Automatic Generation of Inexact Digital Circuits by Gate-level Pruning

Jeremy Schlachter, Vincent Camus, Christian Enz
Integrated Circuits Laboratory (ICLAB)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
Lausanne, Switzerland
jeremy.schlachter@epfl.ch

Krishna V. Palem
Department of Computer Science,
Rice University
Houston, TX 77005, USA

***Abstract*—Inexact or approximate circuits show great ability to reduce power consumption at the cost of occasional errors in comparison to their conventional counterparts. Even though the benefits of such circuits have been proven for many applications, they are not wide spread owing to the absence of a clear design methodology and the required CAD tools. In this regard, this paper presents a methodology to automatically generate inexact circuits starting from a conventional design by adding only one small step in the digital design flow. Further, this paper also demonstrates that achieving pruning at gate-level can lead to substantial savings in terms of power consumption, critical path delay and silicon area. An order of magnitude area and power savings is demonstrated for a 64-bit gate-level pruned high-speed adder for a 10 % relative error magnitude.**

I. Introduction

In the past three decades, technology scaling has been leading the improvements of digital circuits. The well-known Moore's law has enabled the number of transistors in a dense integrated circuit to be multiplied every two years by a factor of two. However, it won't be possible to reduce transistor sizes indefinitely due to fundamental physical limitations. In fact, those limits have already started to appear as the complexity of deeply scaled processes is continuously increasing, making it more and more challenging to reach new technology nodes. In addition, process variations have increased drastically with scaling, forcing to design circuits with large safety margins, at the cost of performance and power efficiency.

Knowing the predicted end of Moore's law, researchers started studying how to continue improving circuits by other means than scaling. In this perspective, inexact or approximate circuits have been gaining a lot of interest [1]. For the past few years, circuits wherein erroneous computations can occasionally happen appeared increasingly in the literature. In particular, it has been proven in [2] that reducing or deleting the safety margins by means of voltage over-scaling, can lead to a drastic reduction of the power consumption, at the cost of a low overhead error monitoring and correction system.

The approach adopted in this paper is slightly different and was first introduced in [3]: for some applications exactness can be traded against power, area and delay savings without any error monitoring and correction circuitry overhead. In some cases, such as video or audio processing where the final output is interpreted by human senses, a small amount of error may not even affect the end-user.

A lot of works published in the literature are exploiting the trade-off between energy consumed and accuracy. In the early stages of inexact computing, VLSI systems were taking advantage of the quadratic energy savings allowed by voltage over-scaling [2]. Later on, many works have been focusing on complexity reduction. Gupta *et. al* presented various approximations of the full adder cell [4] allowing power savings of up to 60 %. In [5], the authors present an under-designed multiplier achieving up to 45 % energy savings. This complexity reduction served as a basis to create inexact arithmetic circuits such as adders [4], [6] and multipliers [7]. However, these computational blocks have been tested empirically, and all present different amounts of errors. Hence, it is not clear from a designer point of view, how these approximate adder and multiplier cells should be instantiated in a circuit. This problem was initially addressed by Lingamneni *et. al* with the *Probabilistic pruning* approach [8], where some adders' propagate and generate blocks were simply removed, leading to substantial area, power and delay savings. The amount of error introduced by those circuits was simply proportional to the number of pruned circuit elements. In [8], the probabilistic pruning process was mostly done by hand, whereas this paper proposes a systematic pruning methodology resulting in the following improvements:

- A simple method to automatically generate inexact versions of a conventional exact circuit with increasing amount of error, adding only one design step in the CAD digital flow.
- Significant improvements compared to former techniques are obtained by achieving pruning at gate-level. For example, for a 64-bit high speed adder with 10 % relative error magnitude, this finer granularity allows to reduce power consumption, critical path delay and silicon area by a factor 7.82, 1.07 and 21, respectively.
- The gate-level pruning can be applied to any arithmetic unit, independently of the architecture.

This paper is organized as follows: section II presents the methodology and the tools developed to automatically prune circuits at the gate level, and section III shows the possible savings allowed by this technique on various adders and multipliers.

This work is funded by the Swiss National Science Foundation under grant number 200021_144418.

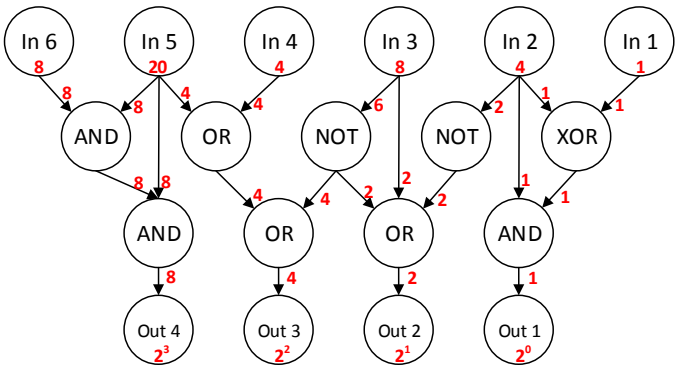


Fig. 1. Directed acyclic graph representation of a gate level netlist and the associated significance attribution

II. Automatic Circuit Pruning

Probabilistic pruning is a design technique that consists of removing circuits blocks and their associated wires in order to trade exactness of computation against power, area and delay savings without any overhead. The amount of pruning is dictated by the application's error tolerance. A formal definition of probabilistic pruning, as well as the proof of concept, have already been addressed in [8]. The following paragraphs only expose the key points necessary to build an automatic gate-level pruning tool using existing CAD software.

A. Significance and activity based pruning

A circuit can be represented by a directed acyclic graph as depicted in Fig. 1, where the nodes are components such as gates, and whose edges are wires. The decision to prune a node is based on two criteria: the significance, which is a structural parameter, and the activity or toggle count. The nodes with the lowest significance-activity product (SAP) are pruned first. By doing so, the error magnitude grows with the amount of pruning. Alternatively, depending on the application's requirement, the designer may choose to prune nodes according to the activity only, in order to minimize the error rate.

The activity of each wire is extracted from the .SAIF file (Switching Activity Interchange Format) obtained through gate-level hardware simulations. This file contains the toggle count (TC) of each wire, as well as the time spent at the logic levels 0 and 1 (T0 and T1) respectively. Note that to get an accurate activity estimation, the system should be simulated with an input stimulus representative of the *real operation* of the circuit.

The significance of each primary output is set by the designer depending on the application's requirement. The experiments performed on adders and multipliers in this paper assume a weighted significance attribution, where each bit position has a significance 2 times higher than the previous when moving from the LSB to the MSB. Reverse topological graph traversal is then performed to compute each nodes' significances as follows:

$$\sigma_i = \sum \sigma_{desc(i)} \tag{1}$$

where σ_i is the significance of the node i and $\sigma_{desc(i)}$ is the significance of the direct descendants of node i. An example of weighted significance attribution is shown in Fig. 1.

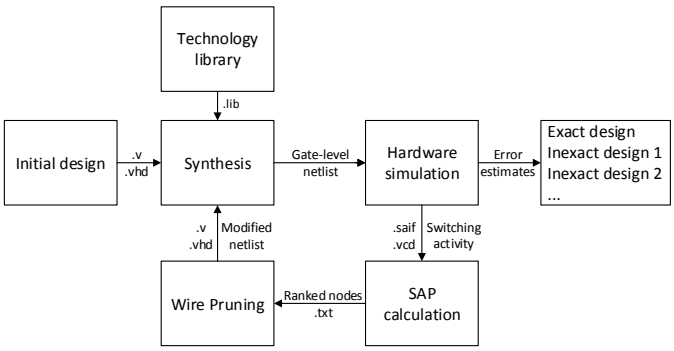


Fig. 2. Functional diagram of the automatic pruning tool

B. Node pruning

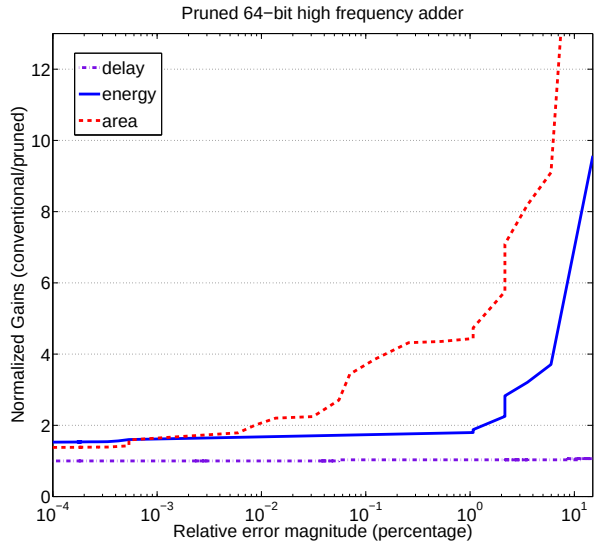
Once the nodes are ranked according to their significance-activity product, the gate-level netlist is modified in order to remove *unessential* nodes from the design. For the sake of simplicity, and in order to maximize the use of the existing EDA tools, the probabilistic pruner does not literally remove the gates form the netlist, but it disconnects the corresponding wires. Gates whose outputs are unconnected will automatically be removed by the synthesis tool. However, leaving gate inputs unconnected would fail the re-synthesis of the design. For this reason, and in order to minimize the error, those inputs are set to 0 if they statistically spend most of the time at 0. Otherwise they are connected to 1. The synthesis of the modified netlist therefore improves the design in two ways:

- One or more gates having their outputs unconnected are removed, allowing direct area, power and delay savings.
- Gates having their inputs set to 1 or 0 can then be replaceyd by lower complexity ones.

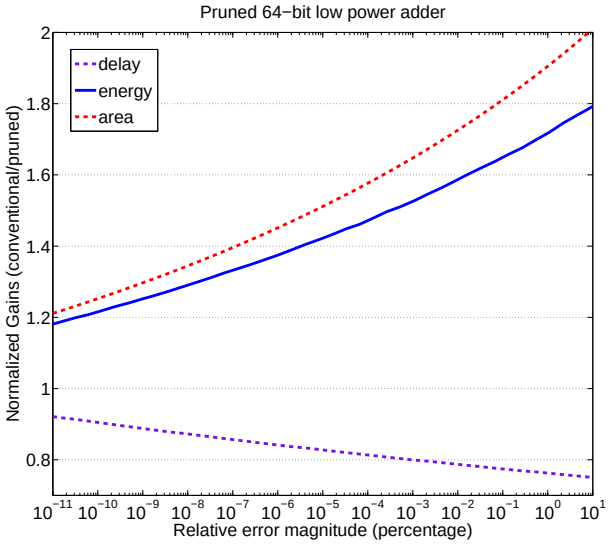
Furthermore, the resulting circuit is optimized for the timing and area constraints set by the designer. Fig. 2 shows the functional diagram of the presented pruning tool. The initial design is synthesized and mapped to a technology in order to get the gate level netlist. This netlist then enters a pruning loop composed of four steps:

1) Hardware simulation to monitor the activity of the circuit and to check if the amount of error introduced by the pruned netlist can still fit the application.
2) The significance-activity product is calculated depending of the designer's requirements (weighted or uniform pruning).
3) Wires are pruned according to the ranking of the nodes.
4) Re-synthesis of the netlist is performed in order to remove or replace non-essential gates.

Synthesis and hardware simulations are performed using existing software, whereas scripting languages are used for SAP calculation and wire pruning. This framework outputs all the gate-level netlists ranked by growing order of inexactness, i.e., by decreasing energy-delay-area product. A significant advantage of the proposed tool and methodology is that they can be embedded in an existing standard digital flow, making them fully compatible with any synthesizable HDL code. Moreover, that same flow can be used indifferently for inexact ASIC or FPGA design.



(a)



(b)

Fig. 3. Energy, delay and area savings for pruned adders implemented in UMC 65 nm technology

III. Application of Gate Level Pruning on Arithmetic Circuits

In the previous work [8], *probabilistic pruning* was done by hand on various 64-bit adders such as Kogge-stone, Han-Carlson, etc. However, it is very rare that the designer selects one of these specific architecture. In fact, HDL languages offer the power of functional descriptions, meaning that a designer does not have to dig into the details of low-level structural descriptions. As an example, an adder or a multiplier can easily be described using the standard *sum* and *product* instruction. At synthesis time, the functional description is automatically mapped to hardware with an optimal architecture under the given constraints. This decision is usually taken by the synthesis tool according to the user-specified design constraints. One of the key strength of the proposed tool is that it is able to prune any digital circuit even those produced by high-level behavioral description, the only condition being that the HDL code is synthesizable.

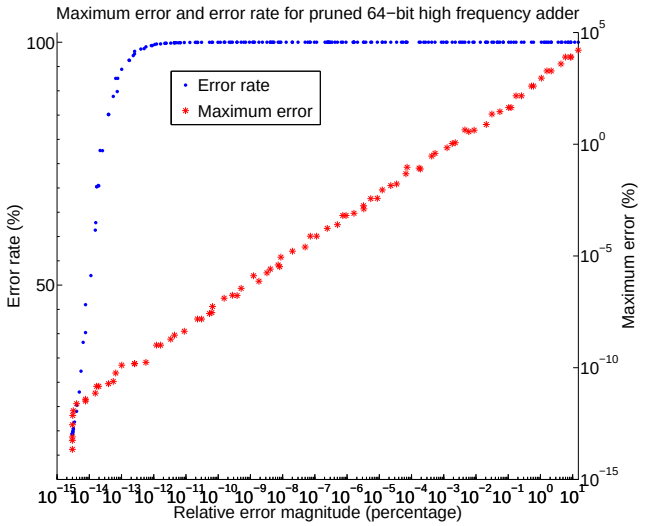


Fig. 4. Error characteristics of a pruned 64-bit high speed adder implemented in UMC 65 nm technology

TABLE I. Comparison of the two pruning techniques for 10 % relative error magnitude

Pruning technique	Area gains	Energy Gains	Delay Gains	EDAP Gains
Gate-level	21X	7.82X	1.07X	175X
Previous work [8]	1.8X	1.8X	2.3X	7.5X

Hence, significant gains improvements are shown thanks to the gate-level pruning, and the demonstration is made that this CAD framework is able to prune circuits described using the standards *sum* and *product* HDL operands, allowing the synthesis tool to select the more appropriate architecture.

A. Error characteristics

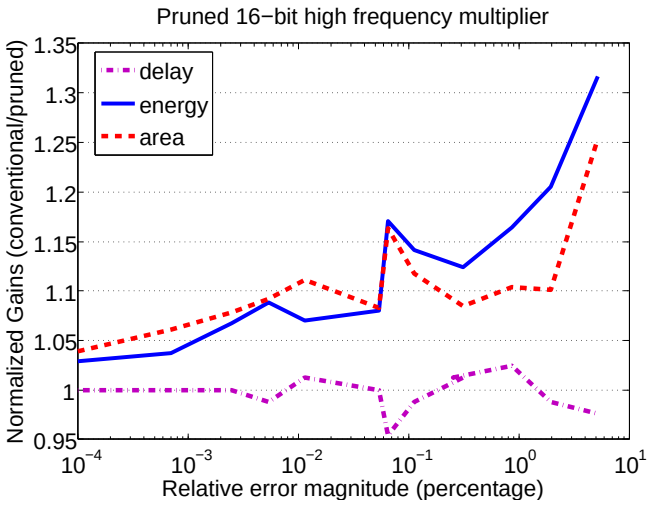
In order to get an accurate error characterization of arithmetic circuits, extensive simulations need to be performed. Thus, a 64-bit adder would have to be simulated with 2^{128} different input combinations which is a cumbersome process. Moreover, the simulation time would need to be multiplied by the number of inexact netlists generated by the pruning tool. Hence, simulating arithmetic circuits with a set of one million uniformly distributed random inputs allows to estimate the error characteristics of each inexact design within a reasonable simulation time. In the specific case of adders and multipliers where the outputs of the circuit are weighted, the relative error magnitude (REM) is calculated as

$$REM = \frac{1}{K}\sum_{k=1}^{K}\left|\frac{O_k - O'_k}{O_k}\right|, \qquad (2)$$

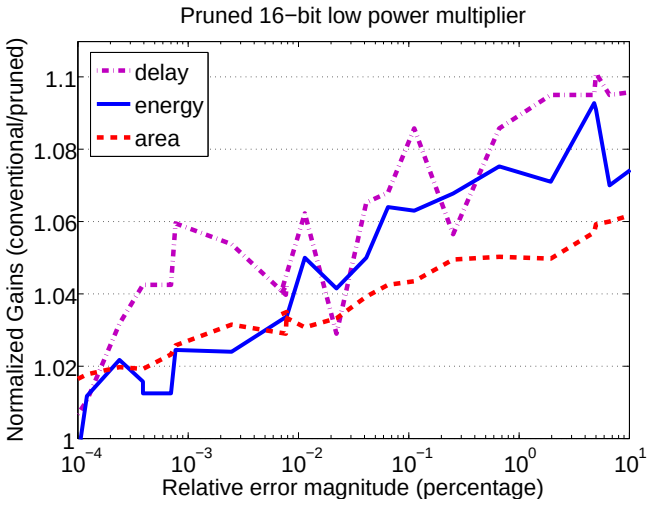
where K is the total number of computations, O_k are the exact results and O'_k are the approximate results provided by the pruned circuit.

B. Pruned adders

Fig. 3 shows the savings of pruned 64-bit adders, for the two extreme points of the design space, namely high frequency and low power. Here, the synthesis tool selects the best architecture for the given constraints, providing an optimized netlist. It is shown Fig. 3(a) that allowing an accuracy degradation of 10% can lead to factor 7.82, 21 and 1.07 savings in terms of power,



(a)



(b)

Fig. 5. Energy, delay and area savings for pruned multipliers implemented in UMC 65 nm technology

area and critical path delay, respectively. Table I compares the normalized gains for different granularities of pruning and shows that the gate-level technique allows significant improvements compared to the previous work [8]. Indeed, a highly parallel adder (i.e. high frequency) is an optimal target for the pruning tool due to high hardware redundancy. In contrast, a low power synthesis outputs a serial, low area architecture such as the ripple carry adder. Removing a small part of this adder rapidly breaks the carry chain and increases the relative error magnitude. As a consequence, for a given relative error magnitude, the pruning technique allows higher savings for parallel architectures rather than for serial ones. Thus, pruning any other adder in the design space would result in gains in-between those obtained for the two extreme design points, i.e. high frequency and low power.

Unfortunately, those huge savings come at a certain cost, Fig 4 plots the error rate and the maximum error, versus relative error magnitude for the high frequency 64-bit adder. As soon as one ore more LSBs outputs' are pruned, the error rate rapidly reaches 100%, however the magnitude of those error can remain as low as $10^{-10}\%$. A special care should be taken when using highly pruned arithmetic units, for instance, for a 10% relative error magnitude, where the savings are huge, the maximum error can be as high as $10^{4}\%$ which for some application may be clearly prohibited. However the error magnitude is bigger than $10^{4}\%$ only twice over the set of one million random inputs.

C. Pruned multipliers

Using the same pruning tools on a 16-bit high frequency multiplier shows savings up to 1.3 x in energy and 1.25 x in area (Figure 5). These lower savings can be explained by the fact that an error can propagate in a multiplicative path. In addition to that, pruning is applied on a lower bit-width hardware, giving less margin to this technique. Again, pruning is more efficient on high frequency multipliers than low power ones.

IV. Conclusion

This paper presented how to automatically generate inexact circuits using gate-level pruning. The proposed methodology and tools are in a large proportion built on existing software and can hence be fully integrated in a standard digital design flow, compatible with any synthesizable design. As a consequence, this work should help designers to trade accuracy for power, area and delay savings for a wide range of applications. Simulations have been showing that the possible savings mostly depend on the targeted circuit. For pruned adders, power consumption and silicon area can be reduced by approximately one order of magnitude at the cost of 10 % relative error magnitude while multipliers show lower, but non-negligible savings: up to 25 % energy and area reduction. However those huge savings must be nuanced as the maximum errors could limit the use of those highly pruned circuits in many applications.

References

[1] K. Palem, A. Lingamneni, C. Enz, and C. Piguet, "Why design reliable chips when faulty ones are even better," in *ESSCIRC (ESSCIRC), 2013 Proceedings of the*, Sept 2013, pp. 255–258.

[2] D. Blaauw, S. Kalaiselvan, K. Lai, W.-H. Ma, S. Pant, C. Tokunaga, S. Das, and D. Bull, "Razor II: In Situ Error Detection and Correction for PVT and SER Tolerance," in *Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. IEEE International*, Feb 2008, pp. 400–622.

[3] S. Cheemalavagu, P. Korkmaz, and K. V. Palem, "Ultra low-energy computing via probabilistic algorithms and devices: CMOS device primitives and the energy-probability relationship," in *Proc. of The 2004 International Conference on Solid State Devices and Materials*, 2004, pp. 402–403.

[4] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," in *Low Power Electronics and Design (ISLPED) 2011 International Symposium on*, Aug 2011, pp. 409–414.

[5] P. Kulkarni, P. Gupta, and M. Ercegovac, "Trading Accuracy for Power with an Underdesigned Multiplier Architecture," in *VLSI Design (VLSI Design), 2011 24th International Conference on*, Jan 2011, pp. 346–351.

[6] H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, no. 4, pp. 850–862, April 2010.

[7] K. Bhardwaj and P. Mane, "ACMA: Accuracy-configurable multiplier architecture for error-resilient System-on-Chip," in *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2013 8th International Workshop on*, July 2013, pp. 1–6.

[8] A. Lingamneni, C. Enz, J. L. Nagel, K. Palem, and C. Piguet, "Energy parsimonious circuit design through probabilistic pruning," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2011*, March 2011, pp. 1–6.