AN ABSTRACT OF THE DISSERTATION OF

Eric Kwesi Donkoh for the degree of Doctor of Philosophy in Electrical and Computer Engineering presented on June 13, 2013.

Title: Design and Modeling of Low-Power Register File Memories

Abstract approved: _____

Patrick Y. Chiang

With the evolving popularity of new computing platforms such as Ultrabooks, Tablets, and Smart Phones, and the shift to multi-core computing, power is now the key performance limiter, a departure from the traditional frequency limitation. As such, increasingly low-power design solutions feature prominently in early architectural and design space exploration in CPU/SoC design. On a high performance CPU, majority of these early studies involve memories, especially Register Files. Register Files (RF) are the preferred memory element for fast data access and are therefore ubiquitous in modern microprocessor design, contributing approximately 30% of Intel's 32nm CPU core power.

The goal of this research is two-fold. First, it explores low-power design techniques to reduce RF leakage and dynamic power at minimal delay and area cost. We analyze RF power distribution, data residencies, signal activities, and logic dependencies in modern 32nm/22nm high performance microprocessors. We then propose new circuit techniques to reduce power in critical memory logic blocks such as the bitcell, write data distribution, read access data path, and decoder. We use innovative transistor stack-forcing techniques to reduce RF read bitline and decoder leakage by as much as 90% and delay by 30% at minimal to no area overhead compared to existing stacking approaches.

An essential component of low-power design is an accurate predictive model (power, area, and timing) for early architectural and design space tradeoff analysis. On a high performance CPU, greater than 75% of RFs are custom designed due to design complexities and constraints (power, area, timing, low-voltage operation requirements). Existing models are particularly unsuited for custom RF because these models typically assume a generic RF circuit implementation and are therefore inaccurate for predicting unique RF topologies without

requiring new model development. Furthermore, these models do not accurately address common design optimizations such as device sizing, data gating, segmentation, and device stacking that significantly impact the power profile of an RF. In the second part of this research we've developed a customizable predictive model that addresses these key limitations. The proposed model is a hybrid of empirical reference data and analytical equations. We use an empirical reference implementation data of a topology under study to capture topology-specific characteristics and analytically model the impact of cross-topology features such as changes in bit-width, entry-count, ports, and common circuit-level design optimizations such as segmentation, gating, device stacking, and sizing. We show how the proposed model can be customized for different RF topologies and other memory structures such as SRAM and ROM using the same model equations. We also demonstrate how the new predictive model, with <10% error, is used in the real world tradeoff analysis in the design of state-of-the-art high performance CPUs and SoCs.

DESIGN AND MODELING OF LOW-POWER REGISTER FILE MEMORIES

by
Eric Kwesi Donkoh

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 13, 2013
Commencement June 2014

Doctor of Philosophy Dissertation of Eric Kwesi Donkoh presented on June 13, 2013

APPROVED:

Major Professor, representing Electrical and Computer Engineering

Director of the School of Electrical Engineering and Computer Science

Dean of the Graduate School

I understand that my dissertation will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my dissertation to any reader upon request.

Eric Kwesi Donkoh, Author

ACKNOWLEDGEMENTS

I will like to express my appreciation to Prof. Patrick Chiang for his flexibility towards the selection of research topic and his guidance towards this work. I would also like to express my gratitude to my committee members, Shih-Lien Lu, Bibiche Geuskens, Prof. Arun Natarajan, and Prof Abi Farsoni for expending their valuable time in guiding my PhD program.

Great thanks to my co-workers Alicia Lower, Teck-Siong Ong, Yan Nee Too, Emily Shriver, and Ee Wah Lim for their support and contribution to various aspects of this research. An appreciation to co-worker Ataur Patwary for sharing his valuable OSU experience. I would also like to thank Nanda Siddaiah and Kurt Kreitzer for their support of this work.

My sincerely thanks to the EECS graduate coordinators Nicole Thompson and Ferne Simendinger for their invaluable assistance in navigating the PhD process.

Finally, a big thanks to all those who directly or indirectly impacted this work.

This Dissertation is dedicated to Family and Friends

No One Has A Monopoly On Ideas – Be Innovative!!!

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF TABLES

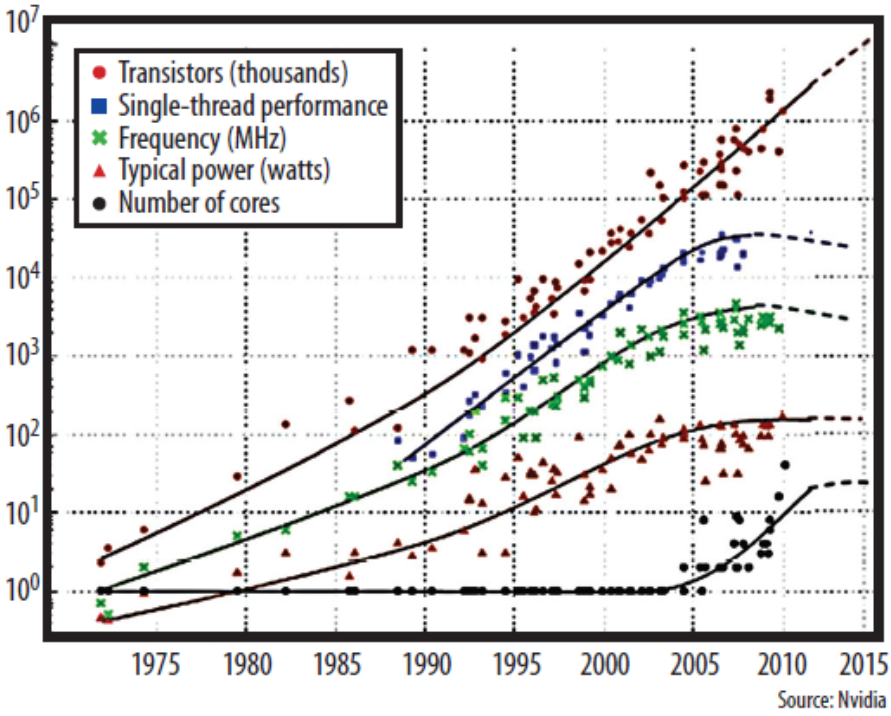
1 INTRODUCTION

1.1 COMPUTING TREND

Historically, compute performance has increased at a predictable pace, doubling every 18-24 months. This trend was until recently largely sustained by increased transistor count coupled with increased processor frequency (enabled by architecture, design, process). During this era, frequency was the primary measure of performance. With increased transistor count and ever higher frequency, CPU power increased accordingly. This trend continued until a power wall was hit and the unconstrained power trend could no longer be sustained. To continue delivering the performance improvement cadence required to sustain the industry without a corresponding increase in power, new techniques and approach at all levels of processor design: system, architecture, circuit, transistors are required.

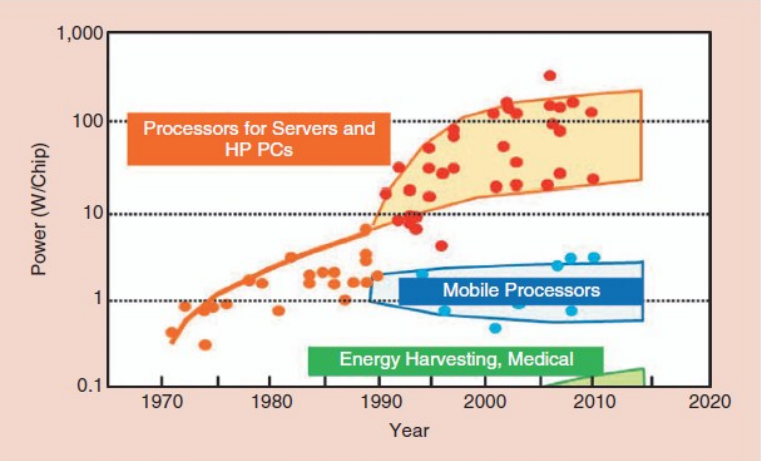
The historic computing trends are aptly summarized in Figure 1.1[1]. The use of frequency as the primary performance enabler saturated around 2004-2005 with the introduction of multi-core computing to the mainstream. With multi-core, overall system performance can be improved by increasing the number of core, without necessarily improving single thread performance per core as multiple threads can be executed concurrently over multiple cores. Multicore however does nothing to arrest the transistor count trend which is projected to continue its current trajectory as more cores are added to the die. Furthermore, while multiple cores reduced the need for higher frequency per core, the power wall still exists as total power budget does not scale with cores. For example two cores executing concurrently at low frequency must fit into the same power envelop as a corresponding single core CPU at higher frequency. With no more head room for power increase, reducing power per core becomes essential as well as managing power between cores. The maximum frequency that each core can operate is limited by the available total power envelope. Consequently power, not frequency, becomes the performance limiter.

Figure 1.2 shows the increasing power trend in major computing platforms: servers, high performance PCs, mobile and low power processors, as more transistors are required on the die to sustain the evolution of more compute intensive applications such as natural speech synthesis, 3-D applications, games etc. As shown in Figure 1.3, mobile System-On-Chip (SoC) power is expected to increase significantly in the coming years. At the same time energy requirement by high performance systems like server will grow exponentially as demand by large server farms explodes with increased demand for server based services like cloud computing and web servers.



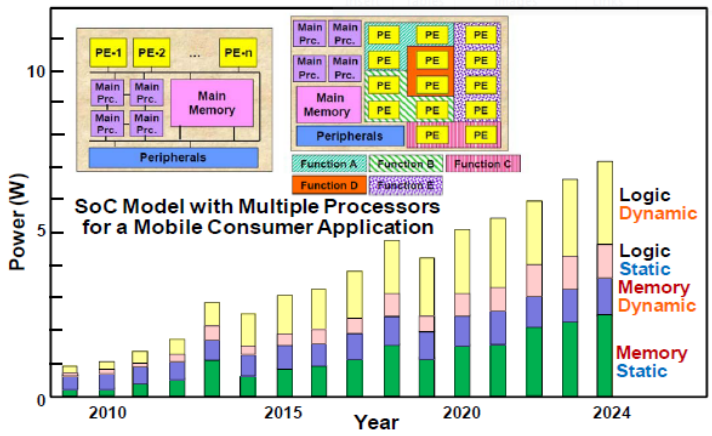
[N. Leavitt. Computer '12]

Figure 1.1 Historic and projected trends in transistor count, single-thread performance frequency, total power, and number of cores. While transistor count continues the historic growth trend, frequency and single-thread performance saturated with the introduction of multi-cores. [1]



[T. Masuhara SSCS'13]

Figure 1.2 Compute platform power trends: Server, high performance CPU, mobile processor and low power DSPs. [2]



[T. Masuhara ASSCC'11]
Figure 1.3 ITRS projected mobile multi-processor power trend [3]

1.2 NEW COMPUTING PLATFORMS CHALLENGES

As the computer industry seem to have figured out how to improve performance with multi-core, there is a dramatic new computing paradigm shift that put the need for even lower power at the forefront of CPU design. This was set forth by the strong move to mobile computing in recent years. New computing platforms like Ultrabook, Tablets, Smart Phones, and Wearables (Figure 1.4) have become an integral part of daily life. This is expected to grow exponentially in the next few years. As mobile platform dominate the user experience, battery life and skin temperature become critical. This also means that desktop level performance has to be miniaturized into these small form factors. The challenge therefore is how to bring the same high performance user experience to these small platforms.

Figure 1.4 Proliferation of new platforms with varying degree of power requirements. [Source Intel]

The new computing paradigm requires an increased performance at lower power per generation. This requires an inverse relationship between performance and power which is not the conventional trend in computing. This has necessitated the need for more innovative ways to bend the power curve while continuing the historic performance trend predicted under Moore's Law. If transistor count continues to be the primary means of improving performance, then how do we continue increasing transistor count and at the same time reduce power. Furthermore, while performance increases approximately as a square root of area (transistor count) [4], power increase linearly with area (transistor count). Hence increasing transistor count to achieve performance is unsustainable if power is not curtailed.

1.3 MEMORY POWER

Memory size is one of the primary drivers of die size growth over the years. On the same architecture, memory size directly correlation with performance. Increase memory size is therefore one of the common performance knobs. As shown in Figure 1.5, cache size is expected to continue the current trajectory over the coming years. As more memory is integrate onto the die and increasing proportion of the die area is allocated to memory, memory power becomes a singularly important determinant of overall die power. Another factor accounting for memory's increasing impact on die power is the integration of high performance graphics (GPU) on the same die as the CPU and its associated memory requirement. This is exemplified by the latest Intel Haswell processor (Figure 1.6).

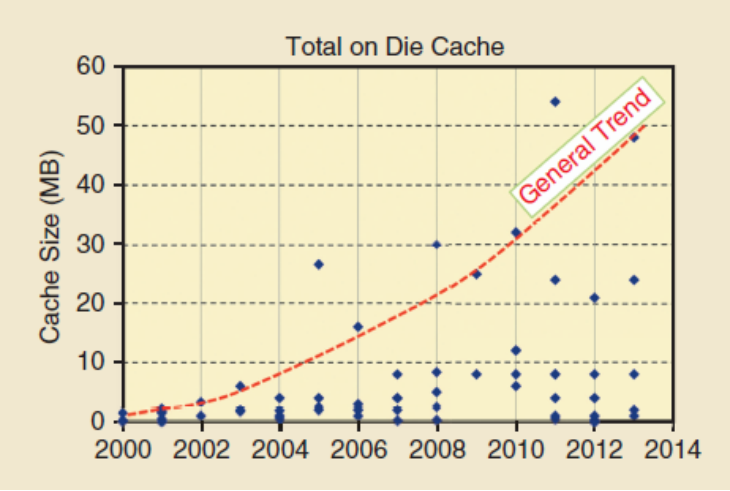
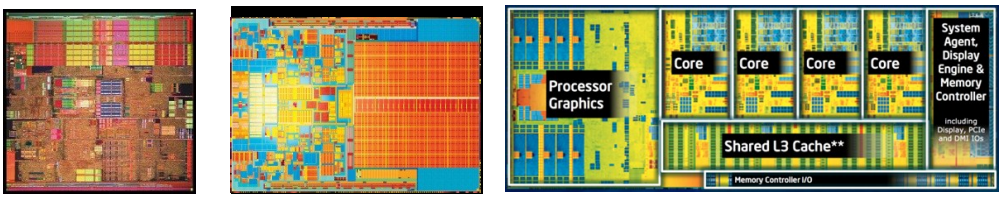


Figure 1.5 Cache size trend [5] [source SSCS'13]



(a) (b) (c)

Figure 1.6 Generations of Intel processor showing increase in memory size (a) 90nm Prescott (b) 45nm Nehalem (c) 22nm Haswell

1.4 CPU POWER BREAKDOWN

Large on-die memories and caches are typically implemented as either SRAM or Register Files. SRAMs are used for high density, long latency memories like Level-2/3 caches while register files (RF) are used for fast access memories like level-0 caches. Register files are also the primary memory element in the CPU/GPU execution engine because of their large storage density compared to latches and flops, and small access latency and variability tolerance in low voltage operation compared to SRAM. Both high demand on memory and performance has resulted in increased number of RF over CPU generations.

RFs therefore contribute substantially to total CPU die power. On the 32nm Westmere (WSM) IA Core [6], RFs accounted for 30% of the die leakage and dynamic power on a typical TDP benchmark (Figure 1.7). Compared to SRAM that uses extreme low leakage devices due to their lower performance requirement, RF uses the regular high performance logic devices. Furthermore, the use of register files in the high activity execution engine account for their high dynamic power. Thus technique to reduce leakage and dynamic power of RFs is paramount in low power regime.

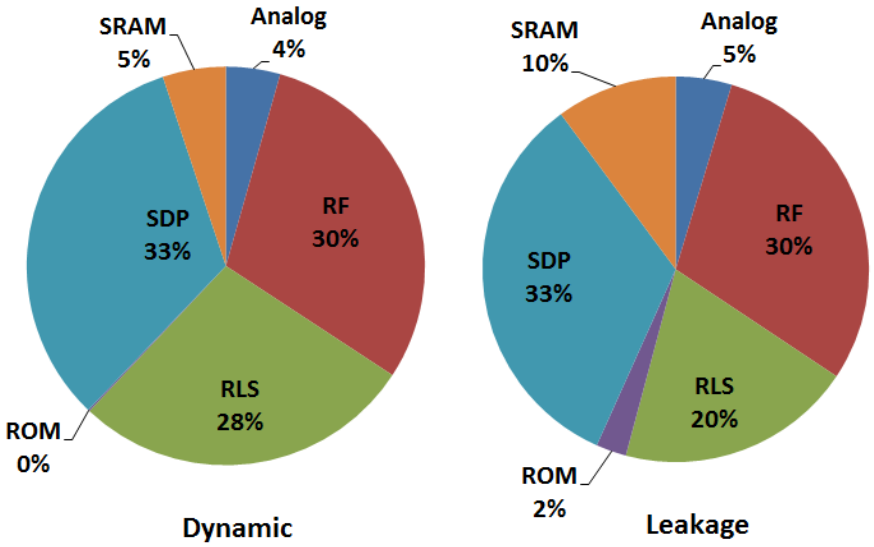


Figure 1.7 Dynamic and Leakage power distribution in 32nm WSM IA Core by design styles: Register Files (RF), Synthesized Block (RLS), Read Only Memory (ROM), Structural Data Path (SDP), and SRAM

1.5 REASERACH GOAL

The goal of this research is to two-fold:

1. Explore new design techniques to reduce register file power.
2. Develop an accurate early power estimation model for architecture and design space exploration that addresses key limitation of existing models.

1.5.1 Register File Low Power Design

Circuit level power optimization is one of the key pillars of power opportunity stack (Operating System, Software, Architecture, RTL, Circuit, Devices). Circuit performance limitation generally imposes constraints on what can be achieved at the architectural or even with devices. The goal of circuit/logic level power optimization is to make maximum use of the opportunities provided by the available devices and also provide opportunities for further power reduction at the architectural level. Circuit/Logic power design is therefore an intermediary between device level opportunities and architectural level possibilities.

In this research, power distribution in a real world high performance CPU design (32nm IA core) Register Files were analyzed to identify key power consuming logic blocks and their design sensitivities. New circuit techniques to reduce power in these high power consuming logic blocks: memory, write bitline, read bitline, decoder, and clocks were then developed. New stack-forcing leakage techniques that minimize area and delay overhead compared to existing stacking techniques are proposed. We use observation of data activity and logic state residencies (static probabilities) under typical application workloads to develop techniques to reduce write data distribution and memory bitcell power.

The contributions of this research to low-power register file design are as follows.

1.5.1.1 Read Bitline

This research a proposes new scalable RF read topology that combines the advantages of dynamic read bitline and the static wordline to reduce both leakage and dynamic power.

For performance and area efficiency, RF read access is usually implemented with wide-NOR dynamic logic. However, continued device and threshold voltage scaling [8] poses a number of challenges: (1) Increased device leakage from threshold scaling (2) Reduced noise robustness due to increased bitline DC droop from device leakage (3) Increasing wire constrained bitcell size as devices continue to scale better than wires across process generations. Alternative circuit techniques that have been proposed to address these challenges generally tradeoff area, standby leakage, or delay. For example, leakage-tolerant techniques [9-13] address bitline noise robustness by reducing bitline active leakage (DC droop) but do not address standby leakage. These techniques incur significant bitline delay penalty and do not scale with increase in number of pulldowns per bitline segment. This research proposes a new technique that uses a static unclocked wordline with dynamic bitline sub-segmentation and stack-forcing to reduce active and standby leakage, and delay. The proposed technique improves read bitline segmentation scalability to a larger number of entries per local bitline segment. For an example 32nm-CMOS 4-write, 6-read port 32 bits x 168 entries register file, the proposed bitline segmentation technique improves local bitline delay by 33%, lowers standby leakage by 70%, and reduces the number of clocked wordlines by 83% when compared with a conventional design. Furthermore, wordline sharing is shown to reduce the number of wordlines by as much as 58%.

1.5.1.2 Memory and Write Bitline

This research proposes new techniques to reduce dynamic and leakage power in register file data distribution using data state residencies and activity factor profile.

Within RFs, the largest single contributor to dynamic power in IA 32nm CPU is the write data distribution. The write data distribution (global and local) accounts for 25% and 22% of RF dynamic and leakage power respectively. Write data distribution power has been increasing over process generations as wire scaling challenges and increasing low-voltage operation constraints have necessitated the use of dual-ended write SRAM bitcell topologies [14]. This research explored various data gating topologies (Global, Midway, Local), logic implementations (NAND, NOR, Tri-State), and techniques (Stack-Forcing, State-Forcing) to reduce both dynamic (by as much as 96%) and leakage power. Due to area and delay overhead careful tradeoff analysis is required to justify its implementation. A simple and accurate data gating break-even analysis model was developed to enable designer evaluate the impact of data gating and potential power benefits on their specific design before implementing data gating. The model comprehends "Data" and "Enable" switching activity, signal probability, and logic implementation overhead with an average estimation error range of ±5%

1.5.1.3 Low Voltage Design

This research also explores candidate topologies for Near/Sub-threshold RF design for optimal energy-delay computation.

Low voltage operation is one of the most effective ways to reduce energy consumed per instruction as dynamic and leakage power have quadratic and exponential dependencies on voltage respectively. Optimal computation per energy occurs at near threshold operation. Enabling Dynamic Voltage and Frequency Scaling (DVFS) to operate much lower voltage in lower power states (C-States) or performance state (P-State) significantly reduces power consumption. One of the current topic of research is in near-/sub-threshold operation. To improve chip area density, memory blocks such as SRAMs and Register Files are built with smallest possible device sizes that meet performance constraints on a technology node. They are therefore most susceptible to increased variation effects as process technologies continue to scale critical device dimensions. Consequently SRAM/RFs have traditionally been the bottleneck to low voltage operation. This work explores the challenges and limitation of Near-/Sub-Threshold RF/SRAM operations. We review current design techniques employed to reduce RF/SRAM minimum operating voltage ($\boldsymbol{V_{min}}$). We compare different topologies and show that in 22nm, with 3σ variation, an interruptible latch based design can reduce an RF write $\boldsymbol{V_{min}}$ by 200mV and delay by up to 3X when compared to conventional 8T dual-ended bitcell. We also propose using a tri-state read ports to improve RF Read $\boldsymbol{V_{min}}$ by 300mW and up to 2X delay improvement over conventional read.

1.5.1.4 Decoder

Address decoding is a common logic function for SRAM operation. An RF/SRAM decoder that simultaneously reduces area, timing, and power using device sharing and power gate techniques is proposed. It exploits the mutual exclusivity of the decoding logic outputs and shares devices to achieve as much as 30% NMOS (PMOS) device width reduction of a 3-to-8 NAND (NOR) decoder in a 32nm technology. A decoder power gate scheme that eliminates floating outputs through state-forcing and reduces leakage by 90% is proposed. Analysis of the shared decoder and sizing guidelines using simple Elmore Delay model were developed to guide designers in optimal shared decoder gate sizing.

1.5.2 Array Power Modeling

One essential aspect of low-power design exploration is an accurate power, area, and timing prediction for early tradeoff analysis. As project schedule continue to shrink due time to market pressures, critical project decisions (area, power, and frequency) need to be made early. This requires early architectural and design tradeoff studies of which a significant proportion involves either changes to an existing arrays or addition of new arrays: Register Files, SRAM, Read-Only Memory. With register files accounting for 30% of high performance CPU power, RFs power tradeoff feature prominently in early design tradeoff analysis.

Majority of register file (>75%) on high performance CPUs are custom designed. There could be as many as sixty (60) unique and custom register files on modern high performance CPU design. These register files cannot be easily compiled due to complexity, performance, area, or other constraints. Even when a compiler is possible, it may take up to 6-months to develop a new compiler or update an existing compiler to a new process node for a single topology. Each manual design however introduces additional uniqueness to the RF that impacts its power (different floor plan, driver size, clock distribution etc.). How do we accurately model all these distinct RF topologies and implementations for architectural and design space exploration?

1.5.2.1 Existing Models

Due to this wide range of RF array implementations, RF modeling is a particularly difficult challenge. To constrain the modeling problem, existing parametric RF/SRAM estimation models typically assume a specific circuit topologies and implementations. Model equations are then developed based on the assumed implementation. To model a different topology, these architectural power models and

performance simulators either use the existing power model essentially unchanged (inaccurate for the new topology) or develop new models for the different topology (time consuming). Existing architectural models like Wattch [16] that do not take circuit implementation into consideration can have error as high as 94% [17]. Unified models that cover different RF topologies without the need to regenerate model equation are unavailable.

Existing models also do not provide the capability for architects and designer to explore the numerous circuit implementations options such as device sizing, data gating, segmentation, and device stacking that significantly impacts the power, area, and timing profile of an RF without requiring a new models.

The two main modeling methods used are analytical and empirical.

Analytical models like PRACTICS[20], NVSim[21] McPAT[22], and others [23-25], using CACTI [18] approach, derive analytical equations that model the key array capacitances and device dimensions based on a specific topology and process technology parameters. To adapt these models to different topologies and technologies require changes to the analytical formulas and parameters.

Empirical models rely on equations derived from data from entire suite of arrays. A major drawback of regression based models [25, 26] is that they require the implementation of several RF configurations to curve fit the empirical data for each topology and technology. There could be as greater than sixty (60) distinct custom RF topologies/implementations on a high performance CPU. A curve fitting modeling approach will require large number of samples of each unique topology which do not exist for custom RFs. Moreover, empirical models are only valid for the specific circuit topology and technology used to generate the model coefficients. Thus, they usually present a method rather than reusable model equations.

1.5.2.2 Proposed Model

The proposed model developed during this research is a hybrid of empirical reference and analytical equations.

The distinct features of the proposed model are:

(1) A reusable power, area, and delay model equations that can be customized for different array topologies by only modifying the model input parameters without requiring new equations, significantly reducing the time required to model new topologies.

(2) A unified model equations for all stages (components) of the array i.e. wordline, bitline, decoder etc. all use the same analytical equations. A single model equation with parameterization that can capture the uniqueness of any register file. In existing models, each RF stage is modeled with different equations. Hence new topologies require new equation development.

(3) Requires a single empirical reference data-point of any *MxN* (M-bits x N-entry) configuration of a topology to accurately model any other configuration of that topology.

(4) Incorporates accurate models for estimating the impact of practical array circuit implementations such as segmentation, data gating, stacking, and driver sizing. This significantly improves model accuracy and provides a tool for architects and designers to do real world tradeoff analysis of possible implementations.

(5) Incorporates activity factor and static probability propagation for benchmark modeling. New benchmarks are modeled which provide architects a tool for evaluating potential impact of a new benchmark on the design.

The proposed model is a set of unified topology-independent analytical equations that become customized for a topology by their reference empirical data and model parameters. On distinct topologies of multi-port read, single- and dual-ended writes, this hybrid reference based approach demonstrates an average error range of <10% for delay, leakage, dynamic, and area prediction. We show that for a specific topology, any reference configuration can be used for accurate prediction. We also show how the model is used to make real world architectural and design tradeoff decisions.

1.6 RESEARCH METHODOLOGY

The research was based on circuit simulations using state of the art analysis tools and driven by challenges facing the high performance microprocessor design at the 32nm and 22nm technology node and beyond. The examples in this research are real world industry design and the design challenges addressed are current and forward looking.

The power, area, and delay models were developed to address practical challenges in microprocessor design. This was driven by model needs and requirement from designer and architects perspective. Specifically, we analyzed what relevant real world tradeoffs are made by circuit designer during the design of high performance processor and how power, area, and delay model are actually used by designers in making those tradeoffs. In addition, feedback from architects on what design data is needed for early

architectural decision making was incorporated in the model development. Models were validated against real world Register Files on Intel's latest state of the art 32nm Haswell processor.

1.7 DISSERTATION OVERVIEW

The organization of the dissertation is as follow:

- **Chapter 1** presents an overview of current industry trends and the research goals
- In **Chapter 2** device and circuit level power overview is presented. Fundamental power limitations, challenges, and common power reduction techniques are also discussed in this chapter.
- In **Chapter 3** we present an overview of register file design. We also look at architectural and functional usage of register files, discuss RF circuit implementations, and review RF read and write topologies and power reduction techniques.
- **Chapter 4** presents a review of near-/sub-threshold RF and SRAM design challenges and solutions. A comparison of different RF topologies under low voltage operation is also presented.
- **Chapter 5** introduces the proposed new RF read topology that employ bitline stacking and sub-segmentation, coupled with static wordline to significantly reduce bitline leakage and area.
- In **Chapter 6**, write data gating strategies are presented. We explore memory and write data leakage and dynamic power dependencies on data residencies and propose new techniques to reduce both leakage and dynamic power.
- **Chapter 7** discuses a new decoder technique that reduces decoder area, power, and delay by sharing mutually exclusive devices among decoder output.
- In **Chapter 8**, we present the proposed new array model that was developed during this research. We also discuss how the model in used in real world CPU design decision making.
- **Chapter 9** presents an interactive web interface that implements the proposed array model.
- Research conclusions are presented in **Chapter 10**.

1.8 RELATED PUBLICATIONS

[1] Eric Donkoh, Patrick Chiang., "A Low-Leakage Dynamic Register File with Unclocked Wordline and Sub-Segmentation for Improved Bitline Scalability". *International Symp. On Low Power Electronic and Design (ISLPED) 2012*

[2] E. Donkoh, A. Lowery, E. Shriver., "A Hybrid and Adaptive Model for Predicting Register File

and SRAM Power Using a Reference Design" *Design Automation Conference (DAC) 2012*

[3] Eric Donkoh, Teck-Siong Ong, Yan Nee Too, Patrick Chiang, "Register File Write Data Gating Techniques and Break-Even Analysis Model" *International Symp. On Low Power Electronic and Design (ISLPED) 2012*

[4] *"CustPAD: A Customizable Power, Area, and Delay Model for Architectural and Design Space Exploration of Register Files and SRAM Structures Using a Hybrid Reference Design"* Journal Paper in Progress.

1.9 REFERENCE

[1] Neal Leavitt, "Will Power Problems Curtain Processor Performance" *Computer, May 2012, pp. 15-17*

[2] Toshiaki Masuhara "Quest for Low-Voltage and Low-Power Integrated Circuits: Towards a Sustainable Future" *IEEE Solid State Circuit, winter 2013, vol , no.1 pp. 8-26*

[3] Toshiaki Masuhara "Challenges in Low Voltage and Low Power IC Towards a Sustainable Future" *IEEE Asian Solid-State Circuit Conference, Nov 14-16, 2011*

[4] Shekhar Borkar, Andrew Chien, "The Future of Microprocessors" *Communications of the ACM, May 2011, Vol 54, No. 5 pp 67-77*

[5] K.C. Smith et al., "Through the Looking Glass II - Part 1 of 2: Trend Tracking for ISSCC 2013 [ISSCC Trends]" , *SSCS 2013, pp 71-89*

[6] N. A. Kurd et al., "Westmere: A family of 32 nm IA processors," *IEEE Int. Solid-State Circuits Conf. (ISSCC),* San Francisco, CA, Feb. 2010.

[7] R. Kumar et al., "A family of 45nm IA processors," *ISSCC'09 Digest of Technical Papers*, pp.58-59, Feb. 2009

[8] P. Packan et. al., "High Performance 32nm Logic Technology Featuring 2nd Generation High-k + Metal Gate Transistors," *IEDM 2009, pp 1-4*

[9] S. Tang et. al., "A leakage tolerant dynamic register file using leakage bypass with stack forcing (LBSF) and source follower", *2002 Symp. VLSI Circuits, pp 320-321*

[10] A. Agarwal et al., "A Leakage-Tolerant Low-Leakage Register File with Conditional Sleep Transistor", *Proc. SOC Conference,2004, pp 241-244*

[11] A. Alvandpour et. al., "A sub-130-nm conditional keeper technique," In *IEEE* Jour. *Of Solid-State Circ.,* May 2002.

[12] R Krishnamurthy et. al., "A 130-nm 6Ghz 256x32bit leakage tolerant register file," In *IEEE JSSC.,* 2002.

[13] S. Borkar, "Circuit techniques for subthreshold leakage, avoidance, control, and tolerance*," IEDM Tech. Dig., pp. 421-424,Dec, 2004.*

[14] K. Anshumali et al. " Circuit And Process Innovations to Enable High-Performance and Are Efficiency on the Nehalem and Westmere Family of Intel processors" *Intel Technology Journal, vol 14, pp 104-127*

[15] S. Narendra, et al, "Scaling of Stack Effect and its Application for Leakage Reduction", *ISLPED, 2001*

[16] D. Brooks, et al, "Wattch: A Framework for Architectural-Level Power Analysis and Optimization," *ISCA 2000*.

[17] Mamidipaka, M.; Khouri, K.; Dutt, N.; Abadir, M. "IDAP: a tool for high-level power estimation of custom array structures" *IEEE Trans on Computer-Aided Design Vol 23, No. 9 2004, pp 1361-1369*

[18] S. J. E. Wilton and N. P. Jouppi, "CACTI: An enhanced cache access and cycle time model," *IEEE J. Solid-State Circuits*, vol. 31, no. 5, pp. 677–688, May 1996.

[19] N. Vijaykrishnan, et al. "Energy-driven integrated hardware-software optimizations using SimplePower", *ISCA 2000*.

[20] A. Zeng, A.; Rose, K.; Gutmann, R.J. "Memory performance prediction for high-performance microprocessors at deep submicrometer technologies" *IEEE Trans. On Computer-Aided Design. Vol 25, No. 25. Sept 2006 pp1705-11718*

[21] X. Dong, et al, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory" *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, pp. 994–1007.*

[22] Sheng Li et al., "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures" *MICRO Dec 12-16 2009*

[23] Agrawal, B.; Sherwood, T., "Ternary CAM Power and Delay Model: Extensions and Uses" *IEEE Trans on VLSI, Vol 16, No 5, 2008, pp 554-564*

[24] Xuemei Zhao, et al: "Design and Realization of a Low Power Register File Using Energy Model", *PATMOS 2002:* pp. 268-277.

[25] Minh Q. Do, et al, "Parameterizable Architecture-Level SRAM Power Model Using Circuit-Simulation Backend for Leakage Calibration," pp.557-563*, ISQED 2006.*

[26] S. L. Coumeri, D. E. Thomas Jr, "Memory Modeling for System Synthesis", *IEEE Transactions on VLSI,* June 2000.

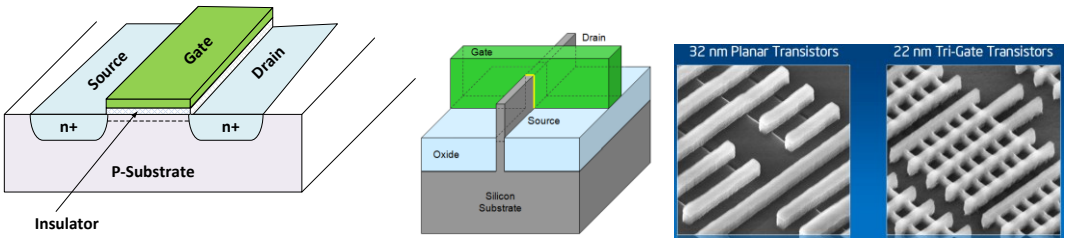
[27] David Browning et al., "Greater Mobility Through Lower Power" Intel technology journal, vol 12, issue 03, oct 2008, pp 211-217

2 BACKGROUND – POWER OVERVIEW

2.1 CMOS TRANSISTOR OVERVIEW

2.1.1 Transistor Operation

The cross-section of a planer and tri-gate CMOS transistors are is shown in Figure 2.1. For illustrative purposes the discussion will be based on the NMOS transistor. The same principle works for the PMOS except electrons are replaced by holes and vice-versa. An NMOS transistor is formed by a highly doped n+ region in a lightly doped p-type substrate, forming an interchangeable drain and source terminals. A metal or poly-silicon gate between the drain and source sits on an insulating silicon dioxide/high-K dielectric material. In a tri-gate transistor, the diffusion wraps around the gate forming a 3-dimensional structure of fins. With tri-gate, a conducting channel is formed on three sides of a vertical fin. The transistor behavior is controlled by applying bias voltage to the terminals.



(a) (b) (c)

Figure 2.1 (a) Planner/2-D transistor (b) Tri-Gate/3-D transistor (c) images of 2D and 3D transistor [source: Intel]

The lightly doped p-type substrate acts as an insulator between the two highly doped n+ terminals. When voltage source is applied to the drain or source with gate at zero bias voltage (relative to the substrate), a diode is formed between the p-type and the n+ source. The p-type substrate will continue to act as an insulator between n+ source and drain terminals (Accumulation - Figure 2.2 (a)). In the zero gate bias state, an application of a positive voltage to the n+ terminal forms a reverse bias *pn*-junction and hence no current flows between drain and source. When a positive bias voltage (relative) to the substrate) is applied to the gate, substrate electrons are attracted toward the gate while holes are repelled away from the gate as illustrated in Figure 2.2. Initially attracted electrons recombine with

holes underneath the gate to create a depletion region. As voltage is increased, electrons begin to accumulate under the gate as more electrons are attracted and holes are further repelled. When a sufficiently high gate bias voltage is applied, enough electrons accumulate under the gate to form a continuous channel between the source and the drain. The result is an inverted channel under the gate with electrons replacing holes (Inversion mode). This creates a conduction channel where electrons can flow from source to drain through the channel when a positive drain-to-source bias voltage is applied. The minimum gate voltage required to create a conductive inversion channel is the threshold voltage. The threshold voltage defines the boundary condition for a CMOS transistor operation.

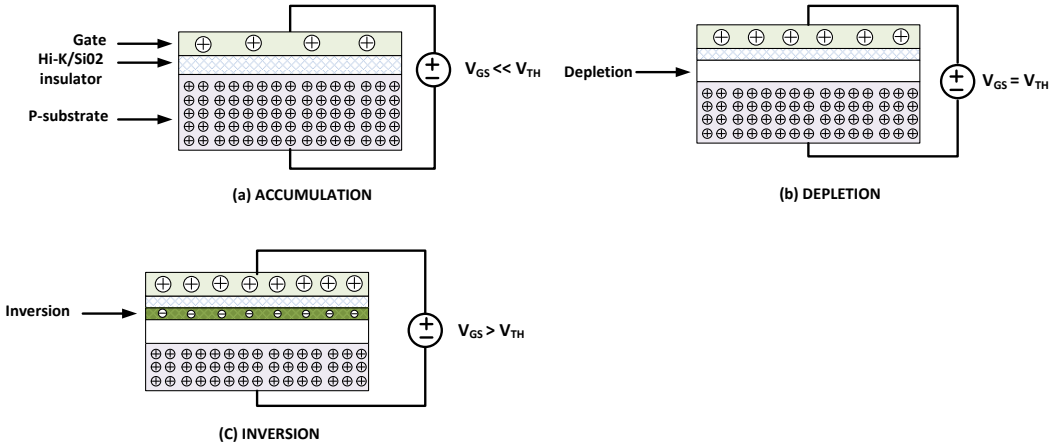


Figure 2.2 Figure : Area under the NMOS transistor gate in (a) Accumulation (b) Depletion and (c) Inversion modes as voltage is applied between gate and substrate.

2.1.2 The threshold voltage

An NMOS transistor threshold voltage can be expressed as [1-2]:

$$V_{TH0} = V_{FB} + 2\phi_B + \frac{Q_B}{C_{ox}} \tag{2.1}$$

Where

$$Q_B = \sqrt{2\varepsilon_{si} q N_A (2\phi_B)} \tag{2.2}$$

$$\phi_B = \frac{kT}{q} ln\left(\frac{N_A}{N_i}\right) \tag{2.3}$$

$$V_{FB} = \phi_{ms} + \frac{Q_{fc}}{C_{ox}} \tag{2.4}$$

Q_B is the bulk charge term and ϕ_B is the bulk Fermi potential (the delta between doped and intrinsic semiconductor Fermi energy level). N_A, N_i are the doped and intrinsic (un-doped) substrate carrier densities respectively. q, k, T are the electron charge, Boltzmann constant, and temperature. The term kT/q is the thermal voltage. ε_{si} is the substrate silicon dielectric constant, C_{ox} is the gate oxide capacitance, ε_{ox} is the gate silicon oxide dielectric constant, V_{FB} is the flat-band voltage, ϕ_{ms} is the work function delta between gate material and the silicon substrate ($\phi_{gate} - \phi_{si}$), Q_{fc} is the fixed oxide surface charge (from doping effect and oxide imperfection)

2.1.2.1 Body Bias

The threshold voltage is modulated by the substrate to source bias voltage V_{SB} (body effect). The present of the bias voltage offsets the amount of energy required to create the inversion layer, hence modulating V_{TH}.

$$V_{TH0} = V_{FB} + 2\phi_B + \frac{\sqrt{2\varepsilon_{si} q N_A (2\phi_B + |V_{SB}|)}}{C_{ox}} \quad 2.5$$

$$V_{TH} = V_{TH0} + \gamma\left(\sqrt{2\phi_B + |V_{SB}|} - \sqrt{2\phi_B}\right) \quad 2.6$$

Where $= \frac{\sqrt{2\varepsilon_{si} q N_A}}{C_{ox}}$;

V_{TH0} is the zero bias threshold voltage ($V_{SB} = 0$) and γ is the body bias parameter. The change in V_{TH} due to body bias voltage is therefore:

$$\Delta V_{TH} = \gamma\left(\sqrt{2\phi_B + |V_{SB}|} - \sqrt{2\phi_B}\right) \quad 2.7$$

2.1.2.2 Channel Length modulation(λ)

As the drain voltage V_{DS} is increased the drain to source electric field increases and the depletion region extends from the drain towards the source. This decreases the effective channel length, increasing the drain current (as W/L is increased). The effective channel length [1]:

$$L_{eff} = L - L_{short} \quad 2.8$$

$$L_{short} = \sqrt{\frac{2\varepsilon_{si}}{q N_A}(V_{DS} + V_{GS} - V_{TH})} \quad 2.9$$

The channel length modulation effect is captured empirically by the channel length modulation factor λ

2.1.2.3 Gate length dependency

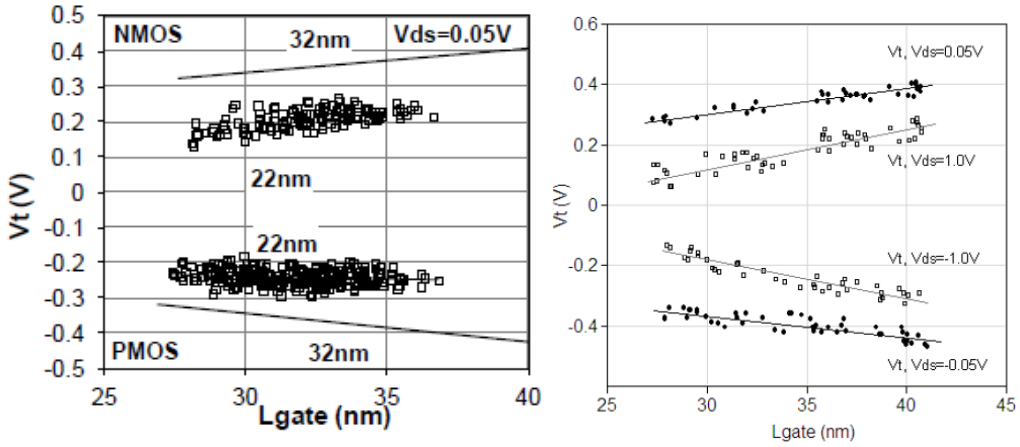
As gate length is decreases, the required work function to create a conducting channel between the drain and source reduces accordingly. This threshold dependency on gate length is used to control leakage by using long channel devices. However, this also has implications concerning process variation impact on leakage. As devices scales down and channel length shrinks, channel length control becomes a difficult task and new techniques to control V_{TH} variations are required. Figure 2.3 show the V_{TH} gate length sensitivity in 22nm tri-gate [3] and 32nm [4].

2.1.2.4 Drain induced barrier lowering (DIBL)

In short channel devices, as drain voltage increases, the depletion region between drain and channel is increased. This extends the depletion to regions under the gate. The resulting effect is the lower of threshold voltage as the energy required by the gate to create the inversion is reduced.

$$V_{TH} = V_{TH0} - \lambda_d V_{DS} \tag{2.10}$$

When the drain depletion reaches close to the source, the drain voltage exerts influence on the source barrier, causing injection of extra carriers and increasing current. In a worst-case scenario the depletion region from the drain reaches the source, causing punch-through. Figure 2.3(b) shows V_{TH} shift due to V_{DS} at different gate length in a 32nm technology [4]. The DIBL effect increases at shorter gate length.



[C. Auth VLSIT'12] [P. Packan IEDM'09]

Figure 2.3 V_{TH} shift due to V_{DS} (DIBL effect) in (a) 22nm [3] and (b) 32nm [4] Intel technology.

2.1.3 Transistor Operation in Super-threshold

Supper-threshold is defined as operation at $V_{GS} > V_{TH}$. The MOS transistor operates in three basic regions in supper-threshold: cutoff, linear/triode, and saturation (Figure 2.4)

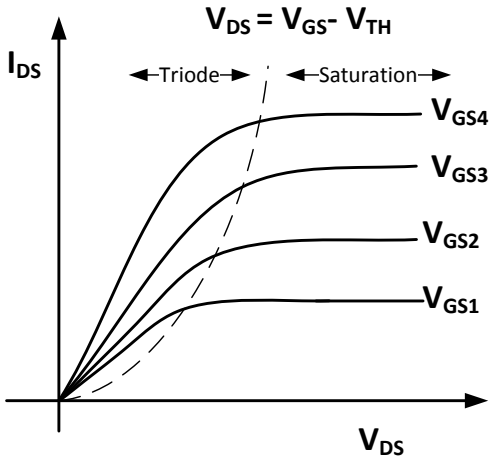


Figure 2.4 CMOS I/V Curve. $V_{GS1} < V_{GS2} < V_{GS2} \ldots . < V_{GSn}$

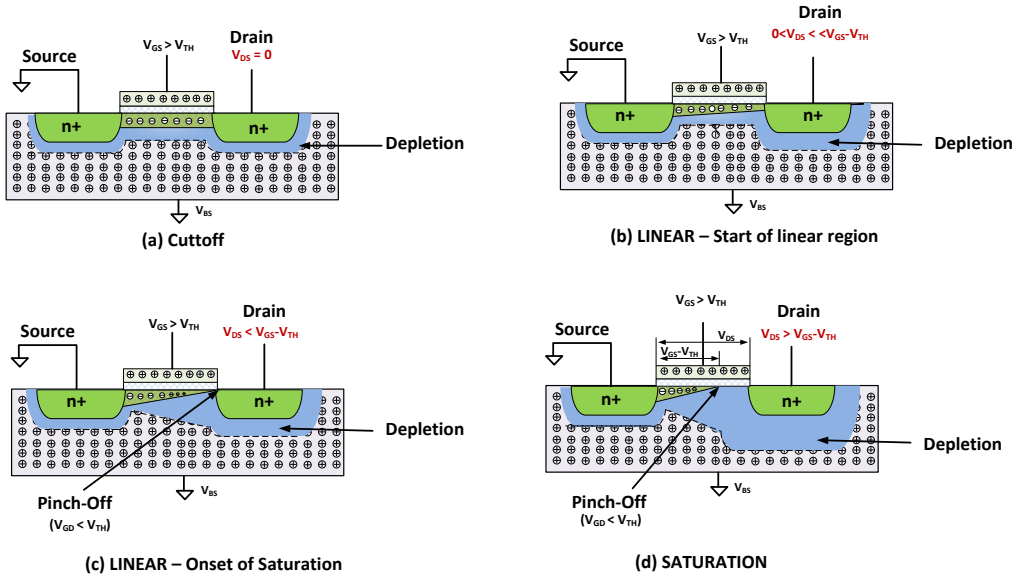


Figure 2.5 CMOS super-threshold operating conditions.

2.1.3.1 Cutoff Region

$V_{GS} > V_{TH};\ V_{DS} = 0$

When $V_{GS} > V_{TH}$ $(@\ V_S = 0; V_D = 0)$, a conduction channel is created under the gate between the drain and source due to a vertical electric field between gate and substrate (Figure 2.5(a)). The conduction channel (inversion layer) is approximately uniform between drain and source at $V_{GS} = V_{GD}$. If $V_{DS} = 0$ no horizontal electric field exist between drain and source and hence no current conduction occurs.

$I_{Dcutoff} = \sim 0$

2.1.3.2 Linear/Triode region

As V_D is increased from an initial value of 0, a horizontal electric field develops between the source and drain as a potential different is developed ($V_{DS} > 0$). This sweeps the electrons from source to drain. As V_D is increased, V_{GD} decreases, reducing the gate-to-drain electric field at the drain. This increases the channel resistance at the drain. The conduction channel (inversion layer) therefore begins to narrow at the drain and is no longer uniform between the drain and source as $V_{GD} < V_{GS}$ (Figure 2.5(b)(c)). Under this condition the conduction channel depth can be modulated by modulating V_{GD}. Hence current is dependent on both V_D and V_G (V_{GD}).

In the linear/triode ($V_{GS} - V_{TH} > 0;\ V_{DS} < V_{GS} - V_{TH}$) operation, the drain current:

$$I_{Dlin} = \mu_n C_{ox} \frac{W}{L}\left[(V_{GS} - V_{TH})V_{DS} - \frac{1}{2}{V_{DS}}^2\right](1 + \lambda V_{DS}) \qquad 2.11$$

μ_n is the carrier mobility, λ is the channel length modulation

2.1.3.3 Saturation

When V_D increases to a voltage where $V_{GD} < V_{TH}$, the drain becomes pinched off (channel cutoff) and the conduction channel from the source doesn't reach the drain. At the source, the full V_{GS} electric field is available for channel inversion ($V_{GS} > V_{TH}$) while at the drain only the gate-to-drain delta, $V_{GS} - V_{DS}$, is available for channel inversion. The conduction channel is non-uniform between source and drain, narrowing to a cutoff before the drain. The pinch-off occurs along the channel where $V_{GS} = V_{TH}$ [1] (Figure 2.5(d)). Under this condition, conduction is due to electron drift as electrons injected into the depletion region are accelerated into the drain by the effect of the drain voltage.

When in saturation mode, the transistor channel is in strong inversion and drain current is independent of drain source voltage V_{DS} in the ideal scenario. In the practice there is a secondary dependency on V_{DS} due to channel length modulation effect λ_d.

In saturation ($V_{GS} - V_{TH} > 0; V_{DS} > V_{GS} - V_{TH}$), the drain current:

$$I_{Dsat} = \frac{1}{2}\mu_n C_{ox}\frac{W}{L}(V_{GS} - V_{TH})^2(1 + \lambda_d V_{DS}) \quad 2.12$$

2.1.4 Transistor Operation in Sub-Threshold

2.1.4.1 Sub-threshold Current

Sub-threshold mode is defined by:

$V_{GS} < V_{TH}$

Sub-threshold current (leakage) has exponential dependency on voltage and V_{DS} (due to DIBL effect). The subthreshold drain current [6]:

$$I_{Dsub} = I_S 10^{\frac{V_{GS} - V_{TH0} + \lambda_d V_{DS} + \gamma_b V_{BS}}{S}}\left(1 - 10^{\frac{-nV_{DS}}{S}}\right) \quad 2.13$$

$$I_S = 2n\mu C_{ox}\frac{W}{L}\left(\frac{kT}{q}\right)^2 \quad 2.14$$

$$S = \frac{\delta V_{GS}}{\delta(\log_{10} I_{DS})} = ln(10)\frac{kT}{q}\left(1 + \frac{C_D}{C_{ox}}\right) = 2.3\frac{kT}{q}\left(1 + \frac{C_D}{C_{ox}}\right) = 2.3n\frac{kT}{q} \quad 2.15$$

Where

$$n = 1 + \frac{C_D}{C_{ox}} \quad 2.16$$

$$C_{ox} = \frac{\varepsilon_{ox}}{t_{ox}}A; \;\; C_D = \frac{\varepsilon_{si}}{d}A$$

$$n = 1 + \frac{at_{ox}}{d} \quad \text{where } \varepsilon_{si} = a\varepsilon_{ox} \quad 2.17$$

d is the depletion layer depth, a is the silicon to oxide permittivity ratio, C_{ox} is the gate oxide capacitance, ε_{ox} is the gate dielectric constant, t_{ox} is oxide thickness, C_D is the depletion layer capacitance, A is the gate area, kT/q is the thermal voltage, S is the sub-threshold slope in mV/decade, n is the subthreshold slope factor, k is the Boltzmann constant, T is the absolute temperature, V_{TH0} is

the zero biased voltage,λ_d is the drain-induced barrier lowering (DIBL) coefficient, $\gamma_b$is the body-bias effect coefficient.

2.1.4.2 Subthreshold slope

Equation 2.13 shows the dependency of subthreshold current on subthreshold slope S. Subthreshold slope indicates the rate of drain current collapse as gate-to-drain voltage is decreased. A lower subthreshold slope is therefore desirable to shut off the transistor quickly when V_{GS} is decreased below V_{TH}. Sub-threshold slope is fairly independent of the V_{DS} but strongly dependent on device parameters C_D/C_{ox} ratio and temperature (Equation 2.15). The subthreshold slope is empirically determined from the slope of $\log(I_{DS})\ vs\ V_{GS}$ at $V_{GS} < V_{TH}$(Figure 2.6) Figure 2.7 shows a 22nm technology subthreshold slope around 70mV/decade [3], an improvement over 100mV/decade in 32nm [4].

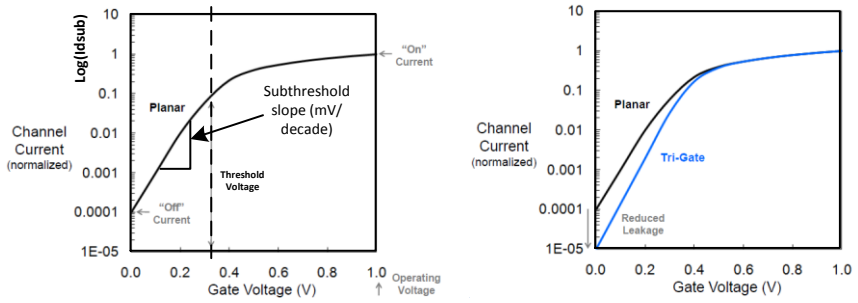
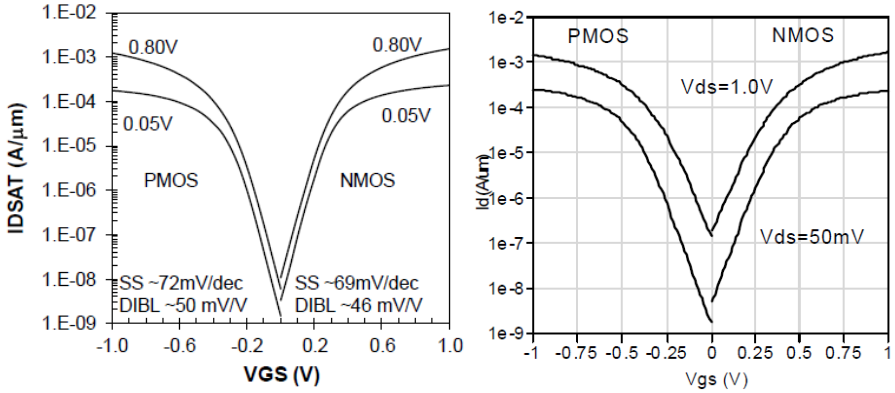


Figure 2.6 (a) Sub-threshold slope definition (b) Planer vs Tri-gate [Source: Intel].



[C. Auth VLSIT'12] [P. Packan IEDM'09]

Figure 2.7 Subthrshold slope in (a) 22nm [3] and (b) 32nm [4]. Subthreshold is ~100mV/decade in 32nm at operationg voltage of 1.0V and at 50mV. Subthreshold slope is independednt of V_{DS}. In 22nm subthreshold slope improved to ~70mV

2.2 LEAKAGE SOURCES

Device Leakage is a key CPU performance constraints facing the industry. As devices are scaled to ever more smaller critical dimension, multiple leakage phenomena begin to come into prominensce. Gate leaakge, for example, has steadily increased over process generation as oxide thichness is decreased (Figure 2.9). Figure 2.8 shows the principal NMOS transistor leakage sources [7].

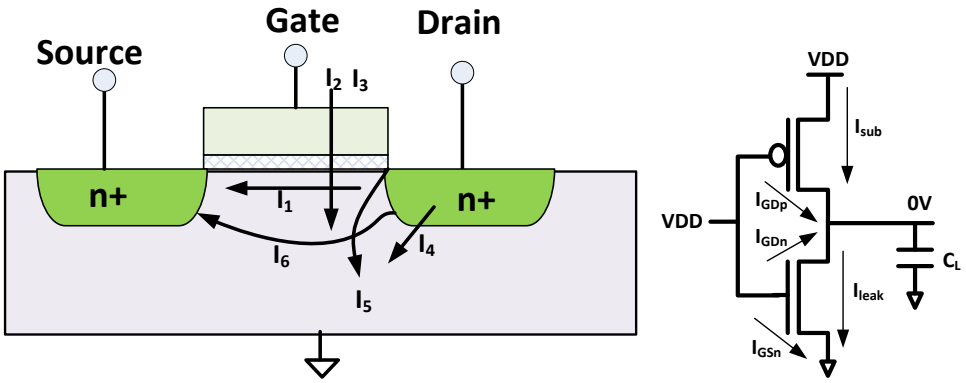


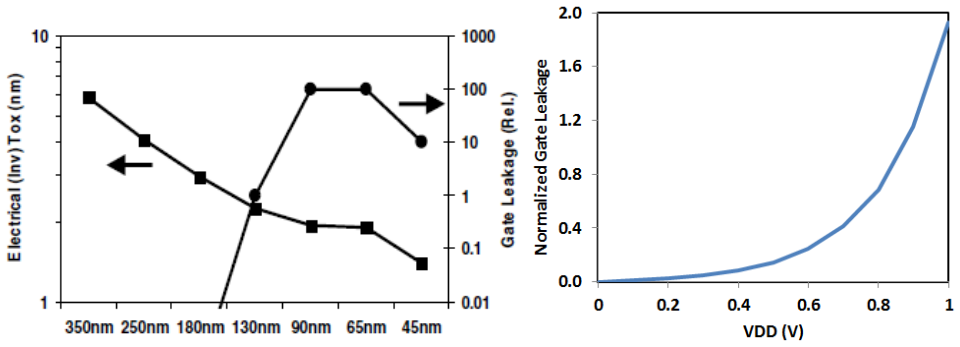
Figure 2.8 Leakage sources in an CMOS transistor. Gate leakage exist whether device is ON or OFF

I1 – Sub-threshold leakage: This is the drain to source current at $V_{GS} < V_{TH}$ as discussed in section 2.1.4. This is mainly due to diffusion current. As gate pitch is reduced with technology scaling, subthreshold leakage increases

I2 – Gate oxide tunneling: As oxide thickness is reduced, an increased electric field across the gate results in electrons tunneling between gate and substrate through the thin gate oxide. Figure 2.9(a) shows the increase in gate leakage over multiple Intel process generations corresponding to gate thickness reduction. An introduction of new gate materials led to one time bending of the curve at 45nm node where Hi-K dielectric material replaced silicon dioxide. Gate leakage has exponential dependency on supply voltage (Figure 2.9(b)) and can be approximated as [6].

$$I_{GD} \propto e^{-T_{ox}} e^{V_{GD}} \tag{2.18}$$

$$I_{GS} \propto e^{-T_{ox}} e^{V_{GS}} \tag{2.19}$$



[P. Auth VLSIT'08]

Figure 2.9 (a)Gate oxide thickness and gate leakage over process generation [8]. Gate leakage increasing with reduced oxide thickness (b) Gate leakage exponential dependency on supply voltage [6].

I3 – Substrate to gate oxide hot carrier injection: At high electric fields, holes and electrons can gain sufficient energy to overcome the interface potential barrier and enter the oxide layer, causing leakage current.

I4 – P-N junction reverse bias current: This is due to source-to-well and drain-to-well reverse bias causing *p-n* leakage current from minority carrier drift and electron-hole pair generation in the depletion region. The reverse bias leakage is a function of junction area and doping concentration [7].

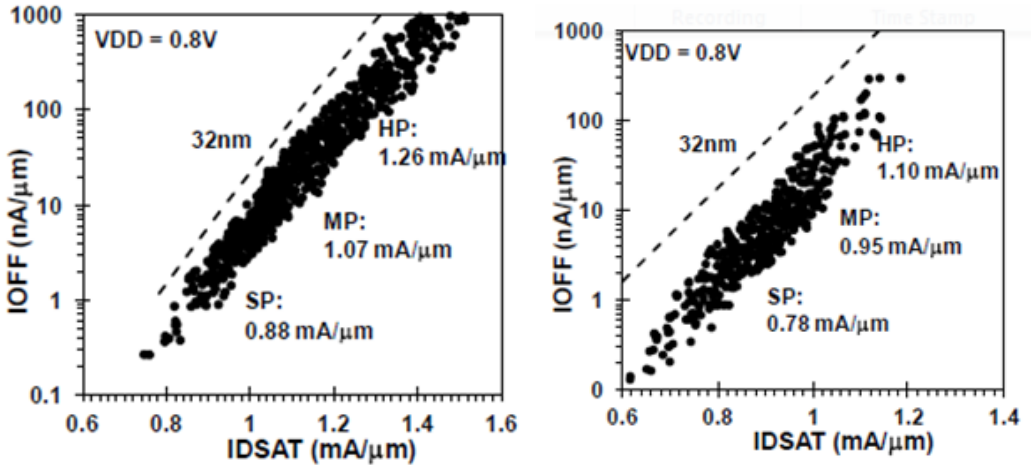
I5 - Gate induced drain leakage (GIBL): GIBL results from high drain junction field effect with biased gate. An accumulation of holes at the surface due to biased gate results in narrow depletion layer at the surface. When negative gate bias is applied relative to V_{DD}, the n+ drain region under the gate become depleted causing field crowding, high field effect and subsequently avalanche multiplication [7]. GIBL increases with thinner oxide thickness and higher V_{DD} and V_{DG}

I6 - Punchthrough: Punchthrough occurs when the depletion originating from the drain due to DIBL effect reaches the source. When the drain depletion reaches close to the source, the drain voltage exerts influence on the source barrier, causing injection of extra carriers and increasing current.

2.3 ION AND IOFF TRADEOFF

Large Ion current (I_{Dsat}, I_{Dlin}) is desirable for high performance logic operations. On the other hand low Ioff is required for lower leakage. As transistor gate length and oxide thickness are scaled over process generations to improve Ion, Ioff increases as well (Figure 2.10). Furthermore as device sizes shrink, the supply voltage is scaled to reduce electric field and improve transistor reliability. This requires a corresponding V_{TH} reduction to maintain a good overdrive voltage ($V_{GS} - V_{TH}$) for high Ion current. Reducing V_{TH} however also increases Ioff. Ioff is also more sensitive to V_{TH} than Ion due to its exponential dependency on V_{TH} . This constrains the potential V_{TH} scaling at a given technology node. To address the conflicting requirement, multiple threshold devices are usually available on a single process node to provide options for trading off high performance (high leakage) with low leakage (low performance). Figure 2.10 shows Ion/Ioff of three device types in 22nm technology [3]. It is then up to the designer to choose the appropriate device type to optimize logic performance and leakage.

	HP	MP	SP
$T_{OX,E}$ (nm)	0.9	0.9	0.9
L_{GATE} (nm)	30	34	34
I_{OFF} (nA/um)	20-100	5-20	1-5



[C. Auth VLSIT'12]

Figure 2.10 Idsat (Ion) vs. Ioff for (a) NMOS and (b) PMOS for various device types (SP, MP, HP) in a 22nm [3]. SP devices have much higher Ion/Ioff ratio due to lower Ioff per um.

2.4 LOGIC AND FUNCTIONAL POWER DISSIPATION

Logic power dissipation can be broadly categorized as follows: switching, short circuit, static, and leakage. The total dissipated power:

$$Power = AF \times (C_L + C_{sc}) \times V_{swing} \times V_{DD} \times f + (I_{leak} + I_{static}) \times V_{DD} \quad 2.20$$

AF is the switching activity factor

C_L is the switching load capacitance

C_{sc} is the short-circuit load capacitance

V_{swing} is the voltage swing during a transition

V_{DD} is the voltage supply voltage

I_{static} is the static current due to bias conditions

I_{leak} is the leakage current

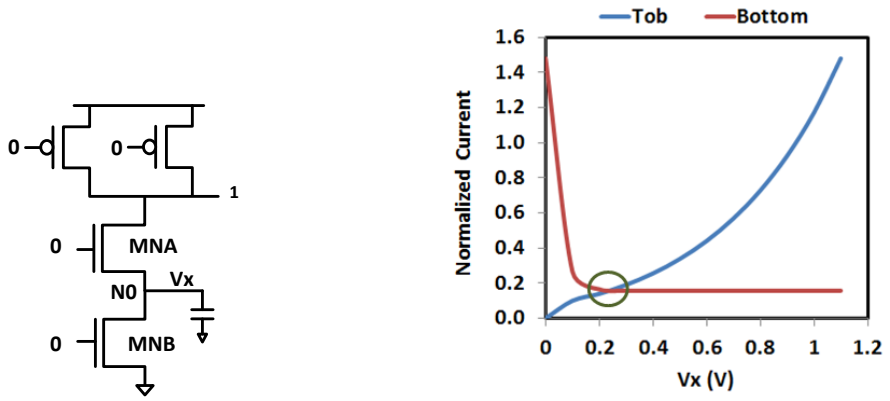
2.4.1 Leakage Power

From design perspective, logic leakage is influenced by the following factors that the designer can control through circuit techniques.
Leakage Power = f (Logic Type, Total Device Width, Device Type, Static Probability, Stacking Factor, Voltage)

Device sizes and types are widely used leakage control techniques. We discuss how stacking, logic type, and static probability characteristics can be used to reduce leakage.

2.4.1.1 Stacking Factor

One way to exploit the exponential leakage dependency on V_{ds} to reduce leakage is device stacking. The principle behind stacking is illustrated by a NAND gate with two stack NMOS devices in Figure 2.11(a) When both NMOS-devices are turned off, the internal node *N0* settles at an equilibrium voltage V_x at which the leakage current of the two series devices *MNA* and *MNB* are equal (Figure 2.11(b))



(a) (b)

Figure 2.11 (a) NAND NMOS Stacking (b) PMOS: Top and bottom devices leakage current dependency on V_x showing convergence point when the two currents are equal.

In the steady state condition the intermediate voltage value is given by [9] :

$$V_x = \frac{\lambda_d V_{dd} + S\, log \frac{w_u}{w_l}}{1 + \gamma_b + 2\lambda_d} \qquad 2.21$$

The converged current is given as [9]:

$$I_{stack} = w_u^{\alpha} w_l^{1-\alpha} I_0 10^{\frac{-\lambda_d V_{dd}(1-\alpha)}{S}} \qquad 2.22$$

where $\alpha = \frac{\lambda_d}{1+2\lambda_d}$

I_0 is the subthreshold current of a single stack transistor with $V_{GS} = V_{BS}$ =0 and $V_{DS} = V_{DD}$

where w_u, w_l are the width of the upper *(MNA)* and lower *(MNB)* stacked devices respectively. V_x is assumed to be greater than $3kT/q$, S is the subthreshold swing, λ_d is the drain-induced barrier lowering (DIBL) factor, and γ_b is the body effect coefficient.

The presence of V_x : (1) reduces the drain-source voltage of both transistors and hence reduces the leakage current due to DIBL. This is more pronounced for the bottom device *MNB*; (2) the gate-source voltage of the top device *MNA* is negative while the source-bulk becomes reverse biased thereby raising its threshold. These factors taken together results in exponential leakage reduction

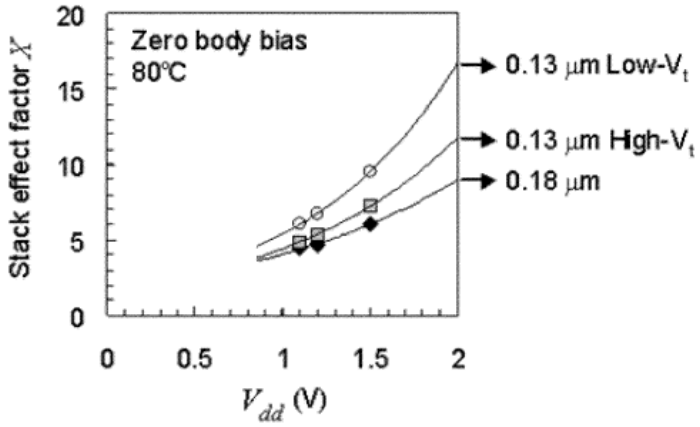
The stack factor of a transistor stack with width w_u and w_l relative to a single reference inverter of width w is given by [9]

$$SF = \frac{I_{inv}}{I_{stack}} = \frac{w}{w_u^{\alpha} w_l^{1-\alpha}} 10^{\frac{\lambda_d V_{dd}(1-\alpha)}{S}} \quad 2.23$$

When $w_u = w_l = w$

$$\frac{I_{inv}}{I_{stack}} = 10^{\frac{\lambda_d V_{dd}}{S}\left(\frac{1+\lambda_d}{1+2\lambda_d}\right)} \quad 2.24$$

The stacking factor is strongly dependent on the DIBL factor λ_d and therefore on the channel length. λ_d increases as channel length is reduced. This results in increased sub-threshold leakage of a single device. However, for a two-stacked logic, a decrease in channel length (increase in λ_d) does not result in increased sub-threshold leakage. Consequently channel length variation has less effect on a two-stack threshold than a single device. The benefit of stacking effect therefore increases over process generations. As shown in Figure 2.12 the stacking factor increases with reduced channel length and lower V_{TH}[9].



[S. Narendra JSSC'04]

Figure 2.12 Stack factor increases with lower V_{TH} and decreasing channel length [9]. Hence the benefit of stacking increases with process scaling.

2.4.1.2 Power Gating

Power gating is an application of the stacking principle to reduce leakage in an inactive logic blocks. Figure 2.13 shows the basic power gate circuits. In inactive mode, when signal PGC = "1" (PGC# = "0") , the NMOS footer and PMOS header are turned OFF. This enforces stacking on the combinational logic and reduces leakage. In active mode, PGC = "0" and a virtual VSS and VDD are established for the combinational logic to operate normally. Typically either PMOS gating or NMOS gating is sufficient to achieve significant leakage.

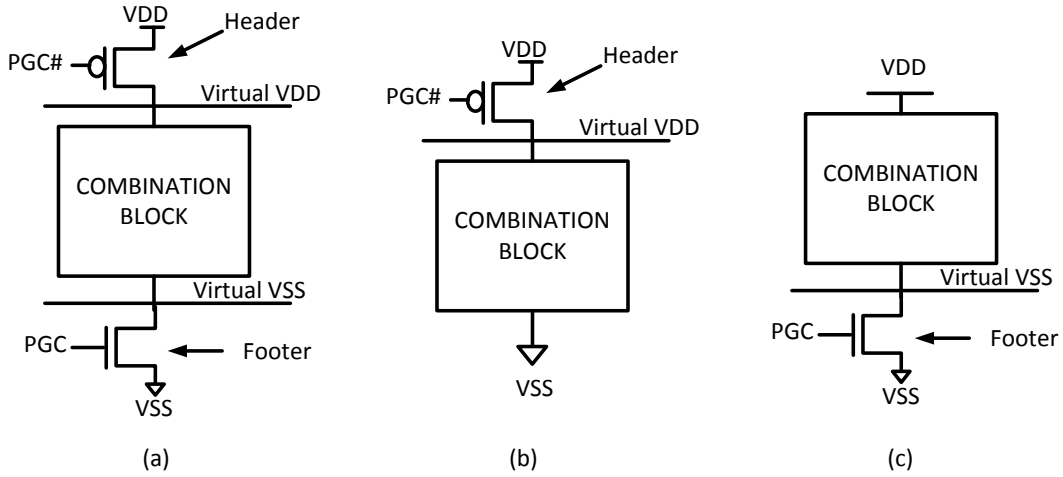
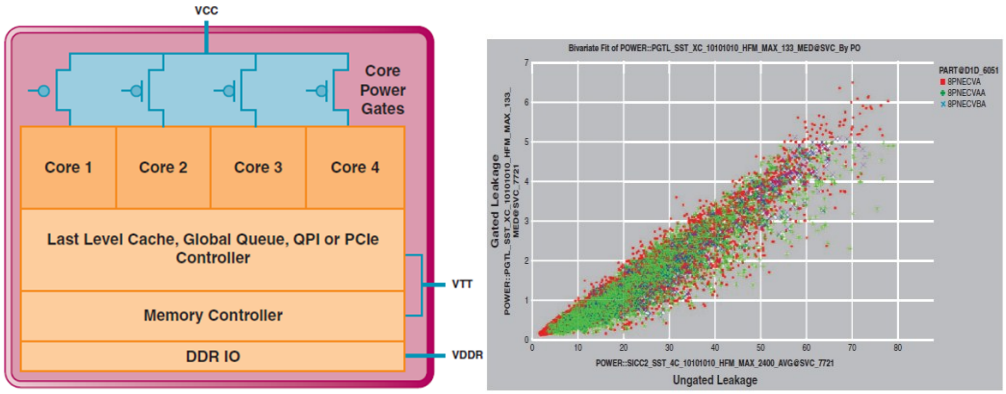


Figure 2.13 Power Gate options (a) Both head and footer (b) power gate with header only (c) power gate with footer only.

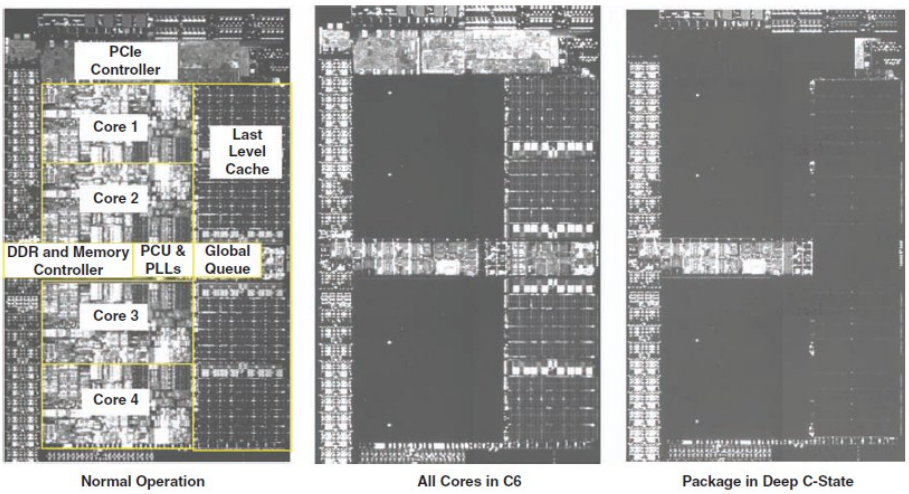
Power gating can be applied at different design hierarchy hierarchies, from a basic logic gate, to a functional logic block, to a whole core. The lower the hierarchy, the more control and effective the leakage saving is. Figure 2.14(a) show power gate used to control the NHM/WSM CPU core [11]. Each core has a power gate to shut it off when in deep idle state. As shown in Figure 2.14(b), a stack factor of > 10 is observed between gated and un-gated core leakage. Figure 2.15 shows a die photo of a power gated NHM based core in idle state

Power gating has a dynamic power and latency cost associated with entering and exiting the power gated state. There is also an area overhead as the power gate devices have to be big enough to supply the current required to drive the gated logic. Power gate implementation therefore involves typical dynamic power, area, and leakage tradeoff. The more granular and lower hierarchy the power gating is, the less the wakeup latency and dynamic power overhead.



[S. Gunther ITJ'10]

Figure 2.14 (a) Power gating scheme of NHM/WSM core (b) Leakage of ungated vs. gated core show 10X core leakage reduction with power gating [11].



[S. Gunther ITJ'10]

Figure 2.15 Die photo of Intel Lynnfield (based on NHM) power gated core [12] in idle state show core shut off during power gated mode.

2.4.1.3 Logic Static Probability and Intrinsic Stack-Forcing

Stacking effect can also be achieved by using the native logic stacking without an additional stacking headers or footers. It is of note that any series connected logic gate (e.g. NAND/NOR) has an inherent stacking condition when more than one of the series devices is turned OFF at the same time. A 2-input NAND gate is intrinsically stacked when both inputs are "0" (series NMOS devices stacked). Similarly a NOR gate is stacked when both inputs are "1" (series PMOS devices). This fact can be used to reduce leakage by state-forcing logic gate inputs to stack-forcing conditions.

Consider a NAND gate with one of the inputs at "0". There are three possible leakage conditions shown in Figure 2.16. Let's assume the leakage current of an equivalent inverter of same size as I_{inv} and the leakage of the NAND as I_{nand}

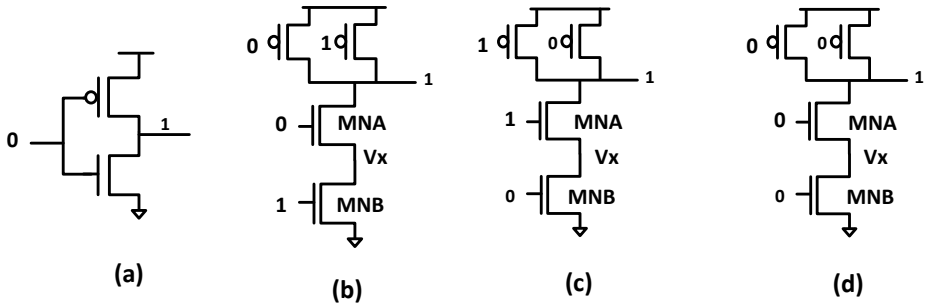


Figure 2.16 NAND gate leakage conditions: (a) reference inverter (b) top device turned OFF (c) bottom device OFF (d) both top and bottom devices OFF

Case 1:

A = 0, B=1. The lower device, *MNB*, is fully ON and the intermediate node $V_x = 0$

$$I_{nand} = I_{inv} \tag{2.25}$$

Case 2:

A = 1, B=0. A voltage drop of $\sim V_{TH}$ develops across device MNA because NMOS cannot pass a full "1". The internal node $V_x = V_{DD} - V_{TH}$

$$I_{nand} = \sim 0.6 I_{inv} \tag{2.26}$$

Case 3:

A=0, B=0. The internal node settles at V_x which reverse biases *MNA* resulting in full stacking

$$I_{nand} = \sim 0.05 I_{inv} \tag{2.27}$$

As illustrated, a series connected logic gate leakage can vary given the same output state depending on specific input state device stacking order.. This knowledge can be used in circuit design to reduce leakage. Leakage can be reduced as shown by connecting low static probability input to the bottom device in an NMOS stack (high SP for top device for PMOS stack). This is commonly encountered when a logic gate has one of its inputs as an enable. It is therefore more power efficient to connect the enable to the bottom device of a NAND and top device of a NOR gate for leakage reduction.

2.4.2 Dynamic Power

2.4.2.1 Switching Power

Power is dissipated in charging and discharging capacitive load during signal transition (from $0 \rightarrow 1$ or $1 \rightarrow 0$).

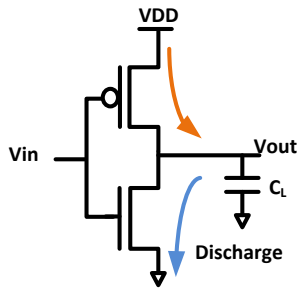


Figure 2.17 Power is dissipated in charging and discharging the load capacitance.

$$P_{dynamic} = C \times V_{dd} \times V_{swing} \times f \times AF \quad (2.28)$$

$$P_{dynamic} = C_{dyn} \times V \times V_{swing} \times f \quad (2.29)$$

Where: $C_{dyn} = C \times AF$ (2.30)

C_{dyn} is the switching capacitance, C of the node capacitance, V_{dd} is the supply voltage, V_{swing} is the swing voltage, AF is the switching activity factor, f is the switching frequency.

At the circuit design level C_{dyn} and V_{swing} are the commonly available knob for improving the dynamic power as the supply voltage and frequency is set at the system level. In addition most static CMOS logic are large signals and undergo full rail-to-rail transition where $V_{swing} = V_{dd}$, further reducing available knob to only C_{dyn}. C_{dyn} can be reduced by reducing load capacitance and switching activity.

2.4.2.2 Reduced Voltage - Threshold Offset Swing

Figure 2.18 shows an NMOS device during a transmission of logic 1. When a voltage $V_{in} = V_{DD}$ is applied to the drain, the output source capacitance is charged slowly. Since both the gate and drain are

connected to V_{DD}, $V_{GS} = V_{DS} = V_{DD} - V_{out}$. As V_{out} increases, V_{GS} and V_{DS} decrease reducing both the channel charge density and drain-source electric fields. This reduces the drive strength and increases the time to charge output capacitance, slowing down the device.

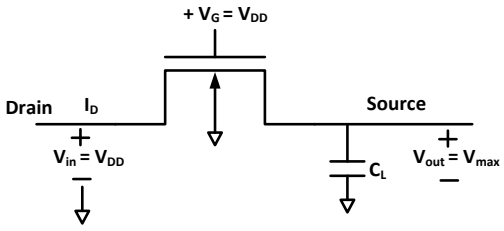


Figure 2.18 MNOS device transmission of logic 1

For NMOS conduction to occur:

$$V_{GS} = V_G - V_S > V_{TH} \tag{2.31}$$

A continuous conduction between the source and the drain requires a minimum voltage difference between gate and source $Min(V_{gsn}) = V_{TH}$ to maintain an inversion layer.

At equilibrium, the maximum voltage of V_{out} is given by [5]:

$$V_{out} = V_{max} = V_G - Min(V_{GS}) = V_{DD} - V_{TH} \tag{2.32}$$

Thus an NMOS device cannot fully pass logic 1. V_{max} is also modulated by body bias effect.

At V_{max}, $V_{SB} = V_{out} = V_{max}$

From equation 2.6 and 2.32

$$V_{max} = V_{DD} - \left[V_{TH0} + \gamma\left(\sqrt{|2\phi_B| + V_{max}} - \sqrt{|2\phi_B|}\right)\right] \tag{2.33}$$

$$V_{max} = (V_{DD} - V_{TH0}) - \gamma\left(\sqrt{|2\phi_B| + V_{max}} - \sqrt{|2\phi_B|}\right) \tag{2.34}$$

The voltage loss of ΔV ($@V_{SB} = V_{out}$) is given by [5]

$$\Delta V = V_{TH0} + \gamma\left(\sqrt{|2\phi_B| + V_{max}} - \sqrt{|2\phi_B|}\right) \tag{2.35}$$

$$V_{max} = V_{DD} - \Delta V \tag{2.36}$$

The ΔV voltage loss property of an NMOS device during logic "1" transmission can be used to reduce the signal voltage swing, V_{swing}, for dynamic power. This can also be used to control device leakage by using a reduced voltage source (NMOS connected to V_{DD})

2.4.2.3 Activity Factor And Logic Dependency

The activity of a logic gate is a function of logic type, input signal residencies, and input waveform (time domain). As shown in Figure 2.19, given the same input activity, the output AF depends on the logic type. For example the AF of a NOR and NAND gate output are 0.75 and 0 respectively given the same input activities A=0.5, and B=0.25. The output AF is also a function of the input waveform as shown in Figure 2.19(e). With the same input AF, the output AF varies depending on the input waveform time alignment. As shown, the NAND gate output AF changes from 0 to 0.5 given the same input AF of A=0.5, B=0.25.

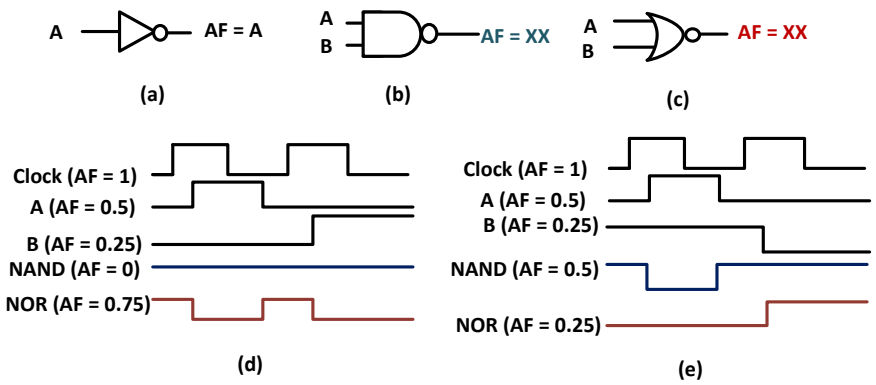


Figure 2.19 Active power logic type and timing dependency. The switching activity depends on interaction of input timing and logic gate type.

2.4.2.4 Short Circuit

Short circuit occurs during signal transition when both PMOS and NMOS are simultaneously ON. This results in a direct current (DC) path between supply voltage VDD and ground Figure 2.20(a). The duration of short circuit is a function of the input slope. A fast input ensures that both NMOS and PMOS devices are turned ON or OFF quickly with minimum ON overlap between the two devices. A slow input slope results in higher short circuit due to increased NMOS-PMOS ON overlap window. The maximum short-circuit current occurs when a fast input slope is combined with a slow output slope. As exemplified in Figure 2.20(c), as the fast inverter output is discharged ($1 \rightarrow 0$), the PMOS

drain-to-source voltage, V_{DS} , increases. Since the slower input signal has not completely transitioned from $0 \rightarrow 1$ the PMOS will still be ON ($V_{GSp} > V_{THp}$) at the same time that the NMOS has turned ON ($V_{GSn} > V_{THn}$) to discharge the output to "0". The increasing PMOS V_{DS} increases the short circuit current through the PMOS.

Ideally a slower output slope is desirable for short circuit reduction. However in addition to impact on logic delay, a slow output slope at one stage will increase the short circuit of the next stage it drives. An optimal short circuit design point is a matched input and output slope of all stage which is impractical in real design. Typically a slope range is set instead for all logic stages to optimize both logic delay and short-circuit.

Short circuit decreases as voltage is reduced and the PMOS-NMOS ON overlap widow reduces as a consequence. At very low supply voltage when $V_{dd} < V_{tn} + |V_{tp}|$, the short circuit is approximately zero as simultaneous conduction of PMOS and NMOS is almost non-existence under this condition.

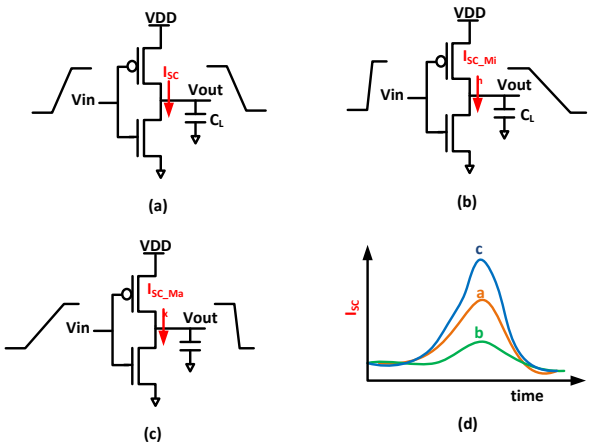


Figure 2.20 Short circuit input and output slope conditions. (a) equal input and output slope , (b) fast input and slow output (c) slow input and fast output (d) short circuit higher for slow input and fast output.

2.4.2.5 Glitch

Glitch occurs when a temporary logic state causes logic output to transition through intermediate states before settling at a final logic state. In practice, all the inputs to a multi-gate logic do not arrive at the same time due to different path lengths. Hence the static logic output will transition through numerous intermediate states before settling at the final logic state when all inputs are available.

Figure 2.21 shows glitches at NAND output due to misalignment of input A and B waveforms in time domain. The output glitch results in power dissipation of equal magnitude as a normal transition so far as there is full transition. The power dissipated by a glitch at a logic stage is independent of the glitch pulse width. However, the glitch pulse width determines how far down the logic chain it gets propagated before it dies off. A narrow glitch pulse will be killed much faster down the chain than a wide glitch pulse.

Glitch is improved by reducing the differences in arrival times at logic gate input. This is hard to archive in practice in random logic due to varied logic paths lengths but could be control in some circumstances by for example placing setup constraints on signals.

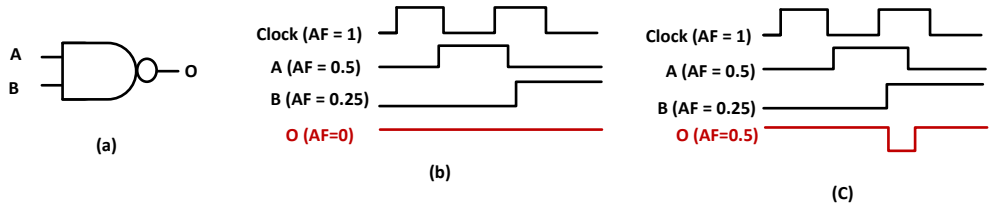


Figure 2.21 Output glitch dependency on input timing waveform. (a) NAND gate (b) Output "O" without glitch and (c) Output "O" with glitch due to differences in input arrival times.

2.4.3 Static Power

Static power results from DC bias current in analogy circuits such as sense amplifiers, voltage converters, voltage regulator and IO logic.

.

2.5 REFERENCES

[1] Neil H.E. Weste, Kamran Eshraghian: "Principles of CMOS VLSI Design – A System Perspective" *2nd Edition, Addison-Wesley, 1993*

[2] S.M. Sze, Kwok K Ng "Physics of Semiconductor Devices", 3rd Edition, Wiley, 2007

[3] C. Auth et al., "A 22 A 22nm High Performance and Low-Power CMOS Technology Featuring Fully-Depleted Tri-Gate Transistors, Self-Aligned Contacts and High Density MIM Capacitors" *Symp. On VLSI Tech. Dig. Of Tech. Paper, 2012, pg 131-132*

[4] Packan et al., "High Performance 32nm Logic Technology Featuring
2nd Generation High-k + Metal Gate Transistors", *IEDM 2009, pp 1-4*

[5] John P. Uyemura "CMOS Logic Circuit Design" *Kluwer academic publishers, 2002*

[6] Jan Rabaey "Low Power Design Essentials" *Springer, 2009*

[7] K. Roy et al., "Leakage Current Mechanisms and Leakage Techniques in Deep-Submicron CMOS Circuits" *proceedings of IEEE vol. 91, No. 2, February 2003, pp 305-327*

[8] Auth et al., " 45nm High-k + metal gate strain-enhanced transistors" *Symp. on VLSI Tech., 2008, pp 128-129*

[9] Siva Narendra et al., "Full-Chip Subthreshold Leakage Power Prediction and Reduction Techniques for Sub-0.18-um CMOS" *IEEE JSSC, Vol 39, No.2, Feb 2004*

[10] S. Narendra, S. Borkar, V. De, D. Antoniadis, A. Chandrakasan, "Scaling of Stack Effect and its Application for Leakage Reduction", *ISLPED, August 6-7,2001*

[11] S. Gunther et al., " Energy-Efficient Computing: Power Management System on the Nehalem Family of Processors" *Intel Technology Journal, 2010, vol 14, pp 50-65*

[12] Martin Dixon et al. "The Next-Generation Intel Core Microarchitecture" *Intel Technology Journal, 2010, vol 14, pp 8-27*

3 BACKGROUND – REGISTER FILE MEMORIES

3.1 MEMORY ELEMENT

Memories are fundamental storage element ubiquitous on CPU for among others holding program/data for execution, storing execution results, for pipeline buffering and state retention, as control registers, and for table lookup. A typical CPU has multiple memory hierarchies characterized by their architectural functionality, access time, and physical size. This requires multiple memory types optimized for the specific requirement of each hierarchy. Table 3.1 show the most common memory types on today's high performance CPU/SoC die.

Table 3.1 Common memory types on a CPU/SoC die

Memory Element	Storage Capacity	Area	Density	Access Time	Power	Design Complexity	Min Vcc
Latch/Flop	Small	Large	Low	Fast	High	Low	Low
Register File	Medium	Medium	Medium	Fast	High	Medium	Medium
ROM	Medium	Medium	Medium	Fast	High	Medium	Medium
SRAM	Large	Small	High	Slow	Low-Medium	High	High
DRAM	X-Large	X-Small	X-High	X-Slow	X-Low	High	High

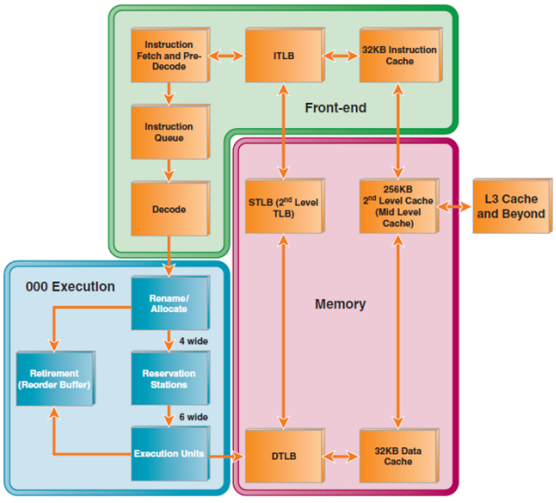
The most widely used storage elements on a modern CPU are latches/flip-flops, register files, read only memories (ROMS), static random access memory (SRAM), and in some cases dynamic random access memory (DRAM). As more system get integrated into a single die in an SoC, different types of memories are integrated onto the same die as the CPU

Latches and flops are mostly used as temporary storage element at transit point in a pipeline execution stream where the life-cycle of a data could be as short as a single cycle. Latches/flops may also sometimes be used for long term storage of small data size. ROMs are non-volatile memory storage element that are permanently programmed to hold specific data like lookup tables and microcode (ucode) fixes and bypass. Register files and SRAMs are the most common storage elements on a CPU core for long term and large size data storage during program execution.

3.2 REGISTER FILE ARCHITECTURAL OVERVIEW

The basic life-cycle of an instruction through the processor core is fetch, decode, execute and store. Registers are fundamental to program execution for storing both operands and results of instruction/data. These may be general purpose registers or specialized registers. The high-level block diagram of IA 45/32nm NHM/WSM core is shown in Figure 3.1[1]. The front-end contains the instruction fetch and decode blocks that deliver micro-operation (μops) instructions to the execution block. The execution unit is then responsible for instruction scheduling, resource allocation, register renaming, re-ordering, and execution. Finally, the memory block which contains multiple levels of memory hierarchies is responsible for the instruction and data load and store operations.

In each of these blocks, register files are the heart of their-operations. The instruction and data caches which stores instructions to be executed require fast access and are therefore implemented as register files. Similarly the data cache in the memory unit requires fast register file access (The 2nd /3rd level caches are implemented as SRAM due to their large size and slower latencies requirement). The Out-Of-Order (OOO) execution block is dominated by register files as each key block has registers. Register Files are therefore central to CPU core as they are involved in the most critical operations. Hence their performance is paramount to CPU performance. Furthermore, given their central role in instruction execution, register files are highly active and therefore consume significant power.



[M. Dixon ITJ'10]

Figure 3.1 NHM/WSM high-level functional blocks [1]. Register files are critical component of each block.

3.3 REGISTER FILES DESIGN

RFs are used to implement a wide variety of functional memory blocks such as control registers, execution registers, caches, tag comparators etc. A register file can therefore be characterized by its architectural functionality. RFs are also characterized by the number of read and write ports, their read and write access methods, or by their physical organization. The term register file is therefore sometimes used interchangeable to refer to architectural/logical functionality or circuit implementation. For example architectural registers can also be implemented as latches or flops, while architectural caches can be implemented as register files.

3.3.1 Register File Organization

The conventional organization of an RF array (similar to SRAM/ROM and other arrays) is as shown in Figure 3.2 The main array components are the storage memory block, the access control logic, and input-output (IO) logic. The memory block is organized as a structured 2-dimentional $M \times N$ matrix of entries (rows) and bits (columns). The array physical organization of bits and entries may be different from the architectural/logical bits-entries size definition. The IO logic block consists of data staging and data flow steering logic for writing into and reading from the array. The access control logic consists of decoder and clocking logic that control memory read and write accesses.

The physical orientation of the memory block, vertical or horizontal (Figure 3.2), depends on the data flow. In a horizontal data flow, the memory data path flows from right-to-left or vise-versa. The IOs are therefore located at the left and right while the access control logic is located at top/bottom of the memory block. In the vertical data path flow, the IOs are located at the top and bottom while the access control logic blocks are located at left/right of the memory block. The array data flow and physical orientation dictates which metal layers key signal can be routed. This imposes routing constraints (width, spacing, capacitance) impacting array size, aspect ratio, performance, and power.

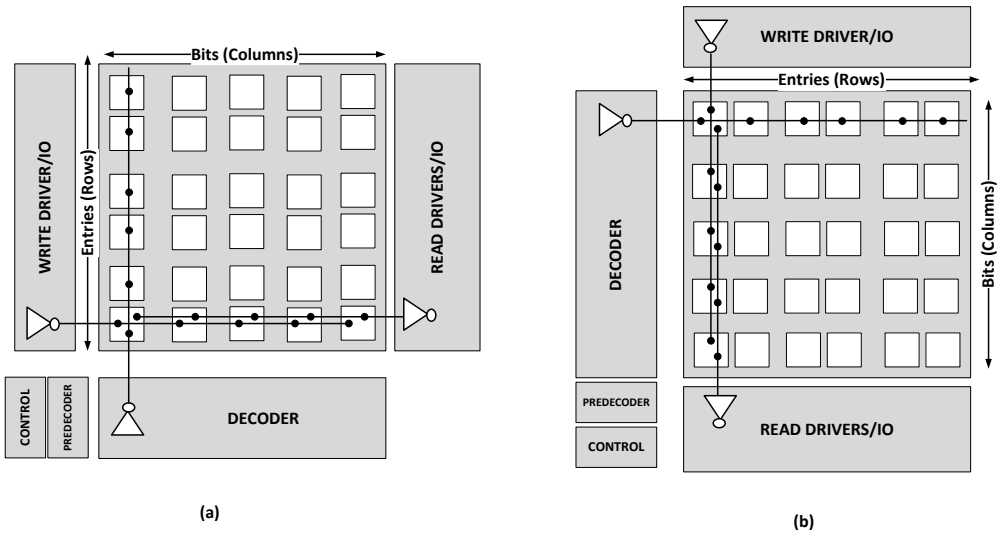


Figure 3.2 RF Array functional block organization (a) horizontal data flow (b) vertical data flow

3.3.2 RF Memory Types

The choice of a register file implementation style is a function of number of architectural and design metrics and tradeoffs: functionality, complexity, area, delay, Vmin, power, noise, reliability, etc. Figure 3.3 shows different types of register file memories (bitcells) commonly used in recent high performance CPU design. In a conventional RF topology, the read and write access ports are decoupled. This allow simultaneous read and write to different entries of the array. This is a critical requirement that typically distinguishes RF from SRAM memories functionally. It is therefore one of the architectural/functional considerations in deciding whether to implement an array as an SRAM or an RF (in addition to size, Vmin, area, performance, frequency of access)

3.3.2.1 Single-Ended

Figure 3.3(a) shows a Single-Ended (SE) RF write bitcell topology where data is directly driven into the bitcell from only one end. A single write data driver and a single write wordline driver are required to write the memory cell. During write "0", a strong "0" is written to "BIT" node from one side of the bitcell through "N1" passgate while the cross-coupled PMOS device "PX" is relied on entirely to complete write a "1" to the "BITX" node. Hence write "0" is slow. During write "1", a strong "0" is written to the "BITX" node through "N2" by the pulldown "NX". Since NMOS is a bad transmitter of logic "1", the cross-coupled PMOS "PB" is relied on to complete write a "1" to "BIT". However "N1" also helps with writing "1" to "BIT" node in the initial write phase transition (when BIT voltage

< V_{GS} - V_{TH}). Write "1" is therefore relatively faster. The advantage of the SE bitcell is that it requires only a single metal track for write bitline signal. It is however slow and has higher Vmin. To improve write "0", a P-Boost device is added to help write "1" to "BITX" through "N2" as in Figure 3.3(b). The SE P-Boost (SEP) essentially uses local write driver inverter in each bitcell. The shared write bitline is still a single track signal but local driver results in higher area overhead.

3.3.2.2 Dual-Ended

A Dual-Write (DE) bitcell is shown in Figure 3.3(c). With DE, a strong write "0" is written from both sides of the array. This implementation is equivalent to SEP except both write bitline drivers are shared by multiple bitcells. Hence the bitcell area overhead is reduced. However, the DE bitcell requires two write bitline tracks. This could impose routing constraints especially for memories with large number of write ports. Dynamic power is also increased due to multiple write bitline transitions during a write. A full transmission gate can be used to ensure strong write "0" and "1" (Figure 3.3(d)). This is useful in low-voltage operation [2]. However this comes with significant area overhead.

3.3.2.3 Set/Reset

An RF bitcell with single sided reset (SR) is shown in Figure 3.3(e). This is a functionality driven memory where a memory reset/set capability is required to clear or initialize the memory content. Thus this can be implemented with both SE/DE/SEP bitcell. In a differential reset both sides of the storage node are reset, improve reset delay. However the additional PMOS device increases bitcell area.

3.3.2.4 Interruptible Latch

An interruptible register file bitcell (INT) is shown in Figure 3.3(f). This is a single-ended write latch with complementary passgate for a strong write "0" and write "1". The write wordline signal (WrWL#) can be generated locally per bitcell or globally and shared across multiple bitcells. This is a typical area and power tradeoff. Local generation reduces routing capacitance, routing congestion, and hence wordline dynamic power. However, this increases cell area which can also lead to increased power. Similarly global wordline can impact area and power in either direction. For example if there are multiple read or write ports, driving two wordline signals can result in bitcell area growth if bitcell is track limited.

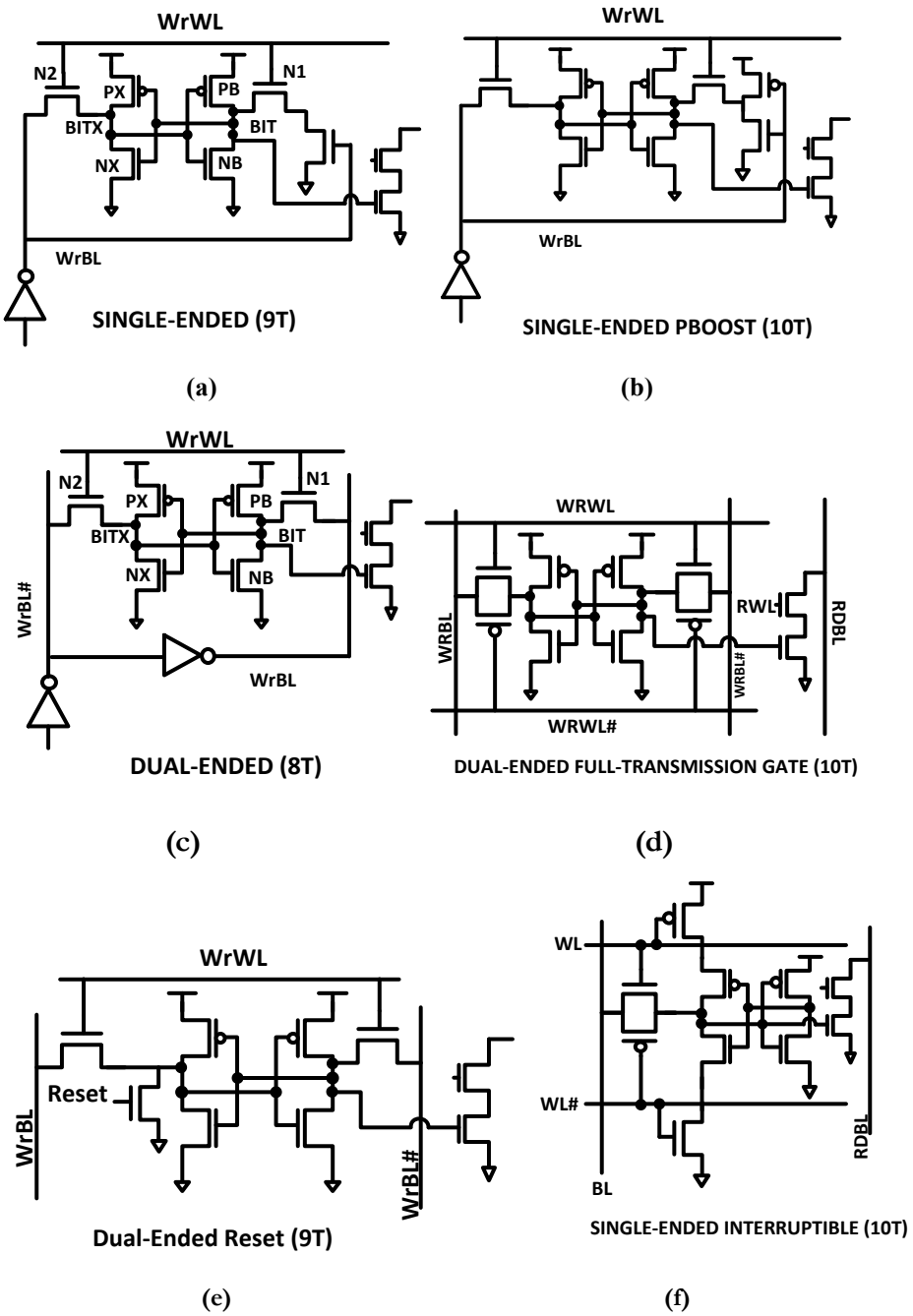
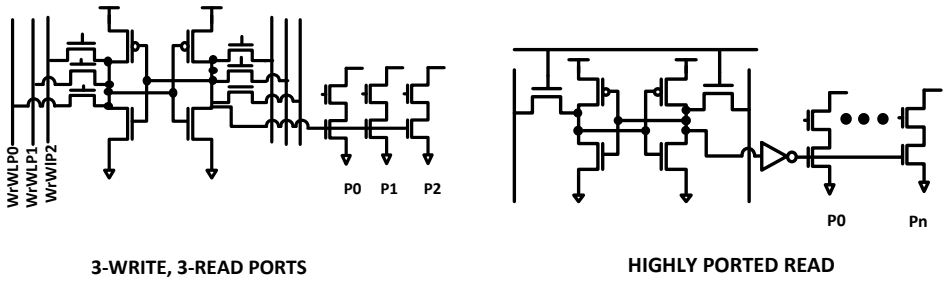


Figure 3.3 Sample register file memory types (a) Single-Ended Write (SE) (b) Single-Ended Write Pboost (SEP) (c) Dual-Ended Write (DE) (d) Dual-Ended Reset (e) Dual-Ended Full transmission gate (f) Single-Ended Interruptible latch

3.3.2.5 Multi-Port

One key functional characteristics of RF is the ability to simultaneously access different entries. Additional read and write ports are added to service multiple request in the same cycle. An example of a 3-write, 3-read port RF is shown in Figure 3.4. A local buffer may be used in a highly ported read bitcell to reduce the RPT capacitive loading on the memory node. This memory node capacitance is critical to writeability.



(b) (b)

Figure 3.4 Multiport bitcell with (a) 3-read, 3-write (b) Highly ported read with local buffer.

As shown from the few RF bitcell examples, the bitcell selection has power, area, and delay implications. The classical choice of area versus performance (power, delay) is a common bitcell design optimization. Making these tradeoffs require assessment of various performance metrics, including schedule and ease of design. Also a particular register files implementation may be favored by the process limitations and constraints – beyond access speed, Vmin, power.

3.3.3 Decoder/Pre-Decoder

Array access is initiated by decoding an encoded address of n bits to 2^n outputs to select a specific array entry. The decoded address is then clocked to generate access wordlines. The selection of decoding schemes depends on the address space size, timing, power, and area constraints. In a full pre-decoding scheme the n bits address is fully decoded into 2^n outputs at the pre-decoder state as shown in Figure 3.5(a). This may be done in multiple stages. The wordline is then simply a clocked version of the fully pre-decoded address. With partial pre-decoding approach, the pre-decoder stage only partially decodes the address with the final decoding integrated into the wordline decoder stage (Figure 3.5(b)) For large address space, this reduces the size of the pre-decoded signal and ease routing constraints.

However, the decoder could be relatively slow due to increase number of devices in series compared to the fully pre-decoded approach. Partial pre-decoding can reduce leakage as it reduces total device width. However, multiple pre-decoded output signals are toggled as the partial pre-decoded outputs are not fully mutually exclusive. Hence, depending on the routing distance from the pre-decoder to the decoder, dynamic power overhead may be significant.

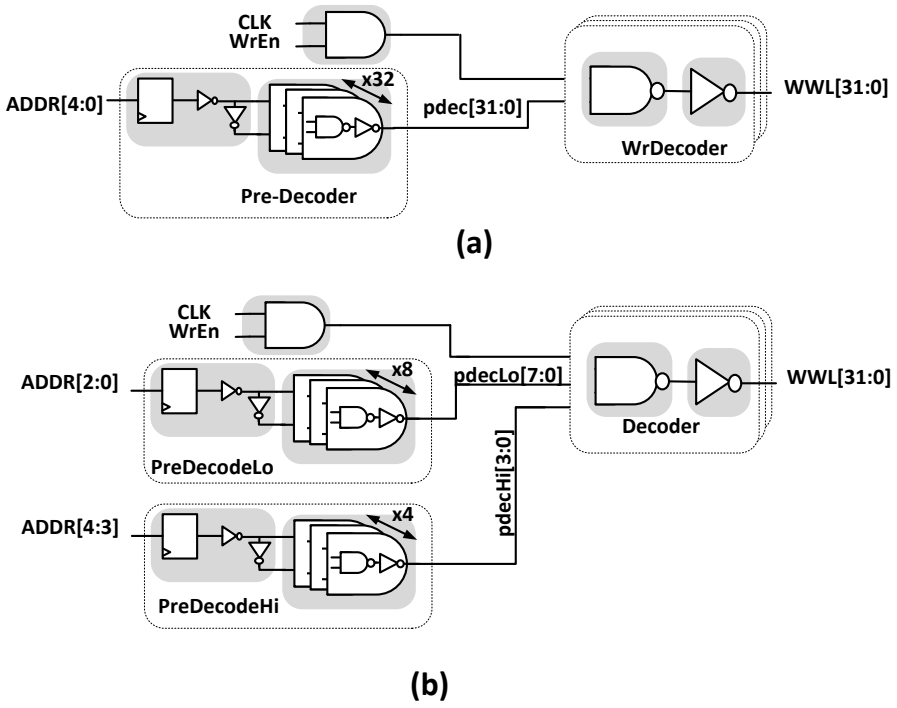


Figure 3.5 Full and partial wordline decoding.

3.3.4 RF Write

A conventional RF write topology is shown in Figure 3.6. New data is staged by a latch or flop and distributed to the array block by a global driver. A local write data driver repeats the data before writing into the bitcell. The data distribution scheme used can vary wildly depending on delay, power, writeability/Vmin and other design tradeoff. Each local write driver is shared by multiple entries with mutually exclusive access, reducing driver overhead. Local bitline is also segmented to reduce the write bitline load. The maximum number of bitcells per segment can typically vary from 8 to 64 bitcells depending on driver size and segmentation strategy. With write multiport, each port will typically have

its own write data distribution and hence local write drivers. In some cases, staging logic/control may be shared between ports if access is mutual exclusive.

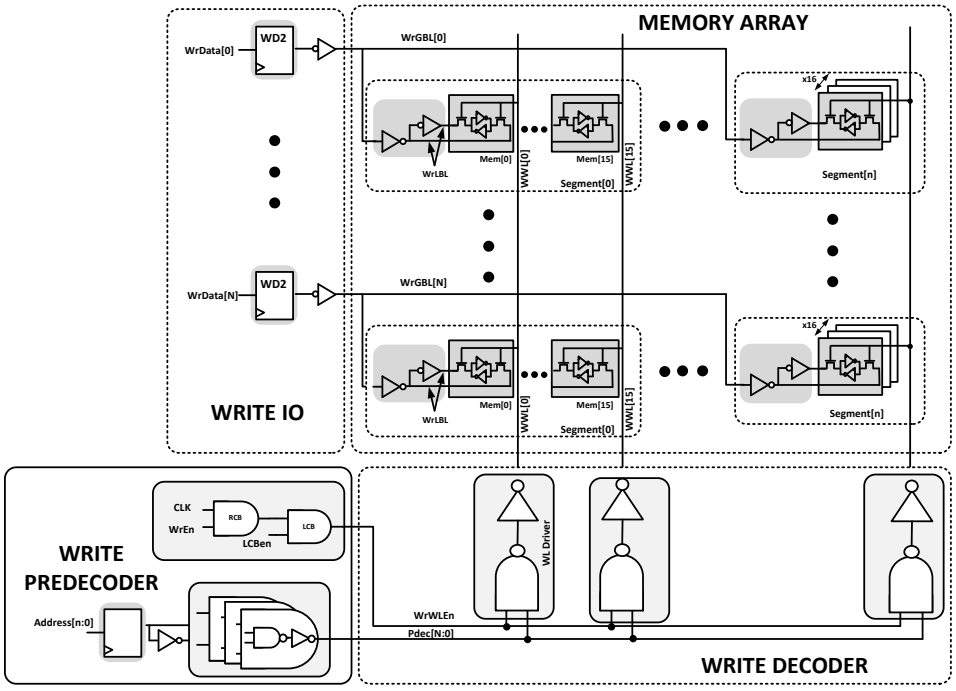


Figure 3.6 Conventional register file write topology

3.3.4.1 DE-Write

The conventional dual-ended write topology relies on writing a strong "0" from either side of the memory nodes (Figure 3.3(c)). When the write data is '0', a strong "0" is written to memory "BIT" node through the "N1" passgate while a weak "1" is passed through "N2" to "BITX". The cross-couple PMOS "PX" completes a strong write "1" to the "BITX" node. When write data is "1", a strong "0" is written though the "N2" pass transistor to "BITX" node, while a weak "1" is written through "N1" to "BIT" node. The cross-coupled PMOS "PB" then completes the strong write "1" to the "BIT" node.

3.3.4.2 RF Write Timing

The write timing sequence is as follows Figure 3.7. A write access address is decoded to select a specific array entry. The write address arrives a phase earlier than the memory access phase. The gated write clocks (RCB/LCB) are activated by a write enable, triggering/initializing write access sequence. The write decoder is then activated by the LCB clock. The write wordline corresponding to the decoded entry select rises, turning ON the bitcell passgates. The data at the bitline input is then written into the memory cell overriding its content.

There are typical three critical paths to an RF write

(i) The address has to be decoded before the write clock is active (address setup)

(ii) The time between the rise and fall edge of the wordline (pulse width) should be large enough to allow complete flipping of the bitcell content (Wordline Write completion)

(iii) When the wordline rises, the time between the arrival of the write bitline data and falling of the wordline (closing of bitcell) should be long enough to enable complete flipping of the bitcell. (Data Write Completion)

Typically, data write completion, (iii), is the worst-case write condition and the write performance limiter. As such the architecture of the write data distribution, the size of drivers, and memory bitcell size are typically influenced by the time it takes to flip the memory content when data arrives at the write driver/bitline input. This has influence on RF power as writeability constrains data driver and memory size.

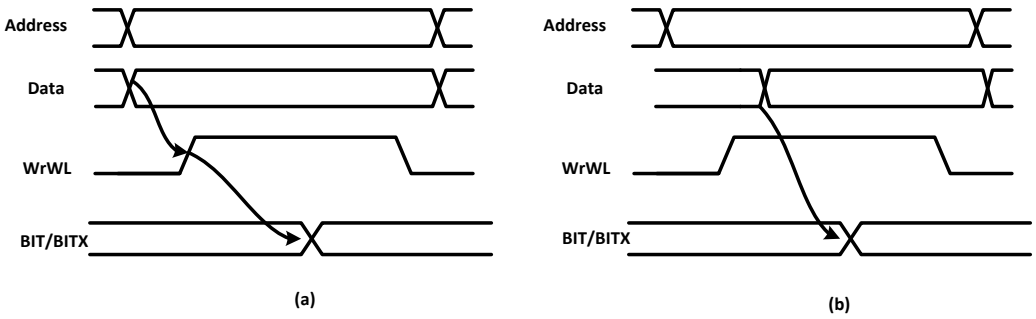


Figure 3.7 RF Write timing waveform (a) Data setup to wordline (b) Data arrives after wordline

3.3.5 RF Read

A conventional RF dynamic read organization is shown in Figure 3.8. The address is decoded to select a distinct array entry of interest. The pre-decoded address is clock gated to generate the read wordline access signals. A read enable initiates data access. The read data path can be implemented with static or dynamic logic. Dynamic read is the preferred read implementation due to its lower area and fast access time compared to static implementation.

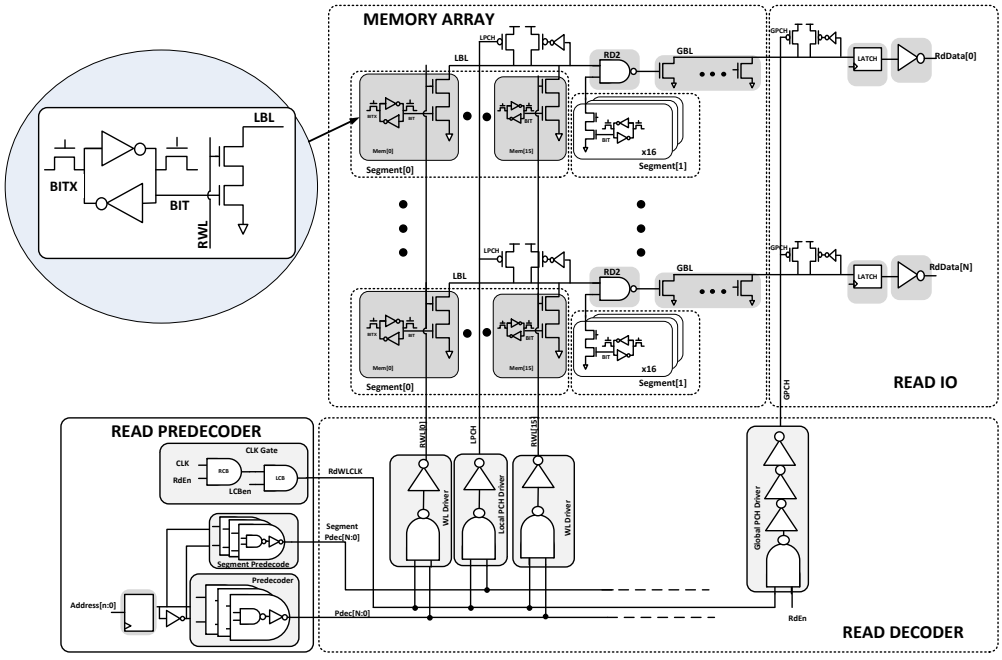


Figure 3.8 Conventional register file read topology

3.3.5.1 Dynamic Read

The read data path of a conventional dynamic/domino RF implementation is shown in Figure 3.9. Read ports (RPTs) from different memory entries are physically connected to form a segment of local bitlines (LBL) in an *Mx1* wide-NOR mux, *M* representing number of entry RPTs. A NAND gate is used to combine two LBL segments. This is further merged at the global bitline (GBL), also implemented with a wide-NOR dynamic logic. In this domino implementation, the LBL and GBLs are precharged to VDD in the OFF state.

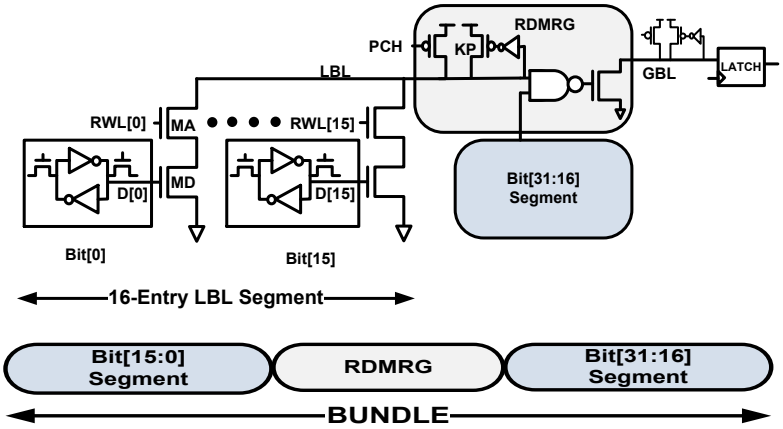


Figure 3.9 Conventional RF dynamic bitline wide-NOR topology with 16-entry per LBL segment and physical organization

3.3.5.2 Dynamic Read Timing

The CONV operation is as follows (Figure 3.10). During read access, the read precharge clock (PCH) goes high and turns OFF the LBL PCH. The RWL initiate read access to a single entry. If the accessed memory data D[0]=1 the LBL is discharged to ground, causing a "domino" chain of evaluation on the path through the NAND, GBL, and latched downstream. If the data D[0]= "0", the LBL retain its precharged value ("1"). The keeper, *KP*, holds the precharged value to prevent false evaluation, withstanding any noise impact from charge sharing, LBL interconnect, propagated noise from the RWL signals, and DC droop from RPT device. The contention between the domino pull-down NMOS and keeper (local bitline and global bitline) is the primary read Vmin limiter in RFs. The keeper needs to be strong enough to hold the precharge value to avoid false discharge from leakage and other noise sources if data is "0". On the other hand it has to be weak enough that the bitline LBL node can be discharged by the Read Port if D[0]=1. The keeper constraint limits Low-Voltage operation of Register Files Read.

There is a short circuit power race between the rising and falling edges of wordline RWL and precharge PCH clock signals. PCH signal should rise to turn OFF the precharging PMOS device before RWL rises to turn ON the RPT NMOS. Conversely at the end of read, the RWL should fall to turn OFF the RPT NMOS before the PCH falls to turn ON the PMOS precharge. Similar short circuit power race exist between the global precharge GPCH and the global bitline NMOS input timing.

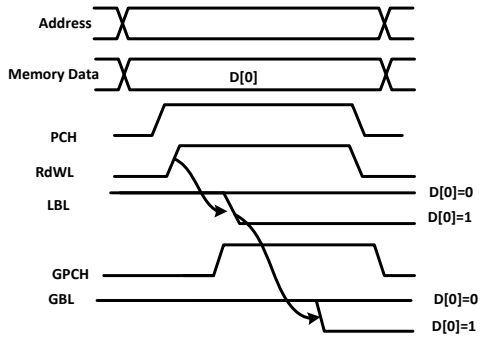


Figure 3.10 Dynamic read timing waveform

3.3.5.3 Static Read

Figure 3.11 Fig shows a static RF read data path implementation. Static read topology is typically used for smaller entry size RFs due to higher delay and area overhead. However, it has the advantages of static wordlines in addition to being contentionless and not susceptible to *Min-Vdd* limitations of a domino read topology.

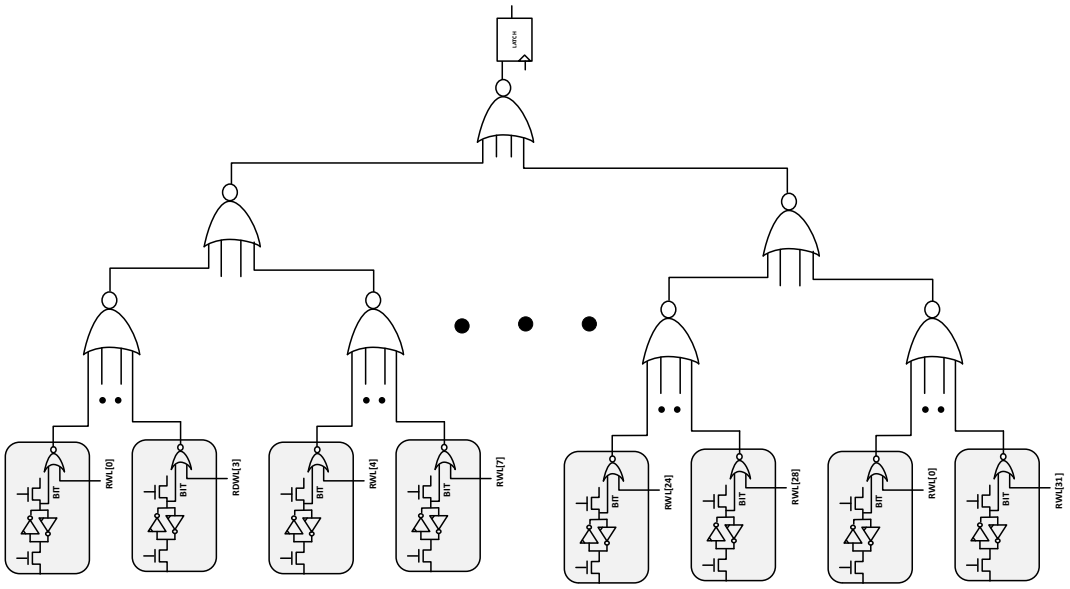


Figure 3.11 Conventional RF static read topology

3.4 REGISTER FILE POWER

3.4.1 Power Distribution

The leakage and dynamic power distribution in all RFs on a 32nm IA core is shown in Figure 3.12. The bitcell leakage which comprises of the memory cell (excluding the read port) is dominant at 39% of total leakage of all RFs. The write data distribution, the local and global write bitline, account for 22% of total RF leakage while the read data path, local and global bitlines, account for 20% leakage. The read and write wordline drivers together account for ~13% of RF leakage. Dynamic power is dominated by write bitline which account for 25% to total dynamic power due to dual-ended write operations.

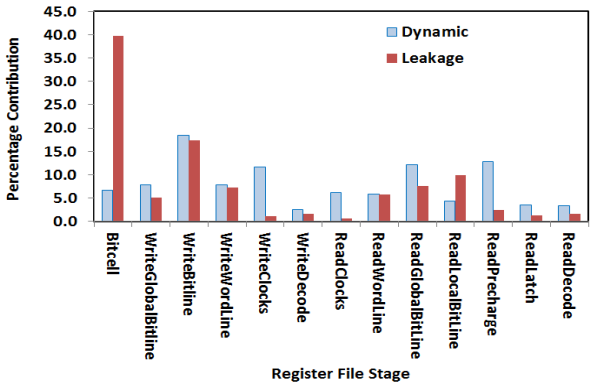


Figure 3.12 Leakage and Dynamic power distribution in 32nm WSM core Register Files

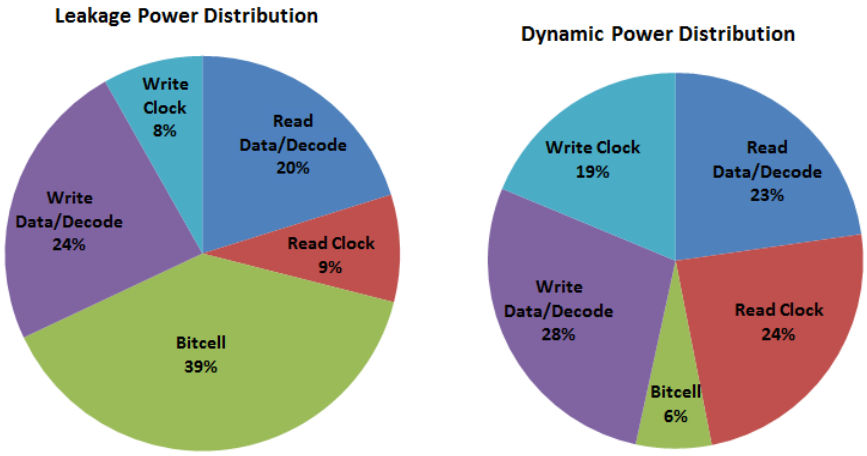


Figure 3.13 Logic operation leakage and dynamic power distribution in a 32nm IA Core.

3.4.2 Memory and Write Power

Figure 3.14 shows the sources of bitcell leakage. In the best case scenario, both the memory node and the corresponding write bitline have same data polarity. The worst case leakage condition occurs when the polarity of the stored memory data is opposite the data at the pass gate input. Since the write bitline is shared by multiple bitcells, this scenario exists any time a different data polarity is written to another entry. When opposite polarity exist across the transfer device, the transfer essentially become a virtual single leaking device. Typical solutions include use of low-leakage (LL) devices on bitcell NMOS pass and retention devices. LL devices however have performance impact that affect writeability and Vmin. Figure 3.14 and Figure 3.15 show the average, min, and max write bitline and memory storage node data static probabilities respectively of various benchmarks in a 22nm Haswell IA Core. Both the data stored in memory and at passgate input are benchmark dependent. These are not known at design time. In chapter 6 we discuss the implication of the memory cell leakage dependencies on stored data and propose way to reduce leakage across the bitcell and the write driver. We also discuss segmentation impact on write data dynamic power and data gating strategies to reduce write dynamic power.

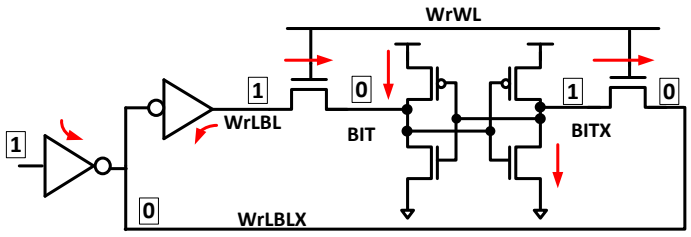


Figure 3.14 Worst case memory leakage occurs when memory and write bitline are in different states.

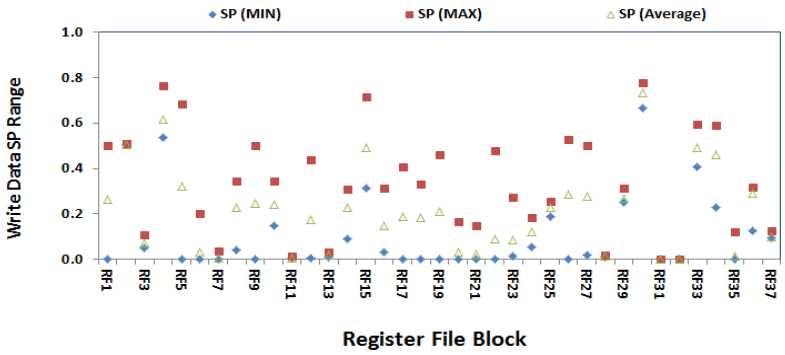


Figure 3.15 The average, minimum, and maximum write data SP of various benchmarks for register files in a 22nm IA core. Though generally most RF on average write a data with polarity=0, this is benchmark dependent.

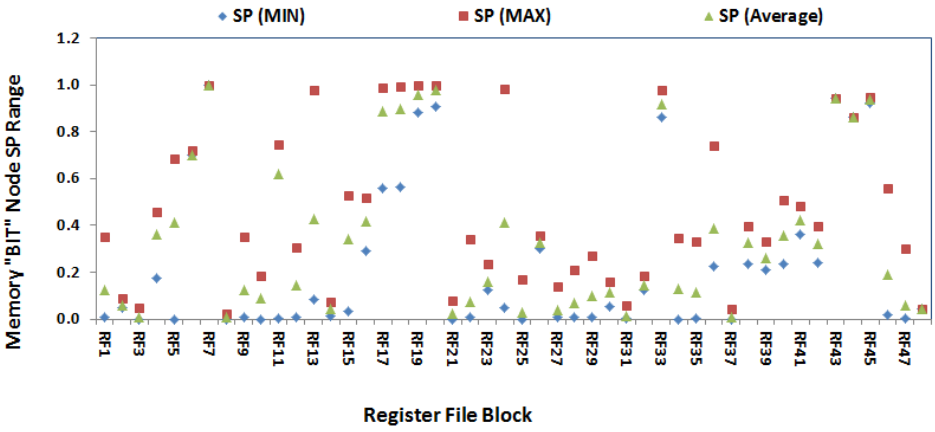


Figure 3.16 The average, minimum, and maximum memory node SP of various benchmarks for register file in a 22nm IA core. Though generally on average most RFs store more "0" than "1", this is benchmark dependent.

3.4.3 Read Power

The leakage scenarios of the dynamic read port are shown in Figure 3.17. The best case leakage condition occurs when the memory is storing "0" on the node connected to the read port (RPT). Under this condition, a natural stacking occurs in the standby mode when the wordline is "0". The worst case read bitline leakage condition occurs when the memory is storing a "1" on the RPT node Under worst-case leakage condition, the LBL RPTs will be leaking through a single device, *MA*. The leakage is usually reduced by using a low-leakage device on the read port. The worst-case leakage condition during read operation results in LBL DC droop which can lead to functional failure if the keeper is not strong enough. To address this, the keeper is sized to meet a DC droop constraint. Larger keeper size however reduces LBL fall delay. Since the content of the array is unknown at the time of design, different benchmarks are analyzed to determine most probable polarity of the memory content. The read port is then connected to the bitcell node with lowest static probability i.e. most probable to store a "0". Typical more "0s" are stored in memory than "1". Hence the read port is typically connected to the "BIT" node for power reasons. This also reduces the read dynamic power. The precharging of the bitline after read "1" bitline discharge incurs a dynamic power overhead. Connecting the RPT to the memory node with the highest probability of storing a "0" eliminates the precharge overhead since the bitline does not discharge during read "0".

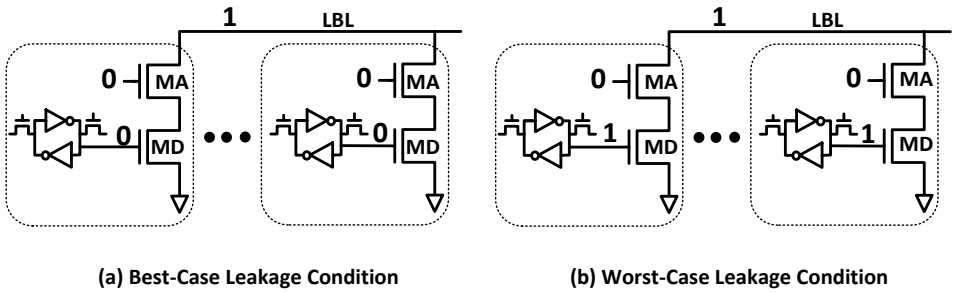


Figure 3.17 Worst case leakage conditions for dynamic read bitline occurs when the memory node connected to the read port is storing a "1"

3.4.4 Multiple–Threshold Device Usage

One of the widely used leakage reduction techniques in RF is the usage of low-leakage (LL) device to take advantage of the process technology. The low-leakage device can be achieved through either long channel devices or high-Vt devices. Multiple-Vt devices are common on modern process technology nodes. The use of low-leakage insertion in RF takes advantage of the structural implementation of RF and the state probability of the various RF nodes. Due to the clocking and domino implementation of RF for example, the inactive leakage states of all read path nodes from wordline to GBL are known at design time. The idea then is to take advantage of the known state probabilities to selectively insert LL devices on leaking devices along the path. For example the wordline is "0" when inactive, hence the wordline driver PMOS will be leaking. The wordline driver PMOS can therefore use a low-leakage device. Similarly, the LBL and GBL are precharged in off-state so the RPTs and GBL NMOS pulldown will be leaking and can be made LL.

The drawback of this technique is that the OFF devices are also on the critical timing path. Low-Leakage devices are comparatively slower. For iso-delay to nominal device therefore, the LL device has to be sized larger thus requiring additional area and increasing Cdyn. The dynamic power drawbacks are typically traded off against leakage reduction benefit of LL devices due dynamic power activity dependency. In order words, the extent of dynamic power overhead is variable since it depends on block activity while the LL device leakage saving is independent of activity and therefor more predictable.

3.4.5 Decoder Power

Stack forcing is one way of reducing wordline drive leakage. A PMOS device is connected in series with the inverter PMOS to enforce stacking (Figure 3.18). The stack control signal is usually an address signal. The stacking invariably incurs delay penalty as the drive strength is reduced. To reduce the impact of delay degradation, the header device is shared across multiple drivers. Since only one driver will be active at a time, sharing minimizes the total required header size. The leakage reduction with wordline stacking is greater than 90%. A shared power gate device size of 4X the reference PMOS device size reduces delay overhead to <20%.

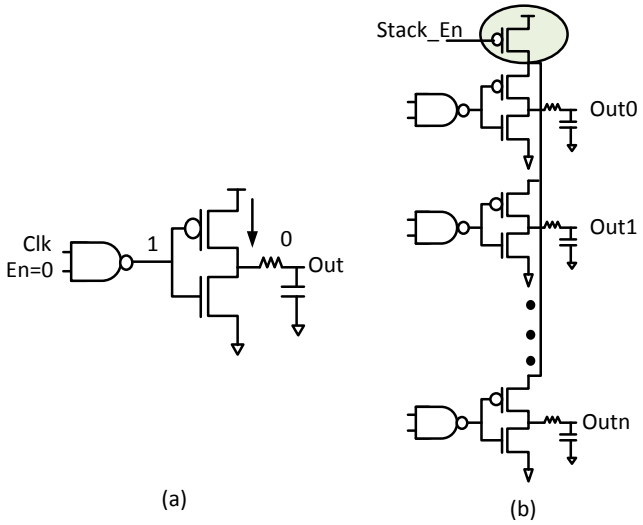


Figure 3.18 Wordline driver local power gate. The power gate device is shared by multiple drivers since only one wordline driver is active at a time, reducing power gate overhead.

3.4.6 Clock/Address Gating

RF clock power can be reduced by gating clock distribution at various levels. The generic gating signal is the read enable signal. There are 3 levels of gating, regional clock buffer (RCB), local clock buffer (LCB), and wordline NAND/NOR (Figure 3.19). The choice of gating stage using the read enable is primarily a function of timing criticality. Ideally the best option is to gate at the RCB level. However, if

the enable signal is timing constrained then the LCB level is the next option. The wordline NAND level is the last resort. The address decoding can also be data gated by read enable to prevent wasteful decoding anytime any of the address signals changes state even when actual access it not needed. This is useful when the unique enable is not available to gate the address latch. Data gating also has a leakage benefit of forcing all the inputs of the decoder NAND to "0" state in an inactive state, enforcing stacking on all instances.

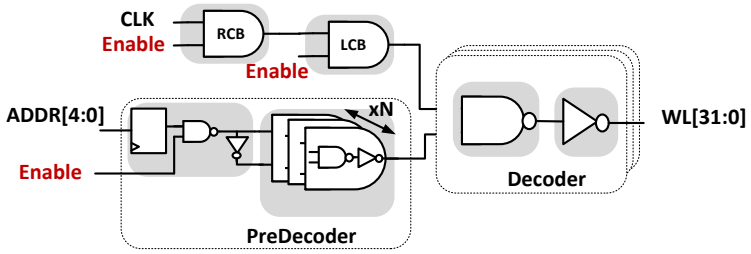


Figure 3.19 Clock and address gating for RF wordline stages

In a more aggressive clock gating, the MSB of the address is used to gate the LCB to generate signal for either the upper entries or lower entries Figure 3.20. The LCB clock output fanout could be high due to the number of driven NAND gates. Clock loading at this stage is gate cap dominated, thus splitting the receiving gate into multiple segments helps Cdyn even though extra routing is required.

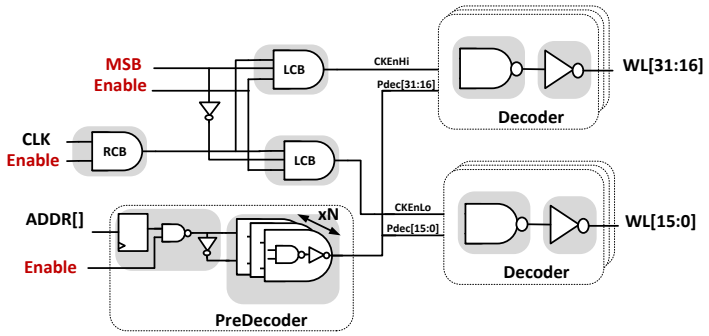


Figure 3.20 Using MSB signal to gate LCB to reduce clock loading

3.5 REFERENCE

REFERENCE

[1] Martin Dixon et al. "The Next-Generation Intel Core Microarchitecture" *Intel Technology Journal, 2010, vol 14, pp 8-27*

[2] Amit Agarwal et al., "A 32nm 8.3 GHz 64-entry x 32b Variation Tolerant Near-Threshold Voltage Register File," *Symp. On VLSI Circuit/Technical Digest of Technical Papers, 2010.*

4 NEAR-/SUB-THRESHOLD REGISTER FILES/SRAM DESIGN

***Abstract*—To improve chip area density, memory blocks such as SRAMs and Register Files (RF) are built with smallest possible device sizes. Memories are therefore most susceptible to increased variation effects as process technologies continue to scale critical device dimensions. Consequently SRAM/RFs have traditionally been the bottleneck to low voltage operation. This work explores the challenges and limitation of Near-/Sub-Threshold RF/SRAM operations. We review current design techniques employed to reduce their minimum operating voltage (V_{min}). We compare different topologies and show that in 22nm, with 3σ variation, an interruptible latch based design can reduce an RF write V_{min} by 200mV and delay by up to 3X when compared to conventional 8T dual-ended bitcell. We also propose using a tri-state read ports to improve RF Read V_{min} by 300mW and up to 2X delay improvement over conventional read.**

Index Terms*—Sub-Threshold, Near-Threshold, SRAM, Register File, *Vmin,

4.1 INTRODUCTION

Processors operating at Near-/Sub-threshold supply voltages have previously been confined to low performance and niche markets such as sensors and medical devices due to frequency limitation. However, the advent of multi-core computing (power now bottleneck), coupled with the need for low power form factors such as tablets and smart phones, has renewed interest in pushing mainstream computing into low operating voltage realm.

Dynamic Voltage and Frequency Scaling (DVFS) has been the dominant technique used to modulate processor operating power and performance based on system demand and power envelop. However, supply voltage scaling is limited by V_{min} , the lowest voltage at which circuit continue to functionally operate correctly. Memory cells, SRAMs and Register Files (RF), are the first to fail as supply voltage is lowered and therefore are the V_{min} limiters. This is because the large number of memory cells on a typical chip necessitates that they are densely designed with the smallest possible device sizes to reduce cost and power. Small devices however, are more susceptible to functional failure in low voltages due to increased variation and reduced drive current, primarily from random dopant fluctuation (RDF).

Typically, high performance circuits are operated in safe super-threshold region (supply voltage well above the device threshold) where the voltage overdrive ($V_{gs} - V_{th}$) is high (Figure 4.1). As the supply voltage (V_{dd}) is scaled down, leakage and dynamic power are reduced.

$$Power = P_{dynamic} + P_{leakage} \quad 4.1$$

$$Power = C \times V_{dd}^2 \times A \times f + I_{leak} \times V_{dd} \quad 4.2$$

Where C is the switching capacitance, A is the activity factor, f is the frequency, and I_{leak} is the leakage current.

Performance however degrades as voltage is scaled down, linearly initially in the super-threshold but exponentially as we enter sub-threshold region (Figure 4.1). As a result, the total energy required to execute an instruction, which unlike power is a function of time (Equation 4.4.), degrades significantly in sub-threshold.

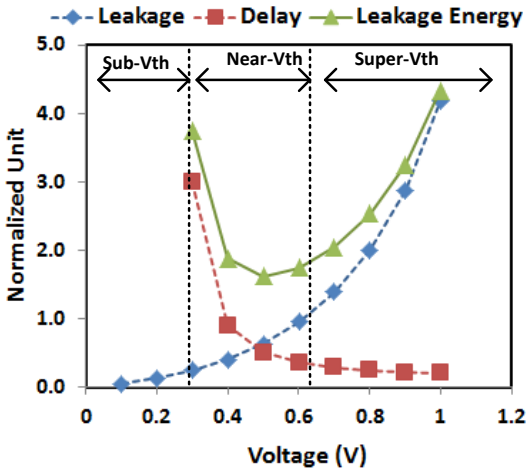


Figure 4.1 Minimum Energy at Near-Vth. Exponential delay in sub-threshold

$$Energy = E_{dynamic} + E_{leakage} \quad (4.3)$$

$$Energy = C \times V_{dd}^2 \times A + I_{leak} \times V_{dd} \times T_d \quad (4.4)$$

where T_d is the circuit delay. Due to leakage dependency on circuit delay, minimum energy in Sub-threshold is set by leakage [1][2]. There exists an optimal operating voltage at which the total energy is minimal (Figure 1). This is found to be around near-threshold region [1][3]. Therefore operating in Near-threshold region, Near-Threshold Computing (NTC), becomes attractive for mainstream computing.

Operating at Near-/Sub-threshold however has significant design challenges and constraints. In addition to performance loss, variation in these regions is exponentially important. There is an order of magnitude increased sensitivity to temperature, threshold voltage (V_{th}), and device channel length (L_e) at low V_{dd} operation. Adding design margin for variability or just using bigger device sizes is costly (area, power) and not sufficient in itself. Therefore, there is a need for architecture and circuit techniques complementing device level process technology to enable voltage scaling into Near-/Sub-threshold. Architectural solutions such as designing only a fraction of SRAM for NTC operation while the rest of the cache is designed for only Super-threshold operation has been proposed[4]. In [3] a system level approach where each core of a multi-core operates at different frequency, with the view that by law of averages, some of the cores will be functional at low voltages is proposed. ECC and redundancy can also extend V_{min} limit by a process generation [5]

This chapter discuses RF/SRAM challenges in Near-/Sub-threshold operation with emphasis placed on RF Near-Threshold operations. We review current circuit level techniques such as read and write assists that are commonly used to reduce V_{min} in section III, IV. We also review SRAM/RF candidate topologies for Vmin reduction in section V. Simulation results are provided in section VI.

We propose using two contention free schemes for low voltage operation. An interruptible latch is shown to reduce write Vmin by 200mV and improves delay by up to 3X in the presence of variation when compare to the conventional 6T/8T SRAM/RF bitcell. To reduce RF read V_{min}, we propose using a tri-state to eliminate keeper contention. The proposed tri-state reduces read V_{min} by 300mW and improve delay by up to 2X at slow process corner (RSSF).

4.2 VARIATION AND RELIABILITY

4.2.1 Variation

V_{th} variation increases as devices get smaller (Equation 4.5. [5]). σ_{vt} has thus been increasing over process generations due to device critical dimension scaling.

$$\sigma_{vt} \propto \frac{EOT}{\sqrt{L_g \times W_g}} \tag{4.5}$$

where EOT is the oxide thickness and L_g, W_g are the device length and width respectively.

In Super-threshold, device On-Current ($I_{on-super}$) has linear dependency on V_{th} and V_{dd} (Equation 4.6). Sub-threshold On-Currents (I_{on-sub}) on the other hand is exponentially dependent on V_{th} and V_{dd} (Equation 4.7). Hence there is a high Sub-threshold sensitivity to V_{th} and V_{dd}. Near-threshold exhibits a hybrid of Supper- and Sub-threshold characteristics. Near-Threshold exponential sensitivity to V_{th}, V_{dd}, and temperature results from greater impact of Sub-threshold current on near-threshold drive current[6][7]. Short channel FET sensitivity of $I_{on-super}$ to V_{th} and V_{dd} is given in [2] as:

$$I_{on-super} = \frac{g_{msat}}{1 + R_s \times g_{msat}} \times (V_{dd} - V_{th} - V_{PO}) \tag{4.6}$$

Where R_s is the FET source resistance, g_{msat} is the saturated transconductance (a function of L_{eff}, C_{ox}, and carrier saturation velocity), and V_{PO} is the pinch-off voltage .

I_{on-sub} sensitivity to V_{th} and V_{dd} is given in [2] as:

$$I_{on-sub} = \frac{W}{L_{eff}} \times \mu_{eff} \times C_{ox} \times (m-1) \times V_T^2 \times exp\left(\frac{V_{gs} - V_{th}}{m \times V_T}\right) \times \left[1 - exp\left(-\frac{V_{ds}}{V_T}\right)\right] \tag{4.7}$$

where $V_T = kT/q$. V_T is thermal voltage, k is Boltzmann constant, q is the electron charge, L_{eff} is effective gate length, W is the gate width, μ_{eff} is effective mobility, C_{ox} is oxide capacitance, m and is the subthreshold slope factor.

4.2.2 Reliability

Near-threshold has advantages in reliability. Aging resulting from negative bias temperature instability (NBTI) for example is less due to reduced electric field from reduced supply voltage and lower temperature. Similarly electromigration is also reduced by lower temperature and current [3]. However, sub-threshold is very sensitive to the impact of these reliability effects. Hence degradations that occur during super-threshold operation mode have disproportionately high impact when processor is operating in sub-threshold mode. Therefore reliability issues become even more significant in circuits designed to operate across wide range of voltages from superthreshold to subthreshold.

4.3 SRAM/RF DESIGN OVERVIEW

4.3.1 SRAM Topology

A conventional 6T SRAM bitcell with differential bitline sensing is shown in Figure 4.2(a). Both read and write share the same wordline (WL) and bitline (BL). Due to the jam feedback structure (back-to-back inverter), the ability to write successfully (writeability) depend on the relative strength of the write driver (WD), access passgate (PG) and pull-up (PU) devices. This contention is the source of SRAM write V_{min} limit. In addition column interleaving introduces half-select problem in SRAM. During write, "WL" activates all bitcells in a row connected to that "WL". However, not all columns are actually written to which potentially leads to static noise margin (SNM) failure.

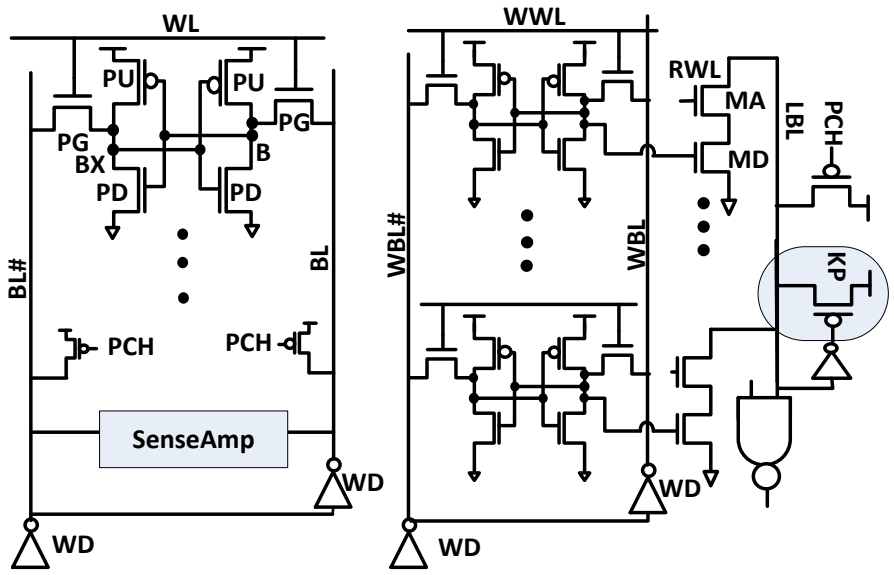


Figure 4.2 (a) Conventional 6T SRAM topology (b) Conventional 8T Register File

4.3.2 Register File(RF) Topology

Register Files are similar to SRAM except that they are designed for fast access compared to SRAM. A conventional RF 8T cell organization is shown in Figure 4.2(b). Unlike SRAM, the read and write ports are decoupled and therefore can simultaneously write and read different entries (rows). The write operation is generally similar to SRAM. The read path however is typically implemented with single ended full swing domino logic. The read bitline (LBL) is precharged to VDD. During read access, the precharge is turned OFF, WL turns ON, and a keeper, KPR, holds the precharged state "1" if data is "0". If data is "1" the bitline is discharged by the read access port, fighting contention from the keeper. This contention is the primary read V_{min} limiter in RFs. The keeper needs to be strong enough to hold the precharge value to avoid false discharge from leakage and other noise sources if data is "0". On the other hand it has to be weak enough that the bitline LBL node can be discharged by the Read Port if data=1. Unlike SRAM, RFs typically do not have half-select issue. Even in cases where there are multiple ways, each way is designed as a separate array with independent access.

4.3.3 SRAM/RF Design Failures

SRAM failures can be categorized as follows:

Write Failure: Inability to write a new value into the memory bitcell. A strong contention from PFET, PU, can prevent the memory bit node (B) from being overwritten with new value if the write driver (WD) and/or passgate (PG) is weak (Figure 4.3).

Read Failure: When WL is turned ON during read, the bit node storing a "0" can be raised above VSS by charge sharing from the highly capacitive and precharged bitline node (Figure 4.2a). This disturb could be large enough to cause the memory node to flip state. Therefore the NFET, PD, has to be strong enough to hold the "0" value. RFs do not have this read failure due to read port decoupling.

Access Failure: In SRAM with differential read, this occurs if there is inadequate differential development for sense-amp to resolve correctly. A weak passgate (PG) and/or pull-down (PD) NFET can result in reduced bitcell current (Icell). In RFs, this results from contention between a strong keeper and a weak read access port. Noise and leakage could also contribute to access failure.

Retention Failure: Cell loses its stored data as a result of droop, leakage, or other causes.

4.3.4 Design Constraints

A weak SRAM passgate and pull-down reduces read current (Icell) and causes read access failure whereas a stronger passgate and weaker pull-down reduces the static noise margin (SNM). On the other hand contention from a stronger pull-up and weaker passgate causes writeability failure. To address contradictory requirements, dynamic voltage control (Assist) techniques are used to modulate the strength of the SRAM devices by increasing (Boost) or reducing (Suppress) the supply voltage during read and write operations.

4.4 LOW VOLTAGE OPERATION TECHNIQUE

4.4.1 Write Assist Techniques

The goal of write assist techniques is to reduce contention by weakening the pull-up PFET and strengthening the NFET pass-gate. This is achieved either through:

(a) VDD Collapse–VDD is suppressed below the nominal voltage to weaken the PU PFET (Figure 4.3). Typically, VDD cannot be dropped below memory cell retention voltage; else unselected columns in an interleaving scheme will be corrupted. In [8] however, a transient voltage collapse (TVC) technique is proposed that collapses VDD below the retention voltage (close to zero) by controlling duration of the collapse. It shows that in 32nm, an SRAM could retain its value with VDD drop >300mV below its data retention voltage, resulting in 100mV improvement over conventional techniques.

(b)Negative Bitline – The bitline is suppressed below VSS to strengthen the NFET passpage PG. This can however lead to excessive electric field and reliability issues.

(c)VSS Raise – The bitcell VSS is raised to weaken PU, reducing PU gate voltage instead of source voltage.

(d)WL Boost – The wordline is raised above nominal VDD to increase the strength of the pass-gate PG NFET. Boosting techniques typically uses charge pump [9] or capacitive coupling to generate boost locally. In [10], a capacitive-coupling technique that uses the intrinsic coupling between write wordline and write bitline to boot write wordline is proposed, achieving 180mV improvement.

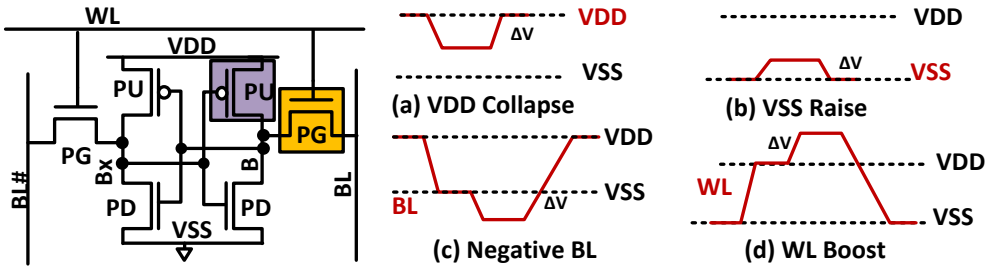


Figure 4.3 Write Assist Schemes (a) VDD Collapse (b) VSS Raise (c) Negative BL (d) WL Boost

To mitigate the impact of write assists on unselected columns, a Regulated Voltage Supply (RVS) is used to reduce the excessive electric field of unselected columns [11].

4.4.2 SRAM Read Assist Techniques

Read assist techniques aim to reduce SNM. The two main approaches are wordline suppression and VDD Boost (Figure 4.4).

(a) With *wordline suppression* technique, the WL voltage is reduced below VDD to weaken the passgate, hence reducing SNM. However, this also lowers cell read current (Icell) which can cause access failure.

(b) *VDD Boost* raises the bitcell VDD above nominal VDD. This strengthens pull-down NFET, PD, increasing SNM and read current (Icell). While this resolves the Icell issues, the excessive supply voltage could affect reliability of the gate.

To reduce the excessive potential difference and reliability issues resulting from voltage offset, RVS approach is proposed in [11]. During VDD Boost, the source (of the unselected column) is also raised to maintain a potential different. Only the selected column experiences the high electric field during the read duration. Similarly, during WL suppression, a negative source is applied to the selected column to increase read current.

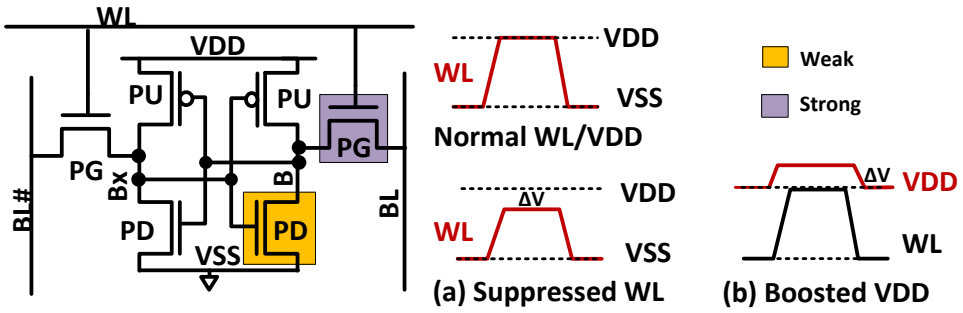


Figure 4.4 SRAM Read Assist Schemes (a) WL Suppression (b) VDD Boost

4.4.3 RF Read Assist

A "Contention Free" Shared Keeper (Delayed Keeper) is proposed in [20] that delays the start of RF keeper-ON during read (Figure 4.5) . This eliminates contention at the start of read, allowing the Read Port devices to pull-down the bitline node. The keeper is turned on later in the evaluate phase to sustain the precharge value if bitcell data is "0". The scheme leaves the bitline floating and is susceptible to noise

during the contention free window. To reduce noise impact requires that the bitline is fully shielded and the wordlines at least half-shielded. A 24% frequency improvement at 340mV is reported. The benefit of the scheme is higher at lower VDD where the impact of contention is more significant. The scheme relies on a difficult delay tuning which itself is subject to variation.

In [13], a conditional keeper is proposed that initially turns on a weak keeper and subsequently a strong keeper if data=1, reducing contention in the early phase of evaluation.

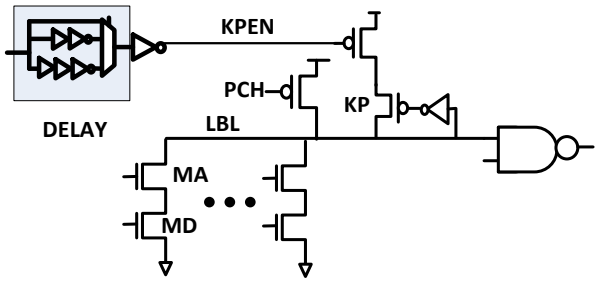


Figure 4.5 RF Read Assist[20]

4.4.4 Assist Techniques Summary

Increasing σ_{vt} and device scaling trend continue to reduce the viability of certain assist techniques. In [12], the effectiveness of write assist techniques is presented. It show diminishing gain with VDD lowering and VSS raising write assist schemes due to an already weak devices in current technology node. Also their effectiveness decreases with decrease voltage scaling. WL-boosting and negative bitline schemes on the other hand are shown in [12] to offer relatively better improvement.

4.4.5 Dual-Voltage Approach:

An alternative solution to processor low voltage operating is to not operate the SRAM at low voltage at all, eliminating it as a V_{min} limiter. The SRAM core is operated at a higher voltage than surrounding static logic. One approach puts only the memory cells on High-VDD supply [14] while keeping WL and BL on normal Low-VDD supply. This introduces supply noise margin concern between the bitline and the memory. Alternatively, SRAM memory, WL, and BL can be on the High-VDD supply [15]. This however

requires level shifter and increased power supply requirement due to increased current demand on the second supply (High-VDD supply).

4.5 SRAM/RF LOW-VOLTAGE TOPOLOGIES

Assist and Dual-Supply techniques are common mode that can be applied to most SRAM/RF topologies. The rest of the chapter will focus on how different SRAM/RF topologies fare in low VDD operation. Most of the topologies described apply to both SRAM and RF. Where necessary, a distinction is made.

4.5.1 Write Topologies

4.5.1.1 Dual-Ended (DE) Single Transmission Gate

In [16], the adoption of the standard RF 8T to improve SRAM cell stability and writeability is proposed. One of the main drawbacks of an 8T SRAM compared to a 6T SRAM is the area overhead. A 20% area penalty was reported at 65nm node [16]. However, this overhead is narrowing as increasingly more area is needed to solve stability issues in 6T SRAMs. A comparison of 6T and 8T bitcell SRAM is provided in [17]. In [18], a 10T differential read with dual port and column bypass to address column interleaving is proposed.

4.5.1.2 Single-Ended Transmission Gate (SETG)

In [19] a 6T single-ended SRAM with a one-sided full transmission pass gate is presented (Figure 4.6). It uses the RF full swing approach instead of SRAM differential operation. However, unlike RF, the design shares the same bitline between read and write. During write, an NFET creates a virtual VDD=VDD-Vtp), thus reducing the PU device strength. A PFET footer, FP, similarly creases a virtual VSS=Vtn. During hold, full supply source voltages are established with a bypass. Proposed design has similar read stability challenges as 6T. However, it can be adapted to implement independent read port (becomes 8T), especially for RF.

4.5.1.3 10T Dual-Ended Dual-Transmission Gate (DETG)

A 10T topology that uses a full transmission gate on both sides of the bitcell for a strong write "1" and "0" proposed in [20] is show in Figure 4.7. The scheme requires two wordline and two bitline signals. The increase in power from two switching wordline is compensated for by the reduced V_{min} in low voltage. Area overhead is however a challenge.

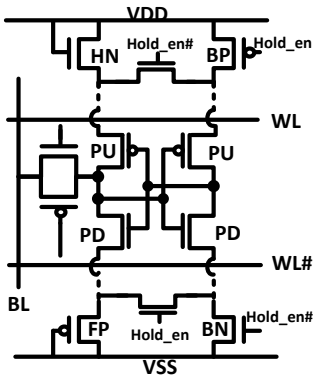


Figure 4.6 Single-Ended Transmission gate [19]

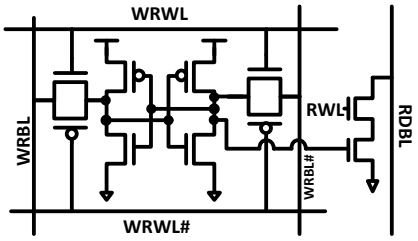


Figure 4.7 10T DETG[20]

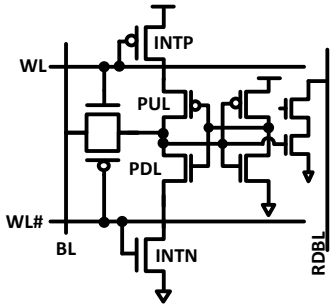


Figure 4.8 8T Interruptible

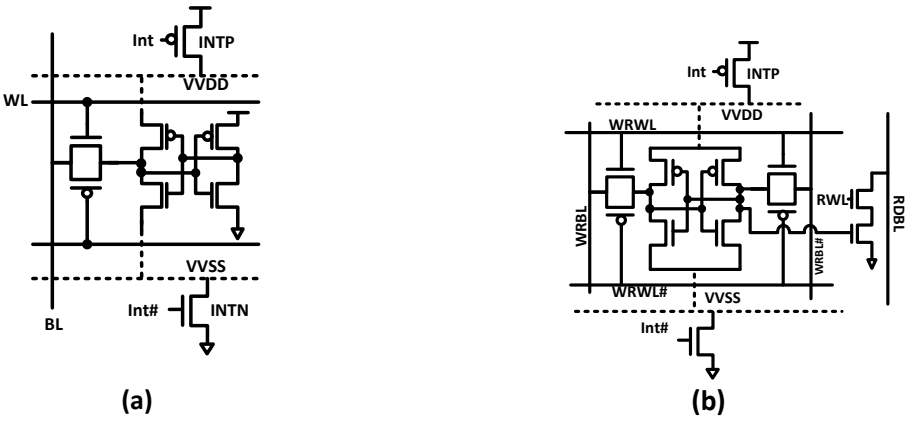


Figure 4.9 (a) 8T SRAM Interruptible with global interrupt (b) DETGI - DETG with Global Interrupt

4.5.1.4 Interruptible Single-Ended Transmission Gate(SEINT)

We propose using a fully interruptible RF/SRAM bitcell that eliminates write contention (Figure 4.8) for low voltage operation. The proposed latch based design uses single-ended passgate. During write, devices INTP and INTN are turned OFF. Unlike the one proposed in [19], the feedback devices PUL and PNL are fully interrupted during write, eliminating write contention. The latch design also uses a single bitline and two wordlines and is therefore more dynamic power efficient compared to DETG/DE since write bitline consumes more power than wordline [21]. To reduce the area overhead, the interruptible devices is shared by bitcells in the same row (Figure 4.9 (a)) by generating a global row interrupt signal. We also propose applying global interrupt scheme to DEGT (Figure 4.9 (b))

4.5.2 Read Topologies

Register File read V_{min} is limited by contention between the read port pull-down series NFETs and the bitline keeper PFET "KP" (Figure 4.2(b)). A stronger keeper is needed to reduce bitline noise of which leakage is typically the dominating component. Thus, RF read V_{min} techniques aims at reducing Read Port leakage to enable to use of a weaker keeper, reducing contention. Bitline leakage reduction techniques include segmentation to reduce the number of read ports (leakage sources) per bitline segment and stack forcing. Most common leakage reduction techniques however incur delay and area penalties. In SRAM, with bitline sensing, leakage reduction improve differential SRAM BL sensing at low voltage operation. Due to reduced bitcell current (Icell) at low voltage (reduced *Ion/Ioff* ratio), leakage from un-selected bitcells can be relatively high enough to compete with selected bitcell *Icell* value. Reducing leakage therefore enhances low voltage mode operation.

4.5.2.1 Data Precharging of Read Port Internal Node (DIP)

A 10T SRAM is proposed in [22] that uses leakage reduction techniques to reduce bitline droop (Figure 4.10 (a)). It uses a single-ended differential sensing in SRAM. In the proposed design, the bitcell data node is connected to a PFET, P1, that precharge the internal node "INT" if "Bit" = "0" to reduce bitline leakage. If "Bit" is "1", the P1 device leakage causes the internal node "INT" to settle at voltage >0. A negative V_{gs} is thus developed across device "NR2", reducing leakage from unselected bitcells. A variant of this technique [23] uses the wordline to precharge the internal node (Figure 4.10(b)).

4.5.2.2 Unclocked Wordline Stack Forced (UWSF)

In [24], we propose a scalable bitline using an *MxN* sub-segmentation to reduce bitline loading and leakage (Figure 4.11). An unclocked wordline (UWL) first activates and discharges the internal bitline

LBLB shared by 4 bitcells. The clocked wordline (SFWL) is then turned on which discharges the LBLA node. Both SFWL and UWL are "0" for unselected bitcells, reducing leakage through stack forcing.

4.5.2.3 Keeper Free Tristate(TRI)

We propose using a tri-state bitline that eliminates contention for low voltage operation (Figure 4.12). Two read wordline signal are required in this design. To maintain the performance advantage of RF read path skewing, the bitline is precharged as usual. This allows the Read Port PFET devices MP0 and MP1 to be sized only as keepers and not for transition, reducing area overhead. Both MP0, MP1 are turned ON when data is "0" (RWL#=0) to hold the bitline precharged value. There is no need for a keeper device. This eliminates keeper contention and improves evaluate delay.

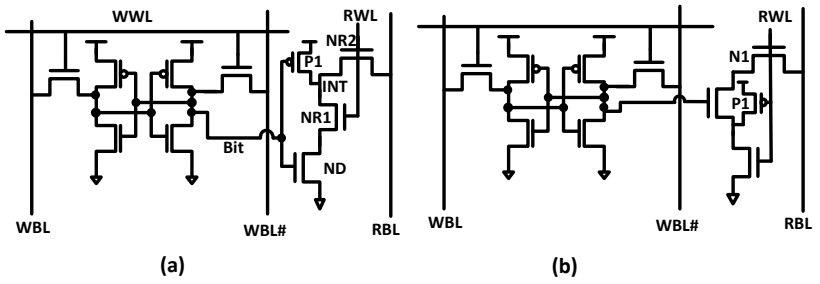


Figure 4.10 (a) Data Read Internal Precharge [22] (b) Wordline Internal Precharge [23]

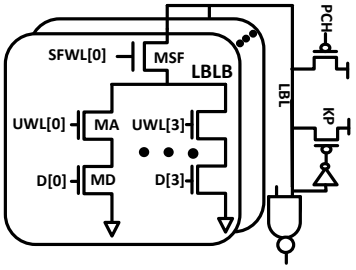


Figure 4.11 UWSF read scheme [24]

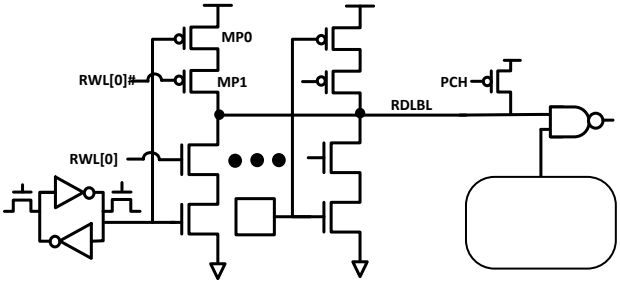


Figure 4.12 Proposed RF Tri-State read with no keeper

4.6 SIMULATION RESULTS

4.6.1 Write Results

We compared the low-voltage operation of the topologies discussed above: DE, SEINT, SETG, DETG, and DETGI (an interruptible DETG). The same device sizes were used for the write driver, the pass gates and the memory retention devices. To compare new topologies against upsizing conventional DE, we created a control DE with 2X the passgate size (DE2X). We then applied a 3σ *Le* and 3σ *Vt* variation to critical devices for worst-case write "0". No variation was applied to the data driver. The delay for write "0" was then measured from 50% write data fall to 20% bit node fall (or bit# rise whichever is slowest).

Figure 4.3 shows the performance of the various low-voltage write topologies relative to the conventional topology. Table 4.1 shows the "shmoo-like" normalized delay at "RSSF" corner. The failing voltage is denoted by "0". Results also show that doubling the passgate size of DE (DEX2) only improves Vmin by less than 100mV iso-delay, while a DETG achieves 150mV iso-delay. SEINT and DETGI achieve up to 200mV improvement iso-delay. SEINT, SETG and DETGI all extend the operating range into subthreshold. DETGI and SEINT can operate down to 200mV though much slower. Delay improvement increases at slow process corners and at lower voltage (Figure 4.13) where the impact of variation and contention is more severe. SEINT shows delay improvement of up to 3X at the same supply voltage over DE while DETGI show up to 2X over DE2. Across different process corners, SEINT and DETGI consistently outperform the other topologies.

4.6.2 Read Results

We compared 16-/32-segments versions of DPI, TRI, and UWSF topologies discussed above to the conventional RF read (CONV). 3σ *Le* and 3σ *Vt* variations were applied to the read port NFET devices (to weaken drive strength) and precharge PFET (increase drive strength). The same Read Port pull-down sizes were used for all topologies and segmentations. The keepers were sized to achieve the same LBL droop.

Figure 4.14 shows the read performance of the different topologies compared to the conventional read topology. The "shmoo-like" plot at "RSSF" corner is shown in Table 8.2. The TRI topology showed the widest operation range up to 300mV at RSSF corner. At iso-delay, the 32-segment TRI is 200mV lower

V_{min} than CONV. TRI has better performance at all voltages than CONV, DPI, and UWSF with up to 2X better delay improvement even at superthreshold voltages primarily due to absence of a keeper. The delay improvement over CONV increases at lower operation voltage (Figure 4.14). 32-segment DPI and UWSF also offer improvement over CONV.

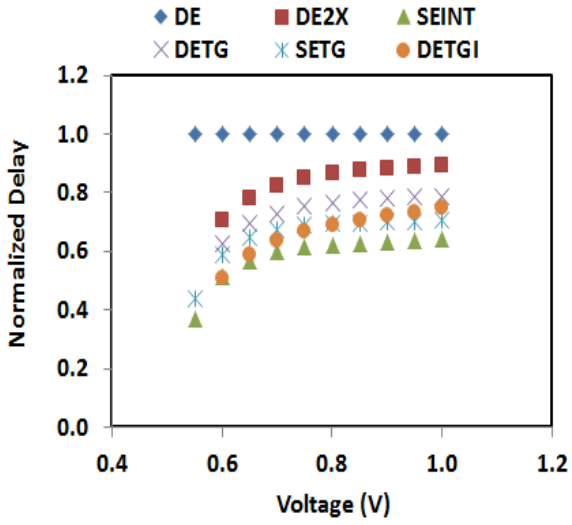


Figure 4.13 Write Delay Relative to DE topology

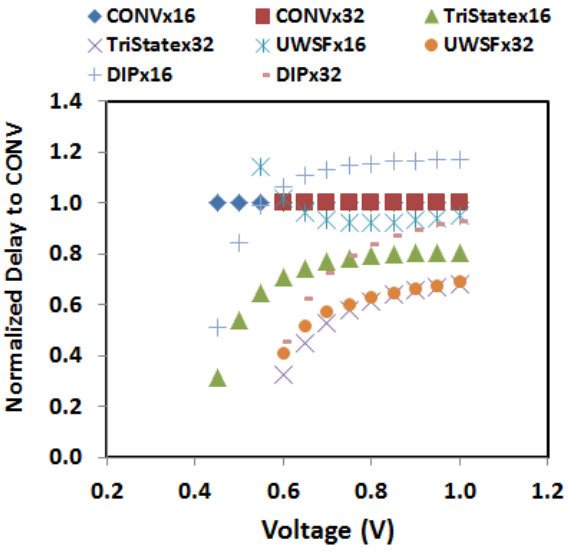


Figure 4.14 Read Delay relative to CONV topology

Table 4.1 Normalized Write Delay at different voltages

Voltage	DE	DE2X	SEINT	DET	SET	DETGI
0.10	0	0	0	0	0	0
0.15	0	0	0	0	0	0
0.20	0	0	0	0	0	0
0.25	0	0	239.14	0	0	333.78
0.30	0	0	123.43	0	177.79	79.89
0.35	0	0	28.11	0	37.36	19.37
0.40	0	0	8.90	0	11.33	6.58
0.45	0	0	4.15	0	5.10	3.46
0.50	0	7.25	2.52	4.16	3.01	2.33
0.55	0	3.14	1.80	2.29	2.09	1.77
0.60	0	2.22	1.41	1.72	1.61	1.44
0.65	0	1.76	1.17	1.41	1.32	1.24
0.70	2.75	1.49	1.00	1.21	1.12	1.09
0.75	1.91	1.30	0.88	1.07	0.98	0.99
0.80	1.55	1.16	0.79	0.96	0.88	0.90
0.85	1.34	1.06	0.72	0.88	0.80	0.83
0.90	1.20	0.97	0.66	0.82	0.73	0.78
0.95	1.09	0.91	0.62	0.77	0.68	0.73
1.00	1.01	0.85	0.58	0.72	0.64	0.70

Table 4.2 Normalized Read Delay at different voltages

Voltage (V)	CONV x16	CONV x32	TriState x16	TriState x32	UWSF x16	UWSF x32	DIP x16	DIP x32
0.10	0	0	0	0	0	0	0	0
0.15	0	0	0	0	0	0	0	0
0.20	0	0	0	0	0	0	0	0
0.25	0	0	0	0	0	0	0	0
0.30	0	0	185.03	475.43	0	0	0	0
0.35	0	0	44.40	113.29	0	0	0	0
0.40	0	0	14.84	37.41	0	0	49.40	106.93
0.45	0	0	6.84	16.86	0	0	15.24	33.21
0.50	0	0	3.98	9.53	0	0	7.64	16.40
0.55	8.36	0	2.71	6.30	0	0	4.82	10.15
0.60	3.99	0	2.05	4.63	9.24	15.57	3.48	7.19
0.65	2.71	0	1.66	3.67	3.99	7.07	2.73	5.55
0.70	2.10	13.66	1.41	3.06	2.62	4.71	2.28	4.55
0.75	1.75	7.31	1.25	2.64	2.02	3.61	1.98	3.89
0.80	1.53	5.29	1.12	2.35	1.68	2.99	1.76	3.43
0.85	1.38	4.26	1.03	2.14	1.48	2.59	1.61	3.10
0.90	1.26	3.64	0.96	1.97	1.34	2.32	1.49	2.85
0.95	1.18	3.22	0.91	1.84	1.25	2.13	1.40	2.65
1.00	1.11	2.91	0.86	1.74	1.18	1.98	1.33	2.49

4.7 CONCLUSION

SRAM/RF Near-/Sub-threshold operation are limited by process variation. Writeability, readability and read stability all rely on a relative strength of PFET and NFET devices. This dependency is a major obstacle to Near-/Sub-threshold operation due to process variation. While *Vmin* improvements have been demonstrated with Assist techniques, they are largely complicated and difficult to implement and control. New circuit topologies that address the fundamental limitations of conventional 6T/8T topologies are desired. Our comparison of different SRAM/RF topologies, considering variation, showed that interruptible SEINT scheme can reduce RF write *Vmin* by up to 200mV and extend its operating range into sub-threshold when compared to a conventional jam topology. SEINT and DETGI can improve delay by up to 3X at the same voltage over DE at RSSF corner. A tristate Read Port (TRI) is also proposed that lowers read *Vmin* by 300mV and improves delay by up to 2X. In both cases Vmin improvement over conventional design increases at lower voltage and also at slower process corners where impact of contention is severest on conventional topology. Based on our studies, we conclude that eliminating contention offers a good path to extending RF/SRAM operating range into Near-/Sub-threshold region.

4.8 REFERENCES

[1] Ronal G. Dreslinski et al., "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," Proc IEEE, vol 98, no2, Feb. 2010.

[2] Dejan Markovic et al., "Ultralow-Power Design in Near-Threshold Region," Proc IEEE, vol 98, no2, Feb. 2010.

[3] Himanshu Kaul et al., "Near-Threshold Voltage (NTV) Design – Opportunities and Challenges," DAC 2012 . June 2012

[4] R. Dreslinski, G. Chen et al., "Reconfigurable, energy efficient near threshold cache architectures," Proc. 41st Annu. MICRO, 2008.

[5] Hiroyuki Yamauchi, "A Discussion on SRAM Circuit Design Trend in Deeper Nanometer-Scale Technologies," IEEE TVLSI, vol 18, no.5 May 2010

[6] Mingoo Seok et al., "CAS-FEST 2010: Mitigating Variability in Near-Threshold Computing" IEEE Trans. On Emerging and Selected Topics in Circuits and Systems, vol1, no.1 march 2011

[7] S. Hanson et al., "Ultralow-Voltage, minimum-Energy CMOS" IBM J Res & Dev, vol 50, no 4/5, July/Sept 2006

[8] Yih Wang et al., "Dynamic Behavior of SRAM Data Retention and a Novel Transient Voltage Collapse Technique for 0.6V 32nm LP SRAM," IEDM Tech. Digest. pp 741-744

[9] Arijit Raychowdbury et al., "PVT-and-Aging Adaptive Wordline Boosting for 8T SRAM Power Reduction. Arijit Raychowdbury et al. ISSCC 2010"

[10] Jaydeep Kulkani et al: "Capacitive-Coupling Wordline Boosting with Self-Induced Vcc Collapse for Write Vmin Reduction in 22-nm 8T SRAM", ISSCC 2012

[11] Hiroyuki Yamauchi, "Embedded SRAM Circuit Design Technologies for a 45nm and Beyond", ASICON '97

[12] Vikas Chandra et al., "On the Efficacy of Write-Assist Techniques on Low Voltage Nanoscale SRAMs": DATE 2010

[13] A. Alvandpour et. al., "A sub-130-nm conditional keeper technique," In *IEEE* Jour. *Of Solid-State Circ.,* May 2002.

[14] K. Zhang et al., "A 3GHZ 70 Mb SRAM in 65nm CMOS Technology with Integrated Column-Based Dynamic Power Supply" JSSC, vol 41, no. 1, Jan 2006

[15] M. Khellah et al. "A 256-Kb Dual-VCC SRAM Building Block in 65-nm CMOS Process With Actively Clamed Sleep Transistor" JSSC, vol 42, no 1, Jan 2007

[16] Leland Chang et al., "An 8T-SRAM for Variability Tolerance and Low-Voltage Operation in High Performance Caches," IEEE JSSC, vol 43, no4, 2008

[17] Gregory Chen et al., "Yield-Driven Near-Threshold SRAM Design, IEEE TVLSI," Vol 18, No 11, 2010

[18] IK Joon Chang et al., "A 32kb 10T Sub-Threshold SRAM Array With Bit-Interleaving and Differential Read Scheme in 90nm CMOS," IEEE JSSC, Vol 44, No.2, Feb 2009.

[19] Bo Zhai et al., "A Variation-Tolerant Sub-200 mV 6-T Subthreshold SRAM," IEEE JSSC, Vol 43, No. 10, Oct. 2008

[20] Amit Agarwal et al., "A 32nm 8.3 GHz 64-entry x 32b Variation Tolerant Near-Threshold Voltage Register File," Symp. On VLSI Circuit/Technical Digest of Technical Papers, 2010.

[21] E. Donkoh et al. "Register File Write Data Gating Techniques and Break-Even Analysis Model" *ISLPED 2012*

[22] Benton Calhoun et al., "A 256-kb 65nm Sub-Threshold SRAM Design for Ultra-Low-Voltage Operation," IEEE JSSC, Vol. 42, No. 3, March 2007

[23] Tae-Hyoung Kim et al., "A High-Density Subthreshold SRAM with Data-Independent Bitline Leakage and Virtual Ground Replica Scheme" . ISSCC Conference, 2007

[24] Eric Donkoh et al., "A Low-Leakage Dynamic Register File with Unclocked Wordline and Sub-Segmentation for Improved Bitline Scalability". ISLPED 2012

5 LOW POWER REGISTER FILE READ DESIGN

ABSTRACT

We propose a register file that uses an unclocked wordline with dynamic bitline sub-segmentation and stack-forcing to reduce both active and standby leakage. The proposed technique improves bitline scalability to a larger number of entries per local bitline segment. For an example 32nm-CMOS 4-write, 6-read port 32 bits x 168 entries register file, the proposed bitline segmentation technique improves local bitline delay by 33%, lowers standby leakage by 70%, and reduces the number of clocked wordlines by 83% when compared with a conventional design. Furthermore, wordline sharing is shown to reduce the number of wordlines by as much as 58%.

Categories and Subject Descriptors: B.3.1 **[Memory Structures]**: Semiconductor Memories.

Keywords: Low Power, Register File, Sub-Segmentation, Unclocked Wordline, Stack-Forcing, Power Gating

5.1 INTRODUCTION

Register Files (RFs) are ubiquitous in modern microprocessor design, contributing significantly to total chip power and area. Approximately 30% of Intel 32nm Westmere (WSM) IA core [1] power is attributable to RFs [2]. For performance and area efficiency RF read access is usually implemented with wide-NOR dynamic logic. However, continued device and Vt scaling [3] pose a number of challenges: (1) Increased device leakage from Vt scaling (2) Reduced noise robustness due to increased bitline DC droop from device leakage (3) Increasing wire constrained bitcell size as devices continue to scale better than wires across process generations.

The most commonly used conventional dynamic wide-NOR RF leakage and DC droop reduction techniques are (i) use of high threshold (HVT) devices on the read ports (RPT) (ii) increase in keeper strength and (iii) reduction in the number of entries per local bitline (LBL) segment. Each of these techniques has associated tradeoffs. Increasing the keeper strength increases contention from the RPT during evaluate, increasing LBL fall delay and short circuit power. Using HVT devices also increases LBL fall delay. Reducing the number of entries per LBL segment increase the number of merge logic blocks

(RDMRG) and hence array area (Figure 2). The RDMRG block in an 8-entry LBL segmentation configuration can constitutes as much as 30% of the total array bundle. Reducing RDMRG count is therefore a desired design objective. However, due to increasing DC droop from RPTs leakage, the number of entries per LBL segment is typically limited to 8 or 16 entries in a high performance 32nm technology.

Alternative circuit techniques that have been proposed generally tradeoff one or more of area, standby leakage, or delay for the other (Figure 5.1). The leakage-tolerant techniques [4] [5] [6] [7] [8] address LBL noise robustness by reducing LBL active leakage (DC droop) but do not address standby leakage. They rely greatly on keeper downsizing to compensate for the delay degradation resulting from the technique. However, this is limited by process minimum device width and the increasing PMOS/NMOS device strength ratio across process generations [3]. These techniques further compensate for the delay degradation by using all Low-Vt (LVT) devices compared to conventional Dual-Vt (DVT). However, in cases where these techniques do not reduce the standby leakage, HVT devices are still needed.

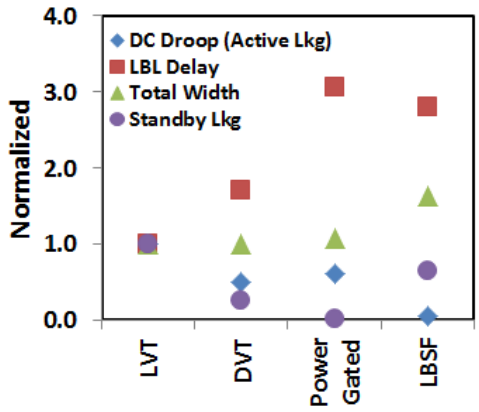


Figure 5.1 Delay, Leakage, DC droop, and Area tradeoff for conventional Low-VT, Dual-VT, Power Gated LBL, and LBSF[4] LBL

Power gating [9] is another common leakage reduction technique. The conventional power gating technique however can incur a high delay and area penalty, making its use prohibitive in key logic blocks such as arrays. Moreover, conventional power gating only addresses standby leakage and does not improve active leakage.

Another dynamic LBL design challenge is grid clock loading and wordline clocking distribution. Wordline shielding is typically required to reduce both wordline and bitline noise. As device dimensions

continue to scale much better than metal layers in the 32nm process technology nodes and beyond, the physical bitcell width is increasingly constrained by wires. Wordline shielding thus becomes costly, especially in multi-ported arrays.

In this chapter, we present an RF bitline technique that (i) uses a static unclocked wordline with a dynamic bitline to reduce the bitline delay (ii) uses drain stack-forcing and sub-segmentation to reduce both standby and active leakage for improved noise robustness (iii) reduces clock loading and wire congestion and (iv) provides a scalable LBL sub-segmentation enabling increased number of pull-down devices per bitline segment. We compare the proposed techniques to (i) conventional DVT LBL (ii) conventional LBL with modified power gate, and (iii) leakage bypass with stack forcing (LBSF) [4] in a 32nm technology. We show that a fully extracted 4-write, 6-read ports 32 bits x 168 entries RF implemented with 24-entry LBL segment using the proposed techniques improves LBL delay by 33%, reduces standby leakage by 70%, and uses 83% less clocked wordlines relative to a conventional design. We also show that the proposed technique can reduce number of wordlines by greater than 50%.

The organization of the chapter is as follow. Section 2 reviews conventional LBL, power gating applied to conventional LBL, and LBSF techniques. Section 3 describes the proposed unclocked wordline stack forced (UWSF) technique. In section 4, we present the comparative simulation results discussing delay, noise robustness, leakage, and scalability. We also present a UWSF implementation results of a 32bit x 168-entries 6-read port array. Summary is presented in section 5.

5.2 RELATED WORK

In this discussion, we define "Active Leakage" as the bitline leakage during read "0" operation (wordline is ON, PCH is OFF). The keeper holds the precharged value and leakage causes bitline DC droop. "Standby Leakage" is defined as the leakage when the bitline is not accessed (all wordlines OFF, PCH is ON).

5.2.1 Conventional RF Implementation (CONV)

A conventional wide NOR dynamic (domino) RF implementation is shown in Figure 5.2. RPTs from different memory entries form a segment of LBL in an *Mx1* NOR mux, *M* representing the number of read ports (RPTs). A NAND gate combines two LBL segments which are further merged at the global

bitline (GBL), also implemented with wide-NOR dynamic logic. The LBLs are pre-charged to VCC during standby. The precharge, keeper, NAND, and GBL NOR devices typically form the read merge block (RDMRG). The CONV read access decoding and clocking scheme is shown in Figure 5.3. The read access wordline signals (RWL) are generated by clocking the decoded address. This clocking of all RWL signals creates clock distribution challenges that lead to grid clock loading, and dynamic power dissipation.

The CONV operation is as follows (Figure 5.3). During read access, the read precharge clock (PCH) goes high and turns OFF the LBL PCH device. The RWL initiate read access to a single entry. If the accessed memory entry is storing data "1", the LBL is discharged to ground, causing a "domino" chain of evaluations along the path through the NAND, GBL, and to the latch downstream. If the stored data is "0", the LBL should retain its precharged value ("1") during the entire read phase with the PCH off. The keeper, *KP*, holds the precharged value to prevent a false evaluation, withstanding any noise impact from charge sharing, LBL interconnect, propagated noise from the RWL signals, and DC droop from RPT device leakage.

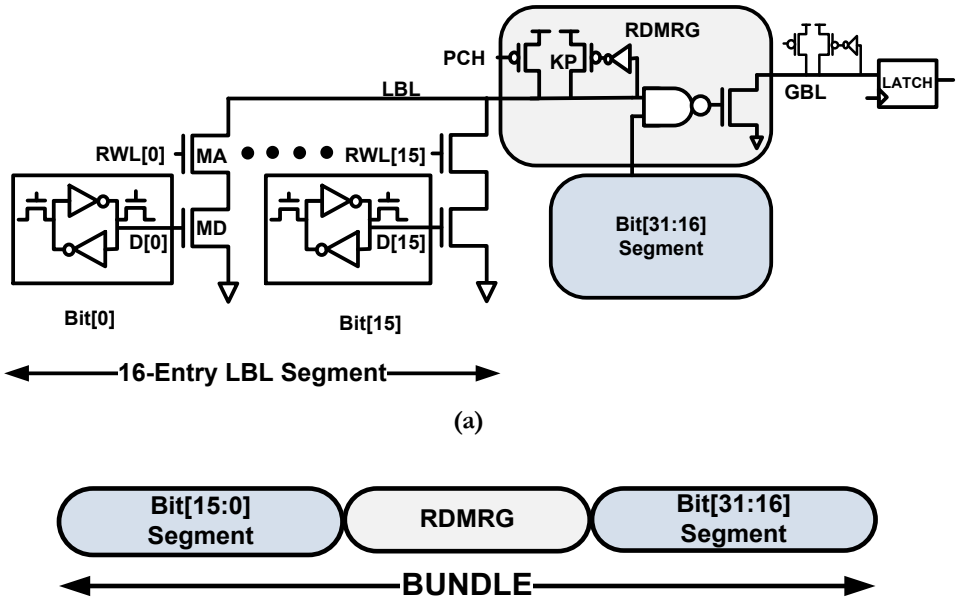


Figure 5.2 (a) Standard RF dynamic bitline wide-NOR topology with 16-entry per LBL segment and (b) physical organization

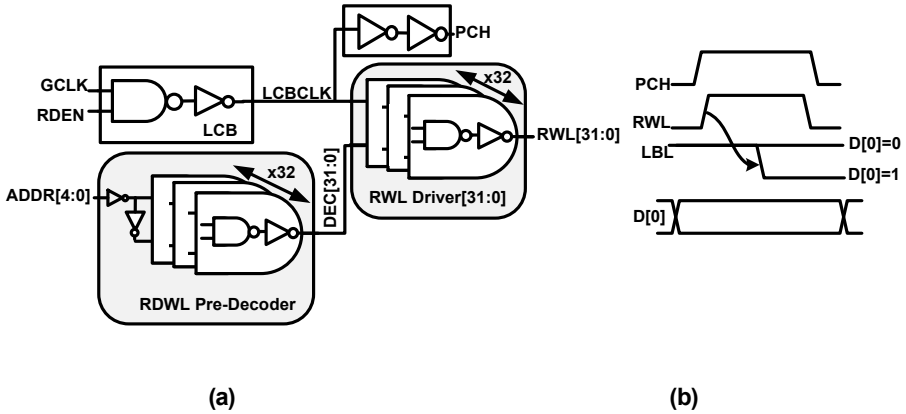


Figure 5.3 CONV LBL wordline decoding and clocking

5.2.2 Conventional LBL Footer Power Gating (CFPG) Technique

In a conventional power gating [9] technique, a footer device (MPG) is inserted in series with the RPT devices (Figure 5.4). When the LBL is inactive, MPG is turned OFF. This enforces stacking on the RPTs. The intermediate node Vx is discharged through leakage and converge to a voltage $Vx > 0$, $Vx > Vg$, creating a negative bias voltage across the top device MA. This significantly reduces leakage. During read access, the PGEN signal is asserted high, discharging node VVS and creating a virtual ground, before the "RWL" and "PCH" signal are asserted in a similar sequence as the CONV operation.

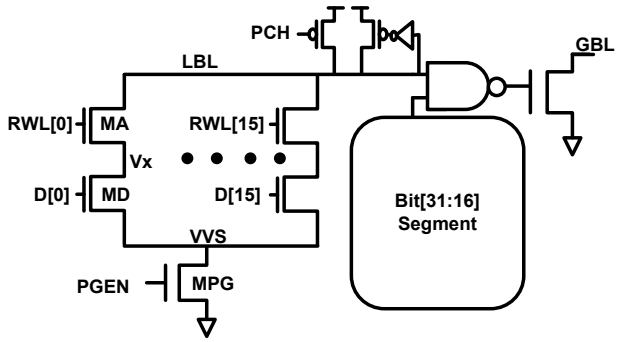


Figure 5.4 Conventional LBL Power Gating

5.2.3 Leakage Bypass with Stack Forcing (LBSF) Technique

The LBSF technique proposed in [4] reduces LBL active leakage by precharging the internal nodes *Vx1 and Vx2* of the RPTs (Figure 5.5). This creates a negative *Vgs* bias across both *MA1* and *MA2* devices, and sets *Vds*='0', cutting off LBL standby leakage through devices *MA1* and *MA2*. However, device *MD* will be leaking due to the internal precharging of node *Vx2*. The timing operation of LBSF is the same as the CONV design.

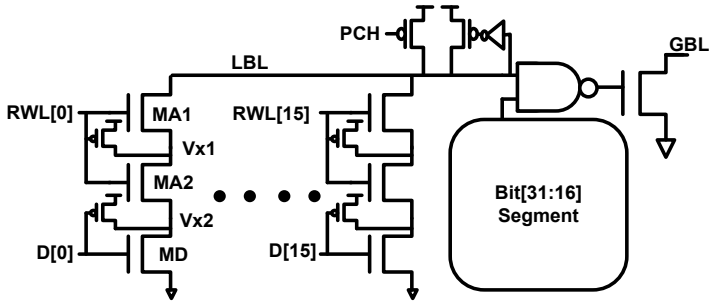


Figure 5.5 LBSF Technique with internal pre-charging

5.3 UNCLOCKED WORDLINE AND STACK-FORCED (UWSF) READ BITLINE

5.3.1 UWSF LBL and Decoder

The proposed UWSF LBL technique has two components: (1) sub-segmentation of the bitline and (2) removal of wordline clocking. The RPTs are grouped into sub-segments. A series NMOS device, MSF, is used to mux multiple RPT sub-segments, partitioning the LBL into two: *LBLA* and *LBLB*. The LBL segment is organized as *M x N* matrix where *M* is the number of entries per *LBLB* sub-segment and *N* is the number *LBLB* sub-segments per *LBLA* segment. A 16-entry LBL segment organized as 4x4 UWSF is shown in Figure 5.6. The *LBLA* mux select, SFWL, decoded from the address, is clocked to satisfy the LBL domino clocking requirement. Thus, the read wordline (UWL) need not be clocked, reducing the number of clocked signals by a factor of *M*.

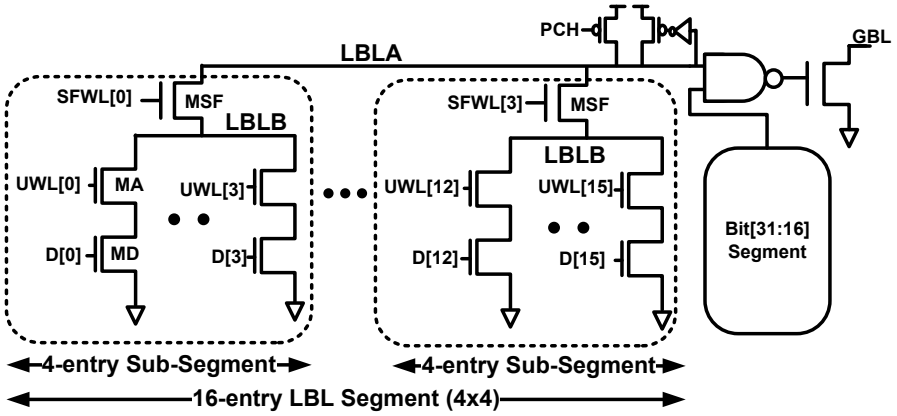


Figure 5.6 UWSF 4x4 16-entry LBL configuration with stack-forced dynamic bitline and unclocked wordline

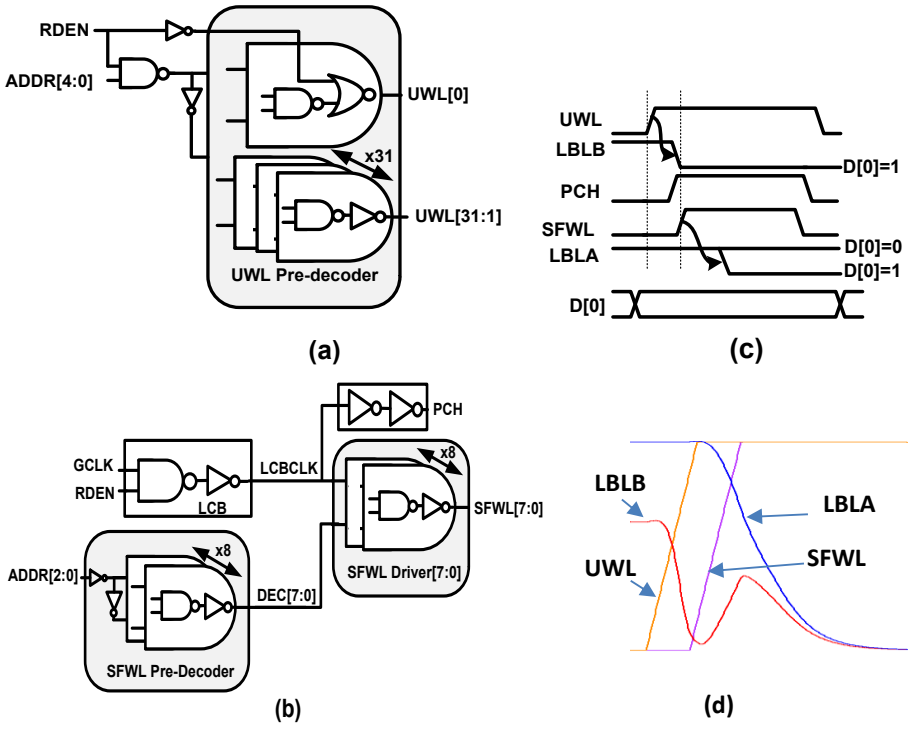


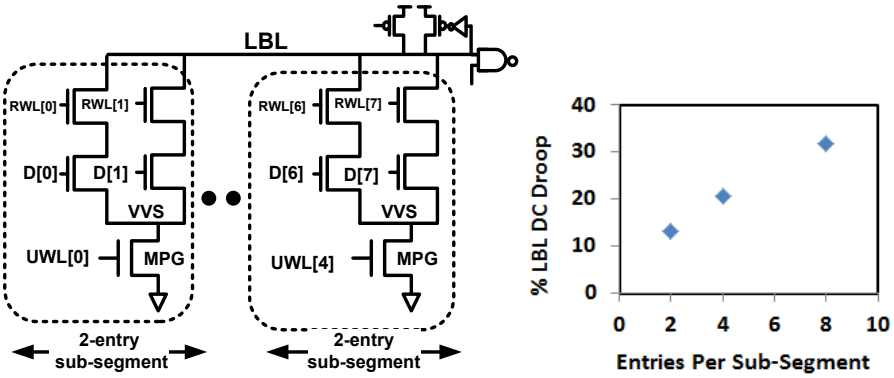
Figure 5.7 UWSF (a) unclocked wordline UWL and (b) SFWL clocking scheme (c) Timing waveform (d) Simulation waveform

A 32-entry decoding scheme of the proposed UWSF is shown in Figure 5.7. ADDR[4:2] is decoded and clocked as SFWL[7:0] to select *LBLA* mux while the RPT select wordlines UWL[31:0] from the pre-decoder are unclocked. Thus, only 8 signals are clocked compared to 32 signals in the CONV design (75% reduction). Without the clocking stage, the UWL signal is 2-gate stages faster than the clocked RWL signal of the CONV design and has no required setup to LCBCLK.

The operation of the UWSF is as follows (Figure 5.7). The faster unlocked wordline (UWL) selects one entry of a sub-segment and starts discharging *LBLB* node. PCH is then turned OFF and SFWL is asserted after 2-gates delay to select an *LBLB* sub-segment, discharging *LBLA*. SFWL timing corresponds to the conventional RWL. In the standby mode, both UWL and SFWL will be reset to "0" to enforce LBL stacking. The SFWL signals are reset by the clock. To reset UWL signals at minimal decoder area overhead without a clock, the read enable signal, RDEN, is used to state-force all addresses to "0". This guarantees a deterministic decoder output, i.e. only the decoder output UWL[0] can be "1". The remaining UWL[n…1] decoder outputs will be "0". UWL[0] is then reset to "0" by RDEN. Alternatively all UWL signals could be reset by RDEN, without state-forcing the address, at a higher area but lower reset switching cost.

5.3.2 Sub-Segmentation and Active Leakage

The CFPG stack-forcing shares a power gate footer among all RPTs of a segment. While this reduces standby leakage, it does not improve active leakage since all the RPTs of the active LBL will have their virtual VSS (VVS) enabled during read access. We propose further sub-segmentation of the CFPG LBL to reduce active leakage by only activating one LBL sub-segment during read access while the remaining sub-segments are kept inactive. An 8-entry 2x4 CFPG LBL sub-segmentation using the UWSF decoding scheme as the power gate control is shown in Figure 5.8(a). The DC droop of a *2x4* sub-segmented LBL is reduced by 58% relative to a single *8x1* LBL segment (Figure 5.8 (b)).



(a) (b)

Figure 5.8 (a) A 2xN sub-segmenation of CFPG. (b) DC droop plot for 2x4, 4x2, and 8x1 CFPG sub-segmentations

5.3.3 Wordline Sharing

Increasingly, RF bitcell layout sizes are constrained by the number of tracks needed to route wordlines. Wordline clocks have more stringent noise constraints that require extra tracks for shielding purposes. To reduce the number of signals, a scheme that shares wordline among multiple sub-segments is employed. Figure 5.9 shows a 2-way sharing scheme of a 2xN sub-segmented LBL configuration. Each wordline is shared by two sub-segments. A 16-entry LBL segment will therefore requires a total of 16 signals (SFWL[7:0] and UWL[7:0]) in a 2-way sharing scheme, breaking even with the CONV design.

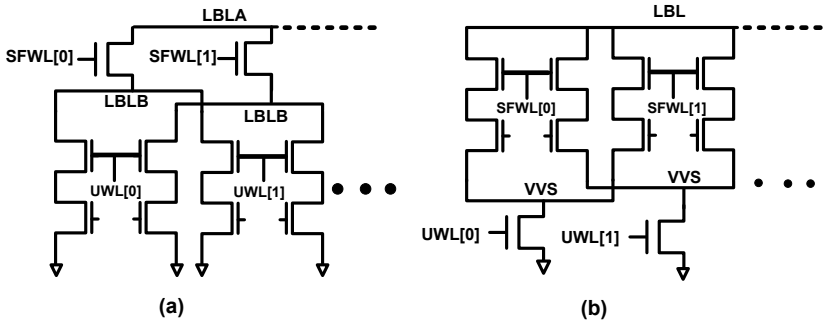


Figure 5.9 A 2xN sub-segmentation with 2-way sharing to reduce number of wordlines (a) UWSF (b) CFPG

Further sharing and sub-segmentation can reduce the number of wordline signals depending on the decoder scheme used. A 16-entry 4x4 sub-segmented 4-way sharing, for example, can decode address bits ADDR[3:2] as SFWL[3:0] to select one of 4 LBLA muxes, while address bits ADDR[1:0] are decoded as UWL[3:0]. Each UWL signal is shared by four RPTs of alternating LBL sub-segment entries. A total of 8 signals (4 UWL + 4 SFWL) are required to access the 16 entries, a 50% reduction. Table 5.1 shows the SFWL and UWL mapping to entries.

Table 5.1 A 4x4 sub-segment with 4-way share mapping

	SFWL[0]	**SFWL[1]**	**SFWL[2]**	**SFWL[3]**
UWL[0]	Entry[0]	Entry[4]	Entry[8]	Entry[12]
UWL[1]	Entry[1]	Entry[5]	Entry[9]	Entry[13]
UWL[0]	Entry[2]	Entry[6]	Entry[10]	Entry[14]
UWL[0]	Entry[3]	Entry[7]	Entry[11]	Entry[15]

5.4 SIMULATION RESULTS

5.4.1 Simulation Setup

We compare the different LBL techniques: (1) CONV DVT (2) UWSF (3) CFPG with sub-segmentation and (4) LBSF DVT. Device width is used as a proxy for device area. Devices were sized such that the total device width is equal, accounting for any auxiliary devices required for the topology. Figure 5.10 shows the device sizes and type used.

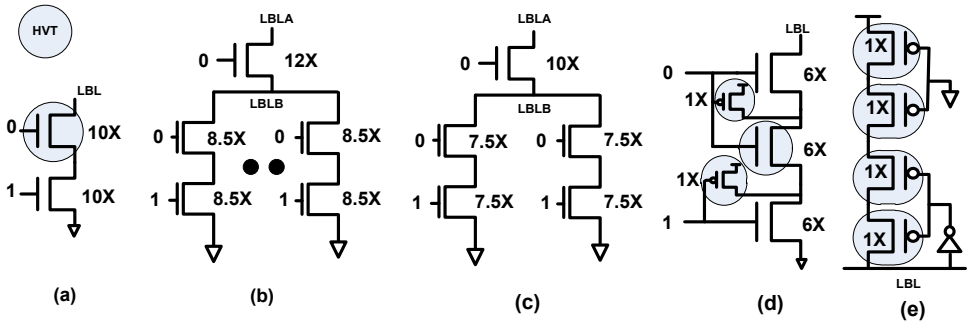


Figure 5.10 Device size, type, and worst-case leakage conditions for (a) CONV (b) UWSF/CFPG 4xN (c) UWSF 2xN (d) LBSF (e) LBSF 4-stack keeper scheme.

The total device width of UWSF includes the LBLA mux select devices. UWSF and CFPG use the same device sizes. The overhead of the mux select devices are amortized over the *M* RPTs sharing the *LBLB* segment. Thus, the total overhead is less compared to LBSF. Three LBL segment configurations were evaluated for each topology: 8-, 16-, and 32-entry per LBL segment. For a given topology, the same RPT sizes are used for all three configurations of that topology without resizing. We evaluated both *2xN* and *4xN* LBL sub-segmentation configurations of UWSF. The CFPG configuration used was a *4xN* sub-segmented LBL.

5.4.2 Leakage Consideration

The worst case leakage conditions are as shown in Figure 5.10. Under CONV leakage condition, the LBL RPTs will be leaking through a single device, *MA* (Figure 5.2). Both standby and active leakage are reduced by using a HVT device. In the LBSF (Figure 4), LBL leakage through device *MNA* and *MNB* are reduced through the internal precharging of nodes *Vx1* and *Vx2* respectively. However, device *MND* will still be leaking. Hence, both CONV LBL and LBSF require DVT to reduce RPT standby leakage. The worst-case standby leakage condition for UWSF has two series devices, *MSF* and *MA*, turned OFF (Figure 5.6), enforcing stacking on the RPTs regardless of the memory data content. This and sub-segmentation reduces both standby and active leakage, enabling the use of all LVT devices. CFPG also stack forces *MA* and *MPG* (Figure 5.4) to reduce standby leakage. Active leakage is also reduced by the proposed sub-segmentation. The standby leakage comparison is presented in Table 5.2. Leakage is reduced by 91% and 93% in the UWSF and CFPG respectively while the DVT LBSF reduces leakage by 80% relative to the CONV DVT (partly due to smaller sizes).

Table 5.2 Leakage comparison of the 4 topologies studied

	CONV	UWSF(4x1)	CFPG(4x1)	LBSF
LKG ratio	1	0.089	0.073	0.202
%reduction	0	-91%	-93%	-80%

5.4.3 Noise Consideration/Keeper Sizing

LBL is subjected to four main sources of noise (i) DC droop from leakage through the RPT, (ii) charge sharing glitch (iii) propagated noise from the RPTs input signals, and (iv) interconnect noise on the LBL wire. Table 5.3 shows the distribution of noise sources for 4 different arrays from extracted layout in a 32nm design. For a 1-port array with full shielding, the noise is dominated by DC droop. However, for a multi-ported array, interconnect and propagated noise components become significant as full shielding is prohibitive.

Table 5.3 Noise breakdown of different ported arrays

Noise Sources	1-Port 8-seg LBL	2-Port 8-seg LBL	6-Port 8-seg LBL
% DC Droop	72%	46%	29%
% Interconnect	14%	29%	24%
% Propagated	14%	16%	34%
% Charge Sharing	0%	9%	13%

5.4.3.1 DC Droop (Active Leakage) and Keeper Sizing

Active leakage manifests as a DC droop during read "0" when the leakage through the RPTs can falsely discharge the LBL in the active window. The keeper should be sufficiently strong to hold the LBL PCH value. Increasing keeper size reduces the droop but also increases the LBL fall delay. Figure 5.11(a) shows the keeper size impact on DC droop on a 32-entry LBL. The required keeper size for UWSF and CFPG is 2.6X smaller than CONV for a DC droop noise floor of 10% VCC. LBSF has the least DC droop due to the precharging of the internal node.

5.4.3.2 Propagated Noise

Wordline noise is propagated onto the LBL through the RPTs. Typically propagated noise is reduced by shielding the wordline signals. In a routing constrained array, this could increase the array size. The propagated noise from UWSF wordline signals (UWL) are attenuated by the mux select device *MSF*. Moreover, inactive sub-segments will have their *LBLB* node at or near "0" voltage. Therefore,

propagated noise from UWL does not impact *LBLA* significantly. Propagated noise from SFWL onto *LBLA* is not attenuated. However, since the total number of SFWL is reduced by a factor of *N* relative to CONV, the total noise impact on *LBLA* is reduced. Figure 5.11(b) shows the LBL sensitivity to propagated noise during read "0". The same keeper size was used for all four techniques. As shown, when noise was only applied to the UWL signal, the propagated noise onto *LBLA* is significantly attenuated. The worst case propagated noise of applying noise simultaneously to both UWL and SFWL is still up to 50X more noise tolerant than the CONV topology at input noise less than 10% VCC. LBSF shows most resilience to propagated noise due to internal precharging.

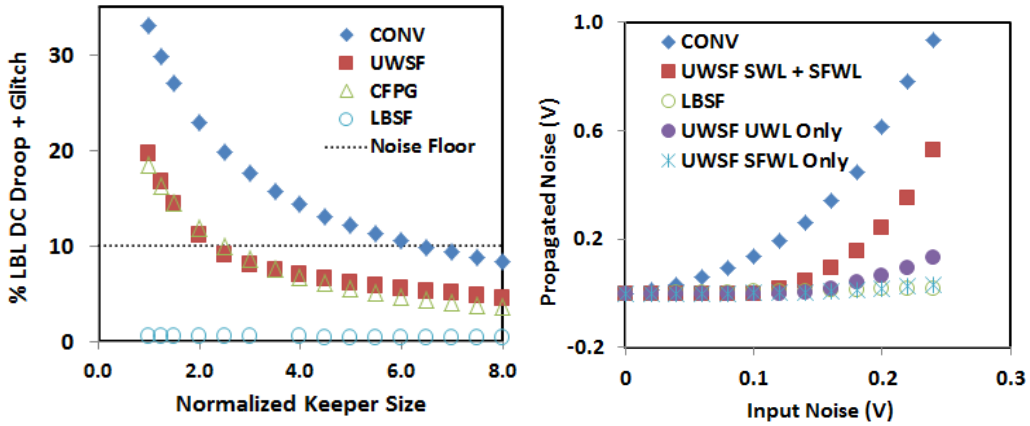


Figure 5.11 (a) Normalized keeper size vs DC Droop (b) LBL propated noise sensitiviy to input noise.

5.4.3.3 Glitch Noise

In standby mode, the UWSF *LBLB* node will be discharged to near "0" due to leakage or to "0" from a previous access to adjacent entry sharing the same *LBLB* node. At the onset of evaluation when the SFWL turns ON, a charge sharing glitch occurs between *LBLA* and *LBLB* due to capacitance and voltage differences. Ideally a smaller capacitance on *LBLB* node is required to reduce the glitch; therefore $N > M$ is desirable for glitch reduction (Figure 5.12 (b)). However, for the same total LBL capacitance (*LBLA* + *LBLB*), the best delay is obtained if a significant fraction of the total LBL capacitance is discharged by the early arrival of UWL before SFWL is turned ON. Thus, delay favors $M > N$. There is therefore an optimal $M \times N$ configuration.

The worst-case charge-sharing glitch and worst-case leakage DC droop occur at different time windows (Figure 5.12 (b)). While the worst charge-sharing occur at the onset of SFWL going active, the DC droop peak occurs at the end of the SFWL active window. The two noise sources are therefore not additive.

Figure 5.12(c) shows a physical organization that groups the shared RPTs into clusters to reduce routing capacitance on the *LBLB* node.

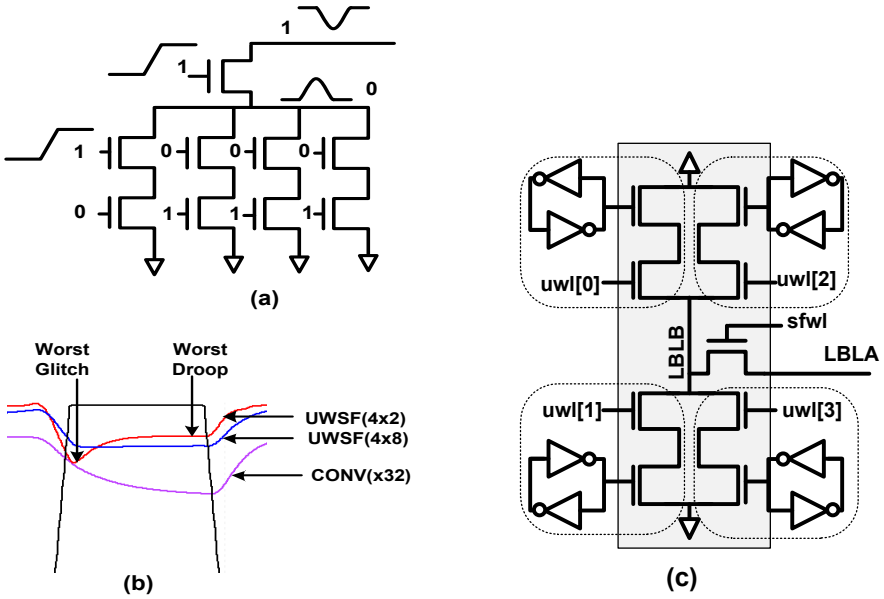


Figure 5.12 (a) UWSF worst-case glitch condition LBL (b) glitch and DC drop waveform (c) a 4x4 layout organization to minimize *LBLA* routing capacitance

5.4.4 Delay & Segment Scalability

In comparing the delay using iso-width devices, the keeper is sized such that LBL DC droop + glitch does not exceed 10% of VCC at the noise corner (ignoring the other noise sources). The UWSF technique has two disadvantages that negatively impact the LBL delay when compared to the CONV topology: (1) a 3-stack pull-down RPTs compared to the 2-stack in the CONV topology and (2) smaller RPT devices for the same total width. However, their impact on LBL fall delay is compensated for by the following advantages: (i) the elimination of UWL clocking to enable an early discharge of *LBLB* node before SFWL is active; this reduces the LBL clock-to-out delay measured as SFWL rise to *LBLA* falls; (ii) the reduced DC droop from reduced leakage, enabling keeper downsizing and less contention; and (iii) the use of all LVT devices due to reduced leakage. While CFPG also shares the same disadvantages over CONV, it does not have advantage (i). LBSF on the other hand does not have advantages (i) and (iii).

5.4.4.1 Keeper Scalability

Figure 5.13(a) shows the keeper size required for 10% dc droop + glitch. The 4x2 and 4x4 UWSF configuration keeper size is limited by glitch. Therefore, the required keeper size decreases as *LBLA* capacitance is increased from increasing the number of sub-segments per *LBLA* segment. Comparatively a 2x8 configuration requires a smaller keeper size due to reduced glitch and reduced active leakage from further sub-segmentation. Though the DC droop specification was 10%, LBSF DC droop was only 1% at the minimum possible keeper size. Even though the keeper could theoretically be downsized further, it could not fully capitalize on the increased noise robustness due to minimum device size limit.

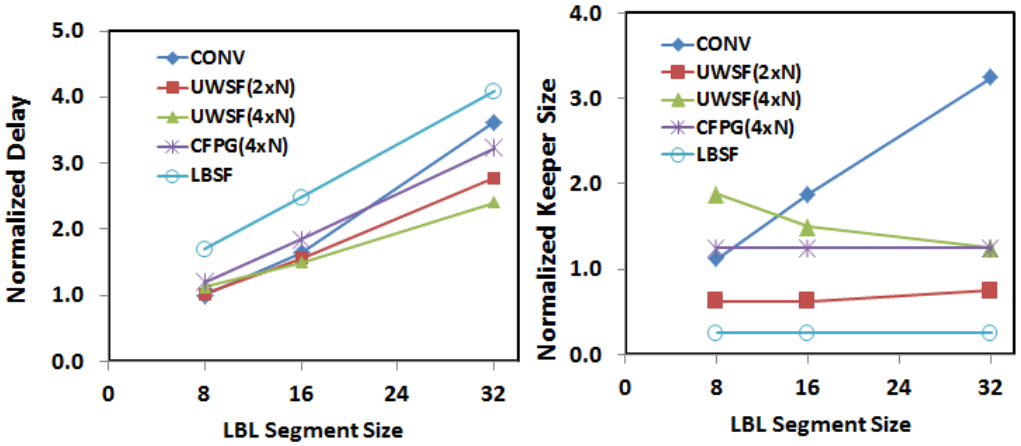
5.4.4.2 Delay Scalability

The delay performance of 8-, 16-, and 32-entry LBL configuration for the aforementioned techniques are presented in Figure 5.13(b). The 16-, 32-entry *4xN* UWSF configurations outperform the conventional design by 9% and 34% respectively. The 8-entry *4x2* LBL UWSF shows a 12% delay degradation over CONV. This is primary due to a larger keeper size required for the charge-sharing glitch, a consequence of a relatively lower *LBLA*/*LBLB* cap ratio. However, an 8-entry *2x4* LBL UWSF configuration only shows 2% delay degradation. The *2xN* 16-entry and 32-entry LBL UWSF also demonstrate 5% and 23% delay improvement respectively over the CONV. As the number of sub-segments per *LBLA* segment is scaled up, the impact of glitch on the *4xN* configuration keeper size is reduced and it outperforms the *2xN* configuration. Increased sharing of *LBLB* improves delay by the early discharge of a relatively larger portion of the total *LBL* capacitance (from the early arrival of the UWL). Increased sharing of *LBLB* also reduces the Mux passgate device area overhead. The optimal *M x N* configurations for the 8-, 16-, and 32-entry LBL segmentations are shown to be 2x4, 4x4 and 4x8 LBL sub-segment configurations respectively.

LBSF has the worst iso-width delay, with degradation of 70%, 51% and 13% for 8-, 16-, and 32-entry segments respectively. CFPG shows 11% delay improvement at 32-entry LBL configuration but degrades by 12% and 20% at 16-, 8-enty LBL configuration. The proposed UWSF shows the best LBL segmentation scaling.

5.4.4.3 Noise Robustness Vs. Delay Tradeoff

Figure 5.14 shows the noise robustness vs. delay tradeoff for 32-entry LBL configurations with increasing keeper size. The proposed UWSF shows the best tradeoff between noise and delay. LBSF is most noise tolerant but has the worst iso-width delay degradation.



(a) (b)

Figure 5.13 (a) Keeper sizing and (b) Delay scalability of 8-, 16-, and 32-entry LBL segmentation

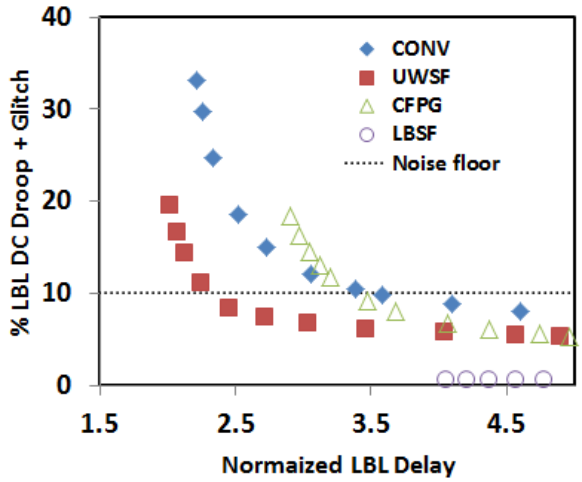


Figure 5.14 LBL noise robustness (glitch + droop) and delay tradeoff for a 32-entry LBL. Noise floor is 10% Vcc

5.4.5 UWSF Multi-Port Implementation Result

The proposed UWSF was implemented on a 4-write, 6-read port 32 bits x 168 entries array. The read ports were clustered [10], making it ideal for sharing. A 6x4 LBL sub-segmentation and a 4-Way UWL sharing schemes were used with each UWL wordline shared by four alternating entries of the 4 sub-

segments. Only 10 signals (6 UWL and 4 SFWL signals) were required to access each 24 entry segment. The total number of clocked signals was reduced from 24 to 4.

Table 5.4 Extracted results of a 4-write, 6-port 32 bits x 168 entries array UWSF LBL compared to CONV LBL

	CONV	UWSF
Delay (Normalized)	1	0.67
Total Noise	1	0.46
Number of RWL per segment	24	10
Number of Clocked Wordlines	24	4
Standby Leakage (Data SP=0.005)	1	0.67
Standby Leakage (Data SP=0.33)	1	0.30

Timing, noise, and power analysis on an extracted layout shows a 54% LBL noise reduction, 33% delay improvement and as much as 70% standby leakage reduction compared to the conventional implementation (Table 5.4). Leakage reduction is dependent on memory data signal probability (SP) [2].

5.5 CONCLUSION

We presented an unclocked wordline technique that in combination with sub-segmentation of the dynamic LBL can implement stack-forcing and improve a 32-entry segmented local bitline delay by 34%, lower worst-case standby leakage by 91%, and reduced number of clocked wordline by over 75% when compared to a conventional design with the same total width. UWSF outperforms LBSF and CFPG in the critical metric of delay, leakage and area. Even though the LBSF was the most noise resilient, this robustness could not be fully converted into reduced keeper sizing to improve delay. Summary of the results are shown in Table 5.5 and Figure 5.15 The proposed UWSF delay scales with increasing entries per LBL segment. This enables increased LBL segment size, thereby reducing the number of RDMRG blocks and array size. We also showed how sub-segmentation can be applied to conventional stack-forcing to reduce active leakage, improve noise robustness, and hence improve delay. Furthermore, wordline sharing using the proposed technique can reduce the number of wordlines by more than 50%. Finally we implemented the proposed techniques on a production 4-write, 6-read port 32 bits x 168 entries array and extracted result showed 54% LBL noise reduction, 33% delay improvement, 70% standby leakage, 58% reduction in number of wordlines, and 83% reduction in clocked wordlines.

Table 5.5 Simulation results summary of 32-entry CONV, UWSF, CFPG and LBSF LBL. Keepers are sized for 90% Vcc Droop + Glitch (@ffff corner).

Configuration	Keeper Size	LBL Fall Delay	LBL Rise Delay	Num of Clock WL	Leakage
CONVx8	1.00	1.00	1.00	8	1.00
CONVx16	1.88	1.64	1.58	16	1.00
CONVx32	3.25	3.63	2.69	32	1.00
UWSFx8(2x4)	0.63	1.02	0.87	4	0.08
UWSFx16(2x8)	0.63	1.56	1.30	8	0.08
UWSFx32(2x16)	0.75	2.78	2.11	16	0.08
UWSFx8(4x2)	1.88	1.12	0.84	2	0.09
UWSFx16(4x4)	1.50	1.50	1.20	4	0.09
UWSFx32(4x8)	1.25	2.41	1.91	8	0.09
CFPGx8	1.25	1.20	0.96	8	0.07
CFPGx16	1.25	1.85	1.48	16	0.07
CFPG32	1.25	3.23	2.47	32	0.07
LBSFx8	0.25	1.70	0.85	8	0.20
LBSFx16	0.25	2.48	1.28	16	0.20
LBSFx32	0.25	4.09	2.06	32	0.20

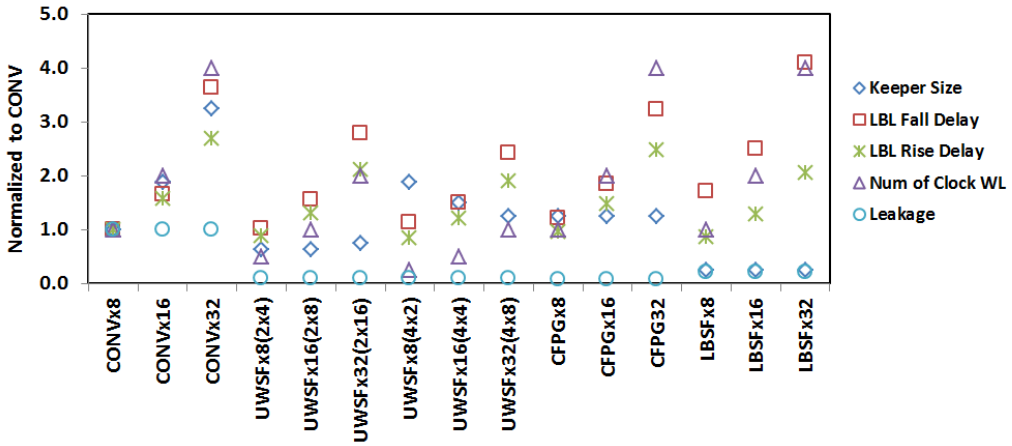


Figure 5.15 A summary of results normalized to CONV design

5.6 RELATED PUBLICATION

[1] Eric Donkoh, Patrick Chiang., "A Low-Leakage Dynamic Register File with Unclocked Wordline and Sub-Segmentation for Improved Bitline Scalability". ISLPED 2012

5.7 REFERENCE

[1] N. Kurd *et al.*: "Westmere: a family of 32nm IA processors", in *Proc. IEEE Int. Solid-State Circ. Conf.*, 2010, pp. 96-97.

[2] E. Donkoh et al. "Register File Write Data Gating Techniques and Break-Even Analysis Model" *ISLPED 2012*

[3] P. Packan et. al.,"High Performance 32nm Logic Technology Featuring 2nd Generation High-k + Metal Gate Transistors," *IEDM 2009, pp 1-4*

[4] S. Tang et. al., "A leakage tolerant dynamic register file using leakage bypass with stack forcing (LBSF) and source follower", *2002 Symp. VLSI Circuits, pp 320-321*

[5] A. Agarwal et al., "A Leakage-Tolerant Low-Leakage Register File with Conditional Sleep Transistor", *Proc. SOC Conference,2004, pp 241-244*

[6] A. Alvandpour et. al., "A sub-130-nm conditional keeper technique," In *IEEE* Jour. *Of Solid-State Circ.,* May 2002.

[7] R Krishnamurthy et. al., "A 130-nm 6Ghz 256x32bit leakage tolerant register file," In *IEEE JSSC.,* 2002.

[8] S. Borkar, "Circuit techniques for subthreshold leakage, avoidance, control, and tolerance*," IEDM Tech. Dig., pp. 421-424,Dec, 2004.*

[9] S. Narendra, et al, "Scaling of Stack Effect and its Application for Leakage Reduction", *ISLPED,2001*

[10] R. Kumar and G. Hinton, "A family of 45nm IA processors," *ISSCC'09 Digest of Tech. Papers*, pp.58-59.

6 LOW POWER REGISTER FILE WRITE

ABSTRACT

Register Files account for 30% of 32nm Intel WSM Core dynamic power of which 25% is due to write data distribution. We analyze Register File data gating strategies used to reduce write bitline dynamic power by as much as 96%. We explore the tradeoff of various data gating topologies (Global, Midway, Local), logic implementations (NAND, NOR, Tri-State), and techniques (Stack-Forcing, State-Forcing) to reduce both dynamic and leakage power. We then present a simple and accurate data gating break-even analysis model. The model comprehends "Data" and "Enable" switching activity, signal probability, logic implementation overhead, demonstrating an average error range of ±5%.

Categories and Subject Descriptors: B.7.1 **[Integrated Circuits]**: Types and Design Styles.

General Terms: Performance, Design

Keywords: Data Gating, Low Power, Register File, SRAM

6.1 INTRODUCTION

As processors continue to scale down to lower operating voltages encompassing new form factors such as Ultrabook and Tablets, power becomes the bottleneck to performance, where performance is redefined to include broader measures such as battery life. Thus, low power design techniques have become major focus in recent CPU designs [1][2], from architecture, through circuit optimization, to device level. Register Files (RFs) are used extensively in modern processors as on-chip memories because of their large storage density and small access latency as well as variability tolerance in low voltage operation [3]. Both high demand on memory and performance has caused the increased number of RF over CPU generations [1][2]. Among the main design blocks in CPU, RFs thus contribute substantially to total CPU power. On the 32nm Westmere (WSM) IA Core [1], RFs accounted for 30% of leakage and dynamic power on a typical benchmark (Figure 6.1).

With its substantial contribution in total power, a number of power saving ideas, focusing especially on RF read power reduction, have been explored [4]. Within RFs however, the largest single contributor

to dynamic power is write data distribution. Write data distribution power has been increasing over process generations as wire scaling challenges and increasing min-vcc constraints have necessitated the use of dual-ended write SRAM bitcell topologies [5].

Figure 6.2 shows power distribution in a 32nm IA core RFs. The write local bitline is the highest dynamic power stage contributing about 18% of total RF dynamic power. The write data distribution (global and local) accounts for 25% and 22% of RF dynamic and leakage power respectively. The SRAM bitcell (excluding read ports identified as "ReadLocaBitline") accounting for 39% of the total RF leakage.

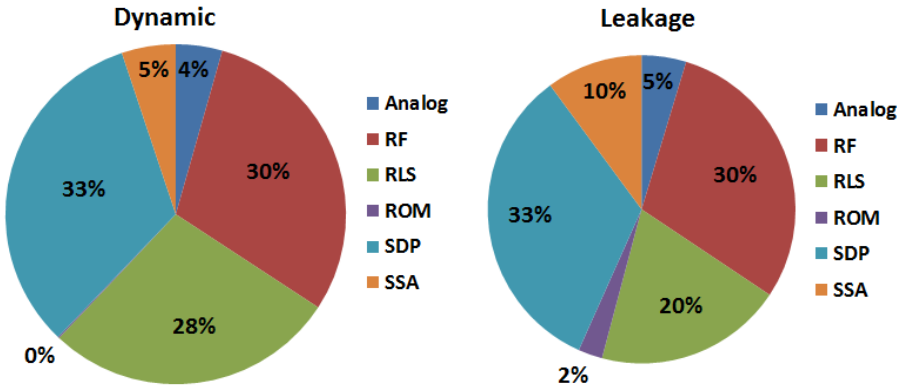


Figure 6.1 Dynamic and Leakage power distribution in 32nm WSM IA Core by design styles: Register Files (RF), Synthesized Block (RLS), Read Only Memory (ROM), Structural Data Path (SDP), Small Signal Arrays (SSA)

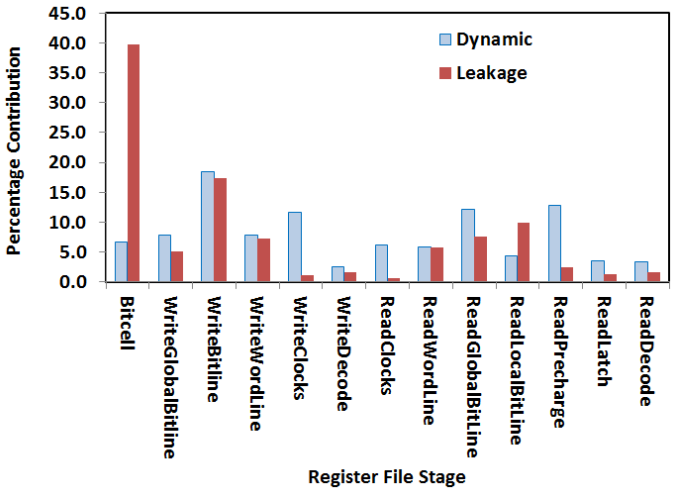


Figure 6.2 Leakage and Dynamic power distribution in 32nm WSM core Register Files

In this chapter, we will focus on write data distribution power. Bitline segmentation to reduce dynamic power has been used on both RF read and write [6]. Other than bitline segmentation, reducing voltage swing on the bitline is also an effective way to decrease power dissipation during writes operation, but it involves additional complexity and challenges [7]. Write data gating with simple logic gate on the other hand is a straightforward static CMOS design, effective yet easy to implement.

In section 2, we explore different write data gating circuit topologies (NAND, NOR, Tri-State) based on the write data static probabilities as well as the granularity of the gating (Global, Midway, Local) permitted by the area to achieve up to 96% of total write bitline dynamic power reduction. Furthermore, we examine the leakage implications of data gating and present techniques to reduce leakage through device stack-forcing [8][9] and state-forcing based on write data and SRAM bit node probable logic state residency (signal probability). We show that the write data and memory data signal probability should be a consideration in the selection of data gating strategy. Finally, in section 3, we present a simple and accurate (±5% error) write data gating Break-Even Analysis Model for estimating write data gating implementation gain and evaluation of multiple options, taking into consideration both circuit implementation, architectural, and write data characteristics.

6.2 WRITE DATA GATING

Write data distribution presents challenges to large RF arrays. Due to performance and write-ability constraints, global and local data buffering are typically needed. The write data is segmented into local (WrBL) and global (WrGBL) bitlines (Figure 6.3). Each local WrBL drives a sub-set of entries called "bundle". The number of entries per bundle (segment bundle size) is usually influenced by memory write-ability, physical and logical memory size, area, timing, and power constraints. For example, a 128 entry array can be implemented as either 8x16 (8 entries per bundle by 16 bundles), 16x8, 32x4, or 64x2 bundle segmentations. Increasing bundle size reduces the area cost of the data distribution. However, this also poses bitcell write-ability challenges. In a 32nm design, segment sizes of 8 and 16 are commonly used.

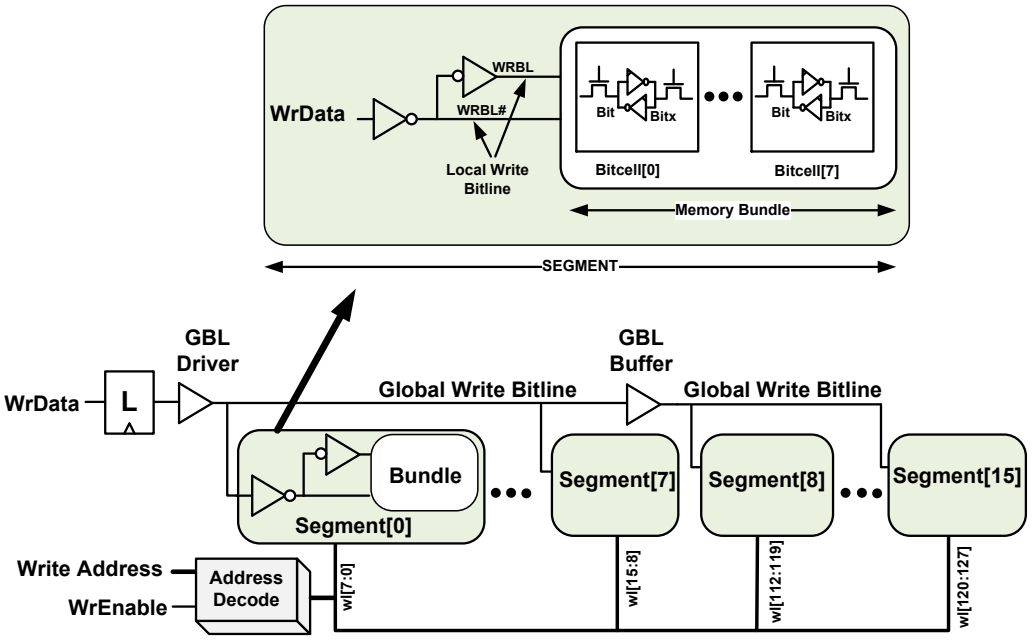


Figure 6.3 A 128-entry RF write data distribution showing global and local bitline segmentations. Each segment is an 8-entry bundle driven by a local write driver

Each memory bitcell presents a switching capacitance (diffusion load) by way of the memory pass-gate connectivity to the local bitline driver. Furthermore, the write data buffers (global and local) increase the switching capacitance and leakage associated with data distribution. These devices in addition to the wire load constitute a large switching capacitance anytime the write data toggles. To reduce the switching capacitance, data gating at multiple stages along the data distribution path can be implemented. At each stage area, delay, timing, and power tradeoffs are critical factors that drive the choice of data gating strategy. The three main data gating strategies are: (1) Global Bitline Gating (2) Mid-Way (Partial) Global Bitline Gating and (3) Local Bitline Gating.

6.2.1 Global Bitline Data Gating

Global data gating is needed when the same write data bus is shared by multiple array blocks with independent access. Typically, array data busses have multiple destinations. The data can be toggling any time there is a write to any of the destination blocks.

Global data gating prevents the switching of the local distribution in blocks that are not the intended recipient of the data. A common case involves a write data bus shared by multiple ways of an array (Figure 6.4). Each Way receives the same write data bus but only one way is intended to be written to at any time. The three common gating options on the global write data are (Figure 6.5): (i) Data Latch Clock Gating (ii) Pre-Latch Data Gating and (iii) Post-Latch Data Gating.

Timing is a primary constraint in deciding how early the data can be gated.

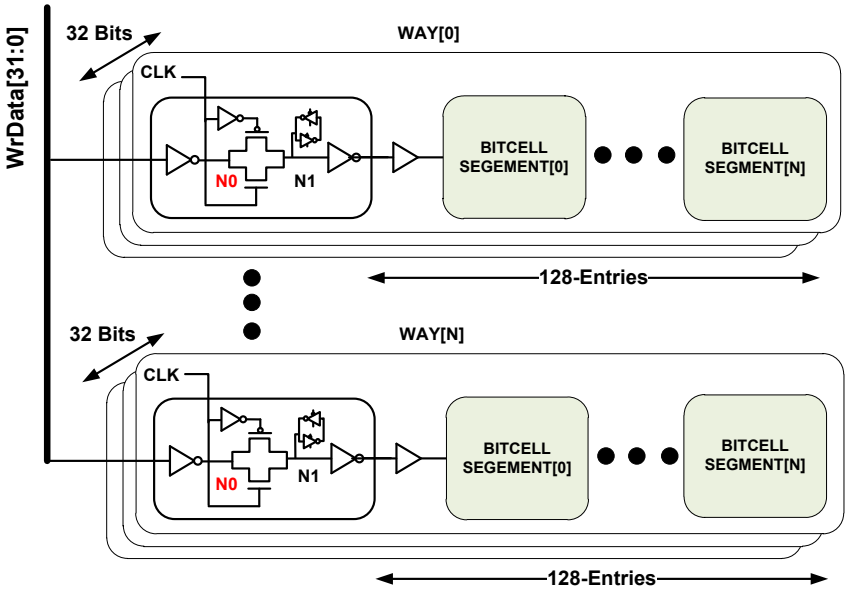


Figure 6.4 An N-way 32bit x 128 entry array. All N-Ways share the same write data bus and therefore receives every data on the bus regardless of the intended destination

6.2.1.1 Global Write Data Latch Gating

The global write data latch clock is gated with an "Enable" which can be a global write enable or a more granular enable such as Write Way Enable. This closes the data latch and prevents WrGBL from toggling whenever WrData toggles unless the block is the intended data destination. The main constraint with latch clock gating is that the enable signal is required a phase before the opening edge of the write latch clock (Figure 6.5 (b)).

6.2.1.2 *Pre-Latch Global Bitline Data Gating*

While the clock gating closes the latch and prevents spurious toggling of the global bitline, it does not prevent the toggling of the internal latch node "*N0*" (Figure 6.5 (a)). This internal node toggling can be significant power dissipation in active and highly distributed write data bus. Pre-Latch data gating prevents this internal latch node from toggling. The pre-latch timing is also relatively relaxed compared the clock gating as the Enable can arrive late and still transparently pass through the latch active window without timing failure (Figure 6.5 (b)).

6.2.1.3 *Post-Latch Global Data Gating*

The post-latch gating has the significant advantage of relaxed timing constraint as the enable signal does not need to setup to the latch. However, it does not reduce the latch power dissipation.

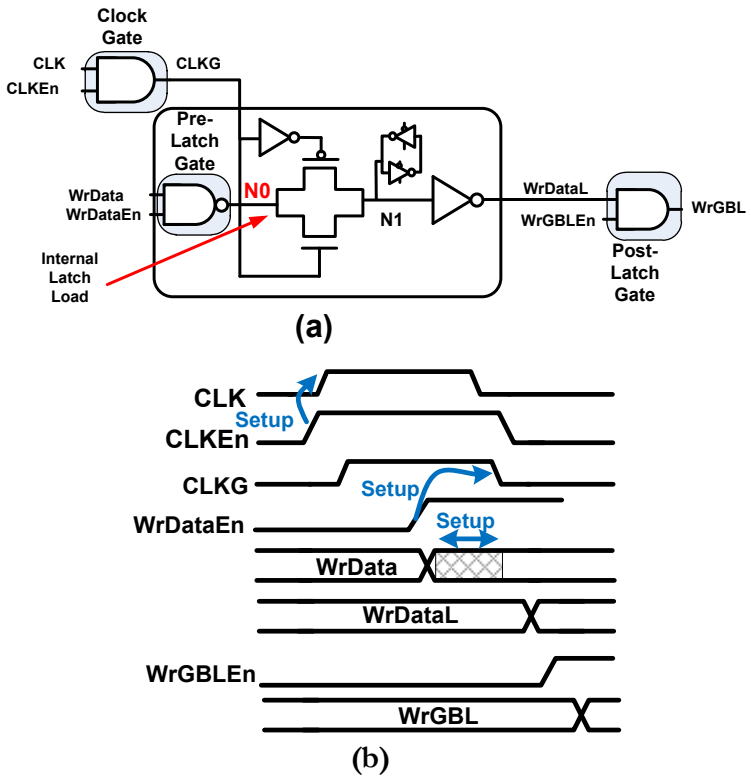


Figure 6.5 (a) Clock Gate, Pre-Latch, and Post-Latch global write data gate options and (b) timing waveforms

6.2.2 Mid-Way Global Bitline Data Gating

Mid-Way gating (Figure 6.6) is a variant of the global data gating that rely on the address to segment the global bitline into two or more segments. The purpose of this is to prevent a fraction of the global bitline from toggling. The MSB of the address is used to gate the activation of the upper address space. If the entry to be written is in the lower address space, the upper segment is inactive, preventing switching. However, if there is a write to the upper address space, there is no power saving as both the upper and lower bitline segments will toggle. This is especially useful if local data gating is not present due to area constraint. Mid-Way gating can be implemented in addition other global gating options.

6.2.3 Local Write Bitline Data Gating

Local write data gating is the finest gating granularity (Figure 6.6). WrBLs are gated per segment using the decoded address. Only the segment that contains the entry to be written is activated for write, decoupling it from the rest of the segments which will remain inactive and hence will not toggle. The timing constraint is relatively more relaxed than global gating.

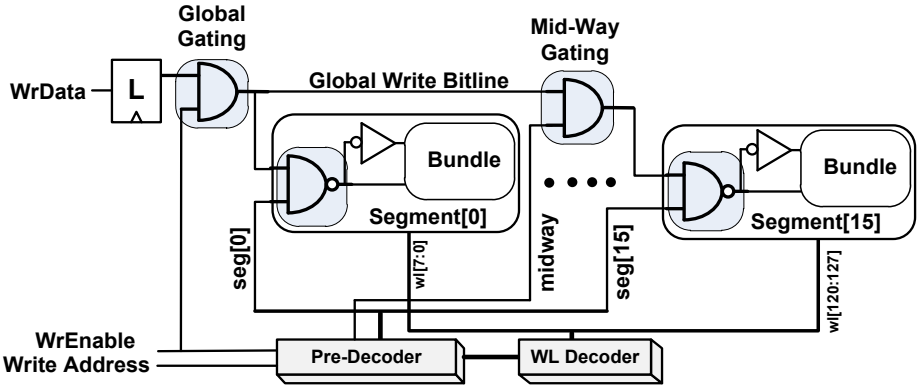


Figure 6.6 Global, Mid-Way, and Local Write Data Gating.

Local write data gating is the most effective data gating strategy due to the high local bitline capacitance. The dynamic reduction depends on the granularity of segmentation. More power reduction is achieved with increased segmentation but at an area cost. Figure 6.7 shows the local bitline dynamic power of a fully extracted 128x32 array with different segment counts and NAND gating. Multiple power benchmarks were run with different write data signal probability (SP). The dynamic power is reduced by

90% when the write bitline is segmented 16 times (8-entry per bundle) with average write data SP=0.5 compared to 38% reduction with 2 segments. The power reduction is also a function of data SP due to reset overhead discussed later. A greater than 2 segments are required to break-even if the SP is close to 1 for a NAND gating logic. The SP therefore becomes important for global gating which is generally a single segment gating.

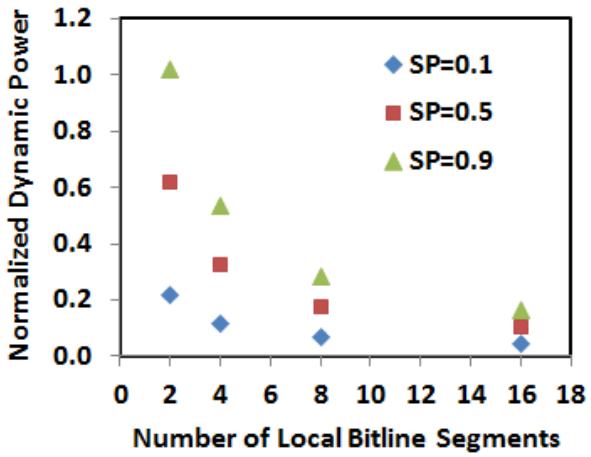


Figure 6.7 Gated local bitline dynamic power relative to un-gated bitline for 2, 4, 8, and 16-entry segment bundles

6.3 GATING LOGIC TYPE AND DYNAMIC POWER IMPLICATIONS

The effectiveness of data gating is dependent on the gating logic type and the write data characteristics. The characteristic of the write data that impacts data gating are: (i) activity factor (AF) – toggling rate, and (ii) signal probability (SP) – the probable logic state residency of the signal. The data SP relative to the gating logic determines the reset switching overhead. Any time the gating logic is de-activated, the logic output will be reset if the data at the output is opposite polarity of the logic reset state. A NAND gate output will reset to logic state "1", while a NOR will rest to "0". Therefore, the gating logic chosen should take into consideration the SP of the write data to reduce switching cost.

Figure 6.8 shows the write data SP for different RF blocks in a 32nm IA core across multiple benchmarks. (SP="1" implies WrData="1", SP="0" implies WrData="0"). While most RF's average

SPs across multiple benchmarks are less than 0.5, there are RFs with high average SPs. This could either be the artifact of the benchmark (random behavior) or systematic occurrence from the RF block designed to store opposite polarity of data. Therefore, the gating logic (NAND/NOR) for each RF block has to take into consideration the most probably data SP. Since the data SP is benchmark dependent and not entirely predictable, an average of a suite of representative application benchmarks are usually used to make the NAND/NOR implementation decision.

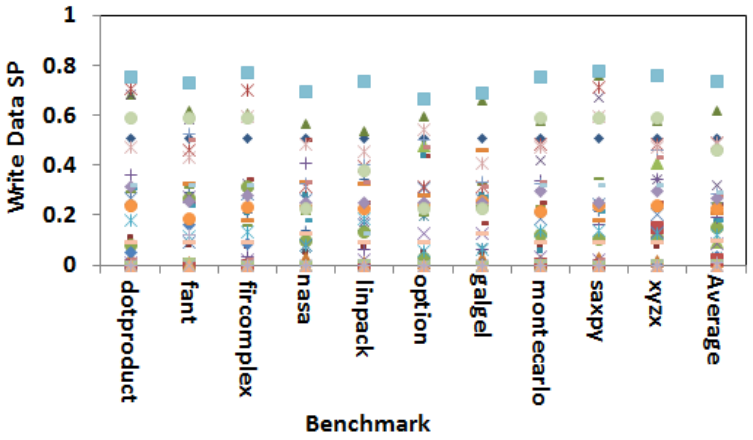


Figure 6.8 RF Blocks write data signal probability distributions for different benchmarks. Each key on the chart represents a separate RF block

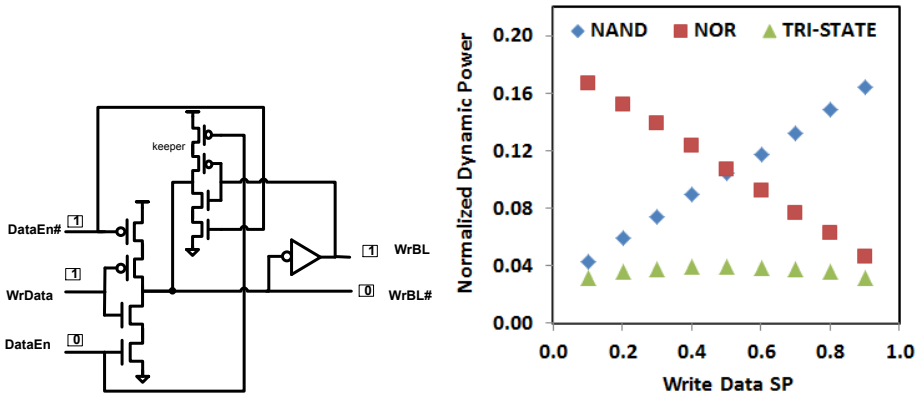


Figure 6.9 (a) Proposed write data tri-state (b) NAND, NOR, and Tri-State logic gating Write Data SP dependency.

We propose a tri-state gating (Figure 6.9 (a)) that eliminates reset overhead by using an internal keeper on the bitline to hold the written value without resetting. The keeper prevents a floating node input to the memory pass-gate. Figure 6.9(b) shows the local data gating dynamic power saving for varying write data SP using a NAND, NOR, and Tri-State logic on a fully extracted 32 bit x 128entry RF block with 16 segments (8-entry per bundle). Dynamic power reduction ranges from 83% to 96%. The maximum reduction using NAND logic is realized when write data average SP < 0.5. On the hand NOR logic gating is preferably at SP > 0.5. The proposed Tri-State is insensitive to SP, showing a consistent 96% reduction. The tri-state however require more area to implement than a NAND/NOR gating.

6.4 LEAKAGE IMPACT OF DATA GATING

Data gating impacts leakage in two ways (1) stack-forcing which reduces the data driver leakage and (2) state-forcing of the memory pass-gate input which impacts memory bitcell leakage.

6.4.1 Gating Logic Leakage and Stack-Forcing

Stack-forcing is a common leakage reducing technique [8]. Data gating enforces stacking depending on the write data logic state. The residency in the stack-forced state depends on the gating logic type used and the write data SP. A WrData="0" enforces stacking on a NAND logic while WrData="1" enforces stacking on a NOR logic (Figure 6.10). Furthermore, the NAND/NOR gating logic output state is deterministic due to the output reset (NAND="1", NOR="0"). This can be utilized to enforce stacking on the subsequent driver pair to further reduce leakage. Figure 6.12(c) (d) shows how this stack-forcing technique is used to further reduce the NAND write data gating leakage (NOR uses same technique).

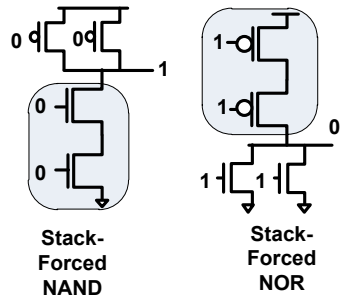


Figure 6.10 Stack-Forcing in NAND and NOR

6.4.2 Bitcell Leakage and State-Forcing

The bitcell leakage sources are shown in Figure 6.11. The bitcell leakage depends on the logic state of the write bitline (WrBL/WrBL#) relative to the internal bitcell node logic state (Bit/Bitx). If the internal memory node and bitline have the same polarity, the channel leakage across the pass-gate is shutoff. However, if the internal nodes have opposite polarity to the bitline, there is channel leakage across both bitcell pass-gates. Using a NAND/NOR logic state-forces the bitline to known states (Figure 6.12 (a)(b)). The leakage impact of gating is dependent on average "Bit" SP. A "Bit" node of SP < 0.5 favors NAND logic gating due to increased probability of same logic state across the pass-gate. On the other hand an SP > 0.5 favor a NOR implementation.

While WrBL logic state is deterministic due to state-forcing from enable reset, the state of the memory content ('Bit") is not deterministic across multiple benchmarks. There is a significant bitcell leakage range between the best case leakage condition (WrBL="0", Bit="0", or WrBL="1", Bit="1") and the worst-case leakage condition ("WrBL="0", Bit="1" or WrBL="1", Bit="0") as shown in Figure 6.13. Therefore, depending on the application benchmark, leakage could be significantly higher than anticipated. Given that the bitcell is the dominant leakage contributor in RFs, this could significantly impact the over array leakage

To force the bitcell leakage to a deterministic value, a back-to-back "NAND_NAND" or "NOR_NOR" logic is used to gate both WrBL and WrBL# (Figure 6.12 (e) (f)). This state-forces logic "1" (NAND) or "0" (NOR) on the input of both pass-gates, guarantee that only one pass-gate will leakage regardless of the bitcell "Bit" state. This enforces a predictable bitcell leakage. Figure 6.12(g) shows a device sharing technique to reduce the area overhead.

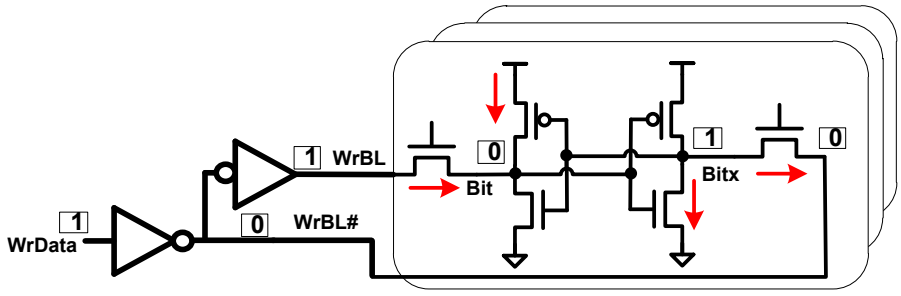


Figure 6.11 Memory leakage sources and worst-cast condition. Both pass-gates are leaking

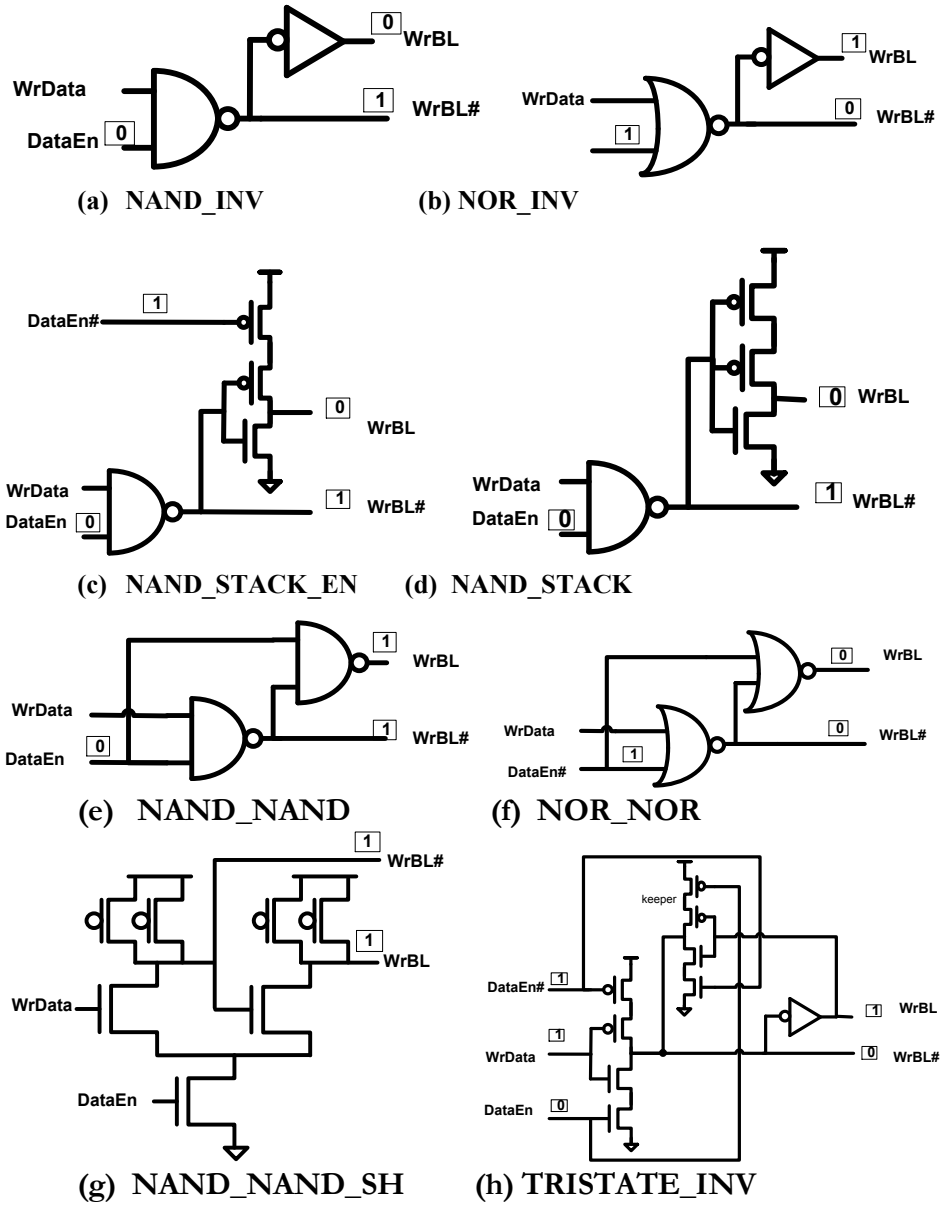


Figure 6.12 Stack-forcing and state-forcing gating techniques. (a) NAND (b) NOR (c) NAND with stack-forced pair (d) NAND with stacking (e) NAND state-forced pair (f) NOR state-forced pair (g) NAND sharing (h)Tri-State with state retention

6.4.3 Simulation Result Comparison

Figure 6.13 compares the impact of the various data gating techniques on the SRAM bitcell leakage. The worst-case leakage is as high as 2.7X relative to the best case leakage for conventional NAND/NOR. State-forcing with "NAND_NAND" or "NOR_NOR" results in a deterministic bitcell leakage.

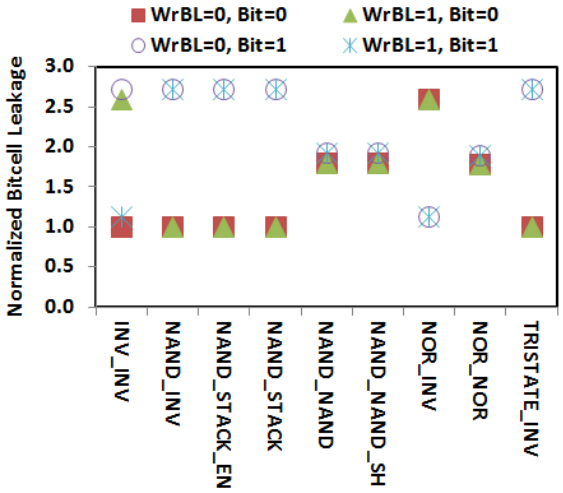


Figure 6.13 Impact of "Bit" and "WrBL" logic state on bitcell leakage for different gating techniques.

Tri-State leakage exhibits similar characteristic as the NAND. Even though it holds the last written value, since "WrBL" is shared by other entries of the bundle, this value could be opposite the polarity of other bitcells of the same segment. However, it guarantees that at least the last bitcell written to does not leak across the pass-gate, reducing the probability of worst-case leakage condition across all bitcells.

Figure 6.14(a) shows the comparative driver leakage of the write data gating techniques. The best-case leakage condition for stack-forced NAND gating ("NAND_STACK" and "NAND_STACK_EN") is 60% less than the reference "INV_INV". They also show 20% improvement over the conventional "NAND_INV" in both best and worst-case leakage scenarios. The comparative delay and area of the various data gating implementations is shown in Figure 6.14(b). The same effective driver sizes were used for all gating logic; as such stacked devices incur a higher area overhead for an equivalent INV_INV delay.

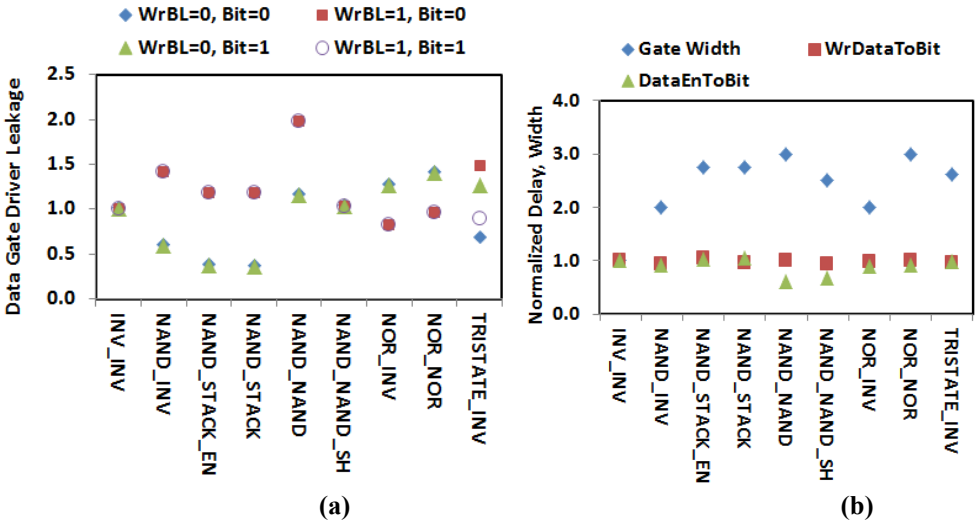


Figure 6.14 Data gating techniques leakage, area, and delay tradeoffs.

6.5 DATA GATE BREAK-EVEN MODEL

We model data gating break-even by signal transition counts. Each data bit is sub-divided by entry into unit load entities (Figure 6.15). Each unit load entity is capable of independently switching (transition) and hence can be independently gated. An M bit x N entry array will have *M x N* independent switchable entities. Our model uses the following parameters:

N_{bits}: Number of Data Bits

$N_{entries}$: Number of Entries

$N_{segment}$: Number of Data Gated Segments

AF_{data} : Average Data Activity Factor

SP_{data}: Average Data Signal Probability

AF_{En}: Enable AF,

SP_{En} : Write Enable Effective SP

N_{En}: Number of enable signals, NAND/NOR=1, Tri-State=2

α : This models the increase in bit load cap by the gating logic devices, captured as additional entity load cap.

β: This is loading on the enable from gating. It is normalized to the entity load

$\rho(0,1)$: Data gate logic Enable active (ON) polarity (NAND/Tri-State=1, NOR=0)

$P_r(0,1)$: Reset penalty, indicate whether reset penalty should be applied.

$P_g(0,1)$: Glitch p0enalty indicates whether glitch penalty should be applied.

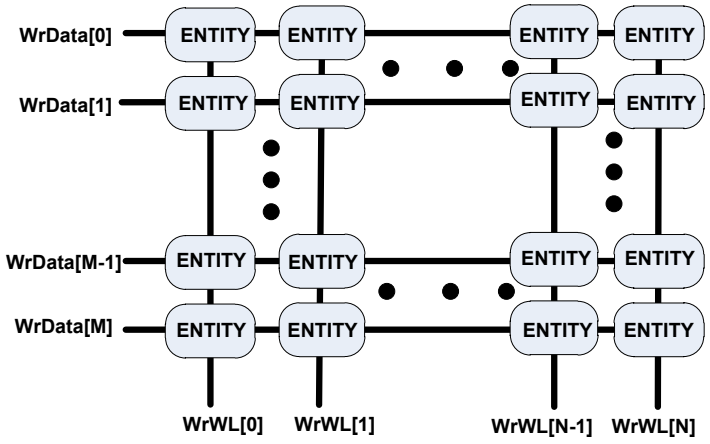


Figure 6.15 An M bit x N entry arrays. Each associated bitcell load is an entity capable of independently switching and can therefore be independently gated

6.5.1 Active Data Transition

Each data bit that can transition is considered an "Active Data". The number of transitions is determined by signal activity factor (AF). An AF=1 is considered as 2 transitions. The total number of active transitions of N_{bits} data per entry entity is given by:

$$DT_{active} = 2 \times AF_{data} \times N_{bits} \quad 6.1$$

6.5.1.1 Un-Gated Active Data Transitions

For an un-gated data, all entry entities are active and can transition during a write. The total number of transitions is:

$$DT_{ungated} = DT_{Active} \times N_{entries} \tag{6.2}$$

6.5.1.2 Gated Active Data Transitions

When the gating logic is activated by the "Enable" signal, only one segment will be active during access. Segment activation is assumed to be mutually exclusive. Therefore only entities of the activated segment can transition. The total gated active transition:

$$DT_{gated} = \left(\frac{N_{entries}}{N_{segment}} + \alpha\right) \times DT_{active} \tag{6.3}$$

where $\propto = \frac{Data\ Gate\ Device\ Load\ per\ Segment}{Data\ Bit\ Load\ per\ Entity}$

The logic gate "Enable" signal probability (SP) modulates post-gating transitions depending on its polarity relative to the logic gate. This is captured as:

$$DT_{gatedEn} = DT_{gated} \times [\rho \times SP_{En} + (1 - \rho)(1 - SP_{En})] \tag{6.4}$$

6.5.2 Data Gating Transition Overhead

There are three data gating transition overheads (i) Reset (ii) Glitch and (3) Enable signal transitions.

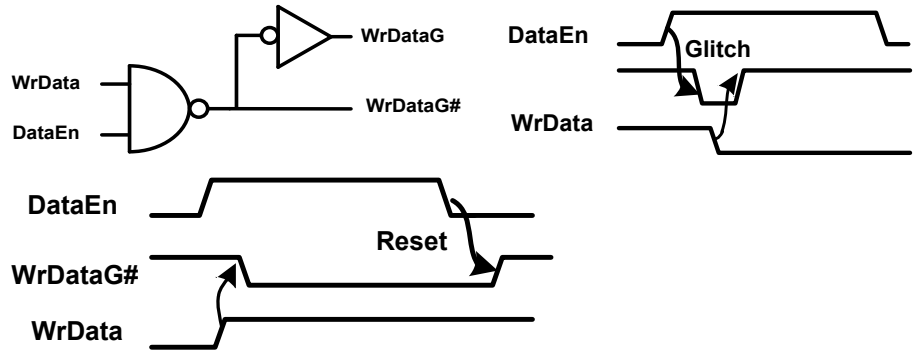


Figure 6.16 NAND Data gate glitch and Reset overhead

6.5.3 Reset Overhead

Any time the data logic enable signal is turned OFF, all gated output are reset to "1" (NAND) or "0" (NOR). Thus there is a reset cost (Figure 6.16) associated with the NAND/NOR gating active segment that manifest itself if the logic gate output data polarity is opposite the reset polarity. Tri-State output is not reset. The reset overhead is modeled as:

$$DT_{reset} = P_r \times AF_{En} \times \left(\frac{N_{entries}}{N_{segment}} + \alpha\right) \times N_{bits} \times [\rho \times SP_{data} + (1-\rho)(1-SP_{data})] \quad 6.5$$

6.5.4 Glitch Overheard

When the gated logic is enabled, the output will transition if the input and output (from previous reset) have the same polarity (Figure 6.16). This transition is wasteful if the gating logic "Enable" signal arrives before the new write data. If the data arrives before the "Enable", then no glitch penalty is incurred because the transition that occurs is the actual useful logic transition.

$$DT_{glitch} = P_g \times AF_{En} \times \left(\frac{N_{entries}}{N_{segment}} + \alpha\right) \times N_{bits} \times [\rho \times SP_{data} + (1-\rho)(1-SP_{data})] \quad 6.6$$

6.5.5 Enable Signal Transition Overhead

This is the transition overhead of the enable signal. The overhead cost of activating the segment.

$$ET_{overhead} = 2 \times AF_{En} \times N_{bits} \times N_{En} \times \beta \tag{6.7}$$

$$\beta = \frac{Enable\ Signal\ Load\ per\ Segment\ per\ Data\ Gate\ Logic}{Data\ Bit\ Load\ per\ Entity}$$

6.5.6 Gated To Un-gated Transition Ratio

Total Number of Gated Transition:

$$DT_{totalgated} = DT_{gatedenable} + ET_{overhead} + DT_{reset} + DT_{glitch} \tag{6.8}$$

The ratio of un-gated to gated transition is:

$$GT_{ratio} = \frac{DT_{totalgated}}{DT_{ungated}} \tag{6.9}$$

6.6 MODEL VALIDATION

The model was validated with a fully extracted layout of a 128entry x 32 bit RF array. We validated the impact of number of segments on break-even point by implement the array as 2x128, 4x64, 8x32 and 16x8 gated segments. We implemented multiple versions of the gating with NAND, NOR and Tri-state logic. We then generated power benchmark traces with stimulus of different AF and SP using an in-house production power analysis tool. We compared GT_{ratio} from the power benchmarks with our model estimation. The model error was ±5% (Figure 6.17) showing good prediction accuracy.

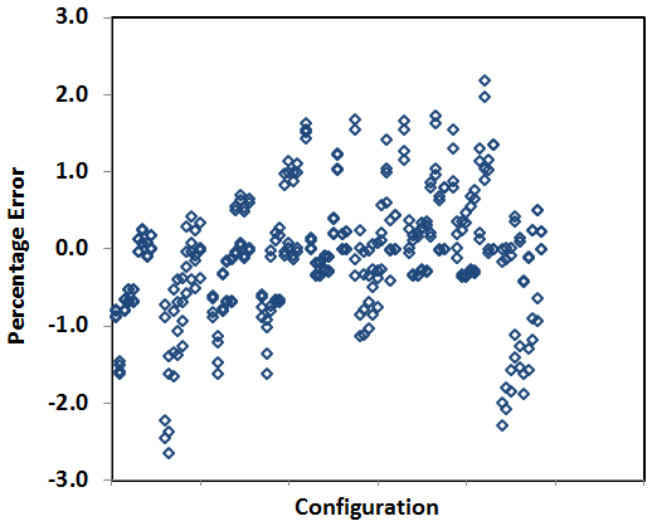


Figure 6.17 Error distribution of data gate model validation

6.7 CONCLUSION

Write data distribution is shown to be the highest power consuming stage in a 32nm WSM IA core RFs. We've presented RF write data gating strategies and explored how global and local data gating can be used to reduce write data distribution dynamic power by as much as 96%. The write data SP is shown to be a relevant consideration in the selection of a NAND or NOR data gate implementation due to reset overhead. A Tri-State logic data gating however is independent of data SP and therefore should be the preferred consideration for low dynamic power implementation. This however comes at an area cost. We further showed that using stack-forcing and state-forcing techniques, both write bitline driver and memory leakage can be reduced. In general, area and timing constrain the granularity of data gating. The ideas presented show how to maximize the power benefit in a tradeoff with area and timing.

We also presented a simple and accurate data gating break-even analysis model for determining the best implementation strategy taking into consideration both the Data and Enable characteristics, logic implementation, and overheads. The model can accurately predict data gating power reduction within 5% error margin.

6.8 ACKNOWLEDGEMENT

I will like to acknowledge my co-workers Teck Siong Ong and Yan Nee Too for their invaluable contribution to this study.

6.9 RELATED PUBLICATIONS

[1] Eric Donkoh, Teck-Siong Ong, Yan Nee Too, Patrick Chiang, "Register File Write Data Gating Techniques and Break-Even Analysis Model" *ISLPED 2012*

6.10 REFERENCE

[1] N. A. Kurd et al., "Westmere: A family of 32 nm IA processors," *IEEE Int. Solid-State Circuits Conf. (ISSCC),* San Francisco,CA, Feb. 2010.

[2] R. Kumar et al., "A family of 45nm IA processors," *ISSCC'09 Digest of Technical Papers*, pp.58-59, Feb. 2009

[3] Chang, L et al., "An 8T-SRAM for Variability Tolerance and Low-Voltage Operation in High-Performance Caches" *IEEE Journal Solid-State Circuits, Apr 2008*

[4] S. Borkar, "Circuit techniques for subthreshold leakage, avoidance, control, and tolerance," *IEDM Tech. Dig., pp. 421-424,Dec, 2004.*

[5] K. Anshumali et al. " Circuit And Process Innovations to Enable High-Performance and Are Efficiency on the Nehalem and Westmere Family of Intel processors" *Intel Technology Journal, vol 14, pp 104-127*

[6] G. Kucuk et al: "Energy–Efficient Instruction Dispatch Buffer Design for Superscalar Processors*" *ISLPED 2001*

[7] K. Kanda et al. "90% Write Power-Saving SRAM Using Sense-Amplifying Memory Cell", *IEEE Journal Solid-State Circuits, Jun 2004*

[8] S. Narendra, et al, "Scaling of Stack Effect and its Application for Leakage Reduction", *ISLPED, 2001*

[9] C.Hua et al. " Distributed Data-Retention Power Gating Techniques for Column and Row Co-Controlled Embedded SRAM" *Memory Technology, Design and testing, 2005*

7 LOW POWER REGISTER FILE DECODER

ABSTRACT

Address decoding is a common logic function for memory operation. A decoder that simultaneously reduces area, timing, and power using device sharing and power gate techniques is presented. It exploits the mutual exclusivity of the decoding logic outputs and shares devices to achieve as much as 30% NMOS (PMOS) device width reduction of a 3-to-8 NAND (NOR) decoder at no delay penalty in a 32nm technology. A decoder power gate scheme that eliminates floating outputs through state-forcing reduces leakage by 90%. Analysis of the shared decoder and sizing guidelines using simple Elmore Delay model are also presented.

Keywords: Low-Power, S-Decoder, Power Gating, Leakage, Register File, SRAM, Dynamic Power.

7.1 INTRODUCTION

In the era of multi-core computing, power has become a more important factor in microprocessor design as frequency alone is no longer a measure of processor performance. Memory power reduction is of particular importance because they constitute a large proportion of processors silicon budget. To this end powers saving techniques are being explored that are either circuit or process technology based. Process technology techniques for leakage reduction include long channel (Le) and high-Vt device usage. Circuit based techniques including power gating have also been explored [1-4].

A memory array is usually organized into data bits and entries with an entry representing all data bits that are accessed at a time. To access the memory element, an encoded address has to be decoded to select a distinct entry from the array. Address decoding is thus one of the common logic operations in memory access. This chapter presents an address decoding scheme that uses devices sharing to reduce the decoder area by up to 30%. It also explores how power gating can be integrated into the proposed design to reduce decoder leakage by as much as 90%, at no area penalty compared to a conventional decoder.

The organization of the chapter is a follows: Section II reviews the conventional decoder and power gating. Section III discusses the shared decoder (S-Decoder) and provides delay and sizing analysis.

Simulation results of the S-Decoder are also presented in this section. Section IV discusses power gating application to the S-Decoder. Conclusions are presented in section V.

7.2 BACKGROUND

7.2.1 Conventional Decoder

Decoders are typically implemented with NAND and NOR static logic (Figure 7.1). For brevity, only the NAND decoder will be discussed. Same approach applies to the NOR decoder where the NMOS and PMOS devices are interchanged. An address of *n* bits is decoded into 2^n outputs. Figure 7.1(a) shows the circuit for NAND-INVERT decoding of a 3 bit address requiring eight 3-input NAND and eight output inverters. The number of NAND PMOS devices depends on whether the decoder is implemented as static or dynamic logic. The total number of NMOS devices needed to implement the NAND logic of a static 3-to-8 decoder will be *3bits x 8 outputs*. An additional *2x8* devices are needed for the final inverter in the NAND-INVERT decoder. The decoder device count can be very high in large arrays with large address space. Sizing of the decoder devices is determined by path timing criticality, area constraints, and power. In a typical decoder, the NAND output falling transition is the timing critical transition since that edge initiates memory access. Hence the NMOS devices of the NAND are sized for speed.

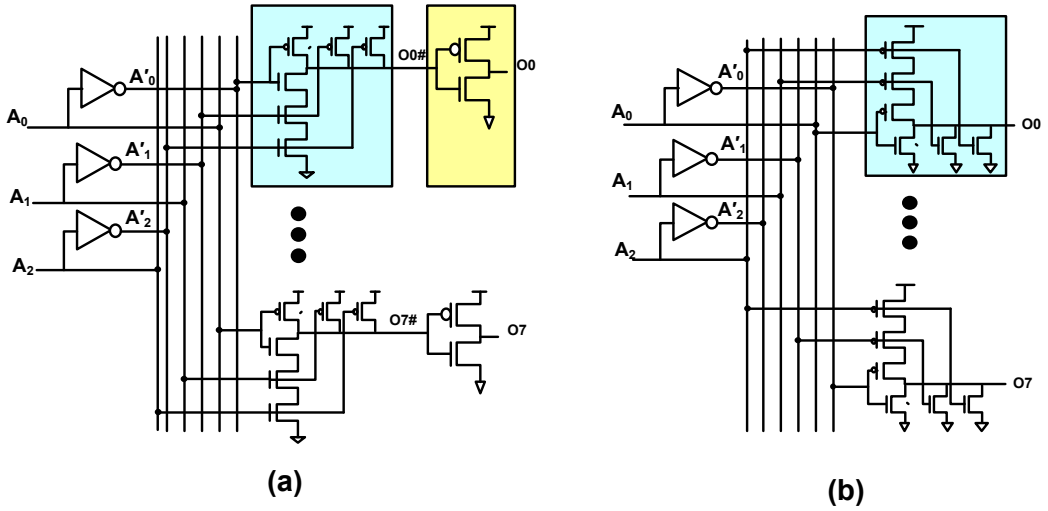


Figure 7.1 Conventional address decoder (a) static NAND-INVERT decoder (b) static NOR decoder

7.2.2 Transistor Stacking and Power Gate Circuit

Stacking reduces leakage by reducing the voltage across an OFF transistor. The principle behind stacking is illustrated with a NAND gate with two stack NMOS devices in Figure 7.2.

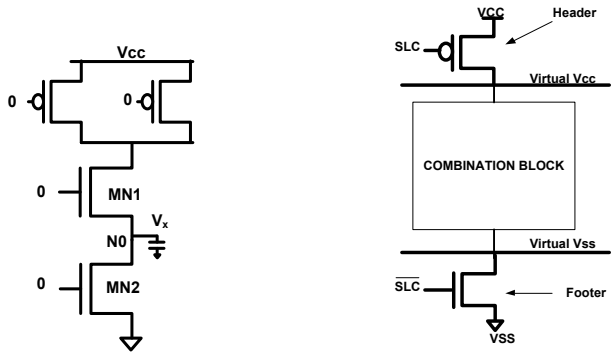


Figure 7.2 (a) Stack-forced NAND gate (b) Power gate logic using header and footer

When both series connected n-devices are turned off, the internal node *N0* finds an equilibrium point V_x at which point the leakage current of the two series devices MN1 and MN2 are equal. The intermediate voltage value is given in [6]:

$$V_x = \frac{\lambda_d V_{dd} + S\, log \frac{W_u}{W_l}}{1 + \gamma_b + 2\lambda_d} \qquad 7.1$$

where w_u, w_l are the width of the upper and lower stacked devices respectively. V_x is assumed to be greater than 3kT/q

The presence of V_x: (1) reduces the drain-source voltage of both transistors and hence reduces the leakage current due to Drain-Induced Barrier Lowering (DIBL). This is more pronounced for the bottom device MN2; (2) the gate-source voltage of the top device MN1 is negative while its body becomes reverse biased thereby raising its threshold and exponentially reducing leakage. The stack factor of a transistor stack with width w_u and w_l relative to reference inverter of width w is given by [6]

$$\frac{I_{inv}}{I_{stack}} = \frac{w}{w_u^{\alpha} w_l^{1-\alpha}} 10^{\frac{\lambda_d V_{dd}(1-\alpha)}{S}} \text{; where } \alpha = \frac{\lambda_d}{1+2\lambda_d} \qquad 7.2$$

Power gating enforces stacking by shutting off current paths of inactive logic to save leakage. An NMOS or PMOS (or both) is inserted in series with active logic block. Power supply to the logic block is then controlled as follows. In normal inactive mode, when power gate control signal PGC = "0", the NMOS footer and PMOS header devices are turned OFF, shutting off the power supply to the logic. In active mode, PGC = "1" and a virtual VSS and VCC is established for the combinational logic to operate normally. Typically either header or footer gating is sufficient to reduce leakage significantly, especially if the combinational logic is in a known state as with decoders.

7.3 SHARED DECODER

7.3.1 Shared Decoder (S-Decoder) Circuit

The outputs of a decoder are mutually exclusive. The NAND output of a NAND-INVERT decoder is mutually exclusive "0 while the inverter output is a mutually exclusive 1" logic state. This mutual exclusivity can be used to reduce the number and size of devices needed to perform a decoding operation by sharing devices between logic outputs. The decoder device sharing concept is best illustrated with a 3-to-8 decoder. The logic operation of a decoder with inputs address bits A_0, A_1, and A_2, and outputs O_0-O_7 is defined by the following Boolean functions:

$$O_0 = A_2'A_1'A_0' = S_0^4A_1'A_0' = S_0^2A_0' \quad \textit{where} \quad S_0^2 = A_2'A_1'$$

$$O_1 = A_2'A_1'A_0 = S_0^4A_1'A_0 = S_0^2A_0 \qquad S_1^2 = A_2'A_1$$

$$O_2 = A_2'A_1A_0' = S_0^4A_1'A_0' = S_1^2A_0' \qquad S_2^2 = A_2A_1'$$

$$O_3 = A_2'A_1A_0 = S_0^4A_1'A_0 = S_1^2A_0 \qquad S_3^2 = A_2A_1$$

$$O_4 = A_2A_1'A_0' = S_1^4A_1'A_0' = S_2^2A_0'$$

$$O_5 = A_2A_1'A_0 = S_1^4A_1'A_0 = S_2^2A_0$$

$$O_6 = A_2A_1A_0' = S_1^4A_1'A_0' = S_3^2A_0'$$

$$O_7 = A_2A_1A_0 = S_1^4A_1'A_0 = S_3^2A_0$$

7.3

In this notation, S^4 denotes logic shared by 4 outputs and S^2 denotes logic shared by 2 outputs. For example, the logic state of S_0^4 is the same for decoder outputs O_0-to-O_3. Since only one of outputs O_0-to-O_3 will make a "1" transition at a time, they can share the same S_0^4 logic. Similarly outputs O_4-to-O_7 can share logic S_1^4. Extending this to all levels O_0-to-O_2 can share logic S_0^2 and so on.

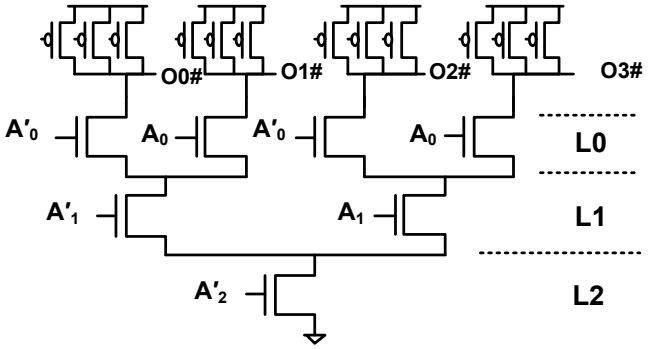


Figure 7.3 Device sharing for outputs O_0-to-O_3 of a 3-to-8 NAND S-Decoder. Similar sharing is used for O_4-O_7

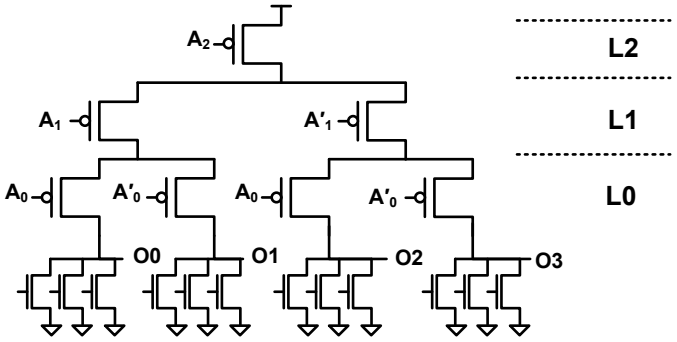


Figure 7.4 Device sharing for outputs O_0-to-O_3 of a 3-to-8 NOR S-Decoder. Similar sharing is used for O_4-O_7

The S-Decoder device sharing scheme for outputs O_0-to-O_3 of a 3-to-8 NAND decoder logic is shown in Figure 7.3. The same circuit is replicated for outputs O_4-to-O_7. A NOR decoder employ the same scheme by sharing the PMOS devices for mutual exclusive "1" outputs (Figure 7.4). A similar device sharing approach is used for any order address decoding by device sharing at different levels. For example, a 2-to-4 address decode will use the same sharing scheme shown for input A_0-A_1 (L0 to L1) while a 4-to-16 decoder will use further sharing beyond level L2. The maximum number of outputs that can share a device at level Ln is 2^n.

7.3.2 Shared Decoder Configurations

There are number of possible shared decoder configurations. The maximum number of devices that can be shared at level N-Ln or P-Ln is 2^n. However, any number of devices less than the maximum can be shared at each level. Input A devices (N-L0 and P-L0) cannot be shared (1-way sharing). Input B devices (N-L1 and P-L1) can have two possible sharing configurations: 1-way (no sharing) and 2-way sharing. Input C devices (N-L2 and P-L2) can have four possible shared configurations: 1-way, 2-way, 3-way, or 4-way sharing. Input D devices (N-L3 and P-L3) can be shared eight possible ways: 1-way, 2-way, 3-way, 4-way, 5-way, 6-way, 7-way and 8-way.

There are two categories of decoder sharing: *uniform* and *non-uniform*. In uniform sharing, the number of ways a device is shared is a multiple of the maximum possible sharing at that level. For example, a 2-way sharing at level L2 is uniform because it's a multiple of the number of possible devices that can be shared, which is 8. With non-uniform sharing, the number of ways a device is shared is not a multiple of the maximum possible device sharing. This leads to irregular sharing as devices at the same level will have different sharing schemes. For example a 3-way sharing at level L2 is non-uniform because it not a multiple of maximum 8 possible shared devices.

A configuration scheme to denote the level of sharing is shown in Table 7.1- Table 7.3. A 2-to-4 decoder has a total of 2 possible configurations; a 3-to-8 decoder has 8 possible configurations (6 uniform configurations) while a 4-to-16 decoder has 64 possible configurations (24 uniform configurations). In Table 4 only the 24 uniform configurations are shown for the 4-to-16 decoder. Non-uniform configurations such as 3-way, 5-way, 6-way, and 7-way sharing are possible. Configuration S8421 denotes a 4-to-16 decoder sharing input D devices among 8 outputs input C devices among 4 outputs, input B devices among 2 outputs, and input A not shared. Similarly configuration S221 of a 3-to-8 decoder implies input C and B are both shared 2-ways and input A is not shared.

Table 7.1 Shared decoder 2-to-4 configurations

A_1	A_0	Configuration
1	1	S11
2	1	S21

Table 7.2 Shared decoder 3-to-8 configurations

A_2	A_1	A_0	Configuration
1	1	1	S111
1	2	1	S121
2	1	1	S211
2	2	1	S221
3	1	1	S311
3	2	1	S321
4	1	1	S411
4	2	1	S421

Table 7.3 4-to-16 Shared decoder uniform configurations. Non-uniform configurations are not shown here

A_3	A_2	A_1	A_0	Uniform Configurations
1	1	1	1	S1111
1	1	2	1	S1121
1	2	1	1	S1211
1	2	2	1	S1221
1	4	1	1	S1411
1	4	2	1	S1421
2	1	1	1	S2111
2	1	2	1	S2121
2	2	1	1	S2211
2	2	2	1	S2221
2	4	1	1	S2411
2	4	2	1	S2421
4	1	1	1	S4111
4	1	2	1	S4121
4	2	1	1	S4211
4	2	2	1	S4221
4	4	1	1	S4411
4	4	2	1	S4421
8	1	1	1	S8111
8	1	2	1	S8121
8	2	1	1	S8211
8	2	2	1	S8221
8	4	1	1	S8411
8	4	2	1	S8421

7.3.3 S-Decoder Sizing And Delay

A shared series chain device will drive ~33% more internal device diffusion capacitance compared to a regular series chain if all the devices in the chain are sized equally. We assume progressive sizing of devices along the series chain. We assume α scaling of every device in the chain relative to the device above it (Figure 7.5). Thus, device M2 is scaled α times M1 and M3 will be α times M2.

$$\left(\frac{W}{L}\right)_2 = \propto \left(\frac{W}{L}\right)_1; \qquad \left(\frac{W}{L}\right)_3 = \left(\frac{W}{L}\right)_2 = \propto^2 \left(\frac{W}{L}\right)_1 \tag{7.4}$$

The device capacitance (C) and resistance (R) are directly proportional to the device dimensions (W/L). The scaling of device sizes by α will result in α scaling of device capacitance and 1/α scaling of the device resistance as shown in Figure 7.5.

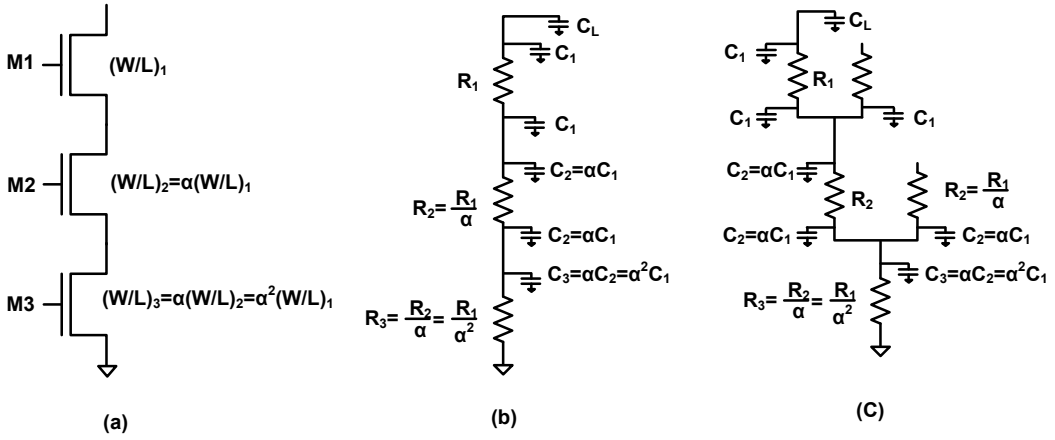


Figure 7.5 Scaling of series connected chain (a) device size scaling (b) conventional series chain scaling (c) shared series chain with progressive α scaling down the chain

The first order switching delay estimation for a series connected devices can be obtained using an RC network capacitive discharge rate with Elmore time constant τ:

$$V_o(t) = V_{max}e^{-t/\tau} \quad where \quad \tau = RC = \sum_{k=1}^{N} C_k \left(\sum_{m=1}^{k} R_m \right) \tag{7.5}$$

The above equation assumes all capacitances in the chain are initially charged to the same value. In reality the internal node is charged to a lower voltage than the output node due to threshold voltage loss and channel body effect. Using the simple Elmore estimation, the time constant for a conventional 3-series connected chain (Figure 7.5(b)) is:

$$\tau_{ref} = (C_1 + C_L)(R_1 + R_2 + R_3) + (C_1 + C_2)(R_2 + R_3) + (C_2 + C_3)(R_3) \tag{7.6}$$

$$\tau_{ref} = R_1C_1(9 + 3\beta) \qquad where\ C_L = \beta C_1 \text{ and } \alpha = 1,\ R_1=R_2=R_3 \text{ and } C_1=C_2=C_3 \tag{7.7}$$

The time constant of a shared 3-series chain with progressive α device size scaling of the series chain Figure 7.5(c) is given by:

$$\tau_{sh} = R_1 C_1 \left(3 + \frac{6}{\propto} + \frac{3}{\propto^2}\right) + R_1 C_L \left(1 + \frac{1}{\propto} + \frac{1}{\propto^2}\right) \quad 7.8$$

$$\tau_{sh} = R_1 C_1 \left(3 + \beta + \frac{(6+\beta)}{\propto} + \frac{(3+\beta)}{\propto^2}\right) \; where \; \beta = {}^{C_L}/_{C_1} \quad 7.9$$

Table 7.4 shows the time constant for shared 2, 3 and 4 series connected RC network chain. In Table 7.5 the α scaling of shared series-chain devices required to achieve the same delay as a regular series-chain is obtained by

$$\tau_{sh} = \tau_{ref} \quad 7.10$$

Table 7.4 Shared series chain RC network time constant

	Shared Series Chain RC network Time Constant
2-series chain	$\tau = R_1 C_1 \left(2 + \beta + \frac{(3+\beta)}{\propto}\right)$
3-series chain	$\tau = R_1 C_1 \left(3 + \beta + \frac{(6+\beta)}{\propto} + \frac{(3+\beta)}{\propto^2}\right)$
4-series chain	$\tau = R_1 C_1 \left(4 + \beta + \frac{(8+\beta)}{\propto} + \frac{(6+\beta)}{\propto^2} + \frac{(3+\beta)}{\propto^3}\right)$

Table 7.5 Shared series chain iso-delay sizing relative to conventional series chain guideline

	Iso-Delay Alpha Equation
2-seires chain	$(2+\beta)\alpha - (3+\beta) = 0$
3-series chain	$(6+3\beta)\alpha^2 - (6+\beta)\alpha - (3+\beta) = 0$
4-series chain	$(12+3\beta)\alpha^3 - (8+\beta)\alpha^2 - (6+\beta)\alpha - (3+\beta) = 0$

The α value is a function of the driven load cap to internal device cap ratio β. A plot of α vs. β is shown in Figure 7.6. A 3-series connected shared chain has iso-delay α-scaling value ranges from 1.37 (β=0) to 1.01 (β=100). With increasing external load (C_L), the additional internal load due to device sharing has less impact on the delay and approaches the conventional unshared series chain. It is assumed that each of the inputs is switching independently in this estimation. In a multi-switch case, the required α scaling will be higher. This is because in addition to the device sizing, the internal node voltage and the input stimulus also affect switching speed.

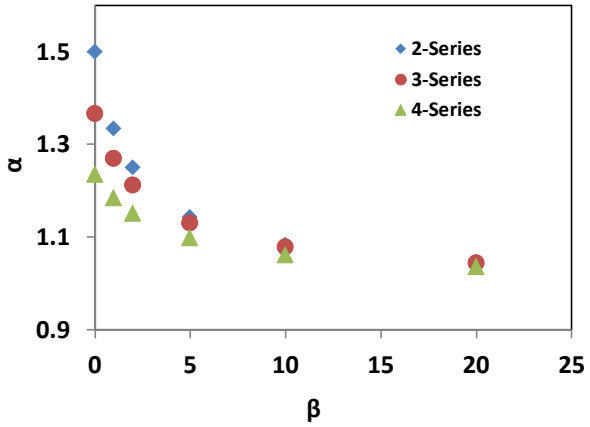


Figure 7.6 Shared series chain sizing $\boldsymbol{\alpha}$ value as a function of $\boldsymbol{\beta}$. As external load increases, the effect of internal cap decreases and $\boldsymbol{\alpha}$ approach 1.0

The worst case delay for a 3-series NAND (NAND3) occurs when all the inputs (A_0, A_1, A_2) switch simultaneously from "000" to "111". In a conventional NAND3, when the input stimulus is "000", the internal nodes, INT1 and INT2, voltages will discharge through the leakage path to an equilibrium voltage level close to zero (Figure 7.7(a)). However, in a shared NAND3, an input stimulus of "000" does not guarantee that the internal nodes are discharged to the leakage equilibrium voltage levels. It is possible to have an input stimulus of "000" and still have the internal node voltage charged to approximately Vcc-Vt through the side-branch paths. As shown in Figure 7.7(b) for output "O_7#", there exists a side-branch from "O_4#" and "O_6#" that charges the internal nodes INT1 and INT2 respectively to Vcc-Vt. Thus during an input transition from "000" to "111", the internal node voltages of the shared NAND will be higher than a regular NAND and hence, the discharge rate will be slower for size device sizes.

The optimal α value for a shared series is empirically obtained as the point of convergence when the delay is equivalent for all the inputs switching independently. Figure 7.8 shows the optimal size for a NAND3 obtained empirically with the convergence point occurring at α = ~1.4.

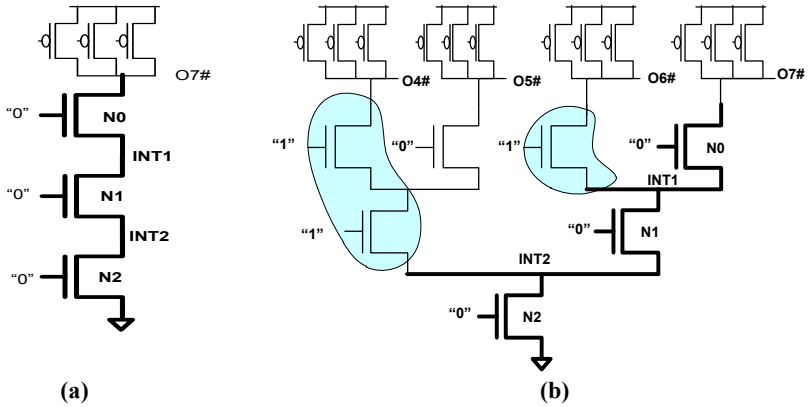


Figure 7.7 (a) Conventional NAND3 (b) Shared NAND - Side branch path in shared series chain charges the internal nodes INT1 and INT2 to higher voltage than standard NAND

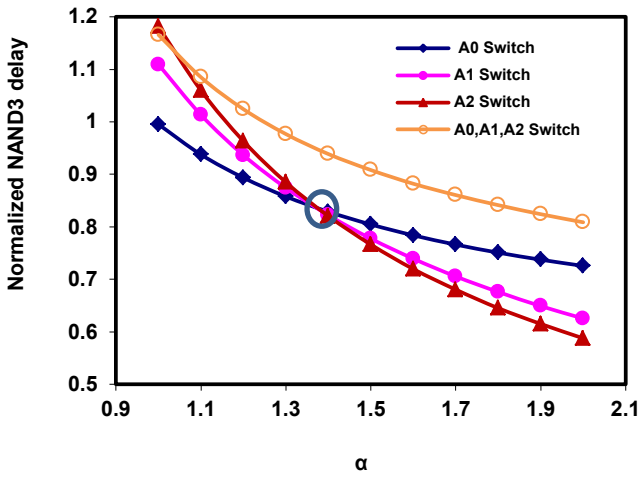


Figure 7.8 S-Decoder NAND3 optimal sizing delay Vs α with input switching independently and simultaneously. Delay is normalized to standard NAND3 delay.

7.3.4 Simulation Results

Circuit simulation was performed in a 32nm process technology. The S-Decoder outperforms the conventional decoder in area, and delay (with equivalent area). The area, delay, and power saving achieved depends on tradeoffs made against design constraints.

Figure 7.9 shows the NMOS width, delay, and leakage tradeoff of a NAND3 S-Decoder relative to a conventional decoder. For this comparison the same input slope is used for the delay simulations, ignoring slope improvement resulting from gate load reduction on the previous stage by sharing the device. In the simulation, input A_0 gates were kept at the same size as the reference, since there is no sharing of these devices. Inputs A_1 and A_2 gates sizing were bounded by their effective size and the total equivalent size relative to a reference conventional NAND. For example, devices shared by 2 outputs will have sizes varied from 1X (effective size) to 2X (total reference size), while devices sharing 4 outputs can have their size varied from 1X to 4X. The decoder NMOS device size can be reduced by 30% with virtually no impact on delay or leakage. Alternatively, a delay improvement of up to 40% can be achieved on an independently switching input at no area cost. However, this comes with a leakage cost due to device progressive sizing and the side branch effect. Leakage is reduced by taking advantage of the improved delay and using low-leakage devices. The S-Decoder delay improvement depends on the input transitions. The delay improvement is higher for each individual input switching independently (i.e. if the other inputs are already setup before that input transitions). All three inputs A_0, A_1, and A_2 switching simultaneously show the least amount of delay improvement over the conventional decoder due to side branch effect discussed earlier. Figure 7.10 shows the 4-to-16 S-decoder tradeoffs relative to conventional decoder. The area reduction increases as more devices are shared. The leakage overhead is reduced by taking advantage of the improved delay and using low-leakage devices.

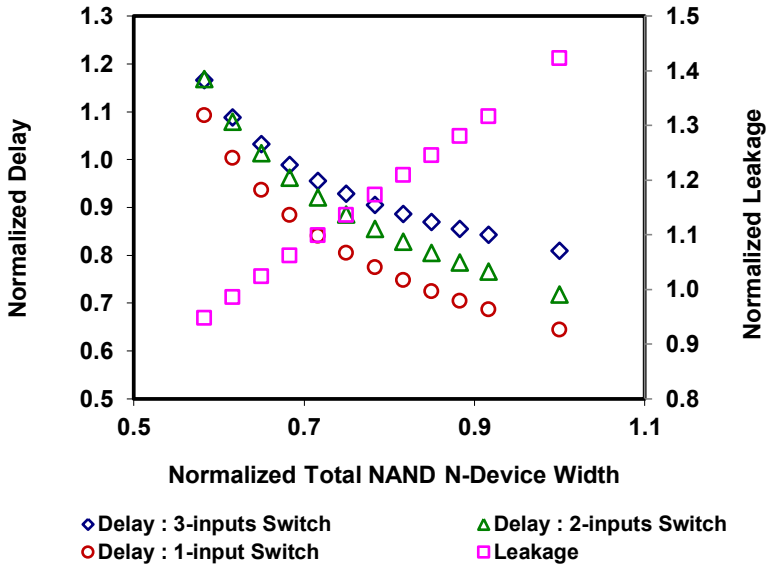


Figure 7.9 S-Decoder NAND3 3-to-8 decoder width, timing, and delay tradeoff relative to conventional decoder. Decoder size is decreased by reducing the size of the shared devices.

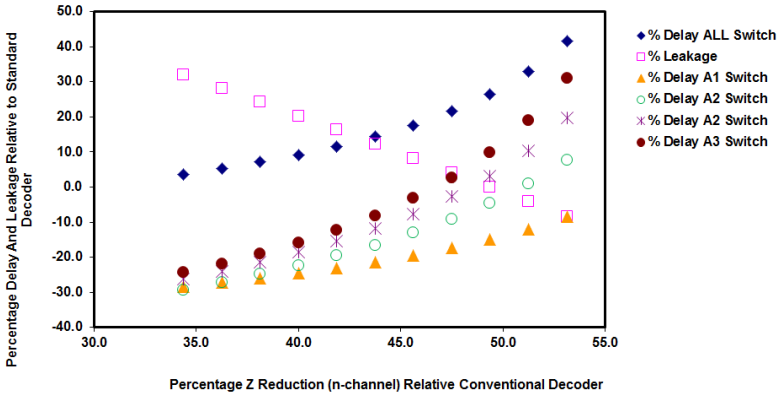


Figure 7.10 S-Decoder 4-to-16 NAND4 decoder area, leakage, and delay tradeoffs relative to conventional decoding

Figure 7.11 shows the delay, leakage and area (measure by n-device width) comparison of selected 3-to-8 shared decoder configurations. The graph compares configurations with same total width and same effective device width. With same total width, the width of the shared device is equal to the total width of the unshared equivalent. Thus a 2-way shared device will have a size of 2X the original size. The effective width sizing uses the same size for all devices, shared and unshared. The S421 configuration shows up to 40% delay improvement. The leakage increase is reduced by using low-leakage devices which can reduce leakage by >75% at 10-15% delay overhead.

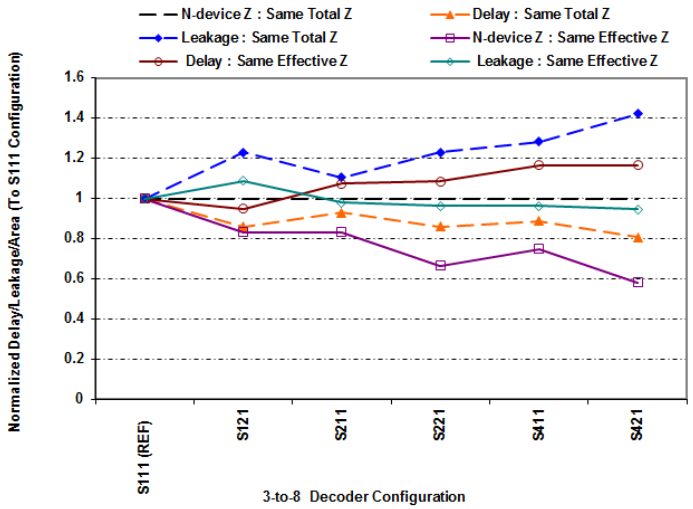


Figure 7.11 A 3-to-8 shared decoder timing, delay, and leakage comparison for various configurations

Figure 7.12 shows the effect of device sharing on leakage, delay and area of the different configurations of 4-to-16 shared decoder. Maximum area saving is realized with maximum sharing configuration S8421. Sharing lower level devices without sharing the corresponding high level devices do not yield optimal delay benefit. For example, sharing L2 devices yields less delay improvement if the L3 devices are not shared (S1411, S2411) as L3 becomes the bottleneck.

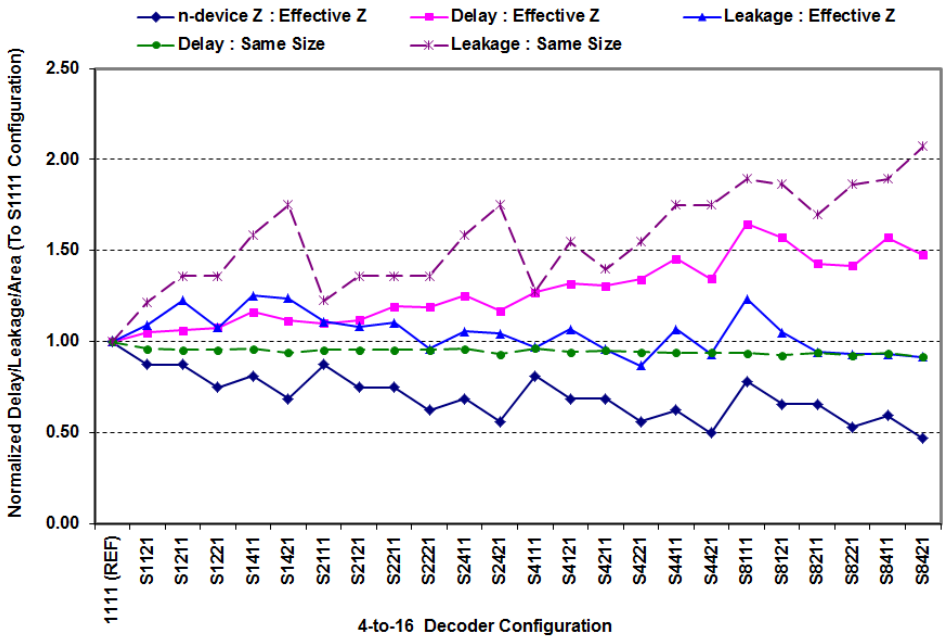


Figure 7.12 A 4-to-16 shared decoder timing, delay, and leakage comparison for various configurations

7.3.5 S-Decoder Standard Cells

Table 7.6 shows the comparison of fully routed layout area and extracted delay of shared decoder standard cells in a 32nm process compared to conventional decoder cells. In building the S-Decoder cell layout, it was necessary in some cell families, depending on the device sizes, to add a dummy device on a shared node to optimize layout footprint. For NAND standard cells with fixed height, the total layout width of the cell is usually determined by NMOS devices. This is because the series NMOS devices are sized larger than the parallel PMOS devices. Thus, the reduction in the number and size of the NMOS devices in shared decoder results in much more compact layout.

Up to 32% standard cell area reduction was achieved with S-Decoder scheme. The cells total device width were reduced by up to 17% (up to 28% for NMOS only) in addition to up to 12% delay improvement.

Table 7.6 Cell layout area and extracted delay comparison of a 3-to-8 S-Decoder relative to a corresponding conventional NAND decoder.

S-Decoder Standard Cell	Area Ratio	Delay Ratio	NMOS Device Width Ratio	NMOS+PMOS Device Width Ratio
Cell Size A	0.80	0.88	0.72	0.83
Cell Size B	0.75	0.94	0.73	0.85
Cell Size C	0.68	0.90	0.72	0.83
Cell Size D	0.84	0.99	0.71	0.83

7.4 SHARED DECODER POWER GATE

7.4.1 S-Decoder Power Gate Circuit

To implement logic power gating, the following basic constraints need to be addressed: (1) availability of a power gating control signal, (2) power gate device area overhead, and (3) delay cost of the power gated output. S-Decoder device sharing enhances the implementation of power gating by reducing the area and delay overhead. The decoder outputs are mutually exclusive therefore only a single power gate device (header or footer) is needed by each stage. A power gate implementation of a 3-to-8 shared NAND decoder is shown in Figure 7.13. An NMOS footer (PGN0) is inserted in series with the last shared device. A Read/Write enable signal is used to control the power gate device in an address decoding. The power gate device is turned OFF when the decoder is inactive in order to reduce leakage. In active mode, the footer is turned ON to establish a virtual VSS for normal operation. In a NAND-INVERT decoder, the output inverter can also be power gated by sharing a header. In a 3-to-8 decoder for example, all eight

inverter output PMOS devices can share the same power gate header, PGP1. This reduces the inverter power overhead. The optimal sizing of the shared PMOS power gate device (PGP1) is determined to be 4 times the reference inverter PMOS for a delay degration within 10% of reference inverter delay.

When power gate is turned ON, the gated logic output nodes may float depending on the input state. For a decoder, there is always one output that is at logic "1" regardless of the input address. Therefore only one decoder output can potentially float when gating is enabled. However, for static decoder, the output node that will float is not deterministic and is dependent on the last decoded address.

To eliminate the output floating (if not desired) at a reduced area cost, the decoder input is forced to a known address when the power gate is activated. This makes the floating decoder output node deterministic. The single floating output can then be forced to a known state during power gating to eliminate the float. This can be done on the NAND output using parallel PMOS device or on the INVETER output using parallel NMOS device. The input is forced to address "000", thus only output O_0 can float when the power gates are activated. Output O_1-O_7 will be driven by their respective non-gated NMOS devices to "0". Output O_0 is then forced to "0" as well using a complementary reset NMOS device. The cost of eliminating floating on the decoder output is reduced since only O_0 needs this reset device.

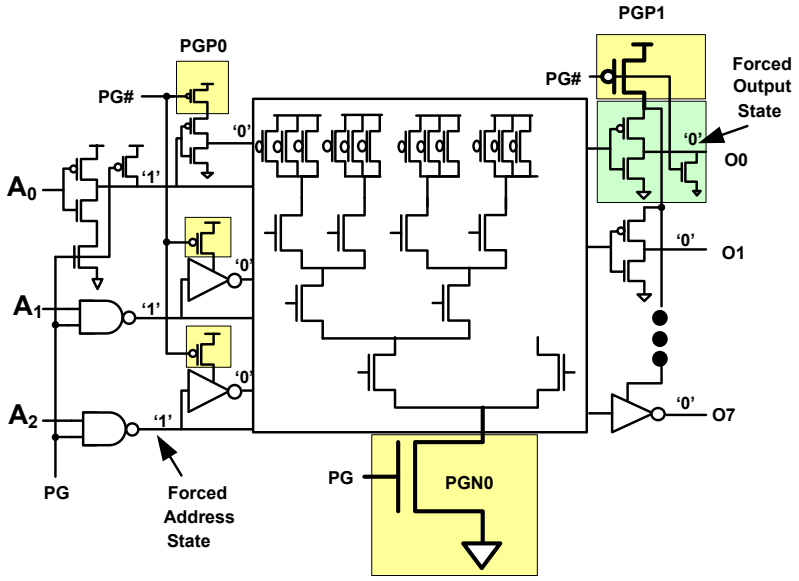


Figure 7.13 A 3-to-8 NAND-INVERT decoder with power gates on both NAND and inverter input gates and forced address to eliminate floating. Only O0 needs to be forced to "0".

7.4.2 Decoder Power Gate Device Sizing

The NAND and NOR power gate devices can be sized using a similar progressive ∝ scaling technique described in section III. The optimal size of the INVETER power gate device (PGI) is primarily a function of the reference inverter P-device (PI) size. The number of outputs sharing a power gate device also has secondary impact on the PGI size due to the intermediate stacked node loading. The power gate device size relative to the driver size is defined by the PG_{ratio}:

$$PG_{ratio} = \frac{Power\ Gate\ Device\ (PGI)\ Width}{Reference\ Device\ (PI)\ Width} \tag{7.11}$$

Figure 7.14 shows the power gating area, delay, and leakage tradeoffs of a 3-to-8 NAND-INVERT decoder inverter. The inverter leakage could be reduced by as much as 95% when the power gate device with a PG_{ratio} of 4 is shared by all inverters. A PG_{ratio} of 8 reduces the delay penalty to 5% of reference inverter delay at 92% leakage reduction if all 8 inverters share the same power gate device.

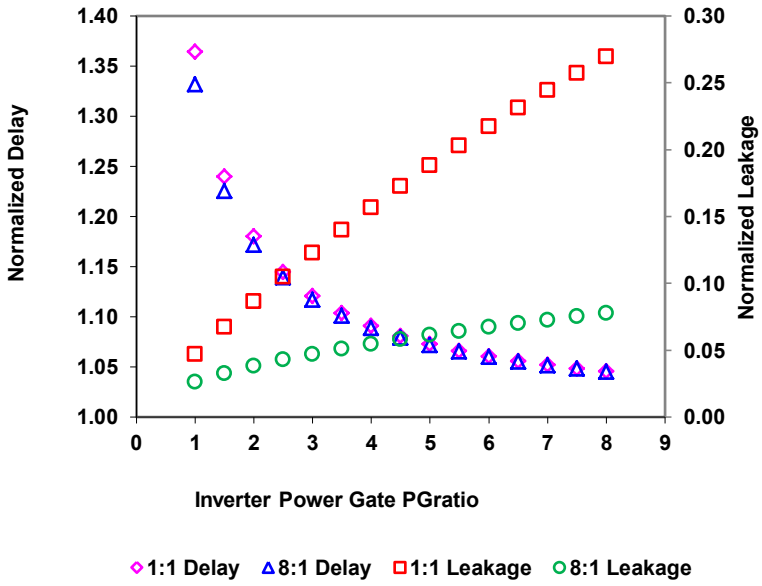


Figure 7.14 Delay and Leakage of power gated decoder inverter relative to non-gated inverter. 8:1 represents power gate device shared by all 8 inverters, 1:1 represents individual inverter power gate devices.

7.4.3 S-Decoder Power Gate Simulation Results

Figure 7.15 shows width, delay, and leakage tradeoffs of a power gated NAND S-Decoder relative to a conventional decoder. The power gate footer PGN0 is sized as another device in the shared series chain with the same ∝ ratio as discussed. A 90% leakage reduction and 10% width reduction is achieved with no delay penalty. Alternatively a delay improvement of up to 25% (for an independently switched input) and a leakage reduction of 89% can be achieved at no area cost relative to a conventional decoder.

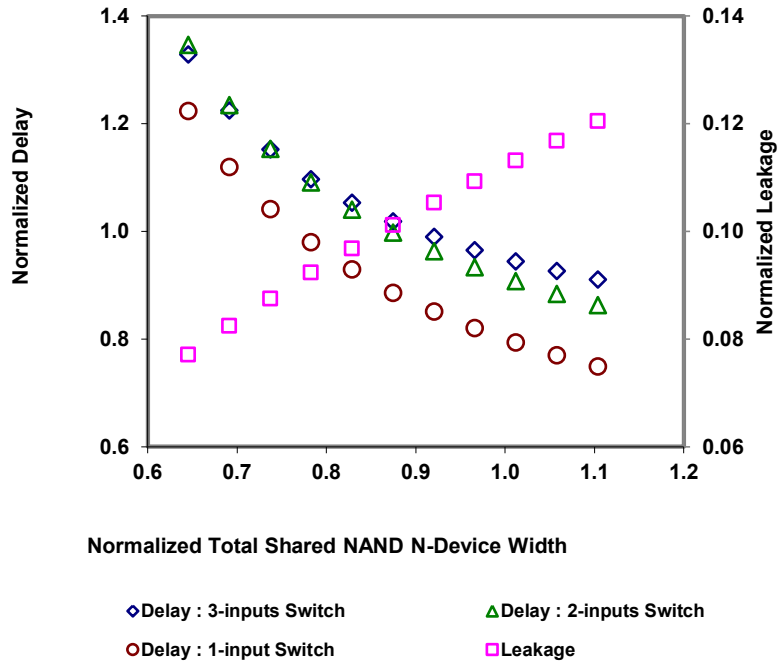


Figure 7.15 Delay, size, and power tradeoff of 3-to-8 shared decoder with power gate compared to a standard decoder.

7.5 CONCLUSION

An S-Decoder that exploits mutual exclusivity of address decoding for area and power efficiency is presented. A 3-to-8 decoder NAND NMOS device or NOR PMOS device sizes could be reduced by as much as 30% at no delay penalty. With power gating, a 90% NAND (NOR) leakage reduction is achieved. The S-Decoder sharing scheme can be used for any order of decoding. Higher order decoding show more gains as more devices are shared. For example, the area gain of 3-to-8 decoder is higher than 2-to-4 decoder since only one level of sharing is possible in a 2-to-4 decoder. Conversely a 4-to-16 decoder has more device sharing levels and hence higher area saving than a 3-to-8 decoder. A simple Elmore delay analysis is sufficient to provide sizing guidelines for the series connected chain of shared devices. A progressive sizing of the series-connected chain is required for optimal delay, power, and area tradeoff. The optimal progressive device size scaling ratio (α) required for the shared series connected chain to achieve the same delay as a conventional series connected chain is found to be 1.5 for a 2-series chain, 1.4 for a 3-series chain and 1.3 for a 4-series chain. Application of power gate and technique to eliminate floating is also presented. The proposed S-Decoder with power gating is a rare case where delay, area, and power can be simultaneous improved relative to conventional design. The extent of the improvement realized however depends on varying tradeoff, and application scenarios.

7.6 REFERENCE

[1] Jan Rabaey, "Low Power Design Essentials", *Springer, 2009*

[2] Giby Samson, Nagaraj Ananthapadmanabhan, Sayeed A. Badrudduza, and Lawrence T. Clark, "Low-Power Dynamic Memory Word Line Decoding for Static Random Access Memories". IEEE Journal of Solid-State Circuits, VOL 43, NO. 11, November 2008, Pg 2524-2532.

[3] M.A. Turi, J.G. Delgado-Frias, "Decreasing energy consumption in address decoders by means of selective precharge schemes" Microelectron. J (2009), doi:10.1016/j.mejo.2009.03.008.

[4] Bharadwaj S. Amrutur, and Mark A Horowitz, "Fast Low-Power Decoders for RAMs", IEEE Journal of Solid-State Circuits, Vol 36, No. 10, October 2001, Pg 1506-1515

[5] John P. Uyemura, "Cmos Logic Circuit Design", *Kluwer Academic Publishers, 2002*

[6] S. Narendra, S. Borkar, V. De, D. Antoniadis, A. Chandrakasan, "Scaling of Stack Effect and its Application for Leakage Reduction", *ISLPED, August 6-7,2001*

[7] Takayasu Sakurai, and A. Richard Newton, "Delay Analysis of Series-Connected MOSFET Circuits". IEEE Journal of Solid-State Circuits, VOL. 26, NO. 2, February 1991, Pg 122-131.

8 REGISTER FILE POWER, AREA, DELAY MODELING

***Abstract*— With the computing industry's paradigm shift from frequency to performance/watt, increasingly low power design solutions, especially for arrays, feature prominently in early architectural and design space exploration. Among arrays, register files (RFs) are the dominant power contributor, accounting for 30% of core power. Most register files are custom designed (>75%) in high performance CPUs. Existing array power models are however particularly unsuited for custom RFs. These models typically assume a generic array implementation and are not easily adaptable to unique topologies of custom RFs. Furthermore, they do not accurately address common design optimizations that significantly impact the power, area, and timing profile of an RF.**

This chapter presents a predictive power, area, and delay models that is customizable to different RF topologies and array structure. The model reduces the changes required to handle new circuit topologies, process corners, and design space exploration. We use a hybrid of an empirical reference data and analytical equations. Topology specific characteristics are captured empirically via a reference implementation. Analytical equations then model the impact of common architectural changes such as bit-width, depth, and ports and design optimizations such as segmentation, gating, device stacking and driver sizing relative to the reference design. The same analytical equations are used for all topologies, customized by the empirical reference and model parameters. On distinct topologies of multi-port read, single- and dual-ended writes, this hybrid reference based approach demonstrates an average error range of <10% for delay, leakage, dynamic, and area predictions. We show that for a specific topology, any reference configuration can be used for accurate prediction. We also show how the model is used to make real world architectural and design tradeoff.

***Index Terms*— Dynamic Power Model, Leakage Power Model, Register Files, SRAM.**

8.1 INTRODUCTION

Accurate power prediction in early-stage architectural and design exploration is ever more critical to the success of high performance/watt CPU/SoC design as new low power platforms like Ultrabooks, Tablets, and Smart Phones evolve. With continued reductions of project development cycles, critical project decisions (area, power, and frequency) need to be made early. This requires early architectural and

design tradeoff studies of which a significant proportion involves either changes to an existing arrays or addition of new arrays: Register Files (RF), SRAM, Read-Only Memory (ROM). This is driven by the fact that arrays collectively account for a significant percentage of die area and power, and are also frequently the performance bottleneck. For example on a 32nm Westmere (WSM) IA Core [1, 2], arrays accounted for 40% of total die power (Figure 8.1). Of this RF power is the most dominant, accounting for 30% of a typical CPU core power benchmark. Accurate array modeling is therefore pivotal to making critical decisions on die size, power, and performance.

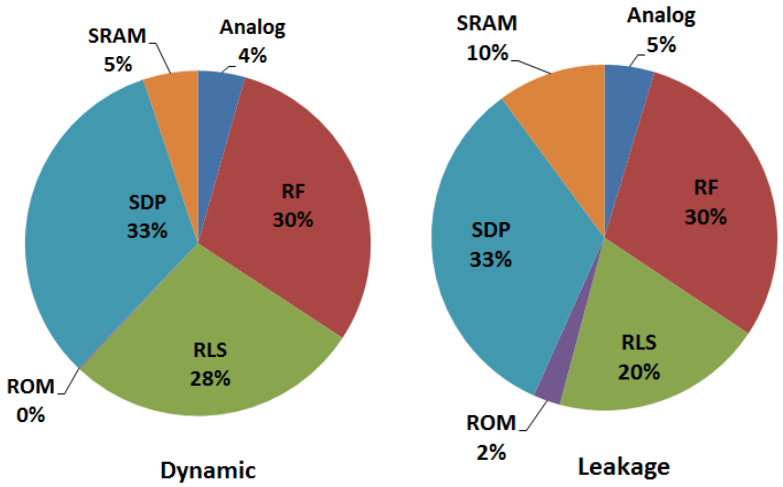


Figure 8.1 Dynamic and Leakage power distribution in 32nm WSM IA Core by design styles: Register Files (RF), Synthesized Block (RLS), Read Only Memory (ROM), Structural Data Path (SDP), SRAM

Register Files (RF) are the preferred memory element for fast data access and are therefore ubiquitous in modern microprocessor design. On a high performance CPU, greater than 75% of RFs are custom designed and not compiled due to design complexities and constraints (power, area, timing, low-voltage operation requirements). Each manual design introduces additional uniqueness to the RF (different floor plan, driver size, routing layer selection, clock distribution, segmentation etc.). Furthermore, techniques such as transistor stack forcing [3-7], power gating [8-10], dual-voltage [11-12], data/clock segment gating [3], read/write assists [13-15], multi-threshold transistor types among others are frequently explored for power, area, timing, and low voltage (V_{min}) optimizations. An RF's circuit implementation is therefore an integral part of its power, area, and timing profile beyond architectural bits, entry, and ports specifications. An accurate array model therefore need to be holistic, taking into consideration both architectural specifications and design implementation. Existing architectural models like Wattch [16] that do not take circuit implementation into consideration can have error as high as 94% [17]. Given the large design

space for RFs however, how do we accurately model all these distinct RF topologies and implementations without requiring new model equations for each distinct topology?

We present a customizable model that addresses the aforementioned model adaptability, reusability, and expands exploration options to include circuit-level design optimizations. Our model is a hybrid of empirical reference data and analytical equations. The empirical "reference design" data of the topology under study, which is an input parameter, captures topology-specific characteristics and process technology dependencies. We then analytically model the impact of cross-topology features such as changes in bit-width, entry-count, and common designer choices such as segmentation, gating, device stacking, and sizing. The same analytical equations are used for all topologies. We do not calculate the device level leakage and dynamic power values which are process technology and circuit topology dependent as in CACTI [18] and related models. Instead we rely on the reference implementation empirical data to capture those dependencies. We then model relative changes from the reference for a new configuration of that topology. Each RF stage is modeled separately using the same model equations. Stage-specific characteristics are therefore captured by model parameters, thereby reducing prediction error and making the model readily customizable to different topologies.

The distinct contributions of the proposed model are:

- A reusable model that can be customized for different array topologies by only modifying the model input parameters.

- A unified model equations for all stages (components) of the array i.e. wordline, bitline, decoder etc. all use the same analytical equations.

- Requires a single empirical reference data-point of any *MxN* (M-bits x N-entry) configuration of a topology to accurately model any other configuration of that topology.

- Incorporates accurate models for estimating the impact of practical array circuit implementations such as segmentation, data gating, stacking, and driver sizing.

- Incorporates activity factor and static probability propagation for benchmark modeling

We validated our model against in-house production timing and power estimation tools on a suite of fully extracted RFs of various topologies and configurations. We show that given any reference configuration of a topology, the model can accurately estimate power, area, and delay for any other configurations of that topology with <10% average error range.

The rest of the chapter is organized as follows. We review related models in section II and present an overview of our hybrid reference model flow in section III. Section IV discusses RF design and illustrates examples of common implementations that affect the overall power, area, and timing of RFs. In section V we discuss our unified stage model approach. The delay model is presented in Section VI, leakage and dynamic power models are presented in Section VII and VIII respectively. We present benchmark and area modeling in sections IX and X respectively. In section XI we show how the model can be customized for SRAM and ROM topologies. Model validation results and model applications scenarios for making real world design decisions are discussed in Section XII and XIV respectively. Conclusion and further discussion is presented in Section VI.

8.2 RELATED WORK

Existing parametric RF/SRAM estimation models, analytical or empirical, are based on specific topologies and circuit implementations. To model a different topology, today's architectural power models and performance simulators [16, 19] either use the existing power model essentially unchanged (inaccurate for the new topology) or develop new models for the different topology (time consuming). Moreover, these models do not provide the capability to explore the numerous circuit implementations available without requiring a new model.

Analytical models like PRACTICS[20], NVSim[21] McPAT[22], and others [23-25], using CACTI approach, derive analytical equations that model the key array capacitances and device dimensions based on a specific topology and process technology parameters. To adapt these models to different topologies/technologies require changes to the analytical formulas and parameters. Our proposed approach does not require any changes to the analytical formulas.

Empirical models rely on equations derived from data from entire suite of arrays. A major drawback of the regression based models [25, 26] is that they require the implementation of several RF configurations to curve fit the empirical data for each topology and technology. Applying statistical techniques, such as design of experiments, typically requires >10 data points to accurately fit the data. There could be as many as sixty (60) distinct custom RF implementations on a modern processor. A curve fitting modeling approach will require large number of samples of each unique topology which do not exist for custom RFs/Arrays. Furthermore, empirical models are only valid for the specific circuit topology and technology used to generate the model coefficients. Thus, they usually present a method rather than reusable model equations.

Liang and Brooks [27] presented a hybrid model that empirically captures the power of three array structures (1-bit x 1-entry, 2-bit x 1-entry, and 1-bit x 2-entry) and composes them analytically to obtain the power of an n-bit, m-entry structure. The analytical equations are stage dependent, with each stage requiring different equation. To reuse the model without modification requires empirical data from the three specific array configurations used to derive the model equations. These 1bx1e, 2bx1e, 1bx2e configurations however do not exist in real design. Moreover, as we show later from our results, using very small array configurations as references to predict larger configurations is less accurate due to circuit and layout anomalies that could be magnified in very small arrays.

IDAP [17] incorporates circuit implementation into its model by building a "Knowledge Base" of various circuit topologies. The knowledge base contains analytical equations for calculating capacitance of essential and influential nodes of each sub-block. New topologies however require the knowledge base to be update with new equations tailored for that topology.

Wattch [16], like most models, uses cell height to calculate stage parasitics using analytical equation and process parameters. Driver size is estimated by scaling driver width based on capacitive load without understanding of realistic driver sizing scenario where the driver is sized at specified intervals and not for every increase in driver load.

8.3 THIS WORK

8.3.1 Reference Design Model Approach

The proposed model is a hybrid of empirical reference data and analytical equations. Our hybrid approach uses an empirical data from a reference implementation, "reference design", to capture the uniqueness of each topology and analytical equations to model the cross-topology features. A "reference design" refers to a circuit implementation of one architectural configuration (bits/entry/ports) of a topology. A single reference empirical data (leakage, dynamic power, area, delay) is required for each distinct topology while the same analytical equations are used for all topologies.

We decompose the reference design logic into distinct stages and model each stage separately. Multiple logic stages may optionally be combined into a single model stage depending on their relevant contribution to power, needed accuracy, and dependency on other stages. By modeling each stage

separately, we can accurately capture the unique characteristics of each stage and can therefore easily customize the model to different topologies through parameterization and reference empirical data.

8.3.2 Model flow

Figure 8.2 shows the proposed model flow. A single reference architectural configuration (bits, entry, ports) of the topology under study is implemented to generate per stage power, area, and delay empirical data. The reference empirical data can be obtained from a full implementation (schematic, layout, parasitic extraction, timing and power estimation) or from a detailed simulation model. Using the reference per stage empirical data, new configurations are estimated by analytically modeling changes relative to the reference. We model the relative per stage change in power, area, and delay resulting from changes in architectural features such as bit-width, number of entries, number of ports, benchmark, and practical circuit implementations such as segmentation, gating, sizing, and device stacking. Each stage can independently be explored by modifying the parameters for that stage. New stages can be defined and added to the model by only capturing their empirical data and describing their topology using model parameters.

In effect our model is a set of unified topology-independent analytical equations that become customized for a topology by their reference empirical data and model parameters.

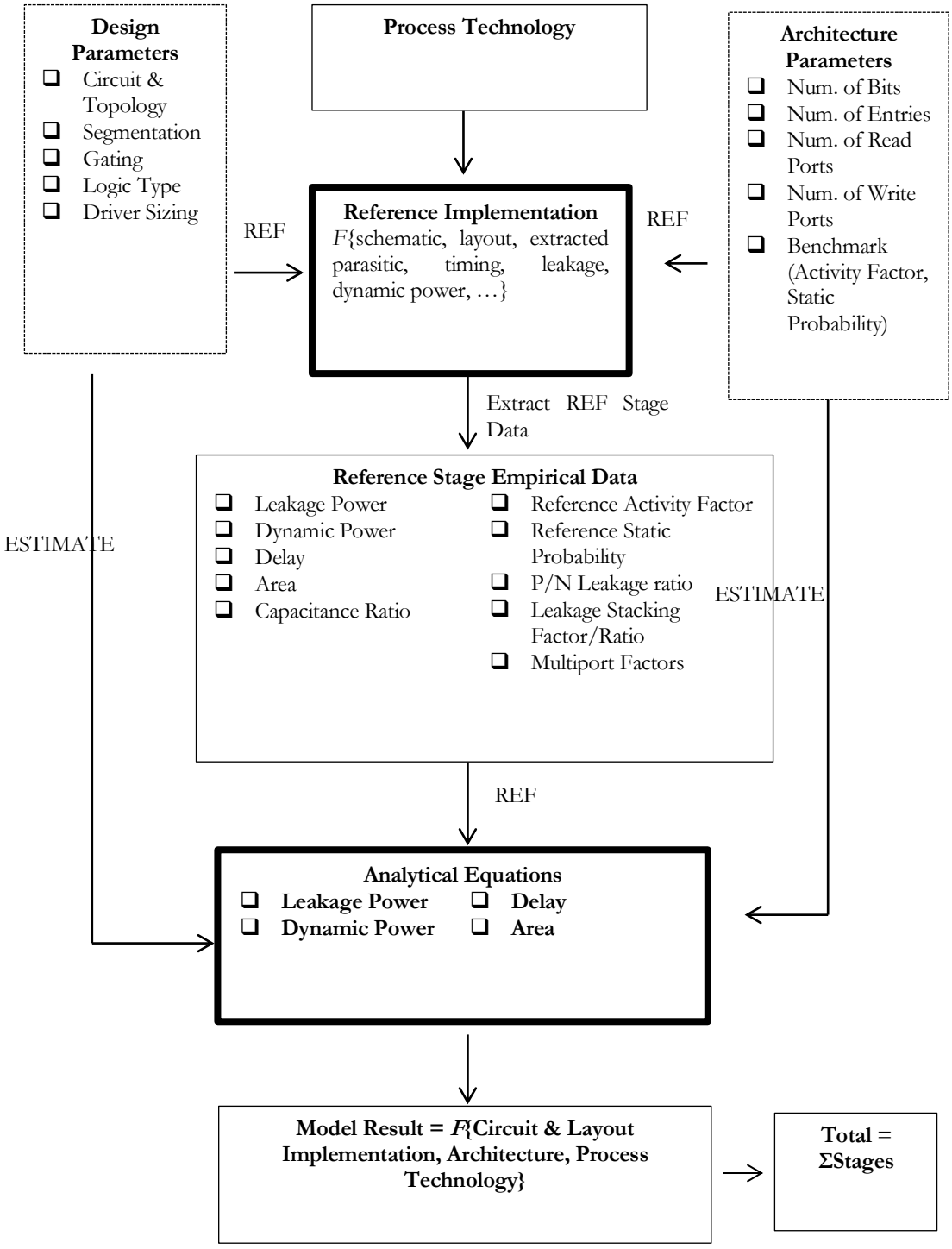


Figure 8.2 Proposed reference design based hybrid empirical and analytical model flow

8.4 REGISTER FILE DESIGN

The organization of a generic RF array is shown in Figure 8.3. The main components are the storage memory block organized as a structured M x N array of entries (rows) and bits (columns), IO drivers, decoder and pre-decoder blocks to generate wordline access signals, and array control logic block. While the high level array organization is fairly consistent across topologies, the detailed design implementations are not. We look at common factors that influence the power, area, and delay variations across RFs.

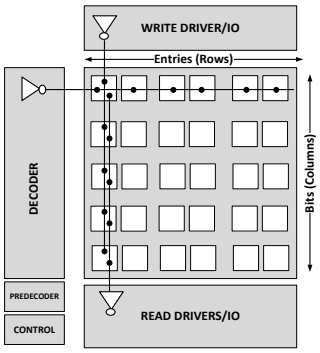


Figure 8.3 RF Array functional block organization.

8.4.1 Register File Topologies

Due to complexity and design constraints (performance, area, power, V_{min}) a wide varieties of RF topologies are implemented on a typical high performance processor. This shear number of variations makes RF modeling intractable using existing models. For example, the RF memory cell could be a Dual-Ended (DE) write where data is written from both sides of the memory (Figure 8.4a) or a Single-Ended (SE) write where data is written from one side of memory (Figure 8.4b). Each of these memories has a different power, area, delay, and V_{min} profile.

A conventional RF write data distribution is shown in Figure 8.5. The local write bitline driver (WD1) could be a NAND, NOR, INVERTERS (without segment enable), or TRI-STATE [3]. Read access may be implemented with static or dynamic full swing, or with small signal sensing logic. Figure 8.6 shows a conventional multi-stage full swing dynamic read implementation with precharged bitlines [5]. The read/write decoding and clocking schemes also have wide variations in their implementations [28-30]. The conventional read/write decoding schemes are shown in Figure 8.7.

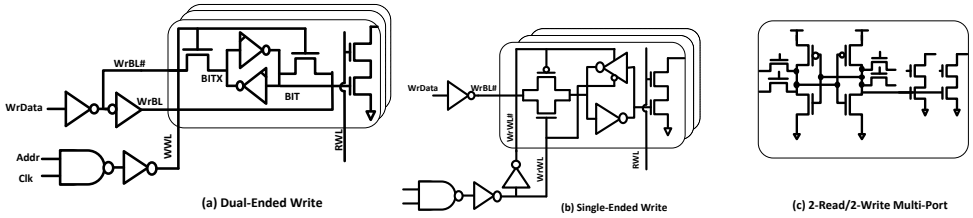


Figure 8.4 Register File memory topologies (a) 1-read, 1-write Dual-Ended (DE) write 8T bitcell (b) 1-read, 1-write Single-Ended interruptible (SE) write 10T bitcell (c) 2-read, 2-write multi-port 12T bitcell.

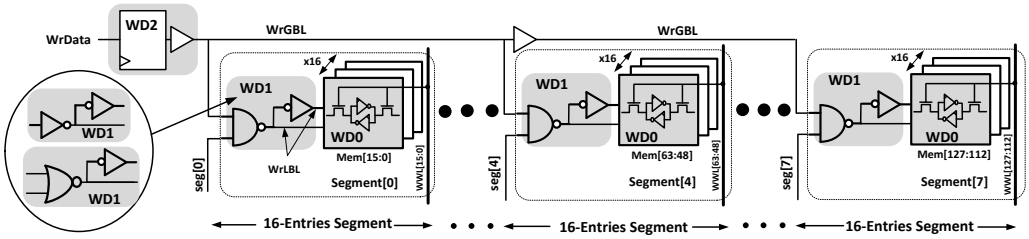


Figure 8.5 Register File write data distribution stages - gated bitline segments. A single segment is enabled during a write to reduce write local bitline toggling. Alternatively a NOR logic or an INVETER (no-gating) may be used. Distinct model stages are high-lighted. Gating is implemented by activating only one WrLBL segment per write.

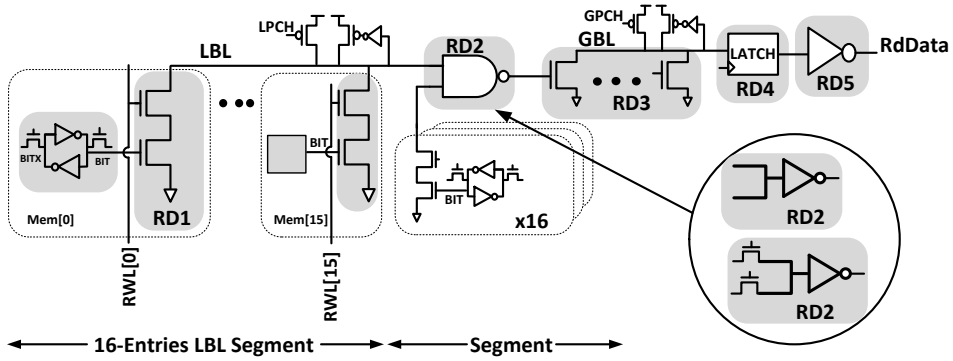


Figure 8.6 Register File dynamic read data path stages. The dynamic read bitlines are segmented into local and global bitline stages. Alternative segment implementations are shown in insert where an inverter or NMOS MUX can be used to merge multiple segments. Distinct model stages are high-lighted. Gating is implemented by activating only one LBL segment per read.

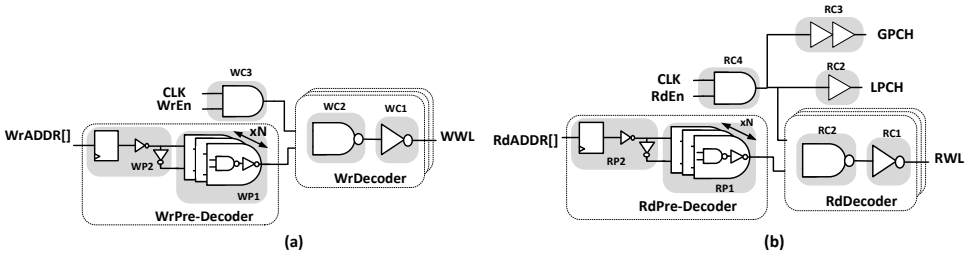


Figure 8.7 Conventional RF Write Decoder stages (b) Conventional RF Read Decode stages. Distinct model stages are high-lighted.

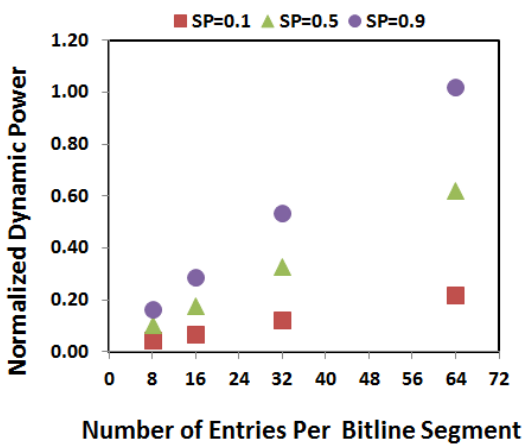


Figure 8.8 Effect of write data segmentation and data SP on dynamic Power

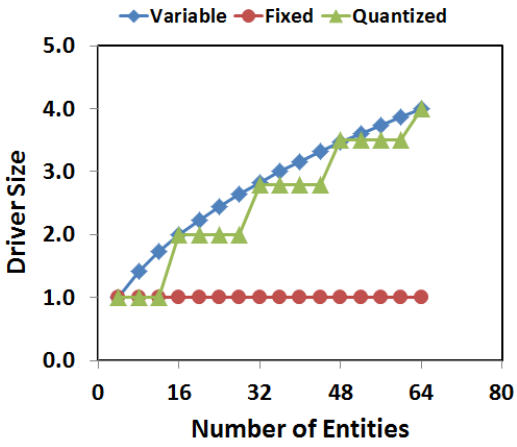


Figure 8.9 Driver sizing scenarios: Variable Sizing, Fixed Sizing, Quantized Sizing

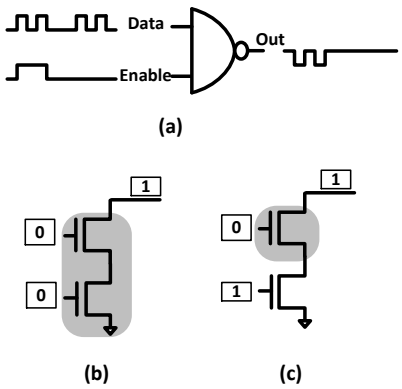


Figure 8.10 NAND (a) AF Propagation (b) Stacked (c)Un-stacked

8.4.2 Segmentation and Gating

In a conventional write data distribution, the write bitline is segmented with dedicated local write drivers for each segment [3]. The write data power, delay, and area are dependent on segment granularity. For example, the write bitline driver can be segmented into 16-entries per segment (Figure 8.5), requiring 4 write drivers (of size A) per bit to drive 64-entries. On the other hand if the segment is 32-entries per driver, only 2 write drivers (of size B < 2X Size A) per bit will be required to drive 64-entries. The driver sizes also depend on driver placement i.e. whether the segment is center or edge driven. The RF dynamic read local bitlines (LBL) are similarly segmented. Two LBL segments are merged by a NAND (RD2) which drives subsequent global bitline (GBL) segments. Alternatively the merging NAND gate, RD2, could be an inverter, or a mux-invert gate (Figure 8.6).

8.4.3 Data & Clock Gating

Data gating impact on power is illustrated by the write data distribution (Figure 8.5). When the local bitline segments are gated, only a single segment will be activated during a write regardless of the number of entries. This bounds the maximum dynamic power dissipation by the local bitline. For example, a gated 16-entry WrLBL segmentation will dissipate the same write bitline dynamic power in a 16-, 64-, or 256- entries RF array. Figure 8.8 shows the dynamic power dependency on Write bitline segment granularity. Increased segment granularity reduces the active load per segment, reducing dynamic power and stage delay. However this comes at an increased leakage and area cost. RF read bitline LBL is similarly segment gated. Clocks may also be segment gated to reduce dynamic power.

8.4.4 Device Stacking

Device stacking is enforced when series connected devices are both switched OFF (Figure 8.10). The leakage through stacked devices could be as much as to 90% lower than un-stacked gate [3]. Therefore, stacking is an important parameter in accurate modeling of state based device leakage. The residency of a logic gate in stack state depends on its input static probability (SP). For example, the leakage of an RF memory read port (RPT), Figure 8.6, is highly dependent on the state of the memory content. If the memory node connected to the RPT, "BIT", is storing a "0", stacking is enforced and RPT leakage is minimum. On the other hand if "BIT"="1", there is no stacking and RPT leakage is maximum. RPT leakage can therefore vary by as much as 10X depending on the stored memory data.

8.4.5 Driver Sizing

There are three basic driver sizing scenarios (Figure 8.9).

(i) Fixed Driver Sizing – In this scenario a fixed driver size is used to drive any number of entities. This typically leads to variable driver delay. In some cases the driver delay and hence size is independent of the number of entities. For example, the memory cell cross-coupled retention devices sizes are usually not dependent on the number of entries or bits.

(ii) Progressive Sizing (Fixed delay)–In this scenario a change in driver load (change in entities) is accompanied by a proportional change in driver size. The goal is to achieve a fixed delay across entities.

(iii) Quantized Sizing – With this common sizing scenario the driver size is not changed for every arbitrary change in driver load but only after a certain load or delay threshold is reached. This could also be necessitated by limitations imposed by fixed library cell sizes.

8.4.6 Multiple-Ports

Multi-Read/Write ports enable simultaneous access to different entries in the same cycle. With dual-ports most resources are duplication per port. However devices like the memory cross-coupled retention devices are shared (Figure 8.4c). The number of ports impacts power, area, and delay by the primary effect of resource duplication and secondary effect of array physical size growth.

8.4.7 Data Static Probability & Activity

The dynamic power of a domino RF read (Figure 8.6) is highly dependent on the polarity of the stored memory data (memory static probability). During read "1", the bitlines are discharged and precharged, while during read "0", the bitlines are not discharged (holds precharged state). Therefore the memory data SP determines the bitline activity and associated power.

8.5 UNIFIED MODEL

8.5.1 Model Stages and Parameters

The write and read model stages of a conventional RF topology (Figure 8.5-Figure 8.7) are shown in Table 8.1. Typically a stage instance is composed of a driver and the load it drives up to the receiver(s). However, multiple logic stages may be combined into a single model stage (e.g. WD2). We also define VirtualVSS, VirtualVDD, VirtualClock stages with specific AF and SP attributes. Table 8.2, Table 8.3, and Table 8.4 show the model architectural, implementation, and empirical parameters. The same model parameters are used to describe all stage.

Table 8.1 Register File Model Stages

Read Data Stages			**Write Data Stage**	
Read Local Bitline	RD1		Memory Storage	WD0
Read Merge NAND	RD2		Write Local Bitline	WD1
Read Global Bitline	RD3		Write Global Bitline	WD2
Read Data Latch	RD4		**Write Clock Stages**	
Read Data Driver	RD5		Write Wordline	WC1
Read Clock Stages			Write Wordline NAND	WC2
Read Wordline	RC1		Write Clock Buffer	WC3
Read Wordlne NAND	RC2		**Write Predecoder**	
Read Local Precharge	RC3		Write Predecoder	WP1
Read Global Precharge	RC4		Write Address Latch	WP2
Read Clock Buffer	RC5		**Virtual Stages**	
Read Predecoder			VirtualClock (SP=0.5, AF=0.5)	
Read Predecoder	RP1		VirtualVss (SP=0, AF=0)	
Read Address Latch	RP2		VirtualVdd (SP=1, AF=0)	

Table 8.2 Model Architecture Parameters

Parameter	Definition
N_b	Number of Bits
N_e	Number of Entries
N_{rpt}	Number of read ports
N_{wpt}	Number of write ports
AF	Stage Output Activity Factor
AF_d	Stage Input Data Activity Factor
AF_{en}	Stage Input Enable Activity Factor
SP	Stage Output Static Probability
SP_d	Stage Input Data Static Probability
SP_{en}	State Input Enable Static Probability

Table 8.3 Model Design Implementation Parameters (per stage)

Parameter	Definition
$N_{b\_perdriver}$	Maximum number of bits per driver.
$N_{e\_perdriver}$	Maximum number of entries per driver.
$N_{b\_persegment}$	Maximum number bits per segment
$N_{e\_persegment}$	Maximum number of entries per segment
$N_{b\_perdepsegment}$	Number of bits per dependent segment
$N_{e\_perdepsegment}$	Number of entries per dependent segment
$G_b(0,1)$	Bit segments gating (Gated=1, UnGated =0)
G_e $(0,1)$	Entries segments gating. (Gated=1, UnGated =0)
D_{Thresh}	Driver sizing delay threshold
V_{swing}	Discharge voltage swing
ρ (0,1)	Logic Gate Enable Active Polarity(NAND = "1", NOR = "0")
G_{logic}	Logic gate type: NAND, NOR, INVERTER, BUFFER
G_{en}	The defined stage that drives the "Enable" node of this stage
G_{data}	The stage defined that drives the "Data" node of this stage

Table 8.4 Model Empirical Parameters (per stage

Parameter	Definition
$D_{stagedelay\_ref}$	Reference Delay
$P_{stageleakage\_ref}$	Reference Leakage Power
$P_{stagedynamic\_ref}$	Reference Dynamic Power
$H_{stageheight\_ref}$	Reference Cell Height
$W_{stagewidth\_ref}$	Reference Cell Width
MPF_{rd}	Read Multi-Port Factor
MPE_{rd}	Read Multi-Port Effect
MPF_{wr}	Write Multi-Port Factor
MPE_{wr}	Write Multi-Port Effect
MPF_{rd_height}	Cell Width Read Multi-Port Factor
MPF_{wr_width}	Cell Width Write Multi-Port Factor
MPF_{rd_height}	Cell Height Read Multi-Port Factor
MPF_{wr_width}	Cell Height Write Multi-Port Factor
Φ_r	Repeated Capacitance Fraction
Φ_{nr}	Non-Repeated Capacitance Fraction
Φ_{ov}	Overhead Capacitance Fraction
$\propto$	Delay power law exponent
β	Driver size to stage delay slope
λ	Fraction of reference leakage that is fixed (leakage from auxiliary devices)

8.5.2 Unified Stage Model Equation

8.5.2.1 Entity Definition

We use the same unified model equations for all stages. References to the number of bits (columns) and entries (rows) are used interchangeably in the model equations based on whether the stage driver load is bits or entry dependent. The subscript "*x*" denotes the entity (bits or entry) whose increase (decrease) results in corresponding increase (decrease) in stage driver load. Subscript "*y*" denotes the orthogonal entity whose change does not affect the driver load. For example, the "*x*" and "*y*" entities of a wordline represents "bits" and "entries" respectively since a change in number of bits changes the load on the wordline driver. On the other hand, for the bitlines "*x*" and "*y*" entities represents "entries" and "bits" respectively since bitline driver load depends on the number of entries. The words "entity" ("entities") therefore refers to bit (bits) or entry (entries) depending on stage.

If(Load Entity="bits")
{
$N_x = N_b$
$N_{x\_ref} = N_{b\_ref}$
$N_{x\_perdriver} = N_{b\_perdriver}$
$N_{x\_persegment} = N_{b\_persegment}$
$N_{x\_perdepsegment} = N_{b\_perdepsegment}$
$G_x(0,1) = G_b(0,1)$

$N_y = N_e$
$N_{y\_ref} = N_{e\_ref}$
$N_{y\_perdriver} = N_{e\_perdriver}$
$N_{y\_persegment} = N_{e\_persegment}$
$N_{y\_perdepsegment} = N_{e\_perdepsegment}$
$G_y(0,1) = G_e(0,1)$
}

If(Load Entity="Entries")
{
$N_x = N_e$
$N_{x\_ref} = N_{e\_ref}$
$N_{x\_perdriver} = N_{e\_perdriver}$
$N_{x\_persegment} = N_{e\_persegment}$
$N_{x\_perdepsegment} = N_{e\_perdepsegment}$
$G_x(0,1) = G_e(0,1)$

$N_y = N_b$
$N_{y\_ref} = N_{b\_ref}$
$N_{y\_perdriver} = N_{b\_perdriver}$
$N_{y\_persegment} = N_{b\_persegment}$
$N_{y\_perdepsegment} = N_{b\_perdepsegment}$
$G_y(0,1) = G_b(0,1)$
}

For the rest of the chapter we use "ref" and "$\_ref$" to denote the reference design input parameters to distinguish it from the parameters of new configuration being estimated where necessary.

8.5.2.2 Segmentation Dependency and Load Entity Scaling

The segmentation of a stage may have dependency on another stage's segmentation. For example, the global bitline stage segmentation (secondary segment) has dependency on the local bitline segmentation (primary segment). The number of stage drivers or the load on the secondary stage is a function of primary stage segmentation. For example, a 128-entry array with 16-entries per LBL segment (32-entries per merge NAND), will require 4 GBL drivers whereas an 8-entry LBL segmentation (16-entries per merge NAND) will result in 8 GBL drivers.

To capture these stage segmentation dependencies in a single stage-independent unified model, two related segmentation parameters, $N_{x\_persegment}$ and $N_{x\_perdepsegment}$ are defined. $N_{x\_persegment}$ represents the number of entities per segment of a stage while $N_{x\_perdepsegment}$ is the number of entities per segment of the corresponding dependent stage. The number of load entities for a stage is therefore scaled by its dependent stage segment as:

$$N_{xl} = Ceil\left[\frac{N_x}{N_{x\_perdepsegment}}\right] \quad 8.1$$

$$N_{xl\_ref} = Ceil\left[\frac{N_{x\_ref}}{N_{x\_perdepsegment}}\right] \quad 8.2$$

$$N_{xlpersegment} = Ceil\left[\frac{N_{x\_persegment}}{N_{x\_perdepsegment}}\right] \quad 8.3$$

N_{xl}, N_{xl_ref} are the number of "*x*" entities scaled by dependent segment. $N_{xlpersegment}$ is the number of scaled "*x*" entities load per segment. If the stage has no dependency, $N_{x\_perdepsegment} = 1$. Similar scaling is defined for "*y*" entities.

8.5.2.3 Stage Capacitance Definition

We categorize the reference empirical stage capacitance into three components as shown in Figure 8.11.

Repeated Capacitance (Φ_r) – This is the fraction of the stage capacitance that is an instantiated multiple of the reference. Φ_r represents the capacitance contribution that changes with load-entity.

Non-repeated capacitance (Φ_{nr}) – This is the fraction of the stage capacitance that is not directly dependent on the number of load-entities but indirectly affected by driver resizing as a result of change in the number of load-entities.

Overhead Capacitance (Φ_{ov}) – This is the fraction of the stage capacitance that is not impacted by change in number of entities. This is usually a fixed load from routing overhead and fixed logic (e.g. bitline keeper logic) associated with the stage

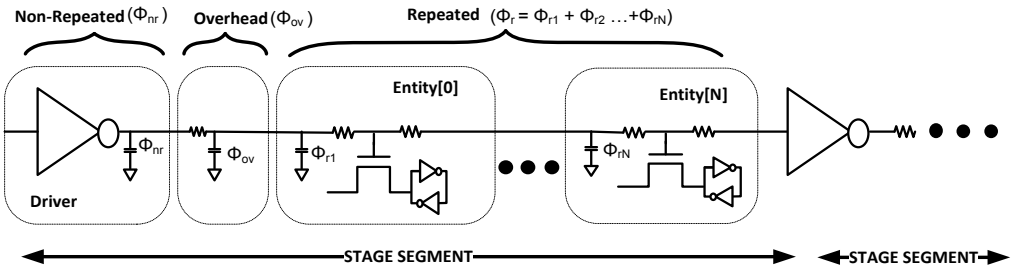


Figure 8.11 Stage capacitance components definition

8.5.2.4 Multiport Factors and Effects

A multi-ported array can be estimated with our model using a corresponding multi-port reference. However, frequently an architectural port exploration is needed on an existing array of different port count. We model ports by defining 1st and 2nd order port impact parameters. The 1st order multi-port factor, $MPort_{factor}$, defines direct ports impact on the number of stage instances. For example, in a conventional design, doubling the number of read ports doubles the number of read wordline drivers but does not affect the number of write wordline drivers.

$$MPort_{factor} = \left[\frac{(N_{rpt}-1)\times MPF_{rd}+1}{(N_{rpt\_ref}-1)\times MPF_{rd}+1}\right]^{rf} \times \left[\frac{(N_{wpt}-1)\times MPF_{wr}+1}{(N_{wpt\_ref}-1)\times MPF_{wr}+1}\right]^{wf} \quad 8.4$$

where : $rf = ceil(MPF_{rd});\ wf = ceil(MPF_{wr});\quad 0 \le MPort_{factor} \le 1$

The 2nd order $MPort_{effect}$ is an indirect port impact where the number of stage instance count is not changed but the stage load or driver size is impacted by the number of ports. For example, doubling the number of read ports does not directly affect the number of write drivers. However the memory cell physical size growth or multiple read port driver placements may affects the load on the write driver due to area growth. This is captured as the 2nd order $MPort$ effect. The MPE_{rd} and MPE_{wr} are the fractional change in reference load per change in number of read and write ports respectively.

$$MPort_{effect} = (N_{rpt} - N_{rpt\_ref}) \times\ MPE_{rd} + \ (N_{wpt} - N_{wpt\_ref}) \times MPE_{wr} \quad 8.5$$

Where $0 \le MPort_{effect} \le 1$

8.6 DELAY MODEL AND DRIVER SIZING

8.6.1 Driver Sizing

We model the different driver sizing scenarios (Figure 8.9) with a driver delay threshold, D_{Thresh}, parameter This is the number of entities interval at which the driver is resized to compensate for the driver delay change. When that specified interval is reached, the driver is resized to drive that number of

entities at iso-delay to the reference. A driver of that size is used to drive the next D_{Thresh} additional number of entities until the threshold is reached before it is resized again.. The number of entities the stage driver is sized to drive, N_{xload}, is bounded by the maximum number of load entities allowed per segment $N_{xlpersegment}$.

$$N_{xload} = Max\left\{\text{Min}\left(\begin{matrix} Floor\left[\frac{N_{xlpersegment}}{D_{Thresh}}\right] \times D_{Thresh}, \\ Floor\left[\frac{N_{xl}}{D_{Thresh}}\right] \times D_{Thresh} \end{matrix}\right), 1\right\} \quad 8.6$$

$$N_{xload_{ref}} = Max\left\{\text{Min}\left(\begin{matrix} Floor\left[\frac{N_{xlpersegment}}{D_{Thresh}}\right] \times D_{Thresh}, \\ Floor\left[\frac{N_{xl\_ref}}{D_{Thresh}}\right] \times D_{Thresh} \end{matrix}\right), 1\right\} \quad 8.7$$

N_{xload} , N_{xload_ref} are the $N_{xlpersegment}$ bounded number of entities the driver is sized to drive. The three driver sizing scenarios illustrated in Fig. 10 are achieved as follows:

- Variable Sizing (Fixed Delay) : $D_{Thresh} = 1$
- Fixed Sizing (Variable Delay) : $D_{Thresh} > N_{xlpersegment}$
- Quantized sizing : $1 < D_{Thresh} < N_{xlpersegment}$

8.6.2 Delay Model

Delay is modeled with power law. We model the fractional change in delay, *delay ratio*, relative to the reference as a function of driven entities. With the power law approach the delay model is to the first order independent of the specific reference configuration (Figure 8.12). The *delay ratio* (∂_x) is generally of the form:

$$\partial_x = \left(\frac{N_{xl}}{N_{xl\_ref}}\right)^{\propto} \quad 8.8$$

$$\log\ \partial_x = \propto \log\left(\frac{N_{xl}}{N_{xl\_ref}}\right) \quad 8.9$$

The $\propto$ value is obtained empirically from the slope of a $\log(\partial_x)$ vs. ($\log(N_{xl}/N_{xl_{ref}})$ plot, which can be obtained from simulation. Figure 8.12 illustrates how alpha value is derived for write wordline.

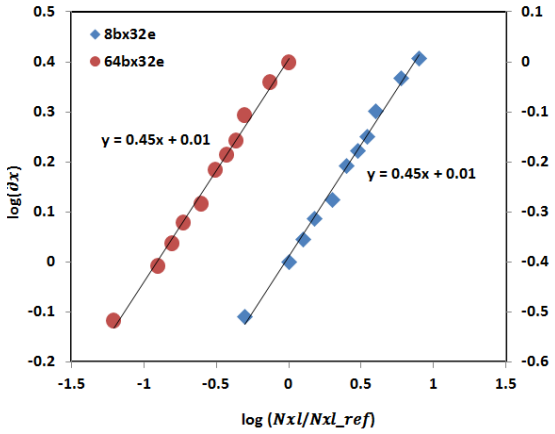


Figure 8.12 The write wordline delay exponent $\propto$ derived from the slope of log graph. The $\propto$ value (slope = 0.45) is independent of the reference configuration used

The maximum value of the *delay ratio* is bounded by the maximum number of entities per segment. A *delay effect* measures the entity delay impact within the bounds of $N_{xlpersegment}$ accounting for multi-port effect. From equation 8.8

$$\partial_{effect} = \left[\left(1 + MPort_{effect} \times \left(\Phi_{rpt} + \Phi_{ov}\right)\right) \times \frac{N_{xload}}{N_{xload\_ref}}\right]^{\propto} \quad 8.10$$

Where Φ_{rpt}, Φ_{ov} are the repeated and overhead cap ratios.

The stage delay is therefore given by:

$$D_{stagedelay} = D_{stagedelay\_ref} \times \partial_{effect} \quad 8.11$$

We use *delay effect* as a proxy for driver sizing. The stage driver size is then a function of the relative change in delay The relative driver size ($\partial_{effectsize}$) required to drive the given entities to achieve the same delay as the reference is:

$$\partial_{effectsize} = \beta \times \partial_{effect} \quad 8.12$$

β is the stage driver size to delay slope. A β=1 implies an $x\%$ change in delay requires an equivalent $x\%$ change in driver size to achieve the same delay as the reference.

8.7 LEAKAGE MODEL

8.7.1 Stage Leakage Power

The stage leakage is modeled by the stage driver instance count relative to the reference ($P_{stageleakage\_ref}$), accounting for relative driver sizing ($\partial_{effectsize}$), multiport impact ($MPort_{factor}$), and stacking factor (SF) and static probability.

$$P_{StageLeakage} = P_{stageleakage\_ref} \times SF \times MPort_{factor} \times \left[\lambda + (1-\lambda) \times \partial_{effectsize} \times \left(\frac{N_{xdriver}}{N_{xdriver\_ref}} \right) \times \left(\frac{N_{ydriver}}{N_{ydriver\_ref}} \right) \right] \times K_l \quad 8.13$$

where

$$N_{xdriver} = Ceil\left[\frac{N_x}{N_{x\_perdriver}} \right] \quad 8.14$$

$$N_{xdriver\_ref} = Ceil\left[\frac{N_{x\_ref}}{N_{x\_perdriver}} \right] \quad 8.15$$

$$N_{ydriver} = Ceil\left[\frac{N_y}{N_{y\_perdriver}} \right] \quad 8.16$$

$$N_{ydriver\_ref} = Ceil\left[\frac{N_{y\_ref}}{N_{y\_perdriver}} \right] \quad 8.17$$

$N_{xdriver}$, $N_{xdriver\_ref}$ are total number of "*x*" entity drivers and $N_{ydriver}$, $N_{ydriver\_ref}$ are number of "*y*" entity drivers. SF is the stage driver stack factor, K_l is an optional stage leakage scale factor.

The total leakage:

$$P_{Leakage} = \sum \{P_{StageLeakage}\} \quad 8.18$$

8.7.2 Static Probability and Stacking Factor (SF)

The static probability (SP) represents the probability that a signal is in logic "1" state. Since a PMOS and NMOS device typically have different leakage. This may be due to unequal driver PMOS-NMOS size and device type, the leakage of a driver is dependent on which device is leaking and hence on the stage SPs.

$$P_{Leakage} = SP_n \times L_n + SP_p \times L_p \quad 8.19$$

SP_n, L_n are the probability of leakage through the NMOS-pulldown (output="1") and the corresponding leakage respectively. SP_p, L_p are the probability of leakage through the PMOS-pullup (output="0") and the corresponding leakage respectively.

Considering stacking, the stage leakage:

$$\begin{aligned} P_{Leakage} &= (P_{sn} \times L_{sn} + P_{un} \times L_{un}) \times SP_n + \left(P_{sp} \times L_{sp} + P_{up} \times L_{up}\right) \times SP_p \\ &= (P_{sn} \times L_{SRN} + P_{un}) \times L_{un} \times SP_n + \left(P_{sp} \times L_{SRP} + P_{up}\right) \times L_{up} \times SP_p \end{aligned} \quad 8.20$$

where $L_{SRN} = {}^{L_{sn}}/_{L_{un}} \quad L_{SRP} = {}^{L_{sp}}/_{L_{up}}$

P_{sn}, P_{un} are the probabilities that the NMOS-pulldown is in a in stacked or un-stacked states respectively. P_{sp}, P_{up} are the probabilities that PMOS-pullup is in a stacked or un-stacked states respectively. L_{SRN} is the NMOS-pulldown stacked to un-stacked leakage ratio. L_{SRP} is the PMOS-pullup stacked to un-stacked leakage ratio. If the stage has no stacking devices, e.g. an INVERTER, $L_{SRN} = L_{SRP} = 1$.

The probability of the stage leaking through NMOS-pulldown, SP_n, is given by the stage output static probability.

$$SP_n = SP_o \quad 8.21$$

$$SP_p = 1 - SP_o \quad 8.22$$

The stage stacking residency is modulated by the input SP. In leakage mode (i.e. NMOS or PMOS leaking), one of the stacked devices will be OFF. We assume that the "Enable" is OFF and the "Data" SP modulates stacked and un-stacked residencies. For example, stacking is enforced in a leaking NAND gate (with Enable="0") when Data="0" and no stacking when Data="1" (Figure 8.10). Conversely for a

NOR gate in leakage mode (Enable="1"), stacking is enforced when "Data=1", whereas "Data=0" renders the NOR gate un-stacked. The NAND/NOR stacking probabilities are complementary.

$$P_{un} = P_{sp} = SP_d \tag{8.23}$$

$$P_{sn} = P_{up} = 1 - SP_d \tag{8.24}$$

From the above equations (Equations 8.20-8.24),

$$P_{Leakage} = [(1 - SP_d) \times L_{SRN} + SP_d] \times L_{un} \times SP_o + [SP_d \times L_{SRP} + (1 - SP_d)] \times L_{up} \times (1 - SP_o) \tag{8.25}$$

The impact of stacking on leakage relative to the reference is obtained from the stacking factor, SF.

$$SF = \frac{[(1 - SP_d) \times L_{SRN} + SP_d] \times SP_o + [SP_d \times L_{SRP} + (1 - SP_d)] \times L_R \times (1 - SP_o)}{\left[\left(1 - SP_{d\_ref}\right) \times L_{SRN} + SP_{d\_ref}\right] \times SP_{o\_ref} + \left[SP_{d\_ref} \times L_{SRP} + \left(1 - SP_{d\_ref}\right)\right] \times L_R \times (1 - SP_{o\_ref})} \tag{8.26}$$

where $L_R = {L_{up}}/{L_{un}}$

L_R is the stage reference PMOS-pullup/NMOS-pulldown unstacked leakage ratio. L_R is a function of PMOS-NMOS device sizing, device type, circuit implementation and others.

8.8 DYNAMIC POWER MODEL

Dynamic power is a function capacitance (C), the supply voltage (V_{dd}), the swing voltage (V_{swing}), activity (AF), and frequency (f). Each of these components can be scaled relative to the empirical reference.

$$P_{dynamic} = C \times AF \times V_{dd} \times V_{swing} \times f \tag{8.27}$$

We model the stage dynamic power by aggregating component segments dynamic power, accounting for gating, activity, voltage swing, multiport, and sizing effects..

8.8.1 Segment Dynamic Power

The dynamic power of a single stage segment:

$$P_{segmentdynamic} = AF_R \times V_R \times f_R \times MPort_{factor} \times P_{dynamicpersegment\_ref} \times N_{xstagesegment} \times N_{ystagesegment} \tag{8.28}$$

Where:

$$V_R = \frac{V_{dd} \times V_{swing}}{V_{dd\_ref} \times V_{swing\_ref}} ; AF_R = \frac{AF}{AF_{ref}} ; f_R = \frac{f}{f_{ref}}$$

$$N_{xstagesegment} = \Phi_r \times N_{rpt} \times \partial_{effectsize_{rpt}} \times (1 + MPort_{effect}) \times k_r + \Phi_{nr} \times \partial_{effectsize_{nr}} \times k_{nr} + \Phi_{ov} \times (1 + MPort_{effect}) \times k_{ov} \tag{8.29}$$

$$N_{ystagesegment} = \left(\frac{N_{ysegment}}{N_{ysegment\_ref}}\right)^{(1-G_y)} \tag{8.30}$$

$$N_{ysegment} = Ceil\left[\frac{N_y}{N_{y\_persegment}}\right] \tag{8.31}$$

$$N_{ysegment\_ref} = Ceil\left[\frac{N_{y\_ref}}{N_{y\_persegment}}\right] \tag{8.32}$$

$$P_{dynamicpersegment\_ref} = \frac{P_{stagedynamic\_ref}}{\left(Ceil\left[N_{xsegment\_ref}\right] \times (\Phi_{ov} + \Phi_{nr}) + N_{xsegment\_ref} \times \Phi_r\right)^{(1-G_x)}}$$

8.33

$$N_{xsegment\_ref} = \frac{N_{xl\_ref}}{min\left(N_{xl\_ref},\ N_{xlpersegment}\right)}$$

8.34

N_{rpt} is the relative number of repeated load-entities (normalized to the reference). $N_{ysegment}$, $N_{ysegment\_ref}$ are the number of "*y*" entity segments. $\partial_{effectsize\_rpt}$, $\partial_{effectsize\_nr}$ are the sizing impact of repeated and non-repeated cap respectively due to delay effects. G_x, G_y are the "*x*" and "*y*" entity segments gating. $G = 1$ if segment is Gated and $G = 0$ if not Gated. k_r, k_{nr}, k_{ov} are optional component capacitance scale factors.

8.8.2 Total Stage Dynamic Power

Two stage segment dynamic power components are defined. $P_{stagedyn\_segment}$ represents dynamic power of a segment with maximum allowable number of entities, $N_{xlpersegment}$. $P_{stagedyn\_partial}$ is the dynamic power of a partially filled segment with number of entities $< N_{xlpersegment}$. If a stage's segments are un-gated, all the segment components are activated. If the stage is segment gated, only one segment with the maximum number of entities per segment, $N_{xlpersegment}$, is assumed to be active.

$$N_{xmaxsegment} = floor\left[\frac{N_{xl}}{N_{xlpersegment}}\right]$$

8.35

$$N_{xpartial} = mod\left[\frac{N_{xl}}{N_{xlpersegment}}\right]$$

8.36

$N_{xmaxsegment}$ is the number of instances of complete segments. $N_{xpartial}$ is the number of entities in the partial segment.

From equation 8.28, the *Max Segment Power*:

$$P_{stagedyn\_segment} = P_{segmentdynamic} \mid N_{rpt} = \frac{N_{xlpersegment}}{Min\left(N_{xl\_ref}, N_{xlpersegment}\right)}$$

8.37

From equation 8.28 the *Partial Segment Power*:

$$P_{stagedyn\_partial} = \quad P_{segmentdynamic} \mid N_{rpt} = \frac{N_{xpartial}}{Min\left(N_{xl\_ref}, N_{xlpersegment}\right)} \tag{8.38}$$

The total stage dynamic power:

$$P_{StageDynamic} = Z_{seg} \times N_{seggate} \times P_{stagedyn\_segment} + Z_{part} \times N_{partgate} \times P_{stagedyn\_partial} \tag{8.39}$$

where

$$N_{seggate} = \left(Max\{N_{xmaxsegment}, 1\}\right)^{(1-G_x)} \tag{8.40}$$

$$Z_{seg} = Min\left(N_{xmaxsegment}, 1\right) \tag{8.41}$$

$$N_{partgate} = 1 - Min\left(N_{xmaxsegment}, G_x\right) \tag{8.42}$$

$$Z_{part} = Min\left(N_{xpartial}, 1\right) \tag{8.43}$$

$N_{seggate}$ is the number of gated segments component. $Z_{part}(0,1)$ zeroes out the "*Partial*" segment component if the stage is gated and one or more segments exist. $Z_{seg}(0,1)$ zeroes out the "*Max*" segment component if total number of entities is less than a segment.

The total dynamic power:

$$P_{Dynamic} = \sum\{P_{StageDynamic}\} \tag{8.44}$$

8.9 BENCHMARK MODELING

We model benchmarks by their AF and SP changes relative to the reference benchmark. AF/SP can be explicitly specified for each stage to model new benchmarks. However, the most common architectural exploration involves specifying AF/SP for key nodes such as read/write enable activity, memory state, and write data activity. We therefore also model AF/SP propagation for new benchmarks estimation.

We define "*Enable"* and "*Data"* stage connectivity parameters (Figure 8.10a). For example, the read local bitline stage, RD1, "*Enable*" and "*Data*" stages are read wordline, RC1, and memory storage, WD0, stages respectively. To use the same unified equations for all stages, we also define three virtual stages, "*VirtualVdd"* (SP=1, AF=0), "*VirtualVss"* (SP=0, AF=0), and "*VirtualClock"* (SP=0.5, AF=0.5), for connectivity to stages without explicitly modeled "*Data"* or "*Enable"* stages. For example the read wordline stage, RC1, is treated as a NAND with "*Enable*" input connected to "*VirtualVdd*" stage and "*Data*" input connected to the NAND stage, RC2. The read clock stage, RC4, has its "Data" connected to "*VirtualClock*" and the "Enable" connected to "RdEn".

The benchmark propagation is computed in two steps to ensure a common mode analysis. We first propagate the reference AF/SP starting from the stages for which new AF/SP have been specified, overriding the existing reference AF/SP of connected stages. We then propagate the new benchmark AF/SP values. For example, let's assume the reference dynamic power is based on "RdEn" AF=0.2 and we need to estimate dynamic power for "RdEn" AF=0.3. We will propagate both the reference "RdEn" AF=0.2 and the new "RdEn" AF=0.3 values to all the stages starting from RC4 stage. We then use the propagated AF/SP from both the reference and new benchmarks in the dynamic and leakage power equations to estimate the relative change in power between the old and new benchmarks.

The propagated stage output SP, SP_{po}, is modeled as:

$$SP_{po} = \rho \times \left(1 - SP_{pen} \times SP_{pd}\right) + (1 - \rho) \times \left[\left(1 - SP_{pen}\right) \times \left(1 - SP_{pd}\right)\right] \quad 8.45$$

SP_{pd} , SP_{pen} are the propagated SP of the stage's input "*Data*" and "*Enable*" respectively. The new benchmark leakage stacking factor (SF) is computed using the propagated SPs in equation 8.26

$$SF = \frac{\begin{array}{c}\left[\left(1 - SP_{pd\_new}\right) \times L_{SRN} + SP_{pd\_new}\right] \times SP_{po\_new} \\ + \left[SP_{pd\_new} \times L_{SRP} + \left(1 - SP_{pd\_new}\right)\right] \times L_R \times (1 - SP_{po\_new})\end{array}}{\begin{array}{c}\left[\left(1 - SP_{pd\_ref}\right) \times L_{SRN} + SP_{pd\_ref}\right] \times SP_{po\_ref} \\ + \left[SP_{pd\_ref} \times L_{SRP} + \left(1 - SP_{pd\_ref}\right)\right] \times L_R \times (1 - SP_{po\_ref})\end{array}} \quad 8.46$$

The propagated stage AF is modeled as:

$$\begin{array}{c} AF_p = AF_{pd} \times \left[\rho \times SP_{pen} + (1 - \rho) \times \left(1 - SP_{pen}\right)\right] + AF_{pen} \\ \times \left[\rho \times SP_{pd} + (1 - \rho) \times \left(1 - SP_{pd}\right)\right] \end{array} \quad 8.47$$

Where ρ is the stage logic gate enable active polarity (NAND = "1", NOR = "0"), AF_{pd} ,and AF_{pen} are the propagated stage "*Data*" and "*Enable*" AF respectively. The normalized AF ratio, AF_R , is computed from the propagated AF values.

$$AF_R = \frac{AF_{p\_new}}{AF_{p\_ref}} \tag{8.48}$$

AF_{p_new}, AF_{p_ref} are the new and reference benchmarks propagated AF respectively.

8.10 AREA MODEL

Array area is modeled by the aggregate of cell instance count relative to the reference. The total cell dimensions are determined from cell entity abutments. The following parameters are defined (Figure 8.13).

$labut\{1,0\}$ – This is the stage driver cell instances abutment relative to the load-entity. $labut = 1$ if the stage load-entity driver cell instances are horizontally abutted and $labut = 0$ if vertically abutted.

$oabut\{1,0\}$ – This is the stage driver cell instances abutment relative to the orthogonal-entity. $oabut = 1$ if the stage orthogonal-entity driver cell instances are horizontally abutted and $oabut = 0$ if vertically abutted.

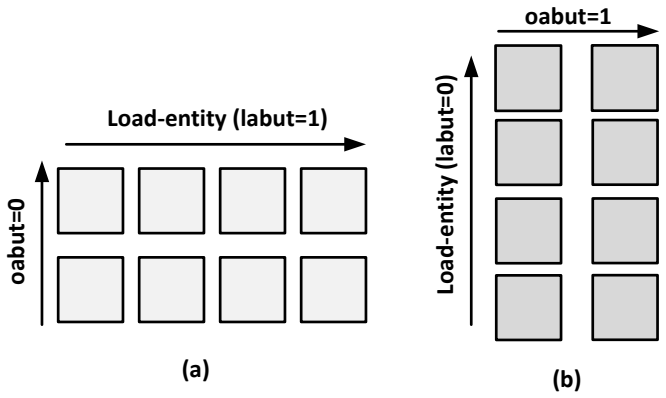


Figure 8.13 Stage cell abutment definition relative to entities (load-entity=4, orthogonal-entity=2) (a)labut=1 (b) labut=0

The stage cell width:

$$W_{stagewidth} = K_w \times MPortW_{factor} \times \left(\partial_{effectsize}\right)^{gw} \times \\ \times \left[\left(\frac{N_{xdriver}}{N_{xdriver\_ref}}\right) \times W_{stagewidth\_ref}\right]^{labut} \\ \times \left[\left(\frac{N_{ydriver}}{N_{ydriver\_ref}}\right) \times W_{stagewidth\_ref}\right]^{oabut} \quad 8.49$$

The stage cell height:

$$H_{stageheight} = K_h \times MPortH_{factor} \times \left(\partial_{effectsize}\right)^{(1-gw)} \\ \times \left[\left(\frac{N_{xdriver}}{N_{xdriver\_ref}}\right) \times H_{stageheight\_ref}\right]^{(1-labut)} \\ \times \left[\left(\frac{N_{ydriver}}{N_{ydriver\_ref}}\right) \times H_{stageheight\_ref}\right]^{(1-oabut)} \quad 8.50$$

gw is the cell delay effect sizing dimension parameter. If $gw = 1$ any cell growth from resizing due to delay effect is only applied to the stage cell width dimension while $gw = 0$ applies delay effect growth to cell height only. When $gw = 0.5$ delay effect size change is applied to both width and height (keeping existing aspect stage cell ratio). K_w, K_h are optional cell width and height scale factors. $MPortW_{factor}$, $MPortH_{factor}$ are the relative change in cell dimension due to change in port count :

$$MPortW_{factor} = \left[\frac{\left(N_{rpt} - 1\right) \times MPF_{rd\_width} + 1}{\left(N_{rpt_{ref}} - 1\right) \times MPF_{rd\_width} + 1}\right]^{rfw} \\ \times \left[\frac{\left(N_{wpt} - 1\right) \times MPF_{wr\_width} + 1}{\left(N_{wpt\_ref} - 1\right) \times MPF_{wr\_width} + 1}\right]^{wfw} \quad 8.51$$

$$MPortH_{factor} = \left[\frac{\left(N_{rpt} - 1\right) \times MPF_{rd\_height} + 1}{\left(N_{rpt_{ref}} - 1\right) \times MPF_{rd\_height} + 1}\right]^{rfh} \\ \times \left[\frac{\left(N_{wpt} - 1\right) \times MPF_{wr\_height} + 1}{\left(N_{wpt\_ref} - 1\right) \times MPF_{wr\_height} + 1}\right]^{wfh} \quad 8.52$$

where

$$rfw = ceil(MPF_{rd\_width}); \quad wfw = ceil(MPF_{wr\_width})$$
$$rfh = ceil(MPF_{rd\_height}); \quad wfh = ceil(MPF_{wr\_height})$$

$$0 \leq rfw,\ wfw,\ rfh,\ wfh \leq 1$$

The total array width and height dimensions:

$$Width_{total} = \sum\{W_{stagewidth}\} \tag{8.53}$$

$$Height_{total} = \sum\{H_{stageheight}\} \tag{8.54}$$

$$Area_{total} = Width_{total} \times Height_{total} \tag{8.55}$$

We use dimension criticality parameters to determine whether a stage cell dimension is critical to the total array width and height.

$$Width_{totalcritical} = \sum\{W_{stagewidth} \times W_{critical}\} \tag{8.56}$$

$$Height_{totalcritical} = \sum\{H_{stageheight} \times H_{critical}\} \tag{8.57}$$

$$Area_{totalcritical} = Width_{totalcritical} \times Height_{totalcritical} \tag{8.58}$$

$W_{critical}(0,1), H_{critical}(0,1)$ are the stage cell dimension criticality factors (Critical factor is set to "1" if a stage dimension is a critical array dimension, else "0") . The array area cell utilization is a ratio of total cell area to total critical dimension area.

$$Area_{density} = \frac{Area_{total}}{Area_{totalcritical}} \tag{8.59}$$

8.11 SRAM MODELING

The same model equations developed above are customizable for estimating SRAM power, area, and delay. The key differences between the conventional 6T-SRAM and the 8T RF described above are small signal sensing and shared read/write port. For a 6-T SRAM modeling, the read and write data path logic are decomposed into stages similar to RFs (Figure 8.14a). The write bitline (WD1) and memory (WD0) stages are similar to that of the RF discussed above. In order not to double count, the memory cell leakage is assigned to the memory stage while the read bitline stage will have no associated driver leakage. The senseamp circuitry is treated as an independent stage (RD1). The decoder and timer logic blocks are similarly decomposed into component stages. The bitline differential sensing swing voltage is modeled by V_{swing}.

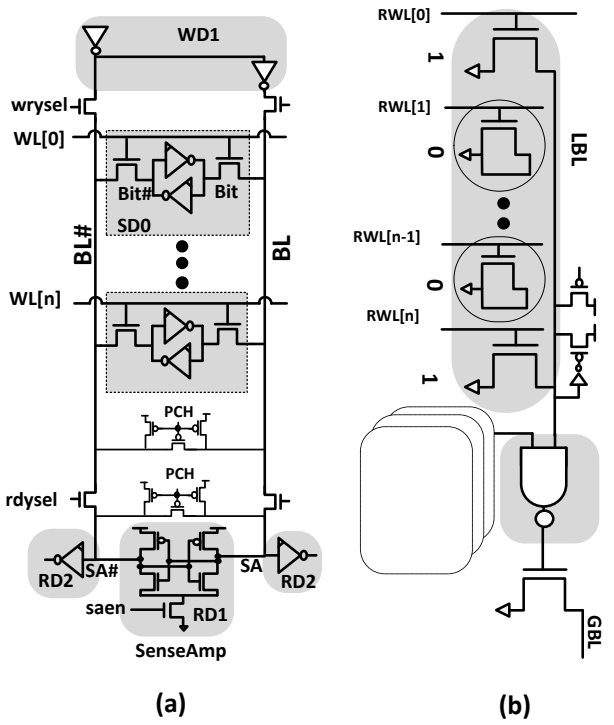


Figure 8.14 (a) Conventional SRAM model stages (b) Conventional ROM model stages

8.12 ROM PROGRAMMABILITY MODELING

The conventional ROM topology (Figure 8.14b) is quite similar to the RF topology. The key differences are the programmability of the read port and the absence of memory storage node or write logic. The ROM programming is typically accomplished by connecting (data = "1") or disconnecting (data = "0") the read port diffusions from the read bitline (diffusion programing). Therefore, the read bitline stage leakage and dynamic power is dependent on programing.

The ratio of programed "1" bit relative to the reference is defined as Pg_{ratio}.

$$Pg_{ratio} = \frac{Pg_1}{Pg_{1\_ref}} \tag{8.60}$$

Pg_1 , Pg_{1_ref} are the percentages of bits programmed as "1". The bitline leakage is scaled by the Pg_{ratio}. From equation 8.13:

$$k_l = Pg_{ratio} \tag{8.61}$$

The read bitline capacitive load is also dependent on programing since the number of connected driver diffusion capacitance is modulated by the programing (diffusion programing). This is modeled by scaling the bitline repeated capacitance component (Φ_r) by Pg_{ratio}. From equation 8.29:

$$k_r = Pg_{ratio} \tag{8.62}$$

8.13 RESULTS AND ANALYSIS

8.13.1 Model Validation

In validating the model, we generated production quality schematic and layout of multiple configurations (bit x entry) of different RF topologies using an in-house compiler. We used a compiler so as to have regularity for model validation. Moreover, we could quickly generate large sample data points for the validation. We ran an in-house timing and power analysis tool on fully extracted layout to obtain the

actual delay, dynamic and leakage power for all configurations. For each distinct topology, we picked a single configuration of that topology as the "reference design". We then used the reference per stage empirical data to estimate the power, area, and delay for the remaining suite of configurations of that topology. We compared our model estimation results against the in-house production analysis tools.

The power validation encompassed the following scenarios 1) Using different *References* (*4bx4e, 16bx32e, 32bx16e, 64bx48e, and 24bx24e*) of the same 1-read, 1-write DE topology to predict the remaining configurations, demonstrating that any configuration of the topology can be used as a reference; 2) *Topology and Multi-Port* validation in which the model is used for distinct topologies (DE and SE write) and multiport prediction; 3) *Delay effect* validation of the model's ability to predict power and delay with timing related gate-sizing changes; (4) *Benchmark* prediction on a custom designed RF with segmentation.

8.13.2 Validation Result

The results of the predicted delay are shown in Figure 8.15. The average error range is 6% for both read and write delay. There were a few outliers at the small bit and entry configuration where anomalies like an extra bitline routing in a 2-entry reference has noticeable impact on the bitline capacitive load.

The leakage error distribution using different reference configurations, topology, multiport, and delay effect is shown in Figure 8.17-Figure 8.20. The average leakage error range is 5%. Outliers are observed when the reference design exhibits circuit characteristics that significantly deviate from the predicted configuration since the model will make its prediction based on the reference. For example, a *4bx4e* reference will have 25% (1 of 4) of the decoder stage, RC2, NAND logic gates in un-stacked leakage state since at least one of its predecoder inputs is always "1". Therefore all estimations based on this reference assume this condition. In actuality a 32-entry fully pre-decoded, for example, will only have 1 of 32 (3.1%) RC2 stage NAND logic gates un-stacked. The total impact of this prediction error is dependent on the contribution of the stage to the total power. This behavior accounts for the relatively large error observed at very small configurations (e.g. *4bx4e*)

Figure 8.21-Figure 8.24 shows the dynamic power error distribution for the various validations. The average dynamic error range is 7%. Dynamic power is sensitive to layout cap ratios of the reference implementation. We observed that the suite of validation RFs had layout anomalies that introduced inconsistencies. For example, the length of lower level connectivity and routing metals, like *Metal1* and *Metal2*, did not necessary scale perfectly across configurations even from a compiler. This error is particularly pronounced with small entry configurations (e.g. *4bx4e*) where minor inconsistencies in layout

can contribute significantly to a total stage capacitance and hence affect the measured dynamic power. This accounted for the 15% error observed when using a *4bx4e* reference or when predicting a *4bx4e* using larger reference. These small array sizes however are not practical configurations on a die.

In summary, the validation results show that given any practical representative reference implementation, the model can accurately predict any configuration of that topology. Results summary is provided in Table 8.5,

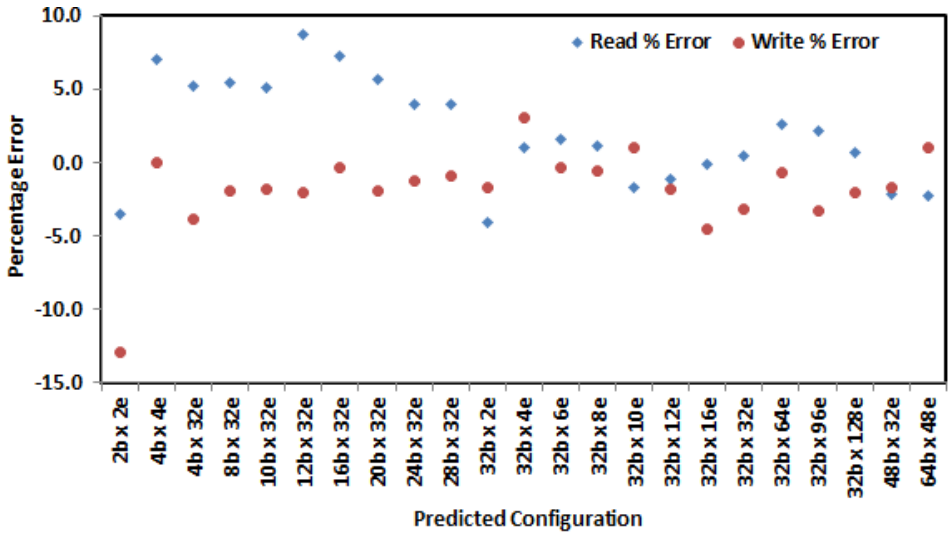


Figure 8.15 Write and Read delay model prediction error the for various DE configurations

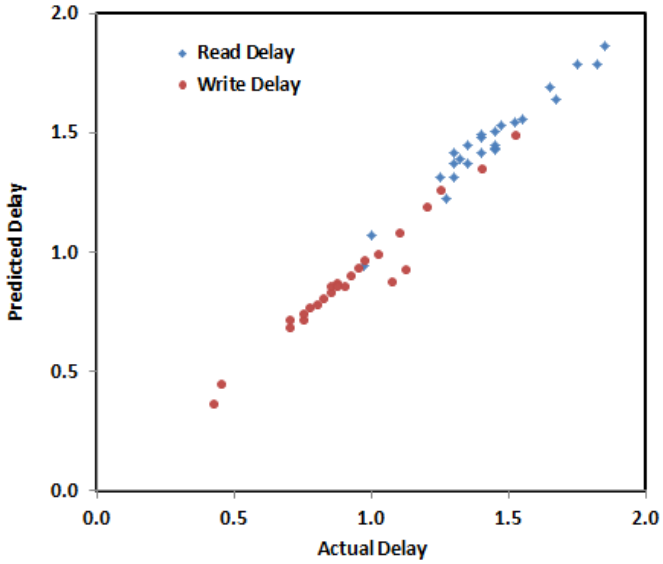


Figure 8.16 Normalized Read and Write Actual vs. Predicted Delay of various DE Configuration

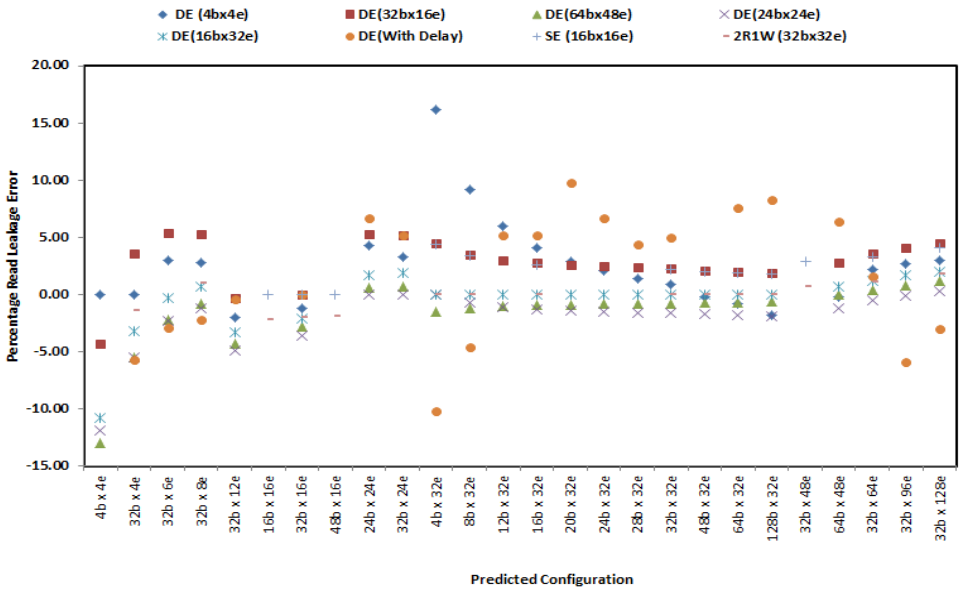


Figure 8.17 Read Leakage prediction error for the various model validations: different reference configuration, topology, multi-ports, and delay effect.

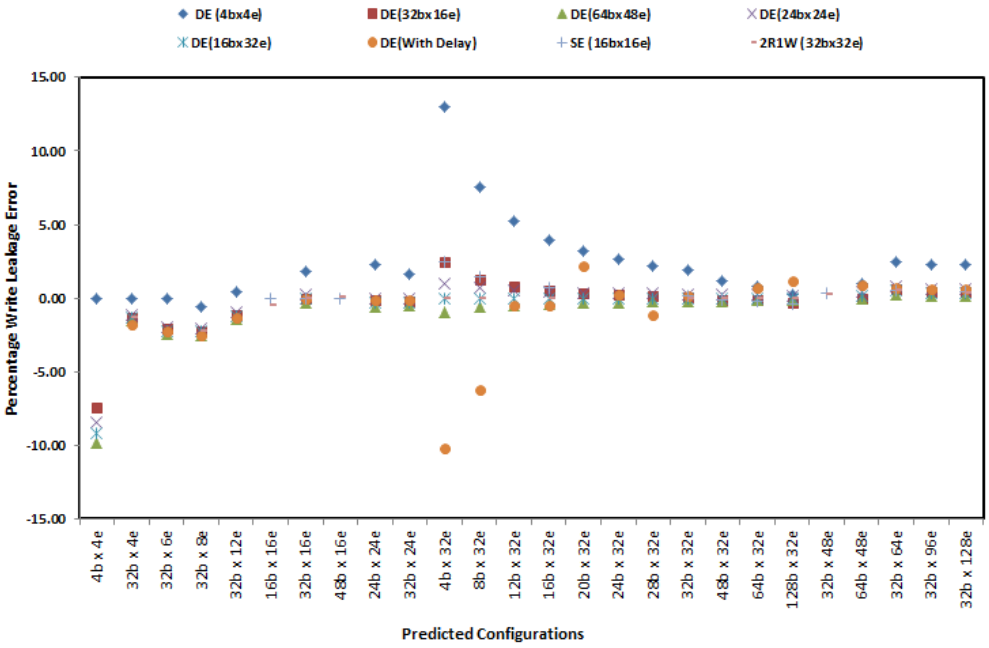


Figure 8.18 Write leakage prediction error for the various model validations: different reference configuration, topology, multi-ports, and delay effect.

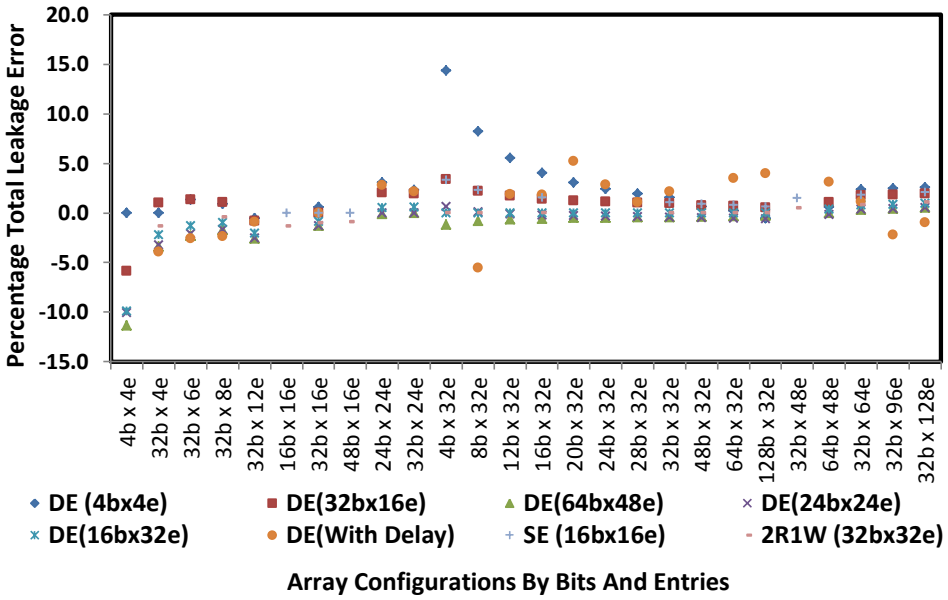


Figure 8.19 Leakage prediction error for the various model validations: different reference configuration, topology, multi-ports, and delay effect.

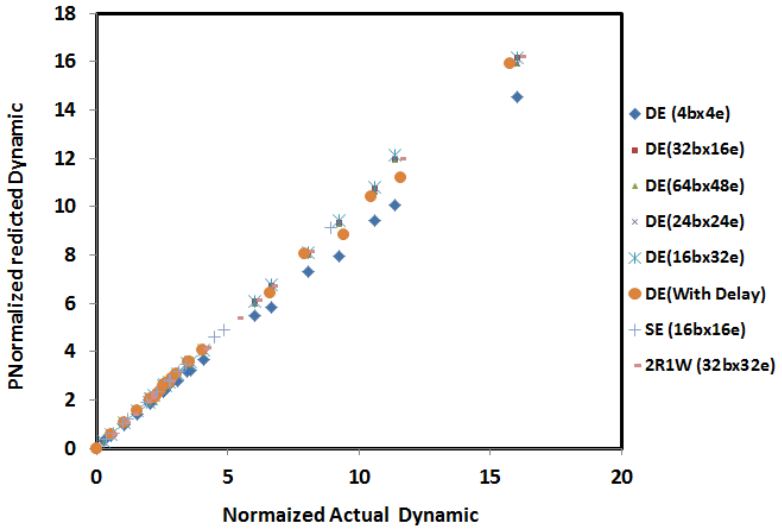


Figure 8.20 Leakage power (actual vs. predicted) for multiple reference designs, topology, multi-ports, and delay effect

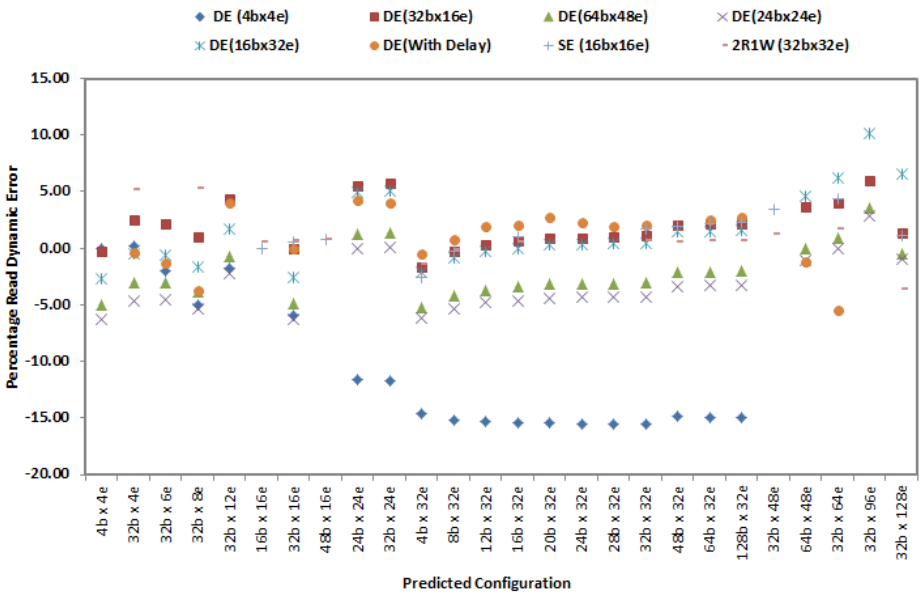


Figure 8.21 Read dynamic prediction error for the various model validations: different reference configuration, topology, multi-ports, and delay effect.

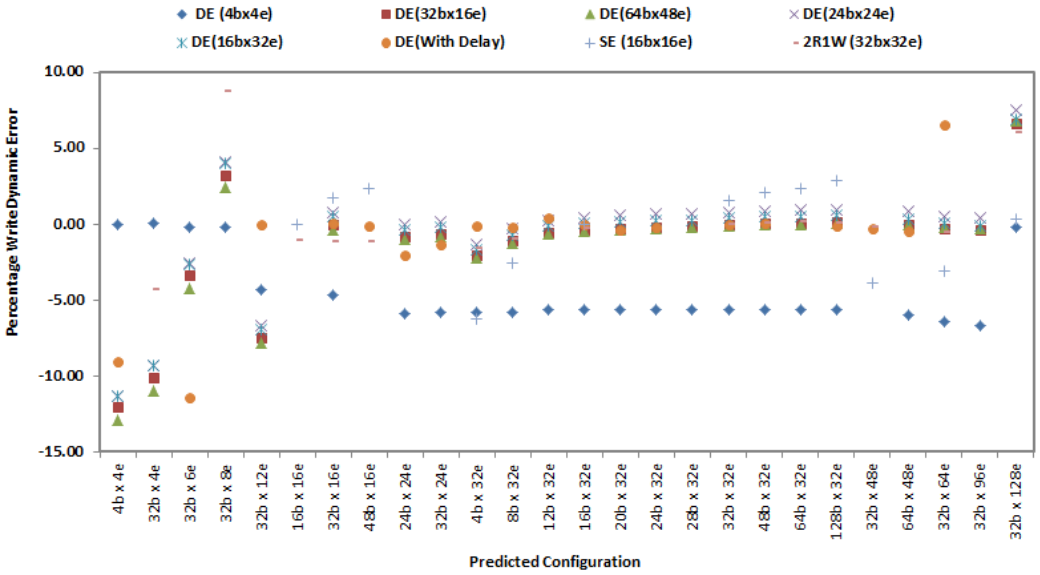


Figure 8.22 Write dynamic prediction error for the various model validations: different reference configuration, topology, multi-ports, and delay effect.

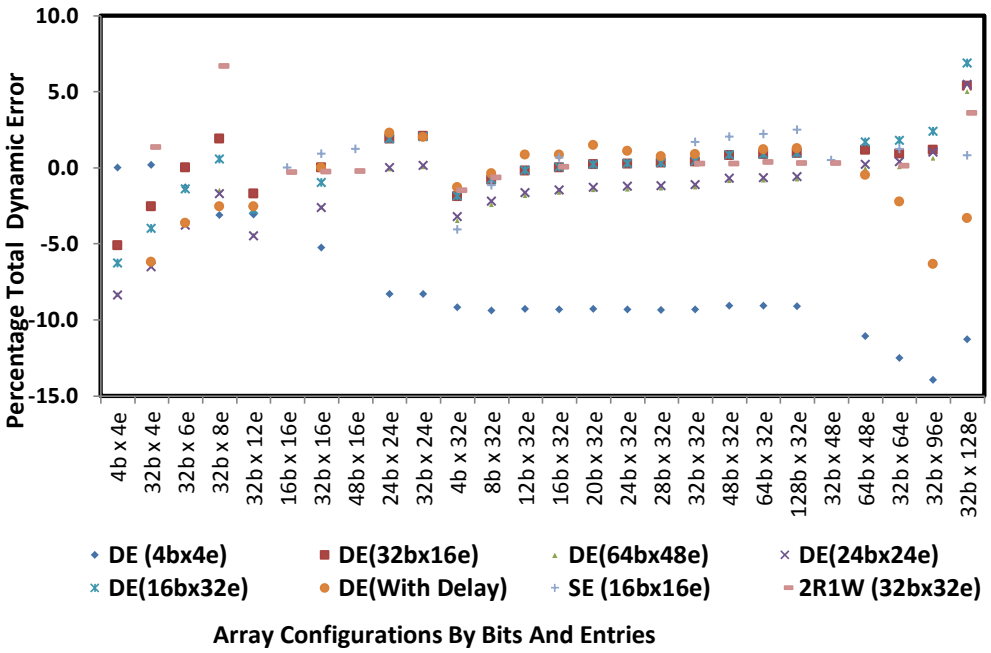


Figure 8.23 Dynamic prediction error for the various model validations: different reference configuration, topology, multi-ports, and delay effect.

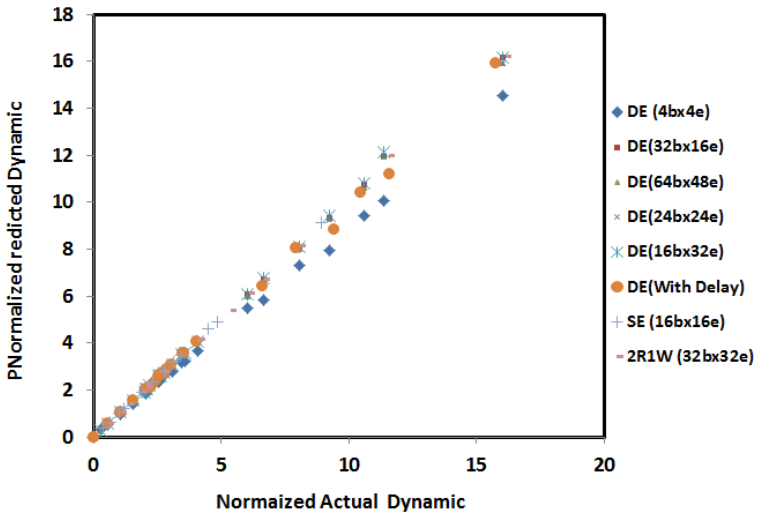


Figure 8.24 Dynamic power (actual vs. prediction) for multiple reference designs, topology, multi-ports, and delay effect

Table 8.5 Leakage prediction error summary

			LEAKAGE					
			Read		Write		Total	
Validation Category	Topology Used	Ref Design Used	Avg Absolute % error	Max Absolute % Error	Avg Absolute % error	Max Absolute % Error	Avg Absolute % error	Max Absolute % Error
Reference & Delay	DE	4B4E	2.9	9.2	2.4	7.5	2.5	8.2
		32B16E	3.2	5.4	1.1	7.4	1.5	5.9
		64B48E	2.1	5.5	1.2	2.5	1.5	3.4
		24B24E	2.3	5.5	1.1	2.0	1.3	3.2
		16B32E	1.5	3.3	0.9	2.3	1.1	2.2
Delay Effect	DE	32B16E	4.7	9.7	1.6	6.2	2.8	6.1
Topology	SE	16B16E	2.2	4.4	0.5	2.5	1.2	3.3
Multi-Port	2R1W (D	32B32E	0.7	2.2	0.3	2.3	0.5	1.4
Average Error for All Validations			2.5	5.6	1.1	4.1	1.5	4.2

Table 8.6 Dynamic prediction error summary

			DYNAMIC					
			Read		Write		Total	
Validation Category	Topology Used	Ref Design Used	Avg Absolute % error	Max Absolute % Error	Avg Absolute % error	Max Absolute % Error	Avg Absolute % error	Max Absolute % Error
Reference & Delay	DE	4B4E	14.3	15.6	4.3	6.7	14.3	15.6
		32B16E	2.3	6.0	2.8	7.5	2.3	6.0
		64B48E	3.1	5.3	3.0	7.9	3.1	5.3
		24B24E	3.9	6.3	2.8	7.5	3.9	6.3
		16B32E	2.7	6.2	2.7	6.9	2.7	6.2
Delay Effect	DE	32B16E	3.7	6.7	3.4	6.6	3.7	6.7
Topology	SE	16B16E	1.7	4.4	2.2	6.3	1.7	4.4
Multi-Port	2R1W (D	32B32E	1.5	5.3	1.6	6.0	1.5	5.3
Average Error for All Validations			4.1	7.0	2.9	6.9	4.1	7.0

8.14 MODEL APPLICATION

8.14.1 Power Distribution and Studies

The model is used to analyze power distribution in an RF to identify high power stages (Figure 8.25). We can then predict the potential impact of RF circuit changes on a per stage basis. This enables us to focus power work on high impact stages. We can identify which low power circuit techniques have high value on any RF based on per stage power contribution. For example, we are able to identify maximum possible leakage reduction if we apply power gating to the read wordline. Similarly if we were to change the memory device type we could quickly estimate the total leakage impact on any array configuration, knowing the memory stage contribution to total array leakage power. We are therefore able to make design driven decision quickly in early design exploration.

8.14.2 Custom Architectural Exploration

We used our power model to perform real world power sensitivity studies. In a typical scenario, an architectural change is proposed to an existing array. Using our model, the empirical data from that specific array is captured as the reference design to model the power sensitivity to the array size for that distinct array. Figure 8.26 shows a distinctive power sensitivity surface plot of a topology under study using its reference.

8.14.3 Topology Comparison and Implementation Driven Architectural Exploration

In another scenario, there exists multiple circuit implementation options for an RF and the question is the best way to implement a specific array given power, area, timing, and other constraints. For this circuit implementation exploration, we modeled the two distinct array topologies, an SE and a DE array, that have different memory bitcells, physical aspect ratios, and write segmentation. Using a single reference design for each topology, we used the model to predict the impact of bit, entry, and other circuit implementation changes on each topology. Figure 8.27 shows the relative power comparison of the two topologies. It also shows the impact of write data gating design space exploration on the two topologies. It can be seen from this study that the DE implementation exhibits significantly higher dynamic power cost at high entry configurations relative to the SE configuration, due to the dual-write operation which is desirable for low-voltage operations. However, when write data gating is implemented, the dynamic power delta between SE and DE is significantly decreased. The power cost of doubling the entries to an existing DE RF can be mitigated by write data gating. This fact is considered by the architect/designer in the analysis and evaluation of an existing DE RF array size increase feature proposal.

8.14.4 Benchmark Prediction

We model different benchmarks to evaluate critical scenarios of interest (Figure 8.28). Architectural and circuit changes unevenly impact different benchmarks. Using the model we are able to quantify potential impact on each specific benchmark, enabling critical decisions. If the critical benchmark is dominated by write power, then write data gating is explored. On the other hand if the critical benchmark is dominated by read, then read segment granularity is of interest.

8.14.5 Early Process Scaling Estimation

In early design phase, one important study is determining process scaling impact on an RF arrays. We can determine the scaling of each custom RF by understanding the uniqueness of each stages and their relative power contribution. We then determine the scale factor for each RF stage that reflects circuit changes and process impact on that specific stage. For example, leakage can be scaled differently for each stage depending on the stage driver device type and the corresponding process scaling per device type. We can also scale stage capacitance components (wirecap, diffusion cap, gate cap) to reflect process

scaling factors for each component on the new process node. We are therefore able to accurately predict and quantify the impact of a new process node on power, area, and delay of each custom RF.

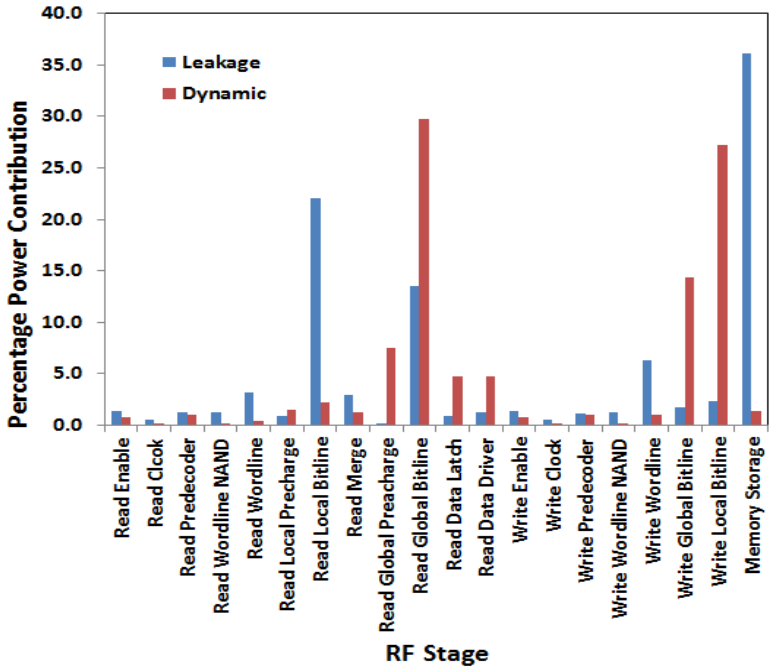


Figure 8.25 Power distribution by stage for a configuration under study

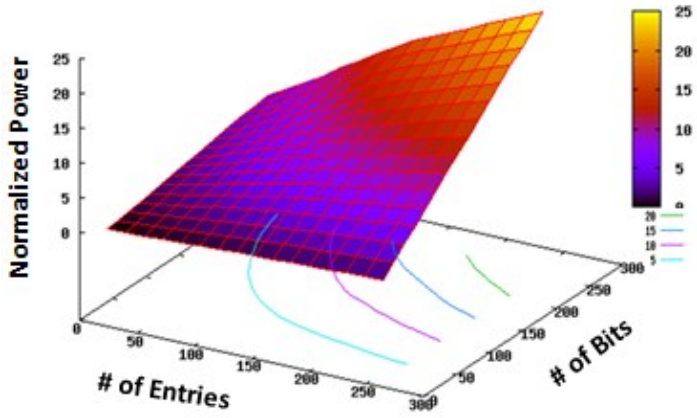


Figure 8.26 A custom RF array power sensitivity to bits and

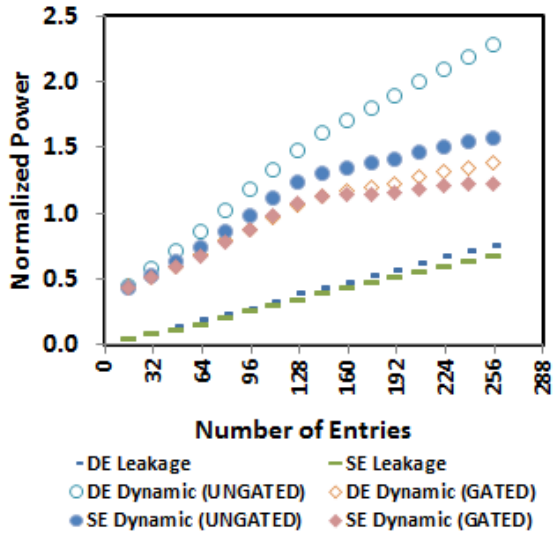


Figure 8.27 Comparison of two RF topologies

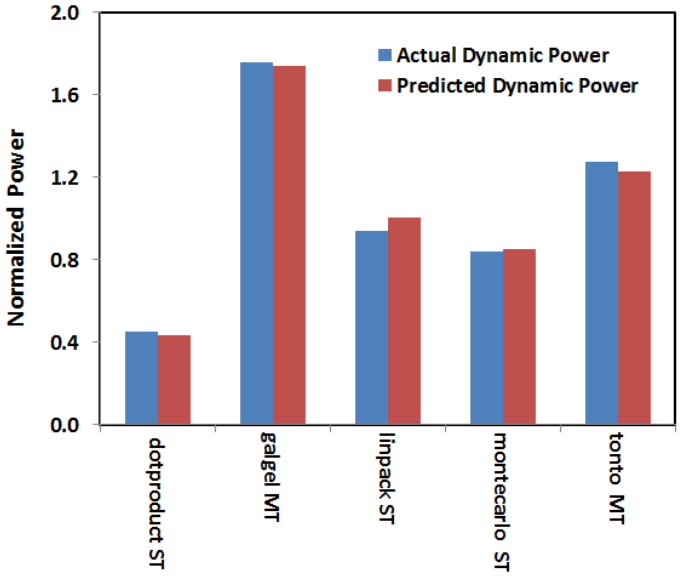


Figure 8.28 Prediction of different benchmarks on 32bx64e custom

8.15 CONCLUSION

In this chapter we presented a customizable hybrid model developed during this research in which the analytical formulas remain unchanged for all topologies while incorporating real design choices such as segmentation, gating, and timing impacts; a combination not in previous RF/SRAM models. The model is customized by using a reference design and decomposing it into component stages, each stage characteristics captured independently by the reference design's empirical data. It allows any representative reference design to be used in estimation and exploration. Changes in topology and/or process require only the empirical data of a single reference to be updated. A typical modern microprocessor has a large number of unique RFs, most of which are manually designed and cannot be compiled, making design exploration across various combinations of bits, entries, gating, and segmentation intractable. Using our model, individual unique RFs can be explored using their respective single reference designs without requiring large number of samples as needed for a curve fit approach. Our model is not tied to any specific technology or design style as we do not model the process, device level, or design environment dependent base values but rely on a reference design empirical data to capture these specifics. Across multiple validation nodes, we show that our reference model approach can accurately predict area, delay, leakage and dynamic power within an average error of 10%. We demonstrated how the model enables tradeoffs both within (e.g. gating, segmentation) and across multiple topologies (SE vs. DE) for optimal implementation and provides wider exploration of circuit implementation options. The proposed reference design approach is also used for delay and area estimations. We also illustrated the model's use cases in modern high performance CPU design.

8.16 ACKNOWLEDGEMENT

I will like to acknowledge Alicia Lowery and Emily Shriver for their invaluable contribution to this work.

8.17 RELATED PUBLICATIONS

[1] E. Donkoh, A. Lowery, E. Shriver., "A Hybrid and Adaptive Model for Predicting Register File and SRAM Power Using a Reference Design" *Design Automation Conference (DAC) 2012*

[2] *"CustPAD: A Customizable Power, Area, and Delay Model for Architectural and Design Space Exploration of Register Files and SRAM Structures Using a Hybrid Reference Design"* Journal Paper in Progress.

8.18 REFERENCE

[1] N. Kurd et al., "A Family of 32nm IA Processors", *IEEE Journal of Solid-State Circuits, 2011, vol 46, pp 119-130.*

[2] K. Anshumali et al. "Circuit And Process Innovations to Enable High-Performance, and Power and Area Efficiency on the Nehalem and Westmere Family of Intel processors" *Intel Technology Journal, 2010, vol 14, pp 104-127*

[3] Eric Donkoh, Teck-Siong Ong, Yan Nee Too, Patrick Chiang, "Register File Write Data Gating Techniques and Break-Even Analysis Model" *ISLPED 2012*

[4] S. Narendra, S. Borkar, V. De, D. Antoniadis, A. Chandrakasan, "Scaling of Stack Effect and its Application for Leakage Reduction", *ISLPED, August 6-7,2001*

[5] Eric Donkoh, Patrick Chiang., "A Low-Leakage Dynamic Register File with Unclocked Wordline and Sub-Segmentation for Improved Bitline Scalability". *ISLPED 2012*

[6] S. Tang et. al., "A leakage tolerant dynamic register file using leakage bypass with stack forcing (LBSF) and source follower", *2002 Symp. VLSI Circuits, pp 320-321*

[7] Tae-Hyoung Kim et al., "A High-Density Subthrshold SRAM with Data-Independent Bitline Leakage and Virtual Ground Replica Scheme" . ISSCC Conference, 2007

[8] S. Borkar, "Circuit techniques for subthreshold leakage, avoidance, control, and tolerance," IEDM Tech. Dig., pp. 421-424,Dec, 2004.

[9] Soumyaroop Roy et al., "State-Retentive Power Gating of Register Files in Multi-core Processors featuring Multithreaded In-Order Cores" IEEE Trans. On Computers, Vol 60, No. 11, Nov. 2011, pp. 1547-1560

[10] Zhinang Hu et al., "Microarchitectural Techniques for Power Gating of Execution Units," ISLPED'04, August 9–11, 2004.

[11] K. Zhang et al., "A 3GHZ 70 Mb SRAM in 65nm CMOS Technology with Integrated Column-Based Dynamic Power Supply" JSSC, vol 41, no. 1, Jan 2006

[12] M. Khellah et al. "A 256-Kb Dual-VCC SRAM Building Block in 65-nm CMOS Process With Actively Clamed Sleep Transistor" JSSC, vol 42, no 1, Jan 2007

[13] Arijit Raychowdbury et al., "PVT-and-Aging Adaptive Wordline Boosting for 8T SRAM Power Reduction. Arijit Raychowdbury et al. ISSCC 2010"

[14] Jaydeep Kulkani et al: "Capacitive-Coupling Wordline Boosting with Self-Induced Vcc Collapse for Write Vmin Reduction in 22-nm 8T SRAM"

[15] Hiroyuki Yamauchi, "Embedded SRAM Circuit Design Technologies for a 45nm and Beyond", ASICON '97

[16] D. Brooks, et al, "Wattch: A Framework for Architectural-Level Power Analysis and Optimization," *ISCA 2000*.

[17] Mamidipaka, M.; Khouri, K.; Dutt, N.; Abadir, M. "IDAP: a tool for high-level power estimation of custom array structures" *IEEE Trans on Computer-Aided Design Vol 23, No. 9 2004, pp 1361-1369*

[18] S. J. E. Wilton and N. P. Jouppi, "CACTI: An enhanced cache access and cycle time model," *IEEE J. Solid-State Circuits*, vol. 31, no. 5, pp. 677–688, May 1996.

[19] N. Vijaykrishnan, et al. "Energy-driven integrated hardware-software optimizations using SimplePower", *ISCA 2000*.

[20] A. Zeng, A.; Rose, K.; Gutmann, R.J. "Memory performance prediction for high-performance microprocessors at deep submicrometer technologies" *IEEE Trans. On Computer-Aided Design. Vol 25, No. 25. Sept 2006 pp1705-11718*

[21] X. Dong, et al, "NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory" *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, pp. 994–1007*.

[22] Sheng Li et al., "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures" *MICRO Dec 12-16 2009*

[23] Agrawal, B.; Sherwood, T., "Ternary CAM Power and Delay Model: Extensions and Uses" *IEEE Trans on VLSI, Vol 16, No 5, 2008, pp 554-564*

[24] Xuemei Zhao, et al: "Design and Realization of a Low Power Register File Using Energy Model", *PATMOS 2002:* pp. 268-277.

[25] Minh Q. Do, et al, "Parameterizable Architecture-Level SRAM Power Model Using Circuit-Simulation Backend for Leakage Calibration," pp.557-563*, ISQED 2006*.

[26] S. L. Coumeri, D. E. Thomas Jr, "Memory Modeling for System Synthesis", *IEEE Transactions on VLSI,* June 2000.

[27] Xiaoyao Liang, Kerem Turgay, David Brooks, "Architectural Power Models for SRAM and CAM Structures Based on Hybrid Analytical/Empirical Techniques"*, ICCAD 2007*.

[28] Giby Samson et al., "Low-Power Dynamic Memory Word Line Decoding for Static Random Access Memories". IEEE Journal of Solid-State Circuits, VOL 43, NO. 11, November 2008, Pg 2524-2532.

[29] M.A. Turi, J.G. Delgado-Frias, "Decreasing energy consumption in address decoders by means of selective precharge schemes" Microelectron. J (2009), doi:10.1016/j.mejo.2009.03.008.

[30] Bharadwaj S. Amrutur, and Mark A Horowitz, "Fast Low-Power Decoders for RAMs", IEEE Journal of Solid-State Circuits, Vol 36, No. 10, October 2001, Pg 1506-1515

9 ARRAYPAD – AN INTERACTIVE WEB INTERFACE FOR ARRAY POWER, AREA, AND DELAY ESTIMATION

9.1 INTRODUCTION

ArrayPAD, an interactive Array Power, Area, and Delay web interface implements the models developed during this research. The purpose of this tool is to provide a user friendly interface for array power, area, and delay estimation across multiple projects. Reference designs of different RF and ROM topologies across multiple projects are made available for power, area, delay estimation and tradeoff analysis.

ArrayPAD can be used by architects and circuit designers for

(1) Architectural exploration to determine the power, area, and timing impact of changing the number of bit, entry, or ports of an existing array.

(2) Estimation of power, area, and timing for a new array

(3) Designer exploration of circuit implementation choices such as bitcell type selection, segmentation, sizing, and gating.

(4) Power, area, and timing tradeoff analysis on multiple array topologies during design planning.

(5) Compare process scaling impact on multiple arrays.

9.2 ARCHITECTURE

The high-level architecture of the web tool is shown in Figure 9.1. The application is implemented with PHP. An XLS configuration file contains model reference input parameter. Users can also enter configuration information through the web interface – enabling users to modify reference parameters for What-If analysis. A Gnuplot generates a 3D plot while SWF is used to generate the 2D plots. A result tables is displayed on the web page. A user can also optionally download the results in an XLS.

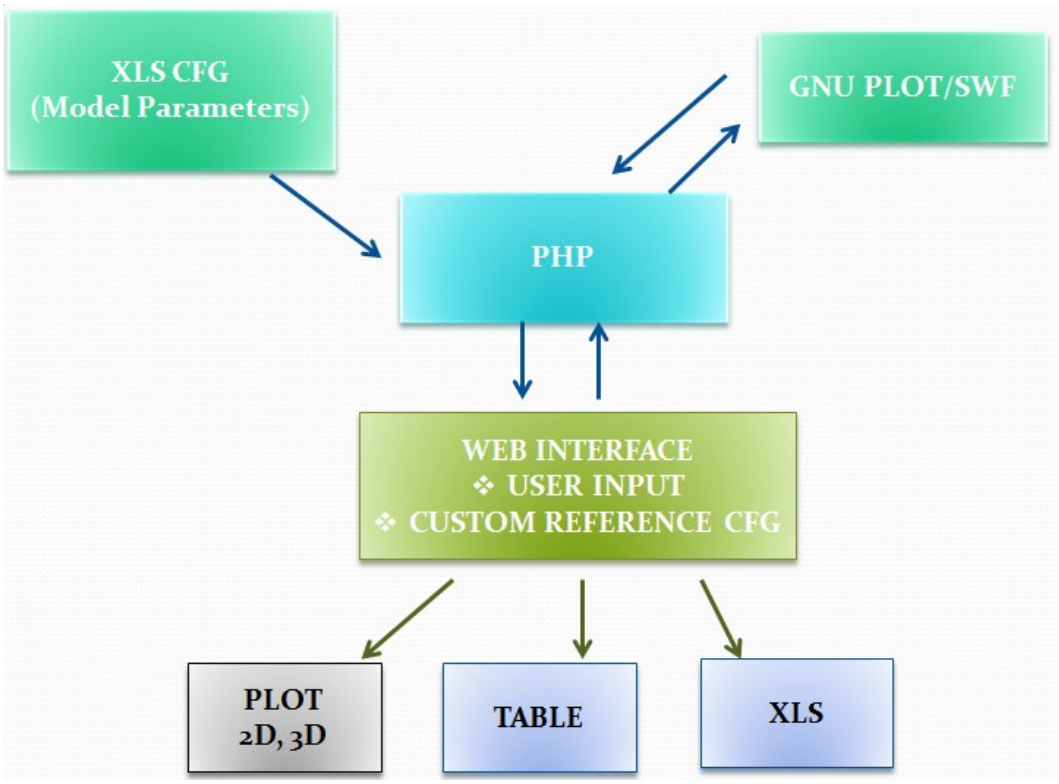


Figure 9.1 Array Models Web Implementation Architecture

9.3 INTERACTIVE INTERFACE

Figure 9.2 shows ArrayPAD's interactive web interface implemented. The main Panels are:

(1) **Reference Design Panel** – User selects a reference design(s) for estimation. Multiple reference designs can be selected. The parameters of the reference design are pre-defined.

(2) **Option Panel** - User specifies the range of Bits and Entries, Number of Ports, to estimate for the selected reference design. User can either sweep Bits and Entries independently or both. User can also specify AF/SP for benchmark analysis.

(3) **Custom Reference Design Table Panel (FOR EXPERT USERS)** – User can specify their own reference parameters. They can also use an existing reference design and make changes

where appropriate. For example a user can load a 16-entry write bitline segmented reference design and change the segmentation to 32-entry to evaluate 16-entry vs. 32-entry segmentation. Figure 9.4 shows the custom reference table form.

(4) **Stage Analyzer Panel** – User selects which array stage results to display. This allow user to control which array stages to study. For example a user might want to analyze all write stages or just write bitline trend.

(5) **Result Panel** – The results panel display the model result based on user selections. The results are displayed in 3 forms.

a. **Plot –** 2D dynamic plots are generated dynamically using SWF. A 2D plot is generated when user sweeps either bits or entries independently. Figure 9.3 shows a 2D plot of comparing three different reference designs. 3D Surface plots are also generated dynamically with GNU Plots when a user sweeps both bits and entries. Figure 9.4 shows 3D surface plot as both Bits and Entries are swept. User can decide the orientation of the contour plot by changing the x-axis to either Bits or Entries.

b. **Raw Table** – The table shows the raw estimated number from the tool, corresponding to the plotted values

c. **CSV download** – The raw results can be downloaded as an XLS file

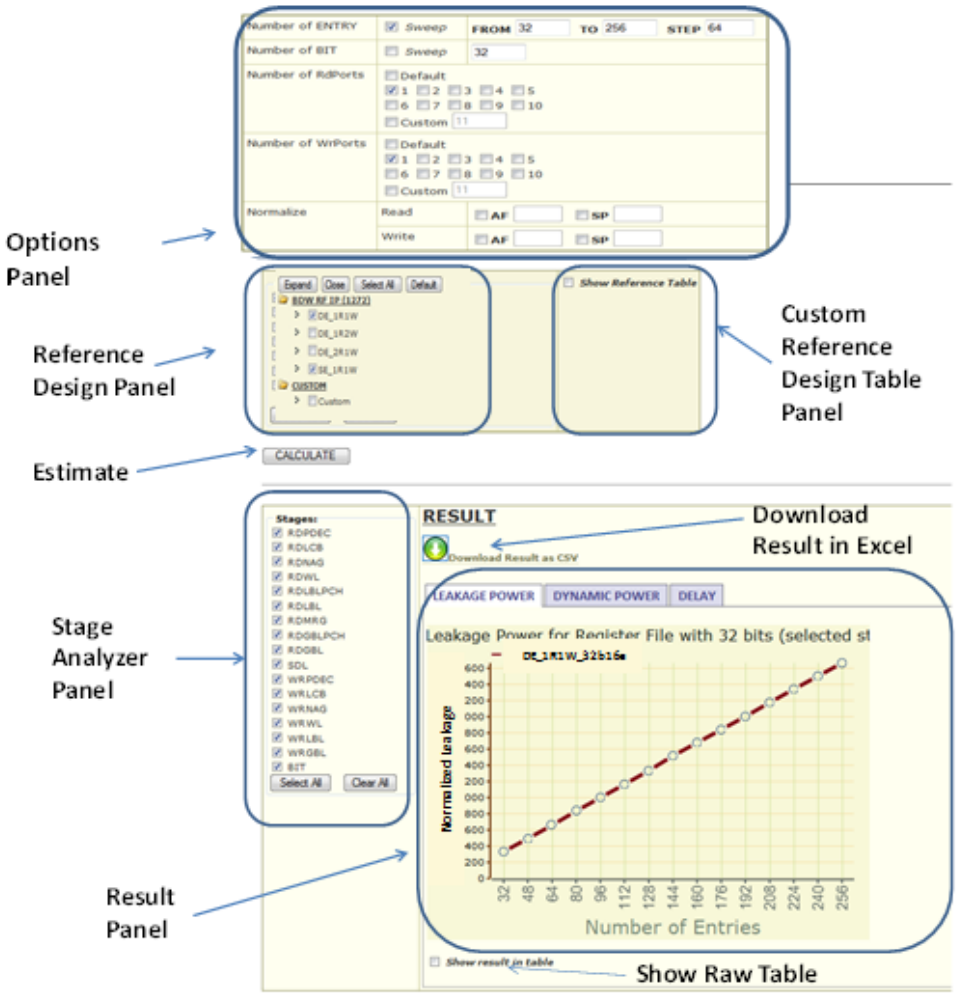


Figure 9.2 Model interactive web interface showing the various panels

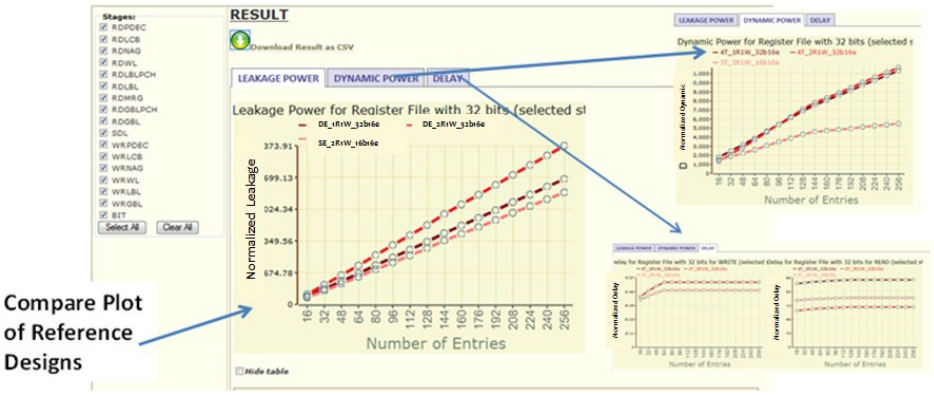


Figure 9.3 Results panel showing dynamically generated 2D charts comparing multiple reference designs: leakage, dynamic, delay estimation. The estimation results can also be saved in an XLS file.

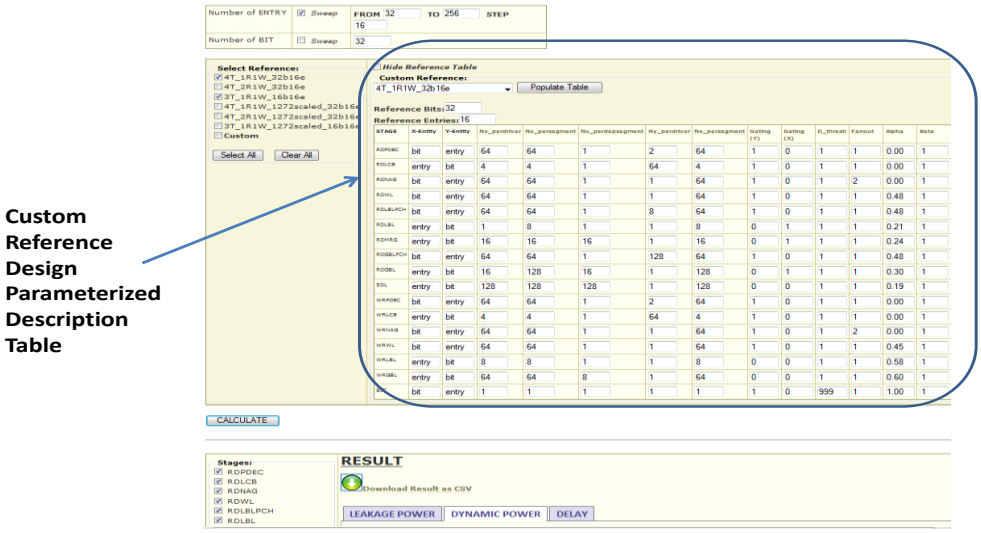


Figure 9.4 ArrayPAD custom reference design input form. Each array stage is defined by a set of parameters that describes topology and reference empirical data

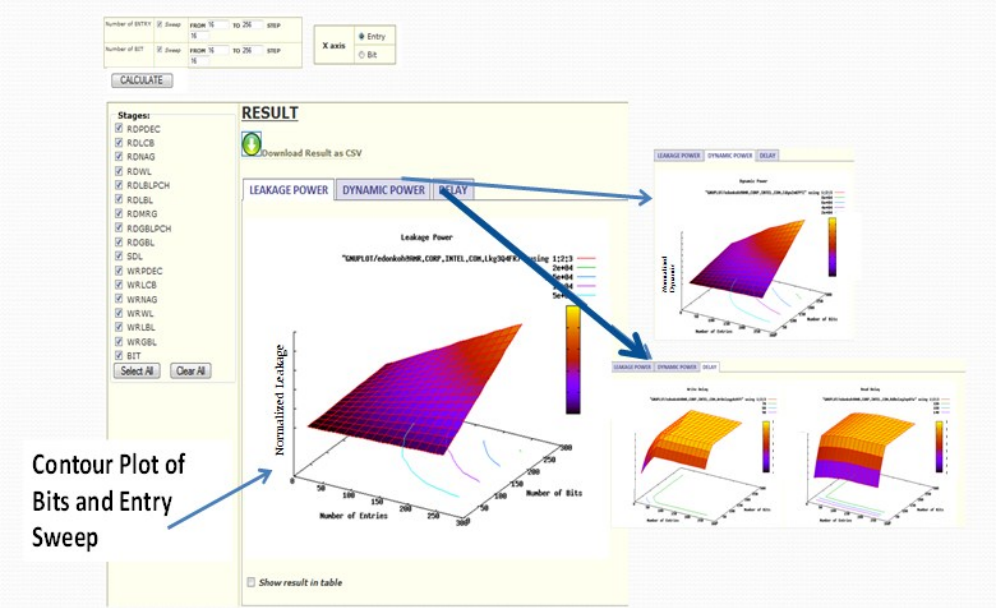


Figure 4: ArrayPAD results panel showing dynamically generated 3D surface plot from sweeping both Bits and Entries

9.4 USAGE MODEL

9.4.1 Basic Usage Steps model

1. Select the reference you want to base your estimate on from the "Reference Panel"

2. Select Number of Bits and Entry range to sweep. User can also enter a specific Bit or Entry. If only Bit or Entry is swept, a line chart is produced. If both Bit and Entry are swept, a 3-D Chart is produced

3. Select the Number of Write and Read Ports you want for the reference

4. User can optionally normalize all stage AF/SP (else the reference default AF/SP is assumed). To check what those are, you can populate the custom reference table and look at the stage AF/SP (Expert Users)

5. Click on “Calculate”

6. The Results Panel shows the results charts and tables.

7. User can choose which stages to include in your results.

8. Results are displayed in a table form

9. Click on “Download Results as CSV” to optionally download the results in an XLS sheet

9.4.2 Expert Usage Model

1. User can make modifications to the reference

2. Click on “Show Reference Table”

3. Select the reference you want to modify from the dropdown menu

4. “Populate Table”

5. Select which stages you want to modify from the stage dropdown menu and make the desired changes.

6. Under the “Reference Panel” , Click on “Custom” folder to expand and make sure the “Custom” checkbox is selected

7. Click on “Calculate”

9.5 ACKNOWLEDGEMENT

I will like to acknowledge Ee Wah Lim for his contribution to building the web interface.

10 CONCLUSION AND FUTURE WORK

One of the goals of this research was to develop new low-power circuit techniques for register files and SRAM-like structures. We've proposed various technique to reduce dynamic and leakage power in key register file blocks: array, write data path, read data path, clocks, and pre-decoder/decoder . Though register files were the focus of this research, these low-leakage techniques are application to other SRAM structures. For example the shared decoding techniques apply to any SRAM-like structure with decoding. The proposed structural techniques such as bitline sub-segmentation are also application to ROM, CAM and others with similar structures.

We proposed new power gating techniques that reduces the overhead and hence the cost of implementing power gating in register files. This is very useful in reducing leakage which is the biggest challenge facing array power. As devices shrink over process generations, threshold scales down, and more cores are added per die, leakage begins to dominate processor power. New practical leakage reduction techniques such as those proposed during this research are therefore critical to future power scaling.

Designing low-power circuit is never done in isolation. Tradeoffs have to be made across various design metric: delay, area, Min-Vcc, schedule, etc. Key to being able to accurately make these tradeoff decisions is the ability to model and estimation proposed changes before implementation. This has traditionally been a difficult task especially for register files due to the large design space available. We've developed a customizable model that can estimate power, area, and delay of any register file and SRAM-like structure. The uniqueness of this model is its customizability, using the same model equations for different topologies. This is a vast improvement over existing model that are not easily adaptable to new topologies. The proposed model also allow designer and architects alike to explore common circuit implementation options such as segmentation, stacking, and gating in their analysis, a feature not present in existing models.

We have implemented the proposed model as an internal tool at Intel. The tool is being used for array estimation, projection, scaling studies, and for tradeoff analysis and decision making for the next generation SoC and high performance CPU design in 22nm and beyond.

10.1 FUTURE MODEL WORK

Future work on the modeling include the following

1. Incorporate voltage, temperature, and thermal models

2. Model more cross topology design optimizations.

3. Explore the possibility of making the model open source to the wider community outside of Intel. This will most likely involve developing references based on external process technologies.