YIELD MODELING AND ANALYSYS OF A CLOCKLESS ASYNCHRONOUS WAVE PIPELINE WITH PULSE FAULTS

By

TAO FENG

Bachelor of Science

Northwest Agricultural University

Xi'an, P. R. China

1996

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
In partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May 2003

YIELD MODELING AND ANALYSYS OF A CLOCKLESS ASYNCHRONOUS WAVE PIPELINE WITH PULSE FAULTS

Thesis Approved:

Thesis Advisor

Dean of the Graduate College

PREFACE

This thesis proposes a new fault model and its modeling and analysis methods in a clockless asynchronous wave pipeline. It is highly desirable to have an adequate and clockless wave pipeline-specific pulse fault rate model for establishing a sound theoretical foundation for clockless wave pipeline design for reliability. The pulse fault model is throughly identified as the unique fault specifically in the clockless wave pipeline in comparison with conventional wave and wave delay faults wave pipelines. The pulse fault rate is statistically yet practically modeled, and extensively evaluated with respect to various design parameters, such as yield, fault coverage, defect-level, and request level length.

This thesis has proposed the pulse fault model for the two-phase clockless asynchronous wave pipeline, and its fault rate has been statistically yet practically derived. The pulse fault model has been throughly identified as the unique fault of the clockless wave pipeline in comparison with wave and wave delay faults of conventional wave pipelines. The pulse fault rate is statistically yet practically modeled, and extensively evaluated with respect to various design parameters, such as yield, fault coverage, defect-level, and request level length. The simulation results have revealed that the proposed pulse fault in association with the request level length has great effect on the integral pulse fault rate. Also, it has been demonstrated that an increased request level length may significantly affect the pulse fault rate to decrease, while in consequence the yield is enhanced and defect level is decreased significantly. In conclusion, the proposed modeling and analysis have provided a theoretical yet practical guideline for a feasible clockless asynchronous wave pipeline design strategy for reliability.

ACKNOWLEDGEMENTS

I would like to express my appreciation to my advisor, Dr. Nohpill Park for his intellectual supervision, crucial guidance and inspiration. I am especially appreciative for his considerateness and patience. Without his understanding and help I would not have finished my thesis smoothly and timely. I am grateful to my other committee members: Dr. K. M. George and Dr. G. E. Hedrick, whose guidance, assistance, encouragement, and friendship were also invaluable.

Moreover, hearty thanks go to my family for their endless love and my friends for their support throughout this whole process. I also would like to give my special appreciation to Mr. Byoungjae Jin, N. –J. Park, Jike Cui, Fan Yang and Miss Min Cai, who provided precious suggestions and assistance for this study.

Finally, Thanks to all the people that work for computer science department for taking such good care of the students during our time at OSU. Thank you for you always shown kindheartedness and thoughtfulness, lend hands without hesitation whenever we need.

Table Of Contents

1 Introduction 1

2 Preliminaries and Review 5

3 Clockless Asynchronous Wave Pipeline 12

4 Proposed Pulse Fault Model 22

5 Pulse Fault Rate Analysis 27

6 Conclusion 35

List of Tables

5.1 Parameter Values Simulated 33

6.1 Simulation Results : strategy 1 40

6.2 Simulation Results : strategy 2 40

6.3 Simulation Results : strategy 3 41

6.4 Simulation Results : strategy 4 41

List of Figures

1.1 Conventional Synchronous Linear Instruction Pipeline 2

1.2 Instruction Wave Pipeline 3

2.1 Conventional Pipeline 5

2.2 Typical Synchronous Wave Pipeline 6

2.3 Clockless Asynchronous Wave Pipeline 9

3.1 Two-phase asynchronous wave pipeline [7] 13

3.2 Transparent Switch 15

3.3 Opaque Switch 15

3.4 Slices of two-phase asynchronous wave pipeline [7] 16

3.5 Beginning of a data wave entering a AWP 19

3.6 Data wave before a transparent switch 20

4.1 Data wave before an opaque switch 22

4.2 Ideal case of operation of data waves and request pulse 23

4.3 Setup time pulse fault 24

4.4 Hold time pulse fault 25

5.1 Pulse Fault Probability 28

5.2 Relative position between the request signal and data wave 29

5.3 d and $(\triangle_{max} - \triangle)$ are at the left side of data skew 31

5.4 Data skew is in the middle of request level 32

5.5 Divided d and $\triangle$ into two parts equally 32

6.1 Length of Request Level vs. Pulse Fault Rate 42

6.2 Length of Request Level vs. Yield 42

6.3 Length of Request Level vs. Fault Coverage 43

6.4 Length of Request Level vs. Defect Level 43

Chapter 1

Introduction

Wave pipeline is a cutting-edge technology for extending the performance of modern microprocessors to their maximum. This technology has been extensively researched to achieve significant performance upgrade [4]. Wave pipeline can be implemented either synchronously or asynchronously. Clockless wave pipeline is one of the asynchronous implementations without internal clock-controlled-registers to further improve the performance of synchronous wave pipeline. Instruction pipeline is a realization of linear synchronous pipeline in which performance is improved through instruction-level parallelism by allowing to start execution of an instruction before the previous ones already in the pipeline are finished. The architecture of conventional synchronous linear instruction pipeline is shown in Fig.1.1.

In order to increase the throughput of the pipeline, the wave pipeline technology was introduced by Cotton [2] in 1969. This pipeline was also referred to as the *maximum rate pipeline*. Cotton observed that the rate at which logic can propagate through the circuit depends not on the longest path delay, but on the difference

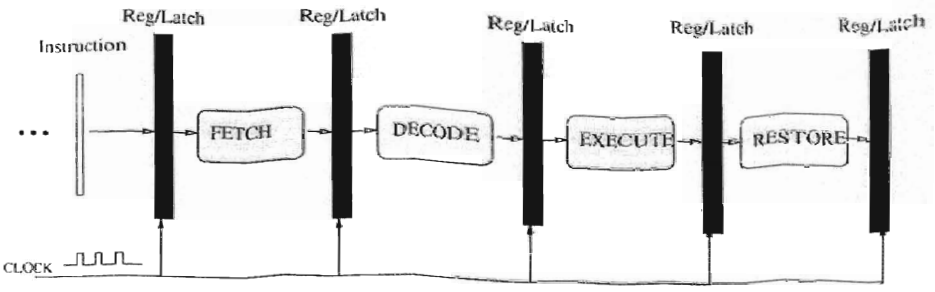

Figure 1.1: Conventional Synchronous Linear Instruction Pipeline

between the longest and the shortest path delays. Hence, by removing the internal latches, balancing all the logic paths within the circuit, and allowing multiple *waves*, the logic bits can propagate through the combinational circuit in each stage without any delay for register-latching. Thus, a clock period much shorter than the one for the longest path in the circuit can be obtained. To achieve this it must be guaranteed that no fast data wave overruns a previous slow data wave that would result in data loss. Therefore, it is critical and the challenging in wave pipeline design that all paths in the combinational logic be well balanced. The balancing of paths can be implemented in two ways: *rough tuning* and *fine tuning*. Rough tuning is to equalize path delays by inserting delay elements to the fast path, and fine tuning is by adjusting gate delays to achieve path equivalence.

Wave pipeline has been extensively researched in both academic and industrial sectors [6], and three to four times of speed-up has been reported [4]. Many research efforts of wave pipeline have been focused on synchronous wave pipeline (SWP), which is a wave pipeline using clock to control the latches operating in parallel. Many works also have been done to research and enhance SWP into an effective and

reliable computer technology [1], such as modeling and analysis of correct timing [12], [3]; development of logic synthesis and computer-aided design (CAD) tools for SWP circuits [16],[13]; and development of new wave pipeline-specific circuit [10].

A conventional synchronous instruction wave pipeline architecture is shown in Fig.1.2. Each active instruction in the pipeline can be regarded as a wave of input data. By removing the internal register-latches, multiple instructions can propagate through the same logic stage simultaneously at a higher clock frequency. Note that this wave pipeline is still synchronous because the primary input (PI) and the primary output (PO) operate synchronously under the control of the clock.

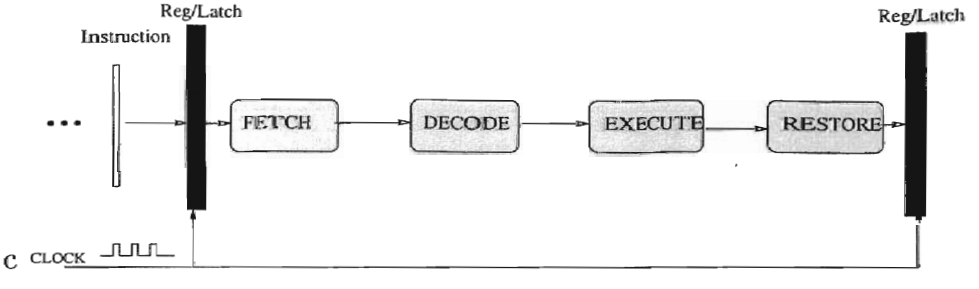

Figure 1.2: Instruction Wave Pipeline

Most recently, researches have been focused on clockless asynchronous wave pipeline (AWP), which replaces the clock with request and acknowledgement signal, or just request signal to realize clockless asynchronous wave pipeline operation. Basic architectures of AWP have been presented in [7], [8]. Since logic stages operate asynchronously, AWP is potentially faster than SWP, and thus more attention has been paid recently. AWP has two basic types. One uses both request and acknowledge signals, i.e., handshaking protocol. This type of AWP is able to buffer signlas to

impose strict control, but extra delay is required due to the waiting period for acknowledgement signals. The other kind of AWP uses only request signal without acknowledgement signals and could run much faster than the former type of AWP. This kind of AWP is to be investigated in this work. However, fault models, testing, fault tolerance and defect-level issues of such AWP have not been adequately and extensively addressed.

The objective of this thesis is to identify an adequate and clockless AWP-specific fault model, and provide a sound theoretical yet practical foundation for AWP design for reliability. Specifically, this thesis will address and propose a method for modeling and analysis of the effect of the new AWP-specific fault associated with AWP design parameters on yield, defect-level, thereby providing an effective and accurate model for AWP testing algorithms and ultimately fault-tolerance.

Chapter 2

Preliminaries and Review

In conventional pipeline, there are registers or latches between any two stages. At any instant of time, there is at most one data active in a single stage as shown in Fig.2.1. Whereas, in wave pipeline, the internal registers or latches are removed, and multiple data can be active in the same logic stages simultaneously as shown in Fig.2.2. The clock cycle time in conventional pipeline is determined by the maximum stage delay, that is $T_{ck} \geq D_{max}$ (where T_{ck} is the clock cycle time and D_{max} is the maximum stage delay).

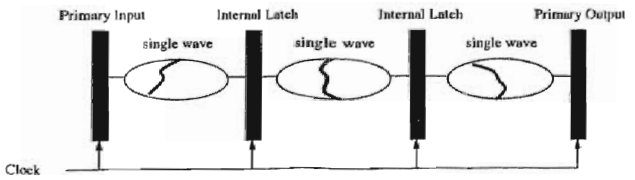

Figure 2.1: Conventional Pipeline

In wave pipeline, the time constraint is more stringent. The clock cycle time is determined not by D_{max} but by the relative difference between D_{max} and D_{min}, that

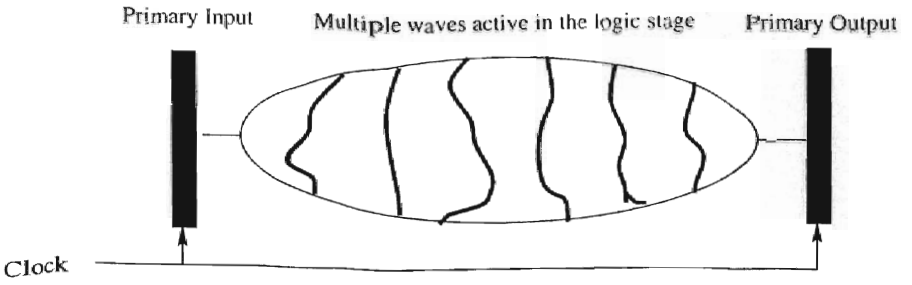

Figure 2.2: Typical Synchronous Wave Pipeline

is $\frac{T_{max}}{N} < T_{ck} < \frac{T_{min}}{N-1}$ [5] (where N is the number of waves in the circuit, T_{max} is the maximum path delay and T_{min} is the minimum path delay.

In order to model the classic SWP timing constraints, the following parameters are defined [5].

- D_{min}, D_{max}: Minimum and maximum propagation delays in a combinational Logic stage.
- T_{ck}: Clock period
- T_{su}, T_h: Register setup and hold times
- D_r: Propagation delay of a register
- $\triangle$: Clock skew between the primary output and input registers. This skew is intentionally inserted between two clock signals and for adjustment of delay.
- $\triangle_{ck}$: Worst case uncontrolled clock skew due to the delay differences along the clock lines.

- $d_{max(x)}, d_{min(x)}$: Maximum and Minimum propagation delays from primary inputs to node x.

- T_{sx}: Temporal separation of the waves at internal nodes

- N: Number of clock cycles needed for a signal to propagate through the logic stages. (i.e., the number of data waves present in the logic stages or the degree of wave pipelining).

- T_L: Time at which a data is sampled at the output register.

Sampling at the primary output registers, the following holds.

$$T_L = NT_{ck} + \triangle \tag{2.1}$$

The output data is to be clocked after the last bit arrived at the primary output and before the first bit associated with the next clock cycle arrives at the primary output, so it has a two sides time constraint.

The constraint for clocking the last bit of a data wave requires that the last bit arrive early enough to be clocked by the output register during N_{th} clock cycle, and can be expressed as follows.

$$T_L > D_r + D_{max} + T_{su} + \triangle ck \tag{2.2}$$

The constraint for clocking the first bit of a data wave requires that the arrival of the next bit does not interfere with the clocking of the current data wave as follows.

$$T_L < T_{ck} + D_r + D_{min} - (T_h + \triangle ck) \tag{2.3}$$

To combine the constraints 2.2 and 2.3, the maximum rate pipelining condition by Cotten can be applied as follows.

$$T_{ck} > (D_{max} - Dmin) + T_{su} + T_h + 2\triangle ck \tag{2.4}$$

The minimum clock period is limited by the difference of the path delays, *i.e.*, $(D_{max} - D_{min})$ and the clocking overhead, *i.e.*, $(T_{su} + T_h + 2\triangle ck)$ resulting from the insertion of clocked registers.

Constraint 2.4 guarantees that waves do not collide or overlap at the output of a logic stage [1] . Also, additional constraints need to be imposed on the internal nodes of the combinational block to prevent wave collision at the individual logic gates, i.e., the next wave should not arrive at a node until the current wave has propagated through, known as the *internal node constraint.*

Let x be an internal node, and $d_{max(x)}$ and $d_{min(x)}$ be the longest and the shortest propagation delays from the primary input to the node x, respectively. The following internal node constraint must be satisfied at each node x of the circuit.

$$T_{ck} > (d_{max(x)} - d_{min(x)}) + T_{sx} + \triangle ck \tag{2.5}$$

It is obvious that T_{sx} is equivalent to the register overhead $T_s + T_h$, and $\triangle_{ck}$ is counted only once since the output register was not accounted for.

Combining 2.2, 2.3 and 2.4, the following expression holds.

$$D_r + D_{max} + T_{su} + \triangle ck < NT_{ck} + \triangle < T_{ck} + D_r + D_{min} - (T_h + \triangle ck) \tag{2.6}$$

Let

$$T_{max} = D_r + D_{max} + T_{su} + \triangle ck - \triangle \tag{2.7}$$

and

$$T_{min} = T_{ck} + D_r + D_{min} - (T_h + \triangle ck + \triangle) \tag{2.8}$$

Then, finally the following expression holds.

$$\frac{T_{max}}{N} < T_{ck} < \frac{T_{min}}{N-1} \tag{2.9}$$

If $N = 1$, then it is the case of conventional pipeline where the clock period is only lower-bounded on the maximum delay in a logic stage. In case of larger values of N, however, it shows that the clock period is also upper-bounded on $\frac{T_{min}}{N-1}$.

Latching a data during data propagation through a stage, the faster bits of a data must wait at the output line of the latch until all the bits reach the latch and lined up. The clock will then turn high, and all of the data bits begin to move forward simultaneously. SWP is register-based, and the input data are globally latched. All the data bits within SWP are latched by the same clock. They always begin to propagate toward next stage at the same time. Whereas, AWP is not register-based and there is only local latching. The bits of a single data wave are latched at the same time locally and asynchronously in each stage as shown in Fig.2.3.

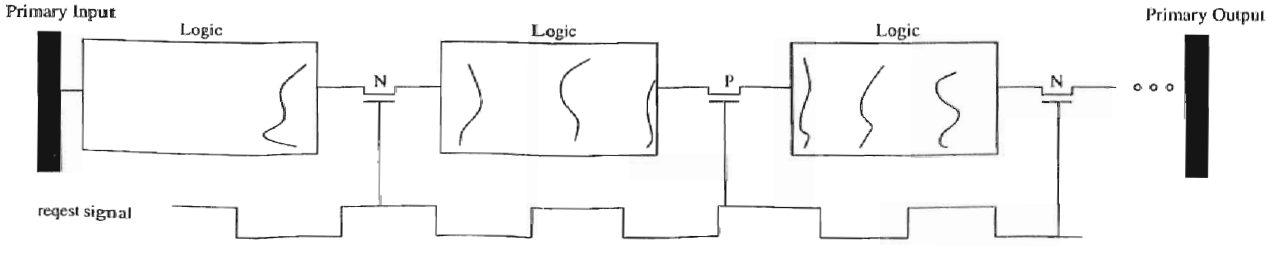

Figure 2.3: Clockless Asynchronous Wave Pipeline

The critical speed-limiting factors in synchronous wave pipeline are uncontrolled

clock skew, sampling time at the primary output, and worst case transition time at each logic stage outputs [1]. The path delay equalization problem could be resolved theoretically while its implementation in the presence of various static and dynamic delays [1] is a real challenge since the sources of delay variations are so extensive. Note that the same kind of delay variations affect AWP as well.

Gate delay varies depending on many factors. Gate delay variations may result from variations of process parameters during manufacturing, e.g., coupling capacitance effect may cause significant changes in the gate delay especially in multilevel metal processes; noise in the power supply voltage can cause cumulative delay dispersions as waves propagate through several logic layers; changing temperature and supply voltage level can impact on gate delay; and some input pattern may also cause significant delay variations.

Testing wave pipeline circuits is a complicated process because of the following factors.

- Operating mode [11]: Since there are multiple data waves propagating through the combinational circuit simultaneously, traditional delay-fault testing is not readily able to detect faults of successive transitions in the circuits.

- New class of faults: Besides traditional stuck-at and delay faults, two new classes of faults are introduced in SWP as follows.

    1. Wave delay fault [11]: There is a two-sided time constraint for the propagation time of any path delay in the combinational logic stage. The valid clock period for an arbitrary SWP is discrete. Therefore, the valid path

delays are also discrete. If a path is too long or too short to satisfy the timing constraint on the primary output lines, it may cause *set-up* or *hold time error* on the primary output.

2. Wave fault [11]: A wave fault is caused on a node if a signal transition is invalidated by the immediate following signal transition. Then, the two successive waves come too close to stabilize signal on a node, or some waves surpass their preceding wave. This fault may cause a spike or even no transition at all, instead of a stable signal level between transitions. The wave faults can be observed and detected at primary output lines.

Chapter 3

Clockless Asynchronous Wave Pipeline

There are three major potential advantages of asynchronous wave pipeline (AWP) [9].

1. Large system can be easily built and modified in a plug and play fashion because it is not necessary to consider the global synchronization of different components.

2. Extra consideration for clock path design and extra power for clock signal can be avoided in AWP.

3. AWP may outperform SWP because fast data does not have to wait for a clock signal to be synchronized to slow data.

However, due to the clockless and asynchronous operation, clockless asynchronous pipeline can not be adequately facilitated by synchronous register-based control.

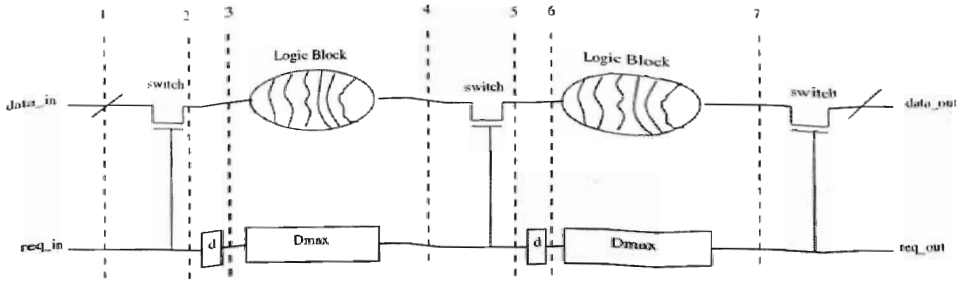

Figure 3.1: Two-phase asynchronous wave pipeline [7]

An asynchronous wave pipeline was proposed by Hauck [7], called *two-phase asynchronous wave pipeline*, which has a two-phase operation by alternating positive and negative level-sensitive switches. Instead of clock signal-based control AWP relies on a *request signal*. Although the duty of the request signal is quite similar to conventional clock signal and is generated by a clock at the primary input, the operation of switches and logic stages are asynchronous while a clock drives all registers in parallel in SWP.

As shown in Fig. 3.1 [7], the switches separate logic stages. A *request line* controls the switches behaving as a reference signal. Data waves and request pulses enter the AWP at the same time. They need to stay coherent until hit the primary output. *Delay elements* are added in the request line to emulate the propagation delay of the data waves. The delay element denoted by d represents the propagation delay of the switches. The other delay element denoted by D_{max} represents the worst-case propagation delay of the logic stage. Ideally, the request signal always get aligned to the last bit of the wave. Although the paths are equalized for delay-balancing, the path delay variations still exist. Hence, data propagating through logic stage may

get skewed since some bits run faster or slower. As a result, the wave spreads wider as the wave propagates through a path for a longer period of time. If the shape (i.e., the width) of the waves is not controlled, then the wave pipeline's utilization will not realize its maximum high.

If a wave becomes exceedingly wide, then it may interfere its adjacent waves. Consequently, less number of waves can be accommodated in the AWP at the same time. A solution to this problem is using *positive and negative switches*, which can be inserted in between the logic stages. The switches can be *transparent* or *opaque* to the data wave. If transparent, the switch turns off for a data wave just to propagate through the switch without any latching as shown in Fig.3.2. Then, the data bits start to propagate on through the next stage before the arrival of the request signal. If opaque, the switch turns on to latch through-data wave as shown in Fig.3.3. The data bits are latched being aligned at the output line of the switch. They can not propagate on forward until the arrival of their corresponding request signal. Hence, the data wave gets aligned at the output line of the switch. Thus, more waves can be active within the AWP simultaneously, and AWP's throughput is increased. Specifically, positive switches are opaque to the data associated with low request signal and transparent to those associated with high request signal, and vice versa for negative switches.

Fig.3.4 [7] shows that data wave flow through a pipeline with pre-set observation points numbered 1 through 7, as shown in Fig.3.1. Traces can be obtained by setting seven observation points and the data wave flow over the same time axis can be recorded. Data and request signal lines are drawn together in the traces. The lined

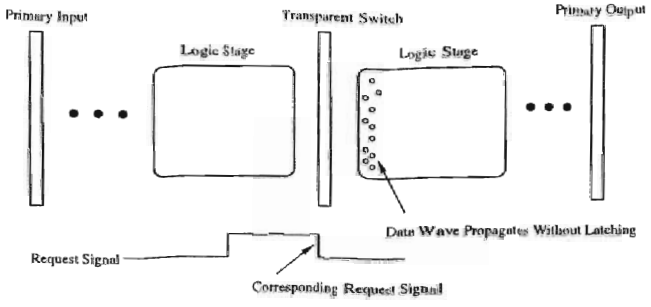

Figure 3.2: Transparent Switch

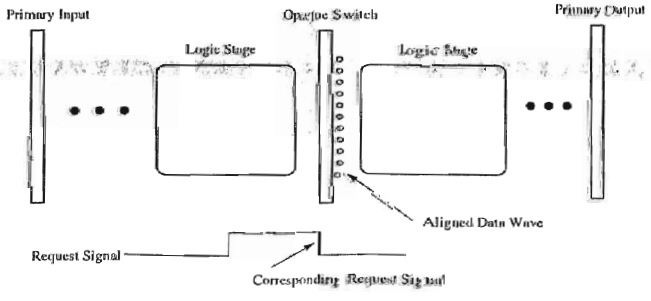

Figure 3.3: Opaque Switch

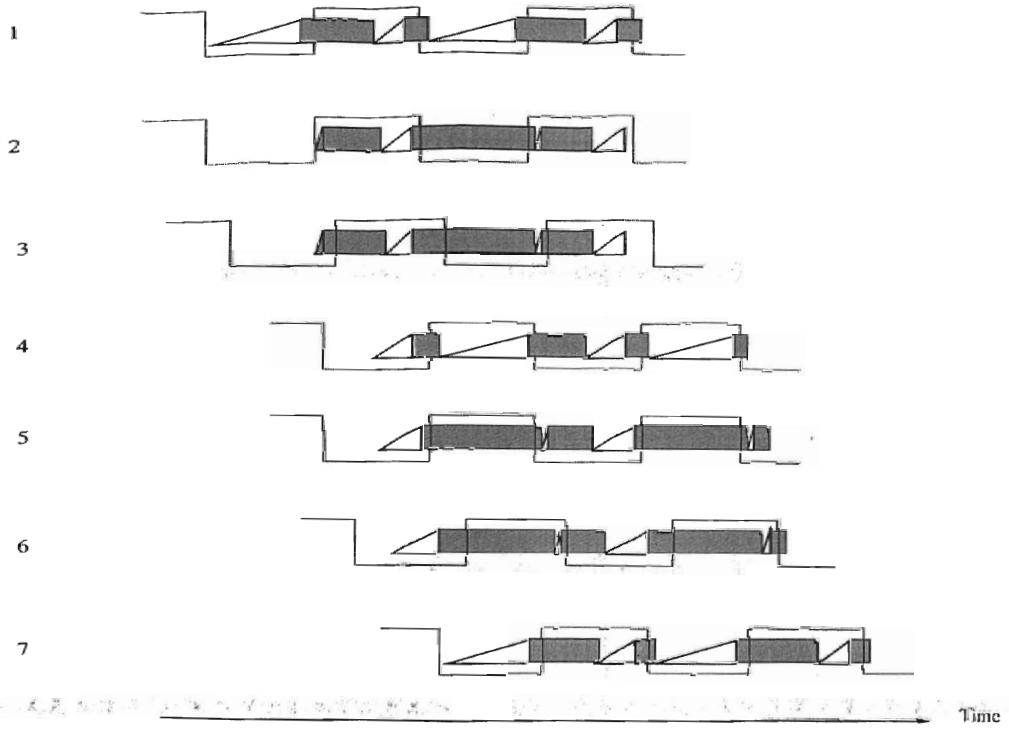

Figure 3.4: Slices of two-phase asynchronous wave pipeline [7]

signals represent the request lines, and the blocked signals represent the dynamic behavior of the data wave[7].

Since different paths have different path delays, the time when the first (fastest) bit of a data wave arrives may vary. This point corresponds to the left end of each triangle of the data wave in Fig 3.4. The rising hypotenuse visualizes the increasing number of bits arriving at the observation point. The right end of the triangle corresponds to the point of time when all bits of the data wave have arrived. From this point the data become stable, and the period of stability corresponds to the shaded rectangles [7].

Trace 1 shows the initial snapshot at the primary input of the pipeline [7]. The first switch we encounter is an N-switch where data associated with low request level are latched and aligned. This low-data corresponds to the left wide triangle lying completely across the low phase. the data has to be stablized prior to the rising edge of the request signal. The following high-data is only half through on its way to the next P-switch where it is to be latched. Therefore, it has only reached half of the maximum spread and thus the triangle is shorter.

In Trace 2, the observation point has passed the N-switch. In fact, no time has elapsed with the request signal, but there is a switch delay on the data path. Hence, the request signal arrives prior to its associated data wave. The high-data has moved closer to the falling edge.

In Trace 3, no time has elapsed with the data wave. The request signal has a delay d such that the data catches up the request signal again, and the data lies across its associated request level, where the low data is aligned. The low data is ready to enter

the next logic stage. Note that the high data wave has passed through the N-switch without being aligned with no change of its shape.

Traces 4, 5, 6 are analogous to Traces 1, 2, 3. In trace 4, there is a significant request signal delay and data wave delay with respect to the corresponding time axis. The low-data waves have now half of their maximum spread because it is on its half way through to the next N-switch. The high-data waves have their maximum spread because they have passed all the way through the next P-switch, where they will be latched and aligned.

In Trace 5, the high-data gets latched and aligned by the P-switch while the low data waves move a little closer to the rising edge. Note that there is no request signal delay. In Trace 6, the request signal is delayed to catch its associated data wave. In Trace 7, the request signal and data wave have completely passed the two-phase logic stage with no overlap between two data waves.

The throughput of AWP is determined by the minimum length of the low or high level of the request signal. The following parameters will be used for the timing analysis.

- *Partial circuit*: any two adjacent logic stages within a AWP circuit and their corresponding switches.
- *Partial path*: a path within a partial circuit.
- d_{max}, d_{min}: maximum and minimum propagation delay of a partial path within any partial circuit.

- D_{max}, D_{min}: maximum d_{max} and minimum d_{min} within the AWP circuit, respectively.
- $\triangle$: $d_{max} - d_{min}$
- $\triangle_{max}$: maximum $\triangle$
- $\triangle_{ps}$: uncontrolled pulse skew.
- T_{su}, T_h: switch setup and hold time, respectively.
- d: propagation delay of a switch.
- L_{min}: minimum length of request level.

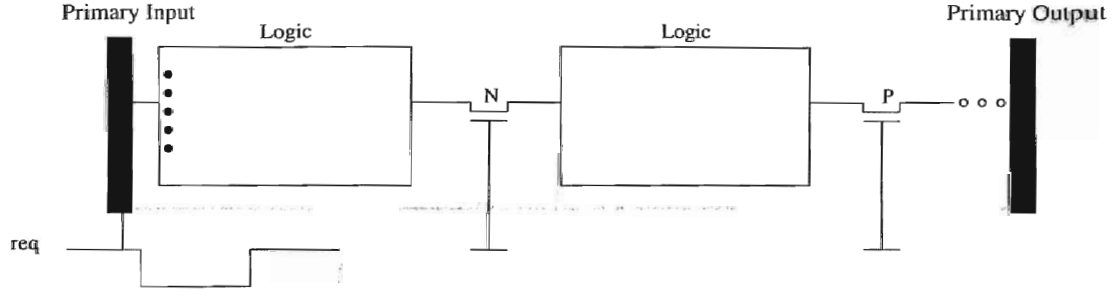

Figure 3.5: Beginning of a data wave entering a AWP

For simplicity, suppose a data wave from the primary input to the primary output is traced to calculate the L_{min}. Normally, all the bits of a data wave enter the AWP at the same time as aligned as shown in Fig.3.5. Also as shown in Fig.3.5, suppose a low request signal is associated with the data wave. The first switch the data wave will hit is an N-switch, and the N-switch is transparent to low data. Therefore, all the bits pass on through the N-switch without being latched as shown in Fig.3.6. After

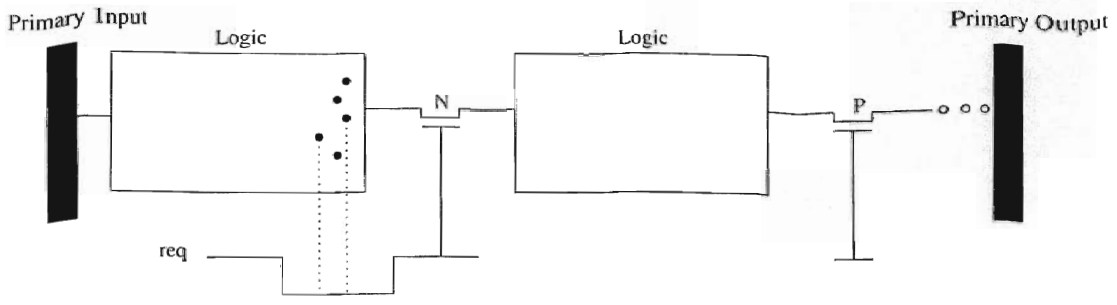

Figure 3.6: Data wave before a transparent switch

propagating through the N-switch, the data wave continues to propagate through the next logic stage and hits the next switch, a P-switch, which is opaque to the low data. Therefore, the data bits gets latched and aligned at the output line of the switch. At this point, the width (i.e., the spread of the data bits in the data wave) may become the widest on this partial path as shows in Fig.4.3. The $\triangle$ can be captured at this point. After passing through this switch, the width of the data wave returns to 0 (i.e., aligned). Then, the data wave will propagate on toward the next partial path to get aligned untill the next P-switch. Then at this point, another $\triangle$ can be captured. Continuing on this way, finally at the primary output, the maximum $\triangle$ can be captured as referred to as $\triangle_{max}$.

In order to make the data wave successfully passing through the AWP without faults, the request signal level must always maintain the proper association with its corresponding data wave. As shown in Fig.4.3, the minimum length of request level is

$$L_{min} = \triangle_{max} + T_{su} + T_h + d + 2\triangle_{ps}. \tag{3.1}$$

The reason for T_{su} and T_h is that the data wave must stay stable before the rising edge and after the falling edge of the request signal. Otherwise, the input data will be observed as an invalid input data by the switch and correct output value cannot be guaranteed. The high-data is delayed by the N-switch and moves closer to the falling edge while it has to become stable before the falling edge [7]. Therefore, the request signal must operate with an adequate delay (i.e., d) to orchestrate with the switch delay.

Chapter 4

Proposed Pulse Fault Model and Yield

In ideal case, there are request signals before and after a data wave as shown in Fig.4.1, where note that there are two $\triangle_{ps}$ are needed.

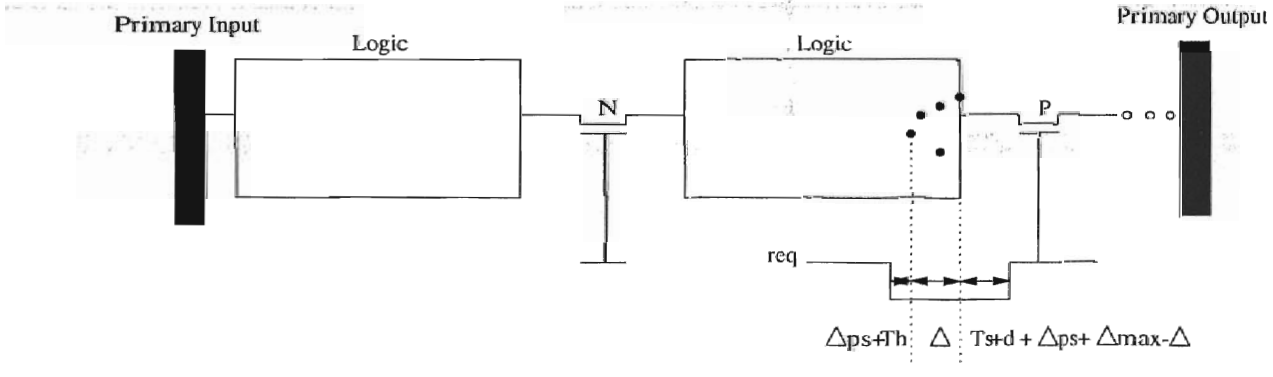

Figure 4.1: Data wave before an opaque switch

Some manufacturing factors such as temperature and supply voltage variations [9] may cause uncontrolled pulse skews on the request signal. This possible pulse

skew must be taken into account to practice reliable AWP operation. Based on the two-phase AWP, a new type of clockless AWP-specific fault is proposed and referred to as *pulse fault*.

The request signal is the reference for data waves. Ideally, both of them should enter the pipeline simultaneously. After entering the pipeline, the data wave and the request signal are supposed to stay coherent until the primary output so that the pulse can implicitly control the switches properly in the course of data wave propagation. The ideal case of AWP operation is shown in Fig.4.2.

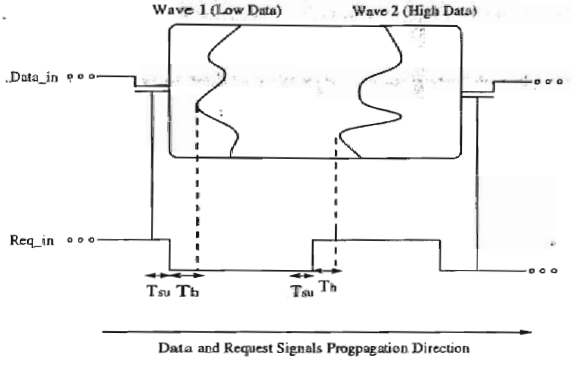

Figure 4.2: Ideal case of operation of data waves and request pulse

In Fig.4.2, the falling or the rising edge is the request pulse signal corresponding to their request level. For proper operation, the data wave must always lie within its corresponding request level, while satisfying the setup time and hold time constraints as well. When a data wave is passing a switch and the corresponding request level fails to completely cover the entire data wave, then the pulse fault occurs. There are two types of the pulse fault proposed. One is that the request level is somewhat slower such that some bits of the data wave proceed faster than the associated request

level, or can not satisfy the setup time constraint. This is referred to as *setup time pulse fault* as shown in Fig.4.3. Setup time pulse fault can occur at either N-switch or P-switch. Suppose a low data wave is about to pass a P-switch, and all the bits are normally to be latched and aligned at the output line of the switch. If some bits of the data wave are too fast and can not stay stable during the setup time of the previous request signal, or some bits even overlap the preceding signal, then these bits will go through the P-switch without being latched. Consequently, these bits mess up with the previous wave's bits and the pulse fault then occurs. Likewise, if a data wave encounters an N-switch, it normally pass through it without being latched. If some bits of the data wave run too fast, then these faster bits will be latched and aligned at the N-switch, just as the bits of the preceding data wave. The consequence is that the faster bits mess up with the previous wave's bits.

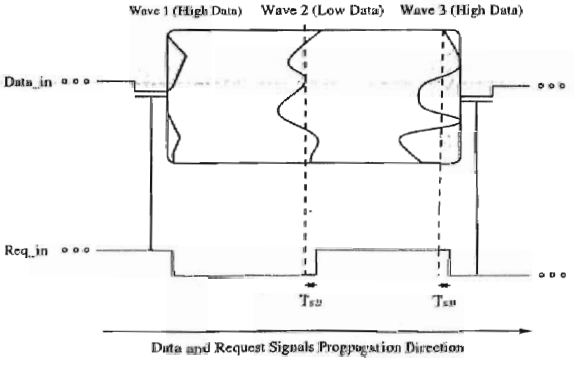

Figure 4.3: Setup time pulse fault

The other type of the pulse fault is referred to as *hold time pulse fault* and is shown in Fig.4.4. It occurs because some bits of the data wave are too slow such that the pulse signal proceeds fater than them , or the last bits can not stay stable during the

hold time of its associated request pulse. Thus, the last bits of the current wave are treated as part of the next wave. The bits of two adjacent waves mess up and the final output becomes invalid. The hold time pulse fault can also occur at either N-switch or P-switch. Suppose there is a low data and an N-switch, and the bits are to pass on through the switch without latching. If the pulse signal proceeds its associated last bit, or some last bits can not satisfy the hold time constraint of the pulse signal, then the last bits will get aligned just as the next wave's bits instead of moving ahead. Likewise, when a low data passes a P-switch, the bits should be aligned at the output line of the switch. However, if some bits are lagging behind the associated request signal, then the pulse arrives and let the aligned bits start propagating on through the next logic stage before these last bits arrive at the switch. Then, these late bits will pass on through the switch without being latched being mis-treated as the bits of the next wave.

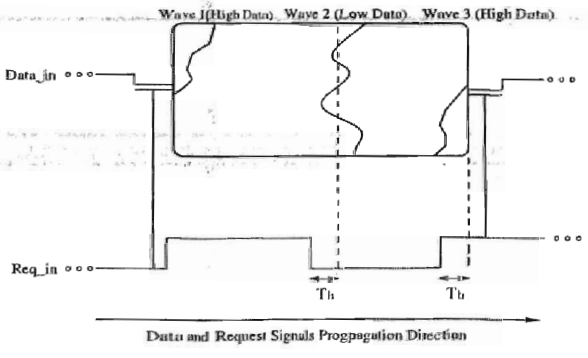

Figure 4.4: Hold time pulse fault

The proposed pulse fault is substantially different and clockless AWP-specific from the *wave delay fault* and *wave fault* with SWP [11]. In AWP, there is no wave delay

fault on the primary output because there is no clock input to the primary output latch; and as long as the request level covers the associated data wave at the primary output, the data is valid. However, wave faults still exist in AWP as they can occur at any internal gates and switches. Note that the pulse fault is inclusive of the wave fault at switches in the sense that since each data wave is associated with a request level, and two request levels can not overlap, in order to have two waves overlapped at a switch, one data wave or both of them must not be covered by the corresponding request line properly. This means that the pulse fault must have occurred already before a wave fault occurs at a switch.

Chapter 5

Pulse Fault Rate Analysis

Path delay of a circuit with fabrication variations can be modeled by using an interval than a single value. Let x denote the average path delay of a partial path. Suppose there are n partial paths in a partial circuit, e.g., x_1, x_2...x_n. All of the partial paths are assumed to be well balanced in the AWP under investigation. Then, without loss of generality, x can be modeled by using the normal distribution with a mean μ and a standard deviation σ as follows.

$$x \sim N(\mu\sigma^2). \tag{5.1}$$

Let $\bar{x}$ be the sample mean, and s be the standard error, i.e. $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$ and $s = \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$.

Then μ and σ can be substituted with $\bar{x}$ and s , respectively, because $\bar{x}$ and s are unbiased estimator of μ and σ.

$$x \sim N(\bar{x}s^2). \tag{5.2}$$

Suppose $f(x)$ is the $p.d.f.$ (probability density function) of x as follows.

$$f(x) = \frac{1}{\sqrt{2\pi}s} e^{-\frac{(x-\bar{x})^2}{2s^2}} \tag{5.3}$$

Then the probability that some bits in the data wave proceed too fast and then overstep their associated request level (i.e., the probability of setup time pulse fault) or some bits in the data wave are too slow such that they lag behind the associated request level (i.e., the hold time pulse fault) is the probability that some bits are out of the range of the request level interval as shown in Fig.5.1 (where the shaded areas represent the probability of pulse fault).

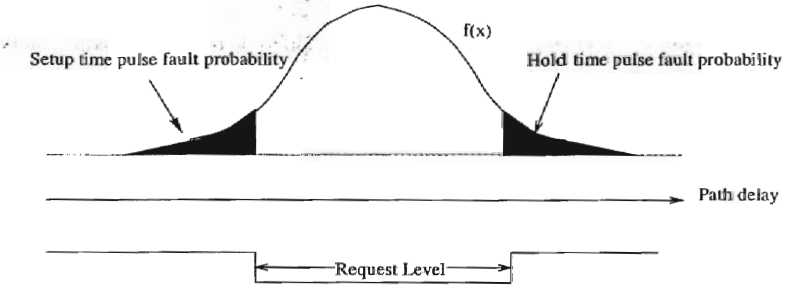

Figure 5.1: Pulse Fault Probability

As shown in in Fig.4.1, the request signal and all the data bits enter the circuit at the same time, and for proper operation the request signal should be slower than the slowest bit of the data wave and reach the switch after the slowest bit. Likewise, the previous request signal should reach the switch before the fastest bit of the associated data wave. The data skew (represented by $\triangle$ in Fig.5.2) is supposed to be covered by the associated request level properly. Therefore, the coverage of $\triangle$ by the request level, i.e., the relative position between the data wave and its associated low or high request level may influence the pulse fault rate to a great extent.

Lets α refer to the difference of propagation time between the slowest bit of the data wave and the request level as shown in Fig.5.2. The associated request signal's propagation delay (denoted as d_s) of the data wave can be expressed as $d_s = d_{max} + \alpha$. The propagation delay from the request signal start to propagation to the previous request signal reach the switch (denoted as d_p) is $d_p = d_{min} - (L - \alpha - \triangle)$. Thus, $L = d_s - d_p$.

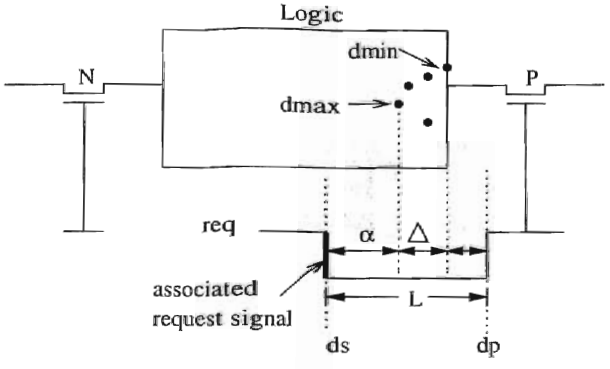

Figure 5.2: Relative position between the request signal and data wave

The pulse fault rate at switch i is then:

$$P_i = \int_{d_{min_i}-(L-\alpha_i-\triangle_i)}^{d_{max_i}+\alpha_i} \frac{1}{\sqrt{2\pi}s} e^{(-\frac{(x-\bar{x})^2}{2s^2})} dx \tag{5.4}$$

If we have n switches in the AWP and P_i is the pulse fault rate at each switch where $1 \leq i \leq n$, then the *total pulse fault rate* (P_{total}) of the AWP is as follows.

$$\begin{aligned} P_{total} &= 1 - \prod_{i=1}^{n}(1 - P_i) \\ &= 1 - \prod_{i=1}^{n}(1 - \int_{d_{min_i}-(L-\alpha_i-\triangle_i)}^{d_{max_i}+\alpha_i} \frac{1}{\sqrt{2\pi}s} e^{(-\frac{(x-\bar{x})^2}{2s^2})} d\bar{x}) \end{aligned} \tag{5.5}$$

The *fault coverage* (referred to as FC [14]) is the percentage of total possible pulse faults that can be detected by a testing process with a certain testability.

$$\begin{aligned} FC &= \frac{\Sigma(P_i \times T_i)}{\Sigma P_i} \\ &= \frac{\Sigma(\int_{d_{min_i}-(L-\alpha_i-\triangle_i)}^{d_{max_i}+\alpha_i} \frac{1}{\sqrt{2\pi}s} e^{(-\frac{(x-\bar{x})^2}{2s^2})} dx \times T_i)}{\Sigma(\int_{d_{min_i}-(L-\alpha_i-\triangle_i)}^{d_{max_i}+\alpha_i} \frac{1}{\sqrt{2\pi}s} e^{(-\frac{(x-\bar{x})^2}{2s^2})} dx)} \end{aligned} \tag{5.6}$$

where T_i is the testability at switch i.

The *yield* (referred to as Y [15]) is the ratio of the number of good products to the total number of products and can be expressed as follows.

$$\begin{aligned} Y &= 1 - P_{total} \\ &= \prod_{i=1}^{n}(1 - P_i) \\ &= \prod_{i=1}^{n}(1 - \int_{d_{min_i}-(L-\alpha_i-\triangle_i)}^{d_{max_i}+\alpha_i} \frac{1}{\sqrt{2\pi}s} e^{(-\frac{(x-\bar{x})^2}{2s^2})} dx) \end{aligned} \tag{5.7}$$

The *defect level* (referred to as DL [17]) is the ratio of products that are defective , but not detected during testing and shipped as expressed as follows.

$$\begin{aligned} DL &= 1 - Y^{(1-FC)} \\ &= 1 - (\prod_{i=1}^{n}(1 - \int_{d_{min_i}-(L-\alpha_i-\triangle_i)}^{d_{max_i}+\alpha_i} \frac{1}{\sqrt{2\pi}s} e^{(-\frac{(x-\bar{x})^2}{2s^2})} dx))^{(1-FC)} \end{aligned} \tag{5.8}$$

Based on the contributing factors for L, e.g., $\triangle$, $\triangle_{ps}$, T_s, T_h, d and $\triangle_{max}$, four possible P_i can be implemented to set up the relative position between data wave and the request level. The first possible P_i is to place the $\triangle$ within the request level just as shown in Fig.4.1, the fault rate at switch i is as follows.

$$P_i = \int_{d_{min_i} - T_{su} - d - \triangle_{ps} - (\triangle_{max} - \triangle_i)}^{d_{max_i} + T_h + d + \triangle_{ps}} \frac{1}{\sqrt{2\pi}s} e^{(-\frac{(x-\bar{x})^2}{2s^2})} dx \tag{5.9}$$

Instead of having d and $(\triangle_{max} - \triangle)$ on the right side of data skew, the second possible P_i is to put them to the left side of $\triangle$ as shown in Fig.5.3. Thus the pulse fault rate at switch i is as follows.

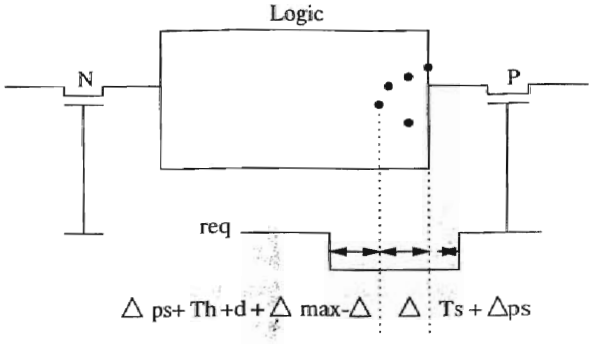

Figure 5.3: d and $(\triangle_{max} - \triangle)$ are at the left side of data skew

$$P_i = \int_{d_{min_i} - T_{su} - \triangle_{ps}}^{d_{max_i} + T_h + \triangle_{ps} + d + (\triangle_{max} - \triangle_i)} \frac{1}{\sqrt{2\pi}s} e^{(-\frac{(x-\bar{x})^2}{2s^2})} dx \tag{5.10}$$

The third possible P_i is to place $\triangle$ in the middle of the request level as shown in Fig.5.4. Thus the pulse fault rate at switch i is as follows.

$$P_i = \int_{d_{min_i} - \frac{L - \triangle_i}{2}}^{d_{max_i} + \frac{L - \triangle_i}{2}} \frac{1}{\sqrt{2\pi}s} e^{(-\frac{(x-\bar{x})^2}{2s^2})} dx \tag{5.11}$$

The fourth possible P_i is to divide d and $(\triangle_{max} - \triangle)$ into two parts equally and let them stride at the two sides of $\triangle$ as shown in Fig.5.5. Thus, the pulse fault rate is as follows.

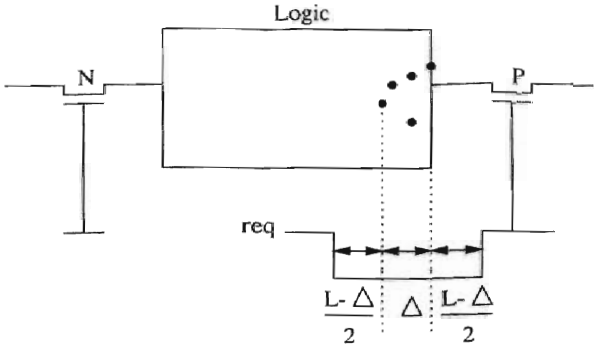

Figure 5.4: Data skew is in the middle of request level

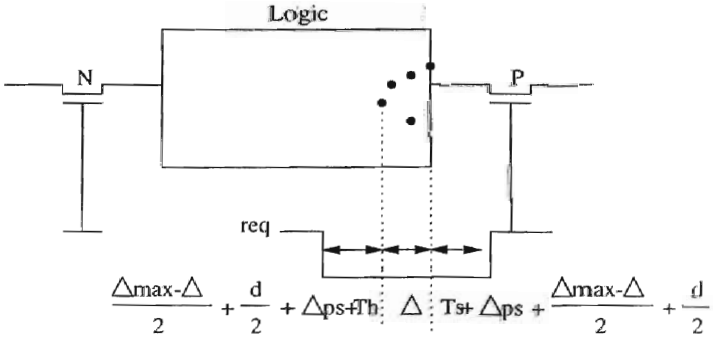

Figure 5.5: Divided d and Δ into two parts equally

Table 5.1: Parameter Values Simulated

-	# path	testability	path 1	path 2	path 3	path 4	path 5	path 6
$S_0 - S_2$	4	0.78	27	24	26	22	-	-
$S_1 - S_3$	5	0.89	60	58	64	62	63	-
$S_2 - S_4$	6	0.95	37	34	33	35	32	30
$S_3 - S_5$	5	0.86	72	75	78	70	77	-
$S_4 - S_6$	6	0.91	40	43	42	44	41	46

$$P_i = \int_{d_{min_i} - T_{su} - \frac{d}{2} - \triangle_{ps} - \frac{\triangle_{max} - \triangle_i}{2}}^{d_{max_i} + T_h + \frac{d}{2} + \triangle_{ps} + \frac{\triangle_{max} - \triangle_i}{2}} \frac{1}{\sqrt{2\pi}s} e^{(-\frac{(x-\bar{x})^2}{2s^2})} dx \tag{5.12}$$

For simulation, suppose there is a circuit with 7 switches (e.g., S_0 - S_6 as provided in Table 5.1). Also, suppose that the number of paths within any partial circuit (the # path in the table), and their path delays are known. Lastly, assume that the testability (i.e., the probability to detect the pulse faults) at each switch is known as provided in Table 5.1 with the unit ns.

From Table 5.1, $\triangle_{max} = 8\ ns$ can be calculated. It is also assumed that $d = 1\ ns$, $T_{su} = 0.6\ ns$, $T_h = 0.2\ ns$, and $\triangle_{ps} = 0.1\ ns$. Then $L_{min} = \triangle_{max} + T_{su} + T_h + d + 2\triangle_{ps} = 10\ ns$.

The final simulation results are shown in Table 6.1, 6.2, 6.3 and 6.4. The length of the request level is set to L_{min} first, i.e., 10 ns, and then to the longer values up to 15 ns. It is shown that the total pulse fault rate at 15 ns becomes very low as shown in Fig.6.1. For this simulation, SAS (Statistical Analysis Software) was used to compute the standard deviations of those path delay samples in Table 5.1. Then, from the $c.d.f.$ (cumulative density function) of Equation 5.4, the pulse fault probability were calculated.

By using Equation 5.5, the total pulse fault rate can be calculated. Then, the

yield, the fault coverage and the defect level can be calculated as well. The simulation results are shown in Fig.6.1, Fig.6.2, Fig.6.3 and Fig.6.4.

In summary, the simulation results have demonstrated that the proposed pulse fault in association with the request level length has great effect on the integral pulse fault rate, as shown in Fig.6.1. All four possible P_i drop to less than 5% when the request level length extends from L_{min} (i.e., 10 ns) up to $15ns$. An increased request level length may significantly affect the pulse fault rate to decrease, while in consequence the yield is enhanced and defect level is declined significantly. The fault coverage is evaluated with an arbitrary sample testability distribution as shown in Table 6.1. Comparing the four strategies for P_i, it can also be observed that strategies 1 and 2 demonstrate similar performance, and strategies 3 and 4 demonstrate similar performance, in which the last two strategies demonstrate less pulse fault rate. Thus strategies 3 and 4 can be suggested to be more suitable for implementation in practice.

Chapter 6

Conclusion

This paper has proposed the pulse fault model for the two-phase clockless asynchronous wave pipeline, and its fault rate has been statistically yet practically derived. The pulse fault model has been throughly identified as the unique fault of the clockless wave pipeline in comparison with wave and wave delay faults of conventional wave pipelines. The pulse fault rate is statistically yet practically modeled, and extensively evaluated with respect to various design parameters, such as yield, fault coverage, defect-level, and request level length. The simulation results have revealed that the proposed pulse fault in association with the request level length has great effect on the integral pulse fault rate. Also,it has been demonstrated that an increased request level length may significantly affect the pulse fault rate to decrease, while in consequence the yield is enhanced and defect level is decreased significantly. it has been observed and analyzed that placing $\triangle$ in the middle of L or divide d and $(\triangle_{max} - \triangle)$ equally and let them stride at both sides of $\triangle$ can reduce pulse fault rate drastically. In conclusion, the proposed modeling and analysis have provided a

theoretical yet practical guideline for a feasible clockless asynchronous wave pipeline design strategy for reliability.

Bibliography

[1] Burleson.W.P , Ciesielski.M, Klass.F, Liu.W, *Wave-Pipling: A Tutorial and Research Survey* IEEE Transactions on very large scale integrate (VLSI)system, VOL.6, NO. 3, September 1998.

[2] Cotton. L. W, *Maximum-rate pipeline system* AFIPS Proceedings Spring Joint Computer Conference, vol. 34, Montvale, NJ: AFIPS Press, pp.581-586, May (1969)

[3] Fishburn.J, *Clock Skew Optimization* IEEE Trans.Comput., 1990

[4] Gray.T, Hughes.T, Arora.S Liu.W, Cavin.R, *theoretical and Practical Issues in CMOS Wave Pipelining* VLSI Design 1991, pp, 9.2.1-9.2.5

[5] Gray.C, Liu.W.T, Cavin.K.R, *Timing Constraints for Wave-Pipelined Systems* IEEE Transactions on computer-aided design of integrated circuits and systems, vol., 13, NO, 8, August 1994.

[6] Heald.R, Shin.K, Reddy.V, Lynch.W, $64-Kbyte-Sum-Addressed-memory$ *Cache with* $1.6-ns$ *Cycle and* $2.6-ns$ *Latency* IEEE Journal of Solid-State Circuits, vol. 33, no. 11, PP. 1682-1689, Nov. (1998)

[7] Hauck.O, Garg.M, Huss.A.S, *Two-phase asynchronous wave-pipelines and their application to a 2D-DCT* Advanced Research in Asynchronous Circuits and Systems, 1999. Proceedings., International Symposium on , 1999 page(s): 219-228

[8] Hauck.O, Huss.A.S, *Asynchronous wave pipelines for high throughput data paths* Electronics, Circuits and Systems, 1998 IEEE International Conference on , Volume: 1 , 1998 Page(s): 283 -286 vol.1

[9] Hermanns.S, Huss.S.A, *Embedding of Asynchronous Wave Pipelines into Synchronous Data Processing* SAME Conference 2001, 14., 15. November 2001, Sophia Antipolis, Valbonne, France

[10] Lein.H.W, Burleson.W, *Wave domino logic: theory and applications* Proc.Int.Symp.Circuits Syst., 1992

[11] Shyur.J.C ,Chen. H.P, Parng.T.M, *On testing wave pipelined circuits* Proceedings of the 31st annual conference on Design automation conference, ACM Press, Series-Proceeding-Article, 1994, Page(s): 370-374

[12] Sakallah.A.K, Mudge.N.T, Burks.T.M, Davidson.S.E, *Synchronization of pipeline* IEEE Trans.Computer-Aided Design, vol.12, 1993.

[13] Shenoy.V.N, Brayton.K.R, Sangiovanni-Vincentelli.L.A, *minimum padding to satisfy short path constraintts* proc. Int. Wrokshop logic Synthesis, 1993

[14] Wu.W.C, Lee.C.Len, *A probabilistic testability measure for delay faults* Proceedings of the 28th conference on ACM/IEEE design automation conference, p.440-445, June 17-22, 1991

[15] Venkataraman.A , Koren.I *Determination of yield bounds prior to routing* Proc. of the 1999 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp. 4-13, November 1999.

[16] Wong.D, Micheli.G.De , Flynn.M, *Inserting active delay elements to achieve wave pipelining* Proc.Int.Conf.Computer-Aided Design, Nov. 1989, pp. 270-273

[17] Williams.T.W, Brawn.N.C, *Defect level as a function of fault coverage* IEEE Trans. on computers, C-30, 1981.

Table 6.1: Simulation Results : strategy 1

L	total fault rate	yield	fault coverage	defect level
10	0.367494451	0.632505549	0.871709406	0.057072231
10.2	0.352666198	0.647333802	0.871896485	0.054187892
10.4	0.325746021	0.674253979	0.871639409	0.049334605
10.6	0.30445439	0.69554561	0.871477465	0.045589325
10.8	0.284201208	0.715798792	0.871348472	0.042103392
11	0.264713337	0.735286663	0.871195594	0.038832599
11.2	0.246210244	0.753789756	0.871071585	0.035784593
11.4	0.228198915	0.771801085	0.870848159	0.032900601
11.6	0.211871715	0.788128285	0.870780798	0.030297901
11.8	0.19603997	0.80396003	0.870637117	0.027833042
12	0.181111714	0.818888286	0.870492922	0.025544572
12.4	0.153968671	0.846031329	0.870197947	0.021468949
12.8	0.130207994	0.869792006	0.869896313	0.017985904
13.6	0.091489187	0.908510813	0.869258668	0.01246608
15	0.047429059	0.952570941	0.868073019	0.006389921

Table 6.2: Simulation Results : strategy 2

L	total fault rate	yield	fault coverage	defect level
10	0.347888027	0.652111973	0.864479146	0.056293856
10.2	0.326300201	0.673699799	0.864299125	0.052186822
10.4	0.305504602	0.694495398	0.864137228	0.048324778
10.6	0.285611382	0.714388618	0.86397734	0.044717579
10.8	0.266576551	0.733423449	0.863824971	0.041339828
11	0.275411848	0.724588152	0.868943331	0.04134129
11.2	0.231258524	0.768741476	0.863525426	0.035256376
11.4	0.214940745	0.785059255	0.863350419	0.032527872
11.6	0.199231543	0.800768457	0.863322501	0.029911017
11.8	0.184627363	0.815372637	0.863202038	0.027535627
12	0.170815811	0.829184189	0.863096544	0.025317783
12.4	0.145645552	0.854354448	0.862902869	0.02134915
12.8	0.123670156	0.876329844	0.862762008	0.017954032
13.6	0.08799714	0.91200286	0.862543906	0.012581559
15	0.046916725	0.953083275	0.862305445	0.006594794

Table 6.3: Simulation Results : strategy 3

L	total fault rate	yield	fault coverage	defect level
10	0.242881202	0.757118798	0.884499981	0.031625281
10.2	0.225890719	0.774109281	0.884699873	0.029090192
10.4	0.209854335	0.790145665	0.88488168	0.02675043
10.6	0.194734407	0.805265593	0.885073572	0.024583895
10.8	0.1806017	0.8193983	0.885261675	0.022594974
11	0.167284435	0.832715565	0.885447901	0.020751921
11.2	0.154742773	0.845257227	0.885635815	0.019042608
11.4	0.143148225	0.856851775	0.885821934	0.017484744
11.6	0.132308882	0.867691118	0.885998982	0.016048789
11.8	0.122179681	0.877820319	0.886179765	0.01472284
12	0.112750677	0.887249323	0.886354305	0.01350335
12.4	0.095814679	0.904185321	0.886695884	0.011347226
12.8	0.080036914	0.919963086	0.886973733	0.009384535
13.6	0.058191685	0.941808315	0.887616937	0.006715111
15	0.031111724	0.968888276	0.888356046	0.003522397

Table 6.4: Simulation Results : strategy 4

L	total fault rate	yield	fault coverage	defect level
10	0.241383233	0.758616767	0.883601987	0.031644441
10.2	0.224389178	0.775610822	0.883804451	0.029094173
10.4	0.208302282	0.791697718	0.884009554	0.026728831
10.6	0.193235642	0.806764358	0.884210603	0.02455619
10.8	0.18081626	0.81918374	0.884470613	0.022778536
11	0.16576059	0.83423941	0.884624314	0.020692996
11.2	0.153323796	0.846676204	0.884830951	0.018985839
11.4	0.141694096	0.858305904	0.88503183	0.017413136
11.6	0.13083329	0.86916671	0.88523444	0.015963673
11.8	0.120717027	0.879282973	0.885437317	0.014630241
12	0.11280932	0.88719068	0.88569498	0.013588609
12.4	0.094483412	0.905516588	0.886027256	0.011248021
12.8	0.08006562	0.91993438	0.886408548	0.009434751
13.6	0.057136576	0.942863424	0.887132926	0.006618404
15	0.031118336	0.968881664	0.888152685	0.003529563

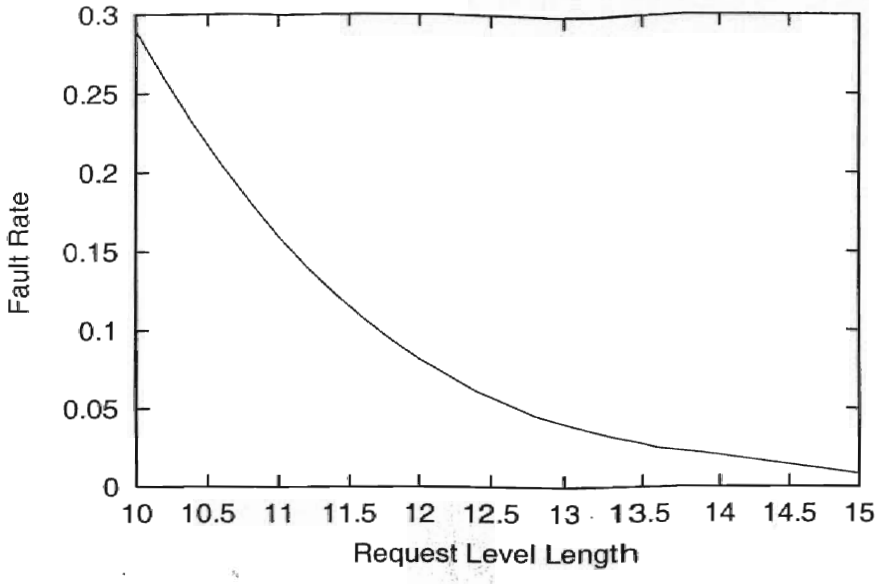

Figure 6.1: Length of Request Level vs. Pulse Fault Rate

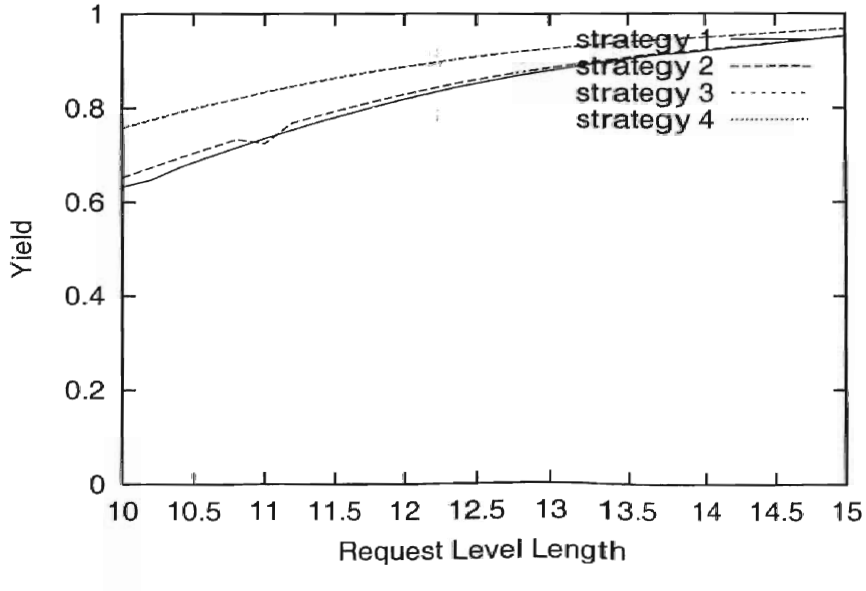

Figure 6.2: Length of Request Level vs. Yield

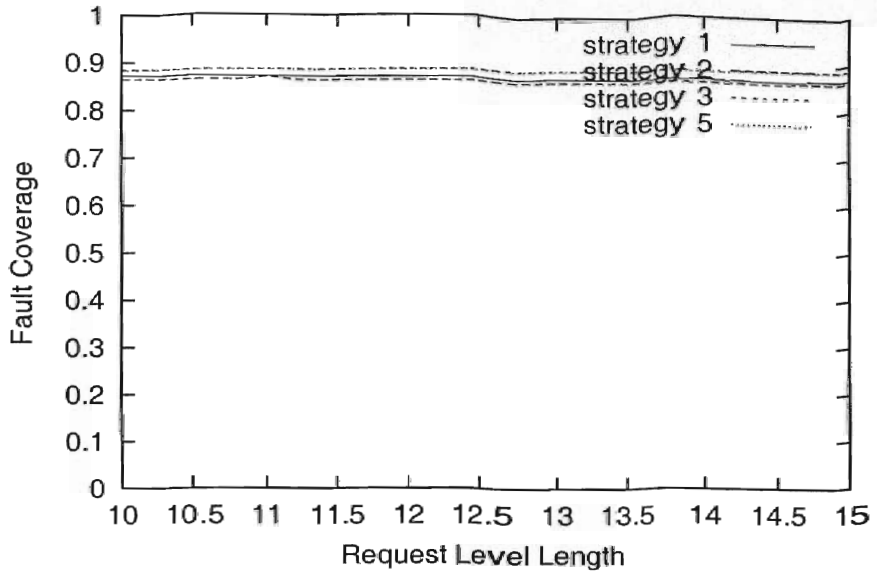

Figure 6.3: Length of Request Level vs. Fault Coverage

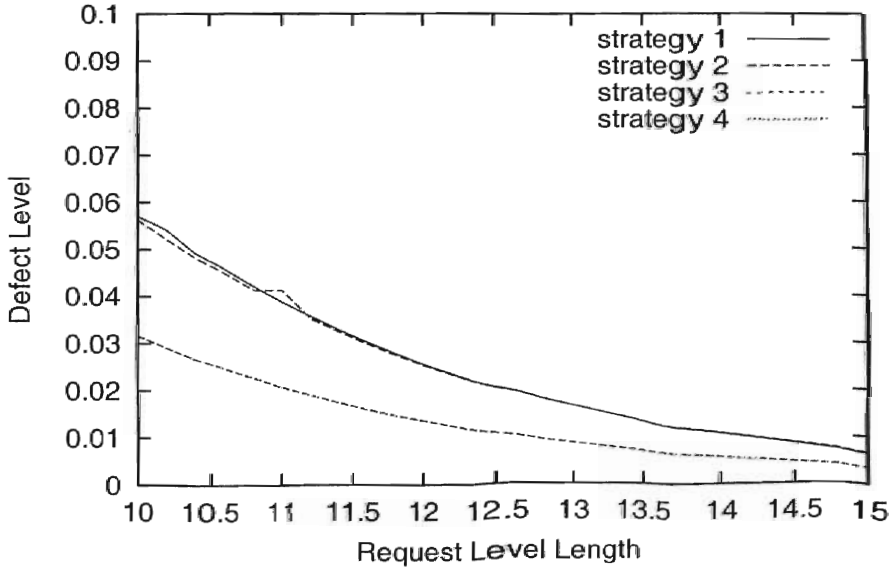

Figure 6.4: Length of Request Level vs. Defect Level

VITA

Tao Feng

Candidate for the Degree of

Master of Science

Thesis: YIELD MODELING AND ANALYSIS OF A CLOCKLESS ASYNCHRONOUS WAVE PEPELINE WITH PULSE FAULTS

Major Field: Computer Science

Biographical:

Personal Data: Born in Qingzhou, Shandong, P.R.China, October 29, 1974, son of Guangsheng Feng and Sumei Wang.

Education: Graduated from Qingzhou 1st School, Qingzhou, Shandong, P. R. China in July 1992. Received Bachelor of Science degree in Agricultural Chemistry from Northwest Agricultural University, Xi'an, Shanxi, P. R. China in July 1996. Completed the requirements for the Master of Science degree in Computer Science at the Computer Science Department of Oklahoma State University in May 2003.

Experience: Agricultural Engineer in Chinese Tobacco Company from July, 1996 to August, 2000.