RING-BASED RESONANT STANDING WAVE OSCILLATORS FOR 3D CLOCKING APPLICATIONS

A Thesis

by

ANDREW JAMES DOUGLASS

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Sunil P. Khatri
Committee Members,	Jiang Hu
	Anxiao Jiang
Head of Department,	Miroslav Begovic

May 2019

Major Subject: Computer Engineering

Copyright 2019 Andrew James Douglass

ABSTRACT

Ring-based resonant standing wave oscillators have been shown to be a useful clocking technique that can distribute and generate a high frequency, low skew, low power, and stable clock signal. By using through-silicon-vias, this type of standing wave oscillator can be used to generate the clocking scheme for 3D integrated circuits. In this thesis, we propose the use of such 3D standing wave oscillators and show how independent 3D oscillators in different stacks can synchronize through the use of a redistribution layer stub. Inter-chip clock synchronization is then accomplished without the need for a PLL. In addition, we propose the first 3D ring-based resonant standing wave oscillator bootstrap and reset circuit to initialize and stop oscillation.

Using a 3D ring-based resonant standing wave oscillator, we propose a ring-based data fabric for 3D stacked DRAM and compare the results with existing approaches such as High Bandwidth Memory (HBM) or Wide I/O memory. We show that our Memory Architecture using a Ring-based Scheme (MARS) can provide the increases in speed necessary to overcome current memory bottlenecks, and can scale effectively as future 3D stacks become larger. Our MARS can trade off power, throughput, and latency to match different application requirements. By using a narrow bus, and connecting it to all channels, the MARS8 can provide an alternative memory configuration with $\sim 6.9\times$ lower power consumption than HBM, and $\sim 2.7\times$ faster speeds than Wide I/O. Using multiple ring topologies in the same stack, the channel count can double from 8 to 16, and then to 32. This is possible since MARS uses about $4\times$ fewer TSVs per channel than HBM or Wide I/O. This provides speeds up to $\sim 4.2\times$ faster than traditional HBM. This scalable architecture allows higher throughput and faster system performance for next-generation DRAM. The MARS topology proposed in this thesis can be used in a variety of computing systems, from lightweight IoT to large-scale data centers.

ACKNOWLEDGMENTS

I would like to thank my thesis advisor, Dr. Sunil Khatri, for his guidance and continual support throughout the course of my college career. Thanks also go to my friends, colleagues, and the faculty staff for making my time at Texas A&M University a great experience. I also want to extend my gratitude to the faculty of the office of graduate and professional studies.

In addition, I acknowledge the US Government's support in the publication of this research. This material is partly based upon work funded by AFRL/AFOSR, under AFRL Contract No. FA8750-16-3-6003. Any findings/conclusions/recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of AFRL.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was presented to a thesis committee consisting of Professor Sunil Khatri and Professor Jiang Hu of the Department of Electrical and Computer Engineering as well as Professor Anxiao Jiang of the Department of Computer Science and Computer Engineering.

The connection of multiple standing wave oscillators via an RDL stub was assisted by Dr. Mark Linderman of the United States Air Force Research Lab (AFRL). All other work conducted for this thesis was completed by the student independently.

NOMENCLATURE

PVT	Process, Voltage, and Temperature
IC	Integrated Circuit
VLSI	Very Large-Scale integration
IP	Intellectual Property
ASIC	Application-Specific Integrated Circuit
PCB	Printed Circuit Board
TSV	Through-Silicon Via
DRAM	Dynamic Random-Access Memory
RAS	Row Access Strobe
CAS	Column Access Strobe
JEDEC	Joint Electron Device Engineering Council
DIMM	Dual In-line Memory Module
DDR	Double Data Rate
DMC	Dynamic Memory Controller
MC	Memory Chip
CPU	Central Processing Unit
GPU	Graphics Processing Unit
RRSWO	Ring-based Resonant Standing Wave Oscillator
PLL	Phase-locked loop
VCO	Voltage-Controlled Oscillator
RDL	Redistribution Layer
RLC	Resistor, Inductor and a Capacitor

HBM	High Bandwidth Memory
MARS	Memory Architecture using a Ring-based Scheme
IES	Insertion-Extraction Station
FIFO	First-In, First-Out
CMOS	Complementary Metal-Oxide-Semiconductor
PTM	Predictive Technology Model
PCM	Phase-Change Memory

TABLE OF CONTENTS

Page

ABSTRACT ii

ACKNOWLEDGMENTS iii

CONTRIBUTORS AND FUNDING SOURCES iv

NOMENCLATURE v

TABLE OF CONTENTS vii

LIST OF FIGURES ix

LIST OF TABLES xii

1. INTRODUCTION 1

1.1 Overview 1
1.1.1 Clock Network Design Challenges 2
1.1.2 Desired Clock Network Properties 3
1.1.3 Traditional Clock Distribution Schemes 6
1.2 3D IC Stacks 10
1.3 DRAM History and Background 12
1.3.1 DRAM Terminology 14
1.3.2 Graphics DRAM 16
1.3.3 Low Power DDR 17
1.3.4 Current Dual In-line Memory Module Issues 18
1.3.4.1 Heat and Power Consumption 18
1.3.4.2 Pin Count 18
1.3.4.3 Refresh Rate 19
1.3.4.4 Commodity Status 20
1.3.4.5 Scalability 20

2. SCALABLE AND SYNCHRONOUS 3D CLOCK DISTRIBUTION 21

2.1 Introduction 21
2.2 Terminology 21
2.3 Previous Work 22
2.3.1 Resonant Standing Wave Oscillators 24
2.4 Our Approach 28
2.4.1 Bootstrap Circuit 29

2.4.2 Standalone 3D RRSWO 31
2.4.2.1 Experimental Results 32
2.4.3 Synchronized 3D RRSWOs 38
2.4.3.1 RDL Stub-based Synchronization of two 3D RRSWOs 38
2.4.3.2 Tristateable Inverter Pair Circuit 39
2.4.3.3 Experimental Results 40
2.4.3.4 Monte Carlo Simulations 52
2.5 Conclusion 54

3. 3D RRSWOS FOR DRAM APPLICATIONS 56

3.1 Introduction 56
3.2 3D Stacked Memory 56
3.2.1 High Bandwidth Memory 57
3.2.2 Wide I/O Memory 59
3.3 Previous Work 59
3.4 Our Approach 61
3.4.1 Ring-Based Interconnect for 3D DRAM 61
3.4.2 Experimental Results 69
3.4.2.1 3D Ring Clock 71
3.4.2.2 DRAM Latencies 74
3.4.2.3 DRAM Power Consumption 83
3.4.2.4 DRAM Bus Utilization 90
3.5 Conclusion 92

4. SUMMARY AND CONCLUSION 94

4.1 Future Work 95

REFERENCES 97

LIST OF FIGURES

FIGURE Page

1.1 Clock distribution network skew example 4

1.2 Clock distribution network jitter example 5

1.3 Common clock tree distribution schemes 6

1.4 H-tree Clock Network 7

1.5 Grid Clock Network 9

1.6 Example 3D IC Stack 11

1.7 FPM DRAM read operation with row kept open 13

1.8 DDR SDRAM read operation 14

1.9 Traditional DIMM architecture with multiple channels 16

1.10 GDDR5 top view based on Nvidia's GeForce GTX Titan Black [1] 17

1.11 Growing refresh rates as DRAM banks increase in size 19

2.1 Resonant clock ring with recovery circuits [2] 25

2.2 Differential amplifier clock extraction circuit 26

2.3 Increasing the perimeter of the RRSWO 27

2.4 Bootstrap reset circuit 29

2.5 Bootstrap reset waveform for single 2D RRSWO 30

2.6 2D to 3D adaptation for an RRSWO 32

2.7 Clock TSV cross sectional view 33

2.8 Bar chart legend 33

2.9 Overlaid standing waveform for example RRSWO 34

2.10 Overlaid extracted clock waveform for example RRSWO 35

2.11 Period of independent RRSWO ... 36
2.12 Skew of independent RRSWO ... 36
2.13 Power of independent RRSWO ... 37
2.14 Synchronizing the RRSWOs of two 3D ICs ... 38
2.15 Tristateable inverter pair ... 39
2.16 Power-up of 2 3D SWOs with the variable size bootstrap circuit ... 41
2.17 Period synchronization of two 3D RRSWOs with 3 segments each ... 42
2.18 Intra-ring skew of two 3D RRSWOs with 3 segments each ... 43
2.19 Inter-ring skew of two 3D RRSWOs with 3 segments each ... 44
2.20 Power consumption and RDL stub current of two 3D RRSWOs ... 45
2.21 Period of two 3D RRSWOs with varying number of segments ... 47
2.22 Intra-ring skew of two 3D RRSWOs with varying number of segments ... 48
2.23 Inter-ring skew of two 3D RRSWOs with varying number of segments ... 49
2.24 Power consumption of two 3D RRSWOs with varying number of segments ... 50
2.25 Percent decrease in oscillation frequency when RDL stub is introduced ... 51
2.26 Period of two 3D RRSWOs for Monte Carlo simulations ... 53
2.27 Skew of two 3D RRSWOs for Monte Carlo simulations ... 53
2.28 Power and RDL stub current of two 3D RRSWOs for Monte Carlo simulations ... 54
3.1 HBM cross sectional view ... 57
3.2 HBM2 pseudo channels overview [3] ... 58
3.3 Cross sectional view HBM (left) and MARS (right) for a DRAM stack ... 62
3.4 Cross sectional view MARS16 (left) and MARS32 (right) ... 63
3.5 IES logic design for data and channel lines ... 64
3.6 IES logic design for row address lines ... 67
3.7 IES logic design for column address lines ... 67

3.8 Input asynchronous FIFO [2] ... 70

3.9 Cross section view of clock TSVs ... 71

3.10 $18GHz$ ring clock signal before differential amplifiers along the 3D ring ... 74

3.11 Total read latency for current memory architectures ... 78

3.12 Latency per command for current memory architectures ... 78

3.13 Queue length per channel for current memory architectures ... 80

3.14 Total read latency for 3D memory architectures ... 81

3.15 Latency per command for 3D memory architectures ... 82

3.16 Queue length per channel for 3D memory architectures ... 82

3.17 Average power consumption for current memory architectures ... 84

3.18 Total energy consumption for current memory architectures ... 85

3.19 Performance per Watt for current memory architectures ... 86

3.20 Average power consumption for 3D memory architectures ... 87

3.21 Total energy consumption for 3D memory architectures ... 88

3.22 Performance per Watt for 3D memory architectures ... 89

3.23 Bus usage for different MARS configurations ... 91

3.24 Bus utilization vs performance reduction for MARS ... 92

4.1 Synchronizing multiple MARS stacks using an RDL stub ... 95

LIST OF TABLES

TABLE Page

1.1 DDR I/O bus transfer rates ... 20

2.1 VDD jitter for varying RDL stub locations connecting two 3D RRSWOs ... 46

2.2 VDD jitter for varying number of segments of two 3D RRSWOs ... 50

3.1 IES versus propagation delay ... 66

3.2 Total number of TSVs ... 68

3.3 3D RRSWO Monte Carlo ... 72

3.4 Ramulator DRAM configurations ... 75

3.5 Theoretical maximum bandwidth ... 76

3.6 CPU trace statistics ... 77

3.7 MARS power consumption breakdown (mW) ... 90

1. INTRODUCTION

1.1 Overview

The technological revolution over the past several decades has revolutionized our daily lives. From high-performance data centers to miniaturized wearable devices, technology has become an integral part of our daily life. As the dependence on technology has grown, so have the technological innovations. Over the years we have seen devices shrink, integrated circuits (ICs) become more complex and processor speeds grow rapidly. Despite these tremendous innovations, many challenges still plague modern IC development.

A common challenge that still faces most IC designs is the ability to generate *and* distribute a clock signal. Clock signals have been heavily utilized since the first ICs in order to allow synchronization between different logical components. The clock signal provides a means to synchronize the movement of data signals throughout an IC in a fast and efficient manner. Most designs consist of banks of synchronized registers that are separated by combinational logic. This creates functional "stages" in the design where the timing of each stage can be evaluated using timing analysis. By analyzing the timing of these stages, designers can ensure the clock meets the requirements of each logical stage. In most cases, this is done by ensuring the clock signal switches at a rate slower than the worst-case propagation delay of any given stage.

A clock distribution network distributes the clock signal from an originating point on the IC to a set of clock regenerators, which each drive the clock signal to a subset of the synchronized registers of the design. As ICs become larger and more complex, this clock distribution network becomes a significant design challenge. Because the proper operation of a clock signal is vital to the overall chip's operation, significant attention is given to the characteristics of the clock distribution network. The design of an efficient clock network can improve both the performance and the power consumption of the IC. In addition, a well-designed clock network helps ensure that the timing requirements are met, for correct communication between functional elements.

1.1.1 Clock Network Design Challenges

Clock signals are foundational to IC development and logic communication. The distribution of these clock signals, however, have many challenges that require careful consideration. Among the difficulties engineers experience when designing these clock distribution networks include large fan-outs, asymmetric physical layouts, technology scaling, as well as process, voltage and temperature (PVT) variations. Addressing these challenges often requires trade-offs and a balance must be struck for each design based on the system requirements.

The first major challenge faced when designing a clock distribution network is the large fan-out of the clock signal. This large fan-out is a result of the clock signal distributing from a single source location to a large number of clock regenerators on an IC. Because the regenerators require a low skew, low jitter rail-to-rail signal, the design of the clock distribution network must handle the large fan-out. This is typically accomplished by appropriately inserting buffers throughout the network to restore the signal strength. The downside of using buffers is that it results in increased clock skew due to PVT variations. The change of voltage, also known as the slew rate, provides a way to measure how "sharp" this clock edge is at the desired regenerator inputs.

The physical layout of the clock network is another challenge that is becoming more difficult as ICs become larger. The challenges with implementing traditional clock distribution networks is that they tend to not follow common very-large-scale integration (VLSI) block partitions. Even though this has historically been addressed by pre-routing the clock network, the recent move towards intellectual property (IP) based design makes it hard if not impossible to pre-route clock signals. This results in irregularities in the layout of the clock network which results in sub-optimal performance. Also, functional logic blocks sometimes maintain their own locally optimized clock networks. The clock distribution scheme must adapt to these different networks.

The clock distribution network has also been severely affected by technology scaling. As new technology generations use smaller process nodes, the long global wires used in a clock distribution network have become more resistive. This is a result of the decreased width of these global wires. With an increased resistance, the propagation delay of the clock signal becomes more important.

Performance demands have resulted in the need for faster switching frequencies while the global wires have become more resistive. Therefore, the ability to ensure proper arrival of the clock edges at different clock endpoints is becoming increasingly difficult with technology scaling, and the increase in wire delays relative to logic delays.

Finally, another major challenge in designing clock distribution networks is that clock paths, interconnect buffers and wire parasitics are highly sensitive to PVT variations. Therefore, as the interconnect, device and operational parameters vary, the performance metrics of a given clock distribution network can fluctuate. This can degrade the reliability of the synchronous elements of an IC and even cause system failure. Therefore, clock distribution networks must be designed to handle process and environmental variations to ensure an IC can perform at its full potential.

1.1.2 Desired Clock Network Properties

The previously mentioned clock distribution design challenges are measured by the ability of the clock network to meet a set of desired metrics. The predominant metrics used to determine the quality of a clock network's design are the slew rate, the clock skew, the clock jitter, and the network power consumption. The goal is to maximize/minimize these performance metrics but this usually comes with trade-offs. The relative importance of these different metrics is dependent on the application.

As previously mentioned, the slew rate of a signal is the change in voltage over time. A higher slew rate is desirable since it means that the endpoints in the distribution network switch faster. A high slew rate ensures that the registers can operate reliably, and thus allows for faster clock speeds (which means increased performance). A low slew rate, however, can result in unexpected operation of the receiving circuits and can reduce system performance.

Another desired metric of clock networks is a low clock *skew*. Skew refers to the difference in arrival time of the same clock signal at different endpoints in the distribution network. Ideally, all synchronous elements in the IC should see a coincident edge of the clock. This ensures that circuits trigger at the same time and ensures reliable communication between functional components. However, slight differences in wire length, wire parasitics, and even temperature can result in the

clock arrival times at different endpoints to differ. Figure 1.1 shows the clock skew that occurs when the delay to two different endpoints do not match. The flip-flop labeled B will see a delayed clock edge compared to flip-flop A at a delayed time interval. This results in B's output appearing at its Q port later than flip-flop A's output, as illustrated by the duration between the dotted lines in Figure 1.1. A higher skew value results in decreased system performance as the clock period must be slowed down in order to accommodate the worst-case skew.

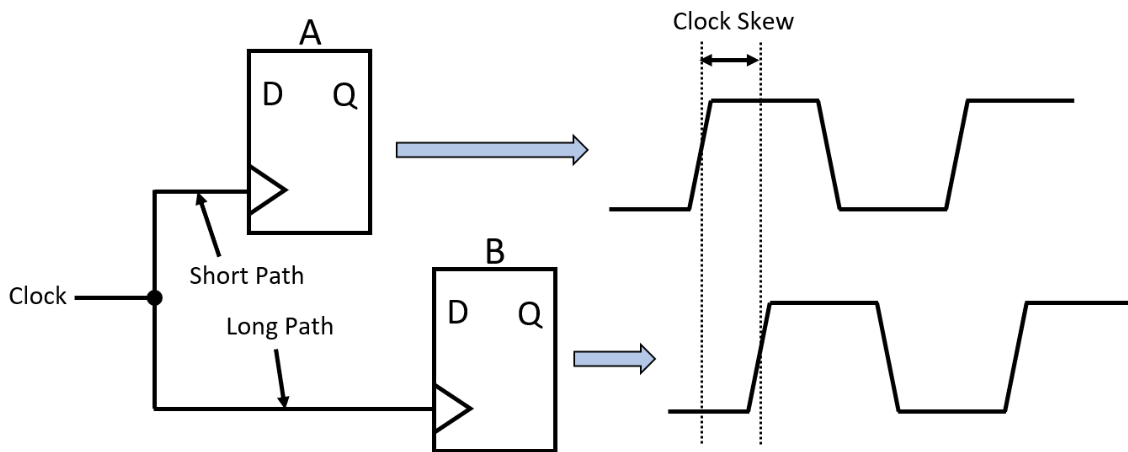

Figure 1.1: Clock distribution network skew example

The duty cycle of the distributed clock is the fraction of the clock period for which the signal is active (high). For example, a 50% duty cycle clock is high for half the period and low for half the period. The desired duty cycle that is presented at the source of the clock distribution scheme should appear at the endpoints. This deviation from the ideal clock period and duty cycle is known as the clock *jitter*. Figure 1.2 shows an ideal clock signal with a 50% duty cycle compared with a clock that has jitter. The dotted lines show the time difference between these clock edges. Jitter, combined with the previously mentioned skew, can result in significant differences in the clock edges seen at the clock network's endpoints. This degrades the quality of synchronization and ultimately leads to decreased performance.

The final major metric used to measure the quality of a clocking scheme is its *power and energy*

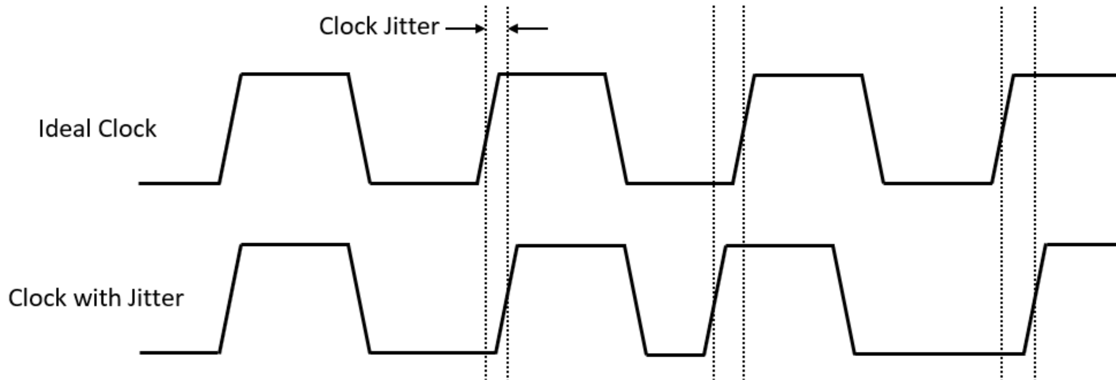

Figure 1.2: Clock distribution network jitter example

consumption. For devices that run off of a battery, the energy consumption is a critical metric. For other systems, power consumption is an important metric. As ICs continue to increase in size, and clock frequencies get faster, the clock distribution network becomes a significant contributor to the overall power budget. A first-order approximation of the dynamic power consumed by a clock signal is represented in Equation 1.1.

$$P_{dyn} = fC_LV_{DD}^2 \tag{1.1}$$

In the above formula, f represents the switching rate (frequency) of the clock, C represents the total load capacitance and V_{DD} represents the supply voltage. The load capacitance is a combination of the capacitance of any gates in the fan-out, as well as capacitive parasitics of the wires that make up the clock distribution network. As expected, with higher performance (higher clock frequency) and larger distribution networks (larger load capacitance) the dynamic power a clock network consumes grows linearly. The power is quadratically related to the supply voltage of the clock drivers. In order to minimize the power consumed, while still meeting the previously mentioned clock metrics, a clock distribution network must strike a balance between its various systems requirements.

1.1.3 Traditional Clock Distribution Schemes

Clock distribution has been a heavily studied area since the development of the first ICs. Many interesting and creative clock distribution networks for both high-performance and low power applications have been proposed. This section discusses many previously proposed and utilized clock distribution networks along with a discussion on their advantages and disadvantages. This will lead to our proposed design and how it can be deployed in an exemplary memory system.

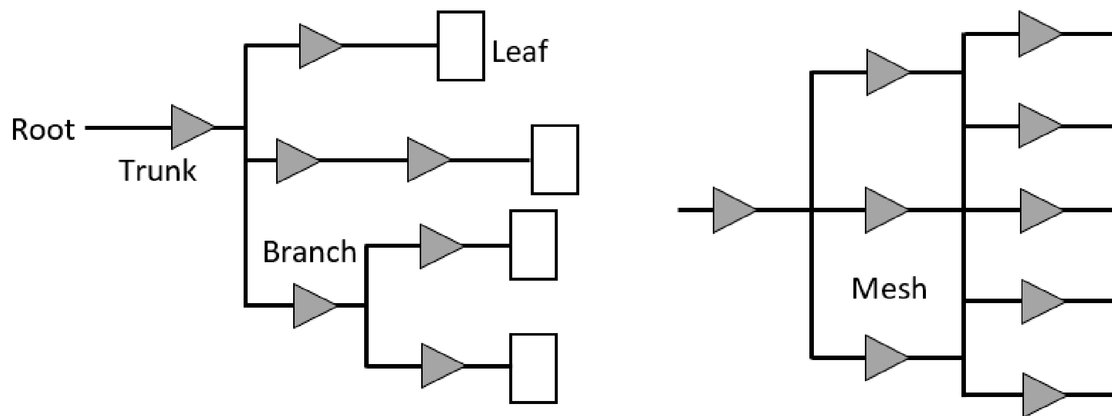

Figure 1.3: Common clock tree distribution schemes

One strategy used for clock distribution is to evenly space buffers from the source to the endpoints to form a tree-like structure. In this tree, the clock source is referred to as the root of the tree with the initial trace from this root being the trunk of the tree. The individual wires that fan-out from the trunk are called the branches with the endpoints being the leaves of the tree. The left side of Figure 1.3 shows this tree-like structure with a series of buffers that drive the clock from any path from the root to a leaf. The tree shown on the left side of Figure 1.3 is often used in ad-hoc, automatically generated clock distribution networks in application-specific integrated circuit (ASIC) designs. The right side of Figure 1.3 shows a mesh based version of a similar clock tree where the branches are shorted to minimize the wire resistance of the tree. This mesh based scheme reduces the resistance by adding wires in parallel, which helps minimize the clock skew

and jitter seen at the leaves [4, 5]. However, the addition of extra wires in the mesh raises the total power consumption of the distribution network.

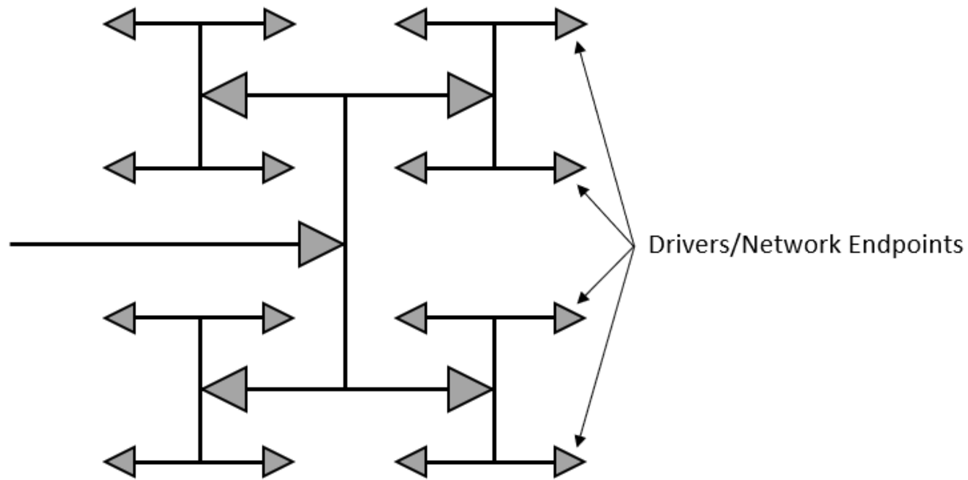

Figure 1.4: H-tree Clock Network

A variation to the previously mentioned tree approach is the “H-tree”. This popular clock topology attempts to match the wire length from the global distribution point (root) to each of the endpoints (leaves) [6, 7]. This is done via structural symmetry, where each path from the root to any leaf of the tree contains identical wire lengths and buffer locations. The symmetric structure of the network, where a pair of bifurcations appear like a letter “H”, gives rise to the H-tree name and is shown in Figure 1.4. Ideally, this network would have identical clock propagation delays to every endpoint and be resilient to PVT variations. Unfortunately, modern IC designs cannot obtain perfectly balanced H-trees due to keep out zones, unbalanced endpoint locations and the growing importance of wire delays, which require multiple intermediate buffers in the H-tree, resulting in susceptibility to PVT variations [8]. Because of these challenges, the H-tree approach typically has trouble maintaining low skew and jitter but performs well when the tree remains small.

Variations of this H-tree topology are also heavily utilized, such as X-Trees and tapered H-trees [9, 10]. A tapered H-tree progressively decreases the width of the wires as the signal propagates

to deeper branches in the tree. By tapering the wires, the tapered H-tree minimizes the reflections of the clock signals at the fan-out points. Tapering wires also reduces the coupling capacitance seen between ground lines and the toggling clock line. This reduction in the coupling capacitance decreases the dynamic power consumption of the clock network. However, tapering the clock signal paths increases the wire resistance which makes the signal characteristics similar to a non-tapered H-tree network [11]. X-trees are similar to H-trees but possess fan-outs of four rather than two [12]. This might fit the available space better for certain ICs and can also be combined with the H-tree structure to create a hybrid approach.

The previously described tree-based schemes rely on an increasing number of branches to distribute the clock to all network endpoints. Distributing the clock with a tree structure may require a substantial number of branches for designs with a lot of endpoints (such as microprocessors). As the tree grows in depth, the skew, jitter and power consumption of the network will increase, resulting in degraded network performance. To overcome these drawbacks, a grid-based structure can be used.

Figure 1.5 shows a two-dimensional grid that distributes the global clock to different local regions on an IC. The dotted lines show a basic X-tree like structure that takes the clock source and distributes it to the locations that are symmetrically selected on the IC. These locations connect to a grid which has fully connected clock traces in both dimensions such that all points within the grid are shorted together. This makes local wiring simple as network endpoints can be directly connected to the global grid structure [13]. By shorting the clock drivers together, the grid distribution network helps minimize the mismatch in path delays to reduce potential skew and jitter [14, 15]. This is accomplished through the balanced load nonuniformities that are a result of the shorted 2D grid structure. The critical characteristics of the grid are the locations of the drivers, wire dimensions and wire spacing. As expected, a dense grid with many parallel wires reduces the network's skew and jitter. However, more toggling wires will increase the dynamic power consumption of the grid-based network, which usually consumes more power than tree-based structures.

Finally, some ICs use a topology with a central spine. This technique is often combined with

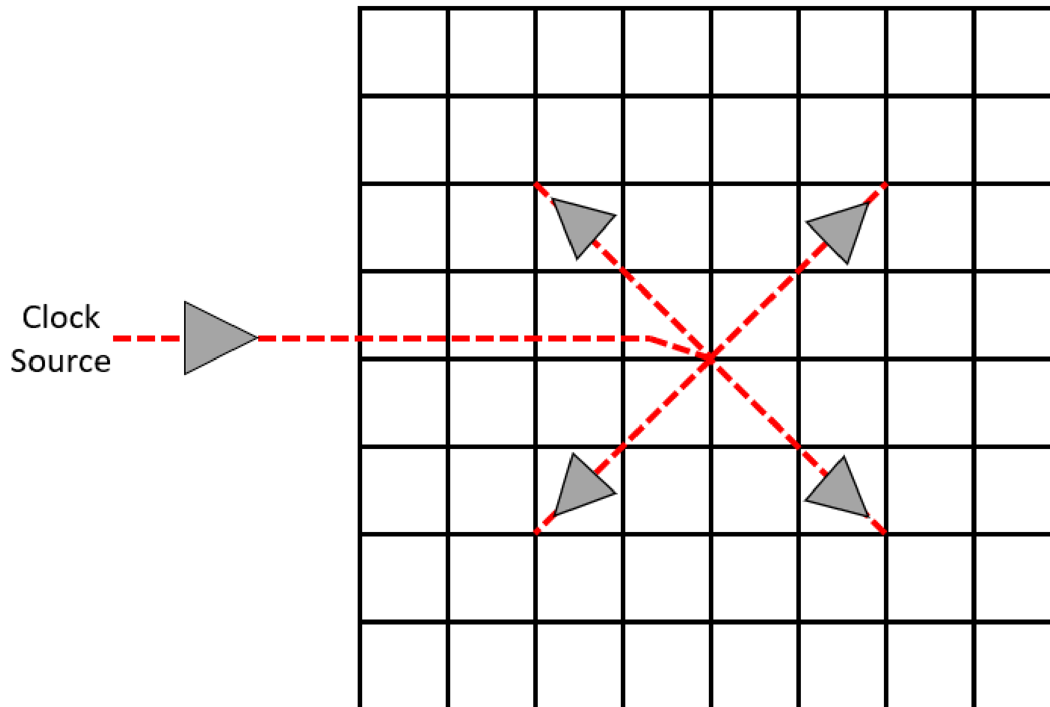

Figure 1.5: Grid Clock Network

other methods where the central spine runs the length of the IC and clock distribution to the end-points is done using a secondary topology that connects to the spine [16]. This method is simple to implement because it breaks up the distribution into two steps, the global spine and the local wiring. The use of multiple spines can help divide the IC into regions, with the secondary topology used for regional clock distribution. A spine tends to minimize wiring area compared to methods like a grid or mesh. By minimizing area, the spine topology often exhibits a lower power consumption than a grid but can experience higher skew for points on the network that lie far from the spine [17]. To combat this, some topologies utilize multiple global spines running in parallel but this increases power consumption.

As shown, these clock distribution networks have advantages and disadvantages that must be balanced to meet the requirements of different applications. In many modern IC designs hybrid approaches are used in order to meet the needs of different functional blocks in the IC. For example, slower operating functional blocks might use a H-tree structure as it can help reduce power

consumption. For higher performance blocks a grid might be favorable since it can lower the clock skew and jitter needed for high clock frequencies.

1.2 3D IC Stacks

Though clock distribution networks have been a well-studied area for decades, new generations of ICs require specialized clock network schemes that do not conform to traditional approaches. One such application that has little prior research in terms of clocking topologies are 3D IC stacks. 3D IC stacks can increase electronic density and performance. Prior systems were limited by 2D IC placement on typical 2D printed circuit boards (PCBs). To overcome this bottleneck, 3D ICs exploit the use of the z-dimension by placing multiple ICs on top of each other to create a "stack". This allows the ICs in the stack to behave as if they were in a single device, thereby achieving higher bandwidth and performance. This is critical in applications such as memory, where bandwidth is becoming an increasing bottleneck to system performance.

Figure 1.6 shows a 3D IC stack with four slave dies, a master die and an interposer. The slave dies contain the desired logic that is often replicated to increase bandwidth and density. The master die is used as a controller to communicate and map commands from external sources to the slave dies. The interposer acts as an electrical interface between the 3D stack and other circuits that might be communicating with it on the PCB [18, 19]. With multiple stacks placed on a common interposer, the communication bandwidth can be increased further. Through-Silicon Vias (TSVs) are wire stubs that are drilled through the ICs to allow communication between the slave dies, the master die and the interposer. This allows the stack to behave as if it were a single high bandwidth device. This high bandwidth is achieved because of the high density and low delay of the TSVs, which allows for larger and faster busses between the dies. Microbumps form the bonding stubs between the die, to allow the TSVs at different layers to be connected together vertically through the 3D stack.

These 3D IC stacks are typically created through a method called die-to-die bonding [18]. This method allows devices to be manufactured separately on each die before stacking occurs. By manufacturing the die individually, each die can be tested independently before the stack is built.

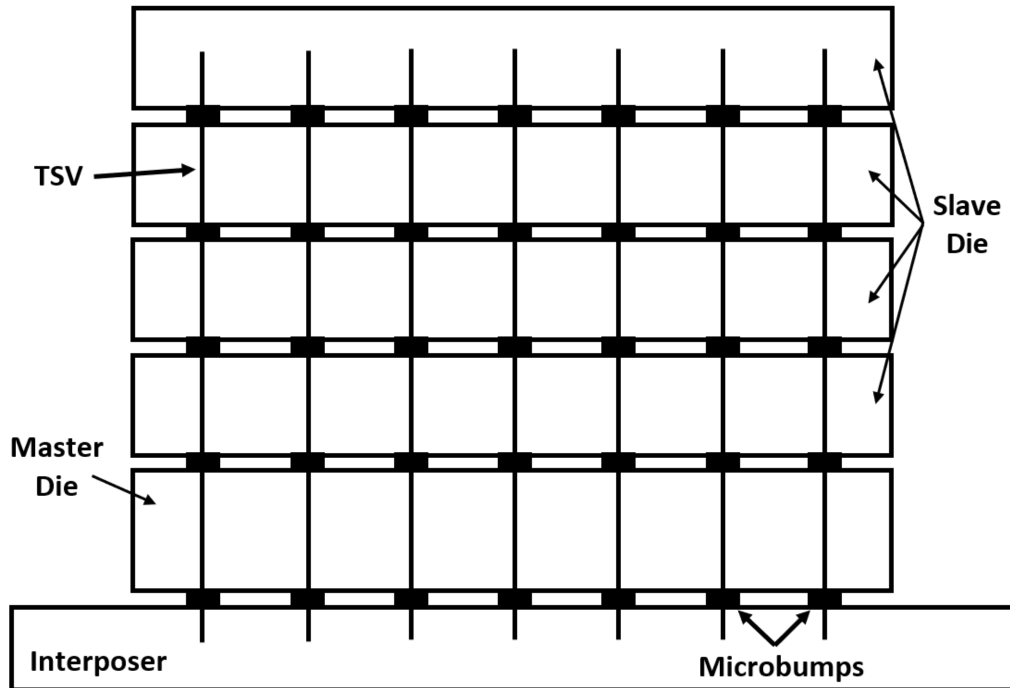

Figure 1.6: Example 3D IC Stack

This increases the stack yield since a single faulty die can be discarded without having to discard the entire stack. The creation of the TSVs can be done before or after the microbump bonding of the die. If TSVs are drilled after bonding, care must be taken to ensure that each TSV aligns with the bonds between ICs in the stack. A single misaligned TSV can result in an unwanted short or open circuit, and cause the entire stack to be discarded. However, TSV drilling after bonding is usually cheaper as the process is only done once for the entire stack rather than for each individual die.

Though 3D IC stacks can yield a significant performance boost and reduced size, power consumption and heat dissipation become a major concern [20]. To ensure proper cooling, devices require package exposure to allow heat to dissipate and prevent the IC from overheating. When additional die are placed on top of each other, the convective mechanism of heat dissipation is replaced by a conductive method of heat dissipation through another die. This makes proper cooling of each die extremely difficult and can cause system failure if heat is not well controlled.

Another challenge with 3D IC stacks is the ability to distribute a clock signal for each die. Traditional clocking schemes are tailored for 2D ICs, and they lack the ability to properly scale in the third dimension using TSVs. In addition, the traditional 2D clock distribution networks are not easily modified to accommodate distribution in three dimensions, and therefore could yield lower quality distribution.

1.3 DRAM History and Background

Current DRAM technologies serve as the workhorse approach for implementing volatile storage in modern computing systems. Non-volatile memories having large memory latencies and caches are too small to hold the memory needed in most modern programs, DRAM memory bridges the gap between non-volatile storage and cache memory, which is vital to system performance. Since DRAM is so important, there has been a lot of research and development into DRAM technology over the decades. This has made DRAM technologies one of the fastest growing aspects of modern computer systems over the past half-century. This section discusses the history of different DRAM technologies and describes the disadvantages of current DRAM architectures.

Fast Page Mode DRAM (FPM DRAM) was one of the first mainstream DRAM designs. FPM DRAM kept the row access strobe (RAS) active while continuously signaling the column access strobe (CAS) to read data from the same row [21]. By keeping the RAS active, the latency needed to read the row is only incurred once for every data segment read from that active row. Despite its simplistic nature, FPM DRAM provided significant performance increase with virtually no added device cost (Figure 1.7).

The next innovation following FPM DRAM was Extended Data Out DRAM (EDO DRAM), which added a group of latches after each column multiplexer. This allowed the DRAM to hold the read data at the output pins while a concurrent column addresses travel across the data bus. EDO DRAM was slightly faster than FPM DRAM because of its ability to hold the state of a row and quickly led to the next DRAM generation known as Burst EDO DRAM (BEDO DRAM).

Similar to EDO DRAM, BEDO DRAM would burst multiple consecutive column addresses worth of data after it received the starting column. This increased the data transfer rate by incurring

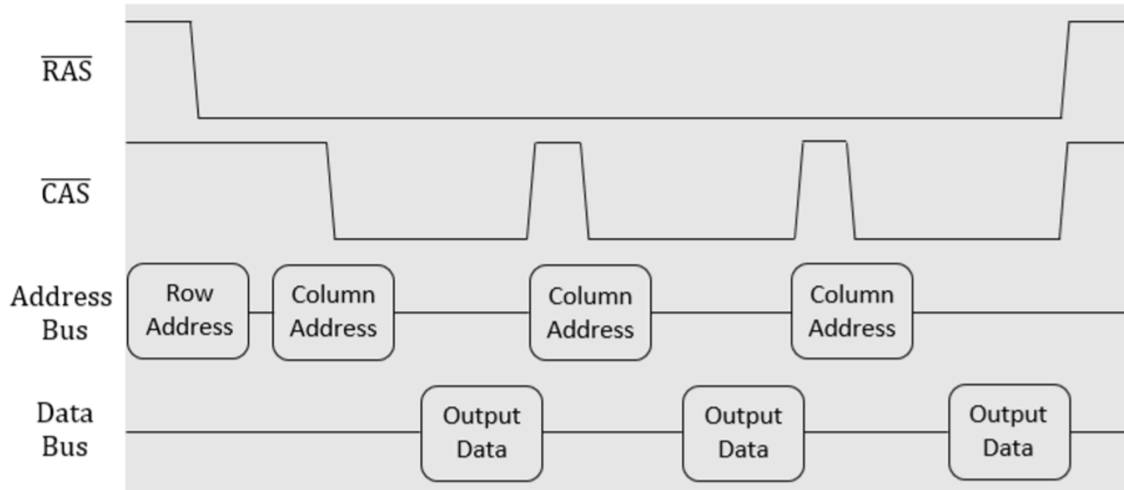

Figure 1.7: FPM DRAM read operation with row kept open

the CAS latency only once before data became available for each strobe of the CAS line. This method relies on the use of spatial locality to increase the overall throughput of a particular process. The bursts, however, were limited, as the data was required to reside on the same row.

The innovations in DRAM that have been discussed so far have been inexpensive to implement but have led to tremendous performance advantages [21]. Such low-cost developments are becoming increasingly rare since the introduction of Synchronous DRAM (SDRAM), which was prevalent in modern computing systems by the late 1990s. Unlike its predecessors, SDRAM used a single-ended clock to synchronize the data being transmitted between the memory module and the dynamic memory controller (DMC). Despite initial SDRAM designs being slower than BEDO DRAM, the introduction of a clock allowed reliable communication with the memory module. SDRAM was the first generation of modern Double Data Rate (DDR) DRAM architectures that currently dominate the market.

DDR SDRAM differs from SDRAM by passing data on both the rising and falling edge of the clock cycle as shown in Figure 1.8. This effectively doubled the data transfer rate without having to double the clock frequency. By utilizing both edges of the clock, DDR is able to maintain low signal integrity requirements for the control, address and data bus that connect the DMC and the

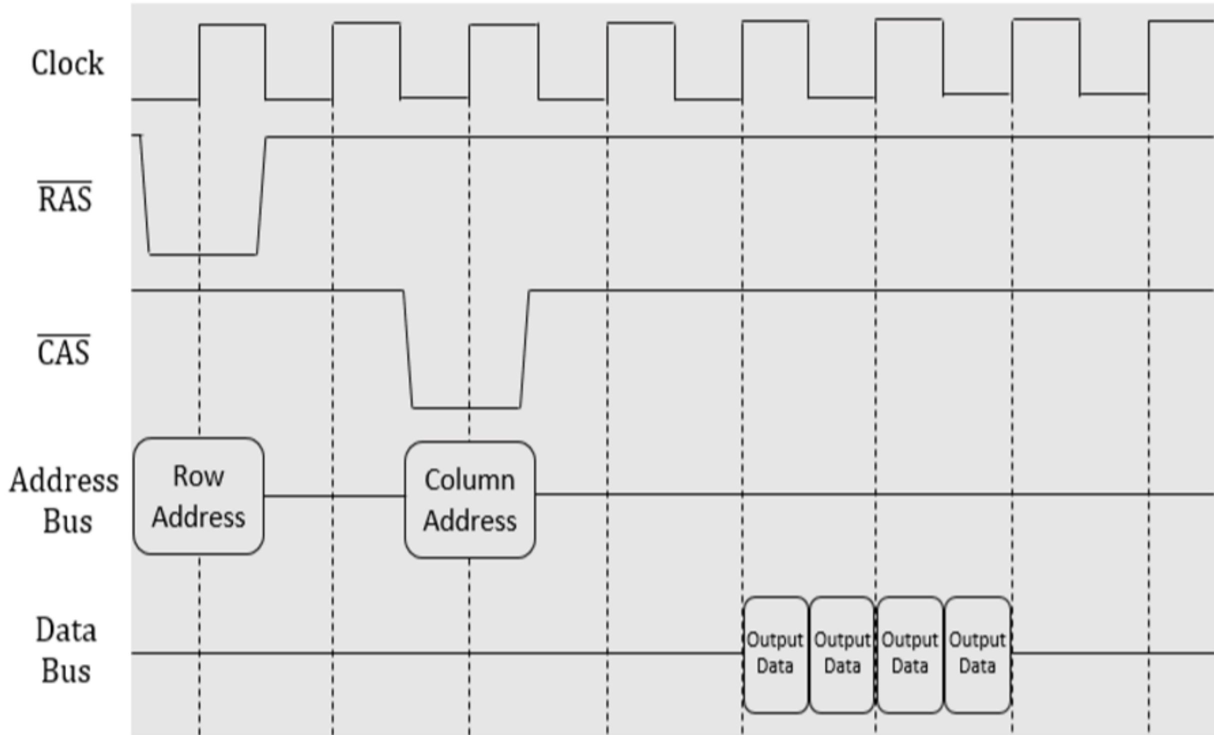

Figure 1.8: DDR SDRAM read operation

memory module. Since the introduction of the first-generation DDR SDRAM in 2000, three more generations of DDR have each shown improvements in the data transfer rate. To achieve these improvements, engineers have created a standard architectural layout with specified timing parameters for communication. By clocking the I/O bus faster, and making the prefetch larger in each generation, engineers have been able to increase memory speeds despite the use of traditionally slow memory chips (MCs). However, despite the increases in transfer rates, the speed of DRAM memory remains significantly slower than modern central processing units (CPUs), with latencies in the low 100s of CPU cycles.

1.3.1 DRAM Terminology

We define some terminology that will assist the reader throughout this thesis. These memory related concepts are heavily used in this chapter and chapter three, and show the advantages of the different DRAM techniques.

- *JEDEC:* The Joint Electron Device Engineering Council (JEDEC) is a joint organization consisting of many industry leaders who create and define new DRAM standards as new generations are developed.

- *DIMM:* Dual In-line Memory Module (DIMM) is a circuit board that consists of DRAM chips that connect to a large data bus with independent electrical pins on each side of the board. These DIMMs plug into a memory slot on a motherboard to allow for modularization.

- *SO-DIMM:* Small Outline Dual In-line Memory Module (SO-DIMM) is a smaller version of a DIMM typically designed for low power mobile devices such as laptops, tablets, and routers.

- *DDR:* Double Data Rate is a computer bus that transfers data on both the positive edge and negative edge of the clock signal.

- *DMC:* Dynamic Memory Controller (DMC) is a hardware component that maps addresses, queues requests and communicates between the CPU and the memory module. The DMC is typically located on the same IC as the CPU.

- *MC:* A Memory Chip (MC) is an integrated circuit that contains memory arrays to store data. DIMMs typically have multiple MCs on the board that connect to form the data bus.

- *Channel:* An independent bus that contains a control portion, address portion and a data portion to communicate memory commands between the DMC and the DIMM.

- *Rank:* A set of DRAM MCs that share a common control bus with the DMC and operate together to provide the total bits required for the data bus.

- *Bank:* A group of memory arrays (usually distributed across multiple MCs) that provide storage of a logical unit so that memory requests can be pipelined to different banks.

- *Sense Amplifier:* A circuit that amplifies small voltage changes on bit lines when reading data, to quickly read the value stored in a memory cell.

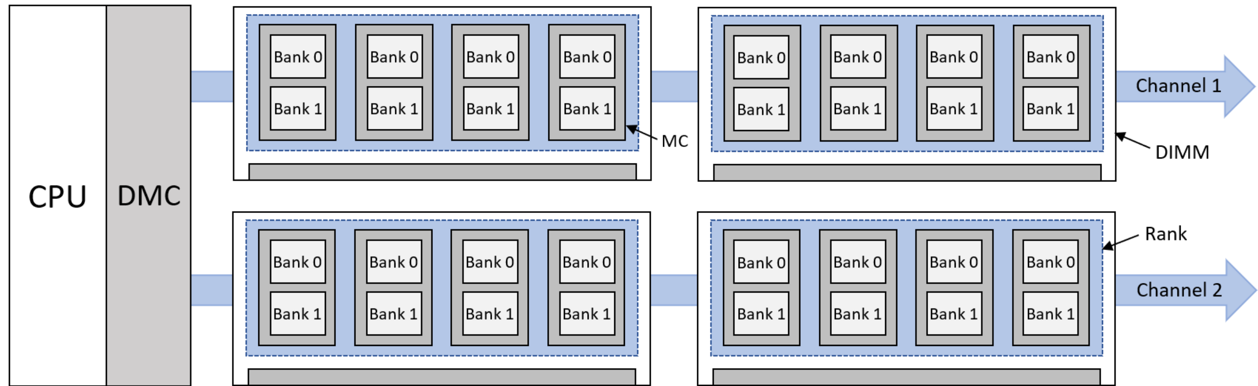

Figure 1.9: Traditional DIMM architecture with multiple channels

- *Refresh:* The process of reading data by the sense amps and writing the read values back to the cell at regular intervals, to prevent loss of information.

1.3.2 Graphics DRAM

Graphics DDR SDRAM (GDDR SDRAM) is a similar memory architecture to DDR but specifically designed for communication with graphics processing units (GPUs). Instead of placing MCs on a dual in-line memory module (DIMM), which is on a separate PCB, MCs are soldered directly to the PCB surrounding a GPU. With a wider data bus than DDR interfaces, GDDR uses a multi-channel architecture to achieve the high bandwidth requirements of graphics applications. Modern GDDR5 DRAM MCs can provide up to 8Gbps per pin. When you place 12 of these GDDR5 MCs are placed around the GPU, a traditional 6-channel (2 MCs per channel) GDDR5 architecture is realized that can achieve up to 384GB/s (using a 64-bit data bus per channel). By utilizing more channels, a wider bus is realized which is able to meet the processing demands of high-end graphics programs. Figure 1.10 shows a modern GDDR5 layout as previously discussed. There is a total of 12 MCs to form 6 independent channels where each MC provides 32-bits of the 64-bit channel data bus.

GDDR5 was a successful development in DRAM technology to meet the higher bandwidth requirements. However, after nearly a decade of use, the need for even higher bandwidth memory has driven the development of what is known as GDDR5X. Standardized by JEDEC in 2016,

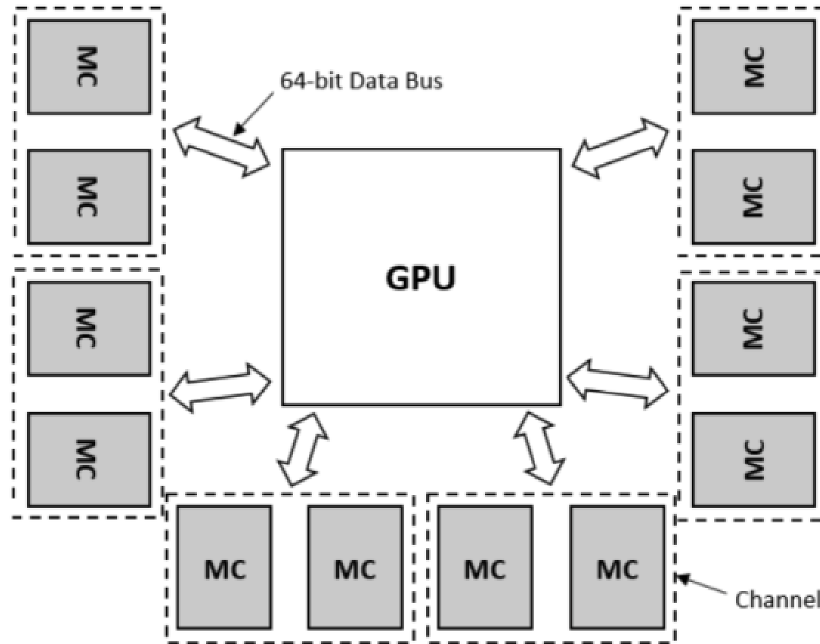

Figure 1.10: GDDR5 top view based on Nvidia's GeForce GTX Titan Black [1]

GDDR5X increased the transfer rate of MCs to up to 14Gbps per pin which is almost double that achievable by GDDR5 [22]. In addition, the prefetch rate was doubled from 8 to 16 by adding more banks in parallel for each MC. The latest GDDR technology to hit the market has been GDDR6 which provides further improvements on its GDDR5 predecessors. GDDR6 improves MCs to 16Gbps per pin while lowering the supply voltage to reduce power consumption [23, 24]. This DRAM technology first started appearing in Nvidia graphics cards in 2018 and is expected to help meet the higher bandwidths required of main memory. It is important to note that despite the increases in transfer rate of the MCs, GDDR5, GDDR5X, and GDDR6 all contain up to 8 channels. The number of channels is limited by the physical space on the PCB that surrounds the graphics processor. A solution to this is to use 3D stacked memory.

1.3.3 Low Power DDR

Low Power DDR (LPDDR) are forms of DDR DRAM designed for mobile systems by targeting lower power consumption, often at the expense of performance. The main difference between LPDDR and traditional DDR technologies is the reduced supply voltage [25]. By lowering sup-

ply voltage, and therefore lowering operating temperature of the MCs, the DRAM requires less frequent refresh commands. In addition to lowering supply voltage, LPDDR typically uses partial refresh of DRAM arrays and deep sleep mode to ensure power is only consumed when necessary [26, 27]. LPDDR has had five generations (LPDDR, LPDDR2, LPDDR3, LPDDR4, LPDDR4X) with a sixth (LPDDR5) currently in development by JEDEC. Similar to traditional DDR technologies, LPDDR has increased the prefetch size for each generation while maintaining a commitment to lower power consumption.

1.3.4 Current Dual In-line Memory Module Issues

1.3.4.1 Heat and Power Consumption

Dual in-line memory modules (DIMMs) are circuits that are designed to seamlessly integrate into CPU boards to provide DRAM-based storage. In fact, they are designed to be simplistic devices because they rely heavily on the DMC on the CPU to control their actions. This is done by the DMC sending a series of requests to the DIMM to carry out. As DIMMs become larger and their hardware becomes more complex, they dissipate more heat and consume more power [28, 29]. With increasing DIMM sizes, the growing use of Buffered DIMMs and faster I/O clocking, DIMMs become large contributors to the total system power consumption and heat dissipation. This will become more problematic as the desire for larger main memory continues and the "Memory Wall" [30, 31] continues to be a bottleneck for system performance.

1.3.4.2 Pin Count

The cost of DRAM cells has continued to decrease since the introduction of DDR SDRAM. This is a result of reductions in minimum feature sizes in the fabrication process, allowing engineers to pack more memory cells into the same MC. However, packaging costs have not decreased at similar rates. This makes the pin count of DRAM devices a large contributor to the total price of the final product [21]. With pin count being a critical factor in final manufacturing costs, architecture trends have tried to minimize the pin count. Reducing pin count also leads to reduced power consumption and heat dissipation. However, reducing pin counts can often lead to reduced

memory bandwidth and therefore requires a careful trade-off study for different applications.

1.3.4.3 Refresh Rate

With DIMMs continuing to grow in size, the amount of memory they can store continues to increase. However, as the name suggests, DRAM is dynamic and therefore needs to be refreshed frequently to prevent data loss. According to JEDEC standards, rows containing valid memory need to be refreshed every $64ms$ in order to prevent data loss. This periodicity decreases to $32ms$ when the DRAM has an operating temperature of over 85^oC [32, 33]. This means that larger memories need to spend more time refreshing rows. With increasing chip density and more rows in a refresh bundle, the time spent refreshing cells becomes a significant problem to memory performance. This will be counterproductive to the advancements made in data transfer rates with future DDR generations. Figure 1.11 shows the proportional relationship between the capacity of the memory and the time spent refreshing cells. The figure shows that as the memory size increases to 32GB, almost a fifth of the time will be used for refreshing stored data. Note that values are doubled when the temperature is above 85^oC [34].

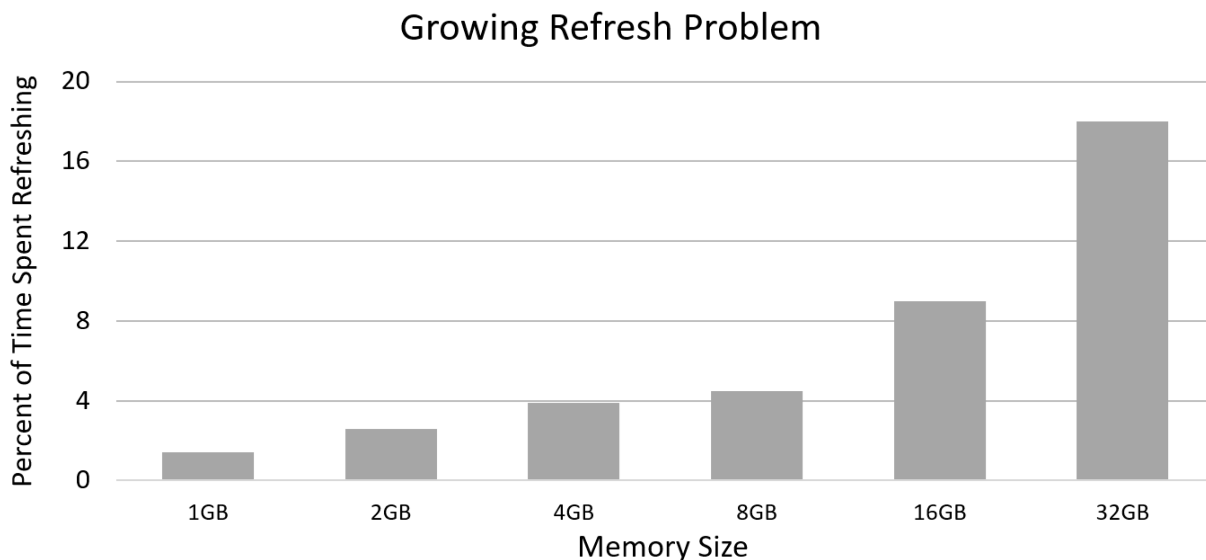

Figure 1.11: Growing refresh rates as DRAM banks increase in size

1.3.4.4 Commodity Status

DIMMs have been marked as a commodity compared to other components within a computer system. As a commodity item, cost plays a major factor when considering different DRAM architectural decisions [21]. This can affect memory speeds as engineers trade off the complexity, power consumption, and the cost while designing the computer system. In order to make a new memory architecture competitive in today's market, the memory designer needs to keep these tradeoffs in mind.

1.3.4.5 Scalability

Since the introduction of the first DDR memory module, engineers have increased transfer speeds by increasing the data rate for a DIMM's I/O bus. As a result, the internal DRAM hardware has several changes with new generations of DDR SDRAM (Table 1.1). To compensate for the slow internal DRAM die, each generation of DDR has increased the prefetch length. This means more data is returned for each column and row address that is transmitted to the DIMM. Though this technique has been successful in the past, larger prefetch architectures will not only be difficult to design in future generations of DRAM but will likely transfer data that goes unused.

DDR Generation	Internal Memory Chip Clock (MHz)	I/O Data Bus Clock (MHz)
DDR	100 - 200	100 - 200
DDR2	100 - 266	200 - 533
DDR3	100 - 266	400 - 1066
DDR4	133 - 266	1066 - 2400

Table 1.1: DDR I/O bus transfer rates

2. SCALABLE AND SYNCHRONOUS 3D CLOCK DISTRIBUTION*

2.1 Introduction

Ring-based Resonant Standing Wave Oscillators (RRSWOs) have been shown to be an effective technique to perform low power clock generation *and* distribution for high-speed clocks [2, 35, 36]. The resulting clock signals can achieve very high frequencies and exhibit low skew. Utilizing through-silicon-vias (TSVs), an RRSWO can form a 3D ring that distributes this high-speed clock to all the die in a 3D IC stack. In this chapter, we present an RRSWO-based technique to generate and distribute a high frequency, low skew clock in a 3D IC. We also present a scheme to synchronize two such 3D RRSWOs without the use of a Phase Locked Loop (PLL). This will reduce manufacturing cost, provide effective synchronization between the 3D ICs and create a scalable system with low clock skew. Finally, we present the design of a robust bootstrap circuit for 3D RRSWOs. We validate our ideas using circuit simulations, including Monte Carlo simulations to quantify the resilience of our approach to process and voltage variations.

2.2 Terminology

Here we define some terminology that will be utilized throughout this chapter. These concepts relate to our clock generation and distribution approach and help the reader identify the advantages of our technique.

- *Standing Wave:* A stationary wave whose phase at any point in space is constant.
- *Mobius Crossing:* A swapping of two parallel wires to create a closed and infinite signal path. This can be used to create a standing wave as the signal travels between the logical low and logical high wires.
- *Virtual Crossing:* A location opposite the Mobius Crossing where the voltage magnitude

*Part of the data reported in this chapter has been updated with permission from "Synchronization of Ring-based Resonant Standing Wave Oscillators for 3D Clocking Applications" by Andrew Douglass and Sunil P. Khatri, 2018. International Conference on Computer Design (ICCD), pp. 318-325, Copyright by IEEE.

remains constant, at approximately half the rail voltage.

- *RRSWO:* A Ring-based Resonant Standing Wave Oscillator is a type of resonant circuit used to generate and distribute a fast, oscillating signal. It utilizes a set of long parallel wires with an odd number of inverter pairs connected across the wire pair to provide negative resistance and sustain the oscillation. With a Mobius Crossing used near one inverter pair in the ring, the generated clock signal throughout the ring will be sinusoidal and possess the same phase at all points along the ring.

- *Q-factor:* A dimensionless parameter which is the ratio of the center frequency of an oscillator to its bandwidth. An oscillator with a high Q-factor is underdamped, indicating that the oscillator has high amplitudes and a small frequency range. In contrast, a low Q-factor means that the oscillator has a large frequency range and smaller amplitudes at their resonant frequency. This makes oscillators with a low Q-factor useful in circuits that require a wider operating frequency range.

- *PLL:* A Phase-Locked Loop is a feedback control system that outputs a clock signal which has a phase and frequency that is identical to the phase and frequency of the control system's input signal. These systems are heavily used in modern ICs as they allow for synchronization of internal clocks in the IC with an external reference clock.

- *TSV:* Through-Silicon Vias are wire stubs that create signal paths between different dies in a 3D stack. Because they connect densely packed 3D dies together, they are less expensive and faster than traditional pin-based connections, such as flip chips and wire bonds.

2.3 Previous Work

The authors in [37] presented an analysis of a rotary traveling-wave oscillator that can operate in the gigahertz range. Though effective in producing a clock in the three to eight gigahertz range, the points along the rotary ring have different phases. As the number of inverters in the rotary oscillator increases, the capacitance increases, thus increasing the period of the square wave clock that

is produced. The authors in [38] also presented a traveling-wave oscillator that utilizes a plurality of inverter pairs throughout the rotary ring. Their oscillator was analyzed at lower frequencies (2-3GHz), and they fabricated a chip that shows the resilience of such a clock distribution technique. Again, the phase change makes it difficult to use in traditional synchronous design. Also, larger inverter pairs cause an increase in total power consumption.

To implement a clock PLL, [36] and [39] utilized a high-frequency standing wave oscillator (SWO). In [39], the authors used a plurality of coupled oscillators. To maintain the oscillation, a cross-coupled NMOS pair is used with a PMOS diode to set the common mode voltage. In [36], the authors present an RRSWO based PLL, with fine frequency adjustment performed by changing the body bias of the PMOS devices in the inverter, and coarse frequency adjustment performed by changing the inductance of the ring. Both techniques present effective means of synchronizing two clocks but require higher power consumption and more circuitry to ensure the clocks remain in phase.

The authors in [40] and [41] have also implemented a PLL. However, they utilized an agile voltage-controlled oscillator (VCO) and an automatically adjusting nominal frequency VCO, instead of an SWO, to achieve frequencies around 1GHz and 10GHz respectively. In [42] the authors also implement a PLL that can operate in the gigahertz range with a particular emphasis on its settling time. They emphasize the charge pump current and the parameters of the loop filter to achieve a locking time of $1.5\,\mu s$. These techniques allow an efficient search for a nominal frequency but suffer from higher energy consumption.

Energy efficiency of synchronization is also important, as shown by the authors in [43] and [44]. They are able to achieve a total PLL power consumption of 4.6mW and 11.4mW respectively. These approaches involve prescient loop filters and a feed-forward compensation technique to minimize power consumption. However, the maximum achievable frequency using these techniques are lower than 3GHz.

For clock distribution schemes like mesh, H-tree, and traveling wave oscillators, there is an inherent skew that results from imperfections in fabrication. There have been many proposed tech-

niques to minimize the skew between endpoints. In [45] the authors propose a skew-compensation circuit. The circuit is designed to detect the propagation delay to an endpoint and compensate for potential differences in endpoint delays. However, the frequency of oscillation was limited to a few hundred megahertz and the power dissipation was significantly higher given the complex circuitry involved.

Another skew compensation approach is [46], in which the skew of each endpoint of the H-tree is individually measured and adjusted to be a fixed value by using binary weighted capacitors at each endpoint. To combat high power consumption, other techniques such as those presented in [47] try to use currents instead of voltages to distribute a global clock signal. They utilize a high-performance current mode pulsed flip flop that uses a driver and a receiver on either end of the clock transmission line. Though they are able to reduce power consumption by up to 42%, propagation delay can increase up to 35%, which can severely limit the speed of the distributed clock.

In [48] the authors implement a clock generating device for miniature implantable medical devices. This is done using an all-digital frequency locked loop which takes advantage of the inductive power supply. The clock generation is done without a crystal, but the oscillation frequency is only around 100MHz. This is significantly slower than the tens of gigahertz oscillation frequency that can be achieved with an RRSWO.

As opposed to the previous approaches, our goal is to provide a resilient technique to synchronize two independent clocks without the use of a PLL. This is done using multiple RRSWOs which allow for fast, low skew oscillation while minimizing power consumption. In the following sections, we will introduce our proposed technique and present the result of simulations to show that our approach can improve on previously proposed clock distribution techniques.

2.3.1 Resonant Standing Wave Oscillators

Several research efforts over the years have tried to develop clock generation and distribution schemes that provide a fast and low skew clock for digital systems. Typically, these approaches tackle clock generation and distribution separately. The RRSWO presented in [2] and [35] combine

low power clock generation *and* distribution approaches for frequencies in the tens of gigahertz range. Additionally, the Q-factor of such ring-based oscillators is high enough to provide low jitter and better PVT tolerance, while also providing the ability to modify the oscillation frequency using PLLs [36, 46]. Despite the advantages of an RRSWO, they have been restricted so far to synchronizing logic on a *single* IC.

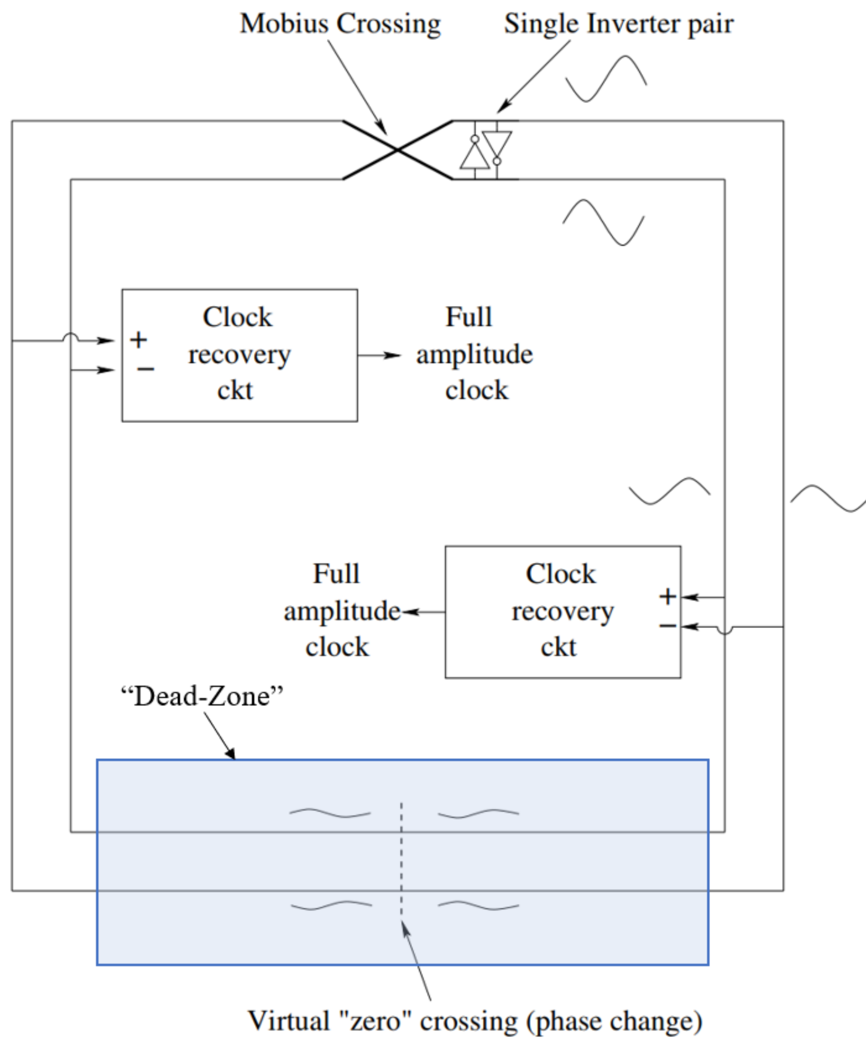

Figure 2.1: Resonant clock ring with recovery circuits [2]

Figure 2.1 shows an RRSWO, which consists of a pair of wires arranged in a ring, with a Mobius Crossing along with a single cross-coupled inverter pair at the Mobius Crossing, to provide

the negative resistance required to sustain oscillation. Waveforms along the ring are sinusoidal with the amplitude diminishing as one approaches the virtual “zero" location. A full amplitude square wave output is extracted from the resonant ring using a series of differential amplifiers which act as clock recovery circuits. The region near the virtual “zero" location, however, is considered the “dead-zone" where the sinusoidal amplitude is too small to reliably recover a full square wave output. As a result, differential amplifiers are not placed in this “dead-zone" region. The differential amplifiers use a direct current (DC) bias transistor to control the tail current, as shown in Figure 2.2. A single stage differential to single-ended output is used with a current mirror active load. This single-ended output is then driven by two inverters in an increasing size inverter chain. This allows the extracted clock output to drive the capacitive load in an appropriately sized region of the IC.

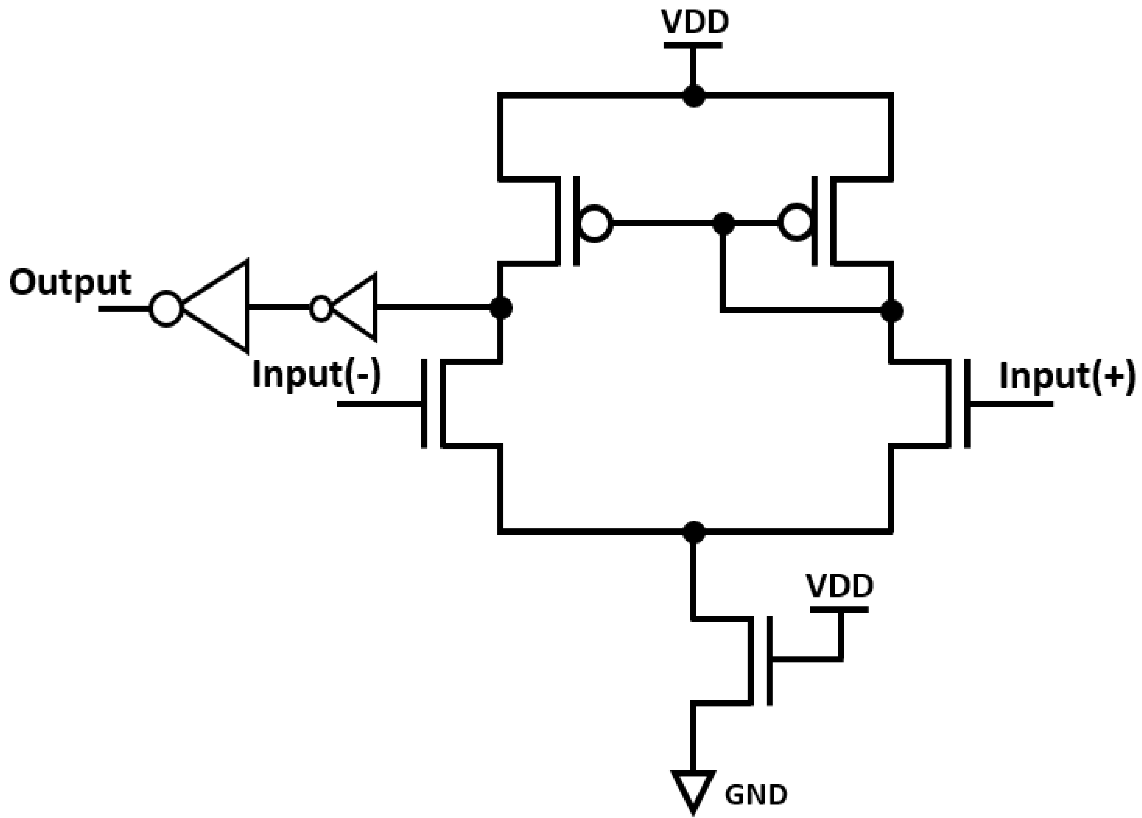

Figure 2.2: Differential amplifier clock extraction circuit

Despite the high frequencies of oscillation, such a ring, as shown in Figure 2.1, has a limited perimeter. Assume that the ring wire pair has a total inductance L and a total capacitance C. This would result in an oscillation frequency as shown in Equation 2.1 below.

$$f = \frac{1}{2\pi}\sqrt{\frac{1}{LC}} \tag{2.1}$$

For higher frequencies (above $10GHz$), the perimeter of the ring (P) needs to be in the $1mm$ to $3mm$ range. This is extremely small compared to typical die sizes, which may be $1cm$ by $1cm$ or larger. This means that the resonant ring is unable to distribute a clock signal across the die area effectively. One alternative would be to increase the perimeter of the ring, but this would decrease the frequency of oscillation since it would cause both L and C to increase.

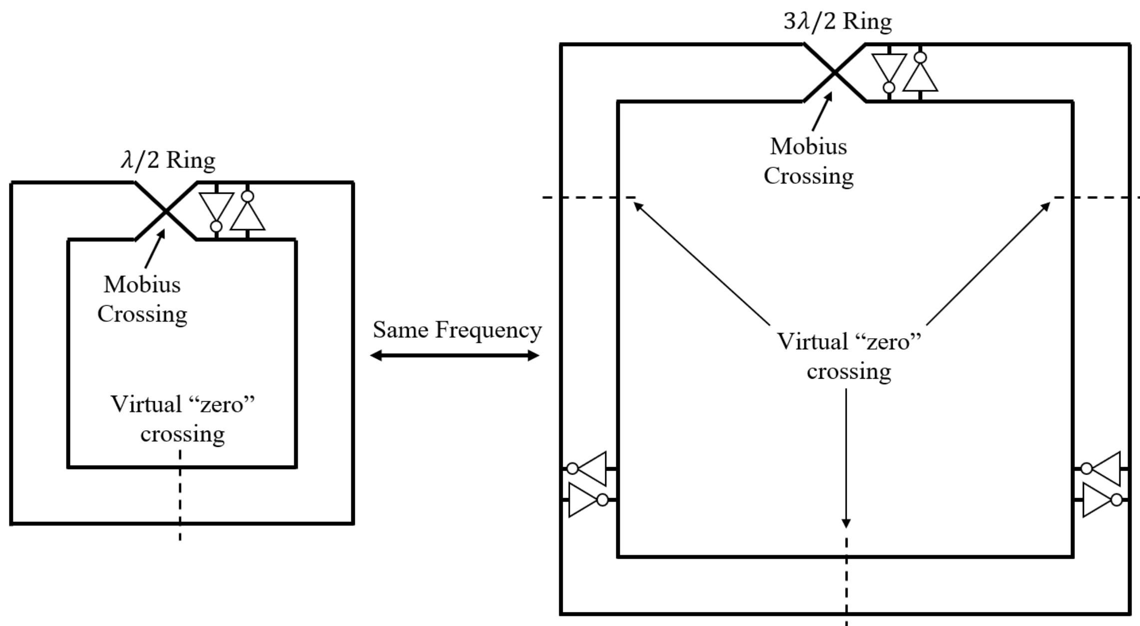

Figure 2.3: Increasing the perimeter of the RRSWO

To increase the perimeter of the ring, without decreasing the oscillation frequency, an odd number of equally spaced inverter pairs can be placed in between the clock wires [2]. Figure 2.3 shows the traditional RRSWO with a single inverter pair with perimeter P (left), alongside

an RRSWO with three times the perimeter (right) and the same oscillation frequency. The larger ring (right), uses three inverter pairs with an equal distance P between each pair. The resulting configuration oscillates at the same frequency as the single inverter pair (left). In general, an RRSWO with k inverter pairs and a total perimeter kP will oscillate at the same frequency as the RRSWO with one inverter pair and a perimeter P (for any odd value of k). Using this approach, the total perimeter of an RRSWO can be increased to “cover" the entire IC. In this way, the RRSWO is able to generate *and* distribute a high frequency, low power resonant clock with very low skew.

The work in this chapter describes a novel circuit to bootstrap the RRSWO. Additionally, in the context of the use of the RRSWO structure in 3D IC stacks, by utilizing a redistribution layer (RDL) stub between two RRSWOs, we demonstrate how we can synchronize two clocks in the tens of gigahertz range with a fast settling time. The use of this RDL stub prevents the need for a PLL. The key contributions presented in this chapter include:

- Presenting the design of a novel bootstrap circuit which allows the RRSWO pair to oscillate from a power-up condition.
- Presenting an RRSWO design which is specifically tailored for 3D IC designs.
- Showing that two separate 3D IC stacks can be synchronized by injection locking, using a wire RDL stub. The resulting system oscillates with minimal skew between the two 3D IC stacks.
- Showing that the above designs are robust to voltage and process variations by means of VDD and Monte-Carlo simulations.
- Showing that our proposed design avoids the use of a PLL, which can be expensive, consume a lot of power and is difficult to operate at the frequencies under consideration.

2.4 Our Approach

The two wires in an RRSWO tend to initialize into an intermediate, non-rail state when powered up. This is because the ring wires are highly capacitive in nature, which causes the wires to

reach a static mid-rail stable equilibrium state (which is approximately VDD/2). Prior to our research, it was assumed that the RRSWO could begin oscillation through the use of the cross talk between the wire pair in the ring. This would require a transition on a wire to create a large enough coupling effect on the mid-rail equilibrium state of the ring. However, upon further simulations, we discovered that the capacitance of any given ring is too high for inter-wire cross talk to cause a large enough disturbance to start oscillation. Therefore, we developed a bootstrap circuit to initialize oscillation for any given RRSWO configuration.

2.4.1 Bootstrap Circuit

The circuit of Figure 2.4 forces the ring to oscillate by asserting and then de-asserting the reset signal (rst). Every inverter pair in the RRSWO is designed using the circuit of Figure 2.4. Every inverter pair will receive the same set of signals and operate in the same manner.

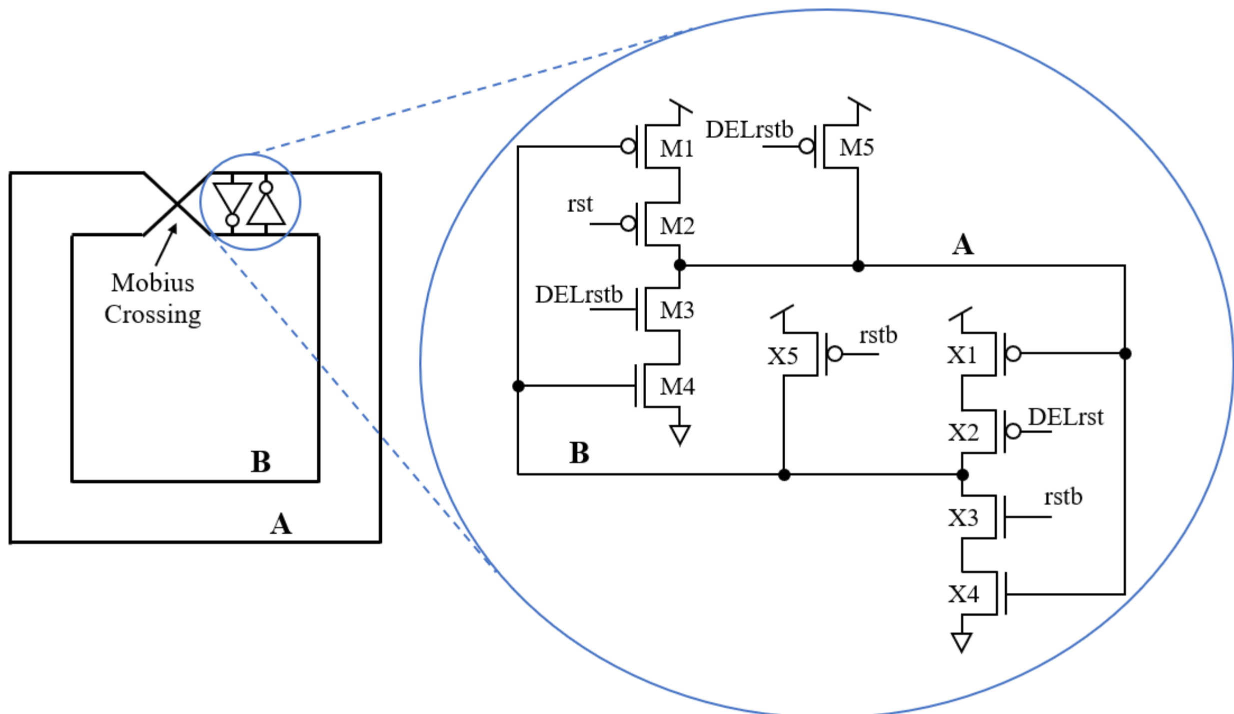

Figure 2.4: Bootstrap reset circuit

When the reset is asserted (rst is driven high and rstb is driven low), both the ring wires are driven into a high state, due to transistors X5 and M5. Note that DELrst and DELrstb are delayed versions of rst and rstb respectively. The transistors M2, M3, X2, and X3 are turned off in this condition. When the rst signal is de-asserted, the upper ring wire (A) remains high longer, since M5 is driven by a delayed rstb signal (DELrstb). This delayed signal is expected to be less than half of the expected clock period of the given ring configuration. Additionally, the lower ring wire (B) is driven low since X4 and X3 both turn on after rst is de-asserted. This gives the upper ring wire (A) and the lower ring wire (B) different logical values which will begin the oscillation of the RRSWO. By designing the delay between the delayed and regular rst (and rstb) signals, a sufficiently large voltage separation can be ensured between the two ring wires, forcing them to enter the oscillatory state.

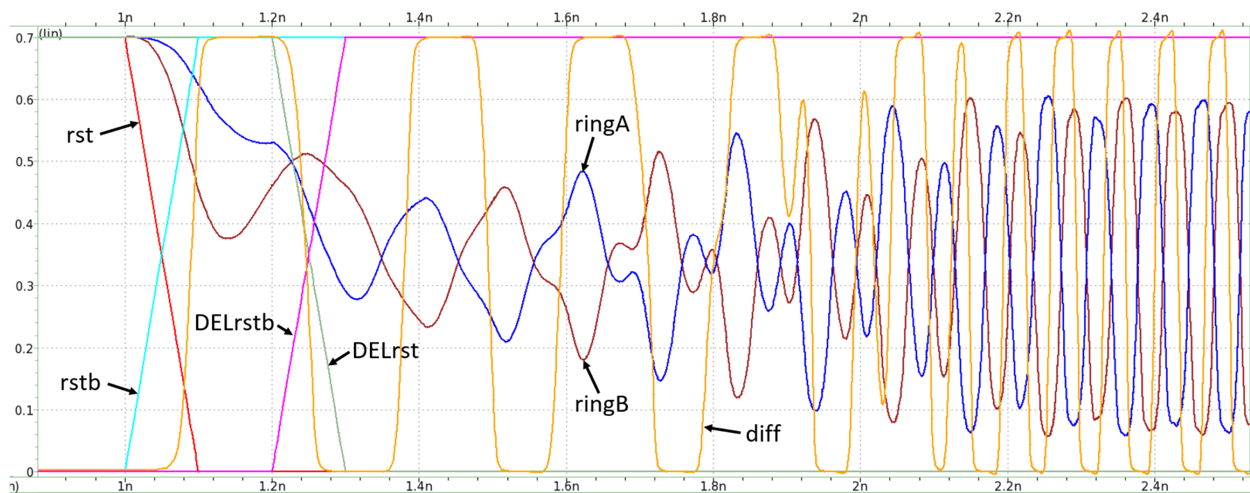

Figure 2.5: Bootstrap reset waveform for single 2D RRSWO

Figure 2.5 shows the waveform for a single 2D RRSWO utilizing the bootstrap circuit for the cross-coupled inverter pairs. The rst signal goes low (rstb goes high) starting at one nanosecond which begins the oscillation process. The ringA and ringB signals represent the clock wires of the RRSWO. These signals start in a high state because they are pulled up by transistors M5 and

X5, as was shown in Figure 2.4. Once rst is de-asserted (rstb asserted), ringA and ringB begin to oscillate. DELrst begins to go low (DELrstb goes high) at 1.2 nanoseconds which turns the cross-coupled inverter pair on. This allows the bootstrap circuit to sustain the RRSWO oscillation. The signal labeled diff represents the extracted full amplitude square wave clock after signals ringA and ringB pass through the clock recovery circuit. By 2.4 nanoseconds (1.2 nanoseconds since rst was de-asserted), the RRSWO enters a stable oscillatory state.

2.4.2 Standalone 3D RRSWO

Figure 2.6 shows a 2D RRSWO on the left that matches the design presented on the right hand side of Figure 2.3. The gray lines that intersect the various shapes symbolically represent the pair of clock wires of the RRSWO. The squares represent the inverter pair locations where the Mobius Crossing can be placed adjacent to any inverter pair. The circles show example locations of the clock recovery circuits (differential amplifiers) that are designed to extract a full amplitude square wave clock from the sinusoidal signal that the RRSWO produces. The dotted lines show the locations of the virtual “zero” crossings where they are exactly halfway between each set of inverter pairs. As previously discussed, the clock recovery circuits are not placed near these virtual “zero” crossings as the amplitude of the sinusoidal wave is not large enough to extract a full amplitude clock.

Using this notation, we can rearrange the 2D RRSWO presented on the left hand side of Figure 2.6 to yield the serpentine structure presented on the right-hand side of the figure. This cross-sectional view of the 3D IC shows the inverter pairs residing on the bottom (master) die. The resonant ring is comprised of TSVs, which are shown by the vertical gray wires on the right side of Figure 2.6. These TSVs are connected through each die by microbumps [49]. The circles show the same clock recovery circuits (differential amplifiers) from the left hand side of the figure that produces the square wave output clock. Traces are extended at the top of the stack to accommodate the virtual “zero" locations. These traces are 2D traces on the top die of the stack. Note that the clock extraction points are evenly distributed throughout the slave dies.

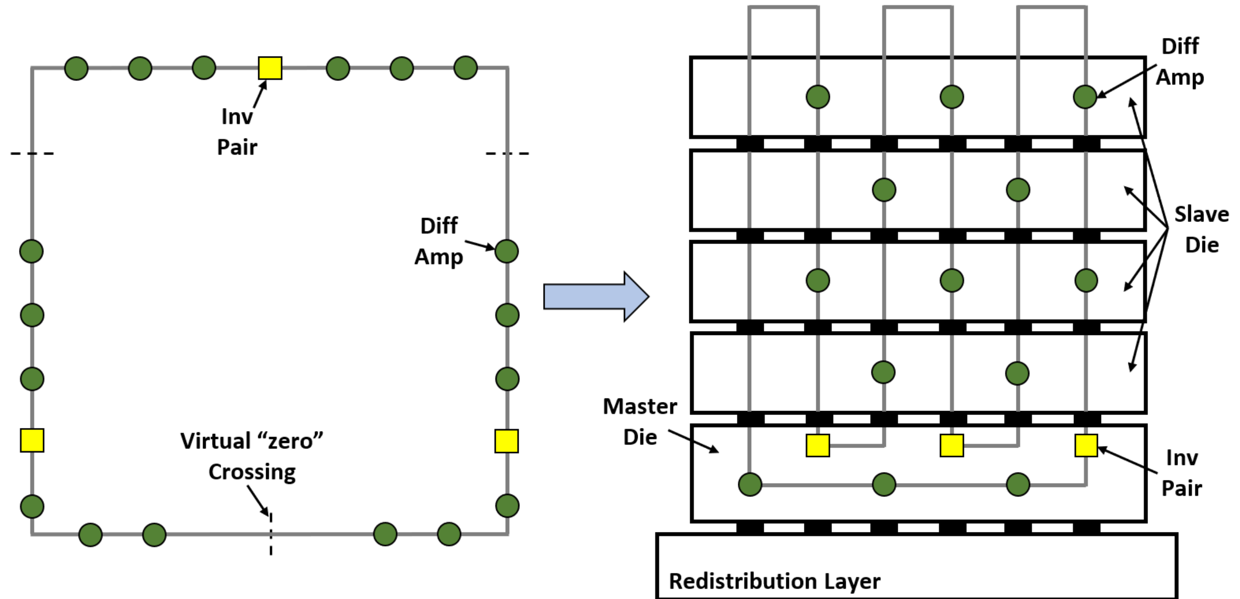

Figure 2.6: 2D to 3D adaptation for an RRSWO

2.4.2.1 *Experimental Results*

The RRSWO bootstrap and synchronization experiments were conducted using the HSPICE circuit simulator [50]. In order to obtain the resistive, inductive and capacitive (RLC) characteristics of the ring, we used Raphael [51] with the clock wires modeled as cylindrical TSVs. These clock wire TSVs had a $2\,\mu\mathrm{m}$ diameter with $4\,\mu\mathrm{m}$ spacing between the surfaces of the clock TSVs as shown in Figure 2.7. The spacing between these clock TSVs and the ground plane encasing them was set to $5\,\mu\mathrm{m}$. Using these parasitics, each section of the ring (the section between two inverter pairs) was modeled using 200 T-sections. Every section is a segment represented by a $2200\,\mu\mathrm{m}$ long pair of TSVs. A $16\,\mathrm{nm}$ predictive technology [52], with VDD at $0.7V$, was used for simulations in HSPICE. In total, each segment can support 200 extraction points, since we utilize 200 T-sections per perimeter segment. However, because of the virtual “zero” location, only 133 of these extraction points are used to extract a full amplitude clock signal. Therefore, these 133 points are the only ones that possess the clock recovery circuits (differential amplifiers).

Figure 2.8 shows the notation used throughout the remaining figures. The bars that are not

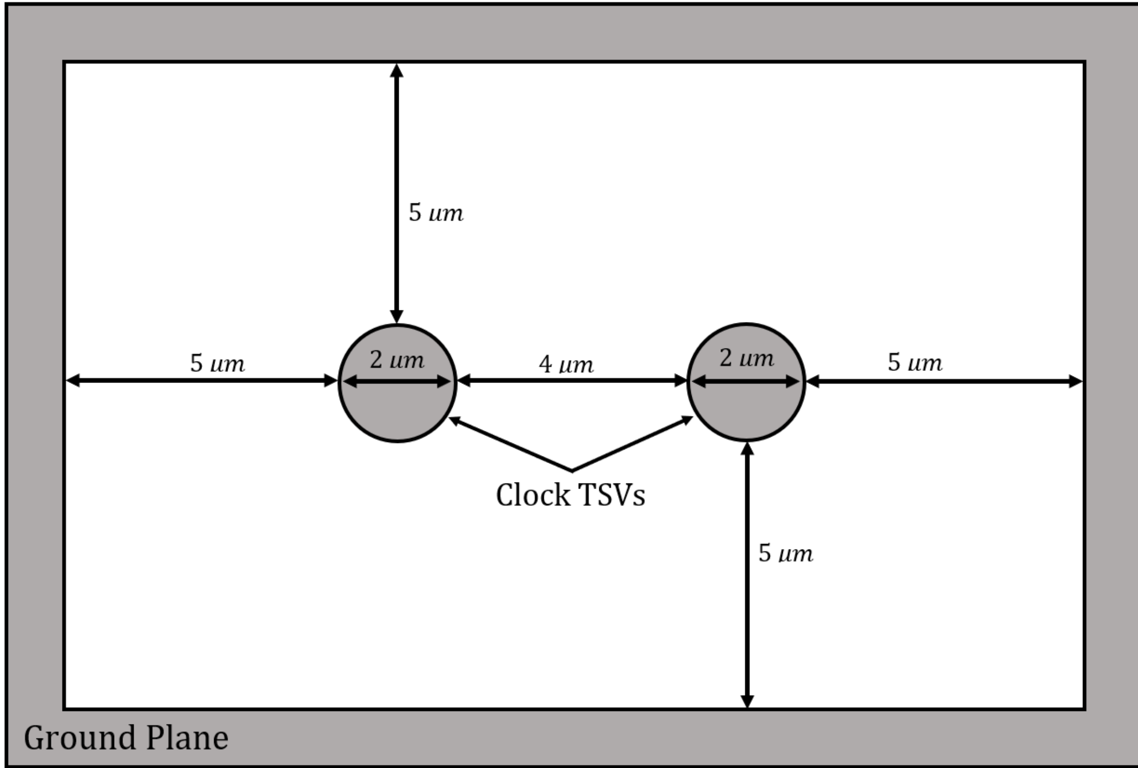

Figure 2.7: Clock TSV cross sectional view

outlined show the average value with the maximum and minimum indicated by the black range markers (left). The right side shows the power bar (which is outlined by the red dotted line) and the current bar (which is outlined by the black bar). These values correspond to the power axis and current axis respectively.

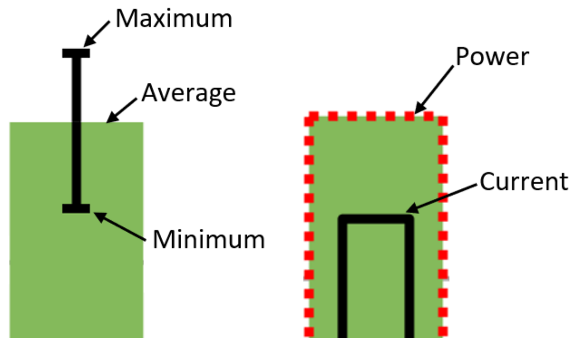

Figure 2.8: Bar chart legend

Figure 2.9 shows the standing wave that can be observed in steady state for a RRSWO. The configuration to obtain this waveform utilized a ring that was $6600\,\mu\mathrm{m}$ in length with three evenly spaced inverter pairs. The full amplitude standing waves are those points closer to the inverter pair while those that do not oscillate are close to the virtual "zero" location. Figure 2.10 shows the output of the clock recovery circuits. This represents the full amplitude clock that can be used to drive synchronous logic. The waveform shows all 399 points that are able to extract a full amplitude clock.

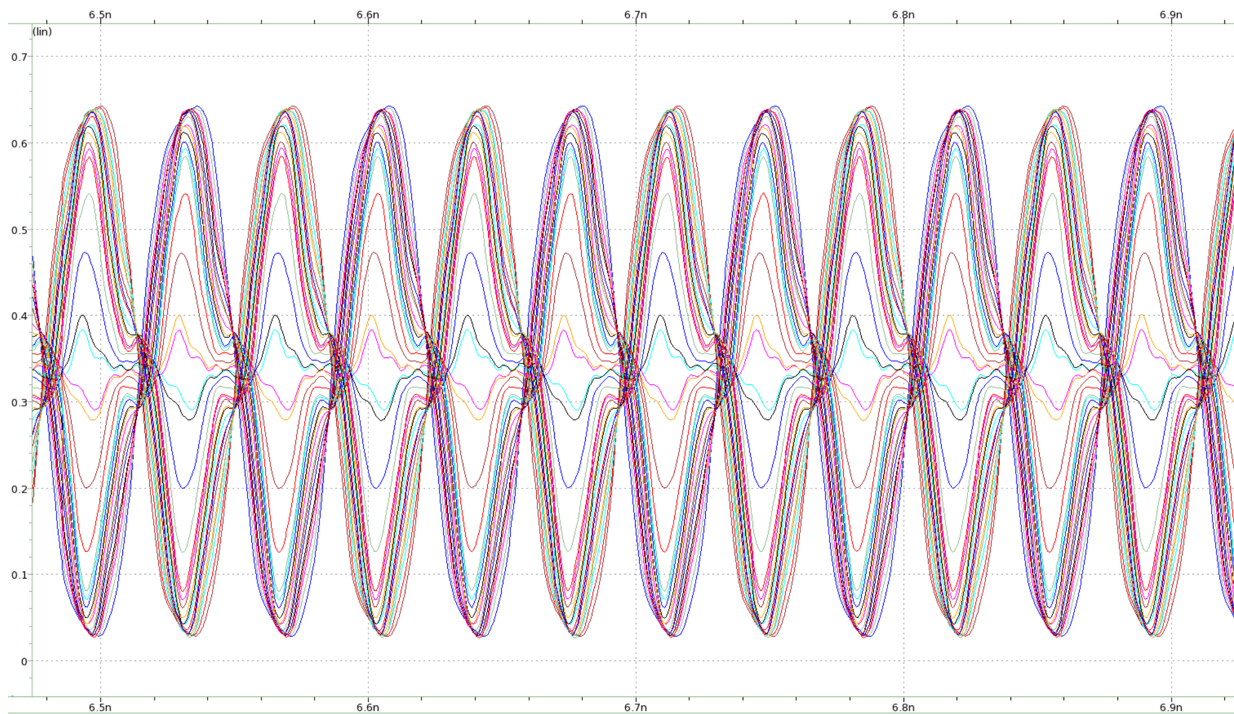

Figure 2.9: Overlaid standing waveform for example RRSWO

Figure 2.11 shows the period of a 3D RRSWO with the size of the inverter pair varied. The values shown represent steady-state behavior, that is after the ring comes out of reset. Three experiments were simulated with each experiment having a different number of segments (and ring length). As the number of segments increased (1, 3 and 9), the total ring length also increased ($2200\,\mu\mathrm{m}$, $6600\,\mu\mathrm{m}$ and $19\,800\,\mu\mathrm{m}$) so that every segment remained $2200\,\mu\mathrm{m}$ in length. As ex-

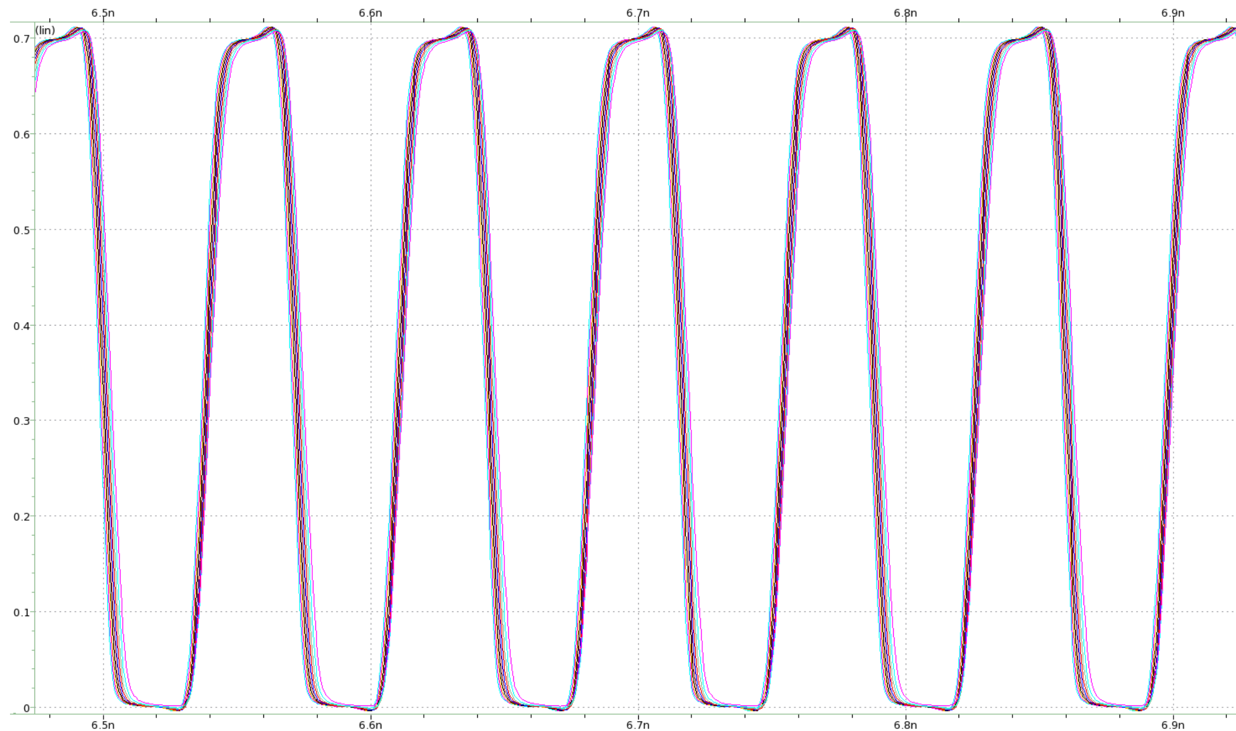

Figure 2.10: Overlaid extracted clock waveform for example RRSWO

pected, from Figure 2.3, the oscillation period of the RRSWO remained relatively constant as the number of segments increased. Note as the inverter size increased the clock period slowly increases, since the total ring capacitance increases (see Equation 2.1). However, there was minimal variation in the clock period as we moved to different points throughout the ring, as shown by the range marker bars. For all configurations, the period remained within $\pm 0.4\%$ of the average. This shows that the RRSWO is able to distribute a clock with minimal jitter to all its endpoints.

Figure 2.12 shows the skew of a 3D RRSWO with the same configurations. The skew values represent the intra-ring skew as we move from the inverter pair location towards the virtual "zero" crossing. As expected, the skew values got larger as we moved farther away from the inverter pair location but remained well-controlled. The average skew remained within 2.8% of the average period while the maximum skew remained within 5.2% of the average period. This shows the ability for the RRSWO to generate and distribute a fast, low-skew clock to large areas of an IC.

Finally, Figure 2.13 shows the power consumption of a 3D RRSWO as we varied the number of

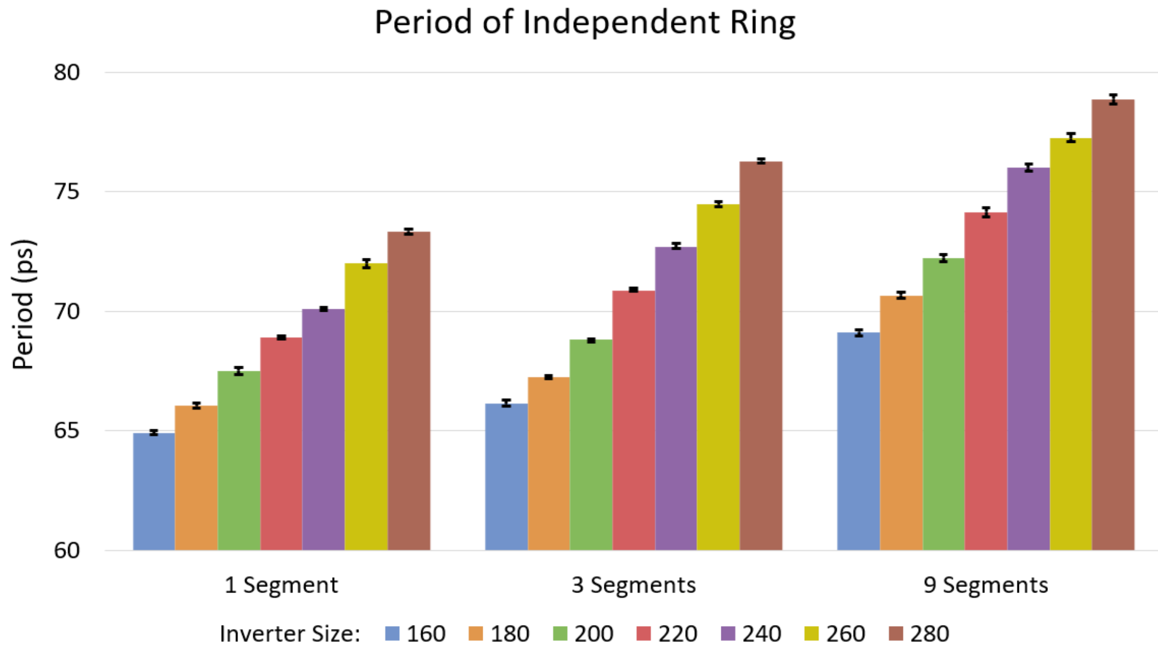

Figure 2.11: Period of independent RRSWO

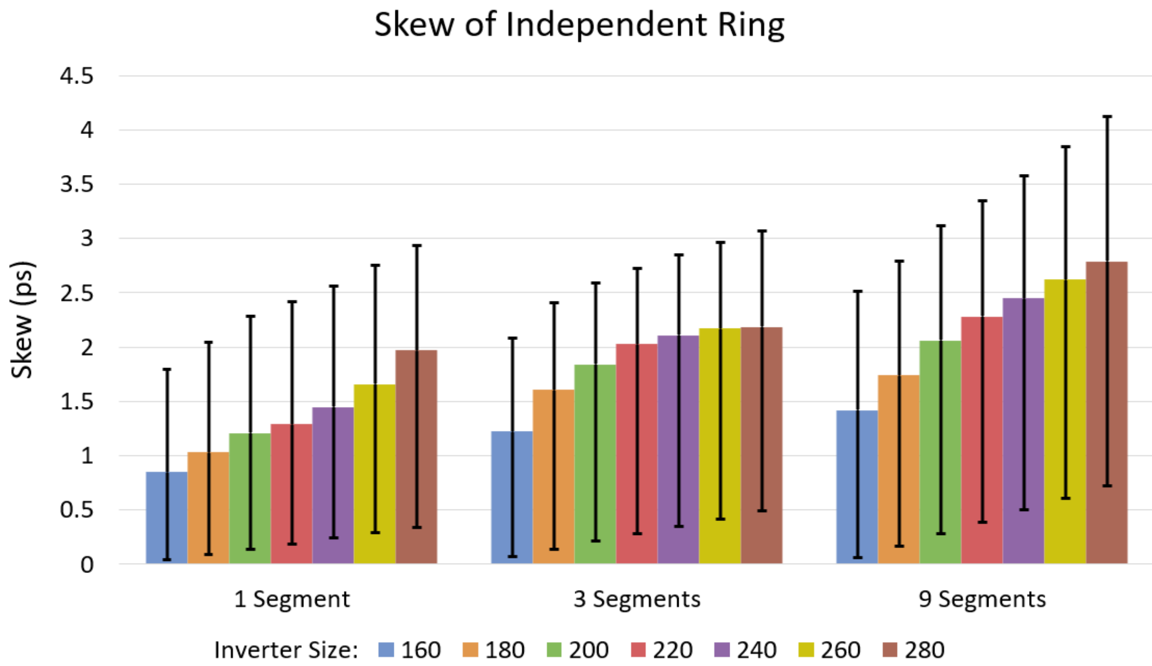

Figure 2.12: Skew of independent RRSWO

segments and inverter pair size. As expected, the power consumption of the RRSWO increases with the number of segments and the size of the inverter pairs. The inverter pairs are the primary source of power consumption in the ring as they maintain the sinusoidal oscillation of the clock wires. Therefore, as the inverter pair size increases, there is an increase in power per inverter pair of about $0.23mW$. This increase is represented by the difference seen from the $160\times$ inverter pair size to the $280\times$. In addition, as expected, the power consumption roughly triples as the number of segments triples. On average, each segment consumes approximately $1.21mW$ of power. This linear scaling in power consumption is significant as other clocking schemes (like the previously discussed grid or H-tree) have exponentially growing wire complexity and therefore exponentially growing power consumption as the size (number of stages) of the H-tree increases. In the following sections, we will evaluate the changes that occur when a redistribution layer (RDL) stub is introduced to synchronize two 3D RRSWOs.

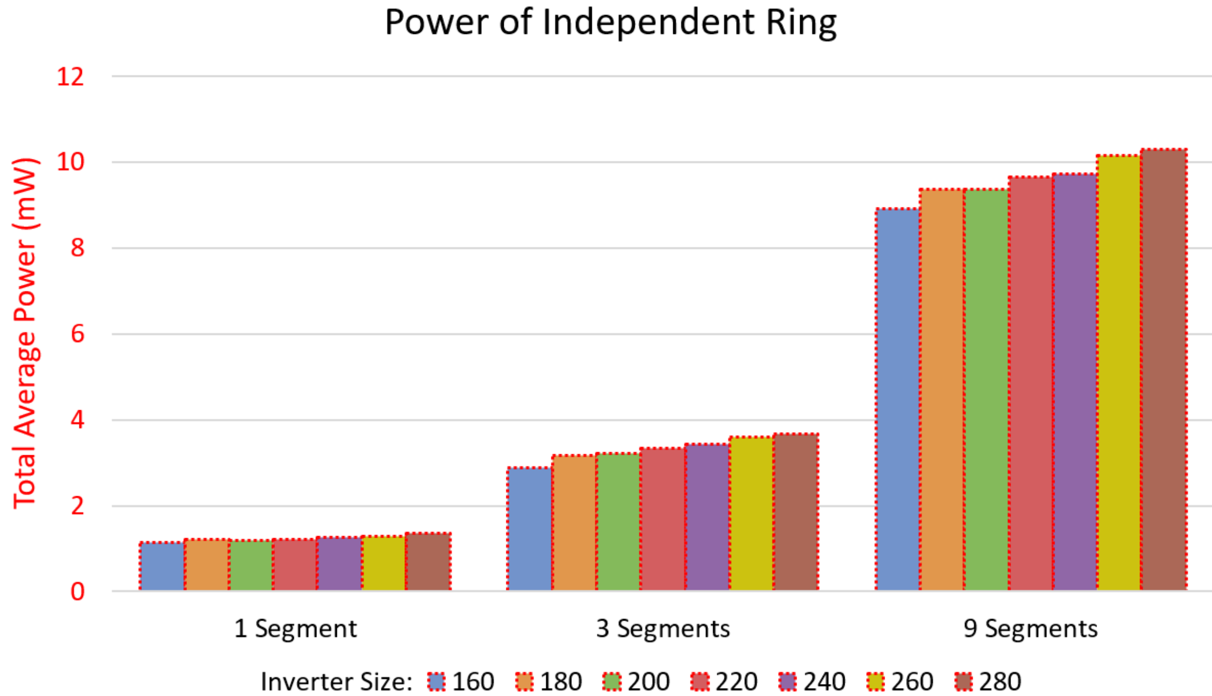

Figure 2.13: Power of independent RRSWO

2.4.3 Synchronized 3D RRSWOs

2.4.3.1 RDL Stub-based Synchronization of two 3D RRSWOs

Until recently, the RRSWO design has been specifically tailored for 2D clock generation *and* distribution. In Section 2.4.2, we presented our RRSWO design that was adapted for 3D ICs. In this section, we will discuss how we injection lock the RRSWOs of two 3D IC stacks.

Two stacks utilizing the 3D RRSWO that was presented in Figure 2.6 can be placed side by side as seen in Figure 2.14. A short RDL stub connects the RRSWO rings of the two 3D stacks at the indicated inverter pair locations to synchronize their oscillations. It is important to note that the RDL stub consists of two identical stub wires. One stub wire connects the outer wire of one RRSWO to the outer wire of the other RRSWO. The second stub wire connects the inner wire of one RRSWO to the inner wire of the other RRSWO. This is necessary in order to ensure the clock wire pair of one ring follows the same phase of the other ring. Use cases of such an architecture include a fast Network-on-Chip [2, 53], PLL related systems [36, 39] and 3D Stacked DRAM.

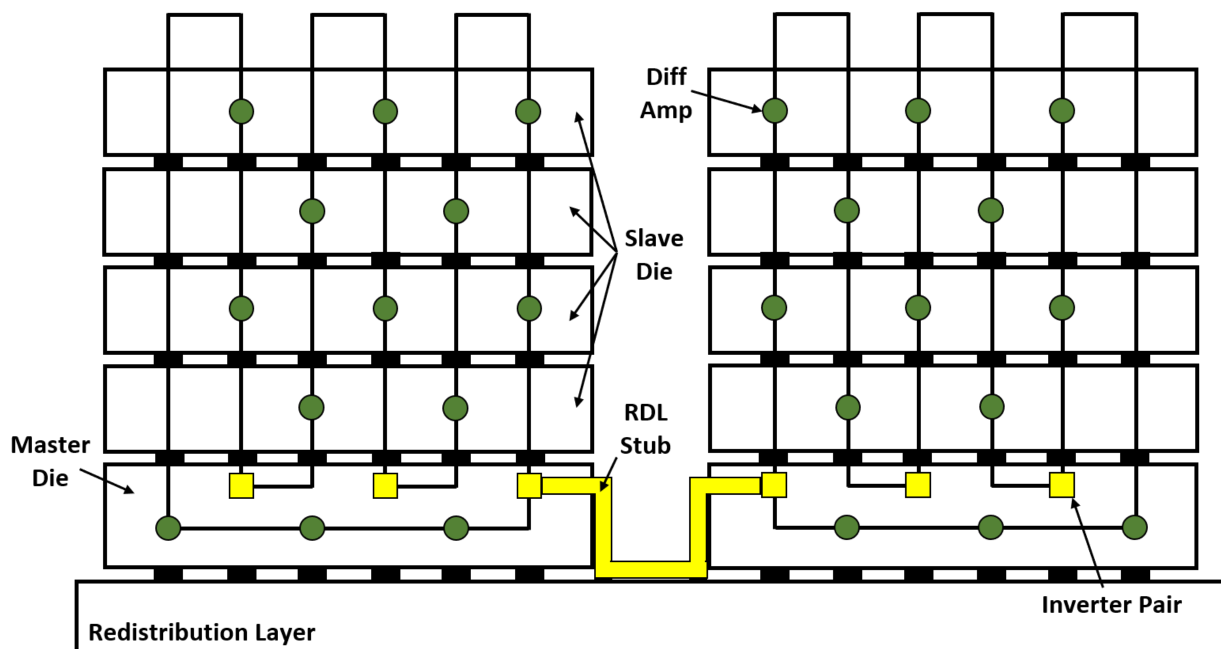

Figure 2.14: Synchronizing the RRSWOs of two 3D ICs

2.4.3.2 *Tristateable Inverter Pair Circuit*

The bootstrap circuit presented in Figure 2.4 forces the ring into an oscillatory state after power up. However, when synchronizing two RRSWOs, the skew between the reset signals of the different inverter pairs in the two different rings can cause synchronization issues. When the size of the inverters are the same in both rings, and the skew between the reset signals of these rings is over half the desired clock period, the rings can synchronize 180 degrees out of phase. Our solution to solve this issue is to increase the size of the inverter pairs of one ring compared to the other. This creates a master ring that will "drag" the slave ring to match its phase and oscillation frequency.

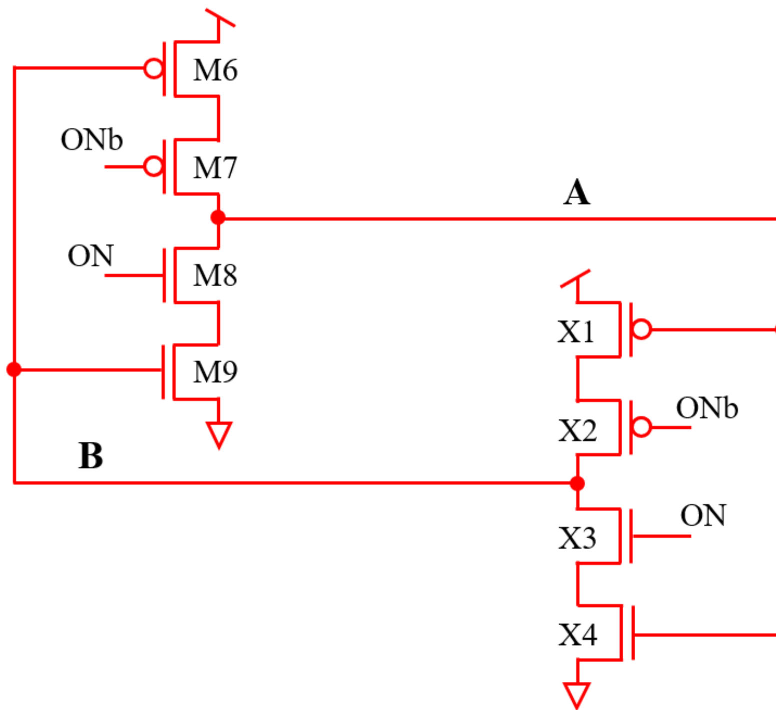

Figure 2.15: Tristateable inverter pair

Figure 2.15 shows the additional tristateable inverter pair that are added in parallel with the inverter circuit in Figure 2.4. When ON is high (ONb is low), then the tristateable inverter pair in Figure 2.15 will turn on, and the additional inverter, when connected in parallel with the circuit in

Figure 2.4, will effectively yield a larger ring inverter pair. The resulting increase in capacitance will slow down the ring slightly but enables us to adjust the size of the inverter pairs within each ring. By setting ON to high, and ONb to low in the first ring (and ON to low, and ONb to high in the second ring), we can force all the inverter sizes of the first ring to be larger than those of the second ring. In other words, the first ring acts as the master ring and the second ring acts as the slave ring. This ensures the rings will never synchronize 180 degrees out of phase. At the same time, the design and layout of both rings are identical, resulting in reduced manufacturing costs. The sizes of the additional tristateable inverters of Figure 2.15 ($20\times$ minimum size) are much smaller than those of Figure 2.4 (at least $140\times$ minimum size), hence the additional inverters do not interfere with the bootstrap operation.

2.4.3.3 Experimental Results

Figure 2.16 shows the reset waveforms and the clock outputs for two connected 3D RRSWOs as they transition out of power-up. The rst signal is the reset for the first RRSWO, while the DELrst signal shows the delayed reset signal. The ringA and ringB waveforms represent the signal of the outer wire of RRSWO-A and RRSWO-B respectively. This is the sinusoidal signal before it gets extracted to a full amplitude clock by the differential amplifiers. The diffA and diffB waveform respectively show the ringA and ringB signals after they pass through the differential amplifier. Following the de-assertion of the reset signals, the output of the two extraction points takes approximately $1.5ns$ to begin oscillating in phase. The signals remain in phase after $3ns$ (not shown).

In the following experiments, we simulated two rings which were connected via an RDL stub. The modeled RDL stub had a total capacitance of $185\,\mathrm{fF}$, a total resistance of $3.4\,\Omega$, and a total inductance of $1.35\,\mathrm{nH}$ (these values were provided by AFRL). There were several parameters varied, such as the size of the inverter pair, the number of ring segments k per ring, and the location of the RDL stub. In addition to these variations, we subjected the rings to cycle-by-cycle supply voltage variations ($60mV$ peak with a $100ps$ period), as well as long term supply voltage variations ($40mV$ peak over a $20ns$ period). For every simulation, we measured the clock period, intra-ring

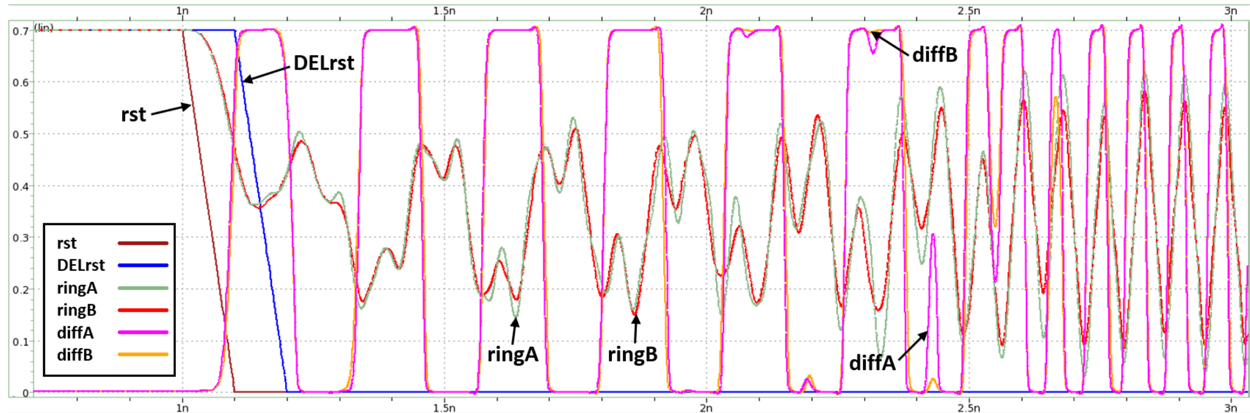

Figure 2.16: Power-up of 2 3D SWOs with the variable size bootstrap circuit

skew, inter-ring skew, power (at multiple points throughout the ring) as well as the current through the RDL stub. The inverter pair size of one ring was kept at $140\times$ minimum size ($140\times$ for the bootstrap circuit with the tristateable inverter pair circuit turned off). The other ring was varied from $160\times$ minimum size to $280\times$, where the $20\times$ tristateable inverter pair circuit is included in these calculated sizes. In other words, the bootstrap reset circuit was varied from $140\times$ to $260\times$ while the $20\times$ minimum size tristateable inverter pair circuit was turned on.

Figures 2.17, 2.18, 2.19 and 2.20 show the results for two independent RRSWOs that are connected via an RDL stub. Three different synchronization points are simulated (0, 45, and 90 degrees, where the virtual zero location is 180 degrees). All configurations consist of two independent rings (each with 3 inverter pairs) that have a total ring perimeter $6600\,\mu m$ ($2200\,\mu m$ per segment). The cross-section of the clock TSVs was kept consistent with the dimensions presented in the standalone 3D RRSWO section (Section 2.4.2).

Figure 2.17 shows the average, minimum and maximum period measured for different inverter sizes and RDL stub locations. This figure illustrates the ability of the RDL stub to synchronize the two rings that would otherwise oscillate independently at different frequencies. The results show that the oscillation frequency drops slightly (compared to Figure 2.11) when a stub is introduced. For example, two independent rings (no RDL stub) with inverter pair sizes of $160\times$ oscillate at 15.3GHz but instead oscillate at 13.3GHz when an RDL stub is introduced. This is expected

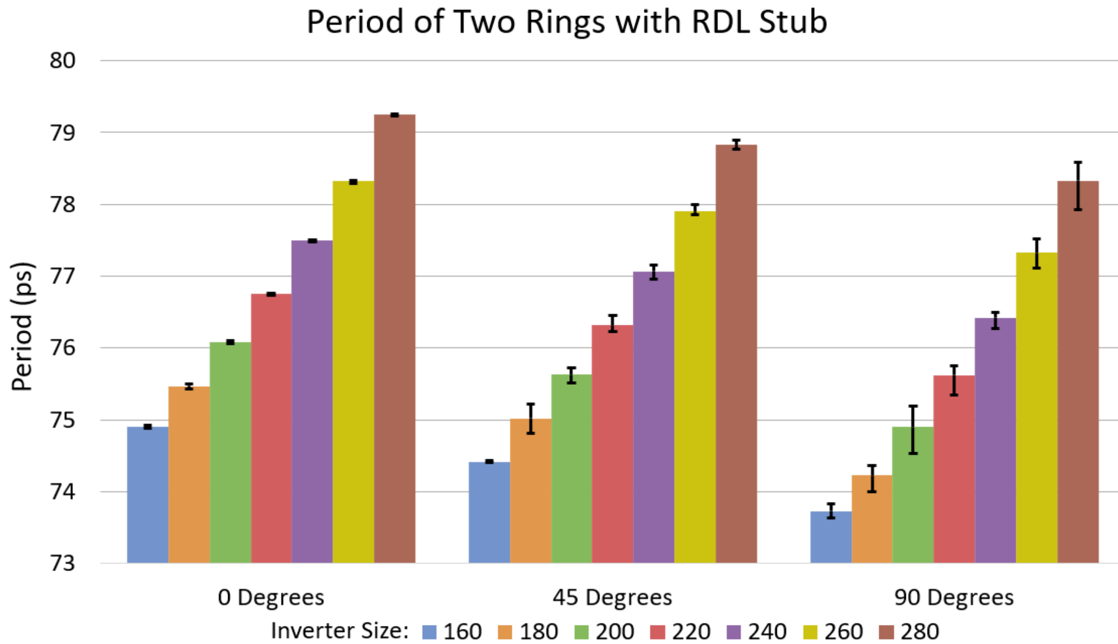

Figure 2.17: Period synchronization of two 3D RRSWOs with 3 segments each

since the ring parasitics increase due to the introduction of the stub. The bars in Figure 2.17 indicate the average period of the synchronized rings while the black markers indicate the minimum and maximum period that was measured for every configuration. A stub located at 0 degrees causes the worst degradation of the clock period but this degradation is less than $\pm 0.5\%$ compared to the clock period for a stub at 45 degrees. However, a 0-degree stub results in the lowest variation in the clock period (no more than $\pm 0.08\%$). Across all the stub locations, the variation of the clock for all inverter sizes and all RDL stub locations remained less than $\pm 0.7\%$ of the total clock period, which is extremely low.

Figure 2.18 and 2.19 shows the well-controlled intra- and inter-ring skew values. In every simulation, the size of the inverter pairs in the first ring was held constant at $140\times$, while the inverter size of the second ring was selected to be $160\times$ through $280\times$ in steps of $20\times$. The skew values are at most $\pm 3.6\%$ of the clock period, which is impressive given the high speed of the oscillator and the fact that no PLL is employed in our simulations. The intra-ring skew is represented by Figure 2.18 where the average skew is indicated as a vertical bar and the black

markers show the minimum and maximum skew. Figure 2.19 shows the inter-ring skew, which is significantly lower than the intra-ring skew values (at most 1.5% of the clock period). This is because the inter-ring skew is measured between identical points on the two RRSWOs while the intra-ring skew is measured across different points found in the same ring. The skew values presented in Figures 2.18 and 2.19 illustrate the effectiveness of our stub-based synchronization scheme.

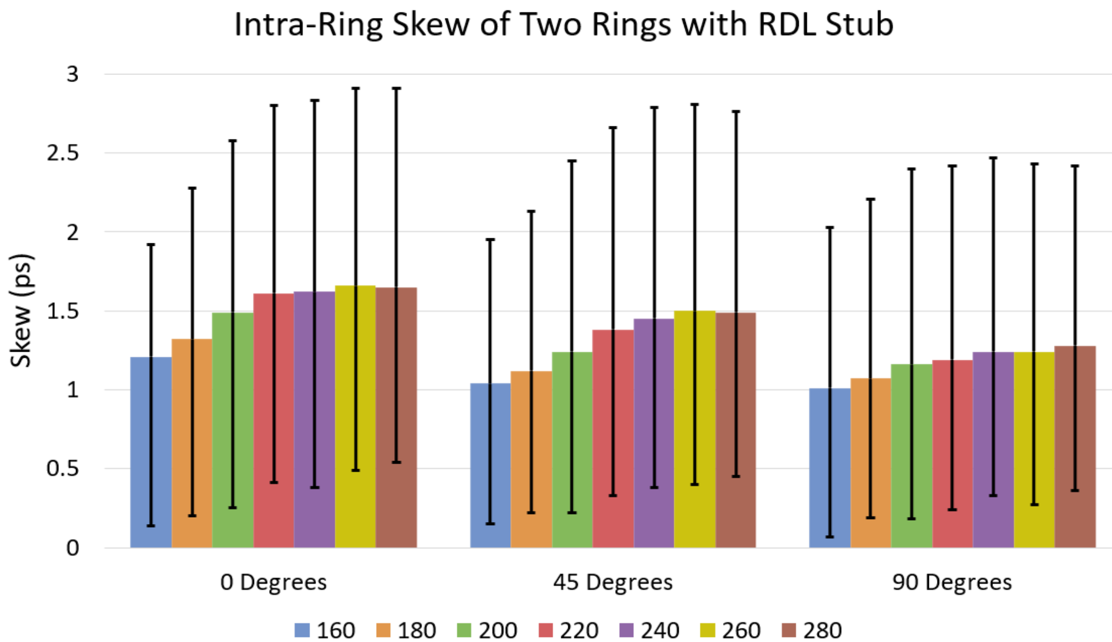

Figure 2.18: Intra-ring skew of two 3D RRSWOs with 3 segments each

Figure 2.20 shows the total average power consumption and the current through the RDL stub. The total average power consumption increases slightly with the inverter size as expected, but remains relatively constant when the location of the RDL stub is moved. The total average power consumption is in the range of 4.1 to 5.2 mW, which is close to the values obtained by the energy efficient PLLs presented in [43] and [44]. Our approach, however, can synchronize clocks over $5\times$ faster than other energy efficient designs. The bars that are outlined in the black solid line show the current through the RDL stub. As expected, the amount of current through the stub increases with

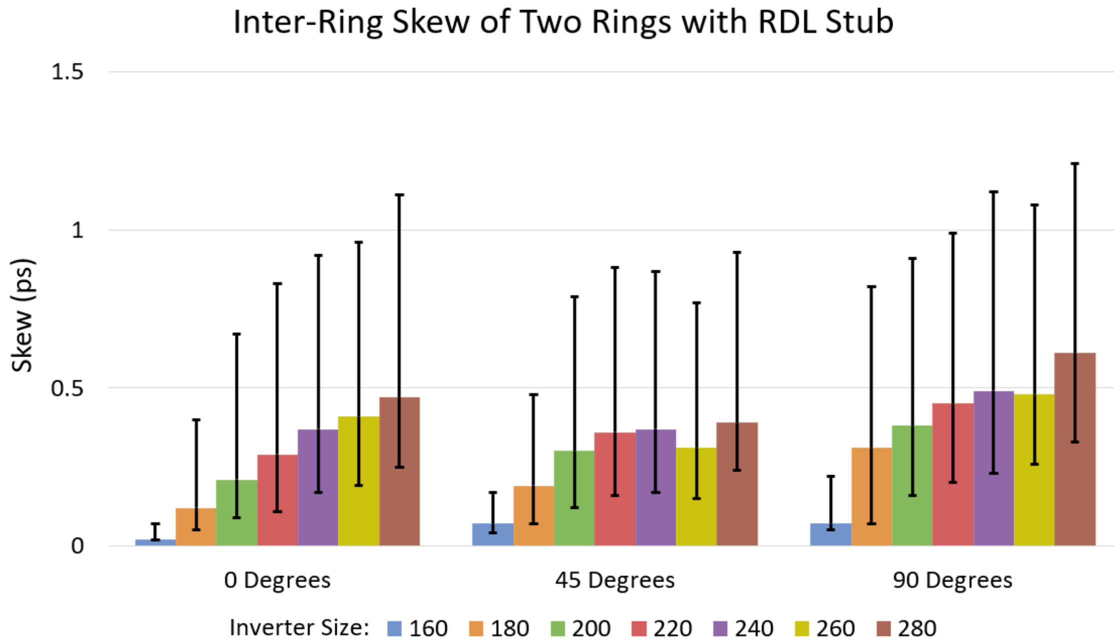

Figure 2.19: Inter-ring skew of two 3D RRSWOs with 3 segments each

the increasing size of the inverter pairs. This is because as the difference in inverter sizes between the two rings increases, the two RRSWOs would nominally oscillate at different frequencies if no RDL stub was used. This RDL current is the smallest when the larger inverter is $160\times$ (recall that the smaller inverter is always $140\times$), as the inverter sizes of the two rings are similar in size and would independently oscillate at a similar period.

When cycle-by-cycle supply voltage variations are introduced, with the RDL stub located at 0 degrees, the average clock period is within $\pm 0.3\%$ of the fixed VDD simulations. The minimum and maximum clock period varies up to $\pm 1.2 ps$. This is within 1.6% of the average clock period and is quite low. Average power consumption and current through the stub with cycle-by-cycle VDD variations remains within $\pm 0.6\%$ and $\pm 4.3\%$ respectively, as compared to the values reported for fixed VDD. Intra- and inter-ring skew however, is higher (by $\pm 4.4\%$ and $\pm 3.1\%$ respectively) when VDD is subject to cycle-by-cycle variations, which is expected considering the higher clock period variations.

With long-term VDD supply voltage variations, the average clock period again remains within

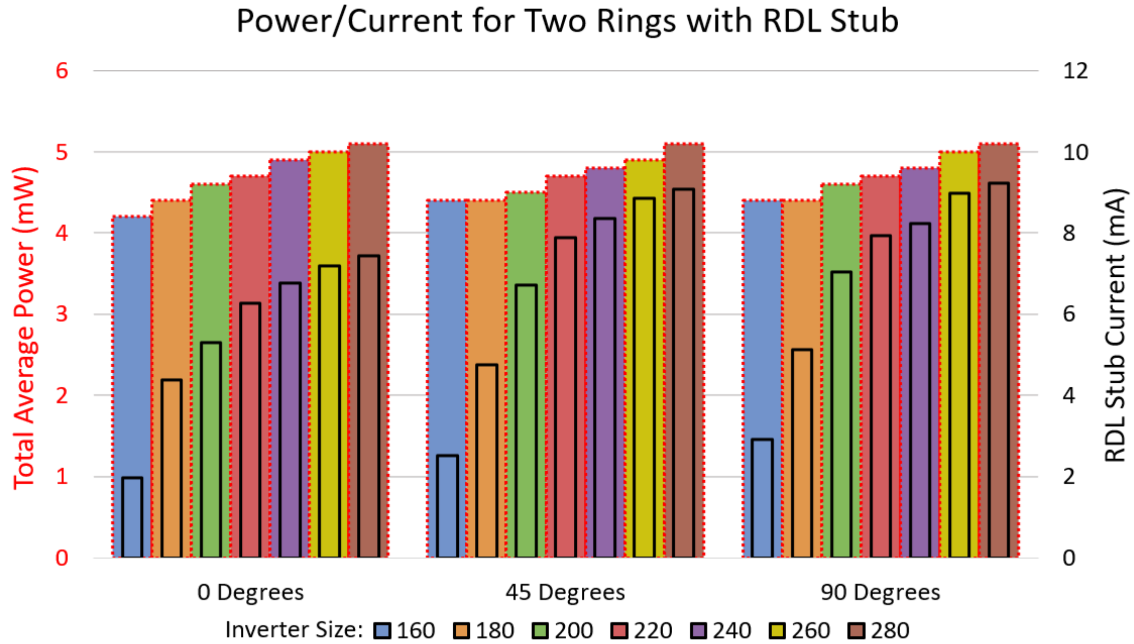

Figure 2.20: Power consumption and RDL stub current of two 3D RRSWOs

$\pm 1.2\%$ of the fixed VDD simulations (Table 2.1). This is significant because most oscillators have a strong dependence of frequency on the supply voltage. On the other hand, our oscillation frequency is dominated by the LC parasitics of the ring, hence its period barely changes with supply voltage variations. The clock period variations decrease compared to the clock period variations in cycle-by-cycle supply voltage (by about $0.1ps$). The intra- and inter-ring skew for long term VDD changes is less than the skew for cycle-by-cycle changes ($\pm 4.1\%$ and $\pm 2.4\%$ with respect to the clock period).

Table 2.1 shows a summary of the variations seen when cycle-by-cycle and long-term supply voltage variations are introduced. These results represent simulations with the RDL stub located at 0 degrees. The period, intra-ring skew, and inter-ring skew values are with respect to the average clock period for the fixed supply voltage simulations. The power and RDL stub current values are with respect to their fixed supply voltage averages. The variations were higher for cycle-by-cycle variations compared to long-term variations. However, the variations all remained extremely low because the RRSWO's oscillation frequency is primarily dependent on the ring's inductance (L)

and capacitance (C), as was shown in Equation 2.1. This is significant as it shows our 3D RRSWOs are resilient to supply voltage variations.

	Fixed VDD	Cycle-by-cycle	Long-term
Period (ps)	$\pm 0.7\%$	$\pm 1.6\%$	$\pm 1.2\%$
Intra-ring skew (ps)	$\pm 3.6\%$	$\pm 4.4\%$	$\pm 4.1\%$
Inter-ring skew (ps)	$\pm 1.5\%$	$\pm 3.1\%$	$\pm 2.4\%$
Power (mW)	$\pm 0.3\%$	$\pm 0.6\%$	$\pm 0.5\%$
RDL stub current (mA)	$\pm 3.9\%$	$\pm 4.3\%$	$\pm 4.2\%$

Table 2.1: VDD jitter for varying RDL stub locations connecting two 3D RRSWOs

The results presented in Figures 2.17, 2.18, 2.19 and 2.20 show that a wire stub connection can be used to synchronize two resonant RRSWOs without using a PLL. These RRSWOs will oscillate with minimal inter-ring skew and consume minimum power while maintaining frequencies above $10GHz$. This is significant because we not only generate and distribute a low skew clock, but we also are able to synchronize these low skew clocks across two 3D IC stacks (all while maintaining power consumption equivalent to energy efficient PLLs).

The experiments presented in Figures 2.21, 2.22, 2.23 and 2.24 show the results of varying the number of segments k in the ring. This was done by adding evenly spaced inverter pairs at a spacing of $2200\,\mu\text{m}$ in each ring. The clock wire TSV dimensions matched the cross-sectional dimensions used in the independent ring simulations (Figure 2.7). In addition to varying the number of inverter pairs, we also varied the size of the inverter pairs in one ring (from $160\times$ to $280\times$ in steps of $20\times$). The inverters in the other ring were fixed at $140\times$ as described earlier. Note that the tristateable inverter pair circuit is turned off in this constant inverter size ring to ensure the two rings never synchronize 180 degrees out of phase. The wire stub connection was left at 0 degrees for all simulations, since it yields the best performance in terms of skew, variation of period, average power and stub current (from Figures 2.17, 2.18, 2.19 and 2.20). In these experiments, the rings

had a total length of $2200\,\mu m$, $6600\,\mu m$, and $19\,800\,\mu m$ (corresponding to 1, 3, and 9 respectively).

Figure 2.21 shows the average, minimum and maximum period of the RRSWOs with the varied inverter size and 1, 3, and 9 segments (see Figure 2.3). As the results show, the average period varied slightly with the increasing number of segments. This variation, however, was within 5% of the average period for a given inverter size. The variation in the clock period remained within 0.22% for a given number of segments and inverter size (as shown by the black markers in Figure 2.21). This shows the ring's ability to generate and distribute a fast, low-jitter clock even as the number of segments was increased.

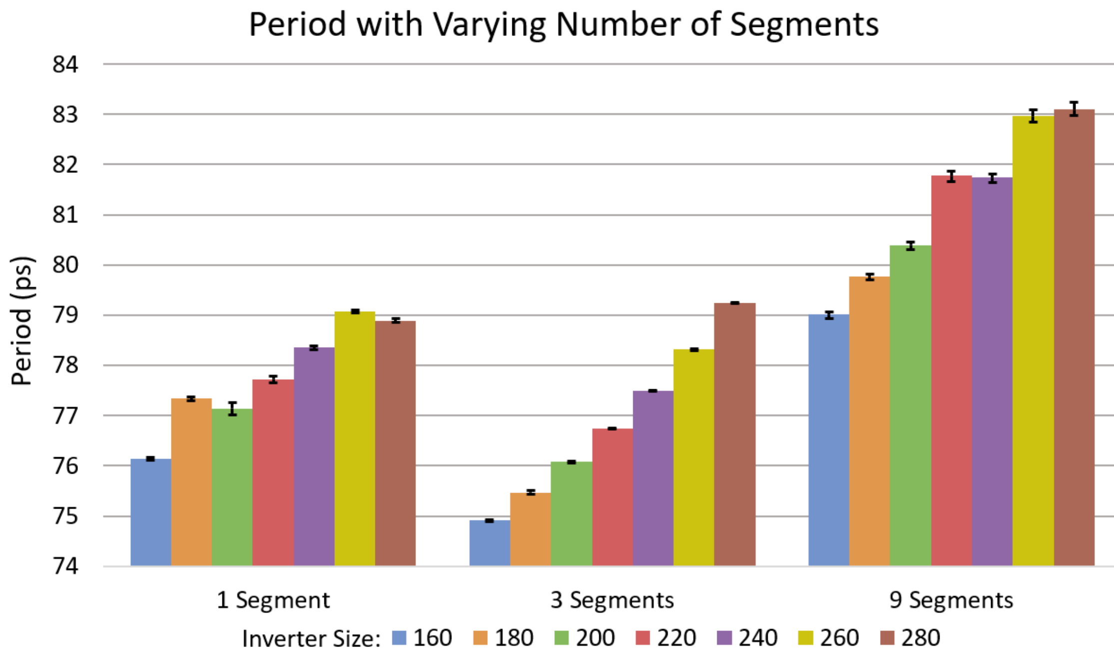

Figure 2.21: Period of two 3D RRSWOs with varying number of segments

Figures 2.22 and 2.23 show the average and maximum intra-ring and inter-ring skew as the number of segments in each ring is increased. The intra-ring skew was slightly higher than the inter-ring skew but remained within ±4.5% of the overall clock period (which is roughly the same for all ring sizes as discussed in Figure 2.12). The inter-ring skew for all configurations shows that

the clock period remained within ±3% at identical points in both rings. This is a critical result in showing that the two rings are synchronized and oscillating in phase without needing a PLL.

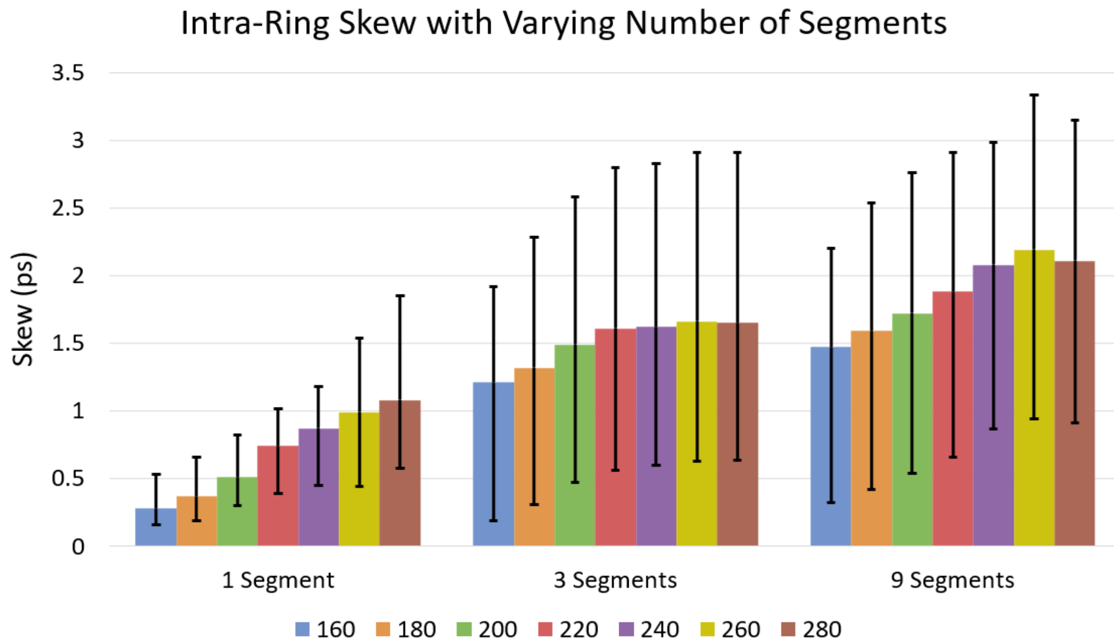

Figure 2.22: Intra-ring skew of two 3D RRSWOs with varying number of segments

The average power consumption, on the other hand, shows a roughly $3\times$ increase in relation to the $3\times$ increase in the number of inverter pairs (Figure 2.24). This is expected as these large inverter sizes are the primary contributors to the power being consumed in the RRSWO. The bars outlined by the solid black line in Figure 2.24 show the current through the RDL stub as the inverter size and number of segments are varied. As expected, the current increases with the increase in the number of segments in the ring and the inverter size. This is because the larger rings (the ones with more segments) requires more drive strength in order to ensure the rings oscillate together.

The experiments that were presented in Figures 2.21, 2.22, 2.23 and 2.24 were repeated for cycle-by-cycle VDD variations and long term VDD variations (Table 2.2). The addition of variations on the supply voltage causes higher variations in the clock period (about ±5% for cycle-by-

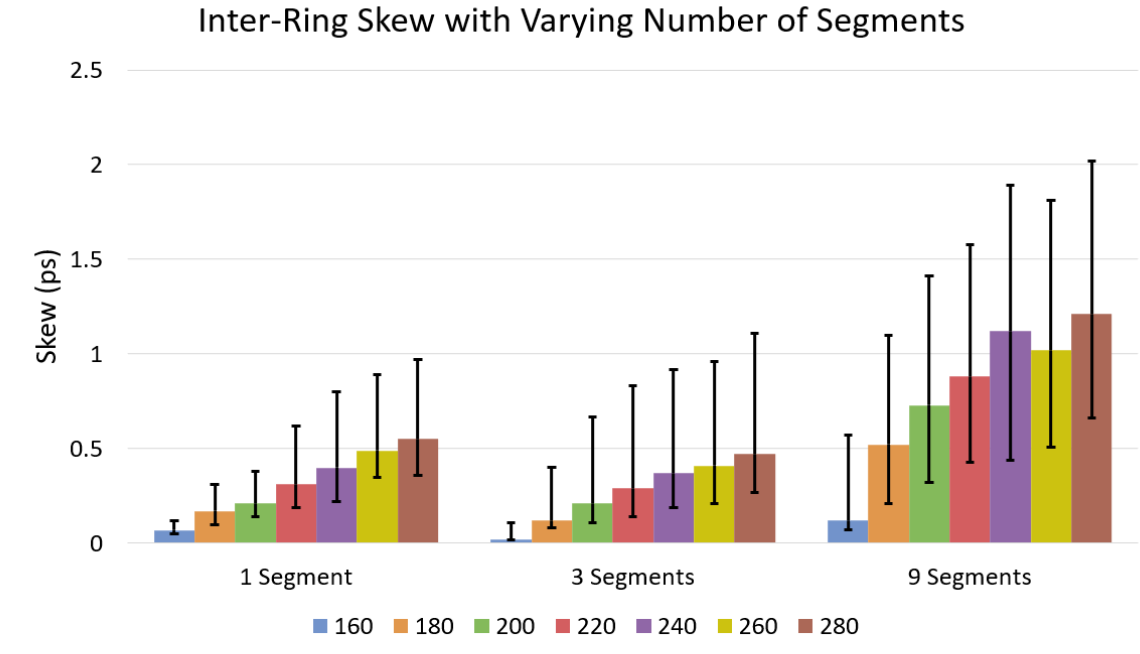

Figure 2.23: Inter-ring skew of two 3D RRSWOs with varying number of segments

cycle and ±4% for long term VDD variations compared to ±3% for a fixed VDD). However, the average clock period remains within ±1.7% of the clock periods for a fixed VDD (Figure 2.21). The intra-ring skew increases to within ±4% of the average clock period (compared to ±3% for fixed VDD) with long-term variations in VDD and increases to within ±6% of the average clock period with cycle-by-cycle variations. Finally, the inter-ring skew increases to within ±6% (compared to ±4.5% for fixed VDD) of the average clock period for cycle-by-cycle variations and to within ±5% of the clock period with long-term variations. The absolute skew numbers are quite small, however, as shown in Figures 2.22 and 2.23.

Table 2.2 shows the summary of the supply voltage variations when varying the number of segments in both 3D RRSWOs. These results represent simulations with 3 segments for both 3D RRSWOs. The period, intra-ring skew, and inter-ring skew values are with respect to the average clock period for the fixed supply voltage simulations. Similar to the varying RDL stub location simulations, the variations were higher for cycle-by-cycle variations compared to long-term variations. Again, the variations all remained extremely low because the RRSWO is primarily

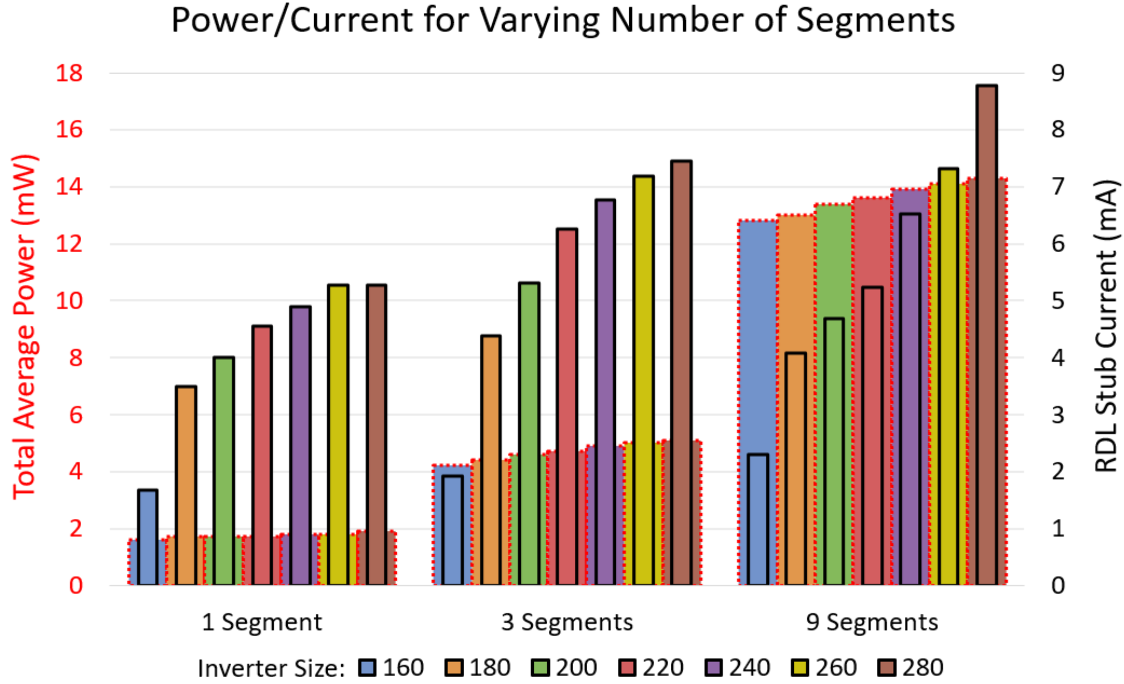

Figure 2.24: Power consumption of two 3D RRSWOs with varying number of segments

dependent on its total inductance and capacitance.

	Fixed VDD	Cycle-by-cycle	Long-term
Period (ps)	±0.6%	±1.7%	±1.2%
Intra-ring skew (ps)	±2.8%	±5%	±3.1%
Inter-ring skew (ps)	±1.5%	±3.2%	±2.4%
Power (mW)	±1.1%	±4.3%	±2.7%
RDL stub current (mA)	±2.2%	±7.2%	±4.5%

Table 2.2: VDD jitter for varying number of segments of two 3D RRSWOs

Finally, we want to show a comparison between the oscillation of the independent ring and a pair of rings with an RDL stub connecting them. Figure 2.25 shows the percent decrease in the oscillation frequency between the oscillation frequency of an independent RRSWO and the

oscillation frequency of two RRSWOs connected via an RDL stub (Figure 2.11 and Figure 2.21). As previously mentioned, the oscillation frequency drops slightly when the RDL stub is introduced because of the parasitics of the stub. This decrease in oscillation frequency dropped by almost $3\times$ as the size of the inverters is increased, as one would expect. It is important to remember, however, that the smaller inverter size results in the fastest clock frequency. In addition, Figure 2.25 shows that there is a smaller degradation in the oscillation frequency as the number of segments is increased. This is because a ring with more segments has larger L and C parasitics and hence insertion of the RDL stub does not increase the total L and C parasitics appreciably. The results presented show that two RRSWOs connected via an RDL stub are still able to maintain oscillation frequencies in the tens of gigahertz range.

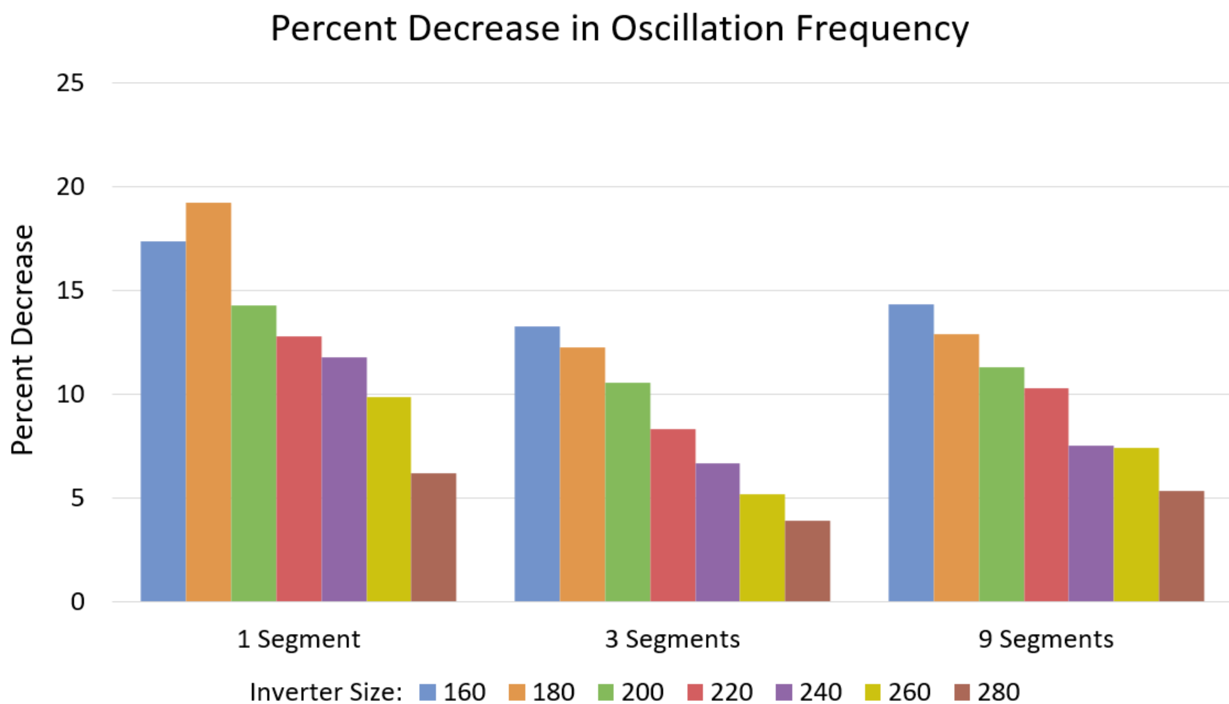

Figure 2.25: Percent decrease in oscillation frequency when RDL stub is introduced

2.4.3.4 Monte Carlo Simulations

The final experiments we performed on the stub-connected RRSWOs were Monte Carlo simulations. We used the 9-inverter pair configuration for these simulations because this configuration has the longest RRSWO rings. These simulations were performed to test the stability of the RRSWO structure. In each resulting figure, we show the statistics as the inverter size of one ring was varied. The total ring perimeter was held at $19\,800\,\mu\mathrm{m}$, the inverter size of the first ring remained at $140\times$, the wire stub remained at 0 degrees, and there were 9 inverter pairs in each RRSWO. For the threshold voltage, $\pm 3\sigma$ was taken to be about $\pm 6\%$ of the nominal threshold voltage [52]. For the channel length, $\pm 3\sigma$ was taken to be $\pm 10\%$ of the nominal minimum channel length [52]. These numbers are reasonable given that the inverter pair devices are chosen to be $2\times$ of the nominal channel length (to reduce the impact of variations) and have large widths as well. Each simulation was run 30 times (this was determined based on total runtime considerations) and the results are shown in Figures 2.26, 2.27 and 2.28.

Figure 2.26 shows the period statistics obtained from the Monte Carlo simulations. Compared to a configuration with 1 or 3 inverter pairs, the 9-inverter pair configuration was least affected by process variations. It resulted in low variations of the period and low intra-ring skew as the inverter sizes increased. The bars in Figure 2.26 represent the average period achieved. The black error markers on the average period bars show the standard deviation (the top of the error marker being one standard deviation above the average while the bottom of the error marker is one standard deviation below the average). This shows that despite the increasing standard deviation with increasing inverter size, 66% ($\pm\sigma$) of our simulations remained within $\pm 3.5\%$ of the clock period average. Similarly, 95% ($\pm 2\sigma$) of our simulations remained within $\pm 7\%$ of the clock period average.

Figure 2.27 shows the intra- and inter-ring skew for the Monte Carlo simulations. The bars that are outlined show the average skew values with the black error markers showing a standard deviation above and below the average for different configurations. The intra-ring skew (indicated by the left set of bars) has a standard deviation is within $\pm 2\%$ of the average clock period for small

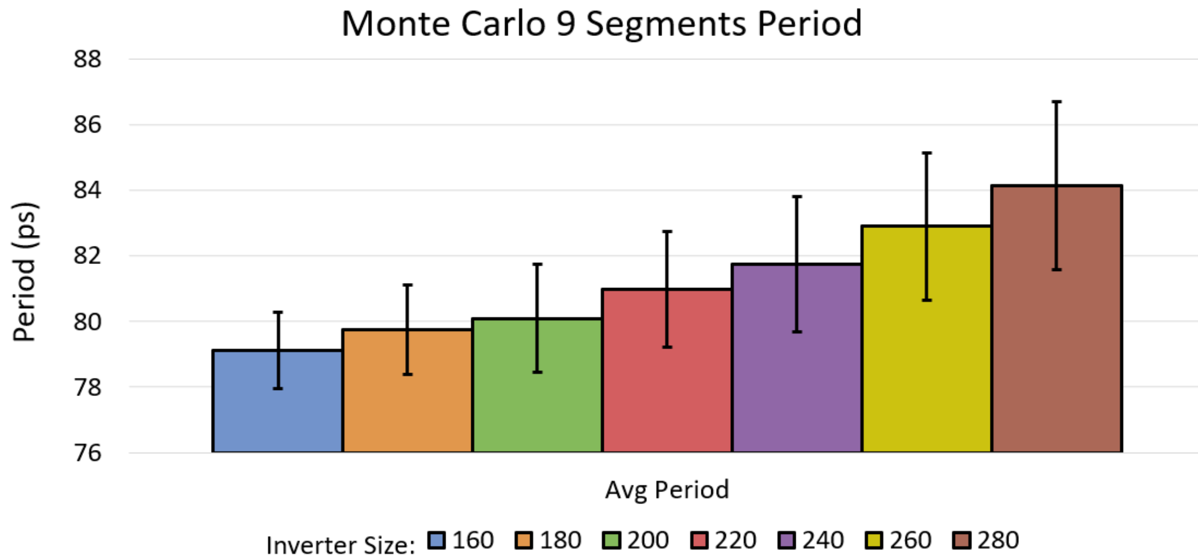

Figure 2.26: Period of two 3D RRSWOs for Monte Carlo simulations

inverter pair sizes while it increases to ±4% as the inverter pair sizes increase. The inter-ring skew standard deviation which is within ±1.5% of the average clock period for small inverter sizes and increases to ±3.5% as the inverter pair sizes increase.

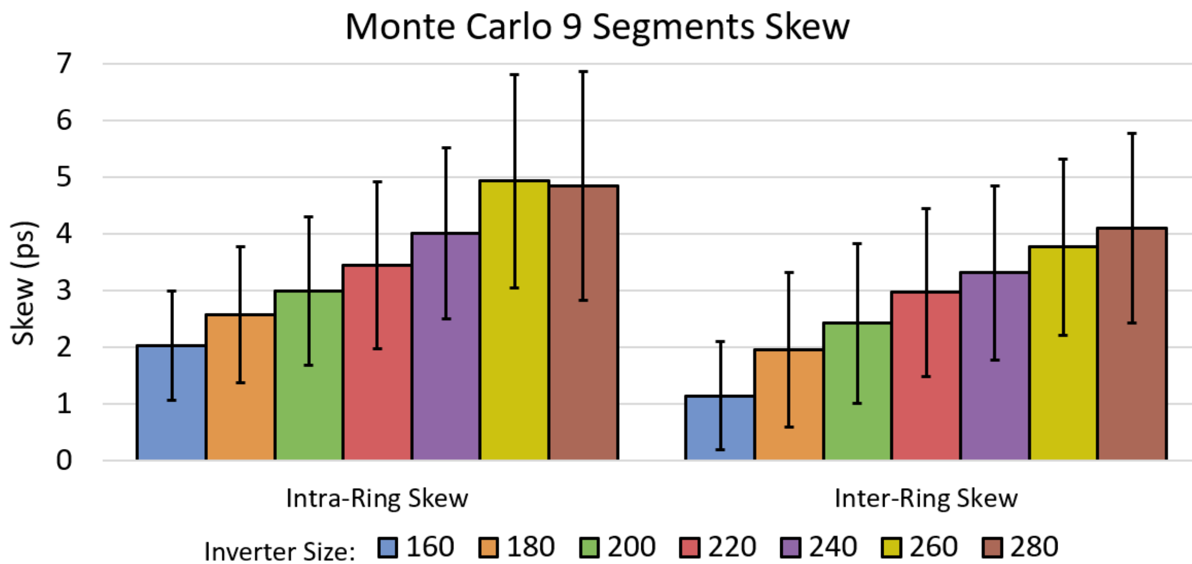

Figure 2.27: Skew of two 3D RRSWOs for Monte Carlo simulations

Figure 2.28 shows the average power consumption and the current through the RDL stub for the Monte Carlo simulations. As expected, the average power consumption and current through the RDL stub gradually increase with increasing inverter size. The error markers show that the standard deviation of each configuration remains within $\pm 8.3\%$ for power consumption and $\pm 14\%$ for RDL stub current. However, for all configurations, and considering two standard deviations above the average, the RRSWO configurations that were simulated still consume less than $16 mW$ of power. This is significant since our topology consists of multiple clock networks (one for each 3D stack) and oscillates in the tens of gigahertz range. Again, the effect of the process variations for the 9-inverter pair configuration is less drastic compared to RRSWOs with fewer segments.

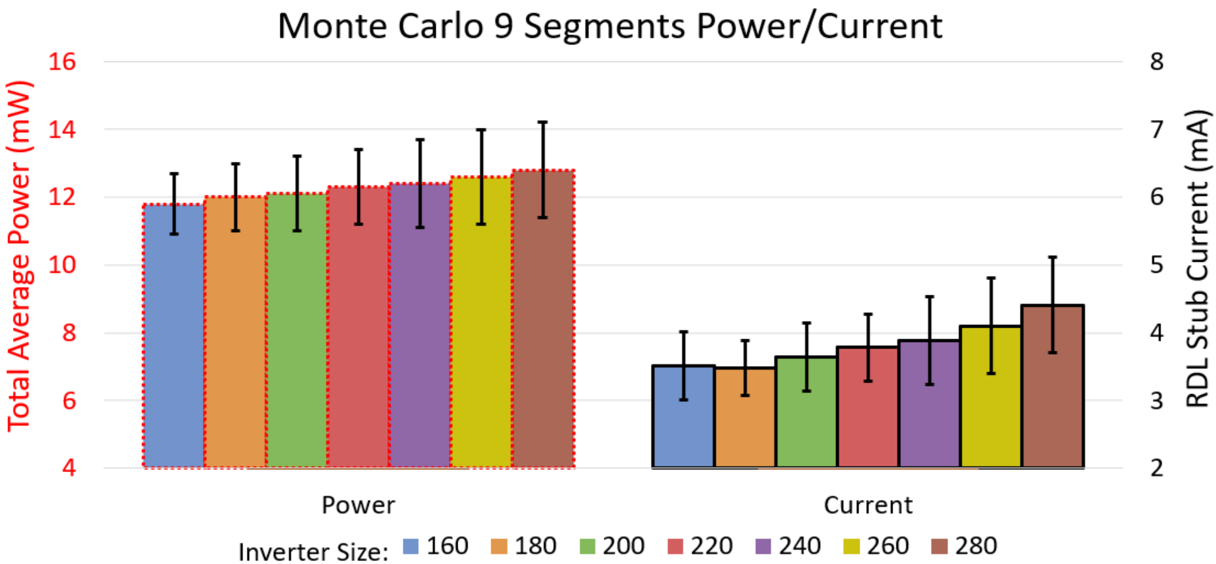

Figure 2.28: Power and RDL stub current of two 3D RRSWOs for Monte Carlo simulations

2.5 Conclusion

This chapter first presented a stand-alone RRSWO design for 3D IC stacks. A bootstrap circuit for stand-alone RRSWOs was also described. In addition, we presented an injection locking approach via a short RDL wire stub to synchronize two independent 3D RRSWOs. With a wire stub,

the 3D ring-based RRSWOs do not need a PLL and can maintain fast, low power and synchronized oscillations. This is significant as it presents a method for synchronizing clocks between ICs without the use of a PLL. We also presented a modified bootstrap design for the stub-connected pair of RRSWOs. Our bootstrap circuit allows the RRSWOs to begin oscillation from an initial state, and synchronize with additional RRSWOs in $5\times$ less time than previously proposed PLL approaches. Finally, we quantified the resilience of the stub-connected RRSWOs to process variations through Monte Carlo simulations. As 3D ICs become more prevalent, the results we presented in this chapter can provide a useful architecture to create a scalable and reliable clocking scheme for 3D systems.

3. 3D RRSWOS FOR DRAM APPLICATIONS*

3.1 Introduction

Despite significant improvements in processor performance, main memory has lagged behind and caused a bottleneck to system performance. To mitigate this "memory wall", engineers have increased clocking rates of the memory data bus for each generation of Double Data Rate (DDR) memory. This enables higher memory bandwidth and has been proven effective for small data burst lengths. For large data bursts, bandwidth tends to be wasted as data is transferred to the cache, and often not used. The probability of transferred data not being used increases as main memory sizes grow and multi-core systems become commonplace. Therefore, different applications utilize different DRAM architectures which vary in bandwidth, cost and power consumption. The following sections discuss the history of these architectures, their disadvantages and the previous approaches to alleviate these disadvantages. The final sections describe our approach and the experimental results of our memory architecture compared to traditional DDR architectures.

3.2 3D Stacked Memory

To improve the performance of traditional DDR technologies that were listed in the first chapter, many industry leaders have developed 3D DRAM stacks to meet the high bandwidth requirements. Unlike the previously mentioned DDR architectures, these 3D stacks can increase memory performance without increasing the form factor. By utilizing compact TSVs, instead of traditional I/O pins, 3D stacked memories have been able to achieve bandwidths significantly higher than previous DDR technologies. With high-end applications requiring larger memory footprints, and faster access times, the following 3D DRAM architectures can help mitigate memory bottlenecks.

*Part of the data reported in this chapter has been updated with permission from "Fast, Ring-Based Design of 3D Stacked DRAM" by Andrew Douglass and Sunil P. Khatri, 2017. International Conference on Computer Design (ICCD), pp. 665-672, Copyright by IEEE.

3.2.1 High Bandwidth Memory

High Bandwidth Memory (HBM) is a high-performance version of 3D stacked memory designed for GPU related applications. Adopted as a JEDEC standard in 2013, HBM targets high throughput by using a 1024-bit data bus that gets divided into 8 independent channels (128 data bits per channel). HBM takes traditional DRAM dies and stacks them up to eight high with a logic die at the base to act as the memory controller. By stacking the memory chips, instead of placing them on the PCB, 1GB of HBM only takes $35mm^2$ of space compared to $672mm^2$ of space in a traditional GDDR5 architecture [54, 55]. With this smaller form factor, HBM memory stacks can be placed on an interposer alongside the GPU (Figure 3.1). This allows wider busses between the DRAM stacks and the processor to support the wider memory channels. However, the use of interposers increases costs since they are significantly more expensive than PCB manufacturing. HBM stacks are typically four DRAM MCs tall, with each MC containing two independent channels. It is important to note that these channels are completely independent and are often not synchronous with one another. The close proximity, higher chip count and wide data bus allow HBM to achieve throughputs of over 100GB/s per stack [55, 56].

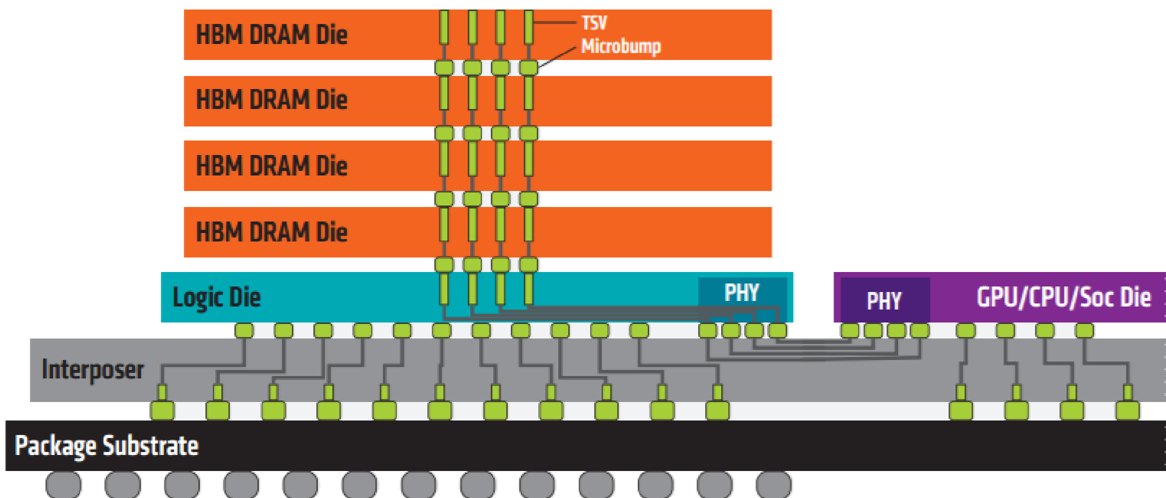

Figure 3.1: HBM cross sectional view

HBM was first deployed in AMD's Radeon series GPUs, which is based on the Fiji architecture, in 2016. In the same year, JEDEC standardized the second generation of HBM known as HBM2. Similar to its predecessor, HBM2 allows stacks that are up to eight MCs high. However, they are able to double the pin transfer rates through the use of pseudo channels. These pseudo channels essentially break individual channels into two independent sets of sub-channels (Figure 3.2). In what is called legacy mode (a mode used to support HBM), each read/write command will transfer 256 bits during 2 cycles. Using pseudo channel mode (the mode used in HBM2), the 128-bit data bus is split into 2 individual 64-bit segments that still transfers 256 bits but reduces the four-bank activation window time [57]. A recent modification to the HBM2 specification allowed for stacks of up to 12 MCs which has increased HBM2's peak bandwidth to 307GB/s per stack.

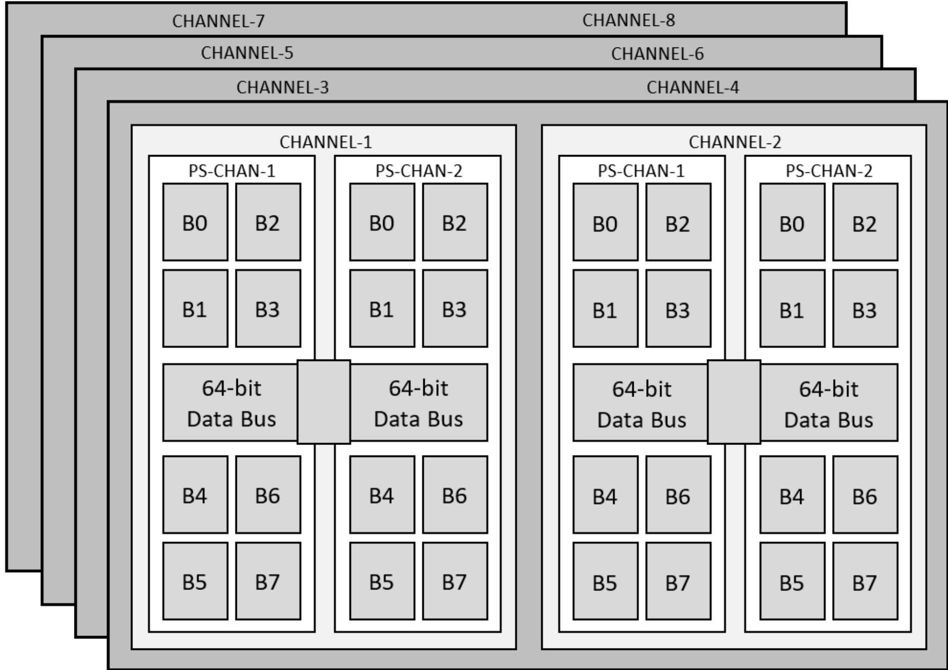

Figure 3.2: HBM2 pseudo channels overview [3]

Industry leaders have also been developing a third and fourth generation of HBM. Each generation is promising higher stacks and denser DRAM dies to allow for higher capacity and higher

bandwidth. However, as these DRAM stacks continue to grow in size, the ability to cool the MCs becomes increasingly difficult. In addition, there is a limited amount of space where TSVs can be placed which severely limits the number of independent channels that can be achieved per stack. Later in this chapter, we will show our proposed architecture, which utilizes the previously discussed 3D RRSWOs in order to achieve an efficient communication topology. Our results will show that this topology can reduce the number of TSVs, and power consumption while maintaining speeds similar to HBM. This will show that our topology is a great candidate for a high-speed interconnect in these future 3D memory stacks.

3.2.2 Wide I/O Memory

Wide I/O Memory is another 3D stacked memory but is targeted for low power applications. Designed with mobile devices in mind, Wide I/O originally promised stacks of DRAM to be placed directly on top of the processor. This results in significant thermal and power issues since the heat from the processor die needs to dissipate through the stack. As a result, most Wide I/O implementations still use an interposer like HBM. Wide I/O promises up to 17GB/s by using four channels with 128 data bits per channel. The reduced bandwidth comes with less than half the power consumption of HBM [58, 59, 60].

Wide I/O 2 builds on its predecessor by improving bandwidth to 68GB/s and minimizing power consumption. Unlike the 512-bit data bus used by Wide I/O, Wide I/O 2 increases the bus width (to match HBM) to 1024 bits [59]. Targeting high-end lower power applications, Wide I/O 2 became a JEDEC standard in 2014. To date, this type of 3D stacked memory has yet to make it to production, partly due to its high price tag for a mobile device.

3.3 Previous Work

There have been several research efforts that address the performance of 3D stacked memory designs. In [61], the authors explored a solution to reduce skew between the 32 data lines used for the data bus in HBM. Increases in skew reduce the window for valid data and result in decreased bandwidth. To achieve a lower read skew, the authors use buffers on the logic die that are adjusted

by successive approximation register logic. For write skew, they form loopback paths on the logic die by controlling the buffer delays. After cancellation, they are able to achieve $15\times$ less skew with a difference of $18ps$ still left between the data lines [61].

The design of [62] proposes a technique to reduce refresh time for Hybrid Memory Cubes, which is another form of 3D stacked memory developed by Intel and other companies. The paper suggests using subarray-level and bank-level concurrency to reduce the time spent performing the refresh operation for the memory cells. By mapping multiple DRAM rows to subgroups which are refreshed concurrently, significant time is saved refreshing the memory. This reduces the power consumption by 5.8% and improves throughput by 6.3% [62].

The work of [63] proposed a new I/O organization to increase memory bandwidth. They combine the internal bandwidth of multiple layers of DRAM die to increase I/O speeds. By time-multiplexing the I/Os across DRAM dies, the authors claim up to 55% improvements in performance for multi-programmed workloads. The most important factor, however, is that it maintains low cost by leveraging internal global bit-lines instead of adding external global bit-lines [63].

The authors in [64] discuss the improvements that can be made in large scale data servers by using the message passing interface packet protocol. This protocol, however, requires high memory bandwidths with short interconnects to ensure memory utilization does not result in system performance bottlenecks. They show how the use of 3D stacked DRAM can be used to service the message passing interface protocol to increase system performance in large scale data servers. This allows many context switches with a fast response time as the amount of memory can be increased while decreasing the trace lengths.

3D memory stacks possess unique characteristics that result in new storage paradigms. One such paradigm is the use of a reshape accelerator for 3D stacked DRAM [65]. Simulations are performed utilizing their proposed reshape accelerator, to map physical addresses via a layout transform to exploit locality. This method of data reorganization uses the internal operation and organization of a traditional 3D memory stack by modifying the logic layer. With only a 0.6% increase in power consumption, this address mapping uses a matrix transform for efficient compu-

tation. In certain applications with poor initial mappings, the authors show that their technique can improve total read latencies by up to $2.2\times$.

3.4 Our Approach

There have been many proposed solutions to solve the performance bottlenecks that have plagued memory system performance over the past few decades. The development of 3D stacked DRAM, such as HBM and Wide I/O, have demonstrated significant improvements over current DDR technologies but resulted in new issues. One issue is the ability to distribute a low power, low skew and fast clock to all DRAM dies. Another issue is that these 3D memory architectures need to utilize as few TSVs as possible, in order to maximize the amount of die area that can be used for DRAM storage. Current 3D memories, such as HBM, utilize a large number of TSVs in order to achieve higher channel counts (which is required to achieve higher bandwidth). This increases the power consumption, decreases available die area and increases manufacturing costs. In order to increase speeds for future 3D memory generations, this chapter explores the use of 3D RRSWOs to provide an efficient, high-speed ring-based communication design which utilizes significantly fewer TSVs.

3.4.1 Ring-Based Interconnect for 3D DRAM

Our Memory Architecture using a Ring-based Scheme (MARS), shown on the right side of Figure 3.3, utilizes a series of address, control and data lines that are routed alongside a 3D RRSWO. These wires intersect each channel in a DRAM stack at Insertion-Extraction Stations (IES'), where data can be injected or removed from the ring. This creates a circular data path where commands can be communicated from the memory controller that lies on the logic die to the DRAM memory stacked above it. A simple RRSWO (discussed in the previous chapter) is used to synchronize all IES' within this circular data path. To ensure that all IES' receive a full amplitude clock, and that no station falls in the "dead-zone" of the RRSWO, longer clock traces are placed at the top of the DRAM stack (Figure 3.3 right), such that the "dead-zone" falls over these longer traces. Figure 3.3 shows the comparison between a traditional HBM stack (left) and a MARS stack (right). The

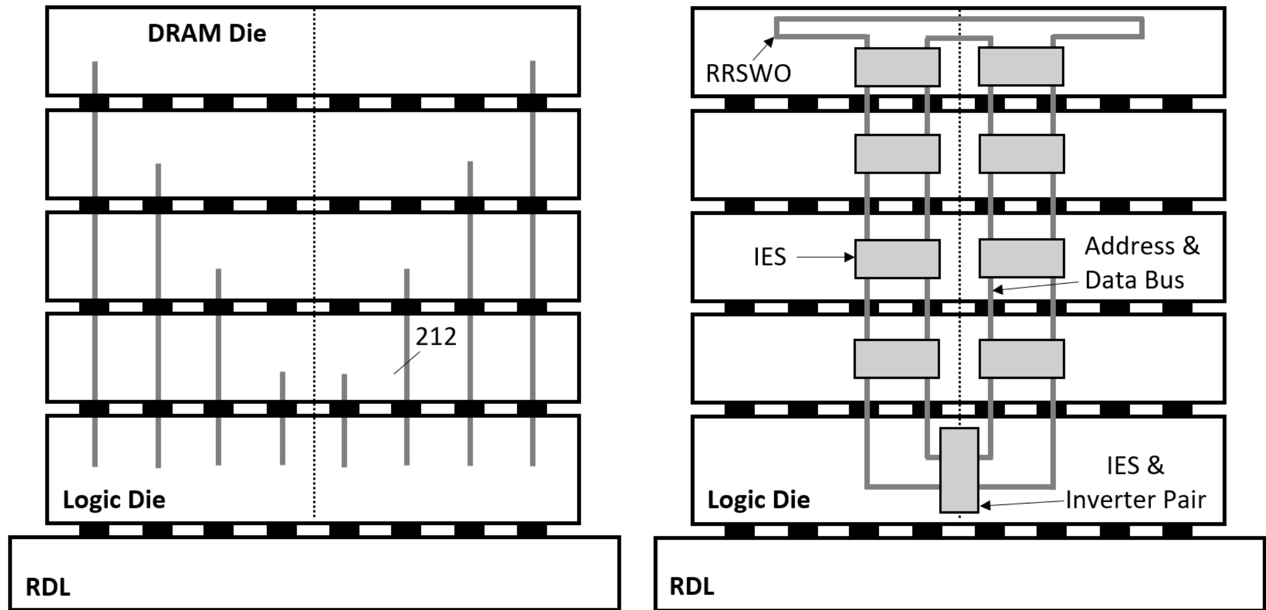

Figure 3.3: Cross sectional view HBM (left) and MARS (right) for a DRAM stack

dotted lines indicate the division between channels that lie on the same DRAM slave die. Therefore, both the HBM stack and the MARS stack in Figure 3.3 represent eight independent channels that lie on four DRAM dies. The configuration shown on the right side of Figure 3.3 is referred to as MARS8.

Using the MARS concept, Figure 3.4 shows how we can scale the architecture to service higher channel counts. The left side of the figure shows a MARS configuration that has 8 DRAM dies and 16 channels. We will denote this topology as MARS16. Notice how the ring-based data fabric has been duplicated where each ring still contains 9 IES' and services 8 independent channels. The right side of Figure 3.4 shows a stack with 16 DRAM dies and 32 total channels. We denote this topology as MARS32. This stack contains 4 independent ring-based data fabrics each connecting to 8 DRAM channels. Note that the height of both stacks remains the same. According to the JEDEC standard, as more ICs are added to the stack, the height of the stack remains the same because the substrate is made thinner by back-grinding, thereby ensuring that the total stack height is constant. This means that the height of all stack remains constant at $720\,\mu\text{m}$ and allows us to use the same RRSWO configuration for each of the MARS topologies described in this thesis.

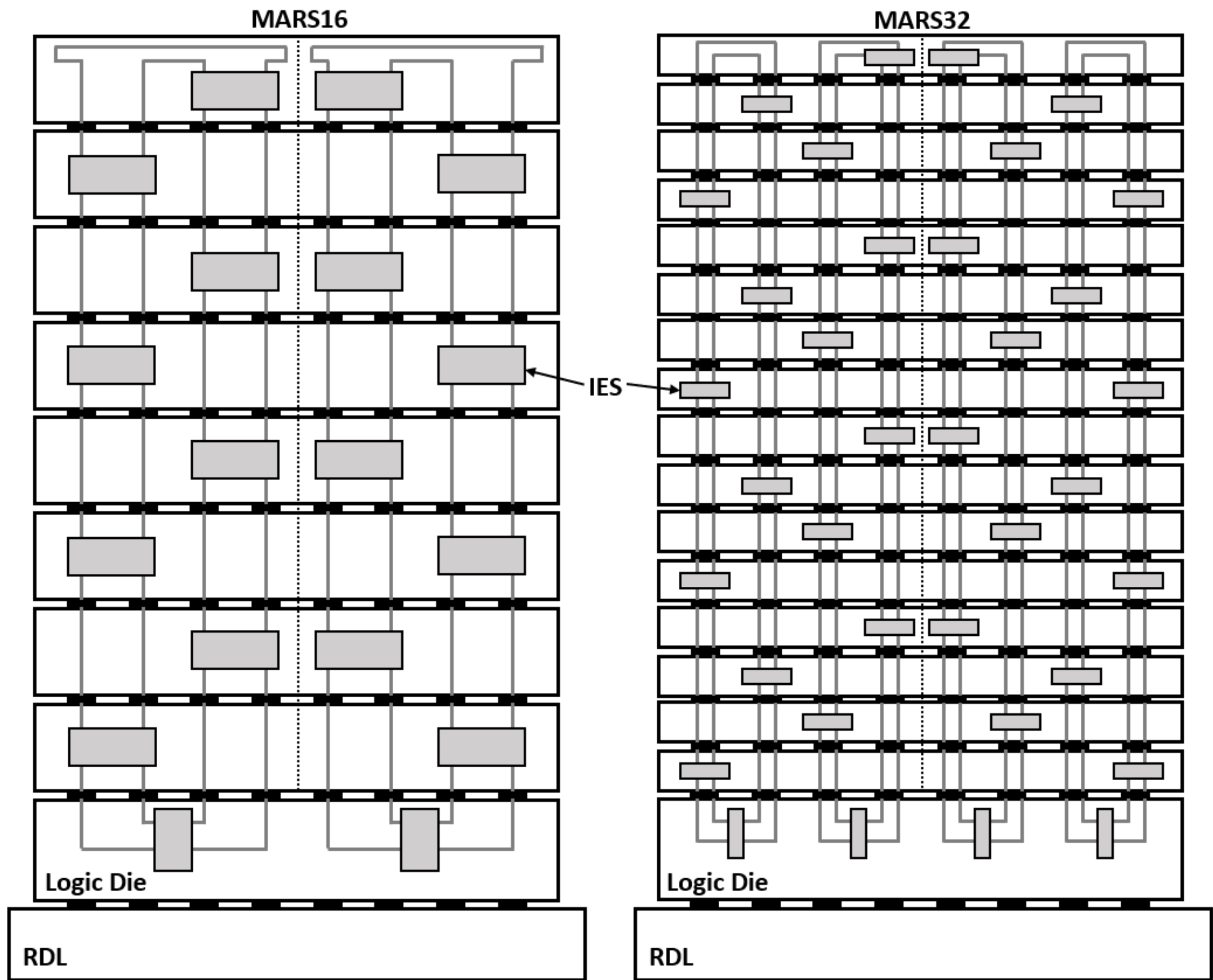

Figure 3.4: Cross sectional view MARS16 (left) and MARS32 (right)

It is important to note that a 16-high DRAM stack is not currently feasible, but we present the MARS32 configuration to show the scalability of our approach and show how it can be used for future aggressive memory stacks.

Our MARS topology contains address, control and data lines that are driven between adjacent IES' in each clock cycle. Because there are a total of 9 IES', including the IES that lies on the logic die, the clock needs to oscillate at 18GHz in order to ensure that the round-trip time of a memory command starting at the logic IES is equivalent to that of an HBM design (0.5 nanoseconds). The RRSWO that was utilized in the previous chapter is well suited to achieve such a frequency with

low power consumption and low skew. The IES' are uniformly spaced throughout the circular data path to ensure information between the stations is synchronized.

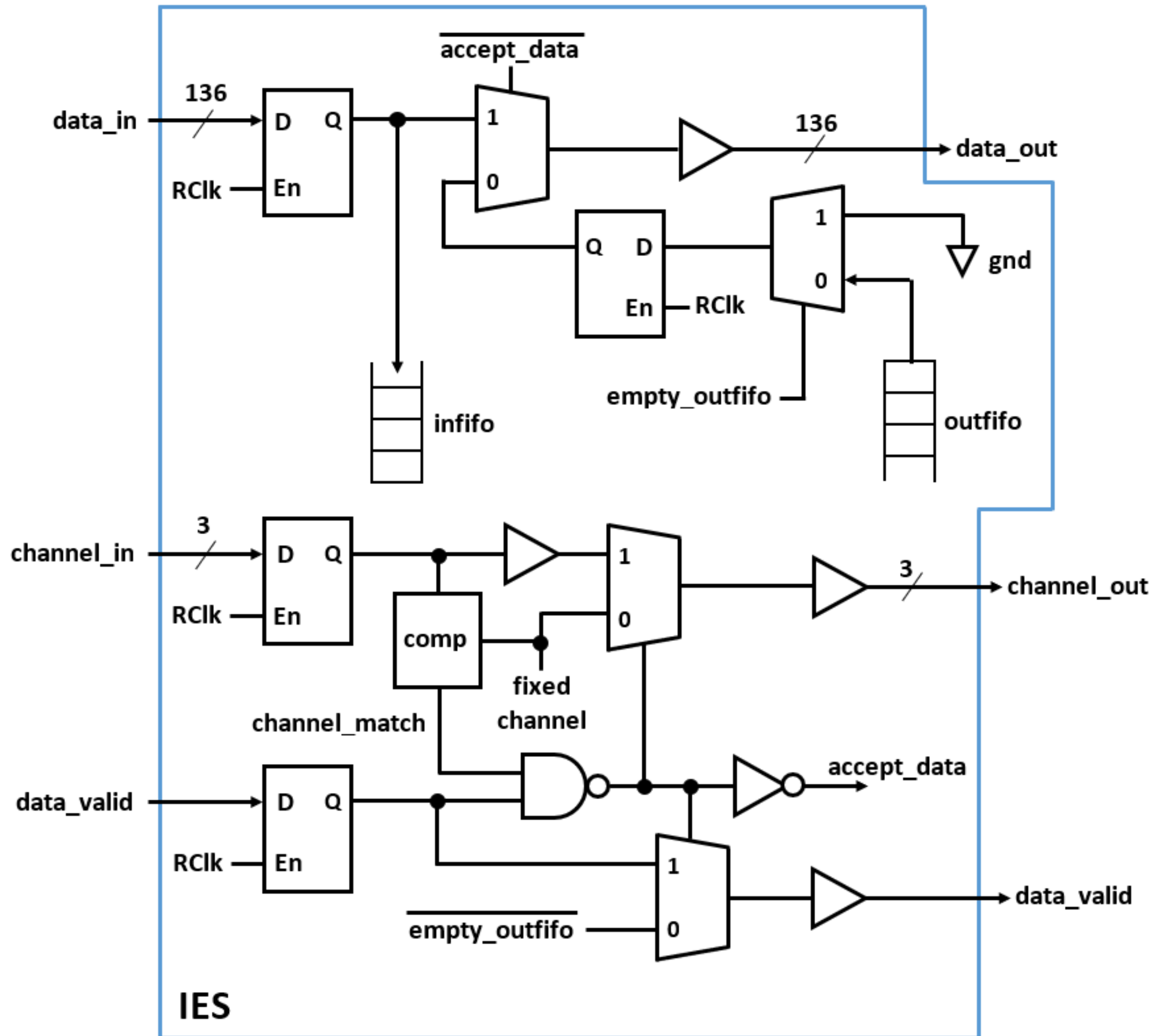

Figure 3.5: IES logic design for data and channel lines

Each IES uses a series of asynchronous FIFOs and multiplexers to communicate between the DRAM channels and the data ring. Figure 3.5 shows the design of the data and channel wires for each DRAM IES. Data enters each IES at the data_in port and continues to the next IES at the

data_out port. This data_in signal consists of the 128-bit wide data bus (HBM also uses 128 data bits per channel) and the 8 redundant data bits. By sharing these lines between all eight channels, the number of TSVs used in the MARS decreases significantly, reducing power consumption and manufacturing costs. The infifo represents an asynchronous FIFO [2] that will only accept data from the ring if accept_data is high. The data stored in this infifo will then be withdrawn by the DRAM channel at the channel's I/O clock rate. Outfifo is an identical asynchronous FIFO that takes data from the DRAM channel and inserts it onto the ring when there is available data *and* when the current slot in the ring is empty. This ensures that the pending DRAM data does not corrupt valid memory commands that are being sent to channels further downstream.

Channel_in indicates which channel the logic die is communicating with. This is a three-bit bus that operates in a typical binary encoding fashion to choose one of the eight DRAM channels to communicate with. The channel_in value is compared against the fixed channel code corresponding to that channel in order to determine if the given memory command on the ring is destined for this channel (indicated by channel_match). The data_valid signal will be high if the given slot in the ring contains valid data. By comparing this signal with the channel_match, we generate the accept_data signal which dictates if the given data in the ring is destined for the current DRAM channel. When the given DRAM channel wishes to return data from a read command, it places the data in the outfifo. If the empty_outfifo signal is low, which allows the data to be sent out on data_out when a free slot is available in the ring (i.e. a cycle in which ring data is not valid). In addition, the channel_out signal will be set to the current fixed channel code to let the IES on the logic die know which channel the data is coming from, and the data_valid would be set to high.

Each IES is clocked by the outputs of the clock recovery circuits of the RRSWO. This is indicated by the RClk signal in Figure 3.5. Because the ring is clocked (by RClk) at $18GHz$, the propagation delay must be minimized to ensure that the signals can travel between IES'. This includes accounting for potential clock skew, propagation delay and logic delay of the IES'. Table 3.1 shows the breakdown of the IES delay, the propagation delay between IES' and the setup/hold time of the IES registers. Here we accounted for VDD jitter and process variations through SPICE

simulations and added some overhead in the timing. With a 55ps clock period, we ensured that all signals were ready at the next IES 19.72ps after being launched by the current IES (yielding a 5.96ps guard band).

Clk to Out Delay	Setup/Hold Time	IES Delay	Propagation Delay	Total
4.03ps	2.12ps	23.17ps	19.72ps	49.04ps

Table 3.1: IES versus propagation delay

Figure 3.6 shows the IES logic for the row address lines. Because commands are executed in the order in which they are sent to each DRAM channel, there is no need to return address information to the logic die. In other words, address values are simply sent to the DRAM channel they correspond to, but the DRAM channel does not need to insert an address into the ring. This significantly simplifies the logic, reducing propagation delay and the total power consumption of each IES. Row_in is 7 bits wide to contain the 6 bit row address and 1 redundant bit (to achieve higher yield by means of post-manufacturing reconfigurability [66, 67]). This row address will always be presented to infifo but only will be accepted if accept_row is asserted. The row_valid signal is a single bit that signifies if the row address bus contains a valid row address in this ring slot. If this bit is high, and this is the expected channel (channel_match is high), accept_row will be asserted and row_valid will be set to 0, for downstream IES'.

The column address logic, which is shown in Figure 3.7, is identical to the row address logic that was presented in Figure 3.6. The only difference is that the column_in bus, which represents the column address, is 9 bits wide. This contains 8 bits for the address and 1 redundant bit for higher post-manufacturing yield. The column_out bus represents the column address moving to downstream IES'. The ability to have a separate row and column address bus allows for row and column addresses to be sent simultaneously to a given DRAM channel in order to increase memory command bandwidth on the ring.

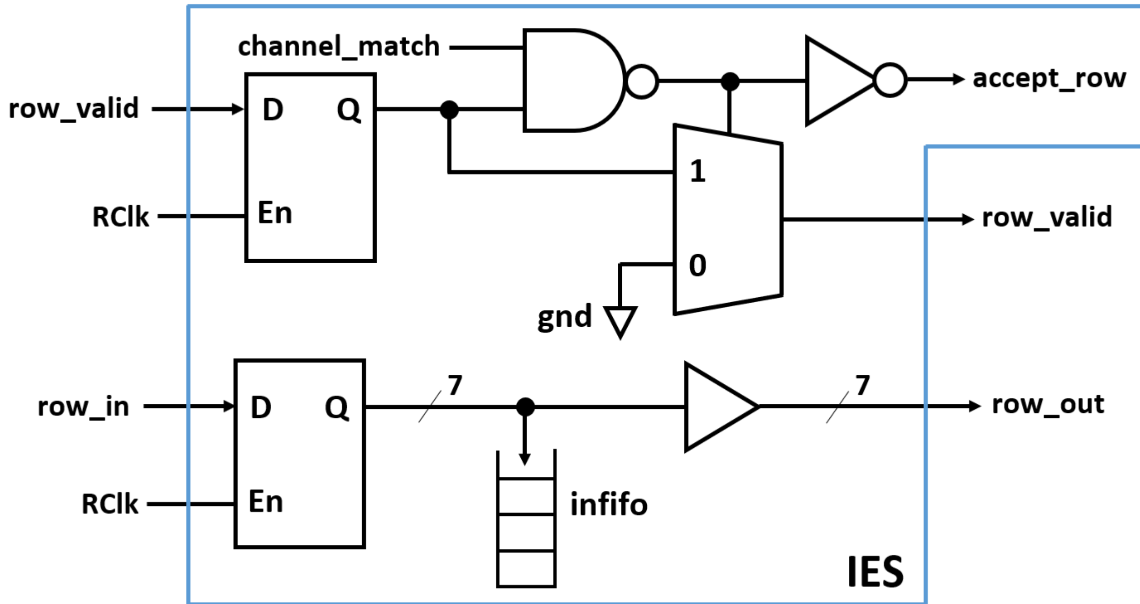

Figure 3.6: IES logic design for row address lines

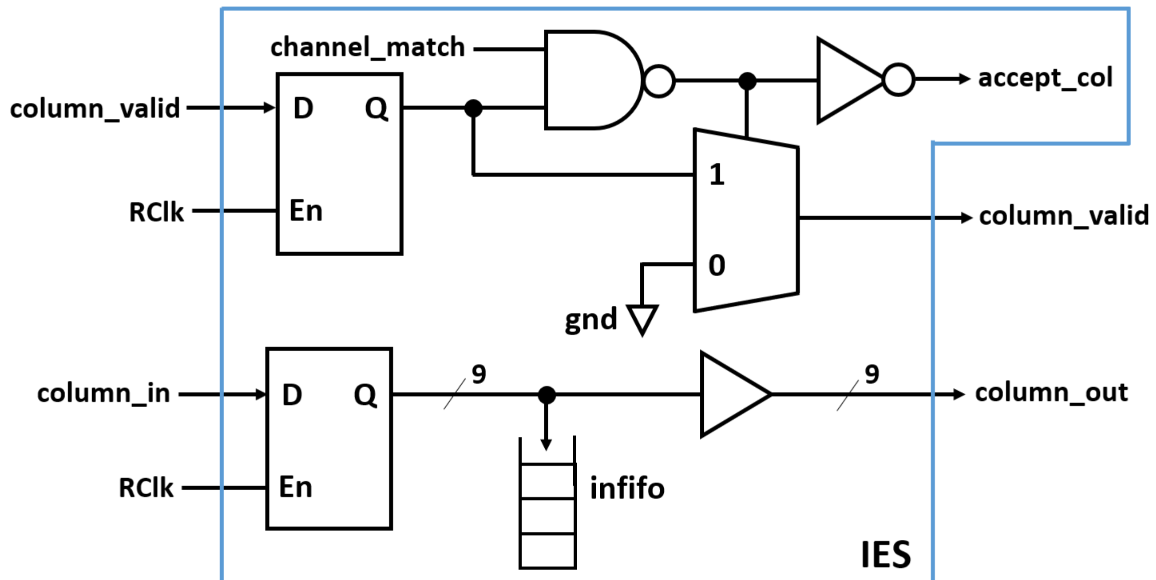

Figure 3.7: IES logic design for column address lines

In addition to the 158 bits that enter the IES in Figures 3.5, 3.6 and 3.7, there are another 32 bits that represent the data bit inversion and the data mask bits (16 bits each where every byte of the data bus has a data inversion bit and data mask bit). A data inversion bit is set to 1 to indicate

Memory Architecture	Number of TSVs
Wide I/O	872
HBM	1696
MARS8	402
MARS16	804
MARS32	1608

Table 3.2: Total number of TSVs

that the corresponding data byte is transmitted in a complemented form (for power reduction). A data mask bit is used to inform the DRAM which bytes on the data bus it should ignore when a write command is issued (allows data values that have a smaller width than the data bus to be stored). These inversion and mask bits behave like data_in in Figure 3.5, and they travel along the ring with the data they correspond to. Finally, there are the 2 clock wires of the RRSWO, 4 parity signals (1 per 32 data lines), 4 data error signals (1 per 32 data lines) to indicate when an error occurs on the data bus and 1 address error signal to indicate when an error occurs on the address bus. This gives us a total of 201 bits (or TSVs) for one side of the ring. However, because the ring is unidirectional, we must double the number of TSVs. This gives us 402 bits (or TSVs) for 8 channels in our proposed MARS. We can compare this to a traditional HBM stack which has a 212-bit bus for each DRAM channel [68]. This means that the total number of TSVs needed for HBM is 1696 for the eight channel configuration which was shown in Figure 3.3. Table 3.2 shows the TSV counts for the eight channel HBM and the different MARS configurations. Note that MARSX represents our proposed MARS with X number of channels. It is important to note that the MARS does not need strobe lines because data is only transferred on the positive edge of the clock compared to using both edges in HBM. This is why the MARS bus is smaller than the bus width of a single channel in HBM (206 for MARS compared to 212 for a single channel in HBM). This shows the significance of our 3D RRSWO structure as it provides a low skew, low power, and fast clock to between all DRAM dies. Wide I/O only utilizes 8 channels and therefore has

approximately half the TSVs that HBM utilizes. The larger memory arrays used in Wide I/O, to reduce power consumption, require a larger address bus which gives Wide I/O 218-bits per channel compared to HBM's 212 bits.

To meet the aforementioned timing requirements, the logic delay in each IES must be minimized. The critical path in this IES logic is from channel_in, through the comparator, to channel_out in Figure 3.5. This is why there is an additional buffer added between the channel_in latch and the multiplexer (to match the path delays and prevent glitches). The comparator utilizes a set of XNOR gates to compare each bit of channel_in with the fixed channel code and then uses an AND gate to generate channel_match (which determines if all bits in channel_in match the fixed channel code). The latches shown in Figures 3.5, 3.6 and 3.7 are D-latches built using CMOS transmission gates. This minimizes the delay seen by the registers, which includes the setup/hold time, but requires equivalent delays from all paths in order to ensure that shorter signal paths do not cause race conditions.

The infifo and outfifo represented in Figure 3.5, 3.6 and 3.7 are both asynchronous FIFOs used to communicate between the DRAM die and the ring-based interconnect. By utilizing the asynchronous FIFO from [2], we are able to ensure reliable data communication between the ring clock domain (Rclk) and the slower DRAM clock domain. Figure 3.8 shows the implementation of the asynchronous FIFO labeled infifo in Figure 3.5, 3.6 and 3.7. The signals prefixed with "R_" are clocked by the $18GHz$ RClk while those signals prefixed with "P_" are clocked with the slower DRAM die clock. The outfifo implementation is identical except that all signals prefixed with "P_" should be swapped with signals prefixed with "R_". The signal R_Write_en represents the accept_data signal from Figure 3.5, where the R_Write_en signal is synchronous with the RClk signal and the P_Read_en is synchronous with based on the DRAM die clock.

3.4.2 Experimental Results

The MARS design was tested using multiple simulation tools to obtain accurate estimates of power consumption and speed. HSPICE [50] was used to ensure that the clock and IES' operated at the desired frequency. The simulations in HSPICE used the 16nm PTM fabrication process [52].

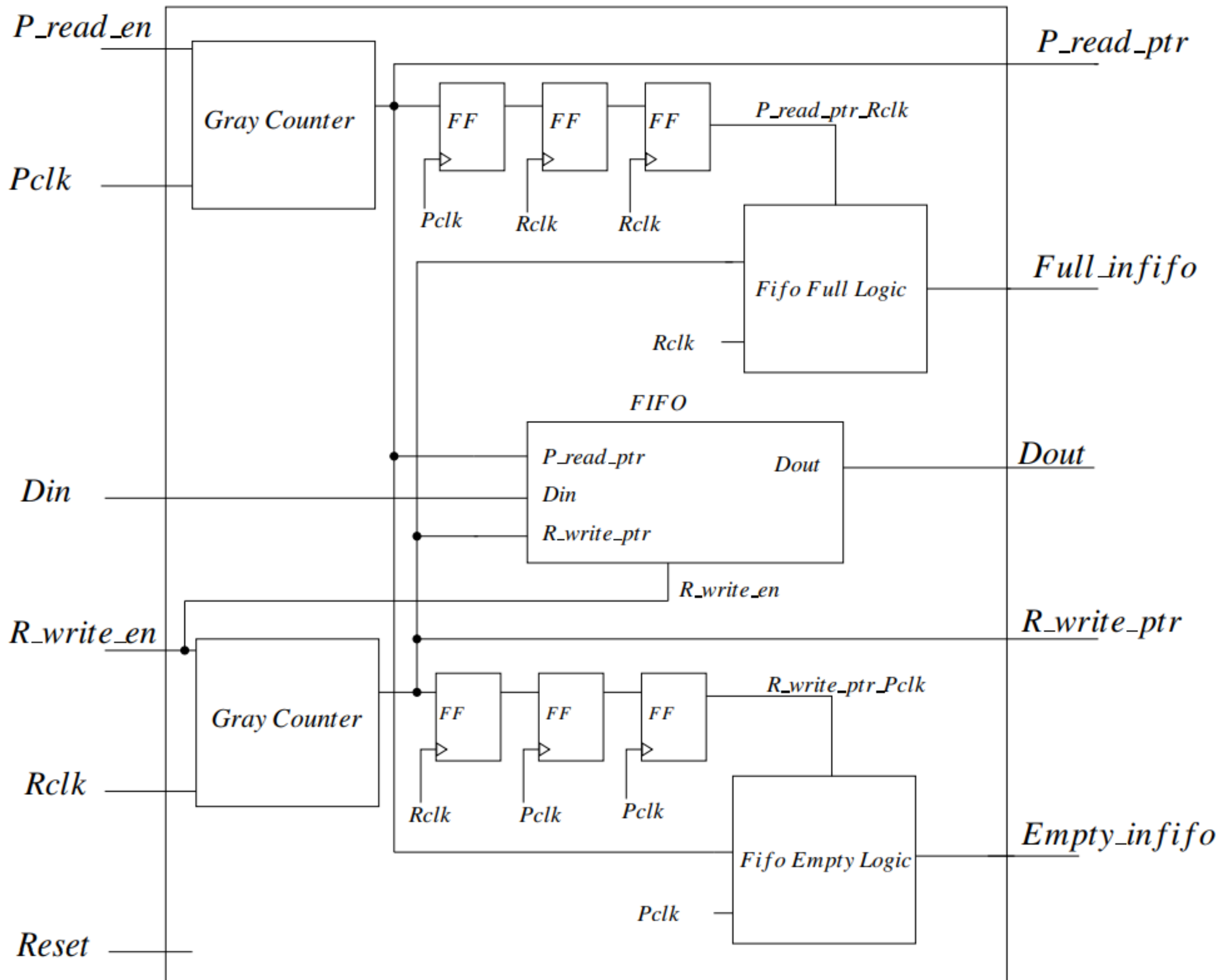

Figure 3.8: Input asynchronous FIFO [2]

Raphael [51] was used to generate 3D parasitics (capacitance, inductance, and resistance) of the wiring. Ramulator [69] was used to measure memory latencies for different CPU traces. Though this program yields only an estimate of memory performance, it presents useful insight into the relative performance of different memory architectures. DRAMPower [70] was used to estimate power and energy consumption for different memory architectures. It performs these estimates based on current consumption statistics published by JEDEC.

3.4.2.1 3D Ring Clock

To model the 3D ring-based clock, we first extracted the capacitance, resistance, and inductance of the clock wires using Raphael as was done in Chapter 2. A distributed model of the clock wiring parasitics was constructed in HSPICE. This distributed model utilized 500 T-sections with a single inverter pair. According to JEDEC, the typical height of a 2GB, 4GB, and 8GB HBM stack is $720\,\mu\text{m}$ [3]. Based on this knowledge, we chose the length of the ring perimeter to be at least $2200\,\mu\text{m}$ for two reasons. First, the ring is at least double the height of the stack because it must travel up the stack and then down as well. Along with the clock wires, the memory commands and data also travel up and back down the stack to reach all DRAM channels. Second, the extra trace at the top of the stack must be at least 33% of the total ring length to accommodate the clock recovery "dead-zone". This left $1474\,\mu\text{m}$ (67% of $2200\,\mu\text{m}$) of the ring capable of extracting a full differential clock, which meets the requirement for the stack height as well. The additional $34\,\mu\text{m}$ ($1474\,\mu\text{m}$ - $1440\,\mu\text{m}$) are used for local wiring on the logic die.

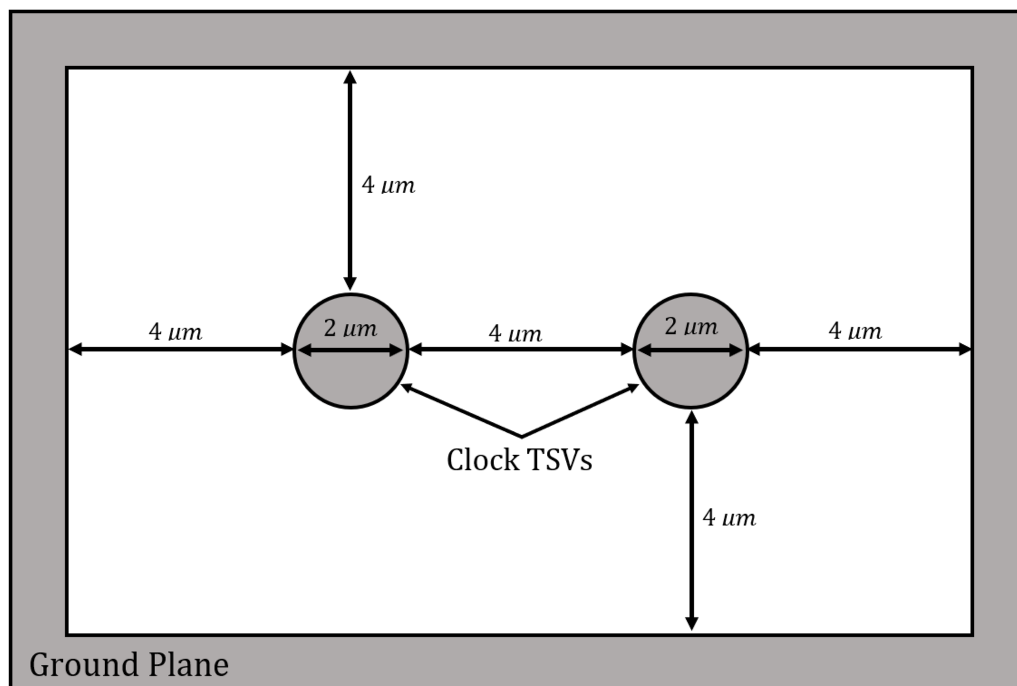

Figure 3.9: Cross section view of clock TSVs

The clock TSVs were surrounded by a ground shield, to provide an electrically controlled environment for all points through each DRAM die. To obtain $18GHz$ clock operation, the dimensions were chosen as shown in Figure 3.9. Here we can see that the diameter of the clock TSVs was chosen to be $2\,\mu m$ while the spacing between these TSVs was chosen to be $4\,\mu m$. The spacing between the clock TSVs and the ground plane was also set to $4\,\mu m$. The ground place is modeled as a rectangular cape and would be implemented as a series of adjacent TSVs.

Once we obtained the dimensions for the ring, we ran Monte Carlo simulations to show the stability of the RRSWO structure. For the threshold voltage, $\pm 3\sigma$ was taken to be about $\pm 6\%$ of the nominal threshold voltage. For the minimum channel length, $\pm 3\sigma$ was taken to be $\pm 10\%$ of the nominal minimum channel length [71, 72]. These numbers are reasonable given that the channel length of the inverter pair devices is chosen to be twice the minimum channel length (to reduce the impact of variations) and have large widths as well. Each simulation was run 30 times (this was determined based on total runtime considerations) and the results are shown in Table 3.3. Note that the inverter size was kept constant at $120\times$ the minimum size. Note that the maximum skew is within the assumed guard band of 5.96ps and the power of the RRSWO is at most $5mW$.

	Average	Min	Max	Std
Period	55.24ps	-2.8ps	+3.7ps	1.7ps
Skew	1.79ps	0.11ps	5.81ps	1.54ps
Power	3.65mW	-0.82mW	+1.42mW	0.69mW

Table 3.3: 3D RRSWO Monte Carlo

In addition to the Monte Carlo simulations, we simulated jitter on VDD to show that it has a minimal effect on our RRSWO. When cycle-by-cycle supply rail variations were introduced, the average clock period is within $\pm 0.5\%$ of the period obtained when using a fixed VDD. The minimum and maximum clock period vary up to ± 0.48ps. This is within 0.87% of the average clock period and is quite low. Average power consumption with cycle-by-cycle VDD variations remains

within ±0.3%, as compared to the values under a fixed VDD. Intra-ring skew is slightly higher (by $\pm 5.1\%$ respectively) when VDD is subject to cycle-by-cycle variations, which is expected considering the higher clock period variations. This jitter, however, is within our assumed guard band of 5.96ps.

With long-term VDD supply rail variations, the average clock period remains within $\pm$0.4% of the period obtained with a fixed VDD. This is significant because most oscillators have a strong dependence of frequency on supply voltage. On the other hand, the oscillation frequency of our RRSWO is substantially dependent on the LC parasitics of the ring wires, hence its period barely changes with supply changes. The clock period variations decrease compared to the clock period variations in cycle-by-cycle supply voltage (by about 0.2ps). The skew for long term VDD changes is less than the skew cycle-by-cycle changes by about 0.73%. The results show here, as well as the results we presented in Chapter 2, are significant as it shows the resilience of our clocking and distribution scheme.

As discussed earlier, the RRSWO was tuned in HSPICE to generate a clock that oscillates at 18GHz. Clock extraction points were placed in the 67% of the ring centered around to the single inverter pair. These extraction points are located at $180\,\mu\mathrm{m}$ increments, coinciding with the location of the IES'. The differential amplifiers that extract a full amplitude square wave clock signal are located at these extraction points. The differential amplifiers, which are referred to as clock extraction circuits in Figure 2.1, use a DC bias transistor to control the tail current (Figure 2.2). A single stage differential to single-ended output is used with a current mirror active load. This single-ended output is then driven to an even number of inverters in a chain with increasing inverter sizes. This allows the extracted clock output to drive the desired capacitive load.

Figure 3.10 shows all the overlaid extracted clock signals from the RRSWO overlaid upon each other. These overlapping waveforms represent the output from all the clock recovery circuits. This signal is the RClk input to each IES in Figures 3.5, 3.6 and 3.7. Recall that this RRSWO is $2200\,\mu\mathrm{m}$ in length with an inverter pair size of $120\times$. The period of the extracted clock is 55.13ps, which equates to 18.14GHz. The average skew between the different IES' is 1.98ps with the minimum

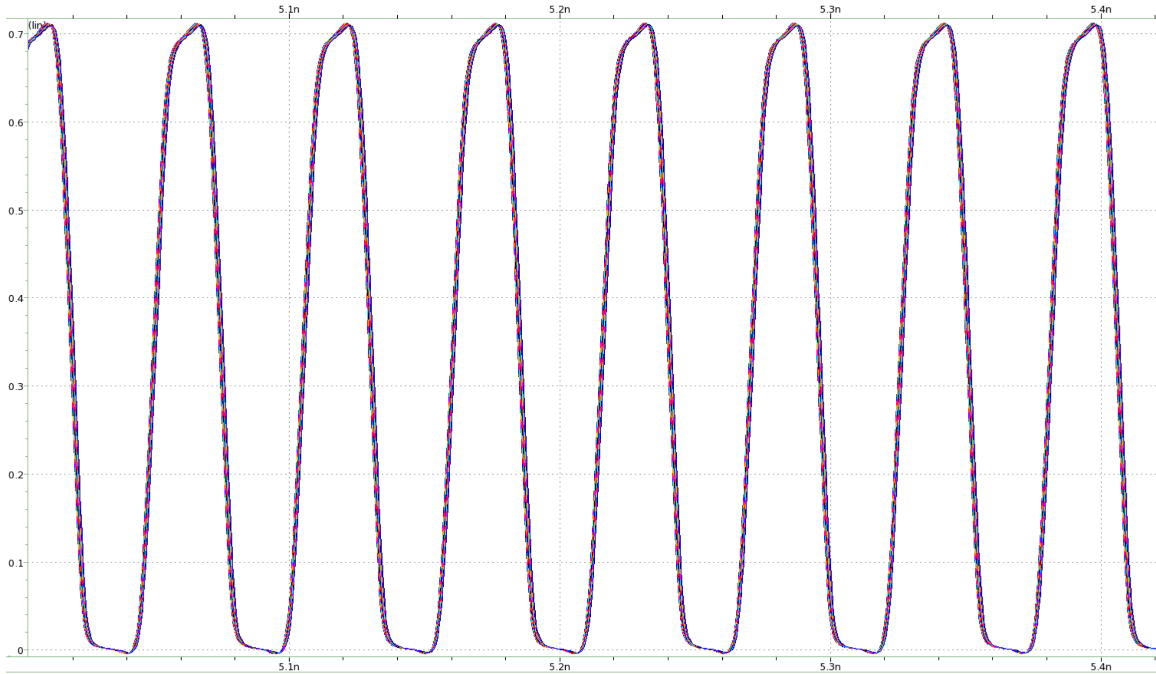

Figure 3.10: $18GHz$ ring clock signal before differential amplifiers along the 3D ring

skew being 0.34ps and the maximum skew being 3.18ps (under ideal conditions). This is $4.5\times$ lower than the skew minimization technique presented in [61]. The total average power consumed by the ring along with the 9 differential amplifiers was 2.83mW. The power consumed by the ring is minimal compared to the average power consumed by the DRAM cores. Based on these results, the RRSWO can deliver a low skew clock to all IES' and maintain a round trip time of 0.5 nanoseconds. This allows comparable speeds to current HBM standards but utilizes approximately $4\times$ fewer TSVs, when MARS uses eight channels.

3.4.2.2 DRAM Latencies

Utilizing the built-in models from Ramulator [69], several modern DRAM architectures (equivalently referred to as configurations in the remainder of this thesis) were simulated with 11 different CPU traces. These DRAM architectures are commonly found in modern computing systems today. Each DRAM architecture was run using multiple channels to accurately represent its fastest configuration. DDR memories have a maximum of 8 ranks per channel since capacitive loading makes more ranks infeasible [73]. The 3D stacked memories, HBM and Wide I/O, had 8 channels

and 4 channels respectively based on the JEDEC standard. Table 3.4 shows an overview of the configuration settings for each DRAM architecture that were simulated.

DDR Arch	Number of Channels	Ranks per Channel	I/O Bus Speed (MHz)
DDR3	2	8	2133
LPDDR3	2	2	2133
DDR4	4	4	2400
LPDDR4	2	2	3200
GDDR5	6	1	5000
WideIO	4	2	266
HBM	8	2	1000

Table 3.4: Ramulator DRAM configurations

Based on the configurations shown in Table 3.4, Ramulator provides the theoretical maximum bandwidth that is achievable. This parameter is typically used by memory manufacturers to market their DRAM architectures. Though a configuration may have a higher theoretical bandwidth than another, this does not guarantee better performance than the other configuration for all applications. Table 3.5 shows the theoretical maximum bandwidth for each configuration that was run with Ramulator. Though GDDR5 has the highest theoretical bandwidth compared to other commercially available DRAM architectures, our results demonstrate that it is not necessarily the fastest architecture among the commercially available architectures when running different CPU traces. As expected, the low power memory architectures have lower theoretical bandwidth. It is important to mention that those memory architectures that have higher channel counts offer higher theoretical maximum bandwidth.

Table 3.6 shows details of the traces we simulated, including total requests, the percentage of those requests that were write commands, and the cache hit rate for every trace that was simulated. These traces are based on a variety of software programs and provide a diverse battery of DRAM

DRAM Architecture	Maximum Bandwidth (GBps)
MARS32	1024
MARS16	512
GDDR5	160
HBM	128
MARS8	128
DDR4	76.8
DDR3	34.1
LPDDR3	25.6
LPDDR4	25.6
WideIO	17.1

Table 3.5: Theoretical maximum bandwidth

benchmarks. It is important to note that with a higher cache hit rate, fewer commands were issued to the DRAM subsystem. This will result in fewer bottlenecks on the DRAM architectures and cause lower latencies in the following graphs.

Every DRAM configuration was run with the cache enabled, to simulate a realistic computer system. This included an L1 and L2 cache with the sizes being 32KB and 512KB respectively. The associativity was set to 8-way and the block size was set to 64 bytes for both cache levels. Utilizing these cache parameters and the traces from Table 3.6, the different configurations were simulated in Ramulator. We first simulated current DDR technologies to form a baseline to compare our MARS against. It is important to remember that the DDR architectures from Table 3.4 represent the highest bandwidth configurations, to compare against our MARS (which is designed for high bandwidth applications).

Figure 3.11 shows the total read latencies for each memory architecture. This represents the summation of the latencies of all the read commands over each trace. The summation of the latencies of all read commands is representative of the effective bandwidth of each memory architecture for different applications. HBM, GDDR5, DDR4, and DDR3 were the fastest memory technolo-

CPU Trace	Total Num of Requests	Write Request %	Cache Hit %
429.mcf	11396914	10.1	57.3
433.milc	4200650	24.5	57.8
434.zeusmp	3123090	47.9	69.4
436.cactusADM	843180	23.4	54.7
437.leslie3d	4052550	26.3	57.9
450.soplex	2303025	34.7	85.2
459.gemsfdtd	5647830	31.5	64.1
462.libquantum	6447882	16.3	54.4
470.lbm	9988010	42.8	41.9
473.astar	1693787	38.9	63.3
482.sphinx3	742650	3.8	79.6

Table 3.6: CPU trace statistics

gies while those that focus on power efficiency, like LPDDR4, LPDDR3, and Wide I/O, were the slowest in terms of total latencies. We note that HBM had lower total read latencies for every trace that was run, and Wide I/O was the fastest among the power-efficient architectures. This is expected as HBM has significantly higher theoretical bandwidth as seen from Table 3.5 and has the highest channel count.

Another performance metric we studied was the per-command latency for each command that was issued to the DRAM (Figure 3.12). Although this metric can measure the average time a single command takes to execute, it does not accurately represent the parallelization of multi-channel architectures. Though Wide I/O and HBM seem to perform poorly on a per-command (or average) latency basis compared to their counterparts, their higher channel counts allow for significant reduction in overall latencies. Figure 3.12 also shows that between the low power memory options (LPDDR4, LPDDR3, and Wide I/O) exhibit a significantly higher per-command latency than the higher performance DRAM architectures (DDR4, DDR3, HBM, and GDDR5). Low power memory options use slower DRAM cores to reduce power consumption, resulting in

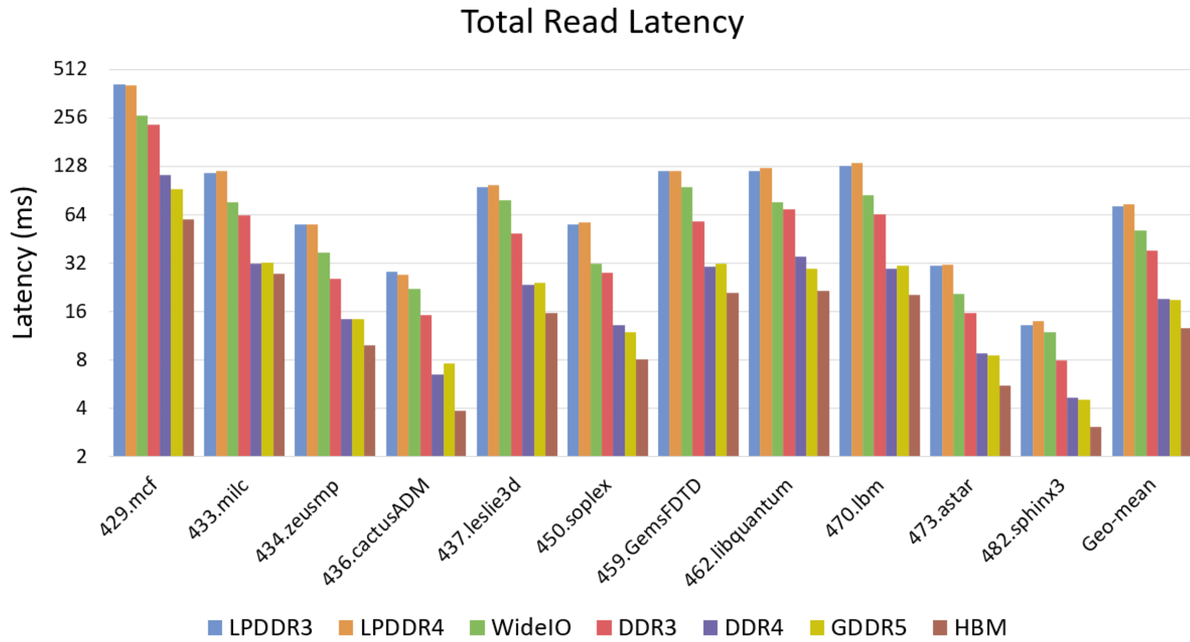

Figure 3.11: Total read latency for current memory architectures

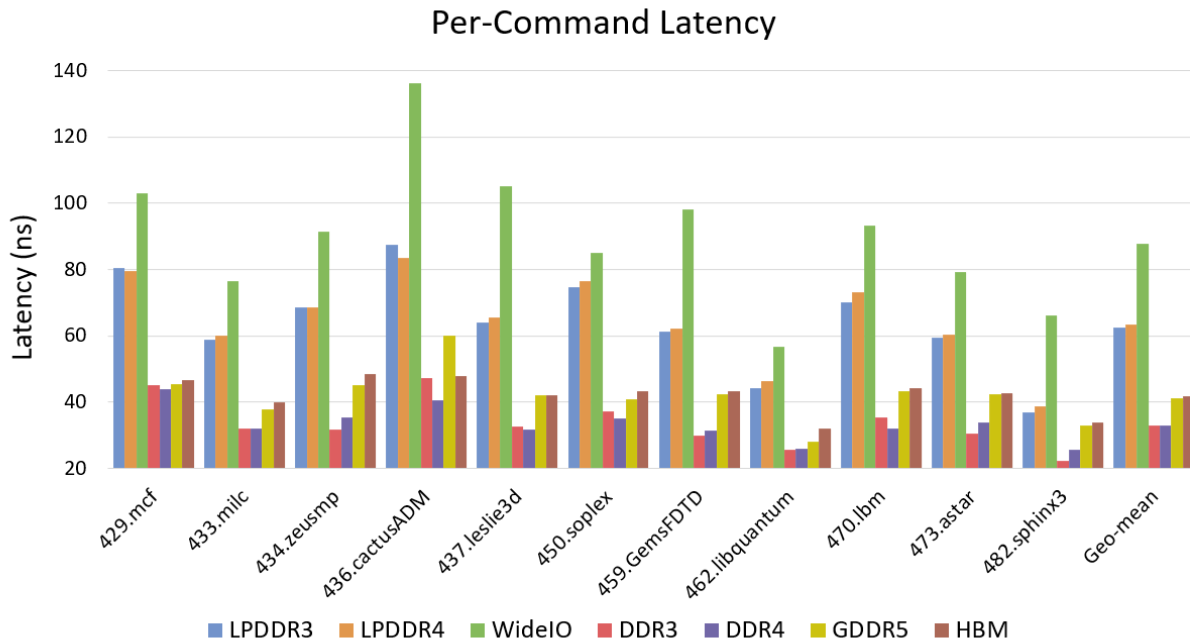

Figure 3.12: Latency per command for current memory architectures

increased per-command latencies.

Figure 3.13 shows the average queue length per channel for different DRAM architectures. The average queue length is a good representation of the parallelization achieved by using multiple channels. This shows that channel count directly affects the average length of the queue since commands will be divided amongst different channels when possible. The longer the average queue length, the longer the latency for the command to be issued to in the memory. Our experiment accounts for refresh commands, which increase queuing bottlenecks as the number of rows to be refreshed grows. As shown in Figure 3.13, the memory technologies with fewer channels (DDR3, LPDDR3, and LPDDR4) have at least $2.2\times$ larger queue lengths compared to memory technologies with higher channel counts (DDR4, GDDR5, WideIO, and HBM). The number of queued commands grows when more frequent requests are issued to the DRAM. This explains the higher average queue length for traces that have more memory requests. Finally, we note that both HBM and Wide I/O have among the smallest queue lengths. Since our MARS approach utilizes high channel counts, we expect the depth of the asynchronous FIFOs in Figure 3.5 to be low as well.

We compare MARS to GDDR5, Wide I/O and HBM since they lead the existing memory architectures in terms of performance and queue lengths respectively. Using the GDDR5, Wide I/O and HBM schemes as a reference, the MARS model (with the fast ring bus) was studied with varying numbers of channels. The three previously mentioned MARS configurations (MARS8, MARS16, and MARS32) were tested for both power consumption and latency.

MARS8 is designed to compete with Wide I/O, to focus on low power while achieving fast memory speeds. The two major differences between MARS8 and Wide I/O is that MARS8 uses a 128-bit wide data bus and 8 channels compared to Wide I/O's 128-bit data bus with 4 channels.

The MARS16 and MARS32 configurations are targeted to compete with HBM, to achieve lower latencies. This involves stacking the DRAM dies to 8-high and 16-high respectively as was shown in 3.4. It is important to note that though 16-high DRAM stacks are not currently available, we propose the 32-channel option to demonstrate the scalability of the MARS paradigm. MARS

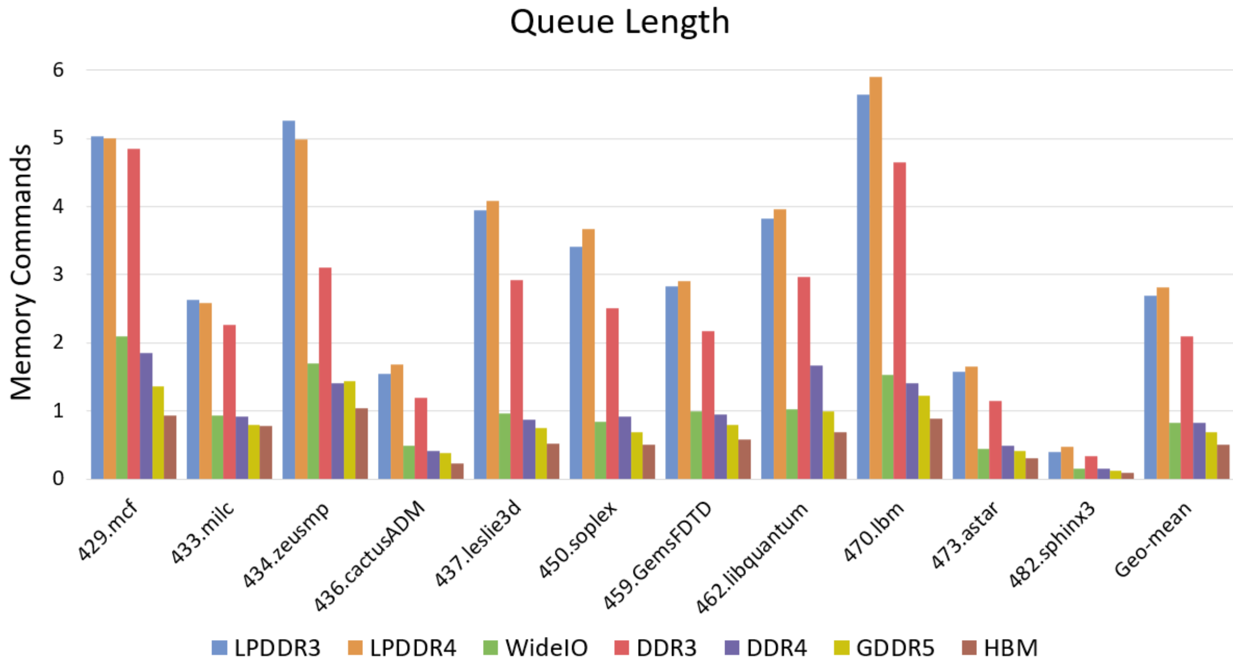

Figure 3.13: Queue length per channel for current memory architectures

has the ability to meet the aggressive stacking configurations that are anticipated in the future, something that HBM cannot support due to the number of the associated TSVs (and the large fraction of the die that they would occupy) and their capacitance.

Figure 3.14 shows the total read latencies for the different memory architectures. This figure presents the three different MARS configurations, as well as GDDR5, HBM and Wide I/O for reference. For every trace, MARS32 had the lowest total latency followed by MARS16, HBM, MARS8, and Wide I/O respectively. This is expected since the extra channels present in MARS32 and MARS16 results in higher throughput over traditional HBM. This is despite the fact that MARS32 uses approximately the same number of TSVs are HBM (and MARS16 uses half as many). MARS8 remains faster than Wide I/O (which it's designed to compete with) because MARS8 uses 8 channels (compared to Wide I/O's 4 channels). Note that the number of TSVs of MARS8 is 2.2× lower than Wide I/O. Also, the y-axis of the graph uses a logarithmic scale and that the set of bars on the far right represent the geometric mean across the simulated traces.

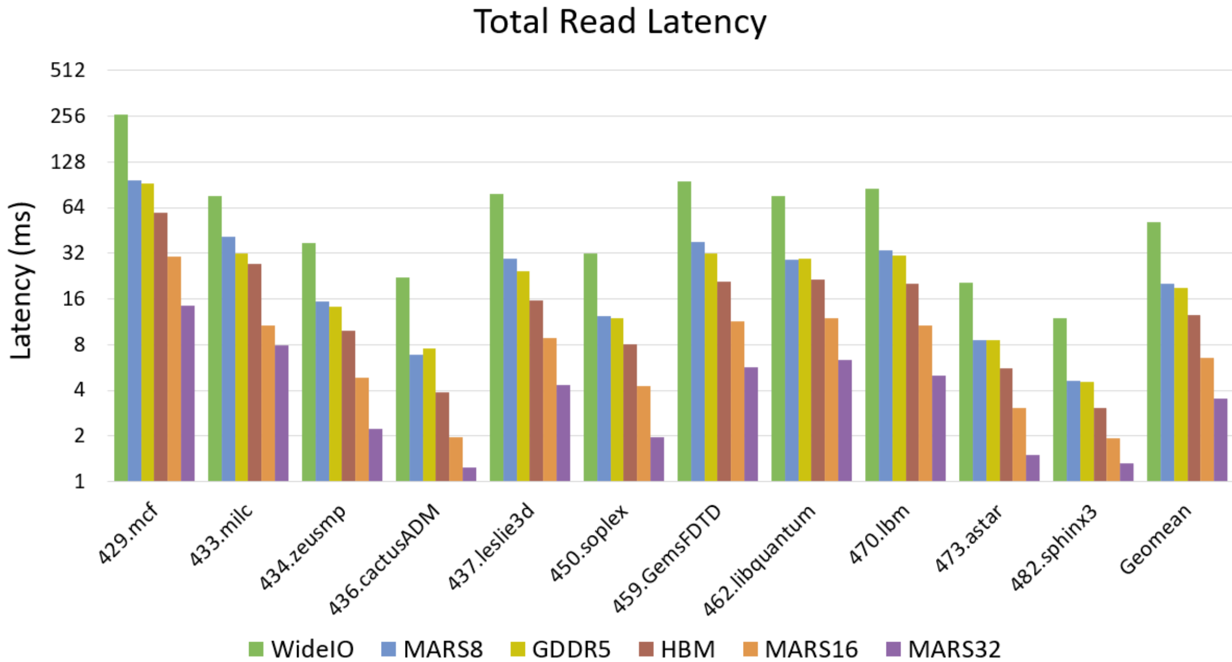

Figure 3.14: Total read latency for 3D memory architectures

The total read latency experiment shows how more channels increases throughput by taking advantage of concurrent command operations. Figure 3.15 shows significantly lower per-command latencies for HBM, MARS32, and MARS16, as compared to Wide I/O and MARS8. Wide I/O and MARS8 have the highest per-command latency because they both use slower DRAM dies to reduce power consumption. Of these architectures, MARS8 has much lower per-command latencies because of its increased channel counts which prevents memory commands from being bottlenecked. GDDR5 has the fastest per-command latency because it uses the fastest DRAM chips to achieve higher bandwidth (since the limited PCB space prevents GDDR5 from increasing channel counts). All other memory architectures had similar per-command latencies since they are all based on the same DRAM dies that are used in HBM. For these architectures, increasing throughput can only be achieved by using more channels, since the per-command latencies are similar.

Figure 3.16 shows the queue length per channel for the three MARS architectures as well as

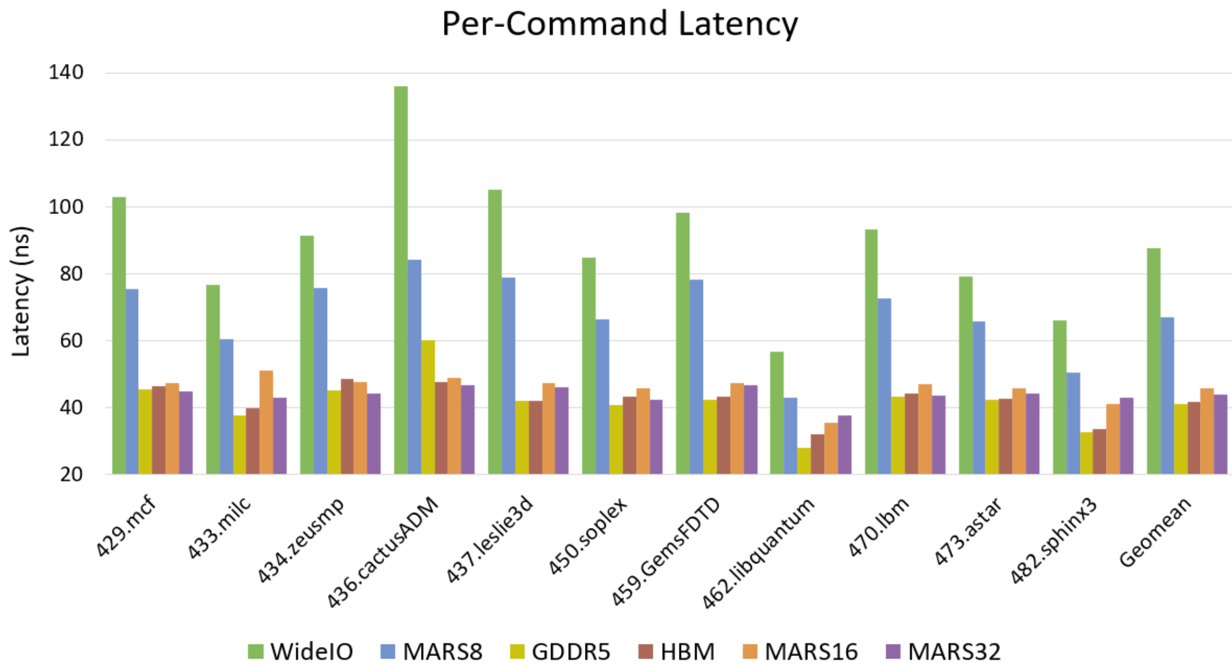

Figure 3.15: Latency per command for 3D memory architectures

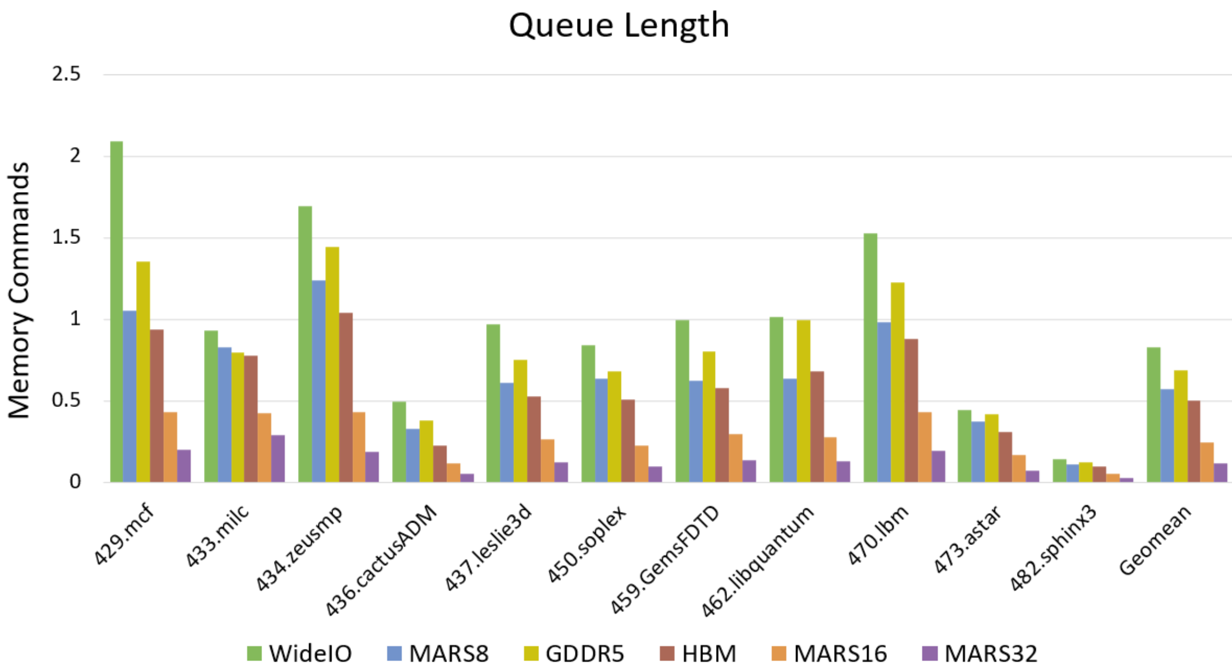

Figure 3.16: Queue length per channel for 3D memory architectures

GDDR5, HBM and Wide I/O. As expected, higher channel counts result in smaller queue lengths per channel because commands can be distributed across more channels. This relationship is substantially linear, as MARS8 and HBM have double the queue lengths as MARS16, which in turn has double the queue lengths compared to MARS32. GDDR5, which utilizes 6 channels, has queue lengths in between MARS8 and Wide I/O. This is expected since MARS8 uses 8 channels and Wide I/O uses 4 channels. The slightly higher queue lengths of MARS8 (compared to HBM), is a result of the slower DRAM cores whose reduced speed causes the queue to get backed up.

Based on the results presented in Figures 3.14, 3.15, and 3.16, we note that MARS can be configured to achieve higher throughput with the addition of more channels. These higher channel counts are possible because of the ring, which significantly reduces the number of TSVs needed for a typical 8 channel architecture. The same channel counts are not feasible in HBM and Wide I/O since they would result in impractically high TSV counts and their associated electrical and floor planning issues. For 16 channels, two rings are used, each of which connects to 8 channels. The same holds for 32 channels where 4 rings are used. By using 32 channels, we show that MARS can easily scale to larger memory sizes and can continue to boost memory performance (or power consumption, as discussed in the next section).

In summary, Figures 3.14, 3.15, and 3.16 show that MARS16 and MARS32 respectively have better total (per-command) latencies by $\sim 2.0\times$ ($\sim 0.9\times$) and $\sim 4.2\times$ ($\sim 1.0\times$), than HBM, which they are designed to compete with. Similarly, MARS8 has better total (per-command) latency, by $\sim 2.7\times$ ($\sim 1.6\times$), then Wide I/O, which it is designed to compete with.

3.4.2.3 DRAM Power Consumption

Ramulator [69] provides an option to record a trace of all the commands issued to the DRAM when running a CPU program. This command trace can be fed into the DRAMPower [70] simulator to estimate power consumption for different DRAM architectures. Each command trace represented the commands per rank, so the total power consumption was calculated by taking the sum of the power used for each rank. The power consumption was also combined with the run time for each trace to find the efficiency (measured in terms of performance per Watt) of each memory

architecture. This provides useful comparisons between memory latencies and power usage of the different approaches. We first compare the power and performance per Watt of the existing schemes (Table 3.4), in order to establish a reference to compare MARS against.

Figure 3.17 shows how low power DRAM architectures, like LPDDR4, LPDDR3 and Wide I/O, consume significantly less power compared to HBM, GDDR5, DDR3, and DDR4. DDR4/DDR3 consumes up to $4\times$ more power because of their increased speeds but consume less than half the power of HBM or GDDR5. Although HBM and GDDR5 are the fastest memory technologies among the existing technologies we studied, the speed gains come at the cost of significantly increased power consumption. With higher power consumption also comes higher heat dissipation. This results in GDDR5 and HBM needing active cooling (using a fan) compared to the DDR technologies that are usually passively cooled on a DIMM. Note the logarithmic scale used to show the power (in milliwatts).

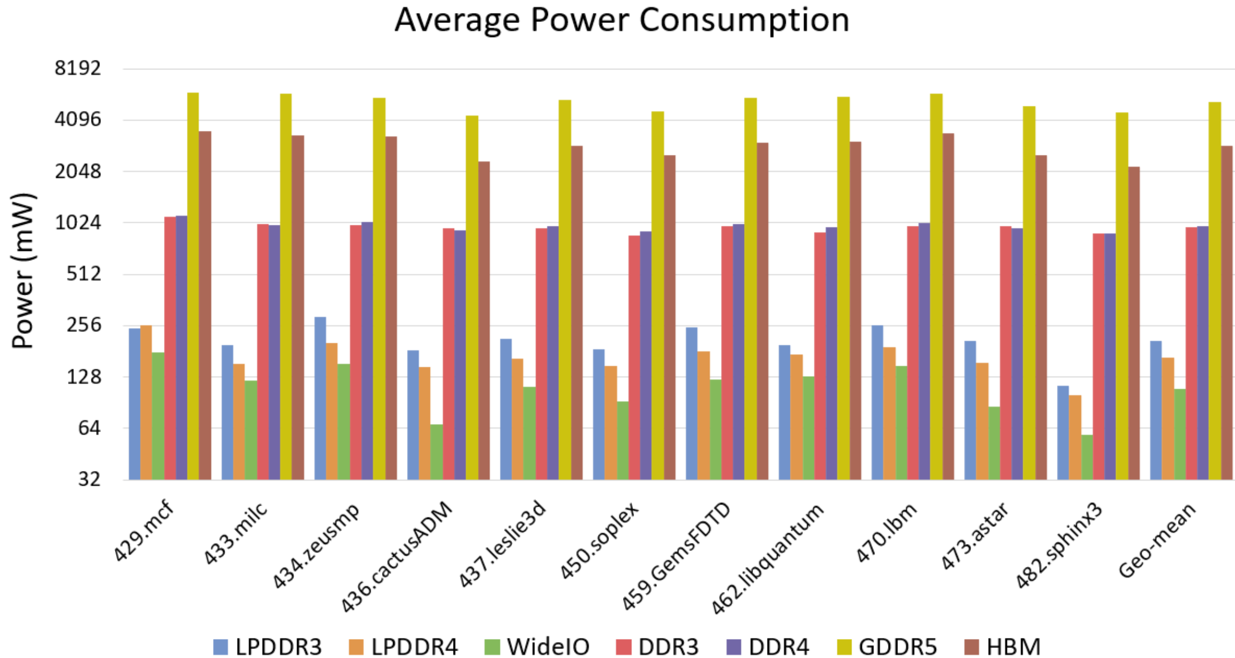

Figure 3.17: Average power consumption for current memory architectures

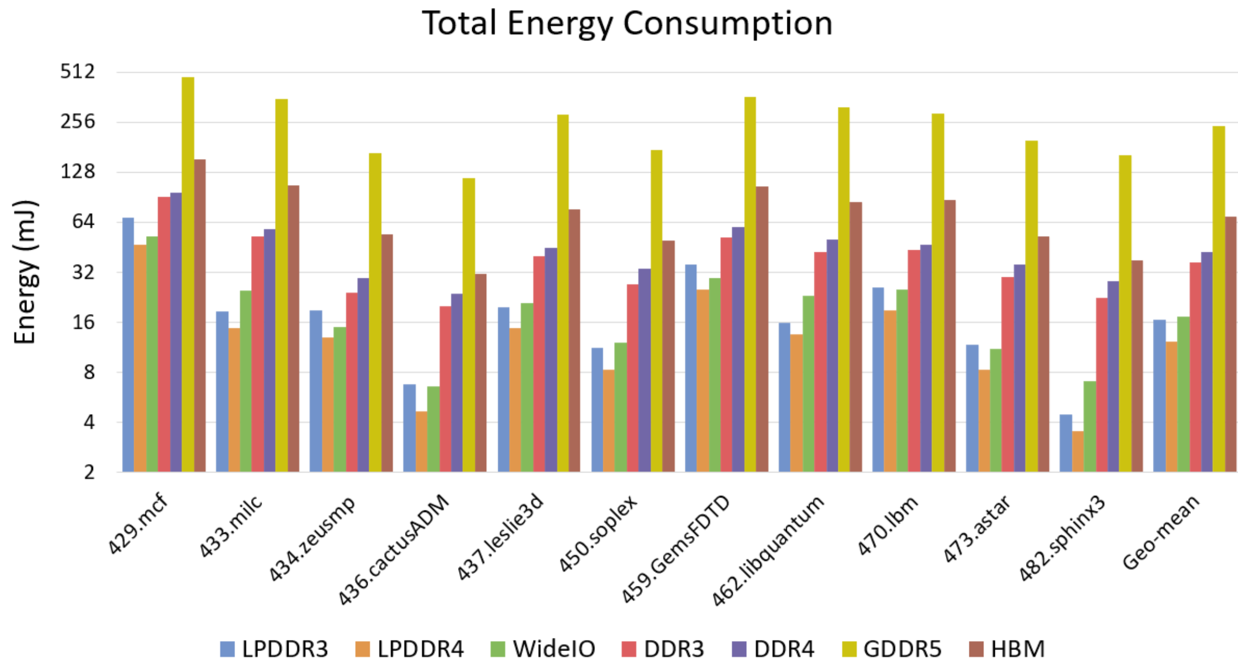

Figure 3.18: Total energy consumption for current memory architectures

The total energy consumption for the current memory architectures is shown in Figure 3.18. GDDR5 and HBM are the highest consumers of energy. However, the difference between the DDR3/DDR4 architectures and the low power architectures is significantly less. DDR4/DDR3 consumes approximately $2\times$ the energy (compared to $4\times$ the power) as LPDDR4/LPDDR3. This is because the low power architectures take significantly longer to process memory commands. GGDR5 consumes significantly more energy than HBM because it relies on faster memory chips instead of more channels to achieve its high bandwidth. Faster memory chips consume more power, but do not yield a proportional speedup, thus increasing the energy consumption of GDDR5 compared to HBM. As a result, HBM is shown to be a more effective means of achieving higher bandwidth since it's energy consumption is almost $4\times$ less than GDDR5.

Using the total power consumption and the read latencies, Figure 3.19 shows the efficiency for each existing memory architecture in terms of performance per Watt consumed. Wide I/O proves to be more efficient than any other memory for all CPU traces because of its low power consumption.

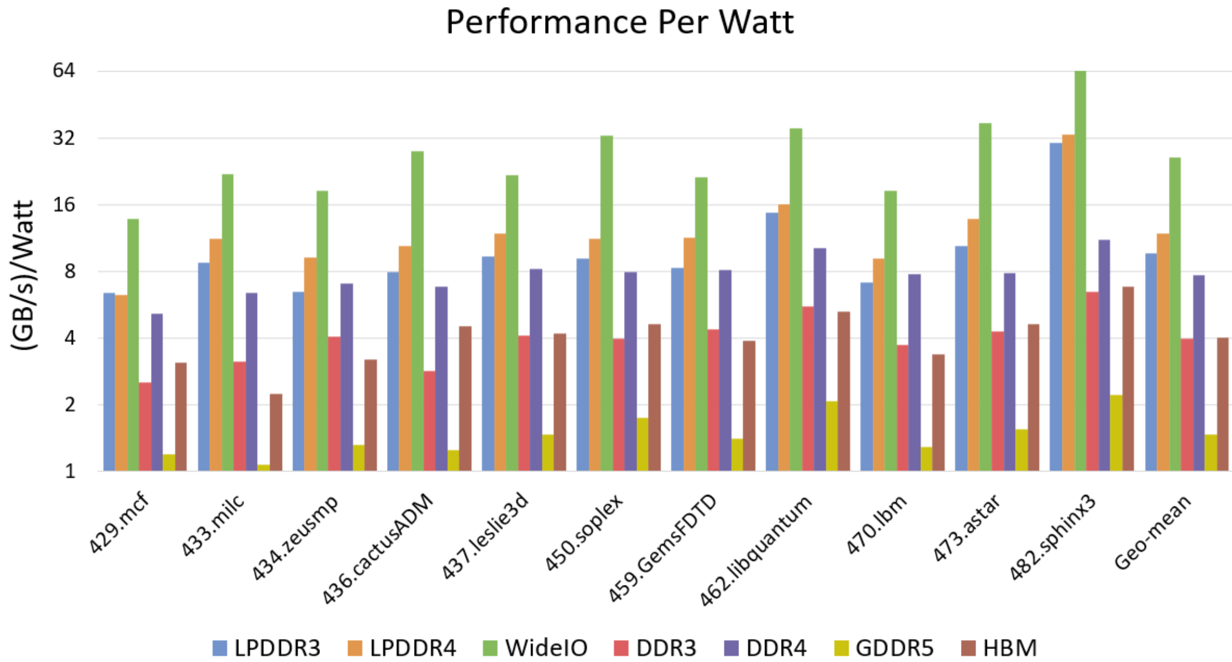

Figure 3.19: Performance per Watt for current memory architectures

Memories that consume more power, like GDDR5 and HBM, were not as efficient since their reduction in latency was less than the increase in power. DDR3 showed good performance per Watt (second only to Wide I/O) which is why its typically found in many consumer devices. We now compare the GDDR5, HBM and Wide I/O schemes with MARS, since GDDR5, HBM and Wide I/O represent the memories with the best performance and efficiency respectively.

By combining the power consumption obtained through DRAMPower with the power obtained for the ring-based interconnect using SPICE, we were able to estimate the total power consumption for our MARS architectures. Note that this includes the power used by the RRSWO, the IES', and the asynchronous FIFOs. The power consumption of the IES', RRSWOs, address and data lines, however, was minimal compared to the power consumed by the DRAM banks.

Figure 3.20 shows the average power consumption for the memory architectures (MARS8, Wide I/O, GDDR5, HBM, MARS16, and MARS32). The Wide I/O memory architecture has the least power consumption followed by MARS8 (are the two low power, high-bandwidth memories). The addition of IES', higher channel count and the higher switching frequency of the MARS

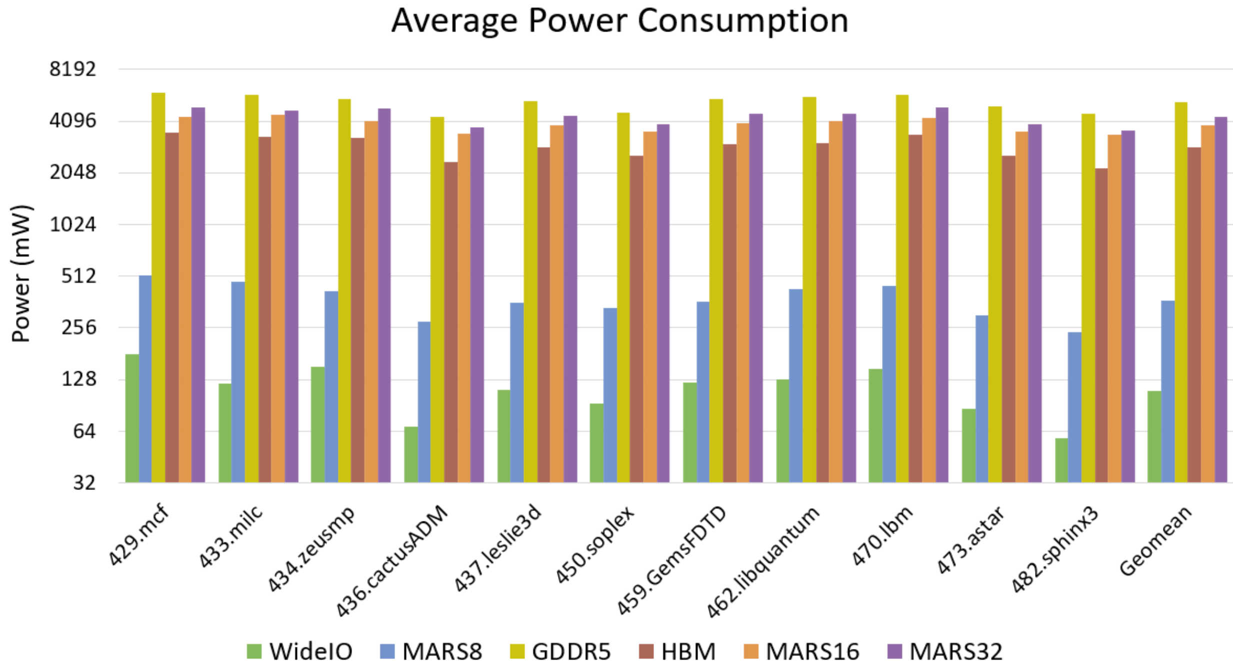

Figure 3.20: Average power consumption for 3D memory architectures

ring, resulted in a power consumption that was on average $2.8\times$ more than the power used by Wide I/O. Recall that the latencies of MARS8 are $\sim 2.7\times$ better than Wide I/O. We expect the power consumption of MARS8 to be reduced by reducing the number of channels or bus width. The power consumed by MARS8 was $\sim 6.9\times$ lower than HBM. MARS16 and MARS32 also saw small power increases over the HBM architecture because of the IES' asynchronous FIFOs and the RRSWO. The increase in power remained relatively small compared to the HBM power consumption because the power consumed by the IES' was smaller in proportion to the power consumed by the DRAM die. Note that the improvement in latency of MARS16 and MARS32 over HBM far outweighs the modest power increase.

Figure 3.21 shows the total energy consumption of the different MARS configurations compared to GDDR5, HBM and Wide I/O. As expected, Wide I/O has the lowest energy consumption because its power consumption is significantly less than the other architectures. The energy consumed by GDDR5 is the highest, followed by HBM. Even though HBM has lower power con-

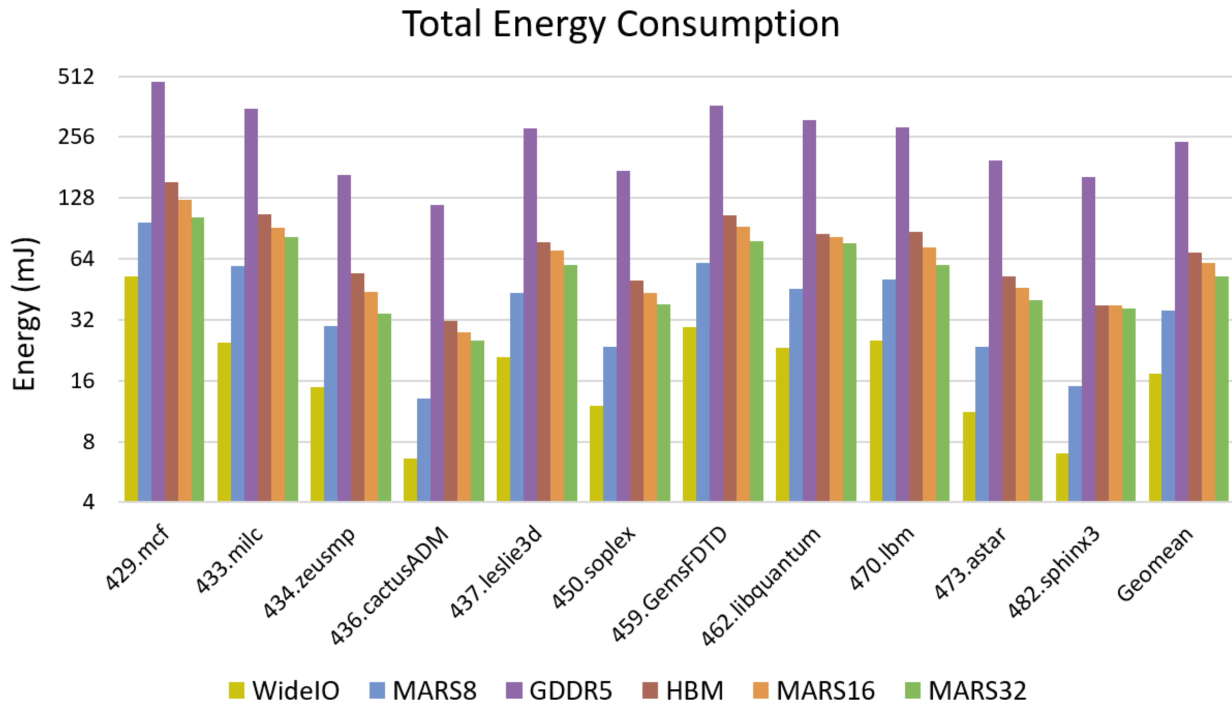

Figure 3.21: Total energy consumption for 3D memory architectures

sumption than MARS16 and MARS32, the higher channel counts of MARS16 and MARS32 yield much higher performance so that the energy consumed by MARS16 and MARS32 is lower than that of HBM. GDDR5 however, has a significantly higher power consumption relative to its performance, which results in increased energy consumption of $3.8\times$ compared to HBM. Again, note that the energy is plotted on a logarithmic scale.

The final statistic that was measured was the performance per Watt of the different memory configurations (Figure 3.22). GDDR5 performs the worst because of its significantly higher power consumption (as the memory chips are clocked significantly faster). HBM also performed poorly because of its high power consumption compared to the bandwidth it can achieve. The MARS16 and MARS32 had much better efficiency than HBM (by $\sim 1.7\times$ and $\sim 2.9\times$ respectively) because they can achieve significantly better speeds without a proportional increase in power consumption. Wide I/O and MARS8 performed considerably better than HBM, MARS16, and MARS32 in terms of performance per Watt. Despite the increase in power consumption for MARS8 compared to

Wide I/O, the use of 8 channels (compared to 4 in Wide I/O) and the larger 128-bit data bus (compared to a 128-bit data bus in Wide I/O) allowed MARS8 to be competitive with Wide I/O in terms of performance per Watt. This shows that MARS8 can provide a faster memory alternative than Wide I/O while being only slightly less efficient.

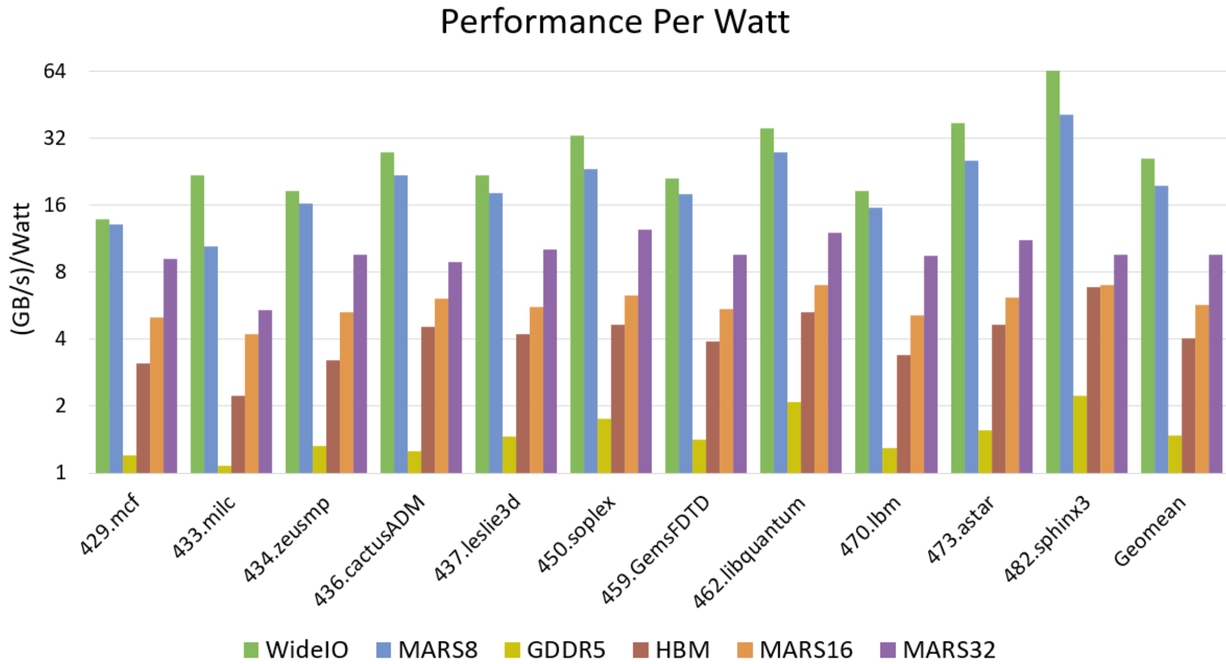

Figure 3.22: Performance per Watt for 3D memory architectures

To summarize Figures 3.20, 3.21 and 3.22, MARS16 and MARS32 consume slightly higher power than HBM (by $\sim 1.2\times$ and $\sim 1.4\times$ on average) while delivering much better performance per watt (by $\sim 1.7\times$ and $\sim 2.9\times$ on average). Similarly, MARS8 delivers similar performance per watt as Wide I/O although its power is $\sim 2.8\times$ higher on average. These numbers can be further optimized by varying the bus widths, clock frequency, and the number of channels for MARS.

Table 3.7 shows the power breakdown of the different MARS configurations. Note that the asynchronous FIFO row includes both the infifo and outfifo for all IES'. We can see that the dominant power consumer for all MARS configurations was the DRAM subsystem. It consumes

70.7%, 94.0%, and 94.1% of the total power consumption for MARS8, MARS16, and MARS32 respectively. As expected, the IES logic and asynchronous FIFOs are larger contributors to the total power consumption in the MARS8 configuration. This is because the DRAM dies are designed for low power and is one reason (along with the doubling the number of channels) that MARS8 consumes $2.8\times$ more power than Wide I/O. For the MARS16 and MARS32 configurations, the IES logic and asynchronous FIFOs consume only 5.6% and 5.7% of the total power consumption which allows us to achieve the $\sim 1.7\times$ and $\sim 2.9\times$ improvement in efficiency (GB per second per Watt) over HBM. Finally, the RRSWOs have minimal power consumption as they consist of only a single inverter pair.

	MARS8	MARS16	MARS32
RRSWO(s)	1.32	2.64	5.28
Asyn. FIFOs	41.45	82.71	122.24
IES logic	67.63	119.03	183.37
DRAM die	266.66	3286.51	4981.23

Table 3.7: MARS power consumption breakdown (mW)

3.4.2.4 DRAM Bus Utilization

In the previously described MARS topologies, we have attached 8 channels to each ring bus (Figures 3.3 and 3.4). However, when studying the bus utilization of different topologies, all architectures discussed (including MARS) had less than 7% utilization[†]. We, therefore, increased the number of channels attached to each ring bus to see its impact on performance. In Figure 3.23, MARSX-Y refers to X total channels in which each ring has Y channels attached to it. For example, the traditional MARS8, MARS16, and MARS32 discussed throughout this paper are

[†]Utilization is defined as the ratio of the number of valid slots in the ring to the total number of slots in the ring, over the duration of a trace.

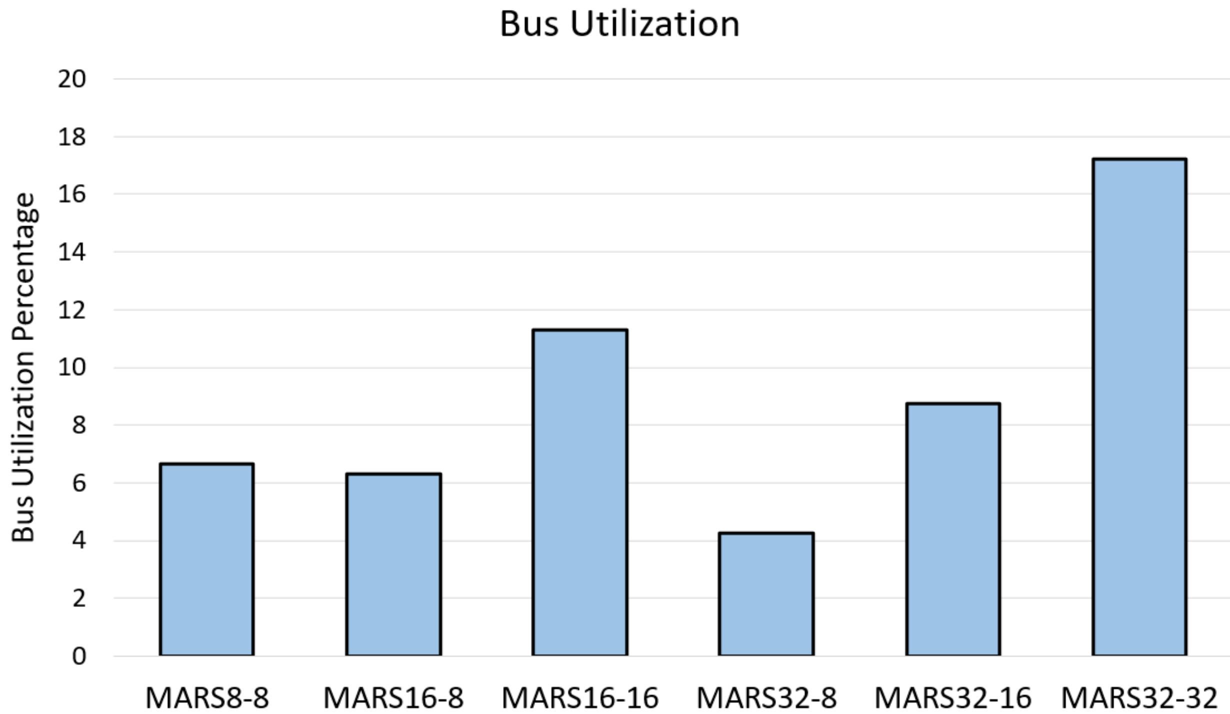

Figure 3.23: Bus usage for different MARS configurations

represented by MARS8-8, MARS16-8, and MARS32-8 (as each of these configurations have 8 channels attached to a ring). As expected, when more channels are attached to a ring the utilization increases almost linearly. This can be seen from the increase from MARS16-8 to MARS16-16 and the increase from MARS32-8 to MARS32-16 to MARS32-32. However, when the number of channels is increased, while keeping the number of channels per ring constant, we saw a slight decrease in bus utilization. This is expected as the memory commands are split amongst more rings in the memory stack.

Figure 3.24 shows the percentage decrease in performance as the number of channels attached to a single ring increases. The performance is based on the total read latency, which has been normalized to the MARS32-8 total read latency, averaged across the different SPEC traces. As we can see, the percentage of the ring that is being utilized increases as the number of channels per ring increases, creating a bottleneck to the ring interface with the DRAM channels. This causes the performance to decrease from MARS32-8 to MARS32-16 (by 4.8%) and MARS32-32 (by 14.7%).

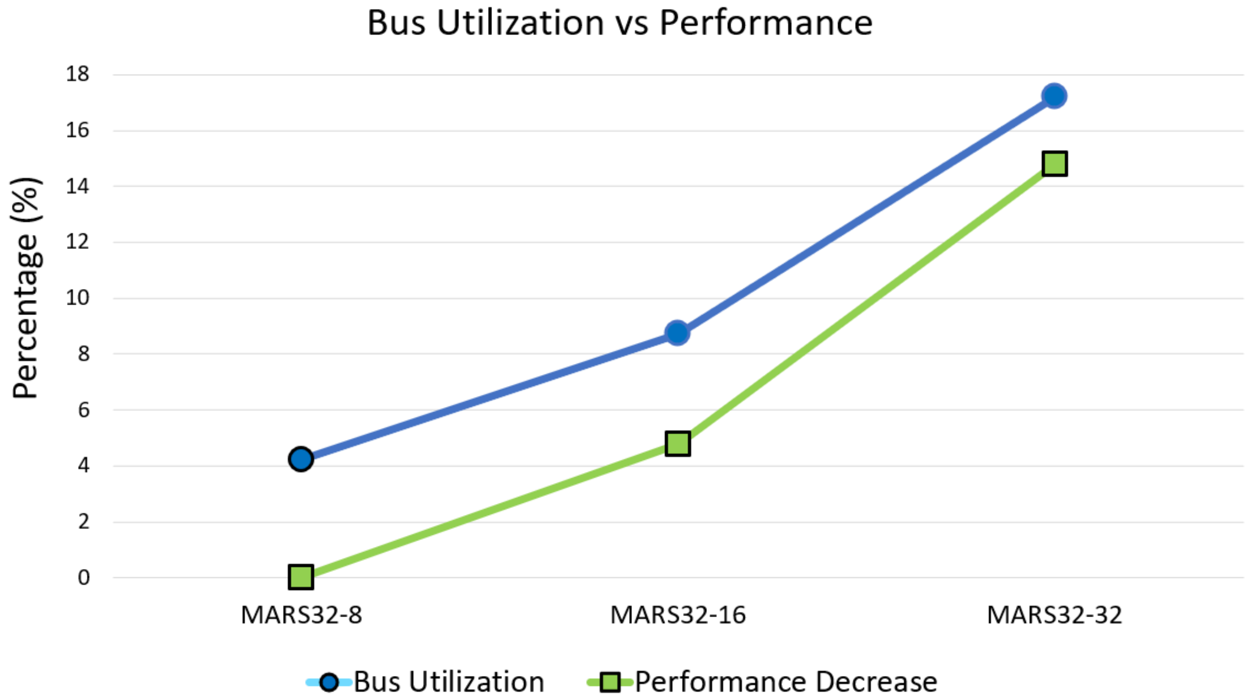

Figure 3.24: Bus utilization vs performance reduction for MARS

This decrease in performance roughly follows the increasing bus utilization, which is also shown in Figure 3.24.

3.5 Conclusion

In this chapter, we presented MARS as an alternative 3D stacked memory architecture to decrease the TSV counts while increasing the channel counts. MARS increases the memory bandwidth by using a fast ring-based data transfer scheme, which utilizes the RRSWO from the previous chapter. The fast data ring can service multiple channels in the same duration it would take to clock each channel individually, thereby reducing TSV counts significantly. By using the techniques shown in this chapter, we show that MARS can be used to create scalable 3D stacked DRAM, to meet the needs of next-generation high bandwidth applications.

The results from our simulations show that our proposed MARS architecture can overcome current memory bottlenecks by trading off power, throughput, and latency. MARS can reduce the

total number of TSVs required per channel compared to current 3D DRAM stacks. This allows us to pack more channels in the same footprint to increase performance with minimal impact on power and routing area. Since it has similar power consumption as traditional HBM architectures, the heat dissipation of MARS will be similar to HBM. We showed that MARS8 can increase throughput by $2.5\times$ over Wide I/O with only a 30% decrease in efficiency (with about $2.2\times$ less TSVs). By increasing the channel count to 16, we showed that MARS16 can achieve $1.9\times$ better throughput over HBM with $1.4\times$ better efficiency (with about $2.1\times$ the TSVs). Finally, we showed that MARS32 can achieve $3.6\times$ better throughput over HBM with $2.4\times$ better efficiency (with similar TSV counts).

4. SUMMARY AND CONCLUSION

In this thesis, we presented a new method for clocking in 3D IC stacks and showed how it can be used to synchronize independent RRSWOs in different IC stacks. This is done through the use of an RDL stub that allows a master RRSWO to "drag" a slave RRSWO to oscillate in phase with it. By utilizing an RDL stub we avoid the use of a PLL for synchronizing clocks, which simplifies logic, reduces power consumption and provides an inexpensive method for synchronization between ICs. In addition, we proposed the first 3D RRSWO bootstrap and reset circuit. Prior to this research it was assumed that inter-wire cross talk could be used to bootstrap this oscillation. In addition, we showed the 3D RRSWOs are robust to VDD jitter, long-term VDD changes, and process variations. This shows the resilience of our RRSWO scheme.

Utilizing these 3D RRSWOs, we also presented a ring-based data fabric for 3D stacked memory called MARS. Our MARS approach provides higher bandwidth over traditional stacked DRAM by using this fast RRSWO. It allows us to connect 8 channels to a common ring while maintaining round trip times similar to traditional 3D memory stacks. We showed how our MARS can reduce the total number of TSVs which allows for higher scalability for future 3D memory technologies. In addition, we showed how our MARS can trade off power, throughput and latency to match the requirements of different memory applications. With a narrow bus, and connecting it to all channels, the MARS8 can increase throughput by $2.5\times$ over Wide I/O with only a 30% decrease in efficiency (with about $2.2\times$ less TSVs). Using multiple ring topologies in the same stack, the channel count can double from 8 to 16, and then to 32. MARS16 (MARS32) can increase throughput by $1.9\times$ ($3.6\times$) over HBM with $1.4\times$ better efficiency ($2.4\times$). MARS16 also reduces the number of TSVs compared to HBM by $2.1\times$ (MARS32 has similar TSV counts to HBM). This scalable architecture allows higher throughput and faster system performance for next generation DRAM. The MARS topology proposed in this thesis can be used in a variety of computing systems, from mobile platforms to large-scale data centers.

4.1 Future Work

Our proposed MARS provides a platform for future research in 3D DRAM technologies and their I/O bus topologies. By varying the number of TSVs, channels and 3D RRSWOs, we show that MARS can accommodate different memory system requirements. In addition to main memory, future research can look into the use of the ring-based data fabric in other 3D memory-based architectures based on Intel's XPoint memory [74, 75], memristor-based memory [76, 77] and phase-change memory (PCM) [78]. These memory technologies will become critical in future computing systems as the demand for fast non-volatile memory increases. With more memory-based architectures scaling in the third dimension, the 3D ring-based architecture proposed in this paper can provide a fast and efficient communication scheme to improve system performance.

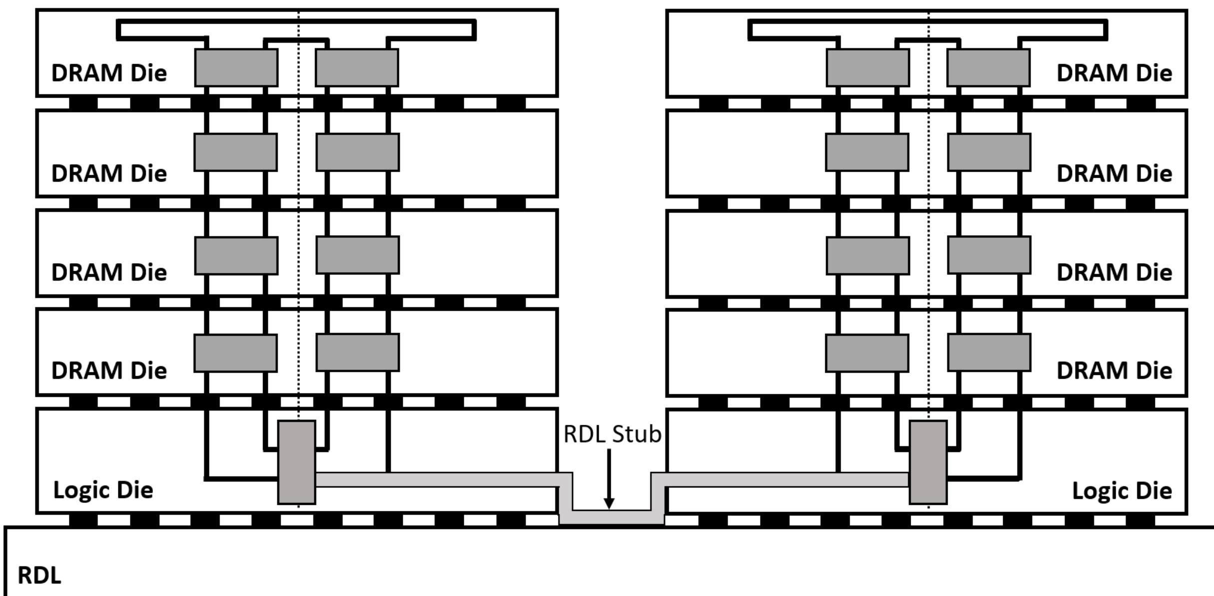

Figure 4.1: Synchronizing multiple MARS stacks using an RDL stub

In addition, we can scale the MARS by using the synchronization methods discussed in the previous chapter. This involves using an RDL stub between different 18GHz RRSWOs that allow DRAM channels in different memory stacks to be synchronized. Figure 4.1 shows two MARS

stacks (each of which are identical to the stacks on the right side of Figure 3.3). As a future experiment, it would be useful to see the performance, power and energy that can be achieved with multiple MARS stacks synchronized in this manner. By synchronizing the different stacks, we could allow for higher channel counts and reliable communication with the DRAM subsystem. This would decrease TSV counts and allow synchronization between the different memory controllers on different ICs. With synchronized memory controllers, we could develop more intelligent memory command scheduling and distribution between the different DRAM stacks to maximum performance and/or power consumption.

REFERENCES

[1] J. Dong, F. Zheng, N. Emmart, J. Lin, and C. Weems, "sDPF-RSA: Utilizing Floating-point Computing Power of GPUs for Massive Digital Signature Computations," in *2018 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 599–609, May 2018.

[2] A. Mandal, S. P. Khatri, and R. N. Mahapatra, "Exploring topologies for source-synchronous ring-based network-on-chip," in *Design, Automation and Test in Europe, DATE 13, Grenoble, France, March 18-22, 2013*, pp. 1026–1031, 2013.

[3] A. Shilov, "JEDEC Publishes HBM2 Specification as Samsung Begins Mass Production of Chips," 2016.

[4] P. Chakrabarti, V. Bhatt, D. Hill, and A. Cao, "Clock mesh framework," in *Thirteenth International Symposium on Quality Electronic Design (ISQED)*, pp. 424–431, March 2012.

[5] E. G. Friedman, "Clock distribution networks in synchronous digital integrated circuits," *Proceedings of the IEEE*, vol. 89, pp. 665–692, May 2001.

[6] J. Rosenfeld and E. G. Friedman, "Design methodology for global resonant H-tree clock distribution networks," in *2006 IEEE International Symposium on Circuits and Systems*, pp. 4 pp.–2076, May 2006.

[7] X. Jiang and S. Horiguchi, "Optimization of wafer scale H-tree clock distribution network based on a new statistical skew model," in *Proceedings IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 96–104, Oct 2000.

[8] W.-K. Loo, K. S. Tan, and Y. K. Teh, "A Study and Design of CMOS H-Tree Clock Distribution Network in System-on-Chip," 10 2009.

[9] W. Loo, K. Tan, and Y. Teh, "A study and design of CMOS H-Tree clock distribution network in system-on-chip," in *2009 IEEE 8th International Conference on ASIC*, pp. 411–414, Oct

2009.

[10] S. Tam, S. Rusu, U. Desai, R. Kim, J. Zhang, and I. Young, "Clock generation and distribution for the first IA-64 microprocessor," *Solid-State Circuits, IEEE Journal of*, vol. 35, pp. 1545 – 1552, 12 2000.

[11] M. A. El-Moursy and E. G. Friedman, "Exponentially tapered H-tree clock distribution networks," in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, vol. 2, pp. II–601, May 2004.

[12] X. Zhang, X. Jiang, and S. Horiguchi, "Variant X-Tree Clock Distribution Network and Its Performance Evaluations," *IEICE Transactions on Electronics*, vol. E90-C, 10 2007.

[13] N. Y. Zhou, P. Restle, J. Palumbo, J. Kozhaya, H. Qian, Z. Li, C. J. Alpert, and C. Sze, "Pacman: driving nonuniform clock grid loads for low-skew robust clock network," in *2014 ACM/IEEE International Workshop on System Level Interconnect Prediction (SLIP)*, pp. 1–5, June 2014.

[14] T. Xanthopoulos, D. W. Bailey, A. K. Gangwar, M. K. Gowan, A. K. Jain, and B. K. Prewitt, "The design and analysis of the clock distribution network for a 1.2 GHz Alpha microprocessor," in *2001 IEEE International Solid-State Circuits Conference. Digest of Technical Papers. ISSCC (Cat. No.01CH37177)*, pp. 402–403, Feb 2001.

[15] P. J. Restle, T. G. McNamara, D. A. Webber, P. J. Camporese, K. F. Eng, K. A. Jenkins, D. H. Allen, M. J. Rohn, M. P. Quaranta, D. W. Boerstler, C. J. Alpert, C. A. Carter, R. N. Bailey, J. G. Petrovick, B. L. Krauter, and B. D. McCredie, "A clock distribution network for microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 792–799, May 2001.

[16] Y. Kim and T. Kim, "Algorithm for synthesis and exploration of clock spines," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 263–268, Jan 2017.

[17] H. Seo, J. Kim, M. Kang, and T. Kim, "Synthesis for power-aware clock spines," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 126–131, Nov 2015.

[18] V. Kumar and A. Naeemi, "An overview of 3D integrated circuits," in *2017 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization for RF, Microwave, and Terahertz Applications (NEMO)*, pp. 311–313, May 2017.

[19] Y. Xie and Y. Ma, "Design space exploration for 3D integrated circuits," in *2008 9th International Conference on Solid-State and Integrated-Circuit Technology*, pp. 2317–2320, Oct 2008.

[20] P. Jain, T. Kim, J. Keane, and C. H. Kim, "A multi-story power delivery technique for 3D integrated circuits," in *Proceeding of the 13th international symposium on Low power electronics and design (ISLPED '08)*, pp. 57–62, Aug 2008.

[21] B. Jacob, S. W. Ng, and D. T. Wang, *Memory Systems: Cache, DRAM, Disk.* 01 2008.

[22] M. Brox, M. Balakrishnan, M. Broschwitz, C. Chetreanu, S. Dietrich, F. Funfrock, M. A. Gonzalez, T. Hein, E. Huber, D. Lauber, M. Ivanov, M. Kuzmenka, C. N. Mohr, J. O. Garrido, S. Padaraju, S. Piatkowski, J. Pottgiesser, P. Pfefferl, M. Plan, J. Polney, S. Rau, M. Richter, R. Schneider, R. O. Seitter, W. Spirkl, M. Walter, J. Weller, and F. Vitale, "An 8-Gb 12-Gb/s/pin GDDR5X DRAM for Cost-Effective High-Performance Applications," *IEEE Journal of Solid-State Circuits*, vol. 53, pp. 134–143, Jan 2018.

[23] K. Hwang, B. Kim, S. Byeon, K. Kim, D. Kwon, H. Lee, G. Lee, S. Yoon, J. Cha, S. Jang, S. Lee, Y. Joo, G. Lee, S. Xi, S. Lim, K. Chu, J. Cho, J. Chun, J. Oh, J. Kim, and S. Lee, "A 16Gb/s/pin 8Gb GDDR6 DRAM with bandwidth extension techniques for high-speed applications," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pp. 210–212, Feb 2018.

[24] Y. Kim, H. Kwon, S. Doo, Y. Eom, Y. Kim, M. Ahn, Y. Kim, S. Jung, S. Do, C. Lee, J. Kim, D. Kang, K. Park, J. Shin, J. Lee, S. Oh, S. Lee, J. Yu, J. Kwon, K. Yu, C. Jeon, S. Kim, M. Won, G. Cho, H. Park, H. Kim, J. Lee, S. Cho, K. Park, J. Park, Y. Lee, Y. Kim, Y. Seo, B. Cho, C. Shin, C. Lee, Y. Lee, Y. Song, S. Bang, Y. Park, S. Choi, B. Kim, G. Han, S. Bae,

H. Kwon, J. Choi, Y. Sohn, K. Park, and S. Jang, “A 16Gb 18Gb/S/pin GDDR6 DRAM with per-bit trainable single-ended DFE and PLL-less clocking,” in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pp. 204–206, Feb 2018.

[25] Y. Cho, Y. Bae, B. Moon, Y. Eom, M. Ahn, W. Lee, C. Cho, M. Park, Y. Jeon, J. Ahn, B. Choi, D. Kang, S. Yoon, Y. Yang, K. Park, J. Choi, J. Lee, and J. Choi, “A Sub-1.0V 20nm 5Gb/s/pin post-LPDDR3 I/O interface with Low Voltage-Swing Terminated Logic and adaptive calibration scheme for mobile application,” in *2013 Symposium on VLSI Circuits*, pp. C240–C241, June 2013.

[26] D. Kim, S. Srikant, B. Pu, and S. Moon, “An early-EMI analysis method for the LPDDR interface,” in *2017 IEEE International Symposium on Electromagnetic Compatibility Signal/Power Integrity (EMCSI)*, pp. 782–785, Aug 2017.

[27] N. Srivastava, A. C. Scogna, H. Shim, A. Baek, Y. Lee, and D. S. Kim, “SI/PI co-optimization for LPDDR3 mobile interface,” in *2017 Asia-Pacific International Symposium on Electromagnetic Compatibility (APEMC)*, pp. 206–208, June 2017.

[28] D. Wu, B. He, X. Tang, J. Xu, and M. Guo, “RAMZzz: Rank-aware DRAM power management with dynamic migrations and demotions,” in *SC ’12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pp. 1–11, Nov 2012.

[29] B. Liu, J. F. Frenzel, and R. B. Wells, “A multi-level DRAM with fast read and low power consumption,” in *2005 IEEE Workshop on Microelectronics and Electron Devices, 2005. WMED ’05.*, pp. 59–62, April 2005.

[30] S. Lu, T. Karnik, G. Srinivasa, K. Chao, D. Carmean, and J. Held, “Scaling the "Memory Wall": Designer track,” in *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 271–272, Nov 2012.

[31] P. Jacob, A. Zia, O. Erdogan, P. M. Belemjian, J. Kim, M. Chu, R. P. Kraft, J. F. McDonald, and K. Bernstein, “Mitigating Memory Wall Effects in High-Clock-Rate and Multicore

CMOS 3-D Processor Memory Stacks," *Proceedings of the IEEE*, vol. 97, pp. 108–122, Jan 2009.

[32] P. Nair, C. Chou, and M. K. Qureshi, "A case for Refresh Pausing in DRAM memory systems," in *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 627–638, Feb 2013.

[33] I. Bhati, M. Chang, Z. Chishti, S. Lu, and B. Jacob, "DRAM Refresh Mechanisms, Penalties, and Trade-Offs," *IEEE Transactions on Computers*, vol. 65, pp. 108–121, Jan 2016.

[34] M. Guan and L. Wang, "Temperature aware refresh for DRAM performance improvement in 3D ICs," in *Sixteenth International Symposium on Quality Electronic Design*, pp. 207–211, March 2015.

[35] V. H. Cordero and S. P. Khatri, "Clock Distribution Scheme using Coplanar Transmission Lines," in *2008 Design, Automation and Test in Europe*, pp. 985–990, March 2008.

[36] V. Karkala, K. C. Bollapalli, R. Garg, and S. P. Khatri, "A PLL design based on a standing wave resonant oscillator," in *2009 IEEE International Conference on Computer Design*, pp. 511–516, Oct 2009.

[37] Y. Chen and K. D. Pedrotti, "Rotary Traveling-Wave Oscillators, Analysis and Simulation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, pp. 77–87, Jan 2011.

[38] J. Wood, T. C. Edwards, and S. Lipa, "Rotary traveling-wave oscillator arrays: a new clock technology," *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 1654–1665, Nov 2001.

[39] F. O'Mahony, C. P. Yue, M. A. Horowitz, and S. S. Wong, "Design of a 10GHz clock distribution network using coupled standing-wave oscillators," in *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*, pp. 682–687, June 2003.

[40] W. B. Wilson, U.-K. Moon, K. R. Lakshmikumar, and L. Dai, "A CMOS self-calibrating frequency synthesizer," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 1437–1444, Oct 2000.

[41] T. Lin and Y. Lai, "An Agile VCO Frequency Calibration Technique for a 10-GHz CMOS PLL," *IEEE Journal of Solid-State Circuits*, vol. 42, pp. 340–349, Feb 2007.

[42] G. Bhargav, G. Prasad, S. D. Canchi, and B. Chanikya, "Design and analysis of phase locked loop in 90nm CMOS," in *2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN)*, pp. 1–7, July 2016.

[43] T. M. Sathish Kumar and P. Periasamy, "Energy Efficient All-Digital Phase Locked Loop Architecture Design on High Resolution Fast Clocking Time to Digital Converter (TDC) Using Model Prescient Control (MPC) Technique," *Wireless Personal Communications*, 02 2018.

[44] S. V. R. Kaipu, K. Vaish, S. Komatireddy, A. Sood, and M. Goswami, "Design of a low power wide range phase locked loop using 180nm CMOS technology," in *2016 International Conference on Signal Processing and Communication (ICSC)*, pp. 443–447, Dec 2016.

[45] H. Sutoh, K. Yamakoshi, and M. Ino, "Circuit technique for skew-free clock distribution," in *Proceedings of the IEEE 1995 Custom Integrated Circuits Conference*, pp. 163–166, May 1995.

[46] A. Mandal, K. C. Bollapalli, N. Jayakumar, S. P. Khatri, and R. N. Mahaptra, "A low-jitter phase-locked resonant clock generation and distribution scheme," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, pp. 487–490, Oct 2013.

[47] R. Islam and M. R. Guthaus, "Current-mode clock distribution," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1203–1206, June 2014.

[48] H. Luo, D. Liu, C. Gong, and M. Li, "A digital crystal-less clock generation scheme for wireless biomedical implants," in *2016 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 480–483, Oct 2016.

[49] C. Lee, T. Chang, Y. Huang, H. Fu, J. Huang, Z. Hsiao, J. H. Lau, C. Ko, R. Cheng, P. Chang, K. Kao, Y. Lu, R. Lo, and M. J. Kao, "Characterization and reliability assessment of solder

microbumps and assembly for 3D IC integration," in *2011 IEEE 61st Electronic Components and Technology Conference (ECTC)*, pp. 1468–1474, May 2011.

[50] "HSPICE: Circuit simulation and analysis tool," *Synopsys Inc.*

[51] S. Inc., "Raphael: 2D, 3D resistance, capacitance and inductance extraction tool."

[52] Y. Cao, "Predictive Technology Model, Nanoscale Integration and Modeling (NIMO) Group, ASU," 2008.

[53] F. O'Mahony, C. P. Yue, M. A. Horowitz, and S. S. Wong, "Design of a 10GHz clock distribution network using coupled standing-wave oscillators," in *Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451)*, pp. 682–687, June 2003.

[54] K. Tran, "The era of high bandwidth memory," in *2016 IEEE Hot Chips 28 Symposium (HCS)*, pp. 1–22, Aug 2016.

[55] H. Jun, J. Cho, K. Lee, H. Son, K. Kim, H. Jin, and K. Kim, "HBM (High Bandwidth Memory) DRAM Technology and Architecture," in *2017 IEEE International Memory Workshop (IMW)*, pp. 1–4, May 2017.

[56] J. C. Lee, J. Kim, K. W. Kim, Y. J. Ku, D. S. Kim, C. Jeong, T. S. Yun, H. Kim, H. S. Cho, S. Oh, H. S. Lee, K. H. Kwon, D. B. Lee, Y. J. Choi, J. Lee, H. G. Kim, J. H. Chun, J. Oh, and S. H. Lee, "High bandwidth memory (HBM) with TSV technique," in *2016 International SoC Design Conference (ISOCC)*, pp. 181–182, Oct 2016.

[57] H. Jun, S. Nam, H. Jin, J. Lee, Y. J. Park, and J. J. Lee, "High-Bandwidth Memory (HBM) Test Challenges and Solutions," *IEEE Design Test*, vol. 34, pp. 16–25, Feb 2017.

[58] K. Chandrasekar, D. Oh, and A. Rahman, "Timing analysis for Wide IO memory interface applications with silicon interposer," in *2014 IEEE International Symposium on Electromagnetic Compatibility (EMC)*, pp. 46–51, Aug 2014.

[59] C. Wang, H. Cheng, M. Chung, P. Pan, C. Chiu, and C. Hung, “High bandwidth application with Wide I/O memory on 2.5D-IC silicon interposer,” in *2013 3rd IEEE CPMT Symposium Japan*, pp. 1–4, Nov 2013.

[60] M. H. Hajkazemi, M. K. Tavana, and H. Homayoun, “Wide I/O or LPDDR? Exploration and analysis of performance, power and temperature trade-offs of emerging DRAM technologies in embedded MPSoCs,” in *2015 33rd IEEE International Conference on Computer Design (ICCD)*, pp. 62–69, Oct 2015.

[61] K. Ahn and C. Yoo, “Skew cancellation technique for >256-Gbyte/s high-bandwidth memory (HBM),” *Electronics Letters*, vol. 52, no. 13, pp. 1155–1157, 2016.

[62] I. G. Thakkar and S. Pasricha, “Massed Refresh: An Energy-Efficient Technique to Reduce Refresh Overhead in Hybrid Memory Cube Architectures,” in *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, pp. 104–109, Jan 2016.

[63] D. Lee, G. Pekhimenko, S. M. Khan, S. Ghose, and O. Mutlu, “Simultaneous Multi Layer Access: A High Bandwidth and Low Cost 3D-Stacked Memory Interface,” *CoRR*, vol. abs/1506.03160, 2015.

[64] P. Jacob, O. Erdogan, A. Zia, P. M. Belemjian, R. P. Kraft, and J. F. McDonald, “Predicting the performance of a 3D processor-memory chip stack,” *IEEE Design Test of Computers*, vol. 22, pp. 540–547, Nov 2005.

[65] B. Akin, F. Franchetti, and J. C. Hoe, “Data reorganization in memory using 3D-stacked DRAM,” in *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, pp. 131–143, June 2015.

[66] H. Jeon, G. H. Loh, and M. Annavaram, “Efficient RAS support for die-stacked DRAM,” in *2014 International Test Conference*, pp. 1–10, Oct 2014.

[67] S. Gong, J. Kim, S. Lym, M. Sullivan, H. David, and M. Erez, “DUO: Exposing On-Chip Redundancy to Rank-Level ECC for High Reliability,” in *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 683–695, Feb 2018.

[68] H. Jun, J. Cho, K. Lee, H. Son, K. Kim, H. Jin, and K. Kim, “HBM (High Bandwidth Memory) DRAM Technology and Architecture,” in *2017 IEEE International Memory Workshop (IMW)*, pp. 1–4, May 2017.

[69] Y. Kim, W. Yang, and O. Mutlu, “Ramulator: A Fast and Extensible DRAM Simulator,” *IEEE Computer Architecture Letters*, vol. 15, pp. 45–49, Jan 2016.

[70] Chandrasekar, Karthik, C. Weis, Y. Li, S. Foossens, M. Jung, O. Naji, B. Akesson, N. When, , and K. Goossens, “DRAMPower: Open-source DRAM Power and Energy Estimation Tool.”

[71] S. Zanella, A. Nardi, A. Neviani, M. Quarantelli, S. Saxena, and C. Guardiani, “Analysis of the impact of process variations on clock skew,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 13, pp. 401–407, Nov 2000.

[72] W. . D. Lam and C.-K. Koh, “Process variation robust clock tree routing,” in *Proceedings of the ASP-DAC 2005. Asia and South Pacific Design Automation Conference, 2005.*, vol. 1, pp. 606–611 Vol. 1, Jan 2005.

[73] K. Fang, H. Zheng, J. Lin, Z. Zhang, and Z. Zhu, “Mini-Rank: A Power-Efficient DDRx DRAM Memory Architecture,” *IEEE Transactions on Computers*, vol. 63, pp. 1500–1512, June 2014.

[74] F. T. Hady, A. Foong, B. Veal, and D. Williams, “Platform Storage Performance With 3D XPoint Technology,” *Proceedings of the IEEE*, vol. 105, pp. 1822–1833, Sep. 2017.

[75] J. Zhang, P. Li, B. Liu, T. G. Marbach, X. Liu, and G. Wang, “Performance Analysis of 3D XPoint SSDs in Virtualized and Non-Virtualized Environments,” in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 1–10, Dec 2018.

[76] S. Hamdioui, H. Aziza, and G. C. Sirakoulis, “Memristor based memories: Technology, design and test,” in *2014 9th IEEE International Conference on Design Technology of Integrated Systems in Nanoscale Era (DTIS)*, pp. 1–7, May 2014.

[77] H. Kim, M. P. Sah, C. Yang, and L. O. Chua, “Memristor-based multilevel memory,” in *2010 12th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2010)*, pp. 1–6, Feb 2010.

[78] H. . P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, “Phase Change Memory,” *Proceedings of the IEEE*, vol. 98, pp. 2201–2227, Dec 2010.