ROBUST CHECKERS FOR SELF-CALIBRATING DESIGNS

THÈSE N^{O} 3647 (2006)

PRÉSENTÉE LE 24 NOVEMBRE 2006

À LA FACULTÉ INFORMATIQUE ET COMMUNICATIONS

Laboratoire d'architecture de processeurs

SECTION D'INFORMATIQUE

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Frédéric WORM

ingénieur en systèmes de communication diplômé EPF
et de nationalité française

acceptée sur proposition du jury:

Prof. E. Telatar, président du jury
Prof. P. Ienne, Prof. P. Thiran, directeurs de thèse
Prof. G. De Micheli, rapporteur
Prof. T. Mudge, rapporteur
Prof. N. Shanbhag, rapporteur

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Lausanne, EPFL
2006

To my parents

Acknowledgements

First of all, I would like to express my wholehearted gratitude to both of my advisors. I am deeply convinced of the professionalism of their abundant and high quality guidance. I feel obliged to mention quantity since I have tasted the *luxe* of weekly meetings with two professors for at least half of the thesis...

One of the contribution of my work is to exploit the strong complementarity of two checkers and combine them into a novel highly robust checker architecture. Likewise, on a human perspective, the thesis has brought together two complementary characters, namely my advisors.

I am hardly exaggerating by describing one as a wild and rapid thinker, exploiting roundtable discussions to produce plenty of illuminating graphs (when his pencil is not lost), and, on top of that, mastering fully the finest nuances of written and oral communication.

The other one is an embodiment of cold logic, rather reluctant to embark on any speculative thinking, and always striving for the perfection of written explanations (in a sense, the simplest possible one). A picture being worth a thousand words, I imagine a bulldozer proceeding at its own pace, certainly, but always forward. Through the work done with them, I will take with me features from both worlds.

While I am well aware that many Ph.D. students would be jealous due to my exceptional access to professorial resources, I have to admit that the beginning of the thesis actually saw me coached by three professors: my advisors and Professor De Micheli. I thank Nanni truly for his counselling and bootstrapping of the project.

I am also grateful to Professor Mudge and Professor Shanbhag who reviewed thoroughly the thesis.

Finally, I would like to thank all LAP members that I have met during my thesis: LAP has definitely be a nice working place. A special thank to the secretaries: Brigitte for her kindness, and Chantal for her ever-lasting cheerful humor and recurrent laugh. I am very grateful to Ajay and his colleague for their fruitful help on solving a particular mathematical problem I encountered. Many thanks also to Miljan who has preceded me in the various tasks required to finalise the thesis. Lastly, I want to thank Michael for the many aviation-related debates held during coffee breaks and apologize to all other "coffee breakers" who suffered from the apparent monotony of our discussions.

Abstract

So far, performance and reliability of circuits have been determined by worst-case characterisation of silicon and of environmental conditions such as noise and temperature. As nanometer technologies exacerbate process variations and reduce noise margins, worst-case design will eventually fail to meet an aggressive combination of objectives in performance, reliability, and power. In order to circumvent these difficulties, researchers have recently proposed a new design paradigm: self-calibrating circuits. The design parameters (e.g., operating points) of a self-calibrating circuit are tuned at run-time by a controller. The latter receives feedback from a checker that monitors correct operation of the circuit. A self-calibrating circuit can thus trade dynamically reliability for power or performance, depending on actual silicon capabilities and noise conditions.

This thesis pioneers the use of digital self-calibration techniques to dynamically tune the operating points of an on-chip link based on the detection of run-time transfer errors. In particular, we show that the energy overhead induced by the checker and operating point controller is offset by the operating of the link at sub-critical voltage. Such a system-level study strengthens the interest into self-calibrated links by demonstrating their feasibility.

The primary focus of the thesis bears on the development of robust and low overhead checkers for a self-calibrating on-chip data link subject to errors caused by operation at sub-critical voltage. Such errors—we call them timing errors—may be numerous and cause error rates as large as 100%. We abstract timing errors by the failure of bit transitions and propose ad-hoc coding techniques to detect them reliably. We emphasise the originality of the coding requirements by showing that (i) traditional error correcting codes (like CRCs) fail to detect timing errors under over-aggressive operation of the link, and (ii) asynchronous codes such as dual-rail detect all timing errors, but incur a significant bandwidth overhead in the synchronous context of our problem. Next, we introduce a novel code-based checker satisfying such requirements and featuring unique detection capabilities towards both timing and additive errors.

Then, we contrast the error detection capabilities of the code-based checker with the one of double sampling flip-flops. We stress the complementarity of the two approaches and show how to optimally combine them into an even more robust checker featuring a very limited wiring and circuitry overhead. Finally, we extend our work to computing elements by giving preliminary research directions on the detection of timing errors resulting from the self-calibration of the operating points of an adder.

The main contribution of this work is to propose novel checker architectures based on codes and/or double sampling flip-flops to detect massive timing errors caused by self-calibration of the link operating points. A requirement rendering our work unique is that reliable operation of the checker should be ensured over the whole range of bit error rates from 0 to 100%. Furthermore, we have developed a unified framework bringing fundamental insights into the timing error detection capabilities of various practical encoding schemes.

Keywords

self-calibrating VLSI design, electrical parameter variation, error detection, self-synchronisation, low-power on-chip link.

Résumé

La performance et la fiabilité des circuits intégrés sont actuellement déterminées par des hypothèses pessimistes de type "worst-case" concernant les charactéristiques des transistors et l'environnement du circuit tels que le bruit ou la température. Cette approche est tôt ou tard vouée à l'échec car la miniaturisation croissante des circuits intégrés s'accompagne d'une variation significative des charactéristiques des transistors ainsi que d'une réduction des marges de sécurité par rapport au bruit. Ainsi, il devient de plus en plus difficile, voire impossible, de satisfaire simultanément les objectifs fixés en termes de performance, fiabilité, et faible consommation. Récemment, des chercheurs ont proposé une approche différente qui consiste à *auto-calibrer* les paramètres importants au lieu de les déterminer par des hypothèse pessimistes. Par exemple, la fréquence et la tension d'un circuit est ajustée dynamiquement par un contrôleur en fonction de la de la détection en ligne des erreurs causées par le processus d'auto-calibration. Ainsi, il est possible d'utiliser des points de fonctionnement beaucoup plus agressifs que ceux résultant d'hypothèses pessimistes.

Cette thèse est consacrée à l'étude et au développement du circuit chargé de détecter en ligne les erreurs causées par l'auto-calibration des points d'opération d'un lien sur puce. Durant ce processus, le taux d'erreur peut tout à fait atteindre 100%. Nous modélisons ces erreurs par l'échec des transitions de bit à bit durant la communication. Dans un premier temps, nous proposons une technique ad-hoc pour la détection de telles erreurs basée sur le codage des données. En comparaison des codes asynchrones (comme Dual-Rail), cette technique permet de réduire significativement la quantité de redondance tout en garantissant une faible probabilité de non-détection des erreurs. En simulant un lien sur puce capable d'auto-calibrer sa fréquence et tension d'alimentation, nous démontrons que le bilan énergétique d'un tel système est favorable car l'energie supplémentaire dépensée par le contrôleur et les bits de redondance est largement compensée par les gains dus à l'alimentation du lien à une tension sous-critique. Ensuite, nous comparons la technique de détection d'erreur proposée (i.e., encodage des données) avec une autre méthode basée sur l'emploi de bascules à double échantillonage. Après avoir mis en évidence la forte complémentarité en terme de pouvoir de détection d'erreurs de ces deux techniques, nous montrons comment elles peuvent être simplement combinées. Il en résulte un circuit extrêment robuste, même avec une faible redondance. Enfin, nous donnons quelques pistes de recherches préliminaires quant à la détection des erreurs dues à l'auto-calibrage des points d'opération d'un additionneur.

La contribution principale de cette thèse est de proposer de nouvelles techniques utilisant des codes et/ou des bascules à double échantillonage afin de détecter les erreurs dues à l'auto-calibrage des points d'opération d'un lien sur puce. D'un point de vue théorique, nous présentons une taxinomie complète des encodages ayant un intérêt pratique dans les circuits synchrones et ayant des capacités d'auto-synchronisation.

Mots Clés

auto-calibration pour circuits intégrés sur puce, variation des paramètres électroniques, détection en ligne d'erreurs, auto-synchronisation, lien sur puce avec faible puissance.

Contents

List of Figures

Chapter 1

Introduction

> Out of intense complexity, intense simplicities can emerge.
>
> Sir Winston Churchill

We start by briefly introducing worst-case design in Sec. 1.1. In particular, we explain why this technique is facing increasingly large difficulties as CMOS technology scales down. Next, Sec. 1.2 describes the concept of *self-calibrating* circuits. We motivate the interest in this novel technique that we present as an alternative to worst-case design. Sec. 1.3 states qualitatively our goal and the problem we focus on. Finally, Sec. 1.4 gives the general structure of the thesis.

1.1 Worst-Case Design

Worst-case design is a widespread engineering technique extremely simple in its principle: it consists in taking a safety margin sufficient to ensure that certain design objectives are met even in conjunction of different worst-case situations. Engineering knowledge is required to quantify what is both "sufficient" and cost-effective.

Described at a high level, worst-case design in the VLSI field is the technique designers have used so far to ensure that a circuit meets its performance, reliability, and power objectives, even though there are uncertainties at design time. It consists in making worst-case assumptions about various quantities that are unknown at design time. Essentially, these assumptions bear on the following aspects.

- **Process capability.** This aspect relates to the speed at which silicon operates. It is modelled as a probabilistic distribution, either continuous— speed is then a positive real number— or discrete (e.g, speed is either slow, typical, or fast). The speed of silicon assumed at design time is the one of a slow process. If silicon speed is modelled as a Gaussian random variable, then the speed assumed at design time is usually at a $3\,\sigma$ variation from the typical speed (with σ the standard deviation of the Gaussian random variable).

- **Noise and environmental conditions.** Different noise sources—capacitive and inductive coupling, IR drops, etc—are modelled by probabilistic distributions. Their effect is considered additive. Environmental conditions such as temperature are also taken into account.

Worst-case design aims thus at meeting required performance, reliability, and energy consumption levels under some very pessimistic noise, process, and environmental conditions considered as worst-case. For example, a circuit designed with slow process should operate at the maximum required performance, meet energy consumption and reliability objectives under the worst noise conditions. Only a tiny fraction of all the manufactured chips do not meet these design objectives: the remaining ones constitute the yield. In addition, worst-case design ensures correctness-by-design, since actual process capabilities and actual operating conditions are necessarily more favourable than the ones assumed at design time. This nice property comes at the expense of another: conservativeness. Indeed, worst-case design is intrinsically conservative, because the conjunction of all the worst-case assumptions is unlikely.

Recently, researchers have raised concerns about the viability of worst-case design as CMOS technology continues to scale down [Austin *et al.*, 2005]. The downside of technology scaling is an increase in *variability* and design *complexity*.

- **Variability.** As CMOS technology moves to nanometer geometry, circuits become more vulnerable to internal and external noise sources. The ever decreasing supply voltage reduces noise margins, which in turn exacerbates the effect of various noise sources and the susceptibility to alpha particles (soft errors) [Nicolaidis, 2005]. At the same time, we see increasingly more variation in the devices dimensions and behavior. This fact is referred to as *parameter variation* and is explained by several reasons. First, due to extreme scaling, random variations in the concentrations of dopant atoms in transistor channels affect their characteristics, such as speed and leakage current. Typically, chips manufactured in 130nm technology exhibit a 30% variation in operating frequency and 5 to 10 times variation in leakage power [Borkar, 2005]. Leakage variations of the order of 20 times are reported in 90nm technology. Secondly, lithography—the method used for patterning transistors—is pushed to its limits, causing rough line edges. It is thus difficult to scale device dimensions without creating deep sub-micron noise effects, or even defects. For example, lateral coupling dominates in nanometer wires, which causes serious crosstalk. Moreover, it is anticipated that, in future technologies, as much as 10% of the transistors may have a variation in critical dimensions larger than $6\,\sigma$ [Borkar *et al.*, 2004]. Another factor contributing to differences in transistor behavior is the variation in the heat flux over the chip. While the two previous factors cause static variation, variation of temperature over the chip is dynamic, as it varies in function of time, workload, and location.
- **Complexity.** According to Moore's law, which has been tracked accurately until now, the number of transistors available to designers doubles every 18 months. Due to this exponential increase of the transistor budget, circuits of unpreceded complexity can be manufactured. However, at the same time, the verification of such circuits requires huge efforts. In particular, verifying the different corner cases (such as process, voltage, and

temperature) is a tedious work, although most of these corners are quite unlikely to be encountered during actual operating conditions. The main culprit here stems from one attribute of worst-case design: correct operation has to be ensured. As a result, designers have the challenging task of verifying highly complex circuits under highly unfavorable conditions.

In summary, worst-case design has been successful so far, since manufacturing has been nearly defect-free, process has exhibited low variation, and susceptibility to noise has been controlled. Nanometer technologies, which are forecast in the near future, exhibit none of these properties. These observations motivate concerns on the capability of worst-case design to build reliable, fast, high-yield, and low-power chips. Fig. 1.1 illustrates the inefficiency of worst-case design when facing large delay variation (the spread of distributions has been exaggerated for the purpose of illustration). As a result, costly efforts made by

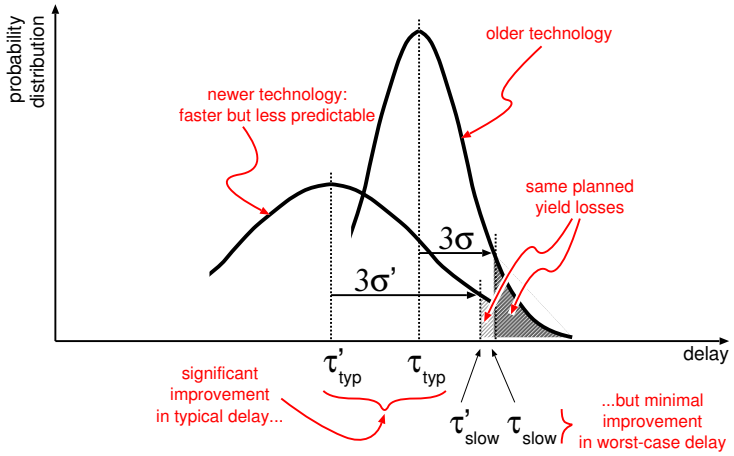

Figure 1.1: Delay distribution of a new and an old CMOS technology. Even though the average delay of the newer technology is improved with respect to the older one, we observe barely no improvement on worst-case delay due to the much larger spread of the newer technology.

technologists to reduce typical delay are cancelled by a simultaneous increase in delay variation.

The next section introduces the concept of self-calibrating design, and explains why it is expected to handle better uncertainty than the worst-case technique just presented.

1.2 Self-Calibration

Worst-case design does not enable to separate concerns such as reliability, performance, and energy consumption. As a result, exploiting the trade-offs between these objectives becomes quite intricate. For example, if a manufactured circuit is faster than expected or required, worst-case design prevents it from either operating faster, as actual silicon capabilities would permit, or from saving energy by working at lower voltage without any performance penalty. Self-calibration is meant to exploit such trade-off. Fig. 1.2 illustrates this point with a simple qual-

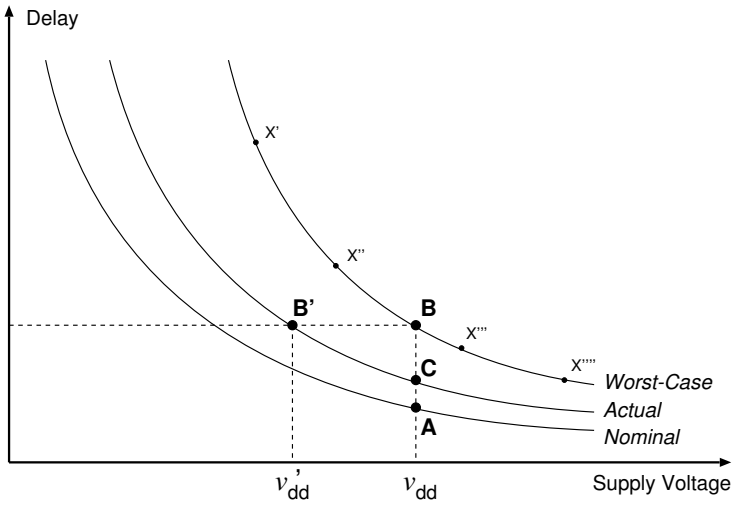

Figure 1.2: Illustration of the potential waste resulting of worst-case design.

itative example. The *nominal* relation between delay and supply voltage might be worsened by the deviation of a number of physical phenomena whose cumulative effect is expressed in the *worst-case* relation. Points X' to X'''' could be used by *Dynamic Voltage Scaling* (DVS) techniques that we discuss in Sec. 2.1. At a given supply voltage v_{dd}, a designer assumes the most conservative delay—that is, that the operating point is not, for instance, A but B—and implements the design accordingly. In fact, the actual device at a particular instant is very likely to be operating in much more favourable conditions and, for instance, the delay may be actually as in C. This implies the following energy waste: operation at the reduced voltage v'_{dd} (B') would yield the same performance the system has been designed for. A less conservative operation in B' rather than B would achieve the very same user function in the same time but would have saved a potentially significant amount of energy—roughly proportional to the difference of the square of the supply voltages v_{dd} and v'_{dd}. As this example illustrates, worst-case design typically results in a waste of resources—usually silicon area and, more critically, energy.

Digital self-calibration has recently gained momentum in the VLSI research community. At the device level, examples include the self-calibration of a keeper[1] based on actual measurement of the leakage current [Kim *et al.*, 2005] and the adaptive source biasing of an SRAM array [Ghosh *et al.*, 2006]. At the circuit level, several architectures have been proposed where the operating voltage and frequency are tuned dynamically as a function of detected errors. Through the work described in Chapter 4, this thesis has actually pioneered the use of digital self-calibration to tune the operating points of an on-chip link [Worm *et al.*, 2002; 2005]. Shortly after our initial work of 2002, researchers from the university of Michigan have developed self-calibration techniques based on a checker consisting of double sampling flip-flops [Austin *et al.*, 2004; Kaul *et al.*, 2005; Das *et al.*, 2006].

[1] A keeper is a device usually added to large fan-in domino gates.

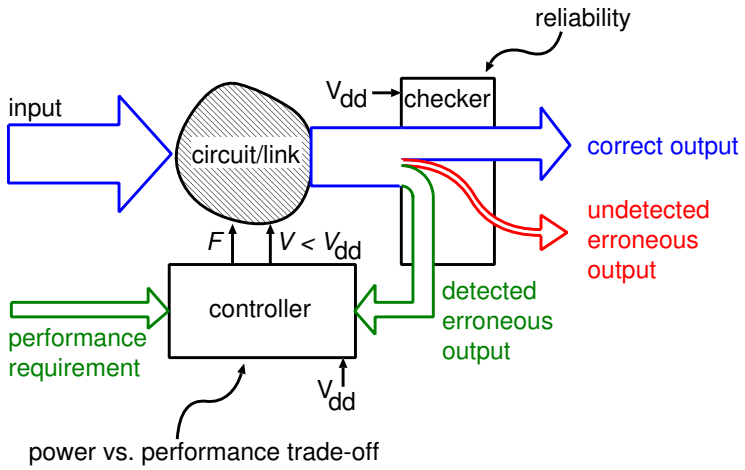

Figure 1.3: A circuit with self-calibrated frequency and voltage supply. Contrary to worst-case design, concerns about, on the one hand, reliability are confined to the checker, while, on the other hand, power and performance trade-offs are determined by the operating point controller.

Fig. 1.3 depicts the structure of a circuit where the supply voltage is self-calibrated as a function of reported errors. The output is verified by the checker. The latter informs the controller of detected errors. Depending on the occurrence of detected errors and on the actual performance requirement, the controller adjusts frequency and/or voltage. Correction of an erroneous circuit output is either achieved by the checker, or by re-evaluating the output. While the value of frequency and voltage do not result from worst-case assumptions about the circuit or link, the checker and controller are assumed to be designed worst-case. Clearly, self-calibration incurs some hardware overhead (mainly the checker and operating point controller logic). Yet, the overhead is expected to be offset by operation at sub-critical voltage. We will verify this expectation in Chapter 4.

Circuits with self-calibrating operating points have the following peculiarities. Contrary to fault-tolerant circuits (that we discuss in Sec. 2.3), the checker is part of a closed loop control system. This feature complexifies the estimation of reliability. For example, the controller should not request operating points where the checker reliability is poor. Moreover, during the self-calibration process, the circuit may be operated temporarily at sub-critical voltage, i.e., under an error rate as large as 100%. Detecting errors even under such conditions puts an original and challenging requirement on the checker.

The next section states the goal of this work and formulates qualitatively the problem we address.

1.3 Goal and Problem Statement

A circuit with self-calibrated operating point requires to (i) detect and correct corrupted circuit outputs, and (ii) control the operating points based on feedback about errors and performance requirements. We want to address items (i)

and (ii) by separating concerns between, on the one hand, reliability—confined to the checker—and, on the other hand, power and performance trade-offs that can be exploited by the operating point controller. That is, ideally, reliability should be determined exclusively by the checker, while the operating point controller would determine power and performance of the circuit. Such a situation can only be attained if reliability is ensured by the checker under *any* circuit error rate. Otherwise, the operating point control policy needs to account for reliability concerns—e.g., avoiding weak spots of the checker—in addition of power and performance objectives. This situation complexifies design trade-offs, or may still require worst-case assumptions to ensure reliability, like, in the case of double sampling flip-flops. In short, we aim at developing highly robust checkers enabling a reliability-agnostic control of the circuit operating points.

The work presented in this thesis focuses on item (i), in the context of an on-chip link with self-calibrating operating points. More specifically, our goal is to design a low-overhead link checker, reliable under any error rate ranging from 0 to 1. This translates into the following problem. Let t_p be the propagation delay through a line of a link, whose operating frequency and voltage are self-calibrated. Even though the link may be operated at sub-critical voltage, t_p is not characterised by any worst-case assumption. The link error rate may thus reach 100%, which happens if the propagation time t_p exceeds significantly the sampling period. In this context, we want to design a reliable checker for the link. The worst-case solution of this problem is to use conservative operating points that ensure statistically $t_\mathrm{p} < T_\mathrm{c}$.

We will argue in Sec. 4.3 that, if this goal is fulfilled, frequency and voltage can be determined in a decoupled manner. On the one hand, frequency is set based on performance requirement only. This question is well researched, as we mention later in Sec. 2.1. On the other hand, voltage is adjusted based on detected errors by simple policies, such as tracking a given error rate, or requesting to increase voltage as soon as an error is reported. While the focus of this thesis is on solving item (i), the solutions proposed for this problem enable to address item (ii) as well using the dynamic voltage scaling technique we describe in Sec. 2.1. The benefits of this approach are as follows:

- separation of concerns between reliability, power, and performance;
- use of well studied dynamic voltage scaling techniques to determine operating frequency;
- simple control of the operating voltage based on detected errors.

1.4 Outline

Chapter 2 reviews various techniques related to some extent with our goal such as dynamic voltage and frequency scaling, double sampling flip-flops, on-chip data encoding, asynchronous, and fault-tolerant circuits. Each time, we emphasise in what the requirements of our problem are unique.

Chapter 3 defines the models of the bit error rate and communication channel assumed in this work. The bit error rate model is not developed in the purpose of high accuracy, but rather in order to generate errors during simulations and compare qualitatively different checkers under a common error model. Regarding the channel, the model introduced defines formally timing errors, i.e., how we abstract errors caused by operation at sub-critical voltage.

Chapter 4 gives a detailed system-level description of a self-calibrating link. The goal of this chapter is twofold. First, we study the energy balance of the self-calibrating link. We observe that tangible energy gains can be achieved without sacrificing performance nor reliability. This result settles the interest into checkers such as the one we study, because it shows that a self-calibrating—despite requiring additional elements—is not unrealistic. Second, we introduce a particular embodiment of the code-based checker we use to detect massive timing errors. We state up-front that the main contributions of this thesis are not to be found in this chapter, but rather in the two following ones where we study in an analytical framework the self-synchronising properties of more general encoding families and propose more robust checker architectures.

In Chapter 5, we develop an analytical framework to express and study in a synchronous context the self-synchronisation properties required to detect timing errors. More specifically, we give an achievable lower bound on the wiring overhead required to detect all such errors—a property that we call *hard* self-synchronisation. We address this question first in all generality, and then focus on a particular encoder structure. Next, we study *soft* self-synchronising encodings, i.e., schemes that use bandwidth more efficiently than hard self-synchronising ones (and thus do not detect all possible timing errors). Moreover, we show how the encoding proposed in Chapter 4 is obtained by reducing the wiring overhead of a particular hard self-synchronising encoding.

In a first part, Chapter 6 focuses on checkers based on double sampling flip-flops. Having explained in Chapter 2 how they detect and correct timing errors, we now contrast their error detection capabilities with our coding technique. We emphasise their complementarity. Next, in a second part, we exploit this fact and show how to simply combine them, while preserving the benefits of each one. The resulting checker architecture features an unmatched robustness to timing errors over the whole range of error rate. Moreover, we show that the combination enables to relax the coding requirements: very basic and tiny codes can be used in the combined checker. We take advantage of this opportunity to extend our scope to computation. We give preliminary checker architectures for an adder with self-calibrated supply voltage. The main achievement of this chapter is to propose novel checker architectures meeting the goal stated in Sec. 1.3.

Finally, Chapter 7 concludes by summarising our achievements with respect to the goal introduced in this chapter and points out perspectives opened by this work.

Chapter 2

How to Design Differently Than Worst-Case?

This chapter briefly reviews existing pieces of work that relate to our goal. Some of them constitute enabling techniques (e.g., dynamic voltage and frequency scaling), while other provide complementary solutions or address slightly different objectives. We proceed as follows. Sec. 2.1 introduces voltage and frequency scaling techniques, which we use to handle uncertainty not only about workload—in fact, its traditional application—but also in noise and process condition. Then, Sec. 2.2 describes alternatives to worst-case design, i.e., techniques that do not satisfy design objectives such as power, performance, and reliability by relying on pessimistic worst-case assumptions. In their intent, such techniques are similar to the one we focus in this work. At last, Sec. 2.3 discusses various contributions to fault-tolerance in VLSI systems. To conclude, we stress that no single piece of work mentioned in this chapter meets our goal, which motivates the originality of our approach.

2.1 Dynamic Voltage and Frequency Scaling

Dynamic voltage and Frequency scaling (DVFS) enables to adjust the operating frequency of a circuit in function of actual workload requirements—typically less than peak performance. By operating at reduced frequency, a reduction in supply voltage is in turn possible. The dynamic power P_{d} dissipated in a circuit is given by the well-known equation

$$P_{\mathrm{d}} = \frac{1}{2}\,\alpha\,C\,f\,v_{\mathrm{dd}}^2,$$

where α is the switching activity factor, C the total switched capacitance abstracting the circuit netlist, f is the operating frequency, and v_{dd} is the supply voltage. Because DVFS reduces both the operating voltage and frequency, the resulting decrease in dynamic power consumption is significant. Reducing power consumption is a key concern, especially for battery-powered circuits—there is no equivalent of Moore law for batteries—and high performance computing elements where cooling is critical.

DVFS has been successfully applied to microprocessors [Nowka *et al.*, 2002; Flautner *et al.*, 2001]. For example, it has been implemented for the Pentium processor and is supported by both the Windows 32 and Linux operating systems. In addition, DVFS has been proposed for off-chip links [Shang *et al.*, 2003]. As pointed out in the discussion of Fig. 1.2, such designs still rely on worst-case assumptions to determine the voltage level chosen for each possible operating frequency. The energy overhead required to change voltage is kept low due to the high efficiency of state-of-the-art voltage converters [Sakiyama *et al.*, 1999].

A DVFS-capable processor offers the possibility to operate at less than peak performance. Yet, in a real-time context, deadlines of tasks should not be missed, even though the processor they are running on does not always operate at peak performance. This requirement leads to the formulation of the following scheduling problem. Given a set of possible operating points and a set of tasks (each one is characterised by its arrival time, processor cycles requirement, and deadline), find a schedule of operating points and tasks that minimises energy consumption (approximated by the average square operating voltage) and meets the deadline of each task. This problem has been heavily researched and is solved: for example, we refer the reader to the elegant solution proposed by Gaujal *et al.* [2005]. For applications without any hard real-time constraints, the control of frequency can be determined by simpler algorithms, e.g., using history-based workload prediction [Sinha and Chandrakasan, 2001].

Besides the fact that DVFS techniques rely on worst-case design and are thus affected by the increasing variability of nanometer technology, their usage in future designs suffers from additional limitations. First, an ineluctable consequence of the reduction in supply voltage is that the range of voltage available for further scaling dramatically shrinks. For illustration, typical values of the supply and threshold voltages of a 90nm technology are respectively 1V and 0.2V, which does not leave a large margin for voltage scaling. Somewhat related to this first point is the fact that, as technology progresses, an increasingly large part of the power dissipated is static. While DVFS only decreases the dynamic power consumption, researchers have proposed to combine it with a method also addressing static power [Martin *et al.*, 2002].

In the perspective of this work, DVFS in itself does not provide a solution since each operating pair frequency-voltage is chosen by worst-case assumptions. Nevertheless, DVFS embodies a design technique that handles workload uncertainty differently than by always operating at peak frequency (i.e., worst-case). Our approach consists in extending this idea to accommodate uncertainty not only about workload, but also about noise and process conditions. Moreover, we do so by leveraging existing work about frequency control. With respect to our intents, the significance of DVFS techniques is to provide mature building blocks enabling to dynamically vary the on-chip supply voltage. Many researchers are currently exploiting this opportunity for different purposes (e.g., dynamic thermal management [Brooks and Martonosi, 2001] or dynamic reliability management [Karl *et al.*, 2006]). Yet, all these design methods share with our work the fact that they bring alternatives to traditional worst-case design.

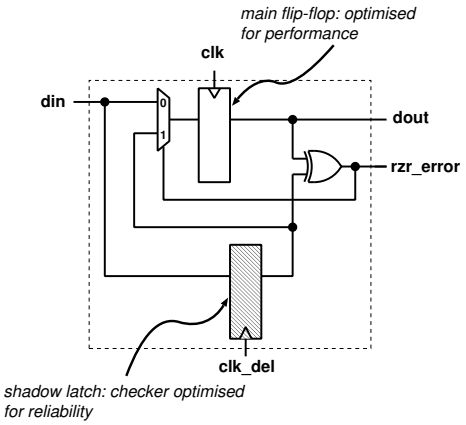

Figure 2.1: A Razor flip-flop. The shadow latch samples the input data some time after the main flip-flop. The sampled values are then compared. A mismatch indicates a timing error because the main flip-flop has sampled data just before it was changing. In this case, the output of the shadow latch is returned to the main flip-flop input, correcting thus the timing error at the expense of an additional cycle latency.

2.2 Alternatives to Worst-Case Design

This section discusses a few examples of designs sharing the feature that the traditional worst-case approach is either marginally or even not at all used to meet design objectives.

2.2.1 Better Than Worst-Case Designs

Better than worst-case designs enable to separate concerns such as performance and reliability by optimising the main circuit for the former while ensuring the latter with a checker [Austin *et al.*, 2005].

Recently, novel designs have been successfully implemented that rely on double sampling flip-flops as a checker to correct timing errors caused by self-calibration of the supply voltage of a circuit [Austin *et al.*, 2004; Kaul *et al.*, 2005; Das *et al.*, 2006; Li *et al.*, 2006]. Previously, double sampling data—i.e., the addition of intra-cycle time redundancy—has been proposed by Nicolaidis as a method for recovering from transient faults, which exploits their locality in time [1999]. As Fig. 2.1 shows, a Razor flip-flop consists of a main flip-flop—fed by the normal clock–and a so-called *shadow-latch* fed by a clock delayed with respect to the main clock. Data at the input is first latched by the main flip-flop. Then, it is latched after some delay by the shadow latch. The setup and hold constraints of the latter are ensured by both worst-case and best-case assumptions about the data arrival time. Provided these assumptions are met, the data held by the shadow latch acts as a "reference". Next, a metastability-tolerant comparator validates the data latched in the main flip-flop by comparing it with the one of the shadow latch. In case a timing error—i.e., a mismatch—is detected, the output of the shadow latch is re-issued at the main flip-flop input, while at the same time raising an error signal. The timing error is thus

corrected at the expense of a one cycle latency (in a processor pipeline, error recovery actually incurs a larger penalty).

A Razor flip-flop corrects timing errors reliably as long as the setup and hold constraints of the shadow latch are met, because the data held by the latter is used to correct timing errors. These constraints are ensured by best-case and worst-case assumptions about the arrival time of data at the input of the shadow latch. The worst-case constraint is somewhat relaxed, because the shadow latch is fed with a delayed clock. More specifically, a too late data arrival results in an undetected timing error since both the main flip-flop and shadow latch hold the same corrupted data: no mismatch is detected. On the contrary, a too early data arrival causes a *short-path* error, whereby the data held in the main flip-flop—although correct—is invalidated due to the early arrival of the the next data sampled in the shadow latch. In such a situation, a timing error is reported although none actually occurred. Razor flip-flops provide a generic method of correcting timing errors because they can replace standard flip-flops after any combinational logic block, as long as the timing constraints of the shadow latch are met and short-path errors are avoided. Yet, depending on the considered circuit, the fact that both the shortest and longest (i.e., critical) paths are constrained may constitute a severe limitation. The latency penalty required to recover from errors is architecture-dependent. For example, error recovery in a pipeline may require to flush it entirely.

As a checker, Razor flip-flops do not meet fully the goal expressed in Sec. 1.3 because reliable operation requires worst-case characterisation of the controlled circuit. We expand on this point later in Chapter 6.

2.2.2 Asynchronous Circuits

In a synchronous circuit, the validity of signals and the progression of events is defined with respect to a global clock. The notion of clock is intrinsically related to the one of worst-case. Indeed, all combinational logic blocks of a synchronous circuit—even though radically different—must evaluate in less than a single clock period. As a result, the performance of a synchronous circuit is defined in terms of worst-case: blocks that evaluate in much less than a clock period do not improve performance at all, since their output is only used at the end of the clock period.

On the contrary, asynchronous circuits do not use a clock to indicate events. Instead, they are able to detect when a combinational block has evaluated (completion detection) and indicate events by changes in handshaking signals. The advantages of asynchronous circuits are numerous.

- **No clock.** Asynchronous circuits do not need complex clock distributions trees. Clock distribution is a notorious source of headache for designers and consume a significant amount of power.
- **Average performance.** Asynchronous circuits do not rely on a clock to indicate the validity of signals. Instead they use completion detection techniques. Therefore, the result produced by a combinational logic block can be exploited as soon as it is available and not at the end of a clock period. That is, asynchronous circuits enable average performance.
- **Tolerance to variations.** Because asynchronous circuits use completion detection techniques, they are tolerant to process and environmental

variations.

- **No idle power.** A clock always toggles—and thus contributes to dynamic power—even though there is no data activity for the circuit. On the contrary, all switching activity occurring on asynchronous circuits is useful.
- **Modularity.** Because, in asynchronous circuits, timing assumptions are explicit in the handshaking protocol, the redesign of a component is easily handled.

Despite these benefits, synchronous circuits constitute the vast majority of all circuits manufactured. The main reasons explaining why asynchronous circuits are confined to niche applications are as follows.

- **Lack of tools.** The design and verification of asynchronous circuits is poorly automated. At the same time, tools for synchronous circuits are extremely optimised—a consequence of their heavy use—and alleviate some of the inherent drawbacks of clocked circuits.
- **Difference and complexity.** Asynchronous circuits are radically different from synchronous ones, which in itself is sufficient to explain why the community of designers is reluctant to adopt them. For example, the operation of asynchronous circuits is concurrent and thus harder to grasp than purely synchronous circuits. In addition, completion detection needs to be implemented with glitch-free logic, which is complex and restricts the set of codes that can be used for this purpose.

Completion detection is performed by either a code membership test or using a matched delay line. In the first method, delay-insensitive codes, such as dual-rail [Varshavsky, 1990] or LEDR (described in Example 2 of Chapter 4), indicate the correct sequencing of events under any timing error rate and, thus, constitute perfect checkers. In fact, timing errors are not even considered in asynchronous circuits since they are defined with respect to a clock (i.e., a too early sampling). The code membership test must be implemented with glitch-free logic, which often requires the use of a spacer and significantly reduces the set of possible codes—in practice, mostly to dual-rail or M-of-N codes [Bainbridge *et al.*, 2003]. In this work, we do not consider applying asynchronous design because, first, as pointed out before, they are poorly supported and, second, they incur a significant overhead in terms of wiring and bandwidth. Yet, the codes used as checker that we develop later in Chapters 4, 5, and 6 borrow ideas from them. In addition, a much larger set of codes is relevant for our problem since there is no strict requirement of glitch-free implementation.

The second method making it possible to detect completion matches the critical path of a combinational logic block with the delay through a chain of buffers (called a delay line). The handshaking protocol relies on the delay line to define the availability of the data output by the combinational logic. In order to operate correctly, the delay through the line should always exceed the delay through the critical path of the logic. For this to hold, the delay line needs to be designed worst-case, even though temperature variations are compensated for, because the delay line and the circuit are exposed to the same environment. Nonetheless, designs based on a matched delay line bring some advantages with respect to synchronous worst-case designed circuits. For example, they tolerate better imbalance between different logic stages, as demonstrated by a technique called de-synchronisation [Blunno *et al.*, 2004; Cortadella *et al.*, 2004]. Besides

their application to detect completion, matched delay lines are also used in voltage controlled oscillators to generate clock signals [Toprak and Leblebici, 2005].

2.3 Fault-Tolerant VLSI Design

Traditional worst-case designed circuits do not tolerate errors at all. As a matter of fact, design parameters—impacting performance and power consumption—are chosen so that, in average, the circuit operates error-free for a specified period of time (i.e., the mean time to failure). As discussed in Chapter 1, this design paradigm is bound to encounter severe difficulties because, among other, soft errors, thermal effects, and increased variation challenge reliability.

Many techniques exist since the beginning of CMOS technology that provide fault-tolerance from the gate level to the application level. Fault-tolerance at the gate level was first proposed in 1956 in the seminal work of Von Neumann in order to compensate the poor reliability of early *NAND* logic gates and majority voters. Each binary logic signal was replaced by a bundle of N lines [1956]. Because each gate was replicated N times, most of the faults could be *masked* by the architecture. The application of this concept to blocks larger than single gates has lead to the well-known N modular redundancy technique, whereby a functional block is repeated N times—typically 3—and the different outputs are resolved by one or several majority voters. Motivated by reliability concerns regarding nanometer technologies, researchers have recently investigated further refinements of modular redundancy [Han *et al.*, 2005; Thaker *et al.*, 2005].

Contrary to data links, computing elements exhibit some intrinsic capabilities of masking faults. For example, when two 1s are input to an *OR* gate, the output is still correct even if a single error corrupts either input. Building on this property, the sensitivity of microprocessors to failure due to soft errors has been recently studied by contrasting combinational with sequential logic and control with execution parts [Saggese *et al.*, 2005a; 2005b].

Other techniques do not mask faults, but instead rely on a checker to detect errors produced by misbehaving circuits—typically, the ones affected by stuck-at faults—or transient faults such as single event upsets [Gorshe and Bose, 1996; Jien-Chung *et al.*, 1992; Nicolaidis, 2003]. Correct outputs produced by the circuit are expected to belong to a particular code. Moreover, the checker itself is not assumed error-free. For this reason, the dual-rail code is often used, which avoids that a single stuck-at fault in the checker corrupts the error signal. Two important properties characterise checkers used in such systems. The first one is defined by the ability to detect all erroneous outputs produced by the circuit for a set of modelled faults. A checker satisfying this property is called *fault-secure*, because it avoids the delivery of corrupted outputs. Yet, some faults could still produce correct outputs and thus remain hidden. The second property aims at avoiding such situations and requires that, for each modelled fault, there exists an input vector producing an error at the circuit output. A checker satisfying this property is *self-testing*, while a checker that is both fault-secure and self-testing is called *strongly fault-secure*. Much of the research in this field targets the development of low-overhead strongly fault-secure checkers for ALUs.

As far as on-chip communication is concerned, a wide range of coding

schemes exist that aim at improving reliability. Some of these techniques often impact not only reliability, but also performance or energy consumption. For example, coding schemes that reduce the switching activity of large inter-wire capacitance decrease energy consumption and, at the same time, increase reliability because bit transition patterns maximising crosstalk are less frequent or even avoided [Zhang *et al.*, 2002; Victor and Keutzer, 2001].

Error correcting codes (ECC) such as CRC codes [MacWilliams and Sloane, 1977; Koopman and Chakravarty, 2004] are widely used in order to increase the tolerance to additive errors. By applying ECC to an on-chip bus, Bertozzi *et al.* have studied the especially important trade-off between reliability and energy consumption [2002]. Indeed, an on-chip bus capable of error detection or correction brings the overhead of both additional wires and the codec circuitry. However, the added redundancy offers the same reliability level as a non-redundant bus at a smaller operating voltage. Although the considered trade-off still relies on worst-case design—in particular, reliability figures are derived from a questionable bit error rate model—our work has similar intents, in that we use a coding technique tolerant to timing errors in order to operate an on-chip link more aggressively than worst-case design enables. Moreover, our work does not depend on a particular bit error rate model since our goal is to ensure reliability under any bit error rate ranging from 0 to 1. Even though traditional ECCs like CRCs increase the robustness to additive errors, Rossi *et al.* have shown that by optimising the inter-wire spacing under a given area constraint, they better tolerate crosstalk caused by inter-wire capacitance [2005a; 2005b].

Finally, fault-tolerance may also also be carried out at the algorithm level, as performed by Shanbhag [2002; 2004]. This technique, referred to by their authors as *algorithmic noise-tolerance*, jointly addresses reliability and energy efficiency issues in digital signal processing systems. It exploits properties of applications run over a DSP (typically, FIR filtering) to enable operation at sub-critical voltage and thus grant significant energy saving. Like better than worst-case designs, this technique exploits data dependencies, i.e., the fact that the critical path is excited only for some particular input data—possibly not occurring frequently. To ensure reliability, architectural enhancements enable to recover from run-time errors and thus guarantee a minimum signal-to-noise ratio. More specifically, blocks implementing linear prediction are added. Furthermore, an original mean of achieving fault-tolerance is called *reduced precision redundancy*, whereby a low precision replica of the main DSP is used to validate outputs of the latter.

While our work shares the feature of operating at sub-critical voltage, algorithmic noise tolerance targets specifically digital signal processing and its goal is essentially to achieve tangible energy saving without degrading the signal-to-noise ratio. Furthermore, algorithmic noise tolerance does not use detected errors as part of a closed-loop control of the supply voltage.

As discussed in this section, fault-tolerance in VLSI systems is achieved by deploying a whole spectrum of techniques that mask the occurrence of faults. Except in the case of algorithmic noise-tolerance, faults are modelled as either relatively infrequent and uncorrelated— typically, transient fault caused by single event upsets—or, on the contrary, as permanent faults—typically, stuck-at faults.

The context of our work is totally different: instead of trying to screen out

the effect of faults, we aim at operating an on-chip link aggressively and thereby, temporarily, under massive timing errors, while at the same time ensuring reliability. Moreover, unlike all techniques discussed in this section, our approach adjusts *dynamically* its level of fault-tolerance to silicon capabilities. As a matter of fact, most of the work reviewed is complementary to our approach, because there is no sense in adjusting the operating points of a circuits subject to rare and uncorrelated errors.

2.4 Conclusions

To conclude, the following points emphasise the novelty of our work by summarising in what its context and requirements are particular.

- **Synchronous operation.** Contrary to the techniques discussed in Sec. 2.2.2, we do not consider asynchronous communication.
- **No worst-case assumption about the link.** We do not rely on any worst-case characterisation of the link. In particular, the reliable operation of the checkers we propose is to be ensured without any assumption about the link itself, even though the checkers themselves can be designed worst-case. This requirement explains the difference of our approach with respect to checkers consisting of double sampling flip-flops (discussed in Sec. 2.2.1).
- **Independence towards bit error rate models.** This point follows from the last stated requirement. We do not make any worst-case assumption about the link and, in particular, about its bit error rate. While we assume a particular type of error—namely, timing errors that we define in Chapter 3—we do not make any assumption on the bit error rate itself, that is, the probability that a timing error occurs given the operating points of the link.
- **Separation of design concerns.** Reliability issues are confined to the checker, since the latter should operate reliably under any possible error rate. Other design objectives such as performance are decoupled from reliability and are determined by an operating point control policy unconstrained by the need of avoiding weak spots of the checker. That is, we aim at a reliability-agnostic control of the self-calibrating circuit.
- **Closed-loop control of voltage.** Contrary to techniques providing fault-tolerance, we apply voltage scaling techniques to adjust dynamically the supply voltage as a function of detected errors. That is, our checker is a part of a closed-loop control system. A practical consequence is that the self-calibration process may cause the link to be temporarily operated under massive error rates as large as 100%.

Chapter 3

Bit Error Rate and Channel Models

This chapter presents the bit error rate (Sec. 3.1) and channel models (Sec. 3.2) of a data link depicted in Fig. 3.1.

On the one hand, the model of the link bit error rate quantifies the probability that a bit error occurs given the operating voltage v_{ch} and frequency F_{ch}. We do not model the bit error rate in the purpose of high accuracy. Instead, the model is needed to generate errors in the simulations presented in Chapter 4 and to compare different checkers under a common error model, which we do in Chapter 6. On the other hand, the channel model describes in what the bit sampled at the link output differs from the one originally emitted when a timing error happens. That is, it explains how errors caused by operation at sub-critical voltage are modelled.

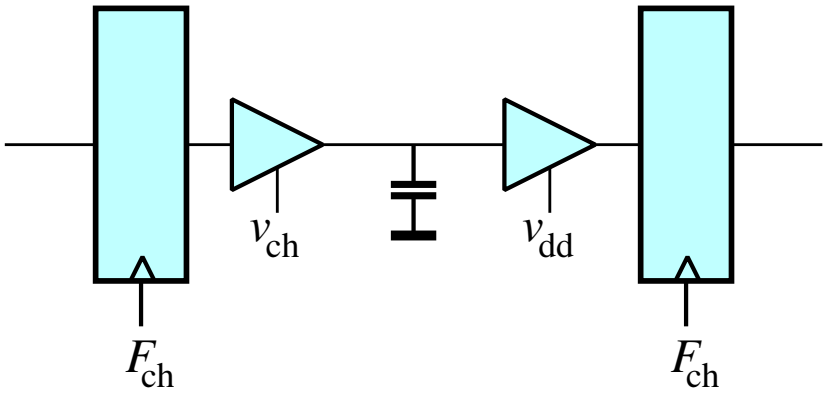

Figure 3.1: A data link supplied at voltage v_{ch} and clocked at frequency F_{ch}.

3.1 Bit Error Rate Model

We consider two possible sources of noise. The first one is an additive white Gaussian noise modelling transient external disturbances, e.g., electro-magnetic interference. The second error source captures the variability of the link delay around its nominal value, representing the effects of temperature, manufacturing conditions, power supply noise, etc. on the link propagation delay. We assume that these two noise sources are uncorrelated.

Regarding the second error source, we would like to model how t_p depends on the link supply voltage. A simple expression can be derived by assuming that delay is measured as the time for the lumped capacitance to attain half a swing (i.e., $v_\mathrm{ch}/2$) and by neglecting velocity saturation and channel length modulation [Weste and Eshraghian, 1993]:

$$t_\mathrm{p} = \frac{C_\mathrm{L}}{k_\mathrm{m}} \cdot \frac{v_\mathrm{ch}}{(v_\mathrm{ch} - v_\mathrm{th})^2}, \tag{3.1}$$

where

- k_m is the transistor transconductance, which depends on the driver transistor dimensions and on some technological parameters,
- C_L is the interconnect capacitance, and
- v_th is the threshold voltage of the devices.

A more complex expression can be used when considering velocity saturation and channel length modulation [Rabaey *et al.*, 2003] as well as sense amplifiers at the receiving end. We refrain from using more complex delay models. Furthermore, we point out that the lumped capacitance model described in Eq. (3.1) is currently widely used in design tools, e.g., by those of Cadence.

In order to model the effect of crosstalk, temperature, and process variation over the link delay, we describe the ratio $C_\mathrm{L}/k_\mathrm{m}$ of Eq. (3.1) as a random quantity. We denote this ratio by α and model it by a Gamma random variable

$$\alpha \sim \Gamma(a, b),$$

where a and b are two real positive parameters characterising the distribution. We use a Gamma distribution because it can model more accurately the line delay than a Gaussian distribution. For example, a Gamma distribution takes only positive values, contrary to a Gaussian. Exponential ($a = 1$) and Erlang ($a \in \mathbb{N}$) distributions are a particular case of the Gamma distribution. While α is modelled as random, both v_ch and v_th are considered deterministic.

Let μ_{t_p} be the mean of t_p and $\sigma^2_{t_\mathrm{p}}$ be its variance under a reference voltage v_dd (later used to characterise nominal operating conditions). Because v_ch and v_th are modelled as deterministic constants, μ_{t_p} and $\sigma^2_{t_\mathrm{p}}$ can be determined as a function of the parameters a and b as indicated below:

$$\begin{aligned} \mu_{t_\mathrm{p}} &= \frac{a\, b\, v_\mathrm{dd}}{(v_\mathrm{dd} - v_\mathrm{th})^2}, \text{ and} \\ \sigma^2_{t_\mathrm{p}} &= \frac{a\, b^2\, v_\mathrm{dd}^2}{(v_\mathrm{dd} - v_\mathrm{th})^4}. \end{aligned} \tag{3.2}$$

The distribution of α is thus entirely determined by the mean and standard deviation of t_p under the reference voltage v_dd. By solving the last equation for a and b, we obtain easily

$$\begin{aligned} a &= \frac{\mu_{t_\mathrm{p}}^2}{\sigma_{t_\mathrm{p}}^2}, \text{ and} \\ b &= \frac{\sigma_{t_\mathrm{p}}^2}{\mu_{t_\mathrm{p}}} \cdot \frac{\left(v_\mathrm{dd} - v_\mathrm{th}\right)^2}{v_\mathrm{dd}}. \end{aligned} \tag{3.3}$$

The bit *timing error rate*, ε_t, is the probability that a bit line is sampled incorrectly. More precisely, ε_t is the probability that t_p exceeds the sampling period T_c (with $T_\mathrm{c} = 1/F_\mathrm{ch}$). By definition of α, $t_\mathrm{p} > T_\mathrm{c}$ if and only if

$$\alpha > T_\mathrm{c} \frac{\left(v_\mathrm{ch} - v_\mathrm{th}\right)^2}{v_\mathrm{ch}}.$$

It follows that

$$\begin{aligned} \varepsilon_\mathrm{t} &= P\left(t_\mathrm{p} > T_\mathrm{c}\right) \\ &= P\left(\alpha > T_\mathrm{c} \frac{\left(v_\mathrm{ch} - v_\mathrm{th}\right)^2}{v_\mathrm{ch}}\right) \\ &= 1 - \mathrm{F}\left(T_\mathrm{c} \frac{\left(v_\mathrm{ch} - v_\mathrm{th}\right)^2}{v_\mathrm{ch}} \mid a, b\right), \end{aligned} \tag{3.4}$$

where $\mathrm{F}\left(\cdot \mid a, b\right)$ is the cumulative distribution function of α:

$$\mathrm{F}\left(x \mid a, b\right) = \frac{1}{b^a \gamma\left(a\right)} \int_0^x t^{a-1} e^{-\frac{t}{b}} dt,$$

and $\gamma\left(\cdot\right)$ is the Gamma function

$$\gamma\left(x\right) = \int_0^\infty e^{-t} t^{x-1} dt, \; x \in \mathbb{C} \setminus \{0, -1, -2, \ldots\}.$$

In Eq. (3.4), the values of the parameters a and b can be determined knowing the mean and standard deviation of t_p, as indicated in Eq. (3.3). The timing error rate ε_t clearly depends on T_c, v_ch and v_th.

In addition, we model the additive noise v_n by a white Gaussian noise, with standard deviation σ_{v_n}:

$$v_\mathrm{n} \sim N\left(0, \sigma_{v_\mathrm{n}}\right).$$

Although on-chip disturbances are more accurately modelled as bursty noise, the white noise model developed here suffices to prove our concept. The *additive bit error rate*, ε_a, is the probability that the additive noise v_n exceeds half the voltage swing v_ch:

$$\varepsilon_\mathrm{a} = P\left(v_\mathrm{n} > v_\mathrm{ch}/2\right) = Q\left(\frac{v_\mathrm{ch}}{2\sigma_{v_\mathrm{n}}}\right), \tag{3.5}$$

with $Q\left(\cdot\right)$ the complementary cumulative Gaussian distribution function:

$$Q\left(x\right) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-y^2/2} dy.$$

Eq. (3.5) is common in the literature [Hegde and Shanbhag, 2000]. Since it accounts only for additive white noise, it is not a function of the sampling period T_c.

Having now modelled the two noise sources, we express the overall bit error rate ε as a function of the timing and additive bit error rates, respectively, ε_t and ε_a. Because they model different physical phenomena, the random variables α and v_n are assumed independent. For the sake of simplicity, we make the following approximation: a bit error occurs either if a timing or additive error occurs. This implies that the contribution of inter-symbol interference to the bit error rate is neglected if $t_\mathrm{p} < T_\mathrm{c}$, and is approximated to one otherwise. Using Eqs. (3.4) and (3.5), the approximation yields directly

$$\begin{aligned}\varepsilon &= 1 - ((1-\varepsilon_\mathrm{t}) \cdot (1-\varepsilon_\mathrm{a})) \\ &= 1 - \left[\mathrm{F}\left(T_\mathrm{c} \frac{(v_\mathrm{ch} - v_\mathrm{th})^2}{v_\mathrm{ch}} \mid a, b \right) \cdot \left(1 - Q\left(\frac{v_\mathrm{ch}}{2\sigma_{v_\mathrm{n}}} \right) \right) \right]. \end{aligned} \tag{3.6}$$

Whether a bit error occurs for given values of v_ch and T_c depends on the realisation of the two random variables α (which in turn impacts t_p) and v_n. Eq. (3.6) is a generalisation of the relation previously introduced [Hegde and Shanbhag, 2000]. The new relation does not model only additive noise—as done in Eq. (3.5)—but also takes into account some other important error sources impacting the bit error rate.

Example 1 illustrates how we model the bit error rate of an on-chip link.

Example 1. (bit error rate of a link). *To model a given technology, we determine the nominal operating points, the device threshold voltage, the mean and variance of t_p under the nominal conditions, and lastly the standard deviation of the additive white noise. We define these parameters as follows.*

- *The nominal operating points are $v_{dd} = 1.2\,V$ and $T_c = 2ns$.*
- *The device threshold voltage is $v_{th} = 0.2\,V$.*
- *The mean and variance of t_p at the nominal operating points are: $\mu_{t_p} = 1ns$ and $\sigma_{t_p} = 0.1ns$.*
- *The standard deviation of v_n is $\sigma_{v_n} = 0.08\,V$.*

The nominal operating voltage v_{dd} and the threshold voltage v_{th} could reflect a $90nm$ or $130nm$ CMOS technology. The values assigned to μ_{t_p} ($1ns$) and T_c ($2ns$) describe respectively the typical and worst-case delay. It is common for the safety margin between the worst-case and typical delay to be as large as 100% [Cortadella et al.*, 2004]. In addition, the value given to σ_{t_p} is $1/10$ of the typical delay, which has been observed in recent technologies. Under the nominal conditions $v_{dd} = 1.2\,V$ and $T_c = 2ns$, these values result into the following error rates:*

- *the timing error rate is $\varepsilon_t = 1.9 \cdot 10^{-15}$;*
- *the additive error rate is $\varepsilon_a = 3.2 \cdot 10^{-14}$;*
- *the overall bit error rate is $\varepsilon = 3.4 \cdot 10^{-14}$.*

Moreover, assuming that errors occurring on different bit lines of a bus are statistically independent, the probability that at least one bit of a 32-bit word is corrupted is

$$\varepsilon_w = 1 - (1-\varepsilon)^{32} = 10^{-12}.$$

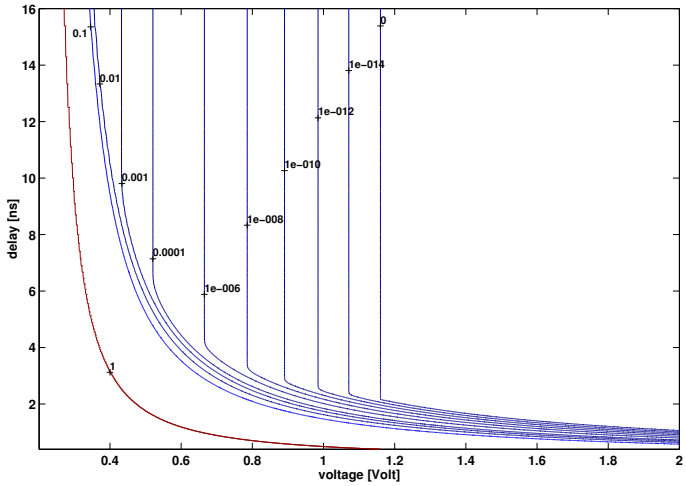

Figure 3.2: Contour plot of the bit error rate ε in the voltage (v_{ch}) and delay (T_{c}) plan.

Fig. 3.2 shows a contour plot of the bit error rate ε in the voltage (v_{ch}) and delay (T_c) plan. Contour lines are vertical, i.e., independent of T_c, as soon as the voltage becomes low enough for the additive noise to exceed the contour level. One recognises the critical zone where the circuit passes from a faulty to a functionally correct state: for delay values sufficiently above the critical value the probability of error is almost zero, whereas the same probability is 1 in regions where the circuit is over constrained. Fig. 3.3 depicts the transition line from 0 to 1 for the bit error rate ε.

We proceed by defining timing and additive errors along with the associated channels. The next section is complementary to this one in that the bit error rate model quantifies the probability of a bit error, while the channel model determines how the received data is obtained from the input data and errors.

3.2 Channel Model

We define first additive errors and the well-known associated channel, which is the *Binary Symmetric Channel* (BSC). Next, we describe timing errors and the resulting channel.

A binary symmetric channel is an additive noise channel. The noise is modelled by a Bernouilli source. The channel output is thus

$$y_k = x_k \oplus e_{k,\mathrm{a}}, \tag{3.7}$$

with x_k the input data and $e_{k,\mathrm{a}}$ a sequence of i.i.d.[1] Bernouilli random variables defined by

$$e_{k,\mathrm{a}} = \begin{cases} 1 & \text{with probability } \varepsilon_{\mathrm{a}}, \\ 0 & \text{with probability } (1-\varepsilon_{\mathrm{a}}). \end{cases} \tag{3.8}$$

[1]Independent and identically distributed.

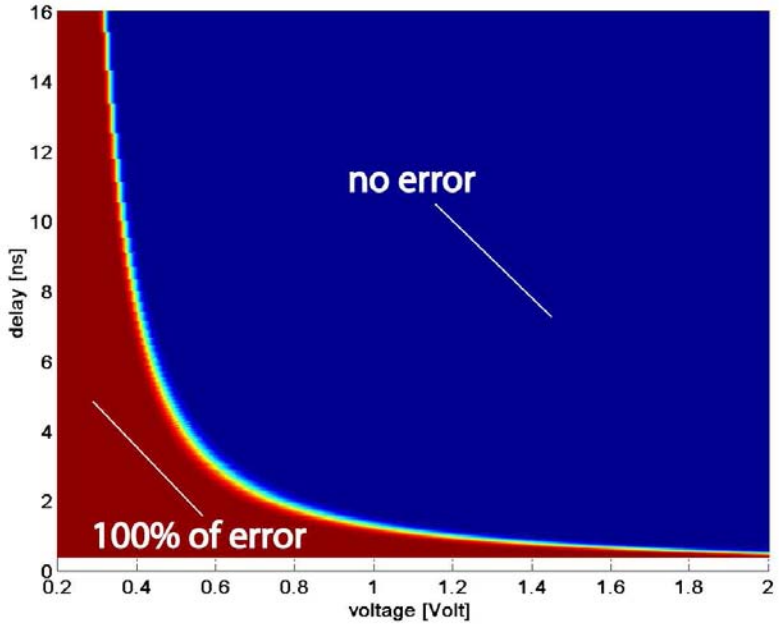

Figure 3.3: Bit error rate landscape in the voltage (v_{ch}) and delay (T_{c}) plan.

ε_{a} can be quantified using Eq. (3.5). The left drawing of Fig. 3.4 depicts the corresponding channel. The channel is said to be symmetric since a 0 has the same probability ε_{a} of being corrupted into a 1, as a 1 into a 0.

We would like to illustrate timing errors by the following phenomenon: assume that a binary signal is sampled synchronously, but with a clock whose period is not guaranteed to be larger than the time elapsed between two signal transitions. In such a situation, the link may be operated beyond its RC delay, which causes inter-symbol interference [Proakis, 2000]. The signal sampled at the receiver end contains not only the current transmitted information but also a contribution from the previously transmitted symbols. In the worst-case, these symbols combine constructively and interfere negatively with the current symbol: an error happens if the interfering contribution is large enough for the sampled value to be in the wrong decision interval. To summarise, by operating the link at sub-critical voltage, there is a risk of sampling the received signal while data transitions have not yet completed: the receiver either samples the previous data, or has metastability problems.

Fig. 3.5 illustrates timing errors. In the top drawing of Fig. 3.5, the data rate is low enough to ensure correct sampling. Therefore, no timing error occurs. Yet, additive noise can corrupt the sampled data, as shown for x_1. The bottom drawing of Fig. 3.5 depicts a situation where the sampling rate is too fast to ensure correct sampling. In this example, x_1 and x_2 are sampled incorrectly.

We adopt a discrete time model representing the sampling instants. We now explain how timing errors affect the sampled data. We denote the input data by x_k and the sampled data by y_k. Moreover, we denote by $e_{k,\mathrm{t}} \in \{0, 1\}$ the failure

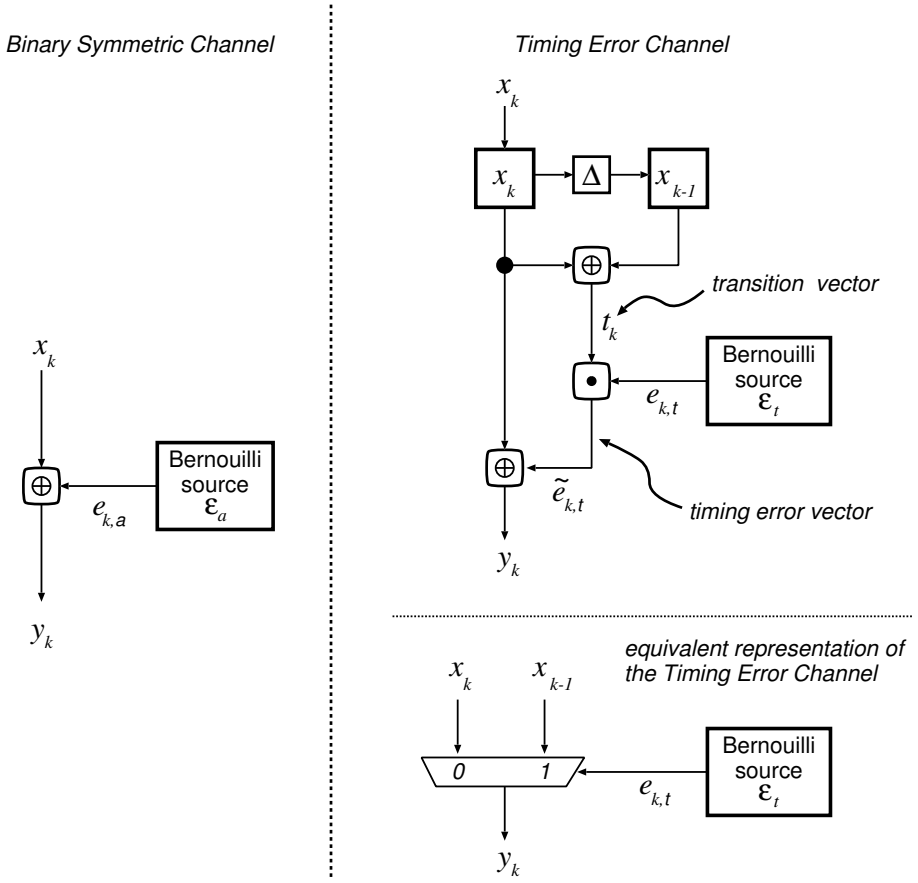

Figure 3.4: Left: a binary symmetric channel BSC(ε_a). Right: a timing error channel TEC (ε_t). The box with symbol Δ denotes a single cycle delay element. The top right representation shows explicitly that errors are caused by the erasure of individual bit transitions. The bottom right representation emphasises that the channel output y_k is either x_k or x_{k-1} depending on $e_{k,t}$.

of data sampling: $e_{k,\mathrm{t}} = 1$ indicates that the sampling was too early, whereas $e_{k,\mathrm{t}} = 0$ indicates a correct sampling. A timing error occurs when $y_k \neq x_k$, which requires the two following conditions to be met:

- the sampling is too early, i.e., $e_{k,\mathrm{t}} = 1$, and
- a bit transition occurs, i.e., $x_k \neq x_{k-1}$.

In the latter condition, note that we have not defined a transition when $x_k \neq y_{k-1}$. Indeed, doing so would preclude some errors (such as the sampling of x_2 in the bottom of Fig. 3.5) from being modelled correctly. Yet, our definition does not capture timing errors caused by a signal transition spreading over more than twice the sampling intervals.

We model the failure of sampling by a sequence of i.i.d. Bernouilli random

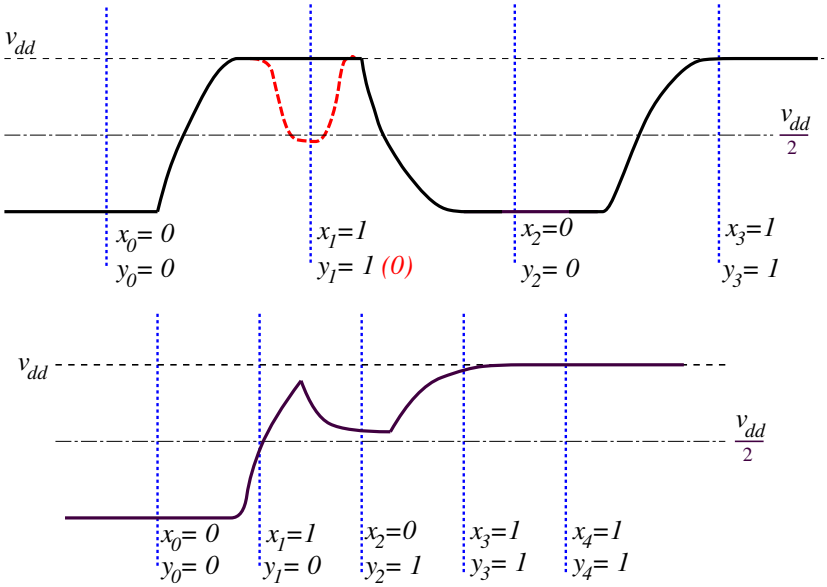

Figure 3.5: Additive (top) and timing (bottom) errors during the transfer of the bit sequence 01011.

variables $\{e_{k,\mathrm{t}}\}$, $k \in \mathbb{N}$,

$$e_{k,\mathrm{t}} = \begin{cases} 1 & \text{with probability } \varepsilon_{\mathrm{t}}, \\ 0 & \text{with probability } (1-\varepsilon_{\mathrm{t}}). \end{cases}$$

That is, we have $P\left(y_k \neq x_k \mid x_k \neq x_{k-1}\right) = \varepsilon_{\mathrm{t}}$, while, for a BSC, $P\left(y_k \neq x_k\right) = \varepsilon_{\mathrm{a}}$. The expression of Eq. (3.4) can be used to determine ε_{t} as a function of the link supply voltage v_{ch} and the sampling period T_{c}.

For a N-bit wide channel, x_k and $e_{k,\mathrm{t}}$ are both N-bit vectors. The relation between the input vector x_k and the sampled data y_k is given by

$$y_k = x_k \oplus e_{k,\mathrm{t}} \cdot \left(x_k \oplus x_{k-1}\right). \tag{3.9}$$

Eq. (3.9) models the fact that $y_k = x_k$ if $e_{k,\mathrm{t}} = \vec{0}$ or $x_k \oplus x_{k-1} = \vec{0}$. On the contrary, a timing error on a bit position i occurs if and only if (i) $x_k^i \neq x_{k-1}^i$ and (ii) $e_{k,\mathrm{t}}^i = 1$ (according to our convention, an index in upper-script denotes the bit position in a vector, while the subscript denotes the time index).

We call a channel described by Eq. (3.9) a *timing error channel*, and we denote it by $\mathrm{TEC}\left(\varepsilon_{\mathrm{t}}\right)$. We introduce a few notations and rewrite Eq. (3.9) as

$$\begin{aligned} y_k &= x_k \oplus e_{k,\mathrm{t}} \cdot \left(x_k \oplus x_{k-1}\right) \\ &= x_k \oplus e_{k,\mathrm{t}} \cdot t_k \\ &= x_k \oplus \tilde{e}_k, \end{aligned} \tag{3.10}$$

where we have defined the *transition vector* $t_k = x_k \oplus x_{k-1}$ and the *timing error vector* $\tilde{e}_k = e_{k,\mathrm{t}} \cdot t_k$.

In this particular case, the timing error vector $\tilde{e}_k$ is not a Bernouilli process as Eq. (3.8). Indeed, only the failure of sampling—which does not necessarily cause

an error—is determined by a Bernouilli random process. A graphic description of the timing error channel is given in the right part of Fig. 3.4. A timing error channel differs from a BSC in two main features.

- Random errors affect data *transitions* and not directly the transmitted data itself.
- Errors are *asymmetric*. While a transition can be cancelled, no spurious transition is ever added.

We use later in Chapter 5 the timing error channel to define self-synchronisation and estimate analytically the reliability of various encoding schemes.

3.3 Conclusions

The primary goal of this chapter is to model at a high level errors occurring on a self-calibrating on-chip link. In this situation, voltage and frequency are not set based on worst-case assumptions about the link delay. Therefore, data sampled on the receiver side of the link may not have completely transitioned to the correct state. In addition, the receiver may encounter metastability problems.

We have modelled operation at sub-critical voltage by the possible failure of bit transitions. This approach is motivated by the fact that inter-symbol interference and inter-wire crosstalk would delay significantly bit transitions under sub-critical operation and has lead to the definitions of timing errors and their corresponding communication channel. These models reflect the original feature of the considered problem. Indeed, in Chapter 5, we define self-synchronisation properties of synchronous encoding schemes based on the timing error channel.

Furthermore, we have formulated an expression of the link bit error rate as a function of the supply voltage and the sampling period (i.e., the inverse of the link frequency). In doing so, we have extended the well known expression of the bit error rate under additive noise [Hegde and Shanbhag, 2000]. We use the bit error rate model to generate errors during the simulations from which the results presented later in Chapter 4 and 6 are derived. However, our approach is independent of a bit error rate model because we study the reliability of checkers under the whole range of possible bit error rate until 100%.

Finally, we point out that metastability is not taken into account in our models. Nevertheless, the checkers presented later include metastability avoidance techniques such as inserting two flip-flops in a row at the link output or the metastability-tolerant comparator deployed in a Razor flip-flop.

Chapter 4

System-Level View of a Self-Calibrating On-Chip Link

This chapter gives a detailed description of a self-calibrating link. Sec. 4.1 discusses its system-level structure and introduces the main idea of the design. Next, Sections 4.2 and 4.3 focus respectively on the encoding and operating point control policy. Then, Sec. 4.4 demonstrates and quantifies the advantages of a self-calibrating link in terms of energy saving and performance in a few particular situations. The goal of this chapter is twofold:

- By comparing the energy consumption of a self-calibrating link with the one of a classic system, we show that, although self-calibration incurs some overhead, energy savings are possible due to operation at lower voltage.
- As a checker, we introduce a particular embodiment of the encoding family that we study in more depth in Chapter 5.

Through the material presented in this chapter, the thesis has pioneered the development of a self-calibrating on-chip link [Worm *et al.*, 2002; 2005]. Nevertheless, our primary contributions—presented later in Chapters 5 and 6—bear on (i) the analytical study of the reliability of the encoding used as checker as well as other more general encoding families, and (ii) the development of checker architectures even more robust than the one deployed in the self-calibrating link we discuss now.

4.1 System-Level Description

We consider a unidirectional point-to-point link interconnecting two subsystems. Fig. 4.1(a) shows a qualitative view of the classic interconnect: at the producer end, a FIFO or a similar buffer is used to decouple the two subsystems which may operate at different frequencies, and a large driver (typically a chain of appropriately sized inverters) charges or discharges the large capacitance represented by the interconnecting wires. A receiver (typically a CMOS gate) compares the level of the line to a threshold and delivers the resulting information to the consumer.

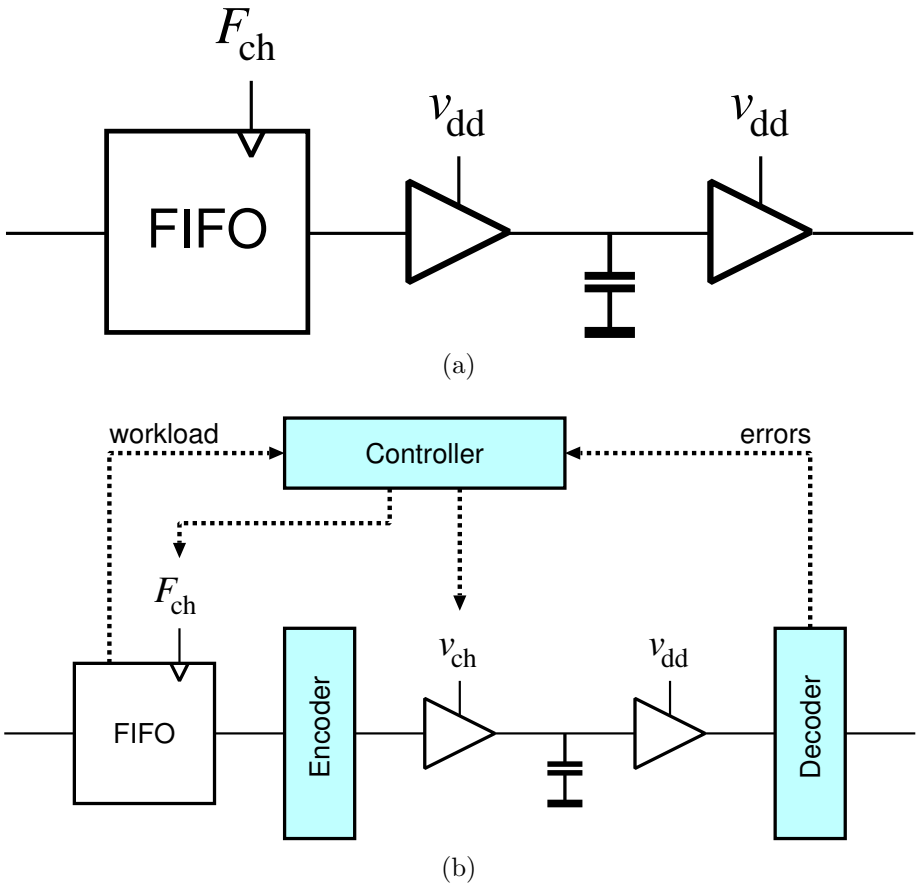

Figure 4.1: The basic idea of a self-calibrating point-to-point unidirectional on-chip interconnect. (a) The classic static scheme, with a FIFO that decouples two subsystems. (b) The proposed self-calibrating scheme with additional elements needed in order to adjust the operating points as required by workload and detected transfer errors.

We add a few elements to the classic scheme, as indicated in Fig. 4.1(b). To reduce the energy consumed per bit, we control dynamically the driver swing and the corresponding receiver threshold. Electrical schemes to reduce the voltage swing of the interconnect are known and well studied [Zhang *et al.*, 2000]. Of course, the variable voltage swing impacts the speed at which the interconnect driver is able to charge or discharge the load capacitance—this phenomenon has been modelled by Eq. (3.1). The maximum reliable operating frequency is thus reduced with lower swings: hence, we need to adapt the communication speed too, as with traditional *Dynamic Voltage and Frequency Scaling* (DVFS) techniques. The key difference with the latter technique stems from the fact that actual operating points (i.e., the pair voltage-frequency) do not result from worst-case assumptions but, instead, are dynamically established by a controller.

The operation with lower swings makes communication more sensitive to several noise sources. To cancel this effect, we introduce an error detection encoding at the word level on the source side—a word is the data unit used for transfers over the link. The link may be operated at bit error rates as large as 1, causing thus multiple errors in a single word. It follows that, in this context, no low-overhead encoding can *correct* errors. As a result, we implement a typical *Automatic Repeat reQuest* (ARQ) strategy, such as Go-Back-N [Walrand and Varaiya, 2000], which relies on the encoding for error *detection* only, while corrupted words are retransmitted. We have chosen in this case to perform error detection and retransmission on a per word basis; however, one could very well imagine an ARQ strategy dealing with packets of several words. In its simplest embodiment, error detection is provided by a code (e.g., CRC) which is transmitted in parallel with the data. DVFS is also applied to the redundant lines that consume additional energy. Nevertheless, the energy saving achieved by operation with a lower swing outweighs the energy required by driving the additional lines, as reported later in Sec. 4.4. Moreover, it has been reported that, for on-chip links, retransmitting corrupted words is more energy-efficient than correcting errors [Bertozzi *et al.*, 2002].

Finally, our scheme requires a controller that decides of the operating frequency and voltage swing: such a controller must be able not only to choose voltage-frequency pairs from a set of operating points as a function of the requested bandwidth; it must also explore the design space to find safe operating pairs. Therefore, it needs as input some information on both bandwidth requirements and channel reliability. In summary, our system

1. uses a variable frequency and swing to trade off speed for energy,

2. implements error detection and ARQ to guarantee reliable communication, and

3. exploits a variable relation between operating frequency and voltage swing to find the best safe operating point in the current environmental conditions, based on monitoring the error rate.

In essence, the controller will provide the minimum voltage swing such that communication is achieved at the requested bandwidth and under a limited retransmission rate. In the ideal case that all transmission errors could be detected, our scheme would trade off transmission energy for additional communication latency (i.e., tardiness). In reality, codes detect only a subset of the

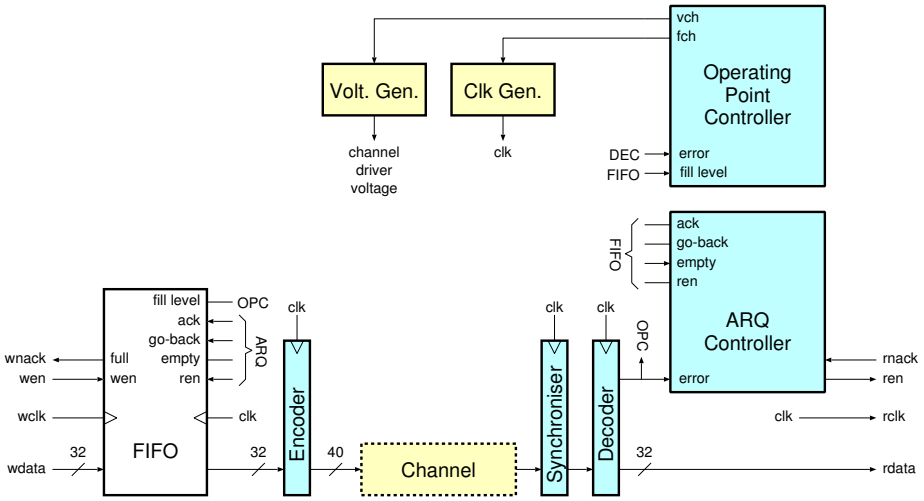

Figure 4.2: A more detailed architecture of the self-calibrating point-to-point unidirectional on-chip interconnect.

possible transmission errors. In our case, undetected malfunctions cause data corruption: we will measure reliability of the link by the residual error rate.

A more practical view of our system is represented in Fig. 4.2. It describes in greater detail the idea of Fig. 4.1(b). The FIFO separates two clock domains: the write clock on the data producer side, and the read clock which is fed to the remaining of the design. Yet, the data interfaces on the producer and consumer ends are the same. The FIFO consists essentially of a dual ported memory with some logic around to control the read/write address and to generate the empty signal. In particular, the FIFO needs to handle data retransmissions. In order to reduce the probability of metastability in the decoding logic, two flip-flops in a row are inserted before the decoding stage [Dally and Poulton, 1998]. This is represented by the synchroniser box.

In practice one can separate completely an ARQ controller and a controller devoted to the choice of the operating point. The former has the sole task to push all words of data through the channel until they are communicated without error and in sequence, ignoring the channel parameters: it addresses thus error recovery. In other words, the ARQ controller decides *which words* to push through the channel in order to deliver only those that are in order and for which no error has been detected. On the other hand, the operating point controller is in charge of picking the lowest frequency and voltage swing required to meet some communication constraint—such as an average delay: it decides *how* to communicate, checks if the choice is appropriate by monitoring the error rate, but ignores *what* is going through the channel.

As Fig. 4.2 suggests, the data path is pipelined into an encoding stage, a synchronizing stage, and a decoding stage. Although the choice of pipelining belongs to implementation issues, our architecture relies on this fact to recover from metastability.

We point out that our architecture can seamlessly be applied to segmented busses. In this case, the same voltage swing is used along all segments—which

is possible as every repeater only consists of an inverter supplied at v_{ch}. As matter of fact, we report in Sec. 4.4 conservative results that only consider the energy consumed by the interconnect wires; in reality, the repeaters will draw additional energy which also scales down with our technique. We have modelled a segmented bus, and found that the energy difference compared to a non-segmented bus amounts to slightly less than 5%.

The focus of this work lies on the overall feasibility and potential advantages of such an adaptive transmission scheme and on the challenges of abandoning a conservative worst-case design style. We will not detail some circuit design aspects such as the implementation of the variable supply transmitters and receivers (which we suppose achievable as a derivation of known techniques [Zhang *et al.*, 2000]), nor the availability of on-chip efficient controllable power supply sources (a key component for any DVFS technique and thus the object of many research efforts [Gutnik and Chandrakasan, 1997; Stratakos, 1998]). In other words, our goal is not to demonstrate that DVFS techniques can be successfully deployed over an on-chip link, but rather to show that abandoning worst-case assumptions to determine operating points is feasible and advantageous in many aspects. Fundamentally, dynamic frequency scaling is not required to prove our concept.

The next section introduces the coding scheme we use to detect timing errors. We study its self-synchronising properties in depth later in Chapter 5, as well as other encoding schemes, which are a more general form of the particular case we present now.

4.2 Encoding

Several challenges are required to make the system robust under the extreme conditions planned. The main difference is that we are not trying to screen out and remove some relatively infrequent errors, which is what error detection codes and ARQ protocols are typically used for. Instead, we operate the system within a small margin from where it becomes no longer operational. In a sense, we will push our system to explore the operating space and, thus, to become at times non-operational. In this section we will analyse some of the related challenges and suggest possible solutions.

Using only a spatial encoding—such as, in the simplest case, adding some parity bits to the data word—is not sufficient. This encoding would be effective to detect, for instance, that a single bit has not yet made a transition, due to crosstalk. Yet, if our sampling rate is so fast that the complete previous word is still present on the interconnect (for example, when the sampling process is like (b) in Fig. 4.3), a pure spatial encoding would diagnose the result to be correct and would not detect that the new word is simply not ready, since a codeword would be sampled twice. Fig. 4.4 contrasts two systematic[1] encoding schemes adding spatial (left) and temporal (right) redundancy. As already pointed out, we observe that a spatial encoding cannot distinguish a new ready codeword from an old one sampled twice consecutively due to the failure of all bit transitions.

[1] An encoding is systematic when its codewords are obtained by concatenating redundant bits to information bits.

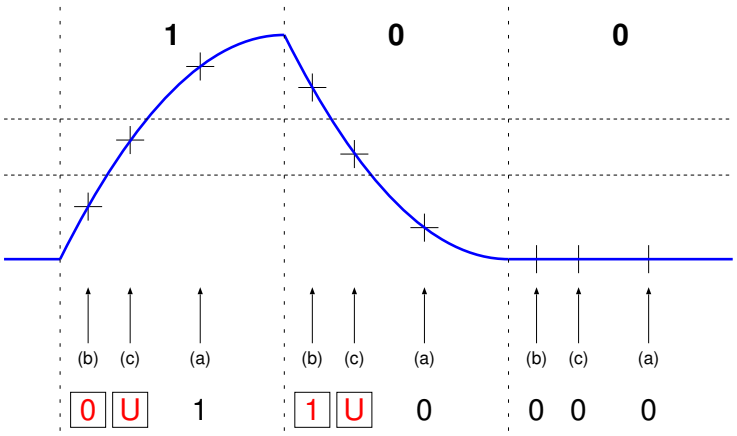

Figure 4.3: A qualitative view of the sources of error in a self-calibrated interconnect operating in too aggressive delay/voltage conditions. (a) Correct operation after a sufficient delay. (b) Bit-errors due to the sampling after a largely insufficient delay. (c) Risk of metastability in the receiver for slightly too aggressive sampling times. (Note that the figure is simplistic in that a new symbol would be emitted at the same time the line is sampled.)

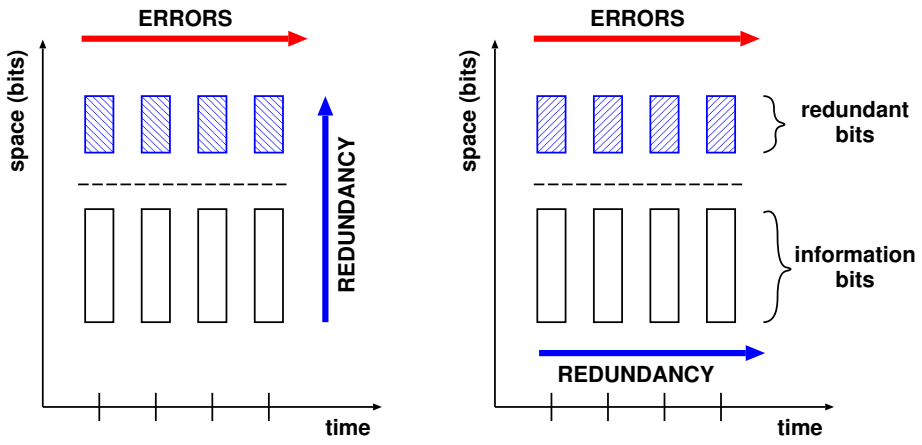

Figure 4.4: Timing errors and encoding schemes with spatial (left) and temporal (right) redundancy.

u	ϕ	r	$\vec{x} = (u \mid r)$
0	0	0	$(0,0)$
0	1	1	$(0,1)$
1	1	0	$(1,0)$
1	0	1	$(1,1)$

Table 4.1: Codewords of the LEDR code. The encoder outputs a sequence of codewords with alternating phase.

Example 2 illustrates a particular encoding adding temporal redundancy, and explains why it detects all timing errors.

Example 2. (Level Encoded Dual-Rail (LEDR) [Dean *et al.*, 1991]). *Dual-rail uses a spacer symbol as temporal redundancy. On the contrary, LEDR includes temporal redundancy implicitly in the codeword sequencing. The encoding is systematic and appends one redundant bit to each information bit. Each codeword is said to have a* phase*, which we define as follows. We denote the information bit by u and the redundant bit by r. The phase, ϕ_k, of the information u_k is a binary information obtained as the parity of the sequence index k:*

$$\phi_k = k \bmod 2 \in \{0,1\}.$$

By extension, the phase of a codeword is the phase of the encoded information. LEDR encodes a single information bit into a 2-bit codeword. The redundant bit r is computed by xor-ing the information bit u and the phase ϕ (i.e., $r = u \oplus \phi$). Equivalently, the phase of an LEDR codeword can be obtained directly by checking whether the information and redundant bits are equal ($\phi = 0$) or different ($\phi = 1$). A codeword contains thus information about data sequencing. Because LEDR computes the redundant bit using temporal information (namely, the phase), it adds temporal redundancy. Table 4.1 lists LEDR codewords.

One verifies easily that LEDR detects all timing errors. Let us consider a 2-bit timing error channel over which information is transmitted with the LEDR code. By inspecting Table 4.1, we observe that exactly *one bit transitions between any two consecutive codewords of opposite phase. As a result, the only timing error that can occur is that a single bit that should transition does not. In this case, LEDR declares an error, since the phase of the received data is unchanged.*

When transmitting more than one information bit, one redundant bit is computed for each information bit by xor-ing the latter with the phase. Fig. 4.5 depicts a possible implementation of an LEDR encoder; while the hardware cost is relatively low, the wiring overhead is significant (100%). In order to use the LEDR as an encoding scheme in the system of Fig. 4.2, the encoder and decoder are provided with a synchronous clock to generate locally the phase bit, since the latter is not transmitted.

Typical self-synchronising codes [Varshavsky, 1990], such as the 1-of-N or LEDR (introduced in Example 2) schemes, incur a large wiring overhead. Instead of using them, we enhance simpler schemes—like classical CRCs—with the notion of phase that has been introduced with the description of LEDR. A particular embodiment is shown in Fig. 4.6. It generates 8 redundant bits

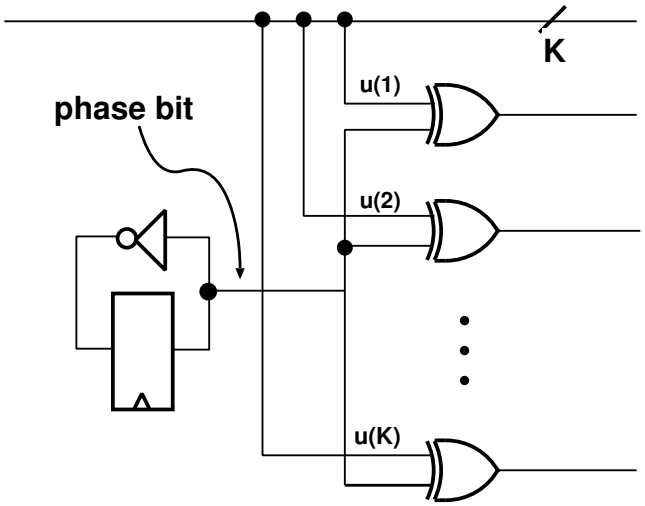

Figure 4.5: Synchronous implementation of a K-bit LEDR encoder. The flip-flop generates the alternating phase bit. LEDR is thus an "alternating-parity" encoding scheme.

computed from all 32 information bits. Our new error detection scheme—that we call *CRC*-8 *alternating-phase* encoding—works by (i) generating a phase bit, i.e., alternatively a 0 and a 1, which is not transmitted but produced independently at the source and destination, and (ii) transmitting eight redundant bits computed using the generator polynomial x^8+x^2+x+1 [Walrand and Varaiya, 2000] and from the 32-bit data padded with the generated bit. The inclusion of the phase bit ensures that any two consecutive identical data pieces cannot have the same encoding—hence, two successive 40-bit encoded words on the channel may be identical only if an error has occurred. The choice of the particular employed CRC-8 is motivated by a study on the error detection capabilities of 8-bit CRC over the binary symmetric channel that concluded that this polynomial has the best overall performance [Baicheva *et al.*, 1998]. We have observed on a few examples that this result still holds over the timing error channel.

The scheme is called alternating-phase encoding because it transmits alternatively codewords from two distinct codes. As depicted in Fig. 4.7, the two codes can be obtained from a single systematic code by discriminating over the most significant bit. In the case where the timing error rate is as large as 1, all bit transitions towards the new codeword fail. Henceforth, the decoder samples twice the same codeword. As already mentioned, any encoding adding only spatial redundancy does not detect this error, which is actually mistaken for the transmission of twice the same information. On the contrary, the proposed encoding detects deterministically such an error. Indeed, the phase bit generated locally by the decoder is included in the parity checks and causes a single bit error, which is always detected by CRC codes. The key feature of this encoding is to include information about data sequencing into the redundant bits, since the latter are computed from parity checks including the phase bit. However, the wiring overhead of the resulting scheme is significantly reduced compared to LEDR, since the parity check of a CRC code involves several information

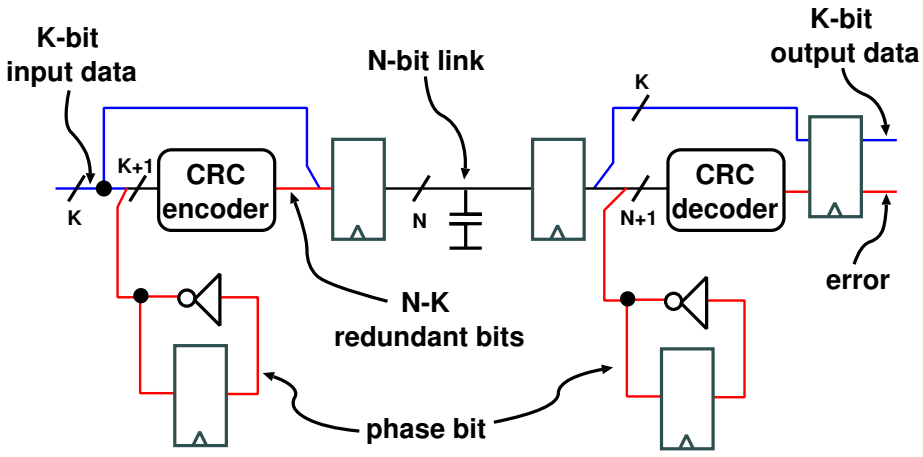

Figure 4.6: The new self-synchronising encoding scheme we propose. The input (respectively received) data is augmented with a phase bit that is not transmitted but generated by the encoder (respectively decoder). The error signal does not only detect bit flips, but also detects when the sampled word is still the last one correctly sent across the channel.

bits, instead of a single one as LEDR does. Possible desynchronisation errors between the toggling flip-flops can be corrected by resetting both the encoder and decoder phase each time an error is detected.

We defer the study of the reliability of this original scheme—as well as other related encodings—to Chapter 5. In the meantime, we have estimated the residual bit error rate of the encoding scheme described in Fig. 4.6. The residual bit error rate is the fraction of transmitted bits affected by an undetected error. We have simulated in VHDL a functional model of the timing error channel and approximated the analytical bit error rate model introduced in Sec. 3.1. We have simulated the transfer of $0.32 \cdot 10^9$ random bits. As shown in Fig. 4.8, no residual undetected error could be observed for raw bit error rates less than 10^{-2}. Moreover, for bit error rates larger than 0.1, the word error rate reaches and remains at 1, while the residual bit error rate reaches a maximum value before eventually decreasing to zero.

Although by no mean specific to the choice of the particular CRC, one point is worth mentioning: as the bit error rate approaches unity, the absolute number of undetected errors increases dramatically. While this is no concern in typical application of error correcting codes, where the error rate is assumed to be always small, a self-calibrating system might operate briefly in regions of extremely high bit-error rate: thanks to the alternating phase bit, our encoding scheme has the important feature of detecting errors when the raw bit error rate approaches unity. As a matter of fact, the residual error rate is 0 if the bit error rate is 1. This feature is essential to prevent the operation point controller using too aggressive, unsafe operating points.

Other error-resilient encoding schemes can be applied. For example, codes with stronger detection probability. It is also possible to avoid the additional lines required to transmit the redundant codes, and to insert error detecting codes in the data stream—as done for data packets. These implies trading off

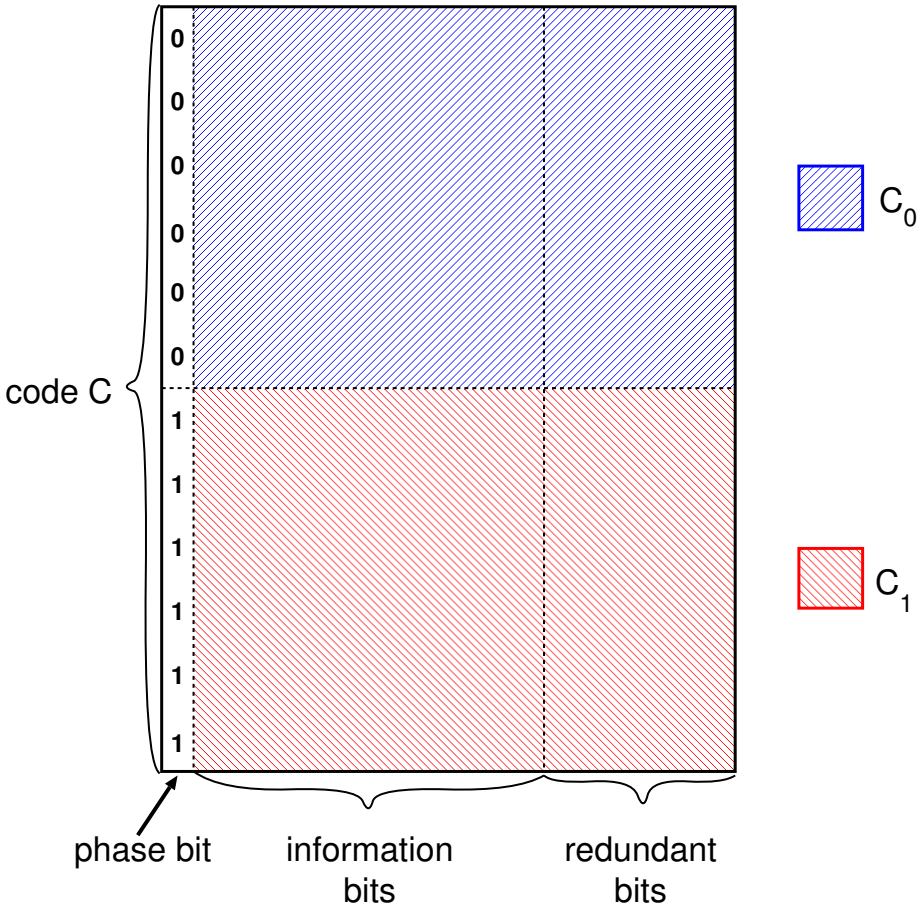

Figure 4.7: The code C_0 (respectively C_1) uses in the alternating-phase encoding consists of all codewords of the code C having 0 (respectively 1) as the most significant information bit.

latency for area. A more subtle trade-off involves the energy consumed and error detecting probability of various coding styles. In this work, we consider the parallel coding scheme of Fig. 4.6 for the sake of simplicity. Nevertheless, we stress that our approach is general, and can be combined with different data formats, including packet encapsulation.

The next section focuses on the link operating point control problem.

4.3 Control

We state the link control problem as a constrained minimisation problem. Then, we explain why and how the voltage and frequency control can be decoupled. Finally, we introduce a particular instance of a decoupled link control which we use later in the simulations presented in Sec. 4.4.

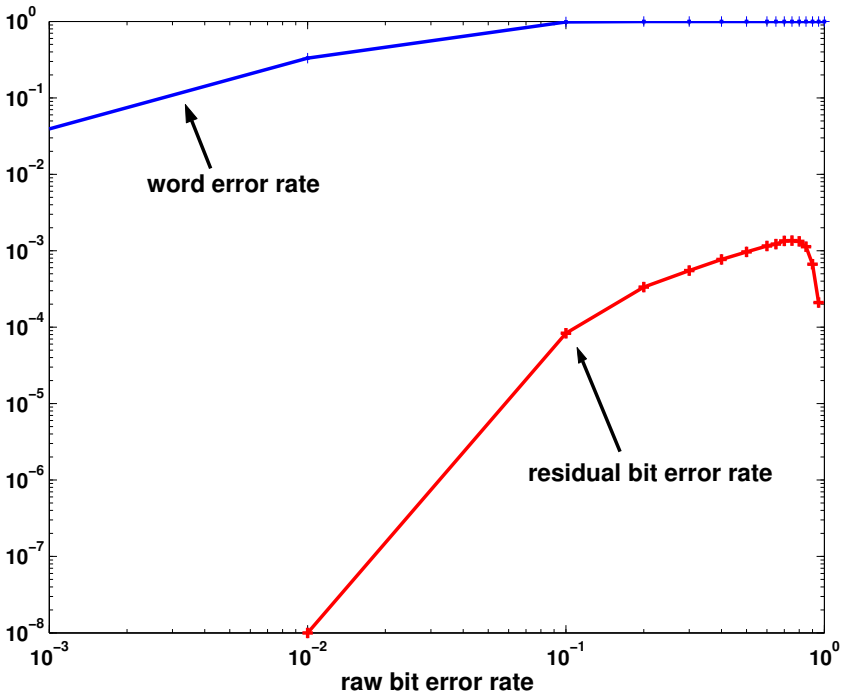

Figure 4.8: Word error rate and residual bit error rate as a function of the raw bit error rate for the CRC-8 alternating-phase encoding generated by the polynome $x^8 + x^2 + x + 1$ (40-bit words).

4.3.1 Statement of the Link Control Problem

The link control problem consists in finding the operating pair $(v_{\text{ch}}, F_{\text{ch}})$ which

1. minimises the energy consumption,
2. meets a performance constraint, and
3. meets a reliability constraint.

Both the performance and reliability constraints are considered as inputs defining some quality of service requirements: the operating frequency and, thus, the corresponding voltage is adjusted in function of workload requirements. Traditional DVFS techniques match energy consumption to the required performance level. In addition, worst-case assumptions determine which voltage is safe for operation at a given frequency, ensuring thus that reliability is met. While more energy-efficient than full time operation at peak performance, this approach does not address concerns raised by increasing process variations, since it relies on worst-case assumptions to determine which voltage is used for a given frequency.

We proceed by stating more precisely the control problem of a self-calibrating link. We define first the energy, performance, and reliability metrics. Referring to Fig. 4.2, we decompose the energy consumed in the transmission scheme into the two following contributions:

- the energy E_{ch} spent over the link to transmit information bits and additional redundancy bits of the error detection code through the link,

- the energy E_{sys} consumed by the logic of the ARQ and operation points controllers, the encoder, the decoder, the synchroniser and the FIFO memory.

E_{ch} corresponds to the energy required to charge or discharge the link capacitance. It scales down with the supply voltage v_{ch}. On the contrary, E_{sys} is independent of v_{ch} since all elements accounted for are supplied at v_{dd}. This decomposition neglects the energy lost in the voltage converter: the efficiency of state-of-the-art voltage converters can be as high as 95% [Sakiyama *et al.*, 1999]. It also neglects the energy spent in three backward control lines due to their very low switching activity.

We are interested in the energy required to transmit a job through the link. We call *job* a collection of information bits that needs to be transmitted over the on-chip link. We define a *transaction* on the link as a cycle during which valid information is transmitted. Transactions on a self-calibrating link include successful word transfers as well as additional transfers caused by retransmissions. However, cycles during which the link is idle are not counted as transactions. Let Φ_{tot} be the number of transactions required for a given job. The energy consumed by the adaptive transmission scheme to transmit the job through the link is

$$\begin{aligned} E_{\text{job}} &= E_{\text{ch}} + E_{\text{sys}} \\ &= \Phi_{\text{tot}} \cdot (K+n)\, C_1\, \overline{v}_{\text{ch}}^2 + \Phi_{\text{tot}}\, C_2 \\ &= \Phi_{\text{tot}} \cdot \left((K+n)\, C_1\, \overline{v}_{\text{ch}}^2 + C_2 \right), \end{aligned} \tag{4.1}$$

where

- K is the number of information bits per word,
- n is the number of redundant bits per word (i.e, $n = N - K$),
- $\overline{v}_{\text{ch}}^2$ is the square voltage averaged over all transactions,
- C_1 is a constant accounting for the switching activity and the bit line capacitance, and
- C_2 is a constant equal to the energy per transaction spent by the ARQ and operation points controllers, the encoder, the decoder, the synchroniser and the FIFO memory.

Eq. (4.1) accounts for the fact that a self-calibrating link introduces an energy overhead both in space (due to the additional redundant lines) and in time (due to the additional transactions incurred by retransmissions). E_{sys} accounts for the energy spent in the elements supplied at voltage v_{dd}. Therefore, it is proportional to Φ_{tot}. Although not indicated expressly, Φ_{tot} is a function of the link word error rate (among other) since it depends on retransmissions. In turn, the link word error rate is a complex function of both v_{ch} and F_{ch}.

Regarding performance, we consider a requirement about the average word transfer delay, which we denote by $\overline{\Delta}$. Constraining the average transfer delay is equivalent to requiring a certain throughput for the link. Moreover, a self-calibrating link cannot ensure a non-trivial (i.e., low) absolute word transfer delay, since retransmissions eventually occur. We denote by Δ_{est} the approximation of the average transfer delay. Δ_{est} is clearly a function of F_{ch}. Because v_{ch} affects the error rate (which, in turn, affects retransmissions and the transfer delay), Δ_{est} is also a function of v_{ch}. We explain later in Sec. 4.3.2 how we estimate the average transfer delay.

As far as reliability is concerned, we would like to meet a constraint about the residual word error rate, ε_{res}. The residual word error rate is the fraction of transmitted words affected by an undetected error. We denote the required residual word error rate as $\overline{\varepsilon}_{\text{w}}^{\text{res}}$. Now, we state the link control problem as follows.

Problem 1. (link control problem). *Consider a job characterised by its average delay requirement $\overline{\Delta}$ and its residual error rate requirement $\overline{\varepsilon}_w^{res}$. Find a sequence of operating pair (v_{ch}, F_{ch}) that:*

1. *minimises E_{job} as expressed in Eq. (4.1),*
2. *ensures that the average delay constraint is met $\Delta_{est} \leq \overline{\Delta}$, and*
3. *ensures that the reliability constraint is met $\varepsilon_{res} \leq \overline{\varepsilon}_w^{res}$.*

The coupling of voltage and frequency makes this problem highly complex. Indeed, E_{job}, Φ_{tot}, Δ_{est}, and ε_{res} are functions of both v_{ch} and F_{ch}. We point out that, in order to obtain meaningful solutions, it is necessary to give both performance and reliability requirements.

In the following example, we illustrate qualitatively how the total energy required to transmit a job varies with the link error rate, which brings some insight into the solution of Prob. 1.

Example 3. (energy optimal error rate). *We illustrate how the total energy E_{job} varies as a function of the word error rate using very high-level energy models. For the sake of simplicity, we consider a system operating at a fixed frequency. We assume that bit errors on a given line are statistically independent from errors occurring on other lines, and that bit errors in a given cycle are statistically independent from bit errors occurring in other cycles. Moreover, we assume that the bit error rate depends on the communication voltage v_{ch} as*

$$\varepsilon = Q\left(\frac{v_{ch}}{2\,\sigma_{v_n}}\right),$$

with σ_{v_n} the standard deviation of the additive noise. Due to the independence of bit errors, the word error rate is

$$\varepsilon_w = 1 - (1-\varepsilon)^N,$$

where N is the word size. The average number of transfer attempts (including the last successful one) is given by $1/(1-\varepsilon_w)$. This relation enables to express how the total number of transactions Φ_{tot} increases with the word error rate. Since E_{sys} is proportional to Φ_{tot}, it also increases accordingly. On the contrary, the energy spent on the interconnect decreases quadratically as v_{ch} decreases. Our goal is to illustrate how the the total energy required to transfer a job varies with the link error rate. Let Θ be value of the ratio E_{sys}/E_{ch} under the worst-case voltage v_{dd}. As the voltage v_{ch} is decreased, we can write

$$E_{ch}(v_{ch}) = E_{ch}(v_{dd}) \cdot \frac{v_{ch}^2}{v_{dd}^2\,(1-\varepsilon_w)} \text{ and}$$
$$E_{sys}(v_{ch}) = E_{sys}(v_{dd}) \cdot \frac{1}{1-\varepsilon_w}.$$

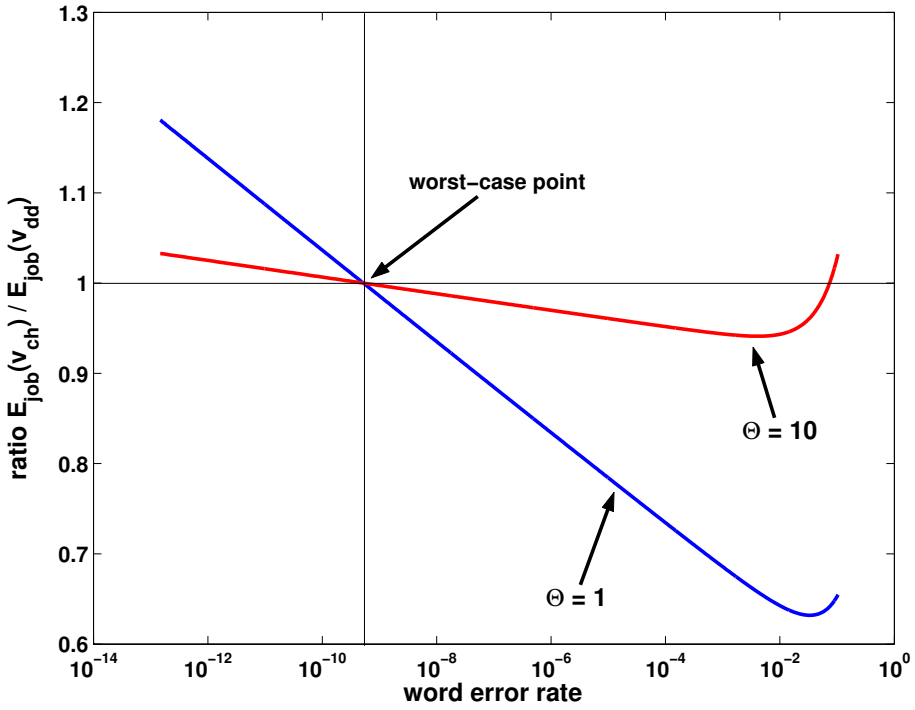

Figure 4.9: The energy scaling ratio of Eq. (4.2) as a function of the word error rate. The parameter Θ is the ratio $E_{\mathsf{sys}}/E_{\mathsf{ch}}$ under the worst-case voltage v_{dd}.

Therefore, the total energy scales as follows

$$\begin{aligned} E_{job}\left(v_{ch}\right) &= E_{ch}\left(v_{ch}\right) + E_{sys}\left(v_{ch}\right) \\ &= E_{ch}\left(v_{dd}\right) \cdot \frac{v_{ch}^2}{v_{dd}^2\ \left(1-\varepsilon_w\right)} + E_{sys}\left(v_{dd}\right) \cdot \frac{1}{\left(1-\varepsilon_w\right)} \\ &= E_{ch}\left(v_{dd}\right) \cdot \left(\frac{v_{ch}^2}{v_{dd}^2\ \left(1-\varepsilon_w\right)} + \frac{\Theta}{\left(1-\varepsilon_w\right)} \right). \end{aligned}$$

Finally, normalising by $E_{job}\left(v_{dd}\right)$, we obtain the energy scaling ratio

$$\frac{E_{job}\left(v_{ch}\right)}{E_{job}\left(v_{dd}\right)} = \frac{\left(\frac{v_{ch}^2}{v_{dd}^2\left(1-\varepsilon_w\right)} + \frac{\Theta}{\left(1-\varepsilon_w\right)} \right)}{1+\Theta} \tag{4.2}$$

Fig. 4.9 plots the ratio expressed in Eq. (4.2) as a function of the word error rate for a 40-bit wide link and for two different values of the parameter Θ. Fig. 4.9 clearly shows the existence of an error rate minimising the total energy: this error rate lies in the 1–10% range, depending on the value of the parameter Θ. However, the error rate minimising the total energy does not necessarily solve Prob. 1 because it may either cause a too long transfer delay or a too large residual error rate. As expected, large values of the parameter Θ decrease the maximum possible energy saving (since most of the energy is spent

in the overhead elements supplied at fixed voltage v_{dd}), and cause the error rate of minimum energy to decrease (since a large error rate would result in a significant amount of energy spent in the overhead elements).

We proceed by arguing why the solution of Prob. 1 can be accurately approximated by a decoupled control of voltage and frequency.

4.3.2 Motivations for a Decoupled Control

As stated in Prob. 1, the link controller has to meet a reliability constraint: the residual word error rate should not exceed a safe value. However, it is impossible for the controller to receive feedback about words delivered and corrupted, since, by definition, these errors are undetected. The controller is only informed of *detected* word errors.

An alternative approach to ensure reliability consists in modelling the checker reliability as a function of the link operating points, and use worst-case assumptions to forbid a-priori operating pairs that do not meet the reliability requirement. While the feasibility of this approach has been demonstrated for Razor flip-flops [Austin *et al.*, 2004], we show later in Chapter 6 that the robustness of these checkers can be increased to a level sufficient to avoid worst-case assumptions about the link error rate by adding little hardware overhead. As a result, we do not follow such an approach that relies on worst-case assumptions to avoid unsafe operating points. On the contrary, our goal is to discover adaptively which operating points are safe in the current noise environment and silicon conditions.

It follows from these observations that, in order to meet a reliability constraint, the link controller has to use the feedback it receives about detected word errors as a correlated feedback on undetected word errors. Because we want to ensure reliability without using any worst-case assumption about the link, the controller should react by avoiding operating pairs that cause retransmissions. We state this important remark as a necessary assumption to solve Prob. 1: *the link controller interprets retransmissions as a sign of unreliability.* Moreover, as far as the CRC-8 alternating-phase encoding is concerned, Fig. 4.8 attests that detected errors need to be avoided in order not to compromise reliability. Indeed, residual errors can be avoided if the link word error rate does not reach 100 %. Accordingly, the link controller should avoid operating points where detected errors are reported. We confirm this fact later in Chapter 5 by studying analytically the reliability of the CRC-8 alternating-phase encoding as a function of the link word error rate.

Because operating points with detected errors are avoided by the link controller, the overall word error rate is expected to be small. Besides reliability concerns, large error rates are likely to be energy-inefficient and cause large transfer delays. Infrequent retransmissions constitutes the key motivation for separating concerns between, on the one hand, energy and, on the other hand, performance. That is, provided that the word error rate is small (typically, around a few percents), Prob. 1 can be accurately approximated by assuming a zero word error rate. In this ideal situation, Φ_{tot} becomes a constant independent of both v_{ch} and F_{ch}. Thus, referring to Eq. (4.1), E_{job} does not depend anymore on F_{ch}. Moreover, Δ_{est} is a function only of F_{ch}, and not anymore on v_{ch}.

In summary, we propose a link control policy that

1. ensures reliability by avoiding operating points where errors are reported,
2. dynamically adapts its operating frequency as a function of the performance requirement, and
3. minimises energy consumption by tracking the lowest possible voltage swing which does not cause transfer errors (retransmissions).

The control policy is *reliable* since it avoids operating the checker where reliability is poor, *energy-aware* since it implements a DVFS technique to determine its operating frequency, and finally *variation-resilient* since it discovers which voltage to use for a given frequency. Fig. 4.10 illustrates how such a control policy selects so-called Pareto points within the design space. Known techniques can be used to solve the frequency selection problem, such as history based prediction of the link workload [Shang *et al.*, 2003]. While we have formulated this problem by constraining the average delay, other formulations—e.g., tracking a target FIFO fill level—would be possible as well. On the contrary, the problem of finding out adaptively the minimum safe voltage for a given operating frequency is more original and related to our intentions.

Having now explained why the link control problem can be decoupled, we proceed by describing a particular decoupled control policy. We study its properties and hardware overhead.

4.3.3 Decoupled Control: a Particular Policy

Fig. 4.11 depicts a simplified operating point control algorithm. The control consists of three main tasks. First, it stores in a table the optimum voltage for each possible frequency. Secondly, it updates the optimum voltages based on reported errors. The update principle is simple: as soon as an error is reported, the optimum voltage (which just suffered an error) is raised. To ensure the most aggressive operation, whenever no error is reported for a given number of cycles, the controller briefly attempts to reduce the Pareto voltage of the current frequency. The threshold *T1* dictates how often attempts to decrease the voltage are made: a counter is incremented each time a word is transmitted successfully. Once the voltage is decreased, the controller enters an *explore* mode whose purpose is to validate the candidate new voltage. The latter is validated only if *T2* consecutive successful word transfers are observed. If that is the case, the controller resets the counter and returns into the *normal* mode. Typical values given to *T1* and *T2* are respectively 500 and 50. The value of *T1* should be large enough to avoid frequent retransmissions caused by tentative reductions of the Pareto voltage.

Finally and independently of this voltage selection process, the minimum frequency meeting the average delay requirement is determined as a function of the FIFO fill level, as indicated by $F_{\text{ch}} = \text{fn}\,(\text{fifo\_level, deadline})$ in Fig. 4.11.

In what follows, we describe in more depth how the average delay is estimated (Sec. 4.3.3.1), we verify some desired properties of the control policy (Sec. 4.3.3.2), and lastly we discuss the hardware complexity of the controller (Sec. 4.3.3.3).

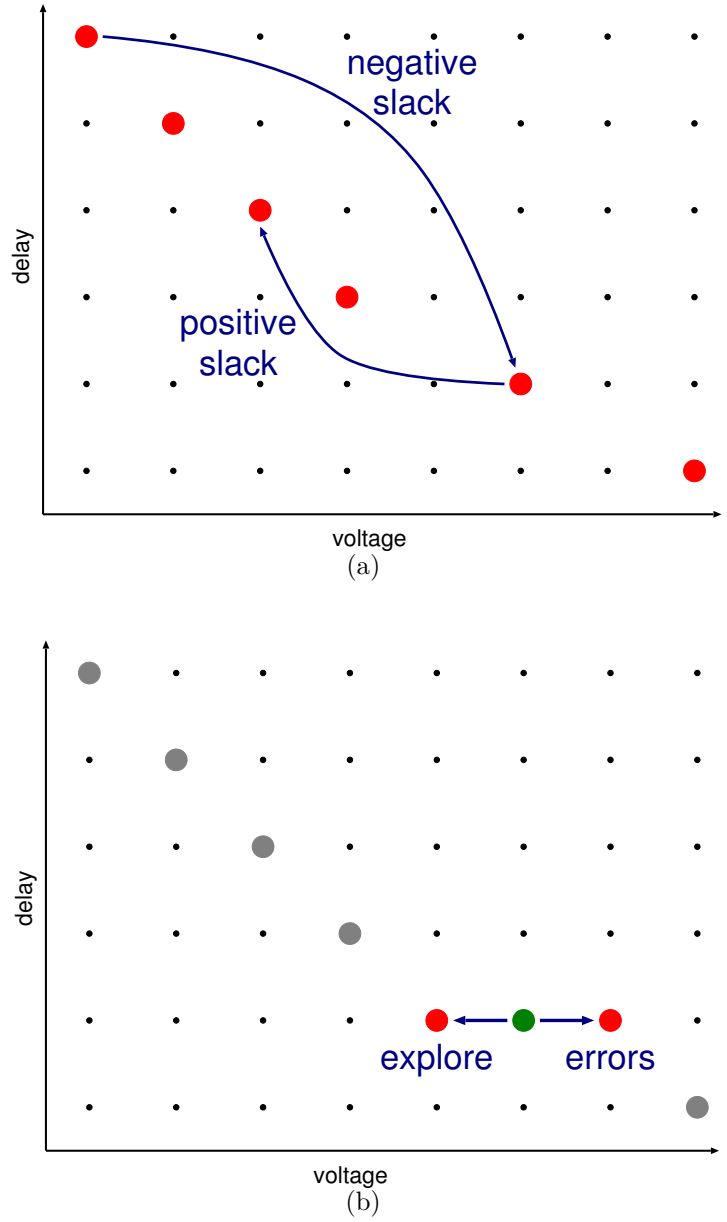

Figure 4.10: Use and estimation of best operating points. (a) The control policy fixes the operating frequency in function of the delay constraint; it sets the operating voltage to the minimum value which has experienced error-free transmission. (b) The controller raises the best voltage for a given frequency when experiencing errors; otherwise, every several cycles, it tries tentatively to reduce it in order to ensure aggressive operation.

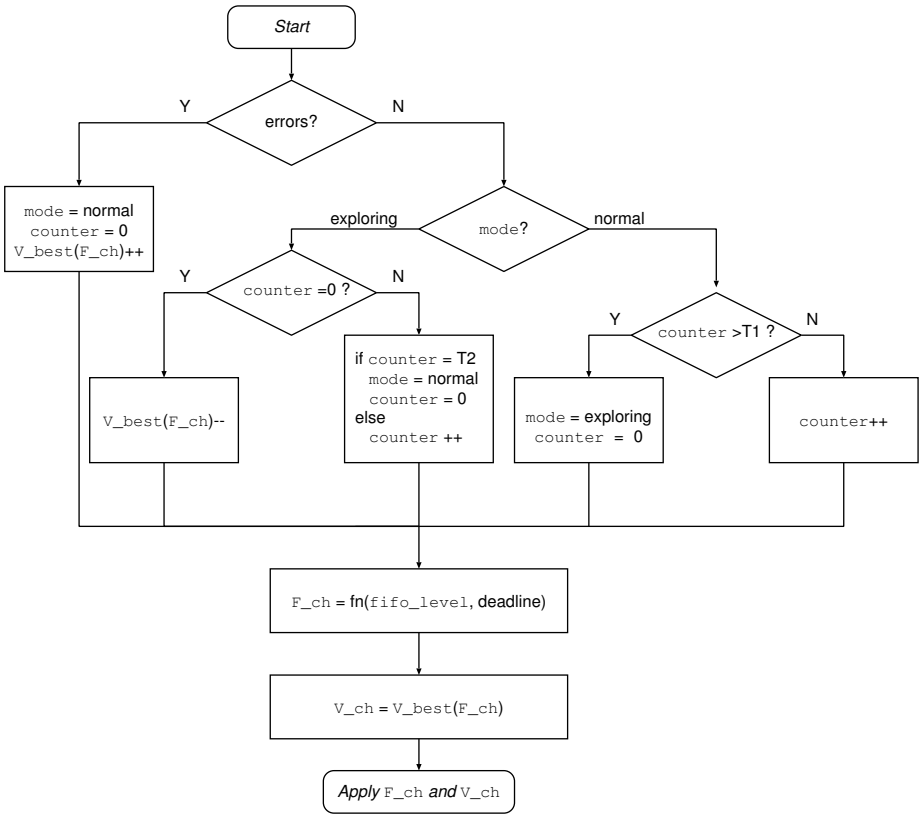

Figure 4.11: Simplified operating point control policy. $T1$ and $T2$ are the values of two constant thresholds.

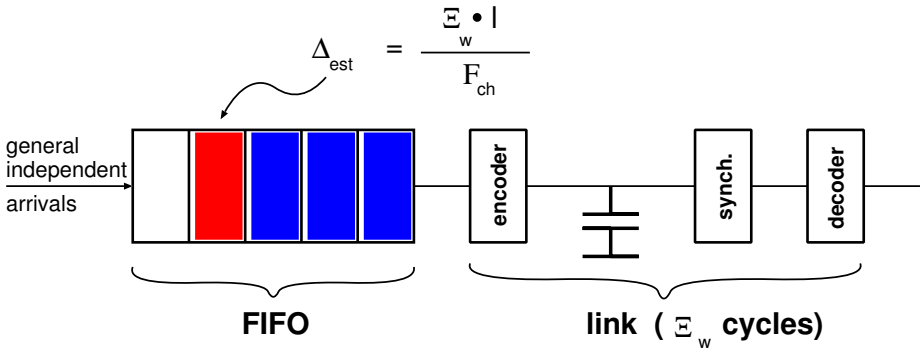

Figure 4.12: GI/D/1 model of the transmission scheme. The arrivals in the queue are *i.i.d* processes. According to Little law, the average transfer delay is proportional to the average buffer fill level. We simply estimate the average transfer delay by the delay experienced by the last queued element.

4.3.3.1 Delay Estimation

Our controller estimates, for each possible frequency, the expected word transfer delay through the system. Then, it selects the smallest frequency meeting the performance requirement.

Because we are limited in resources, we need to estimate very simply the average transfer delay. To do so, our approximation of the average transfer delay, Δ_{est}, is the estimated delay experienced by the last word queued. According to this approximation, Δ_{est}, which includes queueing and transmission, is given by

$$\Delta_{\text{est}}\left(F_{\text{ch}}\right) = \frac{l \cdot \Xi_{\text{w}}}{F_{\text{ch}}},$$

where l represents the queue size, that is, the number of words currently present in the FIFO buffer, and Ξ_{w} is the expected number of cycles needed to send a word through the link. Because our control policy enforces a low error rate, Ξ_{w} can be accurately approximated by the number of cycles required for a word to go through the link when no error occurs—i.e., the depth of the transmission pipeline.

We briefly motivate the approximation made for Δ_{est}. As performed in Fig. 4.12, we can model the transmission scheme by a GI/D/1 queue, i.e., by a single-server queue with a general and *i.i.d.* arrival process and a deterministic service time. According to Little law [Arnold, 1990], the average response time of such a system is proportional to the average fill level. In order to minimise the overhead, we avoid computing an average fill level, and simply sample the average fill level by its instantaneous value. We have also considered finding which element in the queue is the most "slack-critical". The answer is that the most critical is not necessarily the last one. Actually, the problem turns out to be even more complex to solve than computing an average transfer delay.

As far as the implementation is concerned, the controller has to find the slowest frequency among a discrete set meeting the delay requirement. Fig. 4.13 illustrates this selection process. The delay constraint is expressed in cycles with respect to a given frequency. The factors K_0,...,K_Q account for the coefficients

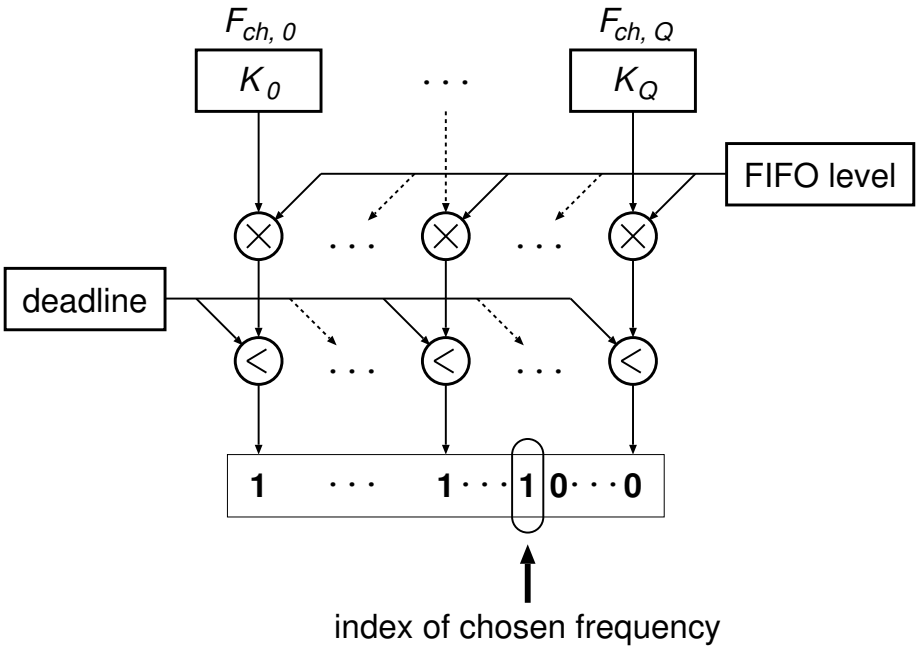

Figure 4.13: Circuit determining the slowest frequency making possible to meet the required delay constraint. By convention, frequency index 0 corresponds to the fastest frequency. K_0,...,K_Q are constants that are hardwired for every frequency.

$\Xi_{\rm w}/F_{{\rm ch},i}$, $i = 0, ..., Q$ and are constant. The circuit needs to multiply these coefficients with the FIFO fill level. Therefore, the complexity of the circuit can be minimised by limiting the number of possible operating frequencies (e.g., to 4) and by picking frequency values whose ratios are powers of two.

4.3.3.2 Properties

We now discuss the *optimality*, *stability*, and *sensitivity* of the control policy to design parameters. We consider two main design parameters. The first one is the threshold $T1$ mentioned in Fig. 4.11 governing how often the controller attempts to use more aggressive voltages. The second parameter—introduced due to implementation reasons—determines how slower the controller runs compared to the data path. The goal of this parameter is to limit the energy overhead of the operating point controller. We now discuss the impact of these two parameters on optimality, stability, and sensitivity. We give later in Sec. 4.4 more experimental details concerning the graphs presented.

- **Optimality.** The algorithm presented in Fig. 4.11 contains two deviations from optimality:

  (i) The slower controller clock that delays voltage and frequency updates.
  (ii) The tracking of the optimal voltage (Pareto voltage) associated to every operating frequency.

While the choice of F_{ch} is not approximated, the delay estimation neglects retransmissions. However, such approximations are inherent to any practical DVFS capable scheme. In summary, the first deviation from optimal control stems from implementation reasons, while the second is intrinsic to the algorithm. We study later the impact of the slower controller clock and focus now about the tracking of the Pareto voltage. We show that the latter hardly affects the control quality. To do so, we simulate a 100,000 words transfer and a-posteriori determine the Pareto voltage of every frequency. Then, we compare the performance of our controller with another *optimal* controller knowing which Pareto voltage to use for every frequency. The difference in performance is minor: our controller saves 25% of energy in this case, while the optimal controller saves one more percent. Both controllers performs the same in term of average transfer delay and residual word errors (none in both cases).

- **Stability.** In our context, stability translates into (i) ensuring that the FIFO level does not grow to infinity, and (ii) after an uncontrolled disturbance (traffic variation or errors), the controller eventually settles on an(other) operating point. Because it tries to obey a delay constraint, the controller does not let the FIFO fill level grow to infinity but, instead, invests more energy by operating at high frequency. The FIFO fill level would only overfill in pathological cases where the classical system's FIFO would too. This is even more evident as the self-calibrating system has a maximum frequency larger than the classic one which suffers worst-case limitations. The second item is granted by the algorithm itself. The only point to discuss stems from the threshold that governs the controller aggressivity. Too low a value leads to undesirable voltage level oscillations. The next paragraphs shows that a wide range of values enable the system to exhibit a sound behavior.

- **Sensitivity.** We show here how relevant metrics such as the energy, transfer delay, and residual word errors are affected by the choice of the controller threshold and by how much slower the controller runs compared to the data path. Recall that the threshold determines how often the controller attempts to set more aggressively the voltage level used for a certain frequency. This threshold henceforth relates with the speed of reaction to changes in noise level. In addition to illustrating the sensitivity to these parameters, the presented results guide us towards a wide range of acceptable values. We have simulated the transfer of 100,000 words. Fig. 4.14 shows that energy saving, words transfer delay, and residual word errors remain in desired intervals over a wide range of threshold values. The top two graphs of Fig. 4.14 do not show any trade-off: a large threshold value results in positive energy savings, a low transfer delay, and no residual errors. The reason is that the error statistics was not modified during simulation. As matter of fact, the Pareto voltages are constant over the whole simulation. On the contrary, small threshold values affect negatively all metrics of interest. First, unsafe operating points are often visited, which causes a significant number of undetected errors. In addition, numerous retransmissions incur a large transfer delay and cancel energy saving.

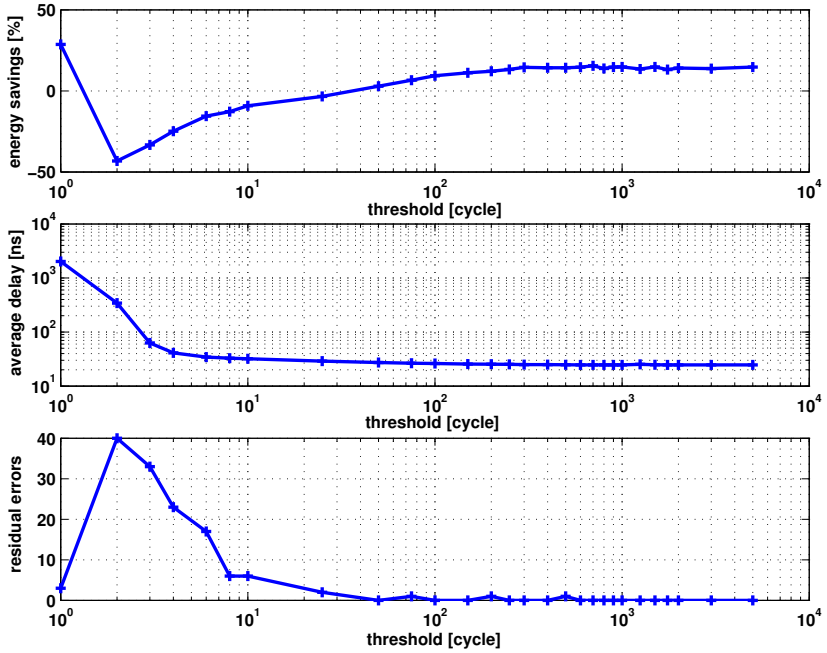

Figure 4.14: Sensitivity of various metrics to the threshold $T1$ (expressed in transmission cycles) mentioned in the description of the controller algorithm.

Finally, Fig. 4.15 illustrates the impact of the ratio between the controller clock and the data path clock on the transfer delay and energy savings. The controller is only allowed to run more slowly. If the controller clock is too slow, the system behaves poorly both in terms of energy saving and average delay. Indeed, energy efficient operating points are excluded, due to the violated delay constraint. On the contrary, once the control policy runs often enough to meet the delay constraint, there is no point in running the controller more often, since its overhead in terms of gates becomes tangible and decreases energy savings. According to Fig. 4.15, running the controller 4 times slower than the data path maximises the energy saving, while barely affecting the transfer delay.

We proceed now by briefly discussing the hardware cost of such a controller.

4.3.3.3 Hardware Overhead

The implementation of the algorithm described in Fig. 4.11—although considerably more detailed than what is shown—has still a relatively low hardware complexity and requires an area equivalent to a few thousand NAND-2 gates. The whole system consists actually of 770 standard cells (230 sequential and 540 combinatorial) from Artisan library in CMOS UMC 90 nm. An accurate estimation of the energy consumed is given later in Fig. 4.17, assuming a discrete operation set of four voltages and four frequency levels. The hardware overhead of the controller increases with the number of possible frequency and voltages due to the need of storing the Pareto voltages in a table.

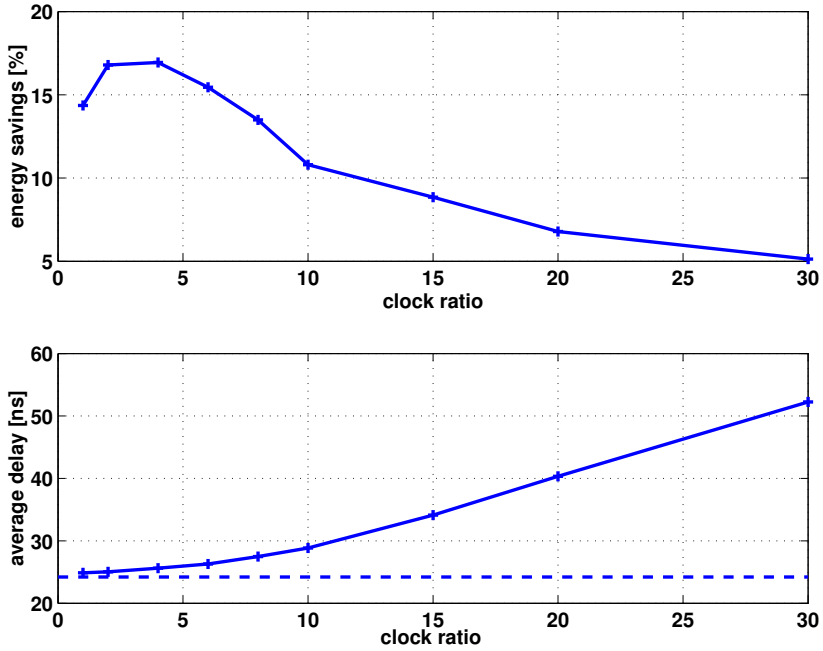

Figure 4.15: Impact of the ratio between controller and data path clock on: (top) energy saving and (bottom) average transfer delay. The dashed line represents the target delay of the classic system.

	classic system	self-calibrating system
voltage swing	1.0V	0.6,...,1.2V
frequency	500MHz	50,...,1000MHz

Table 4.2: Operating range of the classic and self-calibrating interconnect.

4.4 Simulation Results

The goal of this section is to illustrate the advantages of a self-calibrating link over a classical fixed-swing interconnect designed from worst-case assumptions. More precisely, we have compared the self-calibrating link with two kinds of classic fixed-swing systems. The first classic system transmits raw unencoded data on 32 bits. The second one represents the increasing concern with reliability of on-chip communication (see for instance the implementation of practically all internal buses of Itanium 2 [McNairy and Soltis, 2003]). It uses an error detecting code such as an 8-bit CRC to protect 32-bit words and retransmits corrupted words. We refer to the former system as the *classic* one and to the latter as the *classic with codec*. We summarise in Table 4.2 the operating range of the considered systems. We present our results with delays and frequencies relative to the two classic system.

We have measured from simulations the energy consumption, transfer delay, and undetected word errors of the classic and self-calibrating transmission schemes under different—artificial and real-life—traffic traces. A traffic trace is

a file defining the arrival times of words or frames (i.e., a group of words) into the input FIFO.

In order to generate errors during the simulation, we use the bit error rate model introduced in Chapter 3. More specifically, we have modelled the bit error rate and noise sources of a typical 90nm CMOS technology as follows:

- Nominal supply voltage: 1.0V.
- Device threshold voltage: 0.2V.
- Additive noise standard deviation: 0.05V.
- Mean and standard deviation of the propagation delay 1ns and 0.1ns.

Regarding performance, we measure the average transfer delay in case of word transfers, or the absolute delay required to transfer a whole frame. Reliability is measured by counting the number of undetected errors that occurred during the transfer of a given workload. Furthermore, we have estimated the energy required to transfer a workload. We follow the decomposition introduced in Eq. (4.1), which distinguishes the energy E_{ch} spent on the interconnect and the energy E_{sys} consumed by the encoder, decoder, synchroniser, ARQ, and operating point controller. The total number of transactions required to transfer a workload includes the successful transfer and retransmission cycles, and has been measured in the simulation. To estimate E_{ch}, we assume a 1 cm bus length, and a bit line capacitance amounting to 2.73 pF. We have obtained this capacitance value from the technology manual. Moreover, we assume a switching activity factor of 0.5 since our workloads consist of randomly generated data, even for those with a realistic arrival schedule. Finally, we have computed E_{sys} by synthesising the overhead elements required by the self-calibrating system. The error detecting circuitry is as defined in Sec. 4.2. We have approximated the switching activity to 50% on all nets.

In summary, the estimated energy consumption of the self-calibrating system includes the overhead caused by the redundant bit lines, the overhead caused by the additional retransmissions, and the overhead of the logic required by the self-calibration (mainly the codec and operating point controller). However, as mentioned in Sec. 4.3.1, the energy overhead of the voltage converter and of three backward control lines have been neglected.

We present three experiments.

1. The first example focuses on the energy advantage of dynamic bandwidth adaptation considering a realistic MPEG-based workload and an artificial one.
2. The second example shows the energy advantage of dynamically tuning the operating point to actual technology variations.
3. The last example exhibits the robustness of our system to unpredictable noise sources.

4.4.1 Dynamic Bandwidth Adaptation

Modern multimedia algorithms have dynamically varying requirements. Fig. 4.16 shows how the self-calibrating system takes advantage of a time-varying MPEG workload. In the bottom graph, one can observe that the adaptive system tries to exactly match the bandwidth to the current needs: it slows down the communication link to send every MPEG frame as slowly as possible in the

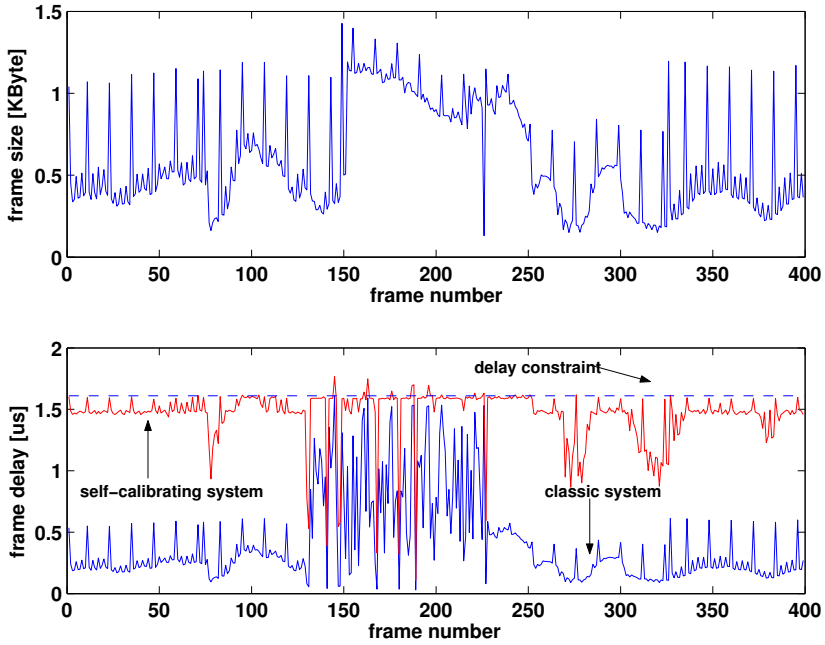

Figure 4.16: Transmission of a variable workload. Top: workload variation in time. Bottom: incurred frame delay in the classic system (low delay) and in the self-calibrating interconnect (delay as close as possible to the imposed constraint—dashed line).

allotted time and, ideally, not any faster. The operation at a lower frequency grants a tangible reduction in average energy: the whole trace, consisting of 400 frames of several Kbytes each, requires 42% less energy with a dynamically self-calibrating system compared to a classic system, and 47% less energy compared to the classic system with codec. Such a saving was achieved by letting the channel controller run at half the frequency than the other components. No undetected error has been reported. Fig. 4.17 depicts the contribution of each component of the self-calibrating system to the energy budget. One can notice that the controller logic and the synchronising registers, which are the only contributions added in our scheme compared to the classic system with codec, impose a relatively limited overhead. On the other hand, it is interesting to note that the energy cost of the codec is not fully negligible and about of the same order of magnitude. Finally, the visible overhead caused by retransmissions amounts to 4%, which accounts for the ARQ controller and the additional memory reads. While this part is clearly quantified, retransmissions also affect the energy budget indirectly, since their delay overhead forces the controller to operate on faster, more energy-costly operating points.

As a last illustration of the energy saving resulting from dynamic bandwidth adaptation, we expose both the classic and the self-calibrating systems to various Poissonian workloads of different traffic intensity. Each workload consists of 100,000 words generated according to Poisson arrivals. We require that, for each workload, the self-calibrating interconnect offers the same average transfer latency as the one experienced by the classic interconnect. Fig. 4.18 illustrates

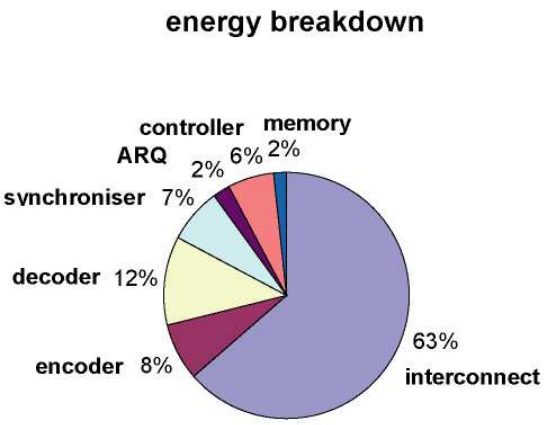

Figure 4.17: Energy breakdown of the self-calibrating interconnect.

the energy saving granted in this case. The graph shows that our controller manages to save energy consistently in all those cases where the average delay is not too constrained—only for tight constrains, the controller energy overhead is larger than the diminishing savings. Moreover, reliability has been ensured as no simulation run suffered from an undetected error.

4.4.2 Exploiting Technology Variations

Fig. 4.19 illustrates the effect of technology on the choice of control points: On a *poor* wafer, simulated with an average propagation delay of 1.125ns (vs. 1ns for the typical delay), the controller chooses mainly Pareto points relatively close to the worst-case delay line. On the contrary, a *good* wafer, simulated with an average propagation delay of 0.875ns, has operating points chosen mostly along a more aggressive delay-voltage line, which reflects the lowest delays experienced by the system. In other words, our control policy is effective in "discovering" the real delay-voltage characteristic of the technology without making any assumptions on it. In both cases, the standard deviation has been reduced to 0.05ns to account for the lower indeterminacy on wafers of a defined quality. These hypotheses result in slightly less than 1% of the wafers being classified as *good* or better than *good*, and *poor* or worse than *poor*. The simulated traffic consists of an artificial workload of 100,000 words with arrival times following a Poisson process. Table 4.3 summarises the energy saving and relative performance of the self-calibrating interconnect compared to the classic and classic with codec systems.

Fig. 4.20 shows the energy-latency trade-off of the self-calibrating link for different wafer quality. As expected, for any type of wafer, energy saving improves with the relaxation of the timing constraints and are dependent on the quality of the wafer: the better the wafer and the larger the average delay that can be tolerated, the more important is the energy saving. This fact can have a very interesting effect in products designed early after the introduction of a

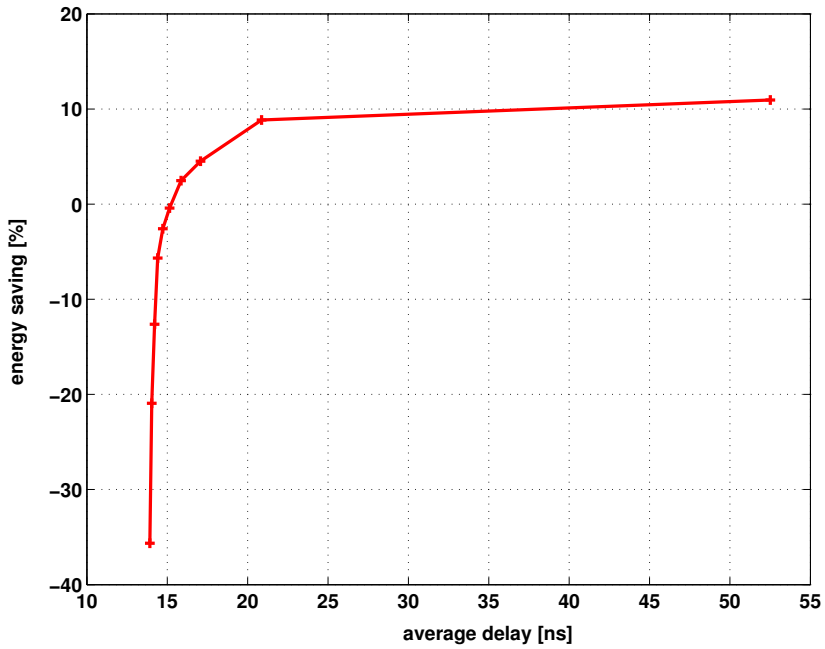

Figure 4.18: Energy saving (with respect to the classic system) as a function of the average transfer delay for different Poissonian workloads. The system becomes energy-inefficient only under high workloads requiring the interconnect to work most of the time at full speed.

new technology node: at design time the technology is poorly controlled and chances are that our system will not save significant amounts of energy. Yet, as chips will go into production and the technology will mature, a more significant saving would be possible. Classic systems would need a redesign, whereas our system will benefit automatically of the technology improvements. Regarding reliability, no residual error has been observed.

4.4.3 Robustness Towards Design Uncertainties

Fig. 4.21 simulates the effect of design hypotheses which have turned out to be too optimistic after manufacturing, for instance due to unexpected sources of on-chip noise. To simulate the self-calibrating system with a higher noise,

	energy saving		average delay variation	
	wafer		wafer	
	good	poor	good	poor
classic	21%	-8%	≤0.5%	≤0.5%
classic with codec	30%	5%	≤0.5%	≤0.5%

Table 4.3: Energy saving and average delay variation for different wafers quality, compared to the classic and classic with codec systems.

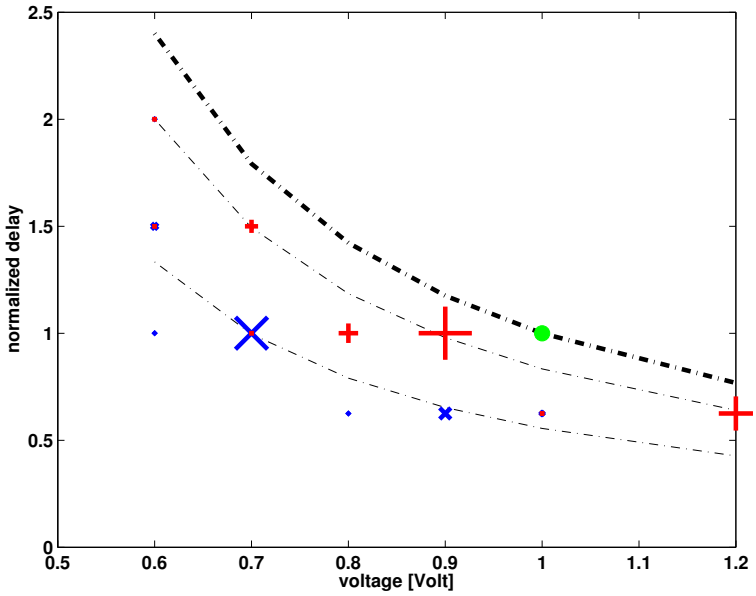

Figure 4.19: Operating points used depending on technology variations. $\circ \rightarrow$ classic system; $+ \rightarrow$ self-calibrating system on a *poor* wafer; $\times \rightarrow$ self-calibrating system on a *good* wafer. The bold line represents the worst-case relation between delay and voltage. According to the model, slightly less than 1% of the wafers are classified as good or better than good, and poor or worse than poor.

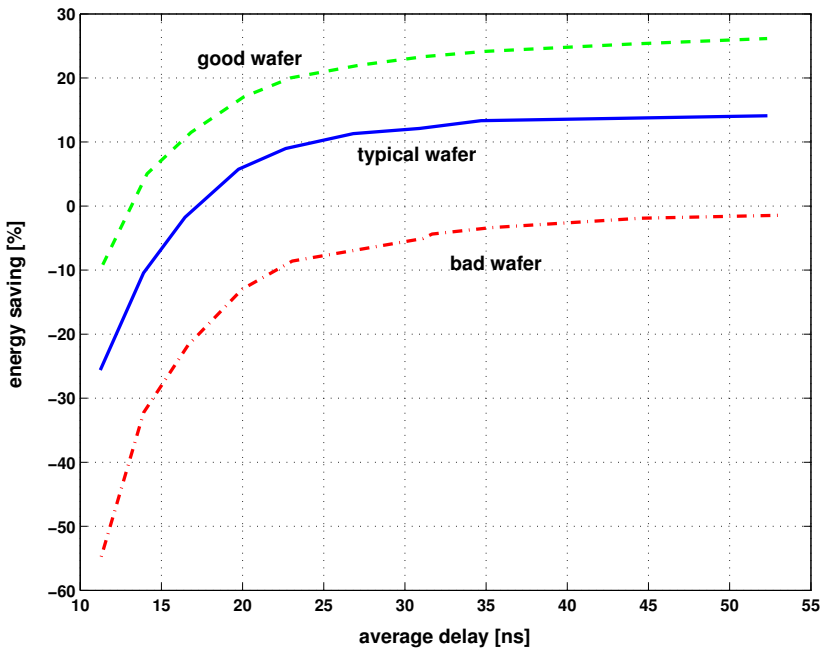

Figure 4.20: Energy saving (with respect to the classical system) as a function of the measured average transfer delay for different wafer quality. The better the wafer, the more important the energy saving.

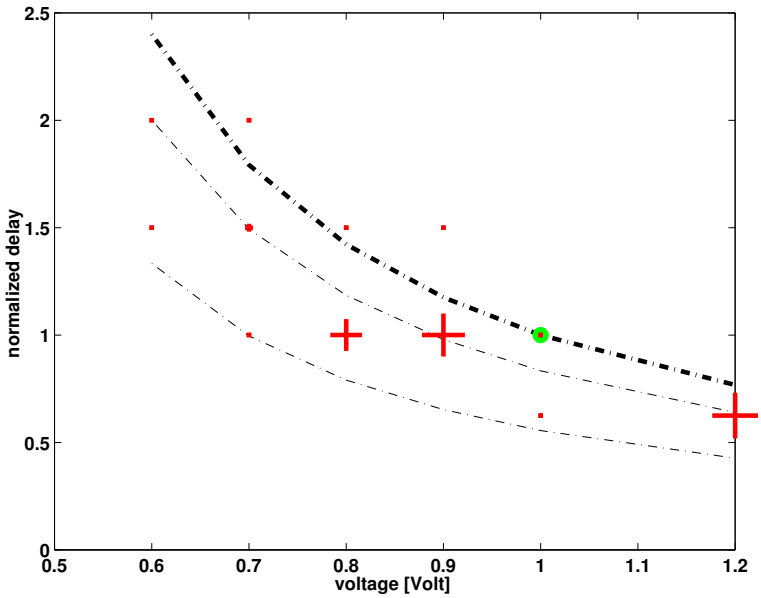

Figure 4.21: Operating points used by the self-calibrating system in the presence of strong noise. $\circ \rightarrow$ classic system; $+ \rightarrow$ self-calibrating system. The classic system has a reduced yield under these conditions, while the self-calibrating one moves to more energy-consuming, but safer operating points.

we raise the standard deviation of the additive noise from 0.05V to 0.1V and the standard deviation of the propagation delay from 0.1ns to 0.15ns. These noise parameters result in a word error rate of approximately 10^{-5} under the nominal operating conditions. It should be stressed that the classic system is not expected to work any more under this condition: if, in the normal design flow, any source of error is overlooked or underestimated—such as crosstalk or other deep sub-micron second-order effects—the manufactured chips may not work or have a very limited yield. The simulated traffic is the same artificial workload consisting of 100,000 words as described in Sec. 4.4.2. As the figure shows, the self-calibrating system adapts to the strong noise by choosing less aggressive operating points and by trading energy for robustness: energy savings shrink to 4% and 16% for the classic system and classic with codec, respectively. That is, it behaves essentially as in the presence of a poor wafer (refer to Fig. 4.19). Two residual errors have been reported during the simulation. As to the average latency, the increase amounts to 4% compared to the desired behaviour of the classic system—but the interconnect operates correctly and avoids the yield reductions incurred by the classic system.

4.5 Conclusions

The content of this chapter constitutes pioneering work about the application of digital self-calibration techniques to an on-chip link [Worm *et al.*, 2002; 2005]. We have described a self-calibrating link at the system-level and explained in detail the additional elements—timing errors detection circuitry, er-

ror recovery (i.e., the ARQ scheme), and operating point controller—imposed by self-calibration.

Regarding the encoding scheme, we have stressed the fact that detecting timing errors requires temporal redundancy. Borrowing from the LEDR encoding, we have combined the notion of phase—a binary information about the data sequencing—with standard error detecting codes such as CRCs to obtain an encoding that detects additive as well as massive timing errors. We study in detail the detection capabilities of this encoding in the next chapter.

Next, we have formulated the link control problem as a constrained minimisation. Contrary to reliability (which is measured through residual errors and has thus to be verified off-line), energy and performance can be observed by the link controller at different levels of indirections. Still, the problem remains complex because retransmissions affect energy efficiency and performance, and depend on both voltage and frequency. Due to the limited resources available to an on-line controller, we approximate the original and complex link control problem by a simple decoupled control of voltage and frequency. Our approximation relies on the observation that retransmissions are expected to occur rarely. The motivations are manifold. First, we have observed by simulation that our particular encoding scheme meets tight reliability constraints only for limited word error rates, and hence infrequent retransmissions. Another more general argument is that retransmitting often is energy inefficient and results in large transfer delays. In a DVFS capable system, large transfer delays reduce the available slack, which, in turn, increases the use of energy-costly operating points. In addition, the decoupled control leverages traditional DVFS techniques in order to determine the smallest operating frequency meeting a given performance requirement. However, the minimum voltage used for a given frequency is tuned by monitoring the occurrence of detected errors, instead of relying on worst-case defined voltage-frequency pairs. In the presented case, the controller has to request a safer operating point as soon as an error is detected in order to avoid operating points where the checker is not reliable enough.

On the one hand, our problem formulation is "link-centric" in the sense that energy and performance metrics are derived by sole considerations of the link and related elements, leading henceforth to a local optimisation. On the other hand, we argue that a global energy optimisation in a network of self-calibrating links is possible if an external unit—such as proposed by Toprak and Leblebici [2005]—provides values of the individual target delay of each self-calibrating link. Extending this principle to the voltage control problem, one could imagine that the local voltage control of a particular link would target an error rate that has been determined in order to minimise, not only the energy consumed by the link but by a larger system, e.g., a link and a cache. To our best knowledge, such questions—outside the scope of the present work—have not been addressed.

Our experimental results have emphasised the versatility of our transmission scheme. We believe that such self-calibrating circuits display features that are essential in future VLSI designs. Specifically, we have shown that a self-calibrating interconnect saves energy under both artificial and realistic workload. Our experiments have demonstrated that operating the link at sub-critical voltage compensates the overhead incurred by additional elements required by self-calibration. Furthermore, the energy saving reported depends on the quality of the manufactured chips. We have also shown that, while the yield of a tradi-

tional system reduces when the design is done with optimistic assumptions on the noise sources, self-calibration allows a circuit to trade energy for robustness without reducing the yield.

In summary, we have studied in detail a particular self-calibrating on-chip link and validated the feasibility of the concept. We have motivated a decoupled control of the link operating points. On the one hand, frequency can be estimated using well-known DVFS techniques which we have introduced in Sec. 2.1. On the other hand, the on-line control of voltage based on monitoring the error rate constitutes, in our opinion, a more interesting and challenging issue. However, there still exists a coupling between the reliability of the checker and the voltage control algorithm. For example, the control policy presented in Sec. 4.3.2 is constrained by the need of avoiding operation under large word error rates. That is, objectives in reliability and energy-efficiency are not decoupled. While double-sampling flip-flops enable to track a target error rate [Kaul *et al.*, 2005], they rely on worst-case assumptions in order to ensure reliability, which is incompatible with our goal. The remaining of this work is therefore devoted to the development of checkers detecting reliably timing errors under any error rate.

The next chapter studies analytically self-synchronising properties of the alternating-phase encoding as well as other more general schemes and gives fundamental bounds on the amount of wiring overhead necessary for detecting all timing errors. Then, in Chapter 6, we will further improve the link checker architecture presented so far, and propose novel architectures bridging the gap between high robustness and low overhead, fulfilling thereby the goal expressed in Sec. 1.3.

Chapter 5

Self-Synchronisation for Synchronous Encoding Schemes

> An expert is a man who has made all the mistakes which can be made, in a narrow field.
>
> Niels Bohr

The goal of this chapter is to derive self-synchronisation properties of synchronous encoding schemes. It reflects the particularity of the considered problem: we use a coding technique in order to detect timing errors reliably under any possible timing error rate. The perspective differs from purely asynchronous communication, because we sample the link output at well-defined instants and would like to indicate—possibly with a non-zero but small error probability—whether the sampled data is correct. On the contrary, asynchronous codes indicate with no error probability the *first* instant when a transmitted piece of data is available.

The self-synchronising scheme introduced in Sec. 4.2, namely alternating-phase encoding, raises many natural questions. For example, why does it not detect all timing errors? Is it possible to find an encoding that detects all timing errors and has the same wiring overhead? In this chapter, we will answer such questions by defining formally self-synchronisation in a synchronous context and giving fundamental bounds about bandwidth efficiency of self-synchronising encodings. To do so, we introduce *sequencing rules*, i.e., rules dictating which symbols can follow which. For example, spacer-based encoding schemes obey a sequencing rule requiring a spacer symbol to be inserted after each codeword conveying information. Self-synchronisation has been extensively studied in a framework assuming only two sequencing rules, namely spacer-based encoding schemes (e.g., dual-rail) and differential encoding where information transmitted on the channel consists of the values of signal changes, rather than signal values themselves.

When considering classic sequencing rules—spacer-based and differential encoding—self-synchronisation is equivalent to the *unordering* property of a

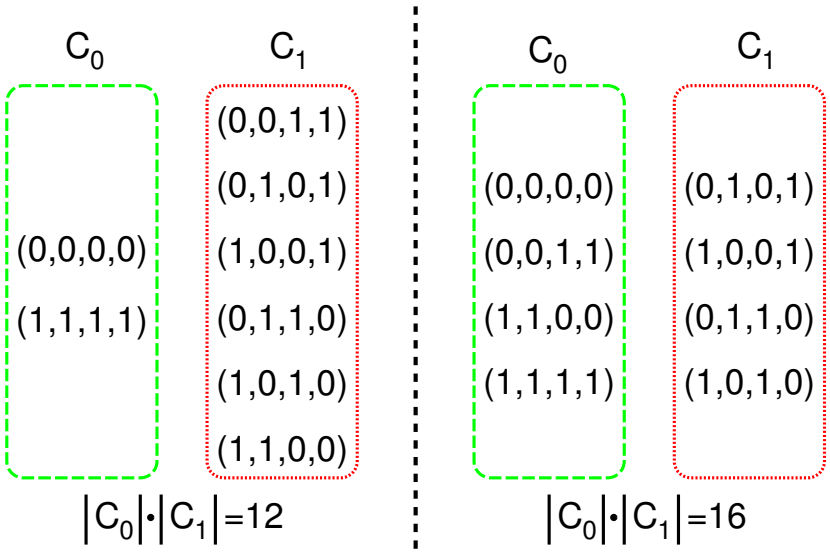

Figure 5.1: Two different self-synchronising encoding schemes that both obey a sequencing rule where two dictionaries (namely, C_0 and C_1) are used alternatively.

code [Verhoeff, 1988; Bose, 1991]. Unordered codes are such that for any codeword p, there exists no other codeword q such that $q^i = 1$ for all bit positions i where $p^i = 1$. In other words, for any two codewords p, q of an unordered code, there exist at least two bit positions i, j such that $p^i = 1; q^i = 0$ and $p^j = 0; q^j = 1$. In this context, research has been essentially focused about *unordered codes of minimum redundancy* [Berger, 1961; Freiman, 1962; Verhoeff, 1988; Varshavsky, 1990]. Such codes are called optimal self-synchronising codes.

Although alternating-phase encoding is a generalisation of spacer-based encoding, it does not fit in this framework because it is neither defined by spacer-based nor by differential sequencing rules. Moreover, its self-synchronisation properties are weaker—and thus novel—since it does not detect all possible timing errors. The originality of our approach is to consider sequencing rules that are more general than—and, indeed, include—the framework described above. While the main motivation is to study self-synchronising properties of a wider range of encodings (e.g., alternating-phase encoding), an additional motivation is to ascertain whether such a generalisation could lead to improvements on bandwidth efficiency. Moreover, we consider a larger family of sequencing rules and codes than pure asynchronous codes because we do not have the necessity of implementing the code membership test with glitch-free logic.

To illustrate the discussion, we give a motivational example showing that minimising the redundancy of unordered codes does not necessarily maximise bandwidth efficiency.

Example 4. (minimum redundancy vs. maximum bandwidth efficiency). *Fig. 5.1 depicts two particular self-synchronising encoding schemes. Both obey a sequencing rule requiring to transmit alternatively codewords from two codes C_0 and C_1. We have not yet defined formally self-synchronisation. Nevertheless, we observe that, for both schemes, the failure of any arbitrary com-*

bination of bit transition towards a new correct codeword never results in the received data to be mistaken for a valid codeword. For example, let us focus on the encoding depicted on the left and assume that the all-zero codeword has been transmitted. Next a codeword from C_1 will follow. As soon as at least a transition from the all-zero codeword towards any codeword of C_1 fails, the received codeword has a Hamming weight different than 2 and thus does not belong to C_1. Similar observations can be made for any codewords of both encoding schemes.

The code C_1 on the left side of Fig. 5.1 consists of all 4-bit vectors of weight 2 and is called a Sperner code. Although the fact is already known, we establish later in Sec. 5.4 that the Sperner code used in spacer-based or differential encoding schemes minimises redundancy, while detecting all possible timing errors. The 4-bit Sperner code is used alternatively with another code C_0 containing the all-zero and all-one spacer. The all-one spacer can be added "for free" in this case and increases thus bandwidth efficiency (we show later in Thrm. 2 that C_0 cannot be further extended without removing codewords from C_1).

On the contrary, none of the code used in the encoding scheme on the right of Fig. 5.1 is a self-synchronising code of minimum redundancy. Yet, the bandwidth efficiency of this scheme is larger than the one of the other encoding based on the Sperner code, since 16 (instead of 12) different codewords combinations can be transmitted over two cycles. Actually, the encoding scheme on the right is the 4-bit LEDR version. We conjecture later in Sec. 5.4.3.1 the optimality of LEDR under sequencing rules alternating coding dictionaries.

The particular case of Example 4 can be generalised: do self-synchronising encoding schemes exist that achieve a better bandwidth efficiency than optimal codes used with sequencing rules of the classic framework? Indeed, when considering any possible sequencing rule, it is not clear whether unordered codes of minimum redundancy (i.e., optimal self-synchronising codes) also maximise bandwidth efficiency.

We proceed as follows. In Sec. 5.1, we briefly review the state-of-the-art of self-synchronising encoding schemes. We discuss in depth later in Chapter 6 other approaches to detect timing errors, such as double sampling flip-flops. Sec. 5.2 summarises the notations used in this chapter. Then, Sec. 5.3 defines the general form of symbol sequencing rules we consider, as well as two particular types which we study in more depth. Moreover, attributes related to sequencing rules—such as self-synchronisation properties and bandwidth efficiency—are defined. Key results are stated in Sec. 5.4, which is devoted to sequencing rules that detect all possible timing errors and maximise bandwidth efficiency. We give the combination of sequencing rule and code that maximises bandwidth efficiency, i.e., show that differential encoding using unordered codes of minimum redundancy is optimal. Next, we focus on other—and thus sub-optimal—particular sequencing rules. Under these more restrictive assumptions (the choice of the sequencing rule is imposed), we show that unordered codes of minimum redundancy are not necessarily optimal. Finally, Sec. 5.5 studies the self-synchronising properties of low-overhead encoding schemes that do not detect all possible timing errors. In particular, we derive the self-synchronising properties of linear codes and alternating-phase encoding with linear codes. We give tight bounds on the residual error rate of the CRC-8 alternating-phase encoding introduced in Sec. 4.2.

5.1 Related Work

Self-synchronising (or delay-insensitive) codes enjoy the property of detecting all timing errors. As a result, they are potential solutions of the considered encoding problem. In practice, self-synchronising codes whose membership test can be implemented with glitch-free logic are used in asynchronous circuits. Dual-rail, LEDR [Dean *et al.*, 1991], and 1-of-N codes are the most well-known examples. Varshavsky studies exhaustively codes used in asynchronous circuits [1990].

A code is said to be *systematic* (or separable) if its codewords are formed by concatenating redundant bits to bits carrying information. The Sperner code is a non-systematic unordered code that minimises redundancy [Freiman, 1962]. It consists of all vectors with 1s in half (rounded to an integer if needed) of the bit positions. On the contrary, the Berger code is a minimum redundancy, systematic, unordered code [Berger, 1961]. The redundant bits of the Berger code are the binary encoding of the number of 0s in the information bits. The Sperner and Berger codes are not used in asynchronous circuits, because their membership test cannot be implemented with glitch-free logic. However, they can be used in the considered encoding problem that has no requirement about how to implement code membership test.

Errors are said to be *unidirectional* when, in any particular received word, either some 1s have been transformed in 0s, or vice-versa, but both cases never occur together in the same word. For example, errors occurring in a logic network consisting only of AND and OR gates are unidirectional, since each particular gate preserves the unidirectional property of errors. On the contrary, XOR gates or inverters do not preserve unidirectional errors. Unordered codes detect all unidirectional errors, since transforming a codeword of an unordered code into another codeword requires to change both 1 into 0 and vice-versa, which is not a unidirectional error. Researchers, especially Bose, have also studied the so-called t-unidirectional error detecting codes that detect up to t unidirectional errors, and all unidirectional errors detecting codes that correct a few (typically, a single) errors [Bose and Rao, 1982; Bose and Lin, 1985; Albassam *et al.*, 1991]. In the considered encoding problem, all unidirectional errors detecting codes are preferred, as the bit error rate can be as large as 100%.

5.2 Notations

We give the notations used in the rest of this chapter. Some special care is need to handle vectors that we address both in time and space.

A letter x (respectively y) denotes the data sent (respectively received). A letter in subscript denotes the time index. For example, x_k is the vector x sent at time $k \in \mathbb{N}$. A letter in upper-script denotes the bit position index. For example, x^i is the i^{th} bit of the vector x. The letter N denotes the bus width or, equivalently, the symbol size, while the letter K denotes the number of information bits per symbol. A code encoding K information bits into a N-bit codewords is referred to as a (N, K) code. We use the term *symbol* to denote any N-bit binary vector. We call a *codeword* a particular symbol that carries information. The operator $\oplus$ denotes the bitwise *exclusive or* of two binary values. Moreover, the operator $\oplus$ applied to a binary vector and a set of

binary vectors is to be understood as follows.

Notation 1. *Let $u \in \{0,1\}^N$ be a binary vector. Let $V \subseteq \{0,1\}^N$ be a set of binary vectors. We denote by $u \oplus V$ the set defined by*

$$u \oplus V = \left\{ w \in \{0,1\}^N \mid w = u \oplus v,\ v \in V \right\}.$$

The operator $\cdot$ denotes the bitwise *and* of two binary values. We denote by $\mathrm{w}(u)$ the weight of a binary vector u. The *weight* of a binary vector is the number of 1 it contains. The cardinality operator is denoted by $|\cdot|$.

5.3 Self-Synchronisation and Sequencing Rules

We first explain our assumption as to how an encoding scheme determines whether the data received at the channel output contains a new piece of information. We call this feature *readiness*, as expressed in the following definition.

Definition 1. (data readiness). *A binary data is* ready *if and only if all transitions from the previous piece of data have completed.*

We formulate the following definition of self-synchronisation.

Definition 2. (hard self-synchronisation). *An encoding is hard self-synchronising if and only if it detects any possible error over a timing error channel* $\mathrm{TEC}(\varepsilon_\mathrm{t})$ *with* $0 \leq \varepsilon_t \leq 1$.

While equivalent to the definition of delay-insensitivity given by Varshavsky [1990], Def. 2 is more suited to a synchronous context. Indeed, the discrete sampling instants are modelled in the timing error channel.

We contrast *hard* self-synchronisation with *soft* self-synchronisation, which is the property to detect not all but many timing errors. Informally, soft self-synchronisation could be defined as the property of an encoding scheme including some time redundancy, but that is not hard self-synchronising. The fact that hard self-synchronising schemes detect all possible timing errors is a particularity of the error model. On the contrary, there exists no equivalent concept for additive errors since the possible outputs of a binary symmetric channel spans the whole space, preventing thus to detect all possible errors. Fig. 5.2 emphasises the difference between a hard and a soft self-synchronising encoding scheme.

We proceed by defining sequencing rules. We assume that time is discrete and determined by the sampling instants. We would like to transfer an information sequence that is encoded into a sequence of symbols $\{x_k\}_{k\in\mathbb{N}}$ over a timing error channel. We denote by x_k the k^{th} symbol emitted by the encoder, and by y_k the output of the timing error channel for the input x_k. According to Def. 1, a data y_k at the output of a timing error channel is ready if and only if $y_k = x_k$. Let D_k be the set of symbols x_k that can be emitted at time k. We call the sequence $\{D_k\}_{k\in\mathbb{N}}$ the *sequence of decoding sets*. Therefore, D_k also constitutes the set of expected symbols at the channel output. If y_k is ready, then $y_k \in D_k$. The converse is not necessarily true. However, as in any syndrome decoding scheme, we assume that a membership test is used to detect whether a given output is ready or not: *The output y_k is declared ready if and only if $y_k \in D_k$.*

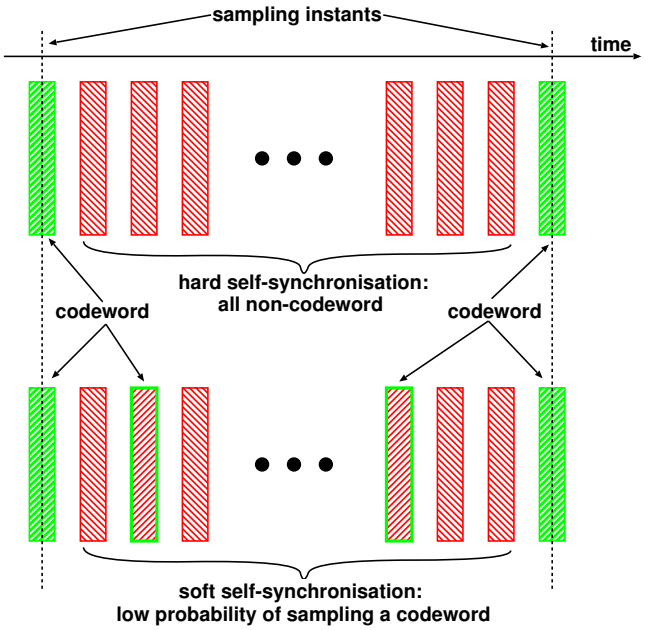

Figure 5.2: Hard self-synchronisation—contrary to soft self-synchronisation—ensures that any intermediate symbol that may be sampled while individual bit transitions are taking place is not mistaken for a valid codeword.

In asynchronous systems, this membership test has to be implemented with glitch-free logic.

So far, we have characterised an encoding scheme by its sequence of decoding sets. We now define sequencing rules as encoding schemes whose sequence of decoding sets obeys a particular rule. While we have explained that the decoding method consists in applying a membership test between y_k and D_k, we have not told how each decoding set is obtained from one time index to the next. In the most general case, a decoding set is a function of the whole transfer history $D_k = \mathrm{F}\left(x_0, x_1, ..., x_{k-1}, k\right)$. For practical reasons, we restrict our considerations to encoding schemes with "Markovian" history. More precisely, we focus on schemes having the property that each decoding set is a function only of the *last emitted symbol* and the *time index*. We call such schemes sequencing rules.

Definition 3. (sequencing rule). *A encoding scheme is a* sequencing rule *if and only if its sequence of decoding sets is such that* $D_k = F(x_{k-1}, k)$*, for all* $k \geq 1$.

Fig. 5.3 depicts the encoder structure of a sequencing rule. We now define a few sets useful to study sequencing rules.

Definition 4. (symbol set). *The symbol set* S *of a sequencing rule is the subset of* $\{0,1\}^N$ *containing all symbols that the encoder may output:*

$$S = \left\{ s \in \{0,1\}^N \mid \exists \textit{a sequence of symbols } x_0, x_1, ..., x_k \textit{ with } x_k = s \right\}.$$

Without loss of generality, we assume that the symbol set is ordered, i.e., we have, assuming an arbitrary ordering, $S = \{s^0, s^1, ..., s^{|S|-1}\}$.

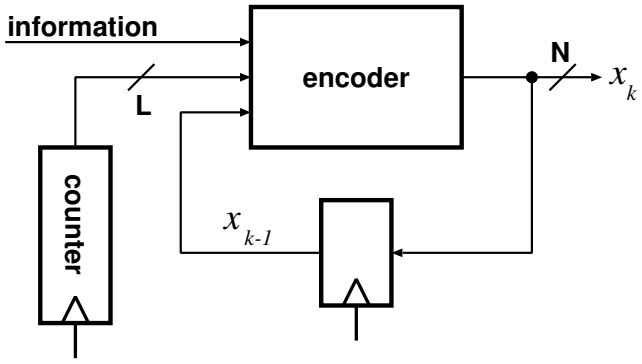

Figure 5.3: Encoder structure of a sequencing rule. The next emitted symbol depends on the previously emitted one, on the number of symbols emitted so far, and on the input information.

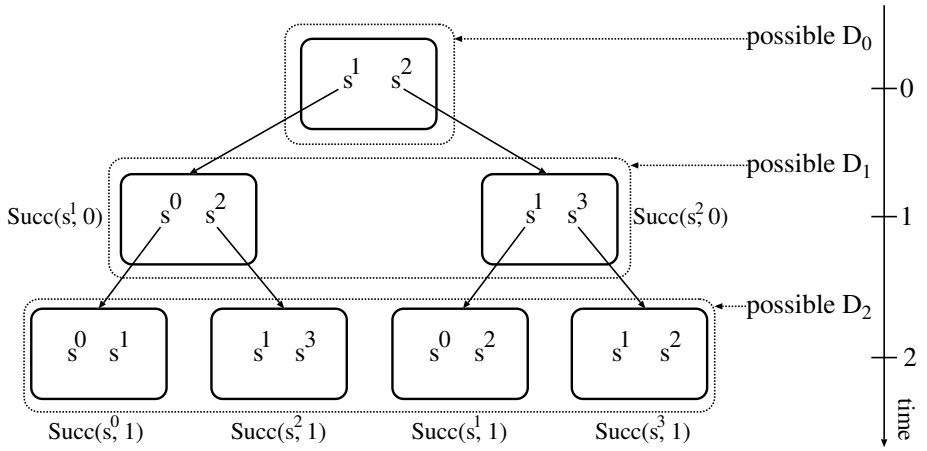

Figure 5.4: The different symbol sequences that can be generated up to time index 2, for a 2-bit sequencing rule with symbol set $S = \{s^0, s^1, s^2, s^3\}$.

We call the *symbol index* the index of a symbol in the symbol set. By convention, we write the symbol index in superscript. It follows from Def. 3 that, for each time index and for each symbol, we can define a set of symbols that may follow it and will constitute the next decoding set if the symbol is emitted.

Definition 5. (the set Succ (s, k)). *For a sequencing rule, we define*

$$Succ(s, k) = \{p \in S \mid \exists \text{ a sequence of symbols such that } x_{k+1} = p \text{ and } x_k = s\}.$$

In a sequencing rule, $D_{k+1} = \text{Succ}(s, k)$ whenever $x_k = s$. Fig. 5.4 illustrates a few sequences of symbols that can be generated according to a particular sequencing rule. We point out that Def. 3 does not describe explicitly how a sequencing rule conveys information, i.e., how particular codes are used in a sequencing rule. We address this question later in Prop. 1 and Prop. 2.

The examples given below attest that sequencing rules encompass a wide family of encoding schemes.

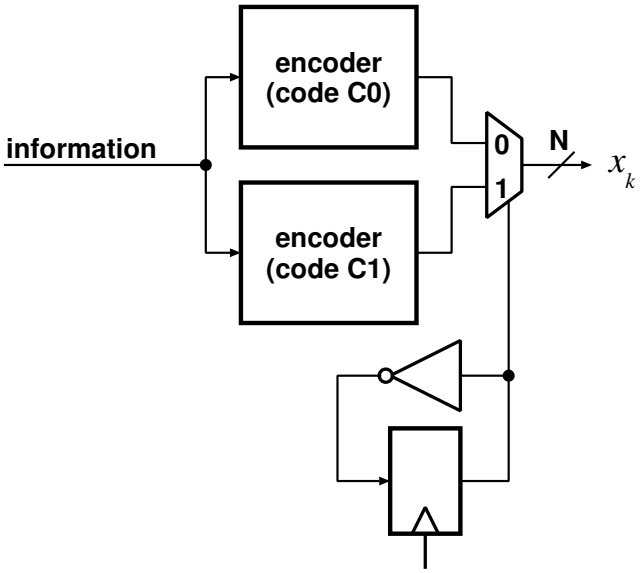

Figure 5.5: Structure of an alternating-phase encoder. The flip-flop acts as a 1-bit counter.

Example 5. (standard codes). *Consider an error detecting code C (e.g., a CRC). The corresponding sequencing rule is easily obtained. We have $S = D_0 = C$ and $Succ(c, k) = C$, for all $c \in C$ and $k \in \mathbb{N}$.*

Example 6. (alternating-phase encoding). *Alternating-phase encoding has been introduced in Sec. 4.2. The scheme transmits alternatively codewords of two codes C_0 and C_1. The symbol set is $S = C_0 \cup C_1$. Moreover, either $D_0 = C_0$, or $D_0 = C_1$. Lastly, for all $c \in C_0$ and $k \in \mathbb{N}$, $Succ(c, k) = C_1$ and for all $c \in C_1$ and $k \in \mathbb{N}$, $Succ(c, k) = C_0$. Spacer-based encoding (e.g., dual-rail) is a particular alternating-phase encoding where one of the two codes contains only a single symbol (i.e., a spacer symbol that carries no information). As shown in Fig. 5.5, the structure of an alternating-phase encoder is a particular case of the encoder structure depicted in Fig. 5.3.*

Example 7. (differential encoding). *Let C be a code. Let $c(u)$ be the codeword obtained by encoding the information u. Differential encoding is a sequencing rule outputting $x_{k+1} = x_k \oplus c(u)$ to transmit the information u in the symbol x_{k+1}, $k > 0$. Initially, x_0 can be an arbitrary symbol. We have, for all $s \in S$ and $k \in \mathbb{N}$, $Succ(s, k) = s \oplus C$. D_0 contains only one arbitrary symbol. $S = C$ if C is linear and $D_0 \subset C$. Differential encoding is also known as* encoding by changes, *or* transition signalling. *LEDR [Dean* et al., *1991] can be described with differential encoding—see later Fig. 5.8(b).*

In Def. 6 and 7, we introduce two particular sequencing rules especially relevant from an implementation point of view: time- and symbol-invariant sequencing rules.

Definition 6. (symbol-invariant sequencing rule). *A sequencing rule is* symbol-invariant *if and only if the set D_k is entirely determined by the knowledge of k, for all integers $k \geq 1$.*

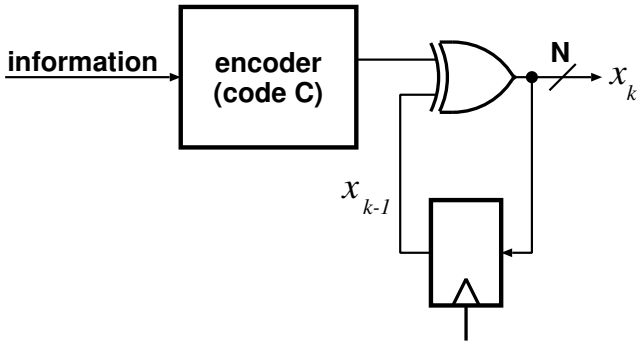

Figure 5.6: Structure of a differential encoder. The next emitted codeword depends on the previously emitted one and on the input information.

Definition 7. (time-invariant sequencing rule). *A sequencing rule is* time-invariant *if and only if the set D_k is entirely determined by the knowledge of x_{k-1}, for all integers $k \geq 1$.*

For symbol-invariant sequencing rules, x_k is entirely determined by k and the information bits, whereas for time-invariant sequencing rules, x_k is entirely determined by x_{k-1} and the information bits. The structure of these sequencing rules is illustrated in Fig. 5.7. Alternating-phase encoding (described in Example 6) is an example of symbol-invariant sequencing rule, since the next decoding set is entirely determined by the number of symbols emitted so far. The alternating-spacer protocol presented in [Sokolov *et al.*, 2005] is also a symbol-invariant sequencing rule. On the contrary, differential encoding is a time-invariant sequencing rule because the next decoding set depends only on the last emitted symbol x_k, but not on the value of k itself.

We characterise in more depth time- and symbol- invariant sequencing rules by giving the relation between codes—i.e., mapping between information and symbols—and the decoding sets of these sequencing rules.

It follows from Def. 7 that a time-invariant sequencing rule is entirely specified by the finite collection of sets $\text{Succ}\left(s^j\right)$ with $j = 0, ..., |S|-1$ (in the previous notation, the index in upper-script denotes the symbol index in the symbol set). Of course, $S = \bigcup_{j=0}^{|S|-1} \text{Succ}\left(s^j\right)$. We consider a time-invariant sequencing rule described henceforth by $|S|$ sets $\text{Succ}\left(s^j\right)$, $j = 0, ..., |S|-1$. Prop. 1 shows that such a sequencing rule can be specified by a relation between a set of distinct codes and each of the sets $\text{Succ}\left(s^j\right)$, $j = 0, ..., |S|-1$, as explained in Prop. 1.

Property 1. (time-invariant sequencing rule). *A sequencing rule is time-invariant if and only if there exists a set of distinct codes $\{C_0, C_1, ..., C_{Q-1}\}$ and a mapping M from $\{0, ..., |S|-1\}$ to $\{0, ..., Q-1\}$ such that for all $j = 0, ..., |S|-1$, $Succ\left(s^j\right) = s^j \oplus C_{M(j)}$.*

Proof. We start by assuming that there exists a set of distinct codes $\{C_0, C_1, ..., C_{Q-1}\}$ and a mapping $M : \{0, ..., |S|-1\} \mapsto \{0, ..., Q-1\}$ such that $\text{Succ}\left(s^j\right) = s^j \oplus C_{M(j)}$. This clearly means that the set $\text{Succ}\left(s^j, k\right)$ is only function of the symbol index j. Therefore, the sequencing rule is time-invariant. Now, we prove the reverse implication and consider a time-invariant sequencing rule. We show how to build a set of distinct codes and a mapping M as

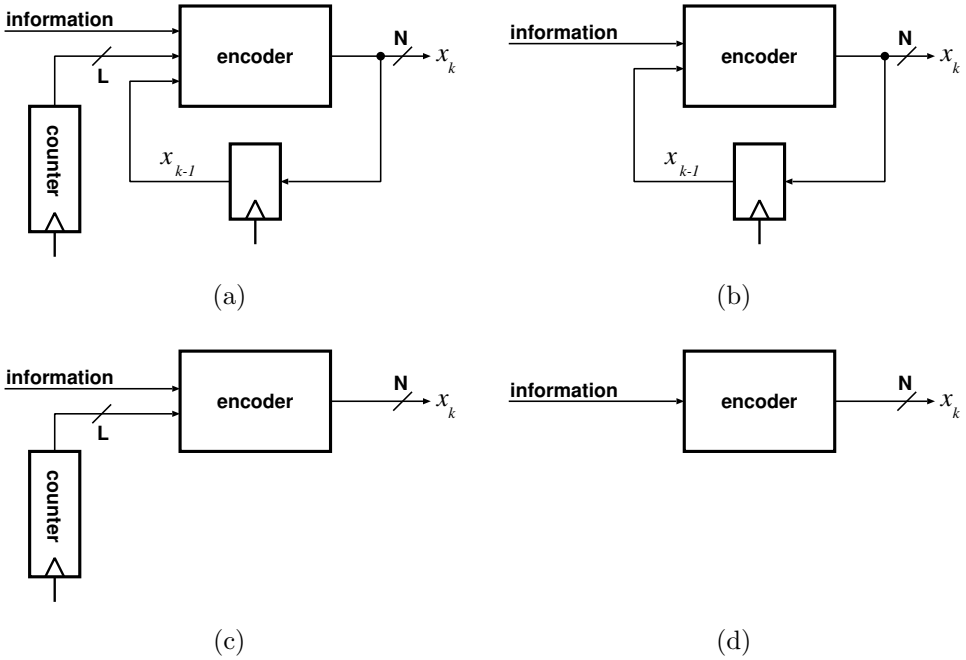

Figure 5.7: Encoder structure of a (a) general, (b) time-invariant, and (c) symbol-invariant sequencing rule. The simplest encoder structure—such as, standard codes as defined in Example 5—is drawn in case (d) and has no self-synchronising property because only spatial redundancy is added.

claimed. We proceed iteratively over the symbol set. Let $C = \emptyset$ be an empty set of codes. Then, we do the following iterations: for all $j = 0, ..., |S| - 1$, we define $C_j = s^j \oplus \mathrm{Succ}\left(s^j\right)$. If $C_j \notin C$, then we add C_j to C and set $M(j) = j$. Otherwise, there exists an integer $i < j$ such that $C_i = C_j$. In this case, we set $M(j) = i$. After a finite number of iterations, we have constructed a set of distinct codes, namely C, and a mapping M as claimed. □

The next example illustrates Prop. 1 by showing that differential encoding is a time-invariant sequencing rule.

Example 8. (differential encoding). *Let $C_0 \subset \{0,1\}^N$ be a code. The symbol set is taken (a priori) as $S = \{0,1\}^N$. The mapping M is defined as follows:*

$$M : \left\{0, 1, ..., 2^N - 1\right\} \mapsto \{0\}$$
$$M(j) = 0, \textit{ for all } j = 0, 1, ..., N.$$

As a result, we have $Succ(s) = s \oplus C_0$, for all $s \in S$, which describes indeed differential encoding. We point out that LEDR is a particular case of differential encoding, where the symbol set is given by $S = \{0,1\}^2$ and $C_0 = \{(0,1),(1,0)\}$.

We proceed by stating the equivalent of Prop. 1 for symbol-invariant sequencing rules.

Property 2. (symbol-invariant sequencing rule). *A sequencing rule is symbol-invariant if and only if there exists a set of distinct codes* $\{C_0, C_1, ..., C_{Q-1}\}$ *and a mapping* M *from* $\mathbb{N}$ *to* $\{0, ..., Q-1\}$ *such that for all* $k \in \mathbb{N}$, $Succ(x_k) = C_{M(k)}$ *and the initial decoding set* $D_0 \in \{C_0, C_1, ..., C_{Q-1}\}$.

Proof. We start by assuming that there exists a set of distinct codes $\{C_0, C_1, ..., C_{Q-1}\}$ and a mapping $M : \mathbb{N} \mapsto \{0, ..., Q-1\}$ such that $\text{Succ}(x_k) = C_{M(k)}$ and $D_0 \in \{C_0, C_1, ..., C_{Q-1}\}$. Obviously, this means that the set $\text{Succ}(x_k, k)$ is only function of the time index k. Therefore, the sequencing rule is symbol-invariant. Now, we prove the reverse implication and consider a symbol invariant sequencing rule. We show how to build a set of codes and a mapping M as claimed. We give an iterative construction. Let C be the set of codes. We initialise C as follows: $C = \{D_0\}$. Then, we do the following iterations: for all $k \geq 0$, define $C_{k+1} = \left\{p \in \{0,1\}^N \mid p \in \text{Succ}(q, k), q \in C_k\right\}$. If $C_{k+1} \notin C$, then we add C_{k+1} to C and set $M(k+1) = k+1$. Otherwise, there exists an integer $j \leq k$ such that $C_j = C_{k+1}$. In this case, we set $M(k+1) = j$. By iterating infinitely, we construct a set of codes, namely C, and a mapping M as claimed. □

We illustrate Prop. 2 by applying it to alternating-phase encoding (which we have defined in Example 6).

Example 9. (alternating-phase encoding). *Consider a set of codes* $\{C_0, C_1\}$. *Let* $S = C_0 \cup C_1$ *be the symbol set. Let* M *be a mapping defined by*

$$M : \mathbb{N} \mapsto \{0, 1\}$$
$$M(k) = k \mod 2.$$

Then, we obtain

$$Succ(x_k) = \begin{cases} C_1 & \text{if } k \mod 2 = 1, \\ C_0 & \text{if } k \mod 2 = 0. \end{cases}$$

Dual-rail is a particular case where $C_0 = \{(0,0)\}$ *and* $C_1 = \{(0,1),(1,0)\}$. *LEDR, which we have already described in Example 8 by a timing-invariant sequencing rule, is another particular case where* $C_0 = \{(0,0),(1,1)\}$ *and* $C_1 = \{(0,1),(1,0)\}$.

Some particular encoding schemes can be described by a time-invariant or by a symbol-invariant sequencing rule. For example, Fig. 5.8 depicts two different encoder structures (one symbol-invariant, the other time-invariant) for LEDR. We point out that there is no general method to express a symbol-invariant sequencing rule as a time-invariant sequencing rule, and vice-versa.

The fundamental difference between time- and symbol-invariant sequencing rules stems from the convention (namely, the mapping M) they define. Symbol-invariant sequencing rules define a convention based on the time index k. Indeed, the encoder and decoder have to keep track of how many symbols have been emitted. In practice, this would be done by the counter (Cntr) drawn in Fig. 5.7(c). On the contrary, time-invariant sequencing rules define a convention bearing on the symbol space: the encoder and decoder have to keep track of the last emitted symbol, as shown in Fig. 5.7(b).

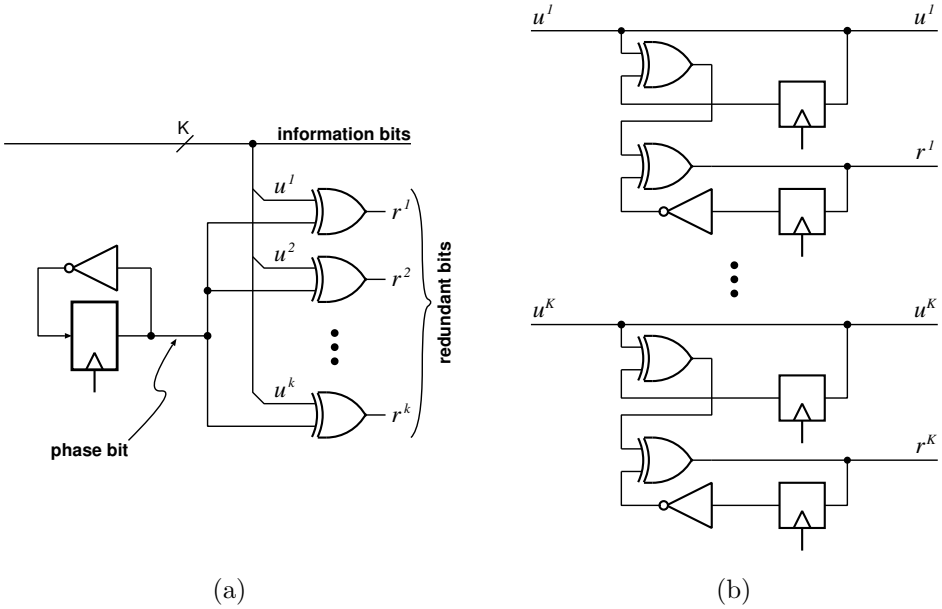

Figure 5.8: Synchronous implementations of a $(N = 2\,K, K)$ LEDR encoder: (a) symbol-invariant and (b) time-invariant. Implementation (a) is symbol-invariant since the decoding set can be determined only knowing the phase, which is a 1-bit counter. Implementation (b) is time-invariant because the next decoding set is a function only of the previously emitted symbol.

Finally, we define *bandwidth efficiency*, which is the metric we use in order to quantify how efficiently a sequencing rule uses wiring resources to transmit information. Let $T \in \mathbb{N}$ be the number of N-bit symbols needed to transfer K information bits with a particular encoding scheme. A maximum of 2^{TN} different sequences of bit vectors can be generated. However, among these, only $\prod_{i=0}^{T-1} |D_i|$ different sequences of symbols carry information. As a result, we define *bandwidth efficiency* for the transfer of K information bits by

$$\eta\,(K) = \frac{\sum_{i=0}^{T-1} \log_2 (|D_i|)}{TN}. \tag{5.1}$$

By definition, $0 \leq \eta\,(K) \leq 1$. For most encoding schemes, bandwidth efficiency is not a function of the number K of information bits transferred, nor of the number of needed symbols T. It means that the use of bandwidth is independent of how much information is transferred. We will use the metric of Eq. (5.1) to compare bandwidth efficiency of different schemes.

As illustration, we compute the bandwidth efficiency of differential encoding with the Sperner code [Freiman, 1962] and of the alternating-phase encoding.

Example 10. (bandwidth efficiency of differential encoding with the Sperner code). *The Sperner code is a non-systematic unordered code minimising redundancy. It consists of all N-bit vectors of weight $\left\lfloor \frac{N}{2} \right\rfloor$ (the choice between the ceiling or floor function is arbitrary). We denote the Sperner code by $W_N = \left\{ q \in \{0,1\}^N \mid w(q) = \left\lfloor \frac{N}{2} \right\rfloor \right\}$. Differential encoding with the Sperner*

code is the sequencing rule defined by the relation $D_k = x_{k-1} \oplus W_N$. *Let* T *be the number of symbols emitted to convey a sequence of* K *information bits (actually,* $T = K/\log_2(|W_N|)$*; however the exact value of* T *is not needed). Here,* $D_i = W_N$, *so that for all* $K \geq 1$

$$\eta = \frac{T \log_2 |W_N|}{T\,N} = \frac{\log_2 |W_N|}{N},$$

which is the redundancy of the Sperner code.

Example 11. (bandwidth efficiency of the alternating-phase encoding). *We consider the alternating-phase encoding (defined in Example 6) using two codes* C_0 *and* C_1. *Let* T *be an even number of emitted symbols, i.e.,* $T = 2 \cdot t$, *with* $t \in \mathbb{N}$. *Then, Eq. (5.1) reads*

$$\begin{aligned}\eta &= \sum_{i=1}^{t} \frac{\log_2 |C_0| + \log_2 |C_1|}{2 \cdot t \cdot N} \\ &= \frac{\log_2 |C_0| + \log_2 |C_1|}{2 \cdot N}.\end{aligned}$$

Having now defined hard self-synchronisation (Def. 2), sequencing rules (Def. 3), and bandwidth efficiency (Eq. (5.1)), a natural question is to bound the bandwidth efficiency of hard self-synchronising sequencing rules. This is the topic of the next section.

5.4 Hard Self-Synchronising Sequencing Rules

First, Sec. 5.4.1 defines *self-synchronising sets* and uses this notion to characterise hard self-synchronising sequencing rules. In Sec. 5.4.2, we derive useful properties of self-synchronising sets, which we exploit afterwards in Sec. 5.4.3 in order to upper bound bandwidth efficiency of hard self-synchronising sequencing rules.

5.4.1 Self-Synchronising Sets

We first state a few definitions.

Definition 8. (ordering relation). *Let* u *and* v *be two* N*-bit vectors. We say that* $u \leq v$ *when the ordering relation holds component-wise, i.e.,* $u \leq v \Leftrightarrow u^i \leq v^i$ *for all* $i = 1, ..., N$.

We point out that the ordering relation of Def. 8 is not conserved by translation: $x \leq y \not\Rightarrow x \oplus z \leq y \oplus z$ for all $x, y, z \in \{0,1\}^N$.

Unordered codes are easily defined using the latter ordering relation.

Definition 9. (unordered codes). *A code* C *is unordered if and only if for all* $u \in C$, *there exists no* $v \in C \setminus \{u\}$ *such that* $v \leq u$.

We introduce now self-synchronising sets.

Definition 10. (self-synchronising set for a symbol). *Let* $s \in \{0,1\}^N$ *be a binary vector. Let* $S \subseteq \{0,1\}^N$ *be a set of binary vectors. We define* S *as a* self-synchronising set *for* s *if and only if the two following conditions are met:*

(i) $s \notin S$ *and*
(ii) for all $p \in S$*, there exists no* $q \in S \setminus \{p\}$ *such that* $s \oplus q \leq s \oplus p$.

We use the notation $s \prec S$ (s *can precede* S) to denote that S is a self-synchronising set for s. In addition, for two sets of vectors $P, Q \subseteq \{0,1\}^N$, $P \prec Q$ means that $p \prec Q$ for all $p \in P$. Def. 10 essentially states that if a self-synchronising set S for a symbol s contains a symbol p, then all symbols q such that $s \oplus q \leq s \oplus p$ cannot belong to S. We emphasise later the difference between self-synchronising sets and unordered codes in Sec. 5.4.2.

We state here the condition under which a sequencing rule is hard self-synchronising.

Property 3. (hard self-synchronising sequencing rule). *A sequencing rule is* hard *self-synchronising if and only if* $x_{k-1} \prec D_k$ *for all* $k \geq 1$.

Proof. We need to show the two following assertions: (i) if an undetected error occurs at time k, then $x_{k-1} \not\prec D_k$, and (ii) if $x_{k-1} \not\prec D_k$, an undetected error can occur at time k. Let $t_k = x_k \oplus x_{k-1}$ be the transition vector. Eq. (3.9) gives the input-output relation of a timing error channel. We rewrite it as

$$\begin{aligned} y_k &= x_k \oplus e_{k,\mathrm{t}} \cdot t_k \\ &= (x_k \oplus t_k) \oplus (t_k \oplus e_{k,\mathrm{t}} \cdot t_k) \\ &= x_{k-1} \oplus \left(e_{k,\mathrm{t}} \oplus \vec{1}\right) \cdot t_k \\ &= x_{k-1} \oplus \overline{e}_{k,\mathrm{t}} \cdot t_k, \end{aligned} \tag{5.2}$$

where $\overline{e}_{k,\mathrm{t}}$ is the bitwise logical negation of the vector e. We show (i) first. If an undetected error occurs at time k, then $y_k \in D_k \setminus \{x_k\}$. Since it always holds that $\overline{e}_{k,\mathrm{t}} \cdot t_k \leq t_k$, using Eq. (5.2) on the left side and the definition of t_k on the right side, one obtains $x_{k-1} \oplus y_k \leq x_{k-1} \oplus x_k$. This violates the second condition of Def. 10 and thus $x_{k-1} \not\prec D_k$. Next, we show (ii). If $x_{k-1} \not\prec D_k$, then, from Def. 10, either (a) $x_{k-1} \in D_k$ or (b) there exist $u, v \in D_k$, $u \neq v$, such that $x_{k-1} \oplus v \leq x_{k-1} \oplus u$. In the former case, if $e_{k,\mathrm{t}} = \vec{1}$, from Eq. (5.2), one has $y_k = x_{k-1}$. If $x_k \neq x_{k-1}$, the error is then undetected because $y_k \neq x_k$ and $y_k \in D_k$. In the latter case, since $x_{k-1} \oplus v \leq x_{k-1} \oplus u$, there exists a vector $f \in \{0,1\}^N$ such that $x_{k-1} \oplus v = f \cdot (x_{k-1} \oplus u)$. Using Eq. (5.2), if $x_k = u$ and $\overline{e}_{k,\mathrm{t}} = f$, then $y_k = v$, which causes an undetected error because $y_k \in D_k$. □

Since $D_k = \mathrm{Succ}\,(s, k-1)$ whenever $x_{k-1} = s$, it is equivalent to state that a sequencing rule is self-synchronising if and only if for all $s \in S$ and $k \in \mathbb{N}$, $s \prec \mathrm{Succ}\,(s, k)$.

We illustrate Prop. 3 by applying it to differential encoding (defined in Example 7).

Property 4. (hard self-synchronisation with differential encoding). *Differential encoding with a code* C *is a hard self-synchronising sequencing rule if and only if* C *is an unordered code.*

Proof. After emitting a symbol x_{k-1}, the next decoding set is $D_k = x_{k-1} \oplus C$. By Prop. 3, hard self-synchronisation is obtained if and only if $x_{k-1} \prec x_{k-1} \oplus C$ for all $k \geq 1$. We show later in Prop. 6 that self-synchronising sets are preserved by translation, i.e. $x_{k-1} \prec x_{k-1} \oplus C$ if and only if $\vec{0} \prec C$. The latter condition expresses the unordering property of the code C. □

By a direct application of Prop. 3 to alternating-phase encoding (defined in Example 6), we obtain the following.

Property 5. (hard self-synchronisation with alternating-phase encoding). *Consider the alternating-phase encoding with two codes C_0 and C_1. The encoding is hard self-synchronising if and only if (i) $C_0 \prec C_1$ and (ii) $C_1 \prec C_0$.*

Proof. With the alternating-phase encoding, $D_k = C_0$ whenever $x_{k-1} \in C_1$ and $D_k = C_1$ whenever $x_{k-1} \in C_0$. The result follows by application of Prop. 3. □

In the particular case of alternating-phase encoding, self-synchronisation does not necessarily require C_0 or C_1 to be an unordered code. It is only the case if one of these codes is a singleton consisting of the all-zero symbol. Indeed, if $C_0 = \{\vec{0}\}$, then Prop. 5 imposes $\vec{0} \prec C_1$, i.e., C_1 has to be unordered to ensure hard self-synchronisation. The other requirement $C_1 \prec C_0$ is satisfied as C_0 is a singleton.

Lastly, we explain in the light of Prop. 3 why standard codes as defined in Example 5 are not hard self-synchronising: the same symbol can be emitted twice consecutively, which violates condition (i) of Def. 10.

5.4.2 Properties of Self-Synchronising Sets

Contrary to Def. 8, self-synchronising sets express a relation that is preserved by translation, as the following property shows.

Property 6. (invariance by translation). *Let $s, p \in \{0,1\}^N$ be two arbitrary symbols and $S \subseteq \{0,1\}^N$ be a set of symbols. Then, $p \oplus s \prec p \oplus S$ if and only if $s \prec S$.*

Proof. ($\Leftarrow$) We first show that if $s \prec S$, then $s \oplus p \prec p \oplus S$. One immediately sees that $p \oplus s \notin p \oplus S$ because $s \notin S$. Hence, condition (i) in Def. 10 is verified. Let $u \in p \oplus S$. We need to show that any $v \in p \oplus S$ such that $s \oplus p \oplus u \leq s \oplus p \oplus v$ is identical to u to prove that condition (ii) also holds. Let $x = p \oplus u$ and $y = p \oplus v$. Clearly, $x, y \in S$. As $s \prec S$, $s \oplus x \leq s \oplus y$ implies that $x = y$ and thus that $u = v$.

($\Rightarrow$) Now, we show that if $s \oplus p \prec p \oplus S$, then $s \prec S$. Since $p \oplus s \prec p \oplus S$, we have $p \oplus s \notin p \oplus S$ and thus $s \notin S$. Hence, condition (i) of Def. 10 is verified. To show that condition (ii) is also verified, let us pick any $x \in S$, and show that any $y \in S$ such that $x \oplus s \leq y \oplus s$ is necessarily identical to x. Let $u = x \oplus p$ and $v = y \oplus p$. Clearly, $u, v \in p \oplus S$ and $u \oplus p \oplus s \leq v \oplus p \oplus s$. Since $p \oplus s \prec p \oplus S$, it follows that $u = v$, which concludes the proof. □

We can now state the relation between unordered codes and self-synchronising sets. By definition, an unordered code C is a self-synchronising set for the all-zero symbol: $\vec{0} \prec C$. By a direct application of Prop. 6, we obtain that S is a self-synchronising set for the symbol s if and only if $s \oplus S$ is an unordered code. Because of these relations, self-synchronising sets can be regarded as a generalisation of unordered codes: self-synchronising sets express an unordering property with respect to a particular symbol. This additional information is essential to determine the bandwidth efficiency of a symbol-invariant sequencing rule (see later, Thrm. 2).

When considering self-synchronising sets which are not singletons, a question arises naturally. Given an arbitrary symbol, what is the largest self-synchronising set for this symbol? Prop. 6 enables a first observation: the size of a self-synchronising set for a given symbol is independent of the choice of the symbol. Indeed, if $s \prec S$, then $p \oplus S$ is a self-synchronising set for the symbol $p \oplus s$. As p can be chosen arbitrarily, $p \oplus s$ can be any arbitrary symbol, as well. Moreover, the set $p \oplus S$ has obviously the same cardinality as S. In order to answer the question, we introduce preliminary definitions and results. Then, Thrm. 1, 2, and 3 state key results.

We define first a maximum self-synchronising set.

Definition 11. (maximum self-synchronising set for a symbol). *Let $s \in \{0,1\}^N$. Let $S \subset \{0,1\}^N$ be a self-synchronising set for s. S is a* maximum *self-synchronising set for s if and only if there exists no other self-synchronising set for s, $T \subset \{0,1\}^N$, such that $|T| > |S|$.*

We also define equivalence classes in $\{0,1\}^N$. Let $s \in \{0,1\}^N$ and consider the equivalence relation between N-bit binary vectors: $p \equiv q$ if and only if $\mathrm{w}(s \oplus p) = \mathrm{w}(s \oplus q)$ (the dependence on s is not written explicitly). From this equivalence relation, we derive $N+1$ equivalence classes.

Definition 12. (equivalence classes in $\{0,1\}^N$). *Let $s \in \{0,1\}^N$. We define $N+1$ equivalence classes in $\{0,1\}^N$: $V_N^i(s) = \left\{p \in \{0,1\}^N \mid w(s \oplus p) = i\right\}$, with $0 \leq i \leq N$.*

Def. 12 implies that $V_N^i(s)$ contains exactly all vectors that are at Hamming distance i from s. Lastly, we define a graph structure on $\{0,1\}^N$.

Definition 13. (graph on $\{0,1\}^N$). *Let $s \in \{0,1\}^N$. We define a graph structure, $\mathbb{G}(s)$, on $\{0,1\}^N$. An arc goes from a node $u \in \{0,1\}^N$ to a node $v \in \{0,1\}^N$ if and only if $u \neq v$ and $s \oplus v \leq s \oplus u$.*

We use the notation $u \to v$ to mean that an arc of $\mathbb{G}(s)$ connects a node u to a node v. As illustration, Fig. 5.9 depicts the graph $\mathbb{G}(010)$. We give basic properties of the graph structure defined in Def. 13.

Lemma 1. *A node belonging to equivalence class $V_N^i(s)$ has edges*

(i) only towards nodes in equivalence classes $V_N^j(s)$, with $j < i$,
(ii) towards i nodes in equivalence class $V_N^{i-1}(s)$, and
(iii) from $N - i$ nodes in equivalence class $V_N^{i+1}(s)$.

Proof. (i) An arc connects node u to node v if and only if $s \oplus v < s \oplus u$, which implies $\mathrm{w}(s \oplus v) < \mathrm{w}(s \oplus u)$. The inequality is strict since, by definition of an arc, $u \neq v$.
(ii) and (iii) Let $u \in V_N^i(s)$. By definition, $\mathrm{w}(u \oplus s) = i$. As a result, there exist exactly i possibilities to transform the vector $u \oplus s$ into a vector $u' \oplus s$ such that $\mathrm{w}(u' \oplus s) = i - 1$ and $u \to u'$. Conversely, there are exactly $N - i$ possibilities to transform the vector $u \oplus s$ into a vector $u'' \oplus s$ such that $\mathrm{w}(u'' \oplus s) = i + 1$ and $u' \to u$. □

For a set of binary vectors Q, define $i_{\max}(Q) = \max_{0 \leq j \leq N} \left\{j \mid Q \cap V_N^j(s) \neq \emptyset\right\}$. In other words, $i_{\max}(Q)$ is the largest index among all equivalence classes that contains at least one element of Q. We show the following claim.

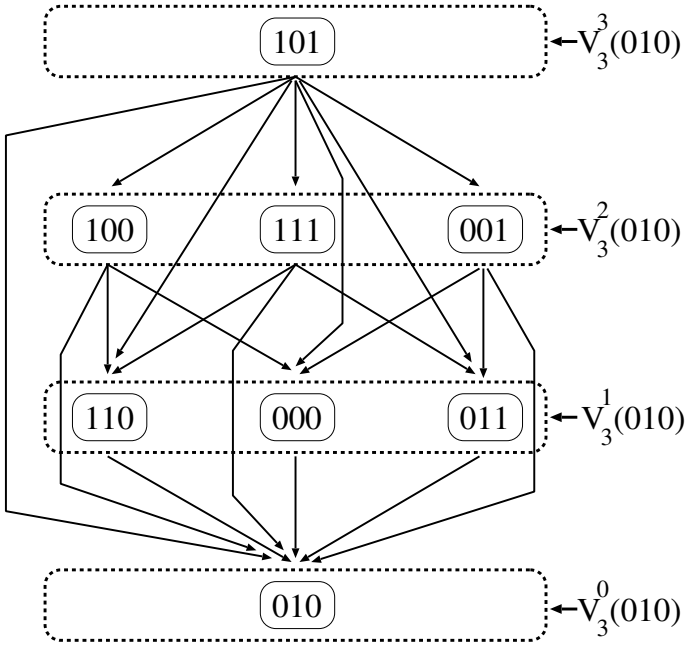

Figure 5.9: The graph $\mathbb{G}(010)$. The equivalence classes are drawn with a dotted line. The claims of Lemma 1 can be verified on this example.

Lemma 2. *Let S be a self-synchronising set for a vector s. Assume that $i_{max}(S) > \left\lfloor \frac{N}{2} \right\rfloor$. Then, there exists a self-synchronising set for s, S', such that (i) $i_{max}(S') = i_{max}(S) - 1$, and (ii) $|S'| = |S|$.*

Proof. Let $R = S \cap V_N^{i_{\max}(S)}(s)$. Knowing R, we define the following set of vectors:

$$A = \left\{ a \in V_N^{i_{\max}(S)-1}(s) \mid \exists r \in R \text{ such that } r \to a \right\}.$$

By definition of A and because S is a self-synchronising set for s, $S \cap A = \emptyset$. We observe that $(S \backslash R) \cup A$ is a self-synchronising set for s. We prove this claim by contradiction, and assume that $(S \backslash R) \cup A$ is not a self-synchronising set for s. This implies the existence of two distinct symbols $a, p \in (S \backslash R) \cup A$ such that $a \to p$. Because $S \backslash R$ is a self-synchronising set for s, we have necessarily $a \in A$. We have therefore a situation where $r \to a \to p$ with $r \in R$. This implies in turn $r \to p$, which is impossible since S is a self-synchronising set.

We obtain S' from S by performing $|R|$ iterations defined as follows. Assume that R is arbitrarily ordered: $R = \{r_1, ..., r_{|R|}\}$. Let $S_0 = S$. Define, for $i = 0, ..., |R| - 1$, $S_{i+1} = (S_i \setminus \{r_i\}) \cup \{a_i\}$, where $a_i \in A$ such that $r_i \to a_i$. We have to show that it is effectively possible to perform $|R|$ iterations, i.e., that there exist at least $|R|$ vectors $a_i \in A$ such that $r_i \to a_i$ for $i = 0, ..., |R| - 1$. This is feasible if and only if $|A| \geq |R|$. We show that the latter inequality holds by reasoning on the number of arcs that connect vectors of R to vectors of A. According to (ii) of Lemma 1, this number of arcs is exactly $|R| \cdot i_{\max}(S)$. However, each vector in A is pointed by at most $N - (i_{\max}(S) - 1)$ vectors of R (due to (iii) of Lemma 1). Therefore, this very same number of arcs is upper bounded by $|A| \cdot (N - (i_{\max}(S) - 1))$, so that we have the relation

$|R| \cdot i_{\max}(S) \leq |A| \cdot (N - (i_{\max}(S) - 1))$. We write thus

$$\frac{|R|}{|A|} \leq \frac{N - (i_{\max}(S) - 1)}{i_{\max}(S)} \leq 1,$$

where the last inequality holds if $i_{\max}(S) \geq \frac{N+1}{2} \geq \lfloor \frac{N}{2} \rfloor$, which is true by assumption.

Finally, we show that $S' = S_{|R|}$ fulfils items (i) and (ii) of the claim. By construction, item (ii) is verified as $|S| = |S_i|$ for $i = 1, ..., |R|$. Moreover, the construction ensures that $i_{\max}\left(S_{|R|}\right) = i_{\max}(S) - 1$. At last, since $S_{|R|} \subset (S \backslash R) \cup A$ and $(S \backslash R) \cup A$ is a self-synchronising set for s, $S_{|R|}$ is also a self-synchronising set for s. As a result, $S_{|R|}$ satisfies (i) and (ii). □

We state a last claim, very similar to Lemma 2. For a set of binary vectors Q, define $i_{\min}(Q) = \min_{0 \leq j \leq N} \left\{ j \mid Q \cap V_N^j(s) \neq \emptyset \right\}$.

Lemma 3. *Let S be a self-synchronising set for a vector s, and assume that $i_{min}(S) < \lfloor \frac{N}{2} \rfloor$. Then, there exists a self-synchronising set for s, S', such that (i) $i_{min}(S') = i_{min}(S) + 1$, and (ii) $|S'| = |S|$.*

A proof very similar to the one of Lemma 2 is possible.

Proof. We define the set $R = \left\{ r \in S \mid r \in V_N^{i_{\min}(S)}(s) \right\}$. Knowing R, we define the following set of vectors:

$$A = \left\{ a \in V_N^{i_{\min}(S)+1}(s) \mid \exists r \in R \text{ such that } a \to r \right\}.$$

We observe that $(S \backslash R) \cup A$ is a self-synchronising set for s. We prove this claim by contradiction, and assume that $(S \backslash R) \cup A$ is not a self-synchronising set for s. This implies the existence of two distinct symbols $a, p \in (S \backslash R) \cup A$ such that $p \to a$. Because $S \backslash R$ is a self-synchronising set for s, we have necessarily $a \in A$. We have therefore a situation where $p \to a \to r$ with $r \in R$. This implies in turn $p \to r$, which is impossible since S is a self-synchronising set.

We obtain S' from S by performing $|R|$ iterations defined as follows. Let $S_0 = S$ and define, for $i = 0, ..., |R| - 1$, $S_{i+1} = (S_i \backslash \{r_i\}) \cup \{a_i\}$, where $a_i \in A$ such that $a_i \to r_i$. We have to show that it is effectively possible to perform $|R|$ iterations, i.e., that there exist at least $|R|$ vectors $a_i \in A$ such that $a_i \to r_i$ for $i = 0, ..., |R| - 1$. Again, this is feasible if and only if $|A| \geq |R|$. According to (ii) of Lemma 1, the number of arcs joining vectors of A to vectors of R is exactly $|R| \cdot (N - i_{\min}(S))$. However, each vector in A is connected to at most $i_{\min}(S) + 1$ vectors of R (due to (iii) of Lemma 1). Therefore, the number of arcs is upper bounded by $|A| \cdot (i_{\min}(S) + 1)$, so that we have the relation $|R| \cdot (N - i_{\min}(S)) \leq |A| \cdot (i_{\min}(S) + 1)$. We write thus

$$\frac{|R|}{|A|} \leq \frac{i_{\min}(S) + 1}{N - i_{\min}(S)} \leq 1,$$

where the last inequality holds if $i_{\min}(S) \leq \frac{N-1}{2} \leq \lfloor \frac{N}{2} \rfloor$, which is true by assumption.

Finally, we verify easily that $S' = S_{|R|}$ fulfils items (i) and (ii) of the claim (same verifications as in Lemma 2). □

We have now derived the properties required to state the following theorem that constructs a maximum self-synchronising set.

Theorem 1. (maximum self-synchronising set for a symbol). *Let* $s \in \{0,1\}^N$. *Define* W_N *as*

$$W_N = \left\{ q \in \{0,1\}^N \mid w(q) = \left\lfloor \frac{N}{2} \right\rfloor \right\}.$$

$s \oplus W_N$ *is a maximum self-synchronising set for* s.

Proof. We have the set equality $V_N^{\lfloor \frac{N}{2} \rfloor}(s) = s \oplus W_N$. Indeed, $s \oplus W_N$ contains all vectors that are at Hamming distance $\lfloor \frac{N}{2} \rfloor$ of s, which is exactly the definition of $V_N^{\lfloor \frac{N}{2} \rfloor}(s)$. Due to (i) of Lemma 1, we obtain that $s \oplus W_N$ is a self-synchronising set for s. Let S be another self-synchronising set for s. We have to show that $|S| \leq |s \oplus W_N|$. We know from Lemmas 2 and 3 that S can be used to generate another self-synchronising set for s, S', such that (a) $|S'| = |S|$, and (b) $i_{\max}(S') = i_{\min}(S') = \lfloor \frac{N}{2} \rfloor$. However, (b) implies that $S' \subset V_N^{\lfloor \frac{N}{2} \rfloor}(s)$. As a result, $S' \subset s \oplus W_N$, which in turn implies $|S'| \leq |s \oplus W_N|$. Finally, combining with (a), we get the desired result $|S| \leq |s \oplus W_N|$. □

As mentioned earlier, the cardinality of a maximum self-synchronising set only depends on the size of the vector space, N, and not on the choice of a particular symbol s.

Although we have proven Thrm. 1 by relying only on properties of self-synchronising sets, we could have applied Prop. 6 to motivate the fact that finding a maximum self-synchronising set for the all-zero symbol—i.e., a maximum unordered code—is an equivalent problem. Since the Sperner code W_N is an unordered code minimising redundancy, it contains the maximum possible number of codewords for a given symbol size. While the proof given by Freiman [1962] is more complex than our approach, it provides an additional result: uniqueness. That is, the Sperner code is the unique unordered code minimising redundancy. As a corollary, the self-synchronising set defined in Thrm. 1 is also unique (besides the arbitrary choice between the floor and ceiling functions).

We give another result about maximum self-synchronising sets.

Theorem 2. (maximum self-synchronising set for a set). *Let* $s \in \{0,1\}^N$ *be a binary vector. Let* S *be a maximum self-synchronising set for* s. *Let* $Q = \left\{ q \in \{0,1\}^N \mid S \text{ is a self-synchronising set for } q \right\}$. *Then,* $Q = \{s, \overline{s}\}$.

Proof. The proof relies on the fact that a maximum self-synchronising set for a vector has a known specific form—as mentioned previously. For brevity, we assume N even. By uniqueness of a maximum self-synchronising set, it holds that for all $q \in Q$, $S = s \oplus W_N = q \oplus W_N$. Equivalently, we write

$$s \oplus q \oplus W_N = W_N.$$

We show that the latter equation has no other solution than $q = s$ and $q = \overline{s}$. Let $t \in \{0,1\}^N$ be a symbol such that $t \oplus W_N = W_N$. We show that the only solutions of the latter equation are $t = \vec{0}$ or $t = \vec{1}$. We assume first that $0 < \mathrm{w}(t) < \frac{N}{2}$. Then, there exists a vector $u \in W_N$ such that $t \leq u$. As

a result, $\mathrm{w}(t \oplus u) < N/2$, which implies $t \oplus u \notin W_N$. This shows that there exists no $t \in \{0,1\}^N$ such that $t \oplus W_N = W_N$ and $0 < \mathrm{w}(t) < N/2$. Now, assume that $\mathrm{w}(t) = N/2$, i.e., $t \in W_N$. Let $u = \overline{t} \in W_N$. Since $\mathrm{w}(t \oplus u) = N$, $t \oplus u \notin W_N$. There exists thus no $t \in \{0,1\}^N$ such that $t \oplus W_N = W_N$ and $0 < \mathrm{w}(t) \leq N/2$. Finally, let us assume that $N/2 < \mathrm{w}(t) < N$. Then, there exists a vector $u \in W_N$, such that $u \leq t$. As a result, $\mathrm{w}(t \oplus u) < N/2$, which implies $t \oplus u \notin W_N$. In conclusion, we have shown that there exist no $t \in \{0,1\}^N$ such that $t \oplus W_N = W_N$ and $0 < \mathrm{w}(t) < N$. As a result, either $t = \vec{0}$ or $t = \vec{1}$. If N is odd, a similar discussion is possible. □

The meaning of the latter theorem is that if S is a *maximum* self-synchronising set for a symbol s, then $s \oplus \vec{1}$ is the only other symbol for which S is also a maximum self-synchronising set.

We give a last result that characterises maximum *systematic* self-synchronising sets. Systematicity of a set is defined as follows.

Definition 14. (systematic set of binary vectors). *Let $S \subset \{0,1\}^N$ and let K, $0 < K \leq N$, be an integer. S is K-systematic if and only if for all $u \in \{0,1\}^K$, there exists a unique $s \in S$ such that s is the concatenation of u and v, for a certain vector $v \in \{0,1\}^{N-K}$.*

For example, a systematic (N, K) code constitutes a K-systematic set consisting of 2^K N-bit codewords, each one being obtained by concatenating redundant bits to information bits.

Theorem 3. (maximum systematic self-synchronising set). *Let $s \in \{0,1\}^N$. Let $S \subset \{0,1\}^N$ be a self-synchronising set for s. Then, S is K-systematic implies $N \geq K + \log_2(K+1)$.*

Proof. We assume that S is K-systematic and consider N as an unknown constant. We are looking for an integer R such that concatenating an R-bit vector to every vector in $\{0,1\}^K$ yields a self-synchronising set for s. K, R and N therefore satisfy $K + R = N$ and we have to show that $R \geq \log_2(K+1)$. We denote by $r(x) \in \{0,1\}^R$ the R-bit vector concatenated to the K-bit vector $x \in \{0,1\}^K$.

According to Def. 12, we define $K+1$ equivalence classes $V_K^i(s_K)$ for $i = 0, ..., K$, where s_K is the K-bit vector formed by the K most significant bits of the N-bit vector s. We also define $R+1$ equivalence classes $V_R^i(s_R)$ for $i = 0, ..., R$, where s_R is the R-bit vector formed by the R least significant bits of the N-bit vector s. We denote by $(u \mid v)$ the concatenation of vector v to vector u. Using the decomposition $\{0,1\}^R = \bigcup_{i=0}^{R} V_R^i(s_R)$, there exist orderings of vectors of $\{0,1\}^R$, $\{0,1\}^R = \{r_0, ..., r_{2^R-1}\}$ such that $s_R \oplus r_i \not\leq s_R \oplus r_j$ if $i < j$. For example, take $r_0 \in V_R^R(s_R)$, $r_i \in V_R^{R-1}(s_R), i = 1, ..., \binom{R}{R-1}$, etc. We claim that the following construction (which describes the Berger code [Berger, 1961]) forms a self-synchronising set for s: for $i = 0, ..., K$ and for all $u \in V_K^i(s_K)$, $r(u) = r_i$. To prove the claim, let us assume that there exist $u, v \in \{0,1\}^K$, $u \neq v$, such that $s_K \oplus u \leq s_K \oplus v$. This implies $\mathrm{w}(s_K \oplus u) < \mathrm{w}(s_K \oplus v)$. By construction, there exist two integers i and j such that $r(v) = r_j$, $r(u) = r_i$, and $i < j$. As $i < j$, we have $r_i \not\leq r_j$, which ensures $s \oplus (u \mid r(u)) \not\leq s \oplus (v \mid r(v))$.

Therefore, the construction gives a self-synchronising set for s with $2^R - 1 = K$, i.e., $R = log_2(K+1)$.

It remains to show that $R = log_2(K+1)$ is indeed the smallest integer needed to obtain a self-synchronising set for s. Take $K+1$ distinct vectors $u_i \in V_K^i(s_K)$, for $i = 0, ..., K$, such that $u_0 \leq u_1 \leq ... \leq u_K$. For S to be a self-synchronising set for s, we must have $r(u_i) \neq r(u_j)$, for all $i \neq j$, $0 \leq i, j \leq K$. As a result, the mapping $r(\cdot)$ defines at least $K+1$ different values, i.e., we have necessarily $2^R \geq K+1$. □

The next section characterises *optimum* hard self-synchronising sequencing rules. More specifically, we use the results derived in this section to find the combinations of sequencing rules and codes that maximise bandwidth efficiency, while ensuring the hard self-synchronising property.

5.4.3 Optimum Hard Self-Synchronising Encoding

We first give an achievable upper bound on the bandwidth efficiency of any hard self-synchronising sequencing rule.

Theorem 4. (upper bound on the bandwidth efficiency of hard self-synchronising sequencing rules). *Differential encoding with the Sperner code (defined in Example 10) has the largest bandwidth efficiency among all hard self-synchronising sequencing rules.*

Proof. Let us consider differential encoding with the Sperner code which is defined by the symbol sequencing rule: $D_k = x_{k-1} \oplus W_N$. According to Thrm. 1, each decoding set D_k is a maximum self-synchronising set for the previously emitted symbol x_{k-1}. As a result, no sequencing rule uses bandwidth more efficiently than differential encoding with the Sperner code. □

Equivalently, the bandwidth efficiency of any hard self-synchronising sequencing rule cannot be larger than $\frac{\log_2 |W_N|}{N}$, which is the bandwidth efficiency of differential encoding with the Sperner code (computed in Example 10).

Furthermore, we bound the bandwidth efficiency of hard self-synchronising sequencing rules using systematic encoding.

Theorem 5. (upper bound on the bandwidth efficiency of systematic hard self-synchronising sequencing rules). *Any hard self-synchronising sequencing rule using a systematic encoding to convey K information bits has a bandwidth efficiency less than or equal to $\frac{K}{N}$, where $N \geq K + \log_2(K+1)$.*

Proof. A systematic hard self-synchronising sequencing rule maximises bandwidth efficiency if and only if it all its decoding sets are maximum systematic self-synchronising sets. According to Thrm. 3, the symbol size N of such a sequencing rule is lower bounded by $K + \log_2(K+1)$. □

Differential encoding with the Berger code—which has been suggested by the construction in the proof of Thrm. 3—achieves the upper bound of Thrm. 5.

We state another result showing that symbol-invariant sequencing rules (defined in Def. 6) cannot achieve the upper bound on bandwidth of Thrm. 4 which is attained by a time-invariant sequencing rule.

Theorem 6. (sub-optimality of symbol-invariant sequencing rules). *Any N-bit, $N > 2$, symbol-invariant hard self-synchronising sequencing rule has a bandwidth efficiency strictly less than $\frac{\log_2 |W_N|}{N}$.*

Proof. We have shown in Prop. 2 that any symbol-invariant sequencing rule is such that for all $k \in \mathbb{N}$, there exist 2 codes C_k and C_{k+1} such that $D_k = C_k$ and $D_{k+1} = C_{k+1}$. To achieve the upper bound on bandwidth efficiency, C_{k+1} must be a maximum self-synchronising set for any symbol belonging to C_k. Then, by Thrm. 2, C_k cannot contain more than 2 symbols. Nonetheless, in order to reach the upper bound of Thrm. 4, C_k must also be a maximum self-synchronising set for any symbol belonging to D_{k-1}. Since $|W_N| > 2$ for all $N > 2$, both C_k and C_{k+1} cannot be simultaneously maximum self-synchronising sets, unless $N = 2$. This shows that $\frac{\log_2 |W_N|}{N}$ is not a tight upper bound on the bandwidth efficiency of hard self-synchronising symbol-invariant sequencing rules. □

To summarise, Thrm. 4 informs that transmitting symbols each time at Hamming distance $\lfloor \frac{N}{2} \rfloor$ from the preceding one is an optimal encoding strategy as far as hard self-synchronisation is concerned. This result is based on Thrm. 1 which exhibits the differential structure of a maximum self-synchronising set. If a systematic encoding is required, differential encoding using the Berger code is optimal. Lastly, by exploiting Prop. 2 and Thrm. 2, we have shown that symbol-invariant sequencing rule are sub-optimal, unless $N = 2$. In the latter case, LEDR (that can be described by both a timing- and symbol-invariant sequencing rule) is optimal.

In order to illustrate the results presented in this section, Fig. 5.10 plots the ratio of the bandwidth efficiency of a few well-known codes to the bandwidth efficiency of the Sperner code, assuming a sequencing rule either with a spacer, or differential encoding. Dual-rail and LEDR always have a bandwidth efficiency that is at least 55% of optimal. Yet, contrary to more bandwidth-efficient encoding, they can be implemented with a glitch-free membership test.

A natural question triggered by Thrm. 6 is as follows. Since symbol-invariant hard self-synchronising sequencing rules cannot achieve the upper bound on bandwidth efficiency mentioned in Thrm. 4 for symbol sizes larger than 2, what is a tight bound on the bandwidth efficiency of such sequencing rules? So far, we could not answer completely this question. Nevertheless, we state the problem and give a conjecture in the next section.

5.4.3.1 Optimum Symbol-Invariant Sequencing Rules

As previously discussed, alternating-phase encoding is a particular symbol-invariant sequencing rule. In all generality, a symbol-invariant encoding alternates transmission between more than two codes, while alternating-phase encoding uses only two codes, C_0 and C_1, as presented in Example 6. In the sequel, we focus the discussion on alternating-phase encoding. That is, referring to Fig. 5.5 , we look for a pair of N-bit codes (C_0, C_1) maximising bandwidth efficiency and ensuring hard self-synchronisation.

We know from Example 11 that, in order to maximise bandwidth efficiency of the alternating-phase encoding, the product $|C_0| \cdot |C_1|$ has to be maximised. Combining with Prop. 5, we obtain that a hard self-synchronising alternating-

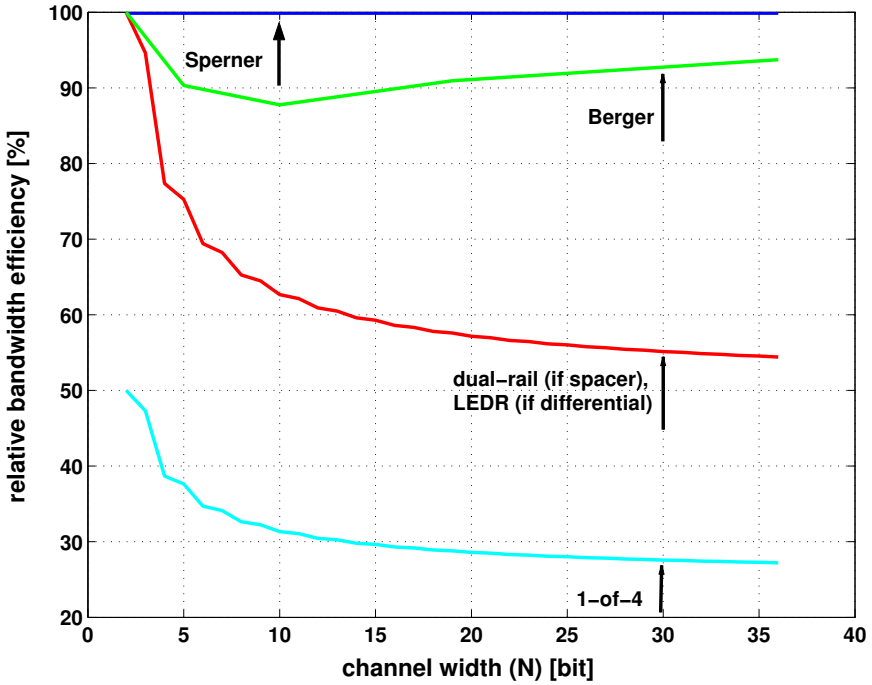

Figure 5.10: Relative bandwidth efficiency of various codes with respect to the optimal solution, i.e., the Sperner code with spacer-based or differential encoding.

phase encoding maximises bandwidth efficiency if and only if the couple of codes (C_0, C_1) solves the following problem.

Problem 2. *Let $C_0, C_1 \subseteq \{0,1\}^N$ be two codes. Maximise $|C_0| \cdot |C_1|$ under the constraint that (i) $C_0 \prec C_1$ and (ii) $C_1 \prec C_0$.*

As a consequence of Thrm. 2, the codes C_0 and C_1 solving Prob. 2 are not unordered codes of minimum redundancy, which we already verified by an example in Fig. 5.1. Furthermore, we have verified by an exhaustive search that LEDR solves Prob. 2 for even values of N not larger than 8 (solutions for odd values are not exactly LEDR, but can be derived from it). We conjecture that LEDR solves Prob. 2 for any even value of N. Since LEDR is a systematic encoding adding one redundant bit to every information bit, its bandwidth efficiency is 50%. According to this conjecture, the bandwidth efficiency of any alternating-phase encoding would be upper bounded to 50%.

While we could not solve Prob. 2, the optimality of LEDR has been shown for a related problem.

Problem 3. *Let $C_0, C_1 \subseteq \{0,1\}^N$ be two codes. Maximise $|C_0| \cdot |C_1|$ under the constraint that for all $s \in C_0$ and $p \in C_1$, $w(s \oplus p) = \lfloor \frac{N}{2} \rfloor$.*

The proof of Prob. 3 is complex and has been omitted in this thesis. Notice that if a pair of codes $C_0, C_1 \subseteq \{0,1\}^N$ satisfies $\mathrm{w}(s \oplus p) = \lfloor \frac{N}{2} \rfloor$ for all $s \in C_0$ and $p \in C_1$, clearly $C_0 \prec C_1$ and $C_1 \prec C_0$, which according to Prop. 5 ensures hard self-synchronisation. However, proving the reverse implication, namely

that if two codes satisfy $C_0 \prec C_1$ and $C_1 \prec C_0$ then $\mathrm{w}(s \oplus p) = \left\lfloor \frac{N}{2} \right\rfloor$ for all $s \in C_0$ and $p \in C_1$, is not obvious either.

Although we could not show the equivalence between Prob. 2 and Prob. 3, the requirement in the latter problem that any codeword in C_0 differs in exactly $\left\lfloor \frac{N}{2} \right\rfloor$ bit positions from any codeword in C_1 is an intuitive extension of the optimality criterion exposed in Thrm. 1. In our opinion, this remark augments the suspicion that LEDR solves Prob. 2.

5.4.3.2 Summary on Hard Self-Synchronising Encoding

So far, hard self-synchronising codes have been exclusively used in asynchronous circuits. Because this application requires the code membership test to be implemented glitch-free, only a few codes meeting this requirement—such as 1-of-N encoding, dual-rail, and LEDR—have been considered. However, as timing errors become increasingly widespread in synchronous circuits, self-synchronisation can now be studied in a broader framework where a glitch-free code membership test is not required.

While self-synchronisation has been mainly considered as a property of unordered codes [Verhoeff, 1988], our approach is more general and defines self-synchronisation as a property of the symbol sequencing rule (Prop. 3). We have shown that, for particular sequencing rules, e.g., differential encoding (Prop. 4) or spacer-based encoding (corrolary of Prop. 5) using the all-zero symbol as a spacer, self-synchronisation is indeed equivalent to the unordering property of a code. On the contrary, for other sequencing rules such as alternating-phase encoding, hard self-synchronisation is not equivalent to the unordering property of a code (Prop. 5).

In Def. 10, we have introduced self-synchronising sets, which are a generalisation of unordered codes. Self-synchronising sets express an unordering property with respect to a particular symbol. In Thrm. 1, we have given the form of a maximum self-synchronising set for a particular symbol s: this set consists of all symbols that are at Hamming distance $\left\lfloor \frac{N}{2} \right\rfloor$ (or $\left\lceil \frac{N}{2} \right\rceil$) from s. This result is fundamental since it shows that maximum self-synchronising sets (i) have a differential form, and (ii) are defined using the Sperner code. Exploiting these facts, we have deduced that differential encoding with unordered codes of minimum redundancy is the combination of sequencing rule and code maximising bandwidth efficiency. This optimal sequencing rule is time-invariant and constitutes the general upper bound on bandwidth efficiency stated in Thrm. 4.

Next, we have discussed hard self-synchronisation for symbol-invariant sequencing rules. The problem formulation leads to the notion of self-synchronising set, not only for a symbol, but for a set of symbols—a concept definitely more general that the unordering property. As a corollary of Thrm. 2, we have shown in Thrm. 6 that symbol-invariant sequencing rules cannot achieve the upper bound on bandwidth efficiency given in Thrm. 4. Furthermore, we have pointed out that unordered codes of minimum redundancy do not maximise bandwidth efficiency under such sequencing rules. Finally, we have stated Prob. 2, i.e., maximising bandwidth efficiency of alternating-phase encoding. We have conjectured the optimality of LEDR under these assumptions and given a related problem (Prob. 3) solved by this encoding scheme.

We proceed now by deriving the soft self-synchronisation properties of the widespread linear codes and of alternating-phase encoding using linear codes.

5.5 Soft Self-Synchronisation With Linear Codes

Linear codes have been extensively studied and have many applications in error detection and correction. Their capability of detecting additive errors is well-known [MacWilliams and Sloane, 1977]. In addition, researchers have developed efficient implementations. Because linear codes are so widespread, we compute their capability of detecting timing errors. More precisely, we derive in Sec. 5.5.1 the undetected error probability of systematic linear codes over the timing error channel. We call the undetected error probability of a code its *residual error rate*. As expected, the error detection capability becomes very poor under large timing error rate (because linear codes only add spatial redundancy). In Sec. 5.5.2, we discuss a modification that improves the detection capability under a large error rate.

5.5.1 Linear Codes Over the Timing Error Channel

The goal of this section is twofold. First, we believe that the timing error channel is an as relevant model of errors occurring on an on-chip link as the binary symmetric channel is. Therefore, we bound and approximate the residual error rate of systematic linear codes over a timing error channel $\mathrm{TEC}(\varepsilon_{\mathrm{t}})$. Second, the results presented in this section will be useful to derive those presented in Sec. 5.5.2.

Deriving the residual error rate is not straightforward—the full derivation is given in Appendix A. In what follows, we only sketch the main steps. We consider a (N, K) code C (i.e., a linear code that encodes K-bit information vectors into N-bit codewords) and make the following assumptions.

- C is *systematic*. That is, its codewords are obtained by concatenating $N-K$ redundant bits to the K information bits.
- C is *linear*. A code is linear if and only if it includes the all-zero codeword and the sum of any two codewords is also a codeword.
- All codewords of C are *equally likely* to be transmitted.

Systematic codes are very convenient because they simplify the encoding procedure. Moreover, linear and systematic codes have the same error detection capabilities as linear and non-systematic codes.

In addition, we make a last assumption which we need to upper bound the residual error rate. The $N-K$ parity checks of the code C can be expressed compactly in a $K \times (N-K)$ matrix, P, whose element $p_{l,i} = 1$ if and only if the information bit x_l is involved in the i^{th} parity check constraint. That is, any codeword $x \in C$ satisfies $N-K$ parity check relations

$$\bigoplus_{l=1}^{K} x_l \cdot p_{l,i} = x_{K+i},\ i = 1, ..., N-K. \tag{5.3}$$

We assume that the code C satisfies Hypothesis 1.

Hypothesis 1. *The $K \times (N-K)$ matrix P defined by the code C has full rank:*

$$\mathit{rank}(P) = \min(K, N-K) = \begin{cases} K & \text{if } K \leq N/2 \\ N-K & \text{otherwise.} \end{cases}$$

Although stronger that the previous ones, the latter assumption is often verified by codes exhibiting strong error detection capabilities. For example, it is verified by the CRC presented later in this sections and by the well-known Hamming code. Parity check codes as well as any linear code defined by two parity checks also verify Hypothesis 1 because the associated matrix P is not full rank only for degenerated cases that have no practical interest.

We proceed now with the derivation of the residual error rate, which we denote by ε_{res}. We use the notations of Sec. 3.2. ε_{res} is the probability that the received data y_k is a codeword different from the one sent, i.e., x_k. Equivalently, $y_k \neq x_k$ if and only if $\tilde{e} \neq \vec{0}$. The undetected probability ε_{res} is then

$$\varepsilon_{\text{res}} = \sum_{\tilde{e} \in C \setminus \{\vec{0}\}} P(\tilde{e}), \tag{5.4}$$

with $P(\tilde{e})$ denoting the probability of occurrence of the timing error vector $\tilde{e}$. Because C is a linear code, the transition vector is a codeword. We express now ε_{res} by conditioning on the transition vector

$$\varepsilon_{\text{res}} = \sum_{\tilde{e} \in C \setminus \{\vec{0}\}} \sum_{t \in C} P(\tilde{e} \mid t)\ P(t). \tag{5.5}$$

In addition, $\tilde{e} \leq t$, because $\tilde{e} = t \cdot e_{\text{t}}$, so that $P(\tilde{e} \mid t) \neq 0$ if and only if $\tilde{e} \leq t$. Therefore, we rewrite Eq. (5.5) as

$$\varepsilon_{\text{res}} = \sum_{\tilde{e} \in C \setminus \{\vec{0}\}} \sum_{\substack{t \in C \\ t \geq \tilde{e}}} P(\tilde{e} \mid t)\ P(t). \tag{5.6}$$

By the codeword equiprobability assumption, the transition vector t is evenly distributed over the code C. Hence,

$$P(t) = \begin{cases} \frac{1}{2^K} & \text{if } t \in C, \\ 0 & \text{otherwise.} \end{cases} \tag{5.7}$$

Combining Eqs. (5.6) and (5.7) yields

$$\varepsilon_{\text{res}} = \frac{1}{2^K} \sum_{\tilde{e} \in C \setminus \{\vec{0}\}} \sum_{\substack{t \in C \\ t \geq \tilde{e}}} P(\tilde{e} \mid t). \tag{5.8}$$

The main difficulty consists in evaluating the sum

$$\sum_{\substack{t \in C \\ t \geq \tilde{e}}} P(\tilde{e} \mid t), \tag{5.9}$$

for each non-zero timing error vector $\tilde{e}$. We derive in Appendix A a lower bound, an approximation, and an upper bound on the sum of Eq. (5.9). We obtain the

lower bound

$$\varepsilon_{\text{res}} \geq \frac{1}{2^K} \sum_{i=1}^{K} A_i \, {\varepsilon_{\text{t}}}^i \left\{1 + (1-\varepsilon_{\text{t}})^{N-K} \left[(2-\varepsilon_{\text{t}})^{K-i} - 1\right]\right\} + \frac{1}{2^K} \sum_{i=K+1}^{N} A_i \, {\varepsilon_{\text{t}}}^i, \tag{5.10}$$

and the approximation

$$\varepsilon_{\text{res}} \cong \frac{1}{2^K} \sum_{i=1}^{K} A_i \, {\varepsilon_{\text{t}}}^i \left\{1 + (1-\varepsilon_{\text{t}})^{\lfloor \frac{N-K}{2} \rfloor} \left[(2-\varepsilon_{\text{t}})^{K-i} - 1\right]\right\} + \frac{1}{2^K} \sum_{i=K+1}^{N} A_i \, {\varepsilon_{\text{t}}}^i, \tag{5.11}$$

where, in Eqs. (5.10) and (5.11), ε_{t} is the timing error rate and A_i is the number of codewords of weight i (i.e., containing exactly i bits equal to 1). The value of A_i can be readily obtained for every code [MacWilliams and Sloane, 1977].

If the code C satisfies Hypothesis 1, the following upper bound holds

$$\varepsilon_{\text{res}} \leq \frac{1}{2^K} \sum_{i=1}^{N} A_i \, {\varepsilon_{\text{t}}}^i (2-\varepsilon_{\text{t}})^{M(i)}, \tag{5.12}$$

where M depends on i as follows

$$M(i) = \begin{cases} K - i & \text{if } i \leq \mathrm{d}\left(C^{\perp}\right) - 1, \\ K - \mathrm{d}\left(C^{\perp}\right) + 1 & \text{if } \mathrm{d}\left(C^{\perp}\right) \leq i \leq N - K + \mathrm{d}\left(C^{\perp}\right) - 1, \\ N - i & \text{if } i \geq N - K + \mathrm{d}\left(C^{\perp}\right), \end{cases} \tag{5.13}$$

where $C^{\perp}$ is the code orthogonal to C and $\mathrm{d}\left(C^{\perp}\right)$ the minimum distance of $C^{\perp}$, that is the minimum number of 1 in any non-zero codeword of $C^{\perp}$. By definition, $C^{\perp}$ contains all codewords for which the inner product with all codewords of C is null. Because C is linear, its orthogonal code $C^{\perp}$ is easily obtained using the matrix P characterising C.

In case $\varepsilon_{\text{t}} = 1$, Eqs. (5.10), (5.11), and (5.12) amount to $\frac{2^K-1}{2^K}$, which is nearly 1 when K is large. As expected in such a situation, the received data y_k is exactly x_{k-1}. But, as x_{k-1} is itself a codeword, an undetected error occurs, unless $x_k = x_{k-1}$.

We have drawn in Fig. 5.11 the bounds and approximation derived for the residual error rate of a particular linear code as a function of the timing error rate ε_{t}. The figure shows that, when the timing error rate is relatively small (i.e., $\varepsilon_{\text{t}} \leq 0.1$), the residual error rate is several orders of magnitude smaller than the timing error rate. However, the residual error rate tends to 1 as the timing error rate increases to 1. This feature is a major impediment for the use of linear codes as an encoding scheme checking timing errors on a self-calibrating link: no negative feedback would be received even when the link is completely inoperative.

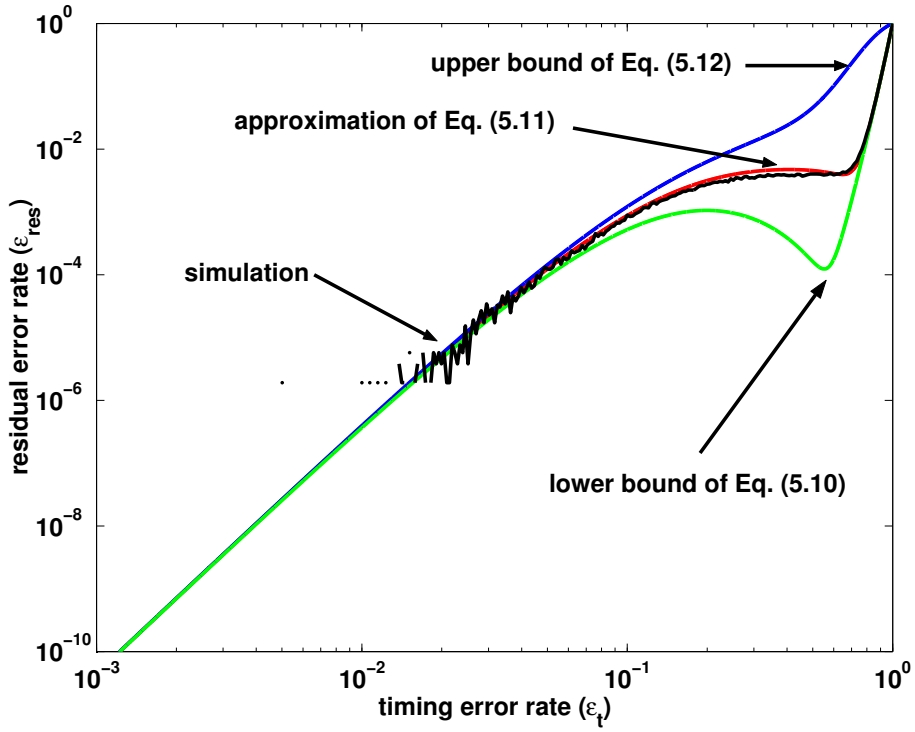

Figure 5.11: Residual error rate as a function of the timing error rate ε_t for the linear $(N = 40, K = 32)$ CRC code generated by the polynome $x^8 + x^2 + x + 1$.

5.5.2 Alternating-Phase Encoding With Linear Codes

In Fig. 4.5, we have shown that the LEDR encoding involves each information bit with the phase bit in exactly one parity check relation. A very straightforward modification enabling to reduce the wiring overhead consists in including more than one information bit into a parity check relation. For example, let us consider the transfer of 32 information bits. Instead of computing 32 redundant bits with 32 independent encoders—as LEDR would do—one could very well use, for instance an 8-bit CRC, to generate only 8 redundant bits. Each redundant bit is computed by only one encoder as a sum of some information bits and the phase bit. This example underlines that two parameters can be varied, namely (i) the number of independent encoders, and (ii) the amount of redundancy added per encoder. Since LEDR is a $(N = 2, K = 1)$ code, the number of independent encoders is in this case equal to the number of information bits to transfer. Fig. 5.12 shows different encoding options, including LEDR. In what follows, we focus on the most general case: the alternating-phase encoding with only one independent encoder. We call the option with only one CRC encoder CRC-based alternating-phase encoding.

As shown in Sec. 5.5.1, linear codes detect timing errors very poorly under large timing error rate ($\varepsilon_t \geq 0.5$), because, contrary to hard self-synchronising codes, they only add spatial redundancy. We have already illustrated the fact that detecting timing errors requires temporal redundancy, i.e., additional in-

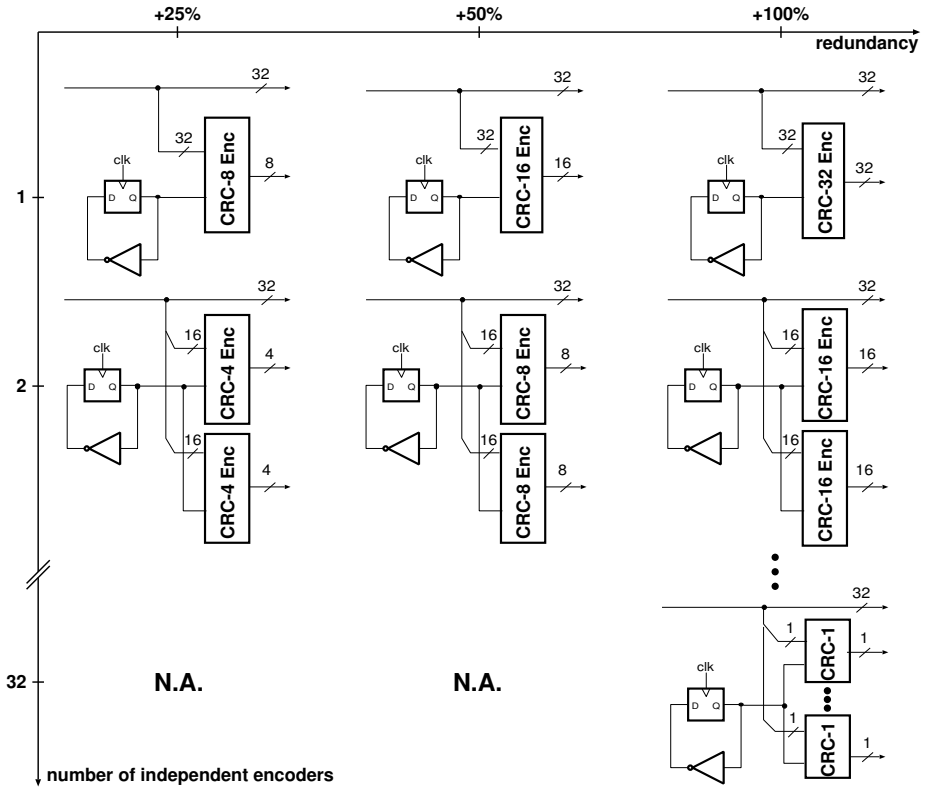

Figure 5.12: Possible options of alternating-phase encoding. The bottom right drawing depicts the LEDR code. All options with more than one independent encoder are particular cases of the top row.

formation about the *sequencing* of data—e.g., refer to Fig. 4.4. Due to the inclusion of the phase bit into several parity checks, CRC-based alternating-phase encoding detects timing errors certainly better than bare CRCs. It remains to quantify exactly the residual error rate of CRC-based alternating-phase encoding. To do so, we make the same assumptions about C as in Sec. 5.5.1 (i.e., C is linear, systematic, all codewords are equally likely to be transmitted and we assume that Hypothesis 1 is verified). We can thus apply the bounds and the approximation of the sum in Eq. (5.9) derived in Appendix A to obtain the following expressions (the proof is given in Appendix B):

$$\varepsilon_{\mathrm{res}} \cong \frac{1}{2^K} \sum_{i=1}^{K} A_i \, {\varepsilon_{\mathrm{t}}}^i \left\{ \left(1-\varepsilon_{\mathrm{t}}\right)^{\mathrm{d}-1} + \left(1-\varepsilon_{\mathrm{t}}\right)^{\left\lfloor \frac{N-K}{2} \right\rfloor} \left[\left(2-\varepsilon_{\mathrm{t}}\right)^{K-i} - 1 \right] \right\}, \quad (5.14)$$

and, if C satisfies Hypothesis 1 of Appendix A,

$$\varepsilon_{\mathrm{res}} \leq \frac{1}{2^K} \sum_{i=1}^{N} A_i \, {\varepsilon_{\mathrm{t}}}^i \left\{ \left(1-\varepsilon_{\mathrm{t}}\right)^{\mathrm{d}-1} + \left[\left(2-\varepsilon_{\mathrm{t}}\right)^{M(i)} - 1 \right] \right\}, \quad (5.15)$$

where d is the minimum distance of the code, A_i is the number of weight-i codewords in the $(N+1, K+1)$ code C, and $M(i)$ depends on the weight i as

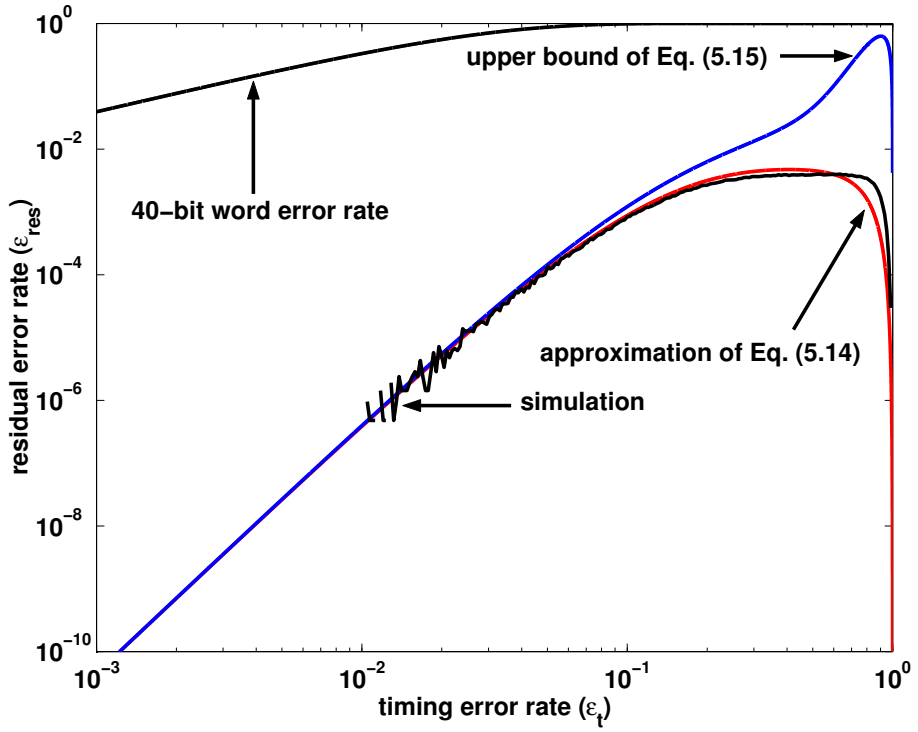

Figure 5.13: Residual error rate (ε_{res}) as a function of the timing error rate ε_{t} for the $(N = 40, K = 32)$ CRC-8 alternating-phase encoding generated by the polynome $x^8 + x^2 + x + 1$. The 40-bit word error rate curve has been plotted to show that the residual error rate of the encoding is very low (less than 10^{-10}) as long as the word error rate remains less than a few percents.

defined in Eq. (5.13). We have plotted the upper bound and approximation of Eqs. (5.15) and (5.14) in Fig. 5.13, for the CRC-8 alternating-phase encoding generated by the polynome $x^8 + x^2 + x + 1$, and validated the quality of the approximation by comparing with a simulation. The residual word error rate is actually maximum for a bit error rate equal to 0.5. In such a case, data received by the decoder is completely mixed between the previous one and the correct one, which is the worst situation for the decoder. However, as the bit error rate further increases, the residual word error rate eventually reaches 0—this is verified from the graph and the equations of both the upper bound and approximation. In this situation, the decoder detects deterministically the error because the phase bit added before the parity checks causes a single bit error, which is always detected. Such a feature is essential for a self-calibrating link that may operate sporadically under such a large error rate.

In this section, we have introduced an embodiment of *soft* self-synchronising codes. By a significant generalisation of LEDR, we have derived a new encoding family: CRC-based alternating-phase encoding. The new encoding has a bandwidth efficiency significantly larger than LEDR ($32/40 = 80\%$ in the case discussed previously vs. 50% for LEDR). The increase in bandwidth efficiency comes at the expense of a loss in reliability: the resulting encoding is

Berger code $(N = K + \log_2(K+1), K)$	$\frac{K}{K+\log_2(K+1)}$
LEDR code $(N = 2 \cdot K, K)$	$\frac{K}{2 \cdot K} = 50\%$
CRC alternating-phase code (N, K)	$\frac{K}{N}$

Table 5.1: Bandwidth efficiency of the compared encoding schemes.

no longer hard self-synchronising—as expected from the conjecture expressed in Sec. 5.4.3.1. Nonetheless, we have bounded analytically and accurately approximated the induced loss of reliability.

5.5.3 Case Study

We illustrate the discussion lead in this chapter by comparing the following systematic encoding options:

- differential encoding with the Berger code,
- the LEDR encoding, and
- the CRC-8 alternating-phase encoding,

under the comparison metrics: bandwidth efficiency, robustness to timing and additive errors. We perform the comparison for a common and fixed wiring resource (i.e., bus width N). More precisely, we consider a narrow bus $N = 10$ and a wider bus $N = 38$. The different metrics are computed as follows. Since the three considered schemes are systematic, the computation of bandwidth efficiency is straightforward. Indeed, the bandwidth efficiency of each encoding scheme is K/N, as shown in Table 5.1.

The Berger encoding and LEDR are hard self-synchronising. On the contrary, the CRC-8 alternating-phase does not detect all timing errors: its residual error rate is approximated by Eq. (5.14).

To assess the robustness towards additive errors, we compute the residual error rate over a binary symmetric channel (BSC). The residual error rate of the CRC-based alternating-phase encoding is easily obtained [MacWilliams and Sloane, 1977]. Denoting ε_{a} the additive bit error rate, we have

$$\varepsilon_{\mathrm{res,a}}^{\mathrm{CRC}} = \sum_{i=1}^{N} A_i \, {\varepsilon_{\mathrm{a}}}^i \, (1 - \varepsilon_{\mathrm{a}})^{N-i}, \tag{5.16}$$

with A_i the number of weight-i codewords. We derive now the residual error rate of the Berger encoding and LEDR over the BSC. To do so, we need a preliminary result.

Property 7. (residual error rate of several identical encoders). *Consider a $(M \cdot N, M \cdot K)$ encoding scheme consisting of M identical (N, K) encoders. Each individual (N, K) encoder transmits over a binary symmetric channel (BSC) with bit error rate ε_a. Assume that the errors occurring on a particular BSC are statistically independent from the ones occurring on the other BSCs. Let ε_{res} be the residual error rate of each individual (N, K) encoder. Let ε_{res}^{tot} be the residual error rate of the global $(M \cdot N, M \cdot K)$ encoding*

scheme. Then, ε_{res}^{tot} *is given by*

$$\varepsilon_{res}^{tot} = \sum_{i=1}^{M} \binom{M}{i} \ {\varepsilon_{res}}^{i} \ (1-\varepsilon_a)^{N(M-i)} \ . \tag{5.17}$$

Proof. An undetected error on the $(M \cdot N, M \cdot K)$ encoding scheme occurs if and only if one of the i, $1 \leq i \leq M$, following events occurs: i among the M independent encoders have each one an undetected error and the $M-i$ others have no error at all. This is exactly the probability computed in Eq. (5.17). □

Using Prop. 7, the residual error rate of LEDR is easily obtained, since a $(N = 2 \cdot K, K)$ LEDR encoder consists of K independent encoders, each one encoding one information bit into a 2-bit codeword. Moreover, the only error a 2-bit LEDR encoder does not detect is a one affecting both bits, because such errors leave the phase of the corrupted data unchanged. These errors happen with probability ${\varepsilon_{\mathrm{a}}}^2$. Applying Prop. 7 with $N = 2$, $M = K$ and $\varepsilon_{\mathrm{res}} = {\varepsilon_{\mathrm{a}}}^2$ yields directly

$$\varepsilon_{\mathrm{res,a}}^{\mathrm{LEDR}} = \sum_{i=1}^{K} \binom{K}{i} \ \varepsilon_{\mathrm{a}}^{2\,i} \ (1-\varepsilon_{\mathrm{a}})^{2\,(K-i)} \ . \tag{5.18}$$

The residual error rate of the Berger code over the binary symmetric channel is derived in Appendix C. In the wide bus scenario presented, the $(N = 38, K = 30)$ Berger encoder consists of two $(N = 19, K = 15)$ encoders. The resulting residual error rate is obtained by using Prop. 7.

We have plotted in Figures 5.14 and 5.15 the residual word error rate of the considered encoding schemes as a function of the additive bit error rate over a narrow and wide bus. We point out that, in the narrow bus case, the 3-bit CRC code presented is the one with the lowest residual error rate among the 8 possible other 3-bit CRC codes.

Finally, we have reported in Tables 5.2 and 5.3 the bandwidth efficiency, the kind of self-synchronisation (i.e., hard or soft) and the robustness to additive errors. Regarding the latter metric, we have computed the orders of magnitude change in the residual error rate for a one order of magnitude change in the raw bit error rate. We call this metric δ. Equivalently, δ is the slope of the curves obtained in Figures 5.14 and 5.15, that is $\delta = \frac{\Delta(\log(\varepsilon_{\mathrm{res,a}}))}{\Delta(\log(\varepsilon_{\mathrm{a}}))}$. In both scenarii,

	CRC APC	Berger	LEDR
bandwidth efficiency	70%	70%	50%
self-synchronisation	soft	hard	hard
δ	2	2	2

Table 5.2: Comparison metrics for the narrow bus (N= 10).

differential encoding with the Berger code outperforms LEDR: the former has a better bandwidth efficiency than the latter and the same robustness to both timing and additive errors. While differential encoding with the Berger code stands out as the best option in the narrow bus scenario, the CRC-8 alternating-phase encoding may be preferred over a wide bus dominated by additive errors. Indeed, this encoding shows a better robustness to additive errors than the

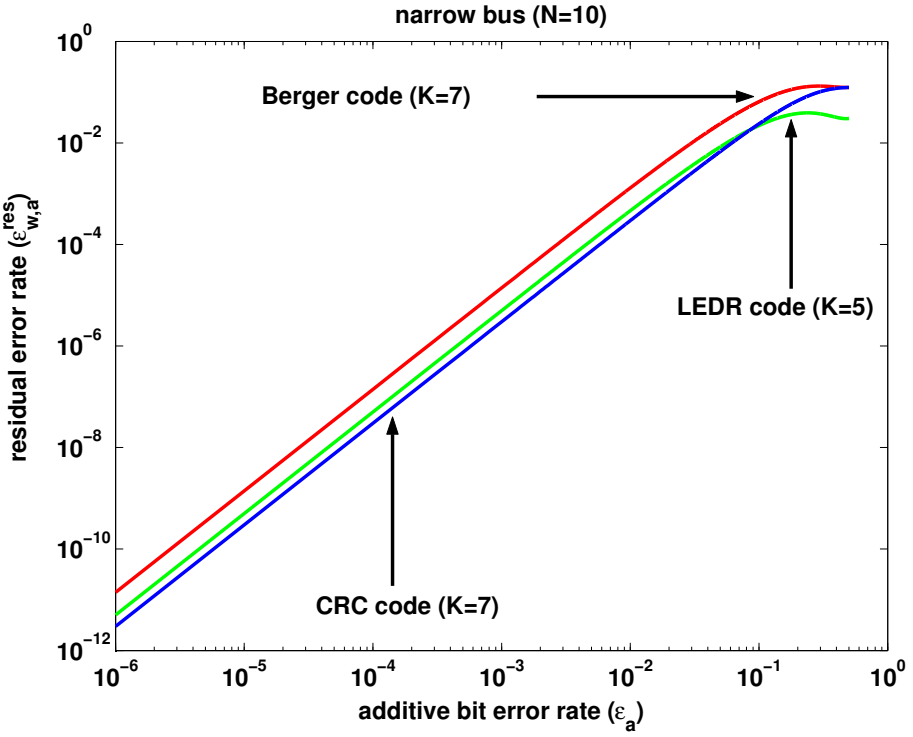

Figure 5.14: Residual error rate as function of the additive bit error rate ε_a over a 10-bit wide BSC for the $(N = 10, K = 7)$ Berger code (top), the $(N = 10, K = 5)$ LEDR code (middle), and the $(N = 10, K = 7)$ alternating-phase encoding generated by the polynome $x^3 + x^2 + 1$ (bottom).

Berger code: its residual error rate decreases approximately of 3.5 orders of magnitude as the bit error error rate decays of one order of magnitude (vs. 2 for the Berger code). Moreover, the hard self-synchronisation property has to be de-emphasised in the sense that no real-life link introduces only timing errors. In fact, all hard self-synchronising encodings have a non-zero residual error rate as soon as additive errors are considered.

	CRC APC	Berger	LEDR
bandwidth efficiency	79%	79%	50%
self-synchronisation	soft	hard	hard
δ	3.5	2	2

Table 5.3: Comparison metrics for the wide bus (N= 38).

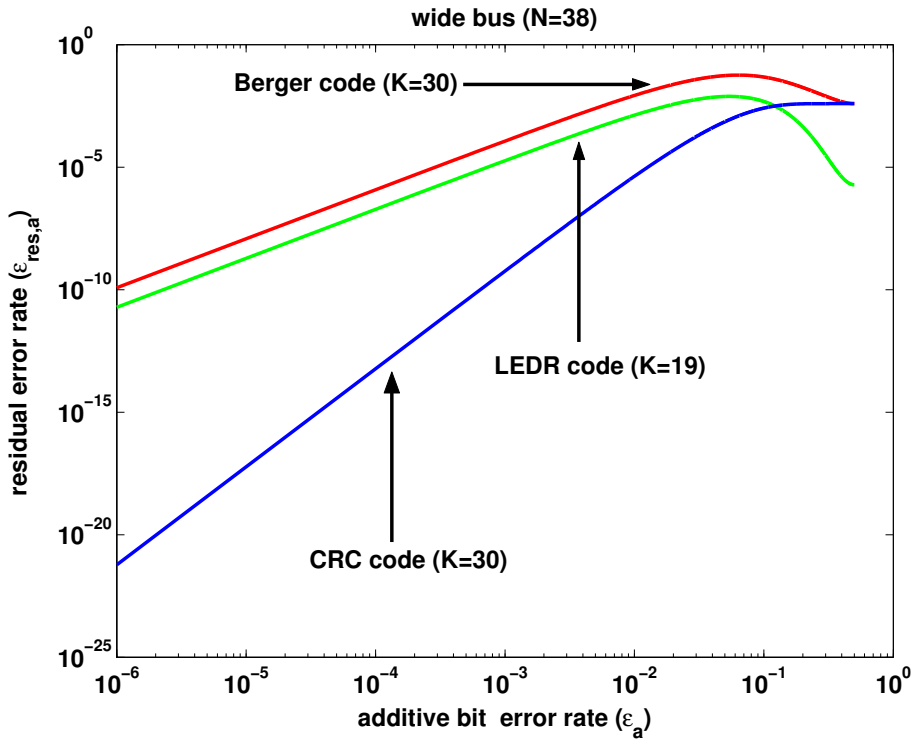

Figure 5.15: Residual error rate as function of the additive bit error rate ε_a over a 38-bit BSC for the $2 \cdot (N = 19, K = 15)$ Berger code, the $(N = 38, K = 19)$ LEDR code, and the $(N = 38, K = 30)$ alternating-phase encoding generated by the polynome $x^8 + x^2 + x + 1$.

5.6 Conclusions

Sec. 5.6.1 summarises the achievements of the chapter. Next, Sec. 5.6.2 discusses the limitations of coding to further improve the robustness of alternating-phase encoding, which links with the next chapter.

5.6.1 Summary of Achievements

Due to the scaling of CMOS technology and reduction in noise margins, we argue that timing errors—and not only additive errors—are becoming a concern in synchronous circuits. Motivated by this observation and the particular features of the problem considered in this thesis, we have expressed and studied self-synchronisation in the context of a synchronous link.

In a first part of this chapter, we have presented a comprehensive taxonomy of hard self-synchronising encoding schemes. The originality of our approach relies on the definition and study of symbol sequencing rules. We have shown that, depending on the considered symbol sequencing rule, hard self-synchronising encoding schemes maximising bandwidth efficiency do not necessarily use unordered codes of minimum redundancy—although the latter have been known so far as the optimal coding solution as far as self-synchronisation is concerned. In

particular, the theorems stated in Sec. 5.4.3 and the conjecture of Sec. 5.4.3.1 bring a fundamental insight into the self-synchronisation capabilities of synchronous systems. We refer the reader to Sec. 5.4.3.2 for a detailed summary about hard self-synchronisation.

In a second part, we have contrasted hard self-synchronisation with soft self-synchronisation, i.e., encoding schemes able to detect many but not all timing errors. We have presented a particular embodiment, CRC-based alternating-phase encoding, that exhibits unique detection capabilities towards both timing and additive errors. First, we have bounded and approximated the residual error rate of systematic linear codes over a timing error channel. This result complements the well-known formula for the residual error rate of linear codes over the binary symmetric channel and, in our opinion, is as much important to estimate the level of reliability offered by these codes. Moreover, the result confirms the expectations that linear codes detect very poorly timing errors under a large error rate. In order to improve the detection capability under large error rate, we have proposed a novel family of encoding scheme: CRC-based alternating-phase encoding which we derived from a significant generalisation of LEDR. As CRC-based alternating-phase encoding enables to reduce the wiring overhead to a level that does not ensure any more hard self-synchronisation, we have developed tight bounds on the residual error rate of this encoding over a timing error channel.

Lastly, we have applied some of the results derived in this chapter to determine which of a few particular hard and soft self-synchronising schemes are most adapted to transmission over a link subject to both timing and additive errors.

5.6.2 Fundamental Limits of Soft Self-Synchronising Encodings

As shown in Fig. 5.13, the CRC-based alternating-phase encoding does not detect timing errors reliably under moderate bit error rates. Thus, the downside is that the operating point controller is constrained to avoid the weak spot of this checker.

In order to improve the robustness to timing errors, one may wonder whether a hard self-synchronising encoding scheme exists with the same additive errors detection capabilities as alternating-phase encoding. Equivalently, can coding improve the latter scheme such that it detects all timing errors and preserve its detection capabilities towards additive errors? Referring to the conjecture stated in Sec. 5.4.3.1, we know that, among the hard self-synchronising encodings alternating two dictionaries, LEDR is the one maximising bandwidth efficiency. As a result, using coding to improve the robustness to timing errors of alternating-phase encoding means modifying the encoding towards LEDR, as suggested in Fig. 5.16. However, such modifications imply two negative aspects. First, the bandwidth efficiency cannot exceed the one of LEDR, i.e., only 50%. Second, LEDR has the particularity of including only a single information bit in each parity check. Therefore, it has a poor robustness to additive errors.

In summary, coding can further increase the timing error detection capabilities of alternating-phase encoding, but only at the expense of bandwidth efficiency and robustness to additive errors. Henceforth, the next chapter proposes double sampling—instead of coding—to enhance *simultaneously* the robustness and bandwidth efficiency of the CRC alternating-phase checker.

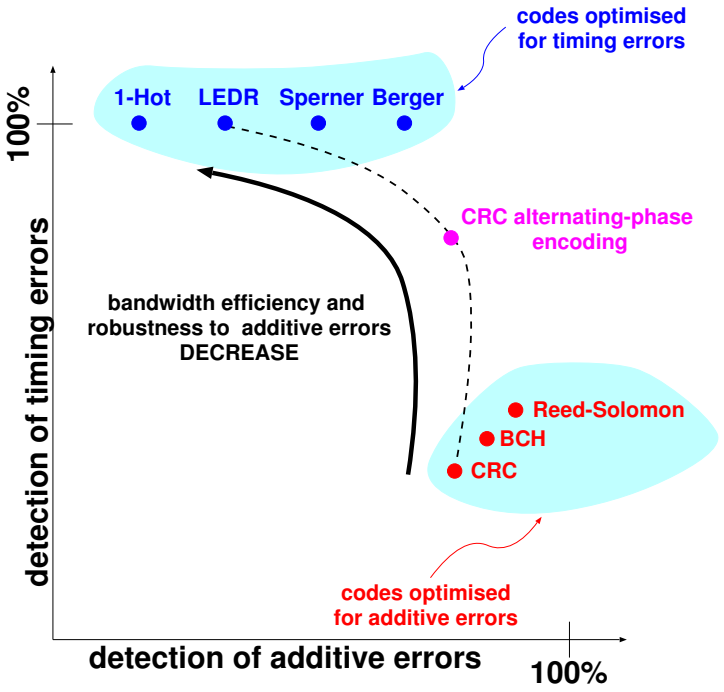

Figure 5.16: The graph sketches how the novel CRC-based alternating-phase encoding compares with existing encoding schemes targeting either timing or additive errors. The results developed in the chapter show that improving the robustness to timing errors of the CRC alternating-phase encoding entails both a reduction of bandwidth efficiency and a lower robustness towards additive errors, as indicated by the thick arrow.

Chapter 6

Double Sampling, Coding, or Both?

> On ne connaît que les choses que l'on apprivoise.
>
> Antoine de Saint-Exupéry.

Before answering the question posed in the title, we contrast qualitatively in Sec. 6.1 the detection capabilities of Razor flip-flops (introduced in Sec. 2.2.1) and soft self-synchronising codes (abbreviated soft SSC and introduced in Sec. 5.5). We emphasise the strong complementary of these two checkers and introduce a novel reliability metric specific to checkers for self-calibrating designs. Exploiting their complementarity, Sec. 6.2 is devoted to the optimal combination of double sampling checkers with soft SSC. First, Sec. 6.2.1 describes in detail a checker combining Razor flip-flops with a soft SSC. In particular, we show in Sec. 6.2.2 the superiority of this combined checker by comparing thoroughly its robustness to timing errors with, individually, Razor flip-flops and the soft SSC.

Then, Sec. 6.2.3 proposes a variation on the combined checker, whereby reliability is further enhanced at the expense of error correction capabilities. Our conclusions on the robustness of the combined checkers—presented in Sec. 6.2.5—is that double sampling without any error correction combined with a soft SSC is optimal as far as reliability is concerned. The hardware complexity of the combined checkers is briefly discussed in Sec. 6.2.4. Interestingly, despite their strong robustness, their hardware overhead is minimal.

Next, Sec. 6.3 develops briefly the question of comparing checkers for self-calibrating circuits. Specifically, we discuss whether information about the operating point control policy is required to compare checkers. In Sec. 6.4, we propose research directions to apply checkers combining double sampling flip-flops with soft SSC not only to communication, but also to computing elements such as adders. Finally, Sec. 6.5 concludes by summarising the contributions of the chapter.

6.1 Qualitative Comparison of Razor Flip-Flops with Codes

In this section, we emphasise the complementarity of Razor flip-flops and soft SSC by a qualitative discussion. The conclusions drawn from the comparison are confirmed later by the simulations presented in Sec. 6.2.2. Finally, we introduce a reliability metric specific to checkers used in a self-calibrating design.

First, we show qualitatively how the *residual* and *reported* error probabilities of each checker vary as a function of the link supply voltage. This information characterises completely the quality of a checker. As with any checker, the data delivered to the end-user should not be corrupted by residual errors. In addition, the checker informs the controller of each detected error, so that unsafe operating points—supply voltage, in the particular case—can be dynamically avoided. The reader may want to read Appendix D where the detection capabilities of both checkers are compared in the whole delay-voltage plane using the error model introduced in Chapter 3. The comparison following below is a snapshot focusing on a fixed delay value. Essentially, the conclusions drawn in Appendix are similar to the ones that follow.

Let t_p be the propagation delay through a bit line. We consider t_p a random quantity parametrised by the supply voltage v_ch: for each particular value of v_ch, t_p is characterised by a probability distribution (details are given in Appendix D). We perform a qualitative comparison because the points we want to make depend on important features of the checkers and not on a particular bit error rate model (i.e., on the actual relation between t_p and v_ch). By definition, a bit line is affected by a timing error whenever

$$t_\mathrm{p} \geq T_\mathrm{c} \tag{6.1}$$

with T_c the sampling period.

A single Razor flip-flop is affected by a residual bit error if and only if

$$t_\mathrm{p} \notin [T_\mathrm{d}; T_\mathrm{c} + T_\mathrm{d}] \tag{6.2}$$

with T_d the delay between the clock fed to the main flip-flop and the clock fed to the shadow latch. If a timing error occurs on a bit line and is such that $t_\mathrm{p} \geq T_\mathrm{c} + T_\mathrm{d}$, then the error is undetected since both the main flip-flop and shadow latch hold the same data piece. Therefore, even if Razor flip-flops on other bit lines detect a timing error, the data output for the particular line where $t_\mathrm{p} \geq T_\mathrm{c} + T_\mathrm{d}$ is corrupted. On the contrary, if the propagation time on a bit line is so short that $t_\mathrm{p} \leq T_\mathrm{d}$, then the next data piece is latched too early by the shadow latch, which causes the correct data held by the main flip-flop to be invalidated since a mismatch is detected. This phenomenon is called a *short-path* error. It is worth noting that the data delivered on a bit line where a short-path error occurs is corrupted—indeed, the next data piece is delivered—even though no timing error actually occurred. As a result, reliable operation of a single Razor flip-flop occurs in limited range of voltages, where the probability of a short-path and an undetected timing error are both acceptably low. In practice, this is ensured by introducing buffers delaying data arrival—avoiding thus short-path errors—while undetected timing errors are avoided thanks to worst-case characterisation of t_p.

Considering a group of Razor flip-flops (such as in a Razorised bus), a word error is reported when at least one of the Razor flip-flops reports an error.

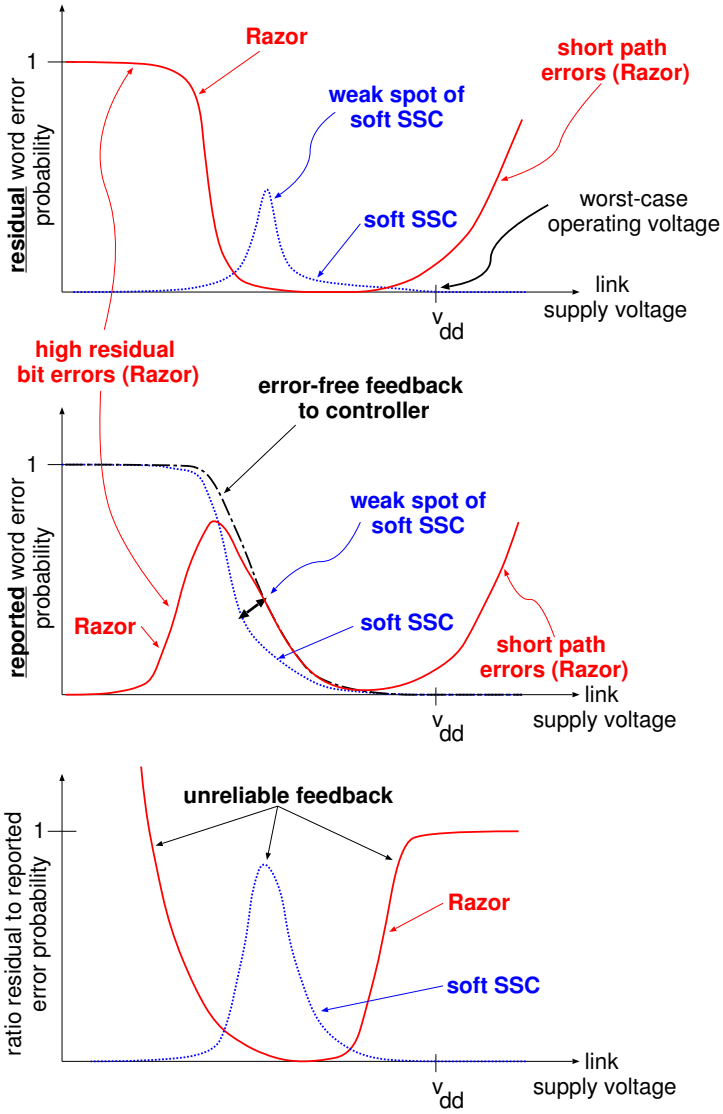

Figure 6.1: Top and middle: residual (top) and reported (middle) word error probability as a function of supply voltage for a bus terminated by Razor flip-flops or a soft SSC. Bottom: ratio of residual to reported error probability for each checker. The reliability metric plotted in the vertical axis expresses clearly and compactly the complementarity of both checkers.

Equivalently, the error signal fed to the controller is obtained by *OR*ing the individual error signals of each Razor flip-flop. A residual word error occurs whenever at least one Razor flip-flop is affected by a residual bit error, even though other Razor flip-flops than the one(s) causing a residual error may report an error. Surprisingly, in a situation where at least one Razor flip-flop is affected by a residual bit error and at least one Razor flip-flop reports an error, residual and reported word errors are not exclusive events.

The top (respectively middle) graph of Fig. 6.1 compares the residual (respectively reported) word error probability of the two link checkers: a group of Razor flip-flops and a soft SSC. The latter operates satisfactorily both under small and large error rates—as expected from Fig. 5.13—where Razor flip-flops suffer from short-path or undetected timing errors. On the contrary, Razor flip-flops provide correct information to both the end-user and the controller in the operating area where the soft SSC is unreliable.

Now, we introduce the reliability metric used to compare the different checkers. In a system with fixed operating points, reliability is measured by the residual error rate, which is the probability that an undetected error occurs. This metric makes sense because the operating points are not expected to be dynamically adjusted due to detected errors. On the contrary, a self-calibrating system uses a checker not only to avoid the delivery of corrupted outputs—like in traditional systems with fixed operating points—but also to provide a feedback to the controller so that unsafe operating points are dynamically avoided. That is, a checker used in a self-calibrating design should provide reliable information about both *residual* errors (feedback relevant for the end-user) and *reported* errors (feedback relevant for controller). The particularity of Razor-based checkers is that reported and residual errors are not exclusive. Thus, there is no simple relation between these two quantities, contrary to soft SSC where any undetected error is necessarily residual and vice-versa. As a result, a relevant reliability metric for a self-calibrating circuit involving Razor flip-flops is thus the ratio ρ of the residual error probability to the reported (i.e., detected) error probability:

$$\rho = \frac{\varepsilon_{\text{res}}}{\varepsilon_{\text{rep}}}. \tag{6.3}$$

The smaller the metric is, the more reliable the checker. The complementarity of Razor flip-flops and soft SSC is obvious by comparing their respective reliability ratio, which is sketched in the bottom graph of Fig. 6.1. This graph conveys compactly the information expressed in the top and middle graphs of the figure.

In addition, the detection capabilities of Razor flip-flops and soft SSC are also complementary in terms of failure mode of the link. That is, as the voltage is reduced to sub-critical values, soft SSC favour a steep transition from low to extremely large error rates. Such a feature minimises the exposure to moderate error rates where they are the least reliable. On the contrary, a more shallow transition from low to high error rates makes the possible operating range of Razor flip-flops larger. Finally, Razor flip-flops and soft SSC differ in how errors are recovered. The former can correct errors—and, thus recover efficiently—while the latter rely on retransmissions, which incurs a larger latency penalty.

Having emphasised the complementarity of the two checkers, the next section proposes to combine them.

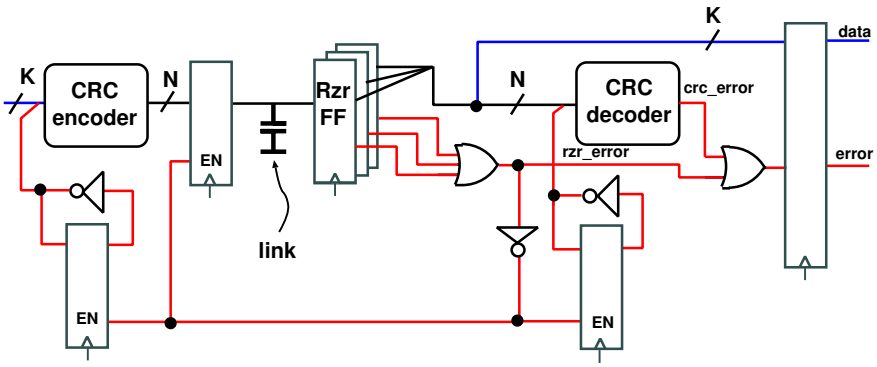

Figure 6.2: The K-bit input data is first encoded into a N-bit codeword. The codeword is transmitted over a link terminated by Razor flip-flops. Finally, the data sampled by the Razor flip-flops is validated in the decoding stage.

6.2 Double Sampling and Codes

The section is organised as follows. First, Sec. 6.2.1 describes a novel checker architecture combining Razor flip-flops with a soft SSC. Sec. 6.2.2 studies its robustness to timing errors and shows its superiority to checkers consisting of only Razor flip-flops or only a soft SSC. However, we argue that, as far as reliability is concerned, the combination of double sampling with the soft SSC is not optimal. Thus, Sec. 6.2.3 proposes a variation on the checker architecture introduced in Sec. 6.2.1 whereby, double sampling used without any error correction capability is combined with a soft SSC. We show that the resulting checker further enhances the robustness to timing errors, while incurring a minimal hardware cost. Finally, Sec. 6.2.4 discusses the hardware complexity of the combined checkers and Sec. 6.2.5 summarises on how to optimally combine double sampling and soft SSC.

6.2.1 Razor Flip-Flops Combined with Codes

Fig. 6.2 depicts a checker combining Razor flip-flops with a soft SSC. The combined checker is organised as a 3-stage pipeline (encoding-transmission-decoding). The basic motivation is to combine serially the two checkers, so that the soft SSC validates the data output by the Razor flip-flops while the latter corrects errors in the range of moderate error rate where the soft SSC reliability is poor.

The binary signal *crc_error* indicates an error in the decoding. The binary signal *rzr_error* indicates the detection of at least one timing error among the N Razor flip-flops sampling the link output. The decoding occurs after the Razor flip-flops, allowing thus to combine error signals from the decoder and the Razor flip-flops very easily: the error signal output along with the decoded data is the OR of *crc_error* and *rzr_error*.

As Fig. 6.2 shows, the phase of the decoder is kept unchanged whenever a timing error is detected because the data corrected by Razor flip-flops is output one cycle after the error is reported. Freezing the phase of the decoder during a cycle has two consequences on the encoder. First, its phase should be frozen as

well, since the two phases must stay synchronised. In turn, as the encoder phase is kept unchanged, the encoder output must also be held during a cycle. If not, two consecutive data pieces would be encoded with the same phase. We point out that the synchronisation of the encoder and decoder phase as performed in Fig. 6.2 poses a significant limitation on the possible values of the parameter T_{d}. Indeed, the *enable* signal feeding the encoder and encoder phase flip-flops (line drawn in the bottom of Fig. 6.2) needs to be de-asserted in the same cycle in which the *rzr_error* signal is asserted. This translates into the following timing constraint (neglecting the set-up and hold time of flip-flops):

$$T_{\mathrm{d}} + t_{\mathrm{p}} \leq T_{\mathrm{c}}, \tag{6.4}$$

where t_{p} designates the time for the *enable* signal feeding the encoding logic to propagate backward from the decoder side. The latter equation necessarily constrains T_{d} to relatively small values. Later in Sec. 6.2.3, we introduce a checker that combines double sampling flip-flops with soft SSC without correcting timing errors. Among other, one advantage of such a checker is that T_{d} is not restricted to small values due to a relation such as Eq. (6.4).

The top diagram of Fig. 6.3 illustrates how a timing error is detected, corrected by the shadow latch, and finally validated by the soft SSC. The timing error occurs in cycle 1, caused by the late arrival of $D2$ at the Razor flip-flops input. In the same cycle, $D1$ is validated by the decoding logic and is output without error in cycle 2. In cycle 2, the *rzr_error* signal is asserted, because the shadow latch samples $D2$ that has finally arrived. The decoding logic also asserts its error signal because it sees now $D1$ with the wrong phase. In cycle 3, $D2$ is output by the Razor flip-flop to the decoding logic, which returns the *rzr_error* signal to low. Moreover, the phase has been kept unchanged to allow for the additional cycle needed for $D2$ to arrive. As a result, the decoding logic validates $D2$ that is output without error in cycle 4.

The bottom diagram of Fig. 6.3 shows how the combined checker behaves in the presence of a short-path error. The error occurs in cycle 1. The shadow latch is corrupted by $D2$ that arrives too early, which triggers the assertion of the Razor error signal. In turn, the assertion of the *rzr_error* signal causes the encoder output and the phase to keep their value in cycle 2. Moreover, in the same cycle, the shadow latch provides the value $D2$ to the decoding logic, causing the *rzr_error* signal to return to 0. Because the phase has been kept to 1, it does not match with $D2$ and thus the CRC error signal is asserted. Next, in cycle 3, $D2$ is still present at the decoder input. The phase has changed to 0. Therefore, the decoding logic validates $D2$ that is finally output in cycle 4.

Short-path errors are intrinsic to double sampling. Henceforth, the combined checker cannot eliminate them. However, as shown in Fig. 6.3, detected and corrected timing errors do not cause the same behavior as short-path errors: the error signals from Razor and CRC errors are not asserted in the same order. Due to this fact, short-path errors can still be diagnosed (i.e., distinguished from timing errors).

Besides the detection of timing and short-path errors, in-order data delivery is also required. As such, the combined checker does not ensure in sequence data delivery since (i) when a timing error is corrected, the encoder output is frozen for a cycle, causing thus the data at its input to be disregarded, and (ii) an error detected only by the decoder triggers a retransmission (which is

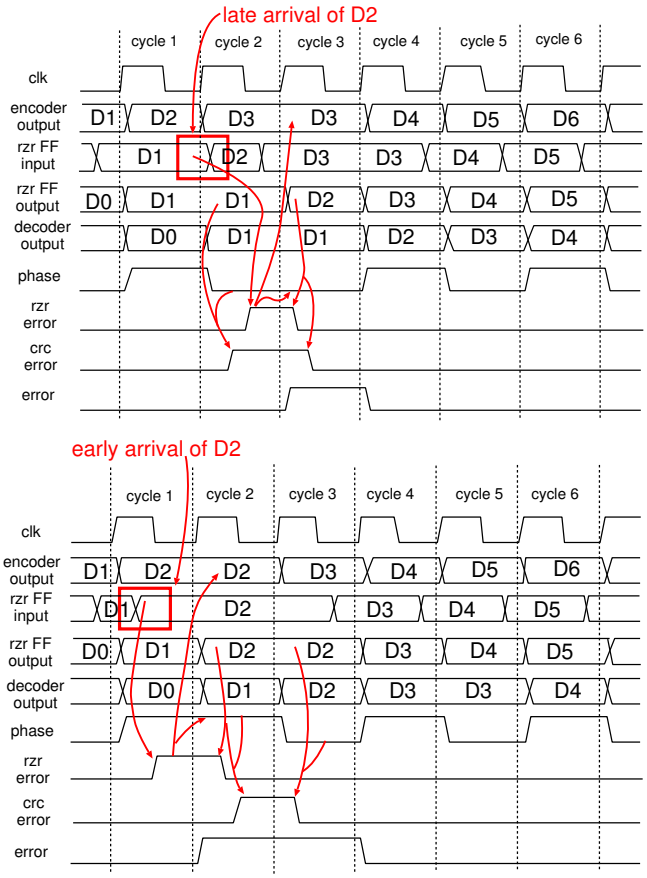

Figure 6.3: Timing diagram of a timing error (top) and of a short-path error (bottom).

equivalent to flushing the pipeline). The task of delivering only correct and in-sequence data pieces is performed by an ARQ controller, such as the one presented in Chapter 4.

6.2.2 Robustness to Timing Errors

We have contrasted qualitatively the detection capabilities of Razor flip-flops and soft SSC in Sec. 6.1. We compare now thoroughly the timing error detection capabilities of the combined checker with, individually, Razor flip-flops or a soft SSC. First, we explain the simulation set-up. Next, we give the comparison results.

6.2.2.1 Experimental Set-Up

We have simulated in VHDL a system consisting of a FIFO, an encoder, a bus terminated by Razor or double sampling flip-flops, a decoder, and an ARQ controller. Each simulation has been run until 500,000 eight-bit data pieces randomly generated were transferred in sequence. For large word error rate, the actual number of word transfers is actually much larger due to frequent retransmissions (up to 10^7 transfers).

The simulation set-up is as follows. We apply the capacitive delay model to express the delay t_p through a bit line of the link as:

$$t_\mathrm{p} = \frac{C_\mathrm{L}}{k_\mathrm{m}} \cdot \frac{v_\mathrm{ch}}{\left(v_\mathrm{ch} - v_\mathrm{th}\right)^2}, \tag{6.5}$$

with k_m the transistor transconductance, C_L the line capacitance, and v_th the device threshold voltage. We refrain from using more complex delay models for two reasons: first, the lumped capacitance model is widely used in the design tools, e.g., by those of Cadence. Second, we study the reliability of the checkers for any error rate from 0 to 100%. In that sense, the presented results are independent of the actual relation between the link supply voltage and the bit error rate. Like in Chapter 3, the ratio $C_\mathrm{L}/k_\mathrm{m}$ is denoted by α and is modelled by a Gamma random variable to account for the variability in the line delay. The coefficient $v_\mathrm{ch}/\left(v_\mathrm{ch} - v_\mathrm{th}\right)^2$ in the right hand side of Eq. (6.5) is considered as deterministic. The bit line delay is thus a random variable parametrised by the supply voltage v_ch.

The bus is modelled with non-synthetisable VHDL and introduces random timing errors. That is, during simulated word transfers, we generate for each bit line a random value for the delay t_p through the line. Henceforth, we determine for each line whether a timing error occurs (i.e., when $t_\mathrm{p} \geq T_\mathrm{c}$) and, if that is the case, whether the error can be detected and corrected by the Razor flip-flop (i.e., when $t_\mathrm{p} \leq T_\mathrm{c} + T_\mathrm{d}$).

The delays t_p through different bit lines are modelled as independent and identically distributed. As the voltage is lowered, the bit error rate rises identically on all lines but each line remains statistically independent from the others.

Fig. 6.4 describes how we model a bus terminated with Razor flip-flops. The model is only used to generate timing errors and mimic the capability of Razor flip-flops to correct them. A Razor flip-flop affected by an undetected timing error outputs the previously sampled data, even during a correction cycle triggered by (at least) one other Razor flip-flop that has detected an error. The

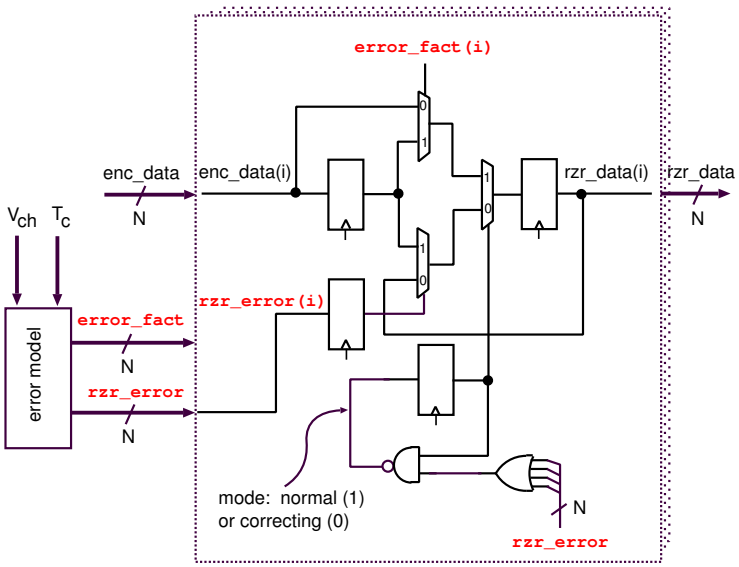

Figure 6.4: Model of a N-bit bus terminated by Razor flip-flops.

i^{th} bit of the signal *error_fact* indicates the presence of a timing error on the i^{th} input bit (i.e., *enc_data(i)*). This timing error is detected by the i^{th} Razor flip-flop if and only if the i^{th} bit of the signal *rzr_error(i)* is asserted. The two N-bit error signals *error_fact* and *rzr_error* are generated randomly at each cycle.

During the simulations, we record the detected and residual word errors for each checker. For the combined checker, errors are reported to the controller if either (i) *rzr_error*= 1 and *crc_error*= 0, i.e., the decoder has validated a word after correction by Razor flip-flops, or (ii) *rzr_error*= 1 and *crc_error*= 1, i.e., the decoder has invalidated a word corrected by Razor flip-flops, and (iii) *rzr_error*= 0 and *crc_error*= 1, i.e., the decoder has detected a word error that no Razor flip-flop detected. Furthermore, a residual word error happens in either of the two following cases. Either all Razor flip-flops are affected by an undetected timing error and the decoder does not detect the resulting word error. Or, at least one Razor flip-flop does not detect a timing error and at least another one reports an error and the resulting word error is not detected by the decoder. All the possible situations are depicted in Fig. 6.5.

We model a link manufactured in a 130nm CMOS technology with the following parameters: the nominal voltage v_{dd} is 1.2V, the threshold voltage v_{th} is 0.2V, and the cycle time T_{c} amounts to 2ns. In addition, $T_{\text{d}} = 0.3\,T_{\text{c}}$, as it has been reported for busses [Kaul *et al.*, 2005]. The value of the parameters a and b characterising the Gamma random variable are chosen so that, under the nominal voltage v_{dd}, the typical delay $\mu_{t_{\text{p}}}$ is 1ns and the standard deviation $\sigma_{t_{\text{p}}}$ is 0.1ns. We ignore the actual failure mode of the link, i.e., the relation of the bit error rate as a function of voltage and frequency. Indeed, it depends on many complex factors—which is actually a reason why worst-case design is so conservative. As a result, we have simulated two opposite noise conditions by assigning different values to the parameter $\sigma_{t_{\text{p}}}$. The first scenario ($\sigma_{t_{\text{p}}} = 0.1$ns under the nominal conditions) models a steep transition from a low error rate

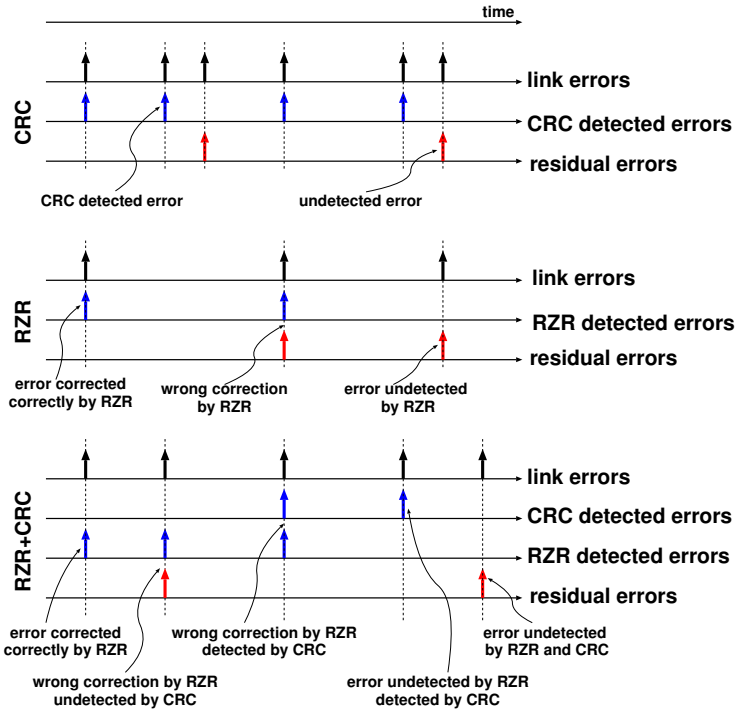

Figure 6.5: All possible error outcomes for each of the 3 checkers (soft SSC, Razor flip-flops, and combined).

to a high error rate. This scenario is called *normal variance*. On the contrary, the second scenario models a more shallow transition from low to high error rates. It is representative of extreme variations in silicon characteristics and is characterised by $\sigma_{t_p} = 0.25$ns at the nominal operating conditions. We refer to this scenario as *large variance*. For illustration, while the probability of a timing error is about 10^{-15} with the normal variance, it amounts nearly to 0.07% under the large variance scenario.

During each simulation, we point out that both voltage and frequency are fixed. However, the voltage is varied from 0.6V to 1.3V. One simulation is performed for each voltage value.

In the simulation, we compute the reliability metric ρ defined in Eq. (6.3) by dividing the total number of residual word errors by the total number of detected word errors. The computed ratio also represents the average number of undetected errors before a detected error occurs.

6.2.2.2 Comparison Results

We compare the following checkers: *RZR*, i.e., 8 Razor flip-flops, *CRC3*, i.e., the 3-bit CRC generated by the polynome $x^3 + x + 1$, *RZR+CRC1*, i.e., a minimal combined checker augmenting RZR with an alternating-phase parity check, and *RZR+CRC3* which combines RZR and CRC3.

Fig. 6.6 plots the reliability metric ρ of these four checkers and the word error rate as a function of the timing bit error rate, under both the normal (top) and large (bottom) variance scenarios.

As expected, Fig. 6.6 shows that the combined checkers always outperform RZR. Moreover, the addition of a single parity bit RZR+CRC1 results in a significant increase in reliability with respect to the RZR checker.

When comparing the RZR+CRC with CRC3 checkers in Fig. 6.6, we observe that the region of poor reliability has shrunk and has been shifted to larger bit error rate values. It follows that the RZR+CRC checkers offer a larger range of bit error rate available to inform reliably the operating point controller that the link is not operative (note that, when the RZR+CRC checker becomes unreliable, the word error rate has already reached 100%). Again, the minimal combined checker (RZR+CRC1) increases reliability very significantly: under the normal variance scenario, residual errors for this checker could not be measured until the word error rate reaches 100%.

While the combined checkers significantly outperform the CRC3 checker for all bit error rates less than approximately 0.7, the single soft SSC checker remains the most reliable beyond these values. Indeed, its residual error rate tends rapidly to zero as the bit error rate further increases because the phase of most (or all of) the bits input to the decoder does not match the phase of the latter. As a result, most of the the errors—or, even all of them—are detected. As Fig. 6.6 indicates, this beneficial phenomenon is not as pronounced for the combined checkers. Specifically, the reliability curve of the combined checkers features a plateau; yet, compared to the one of the CRC3 checker, the plateau is shifted to the right (i.e., towards larger bit error rates). It follows that the single soft SSC checker CRC3 remains the most reliable under very large bit error rates. Fig. 6.7 provides an explanation. Without Razor flip-flops, the bit error rate at the input of the decoder is obviously the same as the link bit error rate $\varepsilon_t = P(t_p \geq T_c)$. On the contrary, the shadow latch of a Razor flip-

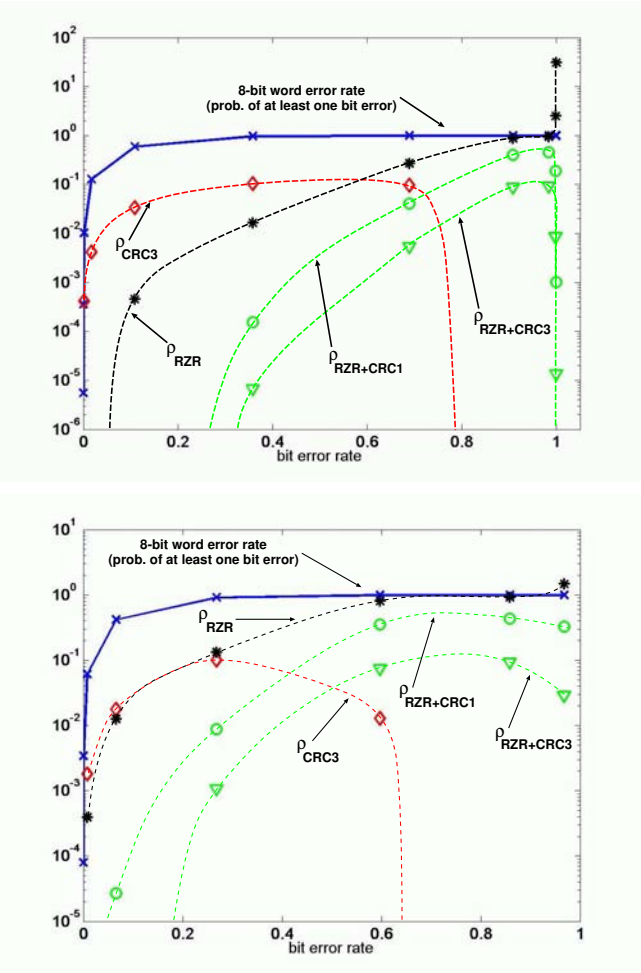

Figure 6.6: The ratio $\rho = \varepsilon_{\text{res}}/\varepsilon_{\text{rep}}$ as a function of bit error rate under normal variance (top) and large variance (bottom). The curves are extended with vertical dotted lines at the first and last simulated points where residual errors could be measured. The maximum unreliability of RZR+CRC1 is comparable to CRC3; however, it occurs at significantly larger bit error rates—and thus smaller voltages.

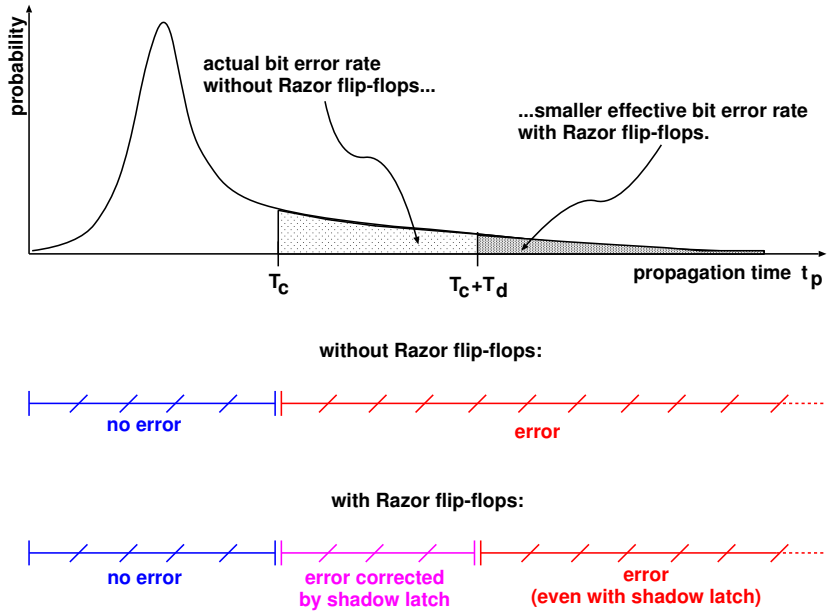

Figure 6.7: Bit error rate as seen by the soft SSC with and without Razor flip-flops. Without error Razor flips-flops, the bit error at the decoder input is the bit error rate of link. With Razor flip-flops, the bit error rate at the decoder input is less than the bit error rate of the link, since some errors are corrected.

flop corrects some errors. Therefore, the bit error rate effectively seen by the decoder of the combined checker is $\varepsilon_t = P(t_p \geq T_c + T_d)$, which is less than the former quantity. As a result, when comparing a single soft SSC checker with a combined checker under the same operating points, *the effect of Razor flip-flops is to reduce the bit error rate measured at the input of the decoder of the combined checker*. If, in addition, the same soft SSC is used for both checkers, then the residual error rate of each checker is obtained by sampling the *same* curve (residual error rate as a function of bit error rate, such as obtained in Sec. 5.5.2) under different bit error rates. However, the detected error rate of combined checker exceeds in any case the one of the single soft SSC checker.

We proceed now by analysing whether operating the decoder of the combined checker under a bit error rate lower than the one of the link is beneficial. To do so, we refer to Fig. 5.13 that plots the residual word error rate of a soft SSC as a function of the bit error rate, assuming a timing error channel. We point out that the residual word error rate is not a monotonic function of the timing bit error rate: as the latter increases, the residual word error rate first increases, then reaches a plateau, and eventually decreases to zero. Because of the non-monotonicity of the function, it is possible that a decrease in bit error rate (due to the correction capabilities of Razor flip-flops) causes an increased residual word error rate (since the residual word error rate is not a monotonic function of the bit error rate). In such a case, correction by Razor flip-flops is detrimental to reliability. Because the values of the word error rate for which the residual error rate reaches the plateau are fairly high (as confirmed in Figures 5.13 and 6.6), we can draw the informal conclusion that correction capabilities of Razor

flip-flops are only beneficial under small word error rates. The next section develops this assertion in more depth.

6.2.3 Giving Up Correction?

The combined checker described so far features aggressive error correction capabilities since the data held by the shadow latch is always considered as a reference even in situations of large error rate where that may not be the case. While error correction capabilities are efficient and safe under small error rates (i.e., when only a few timing errors corrupt a word), they are defective under large error rates because (i) the data pieces held by the shadow latches of some Razor flip-flops—and thus used for correction—may be corrupted themselves, and (ii) the decoder of the combined checker is subject to a smaller bit error rate than the link, which, in this situation, decreases the chance of detecting an error.

These remarks reveal a trade-off between reliability and the correction capabilities of the combined checker. In order to further improve reliability, one could design a different combined checker combining double sampling flip-flops used in the sole purpose of error *detection* with a soft SSC. That is, compared to the RZR+CRC checker, the shadow latch is replaced by a shadow flip-flop whose output is not used to correct timing errors—even though it samples some time after the main flip-flop. The detection of a word error—either by one of the double sampling flip-flops and/or by the decoder of the soft SSC—triggers the retransmission of the entire word. Such a design option favours reliability at the expense of error correction capabilities.

The high level structure of such a checker is depicted in Fig. 6.8. The error signal is asserted whenever a timing error is detected either by the double sampling flip-flops or by the decoder. The assertion of the *error* signal triggers a retransmission. The ARQ controller is a very basic entity that de-asserts the *in_sequence* signal during 2 cycles after the detection of a timing error. Contrary to Fig. 6.2, the main difference stems from the fact that the two checkers are not combined serially, but in parallel: indeed, the double sampling flip-flops and the decoder of the soft SSC are fed with the same data pieces (but at different time instants). Another difference is that there is no need to synchronise the encoder and decoder phase, since the double sampling flip-flops do not provide a corrected data one cycle after the timing error is detected.

Fig. 6.9 illustrates the operation of this checker in the presence of a timing error. In cycle 2, the main flip-flop output is corrupted because it still holds $D1$. On the contrary, the shadow flip-flop has sampled $D2$ correctly. As result, the *ff_error* is sampled high. At the same time, the *crc_error* signal is asserted since $D1$ does not match the phase of the decoding logic. The *error* signal is asserted accordingly in cycle 3. However, contrary to the checker depicted in Fig. 6.2, a *whole* cycle is available for requesting a retransmission. That is, T_{d} is not constrained by the relation such as the one expressed in Eq. (6.4) and may thus assume larger values. Finally, the ARQ controller de-asserts the *in_sequence* signal during cycles 4 and 5 until $D2$ is correctly retransmitted.

We denote by *DSFF+CRCn* a checker such as described in Fig. 6.8, i.e., combining double sampling flip-flops (without error correction) with a n-bit alternating-phase CRC. It can be shown easily that, if submitted to the same data inputs, the DSFF+CRCn checker has a reliability ratio (defined

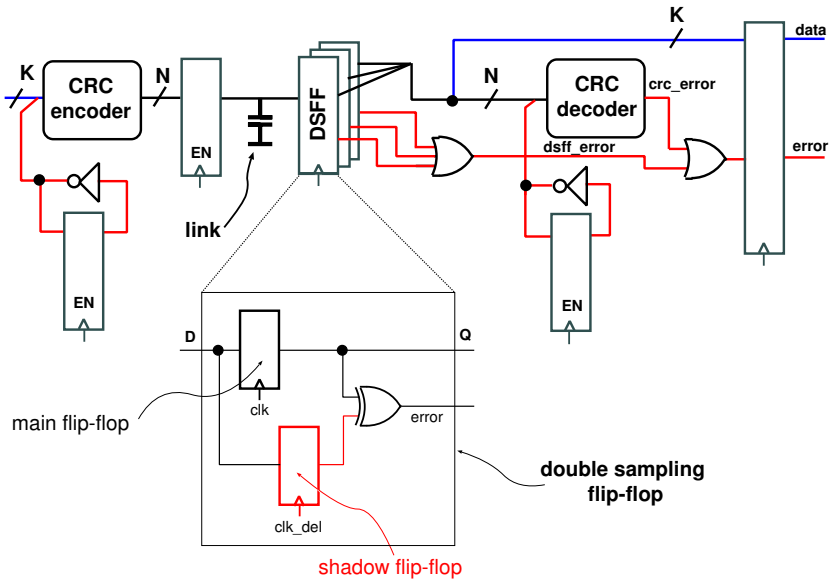

Figure 6.8: A data link using a checker combining double sampling flip-flops with a soft SSC. Timing errors are not corrected. Error recovery (by retransmission) and in-sequence data delivery is ensured by an ARQ controller such as the one described in Sec. 6.2.1.

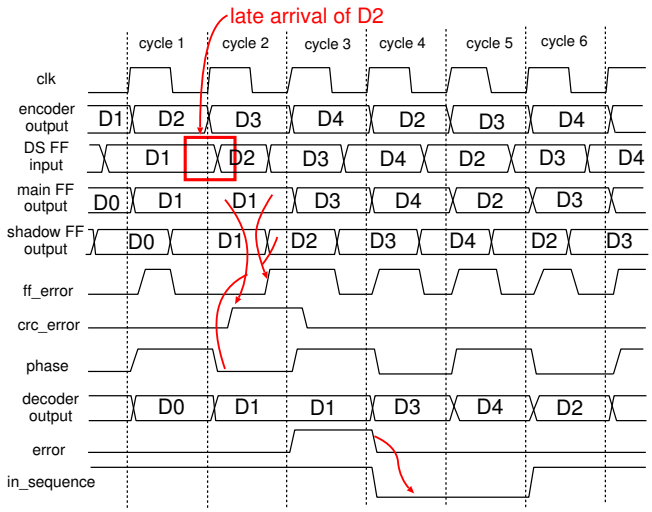

Figure 6.9: Timing diagram describing the operation of the checker depicted in Fig. 6.8 in the presence of a timing error. Note that, contrary to the top diagram of Fig. 6.3, the phase of the encoder and decoder is not affected by the timing error.

in Eq. (6.3)) not larger than the one of the CRCn or RZR+CRCn checkers. To show this claim, we compare first the DSFF+CRCn and CRCn checkers. Because the decoder in both checkers is fed with exactly the same data, it follows that (i) any error detected by the CRCn checker is also detected by the DSFF+CRCn checker, and (ii) any error undetected by the DSFF+CRCn checker is also undetected by the CRCn checker. Therefore, the reliability ratio of the DSFF+CRCn checker is less than or equal to the one of the CRCn checker. We compare now the DSFF+CRCn and RZR+CRCn checkers. Clearly, any error detected by the Razor flip-flops is also detected by the double sampling flip-flops. Moreover, any error that is undetected by the Razor flip-flops but detected by the CRC is, as well, undetected by the double sampling flip-flops and detected by the CRC. Thus, the number of detected errors of the DSFF+CRCn checker is at least the number of detected errors of the RZR+CRCn checker. On the other hand, the RZR+CRCn generates more residual errors than the DSFF+CRCn checker. Indeed, a residual error with the former either went undetected by both the Razor flip-flops and the CRC, or has been detected and wrongly corrected by the Razor flip-flops and then validated by the CRC. The first case also causes a residual error with the DSFF+CRCn checker. On the contrary, the second case causes a retransmission since the error would be detected by the double sampling flip-flops. As a result, the reliability ratio of the DSFF+CRCn checker is less than the one of the CRCn checker.

We have simulated under the same conditions as in Sec. 6.2.2 the reliability ratio of the DSFF+CRC1 checker, i.e., a checker consisting of double sampling flip-flops combined with a parity alternating-phase code. Fig. 6.10 plots the reliability ratio of the DSFF+CRC1 checker and compares it with the ratios plotted in Fig. 6.6. As expected, the DSFF+CRC1 checker outperforms all other checkers under any error rate

Overall, giving up to correction capabilities offers significant benefits.

- **Higher reliability.** As discussed and verified in Fig. 6.10, double sampling flips-flops combined with a single alternating-phase parity bit stands out as the most reliable checker. Furthermore, there is another phenomenon that increases reliability under small error rates. Because the shadow flip-flop output is not used as a reference to correct errors, the effect of short-path errors on reliability is limited to false alarms. On the contrary, the combined checker using Razor flip-flops may output corrupted data in case of a short-path error and compromise thus reliability.
- **Tolerance to soft errors.** Several techniques mitigate the effects of soft errors thanks to double sampling flip-flops. They exploit the fact that soft errors have a limited effect in time by introducing a delay either between the clock or data signals fed into the flip-flops. While Razor flip-flops are unable to recover from soft errors—since the output of the shadow latch is not guaranteed to be correct—double sampling flip-flops can. Indeed, they only indicate a mismatch between the data held in each of them. Retransmitting the corrupted data recovers from the soft error.
- **Low hardware overhead.** The high complementarity of the timing redundancy added by the DSFF+CRC checker enables to minimise the codec circuitry and the wiring overhead, while not compromising reliability. Moreover, the area and power overhead incurred by the shadow flip-flops can be amortised as follows. In a recent work devoted to soft

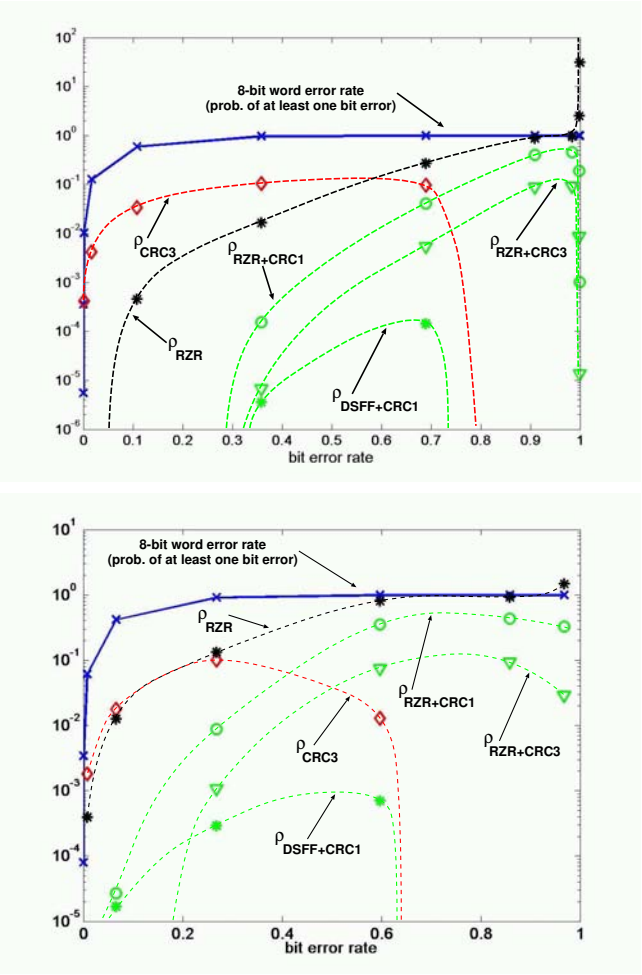

Figure 6.10: The ratio $\rho = \varepsilon_{\text{res}}/\varepsilon_{\text{rep}}$ as a function of bit error rate under normal variance (top) and large variance (bottom). In both scenarios, the DSFF+CRC1 checker outperforms all other checkers, except the RZR+CRC3 checker in a limited range of error rate.

error recovery with double-sampling flip-flops, Mitra *et al.* propose to use scan flip-flops—i.e., additional flip-flops used for testing and debugging purposes and thus unused during normal operation of the chip—as shadow flip-flops [2005]. This idea may also be applied to the double sampling flip-flops in the combined checker proposed in this section.

Hybrid approaches where correction capabilities are dynamically disabled can also be envisaged. We mean that correction is enabled when the data input to the decoder is taken from the shadow flip-flop. Conversely, it is disabled when the decoder is fed with data sampled by the main flip-flop. One possibility consists in disabling the correction capabilities whenever the number of timing errors exceeds a given threshold (possibly a single timing error). While error recovery of such a checker would be more efficient than only retransmitting, its hardware overhead would be increased by the additional circuit indicating when a single or a few bits of a register are set to 1. Another possibility would be to disable correction capabilities only when the supply voltage of the self-calibrating link is lowered to a tentative new level. Otherwise, during normal operation—in fact, when the error rate is expected to remain low—correction is enabled and safely exploited. Dynamically enabling error correction opens a new research direction as far as recovery of on-chip transfer errors is concerned and compares interestingly with hybrid ARQ schemes recently proposed [Murali *et al.*, 2005].

6.2.4 Hardware Complexity of the Combined Checkers

The combined checkers—RZR+CRC and DSFF+CRC—are both organised as a 3-stage pipeline (encoding-transmission-decoding). This fact in itself does not constitute an additional hardware overhead because coding techniques requiring a similar organisation are already deployed over on-chip links, as mentioned in Sec. 4.4.

We argue that the hardware overhead incurred by the DSFF+CRC checker is minimal. First, the area overhead caused by double sampling flip-flops can be mitigated by using scan flip-flops in the purpose of double sampling [Mitra *et al.*, 2005]. Second, we have shown in Sec. 6.2.3 that a single alternating-parity bit protecting 8 information bits offers a high robustness to timing errors over the whole range of bit error rates. Such a checker only adds a single redundant wire and a minimal codec circuitry needed to compute a parity. Moreover, the amount of redundancy (1/8 or 12.5%) is smaller than or comparable to the one added by error control techniques traditionally deployed over on-chip bus—e.g., [McNairy and Soltis, 2003]. The codec circuitry is also less complex as the DSFF+CRCR checker does not perform error correction. We also point out that the hardware overhead of the RZR+CRC or single soft SSC checker is larger than the one incurred by DSFF+CRC, because more redundant bits are required for the same reliability level.

Finally, we have synthetised an ARQ controller for the RZR+CRC checker—in fact, the most complex configuration—and found that it consists of only 6 flip-flops and about 20 gates. Actually, a very similar controller is required in any system that uses codes for error detection.

6.2.5 Double Sampling and/or Codes: Conclusions

Overall, we recommend the checkers combining Razor or double sampling flip-flops with an alternating-phase parity check because they offer a sufficient level of reliability, while incurring a minimal overhead in wiring and codec circuitry. This high level of robustness is explained by the addition of both intra-cycle (i.e., double sampling) and extra-cycle (with the phase bit) time redundancy.

As we have shown in Fig. 6.6, the RZR+CRC1 checker is more reliable than Razor flip-flops under all error rates. Moreover, compared to the single soft SSC checker, the RZR+CRC1 checker offers a larger range of bit error rate to *reliably* report transfer errors to the operating point controller. The RZR+CRC checker also lessens the impact of error recovery on latency since retransmissions are triggered only in situations that would have resulted in an undetected error had Razor flip-flops been used as a single checker.

Then, we have investigated the possibility of using double sampling flip-flops only in the purpose of error detection—error recovery being performed by retransmission. This option is very promising because the resulting checker (i) is more robust to timing errors than any other checker architecture, (ii) is resistant to soft errors, (iii) has a low hardware cost, and (iv) contrary to RZR+CRC checkers, does not necessarily require a small delay between the clock feeding the main and shadow flip-flops (as the constraint of Eq. (6.4) expresses).

A remarkable fact is that the soft SSC validating the data output by the Razor or double-sampling flip-flops needs not include complex spatial redundancy that incurs a large wiring overhead. The reliability ratio plotted in Fig. 6.6 for the RZR+CRC1 checker confirms that the addition of a single alternating-parity bit increases reliability significantly compared to the RZR checker. The RZR+CRC3 checker is certainly even more reliable; however the increase in reliability is not as significant compared to the RZR+CRC1 checker. The reason is that the reliability of the combined checker relies on the soft SSC only from the point on where some Razor flip-flops do not detect timing errors any more, which happens with large bit error rates. Under such conditions, detecting timing errors does not require complex spatial redundancy, but relies mainly on the notion of phase. This phenomenon is even more acute for the DSFF+CRC1 checker.

We proceed as follows. In Sec. 6.3, we answer the question whether comparing checkers requires some knowledge about the operating point control policy. Finally, Sec. 6.4 expands on the opportunity of relaxing coding requirements when codes are used in combination with Razor flip-flops, and opens perspectives towards self-calibrating computation.

6.3 Comparing Checkers With Operating Point Usage

When comparing the reliability metric ρ of two checkers, it could be that one of them is more reliable on a given operating point, while the converse occurs on another operating point. Actually, the reliability comparison of the checkers CRC3 and RZR+CRC1 in Fig. 6.6 illustrates this case. Nevertheless, we could recommend the checker RZR+CRC1 because any practical control policy settles mainly on operating points where the combined checker is more reliable than the CRC3 checker. Therefore, a decision can be made without assuming a particular

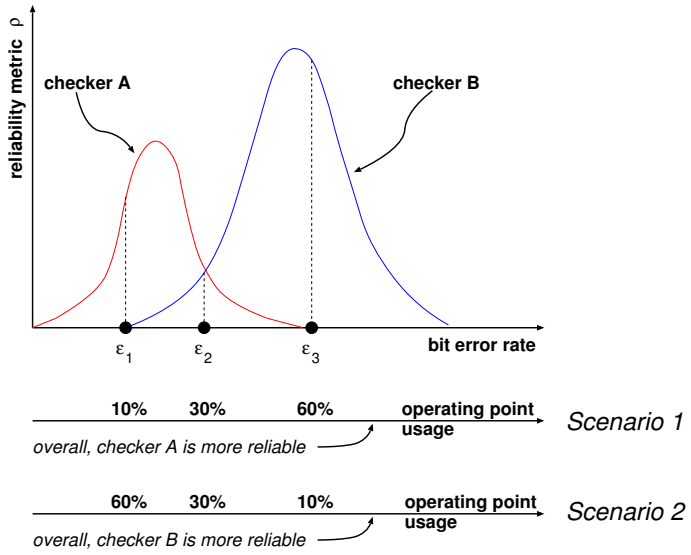

Figure 6.11: Hypothetic comparison of two checkers. With the operating point distribution of scenario 1, checker A is more reliable than checker B because the circuit is mostly used under error rate ε_3. On the contrary, the distribution of scenario 2 uses mainly the error rate ε_1 where checker B is more reliable than checker A.

distribution of the operating point usage (i.e., the fraction of time a particular operating point is used).

However, the knowledge of the operating point usage may be required to determine which of two checkers is more reliable. We mean by operating point *usage* the fraction of the time the operating point is selected (used) by the controller. For example, let us consider the situation depicted in Fig. 6.11. In a situation where the third operating point (characterised by a bit rate ε_3) is mostly used, checker A is overall more reliable than checker B. On the contrary, a different control policy using mostly the first operating point (characterised by a bit rate ε_1) favours checker B.

We denote by Q the number of possible operating points and by u_i the usage of the i^{th} operating point. By definition, the operating point usage satisfies $\sum_{i=1}^{Q} u_i = 1$. In order to discriminate the checkers, the value of the reliability metric ρ of a given operating point should be weighted by its usage frequency. The resulting reliability metric, which we call the *effective* reliability metric and denote by ρ_{eff}, is computed as

$$\rho_{\text{eff}} = \sum_{i=1}^{Q} \rho_i \, u_i, \tag{6.6}$$

with ρ_i the reliability metric of the i^{th} operating point as defined in Eq. (6.3). The effective reliability metric can thus be interpreted as the average (over all operating points used) number of residual errors per detected error.

This metric enables to compare the reliability of checkers, even in situations like the one depicted in Fig. 6.11: a checker A is more reliable than a checker

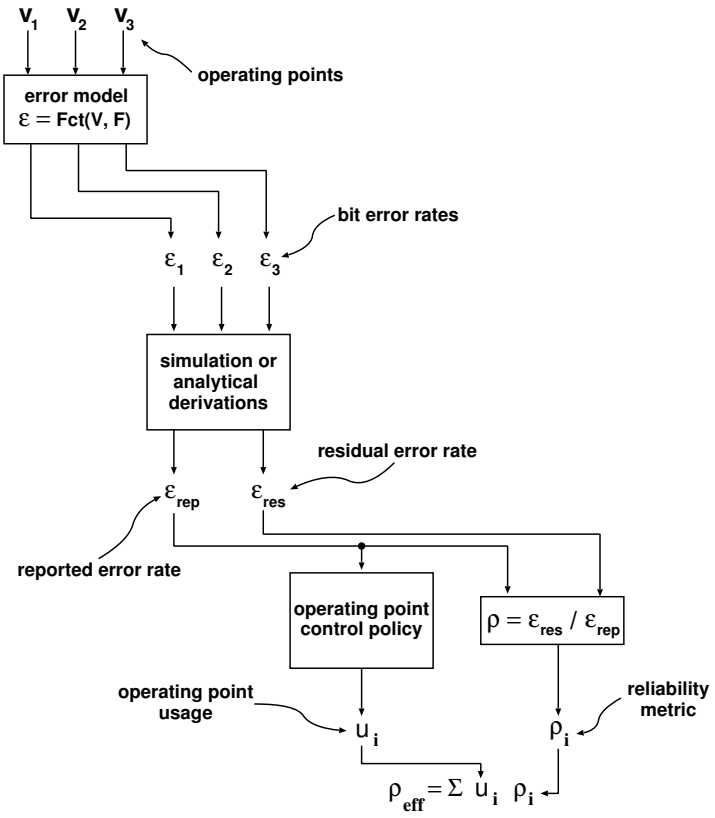

Figure 6.12: Successive steps required for the computation of the effective reliability metric ρ_{eff}.

B if and only if $\rho_{\text{eff,A}} < \rho_{\text{eff,B}}$. Yet, this relation can be deduced without the knowledge of the operating point usage if either $\rho_{i,A} < \rho_{i,B}$ for all $i = 1, ..., Q$—such as when comparing RZR with RZR+CRC1—or if the latter relation is not verified only for operating points u_j such that $u_j \approxeq 0$ for all practical control policies—such as when comparing CRC3 with RZR+CRC1.

Evaluating Eq. (6.6) requires both the quantities ρ_i and u_i. The ratios ρ_i are derived by simulation, as performed in Sec. 6.2.2, while the operating point usage can be obtained by a statistical analysis of the control policy. Since the latter specifies how operating points are adjusted in function of detected errors, a statistical analysis is possible whenever the reported error probability ε_{rep} is known for each operating point. In turn, these probabilities can be obtained from reliability simulations such as the ones performed in Sec. 6.2.2. Fig. 6.12 describes the consecutive steps required to evaluate the effective reliability metric. Even though we did not encounter a practical case justifying the computation of the effective reliability metric, we illustrate in Example 12 the method described in Fig. 6.12.

Example 12. (comparison of the effective reliability of two checkers).

We would like to compare the reliability of the checkers CRC3 and

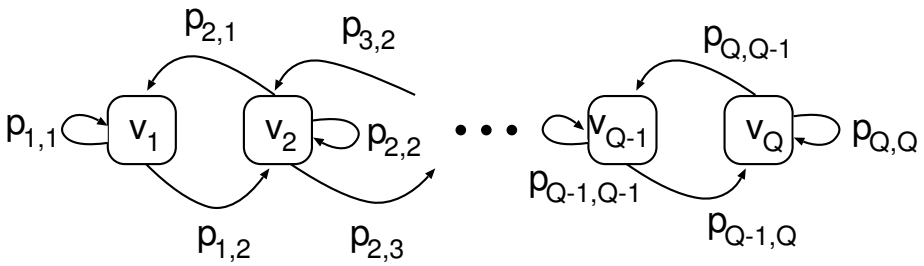

Figure 6.13: The error rate tracking controller modelled as a discrete time Markov chain. The probability of requesting voltage level j given that the current voltage level is i is denoted by $p_{i,j}$.

RZR+CRC1 under the normal variance scenario defined in Sec. 6.2.2 and assuming a voltage control policy that consists in tracking a target word error rate—1% in this example. The voltage controller records the number of reported errors every W cycles. The voltage level is incremented whenever the number of reported errors exceeds a threshold T_h and is decremented whenever the number of reported errors is less than another threshold T_l.

We choose a Markovian control scheme. The control algorithm is a discrete time Markov chain, as shown in Fig. 6.13. The time step of the Markov chain model amounts to W cycles, which corresponds to the time instants where the voltage control decision is taken. The probability $p_{i,j}$ of requesting voltage level j given that the current voltage level is i is computed as follows. At each cycle, an error is reported to the controller with probability ε_{rep}. Therefore, using the fact that errors occurring in different cycles are statistically independent, the probability of increasing voltage is, for $i = 1, ..., Q-1$,

$$p_{i,i+1} = \sum_{k=T_h}^{W} \binom{W}{k} \varepsilon_{rep,i}^{k} \cdot (1 - \varepsilon_{rep,i})^{W-k},$$

while, for $i = 2, ..., Q$, the probability of decreasing the voltage is

$$p_{i,i-1} = \sum_{k=0}^{T_l} \binom{W}{k} \varepsilon_{rep,i}^{k} \cdot (1 - \varepsilon_{rep,i})^{W-k},$$

where $\varepsilon_{rep,i}$ is probability of a reported error in voltage level i. The probability of keeping the voltage unchanged is computed as

$$p_{i,i} = 1 - p_{i,i-1} - p_{i,i+1},$$

because voltage levels can only be incremented, decremented, or kept unchanged. This also implies that $p_{i,j} = 0$ whenever $\mid i - j \mid > 1$.

Let P be the $Q \times Q$ matrix defined by $P = (p_{i,j})_{i,j=1,...,Q}$. The column vector consisting of the steady state usage probabilities U satisfies

$$U = P\,U.$$

The steady state usage probabilities are the unique solution of the latter equation satisfying $\sum_i u_i = 1$.

v_{ch}[V]	0.85	0.90	0.95	1.0	1.05	1.10
ρ	$2.0 \cdot 10^{-4}$	0	0	0	0	0
ε_{rep}	0.48	0.33	$8.0 \cdot 10^{-2}$	$6.3 \cdot 10^{-3}$	$2.3 \cdot 10^{-4}$	$1.6 \cdot 10^{-6}$
u	0	0	0.30	0.70	0	0
ρ_{eff}	$5.0 \cdot 10^{-17}$					

(a)

v_{ch}[V]	0.85	0.90	0.95	1.0	1.05	1.10
ρ	0.10	$3.4 \cdot 10^{-2}$	$4.1 \cdot 10^{-3}$	$4.0 \cdot 10^{-4}$	0	0
ε_{rep}	0.87	0.57	0.10	$7.8 \cdot 10^{-3}$	$3.0 \cdot 10^{-4}$	$1.6 \cdot 10^{-6}$
u	0	0	0.06	0.94	0	0
ρ_{eff}	$6.4 \cdot 10^{-4}$					

(b)

Table 6.1: Voltage (v_{ch}), reliability metric (ρ), reported error rate (ε_{rep}), operating point usage (u), and effective reliability metric (ρ_{eff}) for the RZR+CRC1 (a) and the CRC3 (b) checkers.

Using the reported error probabilities obtained in Sec. 6.2.2, we have computed the operating point usage probabilities assuming a controller characterised as follows. The voltage is updated every 5000 *cycles (i.e.,* $W = 5000$ *and the target count of reported error is* 50*). The voltage is increased whenever more than* 70 *errors are reported (i.e.,* $T_h = 70$*) and decreased whenever less than* 30 *errors are reported (i.e.,* $T_l = 30$*). The link frequency is fixed (*500*MHz), while the simulated voltages range from* $0.6\,V$ *to* $1.1\,V$ *in* $50mV$ *increments. Table 12 summarises the figures obtained, focusing on voltages larger than or equal to* $0.85\,V$*. The numerical comparison of the effective reliability metric clearly confirms the advantage of the RZR+CRC1 checker over the CRC3 checker. The explanation is obvious: the steady state analysis of the control policy reveals that, essentially, the only voltages used are* $0.95\,V$ *and* $1.0\,V$*, where no residual errors could be measured for the combined checker.*

We proceed by presenting perspectives ideas on how to combine Razor flip-flops with codes to build checkers for computing elements, e.g., adders.

6.4 Towards Self-Calibrating Computation

As emphasised in Sec. 6.2.5, the high robustness of the combined checkers results of the combination of double sampling flip-flops with codes—possibly, quite basic—including time redundancy, e.g., the notion of phase. Expanding on this observation, we consider now the combination of double sampling flip-flops with codes on computing elements.

Two main differences stand out compared to communication. The first difference bears on the error model, or more precisely, the lack thereof. Regarding communication, our fault model—i.e., the failure of bit transition—has lead to a particular communication channel: the timing error channel. In our opinion, it is unclear whether developing an equivalent channel for, e.g., an adder is feasible. The failure modes of a computing element are more complex than

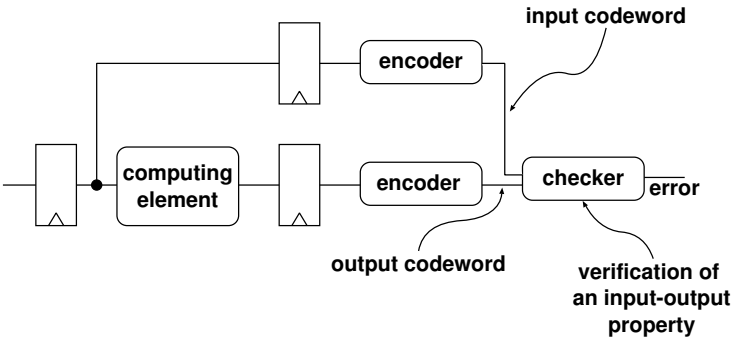

Figure 6.14: An input-output property verified by the computing element under correct operation is used as a checking principle. Without loss of generality, a single input, single output computing element can be considered.

the mere failure of bit transitions. The concept of bit error rate is hard to correlate with a physical event—such as, for communication, the fact that the propagation time through the link exceeds the cycle time. Furthermore, given a particular transition vector, the timing error channel enables to determine all possible channel outputs, which is valuable in order to develop a coding scheme. Again, we see no such equivalent that could list all possible outputs of a computing element operated at sub-critical voltage. Finally, computing elements may be implemented in a wider variety of forms than data links. Thus, different implementations are very likely to behave differently at sub-critical voltage.

The second difference is that, unlike a link, a computing element transforms its input. That is, the set of possible outputs is, in general, different from the set of possible inputs. As a result, the coding scheme deployed must be "preserved" by the computing element. Because communication does not transform intentionally data, a verified input-output property is *identity*. The checking principle used in computed elements is more general, as described in Fig. 6.14. Two requirements bear on the input-output relation under correct operation: (i) it is verified for input codewords, and (ii) it is not verified for inputs that are not codewords. When these requirements are met, the computing element is said to be *code disjoint* [Lala, 2001]. There exist simple input-output properties for the parity and Berger codes under all operations performed by an ALU [Nicolaidis, 2003; Jien-Chung *et al.*, 1992].

We give now specific examples of the checker architecture depicted in Fig. 6.14 by focusing the rest of the discussion on adders. We denote respectively by A, B, C, and S the two input operands, internal carries and sum of an N-bit adder. C contains the carry-in c_{in} and the $N-1$ internal carries. Moreover, we denote respectively by $p(A)$, $p(B)$, $p(C)$, and $p(S)$ the parity of the inputs, internal carries, and sum. The bitwise relation between A, B, C, and S is $S_i = A_i \oplus B_i \oplus C_i$, with $1 \leq i \leq N$ and $c_1 = c_{\text{in}}$. It follows that $p(S) = p(A) \oplus p(B) \oplus p(C)$. By reordering the terms, we obtain the following input-output relation: $p(A) \oplus p(B) = p(C) \oplus p(S)$. That is, one can consider the adder as a single element with a $2N$-bit input, $A \mid B$, producing a $2N$-bit output $C \mid S$, where $\cdot \mid \cdot$ denotes the concatenation operator. From this point

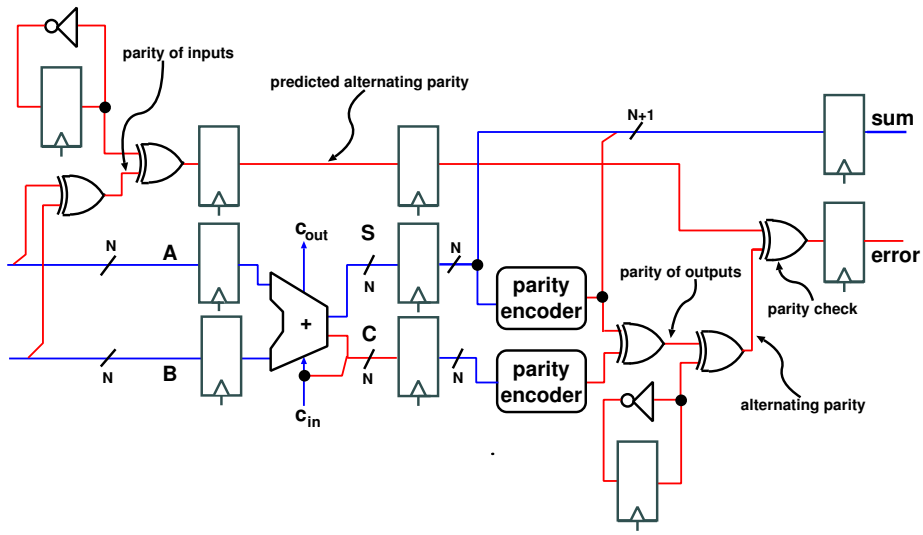

Figure 6.15: An adder with an alternating-parity prediction scheme. A and B are the two parity-encoded operands. S and C are respectively the sum and internal carries.

of view, the adder preserves parity, i.e., $p(A \mid B) = p(C \mid S)$.

Fig. 6.15 is a natural translation of Fig. 4.6 in the context of an adder. The encoding also includes the notion of phase. In Example 2, we have mentioned that LEDR alternates the parity of transmitted codewords. Likewise, the parity predicted in the encoding of Fig. 6.15 alternates at each new addition between the parity $p(A) \oplus p(B)$ and its opposite. While such a scheme prevents from delivering twice the same adder output in case of over-aggressive operation, it is expected to have a poor detection capability under moderate bit error rate, because any even number of bit errors leaves the parity unchanged. Moreover, multiple-bit errors are likely due to the fact that errors affecting the carry propagate. Two non-mutually exclusive options are available to increase the reliability of this scheme. The first one consists in using Razor flip-flops to increase the detection of timing errors, as already discussed in Sec. 6.2.1 for the case of a link. The second option focuses on increasing the code error detection capability and the hardware overhead as well.

On the one hand, the addition of Razor flip-flops is a natural translation of Fig. 6.2 to an adder, as depicted in Fig. 6.16. The resulting checker is expected to be more robust than the one described in Fig. 6.15, since its reliability relies on the single parity check only under error rates large enough to cause some or all Razor flip-flops not to detect errors any more. This is the same reason as already invoked in Sec. 6.2.5 to justify why the coding requirements may be relaxed when combined with Razor flip-flops. Another expectation is that the robustness of the combined checker will hold for different possible implementations of the adder.

On the other hand, we see two ways of improving reliability of the coding scheme.

1. Including more than a single parity check. Parity is preserved by addition

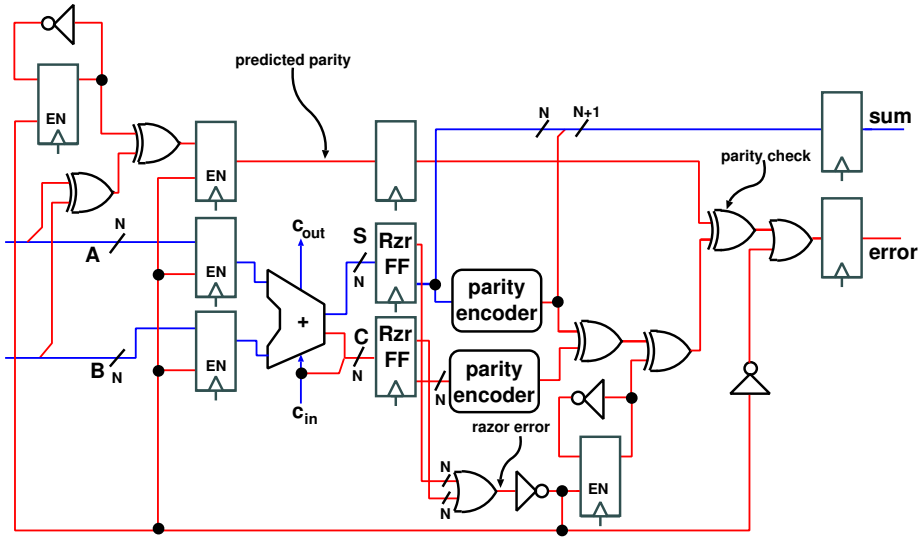

Figure 6.16: An adder combining an alternating-parity prediction scheme with Razor flip-flops. A and B are the two parity-encoded operands. S and C are respectively the sum and internal carries.

as long as it is computed on consecutive bits. Thus, several parity checks can be computed; their supports need not be disjoint.

2. Using other codes preserved by addition, such as the Berger code or low-cost residue codes where the redundant bits are computed as the remainder of the division of the information bits by $2^n - 1$, e.g., 3 or 7.

6.5 Conclusions

The main achievement of this chapter is to propose novel checker architectures for a self-calibrating on-chip link that feature unmatched robustness to timing errors with minimal codec circuitry and wiring overhead. In Sec. 6.1 and Appendix D, we have emphasised the strong complementarity of checkers based on double sampling flip-flops and soft self-synchronising codes (SSC). Next, exploiting this observation, we have proposed two different checker architectures combining double sampling flip-flops with soft SSC.

The first architecture (denoted by RZR+CRC) combines Razor flip-flops with a soft SSC and features

- a larger reliability than each of its counterparts,
- efficient error correction capabilities requiring retransmissions only when Razor flip-flops as a single checker would have caused residual errors, and
- a low hardware overhead incurred by the redundant wires and the codec circuitry, since detection capabilities of the soft SSC are only tried under large bit error rate where the notion of phase—not complex spatial redundancy—mostly matters.

However, we have shown that, under large bit error rates, the error correction capabilities of Razor flip-flops interfere negatively with the soft SSC: in fact, the

single soft SSC checker remains more reliable under large error rates. Although, all in all, the range of error rate ensuring reliable operation is larger for the RZR+CRC checker than for the soft SSC checker, we have developed a modified version of the combined checker where the shadow latch in a Razor flip-flop is replaced by a shadow flip-flop whose output is used in the sole purpose of detecting timing errors. In this latter version, the redundancy brought by double sampling flip-flops is combined optimally—from a reliability standpoint—with the timing redundancy of the soft SSC because the detection capabilities of each checker are exploited without any negative interference, contrary to the the RZR+CRC checker. We have verified by simulation that the checker combining double sampling flip-flops with a single alternating-phase parity bit outperforms all other checkers.

The checker combining double sampling flip-flops with soft SSC achieves the goal formulated in Sec. 1.3, namely the separation of reliability concerns from performance and power objectives. On the one hand, reliability concerns are confined to the checker even without making any worst-case assumptions about the link error rate. On the other hand, the operating point controller adjusts voltage in function of reported errors and operating frequency in function of workload requirements. As a practical consequence, the combined checker enables a larger variety of voltage control policies without risking infringements to reliability. Because of its high robustness to timing errors under any possible bit error rate, the checker does not constrain the operating point controller to avoid weak spots where reliability is not ensured. Equivalently, the novel checker architecture enables a reliability-agnostic control of the link operating points, which is a property neither the single soft SSC checker nor Razor flip-flops enjoy.

Next, we have elaborated on the question of comparing the reliability of checkers for self-calibrating circuits. Since the checker provides information to both the end-user (i.e., the entity using the circuit outputs) and the operating point controller, both residual and reported errors need to be taken into account. There exists not necessarily a simple relation between these two quantities: e.g., residual and reported errors are not exclusive with Razor flip-flops. We have thus proposed a reliability metric specific to checkers for self-calibrating circuits, which accounts for both residual and reported errors . Then, using this metric, we have characterised in Sec. 6.3 situations where the comparison of two checkers does require information about the operating point usage. We have illustrated the procedure required to compute this reliability metric in the particular case of a voltage controller tracking a target error rate.

Finally, expanding on the opportunity to combine low overhead codes with double sampling flip-flops, we have given prospective checker architectures for an adder. Due to the lack of models on the failure of computing elements at sub-critical voltage, future work in this direction should focus on the experimental validation of such checkers, such as the approach followed by Roberts *et al.* [2005].

Chapter 7

Conclusion

First, we summarise our achievements taking especially into account the goal introduced in Chapter 1. In Sec. 7.2, we mention potential areas that could benefit from the checkers we have proposed. Next, Sec. 7.3 mentions a few items that we consider worth investigating in the short-term. Finally, we conclude by discussing long-term perspectives opened by this work.

7.1 Achievements

We say up-front that, through the work presented in Chapter 4, this thesis has pioneered the use of digital self-calibrating techniques to set the operating points of an on-chip link without relying on worst-case characterisation [Worm *et al.*, 2002; 2005]. Having demonstrated the feasibility of a self-calibrating on-chip link, this thesis focuses primarily on the checkers deployed in such circuits.

The main achievement is thus the development of checker architectures meeting the goal stated in Chapter 1, which we recall as follows:

- The checker should demonstrate a high reliability under any possible timing error rate. Although we quantify reliability using a particular error model (namely, the timing error channel), the requirement on the checker to operate reliably over the whole range of bit error rate makes it possible to design without any worst-case characterisation of the link error rate. This requirement is key for the a self-calibrating design that may operate temporarily under error rates as large as 100%—yet and especially under this condition, reliable operation of the checker should be guaranteed.
- The overhead of the checker should be limited in terms of redundant wires added to the link and additional encoding and decoding circuitry.

The checkers proposed in Sec. 6.2 meet these objectives. They offer an unmatched level of robustness to timing errors because they combine two very complementary checkers, namely Razor or double sampling flip-flops adding intra-cycle timing redundancy with CRC-based alternating-phase encoding adding extra-cycle timing redundancy. Due to the strong complementarity of the added timing redundancy, such combined checkers do not require complex and costly spatial redundancy to operate reliably. We have verified by simulation that a

single alternating-phase parity check validating the output of Razor or double sampling flip-flops offers a level of reliability sufficient to protect 8-bit data under any possible timing error rate until 100%. In addition, we have shown that, interestingly, the error correction capabilities of double sampling flip-flops combined with soft self-synchronising codes are detrimental to reliability under large bit error rates. As a result, we have proposed in Sec. 6.2.3 a checker architecture combining double sampling flip-flops—without any correction capability—with codes, which combines optimally the robustness of each individual checker.

As stated in the introduction, such low-overhead and highly robust combined checkers constitute key building blocks in that they enable a reliability-agnostic control of the on-chip link. As a result, the operating point controller is not constrained and complexified by the need of avoiding weak spots of the checker. Simple control policies can be designed to efficiently exploit power-performance trade-offs.

A second achievement of this thesis is to study comprehensively the self-synchronisation properties of synchronous encoding schemes. Based on the observation that operation at sub-critical voltage causes data to be sampled while some individual bit transitions have not yet completed, we have defined timing errors and their associated communication channel. We have exploited the formal definition of timing errors to characterise *hard self-synchronising* encodings that enjoy the property of detecting all possible timing errors. We have bounded the minimum amount of wiring overhead required to ensure the hard self-synchronising property. The bound is achieved by differential encoding with unordered codes of minimum redundancy, i.e., using codes known to be optimal in asynchronous communication. Next, we have studied self-synchronisation for a particular type of encoding (referred to as *symbol-invariant*) that alternates different dictionaries in time. Focusing on symbol-invariant encodings that alternates only two dictionaries, we have conjectured—and verified by an exhaustive search for several symbol sizes—the optimality of LEDR (defined in Example 2). That is, we claim that no other hard self-synchronising alternating-phase encoding uses bandwidth more efficiently than LEDR. Because LEDR does not consist of unordered codes of minimum redundancy, the conjecture shows that, depending on the considered symbol sequencing rule, maximising the bandwidth efficiency of hard self-synchronising encodings does not necessarily lead to unordered codes of minimum redundancy. Therefore, one important contribution is to specify and limit the scope of optimality of the latter encoding and give new properties characterising hard self-synchronisation—as well as the new optimal encodings—valid out of its scope of optimality.

Moreover, we have investigated the possibility of reducing the wiring overhead beyond the limit needed to ensure hard self-synchronisation. Although the resulting encoding schemes do not detect any more all possible timing errors, they still detect many of them and have a lower wiring overhead. Accordingly, we have called these schemes *soft self-synchronising*. In the particular case of LEDR, reducing the wiring overhead below its original amount (i.e., 100%) has lead to a novel encoding scheme: CRC-based alternating-phase encoding. This novel soft self-synchronising encoding features unique detection capabilities towards both timing and additive errors, as shown in Fig. 5.16. Regarding the reduction in the wiring overhead of LEDR, we have bounded and approximated accurately the induced loss in reliability, quantifying thus the trade-off between wiring overhead and self-synchronisation properties.

Finally, we argue that timing errors are as relevant as additive errors to derive reliability figures of standard error correcting codes used over on-chip links. Indeed, crosstalk—a phenomenon raising significant concerns about the reliability of deep sub-micron on-chip links—causes both timing and additive errors. Timing errors result from delayed bit transitions, while additive errors are caused by spurious bit transitions (modelled with a binary symmetric channel). Therefore, Eqs. (5.10), (5.11), and (5.12) bounding and approximating the residual error rate of linear codes over the timing error channel complement interestingly the well-known formula of Eq. (5.16) valid for additive errors.

7.2 Applications

The checkers proposed in this thesis can be deployed over on-chip links where variability in electrical parameters compromises reliable and efficient operation.

We stress that self-calibration of on-chip links can be easily deployed because most of the required building blocks (essentially, the support for voltage scaling and the codec circuitry) are already existing—and even wide-spread in some particular applications. The fact that codes are implemented over on-chip links demonstrates that many applications tolerate transfer variations caused by the recovery from run-time transfer errors. In fact, this feature is very desirable as future technologies will be increasingly error-prone.

Moreover, our approach integrates seamlessly with dynamic voltage and frequency scaling (DVFS) techniques: we propose to control frequency and voltage in a decoupled manner due to the fact that reliability concerns are confined to the checker. As a result, this approach leverages workload prediction methods (such as history-based) that determine the operating frequency in DVFS capable systems.

The network-on-chip design paradigm—a communication-centric approach where a customisable interconnect glues several heterogeneous pre-designed cores—suggests interesting applications for self-calibrating interconnects. Indeed, many such instances use point-to-point links that interconnect computing cores and memories across the whole chip in several cycles. Although network-on-chip embodiments exploit regularity in order to reduce variability, we believe that even regular communication architectures will exhibit a sufficient amount of variability to justify self-calibrating techniques such as the one we propose.

7.3 Future Work

Failure of Circuits at Sub-Critical Voltage

A natural continuation of the work presented in this thesis is to study experimentally how a typical on-chip link fails as its supply voltage is reduced. This knowledge would bring a very complementary insight with respect to the analytical models that we have derived. Moreover, it would enable further refinements of the proposed checkers. Important parameters of the checker combining double sampling flip-flops with codes (such as the delay between the main clock and the clock feeding the shadow latch or the amount of code redundancy) can be adapted in function of the rate at which the link fails, i.e., how steep the actual transition from operating to non-operating occurs in practice. For example, a

very steep transition would favour codes with little redundancy, since mostly the phase matters to detect timing errors under large error rates.

The need to characterise experimentally the failure of computing elements is even more stringent due to the lack of any analytical model in this case. Roberts has already explored this research direction by studying the failure of a multiplier implemented in an FPGA [Roberts *et al.*, 2005]. His work has pointed out inter- and intra-die process variation. In addition, such experimental studies can be performed for synthetic (e.g., random) as well as realistic data sets. Possible data dependencies in the critical path may thus be emphasised. In our opinion, a similar work is required to validate the architectures proposed for self-calibrating adders, e.g., the one depicted in Fig. 6.16.

GALS Designs

GALS (Globally Asynchronous Locally Synchronous) is a broadly-defined design paradigm where locally synchronous islands communicate with each other asynchronously [Chapiro, 1984]. The motivation behind GALS designs is to avoid the distribution of a global clock across the whole chip, to be modular, while at the same time leveraging synchronous design tools because only communication between synchronous islands is performed asynchronously. GALS systems rely on wrappers—referred to as port controllers—to ensure data integrity during transfers between two locally synchronous islands. Port controllers are asynchronous finite state machine that pause (i.e., stretch) the clocks of both synchronous islands while data is being transferred. Applied in a GALS design, soft self-synchronising codes constitute an alternative and attractive solution to traditional asynchronous codes in order to indicate when the data at the receiver end can be safely sampled. Thus, bandwidth could be used more efficiently during data transfers. Although the encoder and decoder logic would be fed by clocks that are not synchronised, the fact that both clocks are paused during the whole transfer ensures consistency between the phase bits generated individually by the encoder and decoder.

7.4 Perspectives

In a near future, we will see chips integrating several billions of transistors that exhibit extreme static and dynamic variations (e.g., in speed or leakage). Handling such sources of variation by worst-case characterisation will be impractical. As a result, today's design technique, namely worst-case design, is fundamentally incompatible with the features chips will soon exhibit. Therefore, we see a urgent need for alternative techniques such as self-calibration.

Although at a very early stage, checkers such as the ones we propose constitute key building blocks required by self-calibrating on-chip links. As such, they contribute to the general trend of building a variety of circuits *reactive* to their operating conditions. This includes environmental factors such as temperature, workload, internal features (e.g., leakage current), application-specific data (e.g., dependencies that Razor flip-flops can exploit) and context (e.g., reconfiguring for a particular environment).

On the one hand, the need for alternative solutions such as reactive circuits is growing because worst-case design poses increasingly large difficulties. On

the other hand, a limiting factor in the development of such circuits stems from the necessity to keep design complexity as low as possible. For example, dynamic frequency scaling for the purpose of thermal management is complex as it renders performance hard to predict: a chip should not fail to operate fast enough simply because it is running hot.

We believe that the work presented in this thesis, together with the techniques mentioned below, are worth developing further because they span the whole design depth and thus explore complementary directions.

- **Self-calibration at the device level.** Kim *et al.* have provided several embodiments, for example, the self-calibration of a keeper based on actual leakage current measurement [2005], or the reduction of hold failure in nano-scaled SRAM [Ghosh *et al.*, 2006]. These techniques target very specific physical phenomena where electrical parameter variations render worst-case design impractical.
- **Self-calibration of circuit operating points.** These techniques include the work presented in this thesis for an on-chip link [Worm *et al.*, 2005], as well as the techniques based on Razor flip-flops [Austin *et al.*, 2004; Kaul *et al.*, 2005; Das *et al.*, 2006]. Such techniques target the detection and correction of timing errors.
- **Algorithmic noise tolerance.** This technique exploits application-specific properties to provide error tolerance [Shanbhag, 2002; Shim *et al.*, 2004]. While application-specific, it is able to recover from any kind of errors—timing or additive.

Through the checkers proposed in this thesis, we make a first step possible towards design techniques that do not rely at all on worst-case characterisation of the designed circuit. In this respect, we propose a more radical paradigm shift than "better than worst-case" designs [Austin *et al.*, 2005]. Although much remains to be done before self-calibrating techniques reach a mature level, we hope that, in the end, the checkers we have proposed will help in establishing alternatives to worst-case design.

Appendix A

Residual Error Rate of Linear Codes Over the Timing Error Channel

In order to derive the undetected errop probability, we start from Eq. (5.8) and focus on the sum

$$\sum_{\substack{t \in C \\ t \geq \tilde{e}}} P\left(\tilde{e} \mid t\right). \tag{A.1}$$

An exact value seems difficult to obtain. As a result, we will bound and approximate this sum. First, we evaluate the summand by showing how the probability $P\left(\tilde{e} \mid t\right)$ is related to the realisation of the error process e_k. Since $\tilde{e} = e \cdot t$, whenever $t_i = 0$, we always have $\tilde{e}_i = 0$. When $t_i = 1$, $\tilde{e}_i = 1$ with probability ε_{t} and $\tilde{e}_i = 0$ with probability $(1 - \varepsilon_{\mathrm{t}})$. We denote by $\mathrm{w}(v)$ the weight of a binary vector v, i.e., the number of bits equal to 1. By definition, there are $\mathrm{w}(t)$ bits of t equal to 1; thus, we obtain that, for any $\tilde{e}$ such that $\tilde{e} \leq t$,

$$P\left(\tilde{e} \mid t\right) = \varepsilon_{\mathrm{t}}^{\mathrm{w}(\tilde{e})} \left(1 - \varepsilon_{\mathrm{t}}\right)^{\mathrm{w}(t) - \mathrm{w}(\tilde{e})}. \tag{A.2}$$

Plugging Eq. (A.2) into Eq. (5.8), we obtain

$$\varepsilon_{\mathrm{res}} = \frac{1}{2^K} \sum_{\tilde{e} \in C \setminus \{\vec{0}\}} \sum_{\substack{t \in C \\ t \geq \tilde{e}}} \varepsilon_{\mathrm{t}}^{\mathrm{w}(\tilde{e})} \left(1 - \varepsilon_{\mathrm{t}}\right)^{\mathrm{w}(t) - \mathrm{w}(\tilde{e})}. \tag{A.3}$$

For convenience, we introduce the following notation.

Definition 15. (the set $Sup(\tilde{e})$). *Let C be a (N, K) linear code and $\tilde{e}$ be a codeword of C. The set $Sup(\tilde{e})$ is defined as the set of all codewords of C that are larger than or equal to $\tilde{e}$:*

$$Sup(\tilde{e}) = \{t \in C \mid t \geq \tilde{e}\}.$$

We call the bit positions i of $\tilde{e}$ where $\tilde{e}_i = 1$ the *"forced to* 1*"* bit positions, because $\tilde{e}_i = 1 \Rightarrow t_i = 1$ for any vector $t \in Sup(\tilde{e})$.

In order to compute the sum of Eq. (A.1), we exploit the fact that the encoding is systematic, which means that the first K bits of any codeword are independent of each other. As a result, for every possible assignment of the first K bits, there exists a codeword that has the assigned bit values in its first K bits positions. We illustrate this property in the following example that shows how a simpler sum than the one of Eq. (A.1) can be bounded.

Example 13. (computation of the sum $\sum_{t\in C}(1-\varepsilon_t)^{\mathbf{w}(t)}$). *Consider the computation of the sum $\sum_{t\in C}(1-\varepsilon_t)^{w(t)}$ for a linear and systematic (N,K) code C. We decompose the weight of every codeword $t \in C$ into the weight of the information bits and the weight of the redundant bits:*

$$w(t) = w_I(t) + w_R(t),$$

where $w_I(t)$ (respectively $w_R(t)$) is the number of 1 *in the information (respectively redundant) part of the codeword t. On one hand, we have clearly $0 \leq w_R(t) \leq N-K$. On the other hand, we know that there are* exactly $\binom{K}{i}$ *codewords such that $w_I(t) = i$. As the first K bits can take any of the 2^K possible assignments of the information bits, we write*

$$\sum_{t\in C}(1-\varepsilon_t)^{w(t)} = \sum_{i=0}^{K}\sum_{\substack{t\in C\\ w_I(t)=i}}(1-\varepsilon_t)^{i+w_R(t)}.$$

Using the bounds on $w_R(t)$, we obtain directly two bounds

$$\begin{aligned}\sum_{t\in C}(1-\varepsilon_t)^{w(t)} &\leq \sum_{i=0}^{K}\binom{K}{i}(1-\varepsilon_t)^{i+N-K},\\ \sum_{t\in C}(1-\varepsilon_t)^{w(t)} &\geq \sum_{i=0}^{K}\binom{K}{i}(1-\varepsilon_t)^{i}.\end{aligned}$$

We would like to apply the very same procedure to compute the sum of Eq. (A.1). To do so, we have to account for the constraint $t \geq \tilde{e}$, which imposes $t_i = 1$ for each bit position i such that $\tilde{e}_i = 1$. In addition, the code C imposes $N-K$ dependence relations between the N bits. The difficulty is that we have to include in the sum only the N-uples that (i) have a 1 at each bit position that is forced to 1 by $\tilde{e}$ and (ii) still satisfy the $N-K$ dependence relations. Fig. A.1 illustrates these constraints. Before showing how this question can be dealt with, we formalise the definition of "dependence-free" bit position.

Definition 16. ((maximum) set of free bit positions for a set S). *Let S be a set of vectors in $\{0,1\}^N$, E be an ordered subset of bit position indices: $E \subseteq \{1,...,N\}$. For any vector $s \in S$, let s_E be the vector of $\{0,1\}^{|E|}$ that is obtained by keeping the entries s_i whose indices i belong to E, i.e., $s_{E,i} = s_i \; \forall i \in E$.*

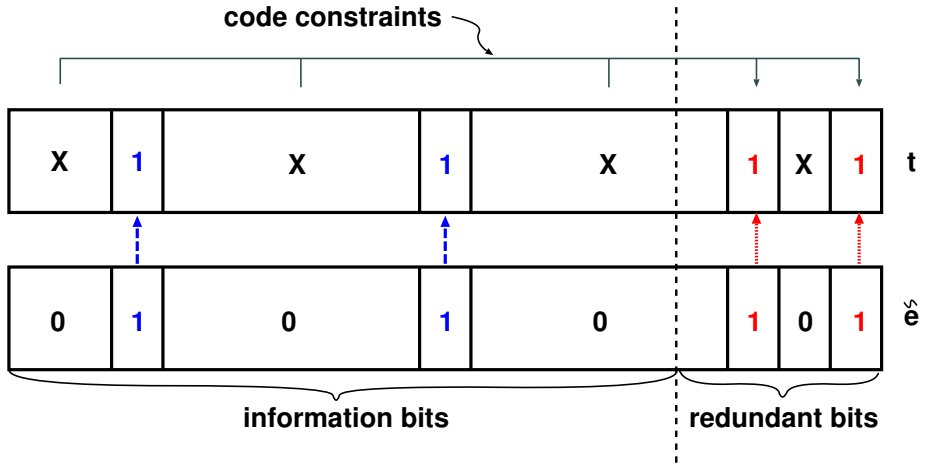

Figure A.1: Information bits of $\tilde{e}$ that are 1 impose that the bits at the same position in the codeword t are also 1 as indicated by the two leftmost vertical arrows. Similarly, the redundant bits of $\tilde{e}$ that are 1 impose that the corresponding bit positions are 1 for each element of $Sup(\tilde{e})$. This constraint is not be met under any possible assignment of unconstrained information bit positions marked by a "X".

We define E as a set of free bit positions if and only if

$$\textit{for all } a \in \{0,1\}^{|E|}, \textit{there exists } s \in S \textit{ such that } s_E = a.$$

The set E is maximum if there exists no other set of free bit positions, $F \neq E$, with $E \subset F$.

In other words, a set E of bit position indices is a set of free bit positions for a set S if and only if, for all possible assignments of the bit positions with indices in E, there exists elements of S having the assigned values. We call the bit positions belonging to a free set of bit positions the *free* bit positions. We illustrate the definition with an example.

Example 14. (set of free bit positions for a code).
Consider a linear and systematic $(N = 5, K = 3)$ code C, where the first redundant bit is the sum of the first two information bits and the second redundant bit is the sum of the last two information bits. This code is given in Table A.1. Suppose that $\tilde{e} = (1,0,0,1,0)$, we obtain $Sup(\tilde{e}) = \{(1,0,0,1,0);(1,0,1,1,1)\}$. By visual inspection of $Sup(\tilde{e})$, we find two sets of free bit positions: $E = \{3\}$ and $E = \{5\}$. The maximum cardinality of a set of free bit positions is therefore 1. It cannot be more, since $Sup(\tilde{e})$ contains two codewords. Now, if $\tilde{e} = (0,0,1,0,1)$, we find: $Sup(\tilde{e}) = \{(0,0,1,0,1);(1,0,1,1,1)\}$, yielding $E = \{1\}$ and $E = \{4\}$ as maximum sets of free bit positions. However, if $\tilde{e} = (0,1,0,1,1)$, we realize that $Sup(\tilde{e})$ only contains $\tilde{e}$ itself, so that the set of free bit positions is empty.

Let us assume that we know a maximum set of free bit positions for $Sup(\tilde{e})$ (we defer the question of how to find it). We call the set M. We point out that

x_1	x_2	x_3	x_4	x_5
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	0	1
1	1	1	0	0

Table A.1: A linear $(N = 5, K = 3)$ code.

M is not uniquely defined, as shown in Example 14; however, $|M|$ is unique because M is a maximum set of free bit positions. We distinguish the following types of bit positions within the set $Sup(\tilde{e})$:

(i) The positions of the bits forced to 1. There are $\mathrm{w}(\tilde{e})$ such bit positions.

(ii) Knowing (i), we determine the free bit positions of $Sup(\tilde{e})$ (i.e., the bit positions with indices in M).

(iii) The remaining bit positions, i.e., the bit positions which are not included in (i) and (ii).

Based on this decomposition and for every $t \in Sup(\tilde{e})$, we write

$$\mathrm{w}(t) = \mathrm{w}(\tilde{e}) + \mathrm{w_F}(t) + \mathrm{w_{rem.}}(t)\,, \tag{A.4}$$

where $\mathrm{w_F}(t)$ designates the number of 1 in the bit positions with indices in M and where $\mathrm{w_{rem.}}(t)$ is the number of 1 in the remaining bit positions (iii). We follow now exactly the same procedure as in Example 13. That is, we decompose the sum in function of the weight assigned to the set of free bit positions:

$$\begin{aligned}\sum_{t\in Sup(\tilde{e})} (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(t)} &= (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(\tilde{e})} + \sum_{\substack{t\in C\\ t>\tilde{e}}} (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(t)} \\ &= (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(\tilde{e})} + \sum_{i=1}^{|M|} \sum_{\substack{t\in C; t>\tilde{e}\\ \mathrm{w_F}(t)=i}} (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(\tilde{e})+i+w_{\mathrm{rem.}}(t)}. \end{aligned} \tag{A.5}$$

The difference with Example 13 is that we do not know how to evaluate easily $|M|$, because we sum only over those codewords that are larger than $\tilde{e}$. As a result, we only bound the cardinality of this set and give later in Lemma 4 an expression for an upper and a lower bound, denoted respectively by M_h and M_l. We obtain trivially from Eq. (A.4)

$$\mathrm{w}(t) \geq \mathrm{w}(\tilde{e}) + \mathrm{w_F}(t), \tag{A.6}$$

and

$$\begin{aligned}\mathrm{w}(t) &\leq \mathrm{w}(\tilde{e}) + \mathrm{w_F}(t) + N - \mathrm{w}(\tilde{e}) - |M| \\ &= \mathrm{w_F}(t) + N - |M|. \end{aligned} \tag{A.7}$$

Lower bounding $|M|$ by M_l and using the upper bound of Eq. (A.7), we obtain

$$\begin{aligned}\sum_{t\in Sup(\tilde{e})}(1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(t)} &\geq (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(\tilde{e})}+\sum_{i=1}^{M_l}\sum_{\substack{t\in C;t>\tilde{e}\\ \mathrm{w_F}(t)=i}}(1-\varepsilon_{\mathrm{t}})^{N+i-M_l}\\ &= (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(\tilde{e})}+\sum_{i=1}^{M_l}\binom{M_l}{i}(1-\varepsilon_{\mathrm{t}})^{N+i-M_l},\\ &= (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(\tilde{e})}+(1-\varepsilon_{\mathrm{t}})^{N-M_l}\left((2-\varepsilon_{\mathrm{t}})^{M_l}-1\right), \end{aligned} \tag{A.8}$$

where, in the second equality, we have exploited the fact that there are exactly $\binom{M_l}{i}$ codewords t of $Sup(\tilde{e})$ such that $\mathrm{w_F}(t)=i$, and, in the last equality, we have used the binomial formula:

$$(1+x)^Q=\sum_{i=0}^{Q}\binom{Q}{i}x^i,$$

with Q an integer.

Similarly, we upper bound $|M|$ by M_h and combine with the lower bound of Eq. (A.6) to get the following upper bound on the sum of Eq. (A.1):

$$\begin{aligned}\sum_{t\in Sup(\tilde{e})}(1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(t)} &\leq \sum_{i=0}^{M_h}\sum_{\substack{t\in C;t\geq\tilde{e}\\ \mathrm{w_F}(t)=i}}(1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(\tilde{e})+i}\\ &= (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(\tilde{e})}\sum_{i=0}^{M_h}\binom{M_h}{i}(1-\varepsilon_{\mathrm{t}})^{i}\\ &= (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(\tilde{e})}(2-\varepsilon_{\mathrm{t}})^{M_h}. \end{aligned} \tag{A.9}$$

The derivation of M_l and M_h is given in the following lemma proven at the end of this appendix.

Lemma 4. (cardinality of a maximum set of free bit positions for $Sup(\tilde{e})$). *Let C be a (N,K) systematic linear code and let $C^\perp$ be the $(N,N-K)$ code orthogonal to C. Let $\tilde{e}$ be a codeword of C and M be a maximum set of free bit positions for $Sup(\tilde{e})$. Then, we have the bounds*

$$|M|\geq M_l=\begin{cases}K-w(\tilde{e}) & \text{if } w(\tilde{e})\leq K,\\ 0 & \text{otherwise.}\end{cases} \tag{A.10}$$

and, if C satisfies Hypothesis 1,

$$|M|\leq M_h=\begin{cases}K-w(\tilde{e}) & \text{if } w(\tilde{e})\leq d\left(C^\perp\right)-1,\\ K-d\left(C^\perp\right)+1 & \text{if } d\left(C^\perp\right)\leq w(\tilde{e})\leq N-K+d\left(C^\perp\right)-1,\\ N-w(\tilde{e}) & \text{if } w(\tilde{e})\geq N-K+d\left(C^\perp\right),\end{cases} \tag{A.11}$$

with $d\left(C^\perp\right)$ denoting the minimum distance of $C^\perp$.

As a corollary of this lemma, $|M| = K - \mathrm{w}(\tilde{e})$ whenever $\mathrm{w}(\tilde{e}) \leq \mathrm{d}\left(C^{\perp}\right) - 1$. The lower bound is not specific to a certain code, whereas the upper bound requires some information specific to the code, namely the minimum distance of the code $C^{\perp}$.

Inserting the values of M_l in Eq. (A.8) yields directly the lower bound

$$\begin{aligned}&\sum_{t \in Sup(\tilde{e})} (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(t)} \\ &\geq \begin{cases} (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(\tilde{e})} \left(1 + (1-\varepsilon_{\mathrm{t}})^{N-K} \left((2-\varepsilon_{\mathrm{t}})^{K-\mathrm{w}(\tilde{e})} - 1\right)\right) & \text{if } \mathrm{w}(\tilde{e}) \leq K, \\ (1-\varepsilon_{\mathrm{t}})^{\mathrm{w}(\tilde{e})} & \text{otherwise.} \end{cases}\end{aligned} \tag{A.12}$$

We obtain a lower bound on $\varepsilon_{\mathrm{res}}$ by plugging Eq. (A.12) in Eq. (A.3):

$$\begin{aligned}\varepsilon_{\mathrm{res}} &\geq \frac{1}{2^K} \sum_{i=1}^{K} \sum_{\substack{\tilde{e} \in C \\ \mathrm{w}(\tilde{e})=i}} {\varepsilon_{\mathrm{t}}}^{i} \left(1 + (1-\varepsilon_{\mathrm{t}})^{N-K} \left((2-\varepsilon_{\mathrm{t}})^{K-i} - 1\right)\right) \\ &+ \frac{1}{2^K} \sum_{i=K+1}^{N} \sum_{\substack{\tilde{e} \in C \\ \mathrm{w}(\tilde{e})=i}} \varepsilon_{\mathrm{t}}^{i} \\ &= \frac{1}{2^K} \left\{ \sum_{i=1}^{K} A_i \, {\varepsilon_{\mathrm{t}}}^{i} \left(1 + (1-\varepsilon_{\mathrm{t}})^{N-K} \left((2-\varepsilon_{\mathrm{t}})^{K-i} - 1\right)\right) + \sum_{i=K+1}^{N} A_i \, {\varepsilon_{\mathrm{t}}}^{i} \right\},\end{aligned} \tag{A.13}$$

with A_i the number of codewords of weight i.

In order to avoid heavy notations, we do not develop the value of M_h needed to upper bound $\varepsilon_{\mathrm{res}}$. We plug Eq. (A.9) into Eq. (A.3), which yields

$$\begin{aligned}\varepsilon_{\mathrm{res}} &\leq \frac{1}{2^K} \sum_{i=1}^{N} \sum_{\substack{\tilde{e} \in C \\ \mathrm{w}(\tilde{e})=i}} {\varepsilon_{\mathrm{t}}}^{i} \, (2-\varepsilon_{\mathrm{t}})^{M_h(i)} \\ &= \frac{1}{2^K} \sum_{i=1}^{N} A_i \, {\varepsilon_{\mathrm{t}}}^{i} \, (2-\varepsilon_{\mathrm{t}})^{M_h(i)},\end{aligned} \tag{A.14}$$

with M_h depending on $\mathrm{w}(\tilde{e})$ as indicated in Eq. (A.11).

In Eq. (A.13), the exponent $N-K$ upper bounds the weight of the redundant bits of t. Approximating the latter quantity to $\left\lfloor \frac{N-K}{2} \right\rfloor$ yields the following approximation of $\varepsilon_{\mathrm{res}}$:

$$\varepsilon_{\mathrm{res}} \cong \frac{1}{2^K} \left\{ \sum_{i=1}^{K} A_i \, {\varepsilon_{\mathrm{t}}}^{i} \left(1 + (1-\varepsilon_{\mathrm{t}})^{\left\lfloor \frac{N-K}{2} \right\rfloor} \left((2-\varepsilon_{\mathrm{t}})^{K-i} - 1\right)\right) + \sum_{i=K+1}^{N} A_i \, {\varepsilon_{\mathrm{t}}}^{i} \right\}. \tag{A.15}$$

To conclude, we prove Lemma 4.

Proof. **(Lemma 4).** Given a codeword $\tilde{e}$, we construct a sequence of $\mathrm{w_R}(\tilde{e})+1$ vectors, $\{\tilde{e}^i\}_{i=0,\ldots,\mathrm{w_R}(\tilde{e})}$ with $\tilde{e}^{\mathrm{w_R}(\tilde{e})}=\tilde{e}$ that enables to bound the cardinality of a maximum set of free bit positions for $Sup\big(\tilde{e}^{\mathrm{w_R}(\tilde{e})}\big)$. Before defining the sequence, we introduce some needed definitions.

We decompose the weight of $\tilde{e}$ into two contributions: $\mathrm{w}(\tilde{e})=\mathrm{w_I}(\tilde{e})+\mathrm{w_R}(\tilde{e})$, where $\mathrm{w_I}(\tilde{e})$ is the weight of the information bits of $\tilde{e}$ and $\mathrm{w_R}(\tilde{e})$ is the weight of the redundant bits of $\tilde{e}$. For convenience, we denote by I the set of information bit positions, i.e., $I=\{1,...,K\}$, and by R the set of redundant bit positions, i.e., $R=\{1,...,N-K\}$. Moreover, we define the two following sets of bit positions.

Definition 17. (wrong information bit positions).

$$WI=\{l\in I\mid \tilde{e}_l=1\}.$$

Definition 18. (wrong redundant bit positions).

$$WR=\{r\in R\mid \tilde{e}_{K+r}=1\}.$$

We order the elements of WR arbitrarily: $WR=\big\{r_1,...,r_{\mathrm{w_R}(\tilde{e})}\big\}$ and use this ordering designate the parity check constraints associated with the redundant bit positions r_i, $i=1,...,\mathrm{w_R}(\tilde{e})$, i.e., the i^{th} parity check constraint is the one defined by $\bigoplus_{l=1}^{K} x_l\cdot p_{l,r_i}=x_{K+r_i}=1$.

The initial vector of the sequence, $\tilde{e}^0$ contains 1 only in the information bit positions j where $\tilde{e}_j=1$:

$$\begin{aligned}&\tilde{e}^0_j=1,\ \text{for all } j\in WI,\ \text{and}\\&\tilde{e}^0_j=0,\ \text{for all } j\in\{1,...,N\}\setminus WI.\end{aligned}$$

The vector $\tilde{e}^i$, $1\le i\le \mathrm{w_R}(\tilde{e})$ is constructed from $\tilde{e}^{i-1}$ as follows:

$$\begin{aligned}&\tilde{e}^i_j=\tilde{e}^{i-1}_j,\ \text{ for all } j\in\{1,...,N\}\setminus\{K+r_i\}\ \text{ and}\\&\tilde{e}^i_{K+r_i}=1.\end{aligned}$$

As $\tilde{e}^{i-1}_{K+r_i}=0$, we observe that $\tilde{e}^i$ differs from $\tilde{e}^{i-1}$ only in bit position $K+r_i$. Clearly, the last vector of this sequence is $\tilde{e}^{\mathrm{w_R}(\tilde{e})}=\tilde{e}$. Since $\mathrm{w_R}(\tilde{e}^0)=0$, $I\setminus WI$ is a maximum set of free bit positions for $Sup\big(\tilde{e}^0\big)$. Then, at each iteration $i, 1\le i\le \mathrm{w_R}(\tilde{e})$, the i^{th} parity check imposes

$$\bigoplus_{j=1}^{K} x_j\cdot p_{j,r_i}=1, \tag{A.16}$$

which we can rewrite as

$$\bigoplus_{j\in WI} x_j\cdot p_{j,r_i}\oplus\bigoplus_{j\in I\setminus WI} x_j\cdot p_{j,r_i}=1. \tag{A.17}$$

Because for all $x\in Sup\big(\tilde{e}^i\big)$ and for all bit position $j\in WI$, $x_j\ge\tilde{e}^i_j=\tilde{e}_j=1$, it holds that

$$\bigoplus_{j\in WI} x_j\cdot p_{j,r_i}=\bigoplus_{j\in WI}\tilde{e}_j\cdot p_{j,r_i}=1.$$

Therefore, we obtain

$$\bigoplus_{j\in I\setminus WI} x_j \cdot p_{j,r_i} = 0 \text{ for all } x \in Sup\big(\tilde{e}^i\big). \tag{A.18}$$

We call Eq. (A.18) the equation *associated with parity check* r_i. Any codeword $x \in Sup\big(\tilde{e}^{\mathrm{w_R}(\tilde{e})}\big)$ satisfies $\mathrm{w_R}(\tilde{e})$ equations such as Eq. (A.18). It follows that the maximum number $|M|$ of free bit positions for $Sup\big(\tilde{e}^{\mathrm{w_R}(\tilde{e})}\big)$ is the maximum number of variables x_j, $j \in I \setminus WI$, that can be assigned freely while satisfying the following system of $\mathrm{w_R}(\tilde{e})$ equations:

$$\begin{aligned}
\bigoplus_{j\in I\setminus WI} x_j \cdot p_{j,r_1} &= 0,\\
\bigoplus_{j\in I\setminus WI} x_j \cdot p_{j,r_2} &= 0,\\
&\ldots\\
\bigoplus_{j\in I\setminus WI} x_j \cdot p_{j,r_{\mathrm{w_R}(\tilde{e})}} &= 0.
\end{aligned} \tag{A.19}$$

Let T be the number of linearly independent equations in the system (A.19). Then, $|M|$ is the difference between the number of information bits not affected by an error, $K - \mathrm{w_I}(\tilde{e})$, and T:

$$|M| = K - \mathrm{w_I}(\tilde{e}) - T. \tag{A.20}$$

Obviously, $T \leq \mathrm{w_R}(\tilde{e})$. Hence Eq. (A.20) yields

$$|M| \geq \max\left(K - \mathrm{w}(\tilde{e}), 0\right), \tag{A.21}$$

which is Eq. (A.10).

In order to obtain an upper bound on $|M|$, we need to lower bound T. We proceed as follows. First, we observe that if $T = \mathrm{w_R}(\tilde{e})$, all equations in the system (A.19) are independent and thus $|M| = K - \mathrm{w}(\tilde{e})$. From now on, we assume $T < \mathrm{w_R}(\tilde{e})$. Let i_0 be the index of the first parity check for which the associated equation is linearly *dependent* on the ones associated with the previous parity checks. That is, the equation $\bigoplus_{j\in I\setminus WI} x_j \cdot p_{j,r_{i_0}} = 0$ is a linear combination of the equations associated with parity check r_{q_1},..., r_{q_Q} for some indices $q_1, ..., q_Q$, with $Q \leq i_0 - 1$. Let $\widehat{C}$ be the systematic code defined by the generator matrix $\widehat{G} = \begin{pmatrix} I_K & \widehat{P} \end{pmatrix}$ where I_K is the identity matrix of size K and where the matrix $\widehat{P}$ is identical to P except in the column r_{i_0}, i.e., $\hat{p}_r = p_r$ for all $r \in R \setminus \{r_{i_0}\}$. The column r_{i_0} of $\widehat{P}$ is obtained as

$$\hat{p}_{r_{i_0}} = p_{r_{i_0}} \oplus p_{r_{q_1}} \oplus ... \oplus p_{r_{q_Q}}. \tag{A.22}$$

Because $\widehat{C}$ is obtained from C by a so called elementary operation (namely, the column r_{i_0} of $\widehat{P}$ is a linear combination of some columns of P), the two codes are said *equivalent*, that is they have the same coding properties. In particular, their minimum distance is the same.

Because P is full rank, so is $\widehat{P}$. As a result, the column r_{i_0} of $\widehat{P}$, $\widehat{p}_{r_{i_0}}$, is not the zero vector. Let ρ be the number of 1 in this column: $\rho = \sum_{j=1}^{K} \hat{p}_{j,r_{i_0}}$. Since

the parity check i_0 is linearly dependent of parity check r_{q_1},..., r_{q_Q}, Eq. (A.22) imposes that $\widehat{P}_{j,r_{i_0}} = 0$ for all $j \in I \setminus WI$. It follows that

$$\rho \leq \mathrm{w_I}(\tilde{e}). \tag{A.23}$$

Let $\widehat{H}$ be the parity check matrix of $\widehat{C}$: $\widehat{H} = \left(\widehat{P}^T \quad I_{N-K}\right)$. $\widehat{H}$ is identical to the parity check matrix H of C, except the row $\widehat{h}_{r_{i_0}}$:

$$\begin{aligned} \widehat{h}_r &= h_r \text{ for all } r \in R \setminus \{r_{i_0}\}, \\ \widehat{h}_{r_{i_0}} &= h_{r_{i_0}} \oplus h_{r_{q_1}} \oplus ... \oplus h_{r_{q_Q}}. \end{aligned}$$

Since the rows of $\widehat{H}$ are codewords of $\widehat{C}^\perp$, the minimum distance of $\widehat{C}^\perp$ is upper bounded by the number of 1 in the row $\widehat{h}_{r_{i_0}}$ of $\widehat{H}$. Therefore,

$$\mathrm{d}\left(\widehat{C}^\perp\right) \leq \mathrm{w}(\widehat{h}_{r_{i_0}}) = \rho + 1 + Q.$$

In the last equality, we have used the fact that $\widehat{h}_{r_{i_0}}$ has exactly ρ 1 in the first K bit positions and exactly $1 + Q$ in the remaining $N - K$ bit positions. Because, in addition, $Q \leq i_0 - 1$, we obtain

$$\mathrm{d}\left(\widehat{C}^\perp\right) \leq \rho + i_0. \tag{A.24}$$

Combining Eqs. (A.23) and (A.24), we get

$$\mathrm{d}\left(\widehat{C}^\perp\right) \leq \mathrm{w_I}(\tilde{e}) + i_0.$$

Now, the number T of equations linearly independent in Eq. (A.19) is at least $i_0 - 1$, by definition of i_0. Thus, $i_0 \leq T + 1$ and

$$\mathrm{d}\left(\widehat{C}^\perp\right) \leq \mathrm{w_I}(\tilde{e}) + T + 1. \tag{A.25}$$

By adding $\mathrm{w_R}(\tilde{e})$ on both sides of Eq. (A.25) and reordering the terms, we obtain finally

$$\mathrm{w_R}(\tilde{e}) - T \leq \mathrm{w}(\tilde{e}) - \mathrm{d}\left(\widehat{C}^\perp\right) + 1.$$

Since the codes $\widehat{C}^\perp$ and $C^\perp$ are equivalent, their minimum distance is the same. Thus

$$\mathrm{w_R}(\tilde{e}) - T \leq \mathrm{w}(\tilde{e}) - \mathrm{d}\left(C^\perp\right) + 1. \tag{A.26}$$

By definition of T, $\mathrm{w_R}(\tilde{e}) - T$ is the number of equations linearly dependent in Eq. (A.19). From Eq. (A.26), we deduce that, if $\mathrm{w}(\tilde{e}) \leq \mathrm{d}\left(C^\perp\right) - 1$, all equations in Eq. (A.19) are linearly independent and therefore $|M| = K - \mathrm{w}(\tilde{e})$.

In addition, if $\mathrm{w}(\tilde{e}) \geq \mathrm{d}\left(C^\perp\right)$, we have the two following upper bounds. First, $N - \mathrm{w}(\tilde{e})$ is a trivial upper bound on $|M|$ because $\mathrm{w}(\tilde{e})$ bit positions are forced to 1 and thus at most the $N - \mathrm{w}(\tilde{e})$ remaining ones can constitute a set of free bit positions. Secondly, we have deduced that $|M| = K - \mathrm{d}\left(\widehat{C}^\perp\right) + 1$ if $\mathrm{w}(\tilde{e}) = \mathrm{d}\left(\widehat{C}^\perp\right) - 1$. Since $|M|$ is a decreasing function of $\mathrm{w}(\tilde{e})$, we have also

$|M| \leq K - \mathrm{d}\left(C^{\perp}\right) + 1$ whenever $\mathrm{w}(\tilde{e}) \geq \mathrm{d}\left(\widehat{C}^{\perp}\right)$. Combining these two upper bounds yields $|M| \leq \min\left(K - \mathrm{d}\left(C^{\perp}\right) + 1, N - \mathrm{w}(\tilde{e})\right)$. Finally, the overall upper bound as a function of $\mathrm{w}(\tilde{e})$ reads

$$|M| \leq \begin{cases} K - \mathrm{w}(\tilde{e}) & \text{if } \mathrm{w}(\tilde{e}) \leq \mathrm{d}\left(C^{\perp}\right) - 1, \\ K - \mathrm{d}\left(C^{\perp}\right) + 1 & \text{if } \mathrm{d}\left(C^{\perp}\right) \leq \mathrm{w}(\tilde{e}) \leq N - K + \mathrm{d}\left(C^{\perp}\right) - 1, \\ N - \mathrm{w}(\tilde{e}) & \text{if } \mathrm{w}(\tilde{e}) \geq N - K + \mathrm{d}\left(C^{\perp}\right). \end{cases} \quad \text{(A.27)}$$

□

Appendix B

Residual Error Rate of Alternating-Phase Encoding With Linear Codes Over the Timing Error Channel

We proceed as in Appendix A. We consider a $(N+1, K+1)$ code C and recall that the same assumptions as the one stated in Sec. 5.5.1 are made: C is linear, systematic, and its codewords are equally likely to be transmitted. Futhermore, we assume that Hypothesis 1 is verified. We point out that the symbol K still denotes the number of information bits, excluding thus the alternating-phase bit. Without loss of generality, we can assume that the alternating-phase bit is located at the least significant bit, i.e., x_1 is the alternating-phase bit of a codeword $x \in C$. We first introduce the following notations.

Definition 19. ($\mathbf{LSB}_0(.)$ and $\mathbf{LSB}_1(.)$). *Let S be a set of vectors of $\{0,1\}^N$, for a certain integer N. We define $LSB_0(S)$ (respectively $LSB_1(S)$) as the subset of vectors of S, which have 0 (respectively 1) as the least significant bit:*

$$LSB_i(S) = \{s \in S \mid s_1 = i\};\ i \in 0, 1.$$

In order to compute ε_{res}, we take the same starting point as in Eq. (5.5)

$$\varepsilon_{\text{res}} = \sum_{\tilde{e} \in C \setminus \{\vec{0}\}} \sum_{\substack{t \in C \\ t \geq \tilde{e}}} P(\tilde{e} \mid t)\ P(t). \tag{B.1}$$

The alternating-phase encoding imposes two constraints:

- The alternating-phase bit is not transmitted and hence is never corrupted by an error. Consequently, $\tilde{e}_1 = 0$ for all $\tilde{e}$. Henceforth, the error vector $\tilde{e}$ belongs to $\text{LSB}_0(C)$.

- $t_1 = 1$, since the least significant bits of two consecutive codewords are always different. Hence, the transition vector t belongs to $\mathrm{LSB}_1(C)$.

By linearity of the code C and due to the codeword equiprobability assumption, the transition vector t is uniformly distributed over $\mathrm{LSB}_1(C)$, i.e.,

$$P(t) = \begin{cases} \frac{1}{|\mathrm{LSB}_1(C)|} = \frac{1}{2^K} & \text{if } t \in \mathrm{LSB}_1(C), \\ 0 & \text{otherwise.} \end{cases}$$

In the last equation, we have used the fact that half of the codewords of the $(N+1, K+1)$ code C have 1 as least significant bit. Plugging the obtained expression into Eq. (B.1) yields

$$\begin{aligned} \varepsilon_{\mathrm{res}} &= \sum_{\tilde{e} \in \mathrm{LSB}_0(C) \setminus \{\vec{0}\}} \sum_{\mathrm{LSB}_1(Sup(\tilde{e}))} P(t)\ P(\tilde{e} \mid t) \\ &= \frac{1}{2^K} \sum_{\tilde{e} \in \mathrm{LSB}_0(C) \setminus \{\vec{0}\}} \sum_{t \in \mathrm{LSB}_1(Sup(\tilde{e}))} P(\tilde{e} \mid t). \end{aligned} \tag{B.2}$$

Now, in order to evaluate $P(\tilde{e} \mid t)$, we make the same reasoning as the one made to obtain Eq. (A.2). The only difference is that, deterministically, the realisation of the error process e is 0 for the least significant bit (i.e., $e_1 = 0$), since the alternating-phase bit is not transmitted. As a result, we remove a factor $1 - \varepsilon_{\mathrm{t}}$ from Eq. (A.2) and obtain

$$\begin{aligned} \varepsilon_{\mathrm{res}} &= \frac{1}{2^K} \sum_{\tilde{e} \in \mathrm{LSB}_0(C) \setminus \{\vec{0}\}} \sum_{t \in \mathrm{LSB}_1(Sup(\tilde{e}))} P(\tilde{e} \mid t) \\ &= \frac{1}{2^K} \sum_{\tilde{e} \in \mathrm{LSB}_0(C) \setminus \{\vec{0}\}} \varepsilon_{\mathrm{t}}^{\mathrm{w}(\tilde{e})} \sum_{t \in \mathrm{LSB}_1(Sup(\tilde{e}))} (1 - \varepsilon_{\mathrm{t}})^{\mathrm{w}(t) - 1 - \mathrm{w}(\tilde{e})}. \end{aligned} \tag{B.3}$$

We focus now on bounding the sum

$$\sum_{t \in \mathrm{LSB}_1(Sup(\tilde{e}))} (1 - \varepsilon_{\mathrm{t}})^{\mathrm{w}(t) - 1 - \mathrm{w}(\tilde{e})}. \tag{B.4}$$

Like in Appendix A, we bound the size of a maximum set of free bit positions for the set $\mathrm{LSB}_1(Sup(\tilde{e}))$ (the proof of Lemma 5 is given at the end of this appendix).

Lemma 5. (maximum set of free bit positions for $\mathrm{LSB}_1(Sup(\tilde{e}))$). *Let C be a $(N+1, K+1)$ systematic linear code satisfying Hypothesis 1 (stated in Appendix A) and let $\tilde{e}$ be a codeword of C such that $\tilde{e}_1 = 0$. The upper bound of Eq. (A.11) also holds for the set $LSB_1(Sup(\tilde{e}))$.*

Contrary to Lemma 4, we have only derived an upper bound since $\mathrm{LSB}_1(Sup(\tilde{e}))$ may be an empty set.

We continue by upper bounding the summand of Eq. (B.4). To do so, we lower-bound $\mathrm{w}(t)$, $t \in \mathrm{LSB}_1(Sup(\tilde{e}))$. We write

$$\mathrm{w}(t) \geq 1 + \mathrm{w}(\tilde{e}) + \mathrm{w}_{\mathrm{F}}(t), \tag{B.5}$$

with the first term of the RHS being the 1 in the least significant bit position, the second term the required 1 in bit positions where $\tilde{e}$ has 1 and the last term the 1 in the free bit positions of $\mathrm{LSB}_1\left(Sup(\tilde{e})\right)$. Moreover, we know that any two codewords of the $(N+1, K+1)$ code C differ in at least d bit positions, where d is the minimum distance of the code. In particular, any codeword $t \in \mathrm{LSB}_1\left(Sup(\tilde{e})\right)$ differs in at least d bit positions with $\tilde{e}$. Because, in addition, $t \geq \tilde{e}$, there exists at least d bit positions j such that $t_j = 1$ and $\tilde{e}_j = 0$. Therefore, the following holds:

$$\mathrm{w}(t) \geq \mathrm{w}(\tilde{e}) + \mathrm{d}. \tag{B.6}$$

By combining Eqs. (B.5) and (B.6), we obtain

$$\mathrm{w}(t) - \mathrm{w}(\tilde{e}) - 1 \geq \max\left(\mathrm{d} - 1, \mathrm{w_F}(t)\right). \tag{B.7}$$

Introducing the bounds of Lemma 5 and of Eq. (B.7) in Eq. (B.4) yields

$$\begin{aligned}
\sum_{t \in \mathrm{LSB}_1(Sup(\tilde{e}))} (1-\varepsilon_\mathrm{t})^{\mathrm{w}(t)-\mathrm{w}(\tilde{e})-1} &= \sum_{\substack{t \in \mathrm{LSB}_1(Sup(\tilde{e})) \\ \mathrm{w_F}(t)=0}} (1-\varepsilon_\mathrm{t})^{\mathrm{w}(t)-\mathrm{w}(\tilde{e})-1} \\
&\quad + \sum_{l=1}^{M_h} \sum_{\substack{t \in \mathrm{LSB}_1(Sup(\tilde{e})) \\ \mathrm{w_F}(t)=l}} (1-\varepsilon_\mathrm{t})^{\mathrm{w}(t)-\mathrm{w}(\tilde{e})-1} \\
&\leq (1-\varepsilon_\mathrm{t})^{\mathrm{d}-1} + \sum_{l=1}^{M_h} \binom{M_h}{l} (1-\varepsilon_\mathrm{t})^{l} \\
&= (1-\varepsilon_\mathrm{t})^{\mathrm{d}-1} + \left((2-\varepsilon_\mathrm{t})^{M_h} - 1\right),
\end{aligned} \tag{B.8}$$

where M_h depends on $\mathrm{w}(\tilde{e})$ and is defined in Eq. (A.11). We obtain an upper bound on ε_res by combining Eqs. (B.3) and (B.8). Again, for the sake of readability, we do not develop further the value of M_h and obtain

$$\begin{aligned}
\varepsilon_\mathrm{res} &= \frac{1}{2^K} \sum_{\tilde{e} \in \mathrm{LSB}_0(C)\setminus\{\vec{0}\}} \varepsilon_\mathrm{t}^{\mathrm{w}(\tilde{e})} \sum_{t \in \mathrm{LSB}_1(Sup(\tilde{e}))} (1-\varepsilon_\mathrm{t})^{\mathrm{w}(t)-\mathrm{w}(\tilde{e})-1} \\
&\leq \frac{1}{2^K} \sum_{i=1}^{N} \sum_{\substack{\tilde{e} \in \mathrm{LSB}_0(C) \\ \mathrm{w}(\tilde{e})=i}} {\varepsilon_\mathrm{t}}^i \left((1-\varepsilon_\mathrm{t})^{\mathrm{d}-1} + \left((2-\varepsilon_\mathrm{t})^{M_h(i)} - 1\right)\right) \\
&= \frac{1}{2^K} \sum_{i=1}^{N} A_i \, {\varepsilon_\mathrm{t}}^i \left((1-\varepsilon_\mathrm{t})^{\mathrm{d}-1} + \left((2-\varepsilon_\mathrm{t})^{M_h(i)} - 1\right)\right),
\end{aligned} \tag{B.9}$$

where A_i is the number of weight-i codewords in the (N, K) code C, and where d is the minimum distance of the $(N+1, K+1)$ code.

We obtain an approximation of ε_res as follows. First, we approximate the quantity $\mathrm{w}(t) - \mathrm{w}(\tilde{e})$ for $t \in \mathrm{LSB}_1\left(Sup(\tilde{e})\right)$, which is the number of bit positions i such that $t_i = 1$ and $\tilde{e}_i = 0$. We write $\mathrm{w}(t) - \mathrm{w}(\tilde{e}) = \mathrm{w_I}(t) - \mathrm{w_I}(\tilde{e}) + \mathrm{w_R}(t) - \mathrm{w_R}(\tilde{e})$ and approximate $\mathrm{w_I}(t) - \mathrm{w_I}(\tilde{e})$ to $1 + \mathrm{w_F}(t)$. Since $\mathrm{w_R}(t) - \mathrm{w_R}(\tilde{e})$ is an integer

between 0 and $N-K$, we approximate the latter quantity to the arithmetic average over the interval $\left\lfloor \frac{N-K}{2} \right\rfloor$, which yields

$$\mathrm{w}(t) - \mathrm{w}(\tilde{e}) \cong 1 + \mathrm{w_F}(t) + \left\lfloor \frac{N-K}{2} \right\rfloor .$$

This approximation leads to

$$\varepsilon_{\mathrm{res}} \cong \frac{1}{2^K} \sum_{i=1}^{K} A_i \, {\varepsilon_{\mathrm{t}}}^i \left\{ \left(1-\varepsilon_{\mathrm{t}}\right)^{\mathrm{d}-1} + \left(1-\varepsilon_{\mathrm{t}}\right)^{\left\lfloor \frac{N-K}{2} \right\rfloor} \left(\left(2-\varepsilon_{\mathrm{t}}\right)^{K-i} - 1 \right) \right\}, \tag{B.10}$$

with A_i is the number of weight-i codewords in the (N,K) code C and d the minimum distance of the $(N+1, K+1)$ code.

To conclude, we prove Lemma 5.

Proof. **(Lemma 5).**

We proceed as in the proof of Lemma 4. In particular, we consider the system of $\mathrm{w_R}(\tilde{e})$ equations constraining the bits positions that do not belong to WI.

$$\begin{aligned}
x_1 \cdot p_{1,r_1} \oplus \bigoplus_{j \in I \setminus \{WI \cup \{1\}\}} x_j \cdot p_{j,r_1} &= 0, \\
x_1 \cdot p_{1,r_1} \oplus \bigoplus_{j \in I \setminus \{WI \cup \{1\}\}} x_j \cdot p_{j,r_2} &= 0, \\
&\ldots \\
x_1 \cdot p_{1,r_{\mathrm{w_R}(\tilde{e})}} \oplus \bigoplus_{j \in I \setminus \{WI \cup \{1\}\}} x_j \cdot p_{j,r_{\mathrm{w_R}(\tilde{e})}} &= 0.
\end{aligned} \tag{B.11}$$

The number of free bit positions for $\mathrm{LSB}_1\left(Sup(\tilde{e})\right)$ is the maximum number of variables that can be assigned freely while satisfying the system of Eq. (B.11) and such that $x_1 = 1$. We rewrite thus the system of Eq. (B.11) as

$$\begin{aligned}
\bigoplus_{j \in I \setminus \{WI \cup \{1\}\}} x_j \cdot p_{j,r_1} &= p_{1,r_1}, \\
\bigoplus_{j \in I \setminus \{WI \cup \{1\}\}} x_j \cdot p_{j,r_2} &= p_{1,r_2}, \\
&\ldots \\
\bigoplus_{j \in I \setminus \{WI \cup \{1\}\}} x_j \cdot p_{j,r_{\mathrm{w_R}(\tilde{e})}} &= p_{1,r_{\mathrm{w_R}(\tilde{e})}}.
\end{aligned} \tag{B.12}$$

If the non-homogeneous system (B.12) has a solution (which may not be the case if the equations associated with parity checks $r_1, \ldots, r_{\mathrm{w_R}(\tilde{e})}$ require $x_1 = 0$), then the number of its solutions is the same as that of the homogeneous system (A.19). As a result, $|M|$ can be upper bounded by Eq. (A.11) as well. However, there exists no other lower bound as the trivial one (namely, 0) because the system (B.12) may have no solution. □

Appendix C

Residual Error Rate of the Berger Code Over the Binary Symmetric Channel

We compute the residual error rate of differential encoding using the Berger code over the BSC. Let C be a Berger code with K information bits (the Berger code requires $K+1$ to be a power of two). The Berger code is systematic; we denote by $Q = log_2(K+1)$ the number of redundant bits. Thus, we write for all $c \in C$, $c = (c_I \mid c_R)$ where $c_R \in \{0,1\}^Q$ is the Q-bit redundant vector of the codeword c computed from its information part $c_I \in \{0,1\}^K$. We partition C into $K+1$ subsets: $C = \bigcup_{i=0}^{K} C_i$ with $C_i = \{c \in C \mid \mathrm{w}(c_I) = i\}$. By definition of the Berger code, $c_R = B(K-i)$ for all $c \in C_i$ where $B(k)$, $k \in \{0, ..., K\}$ is the binary value of the integer k coded on Q bits. We denote by $t \in C$ the transition vector, and by $\tilde{t}$ the sampled transition vector: $\tilde{t} = t \oplus e$, with e the additive error vector. The residual error rate is given by

$$\varepsilon_{\mathrm{res,a}} = \sum_{\tilde{t} \in C \setminus \{t\}} P(\tilde{t}), \tag{C.1}$$

with $P(\tilde{t})$ the probability of occurrence for $\tilde{t}$. By conditioning on the code subset C_i to which the transition vector t belongs to, we get

$$\begin{aligned}
\varepsilon_{\mathrm{res,a}} &= \sum_{j=0}^{K} \sum_{\tilde{t} \in C_j \setminus \{t\}} P(\tilde{t}) \\
&= \sum_{i=0}^{K} \sum_{j=0}^{K} \sum_{\tilde{t} \in C_j \setminus \{t\}} P(\tilde{t} \mid t \in C_i)\ P(t \in C_i) \\
&= \sum_{i=0}^{K} \frac{\binom{K}{i}}{2^K} \sum_{j=0}^{K} \sum_{\tilde{t} \in C_j \setminus \{t\}} P(\tilde{t} \mid t \in C_i),
\end{aligned} \tag{C.2}$$

where we have used the codeword equiprobability assumption in the last equality. We focus now on the sum $\sum_{\tilde{t} \in C_j \setminus \{t\}} P(\tilde{t} \mid t \in C_i)$ and compute it in the

following lemma.

Lemma 6. *Denote by* $F(i,j,\varepsilon_a)$ *the sum* $\sum_{\tilde{t}\in C_j\setminus\{t\}} P\left(\tilde{t} \mid t\in C_i\right)$. *We have*

$$F(i,j,\varepsilon_a) = \varepsilon_a^{w(B(i)\oplus B(j))}\left(1-\varepsilon_a\right)^{Q-w(B(i)\oplus B(j))} \cdot \begin{cases} T_1 & \text{if } j>i,\\ T_2 & \text{if } j=i,\\ T_3 & \text{if } j<i,\end{cases}$$

where $Q = log_2(K+1)$ *and* T_1, T_2, T_3 *are defined respectively in Eqs. (C.3), (C.4), and (C.5):*

$$T_1 = \sum_{k=0}^{\min(i,K-j)} \binom{i}{k}\binom{K-i}{j-i+k} \cdot \varepsilon_a^{j-i+2k}\left(1-\varepsilon_a\right)^{K-(j-i+2k)}, \tag{C.3}$$

$$T_2 = \sum_{k=1}^{\min(i,K-i)} \binom{i}{k}\binom{K-i}{k} \cdot \varepsilon_a^{2k}\left(1-\varepsilon_a\right)^{K-2k}, \tag{C.4}$$

and

$$T_3 = \sum_{k=0}^{\min(j,K-i)} \binom{i}{i-j+k}\binom{K-i}{k} \cdot \varepsilon_a^{i-j+2k}\left(1-\varepsilon_a\right)^{K-(i-j+2k)}. \tag{C.5}$$

Proof. The redundant part of any $\tilde{t}\in C_j$ can equivalently be written as $\tilde{t}_R = B(j)\oplus\vec{1}$. As a result, the redundant part of any $\tilde{t}\in C_j$ differs from the redundant part of t in exactly $\mathrm{w}(B(i)\oplus B(j))$ bit positions. We write thus for any $\tilde{t}\in C_j$

$$P\left(\tilde{t} \mid t\in C_i\right) = P\left(\tilde{t}_I \mid t\in C_i\right) \cdot \varepsilon_{\mathrm{a}}^{\mathrm{w}(B(i)\oplus B(j))}\left(1-\varepsilon_{\mathrm{a}}\right)^{Q-\mathrm{w}(B(i)\oplus B(j))},$$

so that

$$\sum_{\tilde{t}\in C_j\setminus\{t\}} P\left(\tilde{t} \mid t\in C_i\right) = \varepsilon_{\mathrm{a}}^{\mathrm{w}(B(i)\oplus B(j))}\left(1-\varepsilon_{\mathrm{a}}\right)^{Q-(\mathrm{w}(B(i)\oplus B(j)))} \cdot \sum_{\tilde{t}\in C_j\setminus\{t\}} P\left(\tilde{t}_I \mid t\in C_i\right).$$

Now, we work the sum

$$\sum_{\tilde{t}\in C_j\setminus\{t\}} P\left(\tilde{t}_I \mid t\in C_i\right).$$

By definition of C_i and C_j, we get

$$\sum_{\tilde{t}\in C_j\setminus\{t\}} P\left(\tilde{t}_I \mid t\in C_i\right) = \sum_{\substack{\tilde{t}_I\in\{0,1\}^K\setminus\{t_I\}\\ \mathrm{w}(\tilde{t}_I)=j}} P\left(\tilde{t}_I \mid \mathrm{w}(t_I)=i\right).$$

Since $\tilde{t}_I = t_I\oplus e_I$, we rewrite the last sum as

$$\sum_{\substack{\tilde{t}_I\in\{0,1\}^K\setminus\{t_I\}\\ \mathrm{w}(\tilde{t}_I)=j}} \varepsilon_{\mathrm{a}}^{\mathrm{w}(\tilde{t}_I\oplus t_I)}\cdot\left(1-\varepsilon_{\mathrm{a}}\right)^{K-\mathrm{w}(\tilde{t}_I\oplus t_I)}.$$

We consider first the case $i = j$. Let $u \in \{0,1\}^K \setminus \{t_I\}$ be a vector such that $\mathrm{w}(u) = i$ and k be the number of information bit positions p such that $t_I^p = 1$ and $u^p = 0$. Clearly, we have $k \leq i$. Because $\mathrm{w}(u) = \mathrm{w}(t_I) = i$, there exists k other information bit positions q such that $t_I^q = 0$ and $u^q = 1$. As a result, $k \leq K - i$. For a given value of k, there are exactly $\binom{i}{k} \cdot \binom{K-i}{k}$ vectors $\tilde{t}_I \neq t_I$ such that $\mathrm{w}(\tilde{t}_I \oplus t_I) = 2\,k$, with $1 \leq k \leq \min(i, K-i)$. As a result, we can write

$$\sum_{\tilde{t} \in C_i \setminus \{t\}} P\left(\tilde{t}_I \mid \mathrm{w}(t_I) = i\right) \;=\; \sum_{k=1}^{\min(i,K-i)} \binom{i}{k} \binom{K-i}{k} \cdot \varepsilon_{\mathrm{a}}^{2k} \left(1 - \varepsilon_{\mathrm{a}}\right)^{K-2k}.$$

By a similar reasoning, we obtain, for $i < j$,

$$\sum_{\tilde{t} \in C_j \setminus \{t\}} P\left(\tilde{t}_I \mid \mathrm{w}(t_I) = i\right) = \sum_{k=0}^{\min(i,K-j)} \binom{i}{k} \binom{K-i}{j-i+k} \cdot \varepsilon_{\mathrm{a}}^{j-i+2k} \left(1 - \varepsilon_{\mathrm{a}}\right)^{K-(j-i+2k)},$$

and, finally, for $i > j$,

$$\sum_{\tilde{t} \in C_j \setminus \{t\}} P\left(\tilde{t}_I \mid \mathrm{w}(t_I) = i\right) = \sum_{k=0}^{\min(j,K-i)} \binom{i}{i-j+k} \binom{K-i}{k} \cdot \varepsilon_{\mathrm{a}}^{i-j+2k} \left(1 - \varepsilon_{\mathrm{a}}\right)^{K-(i-j+2k)}.$$

□

Appendix D

Qualitative Comparison of Razor Flip-Flops and Codes

This appendix shows the complementarity of Razor flip-flops and soft SSC by comparing qualitatively their detection capabilities in the delay-voltage plan. That is, we express the residual and detected error probabilities of each checker as a function of the link supply voltage v_{ch} and cycle time T_{c}. To do so, we use the bit and word error rate models introduced in Chapter 3, which we need to express the metrics of interest.

Using the lumped capacitance wire model, the delay through a bit line can be expressed as

$$t_{\mathrm{p}} = \frac{C_{\mathrm{L}}}{k_{\mathrm{m}}} \cdot \frac{v_{\mathrm{ch}}}{\left(v_{\mathrm{ch}} - v_{\mathrm{th}}\right)^2},$$

with k_{m} the transistor transconductance, C_{L} the line capacitance, and v_{th} the device threshold voltage.

In order to model the variability of t_{p}, we describe the ratio $C_{\mathrm{L}}/k_{\mathrm{m}}$, which we denote by α, by a Gamma random variable with parameters a and b: $\alpha \sim \Gamma(a, b)$. The two parameters a and b are determined from the mean and standard deviation of t_{p}, as expressed in Eq. (3.2). We use a Gamma distribution because it models more accurately the line delay than a Gaussian distribution. For example, a Gamma distribution takes only positive values, differently to a Gaussian.

As defined in Eq. (6.1), a timing error occurs when t_{p} exceeds the sampling period T_{c}. By definition of α, $t_{\mathrm{p}} > T_{\mathrm{c}}$ if and only if

$$\alpha > T_{\mathrm{c}} \frac{\left(v_{\mathrm{ch}} - v_{\mathrm{th}}\right)^2}{v_{\mathrm{ch}}}.$$

In order to avoid heavy notations, we denote the deterministic factor $\left(v_{\mathrm{ch}} - v_{\mathrm{th}}\right)^2 / v_{\mathrm{ch}}$ by Δ_{v}. We write thus

$$\varepsilon_{\mathrm{t}} = P\left(t_{\mathrm{p}} > T_{\mathrm{c}}\right) = P\left(\alpha > T_{\mathrm{c}} \cdot \Delta_{\mathrm{v}}\right) = 1 - \mathrm{F}\left(T_{\mathrm{c}} \cdot \Delta_{\mathrm{v}} \mid a, b\right),$$

where $\mathrm{F}(\cdot \mid a, b)$ is the cumulative distribution function of α:

$$\mathrm{F}(x \mid a, b) = \frac{1}{b^a \gamma(a)} \int_0^x t^{a-1} e^{-\frac{t}{b}} dt,$$

with $\gamma(\cdot)$ the Gamma function.

We model errors occurring on different bit lines as *independent and identically distributed*. For example, when the voltage is lowered, the probability of a timing error increases identically on all bit lines. Yet, the distributions remain independent, modelling the independence of the noise sources affecting them. Henceforth, the probability that at least one timing error occurs in a N-bit word is

$$\varepsilon_{\mathrm{w}} = 1 - (1 - \varepsilon_{\mathrm{t}})^N . \tag{D.1}$$

Now, we express the residual and reported error probability of Razor flip-flops, and derive first the residual error probabilty. Eq. (6.2) characterises residual bit errors. Accordingly, the residual bit error probability is

$$\begin{aligned} \varepsilon_{\mathrm{b,res}}^{\mathrm{SL}} &= P\left(t_{\mathrm{p}} \notin [T_{\mathrm{d}}; T_{\mathrm{c}} + T_{\mathrm{d}}]\right) \\ &= \mathrm{F}\left(T_{\mathrm{d}} \cdot \Delta_{\mathrm{v}} \mid a, b\right) + 1 - \mathrm{F}\left((T_{\mathrm{c}} + T_{\mathrm{d}}) \cdot \Delta_{\mathrm{v}} \mid a, b\right). \end{aligned} \tag{D.2}$$

We obtain the residual word error probability, denoted $\varepsilon_{\mathrm{res}}^{\mathrm{SL}}$, by observing that Razor flip-flops process each data bit individually: a residual word error occurs as soon as at least one bit line suffers a residual error. That is,

$$\varepsilon_{\mathrm{res}}^{\mathrm{SL}} = 1 - \left(1 - \varepsilon_{\mathrm{b,res}}^{\mathrm{SL}}\right)^N . \tag{D.3}$$

In addition, we are interested in the probability of an error to be reported to the controller. A bit error is reported by a Razor flip-flop either in case of a short-path error—i.e., whenever $t_{\mathrm{p}} \leq T_{\mathrm{d}}$—or in case of a detected timing error—i.e., whenever $T_{\mathrm{c}} \leq t_{\mathrm{p}} \leq T_{\mathrm{c}} + T_{\mathrm{d}}$. Let $\varepsilon_{\mathrm{b,rep}}^{\mathrm{SL}}$ be the probability of a bit error to be reported by a Razor flip-flop. We obtain

$$\varepsilon_{\mathrm{b,rep}}^{\mathrm{SL}} = \mathrm{F}\left(T_{\mathrm{d}} \cdot \Delta_{\mathrm{v}} \mid a, b\right) + \mathrm{F}\left((T_{\mathrm{c}} + T_{\mathrm{d}}) \cdot \Delta_{\mathrm{v}} \mid a, b\right) - \mathrm{F}\left(T_{\mathrm{c}} \cdot \Delta_{\mathrm{v}} \mid a, b\right).$$

Since the error signal reported to the controller is obtained by *OR*-ing the error signals of each individual Razor flip-flop. It is thus given by

$$\varepsilon_{\mathrm{rep}}^{\mathrm{SL}} = 1 - \left(1 - \varepsilon_{\mathrm{b,rep}}^{\mathrm{SL}}\right)^N . \tag{D.4}$$

We derive now the reported error probability for soft SSC. No error is reported to the controller if and only if no error occurs or an error occurs and is undetected. The probability of an error to be reported is thus

$$\varepsilon_{\mathrm{rep}}^{\mathrm{SSC}} = \varepsilon_{\mathrm{w}} - \varepsilon_{\mathrm{res}}^{\mathrm{SSC}}. \tag{D.5}$$

The word error rate ε_{w} is obtained from Eq. (D.1), while the residual error rate $\varepsilon_{\mathrm{res}}^{\mathrm{SSC}}$ can be approximated by the expression given in Eq. (5.14).

For illustration, we describe a 0.13μm technology as follows.

- The nominal operating points are $V_{\mathrm{dd}} = 1.2\mathrm{V}$ and $T_{\mathrm{c}} = 2\mathrm{ns}$.
- $\mu_{t_{\mathrm{p}}} = 1\mathrm{ns}$, $\sigma_{t_{\mathrm{p}}} = 0.1\mathrm{ns}$ under nominal conditions.
- $v_{\mathrm{th}} = 0.2\mathrm{V}$.

Furthermore, $T_{\mathrm{d}} = 0.3\,T_{\mathrm{c}}$, as it has been reported for busses [Kaul *et al.*, 2005]. According to the values given to $\mu_{t_{\mathrm{p}}}$ and $\sigma_{t_{\mathrm{p}}}$ and under the nominal conditions, the timing error probability per bit is $\varepsilon_{\mathrm{t}} \approxeq 10^{-15}$.

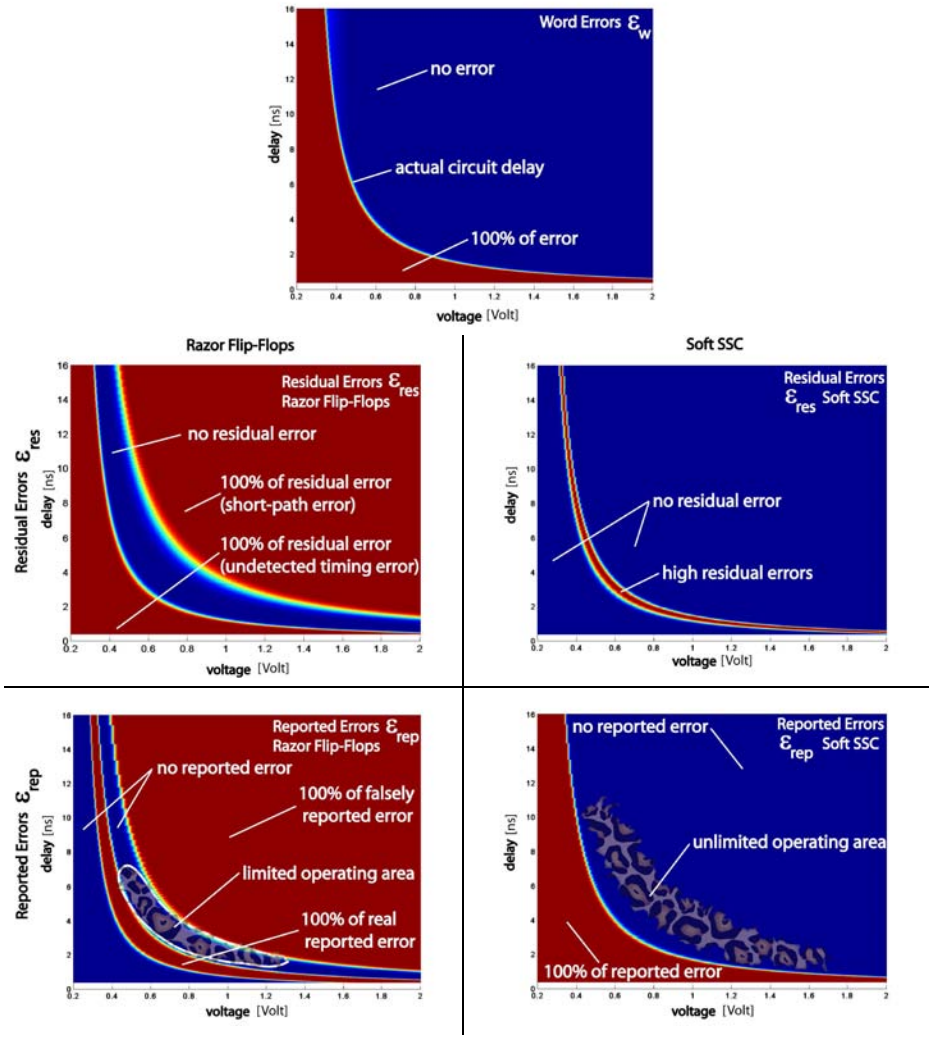

Figure D.1: Word error rate ε_{w} (top), residual error rate $\varepsilon_{\mathsf{res}}$ (middle) and reported error rate $\varepsilon_{\mathsf{rep}}$ (bottom) of Razor flip-flops (left) and alternating-phase code (right) with the 8-bit CRC generated by the polynome x^8+x^2+x+1 for 32-bit information.

We have plotted in Fig. D.1 the word error rate (i.e., Eq. (D.1)), the residual error rate (i.e., Eqs. (D.3) and (5.14)), and the reported error rate (i.e., Eqs. (D.4) and (D.5)) of shadow latches and a soft SSC in the voltage-delay plane. The top graph of Fig. D.1 depicts the word error rate. Areas of zero and 100% error rate are clearly separated. The middle left graph of Fig. D.1 shows that shadow latches are completely unreliable under a large error rate. On the contrary, the soft SSC checker operates reliably except in a tiny stripe where the error rate is moderate (top right of Fig. D.1). Next, by comparing the two bottom graphs of Fig. D.1, we observe that (i) soft SSC provide the controller with a consistent error landscape—i.e., no error is reported under conservative operating points and errors are reported under over-aggressive operation—and (ii) Razor flip-flops report falsely errors under conservative operating points (due to short-path errors), while they do not report any error under over-aggressive operation. As a result, Razor flip-flops require worst-case assumptions about the error rate to make sure they are kept in the constrained but safe operating range where timing errors can be reliably corrected and short-path errors are avoided. There is no such requirement for soft SSC: operation in the whole voltage-delay plane is possible. Reliability is ensured by limiting the probability of visiting unsafe operating points, which constrains the operating point control policy.

Bibliography

[Albassam *et al.*, 1991] Sulaiman Albassam, Bella Bose, and Ramarathnam Venkatesan. Burst and unidirectional error detecting codes. *Proceedings of the IEEE International Symposium on Information Theory*, page 142, June 1991.

[Arnold, 1990] Allen Arnold. *Probability, Statistics and Queuing Theory with Computer Science Applications*. Academic Press, San Diego, Calif., 1990.

[Austin *et al.*, 2004] Todd Austin, David Blaauw, Trevor Mudge, and Krisztián Flautner. Making typical silicon matter with Razor. *Computer*, 37(3):57–65, March 2004.

[Austin *et al.*, 2005] Todd Austin, Valeria Bertacco, David Blaauw, and Trevor Mudge. Opportunities and challenges for better than worst-case design. In *Proceedings of the Asia and South Pacific Design Automation Conference*, January 2005.

[Baicheva *et al.*, 1998] Tsonka Baicheva, Stefan Dodunekov, and Peter Kazakov. On the cyclic redundancy-check codes with 8-bits redundancy. *Computer Communications*, 21(11):1030–33, August 1998.

[Bainbridge *et al.*, 2003] John Bainbridge, William B. Toms, David Edwards, and Steve Furber. Delay-insensitive, point-to-point interconnect using m-of-n codes. In *Proceedings of the 9th International Symposium on Asynchronous Circuits and Systems*, pages 132–40, Vancouver, May 2003.

[Berger, 1961] Jay M. Berger. A note on error detecting codes for asymmetric channels. *Information and Control*, 4(1):68–71, March 1961.

[Bertozzi *et al.*, 2002] Davide Bertozzi, Luca Benini, and Giovanni De Micheli. Low power error resilient encoding for on-chip data buses. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pages 102–9, Paris, March 2002.

[Blunno *et al.*, 2004] Ivan Blunno, Jordi Cortadella, Alex Kondratyev, Luciano Lavagno, Kelvin Lwin, and Christos P. Sotiriou. Handshake protocols for de-synchronization. In *Proceedings of the 10th International Symposium on Asynchronous Circuits and Systems*, pages 149–158, 2004.

[Borkar *et al.*, 2004] Shekhar Borkar, Tanay Karnik, and Vivek De. Design and reliability challenges in nanometer technologies. In *Proceedings of the 41st Design Automation Conference*, page 75, San Diego, Calif., June 2004.

[Borkar, 2005] Shekhar Borkar. Designing reliable systems from unreliable components: The challenges of transistors variability and degradation. *IEEE Micro*, 25(6):10–16, November–December 2005.

[Bose and Lin, 1985] Bella Bose and Der Jei Lin. Systematic unidirectional error-detecting codes. *IEEE Transactions on Computers*, C-34(11):1026–32, November 1985.

[Bose and Rao, 1982] Bella Bose and Thammavaram Rao. Theory of unidirectional error correcting/detecting codes. *IEEE Transactions on Computers*, C-31(6):521–30, June 1982.

[Bose, 1991] Bella Bose. On unordered codes. *IEEE Transactions on Computers*, C-40(2):125–31, February 1991.

[Brooks and Martonosi, 2001] David Brooks and Margaret Martonosi. Dynamic thermal management for high-performance microprocessors. In *Proceedings of the 7th International Symposium on High-Performance Computer Architecture*, pages 171–182, January 2001.

[Chapiro, 1984] Daniel M. Chapiro. *Globally-Asynchronous Locally Synchronous Systems*. Ph.D. thesis, Stanford University, Stanford, Calif., 1984.

[Cortadella *et al.*, 2004] Jordi Cortadella, Alex Kondratyev, Luciano Lavagno, and Christos P. Sotiriou. Coping with the variability of combinational logic delays. In *Proceedings of the International Conference on Computer Aided Design*, pages 505–08, San Jose, Calif., October 2004.

[Dally and Poulton, 1998] William J. Dally and John W. Poulton. *Digital Systems Engineering*. Cambridge University Press, Cambridge, 1998.

[Das *et al.*, 2006] Shidhartha Das, David Roberts, Seokwoo Lee, Sanjay Pant, Blaauw David, Todd Austin, Krisztián Flautner, and Trevor Mudge. A self-tuning DVS processor using delay-error detection and correction. *IEEE Journal of Solid-State Circuits*, 41(4):792–804, April 2006.

[Dean *et al.*, 1991] Mark E. Dean, Ted E. Williams, and David L. Dill. Efficient self-timing with level-encoded two-phase dual-rail (LEDR). In *Proceedings of the 1991 University of California/Santa Cruz Conference on Advanced Research in VLSI*, pages 55–70. MIT Press, March 1991.

[Flautner *et al.*, 2001] Krisztián Flautner, Steve Reinhardt, and Trevor Mudge. Automatic performance setting for dynamic voltage scaling. In *Proceedings of the 7th Conference on Mobile Computing and Networking*, pages 260–71, Rome, July 2001.

[Freiman, 1962] C. V. Freiman. Optimal error detecting codes for asymmetric binary channels. *Information and Control*, 5(1):64–71, March 1962.

[Gaujal *et al.*, 2005] Bruno Gaujal, Nicolas Navet, and Cormac Walsch. Shortest-path algorithms for real-time scheduling of FIFO tasks with minimal energy use. *ACM Transactions on Embedded Computing Systems (TECS)*, 4(4):907–33, November 2005.

[Ghosh *et al.*, 2006] Swaroop Ghosh, Saibal Mukhopadhyay, Keejong Kim, and Kaushik Roy. Self-calibration technique for reduction of hold failures in low-power nano-scaled SRAM. In *Proceedings of the 43rd Design Automation Conference*, pages 971–976, San Francisco, Calif., July 2006.

[Gorshe and Bose, 1996] Steven S. Gorshe and Bella Bose. A self-checking ALU design with efficient codes. In *14th IEEE VLSI Test Symposium (VTS'96)*, pages 157–161, Princeton, New Jersey., April 1996.

[Gutnik and Chandrakasan, 1997] Vadim Gutnik and Anantha P. Chandrakasan. Embedded power supply for low-power DSP. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, VLSI-5(4):425–35, December 1997.

[Han *et al.*, 2005] Jie Han, Jianbo Gao, Yan Qi, Pieter Jonker, and José Fortes. Toward hardware-redundant, fault-tolerant logic for nanoelectronics. *IEEE Design and Test of Computers*, 22(4):328–339, July–August 2005.

[Hegde and Shanbhag, 2000] Rajamohana Hegde and Naresh R. Shanbhag. Toward achieving energy efficiency in presence of deep submicron noise. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, VLSI-8(4):379–91, August 2000.

[Jien-Chung *et al.*, 1992] Lo Jien-Chung, Thanawastien Suchai, and Michael Nicolaidis. An SFS berger check prediction ALU and its application to self-checking processor designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(4):525–540, April 1992.

[Karl *et al.*, 2006] Eric Karl, David Blaauw, Dennis Sylvester, and Trevor Mudge. Reliability modeling and management in dynamic microprocessor-based systems. In *Proceedings of the 43rd Design Automation Conference*, pages 1057–1060, San Francisco, Calif., July 2006.

[Kaul *et al.*, 2005] Himanshu Kaul, Dennis Sylvester, David Blaauw, Trevor Mudge, and Todd Austin. DVS for on-chip bus designs based on timing error correction. In *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition*, pages 80–85, Munich, March 2005.

[Kim *et al.*, 2005] Chris H. Kim, Steven Hsu, Ram Krishnamurthy, Shekhar Borkar, and Kaushik Roy. Self calibrating circuit design for variation tolerant VLSI systems. In *Proceedings of the 11th IEEE International On-Line Testing Symposium*, pages 100–5, Saint Raphaël, France, July 2005.

[Koopman and Chakravarty, 2004] Philip Koopman and Tridib Chakravarty. Cyclic redundancy code (crc) polynomial selection for embedded networks. In *2004 International Conference on Dependable Systems and Networks (DSN 2004)*, pages 145–154, Florence, Italy, June 2004.

[Lala, 2001] Parag K. Lala, editor. *Self-checking and fault-tolerant digital design*. Morgan Kaufmann Publishers Inc., San Francisco, Calif., 2001.

[Li *et al.*, 2006] Hai Li, Yiran Chen, Kaushik Roy, and Cheng-Kok Koh. SAVS: a self-adaptive variable supply-voltage technique for process-tolerant and power-efficient multi-issue superscalar processor design. In *Proceedings of the Asia and South Pacific Design Automation Conference*, pages 158–163, January 2006.

[MacWilliams and Sloane, 1977] Florence Jessie MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*, volume 19 of *Mathematical Library*. North-Holland, Amsterdam, 1977.

[Martin *et al.*, 2002] Steven M. Martin, Krisztian Flautner, Trevor Mudge, and David Blaauw. Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In *Proceedings of the International Conference on Computer Aided Design*, pages 721–25, San Jose, Calif., November 2002.

[McNairy and Soltis, 2003] Cameron McNairy and Don Soltis. Itanium 2 processor microarchitecture. *IEEE Micro*, 23(2):44–55, March–April 2003.

[Mitra *et al.*, 2005] Subhasish Mitra, Norbert Seifert, Ming Zhang, Quan Shi, and Kee S. Kim. Robust system design with built-in soft-error resilience. *IEEE Computer*, 38(2):43–52, 2005.

[Murali *et al.*, 2005] Srinivasan Murali, Theo Theocharides, Narayanan Vijaykrishnan, Mary Jane Irwin, Luca Benini, and Giovanni De Micheli. Analysis of error recovery schemes for networks on chips. *IEEE Design and Test of Computers*, 22(5):434–442, September–October 2005.

[Nicolaidis, 1999] Michael Nicolaidis. Time-redundancy based soft-errors tolerance to rescue nanometer technology. In *17th IEEE VLSI Test Symposium*, pages 86–94, April 1999.

[Nicolaidis, 2003] Michael Nicolaidis. Carry checking/parity prediction adders and ALUs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, VLSI-11(1):121–128, February 2003.

[Nicolaidis, 2005] Michael Nicolaidis. Design for soft error mitigation. *IEEE Transactions on Device and Materials Reliability*, 5(3):405–18, September 2005.

[Nowka *et al.*, 2002] Kevin J. Nowka, Gary D. Carpenter, Eric W. MacDonald, Hung C. Ngo, Bishop C. Brock, Koji I. Ishii, Tuyet Y. Nguyen, and Jeffrey L. Burns. A 32-bit PowerPC System-on-a-Chip with support for dynamic voltage scaling and dynamic frequency scaling. *IEEE Journal of Solid-State Circuits*, 37(11):1441–47, November 2002.

[Proakis, 2000] John G. Proakis. *Digital Communications*. McGraw-Hill, New York, 2000.

[Rabaey *et al.*, 2003] Jan M. Rabaey, Anantha Chandrakasan, and Borivoje Nikolić. *Digital Integrated Circuits*. Prentice Hall, Upper Saddle River, N.J., second edition, 2003.

[Roberts *et al.*, 2005] David Roberts, Todd Austin, David Blauww, and Trevor Mudge. Error analysis for the support of robust voltage scaling. In *Proceedings of 6th IEEE International Symposium on Quality Electronic Design*, pages 65–70, March 2005.

[Rossi *et al.*, 2005a] Daniele Rossi, André K. Nieuwland, Atul Katoch, and Cecilia Metra. Exploiting ECC redundancy to minimize crosstalk impact. *IEEE Design and Test of Computers*, 22(1):57–70, January–February 2005.

[Rossi *et al.*, 2005b] Daniele Rossi, André K. Nieuwland, Atul Katoch, and Cecilia Metra. New ecc for crosstalk impact minimization. *IEEE Design and Test of Computers*, 22(4):340–348, July–August 2005.

[Saggese *et al.*, 2005a] Giacinto Paolo Saggese, Anoop Vetteth, Zbigniew Kalbarczyk, and Ravishankar K. Iyer. Microprocessor sensitivity to failures: Control vs execution and combinational vs sequential logic. In *2005 International Conference on Dependable Systems and Networks (DSN 2005)*, pages 760–769, Yokohama, Japan, July 2005.

[Saggese *et al.*, 2005b] Giacinto Paolo Saggese, Nicholas J. Wang, Zbigniew Kalbarczyk, Sanjay J. Patel, and Ravishankar K. Iyer. An experimental study of soft errors in microprocessors. *IEEE Micro*, 25(6):30–39, November–December 2005.

[Sakiyama *et al.*, 1999] Shiro Sakiyama, Jun Kajiwara, Masayoshi Kinoshita, Katsuji Satomi, Katsuhiro Ohtani, and Akira Matsuzawa. An on-chip high-efficiency and low-noise DC/DC converter using divided switches with current control technique. In *IEEE International Solid-State Circuits Conference, Digest of Technical Paper*, pages 156–57, San Francisco, Calif., February 1999.

[Shanbhag, 2002] Naresh R. Shanbhag. Reliable and energy-efficient digital signal processing. In *Proceedings of the 39th Design Automation Conference*, pages 830–835, June 2002.

[Shang *et al.*, 2003] Li Shang, Li-Shiuan Peh, and Niraj K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *Proceedings of the 9th International Symposium on High-Performance Computer Architecture*, pages 91–102, Anaheim, Calif., February 2003.

[Shim *et al.*, 2004] Byonghyo Shim, Srinivasa R. Sridhara, and Naresh R. Shanbhag. Reliable low-power digital signal processing via reduced precision redundancy. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 12(5):497–510, May 2004.

[Sinha and Chandrakasan, 2001] Amit Sinha and Anantha P. Chandrakasan. Dynamic voltage scheduling using adaptive filtering of workload traces. In *Proceedings of the 14th International Conference on VLSI Design*, pages 221–226, Jan 2001.

[Sokolov *et al.*, 2005] Danil Sokolov, Julian Murphy, Bystrov Alexander, and Alex Yakovlev. Design and analysis of dual-rail circuits for security applications. *IEEE Transactions on Computers*, 54(4):449–60, April 2005.

[Stratakos, 1998] Anthony J. Stratakos. *High-Efficiency Low-Voltage DC-DC Conversion for Portable Applications*. Ph.D. thesis, University of California, Berkeley, Calif., 1998.

[Thaker *et al.*, 2005] Darshan D. Thaker, Francois Impens, Isaac L. Chuang, Rajeevan Amirtharajah, and Frederic T. Chong. Recursive TMR: Scaling fault tolerance in the nanoscale era. *IEEE Design and Test of Computers*, 22(4):298–305, July–August 2005.

[Toprak and Leblebici, 2005] Zeynep Toprak and Yusuf Leblebici. A Low-Power Adaptive Bias/Clock Generator for Fine-Grained Voltage and Frequency Scaling in Multi-Core Systems. *WSEAS TRANSACTIONS on SYSTEMS*, 4(12):2390–2397, 2005.

[Varshavsky, 1990] Victor I. Varshavsky, editor. *Self-Timed Control of Concurrent Processes*. Kluwer Academic, Dordrecht, The Netherlands, 1990.

[Verhoeff, 1988] Tom Verhoeff. Delay-insensitive codes—an overview. *Distributed Computing*, 3(1):1–8, January 1988.

[Victor and Keutzer, 2001] Bret Victor and Kurt Keutzer. Bus encoding to prevent crosstalk delay. In *Proceedings of the International Conference on Computer Aided Design*, pages 57–63, San Jose, Calif., November 2001.

[Von Neumann, 1956] John Von Neumann. Probabilistics logics and the synthesis of reliable organisms from unreliable components. *Automata Studies, Eds. C. E. Shannon et J. McCarthy, Princeton University Press*, pages 43–98, 1956.

[Walrand and Varaiya, 2000] Jean Walrand and Pravin Varaiya. *High-Performance Communication Networks*. Morgan Kaufmann, San Mateo, Calif., second edition, 2000.

[Weste and Eshraghian, 1993] Neil H. E. Weste and Kamran Eshraghian. *Principles of CMOS VLSI Design*. VLSI System Series. Addison-Wesley, Reading, Mass., second edition, 1993.

[Worm *et al.*, 2002] Frédéric Worm, Paolo Ienne, Patrick Thiran, and Giovanni De Micheli. An adaptive low-power transmission scheme for on-chip networks. In *Proceedings of the 15th International Symposium on System Synthesis*, pages 92–100, Kyoto, October 2002.

[Worm *et al.*, 2005] Frédéric Worm, Paolo Ienne, Patrick Thiran, and Giovanni De Micheli. A robust self-calibrating transmission scheme for on-chip networks. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, VLSI-13(1):126–39, January 2005.

[Zhang *et al.*, 2000] Hui Zhang, Varghese George, and Jan M. Rabaey. Low-swing on-chip signaling techniques: Effectiveness and robustness. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, VLSI-8(3):264–72, June 2000.

[Zhang *et al.*, 2002] Yan Zhang, John Lach, Kevin Skadron, and Mircea R. Stan. Odd/even bus invert with two-phase transfer for buses with coupling. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 80–83, Monterey, Calif., August 2002.

Curriculum Vitae

Frédéric Worm was born in 1977 in Geneva, Switzerland. He obtained a master in Communication Systems in 2001 from Ecole Polytechnique Fédérale de Lausanne (EPFL). He spent the last two years of his master at Eurécom, Sophia-Antipolis, France and obtained a *Diplôme d'Etudes Approndies* in Networks and Distributed Systems from Université de Nice et Sophia-Antipolis in 2001. The same year, he joined the Processor Architecture Laboratory at EPFL and started working on his PhD thesis under the joint supervision of Professor Paolo Ienne and Professor Patrick Thiran.

His research interests include bus encoding techniques, VLSI design robust to electrical parameter variations, and Network-on-chip architectures.

Publications

- Frédéric Worm, Patrick Thiran, and Paolo Ienne. Designing robust checkers in the presence of massive timing errors. In Proceedings of 12th IEEE International On-Line Testing Symposium, pages 281-86, Lake of Como, Italy, July 2006.
- Frédéric Worm, Patrick Thiran, Giovanni De Micheli, and Paolo Ienne. Self-calibrating Networks-on-Chip. In Proceedings of the IEEE International Symposium on Circuits and Systems, pages 2361-64, Kobe, Japan, May 2005.
- Frédéric Worm, Patrick Thiran, and Paolo Ienne. A unified coding framework for delay-insensitivity. In Proceedings of the 11th International Symposium on Asynchronous Circuits and Systems, New York, March 2005.
- Frédéric Worm, Paolo Ienne, Patrick Thiran, and Giovanni De Micheli. A robust self-calibrating transmission scheme for on-chip networks. IEEE Transactions on Very Large Scale Integration (VLSI), 13(1), January 2005.
- Frédéric Worm, Paolo Ienne, Patrick Thiran, and Giovanni De Micheli. On-Chip self-calibrating communication techniques robust to electrical parameter variations. IEEE Design and Test of Computers, 21(6), November-December 2004.
- Frédéric Worm, Paolo Ienne, and Patrick Thiran. Soft self-synchronising codes for self-calibrating communication. In Proceedings of the International Conference on Computer Aided Design, San Jose, Calif., November 2004.
- Frédéric Worm, Paolo Ienne, Patrick Thiran, and Giovanni De Micheli. An adaptive low-power transmission scheme for on-chip networks. In Proceedings of the 15th International Symposium on System Synthesis, Kyoto, October 2002.
- Patrick Thiran, Jean-Yves Le Boudec, and Frédéric Worm, Network calculus applied to optimal smoothing. In Proceedings of the 20th IEEE Infocom Conference, Anchorage, April 2001.