How to Live with Uncertainties: Exploiting the Performance Benefits of Self-Timed Logic In Synchronous Design

G.Paci
DEIS, University of bologna, Italy
Giacomo.Paci@unibo.it

A. Nackaerts
IMEC, Belgium
Axel.Nackaerts@imec.be

F.Catthoor
IMEC, Belgium
Francky.Catthoor@imec.be

L.Benini
DEIS, University of Bologna, Italy
Luca.Benini@unibo.it

P.Marchal
IMEC, Belgium
Pol.Marchal@imec.be

Abstract

Ultra low power digital systems are key for any future wireless sensor nodes but also inside nomadic embedded systems (such as inside the digital front end of software defined radios). These systems require the highest possible energy efficiency of logic, which can only be achieved by operating in moderate inversion. Unfortunately, when operating near the threshold voltage, transistors become highly sensitive to process variations, thereby increasing leakage currents and complicating timing closure. Rather than pursuing a worst-case design approach for dealing with these uncertainties, we present a hybrid self-timed/synchronous approach. It will be demonstrated on the VEX VLIW core designed for ultra low-power operations. Experimental results of our approach demonstrate performance benefits up to 2x and significant energy savings at low throughput rates.

1. Introduction

Ultra low power (ULP) digital systems are a generic technology useful for wireless sensor nodes but also inside nomadic embedded systems (such as inside the digital front end of software defined radios). A key requirement for these systems is that they provide the highest achievable computational performance (1-10MOPS) consuming at most 1mW on average [3]. While targeting low to medium performance targets, the highest energy efficiency is achieved by operating in moderate inversion. Unfortunately, at these low operating voltages (near the threshold voltage), transistor switching speed and leakage power are extremely sensitive to process variations, thereby complicating timing closure of synchronous designs. Design margins at all levels of abstraction increase rapidly, causing a delay and power penalties.

In this paper we compare two alternative solutions for dealing with these extreme variations: the classic synchronous design based on corner point analysis and self-timed logic based on dual rail logic. Self-timed logic remains functionally correct under delay variations. Therefore, no additional design margins are needed to cope with process variations. In contrast to synchronous designs, also no need exists to over-design for input data variations. As a result, self-timed logic operates on average many times faster compared to synchronous logic. The introduction of dual rail logic is however far from trivial: (1) usually, synchronization between stages is performed using handshake protocol, but this is not compatible with commercial testing solutions; (2) the overhead of dual rail logic is high in terms of area and power and (3) dual rail logic requires dedicated logic libraries.

In this paper, we therefore propose an alternative technique to convert a synchronous design into a self-timed one. Rather than replacing the clock network with a handshake protocol, we retain the clock, but we apply clock-gating until logic has completed. The completion of logic is detected using dual rail logic. To limit circuit overhead this logic is inserted only in the most critical parts/stages of the design. In this way, we exploit the performance benefits of self-timed logic and remain compatible with existing design solutions for synchronous logic. This hybrid system can be clocked at high frequencies without generating functional errors. In this paper, we delimit how the throughput increases with the clock frequency and in this way, how to optimize the clock frequency for the average case rather than worst case operating conditions. Experimental results on the VEX VLIW core [16] will be presented to quantify this hybrid approach. To this end, we have developed a complete flow to convert a synchronous standard-cell based

combinational logic stage into a self-timed one. Our results indicate that significant performance benefits (up to 2x) can be realized by removing the pessimism of worst-case design. Both energy/throughput benefits strongly depend on the selected circuit topology.

This paper is organized as follows. In section 2, we will first explain the related work. Then, we discuss our strategy to deal with process variations in more detail. Finally, we outline our exploration method (see section 4) and discuss the experimental results on the VEX core (see section 5).

2. Related Work

In recent years, both industry and academia have shown a large interest for ultra-low power systems, as enabling technology for autonomous systems [1-2]. Initial performance/energy requirements for autonomous systems for different possible application domains are described in [3]. We focus in this paper on high bandwidth systems, providing the highest achievable computational performance while consuming less than 1mW on average.

A good approach for achieving this performance/energy target is reducing the power supply [3] and preferably in combination with scaled/domain specific process technologies [4]. The benefits of ultra low voltage operation have been validated down to silicon-level (e.g., the FFT-processor of [6]). However, as the voltage is reduced, the sensitivity to process variations increases [5]. As a result, the system may functionally fail and/or is subjected to large parametric variations (in delay/energy), which at the end result in yield loss. In practice, this limits the maximal achievable energy savings.

Rather than restricting parametric variations at design-time with design margins, our objective is to build systems that measure delay variations and adapt their performance at run-time depending on actual system requirements (similar to the approach followed for memories in [9]). This requires logic that operates correctly under large variations and should not guarantee the performance at design-time. Hence, design margins for timing closure are no longer needed. Logic with Razor-latches partially exhibits this property and can operate in a better-than-worst-case fashion [10]. Unfortunately, it can only cope with limited delay variations (up to 50% of the clock). However, in moderate inversion larger variations may occur in practice (see section 5). An alternative to Razor-latches is self-timed logic, which naturally deals delay-variations [11]. Under normal operating conditions (high voltage and limited variability), self-timed logic has an area, power, performance offset compared to a typical synchronous design. However, in the domain of ULP processing the balance may change in favor of self-timed logic as the amount of variations and the sensitivity to them is much higher. For instance, a better-than-worst-case DLX processor designed in asynchronous logic using matched delay-lines is presented [12]. Despite the fact that this approach is robust to design margins for systematic and environmental uncertainties, it cannot cope with random variations (which are said to be dominant below 90nm). In attempt to deal both with systematic and random variations in a single shot, we study the potential benefits of dual rail encoding logic in this paper. As this logic progresses based on the delay of the logic itself, it may eliminate these variations. In the next sections, we explain how we use dual rail logic for dealing with process variations and quantify its performance benefits.

3. Delay-variation resilient

As indicated in the introduction, performance targets between 1-10MOPS can be achieved while operating in moderate inversion. While operating just above the threshold voltage, systems become very sensitive to process variations, thereby complicating timing closure. Rather than over-designing the system, we explore (partially) self-timed circuits that can naturally live with variations. These will operate on average faster compared to synchronous logic. Self-timed circuits consist of two parts: (1) a mechanism to detect the completion of the combinational logic; (2) a mechanism to synchronize sequential register stages of the logic. Both completion detection and synchronization circuits for self-timed logic are rather complex logic designs. Careful introduction of these components into the system is therefore mandatory. Consequently, to limit overhead, it should be feasible to build a system where *only the most critical components for variations have been made delay-variation resilient and where the performance of the entire system is determined by the completion time of these critical circuits*.

3.1. Completion detection using dual rail logic

Several ways exist to build completion detection logic (see [11]). In the context of this paper, we have used dual rail logic, which detects the operation completion by coding the validity of the output signals.

At the start of a new computation cycle, all signals are in the neutral state. During the computation the signals transit into either a valid 1/0 state. When all signals move into a valid state, this means that the computation has completed. To code the valid/invalid, dual rail logic uses two wires to represent each binary signal (see Table 1).

Table 1. Dual Rail Encoding

State	*Logic value*	*Wires*	
Valid	False	GND	VDD
Valid	True	VDD	GND
Invalid	Neutral	GND	GND
Not Allowed	-	VDD	VDD

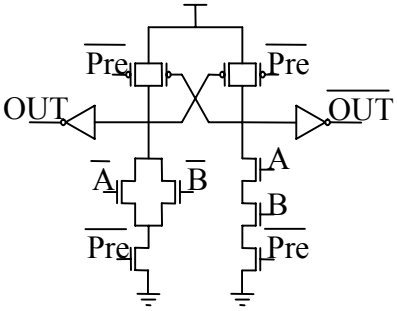


Figure 1. DCVSL NAND

We implemented dual rail logic with DCVSL (Dynamic Cascode Voltage Switch Logic). The main advantage of this logic style is its speed. A NAND2 gate implemented in DCVSL logic is shown in Figure 1. DCVSL is *weakly indicating*, i.e. it starts to compute valid outputs even when not all input signals have become valid. E.g., as soon as either A or B equal zero, the left pull down circuit will discharge the pre-charged left-node, and in this way drive the output of the NAND2 gate to a valid true signal. This makes the circuit faster compared to alternative *strongly indicating* approaches where all input data have to be valid before computing the output (e.g. [13][14]).

Moreover, the pre-charge signal can reset the logic into the invalid state much faster compared to strongly indicating logic. In the latter, all paths have to become valid, before the logic can be invalidated again.

The dual rail encoding is more complex compared to static CMOS (e.g., all wires are duplicated). Therefore, its power overhead has to be carefully analyzed.

3.2. 4-way handshaking protocol vs. clock gating

We have explored two approaches to synchronize sequential logic. First, we have used a 4-way handshaking protocol [12], with a slight modification to work with the completion signal of the logic and for driving the pre-charge signal (see Figure 2-left).

In our view, the introduction of handshake circuits to replace synchronous logic is involved. There are lots of additional timing issues at the physical design level. Furthermore, handshaking complicates testing significantly. Finally, with handshake circuits, it is difficult to partly introduce delay-variation resilient circuits in the most critical parts of the system.

Therefore, as alternative, we examine an approach to replace the critical parts of the system with self-timed logic (see Figure 2-right). Basically, additional logic is inserted to gate the clock. The clock gating is controlled by the completion detection circuits: as long as these critical stages have not finished their computations, the clock is not distributed (i.e. it is locally stopped).

This gated-clock network can be implemented similar to [10]. In contrast to [10], we gate the clock

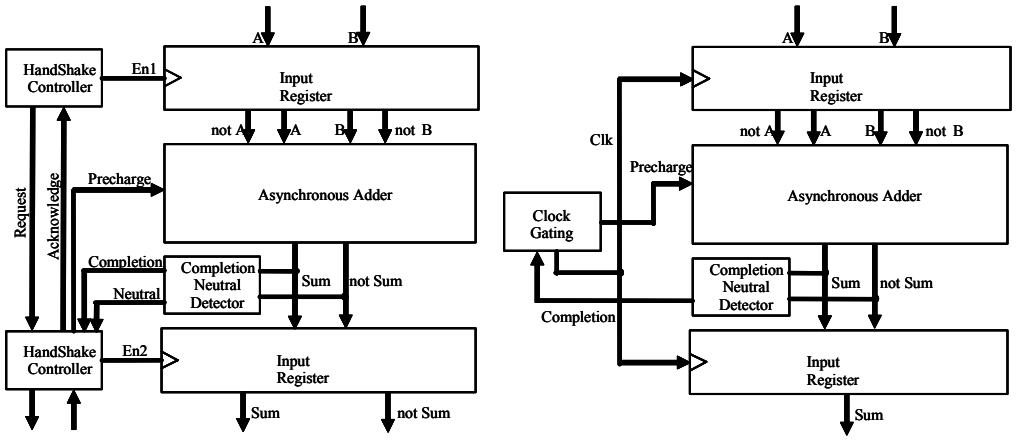


Figure 2. Pipestage structure with handshake and clock gating

until the computations are done rather than stalling it until correct data has been restored.

As a result, the clock does not need to be operated any longer at the worst-case delay. It may be operated faster without causing functional errors. As the logic usually completes faster than the worst-case, the throughput of the system can be significantly improved. The average throughput of the system can be computed with following formula:

Equation 1. Average delay (t_{avg}) of an operation on the clock-gated system; delay of the combinatorial logic (t_{ops}) and delay of the completion logic (t_{cmp})

$$t_{avg} = t_{clk} * \sum_{\forall i>0} i * P(t_{clk} * (i-1) < t_{ops} + t_{cmp} < t_{clk} * i))$$

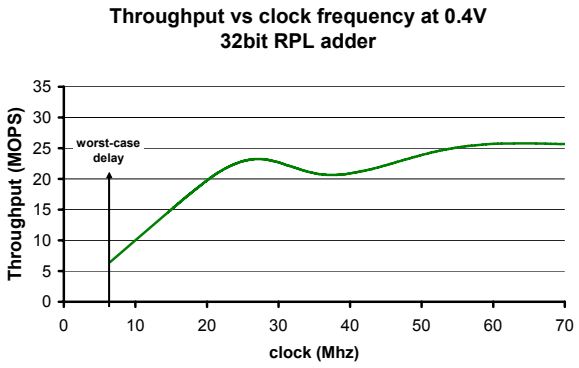


Figure 3. Throughput of self-timed logic (with clock gating) depends on clock speed

As an illustration, we present the throughput of a ripple adder in function of the clock frequency in Figure 3. The arrow represents the synchronous reference adder. It can operate maximally at 6.5Mhz and thus achieve 6.5MOPS. The clock-gated design can be operated faster. Given the fact that in a ripple adder operations for most input patterns complete much faster than the worst-case one, the throughput can be increased almost linearly by speeding up the clock. The benefits saturate at 27Mhz, where a three and an half times higher throughput than for the synchronous one is achieved. Thereafter, the number of errors increases rapidly, explaining the throughput leveling off. It even slightly decreases, because the clock will trigger the registers only on next cycle after the logic completes.

Finally, when increasing the frequency to extremely high values, the performance reaches the performance limit set by an ideal self-timed design. However, in practice, a maximal operating frequency exists in the proposed hybrid self-timed/synchronous approach: the clock frequency can only be increased until the critical path determined by the synchronous part of the design. Increasing clock frequency further would render these synchronous parts of the design functionally incorrect.

4. Evaluation Framework

4.1. Library Design

As baseline for all our experiments, we use both synchronous and self-timed cell-library containing 27 gates: NAND2, NAND3, NOR2, NOR3, EXOR, MULTIPLEXER, INVERTER, BUFFER and DFLIPFLOP. The INVERTER and BUFFER are designed in different versions: minimal size and with increased drive strength 1, 2, 4, 9. The cells are designed in IMEC's 130nm CMOS technology down to layout level. Particularly, the GDSII was generated with Synopsys Cadabra. Based on the layouts of the cells, we have generated the Synopsy target library (.lib) and physical library (.plib).

Table 2. Area Penalty of DCVSL

NAND2	3.5
EXOR	2.2

4.2. Synthesis

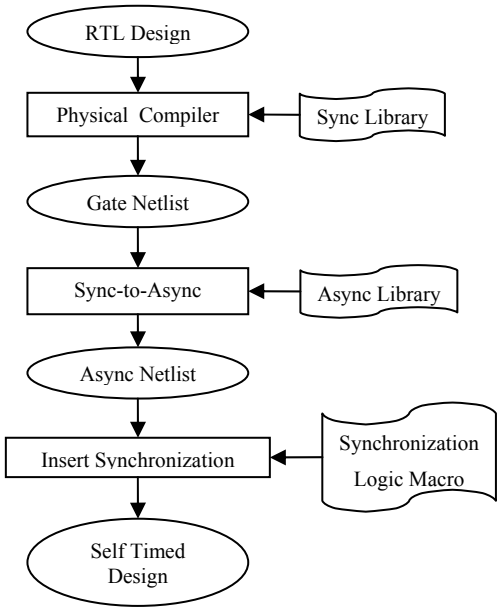


Figure 4. Synthesis of self-timed design

We create a synchronous gate-level netlist from a RTL description of the design using Synopsys Physical Compiler. The RTL design is mapped onto the gates of the synchronous library characterized with the nominal process conditions. Constraints were set to optimize power consumption.

The self-timed design was built using a script that translates synchronous design into a self-timed one. This entails three steps (see Figure 4):

From synchronous to asynchronous gates: we replace every synchronous logic gate with its asynchronous counterpart having a similar input capacitance and drive strength. As the relative strengths/loads between the cells remain similar to the synchronous design, the technology mapping decisions of the synthesis tool can be preserved. Note that better ways exist to synthesize with dual rail gates, exploiting for instance the dual output during logic mapping. However, commercial synthesis tools do not support these optimizations, hence our results will be conservative with respect to an optimized self-timed synthesis flow, but realistic in the context of current commercial toolflows.

Adding signals: we interconnect all extra ports of the dual rail logic, i.e. the pre-charge signal and the signals' complements. No extra flipflops are added to store signal complements.

Adding the synchronization logic: we add completion detection circuit and the synchronization logic (handshake circuit or the clock-gating logic). The completion detection circuit consists of a balanced tree of NOR2 gates implemented in static CMOS logic.

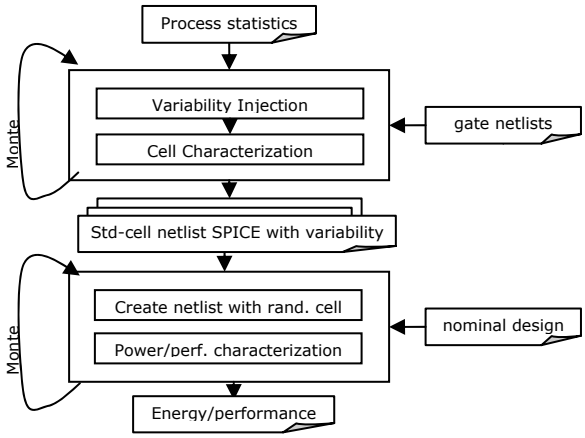


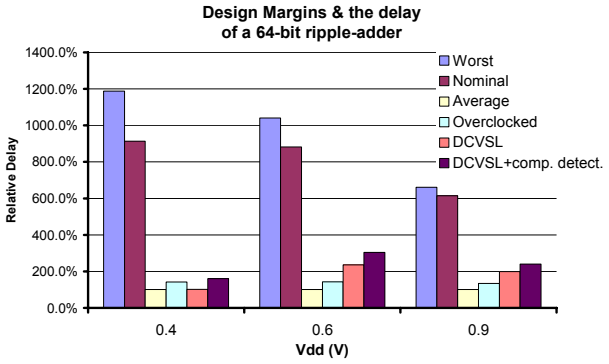
Figure 5. Performance characterization under process variation

4.3. Evaluation under process variations

We evaluate the both synchronous and asynchronous design under process variations. For this purpose, we characterize the standard cells after subjecting the cells' netlists to random β (σ_β = 10%) and Vt (σ_{vt}=5%) variations (see Figure 5). The results of this characterization are converted into a standard cell library. Besides storing the nominal delay/power of each cell, we also store the characterization results of each cell when subjected to random process variations. Then, at the gate-level, we analyze the impact of the process variations. For this purpose, we randomly replace the default cells with these variants. By generating several possible instances, statistics on the design's delay/power distribution are gathered

5. Experimental Results

In this section, we compare the performance/energy of synchronous vs. self-timed for both a ripple and a Brent-Kung adder based pipeline.



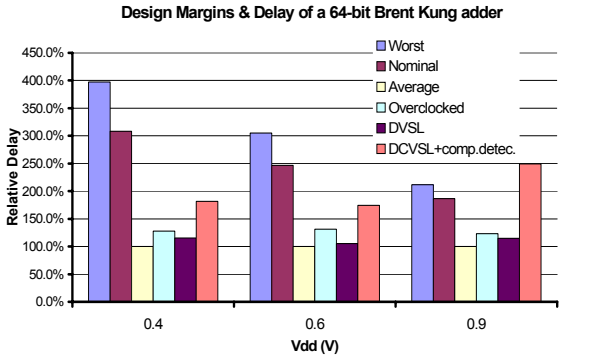


Figure 6. Exploiting variations - PUT

5.1. Uncertainties and their impact on delay

In Figure 6, we indicate how the relative delay for of a 64-bit ripple (rpl) and Brent-kung (bk) adder. In a synchronous design, the system's throughput is defined by the clock frequency, set by the critical path for the worst-case corner. This results in large design

margins, and thus performance loss compared to the average delay of the combinatorial logic, particularly when operating at low operating voltages. E.g., the delay difference between the average performance and the worst-case one for a ripple-adder is 12x at 0.4V. A self-timed logic style that removes both design margins for process and input variations, thus may significantly improve the performance.

5.2. Handshaking vs. clock-gating

As explained in section 3.2, we compare synchronization with either handshaking circuits or using a solution based on clock-gating. Handshaking achieves the average performance of Figure 6. Due to the discrete nature of the clock signal, the throughput of a clock-gating based solution is up to 40% slower, but still outperforms synchronous logic by 8x (at 0.6V) for the ripple-adder and greatly simplifies testing. The operating frequency of the clock-gating based approach has been optimized to achieve the maximum throughput.

5.3. From synchronous to dual rail logic.

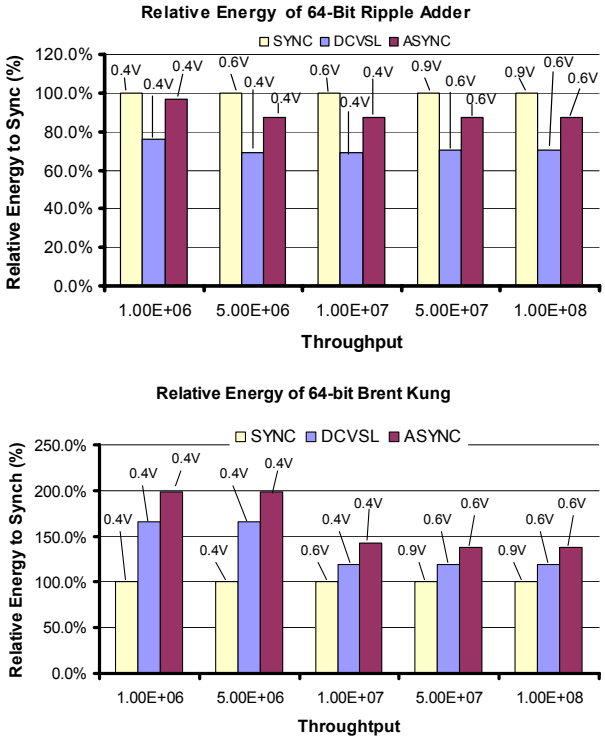


Figure 7. Relative Energy Using Asynchronous Logic Based on DCVSL (MOPS)

As can be seen in Figure 6 the DCVSL logic is in most cases considerably faster than the worst-case synchronous design. Lowering the voltage improves the relative performance of DCVSL (including completion detection logic) compared to the synchronous logic. At lower voltages, synchronous logic performs worse due to increasing short circuit currents and a poor Pmos/Nmos current ratio. The relative overhead of the Completion Detection Circuits (CDC) depends on the topology and operating voltages. As the logic depth of the adder becomes smaller (i.e. when going from a ripple adder to a Brent Kung one), the relative overhead of the CDCs significantly increases. This is why at 0.9V the synchronous BK adder, designed for the worst-case outperforms the dual rail one.

From this experiment we thus conclude that significant performance gains can be achieved while using self-timed designs, particularly when operating at low supplies.

Comparing energy consumption in Figure 7, we indicate how this performance improvement may be converted into energy savings. We plot the energy consumption of both adders designed in static CMOS, DCVSL logic and DCVSL with completion detection circuits. The operating voltage has been optimized to achieve the desired performance target (see Figure 7).

The benefits depend on the selected topology. In case of the ripple-adder, the DCVSL logic improves the energy efficiency between 24%-30% compared to static CMOS, despite its dynamic nature and high area overhead. At 10MOPS, its 30% energy efficiency is the result of a lower operating voltage (0.4V rather than 0.6 V) and less short circuit currents at these low power supplies. The total energy consumption of the asynchronous adder including CDC improves between 4% - 13%. The difference in energy consumption is caused by the CDC, of which the energy efficiency decreases at lower operating supplies

In case of the Brent-Kung adder, self-timed logic is more energy-hungry than the synchronous ones due to complexity of dual rail logic. Therefore, while exploiting self-timed logic to improve the performance of an entire pipeline, designers should carefully analyze how/where to introduce self-timed logic. Replacing all synchronous logic with dual rail one, will result in the highest achievable performance but may come at a energy penalty. A selective introduction is therefore desirable to limit the overhead of the dual rail logic.

5.4. A hybrid self-timed/synchronous approach demonstrated on the VEX VLIW

In this section, we illustrate the energy vs. throughput benefits of a more selective introduction of

dual rail logic on a VLIW core. The VEX VLIW [15] consists of 4 pipeline stages (fetch, decode, execute and write back). The VLIW has been synthesized for energy with a 1.2V library, and targets a frequency of 132 MHz. Initially, its critical path (7.53ns) runs through the ALU (EX-stage), containing a 64-bit ripple adder (ALU-delay=6ns, bypass=0.84ns, others=0.68ns).To improve the average throughput of the system, we replace the ALU with self-timed logic and over clock the system (as introduced in section 3.2). Other parts of the system are kept synchronous. The maximum clock frequency of the resulting system is determined by the longest path in the synchronous parts of the system, i.e. any other stage than the ALU. In this case, this corresponds to the decode stage which has a delay of 3.75ns. Hence, in this case the system can be over-clocked by a factor two. While operating at the double frequency only very few operations of the ALU take more than one cycle. Rather than designing for extremely rate worst-case input patterns and process variations, an average-case design running at the double frequency increases throughput by a factor 1.992.

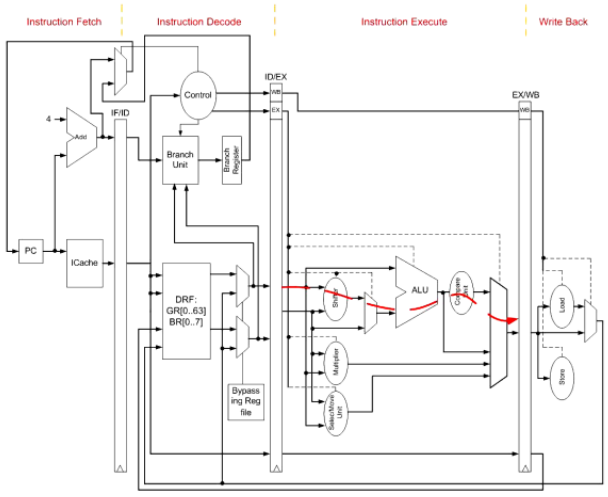


Figure 8. VEX VLIW: We have converted the execute stage into a self-timed one to increase the throughput by taking advantage of process and data-dependent input variations.

Next, we investigate the energy benefits of the average-case design in more detail. In Figure 9, we plot the [throughput-energy consumption] of the VEX core implemented for the worst-case (VEX_WC_RPL) and average case (VEX_AVG_RPL) running at different operating supplies. Operating at the same voltage, the VEX core designed for the average case runs almost two times faster. This can be seen in Figure 9 where VEX_AVG_RPL design is shifted to the right compared to VEX_WC_RPL. At higher voltages, the asynchronous execute stage is 1.56 times more energy-hungry compared to synchronous one (see also previous paragraph). However, the energy penalty per operation for the entire system is only 21%. At lower voltages, asynchronous circuits become more efficient compared to synchronous ones. There replacing the execute stage makes the design both more energy efficient and faster.

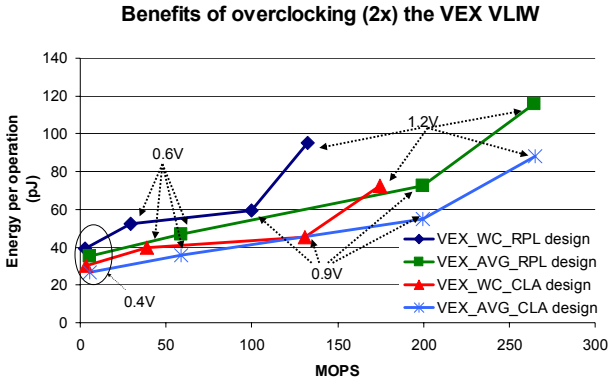


Figure 9. Better throughput/energy consumption for a partly asynchronous design of the VEX VLIW. VEX_WC_X design is a typical synchronous design instance. Its operating frequency is determined by worst-case conditions. VEX_AVG_X is an instance of the VEX designed for the average-case. To limit circuit overhead, only the ALU has been made self-timed logic rather than converting the entire design into an asynchronous one. The design was tested for two versions of the ALU: one where the ALU consists of a 64-bit ripple adder and one where it contains a 64-bit CLA.

Finally, we have replaced the ripple-adder in the VEX ALU design with a faster carry-look ahead (CLA) one. As a result, the delay of the ALU reduces to 5.7ns, but remains the critical path in the design. However, the delay-ratio between this path and worst-case path in the synchronous parts of the design reduces to 1.5. Hence, the throughput difference between the worst-case and average case design is lower compared to the ripple-adder design. Energy trends are similar. From above, we conclude that a selective introduction of self-timed logic in a VLIW may improve energy efficiency for a targeted throughput by removing pessimism on data and process variations. The results are strongly design-dependent though.

6. Conclusions

The highest energy efficiency for ultra low power applications requiring a performance between 1-10MOPS is achieved while operating in moderate/weak inversion. In this operating region, design margins for timing closure due to process and as input variations considerably slow down the throughput. Self-timed logic, such as dual rail logic, can eliminate these design margins. However, self-timed logic implies a large area overhead and may slightly increase the power consumption. Therefore, it is important to only replace those critical parts of the system which are most sensitive to variations. To allow the partial introduction of self-timed logic in an existing synchronous system, we have proposed a synchronization solution based on clock-gating. This solution performs slightly worse than a handshake-based synchronization, but still outperforms synchronous logic. We have demonstrated this approach on a VLIW core. The results indicate that precise performance/energy benefits strongly depend on the specific logic architecture, but removing the pessimism inherent to worst-case design in all cases improves the throughput (up to 2x) and results in more energy-efficient designs at low operating supplies.

7. Acknowledgement

This work has been partially supported by the GALAXY European Project (FP7-ICT-214364).

8. References

[1] D. Butler, "Everything, Everywhere", *Nature*, Vol.440, no. 23, pp402-405, 2006

[2] J. Rabaey, "Traveling the Wild Frontiers of Ultra Low Power Design ", *keynote IEEE PATMOS*, Belgium, 2005

[3] L. Nazhandali et al., "Energy Optimization of Subthreshold-Voltage Processors", *Proc. ACM/IEEE ISCA*, June 2005.

[4] A. Raychowdhury et al., "Computing With Subthreshold Leakage: Device/Circuit/Architecture Co-Design for Ultralow-Power Subthreshold Operation", *IEEE T'VLSI*, Vol 13, No. 11, pp. 1213-1224, Nov., 2005

[5] Y. Cao et al., "Yield optimization with energy-delay constraints in low-power digital circuits", *Proc. IEEE Electronic Devices and Solid-State Circuits*, Dec. 2003

[6] A. Wang et al., "A 180-mV Subthreshold FFT Processor Using a Minimum Energy Design Methodology" *IEEE Journal of Solid-State Circuits,* vol. 40, no. 1, pp. 310-319, January 2005.

[7] D. Sylvester, D. Blaauw and E. Karl, "ElastIC: An Adaptive Self-Healing Architecture for Unpredictable Silicon", *IEEE D&T*, Vol. 23, No. 6, pp 484-490.

[8] M. Mani, M. Orshansky, "A New Statistical Optimization Algorithm for Gate Sizing", *Proc. ICCD* 2004, 272-277

[9] A. Papanikolaou et al., "A system-level methodology for fully compensating process variability impact of memory organizations in periodic applications", Proc. *CODES+ISSS* 2005, p117-122

[10] D. Ernst et al., "Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation", *Proc. Symp. Micro*, 2003

[11] J. Sparsø et al., "Principles of asynchronous circuit design - A systems perspective.", *Kluwer Academic Publishers*, 2001.

[12] J. Cortadella et al., "Desynchronization: Synthesis of Asynchronous Circuits From Synchronous Specifications.", *IEEE Trans. on Vol. 25, Issue 10*, Oct. 2006 Page(s):1904 - 1921.

[13] D. Sokolov et al., "Design and Analysis of Dual-Rail Circuits for Security Applications", *IEEE Trans. on Vol. 54, Issue 4*, Apr. 2005 Page(s):449-460.

[14] A. Kondratyev, K. Lwin, "Design of asynchronous circuits using synchronous CAD tools", *IEEE Design & Test of Computer, vol 19, issue 4*, July-Aug. 2002 Page(s):107 – 117.

[15] J.A.Fisher, P.Foraboschi, C.Young, "Embedded Computing. A VLIW Approach To Architecture, Compilers and Tools", Morgan Kaufmann, San Francisco, 2005.