UC Santa Cruz

UC Santa Cruz Electronic Theses and Dissertations

Title

Voltage Stacking and Timing Speculation with an SRAM Focus

Permalink

https://escholarship.org/uc/item/1hj7d74m

Author

Ebrahimi, Elnaz

Publication Date

2017

Copyright Information

This work is made available under the terms of a Creative Commons Attribution License, availalbe at https://creativecommons.org/licenses/by/4.0/

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

VOLTAGE STACKING AND TIMING SPECULATION WITH AN SRAM FOCUS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

by

Elnaz Ebrahimi

March 2017

The Dissertation of
Elnaz Ebrahimi is approved:

Professor Jose Renau, Chair

Professor Matthew Guthaus

Professor Jishen Zhao

Tyrus Miller
Vice Provost and Dean of Graduate Studies

Copyright © by

Elnaz Ebrahimi

2017

Table of Contents

List of Figures

List of Tables

Abstract

Voltage Stacking and Timing Speculation with an SRAM Focus

by

Elnaz Ebrahimi

Power consumption and delivery have emerged as one of the major challenges facing modern SoC design. As chip designs become more complex with aggressive architectures, pressure on efficient power delivery mechanisms is increasing.

Designing efficient voltage regulators gets harder due to increased current as power increases and voltage scales down. Off-chip voltage regulators can be made more efficient due to fewer restrictions on capacitor sizes but need to take into account parasitics on pads. Process variation further exacerbates the problem, because the design needs to take into account the worst case. As technology scales towards nano-scale and with the prevalence of having multiple voltage domains on the chip, voltage stacking offers an alternative in Power Delivery Network (PDN) design that alleviates conventional power delivery inefficiencies.

The first part of this dissertation explores different types of existing level shifters suitable for a voltage stacked logic, their optimal sizing, and the effect of PVT variation on delay and energy consumption.

In the second part of the dissertation, instead of inserting an SRAM into a voltage domain, as it is the common case, the SRAM logic itself is divided into multiple domain. The symmetric logic of SRAM is leveraged for a stacking technique and is divided into two logic domains, and the supply voltage V_{dd} is doubled. The supply voltage $2V_{dd}$ will distribute evenly

between the stacks and the current demand will decrease up to half. Hence, the same amount of power is delivered, but with half the current.

The third part of this dissertation builds upon the idea of a floating voltage level in a voltage stacked system and on the observation that slower transistors have higher impedance in the presence of process variation. This chapter offers a GPU stacking method based on voltage stacking to manage the effects of process variation and improve the power delivery simultaneously. The evaluation conducted in this dissertation considers Near Threshold Computing (NTC), because the effects of process variation are more severe in this scenario, however, the technique can be applied without the use of NTC. Using GPU Stacking brings the chip distribution closer to the nominal, *i.e.*, no process variation, and is shown to be better than simply using multiple clock domains, which is the current state-of-the-art.

The final contribution of the dissertation looks at SRAM design, specifically into reducing voltage and timing margins. I propose a timing speculative SRAM that extends the existing Replica Bit-Line (RBL) technique to detect read timing failures. And to protect it from incorrect write operations, the SRAM decode logic is extended. The Replica-based Timing Speculative SRAM (RTS) is evaluated as an energy and area efficient design alternative to prior techniques such as Razor-enabled SRAMs.

Dedicated to my parents, Maryam and Hossein

Acknowledgments

The journey of my doctoral studies has been one of the most demanding and yet transformative and rewarding phases of my life. It truly enriched both my academic and my personal life and there are number of people I would like to thank for their immeasurable support and guidance.

Doctoral studies are known to be inherently challenging, and there were many times when I lacked inspiration and energy, however, I was surrounded with family and friends who inspired me and encouraged me to stay positive and continue with persistence. My sincere thanks goes to my advisor, Professor Jose Renau, who never fails to inspire, motivate, and provide guidance. His vast knowledge of computer architecture and his multi-angle research interest opened the door for me to pursue various projects which helped me learn different languages, design techniques, and problem solving skills.

I would like to express my gratitude to my committee members, Professor Matthew Guthaus and Professor Jishen Zhao, who kindly accepted to serve on my dissertation committee. Professor Guthaus has provided me with valuable feedback and research guidance throughout the years. Taking his VLSI courses and seminars has helped me immensely in my research.

During my time as a Ph.D. student, I have been involved in various projects, some of which are not included in this dissertation, yet they have been a product of collaborative efforts with MicroArchitecture Santa Cruz (MASC) laboratory colleagues whom I would like to acknowledge. Rafael Trapani and I have collaborated on a few projects related to “Fluid Pipelines”, a repipelining idea applied to retry-based or elastic pipelines and we have published our results in IWLS’16 [80] and ICCD’16 [79], and presented a poster in DAC’15 [78]. Our most recent collaboration on “Level Shifter Design for Voltage Stacking“ will be presented at

ISCAS'17. I would like to thank Dr. Ehsan Ardestani and Dr. Gabriel Southern with whom I have collaborated on a "Thermal Sampling" project which resulted in multiple peer-reviewed publications: ISLPED'12 [33] and TCAD'13 journal [34]. These projects have strengthened my knowledge of VLSI design and architecture design. And I will be forever grateful to them for such rich joint efforts.

To all my dear colleagues in the MASC Laboratory, Gabriel, Rafael, Ehsan, Daphne, Michael, Ian, Sina, David, Raj, Jason, Tom, Alamelu, Ethan, Blake, Akash, and Ramesh, thank you for the stimulating discussions, for the sleepless nights we were working together before deadlines, and for the positive environment you helped create which made the long working hours untiring.

The continued love and support that I have received from my family over the course of my doctoral studies has been an integral part of my journey. I am thankful to my husband, Mehdi, for always encouraging me to focus my full energy on my research and for always being patient and compassionate when I worked long hours for days and months. Last but not least, I am thankful to my brother and sister for continuously sending me positive energy. I could not have made this journey without all of your help.

Chapter 1

Introduction

The power delivery network is an integral part of the chip design where it has to be robust enough to ensure every device on the chip accesses stable voltage, and meets the timing and functionality requirements. One of the challenges in current SoC design is that there is not a single power domain on the chip, and the number of domains continues to grow as the technology node scales down. As these voltage domains are sharing the same piece of silicon and routed through the same package, they need to be optimized without jeopardizing the voltage drop [72].

One of the most effective techniques for reducing the power consumption is dynamic voltage scaling [86]. Since the dynamic power dissipation in a digital circuit is cubically proportional to the supply voltage, lowering the supply voltage will decrease the dynamic and short-circuit power dissipation [31]. However, it will increase the propagation delay of the circuit and require slowing down of the clock frequency of the circuit, leading to an overall quadratic reduction in the energy to complete a task [86]. As a result, the supply voltage of the components present in the critical path cannot be modified due to speed constraints of the de-

sign. One way to overcome this problem is to use multi-VDD systems [49], where components present in the critical path will work with the standard supply (VddH) and those present in the non-critical path will work with the scaled-down supply voltages (VddL). This leads to different voltage islands, and in order to communicate among each other, an interface circuit is required for voltage level conversion. These circuits are called level converters or level shifters [31, 63]. Also, in a chip with multiple supply voltages, there is need for extra voltage regulators, either on-chip or off-chip. Off-chip DC-DC converters suffer from slow voltage transition times, while the high board-level footprint and system cost limit the number of voltage domains that can be implemented. On-chip voltage regulators enable fast voltage transition times and facilitate multiple voltage domains, but such regulators suffer from low conversion efficiency, especially for high step-down ratios [52].

In addition to adding propagation delay, lowering the supply voltage leads to a rapid increase in the supply current requirements. With increasing current transients and average current levels, more on-chip decoupling capacitance is required while the resistance and inductance of the power distribution network (including on-chip wiring, pins, sockets and connectors) must be kept stringently low for supply integrity. High current requirements also exacerbate on-chip electromigration concerns [66, 81], reduce efficiency of voltage regulators [7, 53], increase voltage noise and losses due to parasitics [25, 52], and increase the number of dedicated power pins [87].

Voltage stacking is an alternative to delivering power in a multi-VDD system that mitigates the aforementioned challenges [25, 52, 87]. Voltage stacking uses the concept of charge recycling [66] and delivers power to logic blocks "stacked" or connected in series as opposed to the usual parallel scheme. A high voltage is applied to the system and divided to deliver power

to each of the logic blocks by charge recycling. Using the stacking technique draws less current to the stacks (voltage domains) without altering the original power budget. Voltage stacking greatly reduces the amount of current drawn. However, it presents a challenge in stack load balancing. If there is current mismatch between the stacks, voltage noise will be introduced in the stacked architecture. Hence, it is a challenge to keep the loads in stacks balanced. Prior approaches for voltage stacking relied on extra voltage regulators to deliver stable voltage to each level [25, 52]. This is mainly required because those approaches applied voltage stacking to full cores that were often running different applications.

Voltage stacking can be applied to SRAMs. SRAM is an indispensable part of most modern Very-Large-Scale Integration (VLSI) designs and dominates silicon area in many applications. 20% of chip area is taken by SRAMs and that is when up to 60% of the total active chip area is taken by memories, which translate to a large amount of chip power consumption [2, 64]. In SRAM voltage stacking the logic can be divided into stacks with a much better balancing than what is observed in a full core. If the activity and power consumption in the voltage stacks are balanced, the shared voltage rails are less susceptible to noise. I leverage the SRAM structure to show how the stacking technique benefits memory subsystems.

The second proposal for voltage stacking application is to use it to compensate for process variation. Process variation affects transistors so that slower transistors will present higher impedance. I will show that assuming balanced activity, and a floating rail between stacking levels, stacking transistors with opposite variation would create a "natural compensation mechanism, because the slower transistor would have higher voltage drop. The challenge in such a scenario would be to guarantee the balanced activity. GPUs parallelize the workload and maintain the performance level across multiple processing units (PEs), and thus are an ideal

candidate for this application.

Power dissipation can be reduced by other techniques in SRAMs such as removing the timing guardbands with the support of an error detection logic. SRAMs are failure prone due to PVT variations and timing guardbands keep SRAMs less sensitive to such variations. However, strict guardbands make it more challenging to meet the frequency requirements, and in that regard SRAMs are overdesigned. There is a demand for architectural alterations which ease the frequency margin without requiring large area or power overhead. We leverage the existing RBL technique to detect timing errors and offer power savings without additional area overhead and design complexity.

In this thesis, I present two different applications and evaluations of voltage stacking, first as a way to reduce overall current in SRAMs operating in nominal voltage levels, with good activity balancing across stack levels and then using GPUs at the near threshold region, to compensate for process variation. I also present a power efficient Replica Bitline Column-based timing speculative technique for SRAMs.

Chapter 2 surveys existing research on level shifter designs suitable for voltage stacked systems. The analysis includes a Pareto optimal plot of existing designs (considering performance, power, and area) and evaluates the robustness of designs while undergoing process variation such as variation of effective length, width, and threshold voltage.

Using one of the level shifters in Chapter 2, I propose a novel voltage stacking technique for SRAMs in Chapter 3. The typical solution to has been to avoid stacking SRAMs or stack the whole SRAM without leveraging the internal structure which happens to be suitable for voltage stacking. This chapter presents the design process of a voltage stacked SRAM, the implementation of the level shifters in the design, and the evaluation of stacked versus non-

stacked SRAMs in terms of energy consumption. This is the first work to propose voltage stacking with a floating voltage rail.

Chapter 4 uses the efficiently stacked SRAM design from Chapter 3 in a GPU environment operating at near threshold voltage. For the first time, this voltage stacking technique is used to compensate for process variation in GPUs. This chapter includes details on how to stack with uneven voltage levels, how to mitigate process variation in NTC, and how to configure the design, post-silicon.

Chapter 5 presents the final contribution of the thesis, a novel timing error detection scheme with minimal area and power overhead. It shows how to leverage an existing RBL technology to detect read timing errors due to process variation with minimal area and power overhead.

Finally, Chapter 6 concludes how power consumption limitations in current architecture design leave researchers in search of methods to design in a power-aware fashion, while the complex and intensive computations will not allow them to compromise performance. This chapter recapitulates elements from previous chapters to further emphasize the need for simple yet novel ideas such as stacked architectures to ease the power delivery and management in chip power network design cycle.

Chapter 2

Level Shifters for Voltage Stacked Architectures

2.1 Introduction

Using a multi-VDD system is an alternative to voltage scaling technique. It counteracts the negative impact on performance, because the critical path components will continue running at V_{dd} level while non-critical components run at a scaled down V_{dd} [63]. In a multi-VDD system, when the DC current flows from a low voltage gate to a high voltage gate, the voltage is not sufficient to turn the PMOS "ON"' and therefore, the PMOS in the high voltage gate is weakly "ON"' conducting static current from the power supply to the ground. The level shifters will remove the static current and restore the full voltage swing from VddL to VddH [31].

Designing a multi-VDD system is inherently complex as there are a few challenges in using a Level Shifter (LS) in the system. They dissipate power and add propagation delay. It is

necessary to optimize the LS circuit for minimum energy-delay product to obtain the potential benefit of using multiple power supply domains. As an LS includes both high voltage and low voltage gate, it will require more area and routing resources. For example, when each functional block on a die needs a different voltage for its desired performance, the number of level shifters can easily grow and become a design area overhead. Techniques such as Dynamic Voltage scaling (DVS) have been widely used in digital signal processing elements for reducing energy consumption [86]. And future low-power systems-on-chips (SoCs) are likely to consist of many scalable voltage domains. This requires level shifters to be able to perform at a high speed with low power [63, 91].

As in the case of multi-VDD systems, voltage stacked systems require level shifters for inter-level communication [25, 53]. Traditional level shifters are inserted to translate or shift the logic levels from the level supplied by one domain to another level supplied by the second domain. In the context of voltage stacking, the level shifters will have a primary voltage rail which sits at the top and a secondary voltage rail which sits in the middle. When placed in a voltage stacked design, they will shift the both rails, either from GND-midrail to midrail-toprail (low to high level shifters) or from midrail-toprail to GND-midrail (high to low level shifters). Although many designs for level shifters exist, an evaluation of different designs in the context of voltage stacking has not been made, so the trade-offs of different designs are not clear. As the starting point for our research, we evaluate existing approaches of LSs for voltage stacking applications. We are especially interested in delay and power, but area and sensitivity to PVT (Process, Voltage and Temperature) variations are also considered. Each of those parameters may have different importance in different designs. For instance, CoreUnfolding [7] allows an entire clock cycle for level shifting, thus delay is less important. However, it requires a large

amount of shifters, which makes area a critical design factor. On the other hand, a voltage stacked SRAM [20] requires minimal impact on timing, but due to the small number of shifters, can tolerate more area overhead per shifter.

The contributions of this chapter are as follows:

- Overview of current level shifter designs.
- Comparison of level shifters in terms of energy, delay, and area.
- Comparison of level shifters in terms of PVT tolerance.

2.2 Overview of Level Shifter designs

This section explores some of the level shifter designs that are suitable for a stacked architecture and their operational behavior. Figure 2.1 to Figure 2.4 include the schematics for a mix of LS designs used in this study and each circuit diagram shows how the LS converts Vin "low" to Vout "high". Note that all of the chosen level shifters are bidirectional. This exploration focuses on converting in a stacked architecture where the primary/top voltage rail is 2V and the middle voltage rail is 1V. The signals are shifted from 0-1V voltage domain to operate in 1-2V voltage domain and vice versa.

2.2.1 Capacitive-Coupling-based (Conventional)

Gu *et al.* designed the capacitive-coupling-based LS shown in Figure 2.1 for a multi-story or voltage stacked power delivery scheme as they did not see the traditional level shifters a good fit for such systems. This LS has a driving inverter, a coupling capacitor, and a receiver with gain stages. Two diodes are connected back to back in order to constrain the voltage

swing at the output node of the coupling capacitor (gate and drain are shorted in the NMOS transistors). Since it always settles near the inverter trip point, a signal transition takes place with a minimal size coupling capacitor [25].

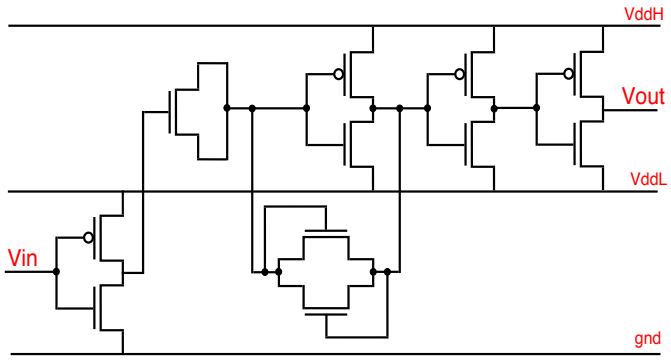


Figure 2.1: Capacitive-Coupling-based (Conventional)

2.2.2 Two-Stage Cross-Coupled (TSCC)

The conventional level shifter shown in Figure 2.2(a), is a differential cascade voltage switched logic gate, using a cross-coupled PMOS half latch operating at the higher supply voltage. To overcome the leakage of weakly conducting PMOS transistors, drive strength of NMOS transistor is enhanced. Low Vin input voltage turns mn1 on, which discharges node A to ground and activates mp2. Node B will be pulled up to VddH and the output voltage will be low. Subsequently, when Vin is asserted, mn2 and mp1 are activated shifting the output voltage up to VddH. The drive of the pull-down transistors needs to be much larger than the PMOS transistors to overcome the PMOS latch action driven with a higher supply voltage. It is a simple design suited for super-threshold conversion [26, 32, 58].

In order to achieve full voltage swing, two cross-coupled level-shifting stages are cascaded. The diode-connected NMOS transistor is employed to weaken the pull-up network

in the 2.2(b) structure which expands the convertible input voltage. The operating range is determined by the transistor size and the threshold voltage. As this is a two-stage cross-coupling structure, the area automatically increases in comparison with Cross-Coupled LS [32, 46, 91].

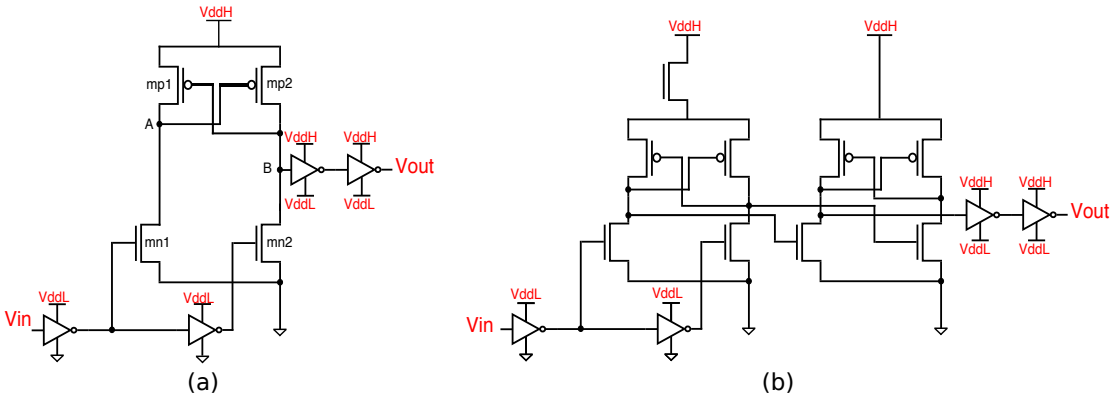


Figure 2.2: (a) Cross-Coupled (CC) (b) Two-Stage Cross-Coupled (TSCC)

2.2.3 Wilson Current Mirror (WCM)

Wilson Current Mirror design is based on the topology of another conventional LS called Current Mirror (CM). CM is a unity gain current amplifier which provides output current proportional to its input current at its high impedance output node. It also maintains the output current constant regardless of the load [3, 58]. The high drain-to-source voltage of PMOS transistors facilitates the construction of a stable current mirror, which offers an effective on-off current comparison at the output node. However, for super-threshold input voltage, a high amount of quiescent current occurs. Because of this high power consumption limits the use of the CM [32]. Figure 2.3(a) shows the schematic of the WCM. It uses a feedback PMOS (mp3) to cut off the static current flowing through mn1 and mp1 after switching. This reduces the standby power in the Current Mirror. However, as the source current is cut off, the mirror

current flowing through mp2 is largely reduced, resulting in weakened pull-up strength and a voltage drop at node A. Although the voltage drop increases the source current through the feedback control, the current increase is too small to pull the voltage at node A back to VddH. The output finally stabilizes at a voltage below VddH, which causes large static current and standby power in the output buffer of the LS [91].

2.2.4 Stacked WCM

Figure 2.3(b) is an enhancement to the WCM design [48] and uses a stacking technique to reduce the leakage power consumption. There is an addition of 3 NMOS transistors in the pull-down network.

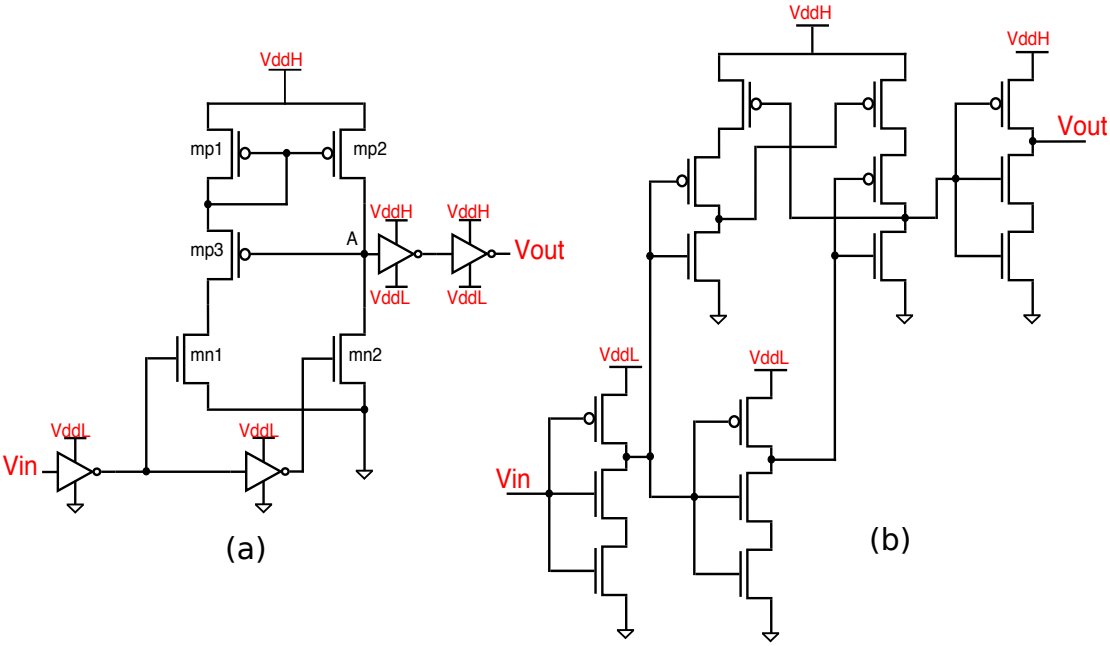


Figure 2.3: (a) Wilson Current Mirror (WCM) (b) Stacked Wilson Current Mirror (Stacked)

2.2.5 Tong

Figure 2.4(a) shows a capacitive-coupled LS that has been used in a 16-core voltage-stacked system [53]. The voltage across the coupling capacitor depends on the difference between the two voltage domains but it can be up to ≈2.7V, much higher than the gate-oxide breakdown voltage of the transistors available. Hence, this approach requires the use of metaloxidemetal (MOM) capacitors [77]. The original design has one 25fF capacitor on each side of the back to back inverters. If we translate each fF to $\approx 1um^2$, the LS area is considerably large. Our experiments show that the 25fF capacitors are over-designed for an LS, and we were able to reduce that number to ≈2.6fF (details in the experimental section), taking into consideration a 30% margin over the minimum operation point. Even with this size reduction, the overall area is still large.

2.2.6 Modified Tong

Since the capacitors occupy a large area in Tong LS (Figure 2.4(a)), utilizing it in a stacked domain can be problematic. We have modified it by replacing each capacitor with two NMOS transistors connected back to back, where the drain and gate are shorted (Figure 2.4. This reduces the area, but is expected to increase the power consumption due to the resistive effect added.

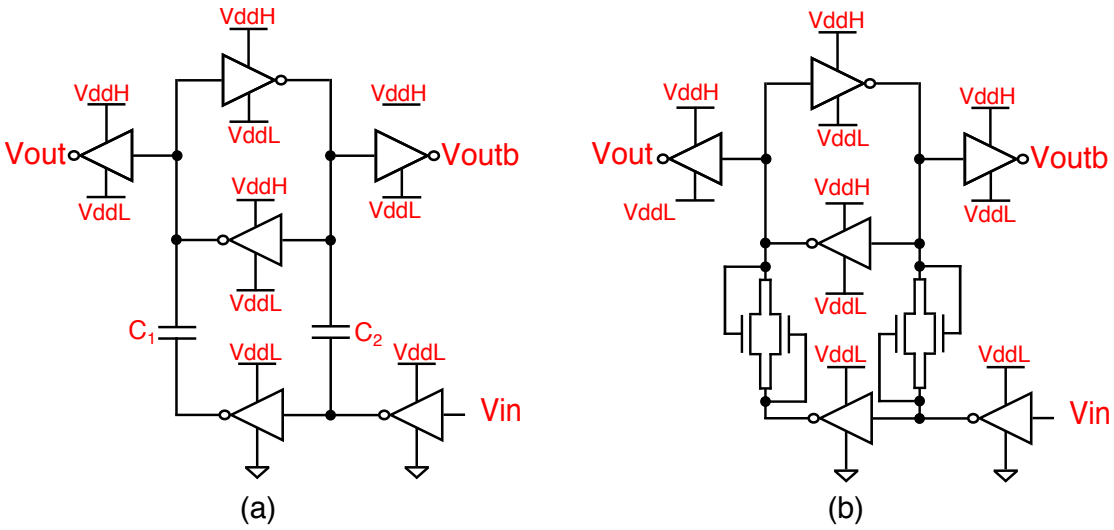


Figure 2.4: (a) Tong (b) Mod-Tong

2.3 Characterization

2.3.1 Transistor Sizing

To setup the level shifters for energy, delay, and area comparison, we begin by determining the optimal size for each LS. Our experiments use the NCSU FreePDK, the Open-Access-based PDK for the 45nm technology node [73]. For each CMOS transistor in an LS circuit, SPICE is run while varying the width from 90nm to 720nm. Energy and average propagation delay for an input signal with 5% slew rate are calculated. The resulting points are plotted in 2.5. Since each LS operates bidirectionally, the same experiment is repeated for level down shifters (Figure 2.5).

There is a Pareto frontier for each LS in Figure 2.5 and any of the frontier points could be of interest for a specific LS depending on its application. The minimum energy-delay product (ED) point is our point of interest for optimal sizing of an LS (It is circled in the plot and listed in Table 2.1. The power and delay are measured for a 1GHz input pulse as *active* energy

and *idle* energy, *i.e.*, when the input voltage is kept constant. We estimate the area considering it is proportional to the sum of widths of transistors in the design and the area of one transistor (45nm$x$90nm).The capacitors in Tong LS dominate the area, 7x the area of Mod-Tong. The top Pareto point is not shown for TSCC down shifter in Figure 2.5, because the delay is more than 250ps.

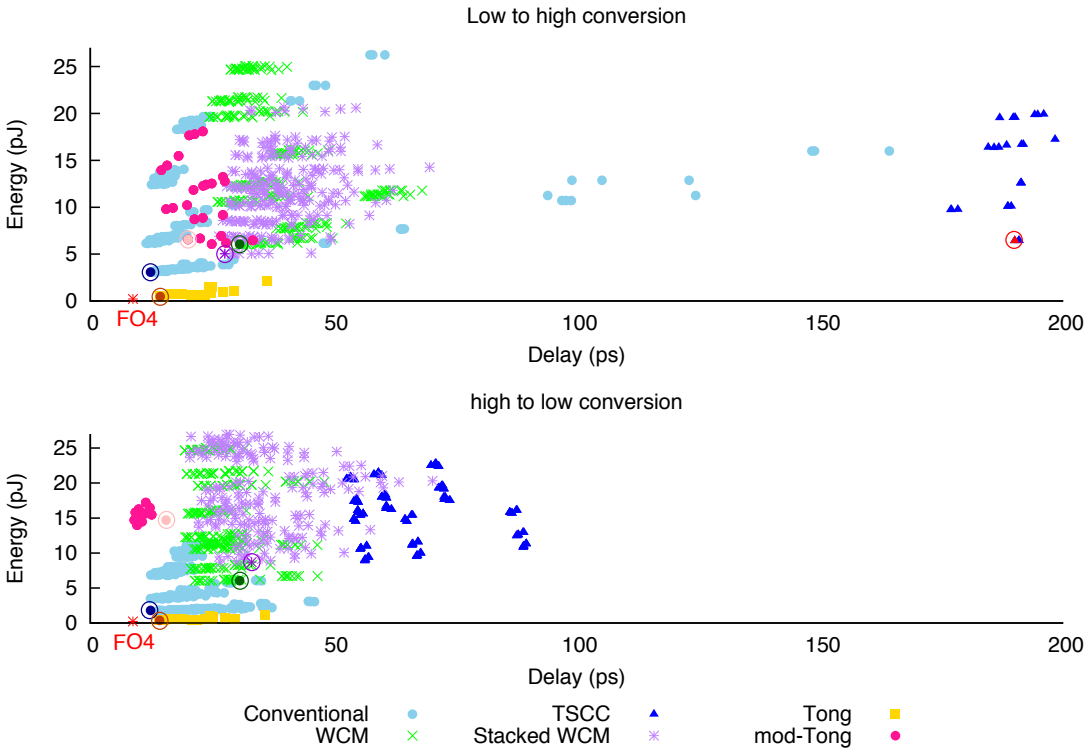


Figure 2.5: Level-up shifters active energy-delay with different transistor widths

Looking at the Pareto points in table 2.1, there is not a single level shifter that is superior in area, delay, and energy. For instance, if minimum delay in the level up shifters is the main design concern, Conventional and Tong are best choices, however, they impact the area significantly as Tong uses two 2.6fF capacitors and Conventional uses a decoupling capacitor and a diode [10]. Also, when down converting, Mod-Tong presents the best delay, but Tong

has the best Energy. Thus, it is not possible to pick a "winner" although Tong, Conventional and Mod-Tong have overall best numbers. TSCC has by far the worst delay in this case.

Table 2.1: Minimum ED Pareto frontier points

Low to high conversion

name	Area	Delay (ps)	Active Power (pW)	Idle Power (pW)
Conv	2790	12.37	3.07	0.53
TSCC	2250	190	6.42	0.53
WCM	2610	30.69	6.04	0.41
Stacked WCM	3060	27.64	5.04	0.37
Tong	2430	14.36	0.48	0.001
Mod-Tong	3330	20.09	6.57	0.75

High to low conversion

name	Area	Delay (ps)	Active Power (pW)	Idle Power (pW)
Conv	2790	13.99	1.79	0.24
TSCC	1980	56.39	8.89	1.11
WCM	2610	21.51	6.04	0.41
Stacked WCM	1980	26.72	8.69	1.43
Tong	2160	14.13	0.36	0.01
Mod-Tong	3690	9.06	14.7	3.40

2.3.2 PVT Variation Effect

An integral deciding factor is LS robustness in presence of Process, Voltage, and Temperature (PVT) variation. To see how temperature affects the delay and energy consumption, we perform an SPICE temperature sweep from 10°C to 90°C. Figure 2.6 shows the trend when converting from high to low.

When shifting down, Tong delay line has a slope of $0.046 ps/^{\circ}$C and there is less than 5ps difference in delay as the temperature rises up to 90°C. Conventional and mod-Tong have close slopes of 0.068 and $0.072 ps/^{\circ}C$ respectively which translates to less than 10ps delay difference across different temperatures. WCM and Stacked WCM each have 3.4x and 4.3x the slope of Tong LS which is equivalent to delay range of up to 16ps and up to 13ps respectively. When shifting up, as the temperature increases, so does the average propagation delay, however, the delay increase is minimal for Conventional, Tong, and mod-Tong: 0.065, 0.053, and $0.082 ps/^{\circ}C$. TSCC follows an inverse trend of decreasing delay (22 times the slope of Tong's), where rise and fall delay vary $\approx$ 100ps from start to finish. The delay itself is large as the whole circuit is slower when up shifting than it is when down shifting (Table 2.1). WCM and Stacked WCM each have a slope twice as steep compared to Tong which translates to 10ps delay range.

Active energy has a small decreasing trend during the up and down conversion in all the level shifters except for Tong where the line slope is $\approx 0 pJ/^{\circ}$C. For up conversion, Tong and Conventional with slopes of 0.003 and -$0.058 pJ/^{\circ}C$ are the best candidates and mod-Tong has the highest slope, -$0.023 pJ/^{\circ}C$. However, for all the level shifters the range that energy varies is less than 2pJ. During the down conversion as temperature increases, TSCC and Stacked WCM are affected the most with slopes of -0.065 and $0.093 pJ/^{\circ}C$ which translates to an energy range

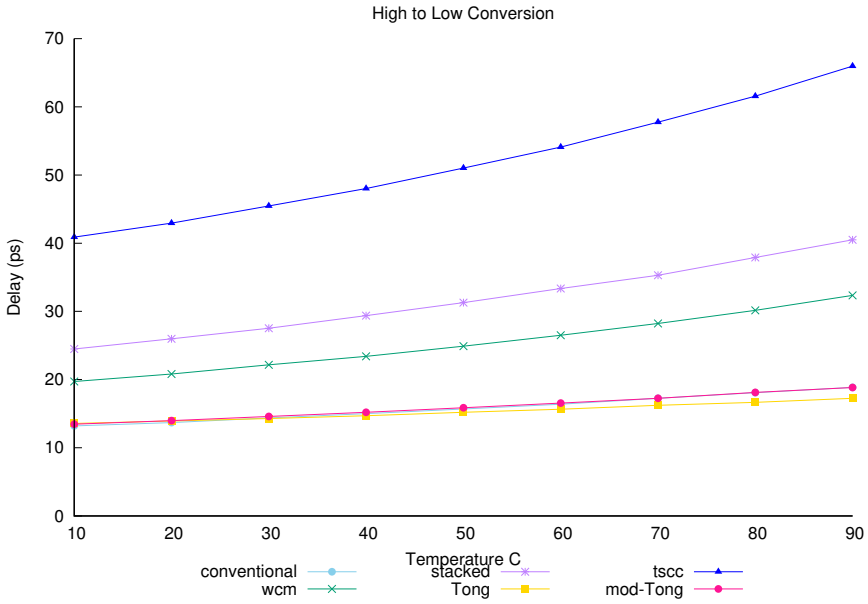


Figure 2.6: Delay vs. temperature. In high to low conversion, as the temperature rises, the average propagation delay will increase. Tong is the least sensitive and TSCC is the most sensitive to temperature variation.

of 7pJ and 8pJ. The least sensitive to varying temperature are Tong and Conventional with slopes of 0 and -0.006$pJ/^{\circ}C$. Overall, as the temperature increases up to 90°C, taking delay sensitivity into consideration takes priority over the power sensitivity.

Continuing the PVT variation effect analysis, we use an HSpice Gaussian distribution function with absolute variation to vary the threshold voltage +/-6% with 3σ value or 99.7% yield and run 5000 Monte Carlo simulations. In this experiment, we vary NMOS and PMOS threshold voltage by +/-6% and measure delay and energy in the *active* mode. Figure 2.7 is the final plot and an FO4 delay point has been included as a point of reference. When up shifting, the energy variation is a few pJs for all the level shifters , however, the delay variation is not small. For example, WCM delay varies ≈30ps. TSCC points have been removed from the up

shifters plot, where the delay varies from 100ps to 450ps whereas energy varies from 4.3pJ to 6pJ and is comparable to that of WCM and Mod-Tong. Tong seems to be the least sensitive to the Vth variation due to having capacitances. There is an energy-delay trade-off between Mod-Tong and Stacked WCM. Mod-Tong has a smaller delay variation whereas Stacked WCM power consumption varies less. When downshifting, the energy variation for all is less than 5pJ. And again, TSCC is the most sensitive with the largest delay range of ≈10ps. TSCC is a two-stage LS, and therefore, is affected by variation differently. Other LS types have nearly half the number of transistors as TSCC and therefore, their delay range is almost half as TSCC.

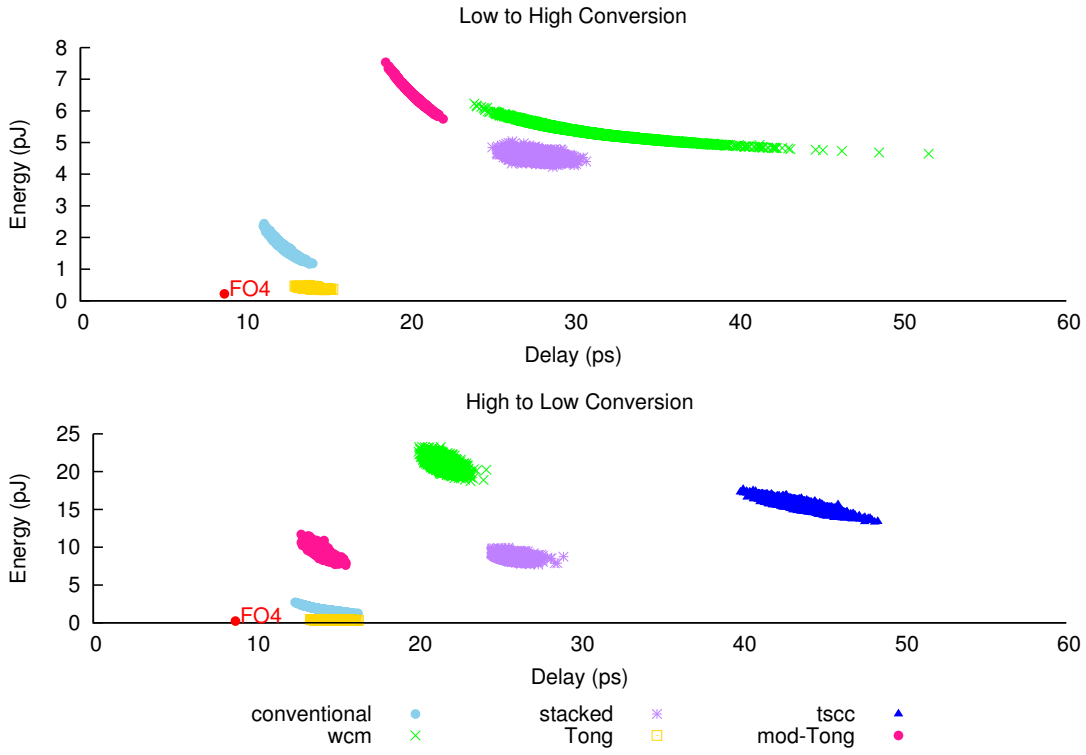


Figure 2.7: Active energy-delay: +/-6% Vth variation MC simulations

We repeat the same experiment and vary the supply voltages by +/-5% (Figure 2.8). Unlike previous experiment, Tong seems to be sensitive as the delay varies both in up conver-

sion and down conversion with an 11ps and 18ps range respectively. In up conversion, mainly the delay difference separates the choices. The most unpredictable delay belongs to TSCC, 100ps, and Tong comes in second. However, the energy variation is less than 2pJ for TSCC. Conventional or Mod-Tod might be better choices for down conversion as both energy and delay vary a few units. When downshifting, WCM and Stacked WCM are more sensitive to variation as their energy consumption differs from their Pareto frontier values (Table 2.1). Devices based on capacitance have lower variation on energy, since they tend to not dissipate power.

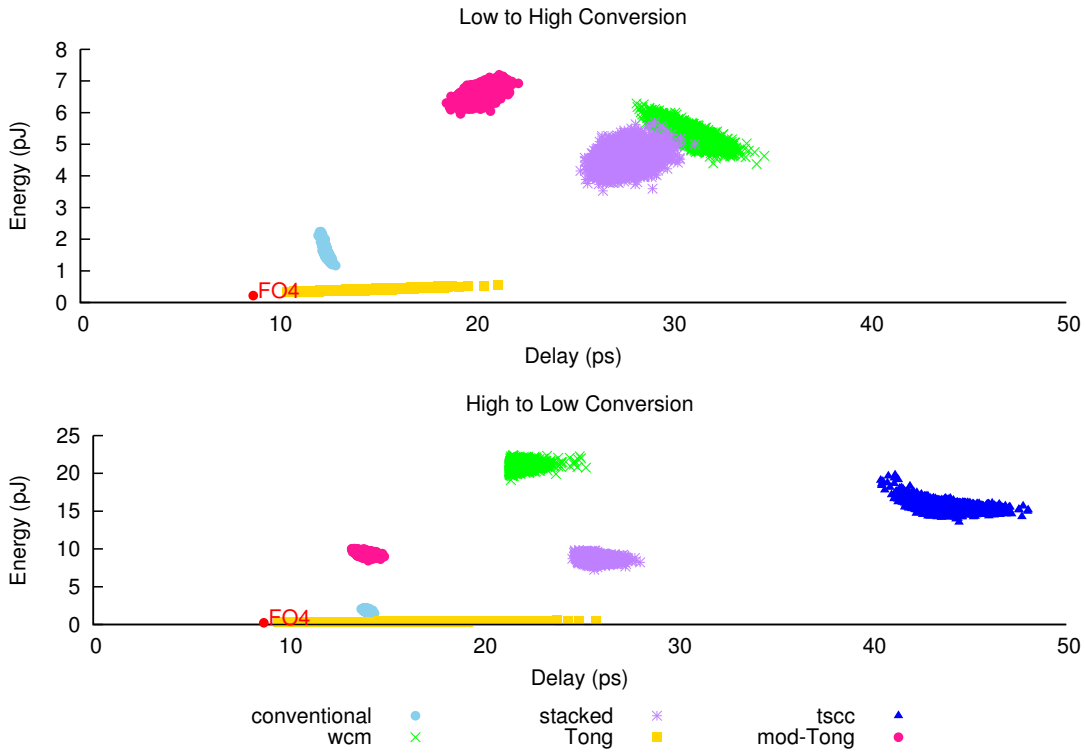


Figure 2.8: Active energy-delay: +/-5% V_{dd} variation MC simulations

Since Tong uses two capacitors, variation could affect the operational behavior of the LS. We repeat the experiment by applying variation to the capacitors. Similar to V_{dd}, the transient sweep is done by using +/-5% variation with 3σ. However, it hardly has any effect on

the delay and energy consumption of the LS (Figure 2.9).

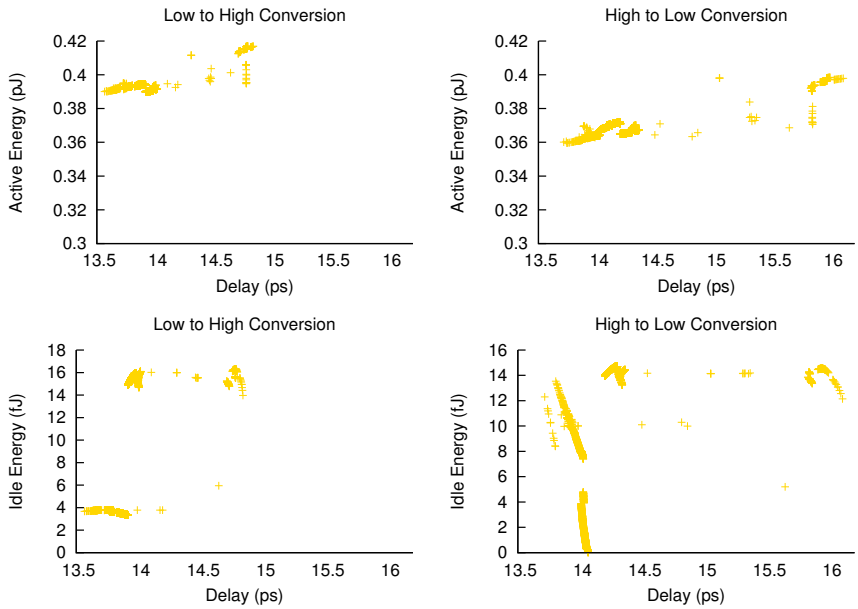


Figure 2.9: Active and idle energy-delay: +/-5% capacitance variation MC simulations

2.4 Conclusion

In this study, we discussed the trade-offs between different designs of level shifters existing in the literature, focusing on the applications in voltage stacked systems. The performance, power, and area of those designs greatly vary depending on the transistor sizing. To find the optimal sizing for each LS, we run experiments varying transistor widths and obtain a Pareto frontier of delay vs. energy. We choose the minimum energy-delay product point as the optimal point for each LS. In terms of power and delay, Tong [77] offers the best design points, but requires large MOM capacitors, that may make it unsuitable for applications where a considerable number of shifters may be required, therefore, we analyze various designs to

show how critical each design factor becomes in different contexts.

Chapter 3

Voltage Stacking in SRAMs

3.1 Introduction

Delivering power to the logic in a chip is one of the major challenges in current chip design [52]. As technology scales, the reduction in voltage supply has led to a sharp increase in the total current that needs to be delivered to a chip. Increased current has several drawbacks: reduced efficiency of voltage regulators [53], increase in voltage noise and losses due to parasitics [25,52], increase in the number of pins dedicated to power [87], and electromigration.

Voltage stacking is an alternative method to deliver power to components that are placed in series, rather than in parallel. The charge between the layers is recycled, *i.e.*, it passes through multiple components [66]. Thus, for the same power budget, voltage stacking allows for delivering less current at an increased voltage level. This effectively reduces the total chip current by n, where n is the number of stack levels used in the design [53]. Voltage stacking of CPU and GPU cores has been proposed by several groups to mitigate these problems [25, 52, 87, 88] and it will be further discussed in Chapter 4. In this study we focus on voltage stacking

in the context of SRAMs.

Previous work has focused on the voltage stacking of cores and logic components of a chip, however, chips dedicate a large portion of their area to SRAMs and cache blocks [2], and thus SRAMs account for a considerable amount of the chip power. Research groups have been able to reduce the current drawn from on-chip SRAMs by focusing on reducing their total power consumption [2, 28, 68]. Although reducing the power consumption is a goal by itself, it has limited impact when it comes to reducing the total current. Voltage stacking mitigates such problem by reducing the current drawn by a factor of n, where n is the number of voltage stacks.

The main challenge in voltage stacking is to balance the activity between the stack levels to maintain the voltage at each level roughly constant [52]. This is a challenge that we will address similarly in Chapter 4. An additional Voltage Regulator (VR) can be used in the intermediate node to guarantee that the voltage stays within specified values [53]. SRAMs have fairly predictable activity during operation, making them ideal candidates for voltage stacking. Voltage stacking SRAM banks has been proposed as a technique to reduce the stand-by power of those components [2]. In this approach, when the banks are not in use, power switches change the banks to a stacked configuration where only half V_{dd} is applies to each bank. This reduces the leakage power during the stand-by mode.

We propose applying voltage stacking to the SRAM structure. To guarantee that the activity is the same across the stack levels, the stacking is done by splitting each word and stacking the word parts which guarantees that the access to both stack levels occurs in the same cycle. Besides the RAM core, sense amplifiers and prechargers are also stacked. Level converters are used where needed to guarantee that each component receives the appropriate voltage level.

Our experiments show that SRAM stacking is able to reduce the current drawn by the SRAM by 40% during write, 36% during read and 44% during standby mode. We will explain that on a circuit level, stacking has multiple benefits such as reducing the IR drop and decreasing number of power rails which has a linear impact on the number of power delivery pins.

To the best of our knowledge, this is the first study that proposes voltage stacking in SRAMs at all times. It reduces the pressure of on-chip power grid design, by reducing the current, especially when combined with core voltage stacking.

3.2 Related Work

Voltage stacking has been proposed in the context of CPU cores to increase the efficiency of voltage regulators as it reduces the high current demands for current chips [52, 66]. Voltage stacking does not reduce the power consumption in the chip, but rather allows for operation at higher voltage and lower current, which is beneficial for VR design both by reducing power losses in the VR and reducing its area [52].

Voltage stacking has also been proposed to reduce the number of pins dedicated to power in a chip [87]. Since the number of power pins is roughly proportional to the current, reducing the current by a factor of n (in a n stacked configuration) would result in a reduction in the number of power pins by the same factor.

Gu *et al.* [25] note that voltage stacking can reduce voltage noise (IR and $L\frac{dt}{dt}$) and IR drop in the power grid, which can ultimately lead to the reduction of power in the parasitics of the system. We also observe that with the reduced noise it is possible to reduce voltage margins,

and increase power savings, although we do not evaluate this scenario in this manuscript, as it would require a parasitics characterization to obtain meaningful results.

Cabe *et al.* [2] proposed to dynamically stack SRAM blocks while they are inactive. This allows for a reduction in leakage power during the stand-by phase of the banks. The technique uses power switches that choose between the stacked mode when in stand-by or regular mode when in regular operation. However, this technique provides a constant V_{dd}, regardless of the circuit state (stacked or not), which means when stacked, half the V_{dd} is applied to each stack level, with doubled V_{dd}, thus the full V_{dd} is applied to each level at all times. We propose maintaining the circuit stacked during all phases of operation. This allows for a reduction in leakage during all times, as well as other benefits of voltage stacking, such as power pin reduction, VR efficiency increase, and voltage noise reduction.

3.3 SRAM Stacking Model

Voltage stacking helps reduce the current draw while applying a higher power-supply voltage to the logic blocks in the design. To take advantage of the charge recycling, instead of running the SRAM at V_{dd}, we divide it into two logic domains connected in series, and apply $2V_{dd}$. If each logic domain consumes the same amount of power, the voltage will distribute evenly between them. The logic domain loads however have to be selected in such way that they have well-balanced charge utilization for achieving a high efficiency [66]. If the power consumption of the two stacks is the same, as the voltage supply is multiplied by n, where n is the number of stacks, the current draw will be reduced to $\frac{1}{n}$. Two stacks are used and Equation 3.1 shows that theoretically this leads to 50% reduced current in the SRAM.

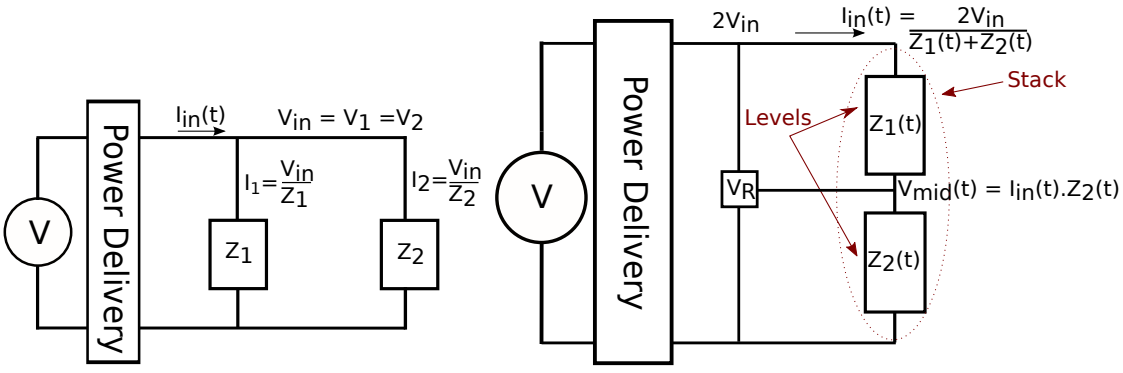
$$p1 = p2$$

$$V_1 \times I_1 = V_2 \times I_2$$

$$V_1 \times I_1 = 2 \times V_1 \times I_2$$

$$I_2 = \frac{I_1}{2} \tag{3.1}$$

In terms of circuit design, Figure 3.1 shows how the conventional model differs from the stacking model. In Figure 3.1a, the two circuits are running in parallel and the same voltage differential is applied to each, the total current drawn from the power source is then the summation of the current of all the components in the circuit. In Figure 3.1b, the stacks run in series and the configuration reduces the current draw, and the IR drop across the stacked components is reduced by a factor of n where n is the number of circuits in series. The current is recycled through the stacks connected in series. V_{mid} will fluctuate depending on the load and impedances present in each stack. If the impedances are similar, the V_{mid} balances in the middle and is the most efficient stacked configuration.



(a) Conventional **(b)** Stacked

Figure 3.1: Conventional vs. stacked power delivery mode

Figure 3.2 shows a high level view of stacked SRAM. The SRAM size is 1Kb, 32 32-bit words, and it has 2 read ports and 1 write port. In order to divide it into two vertical logic domains, we cut the 32-bit wordline in two, which leaves 16 bit for each stack. Bits 0-15 go to the bottom stack and bits 16-31 go to the top stack. Consequently, each stack will have 16x32 bitcells, 16 precharge circuits, 16 write drivers and 3x32 wordline buffers.

The two stacks operate at 0-V_{dd} and V_{dd}-2V_{dd} where V_{dd} is 1.1V. The voltage level of all the signals entering the top stack (2.2V) will need to be shifted; hence, level shifters are placed in 3 locations. First, the input data to write drivers has to be converted to the same voltage level as the top stack. The wordlines will pass both stacks in a row of bitcells, therefore, each decoder will drive two wordlines. Second, the top stack wordline voltage level is shifted. To avoid having the wordline be driven by the level shifters, we placed them before the read and write address buffers. Third, the sense amplifier outputs exiting the top stack will have to be converted back to the V_{dd}; hence, level shifters are placed right after the sense amplifiers. The read and write address decoders, placed in the middle of the RAM core, are the only components which are not part of the stacked architecture.

The schematic of the level shifter used throughout the SRAM is shown in Figure 3.2. There are two capacitors C1 and C2 ,which are sized 15fF each. The design is adopted from Lee's 16-core design [53].

3.4 Setup

We implemented our voltage stacking technique on SRAMs generated by FabMem. FabMem is a multiported RAM and CAM compiler for design space exploration and given

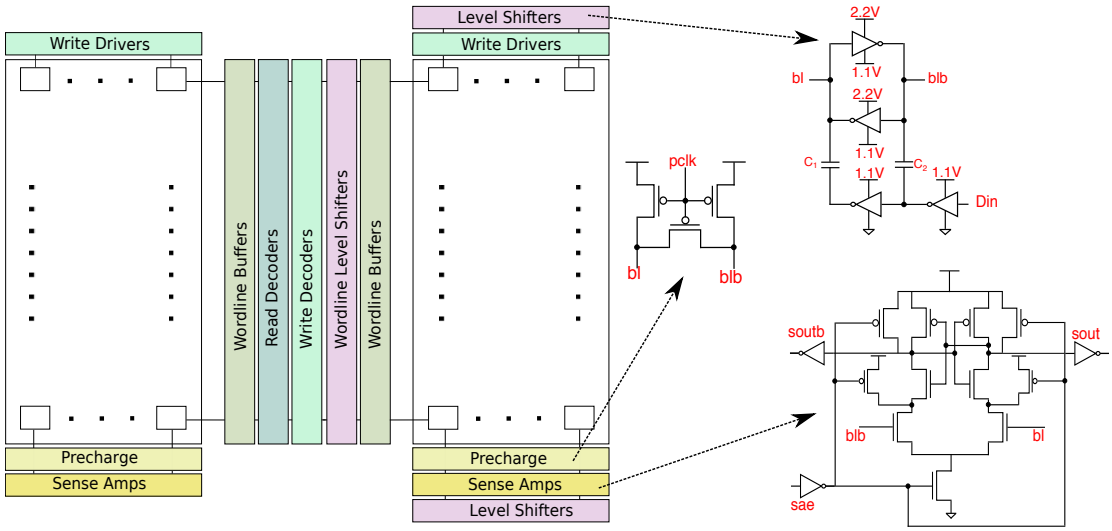


Figure 3.2: Voltage stacked SRAM

the configuration it can generate netlists and layouts and estimate read/write delay and energy consumption [71]. FabMem uses the NCSU FreePDK, the Open-Access-based PDK for the 45nm technology node [73].

Using FabMem, we generated an SRAM with a configuration that is similar to the size of a typical Register File: A 2-read 1-write 1Kb consisting of 32 32-bit words. We use one RF for the base case SRAM and another for the stacked version.

A few alterations had to be made to some of the SRAM components such as the sense amplifier and the precharge circuits. In our experiments, we use current latch mode sense amplifier [60] as opposed to the FabMem default voltage controlled sense amplifier circuitry, because the current latch mode works with the stacked SRAM design. The precharge circuitry used is shown in Figure 3.2.

All the energy related measurements were taken using Synopsys HSpice version I-2013.12-1.

3.5 Evaluation

In this section, we discuss the overall results and share insights as how effective SRAM voltage stacking is.

The two SRAM netlists are simulated using HSpice at $V_{dd} = 1.1V$ with frequency of 500MHz. The simulation results compare the stacked SRAM against the non-stacked SRAM, which we refer to as the base case in this section. Figure 3.3 shows the energy consumption breakdown. Each colored section of the histogram bar shows the total energy consumption of a particular component. The energy numbers pertain to the logic that is stacked. For instance, the level shifters are placed before the buffers driving the wordlines, therefore, these buffers are part of the stacked logic and the read and write address decoders are not. We refer to the read and write wordline buffers as "decoder".

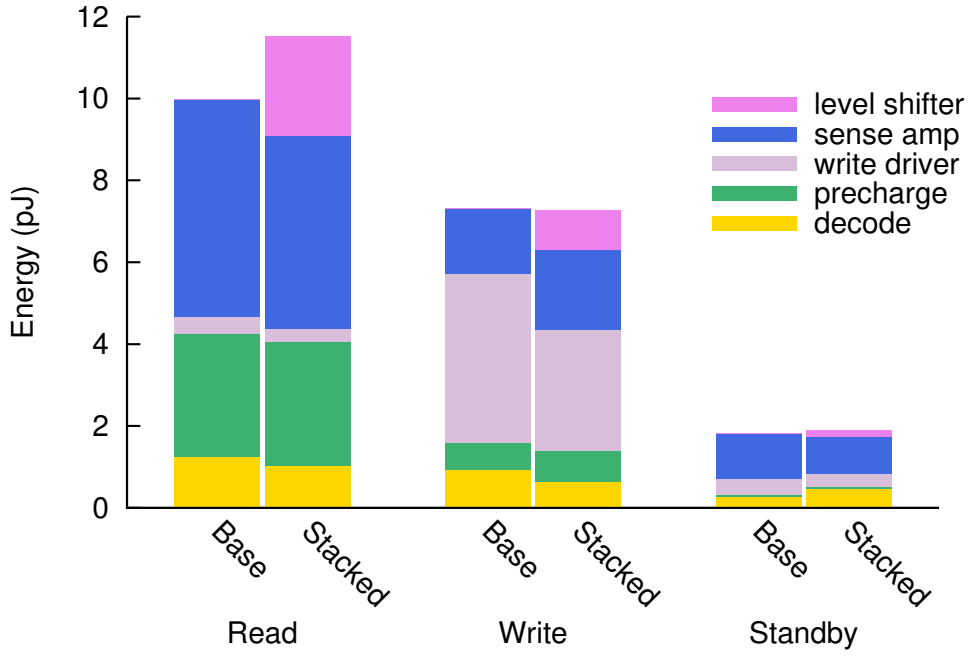


Figure 3.3: Energy breakdown in voltage stacked SRAM vs. non-stacked.

The simulation includes periods of initialization, write, read, followed by standby. Each write or read period consists of 10 write or read operations. Overall, the power consumption increases by 20%, 23%, and 13% during write, read, and standby modes respectively. During the read operations, the stacked SRAM consumes a bit more energy than the base case, however, looking at the breakdown, the excess energy usage is due to having the level shifters in the top stack. For instance they draw 8% more current during the read operation than the precharge circuits. The other components such as sense amplifier, precharger, decoder, and write drivers almost consume the same amount of energy. As the SRAM size increases, the overhead of level shifters will become less of an issue. The standby energy consumption is the same meaning at $2V_{dd}$ the SRAM has the same leakage as operating at V_{dd}. Figure 3.4 shows the current consumption in the top stack of the stacked SRAM versus the base SRAM. Overall, stacked components draw 40%, 36% and 44% less current than the base components during the write, read, and standby modes. Comparing the current and energy breakdown plots for the write operation, we notice that that the write drivers do not have a balanced load, leading to less than expected current savings. For larger SRAMs however, we would expect that this gap will decrease, since the relative power of write drivers are smaller.

We also evaluate the impact of stacking on the SRAM performance. For bitline transitions shown in Figure 3.5, we simulated both SRAMs with 1GHz frequency. Two read output bits, are selected from the top and bottom stacks to show how level conversion affects the SRAM read speed in each stack. As Figure 3.5b shows, there is no delay for the bottom stack bit, however, Figure 3.5a shows the stacked bit has some delay when transitioning from 0 to 1. It is a $60ps$ delay which translates to 6% frequency hit. In reality, the performance hit should be even smaller, because the stacking can be applied to the slower part of the SRAM. We did not

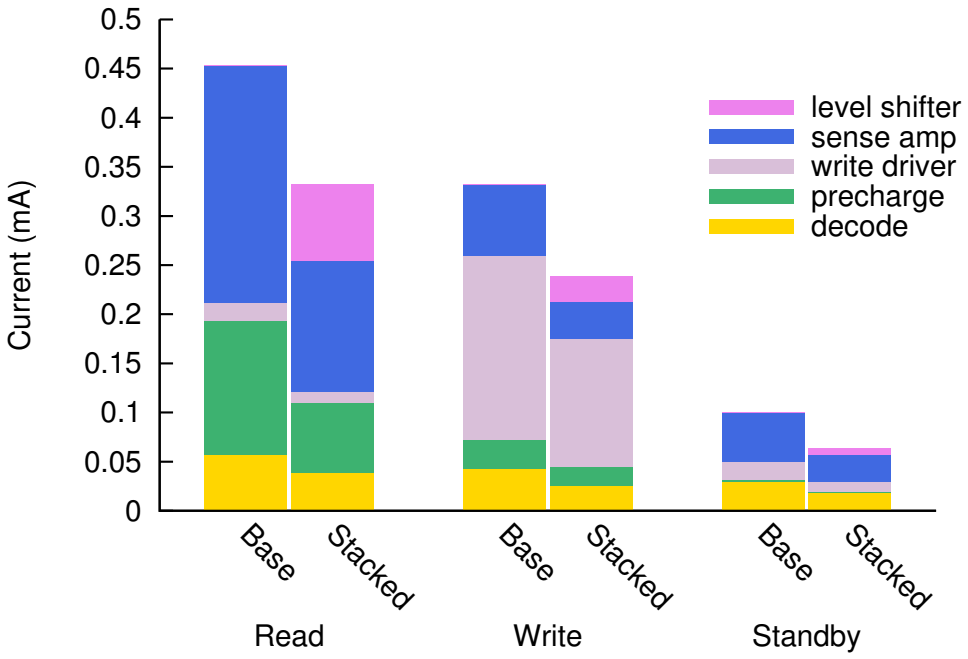
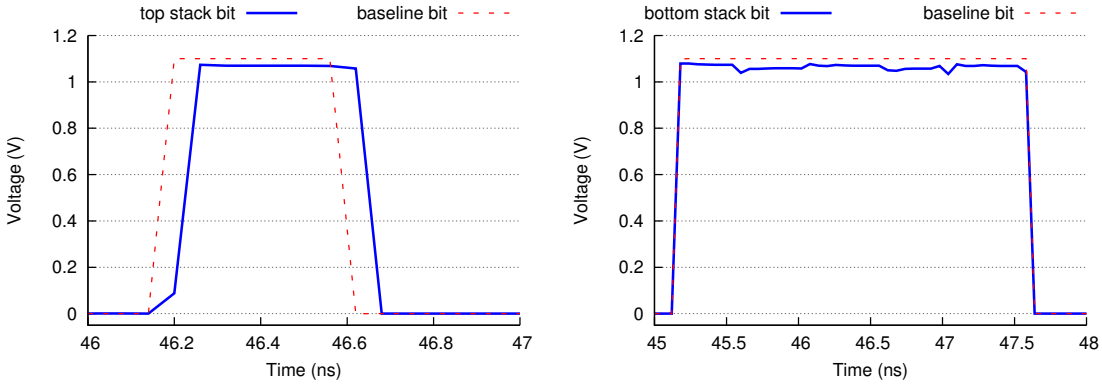


Figure 3.4: Current breakdown in voltage stacked SRAM vs. non-stacked.

perform this optimization, because it is layout dependent. Thus, SRAM stacking has limited impact on performance.

Finally, we evaluate the voltage noise level in the V_{mid} due to the fluctuation in the load. One of the challenges in voltage stacking is load balancing. As expected in a stacked architecture V_{mid} fluctuates and introduces noise to the power deliver network. Since we have used an SRAM architecture, the nature of the circuitry allows for a division of stacks in such way that the V_{mid} is maintained within an acceptable range. Figure 3.6 shows the stability of V_{mid} during initialization, write, and read operations. In the case of V_{mid} noise, one option is to place a voltage regulator at V_{mid} to guarantee that V_{mid} stays at V_{dd} which is 1.1V. In our designed SRAM, the fluctuations observed are within the acceptable range and thus there is no need for an extra voltage regulator, as it is used in stacking for cores [52].



(a) Top stack read delay

(b) Bottom stack read delay

Figure 3.5: Stacking introduces a 60ps delay in the top stack output bitline, which is equivalent to 6% performance hit.

Voltage stacked SRAMs have a small area impact; the level shifter is the main source of overhead. Even in a small SRAM, such as the one we evaluate, the level shifters represent less than 6% of the transistors.

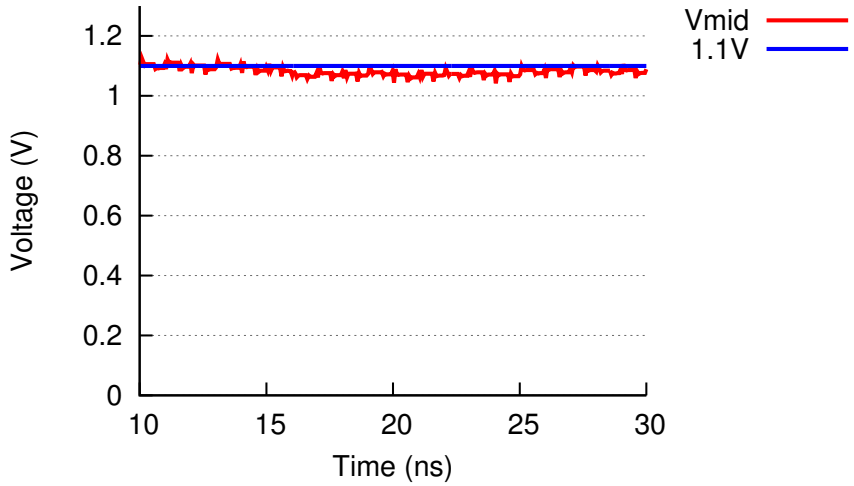


Figure 3.6: Voltage noise for stacked SRAM is within acceptable levels, even without a voltage regulator.

3.6 Conclusion

As the technology scales, and power supply is reduced, delivering power to the logic on a chip becomes a major challenge as its current demand increases drastically. Increased current has many drawbacks [25,52,53,87]. Voltage stacking is an alternative method to deliver power to components on a chip that is stacked or placed in series. A large portion of the chip area belongs to SRAMs and caches. In this research study, we focus on SRAMs and stack an SRAM the size of a typical Register File. Our design keeps the SRAM in stacked mode at all times. Other research work have been proposed where they keep the SRAM banks in stacked mode during the standby mode [2]. We divide the SRAM words into 2 stacks and are able to reduce current by 36%-44% while not letting energy consumption be more than 23%. In addition, doubling V_{dd} and reducing the current could save as much as 10% power from the increased VR efficiency [52], but this was not considered in this paper.

Chapter 4

GPU NTC Process Variation Compensation with Voltage Stacking

4.1 Introduction

Near Threshold Computing (NTC) can improve energy efficiency by further reducing the operating voltage [19]. It has been shown that the performance impact resulting from NTC can be mitigated through parallelism. An ideal candidate for such operation is a GPU [9, 19]. Nonetheless, NTC complicates power delivery. It requires more current to flow at a lower voltage [52] and the increased current flow reduces the efficiency of the power delivery circuitry. It also makes the system more sensitive to process variation [39].

To manage the additional sensitivity to process variation (PV) introduced by NTC, some researchers have proposed frequency scaling or having multiple voltage domains. Having multiple voltage domains requires additional power rails which will further exacerbate the current delivery problem. Frequency scaling [40] addresses some of the issues regarding our

proposal model which we will address in detail in Section 4.3.

In Chapter 3, we discussed why voltage stacking is a beneficial alternative power delivery mechanism and how on a circuit level it reduces the current drawn from the main Voltage Regulator (VR) that makes it more efficient taking less area. Not only has it been applied to SRAMs, but also it has been proposed in the context of CPU cores [52], and GPU cores [88] .

Voltage stacking can be analyzed from another interesting angle that it can compensate for the process variation effects. The proposed GPU stacking methodology lets the voltage node between the stacked elements (V_{MID}) float[1]. This floating node is the key to process variation compensation. GPU stacking alleviates the current delivery challenges, and intrinsically mitigates process variation effects without requiring multiple voltage domains. GPU stacking automatically creates a voltage domain per level in the stack without the cost of multiple power rails. We build on top of this premise, and discuss how it can be leveraged for managing process variation. To the best of our knowledge, this is the first time that voltage stacking has been utilized in NTC GPUs.

Voltage stacking of many cores has its own challenges, among which is the load mismatch between the stacked cores [7]. Cores go through different phases while running applications, which can result in transient impedance mismatch of the stacked cores, and yield timing failures. As a result, stacking is successful when the cores have a matching workload. GPGPUs are instances of such designs. Not only are the cores identical, but the applications running on them are roughly homogeneous[2].

[1]For safeguarding reasons, a few voltage regulators are used to cap the maximum and minimum levels of voltage but the voltage does float between maximum and minimum values.

[2]Even though divergence exists in modern GPUs, the amount of load mismatch observed in traditional GPGPUs benchmarks can be handled by our technique, as it will be shown in our evaluation.

The evaluation of GPU stacking is carried out in the near threshold region. Although the use of this method in the near threshold region is proposed, it is not a requirement. The use of NTC in this research study is justified by the increased sensibility of NTC to process variation effects. The first part of this experiment consists of finding the ideal GPU configuration for the NTC region. By carefully sizing the GPU to NTC, power consumption is reduced by 43% with only 4.8% performance degradation compared to the baseline.

Based on our experiments, there is a potential for self balancing in the stacked configuration. We observe that stacking of cores with opposing process variation trends is a better choice to gain the best self balancing results. This is more likely to happen when there is a certain distance between the stacks. This study proposes the stacking of SIMD Lanes, because having a large number of lanes provides more opportunities for PV compensation. Analyzing several PV maps, GPU stacking shows that it is able to deliver, on average, $\approx 80\%$ of the nominal performance in a multi-frequency domain scenario (as opposed to $\approx 60\%$ in a non-stacked configuration).

The floating V_{mid} node could be a source of problems in cases of extreme load mismatch, or high temperatures. Thus, we propose the use of Dummy Activity (DA), and a small voltage regulator to keep the voltage within safe operational margins (*i.e.*, to avoid voltage starvation in one of the levels). We use SPICE simulations to verify the reliability of the power delivery network (PDN), and show that GPU stacking does not incur extra voltage noise. In our simulations, DA and the additional VR were not required, given the stability of the operational voltage.

The following lists the contributions of this research study:

- The first study to show how voltage stacking alleviates process variation effects.

- The first study proposing stacking with an uneven voltage division (V_{MID}).
- A study to propose a technique to make the post-silicon configuration of the design feasible.

The remainder of this chapter is organized as follows: We start by presenting the related work in Section 4.2, then in Section 4.3, we investigate, with an example case, how Voltage Stacking can mitigate the effects of process variation, and discuss the application of this technique in a GPU architecture. In Section 4.5, the trade-offs of designing a GPU at near threshold are discussed, and the baseline architecture is defined. Section 4.4 and 4.6 present the evaluation setup and the results for the conducted experiments. And finally, Section 4.7 concludes the findings of this study.

4.2 Related Work

GPU Stacking: As supply voltage decreases, the efficiency of power delivery components degrades. On-chip voltage regulators [44] have been proposed to increase the PDN efficiency, as well as the series configurations of units rather than parallel [25, 52, 66]. Such configurations are known as Multi-Story Power Delivery [25], charge recycling [66], or mainly as voltage stacked systems [52]. Our proposed technique depends on a series configuration of cores, yet for a different purpose. Such configuration is referred to as GPU stacking throughout this dissertation. Note that no previous research on voltage stacking exploits the stacking method to control or neutralize process variation effect.

Process Variation: Process variation increases as feature size shrinks. And lowering V_{dd}, as a power management technique, further exacerbates the effects of process variation.

Karpuzcu *et al.* [39] built a model on top of the VARIUS [70] tool which models process variation for logic and memory cells, while targeting NTC [9, 19], called VARIUS-NTV. They observed significant variation in core frequency, and power consumption as a result of process variation.

Lee *et al.* study the impact of frequency variation on the throughput of a GPGPU [50]. Adaptive Body Bias (ABB) leverages the power-performance trade-off to manage process variation effects [84]. Slower devices due to process variation, can run faster by consuming more power, and vice versa. Adaptive Supply Voltage (ASV) is another technique in managing process variation [12]. Supply voltage of a region in the design can be adjusted to compensate for performance loss due to variation. ABB and GPU stacking are orthogonal techniques, and could be used together, although ABB efficiency is expected to reduce with technology scaling [40]. Our proposed technique intrinsically performs ASV.

NTC: An extensive amount of research targets increasing the power/energy efficiency of processors. Apart from the many proposed techniques of how to reduce the utilization of resources [18, 59], or how to promote the use of more power efficient structures [13, 15], there are a significant number of proposals that attempt to utilize the power-performance trade-off. DVFS and Power Gating are among the techniques widely studied at an architectural level, for the same purpose [29, 30]. Intel Turbo Boost technology is another example of utilizing voltage and frequency scaling to adapt to runtime conditions.

Dreslinski *et al.* [19] study devices for near threshold operation, and Chang *et al.* propose the optimization of device parameters for NTC [9]. They propose a slightly modified SRAM cell to address the stability challenges introduced in near threshold regions. Lower V_{dd} exacerbates the effects of process variation. In the context of NTC, again we refer to VARIUS-

NTV model built by Karpuzcu *et al.* [39] which models process variation. The same team published another study of NTC in many cores where they argued in favor of fine-grained core assignment and DVFS [40].

Energy efficiency in GPGPUs: With the rising popularity of GPGPUs, several research groups discuss strategies to make them more energy efficient [24, 51, 69, 89]. Lee *et al.* study the impact of frequency variation on the throughput of a GPGPU [50]. This methodology, however, is the first to extend the evaluation to NTC trade-offs and it is an extension of previous research addressing NTC challenges. Massive data parallelism and extremely repetitive nature of the GPGPU applications is leveraged to adapt the operational region and configuration to the runtime application demand.

4.3 GPU Stacking

Revisiting Figure 3.1, if the GPU core units in the stack consume the same amount of power, the voltage across each GPU core will be equal to V_{dd}. Because as less current flows through the system, the power delivery subsystem could operate more efficiently. However, in the context of GPUs, balancing the overall load across the stacked elements, is not as easily manageable as SRAMs. This is because a change in behavior of one core directly affects the other core. If these changes have significant magnitude, they could result in timing failures as one core could throttle the power delivery to the other. Nonetheless, stacking exhibits promising behavior in the presence of process variation that can be leveraged towards more variation resilient designs.

Before presenting the final model, we formulate the high level effects of process vari-

ation expected in a circuit and explain the high level mechanisms by which voltage stacking can compensate for the effects of process variation. Then, an example case is presented that quantifies how much PV is compensated under simplistic and idealized scenarios. We conclude our preliminary findings and apply them to the GPGPU architecture to mitigate the PV effects. The section ends with a discussion of the architectural choices made for managing extreme cases.

4.3.1 Process Variation

VARIUS-NTV uses Equation 4.1 formulates the gate delay for a MOS transistor, where V_{dd} is the supply voltage to the core, L_{eff} is the effective channel length, K and M are fabrication constants [39].

$$delay \propto \frac{V_{dd} \times L_{eff}}{K \times ln^2(e^{M \times (V_{dd} - V_{th})} + 1)} \tag{4.1}$$

Due to the exponential component with the $(V_{dd} - V_{th})$ term, the gate delay exhibits more sensitivity to V_{th} variation at supply voltages close to V_{th}. Dreslinski *et al.* [19] point out that this effect can be managed effectively by Adaptive Body Biasing (ABB), but it is a consensus that ABB will have limited effect on new technologies [40], which are more susceptible to the process variation effects we aim to mitigate.

Since the distribution of variation could exhibit discrete effects based on the granularity of functional blocks or cores, for the purpose of trend evaluation, we use a guardband similar to Chang *et al.* [9] to take the average penalty of variation and fault into consideration.

4.3.2 High Level Idea of Process Variation Compensation

GPU Stacking provides a unique opportunity to manage the effects of process variation. An increased channel length (L_{eff}) or an increased threshold voltage (V_{th}) due to process variation will result in higher impedance of the channel and slower device.

In a conventional parallel power delivery system, the adverse process variation results in a lowered current Iin_i through the core$_i$, as $Iin_i = V_{dd}/R_i$. Since gate delay is inversely proportional to the Iin_i, it will result in a higher delay and a slower core. To compensate for the lower Iin_i, higher voltage can be applied to the core, or the body bias could be adjusted to reduce the gate delay. In short, the adverse effect of process variation can be compensated by delivering higher voltage.

In a stacked configuration, the same current passes through the stacks. Therefore, higher impedance of a core, due to adverse process variation, results in a higher voltage across that core ($V = IZ$).

Equation4.2 shows the voltage across each core. Index i is the core or *lane* number. Depending on the switching activity of the circuit, the equivalent supply to ground impedance of a core changes during execution. This will be referred to as $Z(t)$. Process variation will bias Z according to the magnitude of the variation. Switching activity (or the running application) will change the transient aspect of Z. Using a performance and power simulator, we can obtain the core power traces and when physical dimensions of the PDN are given, using a circuit simulator, we can analyze the impedance change over time [55]. In our experiments, since the power consumption changes over time, the impedance of the circuit has also varied over time (A more detailed explanation in Sections 4.4 and 4.6.

$$V_i(t) = V_{in}(t) \times \frac{Z_i(t)}{\sum Z(t)} \tag{4.2}$$

Utilizing the inherent feature of stacking is a key contribution of this work. The core with higher impedance due to adverse process variation will have a higher voltage drop across its power terminals. This results in a core speed up, relative to its speed without the higher voltage and with respect to a conventional power supply system, as delay is inversely proportional to the voltage. This, of course, comes at the cost of a lower speed for the other core in the stack. So stacking enables the slower core to run faster, relative to its speed in a conventional configuration, at the cost of the faster core running slower. In other words, the effects of process variation are intrinsically balanced in a stacked configuration.

Ideally, the variation effects in a stacked configuration converge to an average variation. With a simplification[3], one can expect the frequency of the stacked cores to converge to the average of the two cores in a conventional configuration.

For example, a core with 10% variation compared to the nominal value runs at $0.9f$ in a conventional configuration, and a core with -10% variation runs at $1.1f$. Stacking these cores would result in both cores exhibiting delay properties similar to the nominal values and run at about the nominal frequency ($\frac{+1.1f+0.9f}{2} = f$). After all, the nominal value is nothing more than the mean properties across all the samples. However, experiments in Section 4.3.3 show that while the latencies of devices in the stack merge toward each other, they do not exactly converge to the nominal latency.

[3] L_{eff} and V_{dd} both have linear effect on the delay, however, the impact of V_{th} on delay is not linear.

4.3.3 Detailed Analysis of Process Variation Compensation with Voltage Stacking

Section 4.3.2 explained the hypothesis that GPU stacking can facilitate a pathway for process variation compensation. However, the simplistic model does not consider the details of such configuration. SPICE simulations (at 45nm technology [90]) are performed for an example case where inverters are configured in conventional and series with two stack levels. Each inverter drives a high load of 16 other inverters, 4× Fan out of 4, to exacerbate the delay effects so that we can better observe the output signal changes due to process variation. Figure 4.1 is the stacked inverters, and the blue inverter is the device under test. The stacked configuration is supplied with $1.2V$. Using SPICE simulations, the timing characteristics of the inverters with and without presence of process variation are analyzed. For the stacked configuration, the variation is set to affect the header inverter positively (*i.e.*, shorter L_{eff}, thus faster), and the footer negatively (*i.e.*, longer L_{eff}, thus slower).

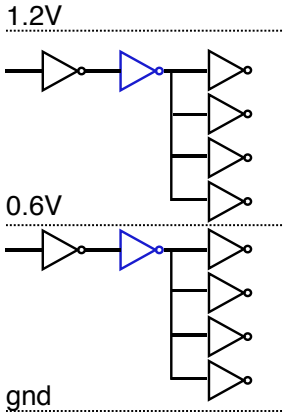


Figure 4.1: Example of how the case study inverters are used in the stacked logic. The blue inverters are the devices under test.

In this example, there is a 20% process variation in effective channel length (L_{eff}). If

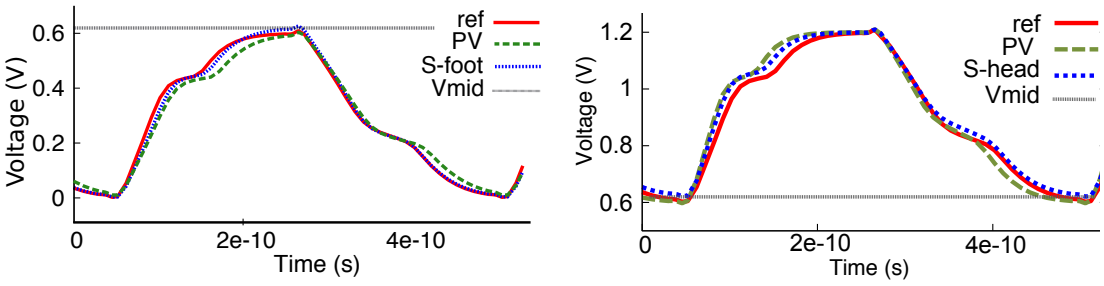


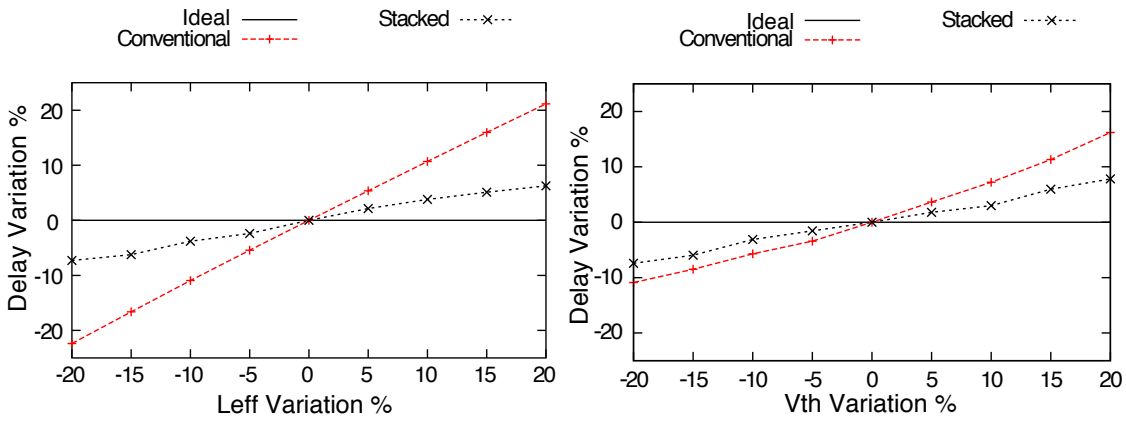
Figure 4.2: Stacked configuration ($S-foot$ and $S-head$) intrinsically mitigates the variation effects. These signals are closer to the case without process variation ref, than the non-stacked baseline PV. Since V_{mid} has shifted, there is more voltage available for the slower part of the design.

the channel length increases, the response time will be slower than the nominal non-variation case (lines PV and ref at the bottom of Figure 4.2). And if the channel shortens, the opposite effect will be observed (PV will be faster than ref at the top of Figure 4.2). However, this also implies different power consumption for each, as seen above.

The simulation results show that with voltage stacking, the voltage rail between the two levels of the stack (V_{mid-pv}) settles around 0.63V. This is because the header transistors have less resistance due to the shorter L_{eff}. The header transistors are effectively supplied with 1.2V - 0.63V = 0.57V, and the footer transistors with 0.63V. In this case, there is a mitigation of the delay variation, shown by $S-head$ and $S-foot$, which are closer to the scenario without process variation. This is a reduction of more than half the delay variation introduced by process variation. Next, we evaluate the effects of V_{th} variation on the inverter delays. The same scenarios are simulated, except that this time the variation is on V_{th}. The transient response for this case is similar to that of Figure 4.2, and thus the graph is not included.

Figure 4.3 summarizes all the experiments for different variation values from -20%

to +20%. We evaluate both L_{eff} and V_{th}. Note how the delay variation is smaller with the use of stacking. Only one line is presented for the stacked configuration, since the stack position (header or footer) does not change the result.



(a) L_{eff} **(b)** V_{th}

Figure 4.3: Mitigation of the variation effects compared to conventional configuration.

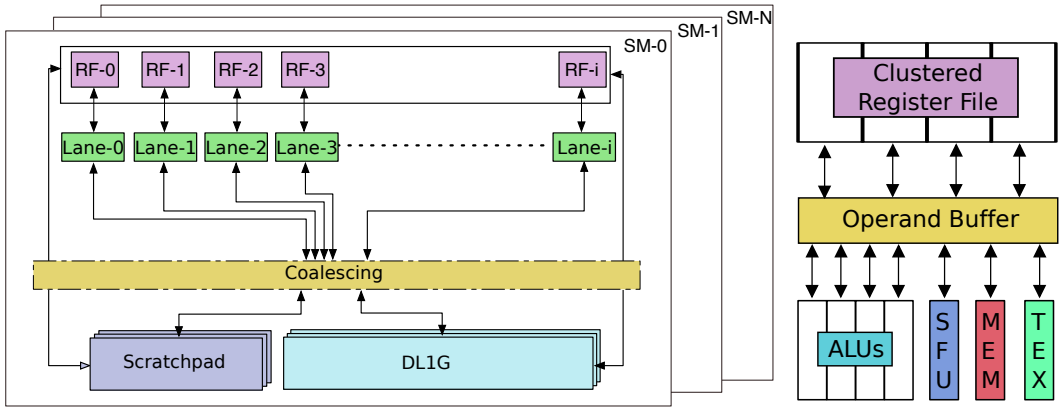
The SPICE model confirms the premise of intrinsic mitigation of process variation in stacked configuration. In the remainder of this chapter, "variation" is used to name the combination of the effects of different sources of process variation, and its total effect on the frequency is measured, unless otherwise specified.

4.3.4 General Purpose GPU

So far we have assumed that the stacked devices are identical and are utilized in a similar way. However, in reality even identical cores can run different applications or different phases of the same application. As explained before, this complicates the operation in the stacked configuration.

General purpose computing on graphic processing units (GPGPU) is becoming per-

vasive as it provides excellent computing power for massively parallel applications. GPGPUs are mainly designed as a cluster of simple processors, depicted in Figure 4.4. Identical simple processors (*lane*[4]) operate in lock-step inside a stream multiprocessor cluster (SM), running identical threads (though processing different data). The homogeneous structure of GPGPUs, both in hardware and application, makes them suitable for voltage stacking in order to manage process variation. GPU stacking stacks *lane*s inside SMs. All the other structures remain in a conventional configuration. The choice of stacking *lane* provides room for more configurability, due to the larger number of *lane*s. It also provides a fine-grained mitigation of process variation, while at the SM level, techniques like multi-clock and multi-voltage domain are possible. Thus, GPU stacking needs an extra voltage domain. In the following paragraphs, we use the terms core, SIMD lane, and *lane* interchangeably.



(a) GPGPU **(b)** SIMD Lanes

Figure 4.4: GPGPUs present a large number of identical SIMD lanes, ideal for stacking.

[4]Some authors use the term *lane* to refer to each of the small execution cores within what we call a SIMD lane. We use the same definition as Gebhart *et al.* [23], and thus count one lane per load/store unit.

4.3.5 Process Variation Model

Variation is divided in two categories: systematic and random. They differ on the granularity at which they occur. Random variation occurs at the transistor level. Systematic variation occurs at a much coarser granularity: within-die (WID) and die-to-die (D2D). Significant variation can be seen on the order of half the chip length [39]. This also justifies the use of *lane* as stacking elements. As seen in Section 4.3.3, the base of the compensation effect comes from a deviation in V_{mid} from $V_{dd}/2$. For high variation levels, V_{mid} could approach the failure voltage, especially in NTC, where the margin is smaller. GPU stacking addresses WID variation, since V_{mid} will vary depending on local average of variation.

4.3.6 Which Lanes to Stack?

Stacking neighboring *lane*s can mitigate the process variation effects regionally. For example, consider the die shown in Figure 4.5. The figure shows four *lane*s with differing amounts of variation compared to the nominal properties. Stacking the *lane*s based on their adjacency (*i.e.*, *lane* 1 and 2 as one stack, and *lane* 3 and 4 as another) will help mitigate the worst case variation (*i.e.*, 20%). The 1-2 stack would operate at about -15% of the nominal frequency ($(1.1f + 1.2f)/2 = 1.15f$, in practice the attenuation would be less). The 3-4 stack would operate at about 15% of the nominal frequency ($(0.9f + 0.8f)/2 = 0.85f$). The actual frequency must be that of the slowest stack.

For the best results, however, the stacking configuration has to be determined based on the observed variation, *i.e.*, to stack a *lane* adversely affected by variation with a *lane* positively affected. For example, stacking *lane* 1 and *lane* 4 together, and *lane* 2 and *lane* 3 as another stack would result in each running at about the nominal frequency. The best strategy is to cluster,

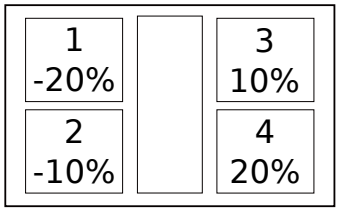


Figure 4.5: Sample die with 4 *lane*s and different variations.

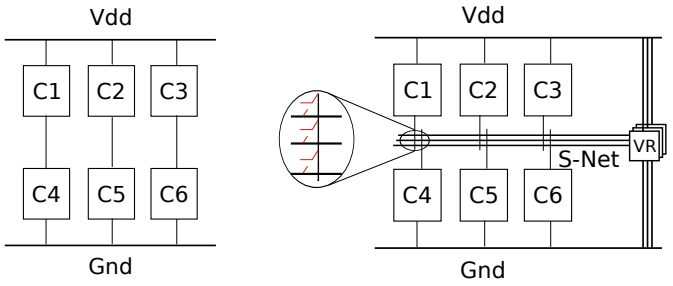
in each side of the stack, *lane*s minimizing the standard deviation of variation (the rationale is that *lane*s with similar process variation require approximately the same compensation). Then, the cluster with maximum process variation average in the header should be stacked with the one with minimum negative variation in the footer. However, this might not be trivial to find, since the number of possible combinations is large. A simpler approach, used in this study, is to have the same number (for instance N) of *lane*s in all the clusters. The clustering is made by simply picking the N *lane*s with maximum variation in the header and clustering them with the N *lane*s with negative variation in the footer. The process is repeated for the remaining *lane*s. This simpler approach works fine in the level of lanes, since their spatial proximity causes similar variations.

Since the variation is not known until after fabrication, a configurable fabric is needed to group and stack the *lane*s based on the observed variation. We adapt the idea of a configurable power delivery fabric [36, 66] to allow the connection of *lane*s that are not neighbors, because in terms of variation management, neighboring *lane*s are most likely, affected similarly. Our proposed fabric capitalizes on the fact that GPU stacking is actually better suited for the connections of logic with opposite variation effects. Note that this may cause a problem, given that it increases the path from V_{dd} to Gnd. Thus, there is a design trade-off here: on one hand, the compensation would be better if the stacked logic were farther apart in the chip, on the other

hand, if this distance is larger, the voltage droop due to voltage rails and switches is increased. That is another reason why stacking lanes is a good design choice as opposed to stacking SMs. Lanes are well constricted in space, within the SM, while SMs will be farther apart in the chip, still an SM is large enough to be used for the purposes of GPU stacking, *i.e.*, there is enough variation in the SM to allow for the type of compensation we aim, as observed in our evaluation.

4.3.6.1 Shared Net Based GPU Stacking

To simplify the design, we propose to cluster the stacking of *lane*s. The clustering is a trade-off between cost and complexity. To cluster *lane*s for stacking, we define Shared Net. Shared Net is a common net that connects a number of *lane*s. For example, V_{dd} is conceptually a shared net. However, we specifically use the term Shared Net for an intermediate net that connects a number of *lane*s in a stacked configuration (V_{mid}). The number of Shared Nets is a design parameter and changes the trade-off between area and compensation granularity (more Shared Nets result in a more fine-grained compensation).



(a) Stacked **(b)** 3 Shared Nets

Figure 4.6: Shared Net configuration simplifies stacking and supports post-fabrication configurability.

Figure 4.6a shows a two-level stacked configuration of six cores with no shared nets. The stacking configuration is static, and is determined at the design time. As the variation is

not known at the design time, and there is a spatial correlation in the variability, this scheme is not likely to provide the compensation opportunities. Figure 4.6b shows the design with three Shared Nets. To simplify the proposed design, each *lane* is fixed to either header or footer at design time. This choice certainly reduces the freedom of the system, but because there are several *lane*s per SM, the loss is minimal. The resistance in Figure 4.6b Shared Nets is not modeled in this study. The scheme is similar to a multi-power domain case, where multiple voltage rails are present and each part of the circuit connects to a different rail. The configurability only applies to the intermediate nets.

The number of Shared Nets is determined at design time based on the expected variation of the technology, or the severity of variation effects on the design metrics and to the level they need to be managed. This decision has to be based on the variation profile for the fabrication technology. Given the knowledge on the variation, it is possible to calculate the expected power consumption for a block (in relative terms), *e.g.*, using Varius-NTV models [39], the power information can then be used to calculate the expected V_{mid} voltage for a given stacking configuration, which can finally be used to calculate the expected performance. The decision on the number of Shared Nets is then a trade-off between the cost of adding an extra Shared Net and the extra performance boost gained. This is explored in our evaluation.

During post-silicon testing, each *lane* can be tested to characterize the observed variation for that particular *lane*. This increases binning time, but current chips already undergo this type of testing for speedgrade purposes. Once the effect of variation is known for each die, the *lane*s can join a cluster based on their observed variation. Note that the clustering is static and done once in the lifetime of the chip in a calibration step, right after fabrication. To allow for the post-silicon configurability, an array of power transistors or fuses can be used between

each *lane* and each Shared Net. Power transistors are present in modern designs for power gating purposes. Although we did not consider them in this design, the reduced current passing through them (due to voltage stacking) will largely reduce their impact on the circuit.

When multiple Shared Nets are present, the clustering is done by stacking the same number of *lane*s (namely *n*) on each side (foot or head) of the Shared Net. The *lane*s are sorted by variation (minimum delay), for both head and foot groups. The *n* first *lane*s in the head group are stacked with the *n* last *lane*s in the foot group. The next *n lane*s in each group are stacked together, and so forth. This configuration will have the maximum compensation within each SM, as *lane*s with opposed variation trends tend to be in opposite sides of the same stack, replicating the behavior observed in Section 4.3.2.

4.3.7 Divergence and Extreme Conditions

The process variation compensation comes from the fact that V_{mid} is "floating", *i.e.*, is not at a fixed frequency. Footer and header groups have different voltages instead. Nevertheless, it is possible that due to load unbalance (caused by GPU divergence), or *e.g.*, , extreme temperature conditions, the voltage difference is such that either level has not enough voltage to guarantee functional correctness. We call this "voltage starvation". We propose different mechanisms to handle such scenarios.

Dummy-activity is inserted through the activation of parts of the lane that are not being used. For instance, if divergence is observed for long periods of time, this could shift the voltage towards the most active *lane*s. This shift can be canceled by adding activity in inactive *lane*s, or inactive parts of *lane*s. When Dummy-activity is inserted, the *lane* does not commit any change to the architectural state, nor does it execute stores, for obvious reasons.

"Lane turn-off" is a more drastic measure for extreme cases. In this case, there is no scheduling for one of the lanes in the level consuming more power than expected. This can only be done in architectures where each *lane* within the SM can execute different code, and would require awareness in the scheduler to be able to maintain correctness. The actual "turning-off" may be done in terms of power-gating, which would require *lane* level power-gating or in terms of scheduling/clock-gating. If after the first *lane* is turned off, there would still be deviation, a new *lane* is then turned-off. This could lead to a big impact on the performance, thus "Lane turn-off" should only be used if Dummy-activity is not able to bring V_{mid} to an acceptable level.

Additional VRs are used for extreme cases. In the stacked logic, it is expected to have Vmid floating, however, within a range that guarantees its functionality. Two small integrated voltage regulators are used, one pull-up and one pull-down, which would be activated when V_{mid} falls below 0.4V or rises above 0.8V, to guarantee correct behavior and avoid bit flips. Since these regulators only cap extreme cases, they are not used in regular operations, therefore, they can be small and their potential inefficiency is not problematic in the overall design. Figure 4.7 is the circuit diagram of our proposed pull-down VR. This is simplified. V_{ref1} is the lower voltage level and V_{ref2} is the upper voltage level, so that V_{mid} will float in the middle. The design is adapted from a Low Dropout Linear VR design [67], which can be used for the pull-up VR.

The natural candidate for triggering these mechanisms is V_{mid}, so in the case of the architectural mechanisms, the scheduler needs to be aware of the voltage during the regular operation of the GPU. VR is always connected and is triggered without any architectural intervention. In our experiments (details in Section 4.6), those mechanisms were never activated for any of the benchmarks tested, even when divergence was observed (for instance in BFS).

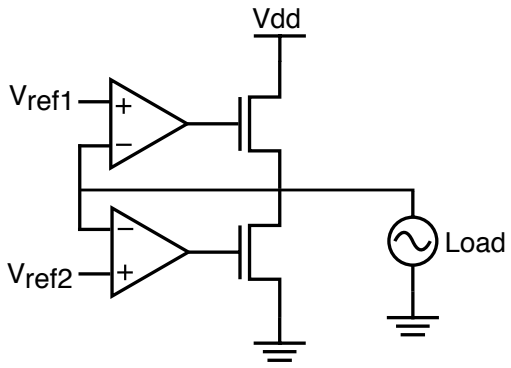


Figure 4.7: The proposed dual VR based on the Low Dropout Linear VR design [67].

4.3.8 Final Design

GPU stacking does not change GPU organization (Figure 4.4), nor does it affect the placement. GPU stacking divides the GPU into different power domains, one for non-stacked structures (*e.g.*, caches, shared memory), and one "super" power domain with the stacked *lane*s (and associated register files). It is a "super" power domain because, to be precise, each stack is a power domain of its own, but this is not known before fabrication. Figure 4.8 shows the proposed micro-architecture, from a power delivery perspective. Each register file (RF) should be stacked with the respective lane, to avoid the need for level shifters between them. The figure shows a solution with four Shared Nets. Note that the stacked SRAMs in Figure 4.8 are similarly stacked to the design proposed in Chapter 3.

4.3.8.1 Area Overhead

One concern about GPU stacking is the area overhead due to Shared Nets. Introducing extra Shared Nets could increase the total amount of metal dedicated to power rails. On the other hand, GPU stacking decreases the overall current of the chip and metal from V_{dd} and *Gnd* rails could be reduced.

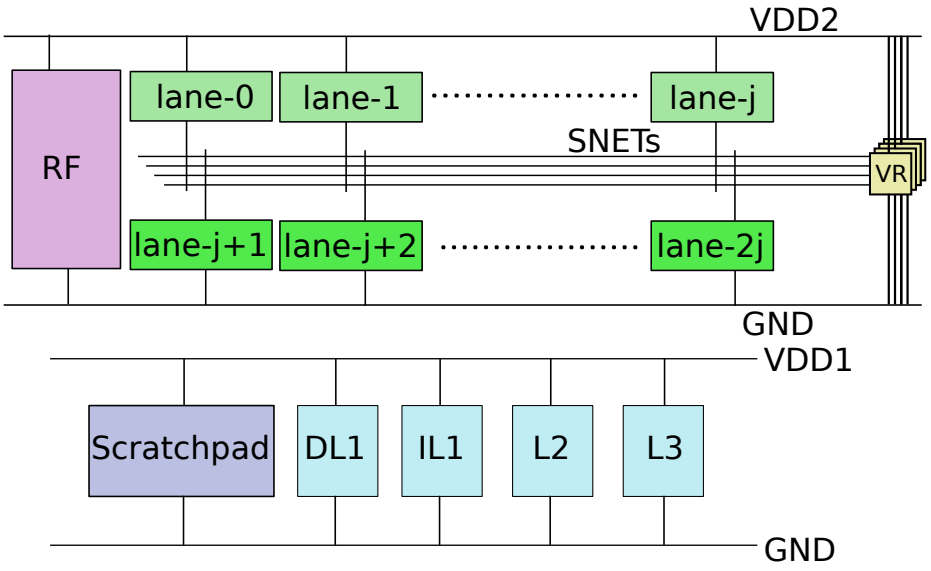


Figure 4.8: GPU stacking allows a more fine-grained voltage adjustment per lane.

Let m_b be the total metal budget for power rails in a chip. For a 2-level stacked system (as the one proposed here), the overall current is reduced by half $I_{stack} = I_{base}/2$. Thus, V_{dd} and *Gnd* could have roughly half the metal budget as needed in the baseline[5] ($m_b/4$ for each). For a system with n Shared Nets, each Shared Net takes I_{stack}/n current. Thus, each Shared Net needs $\approx n$ times less metal than V_{dd} ($m_b/4/n$). Consequently in GPU stacking, the total metal budget for power rails is $m_{vdd} + m_{gnd} + m_{snets} = m_b/4 + m_b/4 + n \cdot m_b/4n = 3/4 \cdot m_b$. In other words, in a GPU stacking configuration it is possible to reduce the amount of metal dedicated to power rails. In our evaluation, we assume that the metal budget is kept constant instead (*i.e.*, V_{dd} and *Gnd* metal is reduced by 1/3 only), which in turns helps improving the PDN by reducing the resistance.

[5]For clarity, we are referring to the amount of tracks dedicated to rails, not their pitch.

4.4 Experimental Setup

Our experimental setup has two main components. In the first part, we evaluate the potential of GPU stacking to compensate for process variation both in performance and power. Then, we evaluate the reliability of the PDN, as it is a main concern in voltage stacking proposals.

4.4.1 Process Variation Modeling

To evaluate the impact GPU stacking has where it compensates for the performance loss due to process variation, process variation needs to be modeled. We use a similar modeling flow to Thomas *et al.* in their *Core Tunneling* study [76]. Figure 4.9 shows the step-by-step flow of VARIUS-NTV [39] which is the process variation modeling tool used throughout this study. VARIUS-NTV [39] has models to compute the effect of parametric variation on logic and memory structures and it is a recommended tool for managing parameter variation when operating in near threshold voltages [38]. It is a tool based on VARIUS [70] and VARIUS focuses on WID variation and divide it into systematic and random components and they assume their effects to be additive. The systematic component is modeled using a multivariate distribution with a spherical spatial correlation structure and the random component, which occurs at the transistor level, is modeled analytically. VARIUS divides the chip into n small equally sized rectangles. Each grid point has a systematic variation of L_{eff} and V_{th} which are assumed to have normal distribution. The random variation of L_{eff} and V_{th} is treated differently because of the level of granularity at which it occurs and it is assumed to be distributed normally and without any correlation [70].

Given the GPU floorplan as the input, VARIUS-NTV provides die maps each with a specific process variation case. The goal is to use VARIUS-NTV to consider the worst process variation over maps or die maps and understand how V_{mid} behaves in extreme cases of process variation effects or an application. VARIUS-NTV also outputs normalized delay, normalized L_{eff}, and effective V_{th} for each component of the GPU die (lanes, caches, register files, etc.). The information from VARIUS-NTV is then used to calculate the expected power for each element in the stack and then the expected voltage on each Shared Net. The calculated V_{mid} is then fed back into VARIUS-NTV to calculate the delay and power after compensation in the stacked configurations.

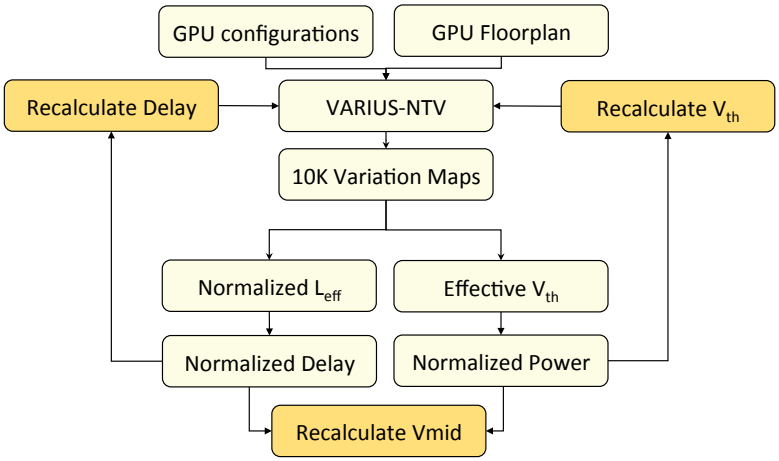


Figure 4.9: Experimental setup with VARIUS-NTV process variation modeling flow. VARIUS-NTV generates GPU variation maps and then calculates the normalized L_{eff} and effective V_{th} for each map. V_{mid} value changes after process variation effects and is calculated and fed back to VARIUS-NTV for compensated delay and power calculations.

This experiment is performed for different number of Shared Nets, where we compare our scheme with a conventional non-stacked baseline and against multi-frequency domain, which has been shown to have promising results in mitigating process variation effects [40].

4.4.2 Simulation Framework

To evaluate the V_{mid} noise and voltage noise behavior in the GPGPU PDN, the time varying impedance in Equation 3.1 needs to be determined. We adapt the same methodology used by Leng *et al.* [55] for forming our simulation framework and modeling our power delivery network. In order to show how impedance in each core changes over time, we use ESESC, a cycle-accurate superscalar simulator [35], to approximate the current variation profile of each core under a certain supply voltage level. This is an approach used in related research studies led by Leng *et al.* [54, 55], Thomas *et al.* [76], and Kim *et al.* [45].

A modified version of ESESC [35] is used to simulate a GPGPU. For power estimation, we use a GPGPU model developed based on McPAT [56], very similar to GPUSimPow [57]. McPAT, integrated with ESESC, takes the microarchitectural activity statistics from ESESC, and calculates the power consumption of each component. This simulation setup provides both dynamic and leakage power. The temperature dependency of leakage is also taken into account. The power traces are then used to generate a time varying impedance model for each core or *lane*.

4.4.3 Power Delivery Network

The power delivery system of a chip consists of off-chip and on-chip components. The off-chip network includes a voltage regulator, capacitors used to stabilize supply voltage and other components. On the chip, power is delivered through a set of pins and C4 pads that deliver the required voltage to the various chip components. To model the entire PDN and evaluate the voltage supplying the cores, we adapt the modeling flow (Figure 4.10) proposed by Leng *et al.* [55].

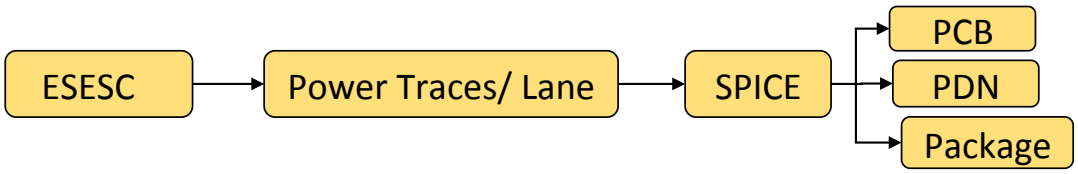


Figure 4.10: The power delivery model tool flow.

To evaluate V_{mid} voltage, voltage noise, and current during execution, the impedance models obtained using our simulation framework 4.4.2 are run by SPICE model that simulates the printed circuit board (PCB), the package, and the on-chip PDN. In this flow, we use the off-chip power delivery model proposed by Leng *et al.* [55] and the on-chip IBM Power Grid benchmark (ibmpg1t) [61]. Figure 4.11 shows a lumped RC model of the complete PDN with the simulation parameters. The grid is represented by the four resistors in the box named "PDN", but the simulation is performed using the full grid. The cores are modeled as variable resistances based on the power traces from ESESC. This methodology is compatible with current industry practices, and short of fabricating a chip, it is the best available method for this type of low-level analysis.

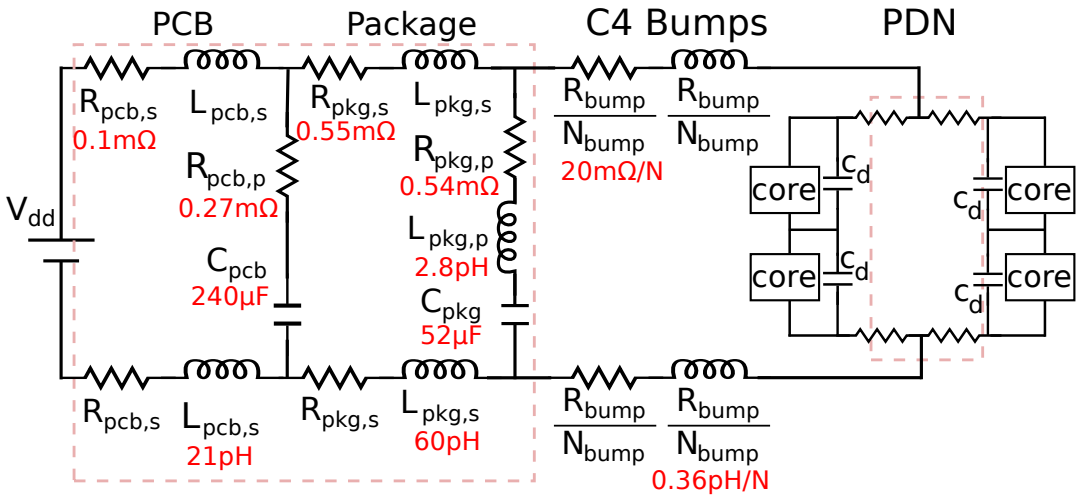


Figure 4.11: The complete power delivery model used for simulations.

Since each stack level will operate at a different voltage level, we will assume to have the level shifter proposed in [25]. Our SPICE simulations show that this circuit has one FO4 delay overhead when communicating between different stacks, however these level shifters can substitute buffers that were present in the design, minimizing the performance impact. Memory and caches are not stacked in this study, and since the process variation is being mitigated due to the stacking in the cores, it is expected that they will end up achieving a higher frequency than memory. To keep the improvement on the logic side, we consider an increase in the number of access cycles rather than reducing the frequency of the core.

Each SM runs at a single frequency. In order to meet timing, this will be the frequency of the slowest *lane* (after compensation). It is possible to have different frequencies for different SMs, but in this study only one frequency is considered for the whole GPU.

4.5 Near Threshold Computing and Baseline Choice

In this section, we analyze how NTC affects choosing architectural parameters in GPUs to determine the baseline configuration for the remainder of this chapter and provide an optimal starting point for the evaluations.

4.5.1 Power-Performance Trade-off

The NTC region is considered to be the most energy-efficient area for operation [39]. Frequency has a linear relationship with the voltage down to the near threshold region. Going from 1V to 0.5V, the device delay increases by factor of 2, while the power is lowered to $(1/2)^3$ of the original value which results in a reduction of energy consumption, as a product of power

and delay, to $(1/2)^2$ of its original value.

The performance loss can be compensated with extra resources to support more parallelism. If we consider $2\times$ more resources to compensate for the $2\times$ increase in delay, the power would increase by a factor of 2, and the delay would decrease by half, leaving the energy reduction unchanged. So ideally, without considering the impact of process variation and faults, energy consumption can be cut to $(1/2)^2$.

4.5.2 GPU Sizing for NTC

We use the simulation flow discussed in Section 4.4.2 to simulate a GPGPU with a range of configurations, created either by varying the number of SMs or the structures within each (*e.g.*, number of *lane*s in each SM). This is summarized in Table 4.1. McPAT [56] tool estimates the power consumption of the GPGPU model and only the on chip structures are modeled for this experiment.

Benchmarks used are from popular suites like Rodinia [11] and Parboil [74] and a few from the CUDA SDK (*bfs, cfd, convolution, hotspot, backprop, lbm, transpose, srad* and *sgemm*).

Figure 4.12 shows the trend on average across all the evaluated applications for energy-delay product (ED), and energy-delay-area product (EDA). The y-axis shows the normalized value for each metric with reference to the $1x$/4SM configuration at 1V supply. The x-axis is V_{dd}. The figure shows the trend for $1x$ configuration–the most energy efficient. In general, energy decreases as the V_{dd} approaches near threshold region. Then the delay starts to degrade more rapidly, increasing the energy consumption, mainly due to clocked logics and leakage.

Parameter	1x	1.5x	2x
Processing Elements (*lane*s) per SM	32	32	64
Register File (RF) per SM	32K	64K	64K
DL1G-Scratchpad memory per SM	32KB	32KB	64KB
Maximum Frequency	1.5 GHz		
Streaming Multiprocessors (SM)	up to 8		
Threads per warp	32		
Maximum Warps per SM	24		
L2	256KB 16w		
L3	4MB 32w		
Memory access latency	180 cyc		
V_{dd}	0.4-1.0 V		
V_{th}	0.30 V		
δV	0.1		
Ambient Temperature	25C		

Table 4.1: Simulation parameters

EDA is of particular interest in this study. Let's consider the 4SM configuration operating at 1.0V as the baseline configuration. The baseline meets the power budget for the chip. Increasing the resources would result in exceeding the power budget. Thus 8SM, for example, is not a feasible configuration in this condition. Now let's consider the *EDA* of 4SM and 8SM configurations. As expected 8SM has a higher *EDA* than 4SM at 1.0 V. As the V_{dd} approaches the near threshold, *EDA* of both configuration decreases. However, at V_{dd} of around 0.6 V, *EDA* of 8SM configuration crosses that of 4SM. This means that the 8SM configuration is more effi-

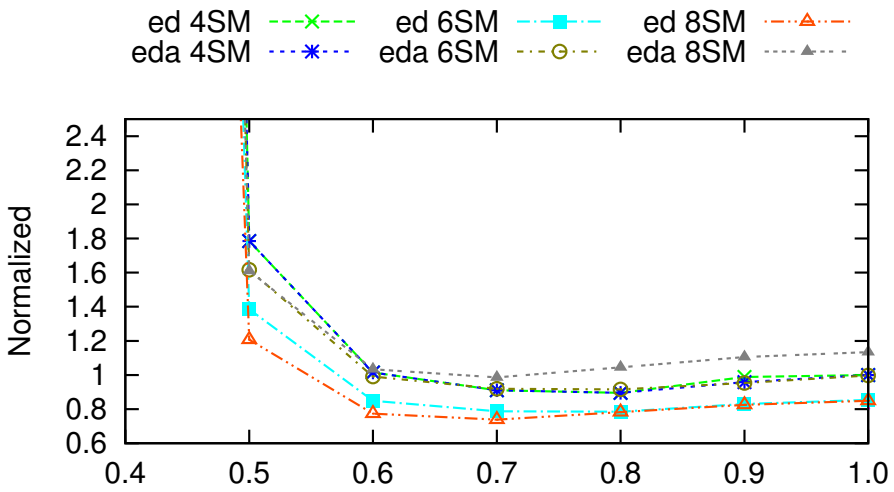
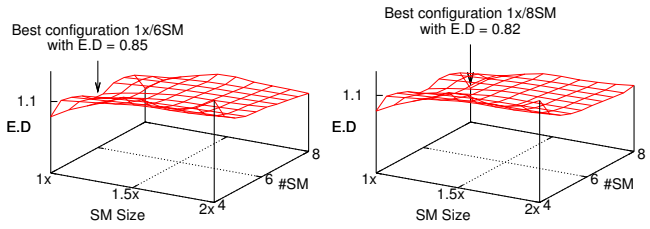


Figure 4.12: Designs with more SMs become more efficient in terms of energy-delay product, as well as energy-delay-area product as V_{dd} decreases.

cient at lower voltages. Also the delay metric for 8SM configuration at this point is the same as the delay of the baseline 4SM running at 1.0 V. This shows that investment in extra resources pays off as the V_{dd} approaches the near threshold region by maintaining the performance within 4.8% of $1x$/4SM configuration operating at 1.0 V, while reducing the power consumption to about 43% of baseline.

Another observation is that the optimal configuration for different metrics changes by changing the V_{dd}. For example Figure 4.13a shows the design space for ED at 1.0 V. The optimal configuration is $1x$/6SM. Bigger structure sizes for cache or number of *lane*s could increase the performance, but the increased power makes such a trade off less desirable due to power budget constraints and possible thermal issues. Figure 4.13b shows the design space for the same metrics at 0.6 V. At this condition, the optimal configuration is 8SM. Also the relative efficiency of bigger structure sizes (*e.g.*, 1.5×) increases. This implies that the architectural parameters, such as cache or RF size, should be reconsidered for maximum efficiency as the

operating voltage changes.



(a) Typical **(b)** Near Threshold

Figure 4.13: The optimized baseline in near threshold region is different from the typical super threshold region. Larger structure sizes for cache or register file or number of lanes could become more desirable in near threshold. This demands reconsidering the architectural parameters to obtain the best energy efficiency, rather than just lowering the voltage.

4.6 Evaluation

We start our discussion by presenting the main results for GPU stacking for mitigating process variation in GPGPUs operating in near threshold region. Then, we analyze the reliability of the PDN, in particular we show that GPU stacking does not increase voltage noise or droop and that the lateral current in the long Shared Net rails is not a problem for GPU stacking. We end our evaluation by discussing some practical aspects for implementing GPU stacking in real chips.

4.6.1 Main Results of GPU Stacking on NTC

We evaluate our method to manage process variation at near threshold (0.6V) supply voltage with single frequency. These results were generated using variation maps generated by

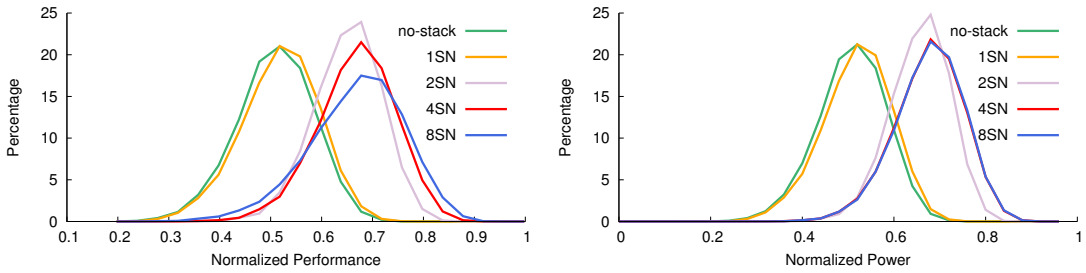


Figure 4.14: The proposed techniques shift the performance and power towards the ideal scaling with no process variation.

VARIUS-NTV (each map corresponds to one die), as described in Section 4.4.1. The baseline GPU was modeled as a $1x/8$SM, as explored in Section 4.5. We then estimate the performance and power for each die. Figure 4.14 shows histograms for performance and power, *i.e.*, the y-axis is the percentage of chips (out of 10K chips) and the x-axis is the performance/power. Performance and power are normalized to the value obtained in the case with no process variation. Both Non-Stacked and Stacked (xSN, where x is the number of Shared Nets) methods are shown. Two and four Shared Nets provide good design solutions, but we consider the four Shared Nets case. In all the cases, the non-variation threshold voltage is set to 0.35. **GPU stacking delivers about 75% of the performance, with 75% power, compared to the no variation conditions**. This represents a reduction in the degradation due to process variation: 37% in performance, and 39% in power compared to the conventional configuration. The increase in power is due to the increased frequency, but *Energy-per-instruction* remains roughly the same. This may seem like a no-gain approach, but means that GPU stacking is able to reduce the effects of process variation, delivering a chip that is closer to ideal scaling.

We now consider the case where multiple clock domains are present. GPUs are specially suited for this type of scenario, because SMs provide a natural bound for clock domain

partitioning. Also, letting each SM run at its maximum frequency yields very good performance results even when process variation is considered [50]. This technique is orthogonal to GPU stacking, and we leverage it in our evaluation. Our results (Figure 4.15) show that the use of multiple clock domains does increase the overall performance of the GPU when process variation exists, but this improvement is limited compared to those of the super-threshold region [50]. Still, GPU stacking improves by 33% the overall performance of the GPU, **delivering about 80% of the nominal performance, with 70% of the power.**

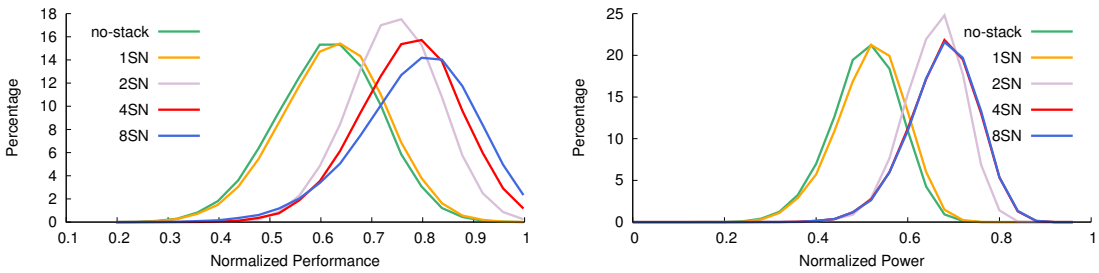


Figure 4.15: When multi-clock domain architecture is used, GPU stacking shifts the performance further towards the ideal scaling with no process variation.

4.6.2 Load Mismatch

In typical GPGPU applications, threads exhibit a very similar execution, or activity rate. This minimizes the possibility of a load mismatch in the stack. In our experiments, we observe that the power consumption of *lane*s are within 5% of each other 98% of the time, and is within 10% of each other 99.2% of the time. $Power_{lane1}/Power_{lane2}$ averages in 1.000024, with standard deviation of 0.070. The sampling rate for our measurement is on the order of 1-10 MHz.

One source of concern is peak difference, which can be as high as 40% in our exper-

iments. We observe that mismatches higher than 30% only occurred for the *backprop* benchmark, but only 0.1% of the time for that benchmark. Since the elevated mismatches are observed in a very short interval, they can be handled by decoupling capacitors between V_{dd}/ V_{mid} and V_{mid}/ Gnd. The mismatch evaluation for the same kernels is carried out in this section.

To examine whether the achievable range of decoupling capacitance in the design is enough for the observed mismatches, we ran a SM-like design through synthesis and back-end design down to GDS. We then extracted the capacitance of the supply nets. Our experiments show that such a design using standard cell decoupling capacitance cells (dcaps) would have total capacitance of $1.4nF/mm^2$. At the super threshold region, with a power density of about $1W/mm^2$, the time constant for the power supply RC circuit would be on the order of a couple of nanoseconds. Such a small time constant is not enough to sustain the transient mismatches that appear on the order of $100ns$ to $1us$. At the near threshold region, with smaller power density, the time constant would be on the order of 10 nanoseconds which is still not enough.

To increase the decoupling capacitance, a technology similar to the one used in fabricating DRAM memory cells can be used. Trenches in the silicon can provide over an order of magnitude more capacitance per area compared to the standard cell $dcaps$. DRAM cells are as big as $4F^2$ to $6F^2$ (F is the feature size, *e.g.*, 22 nm) with about 25-30fF capacitance across generations. With less than 1% density of such cells in the design, the amount of capacitance per area, and consequently the time constant can increases by an order of magnitude.

The same process for trench capacitors can be incorporated into a CMOS process like IBM has done. In such system, a lane operating in near threshold has close to 1000 cycles before the voltage fluctuates by more than 10%. Effectively, giving enough time to change the workload if needed. Nevertheless, we do not see such need for the applications evaluated

because of the high regularity of GPGPU workload. If the trench capacitor is not an option, it is also possible to use on-package capacitors. On-package capacitors are common to stabilize current PDNs. Adding additional capacitors for the Shared Nets will not increase the package pins.

4.6.2.1 Load Mismatch in a process variation scenario

This figure changes when process variation is considered. The load mismatch increases. But as discussed in Section 4.3.3 this mismatch is desirable in GPU stacking as it compensate the delay variation caused by process variation. We evaluate the voltage mismatch between two core-configurations, as explained in Section 4.4.3. We evaluate how the load mismatch affects the voltage available for both levels of the stack by carrying a SPICE simulation of the model presented in Figure 4.11. Each stack contains 16 lanes, 8 in the header and 8 in the footer, a total of 4 stacks (2 SMs with 32 *lane*s each) are hooked in the grid, equally distributed, each V_{mid} has two $5\mu F$ on package decoupling capacitor (one between V_{mid} and *Gnd* and the other between V_{mid} and V_{dd}). BFS and backprop are run, one in each SM (those where the two benchmarks with higher mismatch between lanes).

Figure 4.16 shows the on-chip transient voltage for each stack level during the execution time, one of the stacks is omitted for clarity. The global voltage source is $1.2V$. Instead of plotting the voltage with relation to the global *Gnd*, we plot the local voltage difference, which is more meaningful. Voltage for each stack level stays within 10% the expected voltage for compensation, showing a very good balance.

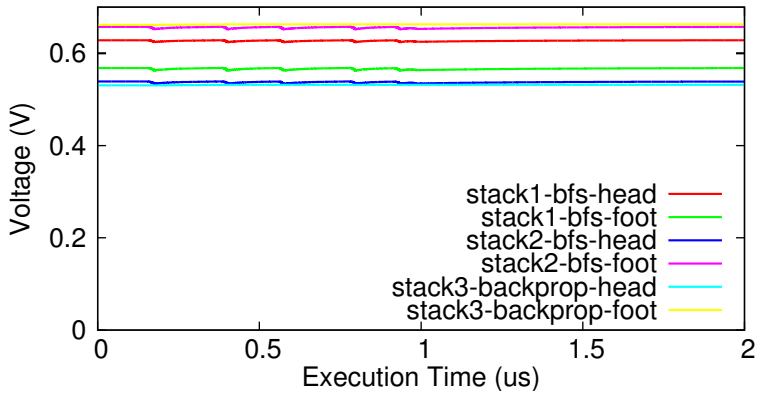


Figure 4.16: It is possible to maintain the voltage in each level of the stacks with on package decoupling capacitors .

4.6.3 Stacking FinFETs vs. Planar CMOS

All the evaluations thus far used traditional planar technology. This section evaluates the use of FinFET devices. BSIMG-CMG SPICE models are used for common multi-gate, i.e., FinFET devices to validate the concept of stacking. BSIM-CMG models are developed by UC Berkeley BSIM Group and designed to be scalable and predictive MOSFET SPICE models that can be used for circuit simulation and CMOS technology development [41]. FinFET models used, aim at the 15nm technology node. Repeating the planar CMOS experiments in 4.3.3, results in the graph in Figure 4.17. It shows an interesting trend in FinFETs. There is an inversion in the V_{mid} trend, which seems to have eliminated the compensation effect of stacking observed for planar devices.

FinFETs behave differently under process variation. For instance, the plots in Figure 4.17 are obtained using a simple inverter example with an FO4 and tracking its delay and power. The Energy-Delay graphs in Figure 4.18 show as we increase L_{eff} variation in FinFETs, the effect on energy-delay is much smaller than planar CMOS. FinFETs are more tolerant of

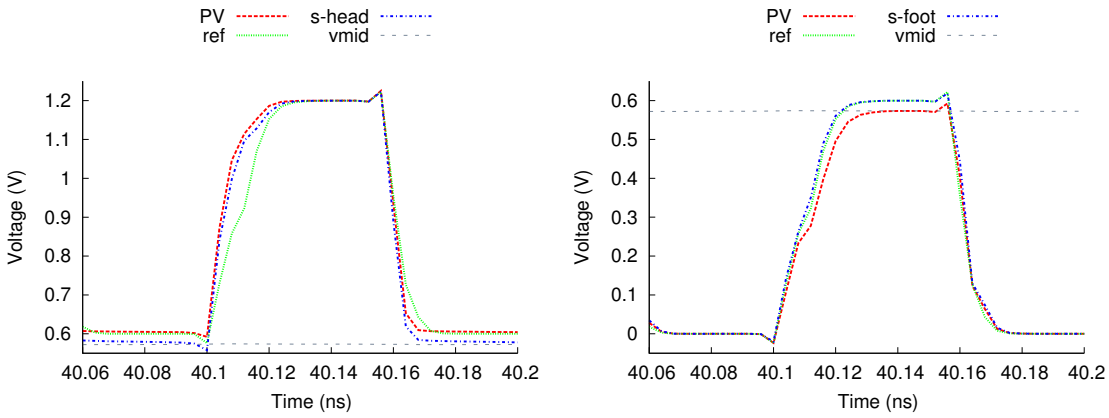
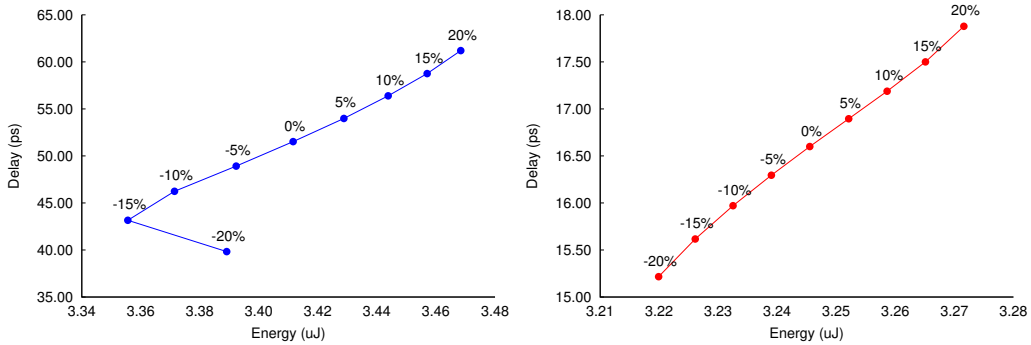


Figure 4.17: The intrinsic compensation of PV effects seen for planar devices is not observed for FinFET devices due to the inverted ED trend between those two technologies.

variation.



(a) Planar CMOS 45nm

(b) BSIM-CMG FinFET 15nm

Figure 4.18: Delay increases more rapidly for Planar CMOS than FinFET as we vary L_{eff}.

Nevertheless, it is still possible to use GPU stacking to compensate for PV effects on FinFET devices. To do so, we propose the use of the extra VRs to force the V_{mid} voltage to a beneficial level. GPU stacking is designed and evaluated in the same way, but during the binning process, a voltage level is chosen for each Shared Net based on the variation of the *lane*s assigned to it. The VRs need to be slightly larger in the FinFET version of GPU stacking

than in the planar version, but is still much smaller than what is used in a regular multi-VDD domain system which requires full fledged VRs for each voltage domain.

Compared with the bulk planar devices, the challenge in analyzing FinFET circuit delay lies in the computation of the different timing parameters. Currently, no FinFET timing model has been presented that is both efficient and sufficiently accurate [85], and no analytical models short of SPICE models exist to accurately estimate FinFET timing.

S. Khatamifard *et al.* performed a case study on thin channel FinFETs. Due to stronger control of the gate over the channel, FinFET features better performance and power characteristics. To see how miniaturization of technology nodes affects digital switches, they build a model using VARIUS [70] modeling tool as the base, called VARIUS-TC [42]. Similarly to VARIUS, it models variation for logic and memory cells, but it targets thin channel FinFETs instead of planar MOSFET. The tool decouples architecture-level analysis from circuit- and device-level characterization by abstraction. It uses look-up tables to capture electrical characteristics which the circuit module uses to extract power and performance characteristics at the logic gate and memory cell level, and then passes them to the architecture module. Overall, VARIUS-TC is able to keep track of architectural implications of process variation in emerging switches [42].

The experiment in Section 4.6.1 is repeated to evaluate the impact of GPU stacking to compensate the performance loss due to process variation in a FinFET scenario. 50,000 variation maps are generated using VARIUS-TC [42]. *lane*s were clustered following the strategy proposed in Section 4.3. VARIUS-TC is then used to calculate the delay and power for each of the *lane*s with the set voltage. Figure 4.19 shows histograms for performance and power. The y-axis is the percentage of chips (out of 50K) and the x-axis is performance/power.

In Figure 4.19 performance and power numbers are normalized to the values of the no-process-variation case and we consider the four Shared Nets case. GPU stacking improves performance by 4% with 4% additional power consumption, compared to the no variation conditions and it still reduced the performance degradation due to PV effects. The improvement itself is much smaller compared to planar MOSFET 4.14, nevertheless, the FinFET baseline understandably is more power efficient and has better performance than the planar baseline.

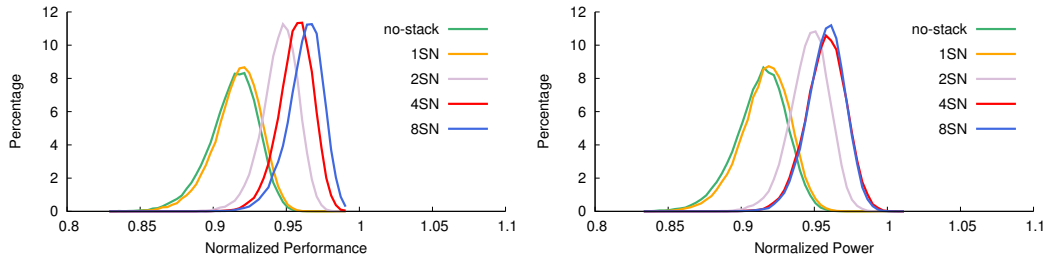


Figure 4.19: The proposed techniques shift the performance and power towards the ideal scaling with no process variation.

Replicating the planar MOSFET evaluation for FinFETs, we let the GPU operate in a multiclock domain and the results in Figure 4.20 show that GPU stacking improves the overall performance of the GPU by 4% with a 4% increase in power consumption.

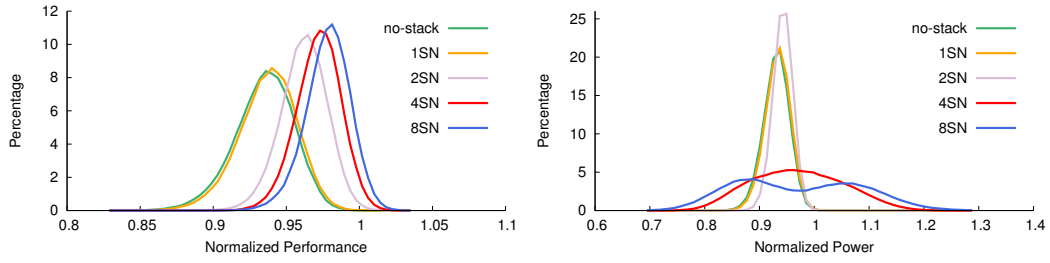


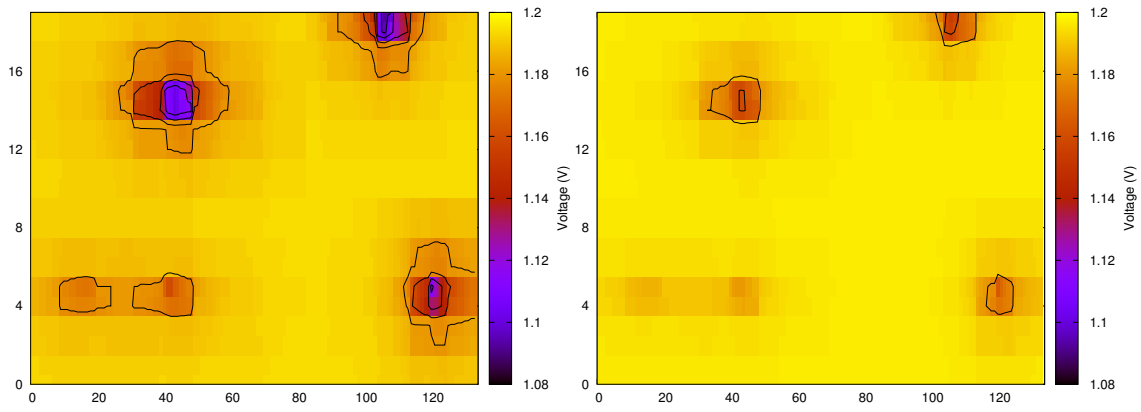
Figure 4.20: When multiclock domain architecture is used, GPU stacking shifts the performance further towards the ideal scaling with no process variation.

When using FinFETs in a stacked architecture, the compensation trend in power and performance is different, in such way that the power delivery mechanism is mainly improved as opposed to that of a multi-VDD domain. The number of power rails decrease and the number and the size of voltage regulators used decreases. For instance, if a voltage regulator is used per Shared Net, in a multi-VDD domain, we need to utilize 8 voltage regulators. In a stacked architecture, we will need to use 1 voltage regulator to keep V_{mid} at an optimal level and to benefit from the stacking technique.

4.6.4 Lateral Current

One of the main concerns in GPU stacking is lateral current, and thus the IR drop. We evaluate it both for V_{dd}, and for V_{mid}. The total current for V_{dd} in GPU stacking is roughly half the current than the conventional case, therefore, one would expect decreased voltage droop. On the other hand, given a fixed budget for power delivery, the insertion of new power rails requires that some of the resources used by V_{dd} and *Gnd* in the conventional scenario be used to Shared Nets instead. Since the current in this rails is expected to be much smaller, the resource reduction in V_{dd} and *Gnd* is small. We consider that 1/3 of the metal used to power delivery is used to Shared Nets. Figure 4.21 shows a 2D color map of the IR drop for V_{dd}, both in the conventional configuration and in GPU stacking for the BFS benchmark from our SPICE simulations, already considering the reduced resources for V_{dd}, the contour lines are traced every $0.02V$. The conventional configuration was scaled up for better comparison.

From the variation maps generated by Varius-NTV [39], we find the worst case (maximum power difference in a stack). We note that, since we are stacking lanes, within an SM, and given the spatial correlation of the variation, in most cases the variation is well below this max-



(a) Conventional

(b) GPU stacking

Figure 4.21: GPU stacking reduces the IR drop by reducing the total current flowing through V_{dd}.

imum. To estimate the resistance, we use the IBM Power Grid Benchmark (ibmpg1t), which is properly scaled to estimate the equivalent grid of one Shared Net. The transient simulation considers the maximum variation case. The maximum observed IR drop in V_{mid} was $65mV$ in our simulations for all benchmarks, which is 5% of the whole supply voltage ($1.2V$).

4.6.5 GPU stacking Practicality Issues

A key concern for GPU stacking is practicality. This can be viewed from different perspectives: in-rush current, non-uniform benchmarks, and difficulty of implementation among many. We will discuss some of these concerns and how to address them.

4.6.5.1 Extreme variation cases

In some extreme cases, the difference in variation between the stack levels may be significantly elevated, reducing the voltage in one of the levels to non-acceptable levels (compared to V_{TH}). After fabrication, chips are divided in performance/power grades or discarded

according to the variation. This should also be the case in the GPU stacking case. To evaluate how GPU stacking affects yield, we use VARIUS-NTV [39] to calculate the expected V_{mid} for each stack. We then compare this value to the V_{th} of each lane in the stack. If the supply voltage in the lane is smaller than $1.2 \cdot V_{th}$ for at least one of the lanes, we consider the chip discarded. Our simulations show that, in such scenario, 1.8% of the chips would be discarded. A more aggressive approach would be to disable lanes which are not meeting such requirement, but we do not consider such scenario.

4.6.5.2 In-Rush current and power gating

GPU stacking reduces the current drawn from the PDN by roughly half during the regular operation. This is also the case for the start-up phase. Our SPICE models were simulated for both stacked and non-stacked configurations, and showed that in-rush is indeed smaller in GPU stacking (data is not included).

Power gating can be applied in addition to the GPU stacking. The only requirement is that pairs of lanes need to be powered off together (one in the head and one in the foot group), but there is no need to power gate one entire SM at a time.

4.6.5.3 Implementation flow

GPU stacking operates in the physical level only and does not make changes in the RTL level. But the physical implementation flow needs to be altered. GPU stacking creates voltage rails that will be connected to the "local" GND rail of some portions of logic and to the "local" V_{dd} rail of others. The idea of having multiple power rails is similar to multiple power domains. Power transistors can then be used to open or close the circuit and choose which

Shared Net each core will tap. Another option is to crack open some connections, since the clustering decision do not change after fabrication.

Voltage stacking also requires isolated wells to avoid unwanted body bias effects between stack levels, thus either Fully-Depleted SOI or a simple triple-well technology are needed [52]. Although this is restrictive, FD-SOI has high availability and triple-well only requires an extra mask during fabrication.

4.6.6 Other advantages of GPU stacking

The increase in the supply voltage has some indirect advantages, some of which are quantified in this section. They share two main causes: voltage regulators are more efficient at higher voltages and the overall current in the circuit is reduced to roughly half [25, 52]. Using results from Hong *et al.* research [27], we estimate memory power as 17% of the total GPU power. The memory power is not affected by GPU stacking.

Some of the main advantages of GPU stacking are related to the voltage regulators: the voltage regulator (VR) area is reduced by roughly $2\times$ with half the current [16]. In the Intel Haswell integrated VR, each $2.8mm^2$ cell can delivery a 25A maximum current. A modern low-end GPU consumes $\approx 55W$ [1]. VR efficiency is a function of both V_{dd} and output current [75]. Considering both the efficiency from increasing V_{dd} from $\approx 0.6V$ (at the near-threshold region) to $\approx 1.2V$, and reducing by half the total current, we expect an improvement in the VR efficiency of 12%. This is a saving of 10% in total power (excluding memory). Also, it is possible to reduce the VR for the logic to 50%, this yields a reduction of 41% in the total VR area.

The number of pins and pads is mainly determined by the total amount of current flowing through them. To keep the current per pin constant, it is now possible to reduce the

Parameter	Expected Variation
Performance vs. PV not-stacked	+37%
Power vs. PV not-stacked	+39%
VR area	-41%
VR efficiency	+10%
Supply pins	-41%

Table 4.2: GPU stacking delivers better performance and power than a non-stacked configuration under process variation. It also allows better VR efficiency, and reduced number of power pins.

number of pins. Again the current related to the logic decreases to roughly half, but the current in memories is the same. Once more, this yields a reduction of 41% in the total number of power pins. Note that the number of pads dedicated to V_{dd} can be also decreased, but pads are now needed for Shared Nets, since on-package decaps are used. The overall number of pads is not expected to change. Table 4.2 summarizes the expected variation in multiple chip parameters due to GPU stacking.

4.7 Conclusion

This part of the dissertation presents GPU Stacking as a method to manage the effects of process variation. We show that the stacking of cores with opposite variations tend to balance the variation effects that would have appeared in a conventional configuration. To maximize the balancing effect, cores with opposite variations should be stacked, which requires post-silicon configurability. We propose a clustering technique to make such a configurability feasible. The homogeneous nature of GPGPU applications make them suitable candidates for GPU stacking.

Previous voltage stacking publications only analyzed multicores and required complex circuitry to stabilize the voltage. This work is the first to use stacking in the context of process variation, and also the first to propose a floating middle rail. We show that the stable nature of GPGPUs allows for the use of only decoupling capacitors to stabilize the power delivery.

This research provides a detailed evaluation of NTC with GPGPUs, and the idea of GPU stacking. We first carefully size a GPU for NTC operation, achieving 43% power savings, with only 4.8% performance degradation. We then apply GPU stacking to manage process variation, which impacts NTC circuits more than circuit in the super-threshold region. The homogeneous nature of GPGPU architectures and applications, make them a very interesting candidate for exploration in the extreme domains, with both low voltages and small feature sizes. We show that stacking can increase performance under process variation at near threshold, on average, by 37% compared to the traditional (not stacked) configuration, delivering 80% of the performance compared to the no variation (ideal) conditions.

Chapter 5

Timing Speculative SRAM

5.1 Introduction

As fast on-chip memories continue to occupy a great portion of the area in System-on-Chip (SoC) designs, their speed and power consumption significantly impacts the system performance. Up to 60% of the total active chip area is taken by memories and 20% of that area is taken by SRAMs [64]. In SRAMs, read operations are the most time consuming operations that affect the access time. Therefore, optimal Sense Amplifier Enable (SAE) signal generation is crucial for having speedy SRAM operations that consume low power. As technology scales down, intra-die variations such as random dopant fluctuations affect Vth and become more pronounced in small circuits such as SRAM cells causing various read and write failures. Timing guardbands prevent such failures, while inherently requiring an over-design to meet the performance goals and an increase in the overall power consumption. Hence, there is a growing demand for architectural schemes and sensing circuits which ease the conventional design methods.

Razor is one of the most common delay-error tolerant flip-flops used in voltage management techniques. It eliminates the frequency margins needed due to PVT variations [21]. Follow up designs to the original Razor proved that recovering from errors is highly costly and resorted to detecting the timing errors only [17]. Read errors have previously been detected using a Razor-based technique [37] in the boundary sense amplifier. Unfortunately, the additional logic for error detection and correction adds significant area to the sense amplifier and, in turn, increases read energy consumption. In the case of having to recover from an error, Razor will add a penalty of one cycle delay to every read operation in order to recover the correct data. In addition, when the error detection scheme is applied on a per bit basis, the final error signal is a logical *OR* of all the error signals which adds additional delay in computing the error signal. These drawbacks limit the overall efficiency as the size of the SRAM increases [17, 21]. In addition, Razor-based SRAM is assumed to be placed in a system that is *Razor-enabled*, however in case of a system error, there is no mechanism for preventing incorrect writes to the SRAM. Therefore, there is a chance of corrupting the SRAM data.

We propose a timing error detecting scheme that utilizes existing hardware called Replica-based Timing Speculative SRAM (RTS) which can

- leverage the RBL concept in timing speculation due to variation,
- prevent erroneous writes,
- provide an area and power efficient error detecting mechanism.

5.2 Related Work

RBL-based SRAMs: During the read operation, sense amplifiers can become acti-

vated too early and misread the data or stay open for a long period dissipating power more than needed for detecting the bitline voltage swing [6]. B. Amrutur proposed the Replica Bitline (RBL) technique to optimize the SAE timing signal and ensure protection against erroneous reads [4] (Figure 5.1. The conventional RBL is a column containing bitcells similar to the SRAM core bitcells. During the read operation, the RBL control signal is asserted. A number of RBL cells that have been hardwired to '0' are going to fully discharge. The column output signal is then buffered, inverted and used as the SAE signal. The buffering is needed to replicate the wordline delay. RBL predicts the maximum discharging delay of the SRAM core bitlines. Once RBL column is activated, the precharged bitline entering the column will begin discharging. The time it takes RBL to discharge through the column bitcells, and fall below the threshold voltage is used to generate the SAE signal.

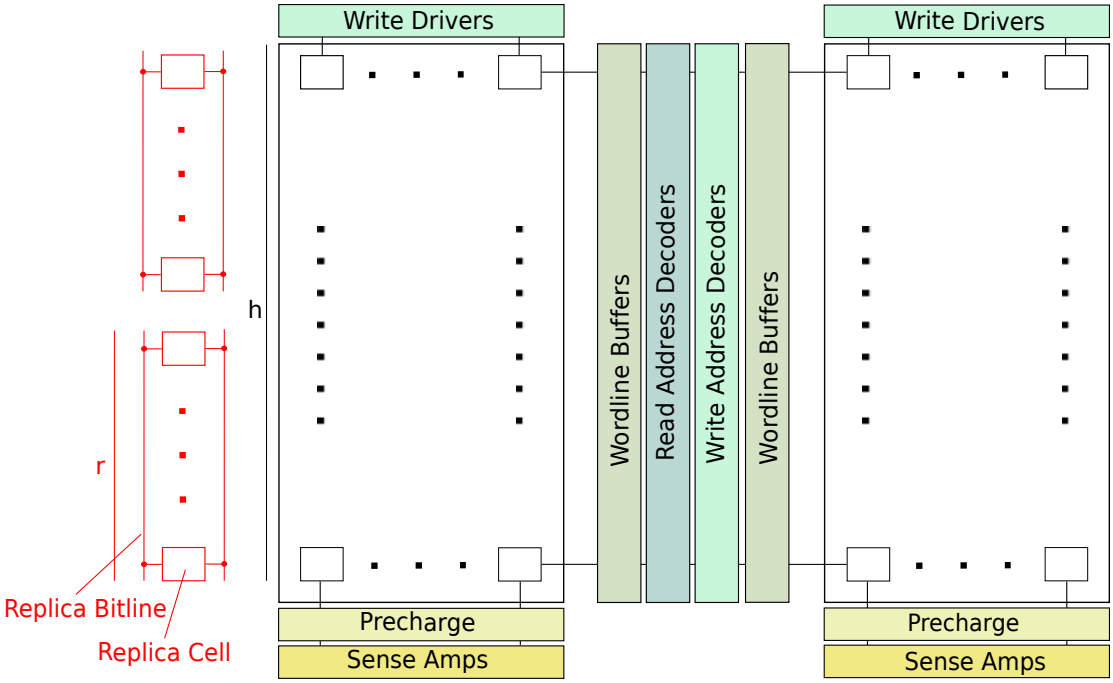


Figure 5.1: Replica-based SRAM [4]. *r* is the height of Replica cells that are active and *h* is the height of the SRAM core.

RBL technique is best suited for SAE timing generation because bitline delay tracking in presence of variation. Many Replica-based SRAM design and testing techniques have been proposed to maintain optimal timing for the SAE signal while keeping the SRAM error-proof during read operation. Viveka et al. proposed a post-silicon tunable testing approach using RBL to minimize margin for generating SAE [83]. The RBL signal can also be used to digitally tune the SA and WL signals in post-fabrication [62]. Arslan et al. proposed a cRBL, a reconfigurable RBL. Another post fabrication technique that uses statistical analysis to select a subset of bitcell drivers in the RBL column that best cancel the local transistor mismatches in memory cells [82].

Concept of Timing Speculation: Ernst et al. proposed RazorI which is a delay-error tolerant flip-flop for error detection in critical paths [21].

Razor eliminates the safety margins by achieving variable tolerance through in-situ timing error detection and correction. It uses a flip-flop and shadow latch to double sample the input. The main flip-flop is triggered at the positive edge of the clock whereas the shadow latch samples input data at the negative edge, so that the data is given 1/2 cycle to stabilize before being sampled by the shadow latch. Since the setup and hold time of the main flip-flop are allowed to be violated, a metastability detector is required at the output of the flip-flop. Razor uses a comparator to detect any discrepancy between the main and the speculative data. In case of error, the error flag is raised and OR'ed with the other "error" flags, and then propagated backwards so that the correct data stored on the shadow latch will be restored at the next clock cycle. Due to large number of logical OR's throughout the design, the "restore" signal can become a critical path itself. Costly correction is one of the reasons that RazorII was proposed later. RazorII performs error detection and leaves the data correction to the architectural replay schemes already present in processor pipelines [17]. Razor technique has been implemented in

many designs [43, 47] and it has been taped out in many chips [8, 22].

Timing Speculative SRAMs Karl et al. use the Razor technique to protect against erroneous reads by implementing a main and a shadow sense amplifier. We will compare against their design and refer to their solution as Razor. Similar to the Razor [21] technique, Karl et al. use a main and a shadow sense amplifier that double samples the bitline voltage swings. During a read operation, the two SAE signals are generated from the falling edge of the clock using an inverter delay chain. The delay chain is tapped to generate the precharge clock which is also used in the SA circuitry. First, the main SA is enabled and shortly after the shadow SA. A comparator is used to detect any discrepancy between inputs and generate an error signal. The generated "error" signal will select the shadow SA value at the output MUX [37]. The Razor will have a penalty associated with using Razor system which is a 1-cycle latency, but it will protect against data dependent delays during readings of the SRAM core. When using Razor, the read operation failures are detected and corrected, however, when there is a write operation, the lack of write failure detection will lead to an erroneous write and corrupt SRAM data unless the error is fixed before the wordline is enabled. For many SRAMs, the allotted time is less than half a cycle. A simple alternative is having a write scheme which is "failure" aware. This alternative is presented in section 5.3.

Another example of the timing speculative SRAM is the Domino Register File design by Kulkarni *et al.* [47], which double samples read output and its delayed version to detect timing errors within a clock window. Similar to Razor and Memory Timing Error Correction designs, it has an area overhead and the error detection mechanism can be defective due to metastability. Khayatzadeh *et al.* propose another Razor like SRAM that has been fabricated implementing Razor at the sense amplifier boundary for detecting read errors. While this is a

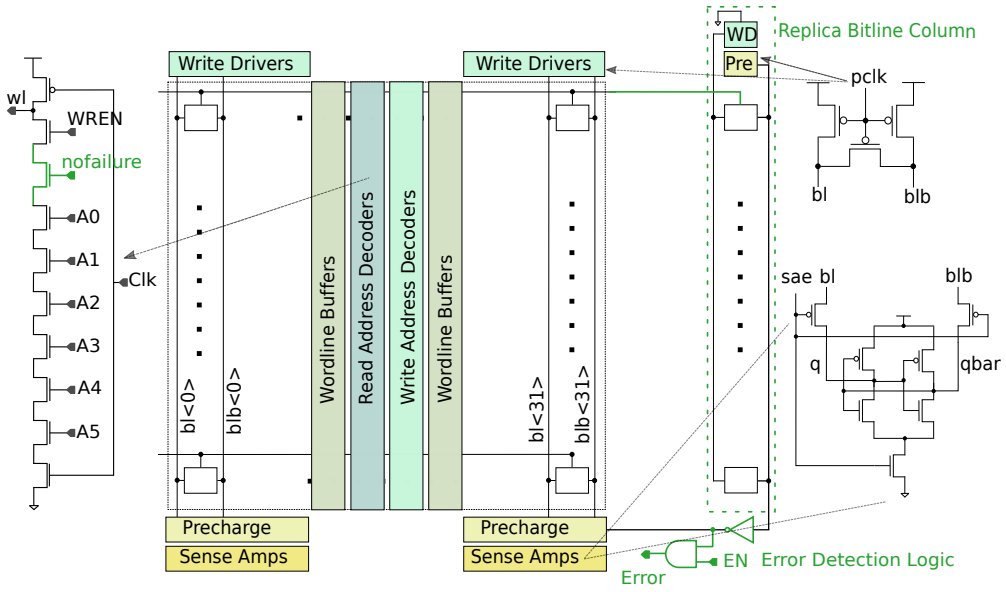


Figure 5.2: Timing Speculative SRAM

novel idea, the issue remains where razor is applied o per bit basis [43] and Razor'ed sense amplifier and the final logical "OR"s will an area overhead.

5.3 Time Speculative SRAM

Figure 5.2 shows a high-level block diagram of our proposed design which includes the enhancements to a generic 1Kb SRAM with two read ports and one write port. We refer to our Timing Speculative SRAM as RTS. The circuit enhancements and added logic are shown in color green *i.e.*, the added logic for generating the SAE, the write address decoder and the error detection logic. The following sections will include the detailed implementation of the new and modified components in the SRAM.

5.3.1 Protecting from Read Time Failures

To detect read time failures, RTS leverages the RBL technique. Replica Bitline Column (Figure 5.2) consists of a column of 6T bitcells that are identical to the bitcells used in the SRAM core. As opposed to adding dummy decoders, we utilize the decoders and wordlines to activate the RBL bitcells. Dummy write drivers produces a '0' bit which is written in all the RBL bitcells. During the precharge period, when clock is '0', the RBL bitlines will be precharged to '1'. When the clock signal transitions to '1', the wordline is raised, the reading begins. The SRAM core wordline (highlighted in Figure 5.2) is extended and connected to the wordline enable of the RBL bitcells. Therefore, the corresponding RBL bitcells will discharged mimicking the SRAM bitcell behavior. The time it takes to discharge the '1' in Replica cells is used to predict the SAE signal as it generates sufficient delay for enabling the sense amplifier. During the precharge period, when the clock signal is '0', the Replica bitcells are precharged to '1' and when the clock transitions to '1' and reading operation begins, the bitcells in the Replica column will begin discharging. The output of the Replica Bitline Column is connected to one large inverting buffer, shown in Figure 5.2, for 2 reasons. First, the RBL output signal needs to be inverted and able to drive all the 32 sense amplifiers and second, we need to compensate for the wordline enabling delay.

The error detection logic is shown in Figure 5.2. It includes another input signal besides RBL, named EN. for RTS. If during the period that EN is *on*, the RBL signal is also *on*, an error flag will be raised. The RBL signal already predicts the worst reading time delay for enabling the SAE. By using the RBL to generate SAE, we allow a window for detecting errors. During the read operation, when the clock is '1', SAE becomes active with some delay. If at

the second half of the negative edge of the clock, SAE signal is not yet lowered, the chance to read the data properly is missed and an error signal will be generated before the rising edge of the clock. The timing diagram in Figure 5.3 show when the error signal is generated.

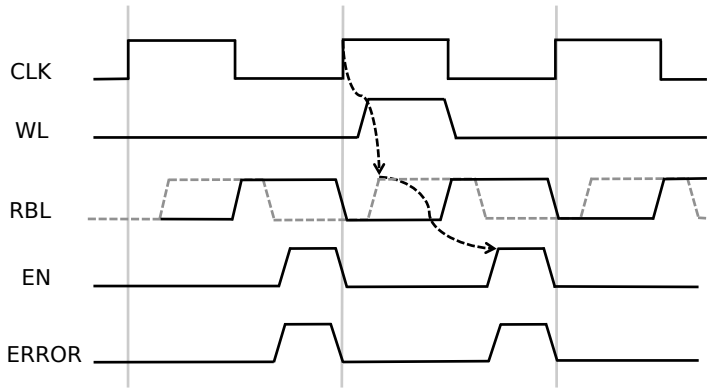


Figure 5.3: RTS error signal generation

RBL is a common technique in modern SRAM designs. Leveraging this existing technology reduces the overhead of area consuming and complex error detection mechanisms. RTS error detection overhead is insignificant regardless of the SRAM size. If the SRAM is large, the area overhead of large error detection mechanisms such as Razor might be acceptable, but in small SRAMs, the area overhead is significant in comparison with other components.

5.3.2 Protecting from Incorrect Writes

Previous state-of-the-art RAM [37] do not protect against incorrect inputs without requiring faster SRAMs. As a result, studies that use Razor-like solutions have assumptions similar to the Bubble Razor design [22], which state "Writes are clocked on the negative edge of the clock when data is guaranteed to be error free". Although writes tend to be faster than reads, requiring them to complete in half a SRAM cycle adds additional overhead. We propose

a minor modification to the write decoder to make it failure-aware.

We define an external failure to be corrupt data that causes the "error" or "failure" flag to be raised in the system. When SRAM is placed within a system that has an error detection mechanism, there exists an "error" or "failure" flag. To prevent an incorrect write, error has to be detected and the wordline disabled in less than half a cycle. However, when the system is equipped with error detecting latches/flops, the failure flag can be used to disable the wordline immediately. In order to protect an upcoming write in case of an external failure, RTS uses a decoder that has an active low failure input. When the write operation starts, at the rising edge of the clock, the address decoding takes place and wordline is selected. If the error flag along with the input data is raised, the wordline will be not be selected. Figure 5.2 shows the modification needed for the write address decoder used in RTS design. By adding a single NMOS transistor to the decoder with an active low failure signal input, the wordline is selected only if there is no failure present in the system. In RTS this external active low signal is called "nofail". The address bit NMOS transistors are connected in series with the "nofail" signal. When there is a failure, the WL will be deselected and the write operation will be discontinued.

5.3.3 Sense Amplifier

RTS uses latch-type sense amplifier (Figure 5.2), used in Amrutur's low power RBL SRAM [4]. It has a pair of cross coupled inverters which are turned on when SAE is active and an adequate input differential is set up. The SAE also enables the two PMOS transistors which are connected to the bitlines. Overall, the sense amplifier behaves similarly to a latch. Compared with other sense amplifiers, it consumes less energy. It will be slower than a current SA, but it will not be problematic as enough timing margin is provided in the generation of the

SAE signal [5].

5.4 Experiment Setup

There are 4 types of SRAMs built as shown in Table 5.2. Each type is designed for 3 different sizes listed in Table 5.1. Size *small* models a typical 1Kb register file for in-order processor. Size *medium* represents a 2Kb register file fit for a 3-way superscalar out-of-order processor, and size *large* models a large 8Kb single cache bank. Area of a size *large* SRAM is 15% more than a size *medium*.

Table 5.1: Experimental SRAM configurations model three typical processor SRAMs.

Size	Ports	Words	Word Width
Small	2r1w	32	32
Medium	6r3w	64	32
Large	1r1w	128	64

5.4.1 Tool Flow

All the SRAMs were designed and implemented using libraries available in FabMem, which is a subset of FabScalar toolset [14]. It uses FreePDK 45nm technology library and customized SRAM components to generate netlists and layouts for multiported SRAMs given the main configurations such as read and write ports, capacity, and word width. Table 5.2 explains each designed SRAM. Type Trad uses the FabScalar circuitry that FabMem provides. RBL builds on the Trad by adding an RBL support and a latch-type SA. RTS is a modified RBL

that incorporates the RBL, latch-type SA and a modified write address decoder. Razor is also a modified version of the Trad where the output sense amplifiers have been replaced with the Razor-enabled version.

Using FabMem, we built the Spice netlist for each of these SRAM types and configurations. The layouts for all of the modified RTS components have been drawn for a closer area approximation of the final SRAM design. All the introduced components have their own customized layout except the Razor-ed sense amplifier, however, we have estimated the area based on the used delay chains and control logic shown in Razor-enabled sense amplifier [37]. Its netlist excludes the extra logic for generating the enable signals (en1 and en2) and they are included as input data to the HSpice stimulus file for analysis.

Table 5.2: Experimental SRAM types

Type	Description
Trad	FabScalar proposed design.
RBL	FabScalar + RBL support + latch-type SA.
RTS	FabScalar + wr decoder + RBL + latch-type SA.
Razor	FabScalar RAM + Razor-ed SA.

5.5 Evaluation

To evaluate RTS, we run SPICE for all the SRAM types in Table 5.2 and compare read and write energy consumption, maximum frequency, and the SRAM area consumed against Razor.

5.5.1 Energy Efficiency

To compare the read and write energy consumption, we run size *small* with a frequency of 1GHz and sizes *medium* and *large* with a frequency of 700MHz. We run HSpice for a period of initialization, followed by 10 consecutive writes and 10 consecutive reads. For multiported structures, all read and write ports are active simultaneously during the read or write period respectively. Figures 5.4, 5.5, and 5.6 show the average energy breakdown for each SRAM type.

RTS and RBL have the RBL in common and, therefore, they consume similar energy levels during read or write operations. Considering Razor is reading and using the large sense amplifiers, it ends up consuming the largest amount of power. In order to have Razor operate faster, either the precharge devices have to be largely sized to elongate the precharge period or the buffer sizes that drive the bitlines to the comparator in the sense amplifier [37].

For small size SRAMs, Razor read energy consumption is 11.96x and 11.89x than RBL and RTS respectively. For medium size SRAMs, Razor consumes 8x more energy than both RBL and RTS. For size large, the read energy consumption for Razor is 3.37x and 3.30x more than RBL and RTS. As we increase the size of the SRAM the energy overhead becomes smaller, however, for typical/small sizes, the overhead is a considerable part of the design budget.

5.5.2 Area

Table 5.3 contains the area of the components used, added, or modified in all the SRAMs. In RBL the total area overhead is mainly due to the Replica Bitline Column and its dummy write driver and precharge circuits. Comparing RBL and Razor, For the size *small*

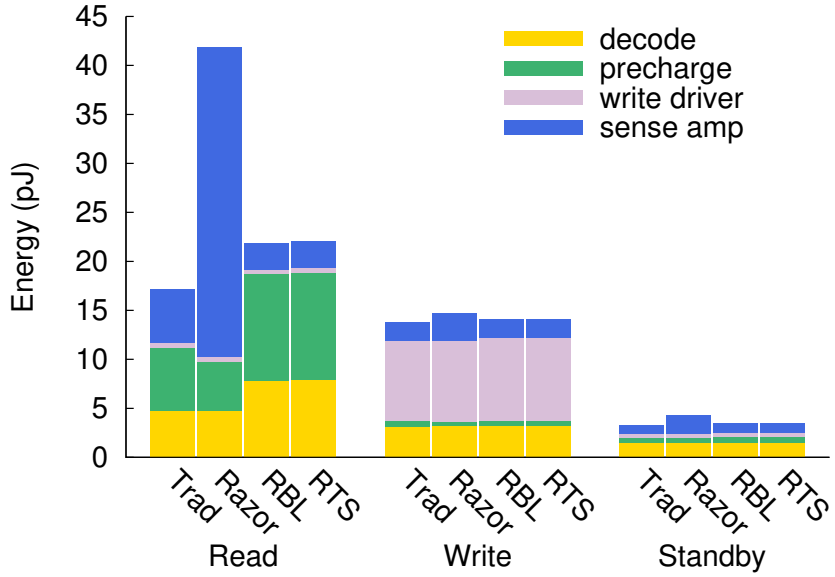


Figure 5.4: Energy breakdown in *small* SRAMs.

SRAM which we have used, RBL overhead is 111.7μm^2 and Razor overhead is 332.3μm^2. Razor has 3 times more area overhead than RBL. The area difference between RBL and RTS, is the error detection (*AND gates*) and the slightly larger write address decoders 3.37 vs. 3.18μm^2. The precharge circuit used for the RBL is identical to the precharge circuit used for the RAM core and to the rest of the SRAMs. The dummy write drivers are also the same as write drivers in all the other SRAMs.

Overall, Razor has the highest area overhead among all the SRAM types and among the error detecting SRAMs, RTS will have a small overhead. If the SRAM has the RBL implemented, the overhead of RTS would be negligible compared to the overall size of the SRAM.

Table 5.4 compares the percentage difference in total area overhead of RTS and Razor compared with RBL for 3 different sizes of SRAM. As the size increases, the overhead of RTS becomes negligible.

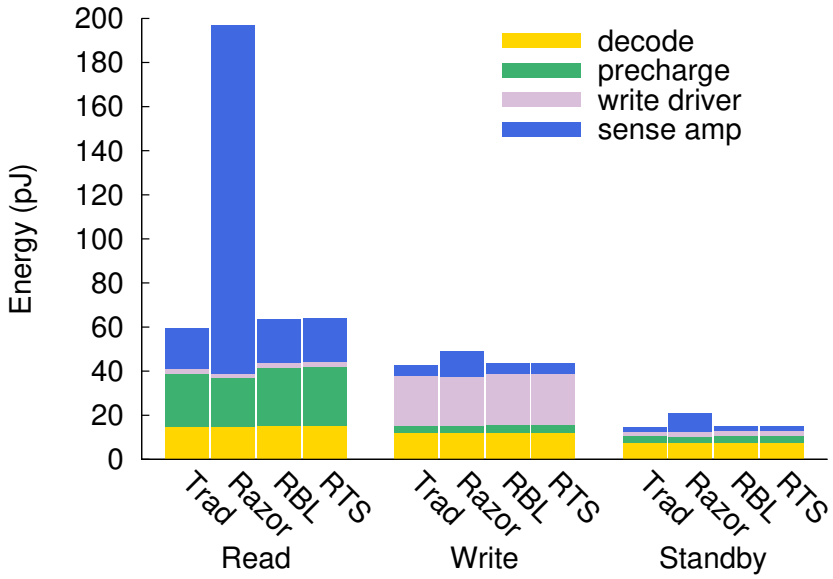


Figure 5.5: Energy breakdown in *medium* SRAMs.

Table 5.3: RTS overhead is mainly due to Replica Bitline Column. The Razor sense amplifier is 5.5x RBL bitcell area and 7.4x the traditional sense amplifier. Units are in μm^2.

Type	SA	RBL	Wr Dec	Xtra Buff	Xtra PreCharge	Xtra Wr Dri	Err logic
Trad	0.98	-	3.18	-	-	-	-
RBL	1.36	1.82	3.18	3.03	0.87	1.51	-
RTS	1.36	1.82	3.37	3.03	0.87	1.51	0.98
Razor	10	-	3.18	-	-	-	-

5.5.3 Process Variation Effects

We performed Monte Carlo simulations to analyze the frequency of the RBL and estimate the maximum clock period for performing reads. In order to perform Monte Carlo simulations and introduce variability, VARIUS is used [70]. VARIUS is a tool that given the

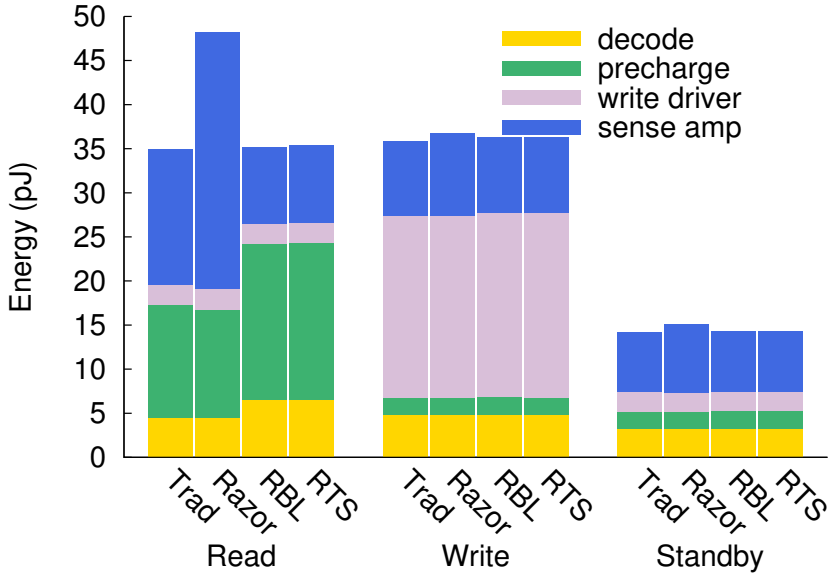


Figure 5.6: Energy breakdown in *large* SRAMs.

Table 5.4: The area overhead percentage difference of RTS and Razor with RBL.

Type	RTS	Razor
Small	0.43	5.46
Medium	0.16	2.45
Large	0.02	2.33

floorplan can create variation maps that considers both systematic and random variations. We generated a floorplan for RBL and run it with 50 variation maps generated by VARIUS. For each map, 10 different data points are read and the maximum output delay or the read speed is measured. Out of the 10 data points, the point with the maximum delay is chosen for that particular map.

Figure 5.7 shows the max clock period density function for the Small RBL which

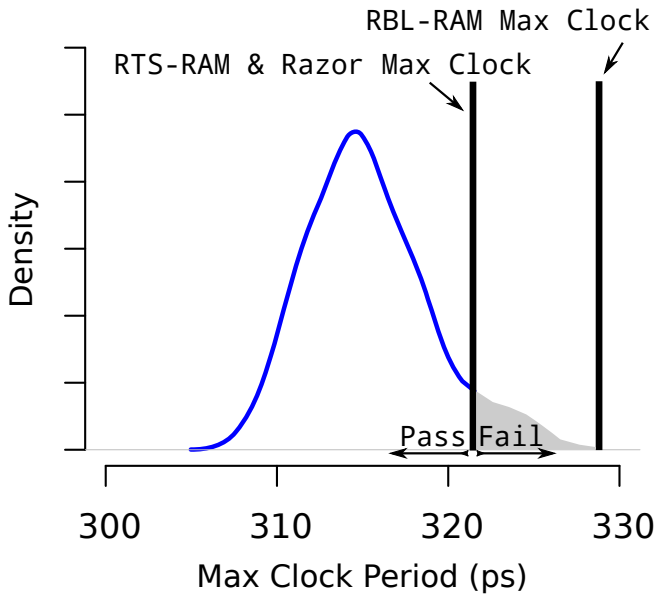


Figure 5.7: RBL and Razor achieve similar results, both allow protection from infrequent failures

corresponds to a typical Register File in simple in-order cores. To have high yield, the RBL should operate around 330ps. Since both RTS and Razor can protect against infrequent errors, it is possible to have a higher operating frequency. In the figure, it shows around 322ps. Both Razor and the proposed RTS can protect against errors.

Razor has a slow dynamic *OR* gate to detect failures. The figure uses a small SRAM, and the dynamic *OR* logic is not in the critical path, but SRAMs with larger number of read bits will affect the overall maximum frequency. This would not happen in RTS, because it uses the Replica Bitline to detect the errors. For simplicity, the Razor additional overhead is not included in Figure 5.7. Overall, RTS achieves the same operating frequency as Razor with area and the energy efficiency.

5.6 Conclusion

The final part of the dissertation includes the proposal for a new way to efficiently implement time speculation in SRAMs. RTS detects read and prevents write failures by offering a simple, yet efficient, solution that avoids costly shadow latches. Previous approaches like Razor required costly shadow latches and error detection logic. RTS leverages RBL to detect timing failures, and modifies the decoder logic to avoid requiring half cycle writes. This simple yet effective design shows to be 22% to 58% more energy efficient in reading operations. All this is achieved with a significant area reduction. The resulting RTS has less an error detection mechanism which is 35% to 73% more area efficient that Razor-enabled SRAM. RTS has 1% total area difference compared with traditional Replica-based RAM while Razor has up to 6% total area difference.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Throughout this thesis, I have addressed several applications of voltage stacking power delivery mechanism. To ease the design space exploration for level shifters, I have presented a detailed comparison of commonly used level shifters in terms of delay versus energy, and more importantly analyzing their correct functionality under voltage, temperature, and process variation. Once the optimal level shifter is determined, it is then applied to a circuit that inherently helps balancing the loads in different voltage stacks. The results shows that the same amount of power is consumed, but the current is cut by a the number of stack levels.

The second application of voltage stacking aims to compensate for process variation in the NTC region in GPGPUs. NTC has the potential to significantly reduce power in GPGPUs, but it makes the circuitry more susceptible to process variation and complicates PDN design. I have proposed a novel method, GPU Stacking, to concurrently manage the effects of variation and improve the power delivery.

I leveraged the logic used in the SRAM architecture to divide it into stacks that allow power distribution evenly. As the final contribution of this thesis, I propose to further leverage the RBL technique widely used in modern SRAM designs to detect timing errors and trim the voltage guardbands.

6.2 Future Work

SRAMs are prevalent in today's industrial and academic SoC designs, and voltage stacking is becoming a more common power management technique used in many designs [7, 20, 52, 88]. Chapter 3 presents how an SRAM is stacked. To extend this evaluation, the small sized SRAM can be replaced by a larger sized SRAMs, for instance, similarly sized to typical cache level sizes. As the size increases, so do the number of level shifters. Area overhead should be less of a concern as the SRAM size itself is large, however, the Pareto frontier curves in Chapter 2 will help the designer choose the optimal level shifter based on required timing and/or area budget. Another evaluation which adds more insights into the concept of voltage stacking is having more than 2 voltage levels. As the memory size increases, the designers might need more than 2 voltage levels.

Additionally, the voltage stacking and timing speculation techniques could be incorporated in an open source memory compiler such as FabMEM [14] and OpenRAM [65] in the future. It would make them applicable to larger size memories, and more importantly, it would make them more portable and accessible especially for academic research. FabMEM generates layout for different SRAM configurations, however, for stacked SRAM, it could only be used if the layout generation tool was modified. OpenRAM is a collaborative project designed and

implemented by VLSI groups from UC Santa Cruz and Oklahoma State University group. It is a Python-based tool which uses SRAM configurations to generate netlists and layouts. It uses the RBL technique to generate the SAE, omnipresent in many current SRAM designs. It provides the opportunity to integrate the timing speculation technique in RTS with the toolset. Since OpenRAM allows the designers to do power analysis, it is a critical step to ensure that the power constraints are met properly and to avoid any later design complications. It would be beneficial for the researchers to be able to apply an alternative power delivery method such as voltage stacking and have quick and yet more deterministic results for energy consumption of their desired memory subsystem.

The next step for researcher working on voltage stacking technique would be to fabricate a chip that uses this methodology in power design, and be able to conduct off the chip experiments and concretely validate that such power delivery mechanism is a definite alternative for chip designers.

Bibliography

[1] NVIDIA GeForce GTX750 specifications. Available on `http://www.geforce.com/hardware/desktop-gpus/geforce-gtx-750/specifications`.

[2] Zhenyu Qi Adam C. Cabe and Mircea R. Stan. Stacking SRAM Banks for Ultra Low Power Standby Mode Operation. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 699–704.

[3] Bhawna Aggarwal, Maneesha Gupta, and A.K. Gupta. A Comparative Study of Various Current Mirror Configurations: Topologies and Characteristics. *Microelectronics Journal*, 53:134–155, 2016.

[4] Bharadwaj S. Amrutur and Mark A. Horowitz. A Replica Technique for Wordline and Sense Control in Low-Power SRAM's. *IEEE Journal of Solid-State Circuits*, 33:1208–1219, August 1998.

[5] Bharadwaj S. Amrutur and Mark A. Horowitz. Speed and Power Scaling of SRAM's. *IEEE Journal of Solid-State Circuits*, 35(2):175–185, February 2000.

[6] Christiensen D.C. Arandilla and Joy Alinda R. Madamba. Comparison of Replica Bitline Technique and Chain Delay Technique as Read Timing Control for Low-Power Asyn-

chronous SRAM. In *Modelling Symposium (AMS), 2011 Fifth Asia*, pages 275–278, May 2011.

[7] Ehsan K. Ardestani, Rafael Trapani Possignolo, Jose Luis Briz, and Jose Renau. Managing mismatches in voltage stacking with coreUnfolding. *ACM Trans. Archit. Code Optim.*, 12(4):43:1–43:26, Nov. 2015.

[8] David Bull, Shidhartha Das, Karthik Shivashankar, Ganesh S. Dasika, Krisztian Flautner, and David Blaauw. A Power-Efficient 32-Bit ARM Processor Using Timing-Error Detection and Correction for Transient-Error Tolerance and Adaptation to PVT Variation. *Solid-State Circuits, IEEE Journal of*, 46(1):18–31, January 2011.

[9] Leland Chang, David J. Frank, Robert K. Montoye, Steven J. Koester, Brian L. Ji, Paul W. Coteus, Robert H. Dennard, and Wilfried Haensch. Practical Strategies for Power-Efficient Computing Technologies. *Proceedings of the IEEE*, 98(2):215–236, 2010.

[10] Leland Chang, Robert K. Montoye, Brian L. Ji, Alan J. Weger, Kevin G. Stawiasz, and Robert H. Dennard. A Fully-Integrated Switched-Capacitor 2:1 Voltage Converter with Regulation Capability and 90% Efficiency at $2.3A/mm^2$. In *VLSI Circuits (VLSIC), 2010 IEEE Symposium on*, pages 55–56, June 2010.

[11] Shuai Che, Michael Boyer, Jiayuan Meng, David Tarjan, Jeremy W. Sheaffer, Sang-Ha Lee, and Kevin Skadron. Rodinia: A Benchmark Suite for Heterogeneous Computing. In *Proceedings of the 2009 IEEE International Symposium on Workload Characterization (IISWC)*, IISWC '09, pages 44–54, Washington, DC, USA, 2009. IEEE Computer Society.

[12] Tom Chen and Samuel Naffziger. Comparison of Adaptive Body Bias (ABB) and Adaptive

Supply Voltage (ASV) for Improving Delay and Leakage Under the Presence of Process Variation. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 11(5):888–899, 2003.

[13] Jin-Hyuck Choi, Jung-Hoon Lee, Seh-Woong Jeong, Shin-Dug Kim, and Charles Weems. A Low Power TLB Structure for Embedded Systems. *IEEE Computer Architecture Letters*, pages 3–3, 2002.

[14] N. K. Choudhary, S. V. Wadhavkar, Tanmay A. Shah, Hiran Mayukh, Jayneel Gandhi, Brandon H. Dwiel, Sandeep Navada, Hashem H. Najaf-abadi, and Eric Rotenberg. FabScalar: composing synthesizable RTL designs of arbitrary cores within a canonical superscalar template. In *Proc. of the 38th Int'l Symp. on Computer Architecture*, pages 11–22, New York, NY, USA, 2011. ACM.

[15] Jason Cong, Mohammad Ali Ghodrat, Michael Gill, Beayna Grigorian, and Glenn Reinman. CHARM: a Composable Heterogeneous Accelerator-Rich Microprocessor. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, ISLPED '12, pages 379–384, 2012.

[16] Intel Corporation. A 400 amp fully integrated silicon voltage regulator with in-die magnetically coupled embedded inductors, 2010.

[17] Shidhartha Das, Carlos Tokunaga, Sanjay Pant, Wei-Hsiang Ma, Kalaiselvan Sudherssen, Kevin Lai, David M. Bull, and David T. Blaauw. RazorII: In Situ Error Detection and Correction for PVT and SER Tolerance. *Solid-State Circuits, IEEE Journal of*, 44(1):32–48, Jan. 2009.

[18] Eric Donkoh, Teck Siong Ong, Yan Nee Too, and Patrick Chiang. Register File Write Data Gating Techniques and Break-even Analysis Model. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, ISLPED '12, pages 149–154, 2012.

[19] Ronald G. Dreslinski, Michael Wieckowski, David Blaauw, Dennis Sylvester, and Trevor Mudge. Near-threshold computing: Reclaiming moores law through energy efficient integrated circuits. 98(2):253–266, Feb 2010.

[20] Elnaz Ebrahimi, Rafael Trapani Possignolo, and Jose Renau. SRAM Voltage Stacking. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1634–1637, May 2016.

[21] D. Ernst, Nam Sung Kim, S. Das, S. Pant, R. Rao, Toan Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: a low-power pipeline based on circuit-level timing speculation. In *Proceedings of the 36th Int'l Symp. on Microarchitecture (MICRO-36)*, pages 7 – 18, Dec. 2003.

[22] Matthew Fojtik, David Fick, Yejoong Kim, Nathaniel Pinckney, David Harris, David Blaauw, and Dennis Sylvester. Bubble Razor: An Architecture-Independent Approach to Timing-Error Detection and Correction. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pages 488 –490, Feb. 2012.

[23] Mark Gebhart, Daniel R. Johnson, David Tarjan, Stephen W. Keckler, William J. Dally, Eric Lindholm, and Kevin Skadron. A Hierarchical Thread Scheduler and Register File

for Energy-Efficient Throughput Processors. *ACM Trans. Comput. Syst.*, 30(2):8:1–8:38, Apr. 2012.

[24] Mark A. Gebhart. *Energy-Efficient Mechanisms for Managing On-Chip Storage in Throughput Processors*. PhD thesis, The University of Texas at Austin, May. 2012.

[25] Jie Gu and Chris H. Kim. Multi-story Power Delivery for Supply Noise Reduction and Low Voltage Operation. In *Proceedings of the 2005 International Symposium on Low Power Electronics and Design*, ISLPED '05, pages 192–197, New York, NY, USA, 2005. ACM.

[26] Amir Hasanbegovic and Snorre Aunet. Low-power Subthreshold to Above Threshold Level Shifters in 90Nm and 65Nm Process. volume 35, pages 1–9, Amsterdam, The Netherlands, The Netherlands, Feb. 2011. Elsevier Science Publishers B. V.

[27] Sunpyo Hong and Hyesoon Kim. An integrated GPU power and performance model. *SIGARCH Comput. Archit. News*, 38(3):280–289, Jun. 2010.

[28] M. Huang, J. Renau, S.-M. Yoo, and J. Torrellas. Cache decomposition for energy-efficient processors. In *International Symposium on Low Power Electronics and Design*, Aug 2001.

[29] Michael Huang, Jose Renau, Seung-Moon Yoo, and Josep Torrellas. A Framework for Dynamic Energy Efficiency and Temperature Management. In *Proceedings of the 33rd Annual ACM/IEEE International Symposium on Microarchitecture*, MICRO 33, pages 202–213, New York, NY, USA, 2000. ACM.

[30] Canturk Isci, Alper Buyuktosunoglu, Chen-Yong Cher, Pradip Bose, and Margaret Martonosi. An analysis of efficient multi-core global power management policies: Max-

imizing performance for a given power budget. In *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 39, pages 347–358, 2006.

[31] Renatas Jakushokas, Mikhail Popovich, Andrey V. Mezhiba, Seluk Kse, and Eby G. Friedman. *Power Distribution Networks with On-Chip Decoupling Capacitors*. Springer Publishing Company, Incorporated, 2nd edition, 2010.

[32] Tejas S. Joshi and Priya M. Ravale Nerkar. A Wide Range Level Shifter using a Self Biased Cascode Current Mirror With PTL-based Buffer. *IJCA Proceedings on National Conference on Emerging Trends in Advanced Communication Technologies*, NCETACT 2015(5):8–12, June 2015.

[33] Ehsan K. Ardestani, Elnaz Ebrahimi, Gabriel Southern, and Jose Renau. Thermal-aware Sampling in Architectural Simulation. In *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '12, pages 33–38, New York, NY, USA, 2012. ACM.

[34] Ehsan K. Ardestani, Francisco J. Mesa-Martinez, Gabriel Southern, Elnaz Ebrahimi, and Jose Renau. Sampling in thermal simulation of processors: Measurement, characterization, and evaluation. *Transaction on Computer Aided Design of Integrated Circuits and Systems (TCAD)*, 2013.

[35] Ehsan K. Ardestani and Jose Renau. ESESC: A Fast Multicore Simulator Using Time-Based Sampling. In *International Symposium on High Performance Computer Architecture*, HPCA'19, 2013.

[36] Svilen Kanev. *Motivating Software-Driven Current Balancing in Flexible Voltage-Stacked Multicore Processors*. PhD thesis, Harvard University Cambridge, Massachusetts, 2012.

[37] Eric Karl, Dennis Sylvester, and D. Blaauw. Timing Error Correction Techniques for Voltage-Scalable On-Chip Memories. In *Circuits and Systems (ISCAS), 2005 IEEE International Symposium on*, volume 4, pages 3563–3566, May 2005.

[38] Ulya R. Karpuzcu, Nam Sung Kim, and Josep Torrellas. Coping with Parametric Variation at Near-Threshold Voltages. *IEEE Micro*, 33(4):6–14, July 2013.

[39] Ulya R. Karpuzcu, Krishna B. Kolluru, Nam Sung Kim, and Josep Torrellas. VARIUS-NTV: A microarchitectural model to capture the increased sensitivity of manycores to process variations at near-threshold voltages. In *Dependable Systems and Networks (DSN), 2012 42nd Annual IEEE/IFIP International Conference on*, pages 1–11. IEEE, 2012.

[40] Ulya R. Karpuzcu, Abhishek Sinkar, Nam Sung Kim, and Torrellas Josep. EnergySmart: Toward Energy-Efficient Many cores for Near-Threshold Computing. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 542–553. IEEE Computer Society, Feb 2013.

[41] Sourabh Khandelwal, Juan Duarte, Navid Paydavosi, Darsen Lu, Chung-Hsun Lin, Mohan, Shijing Yao, Tanvir Morshed, Ali Niknejad, and Chenming Hu. BSIM-CMG 108.0.0 Multi-Gate MOSFET Compact Model. 2014.

[42] S. Karen Khatamifard, Michael Resch, Nam Sung Kim, and Ulya R. Karpuzcu. VARIUS-TC: A Modular Architecture-Level Model of Parametric Variation for Thin-Channel

Switches. *2016 IEEE 34th International Conference on Computer Design (ICCD)*, 00:654–661, 2016.

[43] Mahmood Khayatzadeh, Mehdi Saligane, Jingcheng Wang, Massimo Alioto, David Blaauw, and Dennis Sylvester. 17.3 A Reconfigurable Dual-Port Memory with Error Detection and Correction in 28nm FDSOI. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 310–312, Jan 2016.

[44] Wonyoung Kim, David Brooks, and Gu-Yeon Wei. A fully-integrated 3-level dc/dc converter for nanosecond-scale dvs with fast shunt regulation. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, pages 268–270. IEEE, 2011.

[45] Wonyoung Kim, Meeta S Gupta, Gu-Yeon Wei, and David Brooks. System Level Analysis of Fast, per-Core DVFS Using on-Chip Switching Regulators. In *2008 IEEE 14th International Symposium on High Performance Computer Architecture*, pages 123–134, Feb 2008.

[46] Kyoung-Hoi Koo, Jin-Ho Seo, Myeong-Lyong Ko, and Jae-Whui Kim. A new level-up shifter for high speed and wide range interface in ultra deep sub-micron. In *2005 IEEE International Symposium on Circuits and Systems*, pages 1063–1065 Vol. 2, May 2005.

[47] J. P. Kulkarni, C. Tokunaga, P. A. Aseron, T. Nguyen, C. Augustine, J. W. Tschanz, and V. De. A 409 GOPS/W Adaptive and Resilient Domino Register File in 22 nm Tri-Gate CMOS Featuring In-Situ Timing Margin and Error Detection for Tolerance to Within-Die

Variation, Voltage Droop, Temperature and Aging. *IEEE Journal of Solid-State Circuits*, 51(1):117–129, 2016.

[48] Manoj Kumar, Sandeep K. Arya, and Sujata Pandey. Level Shifter Design for Low Power Applications. *Computer Science & Information Technology, International Journal of*, 2(5), Oct. 2010.

[49] Volkan Kursun and Eby G Friedman. *Multi-voltage CMOS circuit design*. John Wiley & Sons, 2006.

[50] Jungseob Lee, Paritosh P. Ajgaonkar, and Nam Sung Kim. Analyzing throughput of GPGPUs exploiting within-die core-to-core frequency variation. In *Performance Analysis of Systems and Software (ISPASS), 2011 IEEE International Symposium on*, pages 237–246. IEEE, 2011.

[51] Jungseob Lee, Vijay Sathisha, Michael Schulte, Katherine Compton, and Nam Sung Kim. Improving throughput of power-constrained gpus using dynamic voltage/frequency and core scaling. In *Proceedings of the 2011 International Conference on Parallel Architectures and Compilation Techniques*, PACT '11, pages 111–120, 2011.

[52] Sae Kyu Lee, David Brooks, and Gu-Yeon Wei. Evaluation of Voltage Stacking for Near-threshold Multicore Computing. In *Proceedings of the 2012 ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '12, pages 373–378, New York, NY, USA, 2012. ACM.

[53] Sae Kyu Lee, Tao Tong, Xuan Zhang, D. Brooks, and Gu-Yeon Wei. A 16-Core Voltage-

Stacked System with an Integrated Switched-Capacitor DC-DC Converter. In *VLSI Circuits (VLSI Circuits), 2015 Symposium on*, pages C318–C319, June 2015.

[54] Jingwen Leng, Yazhou Zu, and Vijay Janapa Reddi. GPU Voltage Noise: Characterization and Hierarchical Smoothing of Spatial and Temporal Voltage Noise Interference in GPU Architectures. In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pages 161–173, Feb 2015.

[55] Jingwen Leng, Yazhou Zu, Minsoo Rhu, Meeta Gupta, and Vijay Janapa Reddi. Gpuvolt: Modeling and characterizing voltage noise in gpu architectures. In *Proceedings of the 2014 International Symposium on Low Power Electronics and Design*, ISLPED '14, pages 141–146, New York, NY, USA, 2014. ACM.

[56] Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Microarchitecture. 42nd IEEE/ACM Int'l Symp. on*, pages 469–480. IEEE, 2009.

[57] Jan Lucas, Sohan Lal, Michael Andersch, Alvarez-Mesa Mauricio, and Ben Juurlink. How a Single Chip Causes Massive Power Bills GPUSimPow: A GPGPU Power Simulator. In *IEEE International Symposium on Performance Analysis of Systems and Software*, 2013.

[58] Shien-Chun Luo, Ching-Ji Huang, and Yuan-Hua Chu. A Wide-Range Level Shifter Using a Modified Wilson Current Mirror Hybrid Buffer. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(6):1656–1665, June 2014.

[59] Michael J. Lyons and David Brooks. The design of a bloom filter hardware accelerator for

ultra low power systems. In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design*, ISLPED '09, pages 371–376, 2009.

[60] Baker Mohammad, Percy Dadabhoy, Ken Lin, and Paul Bassett. Comparative study of current mode and voltage mode Sense Amplifier used for 28nm SRAM. In *Microelectronics (ICM), 2012 24th International Conference on*, pages 1–6, Dec. 2012.

[61] Sani R. Nassif. Power grid analysis benchmarks. In *Design Automation Conference, 2008. ASPDAC 2008. Asia and South Pacific*, pages 376–381, 2008.

[62] Adam Neale and Manoj Sachdev. Digitally Programmable SRAM Timing for Nano-scale Technologies. In *Quality Electronic Design (ISQED), 2011 12th International Symposium on*, pages 1–7, March 2011.

[63] K.R. Pasupathy and Boby Bindu. Low Power, High Speed Carbon Nanotube FET Based Level Shifters for multi-VDD Systems-On-Chips. *Microelectron. J.*, 46(12):1269–1274, Dec. 2015.

[64] H. Pilo, I. Arsovski, K. Batson, G. Braceras, J. Gabric, R. Houle, S. Lamphier, F. Pavlik, A. Seferagic, Liang-Yu Chen, Shang-Bin Ko, and C. Radens. A 64Mb SRAM in 32nm High-k Metal-gate SOI Technology with 0.7V Operation Enabled by Stability, Write-ability and Read-ability Enhancements. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2011 IEEE International*, pages 254–256, Feb 2011.

[65] Matthew R. Guthaus, James E. Stein, Samira Ataei, Brian Chen, Bin Wu, and Mehedi Sarwar. OpenRAM: An Open-Source Memory Compiler. In *Proceedings of the 35th International Conference on Computer-Aided Design*, page 93. ACM, 2016.

[66] Saravanan Rajapandian, Kenneth L. Shepard, Peter Hazucha, and Tanay Karnik. High-voltage Power Delivery Through Charge Recycling. *Solid-State Circuits, IEEE Journal of*, 41(6):1400–1410, 2006.

[67] Gabriel A Rincon-Mora and Phillip E Allen. A low-voltage, low quiescent current, low drop-out regulator. *IEEE journal of Solid-State circuits*, 33(1):36–44, 1998.

[68] Bharadwaj S.Amrutur. Design and Analysis of Fast Low Power SRAMs. Technical report, Stanford, CA, USA, 2000.

[69] Alamelu Sankaranarayanan, Ehsan K. Ardestani, Jose Luis Briz, and Jose Renau. An energy efficient gpgpu memory hierarchy with tiny incoherent caches. In *International Symposium on Low-Power Electronics and Design*, Beijing, China, Sept. 2013.

[70] Sarangi R. Sarangi, Brian Greskamp, Radu Teodorescu, Jun Nakano, Abhishek Tiwari, and Josep Torrellas. VARIUS: A model of process variation and resulting timing errors for microarchitects. *Semiconductor Manufacturing, IEEE Transactions on*, 21(1):3–13, 2008.

[71] Tanmay Shah. FabMem: A Multiported RAM and CAM Compiler for Superscalar Design Space Exploration. Master's thesis, North Carolina State University, 2010.

[72] E. Sperling. Power Delivery Network Challenges Grow, 2010.

[73] J. E. Stine, J. Chen, I. Castellanos, G. Sundararajan, M. Qayam, P. Kumar, J. Remington, and S. Sohoni. FreePDK v2.0: Transitioning VLSI education towards nanometer variation-aware designs. In *Microelectronic Systems Education, 2009. MSE '09. IEEE International Conference on*, pages 100–103, July 2009.

[74] J.A. Stratton, C. Rodrigues, I.J. Sung, N. Obeid, L.W. Chang, N. Anssari, G.D. Liu, and W.W. Hwu. Parboil: A Revised Benchmark Suite for Scientific and Commercial Throughput Computing. *Center for Reliable and High-Performance Computing*, 2012.

[75] Infineon Technologies. High performance DrMos TDA21220, 2013. Available on `http://www.infineon.com/cms/en/product/power/dc-dc-converter/dc-dc-integrated-power-stage/drmos-integrated-power-stage/TDA21220/productType.html?productType=db3a3044243b532e0124de3165386adc`.

[76] Renji Thomas, Kristin Barber, Naser Sedaghati, Li Zhou, and Radu Teodorescu. Core tunneling: Variation-aware voltage noise mitigation in gpus. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 151–162, March 2016.

[77] Tao Tong, Sae Kyu. Lee, Xuan Zhang, David Brooks, and Gu-Yeon Wei. A Fully Integrated Reconfigurable Switched-Capacitor DC-DC Converter With Four Stacked Output Channels for Voltage Stacking Applications. *IEEE Journal of Solid-State Circuits*, 51(9):2142–2152, Sept 2016.

[78] Rafael Trapani Possignolo, Elnaz Ebrahimi, and Jose Renau. Recycling of elastic systems without throughput penalty, 2015. Work-in-Progress Poster presented at the 52th Annual Design Automation Conference, DAC'15, June 7–11, San Francisco, CA.

[79] Rafael Trapani Possignolo, Elnaz Ebrahimi, Haven Skinner, and Jose Renau. Fluid-pipelines: Elastic circuitry meets out-of-order execution.

[80] Rafael Trapani Possignolo, Elnaz Ebrahimi, Haven Skinner, and Jose Renau. Fluid-Pipelines: Elastic circuitry without throughput penalty. In *Logic Synthesis (IWLS), Proceedings of the 2016 International Workshop on*, Jun 2016.

[81] J.T. Trattles, Anthony G. O'NEILL, and Barrie C. Mecrow. Three-dimensional finite-element investigation of current crowding and peak temperatures in VLSI multilevel interconnections. *IEEE transactions on electron devices*, 40(7):1344–1347, 1993.

[82] Mudit Bhargava Xin Li Ken Mai Umut Arslan, Mark P. McCartney and Lawrence T. Pileggi. Variation-tolerant SRAM Sense-amplifier Timing Using Configurable Replica Bitlines. In *Custom Integrated Circuits Conference, 2008. CICC 2008. IEEE*, pages 415–418, September 2008.

[83] K.R. Viveka and Bharadwaj S. Amrutur. Digitally Controlled Variation Tolerant Timing Generation Technique for SRAM Sense Amplifiers. In *Quality Electronic Design (ASQED), 2013 5th Asia Symposium on*, pages 233–239, August 2013.

[84] H Clement Wann, Chenming Hu, Kenji Noda, Dennis Sinitsky, Fariborz Assaderaghi, and Jeff Bokor. Channel doping engineering of mosfet with adaptable threshold voltage using body effect for low voltage and low power applications.

[85] Yang Yang and Niraj K. Jha. FinPrin: FinFET Logic Circuit Analysis and Optimization Under PVT Variations. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(12):2462–2475, Dec 2014.

[86] Bo Zhai, David Blaauw, Dennis Sylvester, and Krisztian Flautner. Theoretical and Prac-

tical Limits of Dynamic Voltage Scaling. In *Proceedings of the 41st Annual Design Automation Conference*, DAC '04, pages 868–873, New York, NY, USA, 2004.

[87] Yong Zhan and Sachin S Sapatnekar. Automated module assignment in stacked-vdd designs for high-efficiency power delivery. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 4(4):18, 2008.

[88] Qixiang Zhang, Liangzhen Lai, Mark Gottscho, and Puneet Gupta. Multi-story power distribution networks for GPUs. In *Design, Automation, and Test in Europe (DATE), Proceedings of*, Mar 2016.

[89] Jishen Zhao, Guangyu Sun, Gabriel H. Loh, and Yuan Xie. Energy-efficient gpu design with reconfigurable in-package graphics memory. In *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, ISLPED '12, pages 403–408, 2012.

[90] Wei Zhao and Yu Cao. New generation of predictive technology model for sub-45nm early design exploration. *Electron Devices, IEEE Transactions on*, 53(11):2816–2823, 2006.

[91] Jun Zhou, Chao Wang, Xin Liu, and Minkyu Je. Fast and Energy-efficient Low-voltage Level Shifters. *Microelectron. J.*, 46(1):75–80, Jan. 2015.