2-27-2017

Quadded GasP: A Fault Tolerant Asynchronous Design

Kristopher S. Scheiblauer
Portland State University

Quadded GasP: A Fault Tolerant Asynchronous Design

by

Kristopher S. Scheiblauer

A thesis submitted in partial fulfillment of the
requirements for the degree of

Master of Science
in
Electrical and Computer Engineering

Thesis Committee:
W. Robert Daasch, Chair
Ivan Sutherland
Glenn Shirley

Portland State University
2017

Abstract

As device scaling continues, process variability and defect densities are becoming increasingly challenging for circuit designers to contend with. Variability reduces timing margins, making it difficult and time consuming to meet design specifications. Defects can cause degraded performance or incorrect operation resulting in circuit failure. Consequently test times are lengthened and production yields are reduced.

This work assess the combination of two concepts, self-timed asynchronous design and fault tolerance, as a possible solution to both variability and defects. Asynchronous design is not as sensitive to variability as synchronous, while fault tolerance allows continued functional operation in the presence of defects.

GasP is a self-timed asynchronous design that provides high performance in a simple circuit. Quadded Logic, is a gate level fault tolerant methodology. This study presents Quadded GasP, a fault tolerant asynchronous design.

This work demonstrates that Quadded GasP circuits continue to function within performance expectations when faults are present. The increased area and reduced performance costs of Quadded GasP area also evaluated.

These results show Quadded GasP circuits are a viable option for managing process variation and defects. Application of these circuits will provide decreased development and test times, as well as increased yield.

Acknowledgements

I would like to express my sincere gratitude to my thesis advisor Dr. Robert Daasch. He inspired me to begin this work when I had no plans to do research and always welcomed me back after life swept me away for stretches of time.

I would also like to thank Dr. Ivan Sutherland and Dr. Glenn Shirley for their support and guidance, as well as for serving on my thesis committee.

Thanks to everyone in ICDT who lent their support, especially Chris who readily provided much time and assistance in getting me started and answering my questions.

Finally, I also must thank my wife, without whom this endeavor would not have been possible. Her light and spirit is the driving force of my life, and this project was no different. She always had faith I would finish what I started and was as commited as I was to seeing it through.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As devices scale challenges in design and manufacturing arise. Two of these challenges are greater defect densities and increased variability in process, voltage and temperature. It has become critical to find effective solutions to these issues in order to keep design and test times under control. While defects and variability negatively impact both data and clock paths, the focus of this study is on the clock.

Defects on a global clock tree can cause widespread failure and are difficult to diagnose. There has been much research into fault tolerance methodologies for managing defects and providing reliability. With defect densities increasing, while the cost of transistors decreases, these techniques have become more practical.

Variability is a significant issue for synchronous design, which has been the industry standard for many years. Variability causes the worst case delays that synchronous design is based on to become more extreme, and so synchronizing a global clock tree becomes more complex and time consuming. Asynchronous design, which is based on average path delays, is much more tolerant of variability. Self-timed circuit are an application of asynchronous design and will be discussed in this study. Self-timed circuits are effected by local variability, as opposed to the global variability a synchronous clock tree must contend with.

It follows that a combination of these two solutions, fault tolerant asynchronous circuits, would provide designers with a good option to deal with both issues. This research examines the implementation of asynchronous design using fault tolerant methodology and evaluates the resulting impact on performance. A self-timed GasP design is combined with the fault tolerant methodology of Quadded Logic, resulting in a novel circuit called Quadded GasP.

1.1 Objective

There are three primary objectives of this study. The first is to quantify the area and performance impact of converting to Quadded GasP. The addition of fault tolerance comes with a cost in increased area and reduced performance which must be understood in order to consider it a viable option.

The second goal is to verify that Quadded GasP circuits continue to operate correctly in the presence of faults. Fault tolerance is the primary benefit of implementing Quadded Logic. If for any reason Quadded Logic cannot be successfully implemented to correct faults then it is clearly not a practical solution.

Finally, the third goal is to assess the performance impact of faults present in quadded circuits and determine if these circuits perform within the expected operating range of functional circuits. Functional operation in the presence of faults is of little value if the circuit can no longer meet timing specifications.

1.2 Motivation

Greater variability in synchronous designs causes increased margins on clock and data paths, as they are based on worst-case timing. Variability can also make synchronizing a global clock across corners difficult. As a result, meeting timing specifications becomes more challenging and time consuming. However the issue of accounting for variability in the global clock, does not apply to asynchronous design. The data paths are also less affected by variability, as they are based on average path delays, rather than worst case delays. This means less optimization cycles and faster time to market.

One of the barriers preventing wider usage of asynchronous design is testing. Synchronous testing methodologies such as Scan are well understood and widely used. Asynchronous circuits are difficult to control and lack predictable responses. This makes testing problematic and slows the adoption of asynchronous design [1]. By integrating fault tolerance with asynchronous design, the need for testability is reduced, making asynchronous design a more appealing option to circuit designers.

More features get added to chips, components that used to be off die are now integrated. The task of testing gets harder. Incorporating fault tolerance into the chip reduces the amount of circuits that require comprehensive testing, providing reduced test planning and test time.

The effects of defects can be difficult to understand and debug, especially in newer processes. Critical circuits that cause wide-spread issues when they fail can be

particularly difficult to trace. Incorporating fault tolerance, especially in these critical circuits, can afford reduced debug times.

A device that incorporates fault tolerance will benefit from a correction of normally debilitating defects, resulting in increased production yields.

1.3 Contribution

This study analyzes the performance impact of quadding GasP circuits. Contributions of this research are:

- Presenting designs for two configurations of Quadded GasP modules.
- Evaluating the effect on area and performance of applying Quadded Logic to GasP.
- Verifying that Quadded Gasp circuits continue to function in the presence of faults.
- Assessing any impact on performance when correctable faults are introduced into Quadded GasP circuitry.
- Determining whether Quadded GasP performance remains viable when correctable faults are present.

1.4 Organization

This document is organized into five chapters. The second chapter discusses the background of fault tolerance, focusing on Quadded Logic, and self-timed GasP circuits.

Chapter 3 details the methodologies used in this study, including descriptions of the circuit test cases, performance metrics, and fault models. The fourth chapter presents and analyzes the results of the study. Finally, the fifth chapter states the conclusions of the study.

Chapter 2

Background

2.1 Fault Tolerance

Fault tolerance is the concept of receiving reliable operation from a system composed of unreliable components, a topic that was originally pioneered by von Neumann in his Caltech lectures in 1952 [2]. Von Neumann demonstrated that inserting redundant components into existing circuitry could increase the reliable operation of the circuit. Fault tolerance has been the focus of considerable research ever since. Several methodologies for achieving fault tolerance have emerged, such as Triple Modular Redundancy and Quadded Logic [3].

2.1.1 Triple Modular Redundancy

Triple Modular Redundancy (TMR) is a method for applying fault tolerance at the module level. As illustrated in Figure 2.1, TMR is implemented by replicating a module and its inputs three times, then selecting the output using a majority voter. If one of the modules is faulty and produces an incorrect output it will be outvoted by the two remaining fault-free modules, and the correct output will be propagated.

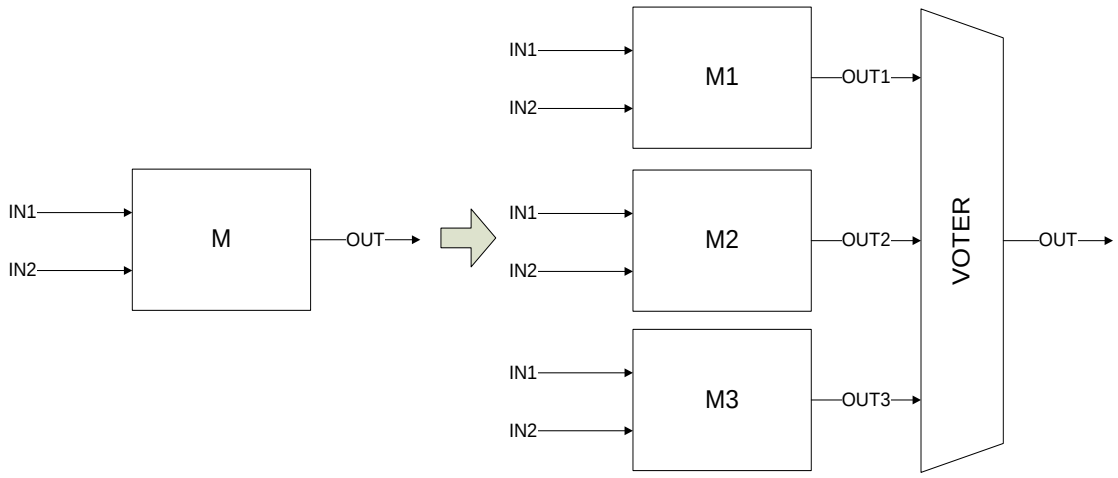

Figure 2.1: Conversion of module M to TMR

An advantage of TMR is that it is relatively simple to implement. It does not require any special cells, complex wiring, or additional circuitry aside from the voter block. However TMR is susceptible to failure if multiple faults are present. The interaction of multiple faults could cause unpredictable behavior, possibly resulting in the voter block propagating an incorrect value. A fault in the voter block itself could also result in erroneous output.

2.2 Quadded Logic

Quadded Logic is a gate-level fault tolerance strategy first proposed by J.G. Tryon in 1962 [4]. Quadded Logic is resilient against any single error and most multiple errors. Rather than replicating entire modules, as in TMR, the gates are replicated. This allows for a finer granularity of fault tolerance. Unlike other fault tolerance strategies, Quadded Logic has no error detection or voting structures associated with it. Rather,

any errors present are masked by construction. The interconnect strategy between quadded gates is the true source of reliability.

Quadded Logic was originally proposed as a solution to the failure of components that could not be easily repaired or replaced. However, the same concept can be applied today to fix defects in ICs. In the future, as CMOS shrinkage becomes limited by atomic sizing, the industry could move towards unreliable transistors: Carbon nano tubes, Silicon nano-wires, Quantum dot cells. Quadded Logic could be used to mitigate the poor reliability of these types of transistors.

When implementing Quadded Logic, each two-input gate in the module is expanded into a stage of four four-input gates, as shown in Figure 2.2. The inputs of each four-input gate in the stage are two copies of the original two inputs. The output of the stage are four signals that are logically equivalent to—but physically distinct from—the original output of the gate.

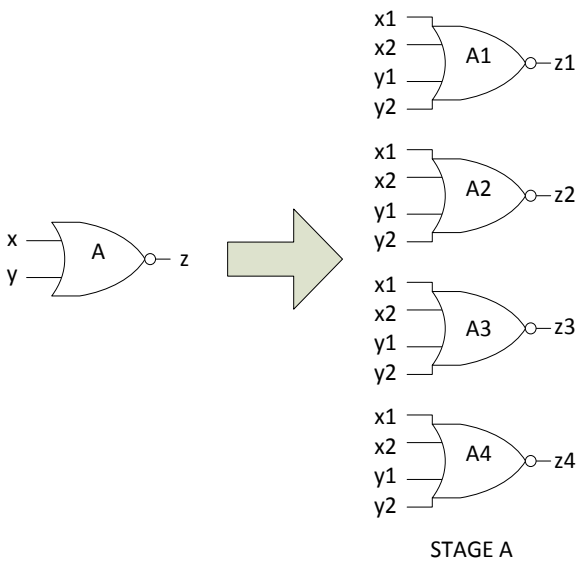

Figure 2.2: Conversion of gate A to a quadded stage

Error correction relies on the interconnect pattern between the stages and the controlling properties of digital logic gates. Each output of a stage of quadded gates is split into two pairs of two lines. One of the pairs is a straight connection to its corresponding gate in the subsequent stage. For example, in Figure 2.3, the straight connection of the first gate in Stage B connects to the first gate in the Stage C. The other signal in the output pair is a cutover connection, which connects to one of the other gates in the following stage. The output signal pairing is denoted as [outputA,outputB][outputC,outputD], where outputA and outputB are paired, and outputC and outputD are paired. There are three possible configurations: [1,2][3,4]; [1,3][2,4]; [1,4][2,3]. The pairing strategy depends on what the interconnect pattern coming into the initial stage is. Figure 2.3 illustrates a [1,2][3,4] connection from Stage A to Stage C, and a [1,3][2,4] connection from Stage B to Stage C.

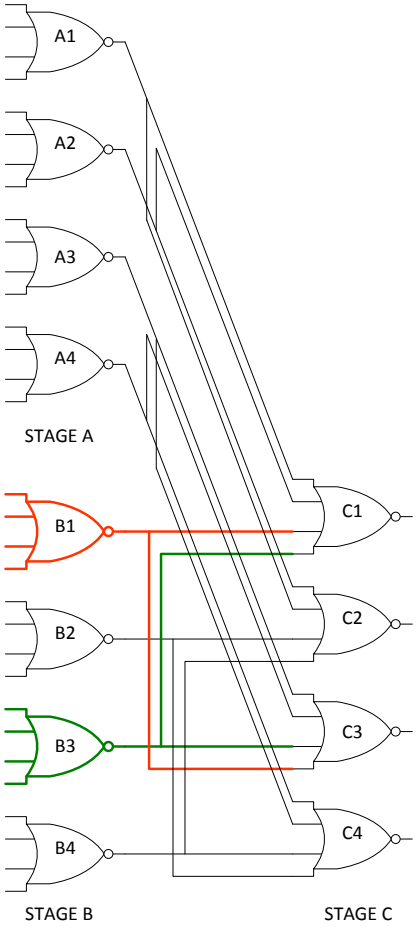

Figure 2.3: A connection pattern of [1,2][3,4] between quadded stages B and C

The error masking capability of Quadded Logic comes from leveraging the controlling versus non-controlling inputs of digital logic gate. A 1 input is controlling for a NOR gate, because the output will be 0 no matter what the other input is. A 0 input is non-controlling for a NOR gate, because the output will be determined by the other input. Quadded Logic insures erroneous signals always reach non-controlling inputs so that they will be masked.

Input 1	Input 2	Output
0	0	1
0	1	0
1	0	0
1	1	1

Table 2.1: Two-input NOR gate truth table

To demonstrate the error correcting capabilities, take two back-to-back inverters as an example. An inverter is equivalent to a two-input NOR gate with both inputs shorted together, which can be quadded as a stage of four two-input nor gates.

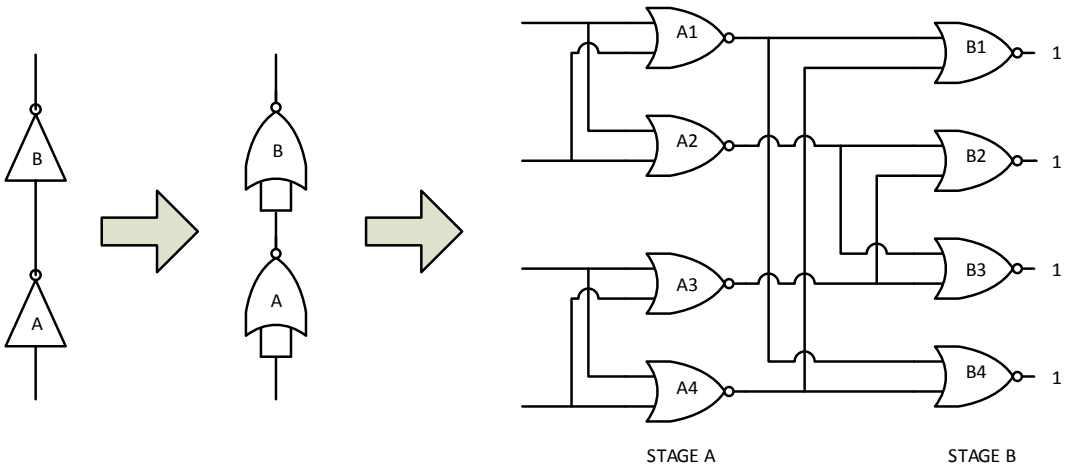

Figure 2.4: Quadding back-to-back inverters

For two stages of NOR gates an incorrect 1 will propagate through the first stage, as a 1 is controlling input for a NOR. However it will be output of the stage as an incorrect 0, which is a non-controlling input of the next stage of NORs. In this way it will be masked by the correct 1 coming from another gate in the preceding stage, as shown in figure 2.5. Alternatively, figure 2.8 illustrates how an incorrect 0 will be immediately masked by the first stage of NOR gates.

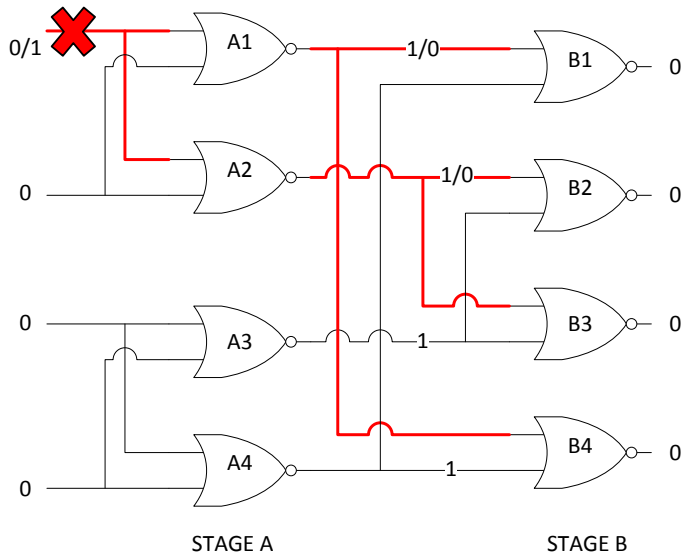

Figure 2.5: An example of SA1 error correction within two stages

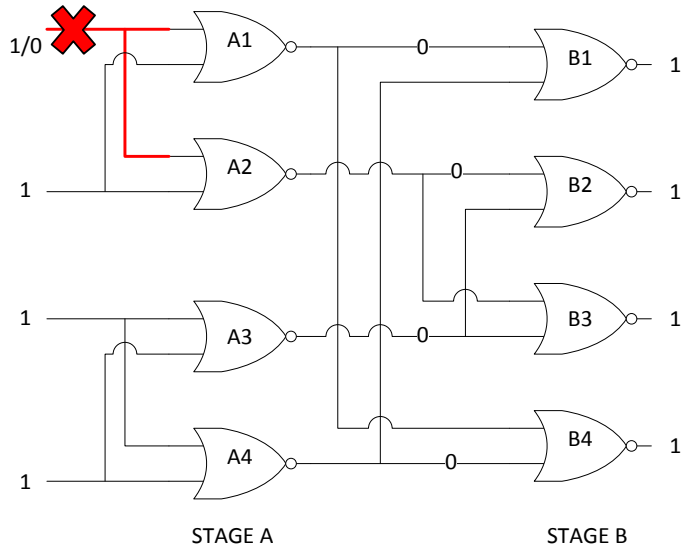

Figure 2.6: SA0 error correction within one Quadded NOR stage

This masking can only occur if it is ensured that any possible erroneous signals are paired with a signal that is correct. This is made possible by the interconnection pattern. The interconnect pattern for the output of the stage must be different than

any of the input patterns into the stage. For example, if the input interconnect pattern is [1,2][3,4] the output pattern must be either [1,3][2,4] or [1,4][2,3] to prevent error propagation. If the pattern coming into the stage is [1,2][3,4], as shown in in figure 2.7, then an incorrect 1 on the input to A1 will propagate as an incorrect 0 on the outputs of A1 and A2. If the following interconnect pattern were also [1,2][3,4] both inputs of B1 and B2 would receive the incorrect outputs from A1 and A2. Thus the error will propagate through the second stage.

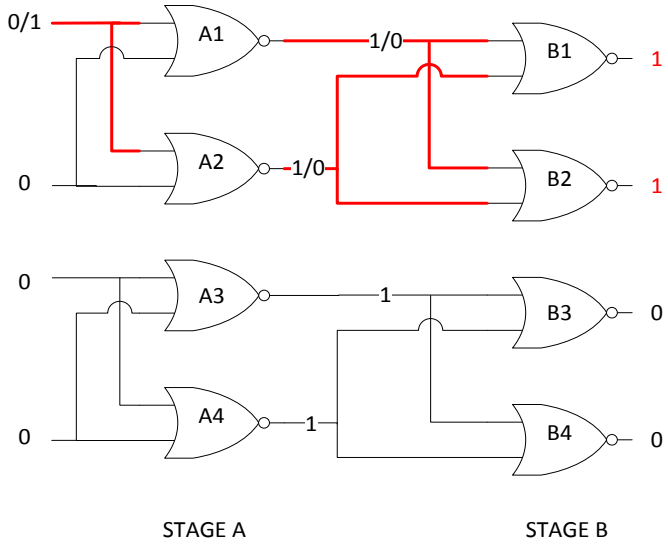

Figure 2.7: Uncorrectable fault propagation due to repeated interconnect patterning

On the other hand, if the Stage B had an input interconnect pattern of [1,3][2,4] or [1,4][2,3], each erroneous signal would have been paired with a controlling correct signal and the error would have been masked. Figure 2.7 is an example of this.

Quadded Logic is also resistant to multiple errors. It has already been shown that Quadded Logic will mask any single error in at most two stages. Therefor any multiple

errors that occur in a Quadded Logic circuit will also be masked if they are two or more stages apart. Even if two errors occur in the same stage, certain combinations will still be masked. For example a SA0 and a SA1 in the same stage will both be masked, as illustrated in figure 2.8.

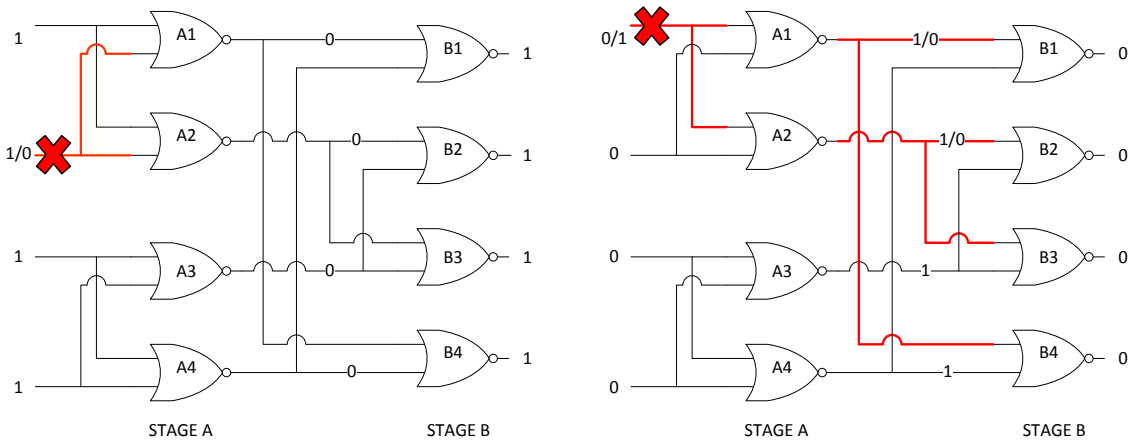

Figure 2.8: Multiple fault correction, SA0 an SA1 in the same circuit

There are two types of errors that Quadded Logic is vulnerable to. Peripheral errors on the inputs and outputs of the quadded circuit, and certain multiple errors that occur within two stages of each other.

Quadded Logic, like any fault tolerance methodology, has critical transition points into and out of the module that are susceptible to faults. The input to a quadded circuit is a single signal that is duplicated four times and fed into the first stage. Any error occurring on these inputs will propagate to all four gates in the stage and will be impossible to mask. Likewise the output of the circuit merges the four quadded signals

of the final stage into a single signal. If there is an error present it will propagate to the output of the circuit.

The error masking ability of Quadded Logic relies on any erroneous signals being paired with a correct controlling signal at the input of a gate. If multiple errors occur in close proximity it can result in the pairing of two erroneous signals, which will prevent either of them from being masked.

Quadded Logic has been around for decades, yet only recently has it again become a topic of research. For much of the history of integrated circuits the cost of area was so high that the addition of redundant transistors was impractical. However in the present the cost of transistors has reduced to the point that it is not uncommon for large portions of chips to be dedicated to non-functional applications as such as DFT and BIST. Additionally wire stacks have grown to the point where converting a single fanout wire to four double fanout wires can be accommodated.

2.3 Self-Timed GasP

GasP is a minimal but fast self-timed pipeline control circuit [5][6]. It has a simple structure of five inverters, a single two-input combinational cell and two pull-up/pull-down transistors, as shown in Figure 2.9. It has two bidirectional state wires which communicate with neighboring GasP modules and one output that controls the pipeline. The Predecessor state wire talks to the preceding GasP stage. It indicates when there is data ready to be transmitted to the next stage of the pipeline. This is the FULL state. The Successor state wire talks to the succeeding GasP stage. It specifies when the next

stage of the pipeline is ready to accept new data. This is the EMPTY state. The Fire output tells the pipeline latches when to accept data from the preceding stage. This occurs when the Predecessor state is FULL and the Successor state is EMPTY. A token is a FULL state that is propagated through a chain of GasP modules. The occupancy is the number of tokens in the chain.

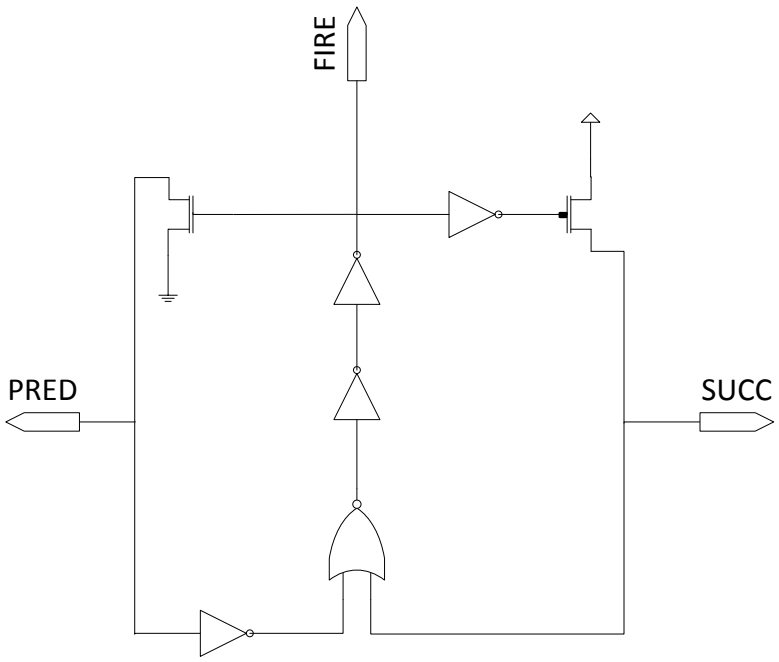

Figure 2.9: Active High GasP circuit

In addition to driving the pipeline latches, the Fire signal performs three operations. It forces the Predecessor state wire to EMPTY, which allows it to accept new data. It forces the Successor state wire to FULL, which allows the data transmitted to keep moving through the pipeline and prevents it from being overwritten until it has done so. It also performs a self-reset, so that it will be ready for the next transmission as soon as the data is ready and the pipeline is clear. Speed of GasP is measured by throughput,

which is the rate at which the GasP module can pass data through the pipeline. Figure 2.12 shows the waveform of a GasP Fire signal. The signal goes high briefly before the self-reset circuitry brings it back low. The throughput is calculated by summing the number of times the Fire signal goes high per unit of time.

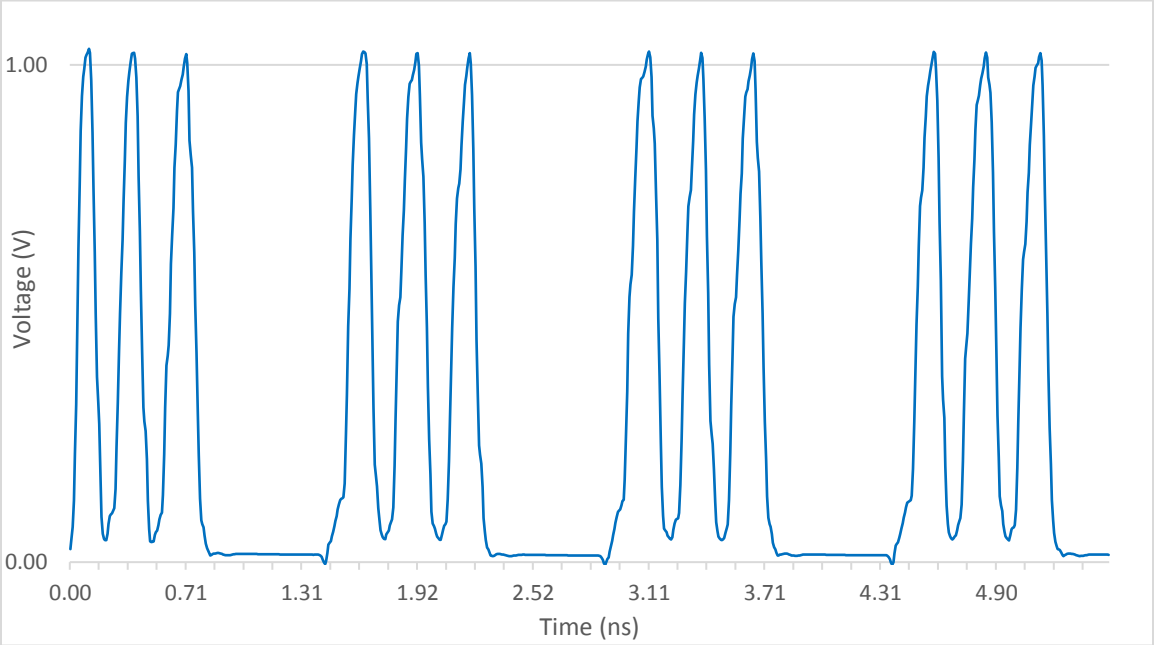

Figure 2.10: Fire signal waveform

The FULL state can be defined as either 0 or 1, and the EMPTY state as either 1 or 0, respectively. A GasP module that uses 0 as the FULL state is called Active Low. It uses a NAND as the combinational gate. A module that uses 1 as the FULL state is called Active High. It uses a NOR as the combinational gate. The GasP module in Figure 2.9 is Active High.

2.4 Performance

The performance of the GasP circuit is measured by its throughput. Throughput is defined as the rate at which data passes through the pipeline. For example, the waveform in Figure 2.10 shows that three tokens are passes approximately every 1.5ns, therefor the throughput of this circuit is 2 tokens/ns. The hallmark of the GasP architecture is its high throughput performance with little cost in area. Implementing GasP with Quadded Logic will decrease the throughput and increase the area, due to replacing inverters and two-input combinational gates with larger and slower two- and four-input combinational gates.

Chapter 3

Strategy

3.1 Technology

TSMC 90nm library was used.

3.2 Quadded Circuit Design

This section will outline Quadded GasP, which are GasP circuits implemented with Quadded Logic. The name Standard GasP is given to the original GasP circuits described in Chapter 2 to avoid confusion.

The GasP module consists of three types of components: inverters, pull-up/pull-down transistors, and a two-input logic gate (NAND or NOR). The translation of the two-input gate is simple, it is transformed into a four-input gate and duplicated four times. The inverters can be translated two ways, either as a stage of two-input NAND gates or NOR gates. As combining different types of gates in a Quadded Logic circuit results in a reduction of error masking ability [7], the type of gate to use for the quadded inverter was determined by the two-input gate. If the two-input gate was a NOR, the inverters were converted with NOR gates. If the two-input gate was a NAND, NAND gates were used.

The pull-up and pull-down transistors were a special case. They only have a single input. It was determined to be sufficient to simply duplicate the transistor four times, placing one on each output of the quadded stage. Any error on these transistors would behave similarly as an error on the output of the preceding stage, and thus be masked similarly.

As discussed in Chapter 2, the primary rule for implementing Quadded Logic with NAND and NOR gates is that the routing pattern of the output of a given stage must be different than the patterns of its inputs. Therefore special care was given to meet this requirement when implementing Quadded GasP. It was especially important to ensure that the state wires, which connected to multiple GasP modules, were properly connected. Figure 3.1 shows that the routing patterns used to connect the quadded stages abide by the interconnection rule.

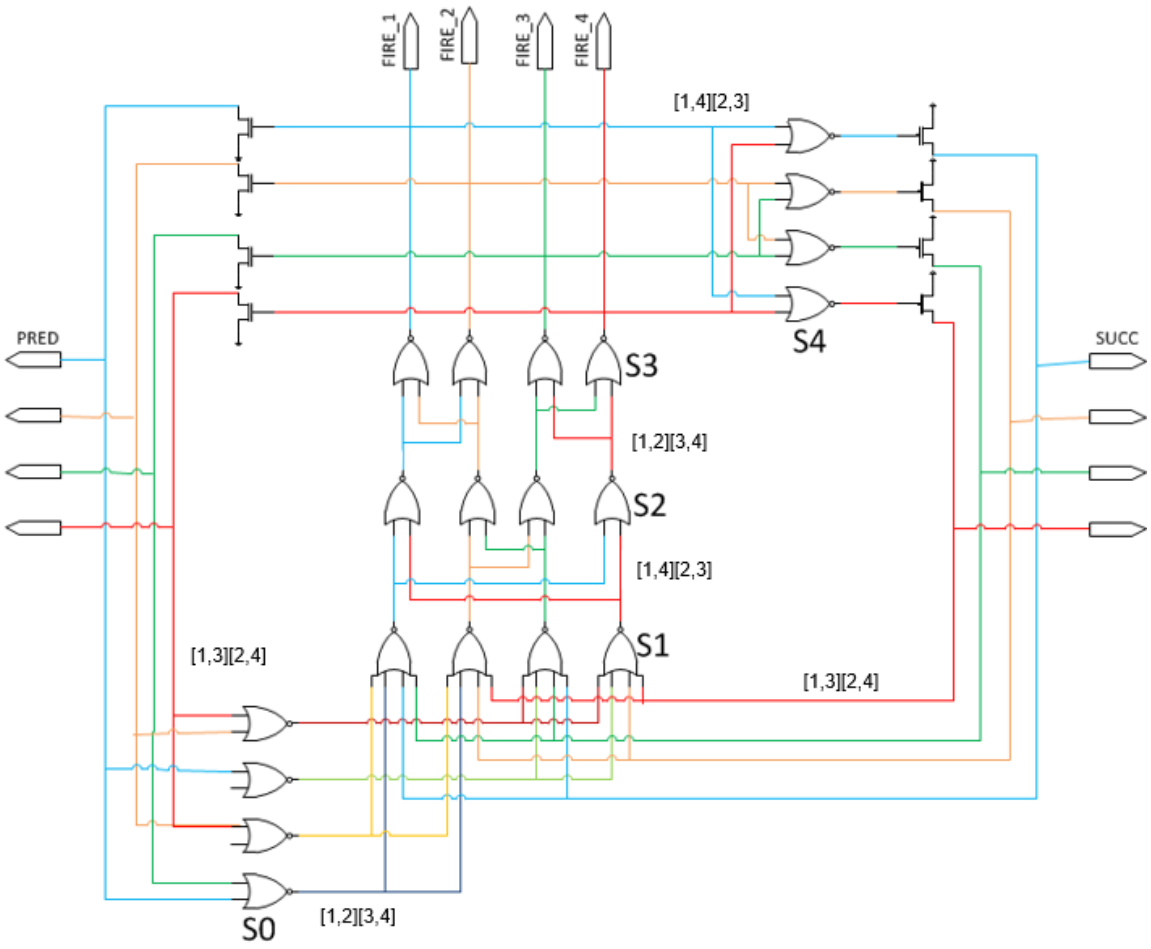

Figure 3.1: Quadded GasP schematic

3.3 Active High & Active Low

GasP circuits can be built to produce either Active-High or Active-Low output signals. Active-High circuits consider a high signal on the state wires to be Full, and a high signal on the Fire ports will trigger the latches and propagate the data path. While Active-Low circuits use low as Full and Fire. The only difference in the structure of the modules is that Active-High uses a two-input NOR gate while Active-Low uses a NAND gate. This gives four GasP implementations to compare: Standard Active-High (SNOR), Standard Active-Low (SNAN), Quadded Active-High (QNOR), and Quadded Active-Low (QNAN). Figure 3.1 illustrates QNOR. The SNAN module is expected to have a higher throughput than the SNOR version, due to the increase in speed of the NAND gate over the NOR. However the difference between quadded versions is expected to be more extreme, as every gate in the module is switched from NOR to NAND. Thus the QNAN is expected to see a larger speed advantage over QNOR, resulting in a smaller gap in performance when compared to the standard circuit.

As a result of quadding, the QNOR and QNAN modules each have four Fire signals driving the latches. This allows the load of the latches to be distributed across the four Fire signals. Each of the four Fire signals in the quadded modules drive ¼ the load of the single Fire signal in the standard modules.

3.4 Load

The speed of a circuit depends in part on the load that it's driving. As such, a load must be defined if the speeds of the two GasP configurations are to be fairly compared. Both

Standard and Quadded GasP drive the same load on their Fire signals, an array of pipeline latches. However, while the Standard GasP drives the entire array with a single Fire signal, Quadded GasP has four Fire signals with which to distribute the load as shown in Figure 3.1. In effect, the Quadded GasP drives one fourth the load of the Standard. This reduced load allows the Quadded GasP circuit to recoup some of its speed disadvantage versus the Standard GasP.

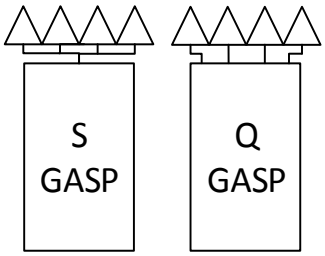

Figure 3.2: Load connections of Standard and Quadded GasP modules

An array of 64 inverters was used during simulation to model pipeline latches. Inverters were chosen for ease of implementation and simulation. 64 were used to model a typical 64-bit data bus. The Fire signal of each Standard GasP module was connected to 64 inverters, while each of the four Fire signals of the Quadded GasP modules were connected to 16 inverters.

3.5 Transistor Stack Connection

The rise and fall times of a combinational gate are, in part, determined by which transistors in the gate are activated by incoming signals. The further away the transistor is from the output node, the longer an input signal can take to propagate through a gate. This is of primary concern to the Quadded GasP circuit, as it has a stage of four-

input gates as well as several stages of two-input gates. Without consideration for the way the stages are connected skewing of the signals could occur. For example if a stage of NAND gates has two high inputs but one of the inputs switches to low. The gates with the switching input connected to the transistor closer to the output will switch its output signal faster. Skewing of the signals within a GasP module results in its Fire signals being launched out of sync.

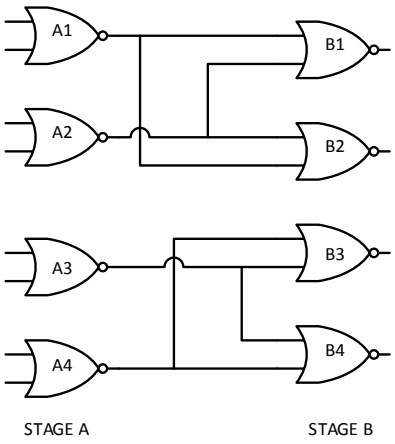

Figure 3.3: Pin connections of a stage of two-input gates

Preventing skewing for a stage of two-input gates is a simple matter of managing which pins are connected to which input signals. Since each output of a stage of gates fans out to two input pins of the following stage, the two signals are connected to opposite pins of their respective cells as shown in Figure 3.3. In this way each signal will have a fast transition and a slow transition, and they will balance themselves out as they propagate through the stages.

The stage of four-input gates requires more special care. Since each signal only reaches two of the gates, there was no way to connect a standard four-input gate in such a way that each signal can be balanced, as was done with the two-input gates. A special four-input gate was needed to perform the balancing of the incoming signals. While the standard four-input gate stacks four transistors on top of each other, the custom gate splits the stack into an array of four sub-stacks, with each transistor width ¼ of original width. This new four-input gate, while logically equivalent, allows the four inputs to be evenly distributed across the array of transistors, so no one signal has a speed advantage. The trend in standard cell design shifts toward the use of multiple unit width transistors rather than wide transistors. Therefore a four-input gate built in this structure would likely require only careful connections of the inputs. Figure 3.4 compares a traditional four-input NOR gate with the special gate used in the Quadded GasP circuit.

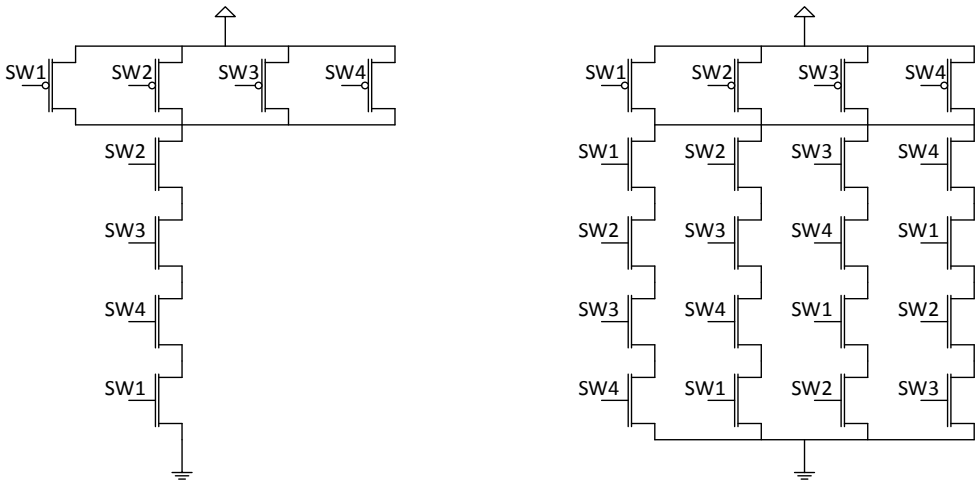

Figure 3.4: Conversion of a traditional four-input NOR to balanced NOR used in Quadded GasP

3.6 Transistor Sizing

Another factor in the speed of a circuit is the sizing of the transistors. In order to fairly compare GasP circuits the transistors in the gates must be appropriately sized to drive a specified load. Logical effort calculations were used to determine appropriate gate sizings for each circuit to achieve maximum throughput while driving the loads defined above.

3.7 Ring of 10

The circuit used to test the different implementations consisted of ten GasP modules stitched together in a ring, as shown in Figure 3.5

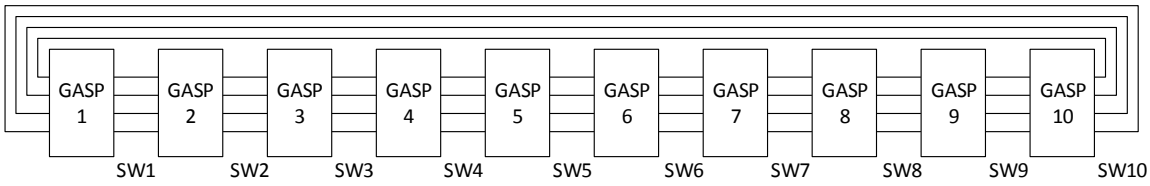

Figure 3.5: Ring of ten GasP modules

The Successor port(s) of the tenth module are connected to the Predecessor port(s) of the first. Each of the ten state wires in the circuit are seeded with either a FULL or EMPTY initial condition. Each FULL state is called a token, and the total number of tokens in the circuit is the occupancy. The throughput of the GasP module is determined by measuring how quickly the tokens propagate around the ring. The circuit is seeded with between one and nine tokens and the throughput is calculated for each occupancy. Zero occupancy is not simulated because there are no tokens to propagate through the circuit. Ten occupancy is not simulated because every stage is FULL and

there is no room to propagate the tokens. Throughput was measured by selecting one GasP module and recording the number of Fire signals output over a specified amount of time.

The ring structure was chosen for its simplicity. It only requires initial conditions to be set on the inter-module state wires—seeding the tokens—and then it continuously runs on its own, which makes it straightforward to measure throughput.

Several modules were necessary in the ring due to the behavior of flattening of the throughput. As tokens are added to the ring, the throughput increases linearly until the ring is half full, at which point the throughput decreases linearly. However the throughput at the mid-point is not a true intersection of these two linear functions, as would be expected. In reality the throughput of a half-full ring is slightly less than this intersection point. Ten GasP modules was determined to be sufficient for observing trends in performance behavior, while keeping simulation runtimes reasonable.

3.8 Canopy Diagram

A canopy diagram graphs the throughput of the GasP ring for each occupancy. As shown in Figure 3.6, the canopy diagram has three general parts. Between occupancy zero through three, referred to as 'low occupancy', the throughput increases linearly. Between occupancy seven through ten, referred to as 'high occupancy' the throughput decreases linearly. Between occupancy four and six these, referred to as 'mid occupancy', the low and high occupancy lines meet. However the slope reduces due to the flattening effect. A ring of ten modules does not provide enough data points to

accurately describe this effect. For the purposes of this study, which is primarily concerned with comparing the effects of quadding and faults on overall performance, analysis is generally focused on the low and high occupancies.

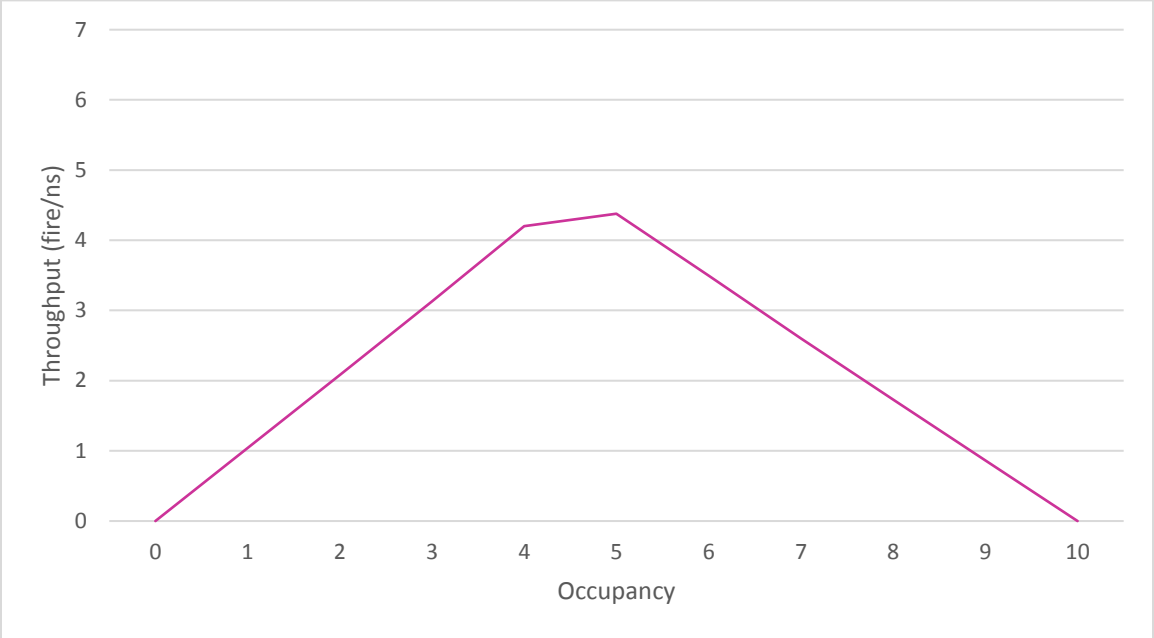

Figure 3.6: Canopy diagram for a ring of ten GasP implementation

3.9 Fault Injection

The benefit of using Quadded GasP over Standard is fault tolerance, however the cost of additional area and throughput reduction would be wasted if the circuit cannot operate. The Quadded GasP circuit was simulated with faults injected to verify it could continue to operate. Throughput measurements were also taken to measure the impact on performance. Several simulations were run using different scenarios of fault types, locations, and numbers.

3.9.1 Type of Faults

Three types of faults were modeled: Two Single Stuck-Ats, which models a node tied to either low (SA0) or high (SA1); and Delay, which models a slow-to-rise or slow-to-fall signal. SA0 faults were modeled by disconnecting a net from its driver and connecting it instead to gnd. SA1 faults were modeled in the same way, but by connecting to vdd. Delay faults were modeled capacitively, by adding a large load to the node, slowing down both the rise- and fall-times to approximately double the fault-free value. An array of inverters was used to model the load.

A stuck-at fault could be considered an extreme example of a delay fault, and as such it could be considered unnecessary to model delay faults. If a fault tolerant circuit operates in the presence of a stuck-at it should also be able to operate when a delay fault is present. However both fault models were used for three reasons. First, to verify that this assertion is indeed correct and Quadded GasP circuits can work in the presence of delay faults. Second, to compare the impact on performance of delay faults versus stuck-at faults in Quadded GasP circuits. Third, to compare the impact on performance of delay faults in Standard GasP circuits versus Quadded GasP circuits.

The circuits were also simulated with a mixture SA0 and SA1 faults, however it was found that the performance impact fell between the extreme cases of only SA0 and only SA1 faults present. As such, only the cases of homogenous faults were analyzed.

3.9.2 Location of Faults

The faults were injected on the inter-module state wires. The state wires were chosen because a single fault in these locations could impact two GasP modules, and due to fault equivalency they also covered many faults on wires internal to the modules. These wires are also generally much longer than internal wires, since they run between modules which can be placed far apart, so they are more susceptible to defects.

To determine the effect of fault location, several simulations were run with a single fault present in the circuit on each bit of a state wire bus (SW1[1], SW1[2],SW1[3],SW1[4]), and with a single fault present on different state wires (SW1[1], SW2[1], SW3[1], etc.). It was determined that the location had no impact, the circuit behaved in the same way no matter where the fault was injected. This was the expected result due to the symmetrical nature of the circuit.

When a second fault was injected, several simulations were run to determine if the proximity of the two faults had any effect on performance. The circuit was configured with one fault present in a fixed location (SW1[1]) and in a second fault present in a location that would change with each simulation (SW3[1], SW3[2], SW3[3], SW3[4], SW5[1], etc.) It was determined that the performance of the circuit was the same regardless of the fault locations.

3.9.3 Number of Faults

The simulations were run with four configurations of total faults in the circuit: no faults present, one fault present, two faults present, five faults present. The fault free

configuration allows a direct comparison between Standard and Quadded GasP. The single fault configuration allows a comparison between fault free and faulty Quadded GasP circuits. The two and five fault configurations were chosen to determine in what manner multiple faults interacted with each other, whether performance degraded linearly, exponentially, etc.

The five locations chosen for fault injection were SW1[1], SW3[3], SW5[4], SW7[2] and SW9[4], as shown in Figure 3.7. These locations were chosen so that each bit on the four-bit state wire bus was represented. The faults were not injected on successive state wires because this combination could lead to two faulty inputs reaching the four-input gate stage, which Quadded Logic is incapable of masking.

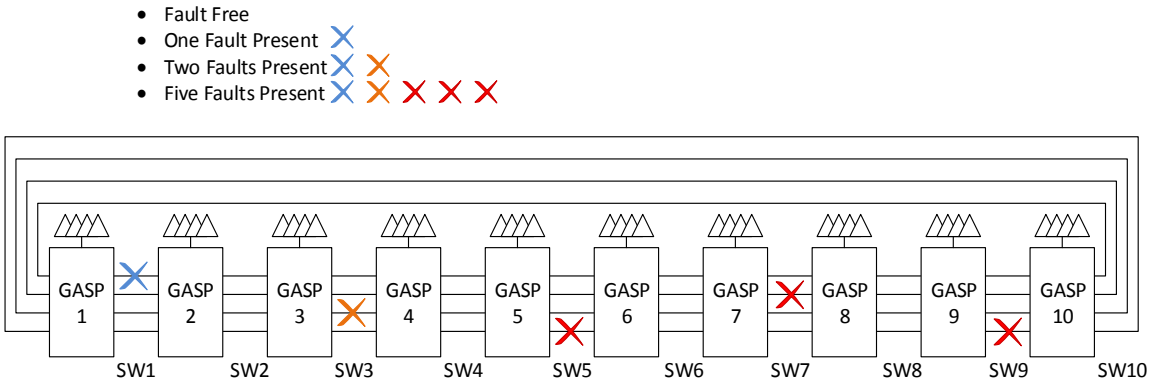

Figure 3.7: Locations of injected faults

3.9.4 Monte Carlo

The use of Monte Carlo simulations allow for another useful comparison: the performance degradation in a faulty circuit versus what would be expected from normal

process variability. It can be determined how a faulty Quadded GasP in a nominal process performs in relation to fault free circuits in the slow corner of the process.

3.10 Simulation

As manufacturing and testing the GasP circuits in silicon was out of the scope of this thesis, they were simulated with Synopsys HSpice. Monte Carlo was used to model process variation and device mismatch. The TSMC 90nm technology library implements a compact modeling scheme for Monte Carlo simulation. Each of the physical attribute of a transistor (tox, leff, etc.) is scaled linearly by the same four random variables. This makes it easy to estimate the expected performance distribution of a circuit due to variability. Two passes of Monte Carlo simulations were run: one pass varying only process and a second pass varying process and device mismatch. As there was no significant difference between the two, only the results of the process only simulations were analyzed.

Chapter 4

Results

In this chapter the area of the four GasP implementations are compared and the performance of the circuits are analyzed under four different fault scenarios: Fault free, delay faults present, SA0 faults present and SA1 faults present. Following this, the performance of the circuits in these scenarios will be contrasted with the estimated fault free performance range caused by normal process variation.

4.1 Area

The area of each of the four GasP modules was assessed by simply summing the combined areas of each gate in the circuit, using the drive strengths used in the performance simulations [8]. The area of an inverter was used to calculate the combined areas of the two pull-up and pull-down transistors. The QNOR was found to be 3.32 times larger than the SNOR, while the QNAND was 3.52 times larger than the SNAN.

	SNOR			QNOR		
Stage	Gate	Drive	Size	Gate	Drive	Size
PU	pmos	6	9.9	pmos	9	15.4
PD	nmos	6		nmos	9	
S0	inv	4	7.7	nor2	6	18.7
S1	nor2	5	16.5	nor4	3	17.6
S2	inv	12	19.8	nor2	4	13.2
S3	inv	27	41.7	nor2	4	13.2
S4	inv	4	7.7	nor2	2	7.7
Total			103.2			342.6
				Increase Factor		3.32

Table 4.1: Area comparison between SNOR and QNOR

This illustrates a benefit of the quadded circuits only being required to drive one fourth of the standard load. The smaller load means that the drive strength of the quadded gates are smaller overall than the standard. Therefor even though the quadded circuits contain four times as many gates, the area increase is less than four times that of the standard circuits.

	SNAND			QNAND		
Stage	Gate	Drive	Size	Gate	Drive	Size
PU	pmos	9	15.4	pmos	11	18.7
PD	nmos	9		nmos	11	
S0	inv	6	9.9	nand2	7	22.0
S1	nand2	8	24.2	nand4	5	31.8
S2	inv	16	25.3	nand2	5	16.5
S3	inv	32	49.4	nand2	5	16.5
S4	inv	2	4.4	nand2	2	7.7
Total			128.5			452.5
				Increase Factor		3.52

Table 4.2: Area comparison between SNAN and QNAN

4.2 Fault Free Performance

To establish a baseline of performance for each GasP configuration, the circuits were first simulated without any faults.

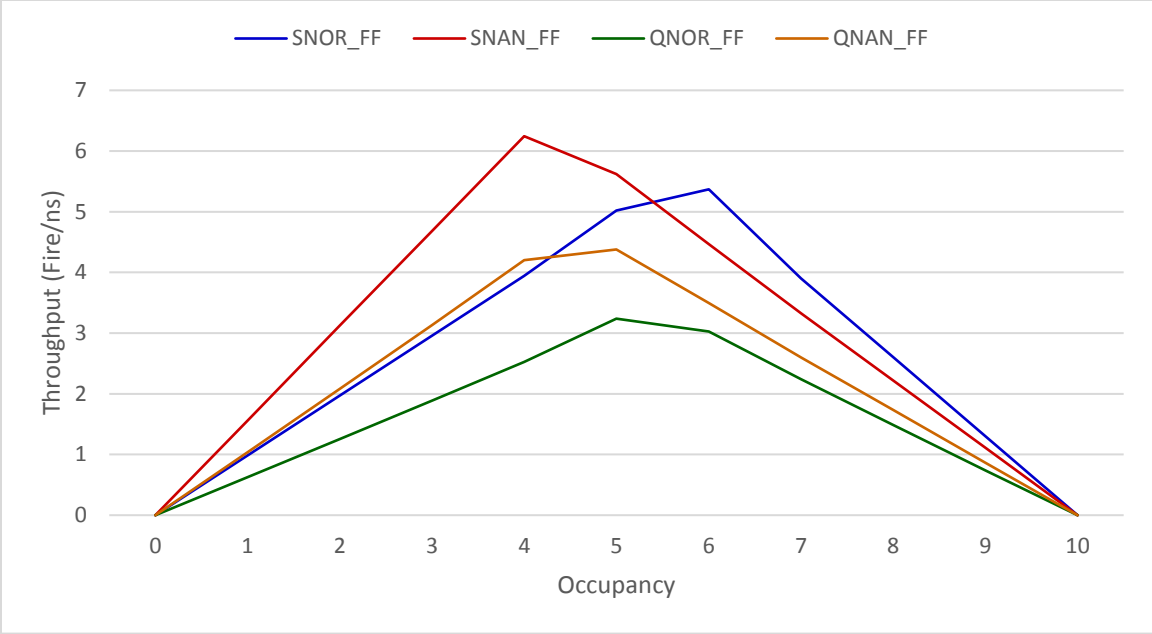

Figure 4.1: Canopy diagram for each fault free Gasp circuit

	Occupancy								
Scenario	1	2	3	4	5	6	7	8	9
QNOR_FF	-36.68	-36.46	-36.27	-35.97	-35.47	-43.62	-42.62	-42.95	-43.12
QNAN_FF	-33.33	-33.47	-33.12	-32.70	-22.10	-21.70	-21.85	-22.05	-22.21

Table 4.3: Performance reduction % for QNOR and QNAN when compared to SNOR and SNAN, respectively

4.2.1 NAND vs NOR

The canopy diagram in Figure 4.1 shows the NAND version of both standard and quadded circuits reach higher peak performance than the NOR versions, as expected. The NAND versions are centered between the 4th and 5th occupancies, while the NOR versions are centered between the 5th and 6th occupancies.

It is also expected for the circuit to behave differently at low and high occupancies, as the token path (Predecessor to Successor) and the vacancy path (Successor to Predecessor) are structurally different. The token path has six gates, but the vacancy

path only has four. In SNOR configurations for example, the token path consists of four inverters, a two-input NOR and a pull-up transistor. The vacancy path has just two inverters along with a two-input NOR and a pull-down transistor.

4.2.2 Quadded vs Standard

The two standard GasP configurations reach higher throughput performances than their quadded counterparts, as shown in Figure 4.1. This is because the cells in the standard path consist only of inverters and a single two-input combinational gate, while the cells in the quadded path are slower two- and four-input combinational cells.

The reduction in performance for the QNAN was less than that of the QNOR as expected. The QNAN performance falls approximately 33% below that of the SNAN circuit at low occupancies, and 22% below at high occupancies. The QNOR performance is 36% lower than the SNOR at low occupancies, and 43% lower at high occupancies.

It should be considered that when designing GasP circuits, it is common to add additional inverters to slow down the token path in order to properly match the delay of the data path. Therefore the degradation in performance, as well as the increase in area, could be mitigated by the reduction of additional inverters required for delay matching.

4.3 Delay Faults

Each of the four GasP configurations were simulated with delay faults injected. In theory the quadded circuits are still able to operate effectively when delay faults are

present. The following experiment allowed verification of that theory, as well as a comparison of the impact of faults on standard and quadded circuits.

4.3.1 NAND vs NOR

Both quadded and standard circuits show similar behavior differences between NAND and NOR configurations. The reduction in performance for NOR configurations is minimal at low occupancies compared to high occupancies. NAND configurations show the opposite effect, a larger performance reduction at low occupancies but a small reduction at high occupancies. It is clear that the cumulative performance reduction due to increasing number of faults is a linear function for all circuits.

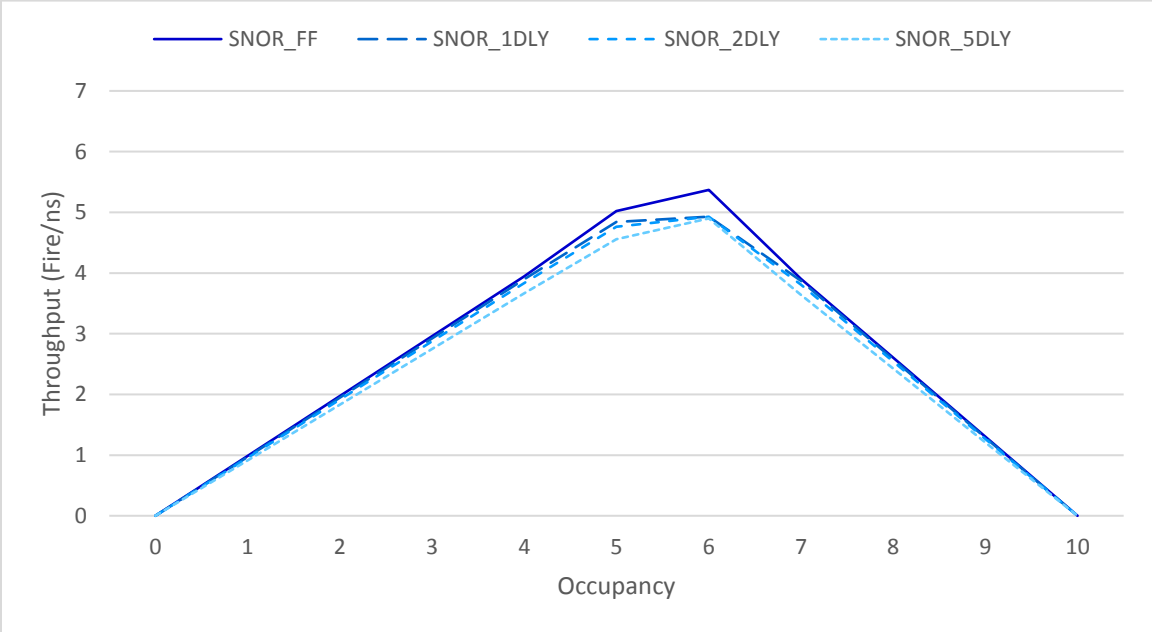

Figure 4.2: Canopy diagram for SNOR with zero, one, two and five delay faults

	Occupancy								
Scenario	1	2	3	4	5	6	7	8	9
SNOR_1DLY	-1.42	-1.27	-1.28	-1.27	-3.51	-8.21	-0.85	-0.88	-1.23
SNOR_2DLY	-2.94	-2.79	-2.81	-2.84	-5.08	-8.23	-2.36	-2.42	-2.77
SNOR_5DLY	-7.29	-7.10	-7.13	-7.15	-9.17	-8.75	-6.74	-6.72	-7.07

Table 4.4: Performance reduction % for SNOR with one, two and five delay faults when compared to fault free

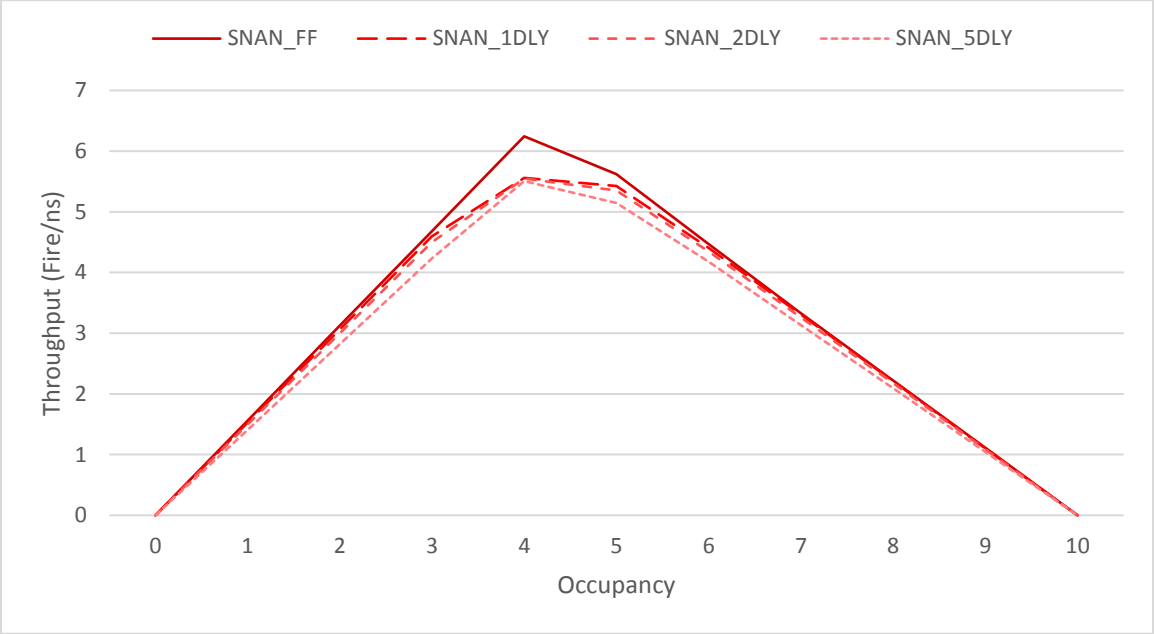

Figure 4.3: Canopy diagram for SNAN with zero, one, two and five delay faults

	Occupancy								
Scenario	1	2	3	4	5	6	7	8	9
SNAN_1DLY	-1.92	-2.05	-1.71	-10.97	-3.47	-1.41	-0.69	-0.54	-0.90
SNAN_2DLY	-4.04	-4.12	-3.82	-11.15	-4.79	-2.71	-2.04	-1.89	-2.25
SNAN_5DLY	-9.74	-9.81	-9.55	-11.82	-8.43	-6.52	-5.86	-5.72	-6.12

Table 4.5: Performance reduction % for SNOR with one, two and five delay faults when compared to fault free

4.3.2 Quadded vs Standard

The comparison between standard and quadded configurations gives a clear picture of the fault tolerance capabilities of the latter. When delay faults are introduced, the

degradation in performance in the quadded circuits is less than half what is seen in the standard circuits. Even with five delay faults injected there is not more than 4% loss in circuit performance for both quadded designs, while the standard circuits lose up to 11%.

The reason that there is a still a reduction in performance in the quadded circuits despite the built in fault tolerance can be explained by looking at stages S2 and S3 as an example. If the output of NOR1 in stage S2 is delayed then the NOR1 and NOR2 gates in stage S3 will only receive a single high input first and then a delayed second input, instead of two simultaneously. This means that until the delayed input arrives, only one of their pull down transistors will be on and the delay through the gate will be slightly longer. The gate will still be able to switch without waiting for the delayed signal, but the transition will be longer than if both inputs arrived on time.

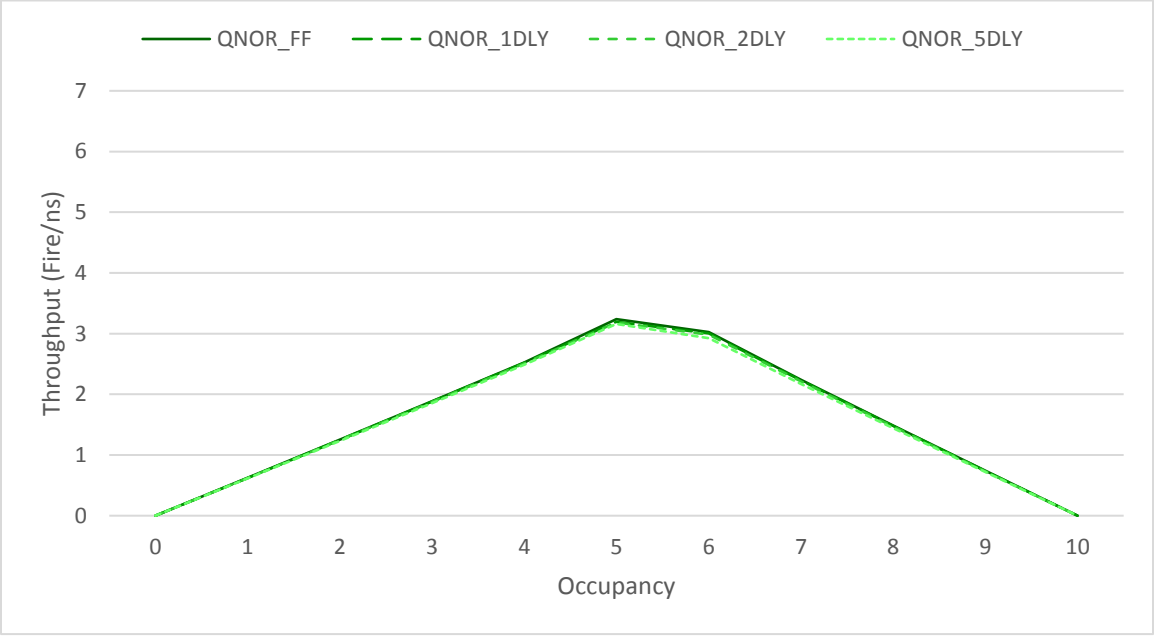

Figure 4.4: Canopy diagram for QNOR with zero, one, two and five delay faults

	Occupancy								
Scenario	1	2	3	4	5	6	7	8	9
QNOR_1DLY	-0.48	-0.24	-0.27	-0.32	-1.33	-0.83	-0.54	-0.47	-0.41
QNOR_2DLY	-0.80	-0.56	-0.64	-0.71	-1.64	-1.45	-1.21	-1.14	-0.95
QNOR_5DLY	-1.76	-1.52	-1.59	-1.62	-2.44	-3.37	-2.99	-2.96	-2.84

Table 4.6: Performance reduction % for QNOR with one, two and five delay faults when compared to fault free

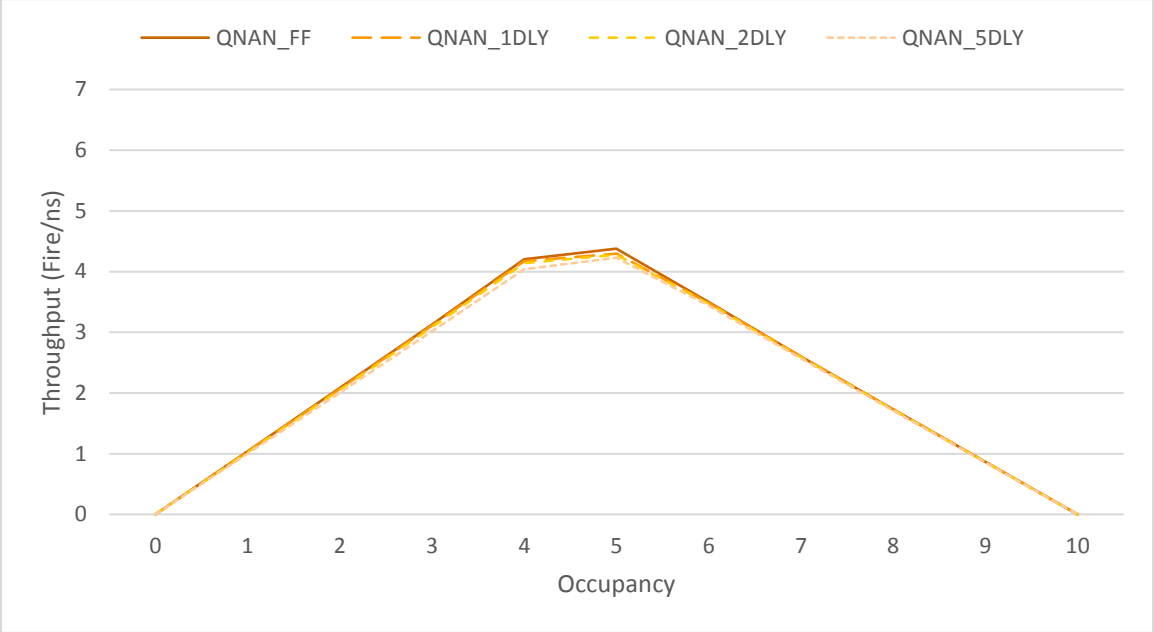

Figure 4.5: Canopy diagram for QNAN with zero, one, two and five delay faults

	Occupancy								
Scenario	1	2	3	4	5	6	7	8	9
QNAN_1DLY	-0.67	-0.67	-0.64	-0.67	-1.87	-0.40	0.00	-0.12	0.00
QNAN_2DLY	-1.44	-1.44	-1.37	-1.45	-2.31	-0.77	-0.35	-0.40	-0.35
QNAN_5DLY	-3.75	-3.70	-3.67	-3.83	-3.43	-1.72	-1.27	-1.39	-1.27

Table 4.7: Performance reduction % for QNAN with one, two and five delay faults when compared to fault free

4.4 Stuck-At 0 Faults

Only the quadded circuits were simulated with stuck-at faults present, as standard circuits cease functioning. The performance between NAND and NOR configurations is compared, as well as the difference in behavior between the delay fault and stuck-at fault models.

The QNOR circuit was found to have a different behavior in the presence of SA0 faults than the QNAN. Tables 4.8 and 4.9 show that the performance in the QNAN circuit degrades more in relation to its fault-free version than the QNOR. At low occupancies QNAN sees a reduction in performance of roughly 8.3%, and 6.4% at high occupancies. However, the QNOR only sees a 3% performance reduction at low occupancies, and an almost negligible 0.3% reduction at high occupancies.

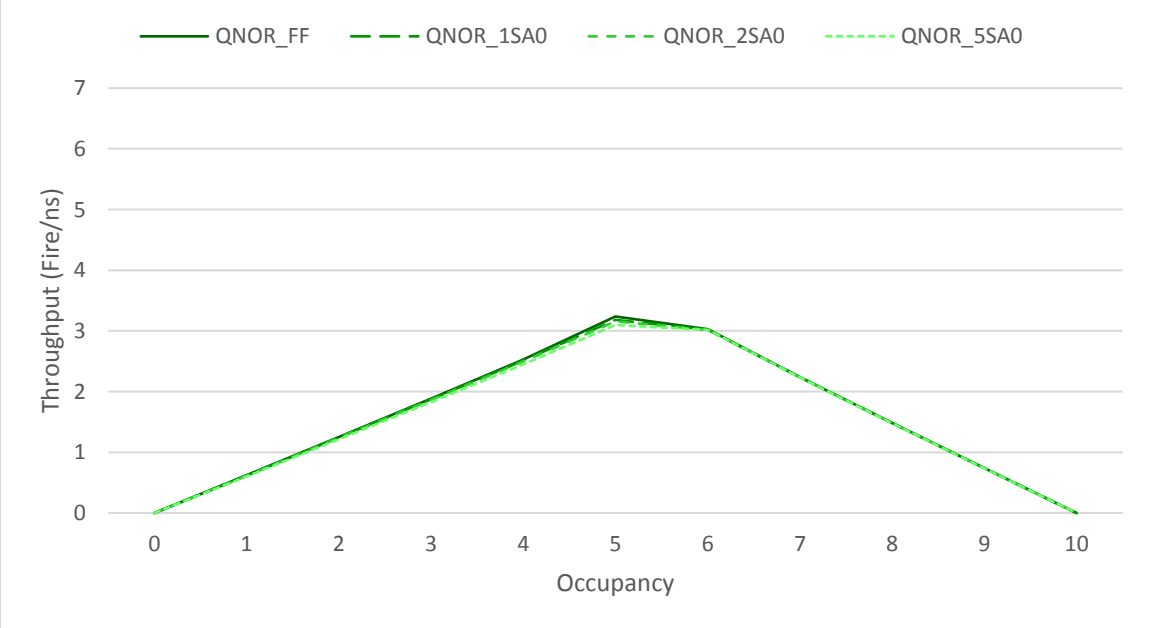

Figure 4.6: Canopy diagram for QNOR with zero, one, two and five stuck-at zero faults

	Occupancy								
Scenario	1	2	3	4	5	6	7	8	9
QNOR_1SA0	-0.64	-0.48	-0.53	-0.55	-1.73	-0.36	0.04	0.00	-0.14
QNOR_2SA0	-1.44	-1.12	-1.17	-1.27	-2.47	-0.33	0.00	-0.07	-0.14
QNOR_5SA0	-3.20	-2.95	-3.02	-3.05	-4.35	-0.36	-0.18	-0.27	-0.27

Table 4.8: Performance reduction % for QNOR with one, two and five stuck-at zero faults when compared to fault free

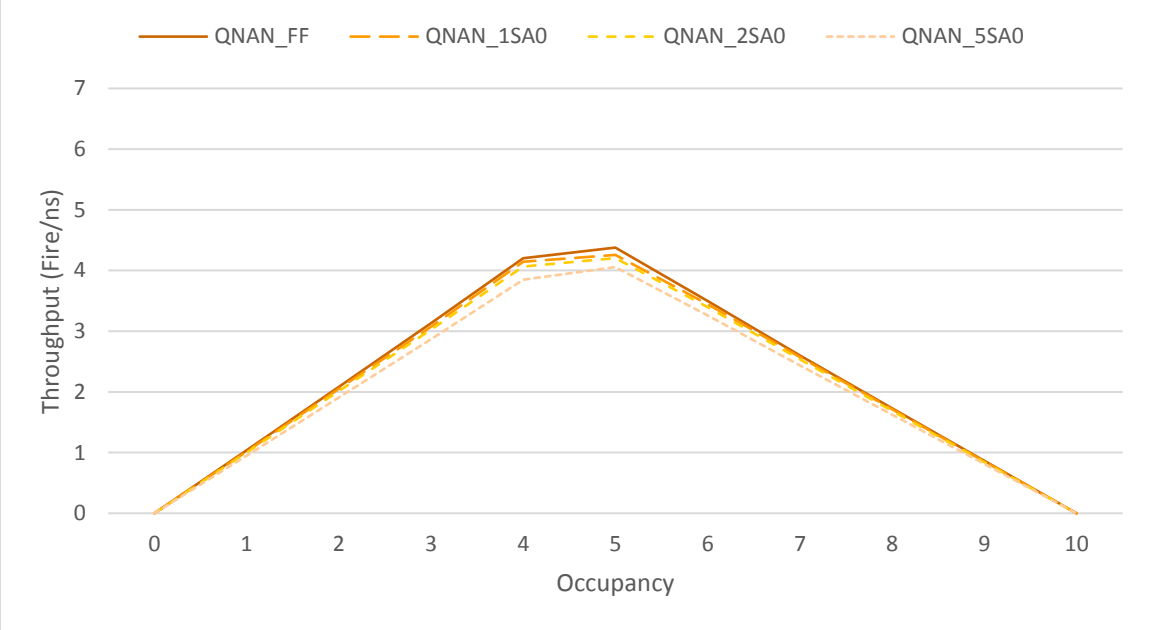

Figure 4.7: Canopy diagram for QNAN with zero, one, two and five stuck-at zero faults

	Occupancy								
Scenario	1	2	3	4	5	6	7	8	9
QNAN_1SA0	-1.92	-1.54	-1.69	-1.33	-2.74	-1.52	-1.00	-1.15	-1.16
QNAN_2SA0	-3.65	-3.27	-3.35	-3.24	-3.95	-2.92	-2.38	-2.60	-2.54
QNAN_5SA0	-8.46	-8.17	-8.27	-8.33	-7.33	-6.78	-6.23	-6.35	-6.36

Table 4.9: Performance reduction % for QNAN with one, two and five stuck-at zero faults when compared to fault free

4.5 Stuck-At 1 Faults

The behavior of QNOR and QNAN in stuck-at one simulation scenarios circuits closely resemble the results of SA0 simulations except reversed. The QNAN performs better

than the QNOR in the presence of SA1 faults. With five faults QNOR performance reduces by 7.2% at low occupancies, and 8.0% at high occupancies. The performance of QNAN with five faults actually increases at low occupancies by 0.7%, while at high occupancies it reduces by 4.0%.

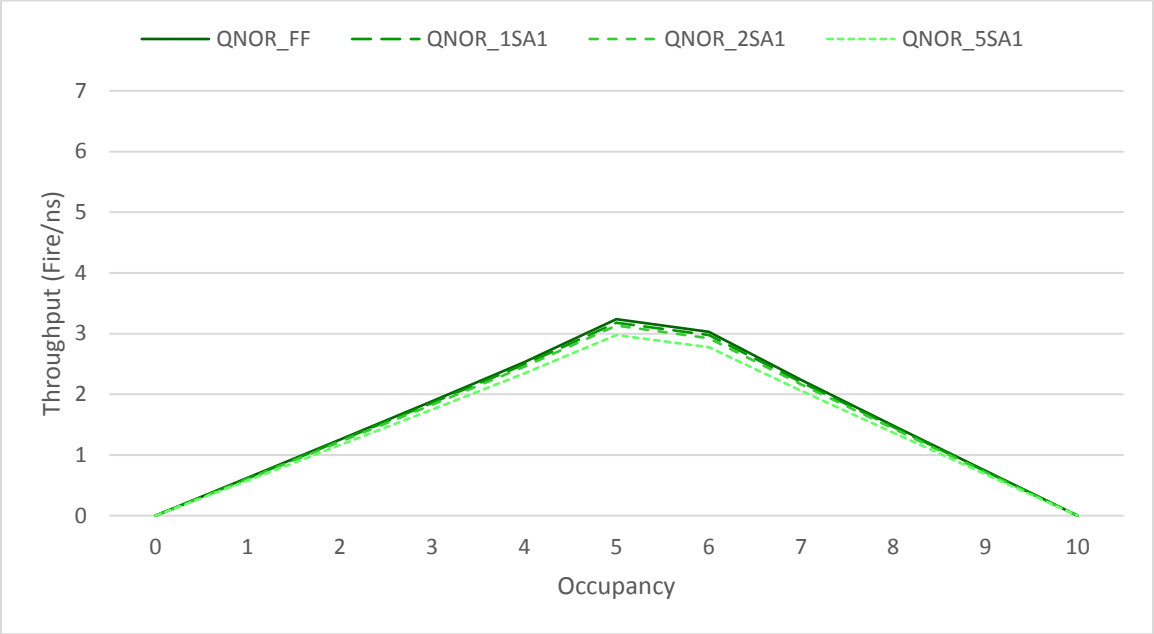

Figure 4.8: Canopy diagram for QNOR with zero, one, two and five stuck-at one faults

	Occupancy								
Scenario	1	2	3	4	5	6	7	8	9
QNOR_1SA1	-1.44	-1.36	-1.49	-1.39	-1.82	-1.65	-1.65	-1.48	-1.49
QNOR_2SA1	-3.04	-2.87	-2.97	-2.89	-3.24	-3.37	-3.26	-3.16	-3.11
QNOR_5SA1	-7.20	-7.18	-7.21	-7.21	-8.09	-8.33	-7.99	-7.81	-7.70

Table 4.10: Performance reduction % for QNOR with one, two and five stuck-at one faults when compared to fault free

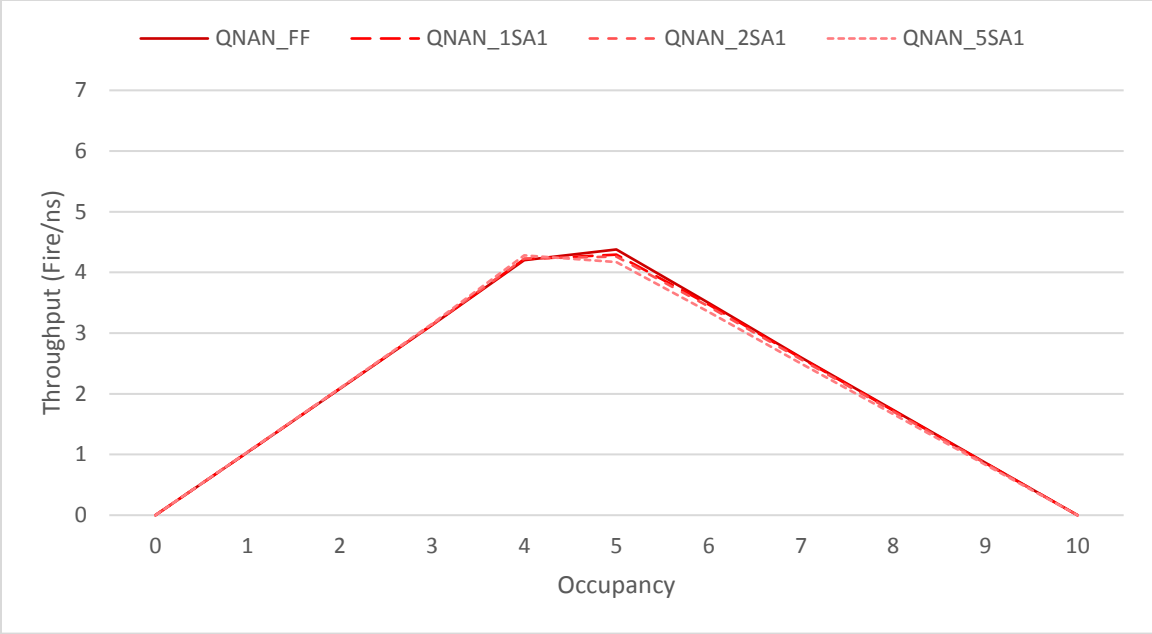

Figure 4.9: Canopy diagram for QNAN with zero, one, two and five stuck-at one faults

	Occupancy								
Scenario	1	2	3	4	5	6	7	8	9
QNAN_1SA1	0.00	0.24	0.16	0.38	-1.94	-0.80	-0.46	-0.46	-0.46
QNAN_2SA1	0.19	0.34	0.29	0.57	-2.67	-1.72	-1.35	-1.39	-1.39
QNAN_5SA1	0.48	0.67	0.67	1.88	-4.80	-4.29	-3.92	-3.87	-3.93

Table 4.11: Performance reduction % for QNAN with one, two and five stuck-at one faults when compared to fault free

4.6 Stuck-At vs Delay

As discussed previously, a stuck-at zero fault could be considered an extreme case of a delay fault, essentially a slow-to-rise fault that never actually rises. In cases where a low-to-high state transition is on the critical path, such as QNOR at low occupancies, this explains why performance reduction is worse for stuck-at zero faults than delay faults. On the other hand, when a high-to-low state transition is on the critical path, such as QNOR at high occupancies, the stuck-at zero more closely resembles an instantaneous

transition. This is why performance in these cases is much better than with delay faults. The same reasoning applies to differences between delay faults and stuck-at one faults.

4.7 Monte Carlo

Simulating the effects of process variation with Monte Carlo provides some insight into how significant the performance reduction caused by these faults can be. The following section analyzes how the performance of faulty circuits falls into the spectrum of normal process variation.

4.6.1 Standard vs Quadded

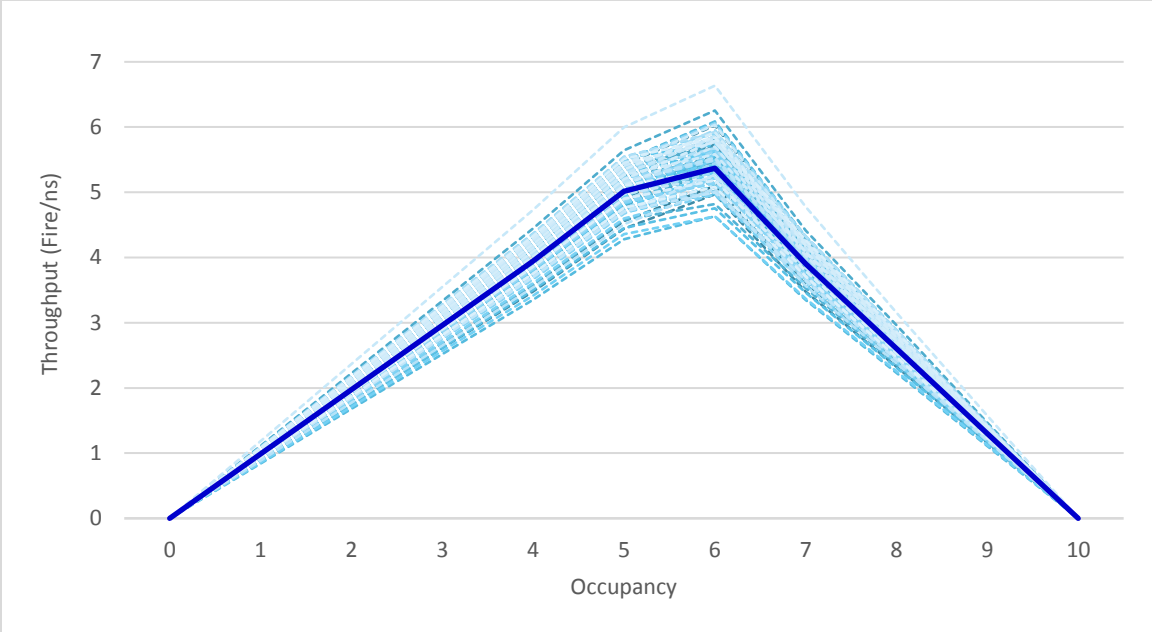

Figure 4.10: Canopy diagram for fault free SNOR TT circuit and 100 fault free Monte Carlo circuits

	Occupancy								
	1	2	3	4	5	6	7	8	9
Mean	0.994	1.990	2.985	3.979	5.056	5.481	3.941	2.626	1.313
Std. Deviation	0.062	0.124	0.185	0.246	0.310	0.356	0.246	0.161	0.081

Table 4.12: Mean and corresponding standard deviation of 100 fault free SNOR Monte Carlo circuits

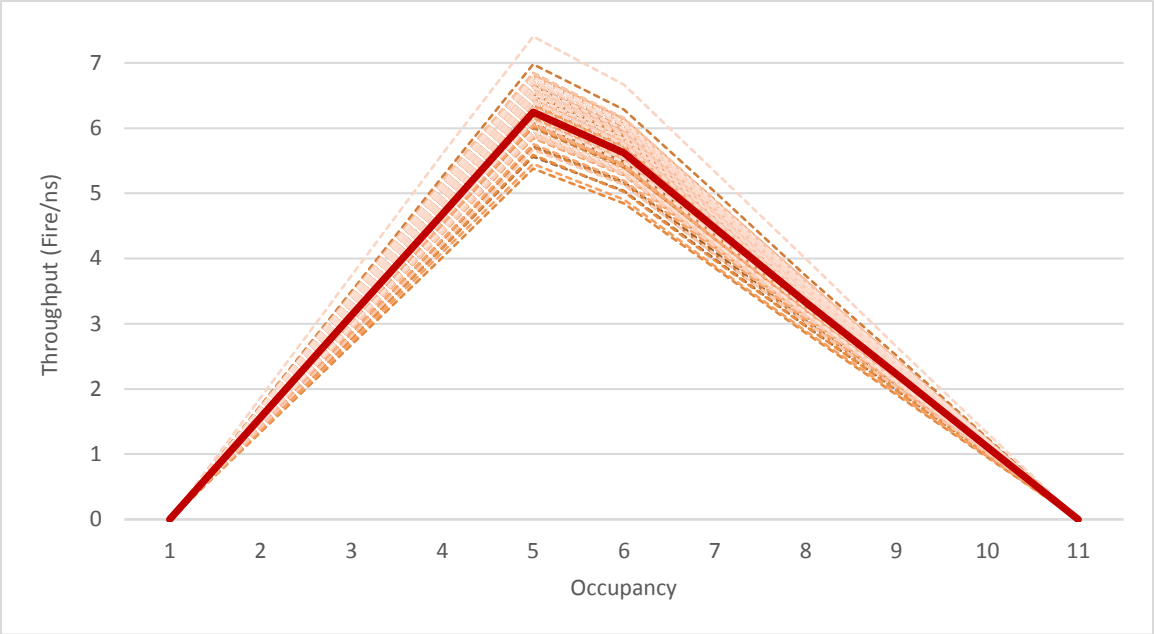

Figure 4.11: Canopy diagram for fault free SNAN TT circuit and 100 fault free Monte Carlo circuits

	Occupancy								
	1	2	3	4	5	6	7	8	9
Mean	1.572	3.148	4.728	6.293	5.662	4.502	3.357	2.240	1.121
Std. Deviation	0.095	0.189	0.284	0.360	0.323	0.263	0.200	0.132	0.066

Table 4.13: Mean and corresponding standard deviation of 100 fault free SNAN Monte Carlo circuits

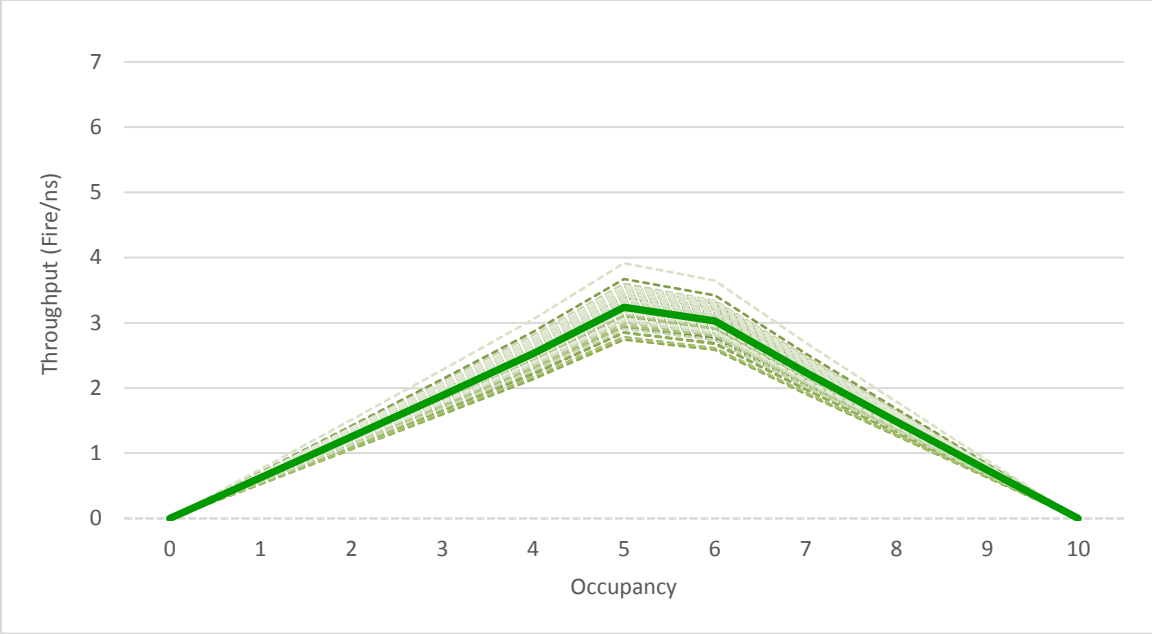

Figure 4.12: Canopy diagram for fault free QNOR TT circuit and 100 fault free Monte Carlo circuits

	Occupancy								
	1	2	3	4	5	6	7	8	9
Mean	0.629	1.263	1.901	2.546	3.265	3.052	2.255	1.496	0.746
Std. Deviation	0.040	0.081	0.122	0.163	0.210	0.188	0.140	0.092	0.046

Table 4.14: Mean and corresponding standard deviation of 100 fault free QNOR Monte Carlo circuits

	Occupancy								
Scenario	1	2	3	4	5	6	7	8	9
QNOR_1DLY	-0.182	-0.166	-0.172	-0.171	-0.333	-0.264	-0.203	-0.196	-0.199
QNOR_2DLY	-0.232	-0.216	-0.229	-0.232	-0.380	-0.365	-0.309	-0.305	-0.286
QNOR_5DLY	-0.380	-0.363	-0.377	-0.373	-0.504	-0.673	-0.594	-0.597	-0.590
QNOR_1SA0	-0.207	-0.203	-0.213	-0.207	-0.395	-0.190	-0.110	-0.120	-0.155
QNOR_2SA0	-0.331	-0.302	-0.311	-0.317	-0.509	-0.184	-0.117	-0.131	-0.155
QNOR_5SA0	-0.603	-0.585	-0.598	-0.593	-0.799	-0.190	-0.146	-0.164	-0.177
QNOR_1SA1	-0.331	-0.339	-0.360	-0.336	-0.409	-0.397	-0.381	-0.359	-0.373
QNOR_2SA1	-0.578	-0.573	-0.590	-0.568	-0.628	-0.673	-0.637	-0.630	-0.634
QNOR_5SA1	-1.222	-1.239	-1.246	-1.235	-1.374	-1.469	-1.391	-1.377	-1.374

Table 4.15: Number of standard deviations away from the fault free mean in faulty QNOR circuits

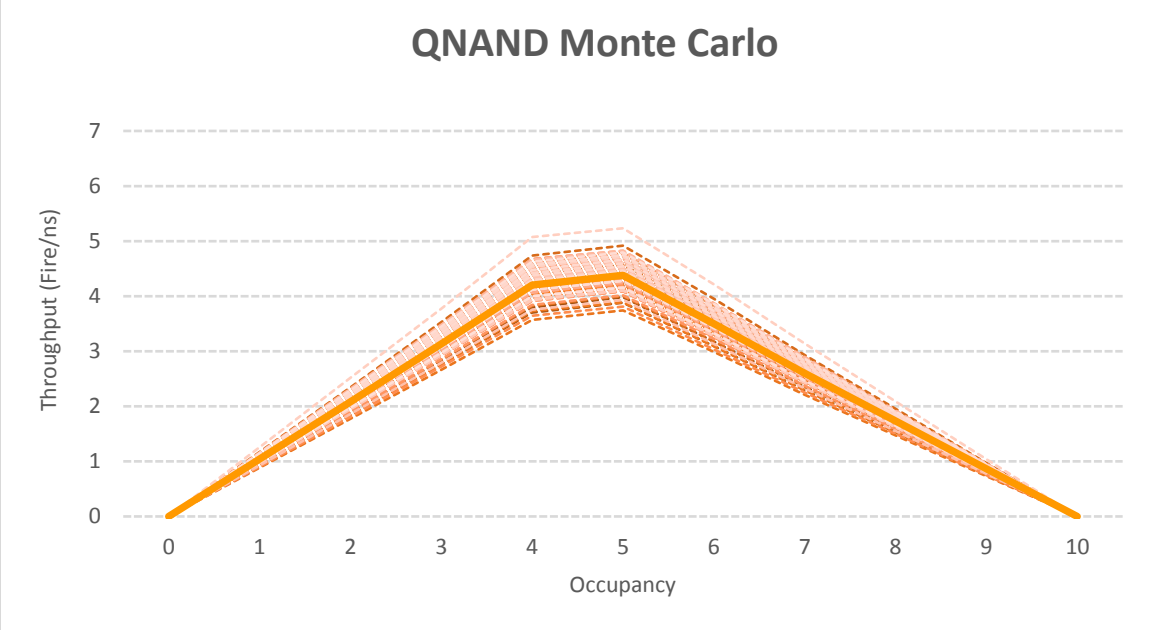

Figure 4.13: Canopy diagram for fault free QNAN TT circuit and 100 fault free Monte Carlo circuits

	Occupancy								
	1	2	3	4	5	6	7	8	9
Mean	1.048	2.100	3.158	4.241	4.412	3.529	2.623	1.746	0.872
Std. Deviation	0.066	0.132	0.199	0.267	0.267	0.220	0.166	0.109	0.054

Table 4.16: Mean and corresponding standard deviation of 100 fault free QNAN Monte Carlo circuits

	Occupancy								
Scenario	1	2	3	4	5	6	7	8	9
QNAN_1DLY	-0.225	-0.251	-0.242	-0.250	-0.434	-0.215	-0.141	-0.146	-0.132
QNAN_2DLY	-0.347	-0.372	-0.358	-0.374	-0.505	-0.274	-0.195	-0.192	-0.187
QNAN_5DLY	-0.711	-0.727	-0.720	-0.748	-0.689	-0.425	-0.339	-0.348	-0.334
QNAN_1SA0	-0.422	-0.387	-0.408	-0.355	-0.576	-0.393	-0.297	-0.311	-0.316
QNAN_2SA0	-0.695	-0.659	-0.669	-0.655	-0.775	-0.616	-0.514	-0.540	-0.536
QNAN_5SA0	-1.454	-1.429	-1.444	-1.456	-1.330	-1.230	-1.115	-1.136	-1.144
QNAN_1SA1	-0.119	-0.108	-0.116	-0.086	-0.445	-0.279	-0.213	-0.201	-0.205
QNAN_2SA1	-0.089	-0.093	-0.096	-0.056	-0.565	-0.425	-0.351	-0.348	-0.352
QNAN_5SA1	-0.043	-0.040	-0.035	0.150	-0.914	-0.834	-0.754	-0.742	-0.757

Table 4.17: Number of standard deviations away from the fault free mean in faulty QNAN circuits

The variation in performance for the quadded circuits is narrower than that of the standard circuits. For example, for an occupancy of three the standard deviation of throughput for SNOR and SNAN is 0.185 and 0.284 respectively, while for QNOR and QNAN it is 0.122 and 0.199. However since the magnitude of the performance is different, standard deviation alone is not enough to compare variation. The coefficient of variation gives the variability independent of the magnitude. It is clear from Table 4.18 that the quadded circuits see no improvement in variability.

	Occupancy								
Module	1	2	3	4	5	6	7	8	9
SNOR	0.062	0.062	0.062	0.062	0.061	0.065	0.062	0.061	0.061
SNAN	0.060	0.060	0.060	0.057	0.057	0.058	0.060	0.059	0.059
QNOR	0.064	0.064	0.064	0.064	0.064	0.062	0.062	0.062	0.062
QNAN	0.063	0.063	0.063	0.063	0.060	0.062	0.063	0.063	0.062

Table 4.18: The coefficients of variation for each GasP circuit

4.6.2 Process Variation vs Faulty Performance

Previously in this chapter the performance reduction of GasP circuits with delay and stuck-at faults was analyzed, enabling comparison between how the faults affected the different GasP configurations. Although this on its own does not quantify how much performance reduction is too much. Clearly the fact that the quadded circuits still operate with stuck-at faults present, while the standard circuits cease to function, is a quality result. However, the fact that a quadded circuit can continue to transmit correct signals does not necessarily mean that it is functional. A fault causing the performance of the circuit to fall out of specification can be just as debilitating as transmitting incorrect data.

The performance distribution of fault free circuits expected due to process variation can be calculated from the results of Monte Carlo simulations. Where the performance of the faulty circuits fall in relation to this distribution can indicate whether they are truly functional or not. For example, if the performance of a faulty circuit falls the equivalent of four standard deviations below the fault free mean it would be well outside the normal expected range of operation and therefor considered a failure. For the purposes of this analysis, it is assumed that performance falling within two standard deviations of the fault free mean is considered viable. This corresponds to 95% of the fault free device population.

Tables 4.15 and 4.16 allow comparison between the performance of the faulty circuits to the mean of the fault free populations. For each fault scenario, the table shows the number of standard deviations away from the mean that the performance falls at each occupancy. The circuit is considered functional if this value is less than two.

The tables show that all fault scenarios are in fact within two standard deviations of the mean. As discussed earlier in this chapter, the two fault scenarios that had the greatest impact on performance were five stuck-at one faults in QNOR and five stuck-at zero faults in QNAN. In both of these worst cases the performance falls within 1.5 standard deviations of the mean, well within the cutoff for functionality. This shows that in the extreme case of having several of the most disruptive faults present in the quadded circuit, performance still exceeds that of the slowest 7% of fault free circuits.

All other scenarios are within one standard deviation, and in many cases even within a half standard deviation. This signifies that the majority of faulty quadded circuits can still provide performance that exceeds the slowest 15.8% of fault free circuits. It can be inferred that the quadded circuits continue to operate with proper performance in the presence of faults.

Chapter 5
Conclusion

Asynchronous circuits are less affected by variability than synchronous circuits. Self-timed circuits, such as GasP, can provide a clocking distribution mechanism that is effected only by local variability. This reduces design and optimization cycles to meet performance specifications, which drives faster time to market. Fault tolerant circuits like Quadded Logic mitigate the need for comprehensive testing, saving time during planning, testing and debug which translates to reduced costs. This research proposed the Quadded GasP circuit, which marries fault tolerance with self-timed asynchronous circuit design, providing a clocking methodology that is robust against both variability and defects.

This study offered two flavors of Quadded GasP circuits. The area and performance cost of converting to these circuits was assessed. It was confirmed that Quadded GasP circuits maintain functional operation in the presence of delay and stuck-at faults. The degree of performance reduction incurred by faults was measured and found to fall within the expected range of operability. It can be concluded based on this research that Quadded GasP is a practical option to manage increased variability and defect density.

5.1 Future Work

A logical follow-on to this research would be to perform the physical layout the Quadded GasP circuit that has been presented. This would allow for a study into the

routability of the quadded nets, as well as a more accurate assessment of the area and performance.

Quadded Transistors is a fault tolerance technique similar to Quadded Logic but rather than adding redundant gate, redundant transistors are added into the gates themselves. There has been study into the concept of joining Quadded Transistors with Quadded Logic, whereby the gates in the final stage are converted into quadded transistor gates [8]. This adds fault tolerance to the final stage of gates which is one of the few weak points of Quadded Logic designs. This would be a step toward further increasing the fault tolerance of Quadded GasP.

GasP is just one of several asynchronous pipeline designs, such as Mousetrap or Dynamic Logic pipelines [9]. Integrating Quadded Logic into these other asynchronous circuits could provide the same benefits as Quadded GasP.

References

[1] S. Zeidler and M. Krstić, "A survey about testing asynchronous circuits," *Circuit Theory and Design (ECCTD), 2015 European Conference on*, Trondheim, 2015, pp. 1-4.

[2] J. von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," *Automata Studies,* C.E. Shannon and J. McCarthy, eds., Princeton Univ. Press, 1956, pp. 43-98.

[3] J. Han, Jianbo Gao, Yan Qi, Pieter Jonker, and Jose A. B. Fortes. "Toward Hardware-Redundant, Fault-Tolerant Logic for Nanoelectronics". IEEE Des. & Test of Computers, 22(4):328–339, 2005.

[4] J.G. Tryon, "Quadded Logic," *Redundancy Techniques for Computing Systems,* R.H. Wilcox and W.C. Mann, eds., Spartan Books, 1962, pp. 205-228.

[5] I. Sutherland and S. Fairbanks, "GasP: A minimal FIFO control," in Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems. IEEE Computer Society Press, Mar. 2001, pp. 46-53.

[6] I. Sutherland, et al., "Method and apparatus for asynchronously controlling state information within a circuit," U.S. Patent 6 420 907, Jul, 16, 2002.

[7] W. R. Daasch, S. Jain and D. Armbrust, "Analyzing the Impact of Fault-tolerant BIST for VLSI Design," *2008 IEEE International Symposium on Defect and Fault Tolerance of VLSI Systems*, Boston, MA, 2008, pp. 152-160.

[8] W. R. Daasch, private communication, Nov, 2016.

[9] J. Han, E. Leung, L. Liu and F. Lombardi, "A Fault-Tolerant Technique Using Quadded Logic and Quadded Transistors," in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 23, no. 8, pp. 1562-1566, Aug. 2015.

[10] S. M. Nowick and M. Singh, "Asynchronous Design—Part 1: Overview and Recent Advances," in *IEEE Design & Test*, vol. 32, no. 3, pp. 5-18, June 2015.