Circuits for High-Performance Low-Power VLSI Logic

by

Albert Ma

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2006

Author .....
Department of Electrical Engineering and Computer Science
May 25, 2006

Certified by .....
Krste Asanović
Associate Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Circuits for High-Performance Low-Power VLSI Logic

by

Albert Ma

Submitted to the Department of Electrical Engineering and Computer Science
on May 25, 2006, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

The demands of future computing, as well as the challenges of nanometer-era VLSI design, require new digital logic techniques and styles that are simultaneously high performance, energy efficient, and robust to noise and variation. We propose a new family of logic styles called Preset Skewed Static Logic (PSSL). PSSL bridges the gap between the two main logic styles, static CMOS logic and domino logic, occupying an intermediate region in the energy-delay-robustness space between the two. PSSL is better than domino in terms of energy and robustness, and is better than static CMOS in terms of delay. PSSL works by partially overlapping the execution of consecutive iterations through speculative evaluation. This is accomplished by presetting nodes at register boundaries before input arrival.

Thesis Supervisor: Krste Asanović
Title: Associate Professor

Acknowledgments

I would like thank God for the opportunity He gave me to do this PhD, the strength to finish it, and the people He gave that supported me all the way. Thank you Krste for your guidance, patience, and grace. You have done the impossible in graduating me. Thanks to Srini and Anantha for being on my committee. Thanks to the SCALE group for you help and support.

I also want to thank my parents, for their sacrifice, love, and support through the years. Finally, I want to thank my wife Sophia, Pastor Paul, Becky JDSN, Pastor Chris, Sally SMN, Heechin JDSN, Jean SMN, and all those in the body of Christ who have prayed for me these ten long years.

This work was partially supported by NSF CAREER Award CCR-0093354, the Cambridge-MIT Institute award 093-P-IRFT(MIT), PERCS project W0133890, and a donation from the Intel corporation.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The relentless drive toward smaller, faster, and cheaper computing systems has, in large part, been enabled by exponential increases in device density and operating frequency through VLSI technology scaling. This, however, has led to exponential increases in power consumption that has reached the limits of reliability and cost effective cooling. In addition, the continued scaling into the nanometer regime has brought with it design robustness issues such as signal integrity, soft error, and environmental and process variability. Furthermore, the issues of power consumption and robustness only get worse with time. This has created, therefore, a crisis in computer system design that threatens to be a stumbling block to future advancement.

Designers of leading-edge computing systems, at any scale, are finding that power consumption and design robustness are first class constraints, and must be taken into account at every level of design. At the circuit level, the choice of logic styles is important as it directly affects power, performance, and robustness. The two prevalent logic styles, static CMOS and domino logic, do not fully meet the needs of future computing. Static CMOS, though energy-efficient and robust, is too slow to be used in timing-critical designs. Domino logic, though fast, consumes too much power and is not robust. In addition, domino logic scales poorly so that its speed advantage is lessened while its power and robustness disadvantages are worsened. We therefore require new digital logic techniques and styles that are simultaneously high performance, energy efficient, and robust to noise and variation.

We propose a new family of logic styles called Preset Skewed Static Logic (PSSL). PSSL occupies an intermediate region in the energy-delay-robustness space between domino logic and static CMOS logic. PSSL is generally better than domino in terms of energy and robustness, and is generally better than static CMOS in terms of delay. PSSL works by partially overlapping the execution of consecutive iterations through speculative evaluation. This is accomplished by presetting nodes at register boundaries before input arrival. This creates timing slack which can be traded for lower delay and/or lower energy. We also show a leakage reduction technique in PSSL that takes advantage of this slack to reduce energy-delay overhead.

Chapter 2 discusses the issues arising from scaling, in particular power and robustness. Chapter 3 describes the two prevailing logic styles: static CMOS and domino. The strengths and weaknesses of each style will be discussed and we will show that these styles need to be improved upon and/or complemented. Chapter 4 describes our novel PSSL logic. We will show its theory of operation and its correctness and derive timing constraints. Chapter 5 discusses related work and how it compares to or complements PSSL. Chapter 6 discusses ways to manage leakage and variability and proposes a leakage reduction technique for PSSL. Chapter 7 is a quantitative comparison of PSSL to other logic styles using several test circuits. Chapter 8 describes a test-chip which implements ALU cores using PSSL, static CMOS, and domino logic styles. This test chip is intended to validate the suitability of our logic style in real circuits and provide another comparison to other styles. Finally, chapter 9 summarizes the contributions of this thesis.

Chapter 2

Background - Scaling and the Challenges for future computing

Integrated circuit technology has advanced tremendously over the past 40 years, as predicted by Moore's Law [1]. Device counts have grown exponentially, from the 2300 transistors of the Intel 4004 processor in 1971, to the 592 million transistors of the Intel Itanium 2 processor in 2004. Simultaneously, clock frequencies have increased exponentially from 0.1MHz in the Intel 4004 to 3.8Ghz in currently shipping Intel Pentium 4's.

Historically, and according to predictions in the International Technology Roadmap for Semiconductors (ITRS) [2], each technology generation, which occur at 2.5–3 year intervals, brings with it a 0.7× scaling in drawn gate length as well as other layout geometry lengths. The physical gate length follows the same 0.7× scaling. Assuming a constant die size, this means a 2× scaling in device count and a 1.4× scaling in total transistor width. In addition, the intrinsic switching speed of a transistor increases at roughly 1.5× per generation.

On the other hand, power consumption has been increasing at 20% per year and has reached power density limits. At the same time, noise, from many sources, as a fraction of power supply voltage, has increased while noise sensitivity has also increased. These factors, together with increased relative process variation and environmental variation, have made predictability and robustness difficult to achieve in new designs. This chapter explains the connection between scaling and power consumption and design robustness.

2.1 Power Consumption

Power has always been one of the foremost issues in system design. No matter what the design scale, there is a direct correspondence between power dissipation and performance/functionality, battery life, cost, and size. A hand-held device, for example, must be small. There is, therefore, no room for a fan or a large battery. Similarly, a personal computer should be inexpensive; few are willing to pay for exotic cooling technologies. In fact,

high performance processors have already reached the power density limit for cost-effective cooling. All these things limit the amount of power a processing chip can burn.

The costs of power dissipation extend beyond the power used for computing. Take a data center for example. Firstly, there is, of course, the electricity bill from the computers. Secondly, there is the electricity bill and maintenance for the air conditioning system which has to remove the heat due to power dissipation. Finally, thermal concerns dictate a maximum power density of a system; in other words, the more power a system burns, the more space it must occupy. Therefore we must add in the rent for the space occupied by the system. In all, one account calculates power dissipation at 25% of the total cost of a data center [3].

Chip power can be divided into two main components: dynamic switching and static leakage. Dynamic power dissipation, ignoring short-circuit current which is usually a small fraction of total dynamic power, is given by $P = \frac{1}{2}CV^2f$, where C is the average total on-chip capacitance switched per cycle. Up until recently, VLSI scaling could be counted on to alleviate the power problem. Ever since the $0.5\,\mu$m generation, the gate dielectric oxide thickness, supply voltage, and threshold voltage have scaled with device dimensions by $0.7\times$ per generation to limit the growth of dynamic power consumption while improving performance.

This, however, is only half the power story. The reduction of oxide thickness and threshold voltage has led to exponential increases in static leakage power. There are six leakage mechanisms in nanometer scale transistors [4], of which the three most significant are subthreshold leakage, gate leakage, and band-to-band tunneling (BTBT) leakage [5]. These are indicated in Figure 2-1. Subthreshold leakage is the current flowing from drain to source (or vice versa) when the transistor is nominally off. This current is inversely exponentially proportional to the transistor's threshold voltage and has therefore grown exponentially. Gate leakage is the current flowing from the gate to the source, drain, or bulk (or vice versa). This is caused by direct tunneling of electrons or holes through the oxide insulator. This current is inversely exponentially proportional to the transistor's oxide thickness, leading to the exponential increase in gate leakage. Band-to-band tunneling is the current flowing through the reverse-biased drain/substrate and source/substrate junctions. This current is exponentially proportional to the doping concentrations on either side of the junction, which have also increased in scaled devices, leading to the exponential increase in BTBT leakage. Subthreshold leakage was the major component of total leakage at technologies larger than 130nm (drawn gate length). However, below 130nm gate leakage dominates. At 45nm gate leakage is about 10 to 100 times larger than subthreshold leakage, depending on temperature. BTBT leakage is the most affected by scaling. BTBT leakage is insignificant at 130nm, is on the same scale as subthreshold leakage at 90nm, and is on the same scale as gate leakage at 45nm [6, 7]. Further, all leakage sources are directly proportional to total transistor width, which increases by $1.4\times$ in each technology generation.

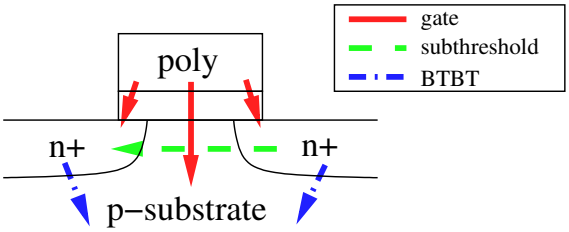


Figure 2-1: Major transistor leakage paths.

There is, therefore, a trade-off between dynamic and static power consumption in choosing voltage levels. Further, leakage power, which was once insignificant, has grown such that leakage and dynamic power are now of approximately the same magnitude [8]. The result is that the scaling of supply and threshold voltages slowed in the 130nm node and voltages have essentially remained flat since the 90nm node. Power has thus become a stumbling block to further scaling. It is impossible to continue simultaneously increasing the active device count and clock frequency while maintaining constant power envelopes if we only relay on scaling and device engineering.

Power dissipation has become such an issue that Intel has changed course on their microprocessor roadmap. Intel had previously sought performance through deep pipelining and high clock frequencies, as in the Pentium 4, without regard to power dissipation. This resulted in power dissipation that reached the absolute limit of cost-effective cooling, and left Intel with no strategy for scaling to higher performance. Intel subsequently switched to seeking balanced power and performance, utilizing greater parallelism with shorter pipelines and lower clock frequencies as in the Pentium M [9].

2.2 Robustness

Robustness is the measure of a design's tolerance to uncertainty. This uncertainty comes from various sources, most importantly from signal noise, single event phenomena (SEP), and variability.

2.2.1 Signal noise and signal integrity

Within a chip, signals are not the nice 0's and 1's of the digital abstraction; real signals have noise. Dealing with this noise is the signal integrity challenge. Signal integrity problems manifest primarily in two ways. Firstly, they can directly cause state, such as dynamic nodes, latch nodes, and memory nodes, to be corrupted, causing incorrect computation. Secondly, they can add significant and unexpected delay. This also causes incorrect computation if the delay is not accounted for in the clock cycle budget. Signal integrity problems can be hard to detect because they are data dependent. In order to safeguard against signal

integrity issues, designers often add extra safety margin, negatively affecting performance and power.

VLSI scaling has made signal integrity critical for a variety of reasons. Clock frequencies and total power draw have increased exponentially in time, leading to large power supply current transients and thus significant noise on power and ground due to resistance and inductance. Techniques to reduce power consumption, such as clock and power gating, further exacerbate the noise problem, creating a new source of noise at different fundamental frequencies from those caused by clocking. Moreover, as technology scales, wires are packed closer together and become relatively longer to connect to more and more devices. Accordingly, coupling capacitance has grown drastically relative to device parasitics so that switching activity on wires has a greater noise effect on neighboring wires. Finally, sensitivity to noise on signal and supply nets has increased because of reduced threshold voltages and supply voltages.

2.2.2 Single Event Phenomena and soft errors

One issue affecting the reliability of computing systems is soft errors. Soft errors are the result of SEP, spatially and temporally random events such as the collision and absorption of high-energy ionizing particles. An SEP manifests itself as a Single Event Upset (SEU), which is the flipping of a state node (RAM, latch, or dynamic node), or as a Single Event Transient (SET), a transient noise pulse that travels through logic and might be captured by a memory. Both SEU and SET can lead to soft error.

Soft errors have long been a concern for memory; their prevention requires the addition of error-correcting-codes (ECC) to the memory. Soft errors have not been a concern for logic because of the larger capacitances found on logic nodes. However, scaling has made soft errors more of a problem because of the reduced energy (proportional to CV^2) at each node. ECC can also be applied to protect logic from soft error; however, this comes at a large area, energy, and delay cost.

2.2.3 Variability

The cost of producing a chip is inversely proportional to the chip yield, that is, the fraction of chips that meet specifications. Chip yield is threatened by device variability. Because of geometry scaling, even tiny absolute deviations in the structure of a transistor represent large relative deviations. The gate oxide, for example, will be only 4 atomic layers high in the 45nm generation scheduled for 2007. Also, as transistor area decreases, the total number of dopant atoms and defects become small. Even the presence or absence of one atom, and its exact location, makes a big difference. Finally, gate length is difficult to control because gate length is so much shorter than the wavelength of light used in the lithography and, in addition, the diffusion of dopants is imprecise.

Leakage current is particularly sensitive to variation because many of its components have an exponential relationship to the aforementioned factors. For example, NMOS transistors in the TSMC 65nm process show about a 1000× variation in I_{off} [10]. Statistical models have shown that 90nm NMOS devices at 300° K display 210%/31%/48% σ/μ variation in subthreshold/BTBT/gate leakage respectively for a 10% 3σ variation of all process parameters [11]. PMOS devices are even more sensitive to process variation. Subthreshold leakage is a strong factor in determining the noise margin of a gate.

Gate delay is also affected by variability, but to a much smaller extent. Rao et al. [12] show gate delays vary by $\pm 15\%$ for a $\pm 3\sigma$ variation in gate length.

Besides process variability, temperature variability is a concern. Most chips dissipation power unevenly throughout the chip, leading to local hotspots. In addition, the locations of hotspots are not entirely predictable as they depend on activity. Temperature has a strong influence on gate delays and on subthreshold leakage.

2.2.4 Improving Robustness

The conventional solution to improving robustness has been design margining, that is, designing for the worst case. This, however, has large energy-delay cost and becomes infeasible as relative uncertainty increases due to scaling. More accurate statistical modeling and analysis has mitigated, but not eliminated the overhead. Also, design margining does not help with soft errors.

More recently, the notion of Better Than Worst-Case Design [13], typified by the architectural technique DIVA [14] and the circuit technique Razor [15], has been proposed to significantly improve robustness. In DIVA, the functionality of critical pipeline feedback loops, such as the fetch-execute loop in a microprocessor, is duplicated outside the critical loop. The outputs of the original block and the duplicate block are compared prior to committing the results to ensure accurate computation. Since the duplicated block is outside any critical loops, its latency is unimportant and it can be designed purely for robustness. DIVA, being an architectural technique, can be used with any logic style. However, it does have significant energy and area overhead, and is limited in its scope.

Razor is a fine-grained technique; each latch or flip-flop is duplicated outside the normal execution path. The data is sampled by the duplicate latch or flip-flop usually half a cycle later. The outputs are then compared. In the case of mismatch, a bubble (or bubbles) is inserted in the pipeline and the cycle is repeated. This allows the pipeline to recover from unexpected timing delay from noise and even Single Event Transients. Further, Razor can exploit data dependent delay variance. Worst case constraints only need to be met for the shadow latch. This allows the clock to be run at a higher frequency than normally possible. Razor has low overhead and wide applicability; however, it is not compatible with all logic styles. Better Than Worst-Case Design techniques may well have to be employed in scaled technologies because of the increased effect of timing unpredictability and soft error.

2.3 Conclusion

Power and robustness are so critical to leading edge designs that they need to be addressed at every level of design. At the circuit level, the choice of logic styles is important. Logic styles differ in terms of energy, delay, and robustness. Traditionally, logic styles have been judged purely by energy, or purely by delay, or, at best, a combined energy-delay metric. However, because every design requires compromises and trade-offs, designers need to pick and choose circuits from different points on an energy-delay-robustness envelope to meet each circuit need. Meeting the needs of future computing will require, among other things, logic styles that combine high-performance, low-power, high-robustness in the face of noise and variability, and ease of implementation and verification. In addition, we want to use logic styles that are compatible with techniques such as Razor to further improve robustness. In the following chapters we'll show why existing logic styles do not meet these needs and how PSSL can fill the void.

Chapter 3

Background - Logic Styles

There two most common basic logic styles are static CMOS and domino (Figure 3-1). In this chapter, we review these two styles and show why they fall short in meeting the energy, delay, and robustness requirements of future computing. A third logic style, Pass Transistor Logic (PTL), is qualitatively similar to static CMOS and can be lumped together with static CMOS for the purposes of this work.

3.1 Static CMOS

A static CMOS logic network is composed of static CMOS gates which are a combination of two networks: a pull-up network, consisting of PMOS transistors, connected to power, and a pull-down network, consisting of NMOS transistors, connected to ground. The networks are constructed such that exactly one of the networks is conducting for any set of inputs. Static CMOS is a universal logic – any logic function can be implemented.

Static CMOS logic is common in ASIC design, where the extra design cost of higher performance logic is often not justified by the relatively low volumes, and where ultimate performance is often not required. However, it can also be found in portions of even the highest performing microprocessor designs, often in non-timing critical circuits or in circuits that cannot be implemented in domino logic.

The appeal of static CMOS logic is its simplicity. The gates are generally relatively easy to lay out. There are no clocks and no feedback involved. The simplicity of static CMOS generally leads to relatively low power dissipation, especially for low fan-in gates. One can, for the most part, ignore transistor width ratios, even sizing altogether, and still obtain a working circuit. Because of this, static CMOS logic is robust to process and environmental variation.

In addition, the gates can recover fully from transient noise. Even if there is a significant noise pulse (from any source including SEP) that flips the output node, the node and all downstream combinational logic is eventually restored to the proper levels. A static logic pipeline run slowly enough (or stopped) is thus immune to soft error from transient noise.

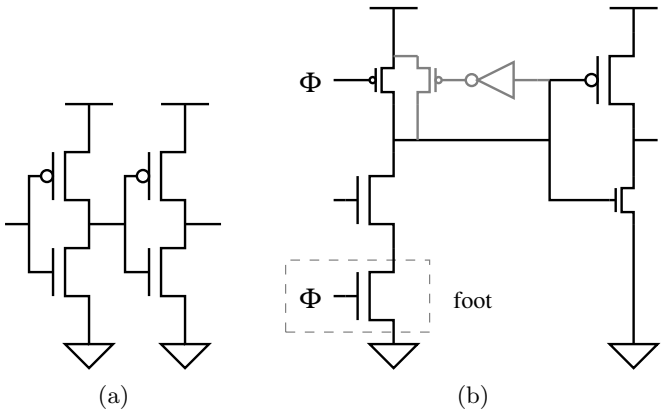


Figure 3-1: Basic logic styles. (a) Static CMOS. (b) Domino. The feedback keeper on the dynamic node, shown in gray, provides noise rejection. Smaller transistors are less critical. The clocked transistor on the NMOS pull-down is not present in the footless variant.

Even better, a pipeline using static gates and Razor latches [15] is also immune to soft error from transient noise (even running at full speed), with little timing overhead and, at the same time, gets speedup from exploiting data-dependent delay variance.

The problem with static CMOS is that it performs too poorly for the most aggressive designs. A full pull-up and pull-down chain are required, meaning that any function requires at least 2 transistors per input. In addition, it is not very efficient in implementing circuits such as `XOR/XNOR`, wide-fanin `NOR`, or binary encoded multiplexers, requiring an exponential number of transistors and/or a n transistor pull-up chain for n inputs.

3.2 Domino

A domino logic network [16] is composed of alternating dynamic and static CMOS gates. In a dynamic gate, the PMOS pull-up chain found in a static CMOS gate is replaced with a clocked pull-up transistor, reducing the input load by a factor of $1 + r$, where r is the PMOS to NMOS width ratio. When the input clock is low, the dynamic output node is precharged high. When the input clock rises, the gate evaluates, conditionally discharging the dynamic node. If the node does not discharge, the feedback keeper maintains the high value at the dynamic node.

Domino logic is frequently chosen for high-speed design because of the higher performance of dynamic gates. An indication of the relative performance of static CMOS and dynamic gates can be found in the theory of Logical Effort [17]. Logical Effort is the measure of output drive divided by input capacitance, relative to an inverter. The contribution of a gate to the total delay of an optimal path is shown to be proportional to the logarithm of the logical effort of the gate. The logical effort of common gates are shown in Table 3.1. A dynamic logic gate generally outperforms the equivalent static CMOS logic gate because

the logical effort for each gate is lower.

	Logical Effort		
Gate Type	Static	Footed dynamic	Footless dynamic
`INV`	1	2/3	1/3
`NAND`	$(n+2)/3$	$(n+1)/3$	$n/3$
`NOR`	$(1+2n)/3$	2/3	1/3
one-hot `MUX`	2	1	2/3
symmetric `XOR`	$n2^{n-1}$	N/A	N/A

Table 3.1: Per-input logical effort of common gates, where n is number of inputs. The ratio of NMOS to PMOS drive is assumed to be 2. Static gates are sized to have balanced rise/fall delay.

The performance of domino logic comes at the cost of power, robustness, and design effort. Domino logic burns more power because of the increased number of transitions on the output net. Figure 3-2 shows the operation of the domino buffer when the data input is always high. Note that nodes A, B, and C switch twice on each cycle, though they are conceptually holding constant values. The nodes in a static CMOS buffer would not be switching at all. More generally, for uniformly random input data, the nodes driven by a domino buffer will toggle twice as often as the nodes driven by a static CMOS buffer, neglecting glitching activity. Note also that the feedback inverter and node B would not exist in the static CMOS version; thus there is extra capacitance being switched. In addition, domino logic presents a much greater clock load than static CMOS, and thus requires greater power to drive. Compared to data nodes of the same capacitance, clock nodes account for more power dissipation because of clock tree buffering and greater switching activity. Further, the feedback keeper also burns power and degrade performance because of contention when the dynamic gate switches. This is shown in Figure 3-2. After precharge, the dynamic node is high, meaning that the feedback keeper is on. When both the data input and the clock go high, the pull-down chain tries to drive the dynamic node low. The pull-down chain has to fight the keeper until the keeper itself changes state. The amount of contention, and hence delay degradation and power waste, depends on the relative sizes of the transistors in the dynamic gate and the feedback keeper.

Another complication is that some form of latching is required in between the clock stages. This is because the final domino precharge in the stage erases the output right when the domino gates in the next stage begin to evaluate. Without a latch, the NMOS chain will not have enough time to fully pull down. The latch (or half-latch) captures the data before the falling edge of the clock that triggers preset, giving evaluate time to work. Extra clock phases or non-50% duty cycle clock waveforms can also solve this issue, at the cost of some complexity.

The absence of a static pull-up chain makes a dynamic gate susceptible to input noise, power and ground bounce, leakage, charge-sharing, and SEP during the evaluate phase if

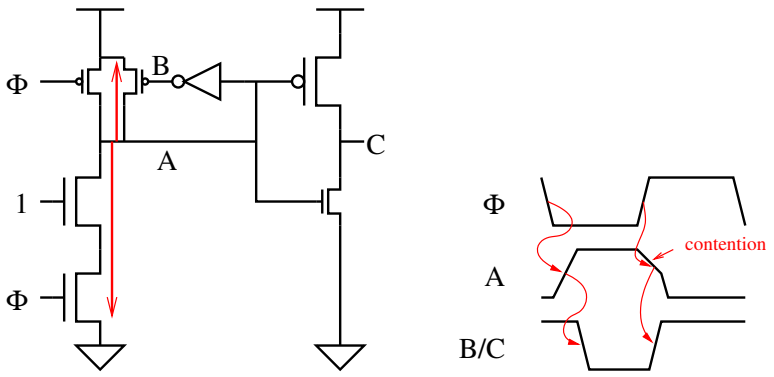
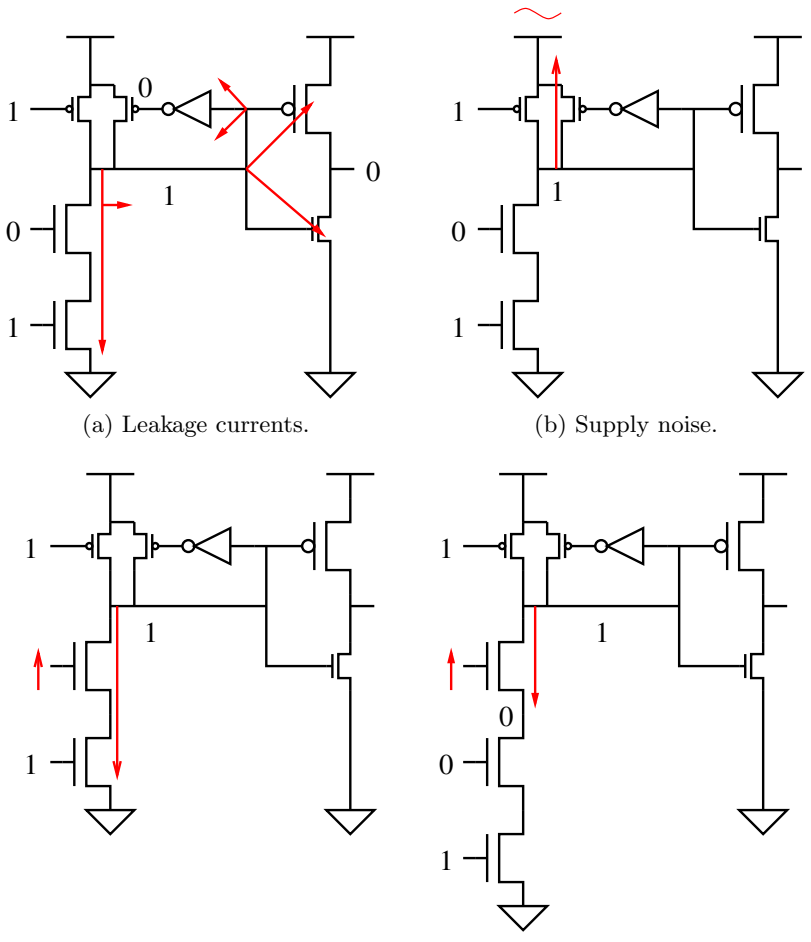


Figure 3-2: Domino switching and contention. Nodes A/B/C always toggle even though the input is held steady. Contention occurs whenever the evaluate stack begins to pull down.

the outputs are not being pulled down (Figure 3-3). Without the feedback keeper in these circuits, the gates would have zero noise rejection and the dynamic nodes will discharge completely given enough time. The feedback keeper placed on the dynamic node maintains the charge on that node, giving the gate some degree of noise-rejection. The noise rejection capability of the circuit depends on the relative sizes of the transistors in the dynamic gate and the feedback keeper. However, note that if the dynamic node incorrectly discharges past a certain point, the result is irreversible and incorrect computation will result; the Razor technique [15] is thus ineffective in improving robustness in domino logic. However, the Razor technique can still be applied to improve performance by exploiting data-dependent delay variance.

In addition, the static gate that follows the dynamic gate in domino logic has a profound influence on the overall noise margin. The static gate tends to be heavily skewed to favor rising transitions, since the falling transition is not a factor in performance. This, however, compounds the heavy skewing in the dynamic gate, resulting in diminished noise margins. In order for domino logic to maintain good noise margin, designers must sacrifice power and performance by increasing the size of the PMOS keeper and the following NMOS chain.

The sizing of the feedback keeper in a domino gate is critical. Figure 3-4a shows the delay vs. keeper ratio (P/N width) of a PTM [18] 32nm 2-input footless `NOR` gate at 0.6V, 100° C. We observe a super-exponential dependence of delay on the keeper ratio. Beyond a ratio of about three, the gate fails to evaluate. Figure 3-4b shows the noise margin (DC large-signal unity gain input level) vs. keeper ratio. There is an exponential dependence of required keeper ratio vs minimum noise margin. We see, then, that we can have large delay and large noise margin, or small delay and small noise margin. There is always a trade-off. The maximum acceptable delay sets the upper bound on the keeper ratio, while the minimum acceptable noise margin sets the lower bound on the keeper ratio. In fact, the trade-off is extremely sensitive; the delay is super-exponentially dependent on the keeper ratio which is again exponentially dependent on the target noise margin. Further, the trade-



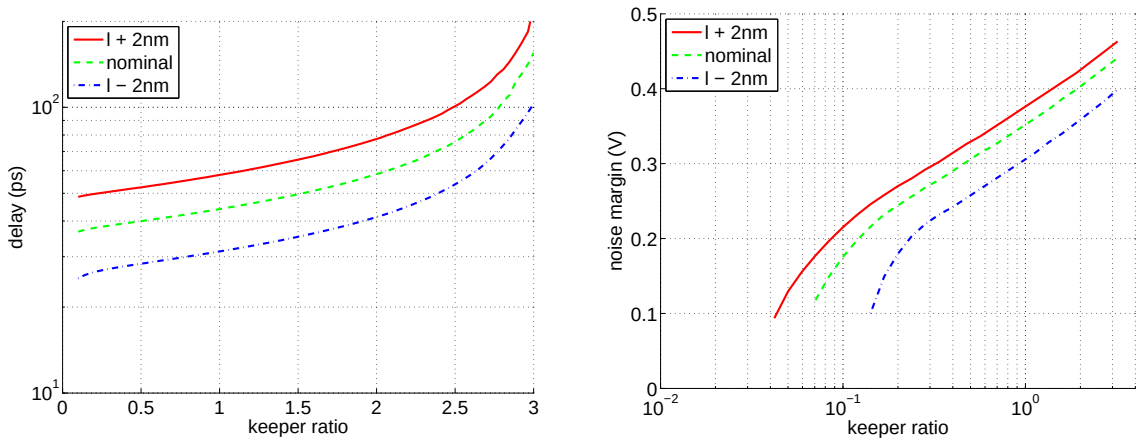
(a) Leakage currents.

(b) Supply noise.

(c) Input noise.

(d) Charge sharing.

Figure 3-3: Sources of noise in domino logic. SEP not shown.



(a) (b)

Figure 3-4: Keeper sizing for a PTM 32nm 2-input footless dynamic NOR gate at 0.6V, 100° C. Channel lengths are varied by ±2 nm. (a) Delay vs. keeper ratio. One input varying with 25ps input rise time. Second input constant 0. 740nm evaluation width. 20fF load. (b) Noise margin vs. keeper ratio. Both inputs varying.

off between performance and noise margin gets worse with scaling. The acceptable range of keeper ratios shrinks as technology scales [19].

To make matters worse, domino logic is sensitive to variation. Figure 3-4 also shows the keeper sizing curves when the transistor channel lengths are varied by ±2 nm, corresponding to approximately 3σ process deviation [2]. Correct operation requires delay to be verified at the slow corner and noise margin to be verified at the fast corner. This further shrinks the range of acceptable keeper ratios. As relative device variability increases, this effect will become larger.

Footless domino pipelines require separate delayed clocks for each stage of logic, each separated by two slow inverters. The timing constraints for the clocks are complex. The rising edge of each stage's clock should precede the rising edge of data for performance. The falling edge of each stage's clock should follow the falling edge of data to prevent contention. This careful clock shaping is also sensitive to variation. The clock generation requires extra area and power and, because of their complexity, are appropriate only in critical datapaths and in wide-or structures such as register files where the clock delay circuitry can be amortized across many gates in a stage. These schemes also require careful analysis. One must account for process and environment variation to ensure accurate tracking of clock and data delays.

Because of the keeper sizing issue, the delay/robustness trade-off, and variability concerns, it is not clear how well dynamic circuits will scale into the nanometer regime. A study of technology scaling on CMOS Logic styles was performed by Anis et al. [20]. They showed that, for the technology nodes from 0.80 μm to 0.25 μm, the performance advantage of domino logic over static logic is reduced. Another analytical study by Anders [19] showed

that if we hold noise margins to a constant fraction of the supply voltage, the performance of dynamic circuits are severely degraded at the 70 nm node, and conventional dynamic circuits cease to function below 70 nm. This last prediction, however, was flawed because it presumed voltage scaling would continue. Nevertheless, domino circuits will continue to face serious noise and scaling issues.

In addition to power-performance-robustness scaling issues, domino logic requires additional design effort because of complex intra-cell routing and routing to reduce noise. Finally, domino logic can only implement non-inverting logic functions, which limits it use. Variations on domino logic such as dual-rail domino, or use of deracers or complementary signal generators [21], enable inverting logic, but at the cost of increased power and area.

3.3 Conclusion

Static CMOS logic and Domino logic occupy very different points in the energy-delay-robustness space. Static CMOS is good in terms of energy and robustness, but is poor in terms of delay. Domino is good in terms of delay, but is poor in terms of energy and robustness. In particular, it cannot take advantage of the Razor technique for robustness against transient noise. Finally, domino has serious scaling issues. In the following chapters, we show how PSSL combines the best features of static CMOS and domino.

Chapter 4

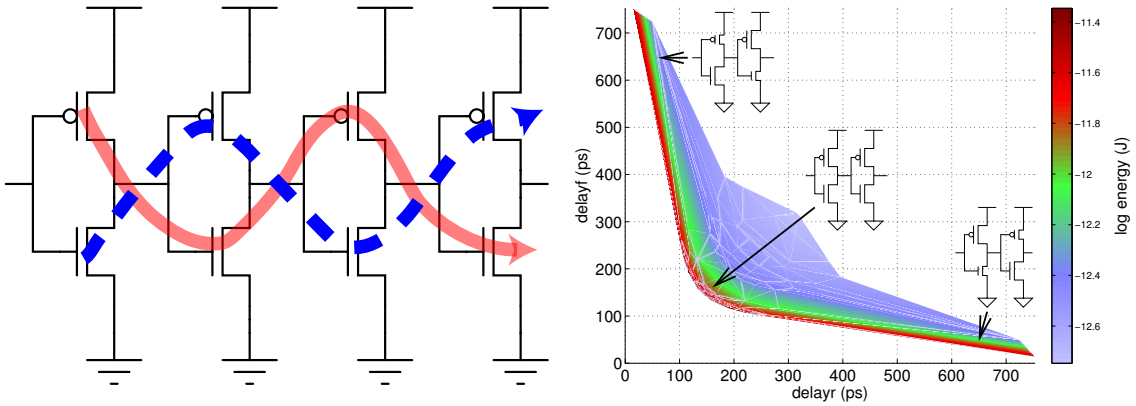
Preset Skewed Static Logic

In this chapter, we present Preset Skewed Static Logic (PSSL). PSSL combines the energy-efficiency and robustness of static CMOS logic with the performance of domino logic. We first show how Skewed Static Logic can improve performance in the presence of timing slack. We then show how to generate slack through preset. We then show the implementation of PSSL logic and PSSL pipelines. Finally we discuss various scaling issues with respect to PSSL.

4.1 Skewed Static Logic

Figure 4-1a shows a chain of four static CMOS inverters. The dashed curve indicates the transistor activation path, that is, the sequence of transistor chains that are turned on, for a rising input transition. The solid curve indicates the transistor activation path for a falling input transition. Note that the total path delay times of a rising input and that of a falling input are not necessarily the same. There is a trade-off between the two delay times and also between delay and energy; this is controlled by varying the sizes of individual transistors. For example, by increasing the size of transistors under the dashed curve, one can speed up the response of the circuit to a rising input transition. This comes at the cost of a slower response to falling input transitions and increased energy dissipation. Figure 4-1b shows this trade-off. The plot shows the energy and delay of two inverters within a long fan-out-of-4 (FO4) chain. The X and Y axes represent delays through rising input and falling input paths. The shade at each x,y location indicates the required energy dissipation to achieve the delays. Note that the shade axis is logarithmic.

More generally, consider any multiple-input, multiple-output acyclic combinational circuit. There can be many activation paths. If there is any difference in the delay times between different paths, then there is slack. By appropriately resizing transistors, one can often use slack to either increase performance or reduce power dissipation.



(a) (b)

Figure 4-1: (a) Inverter Chain (b) Energy-Delay. TSMC 0.18 μm process. FO4 configuration. 10.4fF wire load.

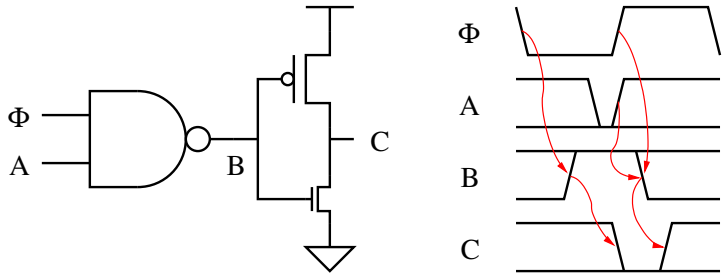


Figure 4-2: Preset Skewed Static Logic. Smaller transistors are less critical.

4.2 Preset

A simple PSSL circuit is shown in Figure 4-2. This resembles the chain of static inverters in Figure 4-1a, except that the first inverter has been replaced by a `NAND` gate with one input tied to the clock. The logical function of this circuit is the same as the inverter chain.

Let us assume that the input `A` is expected to arrive at the rising edge of the clock. The operation of this circuit as is follows. First, the falling edge of the clock initiates the process of *preset*. In preset, all circuit nodes are indirectly forced to pre-determined values. In particular, node `B` rises in turn causing node `C` to fall, thus completing the preset process. The idea behind the preset process is that we are speculatively computing all the nodes of the circuit presuming low input values. This begins one clock phase before the actual input value(s) arrive, so this computation has an extra clock phase to complete.

The rising edge of the clock initiates the process of *evaluate*. Note that the process of evaluate is independent of the process of preset, and, in particular, evaluate can begin before preset completes. If the value of the input node, `A` is low at the rising edge of the

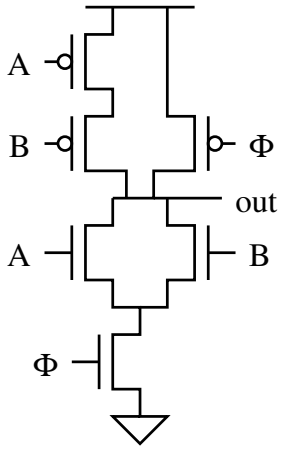


Figure 4-3: A 2-input NOR embedded in PSSL preset-high circuitry.

clock and remains low, nothing further happens in the circuit and evaluate is complete. However, if the input node, A, is high when the clock rises or node A rises while the clock is high, then it causes node B to fall, in turn causing node C to rise, completing the evaluate process.

Whether node A is high or low, eventually node C gets the correct value. However, we have decoupled the computation for low values of A (the preset process) from the computation for high values of A (the evaluate process), giving the former computation extra time and thus creating slack in the path of transistors in the preset process (i.e. the preset path). We can take advantage of this slack by reducing the size of transistors in the preset path to reduce power consumption, or by increasing the size of transistors in the evaluate path to reduce delay. Preset allows PSSL to outperform generic static CMOS logic. However, preset comes at the cost of extra power consumption because of spurious transitions from input mis-speculation and extra clocking overhead.

A NAND gate was used to preset nodes high. A similar analysis holds if we use a NOR gate for preset. This time, preset is initiated by the rising edge of the clock, and node B of Figure 4-2 is preset low.

One can embed logic into the gate used for preset. Figure 4-3 is an example of a 2-input NOR gate folded into a preset-high gate. For logic embedding, it is usually preferable to use preset-high since only a single NMOS transistor is added to the NMOS chain of logic gate. For preset-low, a PMOS transistor is added to the PMOS chain of the logic gate. The latter has a larger energy-delay impact.

4.3 Unateness

Speedup from preset depends upon the decoupling of the computation for high and low inputs. This requires that the preset and evaluate paths go through distinct sets of transis-

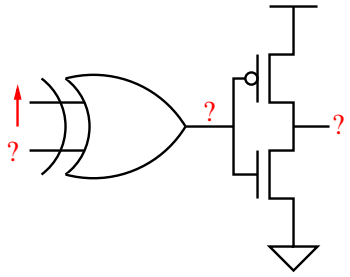


Figure 4-4: Non-unate logic. Since the bottom input of the XOR gate cannot be known *a priori*, a rising input at the top input produces an unknown transition at the XOR gate output, producing unknown transitions for all downstream logic.

tors. If the paths coincide at a transistor, the worst case timing path will apply, reducing the benefits of preset. The paths are distinct only if the boolean logic function being implemented is unate, meaning that any particular input transition in any particular direction can only cause the output to transition in one direction. In others, it is always inverting or non-inverting. Figure 4-4 shows a non-unate logic network, consisting of an XOR gate and an inverter. Since the bottom input of the XOR gate cannot be known *a priori*, a rising input at the top input produces an unknown transition at the XOR gate output, producing unknown transitions for all downstream logic.

Even if a function as a whole is not unate, it may be composed of unate subfunctions. These subfunctions can benefit from preset. Performance can be maximized by locating non-unate blocks as far downstream from preset circuitry as possible.

4.4 Pipelining

We now examine how to create pipelines using PSSL. We present PSSL using the three major clocking schemes: level-sensitive, edge-triggered, and pulsed.

4.4.1 Level-sensitive

Level-sensitive clocking uses alternating transparent latches as timing elements. A two-phase Level-Sensitive PSSL (LS-PSSL) pipeline, shown in Figure 4-5, is the composition of PSSL pipeline stages of alternating phase, separated by transparent latches. One stage begins preset when adjacent stages begin evaluate. In LS-PSSL, the transparent latches serve two purposes. First, they hold pipeline state. Every legal (non-wave pipelined [22]) pipeline must have at least one latch in each full pipeline stage. Second, the latches prevent the preset wave-front from propagating to the following stage until after the preset phase. Otherwise, if the wave-front propagates early, it will cause inter-symbol interference as it becomes indistinguishable from the evaluate wave-front from the previous cycle. However, in contrast to their use in static CMOS pipelines, transparent latches are *not* used for

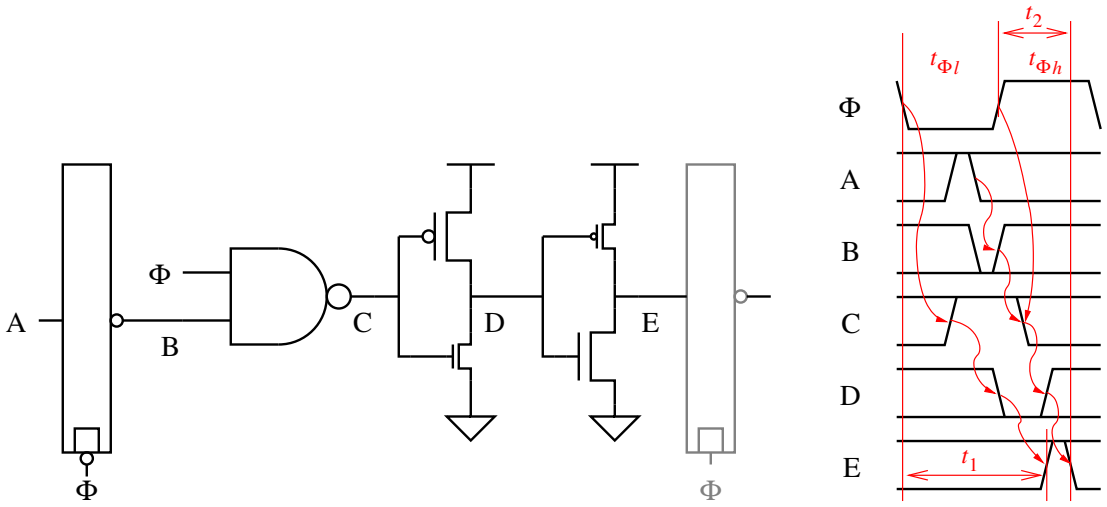


Figure 4-5: Two stage Level-Sensitive PSSL pipeline and timing diagram. Only half of the pipeline is shown. Smaller transistors are less critical. 50% duty cycle clocks are assumed.

synchronization (i.e. delay). Every legal pipeline must have a total of exactly one cycle of delay in each full pipeline stage. In LS-PSSL, the synchronization is performed by the `NAND` gates.

The operation of LS-PSSL, shown in Figure 4-5, is as follows. The falling edge of the clock begins preset, causing `C` to rise, `D` to fall, and finally `E` to rise. This path, whose delay is t_1, must complete in one clock cycle, less setup delay. This coincides with the closing of the second latch at the falling edge of the clock. Therefore we derive the constraint

$$t_1 + t_s < t_{\Phi h} + t_{\Phi l} \tag{4.1}$$

where t_s is the setup time of the latch.

The rising edge of the clock begins evaluate. The value of `A` is effectively sampled by the first latch and `NAND` gate combination at the rising edge of the clock. If it is low, then `C` falls, `D` rises, and, finally, `E` falls. This path, whose delay is t_2, must complete one setup delay before the closing edge of the latch. Therefore we derive the constraint

$$t_2 + t_s < t_{\Phi h} \tag{4.2}$$

Similarly, the equations of the other half of the pipeline (not shown) are given by

$$t_3 + t_s < t_{\Phi l} + t_{\Phi h} \tag{4.3}$$

$$t_4 + t_s < t_{\Phi l} \tag{4.4}$$

The preset path delays, t_1 and t_3 can be twice as long as the evaluate path delays, t_2 and t_4.

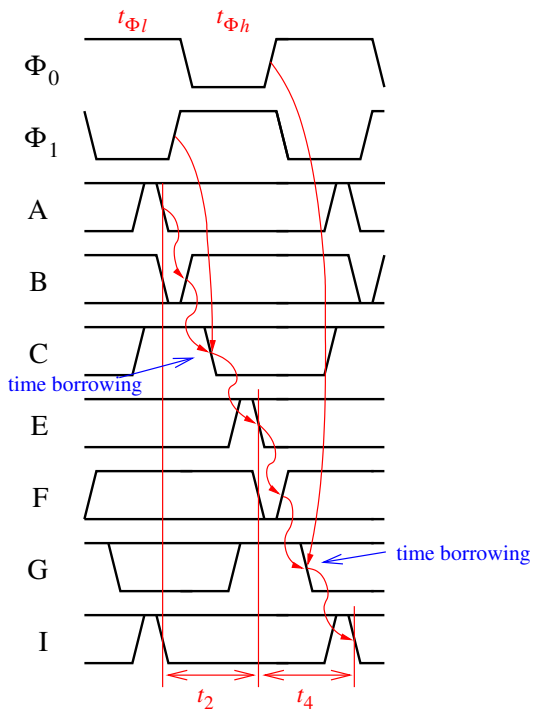


Figure 4-6: Two stage Level-Sensitive PSSL pipeline timing with overlapping clocks.

Note that there is a hard edge on every constraint. Each timing path begins at a clock edge and ends at a clock edge. This means that clock uncertainty, or jitter, needs to be considered in maximum delay timing analysis. Even worse, for t_2 and t_4, the jitter needs to be subtracted from each phase, not just each cycle. However, if we allow the clock waveforms to have greater than 50% duty cycle so that the clocks overlap, then limited time borrowing is allowed as data can flow through the NAND gate before the preceding latch closes[23] (Figure 4-6). Therefore, in this case, jitter does not need to be taken into account for maximum delay analysis. The equations 4.2 and 4.4 are thus replaced by

$$t_2 + t_4 < t_{\Phi h} + t_{\Phi l} \tag{4.5}$$

A problem with using overlapping clocks is that the latches no longer prevent the preset wave-front from advancing early. Therefore, all preset paths must meet a minimum path delay in order to guarantee correct operation, as if the circuit were wave pipelined. These constraints are given by

$$t_{\Phi l} < t_1 \tag{4.6}$$

$$t_{\Phi h} < t_3 \tag{4.7}$$

We can also resolve this by using $\overline{\Phi_n}$ for the latch clock instead of Φ_{n-1}. This allows time borrowing on the evaluate path, but not time borrowing on the preset path.

Latch and preset implementation

Table 4.1 shows eight implementations of the latch and preset gate for level-sensitive PSSL. Row (a) is an implementation of Figure 4-5. Since the latches are used in PSSL mainly to prevent the preset from interfering with the evaluation of data from the previous cycle, the latches can be simplified under certain conditions. These simplified variants are shown in rows (b) through (h).

The variants differ in their properties. This is shown in Table 4.1. The preset/precharge input constraints column indicates the required logic level (on preset) of any input signals that are preset or precharged. The functional constraint column lists which input edges (rising or falling) must arrive prior to the rising edge of the clock, Φ_n, to guarantee the correct functionality of the circuit. The timing constraint column lists which input edges (rising or falling) must arrive prior to the rising edge of the clock for there to be any benefit from preset. The functionality of the circuit is not otherwise affected by not meeting this constraint. The time borrow column lists which input edge, if any, can benefit from time borrowing. Note, though, that a timing constraint on the same edge precludes time borrowing for all practical purposes. Finally the output preset column indicates the logic level to which the output is preset.

The variants shown in rows (g) and (h) combine the latch and preset gate into one dynamic gate. They are equivalent to the variants in rows (d) and (e), except for output inversion. The dynamic gate variants have lower energy and delay and so are usually preferable. Even though PSSL can use dynamic gates, as domino does, they are used for completely different purposes. In domino, the dynamic gates are used for their high performance. In PSSL, the dynamic gates are used to preset downstream logic. Preset is the source of speedup. Even if the PSSL dynamic gate were slower than its static equivalent, PSSL would outperform static CMOS.

The simplified versions allow time borrowing because of the partial removal of latching. An example pipeline and timing diagram are shown in Figure 4-7.

If the preset/precharge constraints are violated, the state node is not protected from corruption by preset. However, the circuit can still operate correctly if the preset paths are sufficiently slow as to not interfere with the evaluation of the previous cycle. This is a form of wave pipelining [22]. Formally, this means that one or both of the following minimum path delay constraints must be met.

$$t_{\Phi l} < t_1 \tag{4.8}$$

$$t_{\Phi h} < t_3 \tag{4.9}$$

A reasonable design, however, should ensure that these minimum path delay constraints apply to only one clock phase so that the clock can be stopped on the other clock phase for standby operation.

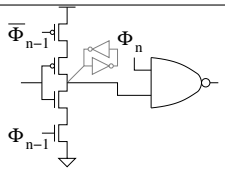
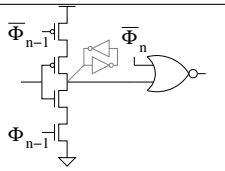
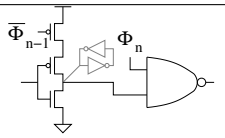
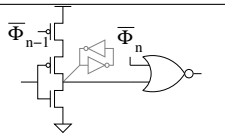
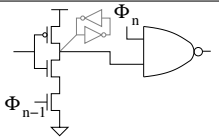
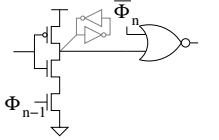
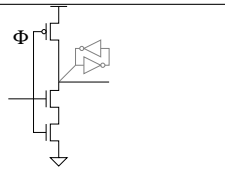
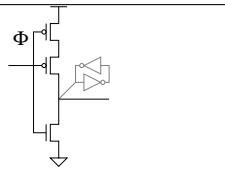
	circuit	input constraints			time	output
		preset/precharge	functional	timing	borrow	preset
a			rise/fall			high
b			rise/fall			low
c		low	fall	rise	rise	high
d		low	fall		rise	low
e		high	rise		fall	high
f		high	rise	fall	fall	low
g		low	fall		rise	high
h		high	rise		fall	low

Table 4.1: LS-PSSL Preset latches and their properties.

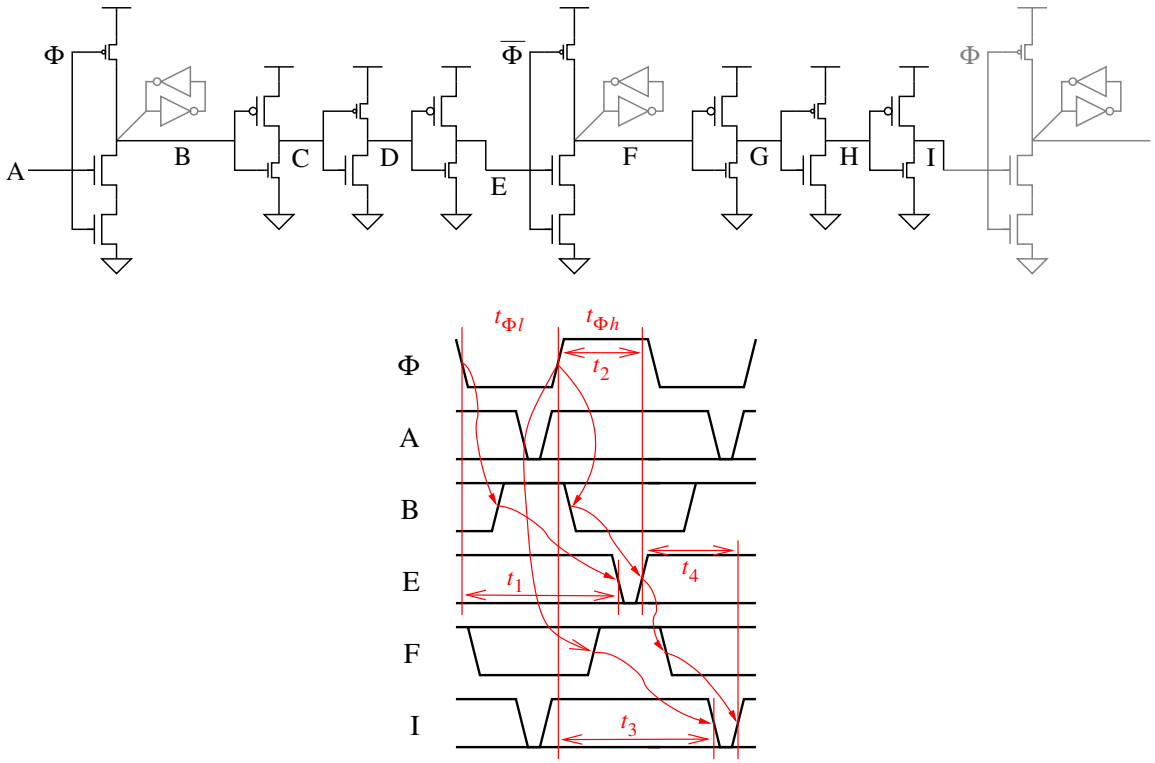


Figure 4-7: LS-PSSL time borrowing. Full pipeline shown. Smaller transistors are less critical.

One complication arises in connecting the output of 2-phase non-overlapping footed domino logic to the input of LS-PSSL. The domino output can be a narrow pulse around the rising edge of the capturing clock. The dynamic preset gates may not work correctly since they begin sampling after the rising edge. The best solution is to use one of the latch/NAND combinations, with $\overline{\Phi_n}$ for the latch clock.

N-phase clocking

It is possible to extend LS-PSSL to arbitrary numbers of clock phases, as in Figure 4-8. As with a 2-phase LS-PSSL pipeline, each preset path is allowed one cycle to complete. The evaluate paths are allowed, on average, $1/n$ of a clock cycle. This means a factor of n speedup on the preset paths. N-phase LS-PSSL, where $n > 2$, differs from two-phase in that time borrowing is allowed on the evaluate paths even when the full latch (see Table 4.1a) is used with 50% duty-cycle clocks[23] since the clocks will always overlap. There is also no preset minimum path delay problem as would occur with a 2-phase overlapping clock design since the rising edge of one clock follows the falling edge (from the previous cycle) of the following clock.

The analysis is different when using dynamic gates for the combined latch and preset (see Table 4.1g and h) since there is no separate clock for latching and preset. Figure 4-9

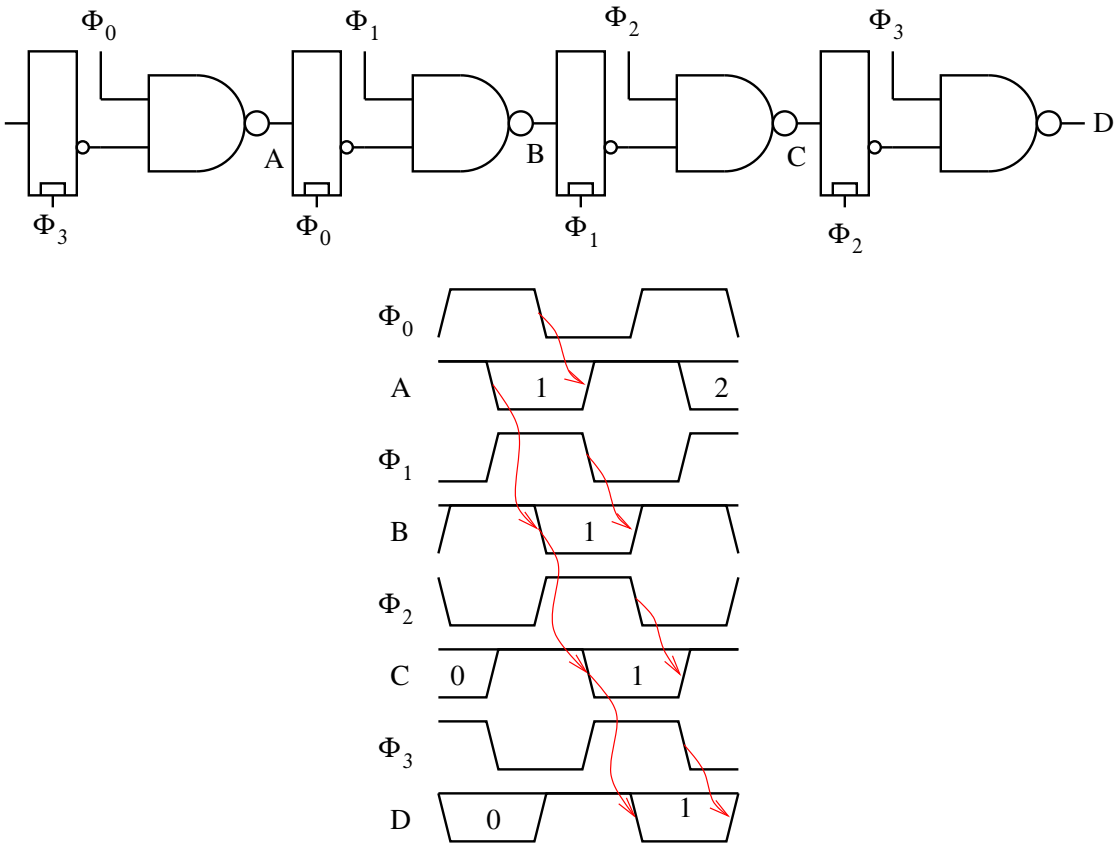


Figure 4-8: 4-phase LS-PSSL. Time borrowing is enabled by the overlapping clocks.

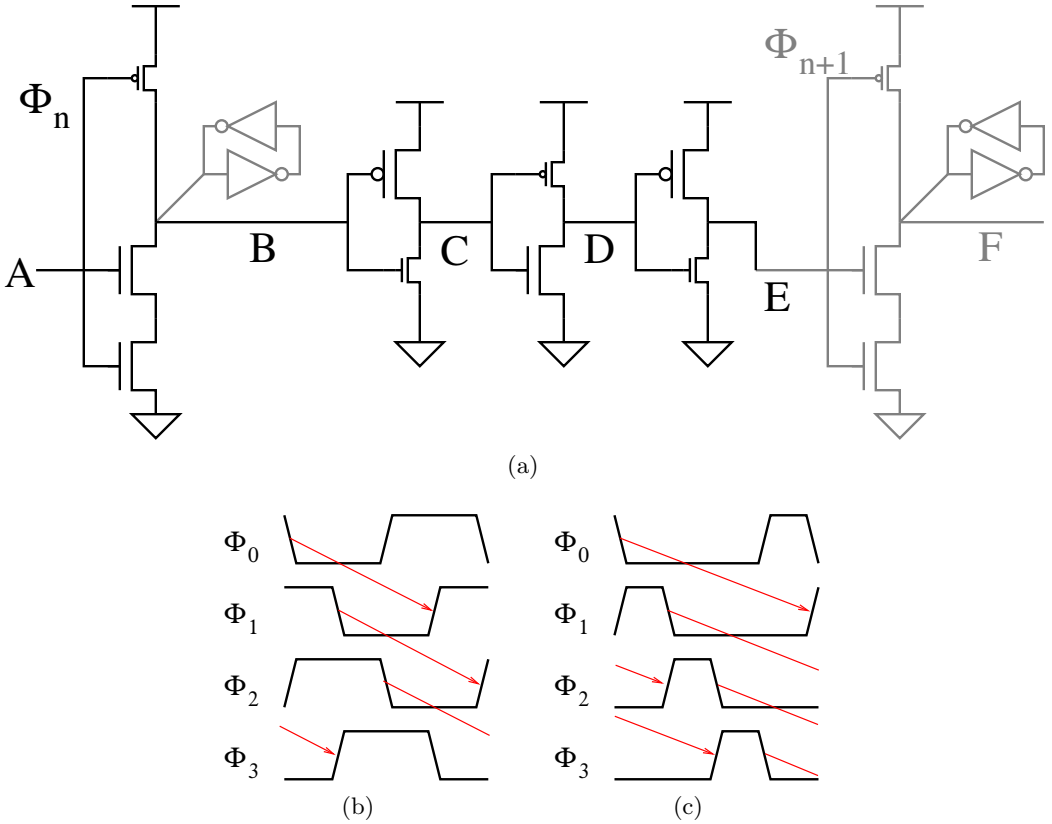


Figure 4-9: N-phase LS-PSSL using dynamic preset and clock waveforms. (a) Partial pipeline shown. Smaller transistors are less critical. (b) 4-phase 50% duty cycle clock input waveforms. (c) 4-phase 25% duty cycle clock input waveforms. Preset path timing is shown in red.

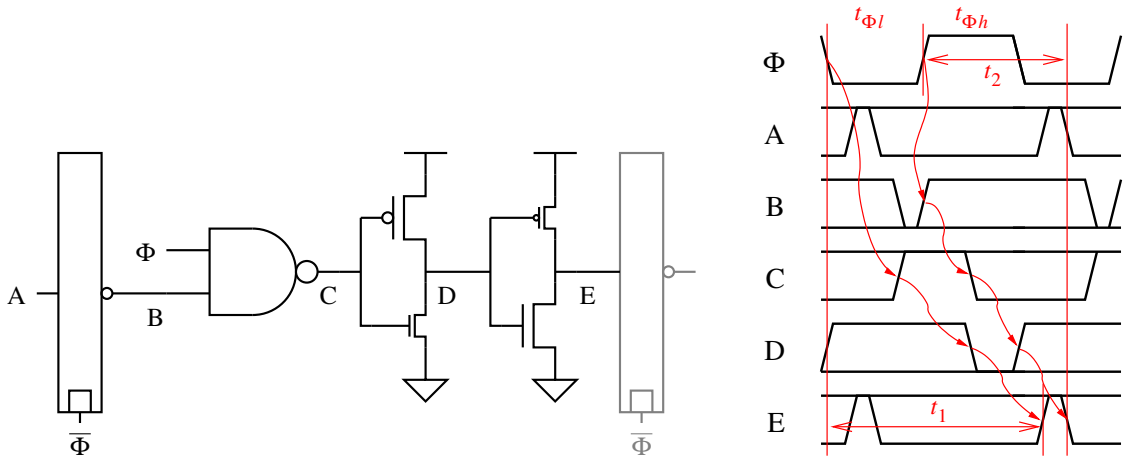


Figure 4-10: Edge Triggered PSSL and timing diagram. Full pipeline shown. Smaller transistors are less critical.

shows the pipeline using dynamic preset, along with two timing diagrams with different clock waveforms. As in the two-phase case, the preset path begins at the falling edge of the clock and ends with the rising edge of the clock to the next stage. Using 4-phase 50% duty cycle clocks, the preset paths can take 3/4 of a clock cycle, for a speedup of 3 on the preset paths. Using 4-phase 25% duty cycle clocks, the preset paths can take one whole clock cycle, for a speedup of 4. In general, n-phase 50% duty cycle clocks achieve $1 + \frac{n}{2}$ speedup whereas $1/n$ duty cycle clocks achieve n speedup.

4.4.2 Edge-triggered

As opposed to level-sensitive clocking, edge-triggered clocking uses a single monolithic timing element (usually a flip-flop). Figure 4-10 shows a pipeline using the same latch and `NAND` gate combination as before. However, this time there is only one set in a full pipeline stage, along with the diagram diagram. The corresponding timing constraints are

$$t_1 + t_s < t_{\Phi h} + 2t_{\Phi l} \tag{4.10}$$

$$t_2 + t_s < t_{\Phi h} + t_{\Phi l} \tag{4.11}$$

$$t_1 > t_{\Phi l} \tag{4.12}$$

The timing paths being and end on clock edges so that there is no time borrowing allowed. Note that there is a minimum path delay constraint on clock phase 2. Violating this constraint would cause inter-symbol interference. This is a fundamental race condition that cannot be avoided. It is impossible to have a data valid window greater than a clock cycle. The constraint means that the clock can only be stopped on clock phase 1 (clock high). With a 50% duty cycle clock, a speedup of a factor of 1.5 can be achieved for the

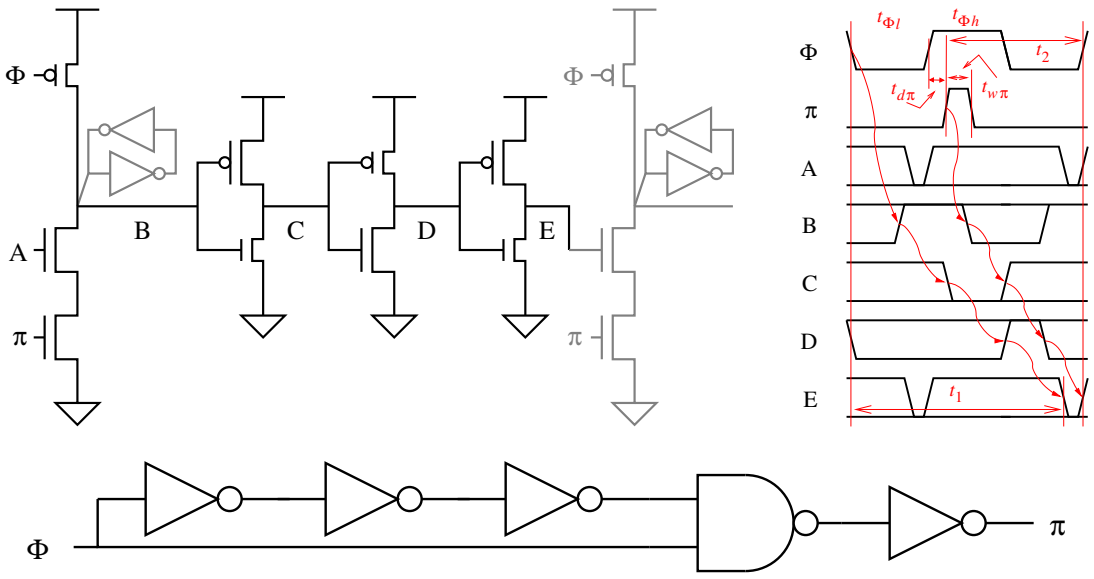


Figure 4-11: Pulsed PSSL pipeline with timing diagram. Full pipeline shown. Smaller transistors are less critical.

preset path. If the clock is a narrow pulse, a theoretical speedup of a factor of 2 can be achieved. However, finite rise/fall times and the requirements of latching impose a lower limit on the clock pulse width.

4.4.3 Pulsed

Pulsed clocking uses transparent latches that are clocked with narrow pulses. Figure 4-11 shows a pulsed PSSL pipeline along with its timing diagram. It uses a novel flip-flop structure which we call the Double Pulsed Set Conditional-Reset Flip Flop (DPSCRFF) [24]. In the DPSCRFF, the path from input to output is only a single stage of logic. This is the key to the design's high-performance. Another advantage is that the data input sees only a single transistor load which reduces required input drive and energy consumption. The pulse, π should be timed to follow the rising edge of the clock, Φ. As with the edge-triggered PSSL, there is an unavoidable race condition, the preset path must take longer than a clock phase. The timing constraints in the general case are given by

$$t_{\Phi l} + t_{d\pi} + t_{w\pi} < t_1 \tag{4.13}$$

$$t_1 < t_{\Phi h} + 2t_{\Phi l} + t_{d\pi} \tag{4.14}$$

$$t_{d\pi} + t_{w\pi} < t_2 \tag{4.15}$$

$$t_2 + t_s < t_{\Phi h} + t_{\Phi l} + t_{d\pi} \tag{4.16}$$

$t_{d\pi} + t_{w\pi}$ is the hold time of the pulsed-latch. No time borrowing is allowed in the general case.

If the intervening logic is strictly inverting, then the timing constraints are given by

$$t_{\Phi l} + t_{d\pi} < t_1 \tag{4.17}$$

$$t_1 < t_{\Phi h} + 2t_{\Phi l} + t_{d\pi} \tag{4.18}$$

$$t_{d\pi} + t_{w\pi} < t_2 \tag{4.19}$$

$$t_2 + t_s < t_{\Phi h} + t_{\Phi l} + t_{d\pi} + t_{w\pi} \tag{4.20}$$

Limited time borrowing is allowed in the evaluate path if it completes in the middle of the latching clock pulse.

If the intervening logic is strictly non-inverting, then the timing constraints are given by

$$t_{\Phi l} + t_{d\pi} + t_{w\pi} < t_1 \tag{4.21}$$

$$t_1 < t_{\Phi h} + 2t_{\Phi l} + t_{d\pi} + t_{w\pi} \tag{4.22}$$

$$t_{d\pi} < t_2 \tag{4.23}$$

$$t_2 + t_s < t_{\Phi h} + t_{\Phi l} + t_{d\pi} \tag{4.24}$$

Limited time borrowing is allowed in the preset path if it completes in the middle of the latching clock pulse.

In this pipeline, all the state is held in the DPSCRFF outputs. Since these are erased when the clock is low, the clock can only be stopped when the clock is high. Fortunately, there is no minimum path delay constraint on phase 1. One of the potential problems with pulsed clocking is the minimum path delay constraint on phase 2. However, this can usually be resolved without slowing down critical paths.

With a 50% duty cycle clock, a speedup of a factor of roughly 1.5 can be achieved for the preset path. A theoretical speedup of a factor of 2 can be achieved if the pulse is made infinitely small and is moved to the right before the falling edge of the clock. However, finite rise/fall times and the requirements of latching impose a lower limit on the pulse width. In addition, the output of the DPSCRFF becomes a narrow pulse, so there is the concern that the data pulse might actually dissipate before reaching the next stage. Fortunately, the skewing of the logic works to stretch out the pulse. However, the pulse integrity would still need to be verified across process corners.

Pulsed-PSSL and edge-triggered PSSL have similar timing properties. Assuming that clock pulse generation is not a problem, it is clear that pulsed-PSSL is superior to edge-triggered-PSSL because of the lower device count and latency of pulsed-PSSL. Therefore we do not consider edge-triggered PSSL in the evaluations.

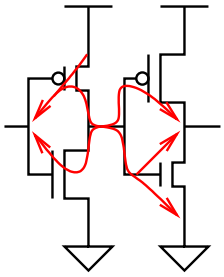


Figure 4-12: Gate leakage, indicated by arrows, in the preset state is predominantly through the smaller preset path transistors.

4.5 Leakage and leakage variability impact

Insofar as PSSL pipelines use static gates, PSSL is robust to leakage's impact on noise margin, even accounting for variability, because static gates have ample noise margin. However, because PSSL static gates will be moderately skewed, the gates will not have the same noise margin as those in a pure static CMOS pipeline. Leakage, however, does increase the amount of energy consumed in the same way as in static CMOS. We do note, however, that transistors in a PSSL pipeline, especially those in the preset path, will be smaller than transistors in static CMOS pipeline of the same performance. In addition, a node in a PSSL pipeline is statistically more likely to be in the preset state than the equivalent node in the static CMOS pipeline. In the preset state, gate leakage occurs predominantly from the transistors on the preset path (Figure 4-12). We thus expect a significant decrease in gate leakage because of the reduced transistor sizes.

Dynamic gates in PSSL pipelines face the same challenges as those in domino. However, the scale of the problems in terms of design and verification is smaller because there are far fewer dynamic gates being used. Also, the dynamic gates in PSSL will have higher noise margin than those in footed domino because the static gates following the dynamic gates will not be as highly skewed in PSSL as in domino. This is because the gates in the precharge path do not contribute to timing, except in extreme cases. This, however, does not apply to footless domino styles as the timing of the precharge paths matter. However, footless domino styles have their own issues as described in Section 3.2.

4.6 Variability impact

Variability is a serious concern for pulsed latches as it becomes difficult to predict and control the final shape of the clock pulses. Clock pulses cannot be allowed to dissipate or fail to swing fully. In order to ensure this in the face of variability, the clock buffer tree must be designed to widen the clock pulse as a safety margin as it passes through. This, of course, creates serious hold time issues. Therefore, it is not clear if pulsed-PSSL will be viable in scaled technologies.

Pulsed-PSSL, Edge-triggered PSSL, and LS-PSSL (in certain cases) make use of wave pipelining. This technique, however, becomes harder to use in the face of variability. Careful analysis of minimum clock width constraints must be performed taking into account variability on timing paths. This only becomes a problem if there is no clock frequency that simultaneously satisfies minimum clock width constraints on the fast corners and maximum clock width constraints on the slow corners.

4.7 Single Event Phenomena

PSSL shares static CMOS's ability to recover from SEP. Even the use of dynamic gates in PSSL does not increase the probability of soft error because the dynamic gate replaces what would otherwise be a latch. SEP, however, can cause transients that are later captured by state elements. As described in Section 2.2, the Razor technique can be applied to reduce or even eliminate the impact of SEP as well as other sources of uncertainty such as variability and noise.

4.8 Conclusion

PSSL is an ideal technology to cope with scaling issues. It is more robust than domino because of its reduced reliance on dynamic logic. As previously discussed, it becomes harder to use dynamic logic as technology scales because of increased noise and decreased noise-immunity. PSSL uses dynamic logic gates, not for its lower logical effort, but for its preset, which can be used to speed up all the downstream logic. Therefore, the reduced performance of dynamic logic due to scaling becomes unimportant.

Chapter 5

Previous Work and Comparison

In this chapter, we describe work that is related to PSSL. We describe in particular several Domino modifications and Domino-Static hybrids, along with two recently proposed exotic logic styles. We show that LS-PSSL is higher-performing, easier to design with, more robust, and more widely applicable than existing styles. We also describe work to improve dynamic circuit robustness to variability. Finally we discuss some work on pipelining and timing elements.

5.1 Logic styles

Conditional Keeper technique

Some variants to dynamic logic have been proposed to enable it to scale better in terms of performance [25, 26, 27, 28]. These techniques modify the feedback keeper to reduce contention. In particular, the conditional keeper technique [26], shown in Figure 5-1, has been successfully used by Intel across three process generations, from 130nm down to 65nm. Here the conventional keeper is replaced with a pair of keepers, a weak fixed keeper and a stronger conditional keeper. The conditional keeper only turns on some time after evaluate begins, giving a small window after the end of precharge for the gate to conditionally pull down. There is a problem, though, if the input data arrives in the middle of evaluate, as will happen with gates located at the end of a phase. This is resolved by generating separate delayed clocks for each stage of logic. Thus this scheme complements footless domino and shares all its design issues, and thus will likely only be used in wide-OR structures such as RAMS and register files.

Noise Tolerant Precharge

Another idea, originally called Noise Tolerant Precharge (NTP), removes the feedback keeper entirely and replaces it with a weak static CMOS pull-up [29, 30], as shown in Figure 5-2. However, a more recent study [31], showed that conventional domino was su-

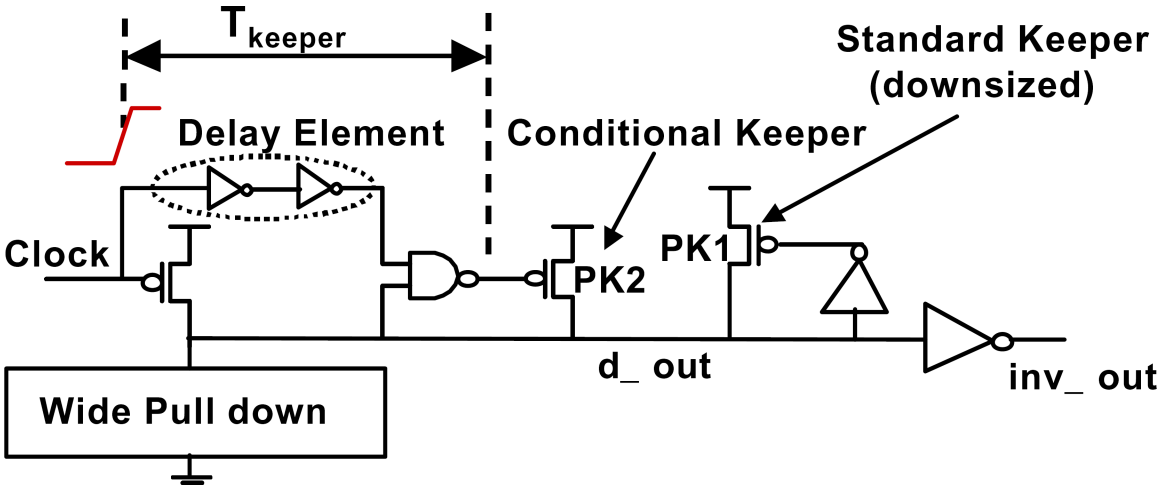


Figure 5-1: Conditional-keeper technique. The conventional keeper is replaced with a pair of keepers, a weak fixed keeper and a stronger conditional keeper. The conditional keeper only turns on some time after evaluate begins to allow time for the gate to switch.

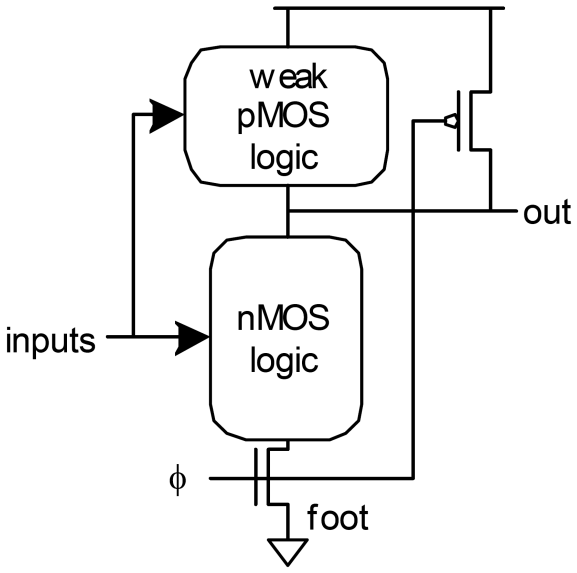


Figure 5-2: Noise-tolerant precharge. The feedback keeper is eliminated and replaced with a weak PMOS pull-up network.

perior to NTP in terms of overall energy-delay-robustness. At the same time, NTP is not a complete replacement for dynamic logic. Dynamic logic has the ability to synchronize, that is, to prevent inputs from affecting outputs. It also has the ability to hold, that is, to keep some memory of old values. These two abilities allow a dynamic gate to function as a latch. NTP, on the other hand, can only synchronize; it cannot hold a value. A pipeline using only alternating NTP and static gates will work through wave pipelining, but only at a minimum clock frequency. In particular, the clock could not be stopped to save power. Therefore, either latches have to be inserted, adding delay and preventing time borrowing, or some gates need to be converted to dynamic gates. NTP complements PSSL. It can be used in place of a static CMOS gate where the inputs are known to be monotonic and the static CMOS gate would have poor performance. It can also be used to replace certain instances of dynamic gates in PSSL where the hold ability is not required.

Skewed CMOS

A static/dynamic hybrid called Skewed CMOS logic was proposed in [32]. This is basically a generalized NTP domino where where the clocked stages can be placed anywhere in the pipeline and both precharge high and precharge low NTP are used. In addition, those stages that are not clocked are skewed to favor the evaluate transition. The resulting circuits are more noise tolerant than domino. An example pipeline along with its timing diagram is shown in Figures 5-3. The precharge paths are allowed to take 3 times longer than the evaluate paths because there are three clocked stages in each phase.

Our work is very similar and shares the idea of using skewed static gates for improved performance and robustness. They are, however, based on different, but complementary, timing principles. Figure 5-4 compares the execution of similar 18-stage pipelines, one in Skewed CMOS and one in LS-PSSL, having the same 3× precharge/preset speedup. In Skewed CMOS, speedup is attained by having multiple sections of dependent logic in the *same* phase precharging simultaneously. Precharge must complete in one phase. The overlap or speedup factor is independent of the number of clock phases. The total number of clocked stages is mn where m is the overlap factor and n is the number of phases. In PSSL, speedup is attained by overlapping the preset of logic in *adjacent* phases. In LS-PSSL, preset must complete in one clock cycle. Here, the overlap factor is identical to the number of clock phases and the total number of clocked stages is m, since there is exactly one clock stage per phase. Therefore, Skewed CMOS requires a factor of 2 greater clock load for similar LS-PSSL pipelines with similar speedup. Edge-triggered and pulsed PSSL have even less resemblance to skewed CMOS. For the former, preset must complete in three phases, the overlap factor is 1.5 for a 50% duty cycle clock, and the total number of clocked stages is 1. One final issue is that static latches are required between stages because precharge can arrive early

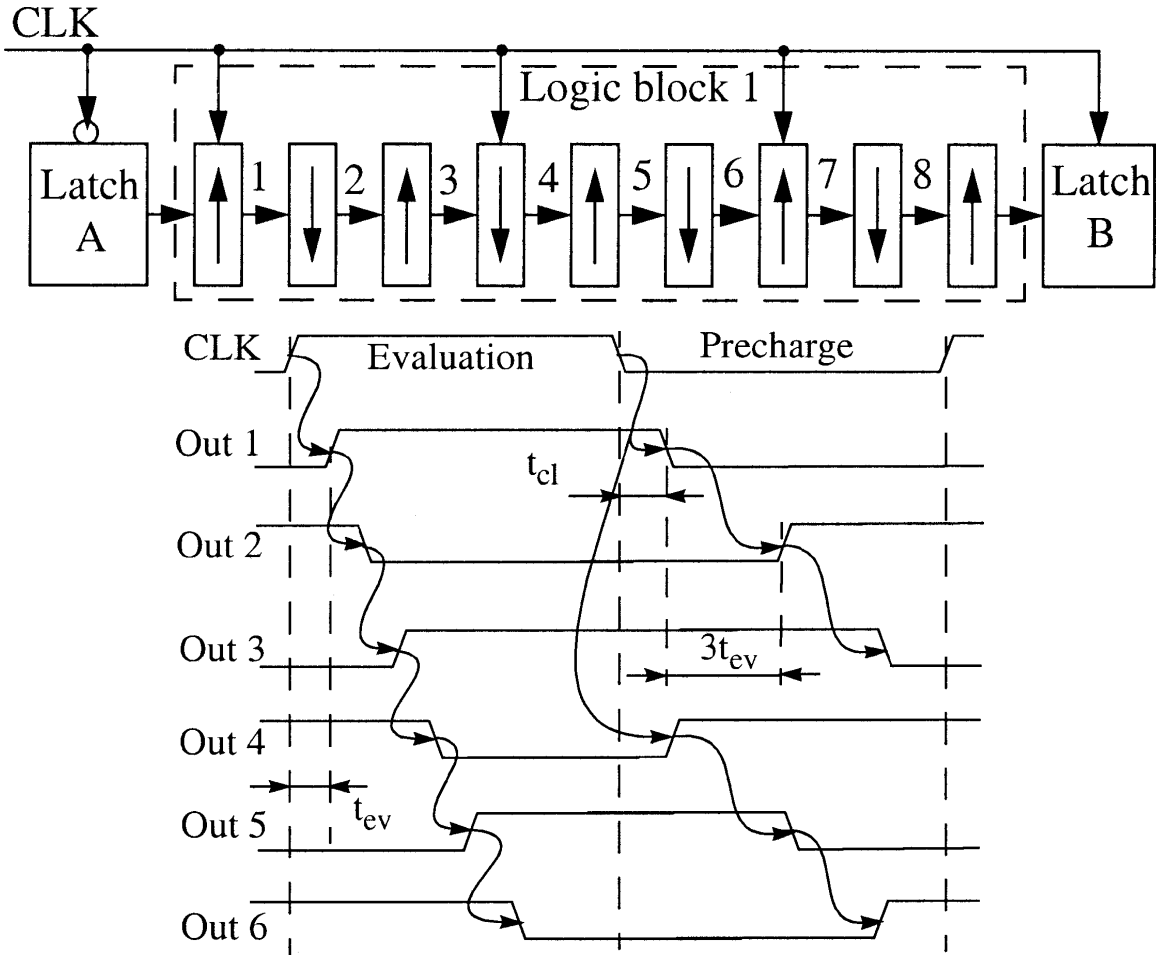
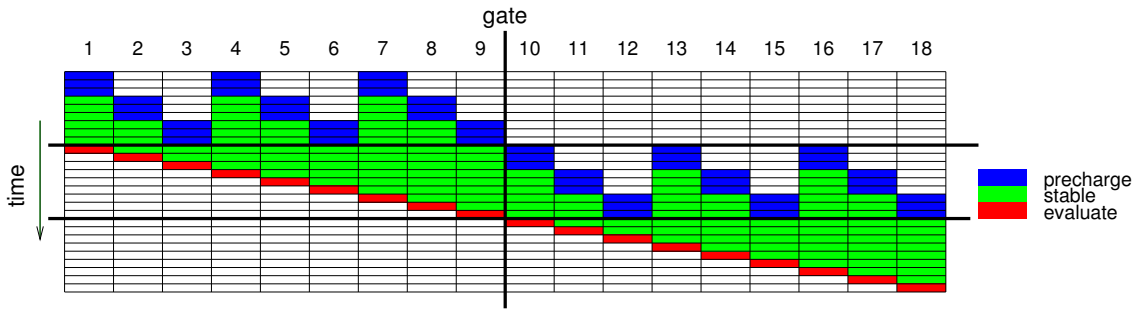
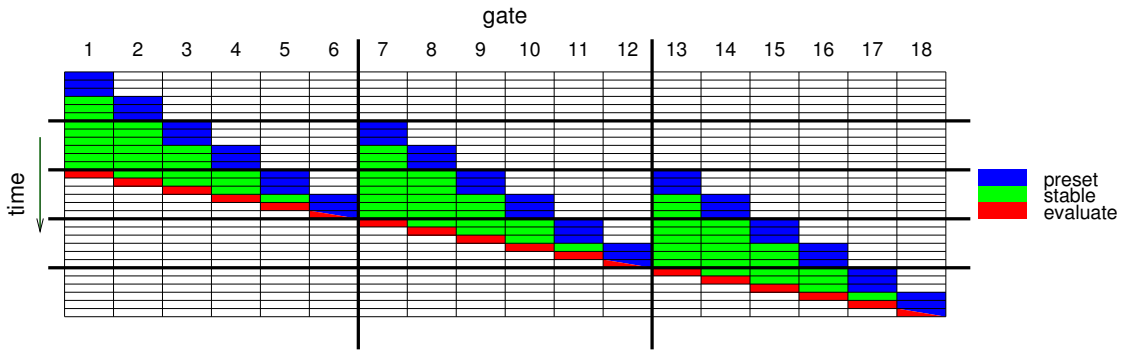


Figure 5-3: 18-stage Skewed CMOS pipeline (half are shown) and timing diagram. The precharge paths are allowed to take 3 times longer than the evaluate paths because there are three clocked stages in each phase.



(a) 2-phase Skewed CMOS with 3 clocked stages per phase. Stages 1,4,7,10,13,16 are clocked.



(b) 3-phase LS-PSSL. Stages 1,7,13 are clocked.

Figure 5-4: Skewed CMOS vs. LS-PSSL execution charts. 18-stage design. Both designs have 3× speedup on the precharge/preset paths.

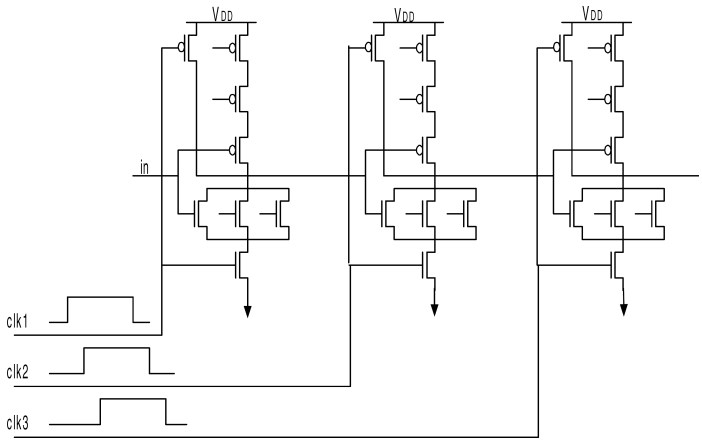


Figure 5-5: Output Prediction Logic.

Output Prediction Logic

Other novel logic styles have also been proposed. In Output Prediction Logic (OPL) [33], shown in Figure 5-5, every gate is preset high using a footed dynamic gate. The rising edges of the clocks to each gate are timed to just precede the falling edges of inputs, such that the output is at the high-gain trip point when the inputs arrive. They showed a 2-3x improvement in delay over static CMOS in a wide range of circuits with an average additional energy cost of about one third. Subsequently, a 64-bit adder was implemented in OPL with a delay of 409ps in 0.18 μm dissipating 325pJ per cycle [34] and another 64-bit adder was implemented in differential OPL with a delay of about 226ps in 0.13 μm dissipating 29.5pJ per cycle [35]. Further circuits were described in [36, 37]. A difficulty with OPL is the requirement for precise generation of clocks, separated by less than an FO4 delay. Further the best performance is achieved right when the circuit is about to fail. If the rising edge of the clock arrives any earlier, the circuit fails. If the clock arrives any later, the circuit is clock-blocked, slowing it down. Each additional circuit stage compounds this slowing, as opposed to other logic/pipelining styles which at most block once on each phase. Such precise control only becomes more difficult in sub-100nm technologies as device variation and noise cause proportionally more delay variance. Also, this technique is applicable only to datapaths where the clock generation overhead can be amortized.

Low-Voltage Swing Logic

Another novel logic style, Low-Voltage Swing (LVS) logic [38], utilizes low-voltage differential signaling and sense-amplifiers–ideas borrowed from RAM design. This allows logic to be built using large diffusion connected networks, resulting in a huge reduction of circuit nodes. This is especially advantageous for Manchester carry chains and very wide muxes or shifters. Combined with reduced voltage swings, the reduction in circuit nodes yields a reduction in power dissipation even while outperforming domino logic. LVS was used in

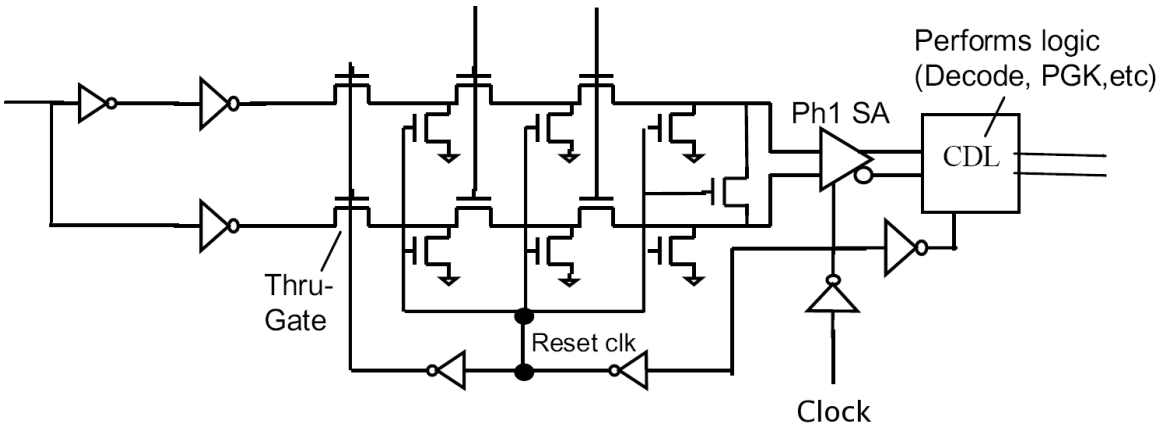


Figure 5-6: Low Voltage Swing Logic.

the design of the Pentium 4 processor at 90nm technology, which ran the integer core at 7GHz. Designing with LVS takes a lot of design effort and care, especially to eliminate non-common-mode noise and to ensure layout matching. This level of effort is normally reserved for RAMs and analog designs and required much work on design tools and methodology. It is worth noting that LVS was abandoned by Intel at the 65nm node in favor of domino [39] because LVS did not scale well enough to justify the design effort.

5.2 Variability

The static gates in PSSL are robust to variation. Variability is primarily a concern for the dynamic preset circuitry in PSSL. Techniques that improve robustness with regard to variability for domino circuits also apply to the dynamic preset circuitry.

Variation-Compensating Dynamic circuit

The Variation-Compensating Dynamic circuit technique [40], shown in Figure 5-7, is similar to the contention reduction techniques described previously. The conventional keeper is replaced with a set of binary-weighted conditional keepers which are digitally programmed to be on or off to customize the overall keeper strength to track process variation. The `NAND` gates used for turning on and off the keepers require extra area and add extra capacitance to the output node. The extra capacitance hurts the energy-delay of the circuit. This technique will be very useful in cases where the output node is already heavily loaded, as in the case of a RAM bit line, so that the added overhead is negligible. However, this technique is not as useful in the general case.

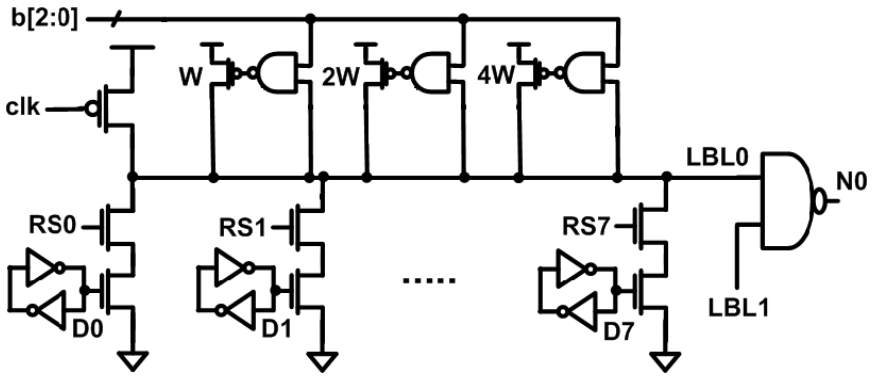


Figure 5-7: Process- Compcompensating Dynamic circuit technique. The conventional keeper is replaced with a set of binary-weighted conditional keepers which are digitally programmed to be on or off to customize the overall keeper strength to track process variation.

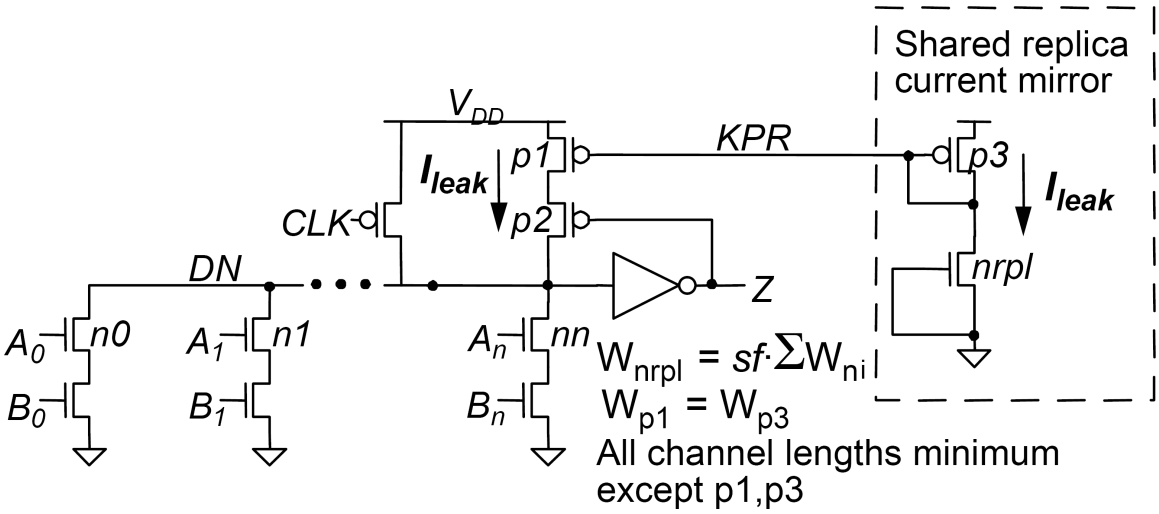


Figure 5-8: Leakage Current Replica Keeper. An analog current mirror replicates the leakage current of a reference pull-down stack onto the keeper. This allows the keeper current to track the actual leakage current.

Leakage Current Replica Keeper

The Leakage Current Replica Keeper [41], shown in Figure 5-8, uses an analog current mirror to replicate the leakage current of a reference pull-down stack onto the keeper. This allows the keeper current to track the actual leakage current with die-to-die process variation and environment variation. There is little penalty with this scheme, but care must be taken with the analog `KPR` signal to avoid noise.

Applicability to PSSL

Schemes that improve dynamic gate robustness to variability inevitably incur area, delay, energy, or design overhead. Since PSSL uses few, if any, dynamic gates, it may be preferable

to over-size the keeper and pay the delay penalty. The penalty from over-sizing in PSSL should be smaller than the penalty for Domino, in proportion to the number of gates in each pipeline stages

5.3 Pipelining

There has been work on improving performance through better pipelining. In standard two-phase footed domino logic, static latches are required between half-pipeline stages because of the early arriving precharge. In [23], Harris describes the removal of static latches, allowing certain dynamic gates to function as latches. This eliminates all synchronization delay and enables time-borrowing to be used improve performance. However, doing this with a 2-phase clocking scheme requires greater than 50% duty cycle clocks, creating clock generation complexity and hold time issues. LS-PSSL with the simplified latches allows time-borrowing even with 50% duty cycle clocks.

In [22] the authors describe wave-pipelining, where several waves of operands are processed simultaneously in a combinational circuit without being separated by latches. Subsequently this was applied to domino logic in [42]. We utilize this technique to allow non-unate signals to be used with the simplified latch/preset circuitry.

5.4 Timing Elements

Flip-flops are critical timing elements in digital circuits and have a large impact on circuit speed and power consumption. Consequently, extensive research has been performed to develop fast and low-power flip-flops [43, 44, 45, 46]. Recently, pulsed latch structures have emerged as the fastest known flip-flop structures [43, 44]. By reducing the transparency period of a latch to a narrow window, the latch can operate as a flip-flop with the additional advantage of allowing limited time-borrowing across cycle boundaries to reduce sensitivity to clock skew and jitter. These structures have the disadvantage of large positive hold times which complicates timing verification. The pulse generators can also consume considerable energy as pulses must be generated locally to avoid pulse distortion. Nonetheless, because of their performance advantages, these pulsed latch structures have been used in several commercial high-performance microprocessors [47, 48]. The DPSCRFF is simpler and faster, and, as we will see later, faster and lower-power than other known pulsed latches while also providing logic speedup from preset.

5.5 Conclusion

There is not likely to be a single logic style or logic element that is best for every instance in every design. However, we believe PSSL complements and improves upon previous work on

logic styles and timing elements. It achieves higher performance, lower energy, and higher robustness. It is also more widely applicable and easier to design with.

Chapter 6

Managing Leakage

As leakage is a major scaling concern, we present a survey of techniques to deal with leakage, highlighting special considerations for PSSL.

6.1 Leakage

The leakage paths (Figure 2-1) in a static CMOS inverter chain are shown in Figure 6-1. There are various ways to reduce the leakage. A survey of leakage reduction techniques can be found in [4]. Four major techniques are multiple-threshold circuits, sleep vector, power gating, and body biasing. We will limit our discussion to the first three. The last technique is, for the most part, orthogonal to logic style.

6.1.1 Multiple-V_{th} circuits

To reduce leakage in active circuitry, one important technique is to use two (or more) types of transistors, each type with different V_{th} voltages. In the simplest scheme, the low-V_{th} transistors are reserved for gates in the critical paths; high-V_{th} transistors are used everywhere else [49]. In theory, this would cut leakage drastically without hurting performance or dynamic power. However, the problem is that few paths are truly non-critical since any timing slack can be traded for reduced dynamic power through transistor sizing and

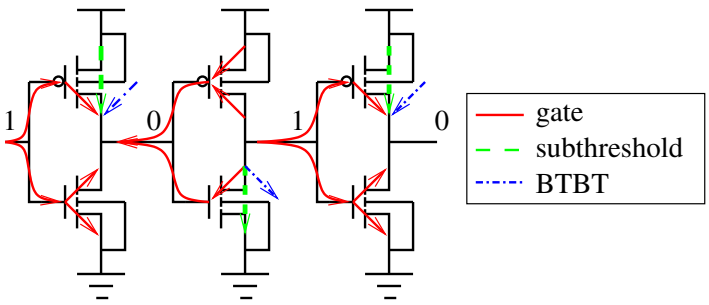


Figure 6-1: Static CMOS leakage paths.

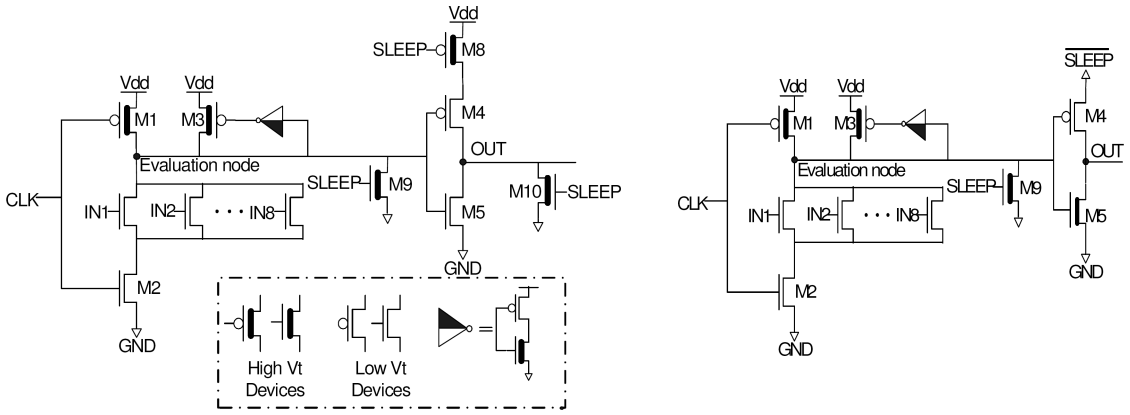


Figure 6-2: Leakage-proof domino circuits. `CLK` and `SLEEP` are forced high during sleep, which sets all the nodes low.

possibly voltage scaling. Therefore, a holistic optimization strategy that considers supply voltage, threshold voltage, and transistor sizing would be more effective [50]. The multiple-V_{th} technique adds a degree of freedom in circuit design, providing a way to trade-off static power vs. dynamic power vs. delay.

This technique can be applied to all the logic styles, with some minor differences. In static CMOS circuits, the V_{th} choice will generally be made per gate. For each gate, every transistor will have low-V_{th} or every transistor will have high-V_{th}. This is because usually all timing paths through the gate are (relatively) critical or they are all (relatively) non-critical. As an optimization, deeply stacked transistors might be left as low-V_{th} since they have lower subthreshold leakage on average. In domino circuits, the transistors in the precharge path, including the static gate pull-down, always have high-V_{th} because they are not in any timing paths. In PSSL circuits, it is beneficial to consider the PMOS and NMOS halves separately since they often have wildly different timing constraints. Transistors in the preset path will generally be less critical than transistors in the evaluate path; therefore, there is generally less penalty for using high-V_{th} devices in the preset path. Using high-V_{th} devices in the preset path also tends to reduce the skew ratio of gates and thus will enhance noise-margin.

6.1.2 Sleep Vector technique

Sleep vector techniques take advantage of the fact that the amount of leakage in a circuit is data dependent. On sleep, a fixed input vector is applied to the primary inputs of a block. This input vector is designed to minimize the leakage in the block [51]. One can further reduce leakage, at the cost of energy and delay, by introducing internal control points that set specific nodes on sleep [52]. Sleep vector techniques, on their own, are highly dependent on circuit topology for their effectiveness and can rarely entirely suppress leakage.

There are various sleep vector techniques specifically suited to domino using multiple-

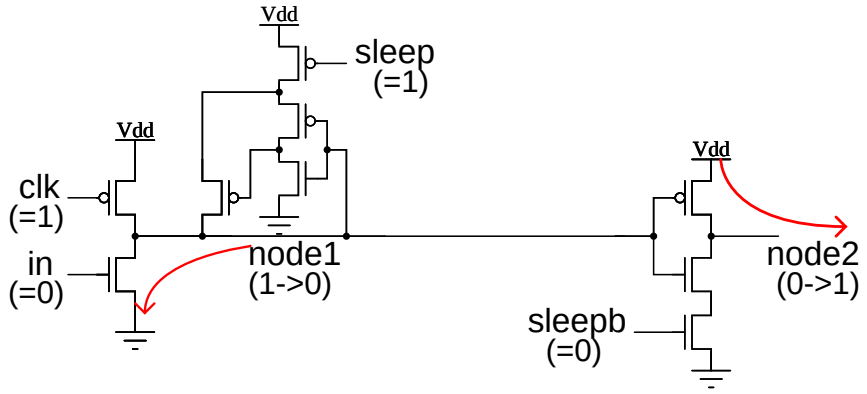


Figure 6-3: Leakage-Biased Domino. Sleep transistors are placed off the evaluate path. Leakage currents bias the circuit into a low-leakage state.

V_{th} devices. These can almost entirely suppress leakage. Figure 6-2a shows Leakage-proof domino [53]. `CLK` and `SLEEP` are forced high during sleep, driving the dynamic and static outputs low and cutting off the static pull-up. There is, however, some performance overhead due to the addition of a high-V_{th} transistor in the critical pull-up chain of the static gate. An improved technique [54], shown in Figure 6-2b, eliminates the performance overhead of the earlier technique. In Leakage-Biased Domino [55](shown in Figure 6-3), sleep transistors are placed off the critical evaluate path. Instead, leakage currents bias the circuit into a low-leakage state.

The above techniques work because of the high-V_{th} transistors in the precharge circuitry of domino logic which do not hurt performance. Similar techniques can also work for PSSL, though at some dynamic power and performance penalty, if all transistors in the preset path have high-V_{th}.

6.1.3 Power Gating

Another class of ideas uses sleep transistors to cut off power and/or ground from the logic. The original idea of power gating comes from Multithreshold voltage CMOS (MTCMOS) logic [56], shown in Figure 6-4. MTCMOS cuts off the circuit from both rails using high-V_{th} transistors. MTCMOS was truly ahead of its time. It was proposed not as a way to deal with leakage (which was virtually non-existent in their 0.5 μm technology), but as a high-performance low-power circuit technique. Since they used transistors with lower than normal V_{th} and used ultra-low supply voltages, they ended up facing the same issues that designers would later face four or five technology generations later. There is also a variant of MTCMOS using only a single sleep transistor.

The problem with MTCMOS is that its high-V_{th} transistors degrade performance. In contrast, Super Cut-Off CMOS (SCCMOS) logic [57], shown in Figure 6-5, does not require

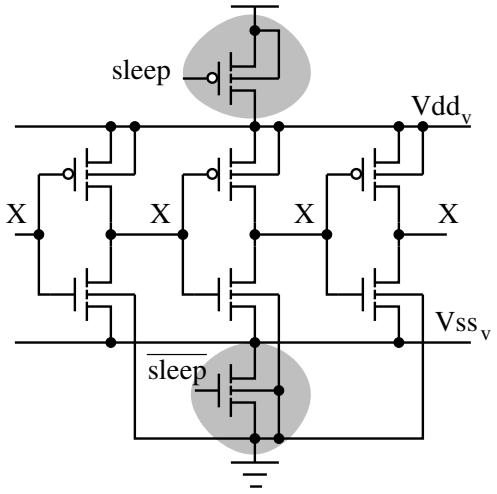


Figure 6-4: Multithreshold voltage CMOS logic. The shaded transistors are high-V_{th} to cut off the circuit from both rails.

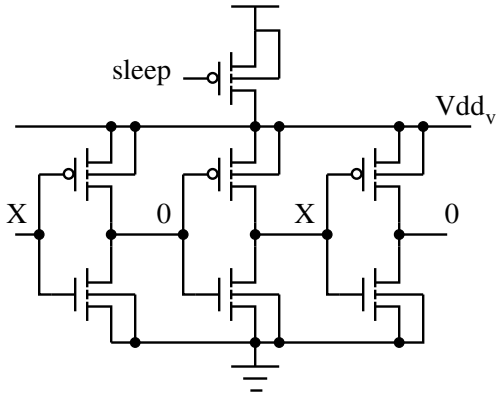


Figure 6-5: Super Cut-Off CMOS logic. The gate of the sleep transistor is boosted above V_{dd} in sleep mode to cut off the circuit from the `Vdd` rail.

high-V_{th} transistors. Rather, the gate of the sleep transistor is boosted above V_{dd} in sleep mode to cut off the circuit from the `Vdd` rail.

A problem with SCCMOS (and single sleep transistor MTCMOS) is the long time required to wake up from sleep mode due to the high-capacitance virtual supply node that needs to be recharged. During sleep, the virtual supply node is completely discharged due to leakage. Zigzag Super Cut-Off CMOS (ZSCCMOS) logic [58], shown in Figure 6-6, addresses this. Low output gates are tied to virtual `Vdd` and high output gates are tied to virtual `Vss` in a zigzag pattern. As in SCCMOS, the gates of the PMOS/NMOS sleep transistors are boosted above V_{dd}/below V_{ss} in sleep mode to cut off gates from the `Vdd`/`Vss` rails. In this configuration, the virtual rails do not swing as much so their recovery time is about 11× less than for SCCMOS. A drawback of ZSCCMOS is that gate leakage is not reduced [59].

Gate-leakage Suppressing CMOS (GSCMOS) logic [60], shown in Figure 6-7, is a modi-

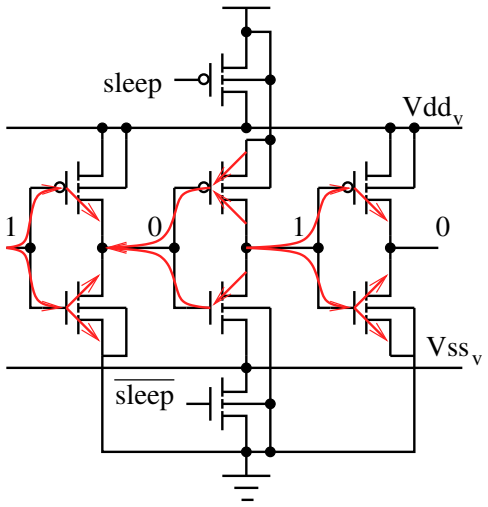


Figure 6-6: Zigzag Super Cut-Off CMOS logic and leakage paths. Low output gates are tied to virtual `Vdd` and high output gates are tied to virtual `Vss` in a zigzag pattern. The gates of the PMOS/NMOS sleep transistors are boosted above V_{dd}/below V_{ss} in sleep mode to cut off gates from the `Vdd`/`Vss` rails. Gate leakage, indicated by the arrows, is not eliminated.

fication of ZSCCMOS to deal with gate leakage. Low output gates are tied to virtual `Vdd1` and high output gates are tied to virtual `Vdd2` and also virtual `Vss` in a zigzag pattern. The gates of the PMOS/NMOS sleep transistors are boosted above V_{dd}/below V_{ss} in sleep mode to cut off gates from the `Vdd`/`Vss` rails.

6.1.4 Applications to PSSL

For deep sleep states, where power consumption is the primary concern, the only true way to eliminate leakage is through power gating. ZSCCMOS is not completely effective and so is not recommended in scaled technologies where gate leakage is large. The MTCMOS and SCCMOS techniques are independent of logic style. For static CMOS logic, using ZSCCMOS and GSCMOS requires applying a sleep vector so that the state of internal nodes is predetermined. Domino and PSSL can use precharge/preset in lieu of a sleep vector.

For fine grained leakage reduction, where low overhead is the primary concern, a sleep vector technique that utilizes high-V_{th} transistors is best. Domino logic works well with these schemes because the transistors in the precharge path do not contribute to timing, so it is convenient to put all the high-V_{th} transistors there. Static CMOS logic, on the other hand, faces a dynamic energy vs. delay vs. static energy tradeoff wherever high-V_{th} transistors are used. PSSL, however, opens up new opportunities for managing leakage reduction. Figure 6-8 compares the leakage paths in Static CMOS and multi-V_{th} PSSL, where all the of transistors on the preset path are high-V_{th}.When the PSSL circuit is placed in the opposite state of preset, all subthreshold leakage is eliminated, and gate leakage is

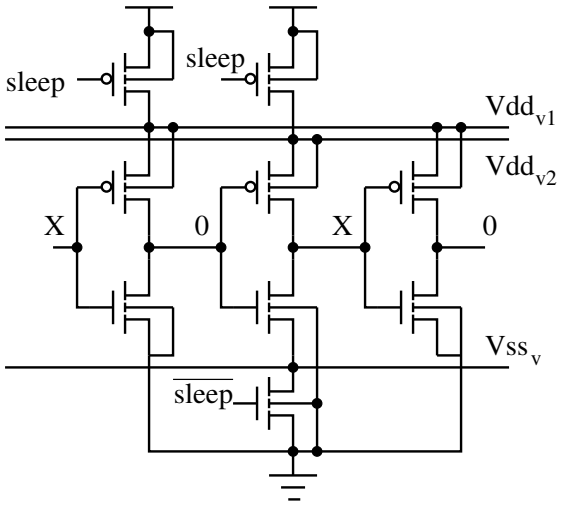
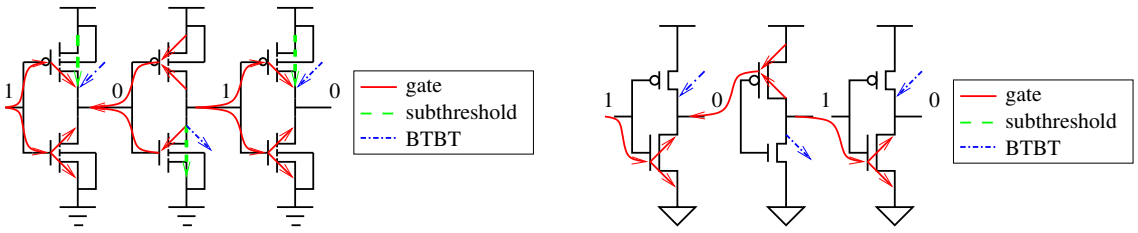


Figure 6-7: Gate-leakage Suppressing CMOS logic. Low output gates are tied to virtual `Vdd1` and high output gates are tied to virtual `Vdd2` and also virtual `Vss` in a zigzag pattern. The gates of the PMOS/NMOS sleep transistors are boosted above V_{dd}/below V_{ss} in sleep mode to cut off gates from the `Vdd`/`Vss` rails.



(a) Static CMOS (b) PSSL

Figure 6-8: Comparison of leakage paths in Static CMOS and multi-V_{th} PSSL. Smaller transistors are on the Preset path and are high-V_{th}.When the PSSL circuit is placed in the opposite state of preset, all subthreshold leakage is eliminated, and gate leakage is reduced by 1/3. BTBT is also reduced because of the reduced size of the transistors.

reduced by 1/3. BTBT is also reduced because of the reduced size of the transistors. This same scheme can be applied with Static CMOS logic; however, with PSSL, there is half the delay penalty (for 2-phase LS-PSSL), since the preset path is stretched over 1 cycle.

6.2 Conclusion

We have surveyed a number of leakage reduction techniques and have found that PSSL creates opportunities for leakage reduction. We have proposed a fine grained leakage reduction technique that takes advantage of the unique properties of PSSL; this technique has reduced overhead when used in PSSL as compared to static CMOS.

Chapter 7

Evaluation

In this chapter we show the effectiveness of PSSL logic quantitatively. Several test circuits were implemented in PSSL and other logic styles, starting from simple inverter pipelines and culminating in the implementation of 32-bit accumulators. In addition, the DPSCRFF flip-flop is also compared against existing flip-flop designs.

7.1 Linear Feedback Shift Register

We implemented a 2-bit linear feedback shift register (LFSR) shown in Figure 7-1 using both static CMOS logic and LS-PSSL. This LFSR implements the sequence 11,01,00,10 (repeated) and was chosen because it is simple and yet exercises the complete set of one-bit state transitions, providing a good test of pipeline functionality.

7.1.1 Methodology

The design of the LFSR in LS-PSSL is shown in Figure 7-2. Of the four phases, three are non-inverting, and thus race-free. However, the last phase is inverting and thus presents a minimum path delay with respect to the high phase of the clock. However, as there are no such constraints with respect to the low phase of the clock, the clock can be stopped in that phase. The inverters in the design are skewed such that the preset path delay is roughly twice the non-preset path delay.

The static CMOS design is similar except that standard latches are used and that, instead of alternately skewed inverters, one type of inverter is used.

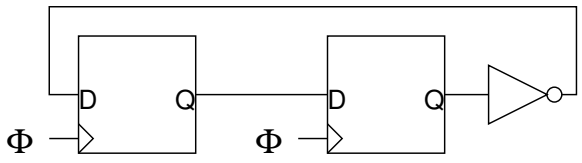


Figure 7-1: Two-bit Linear Feedback Shift Register.

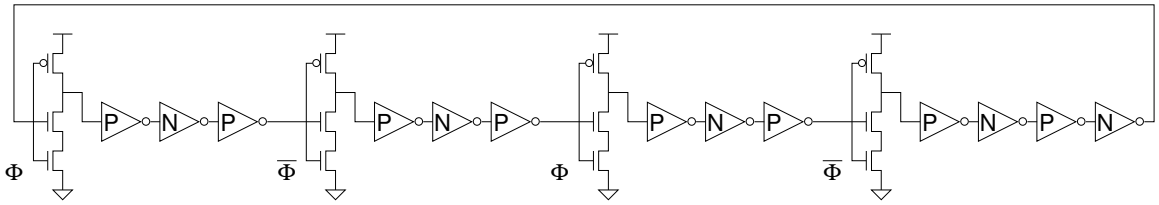


Figure 7-2: Linear Feedback Shift Register implemented using LS-PSSL. Feedback devices have been omitted for clarity. Inverters labeled P/N are skewed to favor the rising/falling transition respectively.

These were tested using HSPICE in a TSMC 0.18 μm process with estimated device parasitics. All nodes were loaded with 12fF of capacitance to represent the loading of wire and of other logic. The minimum clock period and the energy dissipation, excluding clock energy, was measured for each design point.

The static CMOS design was hand optimized to generate the energy-delay curve. We then optimized the PSSL design to have minimum delay subject to a 460fJ energy constraint, which is about the median among the static CMOS energies.

7.1.2 Results

The results are shown in Figure 7-3. The PSSL implementation runs at about 1.5GHz whereas static CMOS implementation runs at about 1GHz when dissipating about 460fJ per clock cycle. This comparison was performed with a fixed load at each node. The improvement would be even better if the load scaled with the sizes of the inverters because of the lower input capacitance of PSSL.

The simulation waveforms for the last stage of the LFSR is shown in Figure 7-4. The top graph shows the output of the dynamic gate. There is an extra pulse whenever the output remains in a 0 state. It thus toggles twice as often as the equivalent node in the static CMOS implementation. Subsequent graphs show the outputs of successive inverters. The pulse is attenuated and eventually eliminated. This occurs because of the skew in the inverter chain. The faster evaluate path eventually overtakes the slower preset path. This reduces the energy penalty of preset so that PSSL can attain better energy-delay. It can be noted that this effect is timing dependent. The faster the circuit is run, the faster the preset pulse is eliminated and the less energy per cycle is expended.

7.2 Shift register using wide fan-in gates

We implemented a 4-bit shift register to further compare PSSL to other logic styles. The design of the shift register is shown in Figure 7-5. In contrast to the 2-bit LFSR of the previous section, the 4-bit design is non-inverting and thus has a straightforward domino implementation. Also, instead of inverters, we used alternating 3-input `NAND` and `NOR` gates.

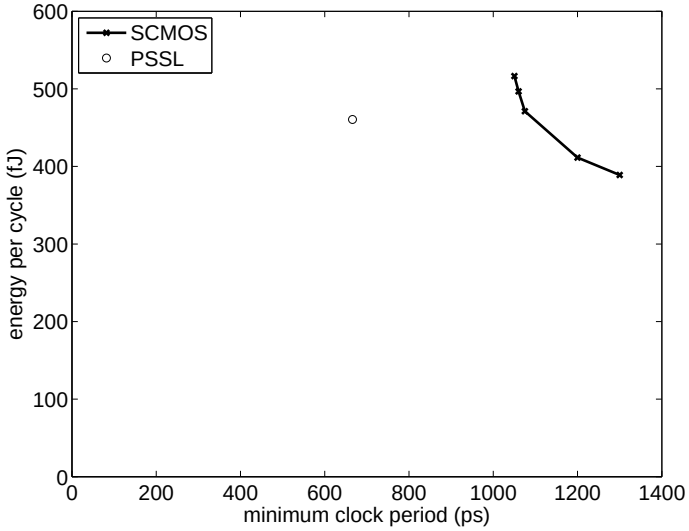


Figure 7-3: Linear Feedback Shift Register energy-delay comparison at 0.18 μm, 1.8V, TT, 25° C.

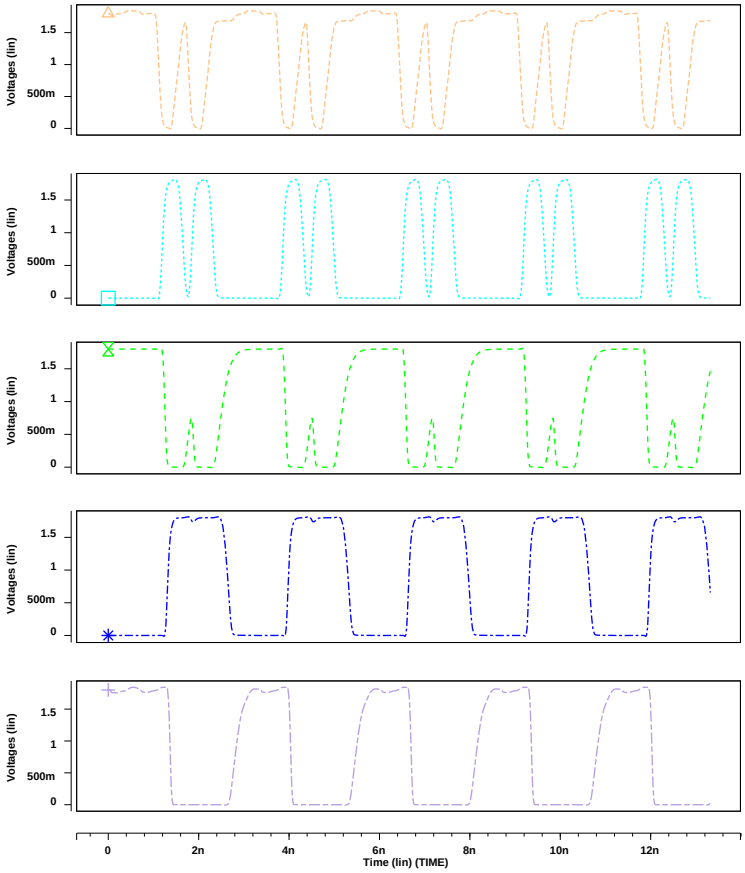


Figure 7-4: LS-PSSL LFSR waveforms. The top graph is the output of the last dynamic gate in the design. Subsequent graphs show the outputs of successive inverters.

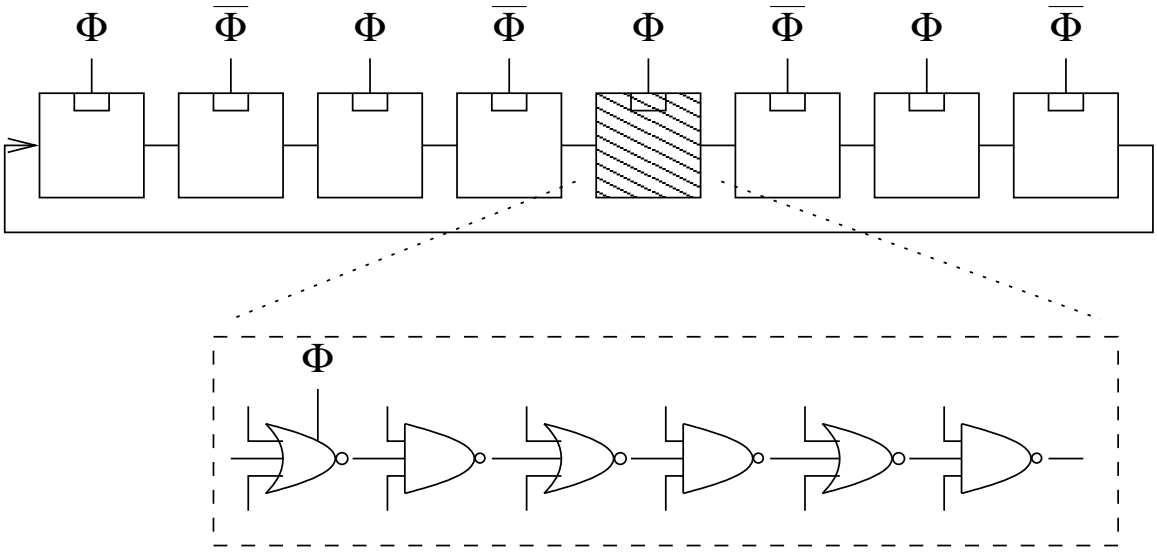


Figure 7-5: Four-bit shift register using wide-fan-in gates.

This structure was chosen because it mimics the implementation of very wide fan-in gates. In addition, this structure highlights the advantages of skewed logic styles such as PSSL and domino. It also provides a vehicle for examining the scaling of wide-fanin circuits into nanometer technologies. Domino, footless-domino, static CMOS, and LS-PSSL designs were compared.

7.2.1 Methodology

We chose to limit our evaluation to designs requiring only 50% duty cycle clocks. For Domino, this would normally mean the insertion of static latches between pipeline stages to solve the precharge race condition. This has the drawback of adding two extra stages of logic in the pipeline. Instead, the last `NAND` gate of each stage has an extra clock NMOS transistor to form a half-latch (with full keeper). For Footless Domino, this means that the delays of the precharge and evaluate phases has to match. Higher performance could be attained by relaxing the duty cycle requirement. However, it is not clear how feasible this would be in practice. The LS-PSSL version uses dynamic gates for preset to minimize delay and to avoid the penalty of clock jitter.

The static CMOS version uses a fully static 8-transistor inverting latch between stages. This intentionally inverts each half-pipe stage so that the slow path through the gates is followed by the fast path. Using time borrowing, the minimum clock period becomes the sum of the delays of the fast and slow paths, instead of twice the slow path as would be the case without the inversion.

We randomly generated variants of each of the four designs. Each variant was simulated in Hspice using the PTM [18] 32nm process at 0.6 volts, 100° C, and with 1fF of capacitive load at each node in addition to estimated device parasitics. For each design, the delay

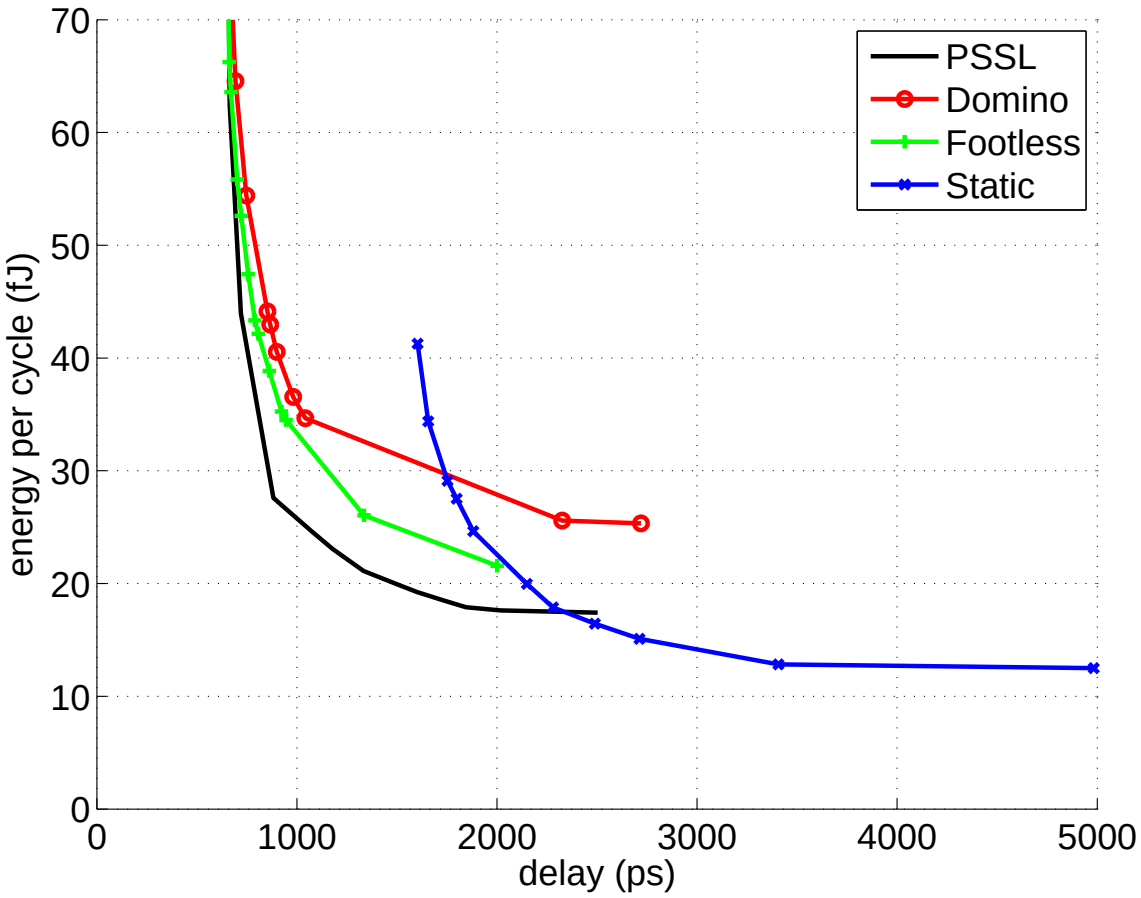


Figure 7-6: Four-bit shift register energy-delay curves at 32nm, 0.6V, 100° C.

was measured, then, running at the maximum clock frequency, the energy per cycle was measured.

7.2.2 Results

The results are shown in Figure 7-6. Only the optimal points, those on the good quadrant of the convex hull, are plotted. Over a wide range of design points, PSSL has the lowest energy and delay. At the 1000ps design point, it is almost twice as fast as Static CMOS for the same energy and uses about 30% less energy than Domino at the same delay. Footless domino is better than Footed domino, but worse than PSSL.

7.3 Flip-flop comparison

We compared the energy-delay curves of DPSCRFF against previously published designs (Figure 7-7) [61]. For the delay metric we chose the minimum D-to-Q delay. This is appropriate because it takes into account the relationship between input arrival time and

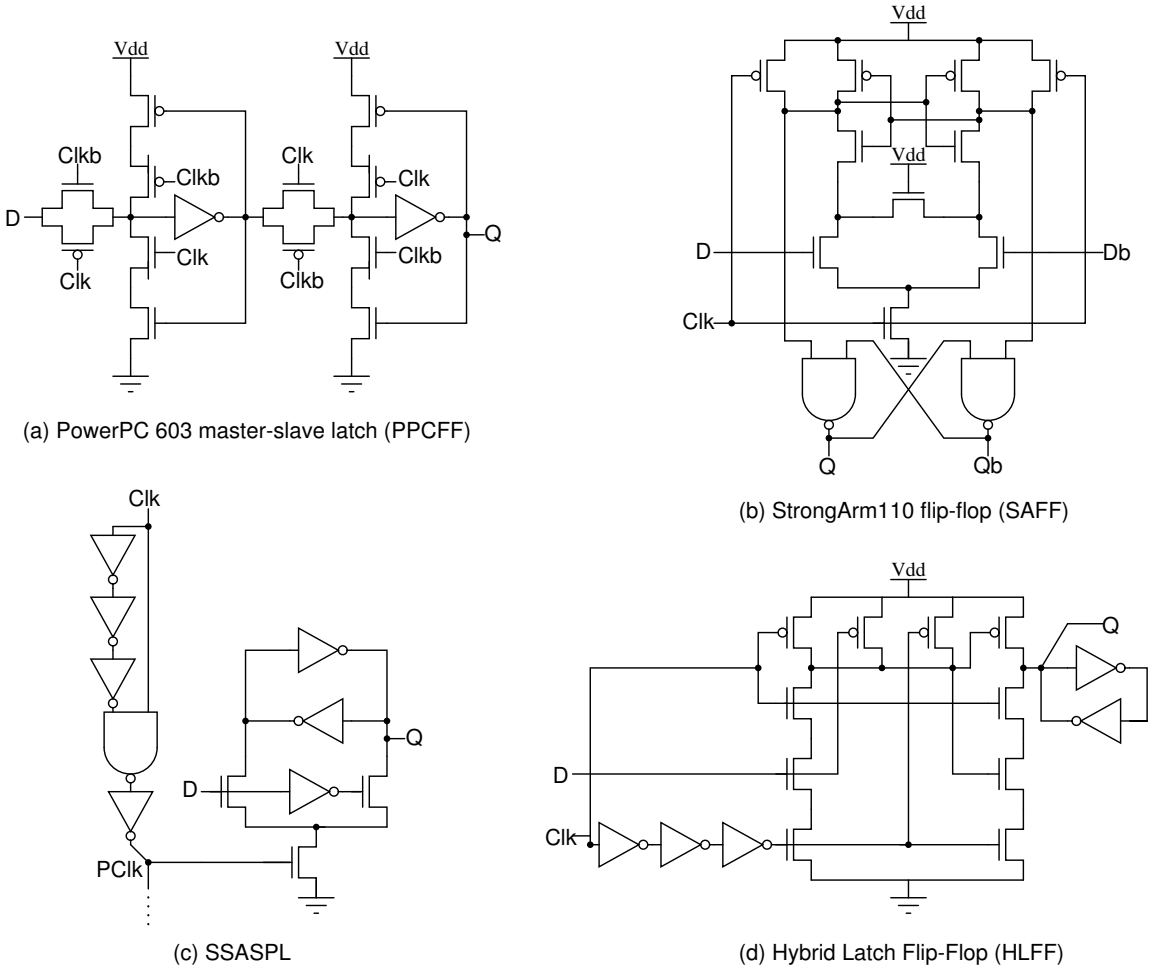


Figure 7-7: Flip-flops for comparison

Clk-to-Q delay [45].

7.3.1 Methodology

All flip-flops were simulated using Hspice from schematic netlists annotated with accurate source/drain parasitic diode parameters using a TSMC 0.25 μm process. Figure 7-8 shows the test-bench used for the evaluation. The test-bench is based on that in [61]. However, we chose more balanced 2/1 inverters instead of minimum sized inverters in the data and clock buffers. As in [61, 45], we subtracted out the energy dissipated in charging and discharging the output load capacitors. In addition, as in [45], we also subtracted out the energy dissipated in the input buffers.

The relative ranking of flip-flops depends on the loading conditions assumed [62]. For this evaluation, we chose a load of 7.2 fF which corresponds to four minimum sized inverters in this technology. This represents a typical light load in a datapath structure [61]. To drive higher loads, it is likely that additional levels of output buffering should be used [62].

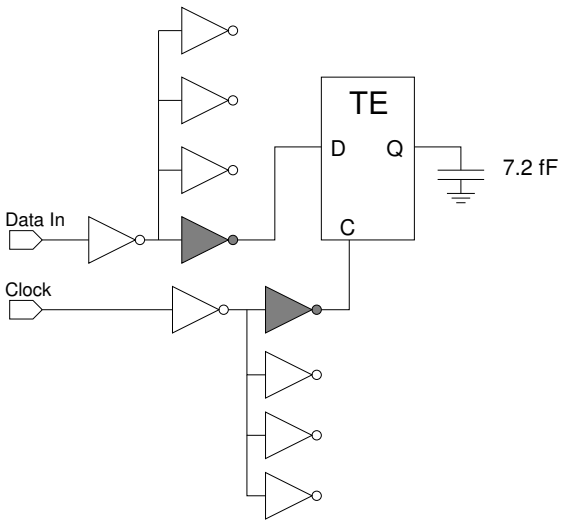


Figure 7-8: Test-bench setup. Balanced FO4 2/1 inverter trees provide realistic input waveforms.

The pulse generators of the DPSCRFF and the SSASPL were connected to four of the flip-flops, and the energy cost of the pulse generation is considered to be amortized between them.

The transistor sizes in the designs were each optimized for several design points. This optimization was performed using data inputs that were stable well before and after the arrival of the clock. The clock was un-gated and the data alternated on every cycle. Clk-to-Q delay and energy were measured. Afterward the minimum D-to-Q delays were found by optimizing the data input arrival times.

7.3.2 Results

Figure 7-9 show the results. As can be seen, the fastest DPSCRFF at 54 ps is significantly faster than the next fastest flops, HLFF and SSASPL, at roughly 150 ps. The lowest-power DPSCRFF at 141 fJ is comparable to the lowest-power flop, PPCFF, at 130 fJ. However, it has a propagation delay of only 167 ps compared to 342 ps.

Figure 7-10 show how the energy dissipation varies with different clock and data input patterns for the different flip-flops. Note that the flip-flops shown in this figure have widely varying propagation delays as shown by the labels in the axes. When the data is held low while the clock continues to run, the energy dissipation of the DPSCRFF is reduced. However, if the clock is running and the data is held high, the DPSCRFF actually dissipates more power than for the full activity waveforms because of its output glitches. When the clock is held stable, no internal nodes change state and only the single data input gate toggles. The DPSCRFF therefore has low energy when the local clock is gated.

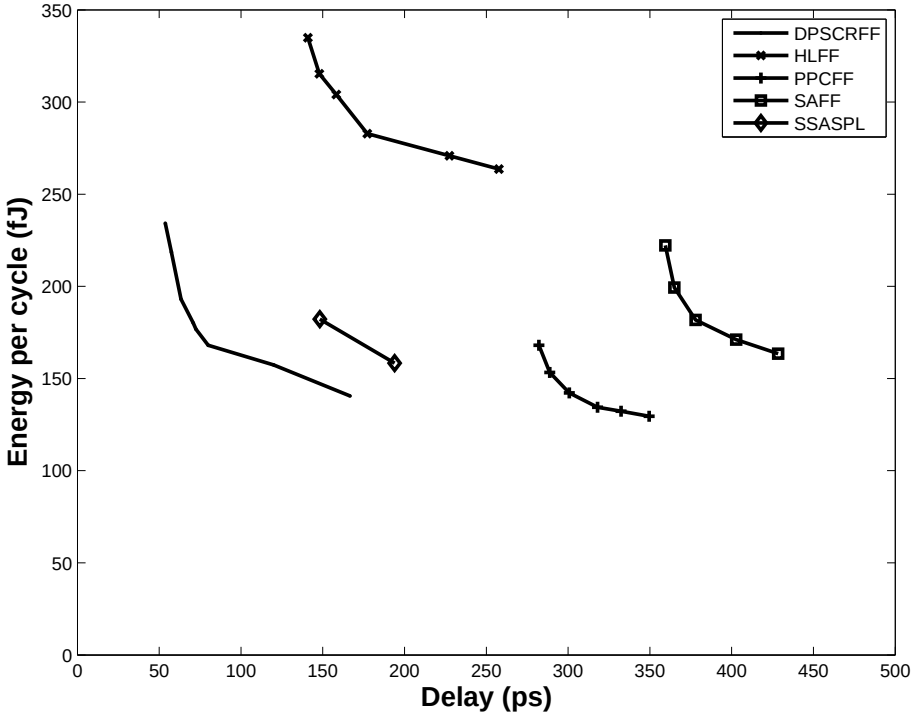


Figure 7-9: Energy versus delay for various flip-flops at 0.25 μm, 2.5V, TT, 25° C.

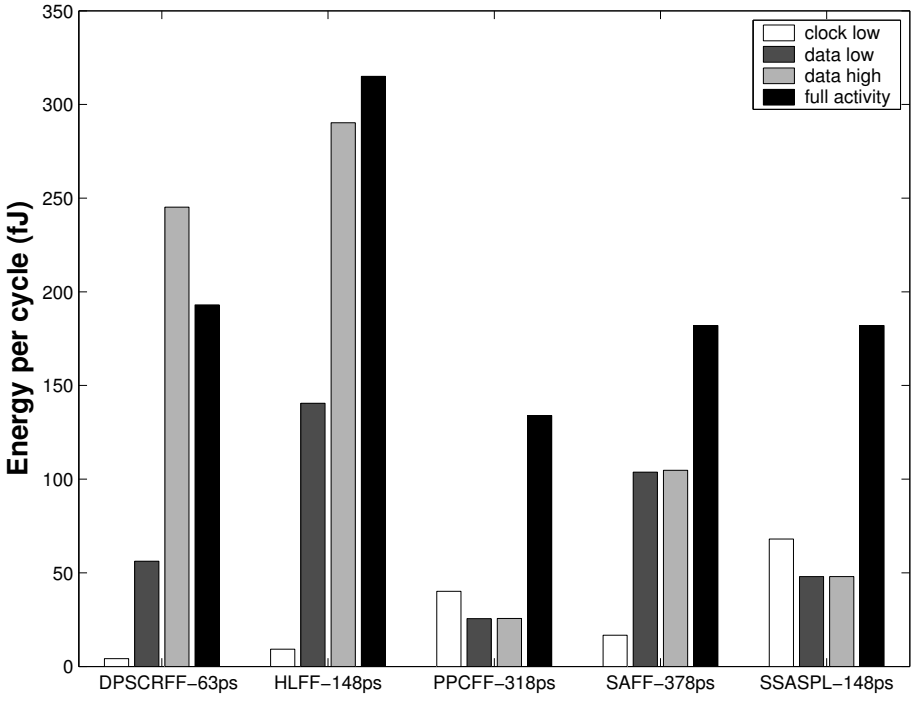


Figure 7-10: Energy Dissipation across different input waveforms for various flip-flops.

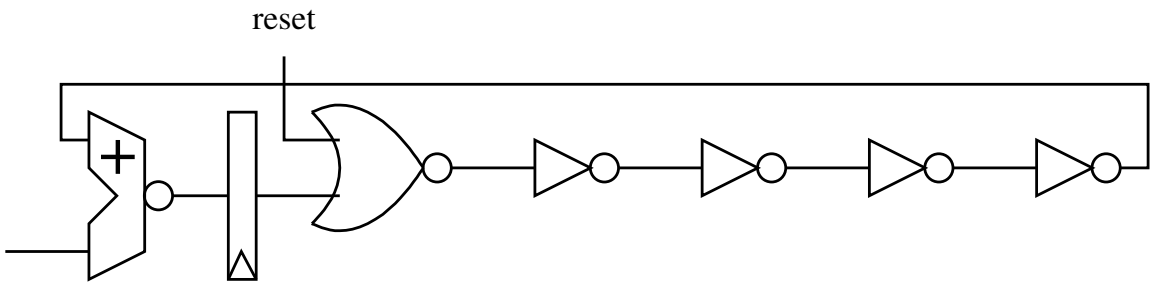


Figure 7-11: Accumulator design. The datapath is 32-bits wide. The inverter chain adds extra delay to facilitate two-phase clocking.

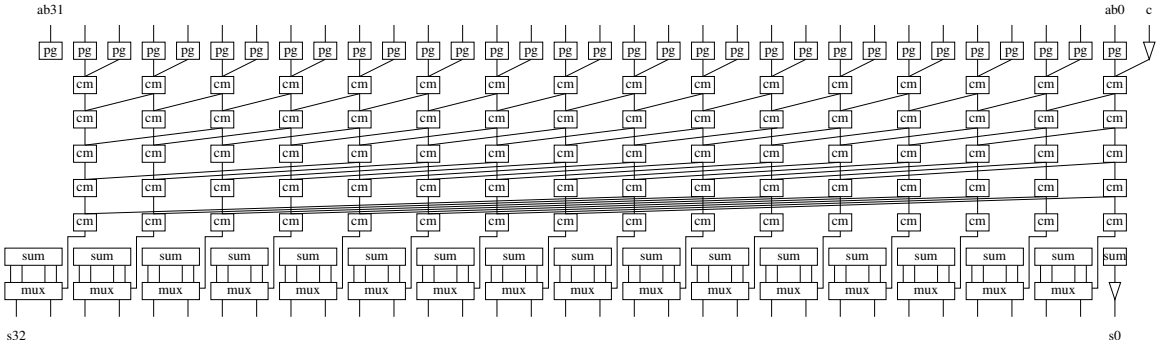


Figure 7-12: Adder architecture. The hybrid carry-lookahead/carry-select implementation computes every other carry bit in the lookahead tree, like a Han-Carlson adder, saving area and energy, but precomputes the final sum bits to reduce the number of logic stages by one.

7.4 32-bit Accumulator

Adders are an important case to consider as they are found in the critical paths of microprocessors and DSP's. We implemented a 32-bit accumulator to serve as a realistic test case for PSSL. The accumulator design is shown in Figure 7-11. Some extra delay was added to the pipeline to facilitate two phase-clocking.

For the adder we chose a hybrid carry-lookahead/carry-select architecture shown in Figure 7-12. Like in a Han-Carlson adder, the carry-lookahead tree only computes every other carry bit, so requires only half the resources as a full carry-lookahead. However, unlike the Han-Carlson which has a final ripple carry calculation to calculate the remaining carry bits, this design precomputes two versions of each 2-bit sum, reducing the number of logic stages by one. Static CMOS, domino, LS-PSSL, and pulsed-PSSL variants were designed and evaluated.

7.4.1 Implementation

A carry-lookahead adder has three major components: carry-generation, partial-sum generation, and final-sum generation. The former two operate in parallel and the results are combined in the last component. Carry-generation is unate. Partial and final sum gen-

eration, however, are not unate. There is, therefore, some amount of trickery involved in designing a PSSL or domino implementation. One solution for domino implementation is to use a dual-rail scheme throughout. However, this is inefficient as the carry-generation component forms the bulk of the circuitry. Another solution is to employ de-racers or complementary signal generators at the boundaries of unate and non-unate sections [21]. This is the solution that is chosen in our design. However, some of the de-racers are in the critical path between carry-generation and final-sum generation. This is unavoidable in the domino and pulsed-PSSL designs. However, one of the advantages of LS-PSSL is that non-unate logic is allowed if certain minimum-path timing restrictions are met. We take advantage of this to remove the de-racers at the output of the carry-generation component. All the versions were designed to be as similar as feasible to one another for fair comparison.

7.4.2 Evaluation

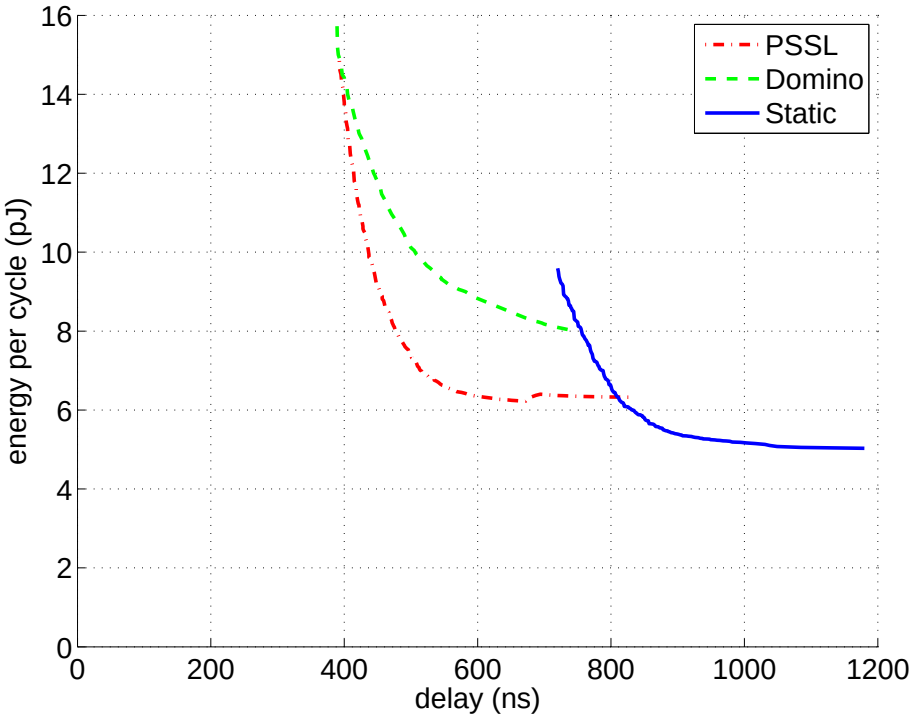
We simulated the designs in the TSMC 0.18 μm micron process at 1.8V and 1.2V. The netlist was annotated to include estimated wire and source/drain parasitics. To measure critical path delay we used Pathmill, which is a transistor level static timing analysis tool. To measure energy consumption we used Nanosim to simulate the accumulator designs over 9 cycles with the same set of random input data. The supply energy draw for the entire design was measured.

In order to generate energy-delay curves we designed and implemented an algorithm to optimally size transistors in the circuit. The algorithm begins with the circuit in the minimum size/lowest energy configuration. In each iteration it selects the transistors which control the node in the critical path of the design with the slowest transition time. Rising transitions are given a $\sqrt{3}$ delay factor over falling transitions. The sizes of the selected transistors are increased so that the transition times on the critical paths will tend to be more balanced. These ideas were drawn from [17]. The algorithm results compared favorably with those from hand optimization.

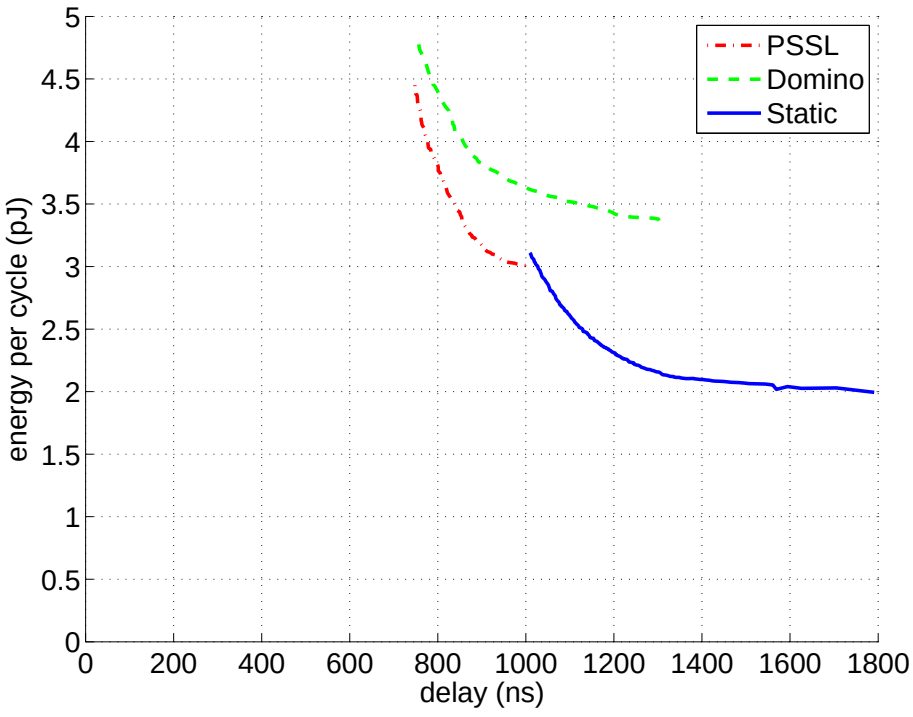
The results are shown in Figure 7-13. At 1.8V LS-PSSL is superior to static CMOS except at the lowest energy points and is superior to domino except at the lowest delay points. At the 500ps design point, it uses 20% less energy than domino and is 33% faster than static CMOS. At 1.2V the results are qualitatively similar. However, the voltage scaling does not degrade the performance of static CMOS as severely so the crossover point between static CMOS and PSSL occurs at a higher point on the curve.

7.5 Conclusion

We have tested PSSL and compared it against other common logic styles, using several test circuits from shift registers to accumulators. We have shown that, over a wide range of design points, PSSL is superior to domino and static CMOS logic in terms on energy and



(a) 1.8V



(b) 1.2V

Figure 7-13: 32-bit accumulator comparison at 0.18 μm, TT, 25° C

delay, and at the same time is more robust than domino. In particular, we have shown 20-30% energy reduction vs. domino and 33-50% delay reduction vs. static CMOS. In addition, we have presented the Double Pulsed Set Conditional-Reset Flip Flop (DPSCRFF), which is twice as fast as previous flip flops described in literature while consuming the same amount of energy.

Chapter 8

Testchip

To demonstrate the advantages of PSSL in real hardware, we built a custom test-chip in the TSMC 180nm process with three ALU cores, using the static CMOS, domino, and LS-PSSL logic styles. In this chapter, we describe the design, implementation, operation, and testing of this chip. We also present initial energy delay performance comparisons between the ALUs implemented in the three logic styles.

8.1 Architecture

A block diagram of the test-chip is shown in Figure 8-1. The ALU cores consume a 40-bit input vector and produce a 32-bit output word on every cycle, at up to roughly 1 GHz, for a total bandwidth of up to about 72Gb/s. A fourth pass-through ALU merely passes inputs to outputs and can be used for end-to-end testing of the test infrastructure. The testchip reuses an existing socket and bus interface in order to limit the hardware and software design effort. Data is communicated on/off-chip through the Asynchronous Host Interface Protocol (AHIP) bus. This 8-bit bidirectional bus operates in the low tens of megahertz with a 50% control overhead, for an effective bandwidth of roughly 50-100Mb/s.

8.1.1 Test infrastructure

To verify correct operation of the cores, a mechanism must exist to provide known test vectors and to check results. The simplest and most flexible mechanism feeds input data and samples output directly from off-chip at full-speed. This is, however, impossible because of the 1000× discrepancy between required and available bandwidth, even if we allow for test compression techniques.

Since the AHIP bus does not have the required bandwidth to supply the ALUs, an internal 128-entry 40-bit Pattern Generator RAM (PGRAM) is included to provide input data to the cores at full speed. The contents of the PGRAM are written offline, through the AHIP bus. In addition, a 128-entry 32-bit Logic Analyzer RAM (LARAM) is included

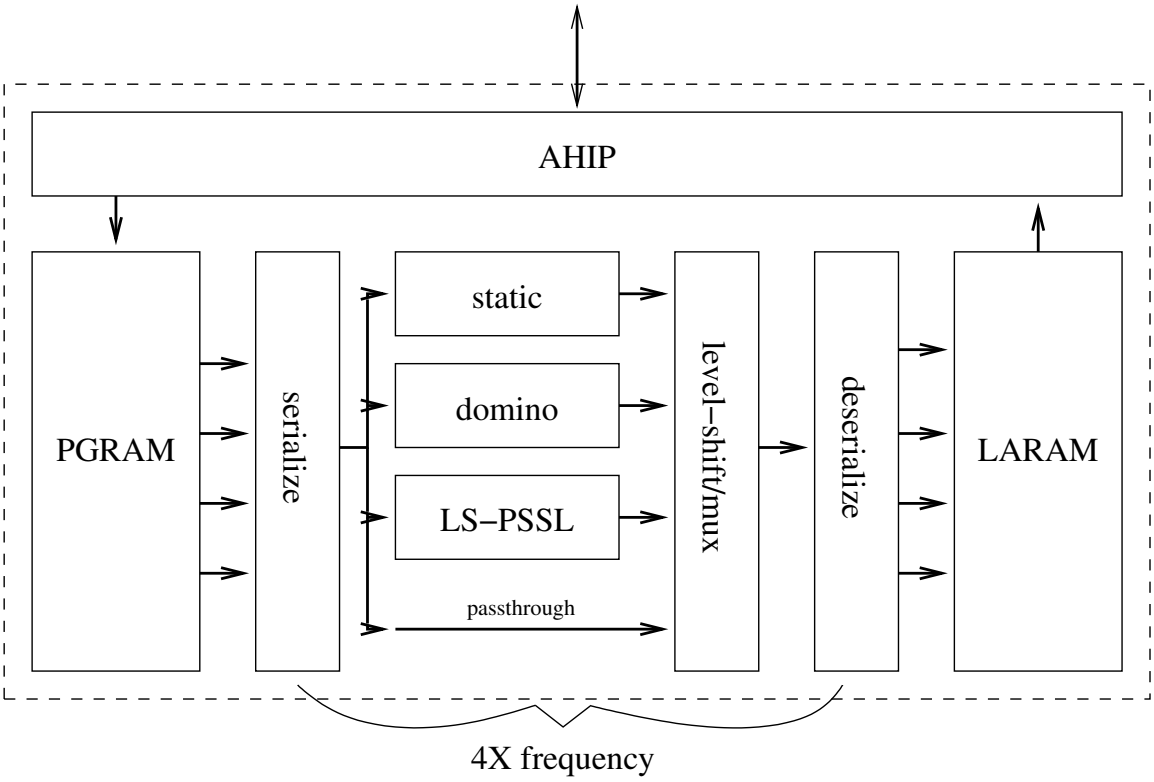


Figure 8-1: Test-chip block diagram.

to capture output data from the cores at full speed. The contents of the LARAM are read offline, also through the AHIP bus. The core operating frequency of 1 GHz is well beyond the capabilities of the test infrastructure, and in particular the RAMS. Therefore, most of the chip runs at a quarter of the core frequency. Together, the LARAM and PGRAM allow full cycle-level control and visibility of the ALU inputs and outputs. The LARAM contents can be compared to expected results offline to determine whether a particular ALU functioned as expected.

An internal register selects the active ALU. This register can be written to through the AHIP bus. Only the active ALU is clocked. The outputs of the active ALU are written into the LARAM.

8.1.2 Measurement infrastructure

The core clock can be provided either by an off-chip clock generator or by an internal voltage controlled oscillator. In either case, the frequency can be measured through an external pin. The frequency of the internal clock generator is set by the voltage on an external pin. The voltage-frequency relationship of the internal clock generator is shown in Figure 8-2.

Each of the ALUs, except the pass-through ALU, is connected to a separate power supply. The voltage for each power supply can be digitally set and measured. In addition,

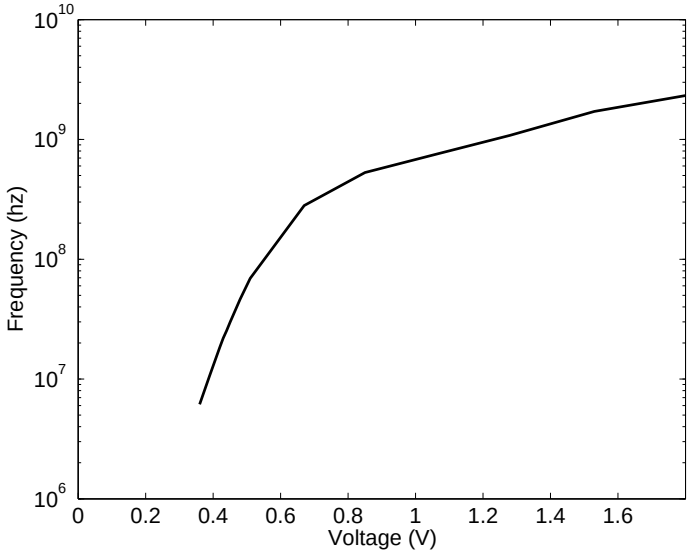


Figure 8-2: On-chip VCO frequency vs. input voltage at 1.8V, 25° C.

the current can be measured and recorded digitally in real time. This allows one to measure the power consumption of each ALU.

8.1.3 Chip operation

On each slow cycle, the PGRAM produces four input vectors. These are fed serially to the ALUs on each core cycle. The outputs of the ALUs are level-shifted to the main chip voltage of 1.8V. On each core cycle, one of the shifted outputs are selected and fed into the deserializer, which produces a 4-word wide vector on each slow cycle, which is then written into the LARAM.

8.1.4 ALU architecture

The architecture of each ALU is shown in Figure 8-3. This architecture is chosen to mimic the critical ALU bypass path of microprocessors. A pair of input muxes select data from either outside the ALU (presumably the register file), the output of the ALU (for the primary pipeline bypass), or from one of two other inputs (representing other possible input sources). The last two inputs are tied to 0 and -1. An `XOR` in the second input path, along with the carry input, allows the selectable computation of $A+B$, $A-B$, $A+B+1$, and $A-B-1$. An output mux selects data from either the output the adder, or from one of three other inputs (representing the outputs of other functional units such as shifters or logic units). This organization is fairly typical. That said, the exactness of correspondence of this datapath to any other ALU datapath is unimportant.

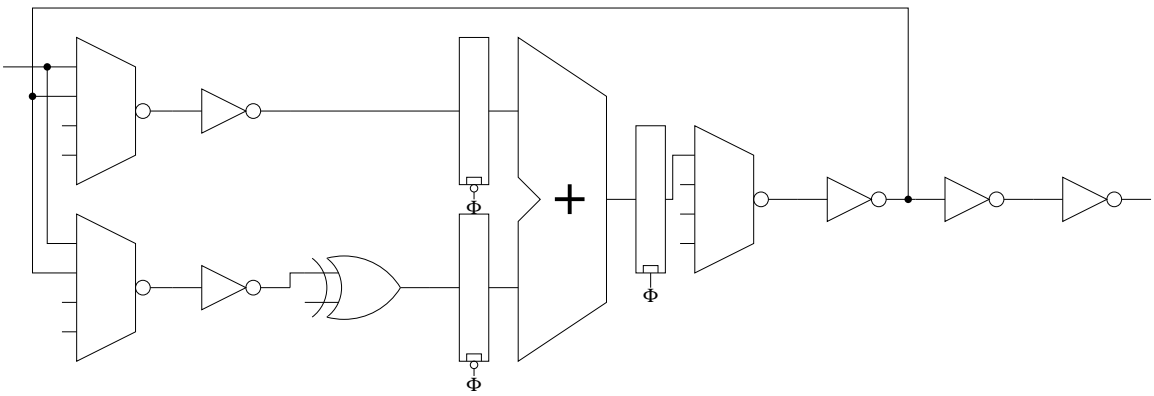


Figure 8-3: ALU block diagram.

8.2 Implementation

The adders in the ALUs are the same as those described in section 7.4, except for transistor sizes.

8.2.1 Transistor size selection

The theory of transistor size selection is as follows. Let $\boldsymbol{S} \subset \Re^n$ be the set of valid transistor size vectors. Let $\boldsymbol{s}$ be a transistor size vector such that $\boldsymbol{s} \in \boldsymbol{S}$. Let $\boldsymbol{V} \subset \Re$ be the set of valid supply voltages. Let v be a supply voltage such that $v \in \boldsymbol{V}$. Let $y(\boldsymbol{s}, v)$ and $x(\boldsymbol{s}, v)$ be the energy and delay, respectively, of the circuit at the given sizes and voltages. A well chosen transistor size design point is such that the energy-delay trade-off from voltage scaling is identical to the energy-delay trade-off from transistor sizing. In general, this is described by a curve in $\boldsymbol{S} \times \boldsymbol{V}$ that satisfies $\nabla y(\boldsymbol{s}, v) = \lambda \nabla x(\boldsymbol{s}, v)$ for $\lambda \in \Re$. Second order constraints also apply to filter out local maxima and saddle points. Different values of λ yield different points on the optimal curve.

For our purposes, we wish to optimize around the point at v=1.8V. Since we are only interest in that point, we can prune the search space by only considering points at 1.8V. We first trace a curve in $\boldsymbol{S}$ that satisfies $\nabla y(\boldsymbol{s}, 1.8) = \lambda \nabla x(\boldsymbol{s}, 1.8)$. This was done with the assistance of a computer program using an algorithm based on gradient descent but modified to prune the search space by only looking at transistors on the critical path. We then retraced the curves at v=1.5V to give an approximation for $\frac{\partial y}{\partial v}, \frac{\partial x}{\partial v}$ at every point.

This algorithm was applied to each of the three ALU designs. The resulting energy-delay curves, based on extracted and back-annotated parasitics, are shown in Figure 8-4. Energy results are from Nanosim spice simulation. Delay results are from Pathmill static timing analysis. Note that we did not consider noise-margin in the transistor sizing.

The energy-delay efficient point is determined by $\frac{dy/d\boldsymbol{s}}{dx/d\boldsymbol{s}} = \frac{\partial y/\partial v}{\partial x/\partial v}$. Using this equation, we chose the PSSL design point of just under 1GHz frequency. We sized the domino design to be approximately the same speed as the PSSL design at 1.8V. We sized the static CMOS

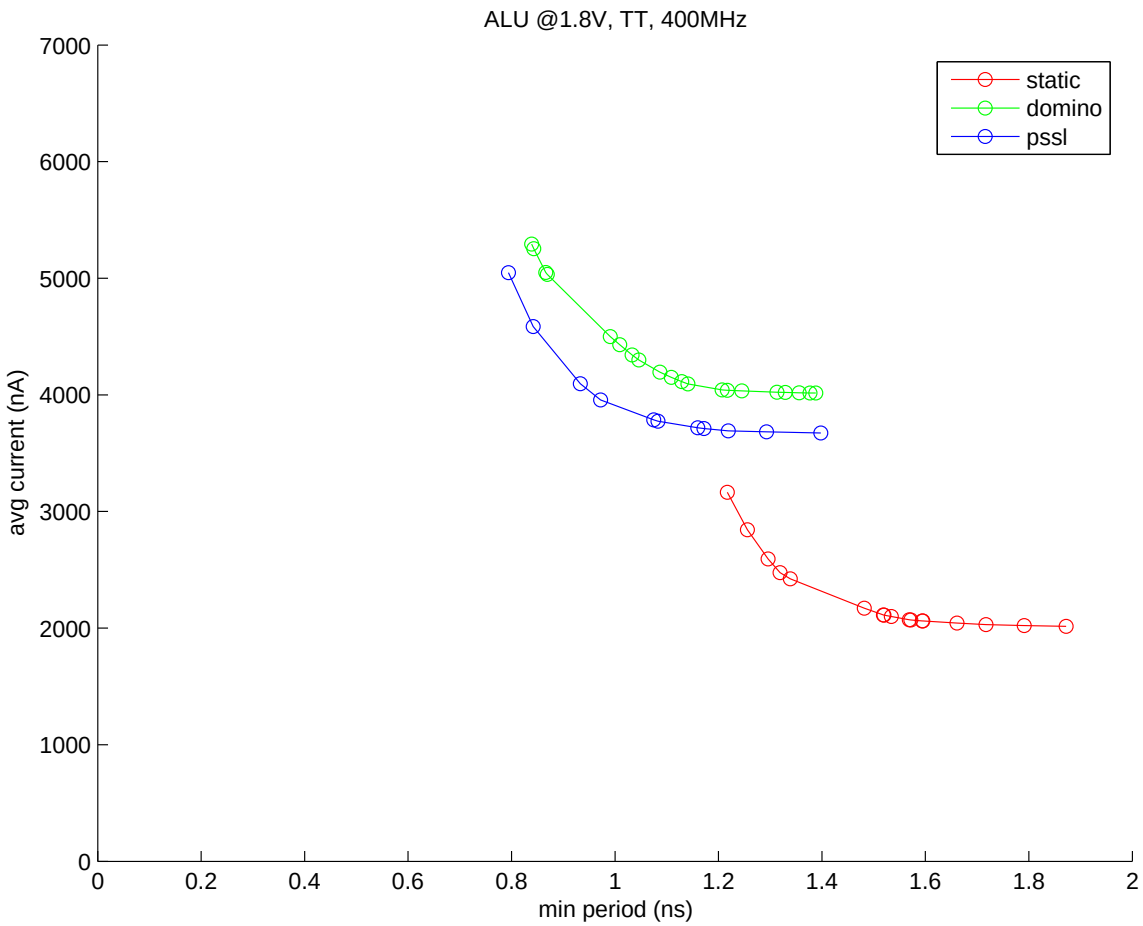


Figure 8-4: ALU energy-delay comparison with varying transistor sizes, TT, 25° C.

design to consume approximately the same energy as the PSSL design at 1.8V.

It can be argued that this comparison is unfair because the domino ALU should have been implemented at a faster design point, then run at a lower voltage for comparison against PSSL. Similar, the static ALU should have been implemented at a slower design point, then run at a higher voltage for comparison. However, we felt that this would make the comparison less clear. Also, it may be unreasonable to increase the voltage of the static ALU beyond 1.8V and risk gate breakdown.

8.2.2 Layout

A die plot of the testchip is shown in Figure 8-5. The chip was built using a semi-custom tool-flow. The datapath leaf-cells were drawn manually, but the datapaths were placed and routed procedurally, using custom software written by this author. All datapath leaf cells and standard cells were designed to a height of 9 wiring tracks, or 5.04 μm. The sizes for each ALU is shown in table 8.1.

style	width (μm)	height (μm)	area (μm^2)
Domino	97.68	176.4	17231
LS-PSSL	112.2	176.4	19792
Static CMOS	169.62	176.4	29921

Table 8.1: Testchip ALU size comparison.

8.3 Simulation Results

A comparison of the energy and delay of the ALUs in the testchip is shown in Table 8.2. The results are from simulation of fully extracted layout at 1.8V, TT, 25° C.

8.4 Test and Measurement Methodology

In order to generate an energy-delay curve for each ALU, we need to find the maximum frequency the ALU can operate at for various supply voltages. This is complicated by the fact that there is no direct control, without additional hardware and software support, over

ALU	speed (MHz)	energy (pJ/cycle)
PSSL	847	23.4
Static	687	25.0
Domino	868	27.0

Table 8.2: Testchip ALU Energy-Delay comparison. Simulated with fully extracted layout at 1.8V, TT, 25° C.

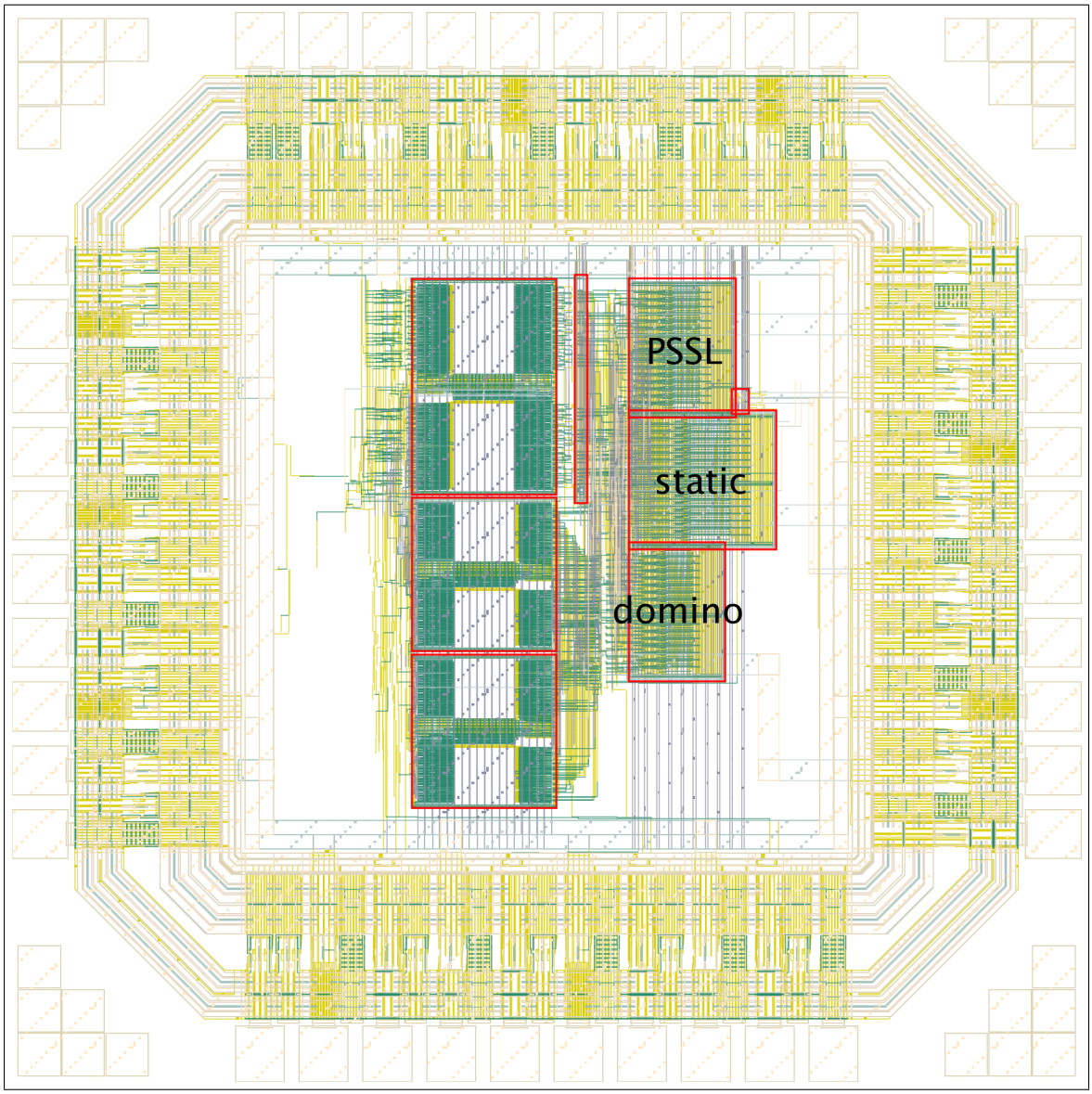


Figure 8-5: Testchip die plot.

the clock frequency and measuring the clock frequency is a manual process through the oscilloscope.

Therefore, we break the process into two steps. The first step is to measure the VCO clock frequency vs VCO input voltage. We start at a low input voltage and steadily increase the voltage by a small increment each time, all the while recording the clock frequencies.

In the second step we try to find the maximum VCO input voltage for each ALU supply voltage for which the ALU generates the correct output vectors. The step will be performed completely automatically by software. The process is as follows:

1. load the test input vectors
2. set the ALU select register
3. set the VCO input voltage to the minimum
4. set the ALU supply voltage to the minimum
5. wait a short time
6. read the output vectors
7. compare output vectors against expected results
8. if the vectors match then
   (a) record VCO input voltage, ALU supply voltage, average current draw
   (b) increase the VCO input voltage by some small fixed amount

   otherwise increase the ALU supply voltage by some small fixed amount.
9. repeat from step 5

We finally combine the data from the two steps to yield the final energy delay curve.

8.5 Conclusion

We have designed and submitted for fabrication a testchip demonstrating functional PSSL ALUs. Initial simulations indicate that the PSSL ALU has lower energy consumption than the equivalent domino logic ALU over some range of delays. We have also shown that it has higher performance than the equivalent static CMOS ALU over some range of energy consumption. This testchip is currently in fabrication.

Chapter 9

Conclusion

In this work we have introduced and demonstrated Preset Skewed Static Logic (PSSL), a novel high-performance, low-power, robust family of logic styles. Over a wide range of design points, PSSL is superior to domino and static CMOS logic in terms on energy and delay, and at the same time is more robust than domino. In particular, we have shown 20-30% energy reduction vs. domino and 33-50% delay reduction vs. static CMOS. In addition, we have presented the Double Pulsed Set Conditional-Reset Flip Flop (DPSCRFF), which is twice as fast as previous flip flops described in literature while consuming the same amount of energy.

9.1 Summary of Contributions

The main contributions presented in this thesis are:

- *PSSL design.* We have described the structure and operation of the novel PSSL logic style, and shown how to build pipelines using three PSSL clocking schemes: level-sensitive, edge-triggered, and pulsed.
- *Leakage reduction techniques.* We have proposed a fine grained leakage reduction technique that takes advantage of the unique properties of PSSL. This technique has reduced overhead when used in PSSL as compared to static CMOS.
- *PSSL evaluation.* We have tested PSSL and compared it against other common logic styles, using several test circuits from shift registers to accumulators. We have also demonstrated the advantages of the DPSCRFF, a novel pulsed latch based flip-flop, for high speed pipelines.
- *Testchip.* We have implemented a testchip in TSMC 0.18 μm technology containing 32-bit ALUs using the logic styles PSSL, domino, and static CMOS to demonstrate the feasibility of PSSL. This chip is currently in fabrication.

9.2 Future Work

There are several research directions that can be pursued based on this work:

Robustness

The robustness advantage of PSSL over domino can be more fully examined and quantified. Issues such as soft error, transient noise, and variability each present different failure modes and these effects should be fully analyzed.

Leakage Reduction

Further leakage reduction techniques should be explored. The leakage reduction technique proposed in this work can form the basis.

Synthesis and Related Tool Flow

Skewed logic styles present challenges to conventional synthesis tool flows. Unique opportunities may exist to improve the quality of synthesis results through novel algorithms and heuristics.

Bibliography

[1] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), April 1965.

[2] International technology roadmap for semiconductors. `http://public.itrs.net`.

[3] Deo Singh and Vivek Tiwari. Power challenges in the internet world. In *Cool Chips Tutorial: An Industrial Perspective on Low Power Processor Design*, pages 8–15, November 1999.

[4] Kaushik Roy, Saibal Mukhopadhyay, and Hamid Mahmoodi-Meimand. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. *Proc. IEEE*, 91(2):305–327, February 2003.

[5] Saibal Mukhopadhyay, Arijit Raychowdhury, and Kaushik Roy. Accurate estimation of total leakage in nanometer-scale bulk CMOS circuits based on device geometry and doping profile. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Sytems*, 24(3):363–381, March 2005.

[6] A. Agarwal et al. Leakage power analysis and reduction: models, estimation and tools. *IEE Proc. Comput. Digit. Tech.*, 152(3):353–368, May 2005.

[7] Arjit Raychowdhury, Saibal Mukhopadhyay, and Kaushik Roy. Modeling and estimation of leakage in sub-90nm devices. In *Int. Conf. VLSI Design*, pages 65–70, 2004.

[8] Gordon Moore. No exponential is forever: but "forever" can be delayed! In *ISSCC*, pages 20–23, February 2003.

[9] Ofri Wechsler. Inside Intel Core microarchitecture. `ftp://download.intel.com/technology/architecture/new_architecture_06.pdf`.

[10] Samuel K. H. Fung et al. 65nm CMOS high speed, general purpose and low power transistor technology for high volume foundry application. In *Symp. VLSI Tech.*, pages 92–93, 2004.

[11] Saibal Mukhopadhyay and Kaushik Roy. Modeling and estimation of total leakage current in nano-scaled CMOS devices considering the effect of parameter variation. In *ISLPED*, pages 172–175, August 2003.

[12] Rajeev R. Rao et al. Parametric yield estimation considering leakage variability. In *DAC*, pages 442–447, June 2004.

[13] David Blaauw Todd Austin, Valeria Bertacco and Trevor Mudge. Opportunities and challenges for better than worst-case design. In *Asia-South Pacific Design Automation Conference*, January 2005.

[14] Todd Austin. Diva: A reliable substrate for deep submicron microarchitecture design. In *MICRO*, pages 196–207, 1999.

[15] Dan Ernst et al. Razor: A low-power pipeline based on circuit-level timing speculation. In *MICRO*, pages 7–18, 2003.

[16] R. H. Krambeck, Charles M. Lee, and Hung-Fai Stephen Law. High-speed compact circuits with CMOS. *IEEE J. Solid-State Circuits*, 17(3):614–619, June 1982.

[17] Ivan Sutherland, Bob Sproull, and David Harris. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann Publishers, Inc., 1999.

[18] Y. Cao et al. New paradigm of predictive mosfet and interconnect modeling for early circuit design. In *CICC*, pages 201–204, 2000. `http://www.eas.asu.edu/~ptm`.

[19] M. Anders et al. Robustness of sub-70nm dynamic circuits: Analytical techniques and scaling trends. In *Symp. VLSI Circuits*, volume 3, pages 23–24, June 2001.

[20] Mohab H. Anis, Mohamed W. Allam, and Mohamed I. Elmasry. Impact of technology scaling on CMOS logic styles. *IEEE Trans. Circuits Syst. II*, 49(8):577–588, August 2002.

[21] S. K. Mathew et al. Sub-500-ps 64-b ALUs in 0.18 μm SOI/bulk CMOS: design and scaling trends. *IEEE J. Solid-State Circuits*, 36(11):1636–1646, November 2001.

[22] L. Cotten. Maximum rate pipelined systems. In *AFIPS Spring Joint Computer Conference*, pages 581–586, 1969.

[23] David Harris. *Skew-Tolerant Circuit Design*. Morgan Kaufmann Publishers, 2001.

[24] Albert Ma and Krste Asanović. A double-pulsed set-conditional-reset flip-flop. Technical Report 844, Laboratory for Computer Science, MIT, 2002.

[25] Muhammad E. S. Elrabaa, Mohab H. Anis, and Mohamed I. Elmasry. A contention-free domino logic for scaled-down CMOS technologies with ultra low threshold voltages. In *ISCAS*, volume 1, pages 748–751, 2000.

[26] Atila Alvandpour et al. A conditional keeper technique for sub-0.13 μm wide dynamic gates. In *Symp. VLSI Circuits*, volume 3, pages 29–30, 2001.

[27] Seong-Ook Jung et al. Skew-tolerant high-speed (STHS) domino logic. In *ISCAS*, volume 4, pages 154–157, 2001.

[28] Mohab H. Anis, Mohamed W. Allam, and Mohamed I. Elmasry. Energy-efficient noise-tolerant dynamic styles for scaled-down CMOS and MTCMOS technologies. *IEEE Trans. VLSI Syst.*, 10(2):71–78, April 2002.

[29] Fumio Murabayashi et al. 2.5V CMOS circuit techniques for a 200MHz superscalar risc processor. *IEEE J. Solid-State Circuits*, 31(7):972–980, July 1996.

[30] Tyler Thorp, Gin Yee, and Carl Sechen. Monotonic static CMOS and dual Vt technology. In *Int. Symp. Low Power Electronics and Design*, pages 151–155, San Diego, CA, 1999.

[31] David Harris, Genevieve Breed, Matt Erler, and David Diaz. Comparison of noise tolerant precharge (NTP) to conventional feedback keepers for dynamic logic. In *Great Lakes Symp. VLSI*, pages 261–264, 2003.

[32] Alexandre Solomatnikov et al. Skewed CMOS: Noise-tolerant high-performance low-power static circuit family. *IEEE Trans. VLSI Syst.*, 10(4):469–476, August 2002.

[33] L. McMurchie, S. Kio, G. Yee, T. Thorp, and C. Sechen. Output Prediction Logic: a high-performance CMOS design technique. In *Int. Conf. Computer Design*, pages 247–254, 2000.

[34] Sheng Sun et al. 409ps 4.7 FO4 64b adder based on output prediction logic in 0.18 μm CMOS. In *ISVLSI*, pages 52–58, May 2005.

[35] K. H. Chong, Larry McMurchie, and Carl Sechen. A 64b adder using self-calibrating differential output prediction logic. In *ISSCC*, volume 49, pages 440–441, February 2006.

[36] Xinyu Guo and Carl Sechen. A high throughput divider implementation. In *CICC*, pages 502–505, September 2005.

[37] Yi Han, Larry McMurchie, and Carl Sechen. A high performance CMOS programmable logic core. In *CICC*, pages 439–442, October 2004.

[38] D. J. Deleganes et al. LVS technology for the Intel Pentium 4 processor on 90nm technology. *Intel Technology Journal*, 8(1):49–58, February 2004.

[39] Sapumal Wijeratne et al. A 9GHz 65nm Intel Pentium® 4 processor integer execution core. In *ISSCC*, volume 49, pages 110–111, February 2006.

[40] Chris H. Kim et al. A process variation compensating technique for sub-90nm dynamic circuits. In *Symp. VLSI Circuits*, pages 205–206, June 2003.

[41] Yolin Lih, Nestoras Tzartzanis, and William W. Walker. A leakage current replica keeper for dynamic circuits. In *ISSCC*, pages 442–443, 2006.

[42] W. Lien and W. Burleson. Wave-domino logic: timing analysis and applications. In *ISCAS*, volume 6, pages 2949–2952, 1992.

[43] H. Partovi et al. Flow-through latch and edge-triggered flip-flop hybrid elements. *ISSCC*, pages 138–139, February 1996.

[44] F. Klass et al. A new family of semidynamic and dynamic flip-flops with embedded logic for high-performance procesors. *IEEE J. Solid-State Circuits*, 34(5):712–716, May 1999.

[45] V. Stojanović and V. Oklobdžija. Comparative analysis of master-slave latches and flip-flops for high-performance and low-power systems. *IEEE J. Solid-State Circuits*, 34(4):536–548, April 1999.

[46] B. Nikolić et al. Improved sense-amplifier-based flip-flop: Design and measurements. *IEEE J. Solid-State Circuits*, 35(6):876–884, June 2000.

[47] M. Golden et al. A seventh-generation x86 microprocessor. *IEEE J. Solid-State Circuits*, 34(11):1465–1477, November 1999.

[48] R. Heald et al. A third-generation SPARC V9 64-b microprocessor. *IEEE J. Solid-State Circuits*, 35(11):1526–1538, November 2000.

[49] Liqiong Wei et al. Design and optimization of dual-threshold circuits for low-voltage low-power applications. *IEEE Trans. VLSI Syst.*, 7(1):16–24, March 1999.

[50] Pankaj Pant, Vivek K. De, and Abhijit Chatterjee. Simultaneous power supply, threshold voltage, and transistor size optimization for low-power operation of CMOS circuits. *IEEE Trans. VLSI Syst.*, 6(4):538–545, December 1998.

[51] Yibin Ye, Shekhar Borkar, and Vivek De. A new technique for standby leakage reduction in high-performance circuits. In *Symp. VLSI Circuits*, pages 40–41, June 1998.

[52] M. C. Johnson, D. Somasekhar, and K. Roy. Leakage control with efficient use of transistor stacks in single threshold CMOS. In *DAC*, pages 442–445, 1999.

[53] G. Yang, Z. D. Wang, and S. M. Kang. Leakage-proof domino circuit design for deep sub-100nm technologies. In *Int. Conf. VLSI Design*, pages 222–227, 2004.

[54] Walid Elgharbawy et al. On gate leakage reduction in dynamic CMOS circuits. In *Midwest Symp. Circ. Syst.*, pages 1390–1393, 2005.

[55] Seongmoo Heo and Krste Asanović. Leakage-biased domino circuits for dynamic fine-grain leakage reduction. In *Symp. VLSI Circuits*, pages 316–319, 2002.

[56] Shin'ichiro Mutoh et al. 1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS. *IEEE J. Solid-State Circuits*, 30(8):847–854, August 1995.

[57] Hiroshi Kawaguchi, Koichi Nose, and Takayasu Sakurai. A super cut-off CMOS (SCCMOS) scheme for 0.5-V supply voltage with picoampere stand-by current. *IEEE J. Solid-State Circuits*, 35(10):1498–1501, 2000.

[58] K. Min, H. Kawaguchi, and T. Sakurai. Zigzag super cut-off CMOS (ZSCCMOS) block activation with self-adaptive voltage level controller: An alternative to clock-gating scheme in leakage dominant era. In *ISSCC*, pages 400–401, 2003.

[59] Mindaugas Dražd̄iulis and Per Larsson-Edefors. Evaluation of power cut-off techniques in the presence of gate leakage. In *ISCAS*, volume 2, pages 745–748, 2004.

[60] Mindaugas Dražd̄iulis et al. A power cut-off technique for gate leakage suppression. In *ESSCIRC*, pages 171–174, 2004.

[61] S. Heo, R. Krashinsky, and K. Asanović. Activity-sensitive flip-flop and latch selection for reduced energy. In *ARVLSI*, Salt Lake City, UT, March 2001.

[62] S. Heo and K. Asanović. Load-sensitive flip-flop characterization. In *IEEE Workshop on VLSI*, Orlando, FL, April 2001.