DynOR: A 32-bit Microprocessor in 28 nm FD-SOI with Cycle-By-Cycle Dynamic Clock Adjustment

Jeremy Constantin*, Andrea Bonetti*, Adam Teman*†, Christoph Müller*, Lorenz Schmid*, and Andreas Burg*
* Telecommunications Circuits Laboratory (TCL), École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
† Faculty of Engineering, Bar-Ilan University, Ramat Gan, Israel
{jeremy.constantin, andrea.bonetti, adam.teman, christoph.mueller, lorenz.schmid, andreas.burg}@epfl.ch

Abstract—**This paper presents DynOR, a 32-bit 6-stage OpenRISC microprocessor with dynamic clock adjustment. To alleviate the issue of unused dynamic timing margins, the clock period of the processor is adjusted on a cycle-by-cycle level, based on the instruction types currently in flight in the pipeline. To this end, we employ a custom designed clock generation unit, capable of immediate glitch-free adjustments of the clock period over a wide range with fine granularity. Our chip measurements in 28 nm FD-SOI technology show that DynOR provides an average speedup of 19% in program execution over a wide range of operating conditions, with a peak speedup for certain applications of up to 41%. Furthermore, this speedup can be traded off against energy, to reduce the chip power consumption for a typical die by up to 15%, compared to a static clocking scheme based on worst case excitation.**

I. Introduction

The concept of dynamic voltage frequency scaling (DVFS) has been one of the leading approaches to energy-efficient operation for the past twenty years. Furthermore, the introduction of in-situ error-detection, such as Razor [4], [5], and tunable replica circuits with error-detection [6] led to an influx of designs targeted at real-time adjustment for voltage overscaling or over-clocking, according to detected errors [7], [8]. While these innovations have succeeded in pushing margin reduction to the limit, they rely on the capability of the microarchitecture to correct a limited number of errors, for example by replay of instructions.

Despite the impressive results and innovative techniques proposed in these and other recent publications, they all adhere to the basic assumption that the clock period must be set according to a global worst-case timing path. However, the study carried out in [9] showed that by applying a non-conventional synthesis strategy to a standard microprocessor core, the longest relevant path for a given pipeline state can vary significantly, depending on the executed instruction types. Therefore, by departing from the conventional global frequency convention and adjusting the clock frequency according to the current pipeline state, margins can be reduced and average throughput and/or energy-efficiency can be improved. In this paper, we present DynOR, a full operational circuit implementing this novel approach, designed and fabricated in a deeply-scaled 28 nm fully-depleted silicon-on-insulator (FD-SOI) technology.

Contributions and Outline: The specific contributions of this work are:

1) A dynamic clock adjustment (DCA) technique and corresponding micro-architecture, which enables the trimming of dynamic timing margins occurring in a microprocessor, which results in an increased instruction throughput.
2) The first silicon implementation of a microprocessor operated by a dynamic clock, which adapts its frequency with a very fine granularity on a cycle-by-cycle basis.
3) A clock generation circuit, enabling this DCA approach.

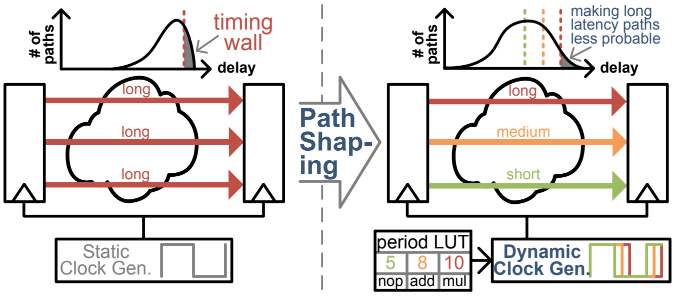


Fig. 1. Dynamic clock adjustment (DCA) approach

The rest of this paper is structured as follows: Section II overviews the concept and methodology of DCA. The chip architecture and clock generation unit are presented in Section III, followed by full chip implementation results and post-fabrication measurements in Section IV. Section V concludes the paper.

II. Dynamic Clock Adjustment Concept

The concept of dynamic clock adjustment, as introduced in [9], is based on the observation that critical paths in a microprocessor are not always excited, and therefore dynamic timing margins exist that can theoretically be exploited on a cycle-by-cycle basis. We observe that with proper design, the excitations of relevant paths largely depend on the instruction types currently residing in the processor pipeline.

The path-delay profile of a design implemented according to a standard synthesis approach results in a so-called "timing wall", as illustrated on the left hand side of Fig. 1, due to optimization steps, such as area recovery. This timing wall severely limits the effectiveness of the DCA approach. It is, however, possible to reshape this path-delay profile, such that all paths that can be made short end up being short (e.g., by employing aggressive critical range optimization), which can significantly reduce the rate of excitation for near-critical paths. Our experiments show that for the core used in this study, area and power penalties due to path reshaping can be limited to 5-13% in 28 nm FD-SOI, depending on the target library and operating voltage [9], while maintaining the frequency maximum of the circuit. By carefully designing the micro-architecture of the processor to support the DCA approach, ensuring that long paths are only activated when necessary[1], we enable the operation of the processor with a dynamic clock

[1] e.g., through shielding of sometimes irrelevant paths such as the ones through the multiplier in the ALU, when other ALU instructions are executed

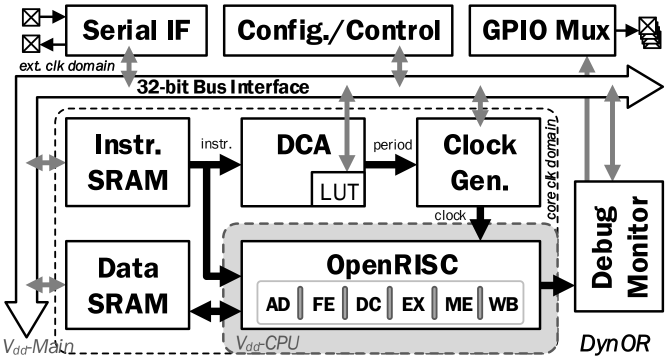


Fig. 2. DynOR architecture

period, which is based on the instruction type that is currently executed, as depicted on the right hand side of Fig. 1. To limit the hardware complexity induced by DCA, we ensure that all paths which do not reside in the execution stage of the pipeline are non-critical, or can be sufficiently accelerated through voltage scaling (i.e., SRAMs). This is possible with low overhead, as indicated by the findings in [9], which show that even in the baseline design (prior to optimization) 93% of all cycles are already limited by the execution stage.

III. DynOR Architecture

In this section we present the full chip architecture of the DynOR system, and describe in detail the key building blocks which enable our dynamic clock adjustment method.

A. Processor Architecture

The DynOR architecture is depicted in Fig. 2. At the core of the chip architecture lies a 32-bit, 6-stage, in-order OpenRISC microprocessor with one integer ALU, supporting single cycle 32-bit multiplications. The processor comprises 16 kB of tightly coupled instruction memory and 8 kB of data memory, both realized using single cycle latency 6-T SRAM macros. Instructions are fetched into the processor pipeline while they are simultaneously analyzed by the DCA module, in order to determine the required next clock period. The clock generation unit supplies the dynamic clock within the core clock domain, which comprises the CPU, the memories, and the DCA, while the peripheral/interface components of the chip are externally clocked. All modules of the core clock domain are connected to those peripherals over a 32-bit bus interface. The CPU has also indirect access to this bus over a multi-cycle memory mapped interface.

A serial interface provides communication for DynOR with the external world, while the central configuration and control module gives access to all clocks, resets, and different mode and calibration settings of the modules of the chip. Debug signals for the operation of the dynamic clock adjustment are observable via a monitor interface. A multiplexed GPIO interface also provides these debug signals, as well as access to a scan chain.

The chip comprises two separate voltage islands (with level-shifted boundaries): one for the OpenRISC core (V_{dd}-CPU) and one for the rest of the chip (V_{dd}-Main). This allows for the operation of the CPU at low voltages beyond the point where the SRAMs are functional, and enables accurate power measurement of the CPU core.

B. Dynamic Clock Adjustment Module

The DCA module provides the functionality for dynamically determining the next clock period, based on the type of the currently fetched instruction. The schematic of the DCA module, together with the clock generation, is shown in Fig. 3. During every cycle in which an instruction is fetched into the FE stage of the CPU, the DCA also receives the same 32-bit instruction from the memory bus. This instruction is then partially decoded (using a subset of the more complex decoding logic of the OpenRISC) to determine its type, such that it can be classified into an instruction group. The chosen groups are as follows: branch, branch-condition, load, store, logic, xor, shift, add, multiply, nop, register-move, and other. The group is then used to address a 12-entry 6-bit lookup table, which provides the critical period, associated with the respective instruction type/group. As can be seen from Fig. 3, this period will come into effect for the cycle when the fetched instruction will reside in the execution (EX) stage of the CPU, which contains the critical paths.

The lookup table of the DCA is populated through calibration of a die for the current operating condition (i.e., supply voltage and temperature). The minimum achievable period value of each entry is determined through binary search, where each run of the search is evaluated by testing the outcome of the program for correctness (including data memory contents), i.e., by checking if the current clock periods cause a timing error for one of the instruction groups, resulting either in an erroneous result or a processor crash. Note that the lookup table produced by a single calibration point is inherently tuned for a specific application (type). To generate a “worst case” table, which provides DCA capabilities independent of the application that is executed, multiple tables from calibrations using different applications can be merged, where for each entry the worst observed period per instruction type is chosen.

C. Clock Generation Unit

The clock generation unit (CGU), shown on the right hand side of Fig. 3, is a digitally-controlled ring oscillator (DCRO) that produces a clock signal whose period can be changed in each clock cycle, as required by the proposed DCA. To modify the propagation delay inside the ring, the oscillator includes a programmable delay unit implemented with a cascade of AND gates that have been placed with a controlled floorplan to produce an accurate range of clock periods. The cycle-by-cycle operation is ensured by using a 1-hot encoded period setting for the programmable delay unit, and by sampling this delay setting with a set of additional registers that have been placed close to the programmable delay unit to minimize both their propagation delays as well as the delay of the clock signal they receive from the ring oscillator. These registers are clocked as soon as the rising edge of the clock has propagated through the programmable delay block. This ensures that the delay setting is stable at the input of the programmable unit as soon as the next falling clock edge arrives at the input of the unit, thereby avoiding both glitches on the clock signal as well as any contamination of the clock period duration.

IV. Test Chip & Measurement Results

The DynOR system architecture, described in Section III, was fabricated in a 28 nm FD-SOI regular-V_T CMOS technology, using 0.24 mm^2 of the complete 1.2 mm^2 die, with

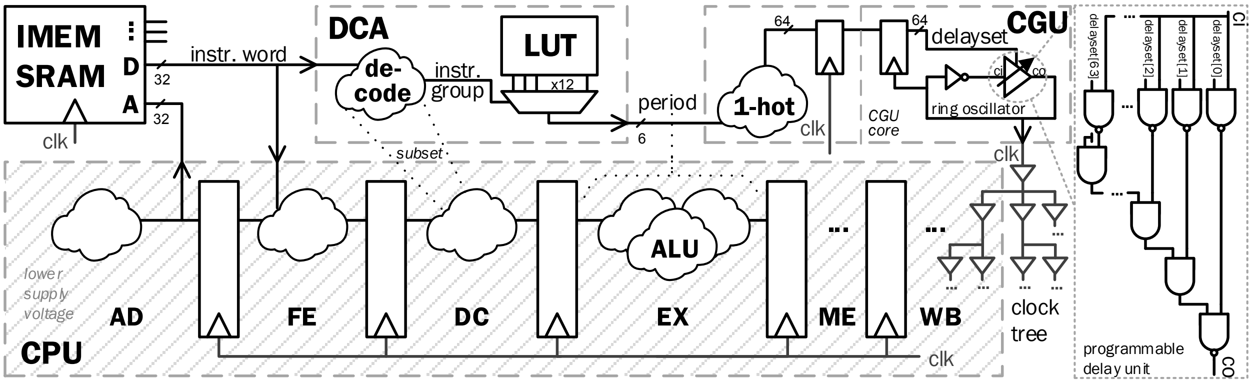
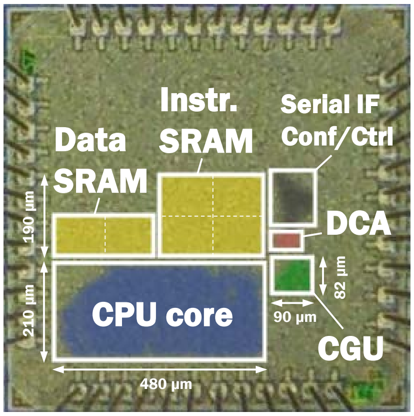


Fig. 3. Schematic of dynamic clock adjustment and generation

a complexity of about 316 k gate equivalents (GEs). The micrograph and main features of the chip are shown in Fig. 4.

Fig. 5 details the distribution of all fabricated dies, regarding their maximum static CPU frequency (F_{max}). At a CPU supply voltage (V_{dd}-CPU) of 0.8 V slow dies operate at an F_{max} which is up to 14% lower than a typical die, while fast dies are up to 9% faster. For a typical die at 1.0 V (V_{dd}-Main) the CGU provides a clock frequency between 365 MHz and 1906 MHz, with a period step granularity of 35 ps (with an average accuracy of ±1.8 ps) over 64 settings. A normal operation range for the CPU at 0.8 V (V_{dd}-CPU) employing DCA is a dynamic frequency range between 509 MHz and 1189 MHz.

In the following, we report the measured speedups and power savings, while varying two main parameters: V_{dd}-CPU and the application executed on the CPU. For each reported V_{dd}-CPU, the corresponding V_{dd}-Main is set to a voltage level that is 200 mV higher (exceptions are 0.6 V and 0.4 V for V_{dd}-CPU, where V_{dd}-Main is only 0.73 V and 0.45 V, respectively). This higher V_{dd}-Main is mainly to allow the SRAMs to provide enough access speed, and in some cases to enable the DCA to operate fast enough, as well. The two chosen applications are matrix multiplication (MM) and median calculation (MC), each with an execution length of about 10^{10} instructions. The MM (using full-range 32-bit values) can be seen as a worst case stress test, since it is heavy in multiplication instructions, and hence excites the worst paths of the CPU with the highest rate. The MC is based on sorting, and in general, performs a more balanced set of instructions, regarding their path excitations.



Technology	UTBB FD-SOI 28 nm RVT
CPU	32b, 6-stage OpenRISC with dynamic clock
Die area	0.24 mm² (chip: 1.2 mm²)
Memory	16 KB (IM) + 8 KB (DM)
Gate Equiv.	172 k (logic) + 144 k (SRAM)
V_{dd} range	0.40 V – 1.10 V (cpu min: 0.26 V)
Freq. range	24 – 1306 MHz
Power range	0.31 – 112 mW
Energy range	13 – 86 pJ/Op

Fig. 4. Die micrograph and chip features

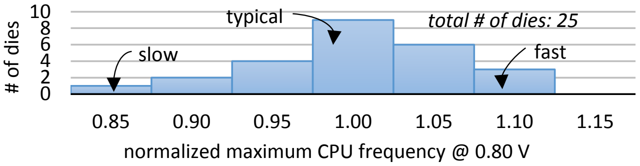


Fig. 5. Speed distribution of fabricated dies

A. Speedup

Fig. 6 reports the speedups (individually for a fast, typical, and slow die), which are all calculated relative to the baseline speed provided by F_{max} for a given V_{dd}-CPU and die at room temperature (25° C). For each parameter configuration, we report the three different speedups, achieved through:

1) Over-clocking the CPU with a higher static clock frequency F_{app}, which the application allows. A speedup over F_{max} is possible here, since we implement a path distribution that avoids the timing wall present in classical designs. Since the MM always excites the worst paths, F_{app} equals F_{max} for this case.
2) DCA using a merged “worst case” period table (as described in Section III-B), which shows the application-independent DCA gains.
3) DCA using a period table that is tuned for the specific application, showing maximum achievable DCA gains.

For the MC on a typical die at 0.8 V, we observe that the implemented path distribution allows for an application-specific speedup of 14% with an increased static clock F_{app}. The proposed DCA approach allows to raise this speedup by an additional 22% to a total of 36% through the means of dynamically adjusting the clock frequency on an instruction basis. Even with a conservative “worst case” DCA table, which operates application-independent, the speedup still amounts to 6% over F_{max}. The application specific speedup for MC enabled by DCA over all die types and different V_{dd}-CPU values is on average 25%, and can reach up to 41%. Furthermore, we see that for the MM a speedup is only enabled through DCA, and even though the application frequently excites the critical paths of the design, which works against our DCA approach, an average speedup of 6% can still be measured, which can help amortize any potential design overhead due to the applied path profile shaping. The average speedup provided by DCA over all operating conditions and applications is 19% for a typical die, and 14% for both the fast and the slow dies.

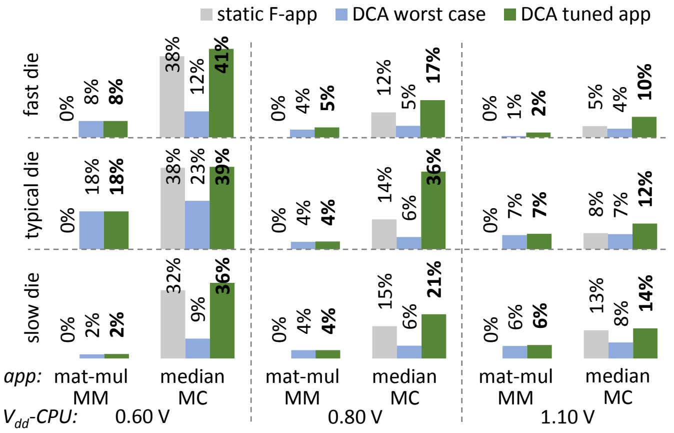


Fig. 6. Measured application speedup with respect to conventional static clocking with F_{max}, for different supply voltages, die types, and applications

In general, we see that lower supply voltages allow for higher speedups, and show the strength of DCA, especially for critical/demanding applications such as MM, while at normal and higher supply voltages, the strength of DCA is shown for more balanced (MC type) applications, where significant additional speedups over F_{app} can be achieved.

B. Power Reduction

The presented application speedups can be traded off against power consumption, by means of supply voltage scaling. In Fig. 7, we report the measured reductions in power consumption enabled by DCA at fixed throughput rates. To this end, we lower the CGU frequencies until the CPU performs the same number of Op/s as it does at fixed F_{max}. We then use the created voltage headroom to lower V_{dd}-CPU accordingly.

The power density of DynOR ranges between 13 μW/Mhz (at 0.4 V) and 87 μW/Mhz (at 1.1 V). Note that since the CPU processes very close to 1 instruction per cycle, 1 μW/MHz here is equivalent to 1 pJ/Op. At a V_{dd}-CPU of 0.8 V we measure power savings of 15% (by decreasing V_{dd}-CPU by 50 mV) for the MC at 521 MOp/s, reducing the chip energy consumption from 50 pJ/Op down to 42 pJ/Op.

Since, to our best knowledge, there are no other fabricated designs which employ a dynamic clocking scheme, a direct comparison with other recent works regarding the effectiveness of the micro-architectural design techniques proves to be difficult. Nevertheless, to put our silicon implementation into a broader context of embedded processing, regarding the overall performance numbers, Table I compares this work with other recent CPU implementations.

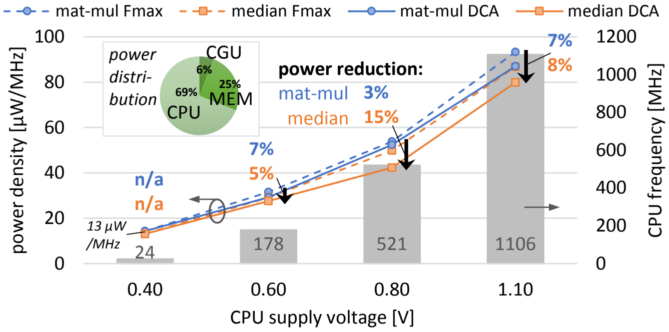


Fig. 7. Measured power reduction at iso-throughput for a typical die, due to supply voltage scaling enabled by DCA

TABLE I. COMPARISON WITH RECENT CPU IMPLEMENTATIONS

	This work	[1]	[2]	[3]
Domain	**Embedded Processing**	Near-Sensor Processing	IoT / MCU Applications	General Purpose Processing
Technology	**FD-SOI 28nm RVT**	FD-SOI 28nm LVT	FD-SOI 28nm	32nm
CPU	**32b OpenRISC**	32b OpenRISC	32b Cortex-M4	32b IA-32 Pentium
# of cores	**1**	4	1	1
I$/D$/L2	**16K/8K/-**	4K/48K/64K	8K/8K/-	8K/8K/-
Voltage range	**0.40 – 1.10 V**	0.32 V – 1.15 V	0.34 – 1.10 V	0.28 – 1.20 V
Max. frequency	**1306 MHz**	825 MHz	834 MHz	915 MHz
Best performance	**1.3 GOPS**	3.3 GOPS	?	1.8 GOPS
Best power density	**13.0 µW/MHz @ 24 MOPS**	20.7 µW/MHz @ 4x40.5 MOPS	8.9 µW/MHz @ 45 MHz	170 µW/MHz @ 100 MHz
Features	**Dynamic Clock Adjustment**	Body Biasing, Shared TCDM	10-T SRAM	2x Superscalar, FPU, Bran. Pred.

V. Conclusion

This work introduces the first silicon implementation and micro-architecture of a 32-bit microprocessor which uses a dynamic clocking scheme to vary its clock period on a cycle-by-cycle basis, according to the executed instruction types. The effectiveness of this dynamic clock adjustment (DCA) approach is enabled through careful path reshaping of the logic in the processor pipeline. Our measurements in 28 nm FD-SOI show that by employing DCA, the throughput of the processor can be increased on average by 19% (up to 41%), while the power reduction reaches up to 15% for a typical die. Moreover, we demonstrate that consistent speedup is possible over varying die speeds, supply voltages, and applications, which can be differently suited for the proposed technique.

Acknowledgment

The authors would like to thank Davide Rossi (Univ. of Bologna) and Philippe Flatresse (STMicroelectronics), for their helpful advice regarding the employed fabrication technology, and Nicholas Preyss (EPFL) for insightful discussions. This work has been supported by: (a) The European Commission's 7th Framework Programme (FP7/2007-2013) under the grant agreement FP7-323872 (Project "SCoRPiO") and (b) the Swiss Nano-Tera Program (Project "IcySoC").

References

[1] D. Rossi *et al.*, "193 MOPS/mW @ 162 MOPS, 0.32V to 1.15V voltage range multi-core accelerator for energy-efficient parallel and sequential digital processing," in *IEEE COOL Chips XIX*, April 2016.

[2] F. Abouzeid *et al.*, "28nm FD-SOI technology and design platform for sub-10pJ/cycle and SER-immune 32bits processors," in *IEEE ESSCIRC*, Sept 2015, pp. 108–111.

[3] S. Jain *et al.*, "A 280mV-to-1.2V wide-operating-range IA-32 processor in 32nm CMOS," in *IEEE ISSCC*, Feb 2012, pp. 66–68.

[4] D. Ernst *et al.*, "Razor: A low-power pipeline based on circuit-level timing speculation," in *IEEE/ACM MICRO*, Dec 2003, pp. 7–18.

[5] S. Das *et al.*, "RazorII: In situ error detection and correction for PVT and SER tolerance," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 32–48, Jan 2009.

[6] J. Tschanz *et al.*, "Tunable replica circuits and adaptive voltage-frequency techniques for dynamic voltage, temperature, and aging variation tolerance," in *Symposium on VLSI Circuits*, June 2009, pp. 112–113.

[7] Y. Zhang *et al.*, "iRazor: 3-transistor current-based error detection and correction in an ARM Cortex-R4 processor," in *IEEE ISSCC*, Jan 2016, pp. 160–162.

[8] E. Beigné *et al.*, "A 460 MHz at 397 mV, 2.6 GHz at 1.3 V, 32 bits VLIW DSP Embedding F_{MAX} Tracking," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 1, pp. 125–136, Jan 2015.

[9] J. Constantin *et al.*, "Exploiting dynamic timing margins in microprocessors for frequency-over-scaling with instruction-based clock adjustment," in *IEEE DATE*, March 2015, pp. 381–386.