Network-on-Chip Synchronization

NETWORK-ON-CHIP SYNCHRONIZATION

A Thesis Presented

by

MARK ANDREW BUCKLER

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL AND COMPUTER ENGINEERING

September 2014

Electrical and Computer Engineering

NETWORK-ON-CHIP SYNCHRONIZATION

A Thesis Presented

by

MARK ANDREW BUCKLER

Approved as to style and content by:

_____
Wayne P. Burleson, Chair

_____
Russell Tessier, Member

_____
Zlatan Aksamija, Member

_____
Christopher V. Hollot, Department Head
Electrical and Computer Engineering

ABSTRACT

NETWORK-ON-CHIP SYNCHRONIZATION

SEPTEMBER 2014

MARK ANDREW BUCKLER

B.S.E.E., RENSSELAER POLYTECHNIC INSTITUTE

M.S.E.C.E., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Wayne P. Burleson

Technology scaling has enabled the number of cores within a System on Chip (SoC) to increase significantly. Globally Asynchronous Locally Synchronous (GALS) systems using Dynamic Voltage and Frequency Scaling (DVFS) operate each of these cores on distinct and dynamic clock domains. The main communication method between these cores is increasingly more likely to be a Network-on-Chip (NoC). Typically, the interfaces between these clock domains experience multi-cycle synchronization latencies due to their use of “brute-force” synchronizers. This thesis aims to improve the performance of NoCs and thereby SoCs as a whole by reducing this synchronization latency.

First, a survey of NoC improvement techniques is presented. One such improvement technique: a multi-layer NoC, has been successfully simulated. Given how one of the most commonly used techniques is DVFS, a thorough analysis and simulation of brute-force synchronizer circuits in both current and future process technologies is presented. Unfortunately, a multi-cycle latency is unavoidable when using brute-force synchronizers, so predictive synchronizers which require only a single cycle of latency have been proposed.

To demonstrate the impact of these predictive synchronizer circuits at a high level, multi-core system simulations incorporating these circuits have been completed. Multiple forms of GALS NoC configurations have been simulated, including multi-synchronous, NoC-synchronous, and single-synchronizer. Speedup on the SPLASH benchmark suite was measured to directly quantify the performance benefit of predictive synchronizers in a full system. Additionally, Mean Time Between Failures (MTBF) has been calculated for each NoC synchronizer configuration to determine the reliability benefit possible when using predictive synchronizers.

TABLE OF CONTENTS

Page

ABSTRACT iv

LIST OF TABLES viii

LIST OF FIGURES ix

LIST OF EQUATIONS xi

CHAPTER

1: BACKGROUND 1

1.1 Networks-on-Chip 2

1.1.1 Traffic Management 3

1.1.2 Low-Swing Interconnect 4

1.1.3 Three Dimensional Interconnect 5

1.1.4 Nanophotonic Interconnect 5

1.1.5 Wireless Interconnect 6

1.1.6 Asynchronous Communication 6

1.1.7 Dynamic Voltage and Frequency Scaling 7

1.1.8 Multiple NoC Structures 8

1.2 Synchronizers 8

1.2.1 Metastability 9

1.2.2 Brute-Force Synchronizers 10

1.2.3 The Effect of Technology Scaling 12

1.2.4 Predictive Synchronizers 13

2: A MULTI-NOC SYSTEM 16

3: SIMULATION OF BRUTE-FORCE SYNCHRONIZAER FLIP-FLOPS 19

3.1 Simulating Tau 20

3.2 Flip-Flop Initialization 21

3.3 Tau and FO4 Delay 23

3.4 Performance and Temperature 25

3.5 Forward Biasing 27
4: DESIGN OF DVFS TECHNIQUES FOR PREDICTIVE SYNCHRONIZATION 29
4.1 Gradual Frequency Change 29
4.2 Instant Frequency Change 33
4.3 The Locally Controlled CDI 37
4.4 RTL Implementation 41
5: SYSTEM LEVEL IMPACT OF PREDICTIVE SYNCHRONIZERS 44
5.1 NoC Organization 44
5.2 Performance Simulation 46
5.2.1 Synchronizer Latency vs CDI Latency 46
5.2.2 Graphite Simulator Changes 48
5.2.3 Simulation Results 49
5.3 Reliability Calculations 51
6: CONCLUSIONS AND FUTURE WORK 53
BIBLIOGRAPHY 54

LIST OF TABLES

Table Page

1: CDI Comparison 42

2: Average Speedup Due to Predictive Synchronizers 50

3: 64 Core System Level MTBF 52

LIST OF FIGURES

Figure Page

1: Mesh Network-on-Chip 3

2: Metastable Waveforms 10

3: Three Stage Brute-Force Synchronizer Circuit 11

4: High Level EOPS Structure [69] 14

5: EOPS Flip-Flop Selector [69] 14

6: Multi-NoC Architecture 16

7: NoC Selection Scheme 17

8: Multi-NoC System Simulation Results 18

9: Synchronizer Bi-stable Element Circuit Response 20

10: Dynamic Latch Flip-Flop 21

11: PowerPC Flip-Flop 22

12: Pseudo-NMOS Flip-Flop 23

13: τ/FO4 Results 24

14: 22nm Planar Technology, Temp vs. Vdd (FSFF left, Pseudo-NMOS right) 26

15: 20nm FinFET Technology, Temp vs. Vdd (FSFF left, Pseudo-NMOS right) 26

16: Voltage Sweep of Non-Biased and Forward Biased DLFF and Pseudo-NMOS 28

17: Continuous Measurement Strategy at Doubled Rate 32

18: Continuous Frequency Measurement Circuit 32

19: Continuous Frequency Measurement Timing 32

20: High Level Diagram of Fast DVFS Strategy 34

21: Fast Change in Receiver Clock Frequency Timing 35

22: The Locally Controlled CDI 37

23: CDI protocol when the receive clock is fastest 38

24: Phase Estimation for Safety Detection 39

25: CDI Protocol when the transmit clock is fastest 40

26: NoC Structures Considered 45

27: Interior Structure of Asynchronous FIFO 47

28: 64 Core Benchmark Speedup When Using Predictive Synchronizers 50

LIST OF EQUATIONS

Equation Page

1: MTBF of Brute-Force Synchronizer 11

2: Calculation of Simulated τ 21

3: τ as Function of FO4 24

4: MTBF of Multi-Synchronizer SoC 51

CHAPTER 1

BACKGROUND

Designers of modern computing devices are constantly challenged to create new systems with higher performance and lower power. The demand for power reduction has contributed to a plateau in clock rates, but Moore's Law has ensured that every year will offer more transistors on-chip than the last. So, designers have chosen to use these extra transistors for additional processors on-chip, relying on parallelism for increased performance. This has resulted in the need for more complex on-chip communication. Networks-on-Chip (NoCs) offered the ability to route packets instead of wires, allowing them to scale better than the bus structures previously used for on-chip communication. When cores and NoCs use DVFS to save power, cores and routers operate with different frequency clocks, requiring synchronizers to reduce the chance of metastability. Brute-force synchronizers present a tradeoff between communication latency and reliability, resulting in additional NoC latency. Delayed NoC communication hurts overall system performance which is dependent upon these global communications.

Dynamic Voltage and Frequency Scaling (DVFS) is an effective and commonly used method of saving power by tuning individual processor cores to optimal operating conditions. Since each core is tuned independently, Systems-on-Chip (SoCs) typically have cores operating at many different voltages and frequencies at one time. Level shifters are an effective and low-impact method of communicating between voltage domains, but clock domain interfaces containing synchronizers are needed for communication between frequency domains.

During synthesis of a design, much care is taken to ensure that data will always be aligned with the system's clock. If the data comes well before the clock, then the clock speed

could be increased, and if the data comes after the clock then the data is missed entirely. If the data comes at the same time as the clock (within the flip-flop's sensitive region) then the flip-flop can be thrown into a state that is neither high nor low: metastable. When communications flow between systems with entirely different clocks, there is significant risk for metastability to occur.

1.1 Networks-on-Chip

Before the concept of a network-on-chip (NoC) was proposed, systems-on-chips (SoCs) relied on complex bus structures to connect processors to memory and I/O. Moore's Law has continued since that time; however, clock rates have stagnated due to power issues. The need for more processing power (without clock speed increases) and the ability to add more transistors has led to an increase in the number of processors on chip. The old bus structures were improved to account for these multi-processor system-on-chips, but eventually the bus designs could not sustain the scaling and complexity of the necessary on-chip interconnect. The NoC emerged as a solution to this problem by "routing packets instead of wires" and has increased in popularity since then [1]. This trend has led to companies such as Arteris, Sonics, Blendics, and iNoCs who provide NoC designs and IP for other SoC companies. Many companies are beginning to choose pre-designed NoC IP solutions over their own designs.

NoC research is primarily on traffic management, low-power interconnect, and clock management. NoC traffic management involves research into topics like cache coherence and compression. Low-power interconnect includes low-swing, three dimensional, nanophotonic, and wireless interconnect. Clock management schemes include asynchronous communication as well as dynamic voltage and frequency scaling.

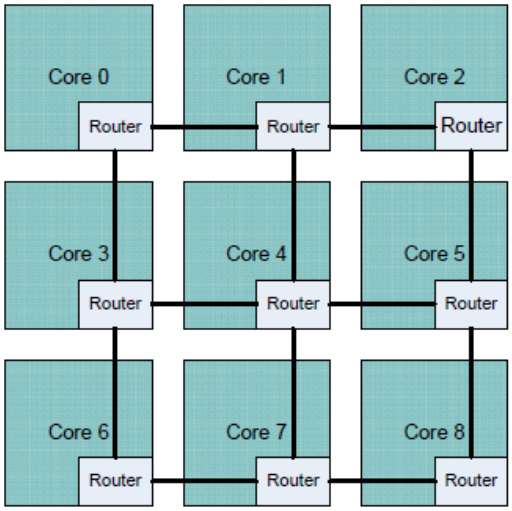


Figure 1: Mesh Network-on-Chip

1.1.1 Traffic Management

In most SoCs the bulk of NoC traffic is for cache coherence. For this reason, design and management of the cache is critical and must be considered when distributing the cache among CPU and GPU cores [2]. Methods have been developed to reduce the power used in cache hierarchy management using both data locality and knowledge of the NoC's physical structure [3, 4]. Coherence-free systems have been proposed to avoid coherence protocols, but industry largely favors cache-coherent systems [5]. One successfully demonstrated method to decrease cache coherency power usage combined bus based snooping coherency and NoC based directory coherency [6].

Power reduction has also been achieved through efficient use of data compression [7], error detection/correction encoding [8], and heterogeneous interconnect [9]. Other techniques achieved power reduction by differentiating among different kinds of traffic (such as 1-to-many/many-to-1 [10] or request/response [11]) and optimizing for each type. Hardware

techniques focus on router designs and microarchitecture [12]. Although buffer-less NoC designs have been proposed, their benefits are minimal (1.5% savings) [13].

Not all traffic has the same effect on application completion time however. If packets can be prioritized based on their application level criticality, then high priority packets can be sent on low latency interconnect and low priority packets can be sent on low energy interconnect [9]. When identified, high priority data can use a circuit switching protocol over a lower performing packet based protocol [14]. Data can also be classified as bandwidth or latency sensitive, allowing for bandwidth and latency optimization of two individual networks. Once established, traffic can be assigned within each network based on the given priority of each packet [15]. Ring based systems [16] and non-uniform cache architectures [17] have also been integrated with packet prioritization protocols.

1.1.2 Low-Swing Interconnect

Low-swing signaling attempts to save energy by reducing the voltage potential between high and low states (lowering the swing) on long on-chip connections. New low-swing techniques have proven to reduce clock power by 66% [18]. With reduced swing comes increased sensitivity to noise, however, requiring special care to ensure reliability [19]. Due to the analog nature of this technique, work has primarily focused on differential signaling, and both voltage- and current-mode transceiver circuit designs [18][20]. Unfortunately, highly custom circuits pose a problem for modern SoCs, which are often designed using synthesized circuits. For this reason, focus also has been given to creating low-swing solutions that can be easily implemented using mainstream SoC design techniques [21].

1.1.3 Three Dimensional Interconnect

The long-awaited emergence of 3D VLSI and die-stacking technology has motivated additional work in the corresponding NoCs. 3D promises shorter interconnect and reduced capacitance, as well as excellent inter-layer connections with the use of through-silicon vias (TSVs) [22]. TSVs also make the circuit design of 3D routers and 3D routing schemes significantly different [14]. Unfortunately, the state of technology today prevents more than two logic layers to be stacked in one package due to thermal concerns. Designs have been proposed with more than two layers, suggesting that one layer could be dedicated to the NoC [23]. These thermal concerns have caused researchers to explore the possibility of thermal-aware 3D NoC architectures that can help mitigate thermal issues [24].

1.1.4 Nanophotonic Interconnect

Although a nanophotonics-based NoC has not yet been developed due to technology limitations, silicon photonics have now been demonstrated in a 90-nm process [25]. This kind of progress has increased interest in nanophotonics as a way to replace traditional metal wires for long-haul connections in NoCs. Full analysis of planned nanophotonic networks has shown significant promise for both increased performance and decreased power consumption using athermal ring resonators and on-chip lasers that enable quick power-gating [26]. Nanophotonics promises bit rates almost independent of distance, higher bandwidth from frequency-division multiplexing (FDM), and lower power due to dissipation at the endpoint only. These promised benefits allow for the potential to improve performance by 60% and decrease power by 80% [27]. NoC laser energy also can be reduced by 49% using busses controlled by distributed on-chip lasers [28]. While these pure photonic designs are very attractive the first practical photonic NoC likely will be some combination of photonics and traditional metal wires [29].

Although recent nanophotonics research is very promising, there is more work to be done on the process side before nanophotonic NoCs can be fully realized.

1.1.5 Wireless Interconnect

Both photonics and wireless NoC designs are part of a trend to integrate formerly off-chip communication techniques into the on-chip network to increase performance and reduce power. Miniature on-chip antennas could be used to transmit and receive information, and the technology already exists to create them on silicon. A wireless NoC would save power and area because small transmitters do not need large capacitive transmission lines and do not require multi-hop connections. Hybrid designs have been proposed with wireless used for long-distance on-chip transmissions [30, 31].

Wireless NoCs can use FDM (similar to the concept's use in nanophotonics) and time-division multiplexing (TDM) along with low-power transceivers to achieve 34% power reduction compared to leading NoCs [32]. Another wireless NoC design uses a sub-divided mesh topology to improve the performance of other wireless NoC designs [33]. Wireless systems face unique challenges however. For now, designers are limited to using existing millimeter-wave antennas using CMOS technology, but future carbon nanotube antennas will significantly reduce the overhead [34]. Use of these carbon nanotube antennas is not possible yet due to the need for process scaling that has not yet been achieved.

1.1.6 Asynchronous Communication

Distributing a global clock across an entire NoC continues to be difficult and very power-hungry as technology scaling continues while die area remains the same. For this reason, the globally asynchronous/locally synchronous (GALS) NoC was proposed. Studies have verified that

GALS NoCs save both energy and latency by removing the global clock but require overhead in the form of synchronizer circuits and extra router wires for flow control [35]. These extra router wires manifest as a requirement for more space for the NoC, sometimes as high as 25% increased switch area (while still maintaining 21% power reduction, given certain factors) [36]. Those numbers were improved to an impressive 57% power reduction when using the butterfly fat tree (BFT) network topology [37].

Area overhead can be reduced with specialized circuitry for routers and other asynchronous components [38]. Even without using the full GALS approach, gains can be achieved with asynchronous circuitry. One recent paper uses router crossbars with built-in asynchronous repeated link circuits. This technique has achieved single-clock-cycle latency along with a 2.2X power savings [39]. Source-synchronous communication using bundled data also has been proposed. This technique routes the clock (as a pulse) along with the data. Source-synchronous systems reduce power through their removal of the global clock [40].

1.1.7 Dynamic Voltage and Frequency Scaling

Similar to other parts of the SoC, the NoC does not always need to operate at its maximum possible level of performance. For this reason, dynamic voltage and frequency scaling (DVFS) can optimize dynamic power. NoCs consider how these DVFS changes affect incoming and outgoing data rates at the node level. Recent work has shown that savings as high as 33% can be seen when applying DVFS to the NoC and low-level cache (LLC) when sharing a voltage/frequency domain [41]. Another proposed design includes dynamic reconfigurable NoC interconnect in addition to DVFS, allowing for energy savings and latency reduction [42]. A simplified binary DVFS control using only a high and a low voltage state also has been proposed to be sufficient for NoC switches [43].

1.1.8 Multiple NoC Structures

While much work has been done to increase performance and decrease energy of single NoC systems, the research community has also considered the possibility of a multi-NoC system. Multi-NoC systems have proven to significantly improve overall system performance while minimally affecting the area-delay and energy-delay products with their hardware overhead [44]. Even naive homogenous network sharing protocols can achieve this benefit due to the inherent increase in both bandwidth and path diversity. Even greater benefit can be achieved with heterogeneous networks however. If specific types of traffic are allocated to specific networks, each network can be optimized for that form of traffic. This has been demonstrated for many forms of traffic including on chip monitoring data [45], cache access requests/responses [11], and main memory access as well as IO [46].

1.2 Synchronizers

Global clocks have been proven to be un-scalable for today's large dies, leading to the increase of GALS systems. Fine-grained DVFS is now used to achieve power savings by dynamically tuning each portion of the system to its current load. This results in each of the cores operating on different frequencies, but still needing to communicate. The goal is to have low-latency, high-throughput, and high reliability inter-clock-domain communication. Even Systems on Chip (SoCs) with a relatively low number of cores depend on high performance communication between clock domains for last-level cache access. Examples include AMD's Bulldozer core systems and Intel's Core i7.

Communicating across clock domains requires a system with flow control to prevent under-running or over-running the receiver, as well as synchronizer circuits to mitigate the risk of metastability (thereby increasing reliability). This thesis defines any system performing both

of these functions as a Clock Domain Interface (CDI). The most common kind of CDI is an asynchronous FIFO [47]. The full and empty signals generated by the FIFO inform the two domains when it is possible to transmit or receive, while brute force synchronizers pass pointers between the domains.

1.2.1 Metastability

Synchronizer circuits are used in many applications to transition data between elements in a MCD (Multiple Clock Domain) device. A common example is in GALS SoCs where multiple parts of a chip operate at different frequencies, yet need to communicate with each other. Another example is high clock rate processor cores that need to talk to other slower cores, or even slower memory. Many designs have been created to solve this problem and all of them aim to remove metastability.

Metastability is a problem caused when unmatched clocks are used to transfer data without any guiding structure. When the clock of a receiving domain rises while the data line is transitioning, a metastable state can be generated within the sampling flip-flop. This metastable state is neither at Vdd or GND, but half way in between (Vdd/2). Generally this occurs within the sampling flip-flop: at the output of the master latch. The reason why this occurs is because when the data and clock change at the same time, the latch does not have enough time to change state by charging or discharging its output. Instead, partial charge exists, balanced like a ball on the peak of a steep hill.

Due to the properties of CMOS circuits, Vdd/2 does not propagate through to the slave latch of the flip-flop. This means that the flip-flop effectively waits for the master latch to settle into a high or low state. There is no guarantee when the latch will settle, introducing randomness into a system that must be designed to be perfectly logically predictable. If the

master latch transitions while the slave latch samples, the same problem will also occur on the slave latch, allowing metastability to exist on the output of the flip-flop as well.

In addition to delay, there is no guarantee that the eventual output value will have been sampled correctly. See the image below for a visual representation of a metastable waveform. In this example the data and clock edges are too close together, causing a metastable output. Since exactly half of the charge is left, the circuit could settle either high or low with an equal probability. So, either the data will successfully make its way to the new logic value (metastable output A) or it will fall back to its original value (metastable output B).

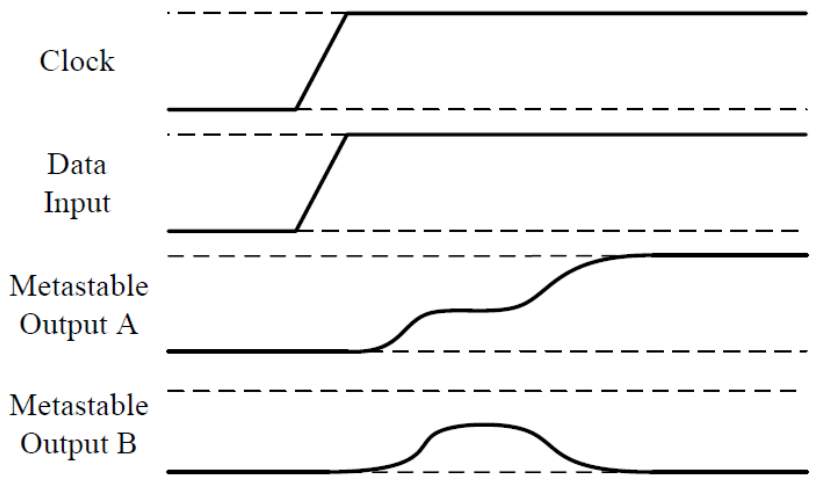


Figure 2: Metastable Waveforms

1.2.2 Brute-Force Synchronizers

All synchronizers are designed to reduce the possibility of a metastability failure, defined as when the output of a sampling circuit exhibits a metastable value. The most common form of a synchronizer is the brute-force synchronizer, which is a chain of flip-flops connected in series. Since metastability can only propagate from one synchronizer to the other if it resolves to

a stable value during a rising clock edge, each additional flip-flop reduces the possibility of metastability. The synchronizer will always retain a finite chance of failure however since settling time is random. Since additional flip-flops also result in an increased latency through the synchronizer, a trade-off between performance and reliability exists.

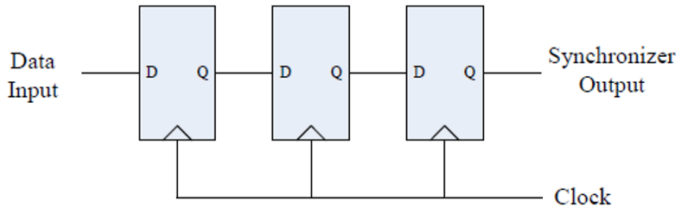


Figure 3: Three Stage Brute-Force Synchronizer Circuit

To quantify the frequency of these failures, brute force synchronizer performance is commonly expressed as the Mean Time Between Failures (MTBF). The expression in **Equation 1** shows dependence on the receiver clock frequency F_C and incoming data frequency F_D. These factors are independent of the flip-flops within the synchronizer, and so these factors do not change with CMOS scaling. The factor *S*, defined as the settling time, is also circuit independent since it is often simplified as the period of the receiver clock period.

$$MTBF = \frac{e^{\frac{S}{\tau}}}{T_w * F_D * F_C}$$

Equation 1: MTBF of Brute-Force Synchronizer

Tw (the sensitive region) and τ (evaluation time constant) are both dependent on the circuit, and are therefore affected by changes in process technology. Although some papers have considered the effects of *Tw* [48], *Tw* only affects MTBF linearly while τ is exponential. For this reason, much of the past work has chosen to focus on τ rather than *Tw* [49, 50]. More complex expressions have been proposed for accurately calculating MTBF for multi-stage synchronizers; however these expressions still show that τ has an exponential effect on MTBF

[51]. For this reason, the evaluation of τ is just as important when considering multi-stage synchronizers.

The value of τ can be affected by many environmental factors, but circuit structure plays a central role. Two circuit parameters of particular importance are transistor strength and node capacitance. A number of circuit topologies have been designed to minimize the necessary node capacitance, while also improving the gain of the flip-flop feedback loop [50, 52-56]. Device sizing is also important however, and so a multitude of techniques have been proposed to effectively manage the tradeoff between device strength and size [56][57][54]. Some techniques have also considered adding supplementary devices with mixed success [55, 58].

1.2.3 The Effect of Technology Scaling

As technology scales both transistor strength and node capacitance change. One key parameter used to compare technology nodes is the fan out of four delay (commonly referred to as *FO4*). Since the early days of CMOS, designers have tracked both *FO4*and τ, finding them to be scale at the same rate [59]. As planar technology has continue to scale, some researchers claim that this trend has continued [56, 60] while others claim that after 65nm this is no longer the case [61, 62].

FinFET devices offer a number of significant benefits in addition to allowing the continuation of Moore's Law. Two of the most well-known benefits of FinFET technology are decreased leakage current and less variation in Vt between process corners [63, 64]. While all circuits will certainly benefit from a decrease in leakage current, this is not a key factor in synchronizers, which focus much more on dynamic behavior. The effect of tighter process corners will have an impact of synchronizers due to the need to design for the worst case. The end result of this is expected to be an overall improvement in synchronizer performance, but

would not affect design trade-offs. Process corners are also heavily defined by laboratory experimentation, and therefore future technology simulation models do not consider them [65].

1.2.4 Predictive Synchronizers

Predictive synchronizers transcend the latency-reliability tradeoff which limits brute-force synchronizers. This means that they can offer near perfect MTBF while still maintaining less than a cycle of latency. This is achieved by exploiting certain relationships between the two clocks to predict their behavior. Each predictive synchronizer has limitations of its own however. Some designs rely on rationally-related clocks and cannot be used with other frequency combinations [66, 67]. Others depend on large delay lines which are continuously matched to the clock period [68]. One system proposed by Dally called the Even Odd Predictive Synchronizer (EOPS) overcomes these problems, but introduces limitations of its own [69].

To accomplish a single cycle of latency, the EOPS leverages the periodic nature of the receive and transmit clocks to estimate the transmit phase from the perspective of the receive domain. These phase estimates are used to choose from one of two flip-flops (Even or Odd) to sample in the receive domain. **Figure 4** shows a high level view of the EOPS.

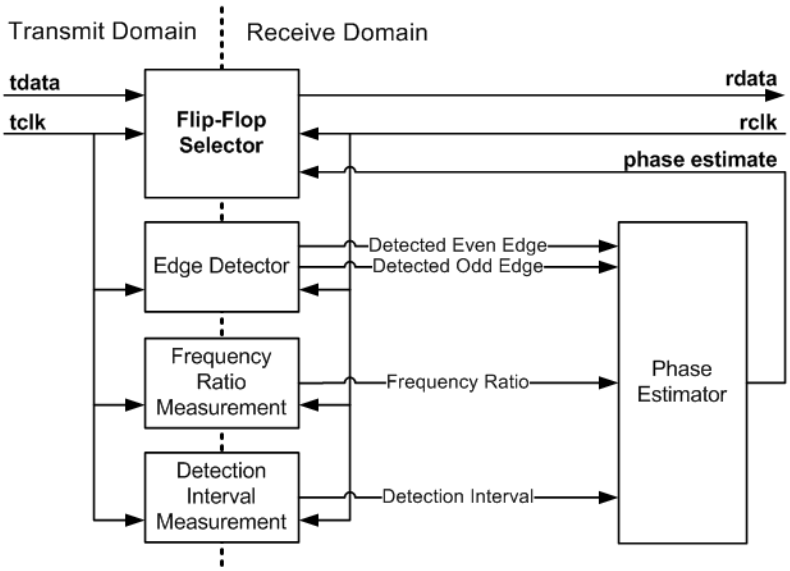


Figure 4: High Level EOPS Structure [69]

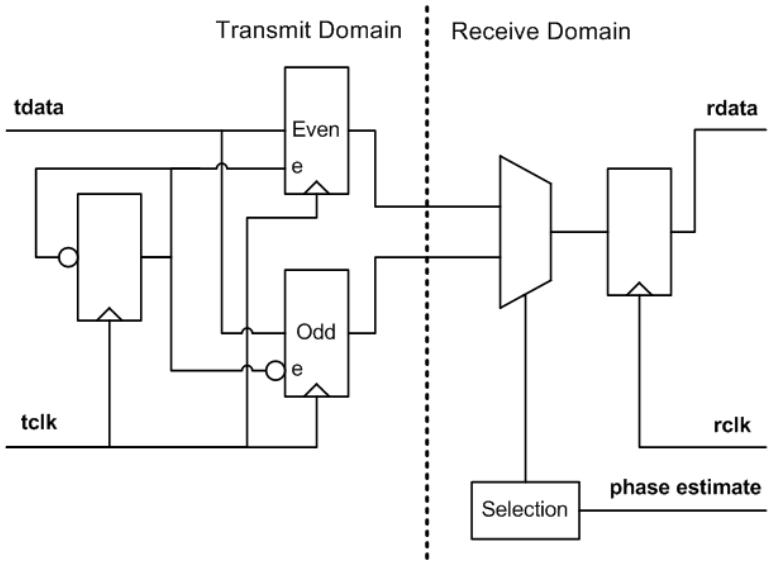


Figure 5: EOPS Flip-Flop Selector [69]

The EOPS needs certain parameters to be measured before it can operate. When clock frequencies change, the system needs to stop data flow while it re-measures. Unfortunately, measurement relies on the overflow of large counters, resulting in a measurement time of over a thousand cycles. For systems that apply DVFS frequently, this long stall in data flow is

unacceptable. This prevents the use of this otherwise robust low-latency synchronizer in practice.

Phase estimation is achieved by first taking a digital measurement of both the ratio between the two clocks, and the size of a "detection interval". The detection interval represents the size of a delay line within a "phase detector" circuit. This alerts the system when two rising edges are within the detection region, meaning they are of the same phase at that time. Since this information needs to be sent through a brute force synchronizer, it is delayed by a certain number of receiver clock cycles.

The system needs current phase estimates however. This can be calculated by multiplying the ratio of the two clock frequencies by the number of delayed receiver clock cycles. This calculation determines how many transmit clock cycles (and fractions of cycles) have occurred since the system detected that the two domains were of the same phase. This in turn gives a phase estimate for the current receiver clock phase, noting that whereas they were within the range of the detection interval, they were not exactly of the same phase. This results in a level of built-in uncertainty in the phase estimate. To define this possible range of phases, upper bound and lower bound estimates are calculated by adding or subtracting the detection interval from the previous calculation [69].

CHAPTER 2

A MULTI-NOC SYSTEM

After analyzing the many techniques for NoC power reduction and performance improvement, one such technique was simulated. An experiment was performed on a simulated multilayer NoC. This does not change the synchronization structure, but it does serve as an introduction to the many possibilities for NoC structure improvements. Implementation of this this system was accomplished in the Graphite multicore simulator [70].

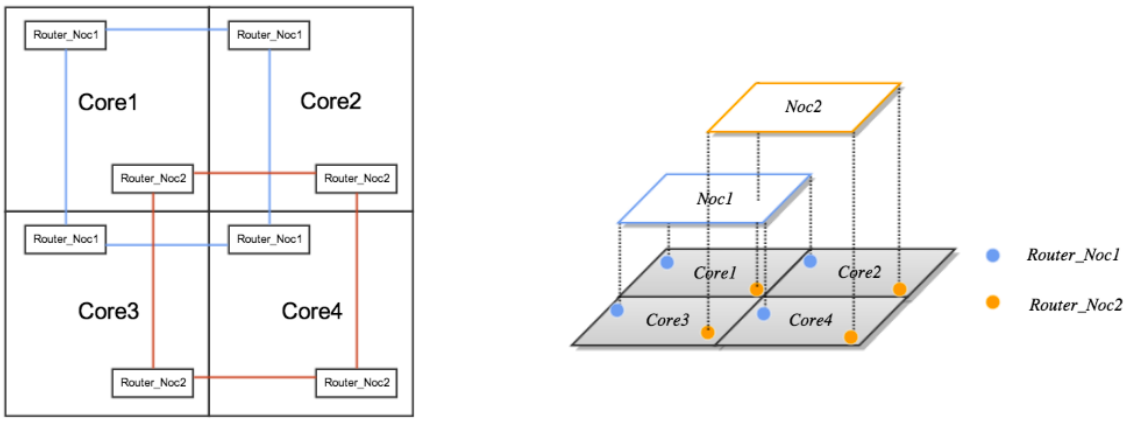


Figure 6: Multi-NoC Architecture

The case shown in **Figure 6** contains four cores. Each core has access to two routers, one for each network. Each router is in communication with a single NoC, resulting in each NoC being separate. Thus each NoC occupies a "layer", although this does not necessarily mean that each network is in a separate physical layer. When a packet needs to be sent, the NoC selector is used to determine which NoC will be used. For the purposes of this experiment, a NoC is selected at random. More sophisticated selection methods could be used in conjunction with a more complex heterogeneous multi-NoC structure. This would allow individual networks to be

optimized for traffic of a certain classification, which the NoC selector would allocate to the appropriate NoC layer.

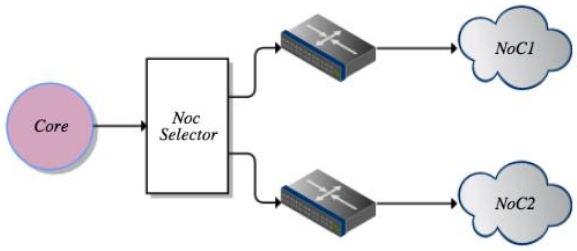


Figure 7: NoC Selection Scheme

The emesh_hop_by_hop model has been selected within the Graphite simulator. This model was chosen since it includes contention delay tracking. Two main files were modified to include the additional NoC: network_model_emesh_hop_by_hop.cc and its associated h file. An additional mesh router, injection router, and network link list were added to each of the router models. To evenly distribute the traffic across the NoCs, a random number is generated and checked to allocate traffic.

Both a single NoC and a double NoC system were simulated with three benchmarks: Spawn_joint_unit_test, Ping_pong_app_test, and Pthread_copy_test. These benchmarks are synthetic, purposefully creating and destroying threads but not necessarily due to a real application. The ping pong app test sends data back and forth between cores, and in simulation proved to inject the most data. It follows that this benchmark also saw the most performance improvement and power reduction. Spawn creates multiple threads of random sizes and locations, creating traffic by requesting data for individual threads. Pthread copy creates multiple threads of the same kind in various locations.

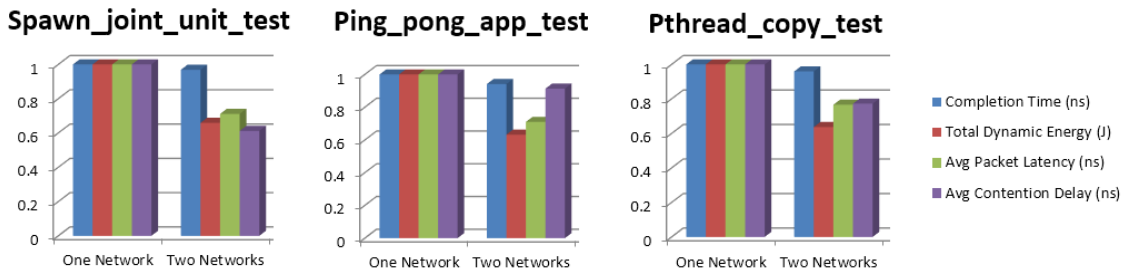


Figure 8: Multi-NoC System Simulation Results

The metrics used to evaluate the performance of these structures on each of these benchmarks are completion time, total dynamic energy (including the second network in cases where it exists), average packet latency, and average contention delay. The results shown in **Figure 8** are normalized to the results for the one network baseline. For all three benchmarks, completion time was reduced around 10%. This is significant since the reduction of completion time directly translates to a performance improvement.

The dynamic energy is also significantly reduced by an impressive 40% by the two network system. It should be noted that this improvement is on the dynamic energy of just the network structure (routers and links) and so the percentage improvement on the entire system would be less. The reasons for reduction in both power and completion time are due to the reduction in congestion. A 30% reduction in contention delay can be seen for the pthread test, and the other tests average to around 15% reduction. The delay is not the only benefit, because congestion also causes issues when packets are dropped due to overfilled routers. This requires re-transmission of data, further increasing the necessary power. With the reduction in congestion, the number of dropped packets have been reduced, hence the reduction in dynamic energy.

CHAPTER 3

SIMULATION OF BRUTE-FORCE SYNCHRONIZER FLIP-FLOPS

Although more sophisticated circuits have been proposed, communication across clock domains in most modern devices requires the use of brute-force synchronizers. Even predictive synchronizers contain brute-force synchronizers out of the critical path. Brute-force synchronizers are made up of highly optimized flip-flops connected in series. Synchronizer flip-flop circuit designers must re-evaluate design trade-offs as process technologies change. Now that semiconductor manufacturers are moving from planar CMOS to FinFET CMOS, designers must prepare for changes of an even larger scale.

Previous work has focused on the effect of technology scaling on synchronizers in planar technology [56, 59-62], but here we aim to demonstrate the effects of FinFET devices. To evaluate their effects, HSPICE simulations using predictive technology models developed at ASU [65] were performed and compared with theoretical analysis. While on-chip synchronizer evaluation techniques have been developed [48], building test chips for multiple technology nodes would be both prohibitively expensive and impossible for nodes not yet developed (such as 10nm and 7nm). Here we present simulation results and theoretical analysis on three high-performance synchronizer flip-flops. The simulation results presented here demonstrate the continual relationship between tau and FO4 delay even in FinFET devices. It was also found that body biasing synchronizer feedback loops is still effective in FinFET devices, but not as significant as with planar devices. FinFET synchronizers were also observed to be more sensitive to temperature variations.

3.1 Simulating Tau

The HSPICE simulation strategy used to generate the results found in this paper was modeled on a hardware testing methodology developed by Dike and Burton [50]. A similar adaption of this technique has recently been used in other synchronizer design research [49]. In the original hardware technique, memory cell nodes are first forced into metastable levels. This is accomplished with initial conditions in simulation. Once forced into metastability, measurements are taken as the circuit begins to self-correct back to a stable value (a logical high or low value). The speed at which the circuit can self-correct is recorded, and used to calculate the value of τ. This gives an accurate value without needing to rely on simulation methods requiring data vs. clock time sweeps, which have proven to be unreasonable given HSPICE's sensitivity [50].

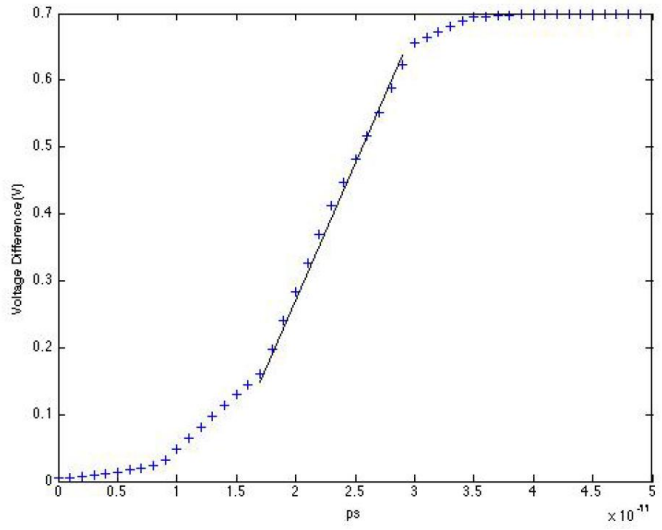


Figure 9: Synchronizer Bi-stable Element Circuit Response

Bi-stable devices within synchronizer flip-flops generally have two sides. This is most easily understood when considering an inverter loop. The loop can be forced into one of two states, but when stable the voltage of one side of the loop is at Vdd while the other is GND.

When both sides are forced to be Vdd/2, the voltage difference between the two nodes is zero. **Figure 9** shows how as the bi-stable element begins to settle, the voltage difference begins to increase (or decrease depending on polarity). The properties of this change are used to calculate τ, as shown in **Equation 2**.

$$\tau = \frac{t_1 - t_2}{ln(V_2/V_1)}$$

Equation 2: Calculation of Simulated τ

3.2 Flip-Flop Initialization

As described, the first step in simulation is forcing the flip-flop under test into metastability. This paper considers three different synchronizer flip-flops which each utilize somewhat different bi-stable element structures. Because of these differences, each flip-flop needed to be initialized in a different fashion. To ensure consistency across the different circuits, all flip-flops were initialized to a standardized Vdd/2.

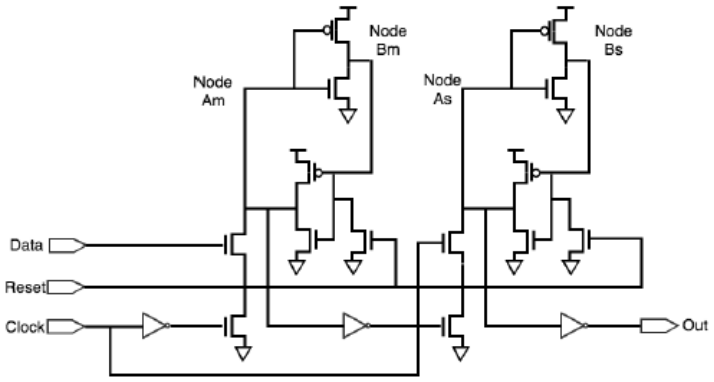


Figure 10: Dynamic Latch Flip-Flop

The first flip-flop considered was the Dynamic Latch Flip-Flop (DLFF), sometimes referred to as a jamb latch flip-flop [50]. The DLFF is so named due to the dynamic nature its

circuit, which requires a separate Reset signal before the device can be re-evaluated (refer to **Figure 10**). While this modification enables the circuit to outperform standard synchronizer flip-flops, it also limits the usability of the circuit in modern devices (which very rarely contain a dynamic reset signal). Since the DLFF utilizes a simple cross-coupled inverter pair, initializing the circuit into metastability is straightforward.

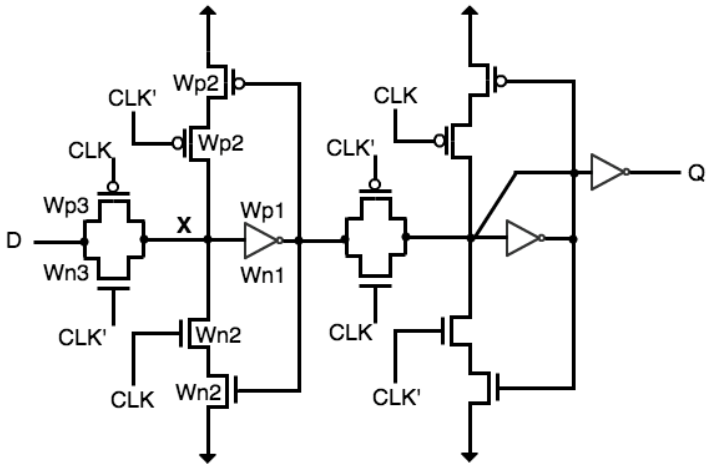


Figure 11: PowerPC Flip-Flop

The second device simulated was the PowerPC flip-flop, recently considered as a suitable sub-threshold voltage synchronizer flip-flop [52]. The PowerPC flip-flop, as shown in **Figure 11**, is initialized by setting the pass gate transistor switch and all Wp2 and Wn2 in the on state. The data input is then set to Vdd/2. The voltage difference is then considered between node X and output of the inverter consisting of Wp1 and Wn1.

The Pseudo-NMOS flip-flop is the third to be simulated, as shown in **Figure 12** [56]. This flip-flop was initialized by setting the q and q bar nodes at Vdd/2, and then setting the clock low such that that mm and mm bar are in the same state. The voltage difference is taken between q and q bar.

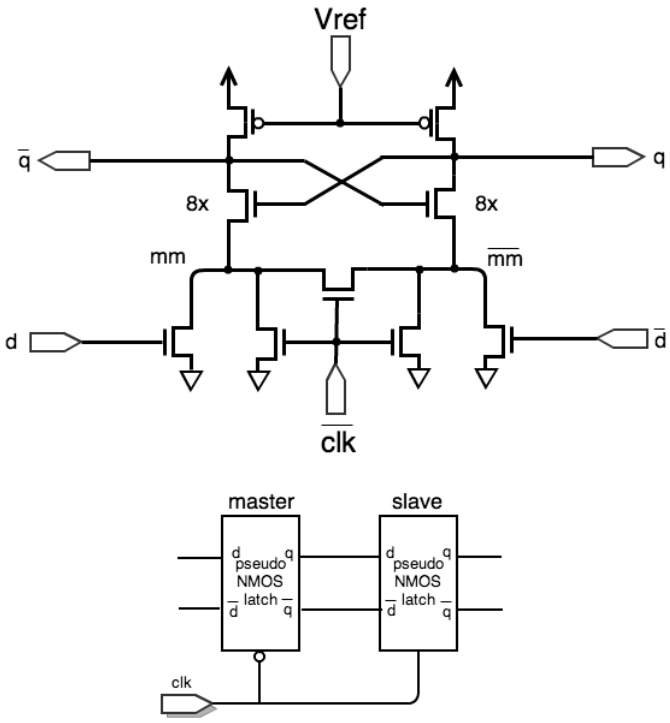


Figure 12: Pseudo-NMOS Flip-Flop

3.3 Tau and FO4 Delay

The link between synchronizer flip flop τ values and FO4 delay has been used in the past to predict synchronizer performance in new technologies. Both factors rely on the device size/strength tradeoff defined by the technology node. As mentioned before however, the research community has disagreed about whether or not this trend continues. Ginosar has proposed that after 65nm, each technology node will see higher τ while FO4 delay will continue to decrease [61]. He justifies this claim by first creating a mathematical relationship between FO4 delay and τ in **Equation 3**.

$$\tau = \frac{\eta}{4}\frac{t_{d,FO4}}{A}$$

Equation 3: τ as Function of FO4

Equation 3 represents τ as a function of FO4 delay, A (inverter gain), and η (derived from a combination of Vdd, Vt, and λ). It was proposed that while FO4 will continue to scale, the inverter gain will not, creating a degradation in τ [61]. Other researchers disagree, and have released results demonstrating τ values which continue to track FO4 delay [56]. Until now however, the effect of FinFETs has not yet been considered. Analysis from UC Berkeley has found that propagation delay in FinFET circuits is independent of electrical width [71]. This serves to maintain the gain of inverter built with FinFET devices, even as they are scaled. The primary factor of consideration relating τ and FO4 delay is inverter gain, and so it stands to reason that τ will continue to track FO4 delay even when FinFET devices are used.

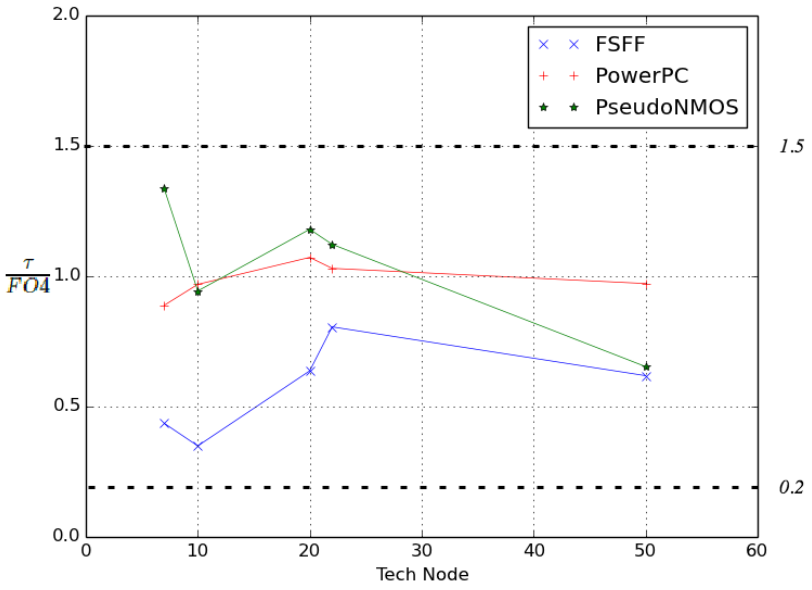


Figure 13: τ/FO4 Results

This analysis is validated by the simulation results shown in **Figure 13**. Using the technique described in the previous section, these simulations were taken for each of the three synchronizer flip flops at nominal Vdd of 0.7V and room temperature of 22°C. The expected range for τ delay has been cited as between 0.2 and 1.5 times the given technology node's FO4

delay [61], and each of the three flip-flops tested fall within this range. While there is some variation between technology nodes, the results still match physically measured results by other researchers for planar technology nodes [56]. The FinFET nodes (7nm, 10nm, and 20nm) stay within the same τ/FO4 range as the values found for planar nodes (22nm, 50nm).

3.4 Performance and Temperature

Most CMOS circuits decrease in effectiveness with an increase in temperature. This is due to the reduced carrier mobility that comes with an increase in particle scattering. However, an additional effect of increased temperature is a decrease in threshold voltage, making devices less effective for a given supply voltage. This is important since synchronizers are especially reliant on threshold voltage. Previous work has verified that the temperature sensitivity for threshold voltage is more important than that of mobility, resulting in synchronizer performance being reduced (an increase in τ) with lower temperatures [48, 72]. FinFET devices are not effected by temperature in the same way as planar devices however. Recent work has found that the previously weaker voltage threshold effect is actually dominant in general logic gates FinFET devices, which led to an increase in logic gate performance at higher temperatures [73]. Unfortunately, this would suggest an especially strong decrease in performance at lower temperatures for synchronizers built with FinFET devices.

This behavior is validated in simulation results, shown in **Figure 14** and **Figure 15**. There is a much more stark degradation of synchronizer performance at lower temperatures when circuits are built with FinFET devices (20nm). When built with planar devices in 22nm technology, the DLFF and Pseudo-NMOS are generally stable over temperature, even with low supply voltages. In fact, the Pseudo-NMOS flip-flop is more dominated by the carrier mobility

effect, resulting in a decrease in τ for lower voltages. A general increase in temperature dependence with scaling can also be observed.

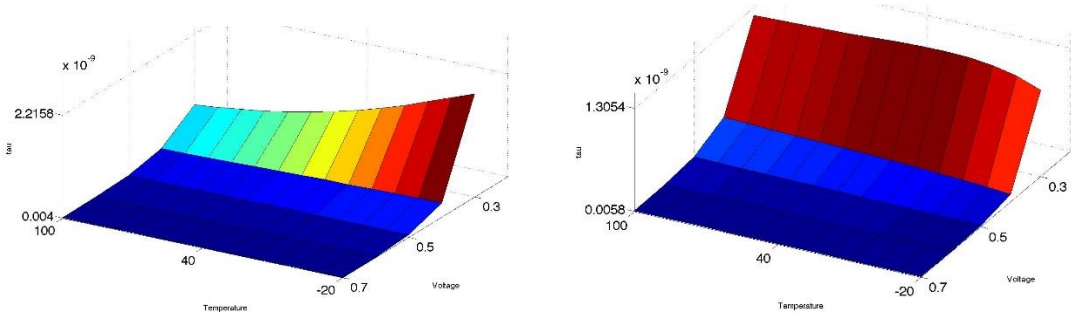


Figure 14: 22nm Planar Technology, Temp vs. Vdd (FSFF left, Pseudo-NMOS right)

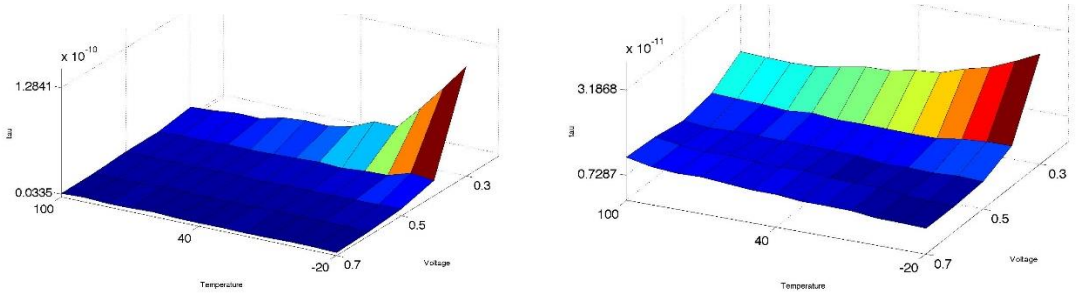


Figure 15: 20nm FinFET Technology, Temp vs. Vdd (FSFF left, Pseudo-NMOS right)

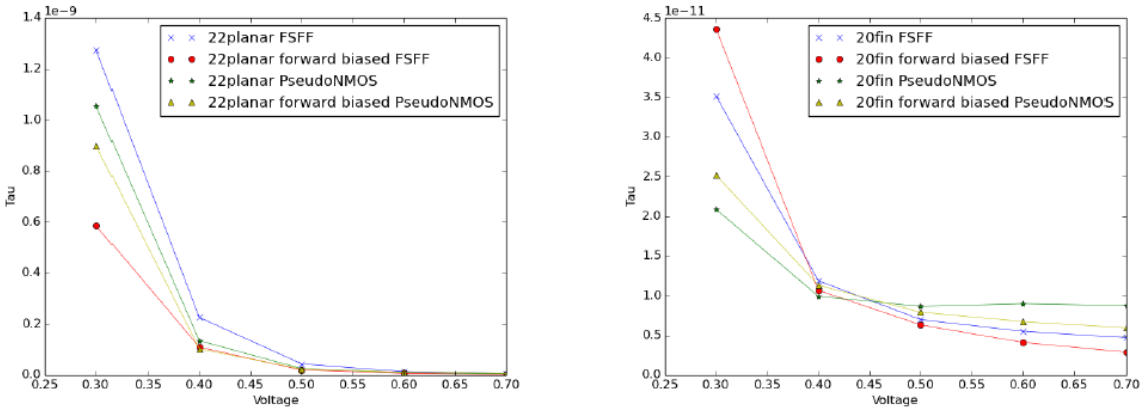
These results also show that increases in τ due to lower temperatures can be prevented with a higher supply voltage. This further demonstrates that the dominant factor in synchronizer dependence on temperature is the change in threshold voltage. In modern SoCs, maximum restrictions on the supply voltage are generally enforced to prevent an increase in power consumption which leads to heat. Thankfully this means that raising supply voltage in a low temperature state to prevent synchronizer performance degradation is not unreasonable. Data for the PowerPC flip-flop was found to be very similar to the DLFF, and so is not displayed here to conserve space.

3.5 Forward Biasing

Designers must also consider how FinFET devices will affect the trade-offs between various synchronizing flip-flop types. When all circuits were implemented with FinFET devices, the Power-PC flip-flop continued to be out-performed by both the DLFF and Pseudo-NMOS. However, the choice between the DLFF and Pseudo-NMOS becomes more complex when FinFET devices are used.

The DLFF was one of the first synchronizing flip-flops to be proposed, and relies on inverter feedback loops to settle metastability [50]. When near-threshold operation became more common as a design practice to reduce power, it was shown that inverter loops provided very poor performance at low voltages. For this reason, the Pseudo-NMOS flip-flop was proposed as a synchronizing flip flop without the need for an inverter feedback loop [56]. Around the same time, another solution was proposed which applied forward body bias to increase DLFF performance at low voltages. It was demonstrated that forward body biasing increases the transconductance of the bi-stable inverter pair, resulting in improved performance [49]. While the Pseudo-NMOS has been compared to the unbiased DLFF, it has not been compared when using this technique. It is important to determine which design outperforms the other at low voltages as well as if the benefit of biasing still exists when using FinFET devices.

For 22nm planar technology, **Figure 16** shows that in low voltages, the DLFF is outperformed by the Psuedo-NMOS flip-flop. When forward body biasing is applied however, the DLFF τ is significantly decreased, surpassing the biased Pseudo-NMOS flip-flop which only decreases slightly. This demonstrates that when designing in planar technology, forward biasing a DLFF is the best solution for near-threshold operation.



(a) 22nm (Planar)

(b) 20nm (FinFET)

Figure 16: Voltage Sweep of Non-Biased and Forward Biased DLFF and Pseudo-NMOS

However, FinFET technology presents an interesting challenge for body biasing techniques. Since FinFET devices sit on top of the bulk, rather than being deposited inside the bulk (as in planar technologies), the increase in the transconductance can no longer be achieved. As can be seen in **Figure 16** showing simulation data for 20nm FinFET, both the DLFF and Pseudo-NMOS receive an increase in τ rather than a decrease in τ when forward biasing is applied. This effectively removes biasing as a suitable technique for near threshold operation.

It can also be seen however that DLFF maintains a significant advantage over the Pseudo-NMOS at nominal supply voltages. This presents a switching point (around 0.45V for 20nm) when above that voltage the DLFF is superior, and below that voltage the Pseudo-NMOS is superior. This is because inverter feedback loops offer a significant advantage at higher voltages. If voltage must decrease, this advantage wears off and the Pseudo-NMOS achieves the lowest τ for near threshold operation.

CHAPTER 4

DESIGN OF DVFS TECHNIQUES FOR PREDICTIVE SYNCHRONIZATION

As described in the background section of this thesis, each predictive synchronizer requires certain assumptions. For example, the phase estimation system within the Even Odd Predictive Synchronizer (EOPS) assumes that the value of the measured frequency ratio and detection interval remain constant. When DVFS is applied, these values need to be re-measured for each change in clock frequency. Naturally, this measurement delays communication for every change in frequency, and is particularly detrimental for systems that apply DVFS aggressively at a fine-grain. The following two sections present solutions for two DVFS scenarios: gradual frequency change, and fast (instant) frequency change.

4.1 Gradual Frequency Change

There are a number of reasons why gradual frequency might occur. For example, changing certain parameters in phase locked loops will result in a slow change in output from the previous frequency to the target frequency. Also, increasing the speed of a clock with a large distribution tree (such as in a large scale GPU) must be done slowly to avoid voltage drop in the power supply. Therefore, when gradual frequency change does occur, special care must be taken. We propose that instead of waiting until clocks have settled on their final values to re-measure, the system will measure continuously.

Both the measured frequency ratio and detection interval needed for the EOPS are represented as digital fractions. The most significant bit is to the left of the decimal point, while the remaining bits are to the right of the decimal point. These digital fractions have a finite number of bits, meaning that their accuracy is limited. This accuracy limitation is accounted for

in the design by being included when calculating the upper and lower bound phase estimations [69]. For example if 10 bits are used, then the measured values can vary by 2^-9 without any harm in functionality. Because the system is built to handle this inaccuracy, another perspective would be that there is a certain range (the range of inaccuracy) in which the actual values of these factors can vary without causing failure in the system. If periodic measurement updates were sent to the phase estimator, and the actual values of the measured variables did not change so much as to go outside of this acceptable range in between updates, then the whole system can continue communication through clock frequency change.

To realize this idea, it is first necessary to generate periodic measurement updates. Basic measurement updates can be achieved by restarting both the frequency ratio and detection interval measurement circuits (same circuits in the Even Odd Synchronizer) each time they finish a measurement. Each time these circuits finish, they will provide updated measurements to the system, allowing it to adapt as clocks change. Unfortunately, the frequency of these updates is limited by the number of cycles the circuits need to finish the measurement.

To assess the adequacy of this design, the maximum speed that a clock can change in frequency in between updates may be calculated. This is the limiting factor that might prevent this design from practical application. If the measured fractions have b bits, then the range of inaccuracy is $\pm 2^{-(b+1)}$. A typical system might have a b of 10, therefore allowing for a variation of ± 0.000488. If we assume that both clocks are 1GHz, for the sake of illustration, then this means that one of these two clocks may change its frequency ±488 kHz and still stay within the measured range.

The time between updates is also determined based on the number of bits used to represent the measured fractions. This is because the number of bits determines the size of the counters that must overflow in the measurement blocks. Time between measurements is equal to 2^b receiver clock cycles, i.e. we must wait 2^{10} = 1024 cycles in between each update for our typical system. This may also be called the cold start time, since this time must be spent on start up to begin estimating phases. When N measurement blocks are used the time between updates is (cold start time / N) as shown in **Figure 17**.

It is important to note that the measured values in each update will be based on the 1024 cycles before the update (the period which the system was measuring). As a result, these updates do not represent the frequency ratio and detection interval value at the exact time of the update. If we assume that the clock frequency is changing at a constant speed between updates, each measured update will represent the frequency ratio and detection interval exactly half way between the start and stop of measurement. This measured value is used by the system until the next update. The time that the frequency needs to stay in the acceptable range is (time between updates) + (time between updates/2). For our typical system, we know that the receiver frequency needs to stay in the acceptable range for 1536 cycles. While this may be acceptable for some applications, others require faster frequency change. This necessitates faster measurement updates.

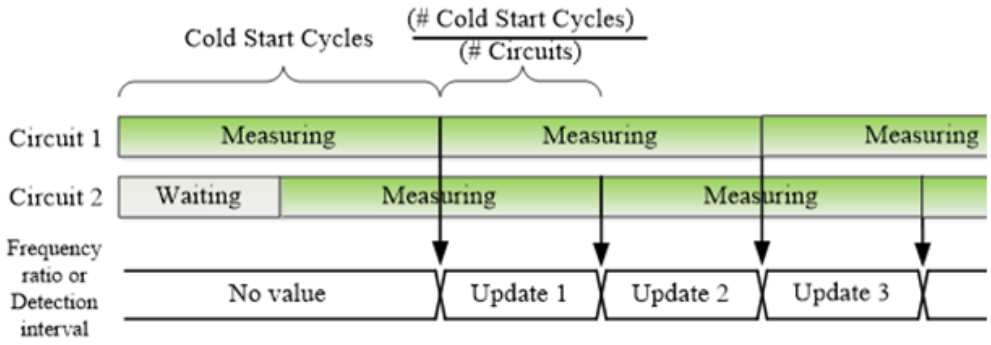


Figure 17: Continuous Measurement Strategy at Doubled Rate

Faster measurement updates may be achieved by including multiple measurement circuits in the design. These circuits are started in a staggered fashion and run in parallel, effectively hiding measurement latency as shown in **Figure 17**. Both the frequency ratio and detection interval can be measured in this manner.

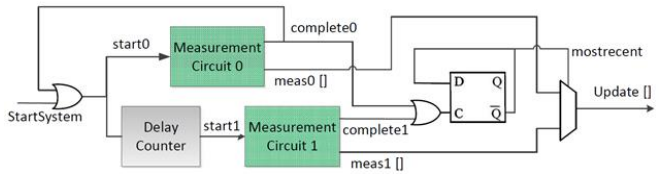


Figure 18: Continuous Frequency Measurement Circuit

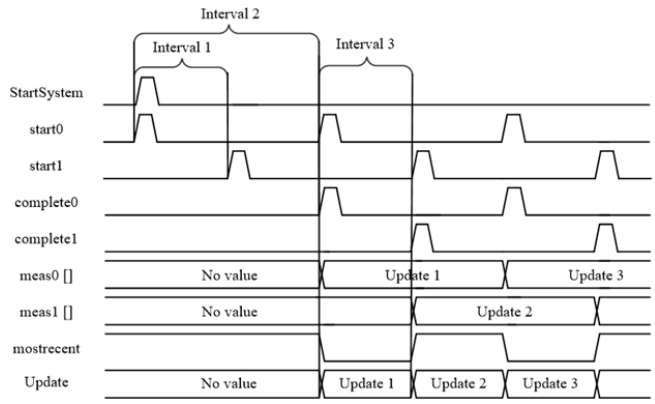


Figure 19: Continuous Frequency Measurement Timing

The circuit used to achieve this technique is shown in **Figure 18** while the timing diagram in **Figure 19** shows its behavior. The circuit is started with a *StartSystem* pulse, which initiates Measurement Circuit 0. This start pulse also enters a delay counter which delays the start signal by half of the measurement completion time (Interval 1). After this delay, Measurement Circuit 1 is started. Eventually, Measurement Circuit 0 finishes (measurement time is shown by Interval 2) and its output value is posted to *Update* as the most recent line is pulled low. This completion causes *start0* to be asserted, beginning the process all over again. After time interval 3, Measurement Circuit 1 finishes. This parallel circuit allows for updates to occur twice as fast as a single measurement structure, doubling the maximum frequency ramp speed. This method could be used for any number of additional measurement circuits, eventually allowing for an updated measurement every clock cycle if so desired.

4.2 Instant Frequency Change

In systems where clocks may change frequency very quickly, such as when digital frequency dividers are used, even parallel measurement circuits may not be fast enough. In this section, we present a solution which directly informs the predictive CDI when frequencies change and what they are changing to. When a frequency transition takes place the CDI pauses communication, reads in the new frequency value, calculates the new frequency ratio and detection interval directly (instead of measuring), and then waits for both domains to track before resuming communication.

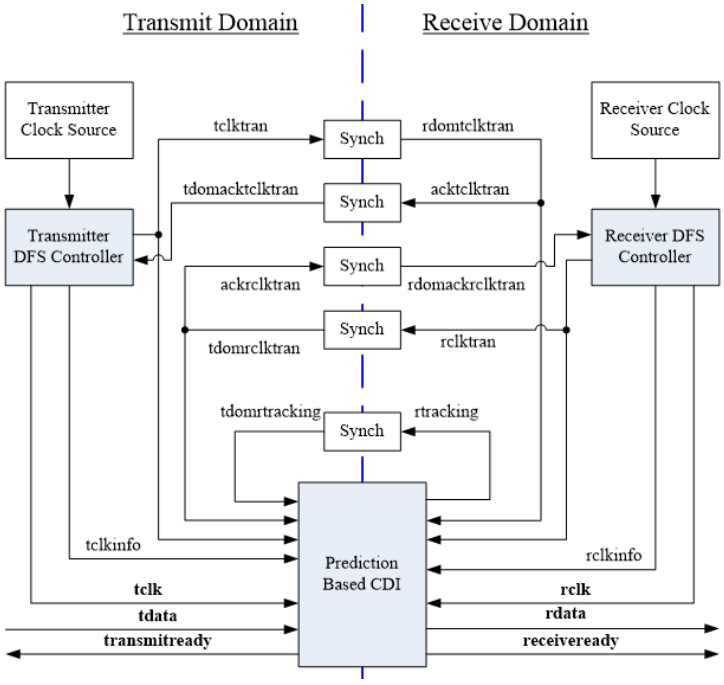


Figure 20: High Level Diagram of Fast DVFS Strategy

The system shown in **Figure 20** uses a number of control signals to communicate the status during the process of a fast frequency change. Each DVFS controller supplies a clock (generated from the controller's clock source) and information about the clock to the Prediction Based CDI. This information could either be the specific frequency of the supplied clock or a code representing which frequency the clock currently is.

Each DVFS controller also has an output for requesting a frequency transition (*rclktran, tclktran*) and an input so that the controller knows when the opposite clock domain has acknowledged the frequency transition (*tdomacktclktran, rdomackrclktran*). These signals cross the clock domain boundary by using brute force synchronizers since the predictive CDI is off-line during frequency transitions. There are also signals for *rtracking* representing when the receive domain is ready. The two domains' clocks, data, and flow control signals (shown in bold at the

bottom of the figure) are the signals used during normal operation when communication is resumed.

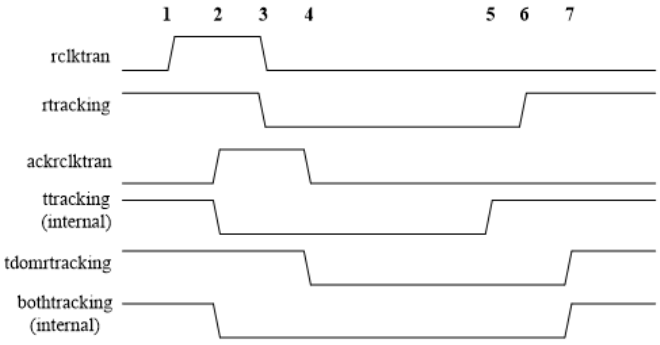


Figure 21: Fast Change in Receiver Clock Frequency Timing

Granted, this technique does require the CDI to pause communication for a certain period of time unlike the continuous measurement technique. To understand how long communication would need to be paused, we consider the timing behavior of the system. As seen in **Figure 21**, the receive domain informs the system of its need to change its frequency by asserting *rclktran* at time 1 as well as updating its *rclkinfo* to inform the CDI of the needed running frequency.

The *rclktran* signal propagates through the brute force synchronizer to the transmit domain by time 2. At this time the transmit domain acknowledges the change via a direct connection to the synchronized transition request signal and de-asserts its tracking flag (*ttracking*). Naturally this implies that both domains are no longer tracking and communication is paused. During this period *transmitready* is set low and *tdata* is kept at a "null" value which the receive domain will ignore.

For the system to resume, the following steps are taken. First, when the transmit domain acknowledges the change in frequency (time 2), it samples the new *rclk* info. This data is known to be constant since it has been a number of synchronizer cycles since the last change

(the *tdomrclktran* signal acts as a data ready signal). The data is then used to calculate both the frequency ratio and detection interval necessary for phase prediction in the transmit domain. The edge detector in the transmit domain is also disabled to prevent the system from re-tracking before the receiver's clock is actually changed.

When the asserted *ackrclktran* signal is successfully synchronized to the receiver domain at time 3, the receiver DVFS controller completes the sequence by changing the receiver clock's frequency. This acknowledgement also triggers the receive domain to calculate its frequency ratio and detection interval which has been updated based on the new *rclk* information. With a new *rclk*, the phase estimator needs to re-track, so the *rtracking* flag is also de-asserted. Having completed its transition, the receiver clock's *rclktran* signal is de-asserted. At time 4 the de-assertion of both the *rclktran* and *rtracking* signals are received in the transmit domain. By this time instant, transmit domain knows that the receiver clock has been changed which prompts the transmit side phase detector to be re-enabled.

Once the receiver clock has been changed and the phase estimators are enabled in both domains, it is necessary to wait for the two domains to begin tracking. At time 5 the transmit domain begins tracking and at time 6 the receive domain begins tracking. When at time 7 the *rtracking* signal is synchronized to the transmit domain, the transmitter can resume communication.

The obvious question is how much time is necessary between stopping and starting of communication after a frequency change. The time range where communication is stopped starts at time 2 and restarts at time 7. During this time, control signals are passed between clock domains and the system waits for both domains to start tracking. Although different frequencies change the exact delay, the delay from two synchronizers comprise 8 cycles if 4 cycle synchronizers are assumed. Our experiments have shown that waiting for tracking rarely

exceeds 12 cycles. This means that the system can re-start communication after only 20 cycles. This is a considerable improvement relative to the thousand cycles necessary for full re-measurement.

4.3 The Locally Controlled CDI

It is important to note that the usefulness of these DVFS enabled phase prediction techniques is not limited to the EOPS. Here, we describe an alternative Clock Domain Interface (CDI) which uses these DVFS enabled predictive phase estimator circuits. In this CDI, flow control is managed locally with no need for a FIFO.

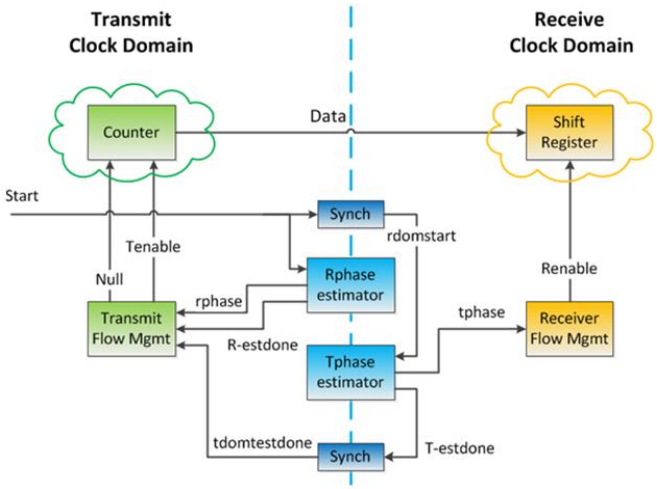


Figure 22: The Locally Controlled CDI

As seen in **Figure 22**, the Locally Controlled CDI uses a phase estimator for each domain. The phase estimates are then sent to the flow management logic which in turn controls the flow of data out of the transmit domain and into the receive domain. The clouds represent the entire transmit and receive domains respectively. The counter in the transmit domain and shift register in the receive domain are only used for testing the system.

The Locally Controlled CDI has two different modes of operation. One protocol is used when the receiver clock is faster while another protocol is used when the transmit clock is faster. The system decides which of these two protocols should be active by checking the ratio between the two frequencies provided by the phase estimator (this is one of the necessary values provided by the DVFS techniques shown in previous sections).

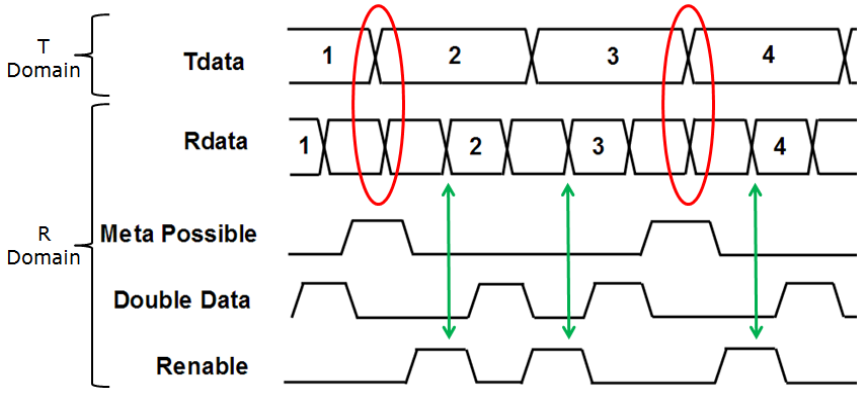


Figure 23: CDI protocol when the receive clock is fastest

Flow control and metastability avoidance is all contained within the receive domain when the receive clock is fastest. This protocol generates three control signals: *Meta Possible*, *Double Data*, and *Renable*. The signal *Meta Possible* informs the system when sampling on a particular edge has the risk of resulting in metastability. To avoid metastability the system needs to avoid sampling data when that data is changing. This is defined as the case when the rising edge of the transmit clock violates the receive domain's setup time or hold time. This situation can be avoided by checking the calculated phase estimates while preparing to sample.

As described in previous sections, the prediction circuits give upper bound and lower bound phase estimates as binary fractions of the transmit clock period. This binary fraction includes bits to the left and to the right of the decimal point. The bits to the right of the decimal point represent where the rising edge of the receive clock falls, within a single transmit clock

cycle. The flow control logic checks these bits to see if the phase estimate overlaps the predefined keep out region around the rising edge. **Figure 24** shows examples of safe sampling (a) and unsafe sampling (b) on transmit domain phase circles. The X and 1-X terms here are predefined as the keep out region (the setup and hold time). When there is overlap, the *Meta Possible* signal is asserted since the edges may be too close. This can be seen in **Figure 23** where the ovals point out the transmit clock and receive clock are too close. Each of these edges is accompanied by a high *Meta Possible* signal.

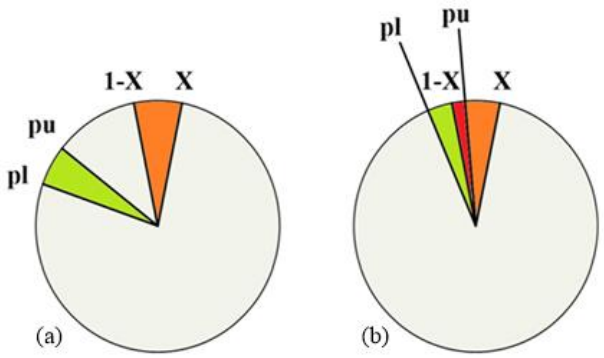


Figure 24: Phase Estimation for Safety Detection

Of course, flow control is also important for a CDI. In the case where the receive clock is fastest the flow control logic must prevent the receiver from sampling data that it has already sampled. This is accomplished with the *Double Data* signal. For this the bits to the left of the decimal point of the phase estimate are utilized. These bits represent which transmit clock cycle the possibly sampled data belongs to. To generate the *Double Data* signal, the phase from the current receive clock rising edge and the phase from the previous receive clock rising edge are compared. If the two phases are in the same transmit clock cycle, then the data was already sampled on the previous edge and there is no need to sample again. If both the *Meta Possible* and *Double Data* signals are low, then *Renable* is high and the receive domain knows that it is the right time to sample.

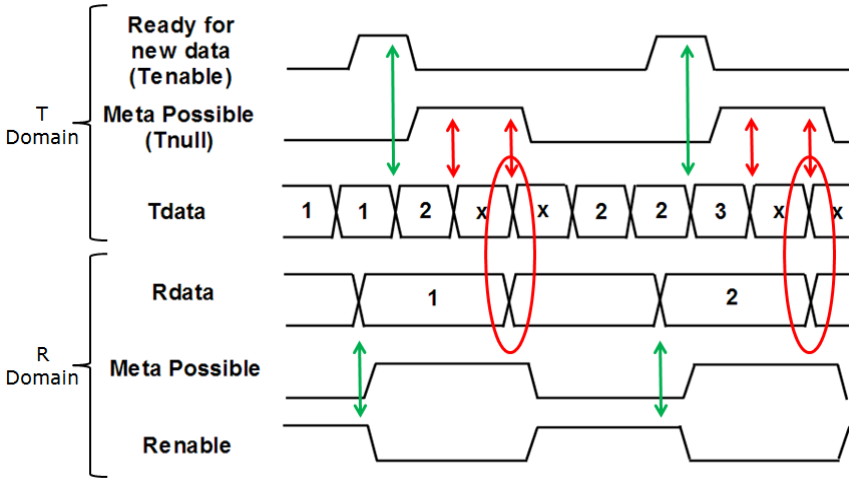


Figure 25: CDI Protocol when the transmit clock is fastest

A different protocol is used if the transmit clock is faster than the receive domain (see **Figure 25**). For this protocol flow control signals are used in both domains. In the receive domain, metastability is predicted in the same way as previously described. Flow control is managed in the transmit domain for this protocol, setting *Renable* high as long as *Meta Possible* is low.

In the transmit domain, both an enable signal as well as a *Tnull* signal are used. The tenable signal represents when the receive domain is ready for new data. This is managed in a similar manner to the *Double Data* signal in the receive clock faster protocol. If the phase estimates of the current transmit clock edge and the previous transmit clock edge are in the same receive clock cycle, then the receive domain has not yet had the chance to sample the data.

Metastability management in the transmit domain is handled differently for this protocol. Data would be lost if the transmit and receive domains disagreed about if a given receive edge would cause metastability. Disagreement is possible since the domains do not send control signals between domains during operation and manage their own phase estimators with

slightly different properties. To prevent this problem from occurring, the transmit domain nulls out data before and after an edge that it detects may possibly cause metastability. The transmit domain also increases its keep-out region slightly to make sure that the transmit domain will always be more protective of possible metastability than the receive domain. This way the only case for a disagreement is when the transmit domain expects metastability but the receive domain does not. In this situation the receive domain only sees the null value and knows to ignore this data which is guaranteed to be safe.

It is also important to note one of the weaknesses of this CDI. This CDI cannot be used when the transmitter and receiver clocks are rationally related as tracking is impossible when clocks are out of phase since rising edges will never occur close to each other. With no rising edges close to each other the system can never calibrate. This limitation restricts the range of frequencies that the CDI can be used with. Also, for the purposes of simplification, the implemented CDI is limited to working with clocks that are no more than twice the other or less than half the other. This could be changed however by including more bits to the left of the decimal point when calculating phase estimates and making few changes to the logic.

4.4 RTL Implementation

These designs have been implemented and verified in Verilog RTL. Synthesis was performed using TSMC 28nm technology standard cells. The test setup consisted of a 5 bit counter in the transmit domain to generate sample data and a shift register in the receive domain to hold the incoming data (as shown in Figure 8). The transmit clock's frequency was set at 1GHz while the receive frequency was swept from 500MHz to 2GHz at 1GHz intervals. This was repeated twice, once with gradual changes in frequency using the circuit shown in Figure 4, and then once with instant changes in frequency using the circuit shown in Figure 6.

Setup and hold time checks were used to verify no risk of metastability during all of operation. The contents of the shift register were monitored to verify that the data was passed without doubles or misses. Transfer of data was successful with the notable exception of rationally-related frequencies. Completion time was also measured to determine the throughput of the system.

Metric	*Brute Force*	*EOPS*	*Locally Controlled*
Latency	4 cycles	**1 cycle**	**1 cycle**
MTBF	Limited	**~Infinite**	**~Infinite**
Layout Size	6606 μm^2	4473 μm^2	**3122 μm^2**
Throughput	**Maximum**	**Maximum**	~15% Reduction

Table 1: CDI Comparison

As shown in **Table 1**, when compared to the Brute force CDI (an asynchronous FIFO using brute force synchronizers) and the EOPS CDI (an asynchronous FIFO using the EOPS), the Locally Controlled CDI has the best latency, Mean Time Between Failures (MTBF) and layout size. Both the EOPS CDI and the Locally Controlled CDI have the same latency and MTBF since they both use the same phase prediction circuits. Latency values shown in this table represent the maximum possible latency. All latencies could vary by a single cycle, since absolute latency depends on when the receiving domain samples within the transmit domains clock period.

The MTBF for the EOPS CDI and the Locally Controlled CDI is near perfect since the only brute force synchronizers (the source for a possible metastability failure) in these CDIs are in the measurement circuits. These brute force synchronizers are out of the critical path and therefore can be made arbitrarily long to achieve almost infinite MTBF. Since the critical path for the brute force CDI goes directly through its brute force synchronizer it must make a compromise between reliability and latency resulting in a sub-optimal MTBF. In terms of layout area, the EOPS CDI is smaller than the brute force CDI since its reduced latency allows it to have a smaller

FIFO, while the Locally Controlled CDI has no need for a FIFO and is therefore even smaller. The one disadvantage to the Locally Controlled CDI is its reduction in throughput, which was verified experimentally. This reduction in throughput is due to the system not sampling data when metastability is possible, effectively missing a cycle.

It should also be mentioned that while the Brute force CDI demonstrates poor latency, MTBF, and area, it does not require any extra circuitry for changes in frequency. Area estimates shown in Table 1 do not include area for extra measurement circuits for gradual frequency changes. Each additional measurement circuit (including both frequency ratio and detection interval) adds 306 μm^2 to the layout size.

The effect of jitter in each domain on the system must also be considered. If precautions are not taken, it is possible for phase predictions to exceed tolerance. If the nature of the jitter is known however, then this can be prevented by artificially increasing the size of the measured detection interval [69]. This technique applies for all uses of the phase prediction circuits (EOPS, Locally Controlled CDI, etc.).

The Locally Controlled CDI's inability to operate with clocks that are rationally related might also be mitigated. To allow for operation over a wider range of frequencies, the Locally Controlled CDI could be augmented with a CDI specifically design for rationally related frequencies [66, 67]. Also, the work presented in this paper manages changes in clock frequencies, not changes in voltage. For this, level shifters must be applied to incoming and outgoing signals.

CHAPTER 5

SYSTEM LEVEL IMPACT OF PREDICTIVE SYNCHRONIZERS

Most synchronizer work focuses on designing synchronizers with an improved latency-reliability tradeoff at the circuit level. Of course, the reason why designers are concerned with the latency and reliability of individual synchronizers is because of the effect that they have on the latency and reliability of the system as a whole. Despite this, the high level impact of synchronizers is rarely studied. In fact, most high level designers ignore the effects of synchronizer circuits in an effort to simplify their simulations. This largely a holdover from when chips contained only a single core, and synchronization was primarily limited to between the core and the cache. Others make the unreasonable assumption that core level DVFS will not be used, allowing the entire SoC to operate on a single clock. As discussed before, a NoC connected many-core SoC must synchronize at many places on-chip. Thus, synchronizer latency and reliability has a greater effect on the overall system. The work shown here quantifies the system level improvement in both latency (via simulated benchmark speedup) and reliability (via MTBF calculations) that can be achieved with the use of predictive synchronizers.

5.1 NoC Organization

When considering synchronizer impact at the system level, it is first necessary to identify which kind of system is being studied. There are many different kinds of NoC as described in the background section describes. For the purposes of this study, only mesh networks will be considered. Clock domains can be organized in a number of different ways within a mesh network however. Naturally, more clock domain interfaces means more synchronizers, and so the NoC organization is crucial to overall system performance. In fact, some NoC structures have been proposed with the purpose of reducing the necessary

synchronizations per message transmission. **Figure 26** circles where the clock domain crossings would be if using one of the three different NoC structures considered. To demonstrate this, an example packet is sent from the bottom left core to the top center core.

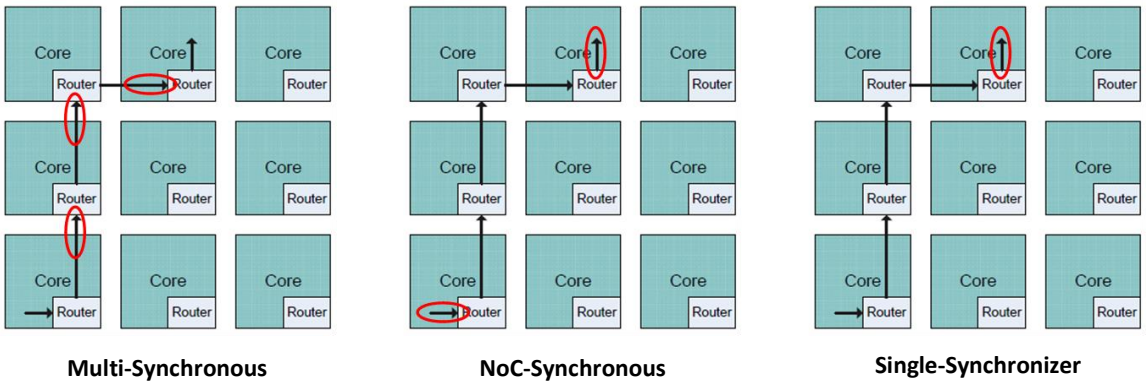


Figure 26: NoC Structures Considered

Routers in the most naïve mesh NoC would operate using the clock of their local core. This would mean that communication between each of the routers would require synchronization. In the example shown in **Figure 26**, 3 synchronizer delays can be observed. In this document we will call this organization method Multi-Synchronous.

Another possible NoC organization would be to operate the NoC with a single clock [30]. We will refer to this organization as NoC-Synchronous. The benefit of this design is that it only requires synchronization when entering the first router, and when exiting the last router (as shown in **Figure 26**). This organization method is not common or even practical however since it requires global clock distribution.

Recent work has taken the reduction of synchronization steps even further. Source synchronous designs [40], and designs which implement custom high-speed global interconnect [39], only require that synchronization take place at the output of the destination router. These kinds of NoC's are referred to as Single-Synchronizer, and an example communication is also shown in **Figure 26**.

Significant work has also been done on asynchronous NoCs [35, 37, 38], but predictive synchronizers require periodic clock domains to operate. Since this work is interested in the benefit of predictive synchronizers, asynchronous NoCs are not considered here.

5.2 Performance Simulation

To quantify the performance improvement granted by the use of predictive synchronizers, high level simulations were completed by modifying the Graphite simulator [70]. The Graphite simulator was chosen as it is an efficient open source cycle-accurate multi-core simulator. In addition to core models, Graphite also offers various mesh NoC models which can be used to acquire realistic performance data. It was important to have a simulator with both realistic core and network models since it was necessary to test NoC configurations with real core generated traffic.

5.2.1 Synchronizer Latency vs CDI Latency

To understand how synchronizer latency was added to the simulator, it is first important to understand latency in the context of a CDI. We have chosen asynchronous FIFOs as our CDI since they are the most common [47]. As shown in **Figure 27**, asynchronous FIFOs contain more than just a table of values. The logic shown in blue is for flow control (preventing under or overflow of the FIFO), which is managed by updating *Full* and *Empty* signals within the *Transmit* and *Receive* domains respectively. To determine if the FIFO is full or empty, the Head and Tail pointers must be compared. Because the *Head Pointer* is in the Transmit domain, and the *Tail Pointer* is in the Receive domain, these pointers must always be updated in the opposite clock domain. So, synchronizers are used to send the *Head Pointer* to the Receive domain and the *Tail Pointer* to the Transmit domain. This design allows the Transmit domain to transmit data

whenever it sees that the FIFO is not full, and the Receive domain to receive data whenever it sees that the FIFO is not empty.

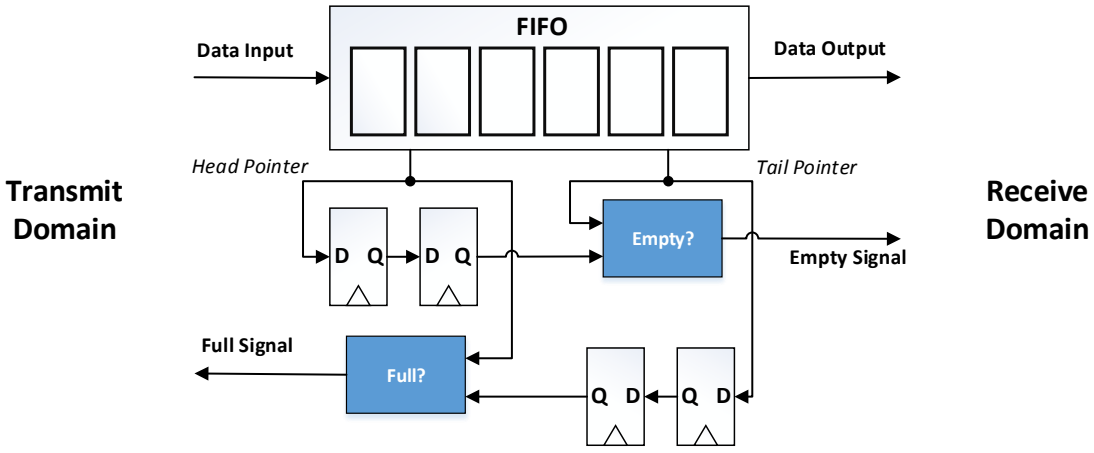


Figure 27: Interior Structure of Asynchronous FIFO

So, we can see that an increase in synchronizer latency does not necessarily mean an increase to CDI latency. First, let us consider the case where the FIFO is empty, and the Transmit domain inserts data into the FIFO. In this example, the Receive domain doesn't know that there is data in the FIFO until the *Head Pointer* is updated in its domain. So for the empty FIFO case, the CDI latency is the same latency of a single synchronizer. However, what if we already have 4 pieces of data within the FIFO, and our synchronizer latency is only 2 cycles? Just as before, the Transmit domain inserts the data into the FIFO and updates the *Head Pointer*. In this case however, by the time the Receive domain reaches the piece of data in question 4 cycles have passed, two cycles after the Head Pointer was updated to represent the new data. So, the synchronization latency has effectively been hidden by the longer queuing latency. This means that synchronizer latency actually enforces a minimum CDI latency, rather than being a direct increase to latency regardless of FIFO content.

5.2.2 Graphite Simulator Changes

Within the simulator code, core models were kept the same while the NoC models were changed to include synchronization latency. The Graphite electronic mesh hop-by-hop network model was chosen as a starting point, as this model already included queuing latency. It is important to note however, that Graphite does not actually implement router queues. Instead, latency is calculated by using a link contention history table. Thankfully our understanding of FIFO behavior allows synchronizer latency to be injected simply by enforcing a minimum delay per CDI. So for each CDI within the network, a statement was added which replaced the contention latency with the synchronizer latency if it was smaller than the synchronizer latency (thereby enforcing the minimum).

The number and location of CDIs depends on the structure of the NoC in question. As described in the section on NoC Organization, the three NoCs considered here are Multi-Synchronous, NoC-Synchronous, and Single-Synchronizer. For Multi-Synchronous, CDI latency was added within every router. Thankfully this was relatively simple due to Graphite's object oriented nature. A single router model is instantiated for every node within the NoC, so CDI latency was injected into this router model. Graphite also distinguishes between injection routers (where the data starts) and standard node routers, so this portion of the code was used for NoC-Synchronous and Single-Synchronizer. Instead of adding CDI latency for every router, either one (for single synchronizer) or two (for NoC-Synchronous) synchronizer latency stages were added at the injection router. Although Graphite does allow for DVFS to be implemented, all cores were set to 1GHz and CDIs were implemented as if all cores were in separate clock domains. This was done to ensure that all variation in execution time was due to synchronization, not changes in DVFS protocol.

5.2.3 Simulation Results

To ensure realistic network traffic, core models were used to execute the SPLASH benchmark suite. Five different executions were run for each benchmark: No synchronization latency, Multi-Synchronous with Brute-Force Synchronizers, NoC-Synchronous with Brute-Force Synchronizers, Single-Synchronizer with Brute-Force Synchronizers, Multi-Synchronous with Predictive Synchronizers, NoC-Synchronous with Predictive Synchronizers, and Single-Synchronizer with Predictive Synchronizers. 64 cores were used for all simulations, brute-force synchronizers were assumed to be 4 cycles of latency, and predictive synchronizers were assumed to be 1 cycle of latency. The predictive synchronizer of choice is a DVFS enabled EOPS.

As could be expected, we observed a significant execution time reduction when using predictive synchronizers. This benchmark directly translates to a system wide performance improvement. For per-benchmark speedup achieved by predictive synchronizers, see **Figure 28**. Speedup numbers are calculated by comparing execution times of systems using brute-force synchronizers and predictive synchronizers. Geometric means were calculated for each of the three NoC types using the zero synchronizer latency results as a base. The ocean benchmarks demonstrated a significantly larger speedup due to their frequent long distance inter-core communications. Water spatial and volrend relied significantly more on local and intra-core communication however.

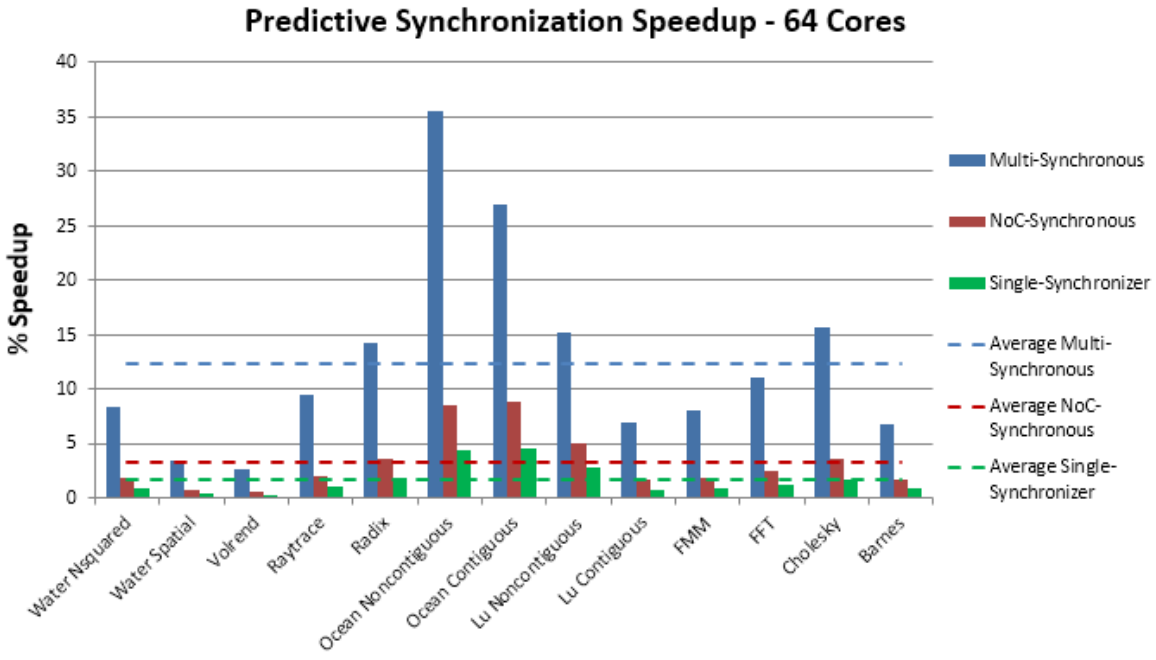


Figure 28: 64 Core Benchmark Speedup When Using Predictive Synchronizers

As can be seen in **Table 2**, the Multi-Synchronous NoC has the highest speedup of roughly 12%. This is to be expected, as we have seen that there is simply more synchronization latency to improve within the Multi-Synchronous NoC. The NoC-Synchronous and Single-Synchronizer NoCs have also been improved, but to a lesser extent. This is acceptable however, as we will see that significant reliability improvements can also be achieved in addition to the small performance improvement.

	Average Predictive Speedup	Geometric Standard Deviation of Brute-Force	Geometric Standard Deviation of Predictive
Multi-Synchronous	12.310 %	1.111	1.030
NoC-Synchronous	3.248 %	1.035	1.010
Single-Synchronizer	1.684 %	1.019	1.006

Table 2: Average Speedup Due to Predictive Synchronizers

5.3 Reliability Calculations

Predictive synchronizers transcend the previously unavoidable tradeoff between synchronizer latency and reliability by moving brute force synchronization off the critical path. For this reason, there is a significant reliability benefit when using predictive synchronizers. As discussed in the introductory section on synchronizers, the MTBF of a single brute-force synchronizer is a factor of the physical synchronizer flip-flop properties (τ and T_w), the number of flip-flop stages (which determines S), and the frequencies of both clock domains (F_D and F_C). If we define a SoC as failing if even one of its many synchronizers fails, then basic probability tells us that the MTBF of a full SoC is the MTBF of a single synchronizer divided by the number of synchronizers on-chip (K).

$$MTBF\left(K_{synchronizers}\right) = \frac{e^{\frac{S}{\tau}}}{T_w * F_D * F_C * K} = \frac{MTBF(1)}{K}$$

Equation 4: MTBF of Multi-Synchronizer SoC

So, when considering the reliability of a full system, it is necessary to know how many synchronizers there are in the full system. Firstly, both commonly used asynchronous FIFO CDIs and predictive CDIs use brute force synchronizers. Standard asynchronous FIFO CDI's contain two synchronizers per CDI. These synchronizers are used to pass the head and tail of the asynchronous FIFO between clock domains. Meanwhile, the predictive circuits used in the EOPS and Locally Controlled CDI contain 16 brute-force synchronizers.

Given that a high MTBF is desirable for the SoC, and that MTBF can be increased by reducing the number of synchronizers on chip, it might seem that the asynchronous FIFO (with less synchronizers per CDI) would always beat the predictive CDI. This is not the case however, since MTBF is also a factor of the number of stages used in the brute-force synchronizers. Not only does the number of stages affect MTBF exponentially rather than linearly, it can be set

arbitrarily high in predictive CDI's since they are off the critical path (and therefore the additional latency is irrelevant to the latency of the CDI).

CDI Type	NoC Type	One Synch MTBF in years	Total Synchs in System	System MTBF in years
Brute Force	Multi-Synchronous	1,492	448	3.33 E 0
Brute Force	NoC-Synchronous	1,492	256	5.83 E 0
Brute Force	Single-Synchronizer	1,492	128	1.17 E 1
Predictive	Multi-Synchronous	32,858,305	3584	9.17 E 3
Predictive	NoC-Synchronous	32,858,305	2048	1.60 E 4
Predictive	Single-Synchronizer	32,858,305	1024	3.21 E 4

Table 3: 64 Core System Level MTBF

A quantitative study on SoC MTBF has been completed, and the results are shown in **Table 3**. Tau and Tw values taken from on-chip measurements by Salomon Beer and Ran Ginosar [58]. The system considered contained 64 cores, frequency values were set to 1GHz, and the number of stages for brute forces synchronizers was 4, and for predictive synchronizers 5 stages were used. 5 stages were simply chosen by adding an arbitrary additional stage, since predictive synchronizers need not limit their number of stages due to latency constraints. It can clearly be seen that the exponential impact of more stages allows the NoCs containing predictive synchronizers to have an MTBF many orders of magnitude higher than that of brute-force FIFO systems.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This thesis has covered significant material in the areas of NoCs and synchronization. Preparation in the form of synchronizer flip-flop and NoC simulation has provided a strong platform for the novel work. After thorough analysis it was concluded that predictive synchronization was necessary to bypass the latency-reliability tradeoff required by brute-force synchronizers. Since the Even Odd Predictive Synchronizer was incompatible with DVFS, novel predictive circuits were created and implemented in Verilog RTL to solve this problem. A novel CDI was also created which could take advantage of these new DVFS capable circuits as well. To see what performance and reliability impact could be achieved by switching to predictive synchronizers, Graphite simulations and reliability calculations were completed.

The results of this study show that it is well worth adding predictive synchronization to modern SoCs. Roughly 12% performance improvement can be achieved if using a Multi-Synchronous NoC, and other NoCs also show some small improvement. Regardless of what type of NoC is used, the reliability improvements are many orders of magnitude, making predictive synchronizers particularly helpful in large scale designs. With all of these benefits it is important to take predictive synchronizers seriously, even if they have a reputation for being too exotic. The next step to allowing predictive synchronizers to go main-stream is to address weaknesses associated with clock jitter and skew. With this last piece in place, predictive synchronization will be able to help processors become significantly faster and more reliable.

BIBLIOGRAPHY

[1] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Design Automation Conference, 2001. Proceedings,* 2001, pp. 684-689.

[2] T. C. Xu, P. Liljeberg and H. Tenhunen, "Explorations of optimal core and cache placements for chip multiprocessor," in *NORCHIP, 2011,* 2011, pp. 1-6.

[3] Hyungjun Kim, P. Ghoshal, B. Grot, P. V. Gratz and D. A. Jimenez, "Reducing network-on-chip energy consumption through spatial locality speculation," in *Networks on Chip (NoCS), 2011 Fifth IEEE/ACM International Symposium on,* 2011, pp. 233-240.

[4] C. Fensch, N. Barrow-Williams, R. D. Mullins and S. Moore, "Designing a Physical Locality Aware Coherence Protocol for Chip-Multiprocessors," *Computers, IEEE Transactions on,* vol. 62, pp. 914-928, 2013.

[5] M. M. K. Martin, M. D. Hill and D. J. Sorin, "Why On-Chip Cache Coherence Is Here to Stay," *Communications of the ACM,* 2012.

[6] Hui Zhao, Ohyoung Jang, Wei Ding, Yuanrui Zhang, M. Kandemir and M. J. Irwin, "A hybrid NoC design for cache coherence optimization for chip multiprocessors," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE,* 2012, pp. 834-842.

[7] Yuho Jin, Ki Hwan Yum and Eun Jung Kim, "Adaptive data compression for high-performance low-power on-chip networks," in *Microarchitecture, 2008. MICRO-41. 2008 41st IEEE/ACM International Symposium on,* 2008, pp. 354-363.

[8] Po-Tsang Huang, Wei-Li Fang, Yin-Ling Wang and Wei Hwang, "Low power and reliable interconnection with self-corrected green coding scheme for network-on-chip," in *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on,* 2008, pp. 77-83.

[9] A. Flores, J. L. Aragon and M. E. Acacio, "Heterogeneous Interconnects for Energy-Efficient Message Management in CMPs," *Computers, IEEE Transactions on,* vol. 59, pp. 16-28, 2010.

[10] T. Krishna, L. Peh, B. Beckman and S. Reinhardt, "Towards the ideal on-chip fabric for 1-to-many and many-to-1 communication," *MICRO,* 2011.

[11] S. Volos, C. Seiculescu, B. Grot, N. K. Pour, B. Falsafi and G. De Micheli, "CCNoC: Specializing on-chip interconnects for energy efficiency in cache-coherent servers," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on,* 2012, pp. 67-74.

[12] J. Kim, "Low-cost router microarchitecture for on-chip networks," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on,* 2009, pp. 255-266.

[13] G. Michelogiannakis, D. Sanchez, W. J. Dally and C. Kozyrakis, "Evaluating bufferless flow control for on-chip networks," in *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on,* 2010, pp. 9-16.

[14] A. B. Ahmed and A. B. Abdallah, "Low-overhead routing algorithm for 3D network-on-chip," in *Networking and Computing (ICNC), 2012 Third International Conference on,* 2012, pp. 23-32.

[15] A. K. Mishra, O. Mutlu and C. R. Das, "A heterogeneous multiple network-on-chip design: An application-aware approach," in *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE,* 2013, pp. 1-10.

[16] Jungju Oh, A. Zajic and M. Prvulovic, "Traffic steering between a low-latency unswitched TL ring and a high-throughput switched on-chip interconnect," in *Parallel Architectures and Compilation Techniques (PACT), 2013 22nd International Conference on,* 2013, pp. 309-318.

[17] E. Bolotin, Z. Guz, I. Cidon, R. Ginosar and A. Kolodny, "The power of priority: NoC based distributed cache coherency," in *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on,* 2007, pp. 117-126.

[18] Yi Liu, Gang Liu, Yintang Yang and Zijin Li, "A novel low-swing transceiver for interconnection between NoC routers," in *Digital Content, Multimedia Technology and its Applications (IDCTA), 2011 7th International Conference on,* 2011, pp. 39-44.

[19] A. Ejlali, B. M. Al-Hashimi, P. Rosinger, S. G. Miremadi and L. Benini, "Performability/Energy Tradeoff in Error-Control Schemes for On-Chip Networks," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 18, pp. 1-14, 2010.

[20] E. Mensink, D. Schinkel, E. A. M. Klumperink, E. Van Tuijl and B. Nauta, "Power Efficient Gigabit Communication Over Capacitively Driven RC-Limited On-Chip Interconnects," *Solid-State Circuits, IEEE Journal of,* vol. 45, pp. 447-457, 2010.

[21] J. Postman, T. Krishna, C. Edmonds, Li-Shiuan Peh and P. Chiang, "SWIFT: A Low-Power Network-On-Chip Implementing the Token Flow Control Router Architecture With Swing-Reduced Interconnects," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 21, pp. 1432-1446, 2013.

[22] T. C. Xu, P. Liljeberg and H. Tenhunen, "Optimal number and placement of through silicon vias in 3D network-on-chip," in *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2011 IEEE 14th International Symposium on,* 2011, pp. 105-110.

[23] V. S. Nandakumar and M. Marek-Sadowska, "Low power, high throughput network-on-chip fabric for 3D multicore processors," in *Computer Design (ICCD), 2011 IEEE 29th International Conference on,* 2011, pp. 453-454.

[24] A. -. Rahmani, K. R. Vaddina, K. Latif, P. Liljeberg, J. Plosila and H. Tenhunen, "Design and management of high-performance, reliable and thermal-aware 3D networks-on-chip," *Circuits, Devices & Systems, IET,* vol. 6, pp. 308-321, 2012.

[25] S. Assefa, S. Shank, W. Green, M. Khater, E. Kiewra, C. Reinholm, S. Kamlapurkar, A. Rylyakov, C. Schow, F. Horst, Huapu Pan, T. Topuria, P. Rice, D. M. Gill, J. Rosenberg, T. Barwicz, Min Yang, J. Proesel, J. Hofrichter, B. Offrein, Xiaoxiong Gu, W. Haensch, J. Ellis-Monaghan and Y. Vlasov, "A 90nm CMOS integrated nano-photonics technology for 25Gbps WDM optical communications applications," in *Electron Devices Meeting (IEDM), 2012 IEEE International,* 2012, pp. 33.8.1-33.8.3.

[26] G. Kurian, Chen Sun, C. -. O. Chen, J. E. Miller, J. Michel, Lan Wei, D. A. Antoniadis, Li-Shiuan Peh, L. Kimerling, V. Stojanovic and A. Agarwal, "Cross-layer energy and performance evaluation of a nanophotonic manycore processor system using real application workloads," in *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International,* 2012, pp. 1117-1130.

[27] R. Morris, E. Jolley and A. K. Kodi, "Extending the Performance and Energy-Efficiency of Shared Memory Multicores with Nanophotonic Technology," *Parallel and Distributed Systems, IEEE Transactions on,* vol. 25, pp. 83-92, 2014.

[28] Chao Chen and A. Joshi, "Runtime Management of Laser Power in Silicon-Photonic Multibus NoC Architecture," *Selected Topics in Quantum Electronics, IEEE Journal of,* vol. 19, pp. 3700713-3700713, 2013.

[29] C. A. D. Adi, H. Matsutani, M. Koibuchi, H. Irie, T. Miyoshi and T. Yoshinaga, "An efficient path setup for a photonic network-on-chip," in *Networking and Computing (ICNC), 2010 First International Conference on,* 2010, pp. 156-161.

[30] M. A. Abd El Ghany, M. A. Wanas and M. Zaki, "Hybrid mesh-ring wireless network on chip for multi-core system," in *SoC Design Conference (ISOCC), 2012 International,* 2012, pp. 167-170.

[31] A. Ganguly, K. Chang, S. Deb, P. P. Pande, B. Belzer and C. Teuscher, "Scalable Hybrid Wireless Network-on-Chip Architectures for Multicore Systems," *Computers, IEEE Transactions on,* vol. 60, pp. 1485-1502, 2011.

[32] D. DiTomaso, S. Laha, S. Kaya, D. Matolak and A. Kodi, "Energy efficient modulation for a wireless network-on-chip architecture," in *New Circuits and Systems Conference (NEWCAS), 2012 IEEE 10th International,* 2012, pp. 489-492.

[33] Ling Wang, Zhen Wang and Yingtao Jiang, "A hybrid chip interconnection architecture with a global wireless network overlaid on top of a wired network-on-chip," in *System on Chip (SoC), 2012 International Symposium on,* 2012, pp. 1-4.

[34] L. P. Carloni, P. Pande and Yuan Xie, "Networks-on-chip in emerging interconnect paradigms: Advantages and challenges," in *Networks-on-Chip, 2009. NoCS 2009. 3rd ACM/IEEE International Symposium on,* 2009, pp. 93-102.

[35] D. Gebhardt, Junbok You and K. S. Stevens, "Comparing energy and latency of asynchronous and synchronous NoCs for embedded SoCs," in *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on,* 2010, pp. 115-122.

[36] M. A. A. El Ghany and et al., "Power analysis for Asynchronous CLICHE Network-on-Chip," *SOCC,* 2010.

[37] M. Rashed, M. A. Abd El Ghany and M. Ismail, "Power characteristics of asynchronous networks-on-chip," in *SOC Conference (SOCC), 2011 IEEE International,* 2011, pp. 160-165.

[38] M. N. Horak, S. M. Nowick, M. Carlberg and U. Vishkin, "A Low-Overhead Asynchronous Interconnection Network for GALS Chip Multiprocessors," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on,* vol. 30, pp. 494-507, 2011.

[39] C. O. Chen, S. Park, T. Krishna, S. Subramanian, A. P. Chandrakasan and L. Peh, "SMART: A single-cycle reconfigurable NoC for SoC applications," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013,* 2013, pp. 338-343.

[40] Yoon Seok Yang, R. Kumar, Gwan Choi and P. Gratz, "WaveSync: A low-latency source synchronous bypass network-on-chip architecture," in *Computer Design (ICCD), 2012 IEEE 30th International Conference on,* 2012, pp. 241-248.

[41] Xi Chen, Zheng Xu, Hyungjun Kim, P. Gratz, Jiang Hu, M. Kishinevsky and U. Ogras, "In-network monitoring and control policy for DVFS of CMP networks-on-chip and last level caches," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on,* 2012, pp. 43-50.

[42] Liang Guang, E. Nigussie and H. Tenhunen, "Run-time communication bypassing for energy-efficient, low-latency per-core DVFS on network-on-chip," in *SOC Conference (SOCC), 2010 IEEE International,* 2010, pp. 481-486.

[43] M. Kumar Yadav, M. R. Casu and M. Zamboni, "A simple DVFS controller for a NoC switch," in *Ph.D. Research in Microelectronics and Electronics (PRIME), 2012 8th Conference on,* 2012, pp. 1-4.

[44] J. Balfour and W. J. Dally, "Design tradeoffs for tiled CMP on-chip networks," *Proc. of the 20th Int. Conference on Supercomputing,* 2006.

[45] Jia Zhao, S. Madduri, R. Vadlamani, W. Burleson and R. Tessier, "A Dedicated Monitoring Infrastructure for Multicore Processors," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. 19, pp. 1011-1022, 2011.

[46] D. Wentzlaff, P. Griffin, H. Hoffmann, Liewei Bao, B. Edwards, C. Ramey, M. Mattina, Chyi-Chang Miao, J. F. Brown and A. Agarwal, "On-Chip Interconnection Architecture of the Tile Processor," *Micro, IEEE,* vol. 27, pp. 15-31, 2007.

[47] R. Ginosar, "Metastability and Synchronizers: A Tutorial," *Design & Test of Computers, IEEE,* vol. 28, pp. 23-35, 2011.

[48] S. Beer, R. Ginosar, J. Cox, T. Chaney and D. M. Zar, "Metastability challenges for 65nm and beyond; simulation and measurements," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2013,* 2013, pp. 1297-1302.

[49] J. Zhou, M. Ashouei, D. Kinniment, J. Huisken, G. Russell and A. Yakovley, "Sub-threshold Synchronizer," *Microelectronics Journal,* 2011.

[50] C. Dike and E. Burton, "Miller and noise effects in a synchronizing flip-flop," *Solid-State Circuits, IEEE Journal of,* vol. 34, pp. 849-855, 1999.

[51] S. Beer, J. Cox, T. Chaney and D. M. Zar, "MTBF bounds for multistage synchronizers," in *Asynchronous Circuits and Systems (ASYNC), 2013 IEEE 19th International Symposium on,* 2013, pp. 158-165.

[52] D. Li, P. I. -. Chuang, D. Nairn and M. Sachdev, "Design and analysis of metastable-hardened flip-flops in sub-threshold region," in *Low Power Electronics and Design (ISLPED) 2011 International Symposium on,* 2011, pp. 157-162.

[53] J. U. Horstmann, H. W. Eichel and R. L. Coates, "Metastability behavior of CMOS ASIC flip-flops in theory and test," *Solid-State Circuits, IEEE Journal of,* vol. 24, pp. 146-157, 1989.

[54] L. Portmann and T. H. -. Meng, "Metastability in CMOS library elements in reduced supply and technology scaled applications," *Solid-State Circuits, IEEE Journal of,* vol. 30, pp. 39-46, 1995.

[55] Jun Zhou, D. J. Kinniment, C. E. Dike, G. Russell and A. V. Yakovlev, "On-Chip Measurement of Deep Metastability in Synchronizers," *Solid-State Circuits, IEEE Journal of,* vol. 43, pp. 550-557, 2008.

[56] Suwen Yang, I. W. Jones and M. R. Greenstreet, "Synchronizer performance in deep sub-micron technology," in *Asynchronous Circuits and Systems (ASYNC), 2011 17th IEEE International Symposium on,* 2011, pp. 33-42.

[57] D. Rennie, D. Li, M. Sachdev, B. Bhuva, S. Jagannathan, ShiJie Wen and R. Wong, "Performance, metastability and soft-error robustness tradeoffs for flip-flops in 40nm CMOS," in *Custom Integrated Circuits Conference (CICC), 2011 IEEE,* 2011, pp. 1-4.

[58] S. Beer, R. Ginosar, M. Priel, R. Dobkin and A. Kolodny, "An on-chip metastability measurement circuit to characterize synchronization behavior in 65nm," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on,* 2011, pp. 2593-2596.

[59] T. J. Chaney and F. U. Rosenberger, "Characterization and Scaling of MOS Flip Flop Performance in Synchronizer Applications," *Proc. of the Caltech Conf. on VLSI,* 1979.

[60] M. S. Baghini and M. P. Desai, "Impact of technology scaling on metastability performance of CMOS synchronizing latches," in *Design Automation Conference, 2002. Proceedings of ASP-DAC 2002. 7th Asia and South Pacific and the 15th International Conference on VLSI Design. Proceedings.* 2002, pp. 317-322.

[61] S. Beer, R. Ginosar, M. Priel, R. Dobkin and A. Kolodny, "The devolution of synchronizers," in *Asynchronous Circuits and Systems (ASYNC), 2010 IEEE Symposium on,* 2010, pp. 94-103.

[62] R. D. S. Beer and R. Ginosar, "Metastability Measurements of Several ASIC and FPGA Synchronizers," *Technical Report, EE Dept, Technion,* 2009.

[63] A. N. Bhoj and N. K. Jha, "Design of ultra-low-leakage logic gates and flip-flops in high-performance FinFET technology," in *Quality Electronic Design (ISQED), 2011 12th International Symposium on,* 2011, pp. 1-8.

[64] H. R. Khan, D. Mamaluy and D. Vasileska, "Simulation of the Impact of Process Variation on the Optimized 10-nm FinFET," *Electron Devices, IEEE Transactions on,* vol. 55, pp. 2134-2141, 2008.

[65] S. Sinha, G. Yeric, V. Chandra, B. Cline and Yu Cao, "Exploring sub-20nm FinFET design with predictive technology models," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE,* 2012, pp. 283-288.

[66] J. -. Chabloz and A. Hemani, "Low-Latency Maximal-Throughput Communication Interfaces for Rationally Related Clock Domains," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on,* vol. PP, pp. 1-1, 2013.

[67] Ru Wang, Huandong Wang, Baoxia Fan and Liang Yang, "RIRI scheme: A robust instant-responding ratiochronous interface with zero-latency penalty," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on,* 2011, pp. 2569-2572.

[68] U. Frank, T. Kapshitz and R. Ginosar, "A Predictive Synchronizer for Periodic Clock Domains," *J. Formal Methods in System Design,* 2006.

[69] W. J. Dally and S. G. Tell, "The Even/Odd synchronizer: A fast, all-digital, periodic synchronizer," in *Asynchronous Circuits and Systems (ASYNC), 2010 IEEE Symposium on,* 2010, pp. 75-84.

[70] J. E. Miller, H. Kasture, G. Kurian, C. Gruenwald, N. Beckmann, C. Celio, J. Eastep and A. Agarwal, "Graphite: A distributed parallel simulator for multicores," in *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on,* 2010, pp. 1-12.

[71] A. B. Sachid and Chenming Hu, "A little known benefit of FinFET over planar MOSFET in highperformance circuits at advanced technology nodes," in *SOI Conference (SOI), 2012 IEEE International,* 2012, pp. 1-2.

[72] Jun Zhou, D. Kinniment, G. Russell and A. Yakovlev, "Adapting synchronizers to the effects of on chip variability," in *Asynchronous Circuits and Systems, 2008. ASYNC '08. 14th IEEE International Symposium on,* 2008, pp. 39-47.

[73] S. Soleimani, A. Afzali-Kusha and B. Forouzandeh, "Temperature dependence of propagation delay characteristic in FinFET circuits," in *Microelectronics, 2008. ICM 2008. International Conference on,* 2008, pp. 276-279.