Technical Report

UCAM-CL-TR-738
ISSN 1476-2986

Number 738

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

High precision timing using self-timed circuits

Scott Fairbanks

January 2009

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted September 2004 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Gonville and Caius College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Abstract

Constraining the events that demarcate periods on a VLSI chip to precise instances of time is the task undertaken in this thesis. High speed sampling and clock distribution are two example applications. Foundational to my approach is the use of self-timed data control circuits.

Specially designed self-timed control circuits deliver high frequency timing signals with precise phase relationships. The frequency and the phase relationships are controlled by varying the number of self-timed control stages and the number of tokens they control.

The self-timed control circuits are constructed with simple digital logic gates. The digital logic gates respond to a range of analog values with a continuum of precise and controlled delays. The control circuits implement their functionality efficiently. This allows the gates to drive long wires and distribute the timing signals over a large area. Also gate delays are short and few, allowing for high frequencies.

The self-timed control circuits implement the functionality of a FIFO that is then closed into a ring. Timing tokens ripple through the rings. The FIFO stages use digital handshaking protocols to pass the timing tokens between the stages. The FIFO control stage detects the phase between the handshake signals on its inputs and produces a signal that is sent back to the producers with a delay that is a function of the phase relationship of the input signals.

The methods described are not bound to the same process and systematic skew limitations of existing methods. For a certain power budget, timing signals are generated and distributed with significantly less power with the approaches to be presented than with conventional methods.

Acknowledgements

My thoughts are ordered and informed by the person of Jesus. Most of all I thank and praise him. He takes my small dreams and shortsighted goals and and returns bigger dreams and endurable accomplishments.

It simultaneously seems most appropriate and inappropriate to thank my wife Elisa. Most appropriate because more-so than anyone, I couldn't have done it without her. Most inappropriate because I think we did it together and as team-mates we celebrate with each other, not thank each other. But there is plenty of celebration and gratitude to go around. Elisa, We did it!! Thank you.

Ivan Sutherland has taken a continued active interest in my personal development. He has my deepest gratitude.

Paul Cunningham leads by inspiration. Thanks Paul for kicking my gracious butt into gear.

My best memories of Cambridge will be the many hours I was able to spend with Reya and Atira. They have given me much joy. Also, I thank them for keeping me from finishing my work too early.

Simon Moore took me under his wing, gave me sound advice, and edited my most errant thoughts before I could inflict them on others. thank you.

Bob Mullins provided hours of thought provoking and just plain provoking conversation on everything from transistors to trans-atlantic relations. When I'm no longer able to accomplish meaningful work I'd be very eager to share an office with him again.

Bill, Ian, Jo, John, Jon, Wes, Mark, Ann, and Yaeko and belatedly Charlie, thanks for the education I couldn't have bought.

Bernard Bell has challenged and stimulated my mind as much as any other.

This list is scandalously brief and incomplete. I am indebted to very many people for their friendship, advice, and tutelage. I wish I could thank them all.

Contents

List of Figures

Chapter 1

Structuring Time

1.1 Basis

Computation is performed using some ordering which advances over time. An electronic computation system needs to provide mechanisms to control the ordering of data as it is moved between memory systems and arithmetic units. This ordering is often imposed by a timing standard which must be communicated. In VLSI, time is communicated by copying the events through various channels to the elements that use the timing signals. The variability in the transmission properties of the channels results in the copied events being located at different instances in time at the dispersed elements. The further an event is communicated from its source, the greater the event can be dispersed in time at a destination. This dispersion in arrival time impedes computation because if there is uncertainty in the timing standard, the computation can't safely progress.

Events tightly bound to certain instances of time are primarily used in two ways in VLSI applications. High precision timing is either used to locate other events in time or dictate in time when other events can occur. These two applications are commonly referred to as sampling and clocking.

Sampling Placing events at *many equi-distant instances* on a *single node* allows another event to be located in time. The many events are like mile posts, or better yet, yard-sticks, in time. Clock recovery circuits and oscilloscopes sample a node at many precise instances in time to locate another signal in temporal space. Analog to digital converters sample a node to determine how some signal changes with respect to time and to describe that relationship digitally.

Clocking Events that are precisely placed at a *single instant* in time but at *many nodes* facilitate communication. Communication is the exchange of information between different

elements in different locations. Communication requires some protocol to signal when to send and when to receive information. In microprocessors this signaling is typically performed with a clock. The domain of the clock is the area containing state-holding elements whose communication is dictated by that clock. Structures within the same clock domain communicate with each other with minimal complexity. Note that clock domains can overlap.

Sampling and clocking are both limited by timing imprecision. Timing imprecision limits the performance of many applications. Sampling imprecision limits the signal frequency that can be observed by a scope. It also limits the amount of information that can be translated from the analog to the digital domain. Clocking imprecision due to the topology of the clock distribution apparatus is called skew. Skew limits the time available in each clock cycle to perform calculations.

1.2 Task

This thesis seeks methods for precisely placing events at both one and many specific locations in space and at specific instances in time.

A high precision timing solution should consider three factors.

Power –Sharper timing resolutions can be attained by applying more power. Events are generated quicker if the supply voltage is raised, giving greater timing resolution, Skew is reduced if separate nodes are shorted through low resistance metals. But the capacitance of that metal must be charged and discharged, requiring power. A new high precision timing solution must increase precision without increasing the power.

Complexity – Modern microprocessors are extremely complex circuits having many metal layers and billions of transistors. The clock distribution network is but one of many systems whose needs and resources must be balanced against the others to achieve high performance. Exotic timing methods, discussed in the next chapter, have been pioneered to address the looming bottleneck imposed by the power-precision trade-off. Yet none of these methods, as far as I know, have gained traction. Primarily they are analog solutions. Digital solutions such as a balanced clock tree are simple and intuitive. When building complex billion transistor circuits, simplicity carries much currency. A new high precision timing solution must be simple which means preferably digital.

Elegance –While balanced clock trees are digital and simple, their topology is incongruous with the rectangular and grid-like layout of the functional units and power distribution system. This makes them difficult to route and balance. It is difficult to locate and balance current return paths and charge coupling to neighboring signals. A new high precision timing

method should share the same rectangular topology as the functional units and the power distribution.

My task is to develop a high precision timing method that is both simple, efficient, and topologically congruent with other structures in a microprocessor.

1.3 Method

This thesis overcomes the challenges encountered in providing a highly precise timing reference and communicating that timing reference. It uses self-timed circuits to generate and distribute high precision timing signals.

The foundation of my method is self-timed or ripple FIFO control. Ripple FIFO control has been used on many test chips to generate and deliver high frequency timing signals (6; 21; 33; 44). The timing signals in self-timed FIFOs are generated in response to other timing signals rather than copied. The FIFO control is closed such that the FIFO forms a seamless ring. Tokens in the ring exert a force on each other causing them to spread evenly around the circumference of the ring. As the tokens ripple around the ring in the evenly spaced pattern they cause the handshaking signals between the stages to assume dependable and predictable timing relationships with each other.

This method relies on digital logic, closed loop feedback, and physics.

Digital Logic Digital logic is used by the handshaking protocols to maintain the ordering of timing tokens.

Feedback The FIFO stage acts as an actuator in a closed loop system. The phase of its inputs are detected and the delay of the FIFO stage is adjusted accordingly.

Physics The delay varies with pico-second resolution by the physics of the novel gates designed for this application.

These three effects act in concert to ensure that tokens spread throughout a ripple FIFO such that the handshake signals between FIFO control stages in one location operate in a predictable timing relationship with respect to a FIFO control stage in another location.

This timing method uses asynchronous protocols and ripple FIFO control circuits that were pioneered and rigorously refined to communicate efficiently. The handshaking protocols were intended to order individual data words in asynchronous pipelines. My application, however, is synchronous and generates timing signals independent of the data. The assumptions of my application allow me to implement the FIFO control logic even more efficiently.

Although my technique relies upon closed loop feedback and the detection of analog voltages my technique can be implemented by digital circuit designers using conventional digital design flows.

Balanced clock trees require that each leaf is the same electrical distance from a shared source. The only spatial constraints on the FIFOs in this application are that each FIFO stage is the same electrical distance from its two neighbors and that the FIFO control circuits make a ring. These constraints are easily reconciled with the existing rectangular and grid-like topology of the functional units and power-distribution grid.

This thesis describes how to accurately place events at specific locations in space and specific instances in time on a VLSI chip. The method is efficient with its use of power, it is easily implemented with available digital design flow techniques, and integrates nicely with the geometry of other circuits.

1.4 Outline

My description continues in six more chapters.

The second chapter surveys existing methods for placing events at specific locations in space and time in CMOS VLSI.

The third chapter introduces various forms of self-timed FIFOs and explains their operation. Previous works investigating the movement of data tokens in self-timed FIFOs is surveyed. Their contributions to this thesis are identified.

The fourth chapter applies the method discussed in Chapter 3 to address the tasks that were identified in this chapter using FIFO control that communicates using two-phase `Micropipeline` signaling.

The fifth chapter discusses high precision timing techniques using a FIFO control circuitry that employs a handshaking signal that communicates by pulses (20).

The sixth chapter extends the ideas presented in the fifth chapter to the specific problem of global clock distribution.

The seventh chapter identifies areas of future work related to this work and provides some concluding remarks.

Chapter 2

Existing methods

2.1 Introduction

This chapter surveys the existing methods for generating and distributing high precision timing signals.

The first section introduces techniques used to generate signals with a precision of less than a single fan-out-of-one inverter delay (FO1). It discusses their limitations, costs and provides published results reported from test chips. I discuss the design of a circuit that generates 30 phases of a signal with a period of 8 fan-out-of-4 gate delays to reveal the constraints and tradeoffs faced in the most prevalent method for generating sub-gate delay timing precision.

The second section surveys the problem and solutions of providing signals of the same frequency and phase at many locations on a large piece of silicon, clock distribution in other words. I explore the fundamental limitations to the most widespread method for tackling clock distribution, which is amplifying the signal using inverters through many equivalent paths branching out from a frequency source located at a single location. I introduce and briefly discuss alternatives proposed to get around the fundamental limitations of this strategy.

2.2 Terms and definitions

Timing uncertainty is contributed from many sources. The total timing uncertainty is the sum of the uncertainty provided by imprecision in fabrication, environmental noise, switching noise, and the systematic skew contributed from an unbalanced design.

Skew is defined as the spatial variation of a timing signal as distributed through a system (24). I distinguish between two types of skew in this thesis. The first type of skew is process skew. Process skew is caused by inaccuracies in the fabrication of a circuit in silicon. The

second type of skew is systematic skew. Systematic skew is the spatial variation in a timing signal arising from limitations in the design.

A metric I frequently use throughout this thesis is the FO4 delay (39). This refers to the delay of an inverter driving four identical copies of itself. While the delay of an inverter with a fan out of four is process dependent, the FO4 delay is relatively process independent. An inverter driving four copies of itself incurs a delay of 75ps in the 180nm process used for simulations in this thesis. That value is highly dependent on the process used. However the delay of a NAND gate driving four copies of itself is about 4/3 of a FO4 delay. This value is fairly accurate, independent of process.

Another metric I employ is the electrical effort, **EE** (39). The **EE** is a design choice. **EE** is defined as the the ratio of the total capacitance driven by a gate divided by the capacitance contributed from the inputs to a gate. An inverter driving two copies of itself or an inverter that drives a single inverter that is twice as big has an **EE** of two.

2.3 Sub-gate delay timing precision

2.3.1 Background

2.3.1.1 Applications

High precision timing finds application in analog to digital conversion, clock recovery, on-chip sub-sampling voltage probe circuits, and high speed serial links. A/D converters and scopes require a large number of samples to reconstruct high frequency signals to either observe the signals or extract information from them.

Clock recovery circuits and serial links require that the voltage transitions on a data line are located in the middle of the data eye of the receiving circuitry to maximize noise immunity, and attainable communication frequencies. High precision timing is used to locate the relative position between the data eye and the incoming data. A feedback circuit is then used to move the data eye to the proper temporal position relative to the incoming data.

Traditionally the means to achieving high precision is to use technologies with a faster intrinsic device speed such as GaAs. Investment in further integrating CMOS is great compared to GaAs and bipolar. Therefore, integrating high precision timing circuitry in silicon is economically attractive. It is the dominant technology, CMOS circuits are fabricated in much greater volume than other technologies.

2.3.1.2 Simple solutions

The simplest solution to precisely resolving time on a single node is by generating a signal of the highest frequency on that node. Conventionally, the highest frequency attainable by digital logic is that produced by a ring of three unloaded inverters. A ring of three unloaded inverters beats out a period equivalent to about 2 FO4 delays.

The ring of three inverters has three nodes. Each inverter in the ring has the same gate delay. A transition on one node in the ring causes a transition on the next node in the ring. We know that each transition is separated by 60° in phase from the transition that caused it because the ring is a closed loop, six transitions are necessary for a single period, and each gate has the same delay. Therefore each event is equally spaced. If we sample a probed node with each of these phases in parallel, then the maximum sampling frequency triples. This technique brings the timing resolution to that of a single unloaded inverter delay.

The precision limit is one FO1 inverter delay using digital logic without further techniques. More phases exist in a ring with more gates but the frequency slows proportionally.

2.4 Existing solutions

Two general techniques exist for on-chip generation of high frequency signals in many phases. One solution, phase interpolation, is open loop and results in systematic skew (36; 42). The other solution, arrayed oscillators (18), is closed loop. Systematic skew is eradicated but it has other limitations. The open loop solution is discussed first. Here skew is the difference between the expected and actual instant of an event in reference to another ideal event that is precisely located in time.

2.4.1 Open loop

Phase interpolation, or phase mixing, is the most common method for achieving sub-gate delay timing precision. Interpolation works by shorting the output of two amplifiers that are being driven by signals of different phase. The result is an averaging of the two outputs.

Figure 2.1 shows a Delay Locked Loop. The DLL sends a reference signal through a string of differential amplifiers. The phase of the signal at the output of the amplifier string is compared with the input signal. The delay of the amplifiers is adjusted according to the control voltage produced by the phase comparator until the input and the output are in a desired phase relationship.

This delay locked loop (DLL) yields ten separate phases, one phase at the output of each amplifier in the loop and its complement. Figure 2.2 shows interpolation being used to double

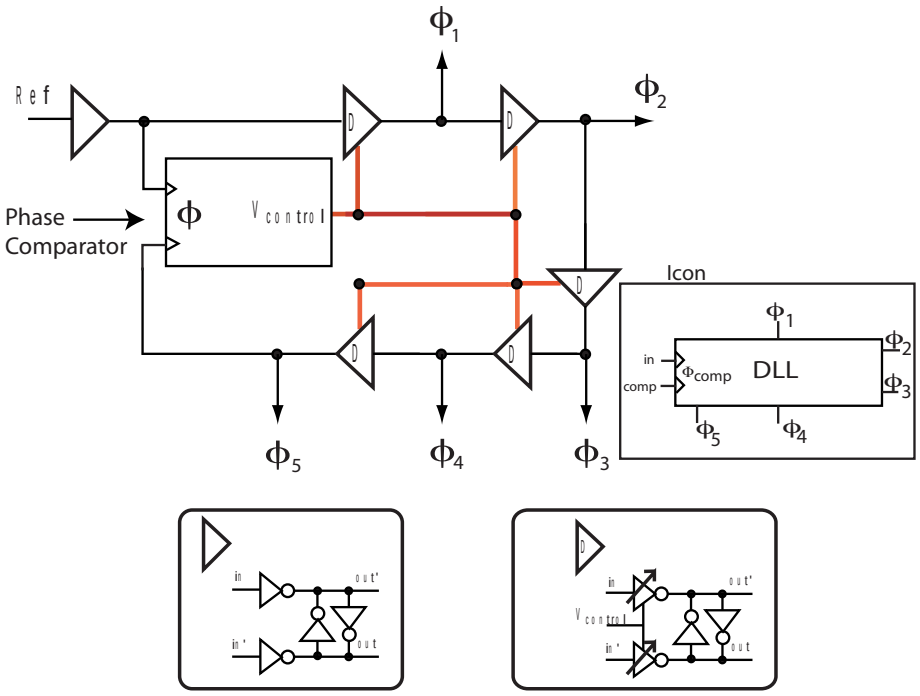


Figure 2.1: Five stage Delay Locked Loop

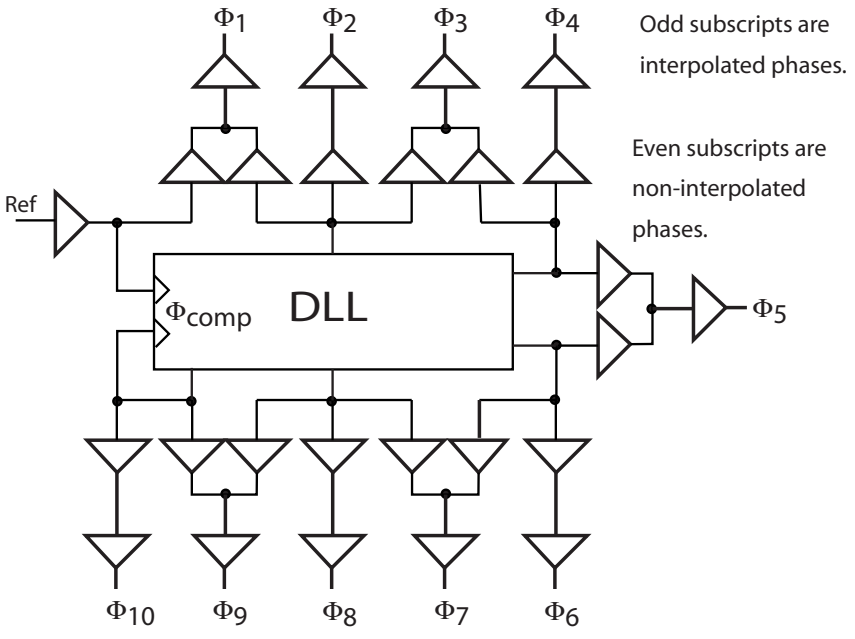


Figure 2.2: Doubling phases through interpolation

the number of available phases and double the timing resolution. This example shows 50/50 interpolation. The new phase is located midway between the buffered versions of the two phases that were mixed to create it. Other phase relationships are derived by varying the amount of current contributed from the original phases.

Two types of nodes exist after the first set of amplifiers coming off the amplifiers in the DLL: interpolated nodes and plain nodes. Ideally the phase of the interpolated nodes are located midway between the phases of the plain nodes on either side. But in reality they don't compare. The interpolated nodes have different slew rates than the events on the plain nodes. These aren't the same signal at different phases, they are different signals which makes talking about phase less meaningful.

The two amplifiers whose outputs are interpolated work against each other at the beginning of the transition on the interpolated node and they work together for the rest of the transition. The voltage on an interpolated node might cross some arbitrary voltage at a time that would place its phase in a correct relationship with a plain node but the different transition times will cause different delays in a following stage. Another stage of amplification will make the slew rates approximately but not exactly equal.

The sizes of the two interpolating amplifiers need to be adjusted until the interpolated transitions are equidistant from the non-interpolated transitions on either side. Some degree of systematic skew exists because of the different shapes of the signal on the interpolated and plain nodes. This can be mitigated but not eradicated with careful transistor sizing.

After one stage of interpolation the number of phases doubled from ten to twenty. To achieve greater precision another stage of interpolation can be performed to double again the number of phases. An arbitrary number of levels of interpolation can be performed. Interpolation ceases to be effective for timing precision when the timing uncertainty from transistor mismatch, systematic skew, and noise exceeds half the timing precision. If, for example, the phases are spaced by 50ps by design, it is useless to interpolate any further if the timing uncertainty is 25ps or greater.

2.4.1.1 An interpolation experiment

Figure 2.3 is a circuit constructed to divide an $8\times$FO4 gate delay clock period (600ps) into 30 phases that are approximately evenly spaced using interpolation. The transistor sizes are shown on the figure. I constrained the choices of transistor sizing to the discrete choices available in a 180nm process.

The last stage of buffers, labeled E in the figure, provide an equivalent drive as circuits described later in Section 4.7.1. The amplifiers labeled A are sized to amplify the 8 FO4 gate delay period. The sizes of amplifiers labeled B, C, and D were calibrated until the systematic

skew was less than a pico-second from the 20ps phase offsets specified. The granularity of available transistor sizes prohibited the systematic skew from further reduction.

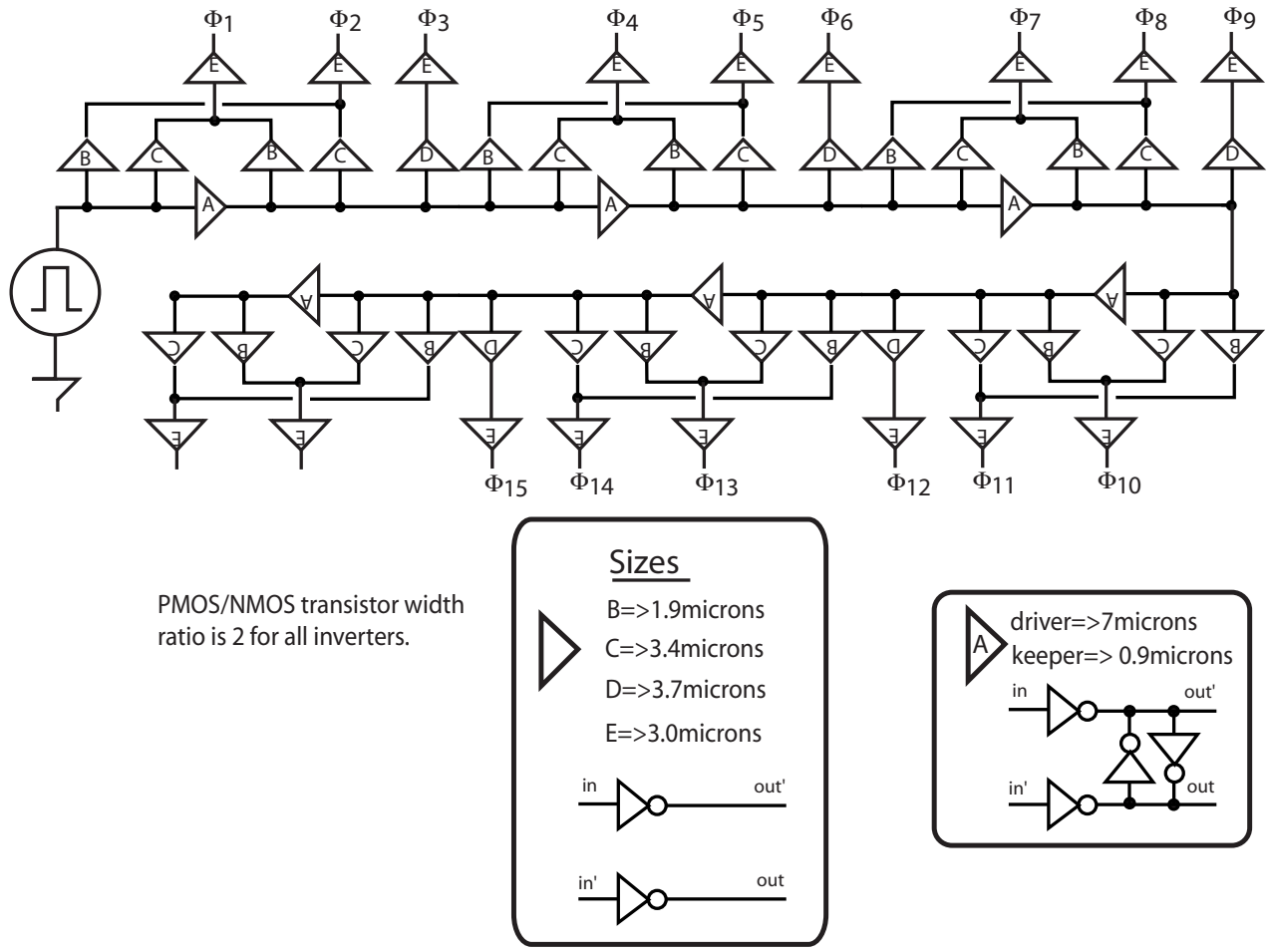


Figure 2.3: Interpolating for 20 phases of 400ps period signal

Transistor mismatch gives rise to skew. I model the delay through an inverter using a Normal distribution as described by Equation 2.1.

$$f(\chi) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{(\chi-\mu)^2}{2\sigma^2}\right) \tag{2.1}$$

The standard deviation, σ, for the inverter delay is linear with **EE**. The relationship between the standard deviation for timing uncertainty with respect to the **EE** of the stage is plotted in Figure 2.4. The data for the plot was obtained by performing a Monte Carlo simulation in Hspice using a transistor mismatch model derived from (27; 28) for a sample of $n = 30$.

The normal distribution for the delay of N series inverters is the convolution of their individual distributions. This results in another normal distribution with a standard deviation equivalent to the square root of the sum of the squares of the standard deviations in each stage. Equation 2.2 expresses this relationship. If each stage has the same standard deviation, then the standard deviation for the string of inverter stages reduces to Equation 2.3.

$$\sigma_N = \sqrt{\sigma_1^2 + \sigma_2^2 \ldots + \sigma_i^2} \tag{2.2}$$

$$\sigma_N = \sqrt{N} \times \sigma \tag{2.3}$$

The fanout through the amplifiers labeled A in the figure is just shy of three. Seven amplifiers is the longest path between the input to the circuit until one of the phase taps.

Figure 2.4 shows that an inverter with a **EE** of 3 has a standard deviation of 1.5ps of skew per stage. The last two stages have a **EE** of about one and therefore about a pico-second of standard deviation skew. The total expected skew from transistor mismatch is then approximately 3.6ps. The total worst case skew expectation is about 5ps including the systematic skew. A FO4 delay in the process used for this simulation is 75ps. This analysis suggests a limit in the timing resolution using interpolation for this configuration to equal 7% of a FO4 delay.

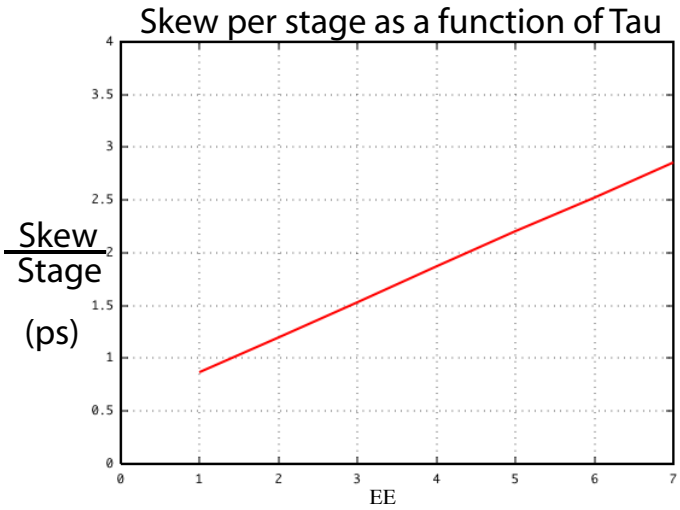


Figure 2.4: Skew vs **EE**

2.4.1.2 Reported interpolation studies

A timing uncertainty equal to a precision of 6.3% of a FO4 delay was reported by Sidiropoulos (36) using phase interpolation. Weinlader (42) achieved a resolution of 3% of a FO4 delay

using a 3 bit interpolation calibrater that allowed them to tune the amount of current from the interpolation amplifiers. This solution required post fabrication tuning and also needed circuitry capable of detecting errors to this precision.

Phase interpolation is a solution that doesn't track supply voltage. If the supply voltage is dropped for power concerns or raised to increase the frequency then the interpolated edge will likely not maintain the desired phase relationship with adjacent edges. Sidiropoulos' solution works at a single supply voltage. Weinlader's solution can be tuned to produce the desired phase relationships assuming that his calibrator has a large enough range.

2.4.2 Closed loop

Maneatis (18) uses a linear array of oscillators with a unique coupling arrangement that forces the outputs of ring oscillators to be uniformly offset in phase by a precise fraction of a buffer delay. Figure 2.5 sketches his technique. Similar to interpolation, each gate in the linear array is built from two shorted inverters. One of the inputs to the gate is taken from the output of the previous gate in the same oscillator. The other input is taken from the output of a gate in an adjacent oscillator. The closing connections in the array oscillator, which connect the array inputs on the top to the array outputs along the bottom, impose boundary constraints that constrain the arrays operation to specific consistent modes of oscillation.

Suppose all rings are oscillating in phase, so that the phase difference between the inputs to each gate is zero, then the corresponding nodes in each oscillator are in phase. This is not a very interesting case.

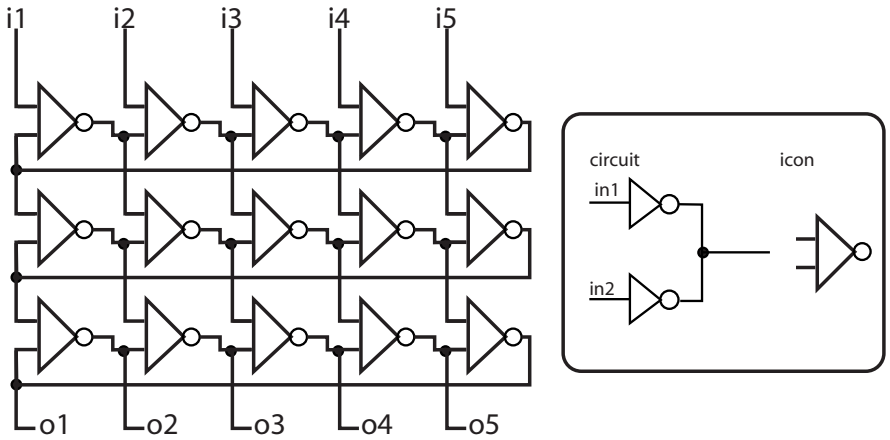


Figure 2.5: Maneatis' Array of Oscillators

Suppose instead the outputs along the bottom were connected to the inputs along the top that had a subscription two greater than itself. For example, o2 connects to i4, o3 connects to i0, and o4 connects to i1. These connections force the inputs along the top of the array to lag two buffer delays in phase behind the corresponding bottom array nodes. The phase difference across all corresponding ring nodes will uniformly span two buffer delays from top to bottom. The phase difference between corresponding nodes in adjacent rings is then two buffer delays divided by the number of rings.

Systematic skew is zero using this scheme. If transistor mismatch is neglected and noise is not modeled, all phases in Hspice have the exact relationship predicted, all signals also have the exact same slew rate. Nodes retain the same phase relationship even as the frequency is adjusted by either homogeneously varying the delays of the mixing gates or scaling the supply voltage.

This approach to generating high precision timing signals, while clever, is rarely employed. Two papers (41; 46) in the literature evaluated open-loop interpolation techniques and Maneatis' arrayed oscillators and chose interpolation for their high precision timing needs. I am not aware of anybody using the Maneatis array in a real application. A couple of weaknesses limit its use.

2.4.2.1 An arrayed oscillator experiment

The most parsimonious way to derive high precision for the smallest amount of hardware would be achieved by using an array of rings with three Maneatis mixer gates. Three gate delay rings provide the highest frequency which can then be divided into the number of phases desired. But the delay through the ring of gates is highly dependent on the number of phases being produced. The frequency of the rings drops substantially when the oscillators are arrayed. When the two inputs to the Maneatis gate are out of phase, then the two inverters form a short between supply and ground while the inputs differ. This current is wasted because it is not used to charge the node and results in greater gate delay.

For example a Maneatis array of three separate three inverter rings coupled together in the prescribed manner, oscillates 1.6 times slower than a ring of three inverters. An array of four separate three inverter rings oscillates at 2.25 times slower than a ring of three inverters. Even though the signal is available in more phases, the timing precision is reduced by a factor of $\frac{4}{3} \times \frac{1.6}{2.25} = 0.92$. With phase interpolation, the frequency of the phases is equivalent to the frequency of the signal being applied at the input of the interpolation gates independent of the amount of interpolation performed.

The array oscillator is naturally laid out as a two-dimensional array of gates with rings extending in one direction while being arrayed in the other. For the gate delays to be equal, the RC delay of the interconnect between oscillators must be equal. This is challenging because of the boundary condition where the bottom outputs must connect with the top inputs. One solution is

to interleave the buffers in both the horizontal and vertical direction, but geometrically complex and circuitous routes result.

2.4.2.2 Reported results from arrayed oscillator chips

The Maneatis phase array was constructed and achieved a timing precision equivalent to 14.3% of a FO4 delay in a 1.2μ CMOS process. While the Maneatis array, isn't widely used, the zero systematic skew is a powerful motivator for using a closed loop design. Another strength of this design is the ability of the array to maintain the correct phase realtionships between stages even when the supply is scaled. A third benefit to the closed loop technique is that the extra amplifier used to make the skew rates of the signals approximately equal is not needed. Each stage of extra amplification contributes skew. The accumulated timing uncertainty through a single stage of amplification is significant in high precision applications.

2.5 Signal Distribution

This section illustrates the obstacles encountered when distributing a timing signal across a large piece of silicon. First the steps for building an ideal signal amplification and distribution apparatus, or a clock tree is described. This exercise exposes the limits of this technique.

To distribute the clock signal, high performance microprocessors almost exclusively use amplification through strings of inverters that branch out from a source located in a single location. This section explores the limits of that approach and demonstrate that the solution is increasingly inadequate as the technology moves deeper into nanometer feature sizes.

High performance microprocessors employ various stratagems to remedy the timing uncertainty inherent in the conventional technique of distributing a signal over a large area. These stratagems combine to various degrees two general remedies.

Power A common remedy is to short the outputs of the nodes of the inverters at one or more levels of the amplification apparatus. This is called a grid. As the amount of metal used to short the scattered drivers is increased, the better this piece of metal approximates a single electrical node. This is the desired result.

Unfortunately the power required to charge and discharge a large chunk of metal a few billion times a second is substantial. This medicine can only be taken in limited doses.

Complexity The gloomiest prognostications related to timing uncertainty have been met and can likely be continued to be met by increasing the complexity of the circuitry used to tune the processors timing signal distribution apparatus. This strategy includes some form of closed loop control. First the phase from two synchronizing signals is compared and

then the delays of the gates driving those nodes are adjusted to bring the nodes into phase (8; 10; 30).

Another solution is taking more timing constraints into account. Time borrowing (14) is a favorite technique of high speed data path designers. Time borrowing is used in latch based designs. Typically successive latches in the pipeline are transparent on alternating opposite half cycles. This means that there is one other latch between them that must be transparent during some fraction of the half cycle when the others aren't. Time borrowing allows you to unevenly distribute the logic between two latches operated by the same phase. This allows the logic located before the latch that is made transparent at the half cycle to borrow time from the logic placed after this latch. To borrow time, the data must arrive at the half-cycle latch while it is still transparent. The timing signal that operates the half-cycle latch has a soft edge. In other words it can arrive at any during some small period. During this period the data signal is not stopped and re-timed by the closed latch. While this technique does buy some skew tolerance, it adds to the number of specially timed signals.

Fragmenting the global skew into multi-level skew hierarchy and being aware of local and global skews allows for more aggressive designs but costs in design time (15).

This thesis doesn't provide new remedies for the imprecision resulting from distributing timing signals. These remedies remain available for any clock distribution technique. This thesis is concerned with delivering the timing signal to the location where it is needed with less timing uncertainty. If the precision is still not satisfactory then the remedies remain available but the dose of remedy will be less.

2.5.1 Limits of amplification

Dally and Poulton's indispensable book Digital Systems Engineering (7) contains a detailed description of an on-chip clock distribution solution. Table 2.1 describes the critical physical parameters for the problem. The solution provided in the text is practical. Here a solution is described that is ideal in distributing a clock with the minimum amount of skew using amplification alone.

A script was written that generates descriptions for ideal H-trees to determine the amount of amplification circuitry required to drive the loads spread across a square piece of silicon from a single frequency source. Assuming Manhattan routing, the H-tree is the ideal amplification topology.

The specifications for my H-tree generator are:

Typical on-chip clock distribution problem	
Number of gates	10^6
Number of clock loads	5 x 10^4
Capacitance per clock load	20fF
Total clock capacitance	1nF
Chip dimensions	16 x 16 mm
Wire resistivity	0.07 $\frac{\Omega}{square}$
Wire area capacitance	0.13fF/μm^2
Wire length capacitance	0.08 fF/μm

Table 2.1: Reproduced from Dally and Poulton's *Digital Systems Engineering* (7)

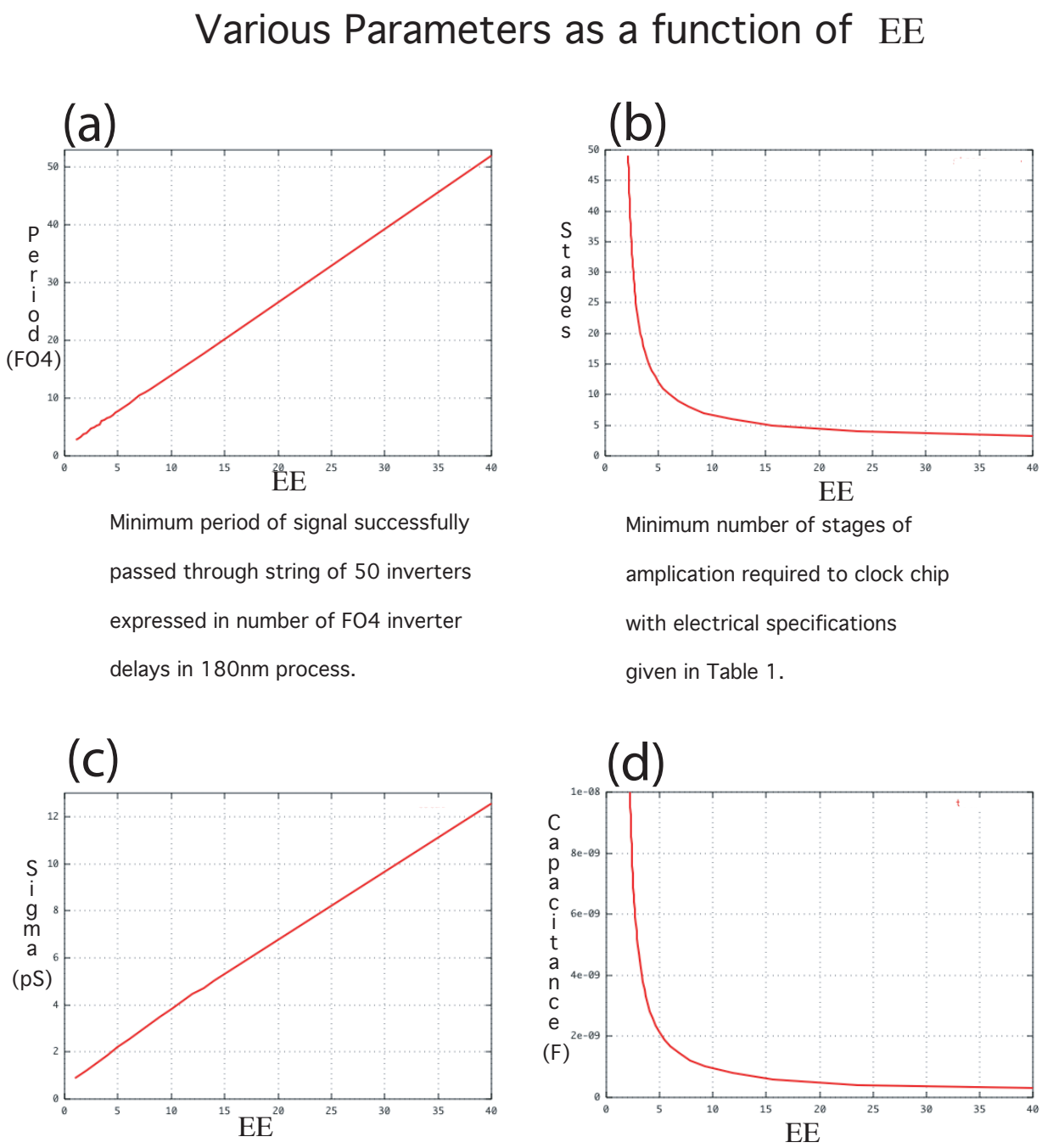
- Given a target value for skew for the lumped time constant of the span between two drivers in the H-tree, the program produces a description of the minimum H-tree necessary to achieve the target. The minimum time constant is specified in multiples of **EE**.
- The program requires that the user specify the size of the initial driver, the length of the side of a square chip, the number of leaves on the H-tree, the targeted time constant at each node of the H-tree, and the load found at each leaf.
- It assumes a best case routing for the H-tree. It assumes that each path from root to leaf is perfectly balanced and routed in the canonical H-tree topology. The wire widths and transistor gate widths are both sized ideally for minimum delay from the discrete choices available. I specified a maximum wire width of 10μm.

The resulting H-tree might be unrealistic to implement but provides a best case signal amplification scheme. It assumes ideal routing, equivalent electrical gain at each stage of amplification, and homogeneously distributed loads. These assumptions yield the fundamental limits to distributing a signal through amplification with digital inverters.

2.5.1.1 Plots

The H-tree generator and Hspice simulations yielded data for a number of plots that show the relationships between parameters critical to constructing an ideal H-tree.

Figure 2.6a plots the minimum clock period for an inverter with unity gain plotted against **EE**. For certain values of **EE** this plot shows the shortest period that can be sent through the inverter before the magnitude of the gain is no longer unity. A signal of this period copied



Minimum period of signal successfully passed through string of 50 inverters expressed in number of FO4 inverter delays in 180nm process.

Minimum number of stages of amplication required to clock chip with electrical specifications given in Table 1.

Sigma for Gaussian(Normal) distribution for skew through a single stage of inversion.

Capacitance found in all drivers and interconnect in ideal H-tree constructed to clock chip with electrical specifications given in Table 1.

Figure 2.6: Crucial functions for driving H-tree

through a string of inverter with the same **EE** will attenuate and perhaps be completely filtered out before it reaches its destination. The period is expressed in units of FO4. The FO4 delay is 75ps using Spice models extracted from a 180nm process.

Figure 2.7 shows a three stage section from the string of inverters in the test setup used to construct Figures 2.6a and 2.6c. Differential signals ripple down two identical strings of inverters. Each stage in the individual strings has two inverters. One inverter conducts the signal to the next stage. The other inverter provides the extra loading to achieve the loading desired. To obtain a data value for a **EE** of two, both inverter are equivalently sized. To obtain a data value for a **EE** of three, the load inverter is twice the size of the inverter used to conduct the signal to the next stage.

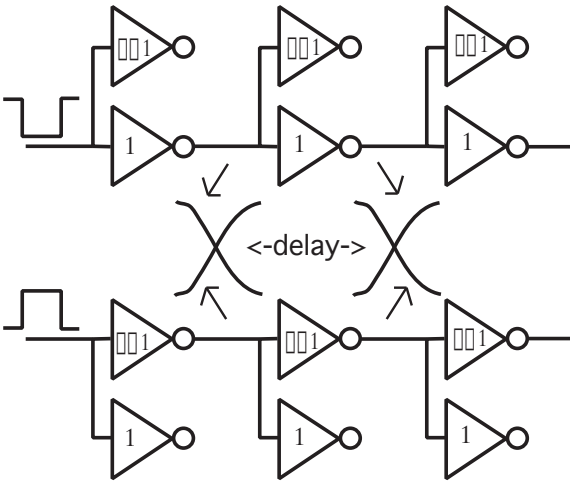


Figure 2.7: Test setup for data in Figure 2.6.

Figure 2.6b describes the number of stages of inversion required to drive the H-tree described in Table 2.1 plotted against **EE**. The lower the value of **EE** the less noise couples onto the clock signal. Low values of **EE** result in quickly driven nodes. The transistors of the inverters spend little time in their linear region where they are most likely to amplify noise.

Figure 2.6c relates the standard deviation of the delay of an inverter. The x-axis shows the **EE** of the inverter. This plot is identical to Figure 2.4.

Figure 2.6d shows the total amount of capacitance found in the inverters and interconnect used to construct the H-tree as a function of **EE**. The total power used by the H-tree is related to the values on this plot through the relationship:

$$Power = \frac{f \times C \times {V_{supply}}^2}{2} \tag{2.4}$$

2.5.1.2 Putting it together

State of the art high performance processors have clock periods of around ten FO4 delays (1). From Figure 2.6a shows that the **EE** for the H-tree must be 7 or less to successfully copy this signal through the string of inverters. From Figure 2.6b shows that a **EE** of 7 requires at least 9 stages of amplification.

I calculate the fraction of the clock period that can be used to perform useful work and what fraction must be set aside because of process skew using Figure 2.6c. The standard deviation for expected skew on a single stage operating with a **EE** of 7 is about 2.5ps.

If we assume $\pm 3 \times \sigma$ skew, then statistically 99.7% of the worst case timing assumptions are met. We expect 15ps of skew per stage or $9 \times 15 = 135ps$ total. That is almost $2 \times FO4$ delays.

Assume we chose to pipeline further to achieve a clock period equal to 8 FO4 delays. This requires a **EE** of about six. A **EE** of six requires 13 stages of amplification. The standard deviation of expected process skew becomes less with lower values of **EE**. But it reduces linearly while the number of stages increases as $\frac{1}{\mathbf{EE}}$. Even though the skew per stage is reducing, the number of stages is increasing faster. The standard deviation for per stage skew for a **EE** of 6 is about 2.2ps. The expected 3σ skew is then 13ps per stage. The expected process skew at the leaves is then $13 \times 13ps = 169ps$.

As the frequency of the clock increases there is less spare time after driving the next amplifier to drive the interconnect. As progressively less time remains to drive the interconnect the number of stages quickly increases. This is the source of the knee in the curve of Figure 2.6b.

These curves show that pipelining much below eight gate delays is infeasible. The knee of the curve shown in Figure 2.6b is approximately at 7**EE**. Further pipelining simultaneously reduces the amount of time available to compute, and increases the skew which eats into the available computation time.

Notice also from Figure 2.6d that the power increases rapidly with extra stages of amplification. My model doesn't accommodate the inevitable extra jitter that results from the exponentially increasing power of the clock tree.

2.5.2 Distribution solutions

This section lists alternatives to the H-tree and its derivatives for delivering a timing signal in a microprocessor. I don't know of any cases where these solution were adopted for commercial use.

I sort the existing alternatives of distributing a clock signal under three categories.

Coupled Oscillators Signal distribution techniques in this category share the technique of taking some number of oscillators, distributing them over the area of the chip, and synchronizing each of them to the average phase of its neighboring oscillators (13; 16; 17; 31).

Pratt (29) articulated the weakness to this method. Phase averaging can produce an undesired stable equilibrium. Mode lock is a stable system equilibrium in which the phase averaging mechanism used to couple the oscillators settles the oscillators in a non-zero phase relationship. If two of the phases contributing to the average are of equal but opposite magnitude then an undesired and stable phase equilibrium occurs. Pratt goes on to show that if the phase detector produces a control voltage that increases linearly for phase differences between +/-90°and decreases between 90°and 270°, then mode lock is avoided. Fortunately, a simple XOR implements this error function. Unfortunately, his implementation requires the routing of long analog control lines that are easily corrupted by noise.

Transmission Line Wood et al (45) distributes a clock signal by creating a rotating traveling wave within a closed-loop differential transmission line. A number of transmission lines are routed throughout the chip and coupled. The dependable LC characteristics of VLSI interconnect is exploited to adiabatically re-circulate the energy in the clock distribution system. 950Mhz and 3.4GHz clock signals were generated in a 0.25μm test chip. Perhaps this solution's biggest weakness is that it is too unconventional.

Mahony et al (25) similarly use transmission lines that carry oscillating waveforms where the energy is adiabatically conserved. Unlike Wood, the network uses standing waves instead of traveling waves. Although the clock signal is largely in phase everywhere, the amplitude varies according to where along the transmission line the clock signal is tapped. The frequency attained is 10GHz in 180nm. I assume this clocking technique is waiting for transistor technology to improve to where it can use a clock of this frequency.

Wireless Here the clock signal is distributed using a wireless signal communicated through waveguides (32) or simply broadcast (12). In both of these cases the signal is distributed with negligible skew. Unfortunately the circuitry required to convert the signal into the electrical domain presents problems.

In the first case the required circuitry is large, expensive, and introduces a substantial amount of timing uncertainty. The transmitter and receiver in the second case are physically very large. The paper presenting this work transmitted a 15GHz clock which must be divided down numerous times to get the clock frequency. Frequencies faster than 15GHz can't be successfully captured and divided down because of the intrinsic speed of the transistors. If the transistors were able to handle higher frequencies then the wavelength of the signal that broadcast the clock would be shorter. Then the antennas and

receivers could have been sized more reasonably. Perhaps this technology will be more attractive in the future.

2.6 Summary

This chapter surveys the conventional methods for generating sub-gate delay timing signals and for distributing a signal over a large area. Skew from process variations quickly limits the attainable precision as the demands for more precision or the area of distribution increases. The reason in either case is simply that more stages of amplifiers are needed as the precision demands increase. Each stage accumulates skew that further erodes into a shrinking period.

Chapter 3

Ripple FIFO basics

3.1 Introduction

This chapter provides the foundations that allow the reader to navigate through the rest of this thesis. A short primer on ripple FIFOs provides a foundational understanding of the terms and concepts used. A description of the operation of three FIFO protocols illustrate how handshakes move data between stages.

A survey of two papers that explore the fine grained movements of tokens in a ripple FIFO provides context for this work. Having introduced ripple FIFOs and explored how far the previous research advanced, I discuss my goals.

3.2 FIFO Basics

First In First Out registers are used to preserve the order of data tokens. As the name suggests, the first data token placed into the FIFO is the first data token that can be accessed at the output of the FIFO some time later. I use FIFOs connected in rings to generate and distribute high precision timing signals.

When the output of the FIFO is connected back to the input, the name FIFO ceases to accurately describe its behavior. This thesis uses 'FIFO' to mean a FIFO wrapped into a ring so that is no longer has an input or output port. The FIFO retains the handshaking mechanisms and protocols to ensure token ordering. But the purpose of the rings is to provide dependable high precision timing signals. Data path delays are the concern of the data path designer. Confounding data path design and control path design is a major obstacle to adaption of self-timed techniques (43).

Ripple FIFOs are distinguished from Address FIFOs. Address FIFOs read data from the same address in which they were written. Read and write pointers ensure that order is preserved. Centralized logic manipulates and updates the complete state of an address FIFO . A stage in a ripple FIFO concerns itself only with the occupancy state of adjacent stages. A FIFO stage moves a token forward when a $FULL$ stage immediately precedes an $EMPTY$ stage. Handshaking signals communicate the occupancy state by obeying protocols. In a two phase `Micropipeline` protolcol (40), stage n signals to a stage $n+1$ the presence of a token by asserting a $Request$ signal. If the second stage is not occupied by a token, then it consumes the token and returns an $Acknowledge$ signal to the first stage, which is then declared empty.

Tokens and holes are useful abstractions for thinking about data movements in a FIFO. In reality a hole and a token are simply states of signaling wires.

Token movements in FIFO stages are often viewed as atomic. A token either occupies a stage or it does not. For my application, it might help to think that a token can be smeared across a number of stages. Token movements are not viewed as being atomic but rather gradual and analog.

3.2.1 Ripple FIFO control variants

Figure 3.1 shows three ripple FIFOs built from a different control. Each FIFO is three stages with a one bit data path. While each of the control types shown in Figure 3.1 is shown controlling a single bit data path, the application described in this thesis is data path agnostic.

3.2.1.1 `Micropipelines`

The first FIFO control shown is a `Micropipeline` (40). `Micropipelines` communicate using event logic. A voltage transition is called an event. A HI to LO voltage transition carries the same meaning as a LO to HI voltage transition.

A datum is communicated between stages in two phases. In the first phase a stage toggles the value on its $Request$ wire. This signals to the next stage that a data is available to be consumed. In the second phase the requested stage latches the data and toggles the $Acknowledge$ wire alerting the first stage that it has consumed the data. The event that $Acknowledge$s the previous stage also serves as a $Request$ to the next stage. A token movement requires only two events.

A token exists where the $Request$ wire leaving a stage is at the opposite potential from the $Acknowledge$ wire entering the stage from the next stage of FIFO control.

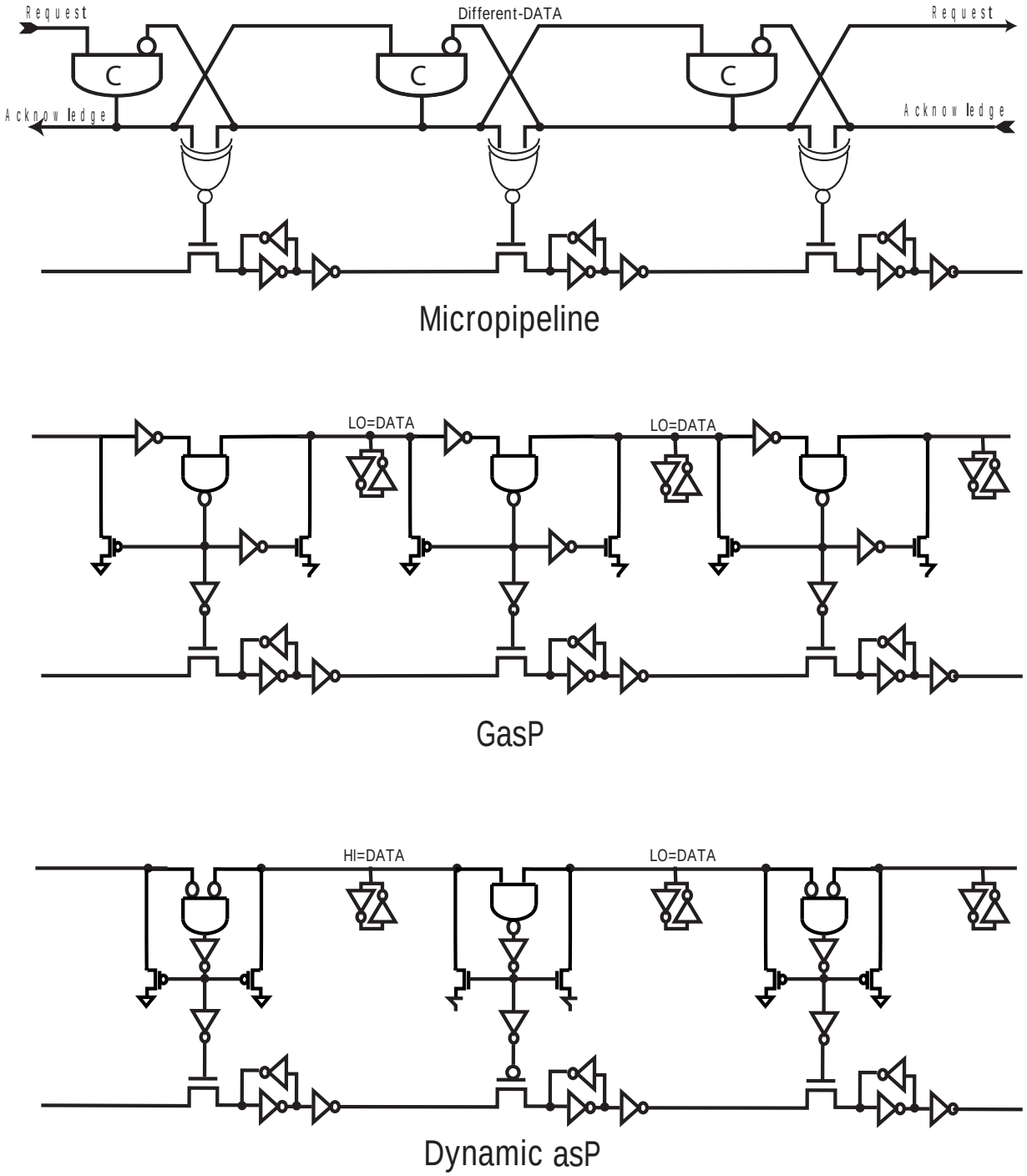


Figure 3.1: `Micropipeline`, `GasP` and `Dynamic asP` ripple FIFO control

3.2.1.2 GasP

The second FIFO control shown is GasP (38). Each GasP stage is connected to the previous and next stage by a state conductor which is simply a length of wire with a staticizing *keeper* made from back to back inverters. A HI potential on the state conductor means that the latch controlled from the previous GasP stage is empty. A LO potential means that the latch is $FULL$ and has data. In GasP, a LO voltage on a state wire locates a token.

When the GasP stage detects that the previous latch is $FULL$ and the next stage is $EMPTY$, the NAND gates asserts and three actions are initiated.

Copy data Two gate delays after the NAND gate asserts, the data latch is made transparent and the data item in the previous latch is copied forward.

State conductors toggled The PMOS transistor with its drain connected to the previous state conductor and the NMOS transistor with its drain connected to the subsequent state conductor both toggle the potential of the two conductors. The previous state conductor is made HI, declaring the latch $FULL$. The next state conductor is made LO, declaring the latch $EMPTY$. The token has been moved forward a stage.

Reset The PMOS transistors in the NAND gate are exercised in unison three gate delays after the NAND asserted to un-assert the NAND gate. Soon after, the latch becomes opaque and the drive is removed from the PMOS and NMOS transistors used to toggle the potential of the state conductors.

Perhaps the most attractive feature of GasP is the single wire used between FIFO stages to communicate. Typically the wire that connects stages is long, while the wires within the FIFO stage are short. Using a single wire to communicate minimizes the power necessary to distribute the control, especially over long distances. The long wire is driven by single PMOS and NMOS transistors. These drivers are about one and half and three times more efficient than an inverter at driving a load.

GasP places four gate delays along the forward path through the FIFO control and two in the reverse. For the application of controlling the movement of data in a self-timed FIFO this makes sense. The delays in the forward path model the delays in the data path. Time is required for data to propagate through a latch. Declaring a latch empty requires only the time needed to change the potential on the state conductor from LO to HI. As little time as possible should be spent doing this.

While having uneven forward and reverse delays is beneficial for the purposes of self-timed systems, it is not ideal for the purposes of distributing high precision and high frequency timing signals. Later, this chapter lays the foundations necessary to understand why.

3.2.1.3 `Dynamic asP`

The third FIFO control shown is called `Dynamic asP` (20). It is a pre-cursor to `GasP`. Its actions are very similar to `GasP` but it has three gate delays in the forward and in the reverse directions.

`Dynamic asP` has two sexes of FIFO control. The two types of FIFO control alternate.The data path is constructed with alternating latch types because they use a different type of control. The two latch types copy data forward when being controlled by opposite voltage polarities. One type of FIFO control detects a LO potential on the two state conductors that connect it to the adjacent stages. The other type of control detects HI potentials and resets them LO.

Confusingly, the meaning of the potentials on the state conductors alternates as well. A HI potential means $FULL$ on the state conductors before the stages with NMOS transistors connected to the state conductors while a LO potential carries that meaning in the other stages.

The same three actions initiated in `GasP` when the move conditions are detected are initiated in Dyanamic asP.

3.2.2 Throughput

The throughput of a ripple FIFO is the average number of tokens that pass through a FIFO stage in some period of time.

A ring of ripple FIFO control loaded with some number of tokens has three regions of operation. They are token limited, hole limited, and self limited. In the token limited region, each stage of FIFO control waits for a $Request$ before initiating a token transfer. While the ring operates in this region, the average number of tokens consumed by a stage in a period of time increases linearly with the number of tokens in the ring. In other words, the FIFO is deprived of the tokens necessary to achieve peak throughput. Similarly, in the hole limited region, the availability of spaces for the tokens to occupy limits throughput. In this region, the token throughput increases linearly with the number of holes.The ring can be described as congested.

A plot of the throughput versus the normalized occupancy for a typical FIFO ring constructed of `GasP` control circuitry is shown in Figure 3.2. This plot uses data from HSpice simulations. When the two lines describing the token throughput as a function of the ring occupancy are extrapolated, they intersect at a point. This point describes the peak throughput neglecting second order effects. The normalized occupancy of this point is described by the equation:

$$\overline{O_{peak}} = \frac{L_{for}}{L_{for} + L_{rev}} \tag{3.1}$$

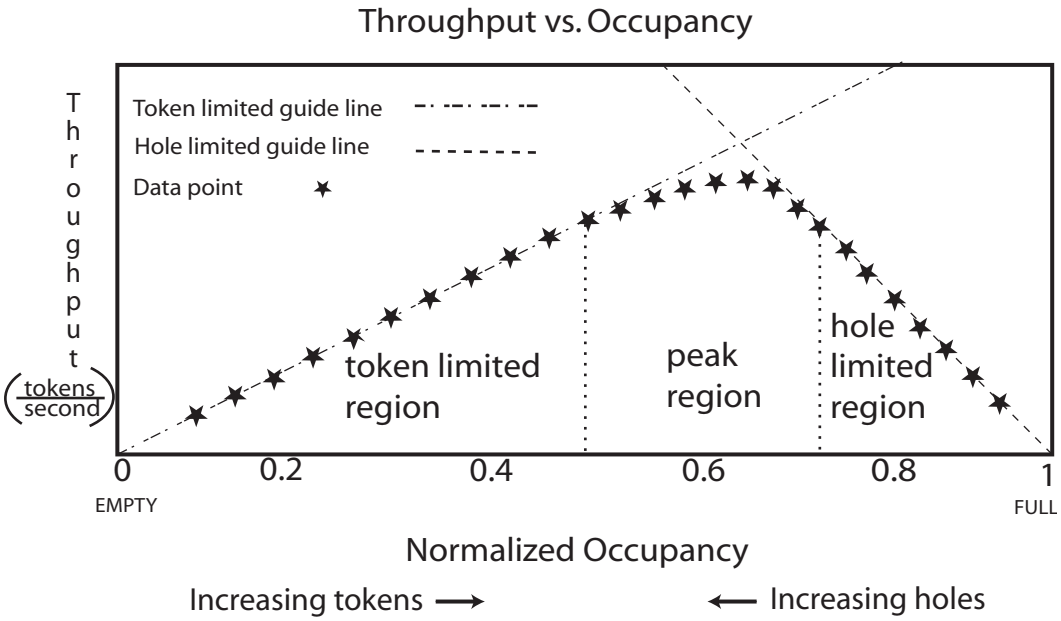


Figure 3.2: Example Throughput vs. Occupancy Plot

The overbar in the equation signals that this is a normalized value and ranges from zero to one.

L_{for} and L_{rev} are the magnitudes of the slopes of the lines that describe the throughput of a FIFO in the token limited and hole limited regions of operation. They are equivalent to the forward and reverse latency of a FIFO stages as measured from the last asserting input needed for a token movement while the FIFO operates deep within the token limited and hole limited region respectively.

The pyramid shape formed by the lines describing the throughput in the hole and token limited regions provides a good first order approximation of the FIFO ring's throughput behavior.

The peak region is centered around the peak occupancy, $\overline{O_{peak}}$, described by Equation 3.1. This is the region where the actual throughput pulls away from the predicted pyramid shape. Two effects cause the token throughput to pull away from that predicted. Depending on the relative strengths of these two effects, tokens might bunch or spread. The strength of each effect is determined by transistor sizing, wire lengths, and the type of control employed. The next section identifies these effects.

Sometimes tokens find themselves in discrete packs in a ring. Often the tokens do 'reverse Indian sprints', where the back token in the pack drops off pace until the head of the token pack catches it. The key to token spacing is amplifying the effect that causes tokens to spread, while mitigating the sources of token gathering.

3.2.3 Locking

If tokens spread evenly throughout a FIFO ring, then the handshaking signals that communicate between adjacent stages must charge and discharge at predictable times relative to each other. In `GasP` (38) a token movement causes a $360°$ phase transition between stages. If six tokens evenly spread around a ring composed of 13 `GasP` control FIFO stages, then we can also think of there being $\frac{6}{13}$ of a token per stage. If a token causes $360°$of phase shift as it moves between two stages, then the phase difference between control wires in adjacent stages is $\frac{6}{13} \times 360°$.

The state in which tokens are spaced equidistantly around the FIFO ring I call the *locked* state. FIFOs with tokens in the *locked* state operate in the narrow range of occupancies that make up the peak region in Figure 3.2.

3.3 Previous Token Spacing Work

This section describes previous work that investigated the fine grained dynamics of token movement in a ripple FIFO.

3.3.1 Token movement in an open FIFO

Ebergen et al (9) explored how data tokens distribute themselves throughout a free running ripple FIFO. They presented the Charlie Plot as a tool to facilitate understanding, see Figure 3.3. The Charlie Plot elegantly demonstrates how the delay for a C-element logic gate is continuous with respect to the separation time between it's inputs asserting. It does this by a clever choice of axes.

The x-axis of the Charlie Plot records half the difference in the separation time of the inputs. The y-axis record the delay as measured from this point in time. If the left and right input to a stage arrived at 3ns and 5ns respectively and the output was generated at 6ns, then a data point for the Charlie Plot is recorded as $\frac{5ns-3ns}{2}$, or 1ns, for the x-coordinate and $6ns - (5ns - 3ns)$, or 4ns, for the y-coordinate.

In this paper, an accurate delay analysis and characterization of the FIFO was performed using a Charlie Plot for an asynchronous `Micropipeline` control stage (40) along with the response times of the source and sink to the FIFO. They predicted how tokens could be expected to distribute themselves over a section of FIFO using these tools and confirmed the predictions using HSpice simulations.

Most important to this present work, the authors identified the phenomenon where a lead token repels a second lagging token in the FIFO as the distance between them shrinks. They explained that this effect is a result of the increased delay of the C-element used within the FIFO

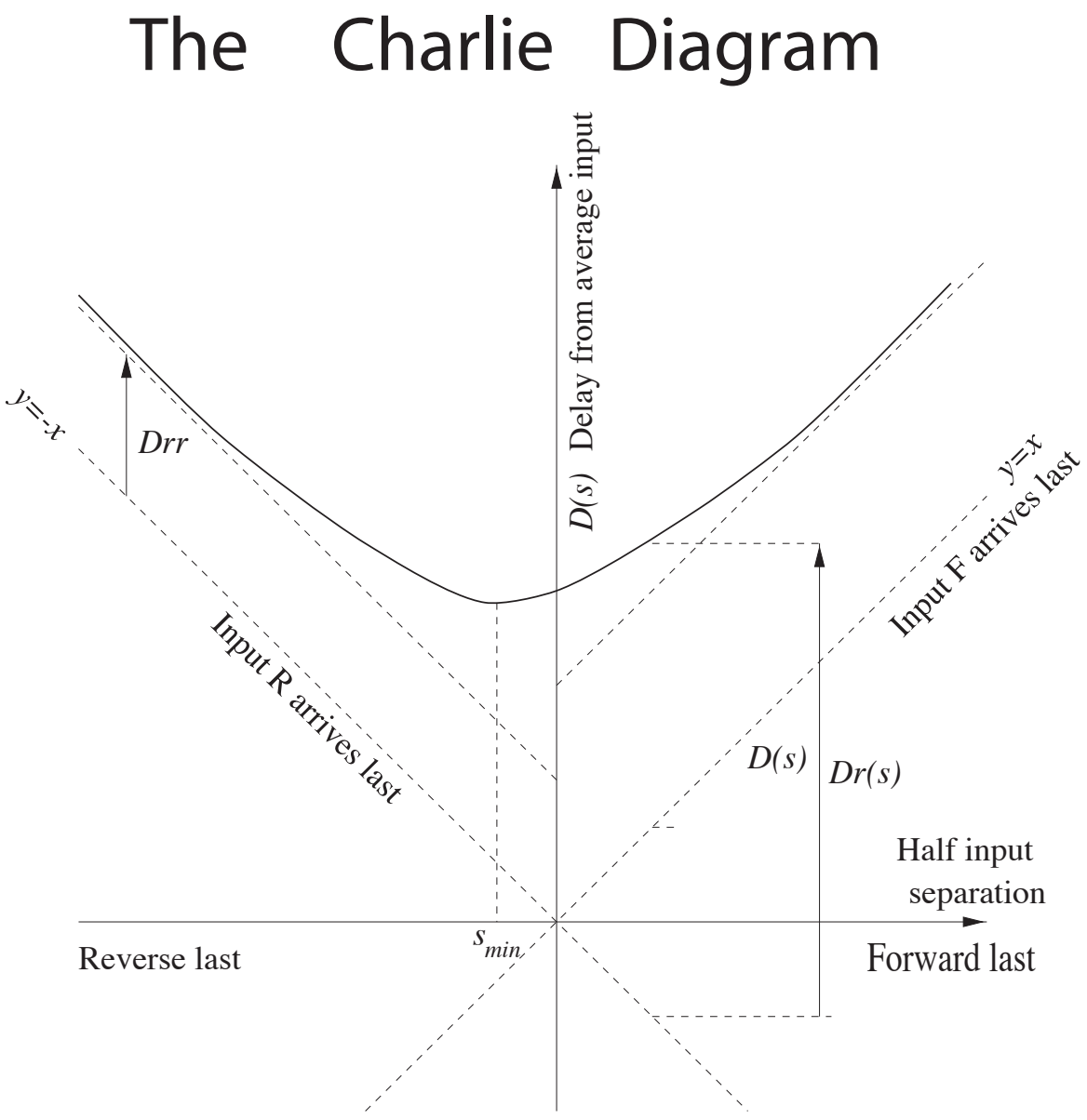

The Charlie Diagram
D(s) Delay from average input
y=-x
Drr
Input R arrives last
y=x
Input F arrives last
D(s)
Dr(s)
Half input separation
Reverse last
s min,
Forward last

Figure 3.3:

control when the temporal separation between its inputs decreases. They called this effect the **Charlie Effect** after Charles Molnar who first identified the source of the token repulsion while viewing his eponymous plot.

The extra delay has three sources. All three sources have a maximum effect when both inputs to the FIFO control arrive simultaneously and the effects taper as the temporal separation between inputs grows. They are:

- The resistance of the series drive transistors increases as the sum of the time that both transistors transition from cut-off to saturation increases. This increases the RC time constant that determines the gate delay.
- The un-exhausted nodal capacitances increase when one input is progressively less setup in advance of the other. This effect also increases the RC time constant of the gate.
- The cross-over or totem pole currents increase because of the brief period when both pull-up and pull-down stacks conduct. Current that would otherwise charge the output is wasted.

The Charlie plot has two asymptotes, plus and minus forty-five degrees with x-axis. These two lines represent the delay of the FIFO control without the Charlie effect. The Charlie plot pulls away from the +/- 45 asymptotes over a range of separations. This range shows where the Charlie effect influences token movement. The distance between the asymptotes and the Charlie plot is proportional to the magnitude of the Charlie effect.

This paper also identified the **drafting effect**. The drafting effect causes tokens to bunch. When one token closely follows another, the handshake signals that effect data movement might not fully transition to the supply voltage or ground before being forced in the other direction. The second token experiences less delay than the first because of the decreased potential that must be overcome to effect the second data movement. The decreased delay results in the distance between the tokens closing even further, in other words bunching. While this effect is identified, the drafting effect is not reflected in the Charlie plot presented in this paper.

This paper identified the two basic forces that govern the spacing of tokens in a ripple FIFO. It created a language that helped to discuss, identify, and quantify forces on things called tokens. This is noteworthy because tokens are really not ‘things,’ but simply convenient mental abstraction. Language using imagery from physics accurately describes how these tokens interact.

3.3.2 Token movement in a closed FIFO

Winstanley et al (44) continues the investigation of token movements in FIFOs by closing the section of `Micropipeline` FIFO into a ring and forcing the tokens to gather into a bunch or spread evenly through the ring using a clever circuit that allows you to apply positive or negative feedback to the output of their control. The feedback circuit requires an external current

reference that allows you to control the amount of feedback needed to overcome the Charlie and drafting effects.

The circuit is reproduced here as Figure 3.4. The transistors that implement the C-element function are on the far left of the figure. The inputs are labeled 'A' and 'B.' There are two feedback paths to the output of the C-element. The first is composed of the inverters 'W' and 'X.' They keep the value of the output when the input values differ. This keeper is needed because when the inputs differ the output is isolated from supply and ground. The second loop is composed of inverters 'X', 'Y', 'Z' and the structure the authors refer to as a 'crummy buffer.' This gate applies a weak amount of feedback that will counteract the value held by the keeper inverters. Notice that it pulls down through a PMOS transistor and up through an NMOS transistor. Also, the amount of feedback is regulated by an off-chip reference current which controls the amount of current feeding the crummy buffer. When the reference current is large, then the output value will partly transition towards the opposite value roughly four gate delays after the C-element output last changed value. The purpose of the second feedback loop is to accelerate transitions when the inputs are greatly separated. Greatly separated inputs occur when tokens in the closed FIFO ring are greatly separated. Because the output transition is accelerated, its effect is to shrink the distance between greatly separated tokens.

They added an extra axis to the Charlie plot characterization to reflect the influence of the drafting effect. The delay of their control is dependent not only upon the temporal separation of its inputs but also heavily dependent on the rate at which it is being exercised.

They fitted their three dimensional Charlie curve used to characterize their circuit with an involved formula. They modeled a hyperbolic dependence of delay on input separation, s. They used a negative-exponential dependence on the time since the last output event. The resulting equations are:

$$Charlie(s, y) = u(s) + \beta(1 - e^{\alpha(y+u(s)}) \tag{3.2}$$

where:

$$u(s) = 1 + \sqrt{1 + (s + 0.1)^2} \tag{3.3}$$

y is the time from the last input event to the average of the arrival times of the input events. β relates the relative strength of the drafting effect. The formula allowed them to make some accurate predictions about the operation of the FIFO ring and helped to troubleshoot problems but ultimately failed to predict whether tokens would bunch or spread for all cases.

Their motivation for exploring why the tokens sometimes bunch and sometimes gather is much different than my own motivations. They wanted to understand the causes and implications of token bunching in asynchronous environments. They explained:

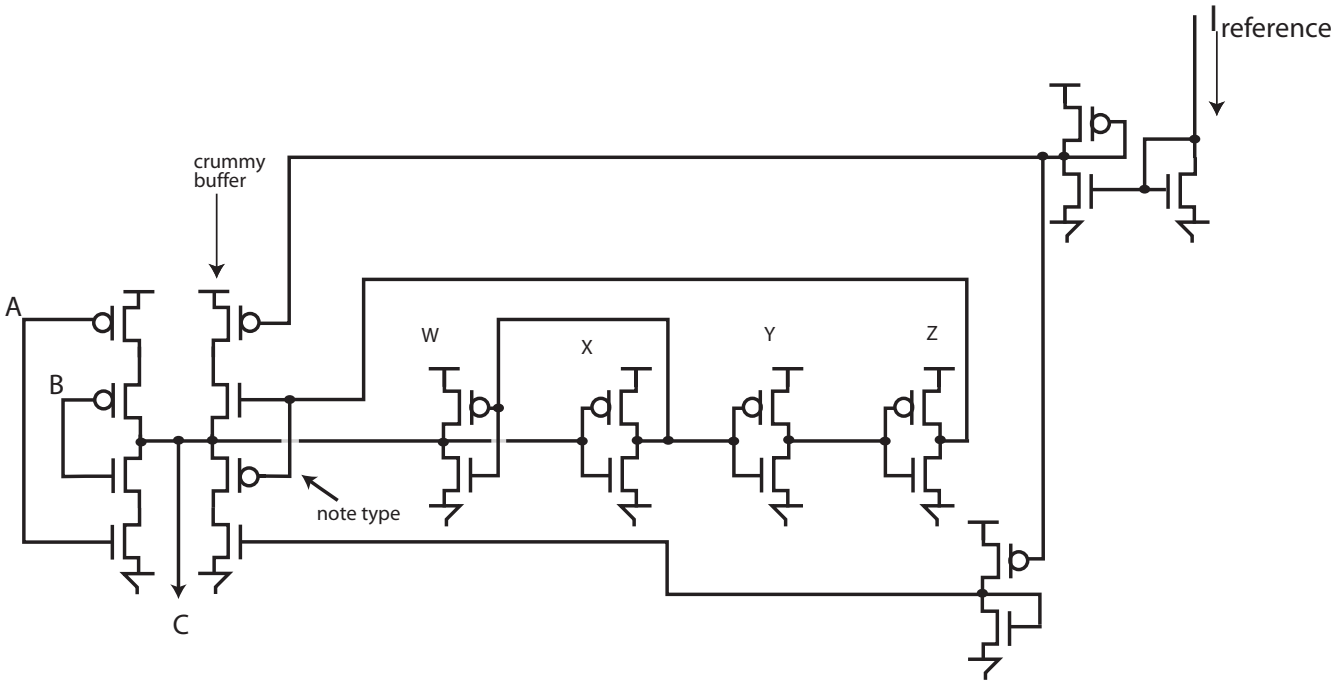


Figure 3.4: Winstanley, Garivier, and Greenstreet's `Micropipeline` FIFO control for token spreading

> *If a self-timed design uses bundled control, then the data path must be capable of operating at the minimum cycle time of the bunched tokens. Long term throughput however is determined by the average cycle time. Bunching results in a performance compromise. We seek to understand token movements to allow us to control bunching and spreading and therefore increase the performance of our asynchronous pipelines.*

They do identify high precision delays and clock distribution as potential applications, but no guidance is provided as to how this might be done.

This paper's most important contribution was showing that a circuit could be constructed that amplifies the effect that causes tokens to spread. They showed that the drafting and Charlie effect were properties that could be controlled.

3.4 Moving Forward

3.4.1 Simplicity and Dependability

Winstanley's work was interesting but esoteric. Remember that their primary motivation was to incrementally improve the timing margins of the data path in self-timed designs. I believe they developed an interesting technology, but for the wrong application. The revolutionary trait of his circuitry, which was identified but not capitalized upon, was that it gave some certainty in the timing relationship between different nodes in different locations. The certainty was derived from closed loop control instead of open-loop amplification. The circuitry of the closed loop control was simple.The feedback is supplied by the existing $Request$ and $Acknowledge$ handshaking wires. But they required an external input and a large number of extra transistors to achieve this.

Some other problems remained that could be obstacles to using their control for high precision timing applications. Most detrimental was that the control was not predictable. The unpredictability had a number of sources. They are:

Symmetry — Their control is not electrically symmetric meaning the Charlie effect has a different magnitude when the ring is starved of tokens or congested with tokens.

Internal Nodes —The C-element transistor stacks contains internal nodes between the source and drains of like transistors. These internal nodes dump varying amounts of charge onto the output nodes depending on how the gate is exercised.

Drafting Effect —Their control had widely different delay characteristics depending upon the rate at which it was being exercised.

Model —Their model captured the basic character of the control but was not able to predict exactly when the tokens would bunch or spread.

The primary impediment to accurate predictions of token spreading is their model. My research dispenses with the Charlie plot for characterization. The Charlie plot highlights the continuity of the delay of the FIFO control as the separation time of its inputs asserting changes. Modeling the function that results from the clever but confusing choice of axis fails to yield intuition of token dynamics. I choose axis that highlight discontinuities in the operation of the FIFO control.

The issues that remained to be resolved to make ripple FIFO control dependable for the purpose of distributing precise timing signals are:

Eradicate or drastically mitigate the drafting effect that causes the tokens to bunch.

Amplify the Charlie effect that causes the tokens to spread.

Remove the electrical asymmetry that results in a different bunching characteristic depending on if the FIFO ring is token or hole limited.

Characterize the FIFO control such that one could predict whether tokens will spread or bunch in a ring FIFO and what the resulting phase relationship between nodes in the ring will be.

Streamline the FIFO control to extend the range of frequencies over which it can be exercised.

3.4.2 Speed, Speed, Speed

Although speed was not a stated goal of the project, the FIFO control in Winstanley was large and slow, perhaps too large and slow to be used as a competitive alternative to other timing methods. The waveforms presented in the paper show a 200MHz signal for a control fabricated in 0.35μm process.

Numerous sources, including my own experiments, have identified a practical limit near periods of 8 FO4 gate delays (41; 46) as being the minimum distributable clock period. Signals with periods shorter than this require almost the full period to just drive the next amplifier stage, let alone any significant span of interconnect. The number of amplifier stages required to distribute the signal then explodes. Yet numerous test chips have been built using ripple FIFOs that send signals with periods shorter than 8 FO4 (6; 33) over long distances. A clear motivation to using ripple FIFOs for distributing high frequency signals is an increased maximum attainable frequency.

Ripple FIFOs generate signals rather than copy them. Signals are generated in response to a detection. As a token moves through a ring, it is not slowly degrading as a transition in a string of inverters. The timing signal is restored in response to another signal reaching a certain threshold.

Ripple FIFO control such as `GasP`, `Dynamic asP`, and `Micropipelines` perform only those actions essential to token movement. `Micropipelines` waste no transistors performing return-to-zero due to the two phase handshaking protocol they employ. Of the six distinct gates within a `GasP` (38) control loop, all are at least as efficient as an inverter[1].

The drive of an inverter in an H-tree is divided at each branch point. The drive from the root of an H-tree is evenly divided in the paths to each of the leaves. Each gate in ripple FIFO control drives other gates that generate signals to eventually again drive it. The only electrons to escape the loop go directly into driving the elements that need to be timed. In the parlance of $LogicalEffortTheory$ (39), ripple FIFOs distribute a signal with less $branchingeffort$.

A ripple FIFO is self-timed. Ripple FIFOs oscillates at exactly the speed at which it can. An H-tree nearly always uses more resources than needed. If an H-tree is under-designed then the clock signal is filtered out before being amplified to the leaves. It is nearly impossible to size all transistors and interconnect to be just big enough to distribute the clock signal. H-trees must be overdesigned to provide for worst case process errors that might cause the clock signal to be filtered.

3.4.3 Goals

To use ring FIFOs as attractive alternatives to other methods for high precision timing signal generation and distribution a number of challenges remain. The range of occupancies that results in token locking in a ring FIFO must be extended. A ring achieves a wider range of tokens locking by amplifying the Charlie effect and mitigating or eradicating the drafting effect. The resulting cycle time should be short. Time resolution and the clocking limits increase with higher frequencies. An understanding of token movements in a closed FIFO must increase. A better understanding of the mechanics of token lock enables accurate predictions of whether a certain number of tokens lock in a FIFO ring. Economy of area, interconnect, and power are desired.

[1]The NMOS transistor in `GasP` control that is used to drive the state conductor $FULL$ has less than a third of the input capacitance necessary to source the same current as an inverter. The PMOS pull-up transistors in the NAND gate can be half-sized because they are always exercised in tandem. This alteration makes this NAND gate as efficient as an inverter at driving a load.

Chapter 4

Micropipelines

4.1 Micropipelines

Micropipelines (40) are a specific type of ripple FIFO. The Micropipeline FIFO stages communicate using a two-phase handshaking protocol (37). A new *Request* or *Acknowledge* is signaled by toggling the binary value that exists on the *Request* or *Acknowledge* wire. *Request* and *Acknowledge* signals alternate. Each *Request* and *Acknowledge* handshake signifies a token transfer between the FIFO stages they connect. It is convenient to locate a token in the space between two FIFO stages where the *Request* signal going forward is at a different potential than the acknowledge signal coming back.

This chapter presents a novel Micropipeline FIFO control. FIFOs are constructed using this control and connected in a ring. Tokens are initialized into the ring. The tokens spread evenly throughout the FIFO ring. Tokens in these FIFOs do not signal the presence of data. Rather the tokens are used solely to generate timing signals.

A single token movement causes $180°$ of phase shift on the handshaking wires that enact it within a ring built from asynchronous FIFO stages that communicate with Micropipeline two phase signaling. If T tokens are confined to an N stage ring and spread evenly through the ring, then the phase shift at the output of two stages which are n stages apart must be:

$$\phi_n = n \times \frac{\mathrm{T}}{\mathrm{N}} \times 180° \tag{4.1}$$

'Fractional number of tokens per stage' and 'phase offset per stage' are then equivalent descriptions. For example, eight tokens locked in a thirteen stage ring has a fractional occupancy of $\frac{8\ tokens}{13\ stages} = 0.615\ tokens/stage$ which is equivalent to saying there is a $110.8°$ phase shift between the request signals sent by adjacent stages.

The Micropipeline control I employ uses differential signaling. Each signal in the Micropipeline is represented with a true signal and its complement. These two signals are

always 180° out of phase. The number of nodes in the ring is the number of stages times two, $2 \times N$.

If the number of stages is a multiple of the number of tokens, then some of these nodes generate signals of the same frequency and phase. For example, if x tokens spread evenly in a `Micropipeline` ring of $2 \times x$ stages, then a signal of the same frequency and phase is found in every other stage. Equation 4.1 says that the phase relationship between the outputs of stages separated by $n = 2$ is $\phi = 2 \times \frac{x/2}{x} \times 180° = 180°$. The output is represented in differential form, so theoretically the true form of the output in one stage is in phase with the complement form of the output two stages before and after.

I refer to the total number of phases in a ring by Φ. If the number of stages, N, is a multiple of the number of tokens, T, then the total number of theoretically unique phases, Φ is equal to twice the number of stages:

$$\Phi = 2 \times N, \qquad \text{if} \quad N, T \; are \; co-prime \tag{4.2}$$

However if the number of stages and the number of tokens are not co-prime, then a sequence of phase relationships between the stage outputs repeats. For example, eight tokens in a sixteen stage FIFO result in the phase of the output signals relative to some chosen stage moving in the direction of token movement to be: 90°, 180°, 270°, and 360°, and then the pattern repeats three more times before arriving back at the chosen stage. When the number of stages is a multiple of the number of tokens, the expression for the total number of unique phases, Φ, in a ring FIFO made of N stages with T tokens is:

$$\Phi = 2 \times (\frac{N}{T}) \tag{4.3}$$

with

$$T \qquad \text{copies of each phase.}$$

For example, Figure 4.1 is a 16 stage ripple FIFO ring. Assume it has four locked tokens. The *Request* signals of subsequent stages are advanced in phase by 45° according to Equation 4.1. Because the number of stages is a multiple of the number of tokens, the total number of phases is eight according to Equation 4.3. The *Request* signal and its complement is available at each stage. Therefore, we expect to find signals of the same frequency and phase at each fourth stage, for a total of four available locations to find any phase.

The ring is arranged on the silicon according to where the phases are needed. The stages in Figure 4.1 arbitrarily form an L shape to highlight this. The route of the ring is determined according to where the timing signals are needed. Broken lines represent four of the *Request* wires between stages in Figure 4.1. These wires carry signals that are in phase.

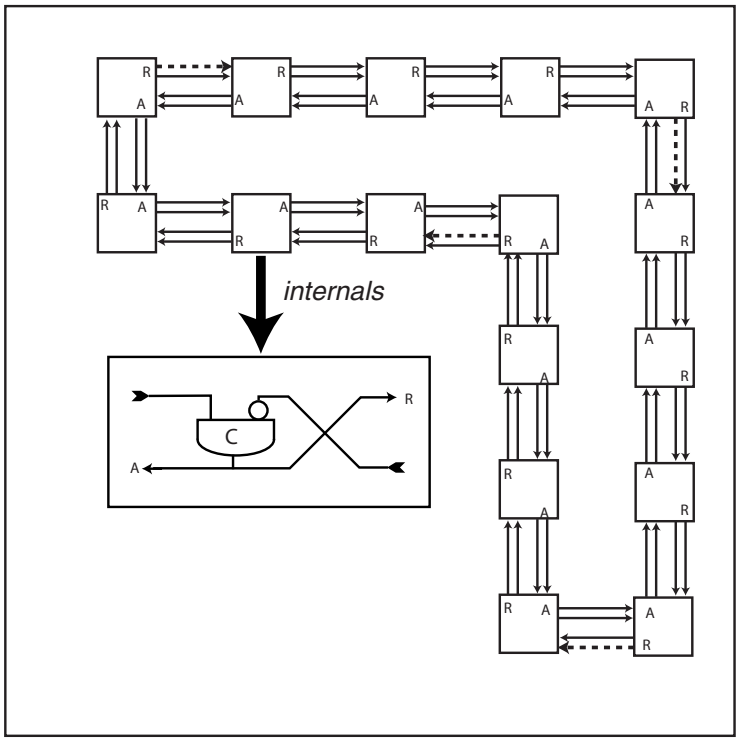


Figure 4.1: FIFO ring for generating and distributing timing signals

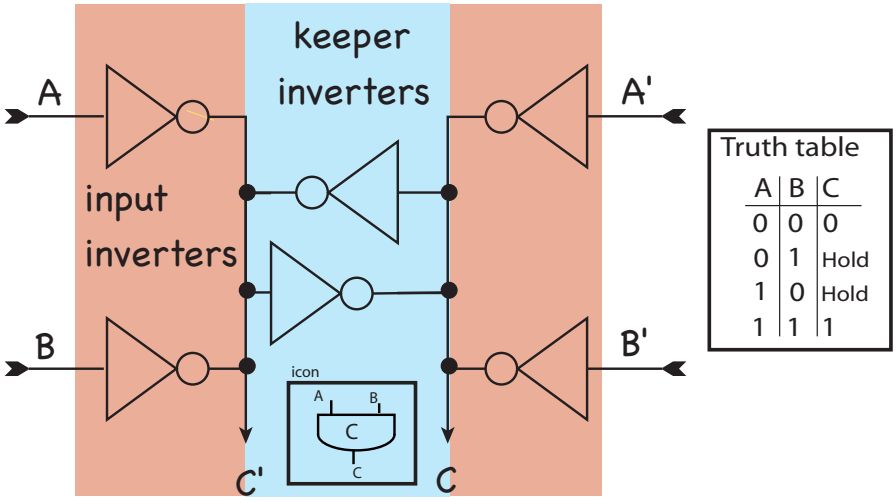


Figure 4.2: Analog C-element

4.2 The FIFO Stage

Figure 4.2 shows a circuit that implements the C-element function (23). This C-element alone is used as the FIFO stage for this application.

Complementary values are held on the output by the cross coupled *keeper* inverters. When the inputs agree, the state of the output is set to the same value as the inputs. When the inputs disagree the binary state of the output is held at the value of the inputs when they last agreed. The actual potential at the output of this gate is removed from the supply or ground by an amount determined by the voltage divider formed by the inverter connected to the asserted[1] input against the keeper and the inverter connected to the unasserted input. I call this an Analog C-element because its output voltage is expected to take on a range of values and its inputs sense this same range of values.

Some things to note about the Analog C-element are:

Symmetry — It is completely symmetric. I don't know of any other electrically symmetric `Micropipeline` stage. When tokens are locked within a FIFO ring built from this control, the duty cycle of the resulting signals are necessarily 50% neglecting potential noise effects.

Robustness — The shortest path back to any node through adjacent stages is three gate delays, ensuring robust oscillations if all inverters are of the same size.

Economy — It uses 12 transistors. None of the transistors are in series. This circuit lays out in an area of $6 \times 6\mu^2$ in a 180nm process using 2μm wide PMOS transistors and 1μm wide NMOS transistors was. The drains of the inverters connected to the inputs should be shared to reduce the drain capacitance for a stage by a third. The square layout lends itself to simple placement and abutment of cells. The short distance between cells results in insignificant wire loads.

Static Current — If the inputs disagree then static current is drawn. This is a problem if this FIFO stage were used in an asynchronous environment where it might wait an unbounded period of time for its inputs to agree. When used in a ring FIFO with locked tokens, the voltage on all nodes is always moving towards new voltage. This C-element is unique in that it does not have any series transistors. Therefore, it produces more current per unit of input capacitance than any other C-element implementation (35).

Performance — When the inputs are in phase, the input inverters work together and are weakly impeded by the keeper inverter, because the inputs to the keepers inverters are switching as they try to resist the voltage change.

[1] If an input to the C-element is 'asserted,' it means that the input is changed since the output last changed.

When the inputs are out of phase, the output settles at a voltage removed from the rail. This results in shorter gate delays when the final input asserts because the output potential must then be charged over a smaller voltage range.

The degree to which the output is removed from the rail depends on how far out of phase the inputs are. The output potential of the gate when the last input asserts effectively encodes timing or token separation information.

Predictability — There are no internal nodes between the rails and the output. Internal nodes store charge that could be dumped onto the output and cause unexpected phase shifts.

Figure 4.3 shows how a number of Analog C-elements are connected to make a ripple FIFO. To make a ring simply connect $Rout$ and $Aout$ from the last stage to Rin and Ain in the first stage.

4.3 The Separation Plot

Figure 4.4 shows the closed loop FIFO stage delay vs temporal separation of the inputs. I refer to this plot as a *Separation curve*. The Charlie plot used in previous studies (9; 44) characterized a FIFO stage with an **open loop** delay versus temporal input separation characterizations to predict their FIFO's behavior. The open loop characteristics of the Analog `Micropipeline` stage fails to predict the behavior of a FIFO built from it. The open loop Separation curve for a FIFO stage built from a single Analog C-element shows a delay that monotonically decays to a constant value as the separation time increases.

The closed loop Separation plot is constructed by measuring the temporal separation time of a stage's inputs and measuring the resulting delay from a FIFO stage that is within a working ring.

The closed loop Separation plot shows that the delay of the stage is at a maximum when its inputs arrive simultaneously, S_{zero}. The delay then tapers to a nadir. As the temporal separation of the inputs then increases, there is a short delay hump where the delay increases by settling to a constant value. I refer to the separations that yield the delay at the nadir of the Separation curve and the top of the hump a bit further along the Separation curve as S_{z1} and S_{z2} respectively because they are the first and second points on the curve with a slope of zero. These separations are important for understanding the dynamics of the tokens as they enter the locked state.

4.3.1 Charlie effect

The Charlie effect manifest itself as increased delay as the temporal separation between inputs decreases. The Separation plot reveals at a glance the magnitude of the Charlie effect. It is

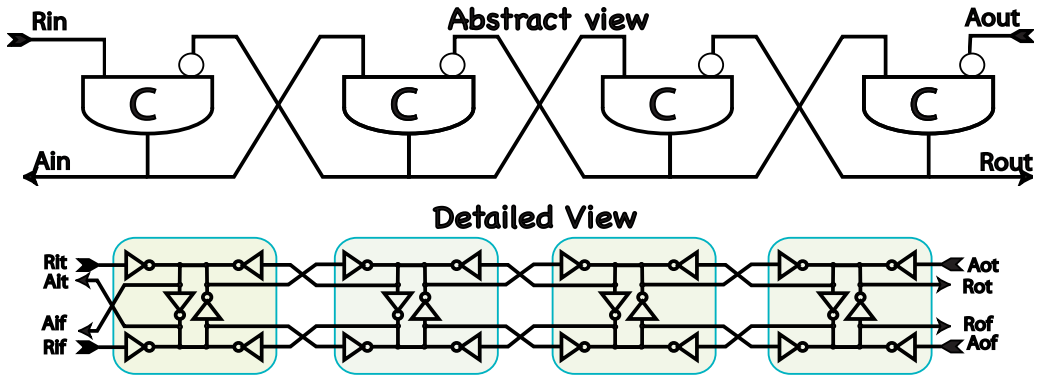


Figure 4.3: Four Stage Micropipeline

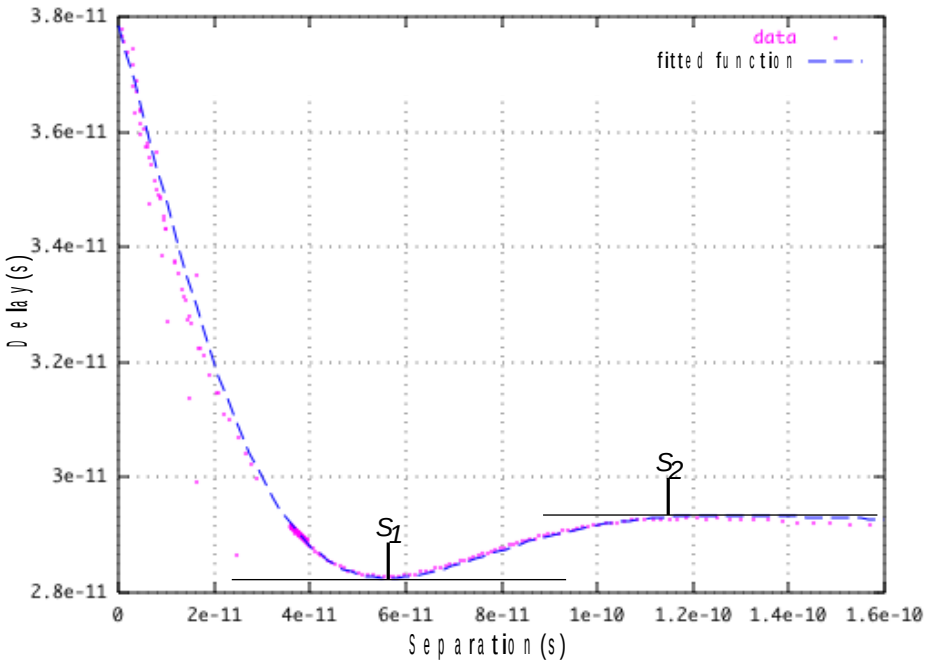


Figure 4.4: Delay vs. Separation with fitted curve

simply the difference between the peak delay when the inputs arrive simultaneously and the delay when the inputs are widely separated. The Separation plot also shows the range over which the Charlie effect acts. This separation is demarcated by the value of S_{z2}.

4.3.2 Drafting effect

Assume a token limited FIFO. In a FIFO constructed from Analog `Micropipeline` control, the transition that effects a move is always arrested from reaching the supply or ground by the Acknowledging signal changing the value on one of the inputs. A signal's transition time in CMOS is much longer than the delay through the gate that drives the signal. A *Request* is processed by the next stage and *Acknowledge*d before the *Request* signal fully transitions in a `Micropipeline` built from Analog C-elements. In a hole limited FIFO the same explanation holds except the roles of the *Request* and *Acknowledge* signals are reversed.

The voltage transition effecting a data movement reaches the same potential and incurs the same delay independent of occupancy or the rate at which the control is exercised.

In (44) a third axis was added to the Charlie Plot to reflect the drafting effect. The drafting effect resulted in reduced delay when the FIFO stage was exercised at a high frequency. The drafting effect is caused when the transition effecting a data movement is arrested from completing its full transition before being driven in the opposite direction. The drafting effect causes tokens to bunch and counters the spreading effect necessary for token lock. The analog C-element doesn't exhibit the drafting effect.

4.3.3 Separation curve character

The nature of the Separation plot for an Analog C-element is described by a cosine within a strongly attenuating exponential. I offer an explanation for the shape shortly. A curve described by Equation 4.4 fits the separation versus delay data.

$$D = D_{hold} + (D_{zero} - D_{hold}) \times e^{\frac{-S}{\delta}} \times \cos \frac{2\pi \times S}{\tau} \tag{4.4}$$

Where D_{hold} is the delay for a stage when the temporal separation of its inputs is the *holding separation*. D_{zero} is the stage delay when the inputs arrive simultaneously. These values are all represented in Figure 4.5. δ is the attenuating time constant for the exponential which describes the separation time required for the delay to decrease by a factor of e^{-1} of the difference between the peak and D_{hold}. τ is the period of the sinusoid. The hump in Figure 4.5 is modeled by the the cosine in Equation (4.4) increasing between π and 2π. The location of the nadir and the peak of the hump are found by finding the first two values for S that force the derivative of Equation (4.4) to equal zero.

The derivative of Equation (4.4) is:

$$\frac{dD}{dS} = -\left(D_{zero} - D_{hold}\right) \times e^{\frac{-S}{\delta}} \left[\frac{1}{\delta}\cos\frac{2\pi \times S}{\tau} + \frac{2\pi}{\tau}\sin\frac{2\pi \times S}{\tau}\right] \tag{4.5}$$

When Equation (4.5) is set to zero, its solutions are found at:

$$S_{zi} = \frac{\tau}{2\pi}\arctan\frac{-\tau}{\delta \times 2\pi} + i \times \frac{\tau}{\pi} \qquad i\text{=0,1,2...} \tag{4.6}$$

Figure 4.4 shows the data used to plot Figure 4.5, fitted with a curve described by (4.4) with the values of 29.25ps and 142ps used for the constants δ and τ respectively. D_{hold} and D_{zero} are measured from Figure 4.5.

The dynamics in the decaying sinusoid of the Separation curve are understood by looking closely at a token movement. When the inputs to a first FIFO stage agree, the output is driven towards its new binary value, signaling the consumption of a token. This new value acts as a request signal to a second stage in the FIFO. If the second stage is empty, it consumes the token and sends an Acknowledgment back to the first stage. The Acknowledgment causes the inputs to the first stage to now disagree. The output voltage of the first stage is divided. The inverter connected to the $Request$ input to the first stage and a keeper inverter are maintaining the previous binary output value, while the inverter connected to the $Acknowledge$ input is attempting to drive the output in the opposite direction.

The output voltage of the second stage dips to the voltage divided value one stage latency later than the first. Its inputs differ for the same reason. The inverter connected to the $Acknowledge$ input of the first stage becomes weakened slightly as it is controlled by the output voltage of that second stage. The resistance between the rail and the output of the inverter controlled by this $Acknowledge$ input in the first stage then increases and changes the potential of the voltage divided output value of the first stage, moving it closer towards the full rail voltage of the binary value the C-element is maintaining. This new voltage divided value, in turn, has a further effect on the voltage divided value of the first stage's two neighbors, but this effect is diminished because the voltage difference is less. This diminished effect on the two neighbors causes the output voltage to once again adjust. Assuming that the gate's logical value does not change, then these adjustments continue with a period described by τ in (4.4). The magnitude of each adjustment decreases exponentially as described by δ.

Because the exponential envelope of the Separation curve so strongly attenuates the sinusoid, only S_{z1} and S_{z1}, the first two solutions of Equation (4.6), determine the conditions necessary for tokens to lock in a FIFO ring. The tokens lock in regions of the Separation plot with a negative slope on the Separation curve. If the oscillation was not so strongly attenuated then additional locking states would exist beyond S_{z2}. The curve is essentially flat beyond S_{z2}.

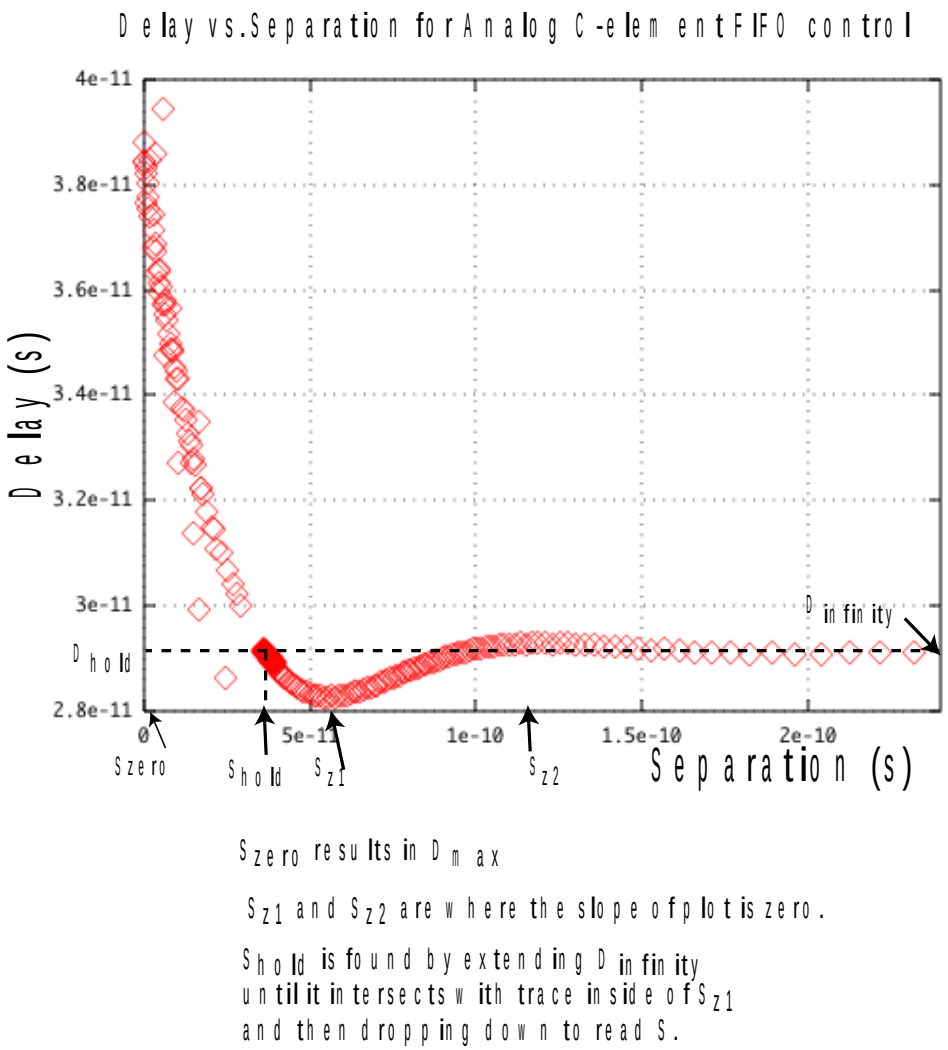


Figure 4.5: Delay vs Separation plot

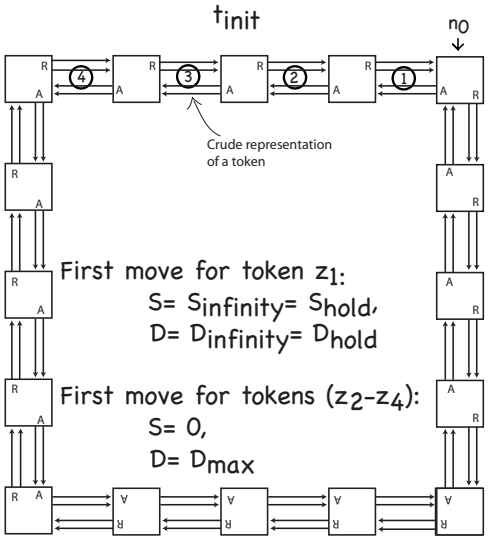


Figure 4.6: Initial state of FIFO ring

The characterization of the delay response of the FIFO stage to any ordering and temporal separation of inputs enable me to predict whether some number of tokens will lock in a FIFO ring of some size.

4.4 The Locking Mechanism

This section details the conditions necessary for a ring of N stages to achieve the locked state with T tokens and H holes. Note that $T + H = N$. Tokens are individually referred to as $\{z_1, z_2, z_3...z_k\}$.

A FIFO ring locks with some range of token occupancies. The lower limit to this range is found when the FIFO ring is token limited, the upper limit is found when the FIFO ring is hole limited. The variables T_{min}, and T_{max} denote these values. It is sufficient to derive the condition for lock in a token limited FIFO. As a direct consequence of the symmetry of the Analog C-element, T_{max} relates to T_{min} in a ring of N FIFO stages by the following relationship:

$$T_{max} = N - T_{min}. \tag{4.7}$$

4.4.1 Token locking condition

The locked state is achieved when each token in the FIFO ring is separated by an equal distance from the token before and after it. If the separation between two tokens is such that all tokens

are operating in a region of the separation curve with a negative and monotonic slope, then the tokens will converge to a locked state. Imagine three of the tokens in a ring. If the first and second token are separated by a shorter distance than the second and third, then the second token takes more time between moves than the third token because of the shorter separation. The third token closes the distance on the second token until their separation equalt the distance separating the first and second token.

The small hump between S_{z1} and S_{z2} presents the possibility that tokens can experience the same delay despite being separated from their neighbor by different distances. This suggest that tokens might lock into something other than the equally spaced state. Revisiting the previous example, assume that tokens one and two are separated by an amount that gives some delay and tokens two and three are separated by a different distance that yields the same delay. If the distance between tokens two and three is nudged a small bit closer because of a noise source then it will move even closer to the second token on its next move. The third begins to catch token two until the separations are equivalent. Tokens separated by different distances but with the same delay are in an unstable state.

For tokens to lock within a FIFO ring, all tokens must be operating in the trough of Figure 4.5 between S_{zero} and S_2.

Before locking, there must be two tokens separated by a distance greater than or equal to the distance between any two other tokens. I refer to the temporal separation of the inputs to a stage resulting from this distance as S_{worst}. The separation is described by the following equation:

$$S = |t_{ack} - t_{req}| \tag{4.8}$$

For the ring to enter the locked state, the following condition must be satisfied:

$$S_{worst} \leq S_{z2} \tag{4.9}$$

When this condition is met, then all stages are operating in the trough of Figure 4.5 that is found between S_{zero} and S_{z2}. Once this condition is met the ring eventually locks.

Assume all tokens T=$\{ z_1, z_2, z_3 ... z_k \}$, are initially bunched in adjacent stages, N=$\{n_k, n_{k-1}, n_{k-2} ... n_1 \}$ shown in Figure 4.6.

4.4.2 Holding Separation

If tokens are initialized in adjacent stages, they start to separate away from each other until they reach a certain distance from their predecessor. If the tokens fail to lock, then the tokens move around the ring in a pack, all tokens separated by this distance except the first and last token which are separated by a greater distance. For the tokens to lock, this distance between the first

and last token after the tokens have spaced themselves must satisfy the condition expressed in Equation (4.9).

To derive the relationship that expresses whether T tokens will lock in a ring of N stages, I first need to find the distance that tokens naturally separate in a free running FIFO ring. I call the temporal separation of the inputs to a FIFO stage resulting from this separation to be the *holding separation*. The delay of a FIFO stage operating at the *holding separation* is called the *holding delay*. S_{hold} and D_{hold} signify these values in later equations. With these values, the separation between the $Request$ and $Acknowledge$ signal is calculated for the worst case separation. This worst case separation is tested against the condition in Equation (4.9).

Initially the distance between the first and last token is great. Assume the separation is at a value greater than S_{z2} in Figure 4.5. Separations greater than S_{z2} have the same constant delay. Once moving, the second token separates from the first token until its delay is equal to this constant delay. If it separates any further, then its delay decreases and becomes less than the delay of the first token. This means that the holding separation is the separation less than S_{z1} that produces the same delay as a token separated from its predecessor by a very large distance. The remaining tokens separate from the token they follow to the holding separation by this same mechanism.

The holding separation is found by extending the line defining the delay encountered by a token separated from its predecessor by a value greater than S_{z2} until it intersects the trace of Figure 4.5 inside of S_{z1}. Drop a line down from this intersection to find the *holding separation*.

4.4.3 Token locking formulation

Assume that tokens initially start in a bunch since this condition is furthest from token lock. Also assume the tokens separate to the holding distance. The instant of a $Request$ from the first token and the instant of an $Acknowledge$ from the last token to the same stage referenced from some point in time equals a function of T, N, and D_{hold}. The magnitude of the difference of these two expressions yields the worst case separation. This difference substitutes into Equation (4.9) to predict whether the tokens lock.

If the separation distance between two tokens is known, then the time between two events produced by a stage is:

$$P = 2D + S \tag{4.10}$$

This equation is used to find the period between tokens entering a new stage when separated by the *holding separation*.

To determine this separation at stage n_i, we start a test clock when stage n_{i+1} produces an event signaling the consumption of the first token. This instant is illustrated in Figure 4.7. This

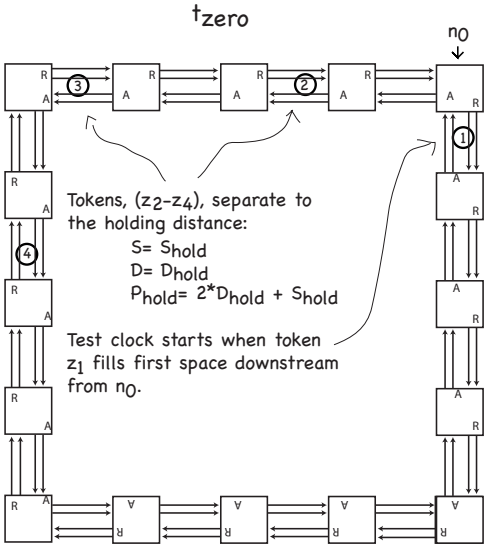


Figure 4.7: Test clock starts

is the reference point in time used to measure the instant the *Request* and *Acknowledge* signals occur.

If tokens $\{z_2 ... z_k\}$ are separated by the holding distance then stage n_{i+1} consumes or takes the last token and generates the acknowledge event sent to stage n_i at:

$$t_{ack} = P_{hold}(T - 1). \tag{4.11}$$

This instant is illustrated in Figure 4.8.

The second input that defines S_{worst} is the request to stage n_i that is generated after stage n_{i-1} consumes the first token. This event occurs at:

$$t_{req} = D_{hold}(N - 1). \tag{4.12}$$

This instant is illustrated in Figure 4.9.

For the tokens to converge to a locked state, the difference between these two events must be less than S_{z2}.

$$t_{req} - t_{ack} < S_{z2} \tag{4.13}$$

$$(N - 1) \times D_{hold} - (T_{min} - 1) \times (P_{hold}) < S_{z2} \tag{4.14}$$

$$(N - 1) \times D_{hold} - (T_{min} - 1) \times (2 \times D_{hold} + S_{hold}) < S_{z2} \tag{4.15}$$

When S_{worst} is less than S_{z2} the first token accelerates towards the last token when the separation between the first and last token is reduced to less than S_{z2}. The separation between

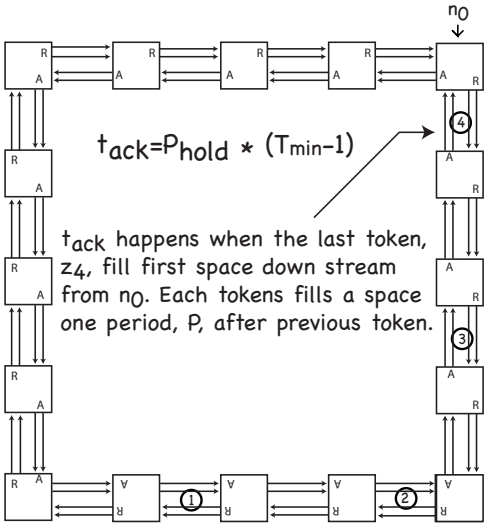


Figure 4.8: Time of acknowledge for S_{worst}

the first token and the second token then extends because of the first tokens decreased delay. The resulting increased delay between the first and second token makes the next move of the second token be effected with less delay. This causes the separation between the second and third token to extend and the third token is accelerated. The tokens then jostle and converge to a stable point where they are all operating at the same separation, and are therefore locked.

Solving this equation for T_{min} and then remembering the relationship found in Equation (4.7) provides the solution for the full range of token occupancies that result in token lock for a ring FIFO built from Analog C-element control.

$$N - \frac{D_{hold}\times(N+1)+S_{hold}-S_{z2}}{2\times D_{hold}+S_{hold}} > T > \frac{D_{hold}\times(N+1)+S_{hold}-S_{z2}}{2\times D_{hold}+S_{hold}} \tag{4.16}$$

4.5 Visuals

This section contains signal traces obtained from HSpice simulations and plots using data culled from HSpice simulations.

Figure 4.10 shows a request signal from a FIFO stage within a FIFO ring of 27 stages with 8 tokens. Arrows point to the transition effecting the move of the eighth token after its first four trips around the ring. Notice the separation between the transitions that signal the movement of the first and eighth token shrinks each time the tokens come around the ring. By the fourth trip around the ring, the tokens are locked into the equidistant pattern.

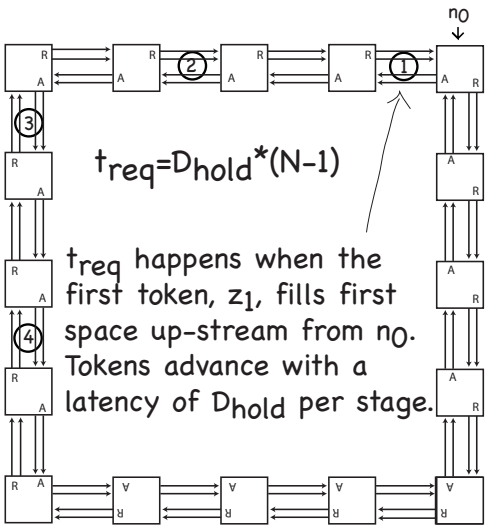


Figure 4.9: *Request* time

Figure 4.11 shows the *Request* signals from each of the four stages in a four stage FIFO ring with two tokens when locked. Each window holds the true and complement request signal from a stage. Notice that the signals in each successive window are advanced in phase by 90 degrees from the signals in the previous window.

Figure 4.12 shows eight plots. Each plot records the movements of a single token. The plot records the separation time at a stage for each of a token's first 250 movements. Eight traces are in the figure because there are eight tokens in the FIFO. The ring FIFO is composed of 24 stages. Notice that the separation times at a stage for the initial movements of the first token are large. The last seven tokens are moved when there is little time separating the inputs to a stage because they are initialized in adjacent stages. Notice the curves have the same character as any simple feedback control system has towards a step response. Initially the target is overshot by a large amount and then there is a large amount of ringing about the target while the controlled variable converges.

In the simulations used to draw Figures 4.12 and 4.13, one hundred token transfers, or moves, required about 7.5ns to 10ns.

Figure 4.13 is similar to Figure 4.12. Instead of plotting the separation times for each move of the eight tokens in a ring of one size, Figure 4.13 plots the separation times of the initially leading token in five rings of different size. The shorter rings converge faster for two reasons. First, the tokens have less distance to spread, but also they operate on a steeper portion of the Separation curve. A control system with a high gain exhibits ringing around the value it tracks as it locks. Systems with less gain exhibit a dampened response. The shorter rings converge to smaller separations. The magnitude of the slope of the separation curve increases as the

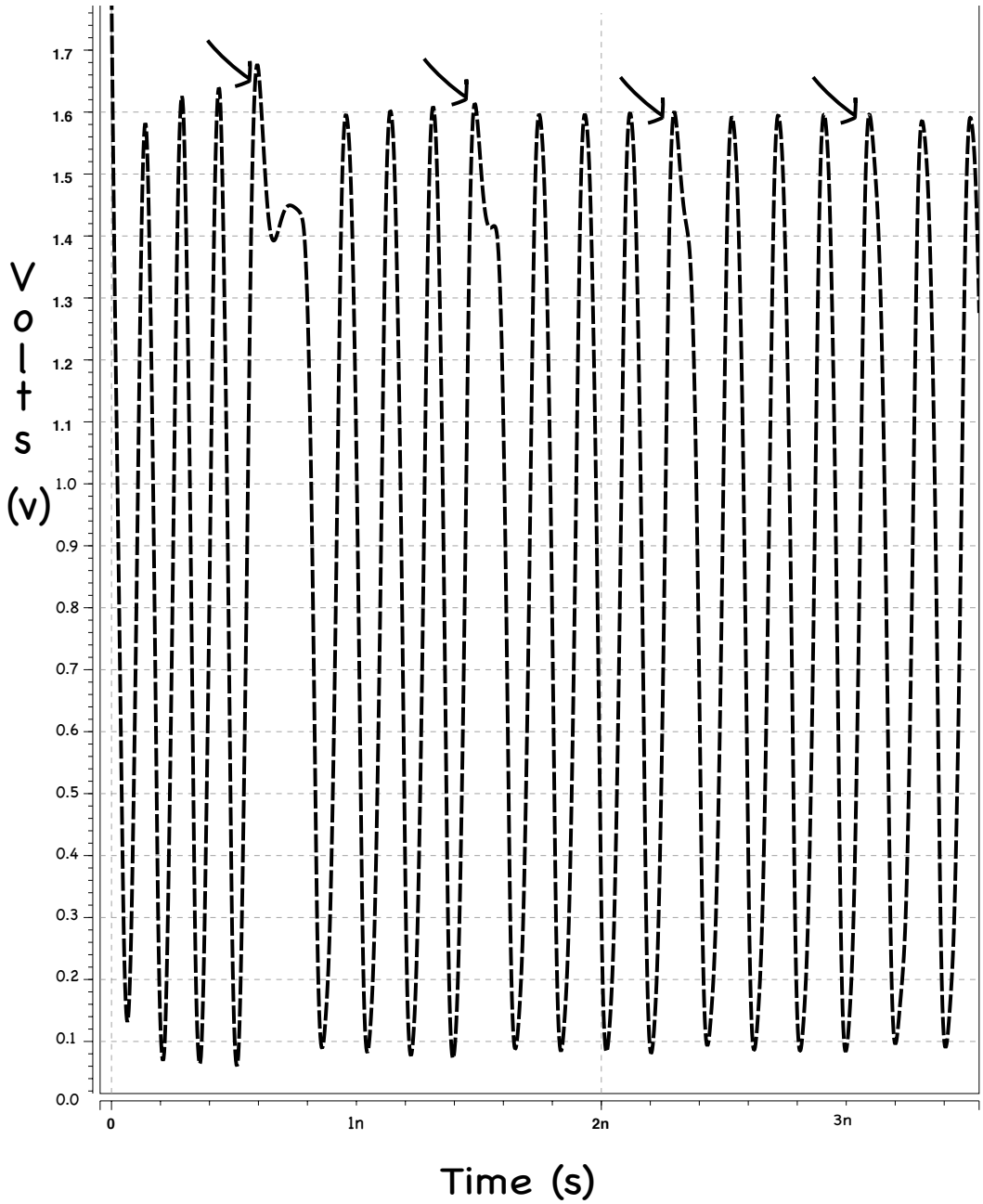


Figure 4.10: A *Request* signal in a ring of 27 FIFO stages with 8 tokens

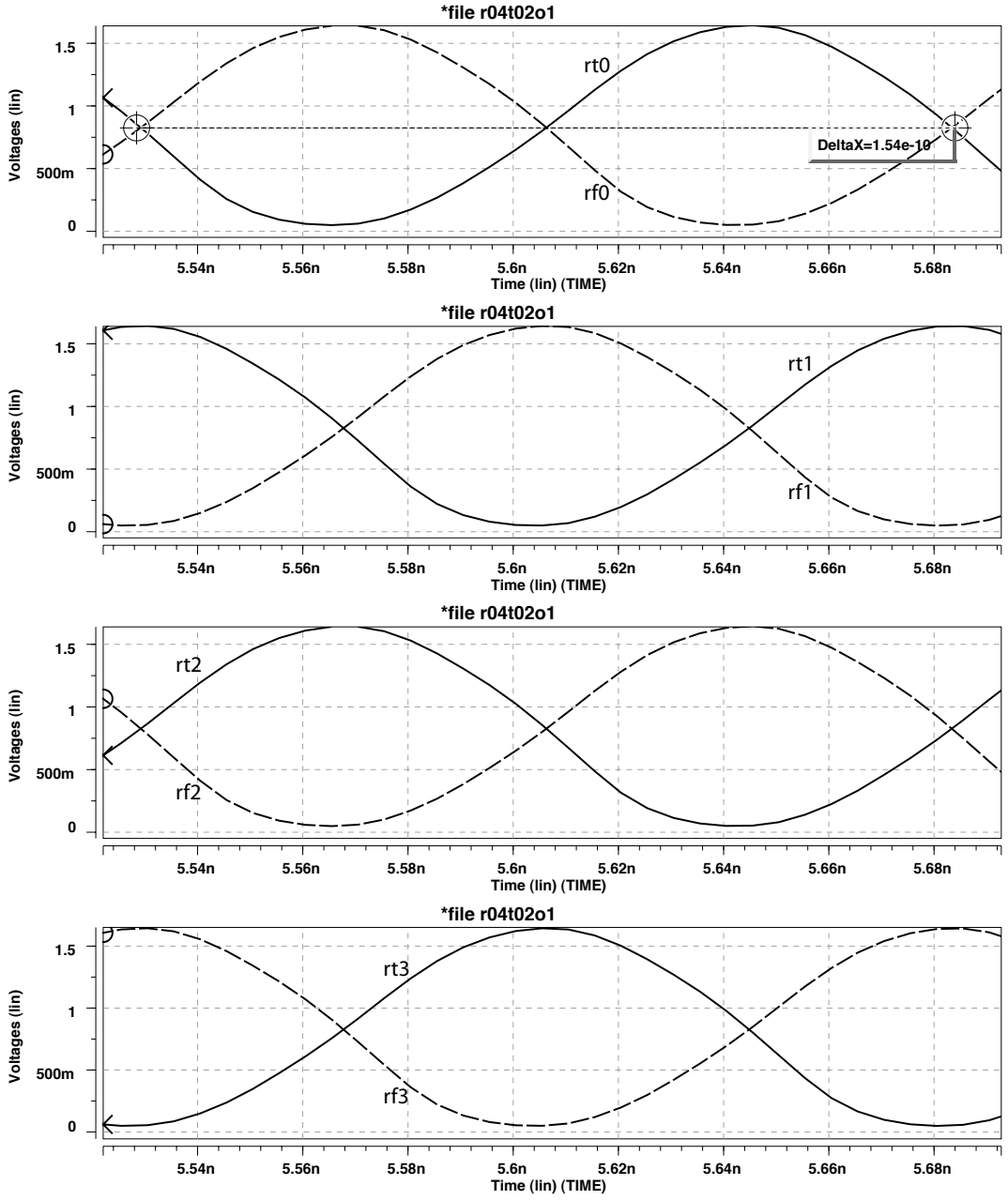


Figure 4.11: The 8 handshaking signals in a four stage ring FIFO with two tokens

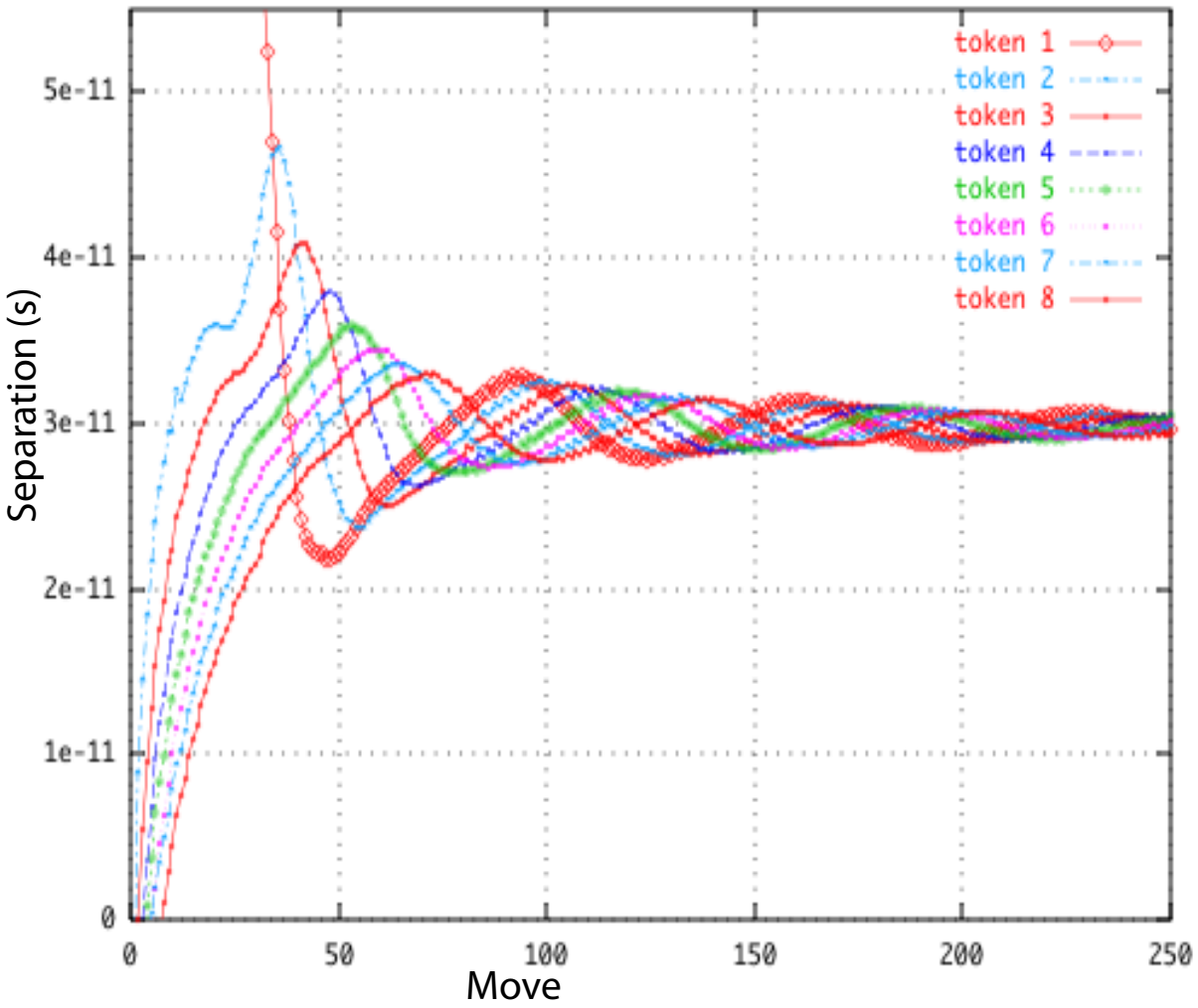


Figure 4.12: Token separation vs move number for the eight tokens in a 24 stage FIFO

separation approaches zero. The magnitude of the slope of the Separation curve is the loop gain of our control system. A token moved out of place by a noise source is corrected quicker if the tokens operate on a steeper part of their Separation curve. The slope at any point on the Separation curve is found from Equation (4.5).

Notice that the first token in the 29 stage FIFO ring fails to lock. The separation distance between the first and last token is on the constant delay portion of the Separation curve just to the right of S_{z2} and so it will not lock. Notice that no rings lock with separations between S_{z1} and S_{z2}.

The trajectory of the first token in the 28 stage FIFO ring looks like it is about to flatten out just beyond the lock range at around 80 moves. Then the last token is delayed just enough so that the distance between it and the first token places the last token over the hump of the Separation curve. The first token is then delayed even less on the next move, causing the separation between the first and last token to close even further. A similar action trickles back through the pack of tokens and the tokens eventually lock.

4.5.1 Relationship between Separation and Phases

I observe that the relative phase of the signals found at the output of adjacent stages can be expressed in two ways:

$$\phi = \frac{\text{tokens}}{\text{stages}} \times 180° \tag{4.17}$$

and

$$\phi = \frac{D}{Period} \times 360° \tag{4.18}$$

where

$$Period = 4 \times D + 2 \times S \tag{4.19}$$

Where D is the stage delay or latency and S is the temporal separation of the inputs to the FIFO stage.

Equations (4.17), (4.18), and (4.19) yield a relationship between the stages and tokens, and the resulting separation and delay in the those stages:

$$\frac{S}{D} = \frac{stages}{tokens} - 2 \tag{4.20}$$

Equation (4.4) also expresses delay as a function of separation. Substituting this equation into 4.20 yields:

$$\frac{S}{D_{hold} + (D_{zero} - D_{hold}) \times e^{\frac{-S}{\delta}} \times \cos\frac{2\pi \times S}{\tau}} = \frac{stages}{tokens} - 2 \tag{4.21}$$

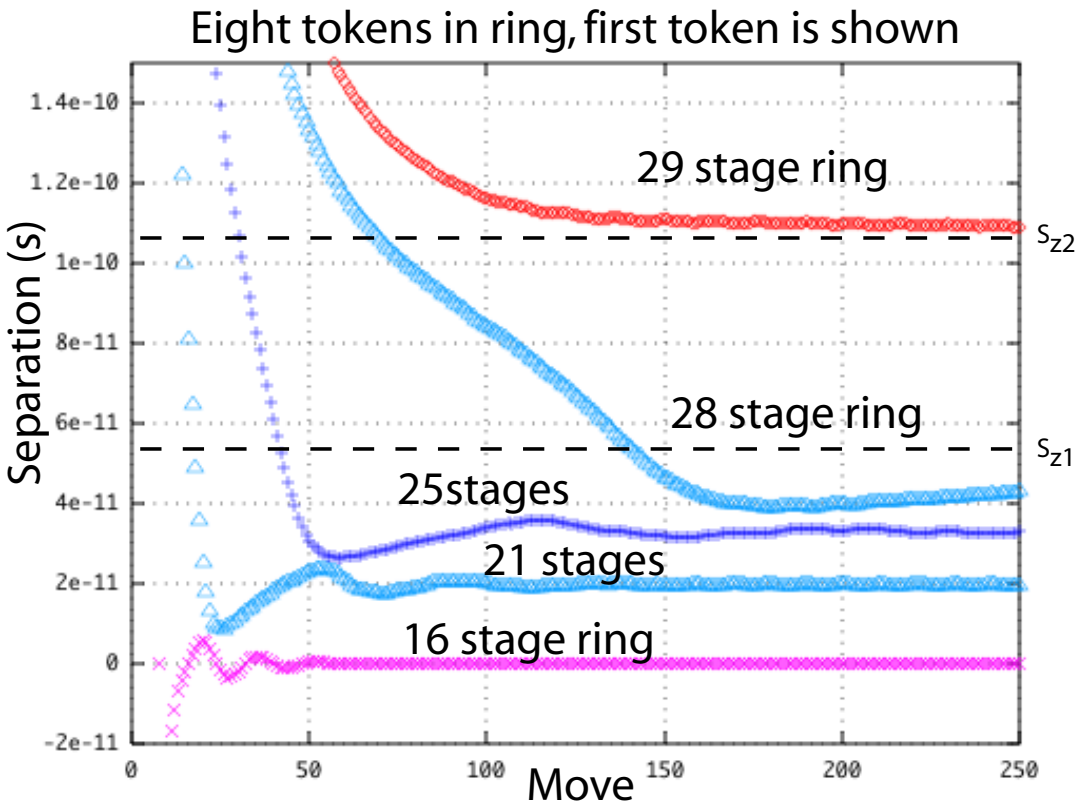


Figure 4.13: Token separation vs. move number for first token in ring.

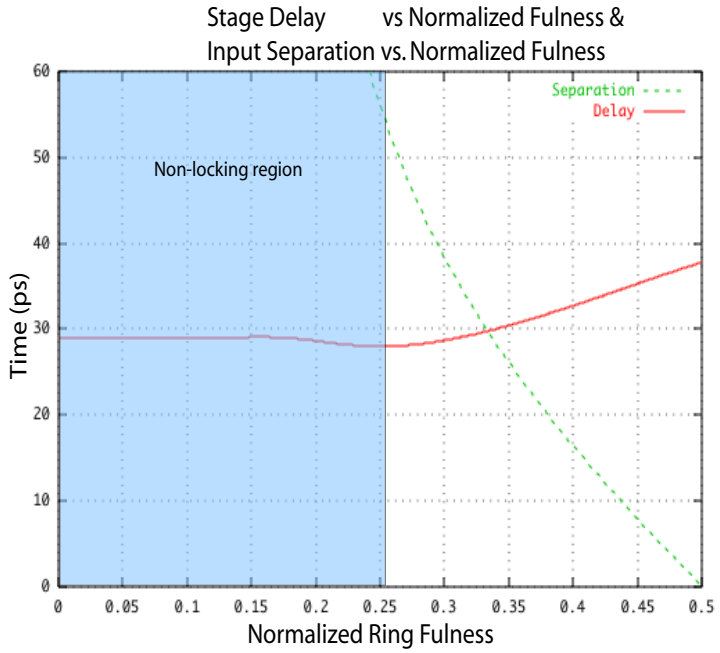


Figure 4.14: Separation and stage delay vs. fractional fulness of ring

Given some number of stages and tokens, the only variable remaining in this equation is separation time, S. Equation (4.20) assumes that the tokens are locked. The tokens will lock if the separation given by this (Equation 4.21) is less than S_{z1}, which is calculated to be 55ps using the values from the curve fit that produced Figure (4.4).

Equation (4.21) yields the temporal separation between the inputs to a stage given the number of stages in a FIFO ring with a specified number of stages. Each separation within S_{z1} yields a unique delay. The period is found from this separation and delay from Equation (4.19). Equation (4.21) quickly generates data for a number of informative plots.

Using 1001 stages in Equation (4.21) and sweeping the number of tokens from zero to half occupancies into the FIFO allows me to plot period, stage delay and temporal separation of inputs all against the normalized fulness of the FIFO ring. Sweeping over the complete range of occupancies is unnecessary because the functions plotted are symmetric about the 50% occupied point due to the symmetry of the Analog `Micropipeline` FIFO control stage.

Using 1001 stages as the representative FIFO achieves fairly continuous plots and sweeps through a large number of different phase relationships between stages.

Notice from Figure 4.15 that in the locking region, the period varies from 151ps to 208ps. A half-occupied FIFO ring results in the shortest period, even though its stage delay is greatest at this point.

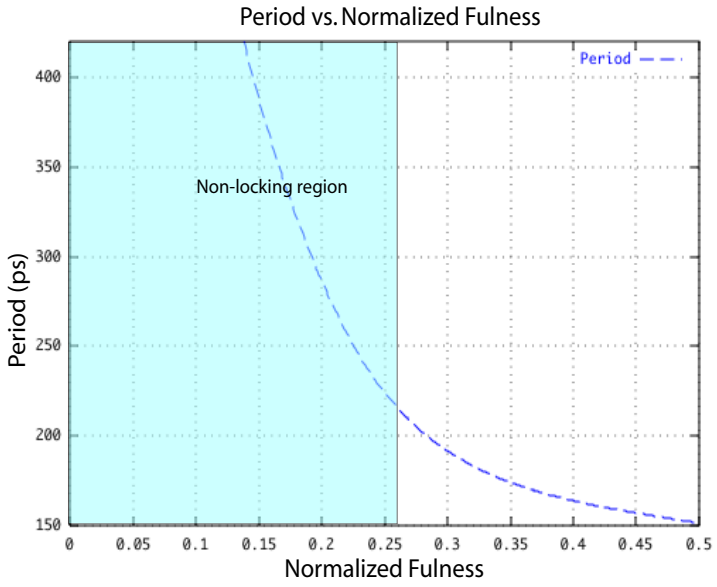


Figure 4.15: Period versus fractional fulness of ring

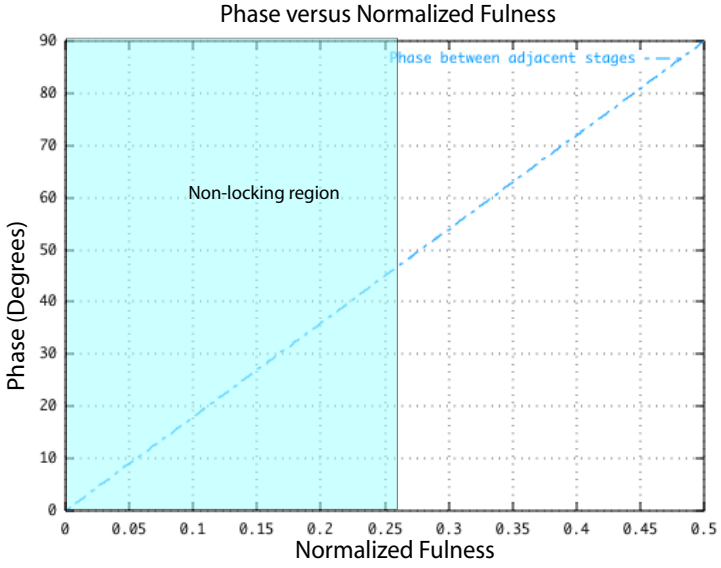


Figure 4.16: Phase between stages versus fractional fulness of ring

Figure 4.16 shows that the phase between adjacent stages increases linearly up to 90 degrees versus fulness.

The HSpice transistor model employed uses thirty parameters. Because of the simplicity of the Analog C-element and the elegance found in a `Micropipeline` ring composed of this gate, a model for the ring with HSpice accuracy can be built with only four variables: tokens, stages, τ and δ. A script requiring seconds replaced time consuming HSpice simulations to accurately find the complete range of periods and stage delays and temporal input separation times for any combination of tokens and stages that result in a lock.

The precise results yield certainty and predictive power. It gives confidence to a designer. The increased understanding also provides a springboard to explore topologies derived from the initial design shown in Figure 4.2 that amplify an aspect of the FIFO control while compromising some other aspect.

4.6 Derived Topologies

This section introduces a number of topologies derived from the basic Analog C-element shown in Figure 4.2 pioneered for specific applications.

4.6.1 Voltage Controlled Oscillator

In this section the Analog C-element is altered so that its delay is a function of a control voltage supplied through an added input to the C-element. When these C-elements are connected in a ring, they form an oscillator whose frequency is a function of the control voltage supplied to the extra input. In other words, the ring is a voltage controlled oscillator (VCO).

Figure 4.17 shows an altered Analog C-element. Series NMOS transistors are included in the pull down paths of the drive transistors. The extra NMOS transistors are controlled by a single control voltage, *cnt*. The oscillation frequency is plotted as a function of this control voltage in Figure 4.18. A $3\times$ adjustable frequency range is achieved using this simple speed control. The frequency is linear with the control voltage over a nearly 400mV range. This is a desirable property in a VCO.

The linear section of the frequency versus control voltage plot is made increasingly more gradual as the keeper inverters are sized larger with respect to the other transistors. The tradeoff is the range of frequencies that can be attained.

All transistors, PMOS and NMOS, are sized identically in the circuit used to generate Figure 4.18. In a typical process the NMOS transistors are approximately 2.5 more conductive than an equivalently sized PMOS transistor. The NMOS series transistor makes the conductivity of the pull-up and pull-down path nearly identical.

The series transistors control the amount of current that the other NMOS transistors can source. They are not used to implement logic and do not contribute unexpected phase shifts.

4.6.2 Maximized locking range

The analog C-element implementation shown in Figure 4.2 allows for the construction of FIFO rings that allow you to place events at different locations on a VLSI chip with a large range of phase relationships. Nevertheless the range of phases is limited because of the range of separations over which the delay of the analog C-element varies. This range is demarcated by the value S_{z2} in Figure 4.5. Here I present a C-element implementation that greatly extends this range.

The control shown in Figure 4.19 seeks to maximize the token occupancy range that results in token lock. Two Analog C-elements are overlapped. The C-element at the top of the figure receives its $Request$ signal a gate delay later than the C-element at the bottom. The C-element on the bottom receives its $Acknowledge$ signal a gate delay later than the C-element on top. The effect of this arrangement is to extend the portion of the Delay vs. Separation curve that has a non-zero slope. This in turn extends the locking range of the tokens.

When a FIFO ring constructed from this control is half-full, its period is 319ps. Hspice predicts that four tokens lock in a FIFO ring as long as 90 stages using this control. This allows you to pick the phase offset between stages to be within a few degrees. The cost of the increased range is double the hardware.

The delay vs. separation curve for this control is shown in Figure 4.20. The curve is fitted with the following function:

$$D = D_c + (D_{max} - D_c) \times e^{\frac{S}{\delta}}. \tag{4.22}$$

The value for D_c is 22ps. $D_{max}$is 82ps. The value for δ is 92ps.

On Figure 4.20a I included the line:

$$D = 80ps - \frac{S}{2} \tag{4.23}$$

This line represents delay-separation pairs that result in a period of 320ps. Notice that the Separation plot for the control shown in Figure 4.20b closely tracks this line for separations below 10ps. This suggests that when tokens operate on this fraction of the Separation curve, the token movements produce the same periods even when the tokens are knocked out of phase by a noise source. A future area of investigation could explore ways to shape the Separation curve by manipulating transistor sizes and interconnect lengths to make the separation curves linear over a large range of separations. This might add a degree of robustness in the face of environmental and switching noise.

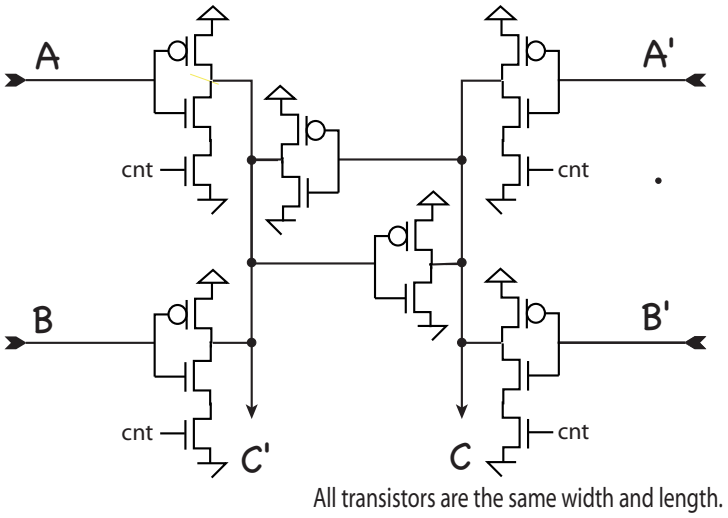


Figure 4.17: Analog C-element with speed control

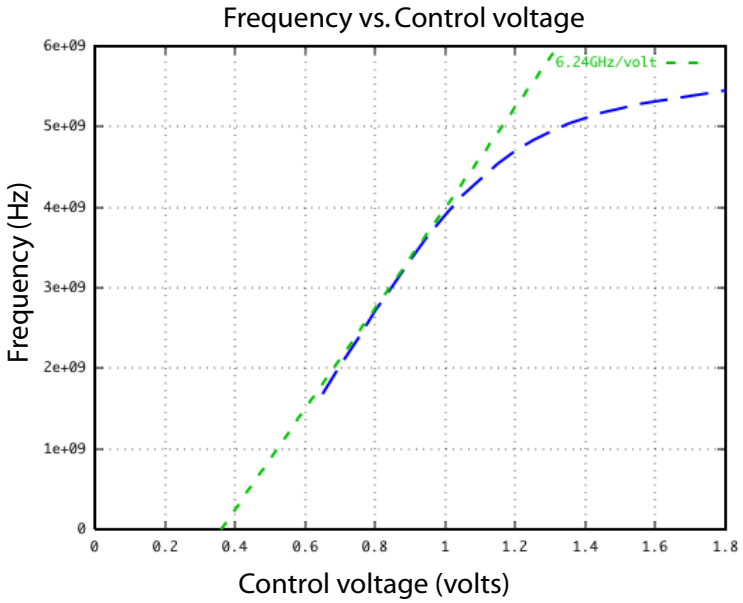


Figure 4.18: Speed control response

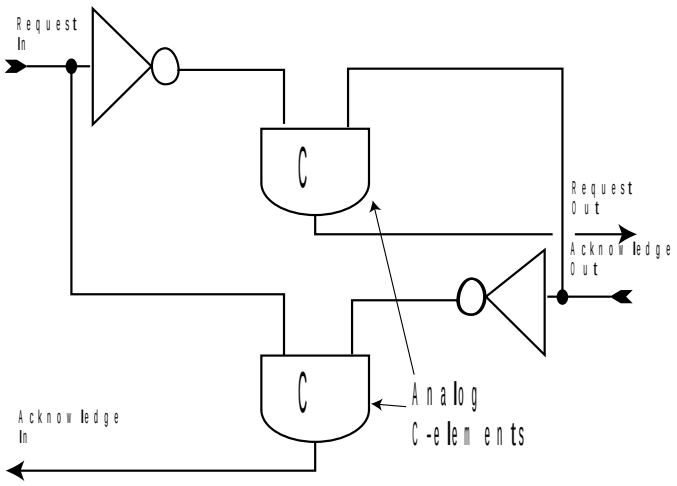


Figure 4.19: Overlapping Analog C-element

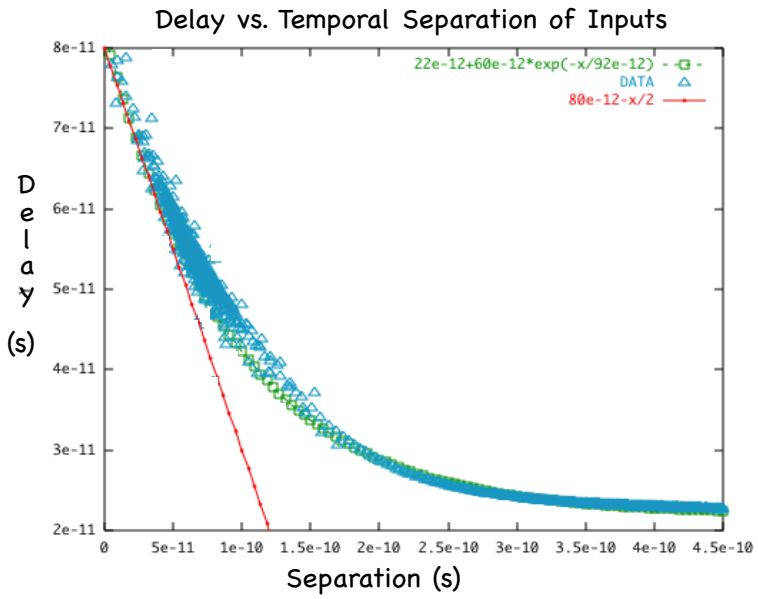


Figure 4.20: Separation curve for overlapping Analog C-element

4.6.3 Locking Different Rings

Assume two rings of analog C-element FIFO control. Each ring is initialized with the same number of tokens. Barring fabrication errors, both rings should produce signals of the same frequency and the various nodes in each ring should have similar phase relationships. However, the phase relationship between the nodes in the separate rings is unknown. This section discusses a method for bringing the nodes in the different rings into a deterministic phase relationship with respect to each other.

4.6.3.1 Synchronizing

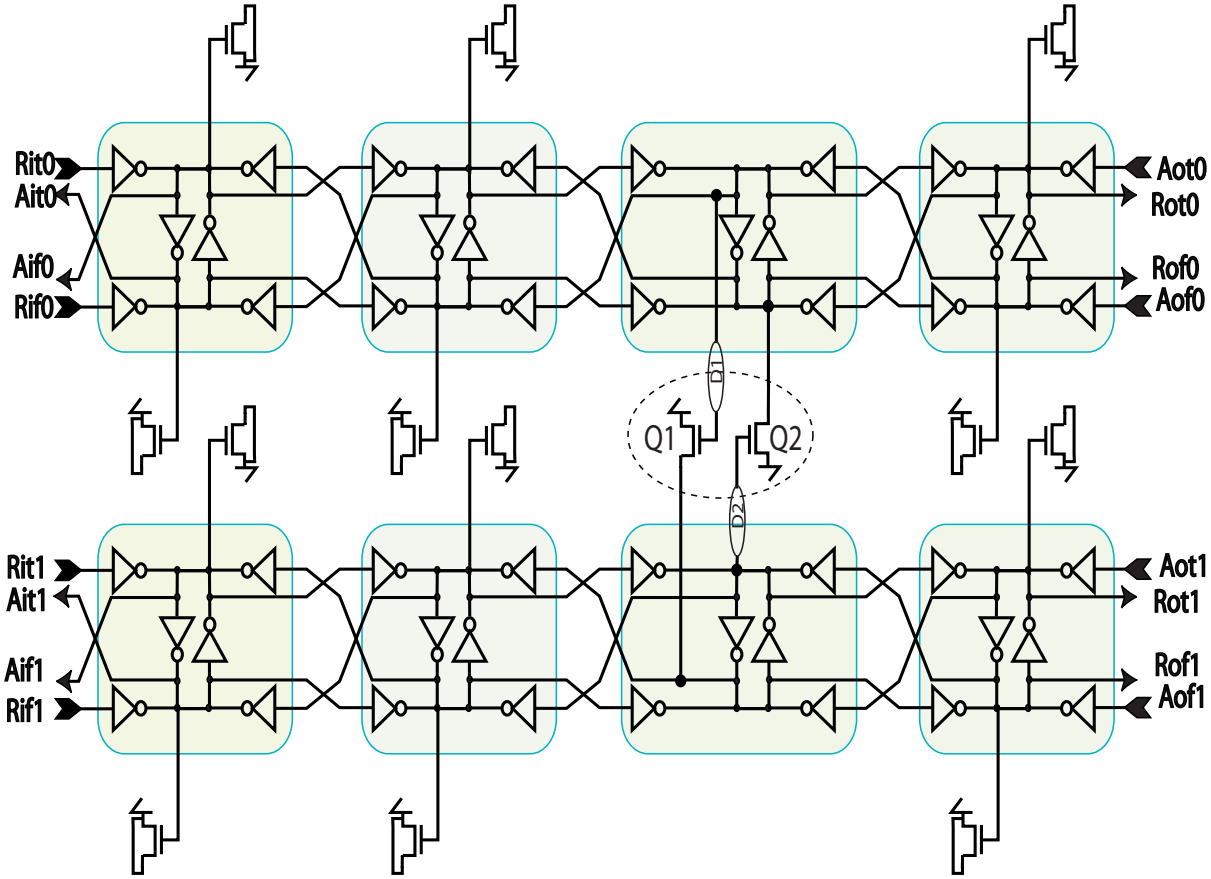


Figure 4.21: Synchronizing separate FIFO rings

Figure 4.21 shows two four stage FIFO segments from two separate FIFO rings. The third FIFO stages in both segments are coupled together by two NMOS transistors that are hi-lighted by a dashed halo. These two transistors phase lock the two rings by charge injection.

Assume that the nodes in the top FIFO section are advanced in phase with respect to the corresponding nodes in the bottom FIFO. The node connected to the gate of transistor $Q1$ then goes HI before the gate of $Q2$. $Q1$ couples negative charge onto the node that is complementary to the node connected the gate of $Q2$. This hastens the transition on that node, bringing the two rings closer to in-phase.

The transistors that are appended to the other nodes in the figure are used to model the capacitance provided by the gates and drains of transistors $Q1$ and $Q2$. Alternatively, these nodes can have other two-transistor phase-locking circuits identical to $Q1$ and $Q2$. This will bring the circuits into the locked relationship faster, though a single set of phase-locking transistors is sufficient.

The larger the phase-locking transistors are relative to the size of the transistors in the FIFO, the faster the separate rings lock. In Hspice simulations, two five stage rings phase lock in less than 5ns regardless of the initial phase offset when the phase-locking transistors are a 1μm wide and the size of the NMOS transistors in the FIFO are 4μm wide.

4.6.3.2 Interleaving events

Assume signals with a $45°$relationship are needed. Equation (4.1) suggests that these signals could be obtained by placing 2 tokens into a ring of eight stages. However, Equation (4.16) predicts that two tokens won't lock in an eight stage ring. A similar circuit to that shown in Figure 4.21 yields the necessary $45°$ without resorting to using another C-element implementation that compromises the high frequency attained by FIFOs built from simple Analog C-elements.

The circuit is shown in Figure 4.22. The gate of transistor Q_1 and the source of Q_2 anchor the two FIFOs together. The source of transistor Q_1 and the gate of Q_2 shift the relative phases of the two FIFOs. The phases of the two FIFOs become locked into a phase offset equivalent to the half the phase difference between the phase difference of the stages that the source of transistor Q_1 and the gate of Q_2 connect. If the connections that provide the phase shift connect to adjacent stages, then the phases from each FIFO interleave each other. In the case of two tokens in a four stage FIFO, the phases between the FIFOs separate by $45°$.

Two FIFOs that produce interleaved events provides a phase resolution that otherwise requires the overlapped Analog C-element in Figure 4.19. However, the gained phase resolution is achieved without compromising frequency.

4.6.4 Maximum Frequency

For many applications the 2FO4 gate delay sampling period provided by the analog `Micropipeline` control is adequate. Some applications require a higher frequency. This section describes

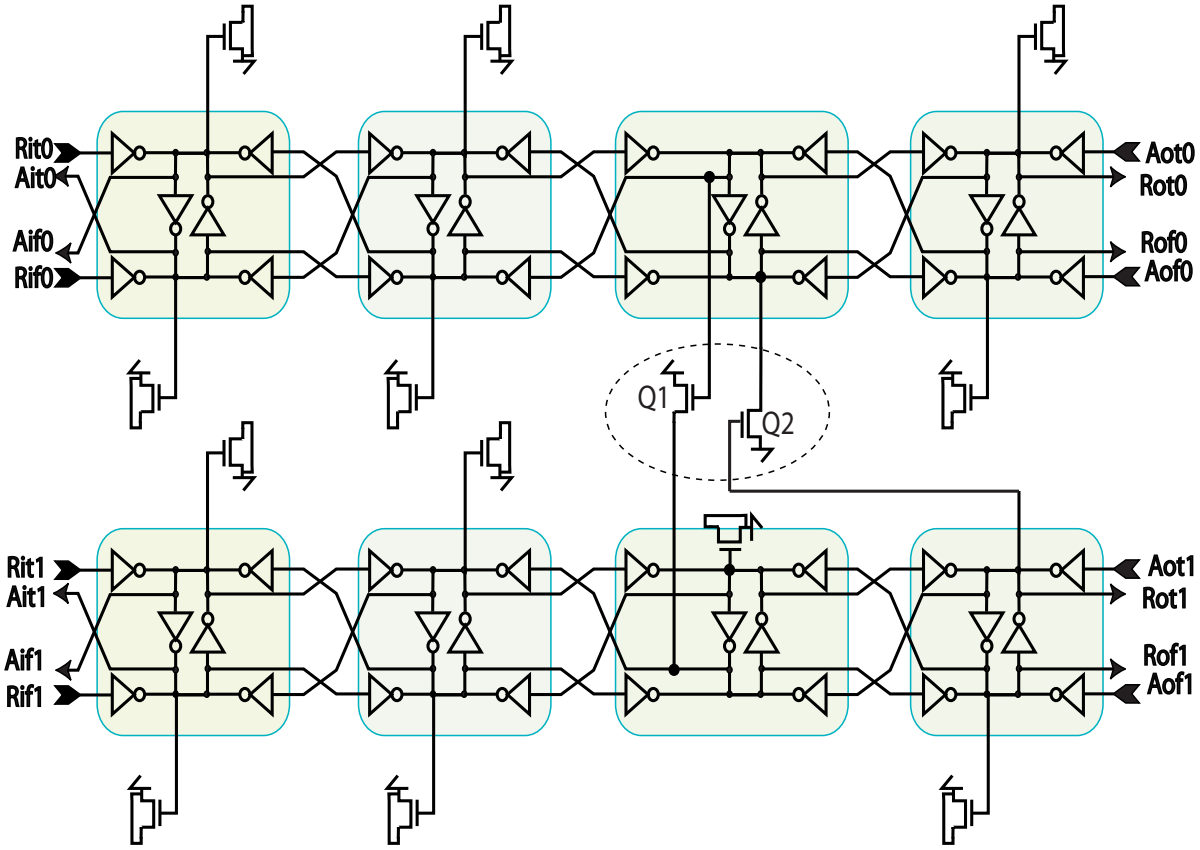


Figure 4.22: Interleaving events from separate FIFO rings

a method to double the frequency provided by analog `Micropipeline` FIFO control connected in a ring. I describe a method for generating differential signals that swing over the full supply voltage with periods equivalent to one FO4 gate delay.

Figure 4.23 shows some idealized waveforms that describe this circuit's functionality. The method requires first generating four evenly spaced phases. A four stage analog `Micropipeline` FIFO ring initialized with two tokens provides two copies of four evenly spaced phases. The four phases are shown as signals A to D in Figure 4.23.

The four phases A to D are used to generate four pulses E to H. Four NMOS and PMOS transistor pairs each generate one of these pulses from two of the initial four phases. These four pulses each control a single transistor in the stage that drive the double frequency node J. In practice the pulses are not so well confined to the periods suggested by the figure. Because of the long slew rates of CMOS circuits and the already high frequency it is difficult to confine the pulses to only a fourth of the base period.

A complementary signal is derived by doubling the circuitry shown and driving the transistors in the first stage by the complement values.

This method generates a signal with a period of 1 FO4. Figure 4.24 shows results from simulating this circuit in HSpice. The top two traces represent signals taken from the fours stage `Micropipeline` ring. The third trace shows one of the four HI pulses and the fourth trace is a LO pulse. HI pulses are more easily confined to approximately one fourth of the period than LO pulses. The bottom window shows the complementary 1 FO4 signals. The peak to peak signal swing is 1.4 V. The transistor values are as listed on the schematic. The signal swing amplifies to 1.8V peak to peak voltage after a single stage of inversion by making the circuitry shown in Figure 4.23 large relative to the amplifying buffer and keeping the amplifying buffer lightly loaded. Simulations used 0.18μm models.

4.7 Performance and Power

This section presents simulated results from FIFO rings built from the Analog C-element FIFO control shown in Figure 4.2. The results are presented for FIFO rings that are very aggressive and high performance. I explore the limits of operation for an Analog `Micropipeline` ring. These are not necessarily designs that I would hazard fabricating. Results are for a FIFO ring that sometimes breaks under reasonable transistor mismatch parameters. Margins can be added once the certainty of the process and the needs of the application are determined.

All HSpice simulations for this thesis were done with models extracted from the same 1.8V 180nm process. The FO4 delay for an inverter with PMOS/NMOS transistor ratio of two in this process is 75ps. A ring of three unloaded inverters oscillates with a period of 158ps.

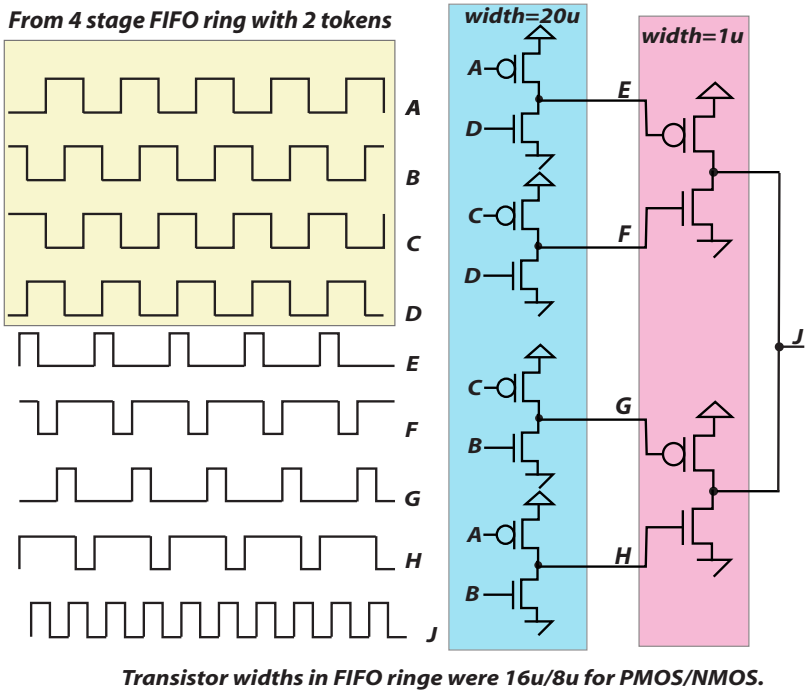


Figure 4.23: Frequency doubling circuit

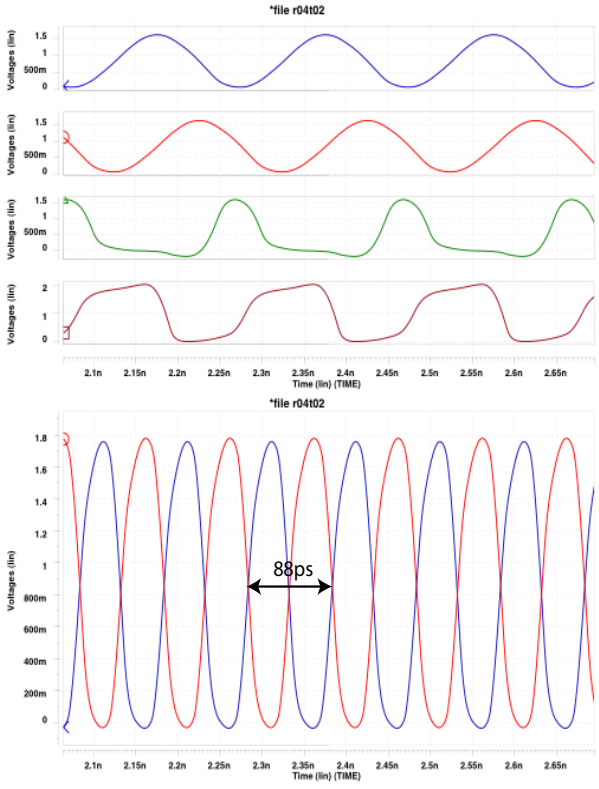


Figure 4.24: Signals from HSpice from maximum frequency circuit

Period — Because the stage delay increases as the Separation time decreases, the frequency of all locked rings is nearly the same. The cycle times for all rings that I simulated, when locked, fell between 153ps, when a ring is half full, and 208ps when 8 tokens occupied an eleven stage ring.

Voltage swing — The peak to peak voltage swing for all nodes in the rings when locked fell between 1.49 and 1.58V. One of the more compelling advantages to using handshake circuitry to distribute high frequency signals is the preservation of signal strength. A signal copied through inverters slowly attenuates. In handshake circuits, the signal must reach a certain amplitude before the receiver detects it. Because `Micropipelines` detect both rising and falling transitions a healthy voltage swing is certain.

Current — The average current draw per stage when the ring is locked is within a couple of percent of 0.8mA regardless of the number of stages or tokens. The peak to peak current draw per stage varies dramatically depending on the number of distinct phase in a ring. An eight stage ring loaded with four tokens has sixteen nodes that represent four copies of four distinct phases. If the same ring is loaded with two tokens, it would have two copies of eight distinct phases. A ring with a high number of distinct phases draws nearly constant current because the current needs of the gates in the various stages are evenly spaced through time. Figure 4.25 shows the relationship between the peak to peak current, or the magnitude of the AC current, per stage with respect to the number of distinct phases found in the ring.

The primary source of jitter is sudden current demands on the supply. Assume a design requires a four phase clock. The peak current requirement from an Analog `Micropipeline` is greatest in this case. Placing ten tokens in a 21 stage FIFO ring yields nearly the same phase relationship between stages, 85°versus 90°, but the peak variable current demanded from the supply is drastically mitigated.

Supply range — The rings operate under a wide range of supply voltages. Figure 4.26 shows the relationship between the frequency at which the ring oscillates and the supply voltage for a four stage FIFO ring with two tokens. The nearly linear relationship suggests that an Analog `Micropipeline` FIFO ring is well suited for use as a phase locked loop.

Usable signal — When the power supply is reduced, see Figure 4.26, in order to lower operating frequency or to reduce power, the resulting voltage swing on the output nodes is centered around the threshold voltage of the inverters that compose the Analog C-element. This obviates the analog circuitry used to restore the voltage swing to levels used by digital gates that are used in other high precision timing solutions (41; 47).

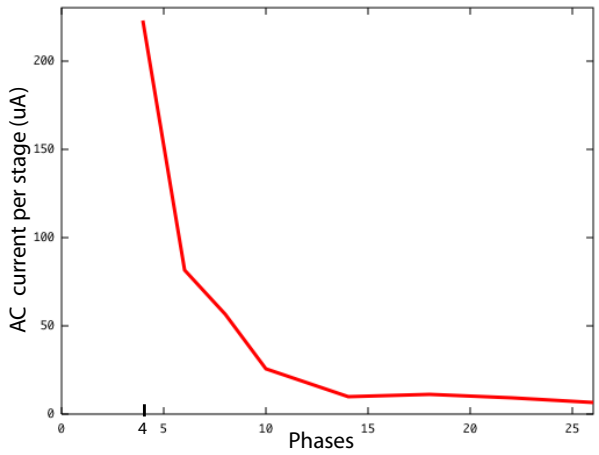


Figure 4.25: I_{pkpk} as a function of the number of distinct phases

Phase compensation — When the power supply is reduced, the various nodes in the ring retain their phase relationship. Extra phases that are derived using phase interpolation lose their timing relationship with adjacent phases when the power supply is changed. Here the number of degrees between adjacent phases remains constant even as the clock period changes.

4.7.1 Tolerance to fabrication variations

Table 4.1 shows how transistor mismatch affects the performance of the FIFO rings. The mismatch model chooses a threshold voltage for each transistor from a 1 sigma Gaussian distribution of 10mV and a 1 sigma Gaussian current matching distribution of 2%. These mismatch parameters are based on results from (28). HSpice was used to perform a thirty trial Monte Carlo simulation.

In the first row the FIFO configuration used is described in terms of the number of stages and its loading. Then the average period of the oscillations for that configuration over the thirty trials is listed. The following rows are split into three sub-columns. The first of the sub-columns list the available phases for that configuration. The phases are listed in the order in which they are encountered in the FIFO stages as you move around the ring in the same direction as the token movement. For example, for the configuration with three stages and two tokens we know there are six electrical nodes in the circuit. Each node carries a signal that is advanced in phase by some multiple of 60°from the node chosen as the reference. In the initial stage chosen as

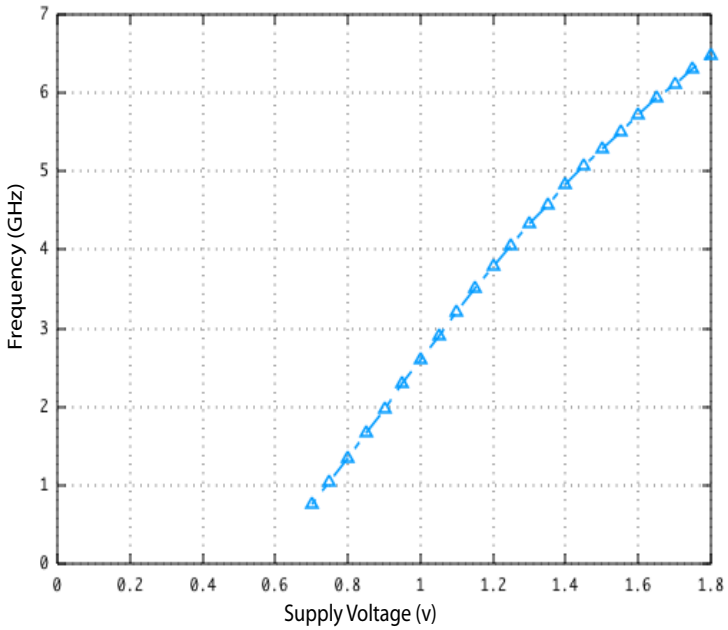


Figure 4.26: Frequency against supply voltage

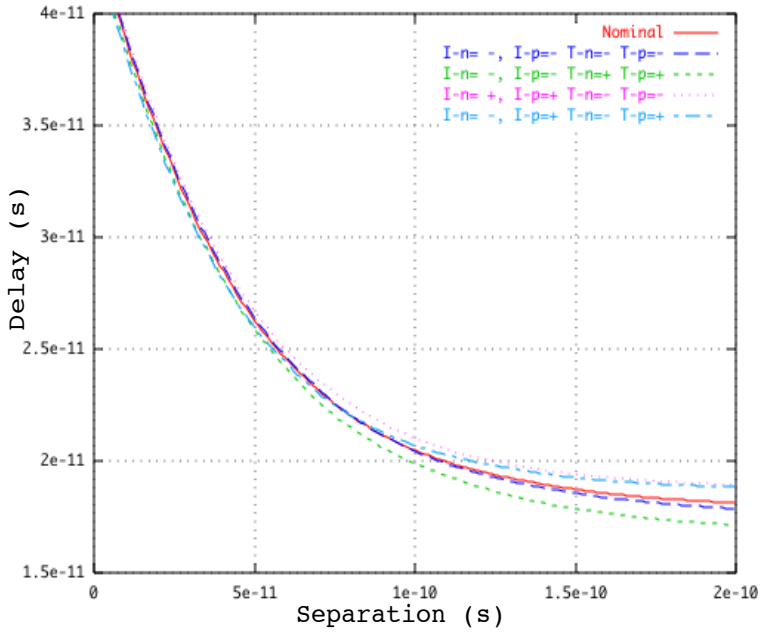


Figure 4.27: Open loop separation curve for Analog C-element with different extremes of transistor mismatch

reference, we expect to find the node that carries the reference phase and the node that produces a signal 180 °removed from the reference node. In the next stage we expect to find signals offset from the reference by 240°and 120°. The next sub-column lists the skew from the measured standard deviation phase expressed in degrees. The next sub-column translates that number into actual delay as expressed in seconds.

Of the 30 Monte Carlo trials for the fifteen stage FIFO ring, the ring failed to lock during one trial, and the ring failed to sustain oscillations for one trial. The results from those two runs are not included in the average deviation calculation for the data in Table 4.1. In the case where the ring failed to sustain oscillations, transistor mismatch caused one of the adjacent stages to respond sooner to a voltage transition from a stage than the other.

This simulation exposes the limits of dependable operation for this FIFO control in the presence of transistor mismatch. Increasing the size of the keeper inverters to 1.5 times the size of the input inverters yielded robust and locked oscillations for all thirty trials. The average period of the signals slows to 203ps with this alteration. Increasing the keeper strength stabilizes the oscillation and keeps the output from switching until the inputs have decisively changed state.

I define the instant of an event in the FIFO rings as when the differential output values cross. An artifact of this choice is that there is no skew between a signal and its complement. The phases in the FIFO rings are listed next to their complement in Table 4.1. These numbers suggest that significantly better phase resolutions can be achieved using `Micropipeline` FIFO rings than through the non-calibrated interpolation techniques discussed in Section 2.4.1. I offer three explanations for the high precision.

Bi-directional signaling. Because FIFO communication signaling is bi-directional, the furthest number of stages between any two nodes is half the number of FIFO stages. Distant nodes are coupled more tightly than single direction signaling such as that found in a DLL.

Separation curve convergence. Figure 4.27 plots the open loop separation curve for the Analog C-element under five different extremes of my chosen transistor mismatch model. Open loop separation curves do not have a hump but rather taper to some constant value. The open loop separation curve is easier to visually compare with other separation curves. Each trace represents a different extreme of threshold voltage and current factor to the NMOS and PMOS transistors. The legend lists the extremes chosen for each plot. Four letters each followed by a negative or plus sign (+/-) in the legend identifies the condition that generated the plot. 'I-n=+' means that the NMOS transistors for this simulation received current factor parameters on the more conductive extreme of the mismatch model.

'T-p=-' means that the PMOS transistors in this simulation received threshold voltage offsets on the later turn-on threshold voltage extreme of the mismatch model.

Notice that as the Separation distance between inputs shrinks, all five plots converge. When a FIFO ring is locked, it will be operating on the dynamic part of the Separation curve near the zero Separation point where these plots converge. This plot says that the locked rings are very insensitive to transistor mismatch when operating in this region. This helps to explain the small amount of skew between the different phases of the FIFO ring.

Threshold voltage variance. Monte Carlo simulations showed that the standard deviation of the threshold voltage for the inverter was 18mV while the standard deviation of the threshold voltage for the C-element was 12mV.

4.7.2 Jitter

Jitter is the timing uncertainty contributed from sources such as environmental or power supply noise. Power supply noise is contributed by the non-zero output resistance of the supply. Current demands cause the supply voltage to dip. This results in unexpected gate delays.

Two AC sources in series with the power supply model the effect of power supply noise. One AC source has a peak to peak voltage of a tenth of the supply voltage, 0.18V, and a frequency of 1GHz. This AC source models a 1GHz system clock. The other AC source is 7.4 GHz and also has a peak to peak voltage equivalent to a tenth of the supply voltage. This source models switching noise contributed by the logic and any environmental noise.

A ring of three inverters with a fanout of 1.304 oscillates at a frequency equivalent to a three stage ring with two tokens in it. Two rings of three inverters and two three stage FIFO rings were constructed. The dirty and clean voltage sources each supply one FIFO ring and one inverter ring each. Each ring oscillates for a few nano-seconds before noise appears on the dirty supply for a period of time equivalent to four periods of the rings when the supplies of the rings are clean. The extra delay caused by the modeled jitter on all nodes is measured and averaged.

The average delay offset between the clean and dirty version of the ring of three inverters was 16.25ps/cycle. The difference between the clean and dirty version of the ring of three FIFO stages was 14ps/cycle. FIFO rings provide a small amount of supply noise rejection over standard inverters.

Differential gates such as the Analog C-element provide common mode noise rejection. Noise that causes the supply to dip will slow the side of the gate that is producing a rising transition but will hasten the side of the gate producing the falling transition. This is the source of the slightly better performance in the face of noise for the experiment presented.

config	3 Stages 2 Tokens			4 Stages 2 Tokens			5 Stages 2 Tokens		
μ_{per}	179ps			154ps			162ps		
	phases	σ (°)	σ (s)	phases	σ (°)	σ (s)	phases	σ (°)	σ (s)
	0°, 180°	0°	0ps	0°, 180°	0°	0ps	0°,180°	0°	0ps
	120°, 300°	1.6°	0.8ps	90°, 270°	1.4°	0.6ps	72°, 252°	2.2°	1.0ps
	240°, 120°	1.6°	0.8ps	0°, 180°	2.5°	1.1ps	144°, 324°	2.2°	1.0ps
				90°, 270°	1.8°	0.8ps	216°, 36°	2°	0.9ps
							288°, 72°	2°	0.9ps
config	7 Stages 4 Tokens			13 Stages 8 Tokens			15 Stages 8 Tokens		
μ_{per}	156ps			165ps			155ps		
	phases	σ (°)	σ (s)	phases	σ (°)	σ (s)	phases	σ (°)	σ (s)
	0°,180°	0°	0ps	0°,180°	0°	0ps	0°,180°	0°	0ps
	103°, 293°	2.7°	1.2ps	111°, 291°	2.2°	1.0ps	96°, 276°	2.5°	1.1ps
	206°, 26°	4.1°	1.8ps	222°, 42°	2.4°	1.1ps	192°, 12°	4.4°	1.9ps
	309°,129°	3.5°	1.5ps	332°,152°	2.8°	1.3ps	288°,108°	6.2°	2.7ps
	51°, 231°	3.2°	1.4ps	73°, 253°	3.1°	1.4ps	24°, 204°	6.9°	3.0ps
	154°, 334°	2.3°	1.0ps	194°, 14°	3.1°	1.4ps	120°, 300°	6.9°	3.0ps
	208°, 28°	2.1°	0.9ps	305°,125°	3.5°	1.6ps	216°, 36°	7.4°	3.2ps
				55°, 235°	4.1°	1.9ps	312°, 132°	7.6°	3.3ps
				166°, 346°	3.3°	1.5ps	48°, 228°	6.2°	2.7ps
				277°, 97°	2.8°	1.3ps	144°, 324°	5.1°	2.2ps
				28°, 208°	2.6°	1.2ps	240°, 60°	6.0°	2.6ps
				138°, 318°	1.7°	0.8ps	336°,156°	7.4°	3.2ps
				249°, 69°	1.7°	0.8ps	72°, 252°	5.7°	2.5ps
							168°, 348°	3.9°	1.7ps
							264°, 84°	2.7°	1.2ps

Table 4.1: Phases of signals available in FIFO rings of various configurations and the precision of those signals

4.8 Initialization

This section explores schemes for introducing tokens into the FIFO ring and starting the ring running.

To maintain the phase relationship between adjacent stages predicted by Equation (4.1), the electrical drive of each stage and the loading on the communication wires and the electrical nodes in each stage must be identical. If there are special stages with extra functionality used for loading, they will likely cause undesirable phase shifts. It is best to give each stage the same functionality and then exercise those stages necessary for loading.

Two methods for loading tokens into the FIFO are shown. Depending upon the application and the constraints of the application, a different FIFO loading scheme is employed.

4.8.1 Brute force method

I call the first method the brute force method because it involves simply overpowering of the drivers on the output node in the Analog C-element. Figure 4.28 presents a schematic. Initially the binary values that result in the desired number of tokens are passed through a shift register. The values enable transistors RT and RF when the $Load$ signal is asserted. The tokens start to move around the ring when the $Load$ signal is removed.

Transistors RT and RF must be able to overcome any value that the C-element might try to keep. The drains of the pass transistors load the output node and compromise performance. For example, if the inverters that compose the analog C-element were sized with 1μm wide NMOS and 2μm wide PMOS transistors, then transistors of width 2μm for RT and RF are necessary to write the correct value. The shortest period achievable grows to 167ps with the added circuitry from the 152ps achievable without the initialization circuitry. If the speed control transistors presented in Section 4.6.1 are used then the speed control can be applied when loading. Transistors RT and RF can then be made smaller, reducing the output load.

4.8.2 Disable driver method

This method requires two special stages. The two stages become a Source and a Stopper for the FIFO ring. Assume that the Source is one stage downstream from the Stopper. The Source places tokens into the FIFO, while the Stopper is a barrier against which the tokens pile-up.

To make a Stopper stage, the drive transistors must be disabled so that they can not overcome the value held by the keeper inverters. Tri-stating the drive transistors is sufficient for implementing this function. The Stopper is shown in Figure 4.29. The value of the signal *start2* is LO while loading.

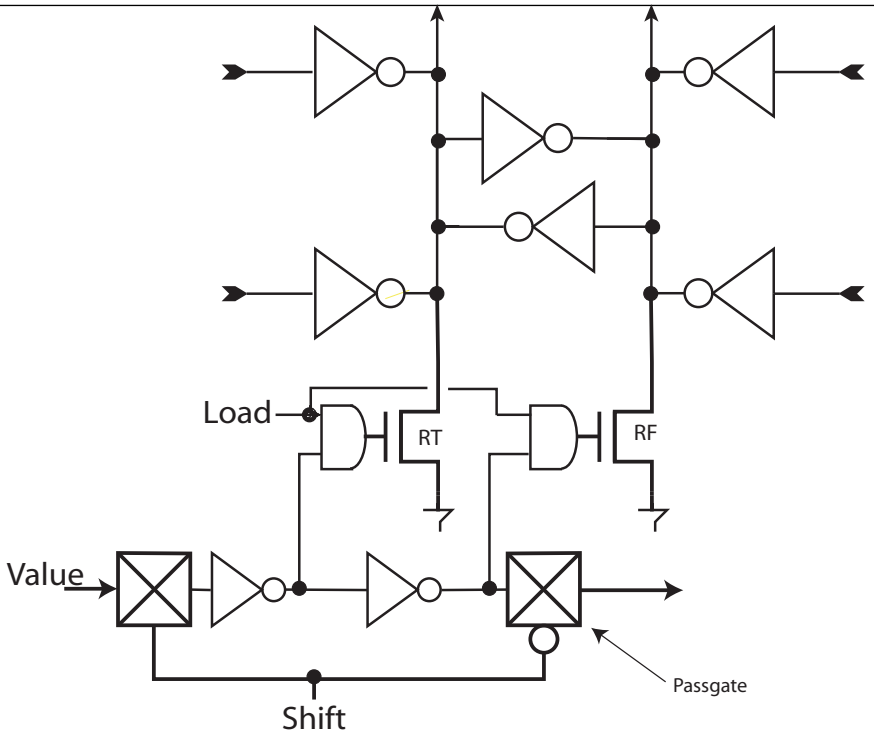


Figure 4.28: Brute force initialization method

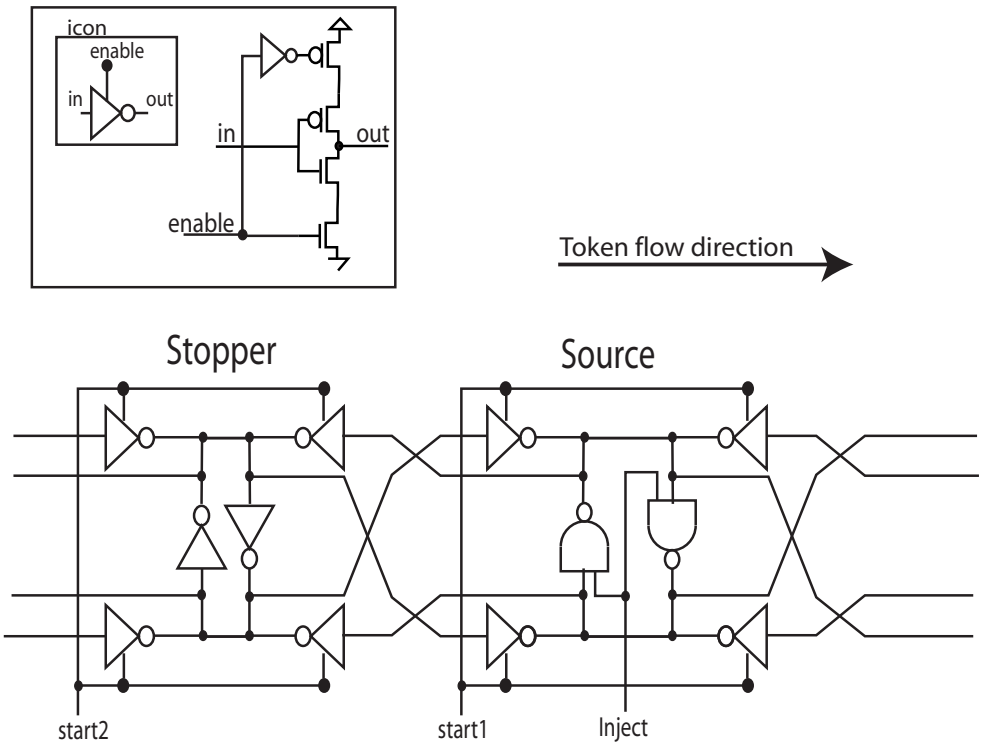


Figure 4.29: Disable driver initialization method

The Source stage puts tokens into the FIFO. Each time the output of the Source is toggled a new token injects into the FIFO. The token ripples around the FIFO until it reaches the first stage before the Stopper that is not occupied by a token. The Starter stage is built by replacing the drive inverters with tri-state drive inverters and replaces the inverters of the keeper with back to back two input NAND gates. One of the inputs to each NAND gate connects to the output of the other NAND gate, the other input connects to an *inject* signal. Each time *inject* toggles, a new token enters the FIFO. *Inject* must be toggled an even number of times. After the desired number of tokens are in the FIFO, the drive transistors to the Source are enabled. Then the tokens are released to flow around the FIFO when the drive transistors in the Stopper are enabled. The Source is also shown in Figure 4.29. The signal *start1* is LO while loading.

Systematic skew is introduced with this method because each stage is no longer identical. The systematic skew can be mitigated by minimizing the electrical differences between the Source and Stopper stages and the normal stages. The series transistors that are introduced in the tri-state buffers and the NAND gate should connect to the supplies and made as wide as feasible. This minimizes the added resistance of these conduction paths. During normal operation these transistors are transparent so the nodal capacitance between series transistors won't affect operation. The PMOS transistor connected to the *inject* input should be minimum sized. While loading speed is not critical and this will minimize its loading on the circuit.

Systematic skew can be completely eradicated making each stage a Source and connecting the *inject* and *start1* and *start2* signals to only two of the stages.

The signal *start2* can be just a delayed version of *start1*. This reduces the number of extra pins necessary to implement this initialization method from three to two.

4.9 Applications

This section discusses potential applications for the technology introduced in the first part of this chapter.

4.9.1 Sub-gate delay timing precision

Professor Mark Horowitz's group at Stanford released three theses (18; 41; 46) with a focus on timing precision in the last ten years. These works culminated in the work by Weinlader which achieved a reported 3% timing resolution using a post-fabrication calibration method. From what they learned on their latest test chip they conjecture they could achieve even finer precisions.

FIFO rings constructed from Analog C-elements offer compelling advantages that could be used in conjunction with calibration techniques to achieve the resolutions seen using interpolation with far less circuitry and power. The foundation of the interpolating schemes is a DLL. The DLL in Weinlader provides ten phases of a ten gate delay period. This period resulted in 900MHz frequency in the process used for fabrication. From these initial ten phases they interpolated to generate 40 phases.

Many evenly spaced phases are required to achieve a high timing precision using a low frequency. If many phases are being used to achieve a high sampling rate then a large number of parallel samplers are needed. Each sampler is controlled by a different phase. Forty channels sampling a single high speed node is certain to affect significantly the signal being monitored because each sampler loads the node being observed.

Physical space is also an issue. Forty samplers consume a significant amount of area. The wavelength of a signal that requires sampling with a period equal to 3% of a FO4 delay is short. Samplers in different locations sample what are effectively different signals.

Section 4.6.4 described a method to generate differential signals with periods equivalent to 1FO4 gate delay. Section 4.6.3.2 described how to interleave phase from two separate FIFO rings. Combining these two techniques generates a signal in quadrature with periods equal to 1FO4 gate delay. This achieves equivalent timing precision to a signal of 10 FO4 gate delays in forty phases. The received signal is far less loaded because I sample with a high frequency and fewer phases.

Issues related to high frequency sampling such as the bandwidth of the sampler need to be addressed but having a high frequency sampler in the design arsenal provides alternative solutions to consider.

4.9.2 High Speed Pipelining

Multiple clock phases enable fine grained concurrency in data path operations. Data paths in state of the art commercial processors exploit the speed advantage of logic styles that use multiple clock phases. The three studies below cite advantages derived from using a plurality of clock periods.

Static Latch Design — Molnar et al (21) investigated the number of clock phases that allowed data to be shifted through a series of latches at the highest frequency. The latches are simulated to be clocked using a *magic* multi-phase clock. In other words the clocks were generated using ideal HSpice elements. The phase offset for the clock signal from stage to stage was adjusted while the frequency of the clock to each stage matched. Initially they did not concern themselves with how these signals would be generated in silicon.

Their experiment used models from a 0.6μm process. They found that a 'near-optimal' performance was obtained using an 8-phase clock with a cycle time of 800ps with the clock signal into each stage lagging the clock signal to the previous stage by 300ps.

The authors concluded that generating high frequency signals with the eight distinct phases provided by their *magic* control was physically unrealizable and compromised with a signal represented in fewer phases. The technology presented in this chapter makes the optimum timing regime for their latch physically realizable.

Skew Tolerant Domino — Harris (14) derives the following equation to relate the maximum tolerable clock skew for Domino logic as a function of the number of clock phases (N) used.

$$t_{max-skew} = \frac{\frac{N-1}{N} \times T_c - t_{hold} - t_{precharge}}{2} \tag{4.24}$$

Using this equation he presents the following table of values for maximum clock-skew and necessary pre-charge time for different values of N assuming a sixteen gate delay clock period.

N	*Precharge (gate delays)*	*Maximum Skew(gate delays)*
2	6	2
3	7.33	3.33
4	8	4
6	8.66	4.66
8	9	5

Table 4.2: Skew tolerance for various numbers of clock phases.

Notice that as the number of clock phases increases, the maximum skew tolerated also increases. Unfortunately the extra clock skew gained by using more phases is not avail-

able at higher frequencies where it is most needed using the current solutions detailed in Harris' book. Analog FIFO rings remove this limitation.

Output Prediction Logic Output prediction logic (34) is a high performance CMOS logic family that claims speedups of 2x to 3x over optimized conventional static circuits. Though promising great speed benefits over even Domino data paths, its adoption has been limited. OPL is heavily dependent on precise multi-phase clock edges for correct operation. OPL's advantage is only realized in very aggressive timing environments. The adaption of OPL requires near pico-second timing resolution at frequencies around 3×FO4 delays. These resolutions are untenable using conventional techniques.

The authors described a 64 bit carry look-ahead adder that they claim is twice as fast as other published 64 bit adders. Their adder is pipelined over six stages and produces a result every 2.8×FO4 gate delays. They require a six phase clock. They achieved a phase resolution equivalent to 0.42×FO4 gate delays. Finer resolutions would provided proportionally faster calculations. My mismatch model suggests that a three stage FIFO ring built from Analog `Micropipeline` control and initialized with two tokens could generate a 1.8×FO4 gate delay clock with a 0.02×FO4 gate delay resolution.

4.9.2.1 Driving long wires

Analog FIFO control effectively distributes timing information across a large area of silicon. Figure 4.30 records the resulting oscillation frequency of a four stage ring FIFO built from analog C-elements as a function of the size of the FIFO control. The ring is initialized with two tokens. Four wires run between each stage. The four wires carry the signal in quadrature phase with zero systematic skew.

In one instance the length of wire modeled a short interconnect. The interconnect was 100μm long and 1μm wide. In another instance the length of wire modeled a mid-level interconnect. Here the interconnect was 500μm long and 2μm wide. Finally I modeled a wire used for global signaling. The interconnect is 1000μm long and 3μm wide.

The X-axis records the total transistor width used in a single stage of FIFO control. The analog FIFO control was constructed using six identical inverters. The number along the X-axis is six times the size of the individual inverters used to construct the Analog C-element.

Notice that each stage drives twice the length of wire that runs between stages because the same signal is used to $Acknowledge$ the previous stage and $Request$s the next stage.

Two wires running alongside each other couple charge to the other. The sidewalls of each wire form a capacitor that couples charge between them. The known phase relationship between the handshake wires allows for them to be arranged in a manner that constructively couples charge. Take the special case where the number of tokens is equal to half the number of stages.

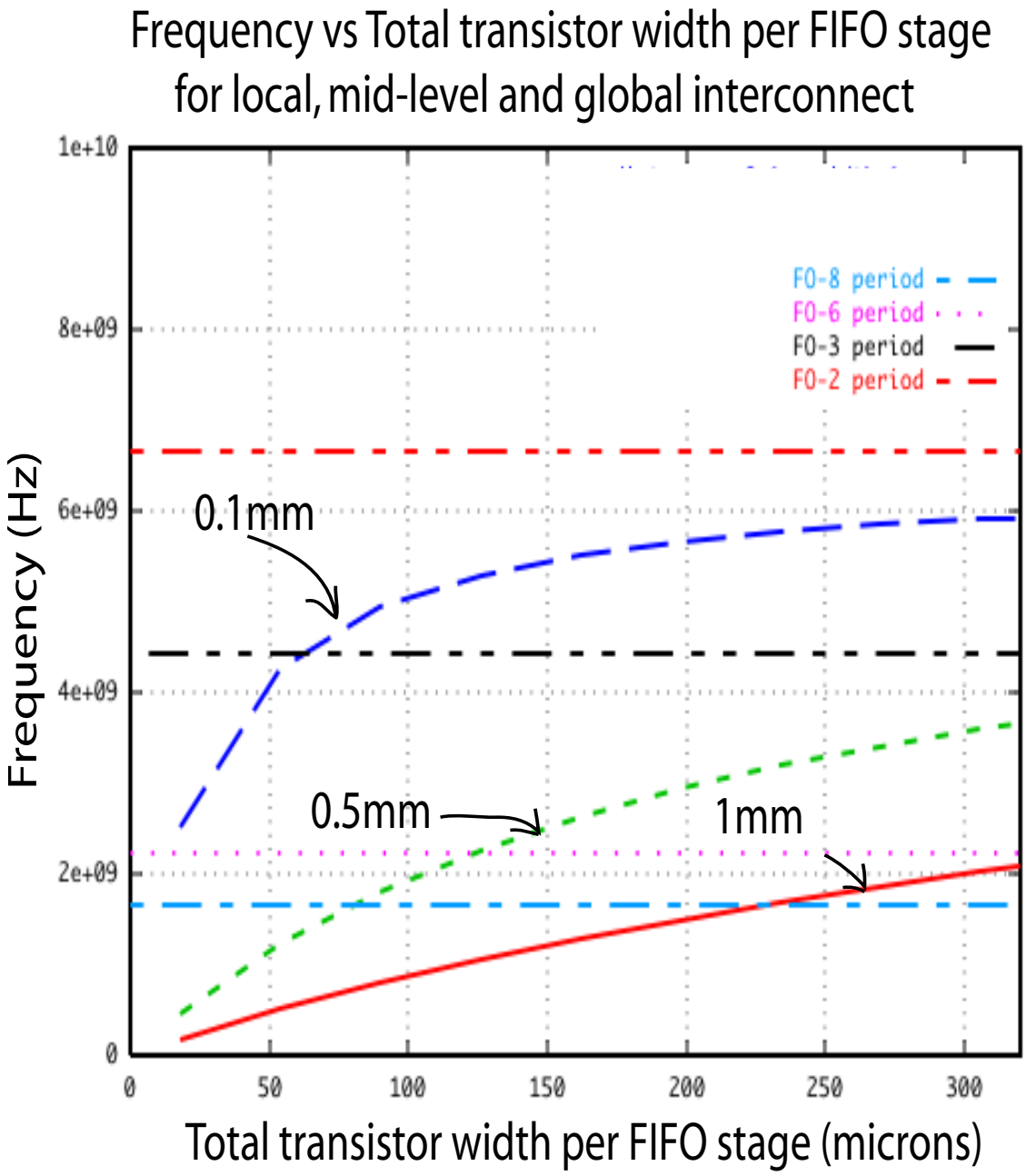


Figure 4.30: Analog `Micropipelines` for distributing signals

The four wires that run between stages oscillate in quadrature. The wires can be interleaved with respect to each other so that they couple charge that hastens rather than hinders voltage transitions (26). If the differential handshaking wires are arranged $Request$, $Acknowledge$, $\overline{Request}$, $\overline{Acknowledge}$ then each wire is shifted by 90° from its predecessor. This arrangement ensures that the nearest transitions on neighboring wires are in the same direction. More charge that hastens the next transition is coupled than charge that hinders the next transition.

Chapter 5

Pulse protocols

This chapter applies the principles discussed in the previous chapter to a self-timed FIFO control that employs a different protocol from Micropipelines to transfer the timing tokens between stages. The full name of this protocol is the dynamic asymmetric pulse protocol, or colloquially `Dynamic asP` (20). Charlie Molnar pioneered this control as a higher performance optimization to the already snappy asP protocol (21).

`Dynamic asP` efficiently communicates timing information. It drives long wires efficiently. I alter standard `Dynamic asP` to amplify the Charlie effect so that tokens placed in ripple FIFOs built with this control and connected in a ring separate evenly around the circumference of the ring resulting in the distribution of timing information.

5.1 Pulse Protocols

Two phase signaling employed by Micropipelines attracts the self-timed circuit designer because it uses the fewest transition to effect a token move, it has a minimal number of delay assumptions, and it is logically elegant. Logical symmetry, while elegant, is also the weakness of two-phase signaling. Two-phase signaling logic gates requires a PMOS transistor stack that mirrors the NMOS stack. PMOS transistors are approximately 2.5 time less conductive than NMOS transistors of equivalent dimensions. The physics of CMOS circuitry moves designers towards design styles that reflect this reality.

5.1.1 Pulse protocol basics

The poor conductivity of PMOS transistors pushed self-timed circuit designers away from two-phase or transition signaling and towards pulse protocols. The protocols are described as pulse because the same signal that initiates a token movement also initiates the removal or de-assertion

of that same signal. This strategy reserves efficient NMOS transistors to implement complex logic functions, while utilizing PMOS transistors to simply reset the logic.

A number of papers exist that describe pulse protocol topologies (20; 21; 33; 38). Each of these design styles use pulses to set and reset the state of a stage to either FULL or EMPTY. When a stage signaling FULL in a linear FIFO precedes a stage signaling EMPTY, a move initiates and the states of the handshaking wires toggle to signal EMPTY and FULL respectively.

A logic gate detects the FULL and EMPTY state in all of the pulse protocols. When the move signal asserts, three actions initiate: the latches are made transparent and the data is copied forward, the state of the preceding and next stage are toggled, and the AND gate returns to its non-asserted state.

5.1.2 Weighing the options

Schematics for three stage FIFOs using `AsP`, `GasP`, and `Dynamic asP` are shown in Figure 5.1. `AsP` ripple FIFO control (21) uses a SR latch built from two cross tied NAND gates to maintain the state of adjacent stages. `Dyanamic asP` and `GasP` (20; 38) ripple FIFO control replaced the cross-tied NAND-SR latch with an SR latch implementation that uses a keeper and two drive transistors of opposite sex on either side of a wire called a state conductor. When a pulse of the proper polarity is applied to the gate of a transistor in this SR latch, the value on the state conductor is driven towards the polarity attached to that transistor's source. The value is held indefinitely by the keeper or until the value on the state conductor is reset by the transistor on the opposite side.

Handshaking is performed with a single long wire running between stages. This is notable considering the growing cost in driving long wires. Consider that the differential two-phase signaling protocol employed in the previous chapter required four long wires between stages.

`Dynamic asP` is disclosed only in the form of a patent (20). The `GasP` control topology supplanted it because it was just as efficient at moving data while using only a single FIFO stage primitive instead of two. `Dynamic asP` needs latches that become transparent on opposite polarities in alternating stages. The FULL signal is also assigned to opposite polarities in alternating stages. A `GasP` control stage has four gate delays going forward and two going back. This apportionment better models the movement of data through a simple data path. Data only moves forward. `Dynamic asP` places three gate delays in both directions.

The symmetry of `Dynamic asP` is not ideal for controlling self-timed FIFOs but it is attractive for the purposes of generating and distributing high precision timing signals. A single type of FIFO control stage exist in `GasP` but the delay characteristics differ depending on if the next stage empties or the previous stage fills last. `Dynamic asP` uses two types of FIFO

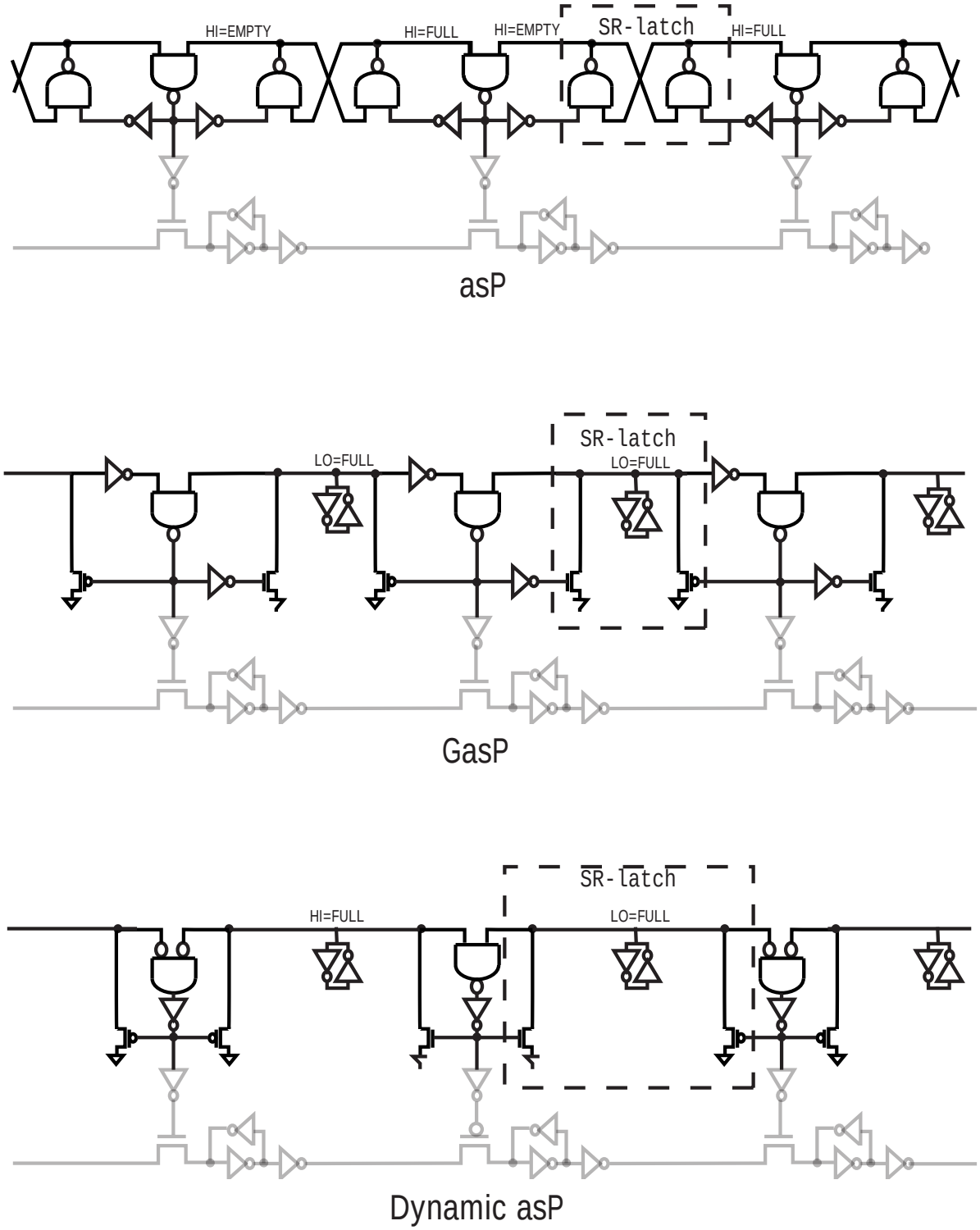


Figure 5.1: `AsP`, `GasP`, and `Dynamic asP` ripple FIFO controls

control stage but each of the FIFO stages can be made completely symmetric. This means the character of the ripple FIFOs composed of this control have the same character whether the FIFO is congested with tokens or sparsely occupied.

I choose to use `Dynamic asP` to demonstrate how self-timed ripple FIFO control can be used to generate and distribute high precision timing signals. I alter the `Dynamic asP` control to amplify the difference in the stage delay between the cases when the inputs to the FIFO stage assert simultaneously and when they assert when separated by a large amount of time. The term Charlie effect was used to describe this phenomenon for Micropipelines in Section 3.3.1 and is adequate here as well considering Charlie[1] pioneered this control.

5.2 The FIFO stages

5.2.1 Salient details

Figure 5.2 shows two stages of `Dynamic asP` FIFO control. The data path is omitted because my application is data path agnostic. I call the `Dynamic asP` control primitive that changes the state of the adjacent state conductors to a HI voltage a Pull-up stage. I call the other control primitive a Pull-down stage.

[1] Charles Molnar, was one of my mentors. He passed away in 1997

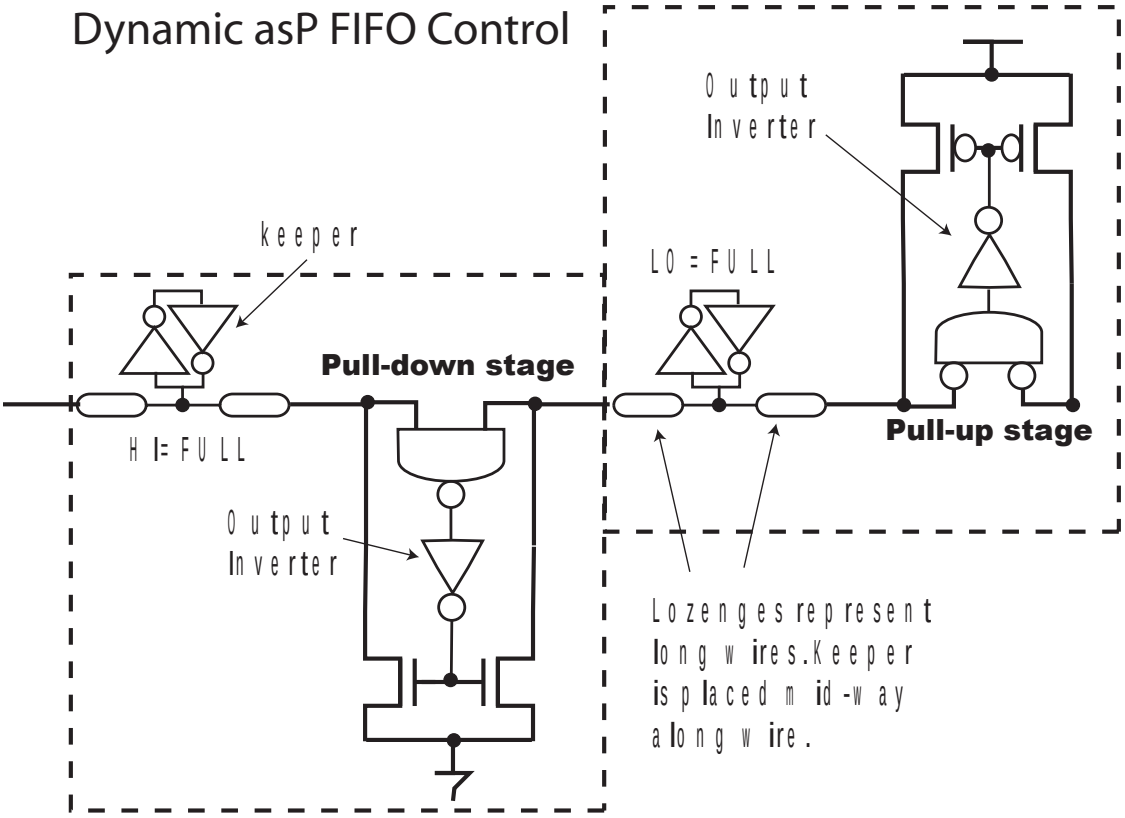


Figure 5.2: One stage of Pull-up and Pull-down `Dynamic asP` FIFO control

Some things to note about `Dynamic asP` FIFO control:

Alternation —The meaning attached to the binary signaling values alternate after each stage. The interesting repercussion of this protocol is that a FIFO that is half occupied and with evenly spaced tokens has the same binary value on each state conductor.

Handshaking — A single wire is used for communication. Signals distributed over long distances using `Dynamic asP` uses little power.

Series transistors — The implementation of the NAND and NOR gates for this application are free of series transistors. I discuss these implementations shortly. No series transistors are used in the `Dynamic asP` control. This is advantageous because series transistors result in different electrical characteristics depending on which input asserts last. Also the node internal to the series transistors dumps charge onto the output node at unexpected times and upsets the precision of timing signals.

Efficiency — Even though only twelve transistors per stages are used, three stages of amplification are provided between the gates that detect a new value on the state conductor through to the transistors that drive the state conductor.

Symmetry — Each type of FIFO control is electrically and logically symmetric.

Occupancy — `Micropipeline` rings with an even number of stages can only have an even number of tokens. On the other hand `Dynamic asP` control can only have an even number of stages but allows any number of tokens not exceeding the number of stages.

Token flow The direction of token flow when the FIFO is not controlling a data path must simply be defined. Otherwise, the control is completely symmetric, it makes no distinction between tokens moving forward and holds moving back.

As in `Micropipeline` control, the Charlie effect causes the tokens to repel each other. This leads them to evenly distribute themselves around the FIFO into a locked state. The handshaking signals between stages then have a predictable phase relationship with respect to each other. The phase between signals on the state conductors that are N stages apart is:

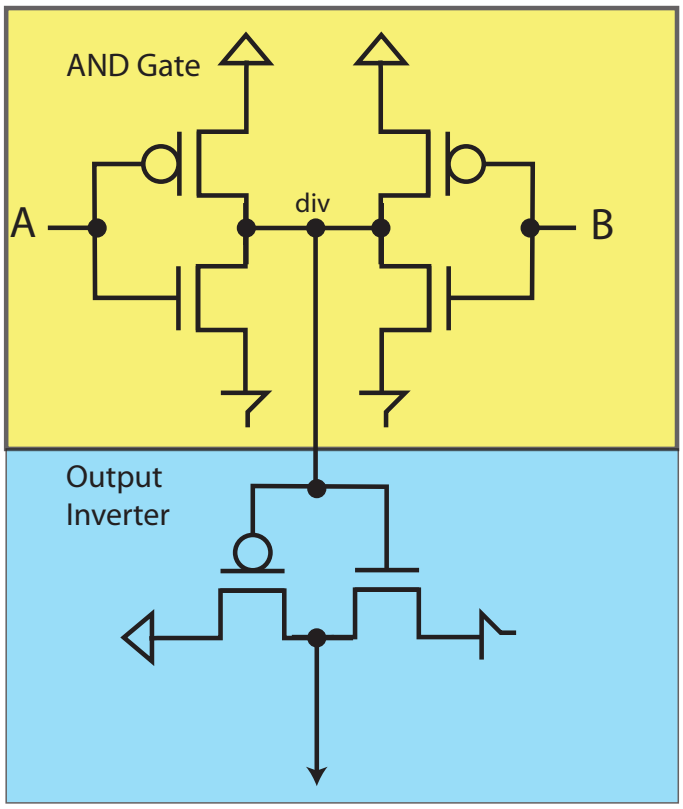
$$\phi_N = (N) \times (\frac{\text{tokens}}{\text{stages}} \times 360° + 180°) \tag{5.1}$$

The phases on the state conductors N stages apart when the tokens are evenly spaced around the ring is N times the fractional number of tokens per stage multiplied by 360 degrees. One hundred eighty degrees is then added to this product. A token movement copies the FULL state from one state conductor to the next. This is the source of the 360° term. The polarity of the FULL signal changes by 180 degrees in adjacent stages. This is the source of the 180° term.

5.2.2 Variable delay ANDs

An unusual implementation of the AND gate is described here that detects the move condition and amplifies the differences delay for the simultaneous and widely separated cases. In the Pull-down FIFO control the move condition is detected by a NAND gate while in the Pull-up control, the move condition is detected by a NOR gate. The NAND and NOR gates operate on the same principle.

Both NOR and NAND gates are referred to as AND gates because a NAND asserts when its first AND second input are HI, while the NOR asserts when its first AND second input are LO. The asserted value for the inputs is HI for a NAND and LO for a NOR.



When ***A*** and ***B*** are at different binary values, the voltage on node ***div*** settles somewhere between supply and ground.

The settling potential is determined by the PMOS/NMOS width ratio of the transistors in the AND gate.

The PMOS/NMOS width ratio of the output inverter determines its logical threshold voltage.

If this threshold voltage is greater than the settling voltage of the AND gate on node ***div***, then the AND implements the functionality of a NOR gate, otherwise the functionality of NAND is implemented.

Figure 5.3: Variable delay AND gate

The variable delay AND is implemented by shorting the output of two inverters as shown in Figure 5.3. A voltage divider forms when the input to the two inverters differ. The placement of

the voltage divided value with respect to the threshold voltage of the gate driven by the voltage divided node determines whether the inverters implement the NAND or NOR function.

This AND gate relies on the same principle discussed in Section 4.2 to amplify the difference in delay between when the inputs arrive simultaneously and when they are largely separated. When neither input is asserted, the output voltage is at one of the rails, supply or ground depending on if the gate is a NAND or NOR respectively. After a single inputs asserts a voltage divider is formed. Time is required for the output voltage to move from the supply to the voltage divided value. The delay of the AND gate varies depending on how far the output voltage has decayed towards the voltage divided value when the second input asserts.

The delay of the AND gate is equal to the time required for the potential to be driven over a voltage range equal to the output voltage when the second input asserts minus the logical threshold voltage of the following stage. When the inputs assert simultaneously, this voltage difference is at a maximum and therefore so is the delay. If the inputs assert with very large temporal separations, then the output voltage has decayed to the voltage divided value when the second input asserts. The delay is minimum. A continuum of monotonically decreasing delay values exist between these two cases.

AND logic is boolean. The outputs and inputs to an AND gate are interpreted to be either a HI or LO. The logic gate driven by the NAND interprets the value according to the value of its logical threshold voltage. An inverter with a high threshold voltage interprets a larger fraction of the potentials between supply and ground to be a logical LO value than an inverter with a low threshold voltage. If the potential of the NAND gate output is greater than the threshold voltage of the logic gate it drives, then the output is HI, otherwise it is LO. For a NAND gate, care must be taken to place the threshold voltage of the following stage to be less than the voltage divided value when the inputs disagree. For a NOR gate the threshold voltage of the following stage must be greater than the potential of the voltage divider when the inputs disagree.

Assume that the AND gate I have described drives an inverter that I refer to as the output inverter. Assume also that we are dealing with a NAND gate. To maximize the delay variability of the NAND gate, the threshold voltage of the output inverter should be moved as close to the ground potential as practical and the voltage divided potential should then be moved as close to the threshold voltage as margins allow without becoming less than the threshold voltage.

The threshold voltage of the output inverter and the voltage divided voltage of the NAND gate is controlled by varying their PMOS/NMOS width ratio. The threshold voltage of an inverter is found by connecting the output of the inverter back to the input and observing the potential at which the this node settles. Figure 5.4 shows that the threshold voltage varies between 0.58V and 1V on a 1.8 volt supply as the PMOS/NMOS transistor width ratio varies between 0.1 to 10. The figure also shows that the same range of width ratios results in a nearly full range of voltage divider values.

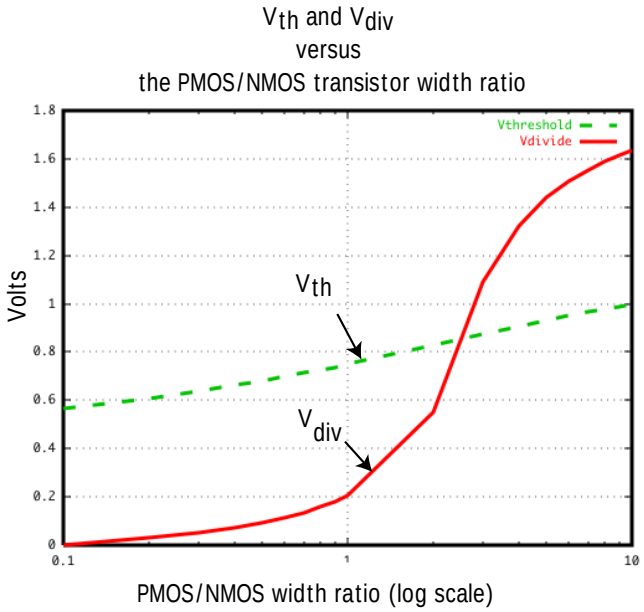


Figure 5.4: Vth and Vdiv as a function of transistor width ratio.

NOR gates built in this style are much more effective than NAND gates. Large PMOS/NMOS transistor width ratios are necessary to implement NAND logic. NOR gates are used in the control stages that drive the state conductors HI. Fortunately the more efficient NOR gate is used in the stages that drive the inefficient PMOS drive transistors. This mitigates the differences in driving capability between the stages.

5.3 Design decisions

The Analog `Micropipeline` control stage had relatively few variables to adjust. Little value was gained by varying the transistor width ratios or sizing the keeper inverters large or small relative to the drive inverters in Figure 4.2. `Dynamic asP`, however has a number of design decisions that affect the character of the FIFO rings.

5.3.1 Locking Range

Forcing tokens to lock in a FIFO ring depends on the range of separations over which the delay of a stage varies as well as the magnitude of the variation. To further amplify both the variable delay and the range of separations the size of the output inverter relative to the size of the AND gate can be increased. This sizing ratio can be arbitrarily increased until the range over which the delays act is sufficient to result in the tokens locking. Keep in mind that the standard

deviation of skew arising from transistor mismatch increases linearly with respect to the amount of electrical amplification required of a stage as shown in Figure 2.6c. Also note that increasing the range in this way increases delay and slows the frequency.

5.3.2 Duty cycle

When the occupancy of the ring is less than half and a number of tokens are locked in an evenly spaced pattern, then the signal on the state conductors connecting Pull-up stages to Pull-down stages is a series of evenly spaced pulses rising from the ground potential. The signal on the state conductors connecting Pull down stages to Pull-up stages is a series of evenly spaced pulses descending from the supply to ground and back. If the ring is more than half full then these conditions are swapped.

When the FIFO ring is exactly half occupied, all state conductors, ideally, have signals of the exact same frequency and phase. The duty cycle of the signal is equal to the delay through the Pull-down stage when the inputs arrive simultaneously divided by the sum of the Pull-up and Pull-down stage delays when the inputs arrive simultaneously. The duty cycle is 50% when the Pull-up and Pull-down stage have the same delay.

5.4 The Separation Plot

As in the last chapter, a Separation plot illustrates how the delay of the control stages vary with respect to the temporal separation of the inputs. Differential input signals exercised both types of control simultaneously. The Separation of the two input signals varied over a range. The delay of the stage is the difference between the time where the differential output signals cross minus the time at which the last asserting differential input signals cross. Figure 5.5 shows the results.

5.5 Visuals

Figure 5.6a shows the signal on a single state conductors in a twenty stage ripple FIFO built from `Dynamic asP` control. The ring is initialized with three tokens. Notice that the tokens, represented by the pulses, start in a pack and separate to an evenly spaced pattern after about 25ps.

Figure 5.7 shows the signal on two state conductor separated by four stages in a twenty stage ripple FIFO built from `Dynamic asP` control. The ring is initialized with ten tokens. Notice that the tokens are initially far out of phase. They lock in phase by about 10ns.

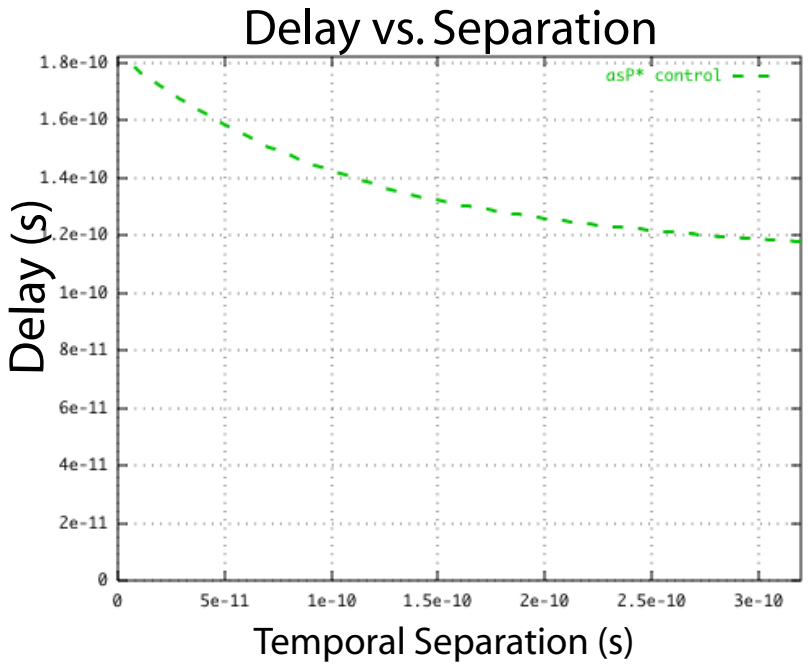


Figure 5.5: Delay vs. temporal separation of inputs for asP* control

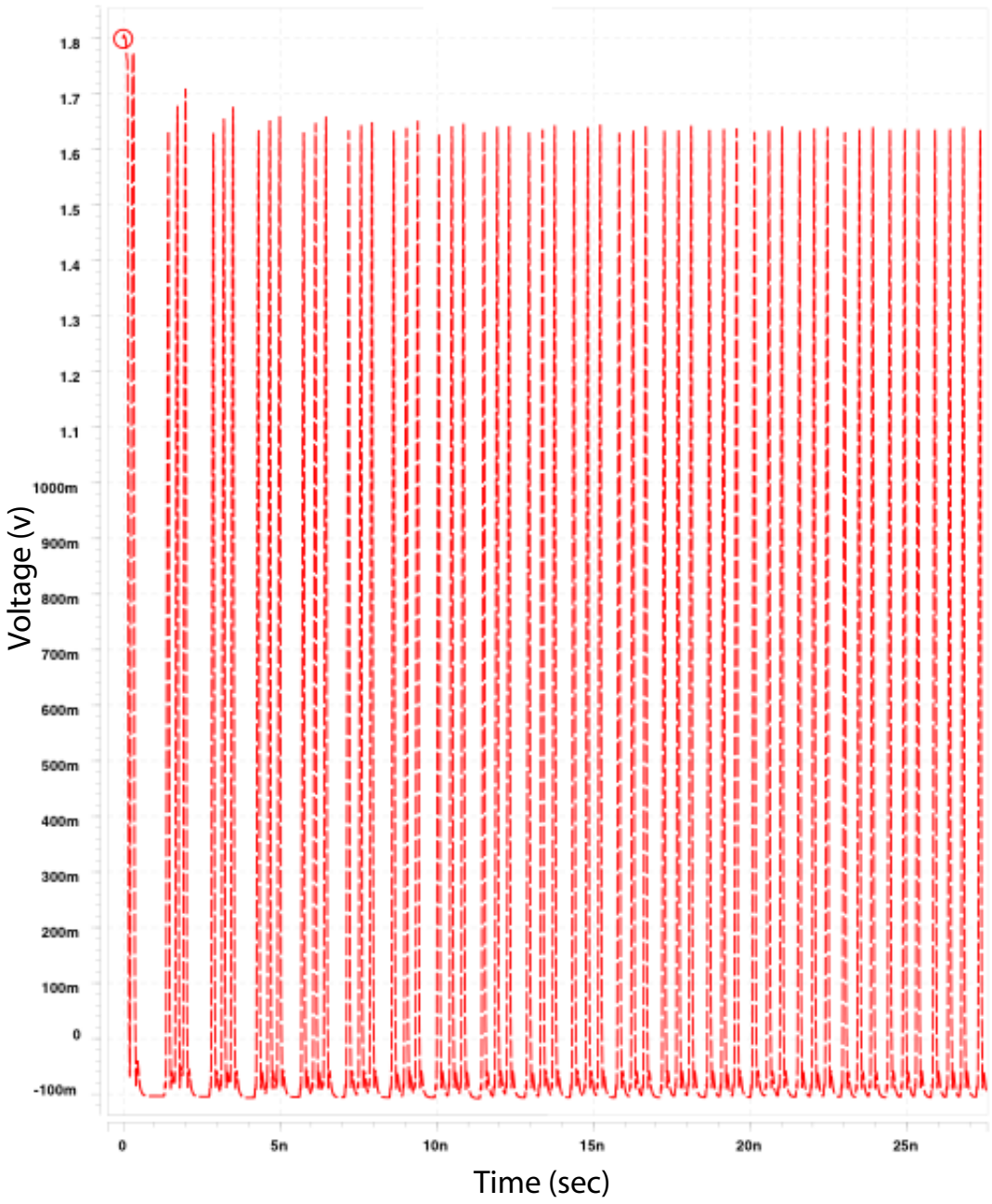


Figure 5.6: Three tokens moving into lock in a 20 stage FIFO ring.

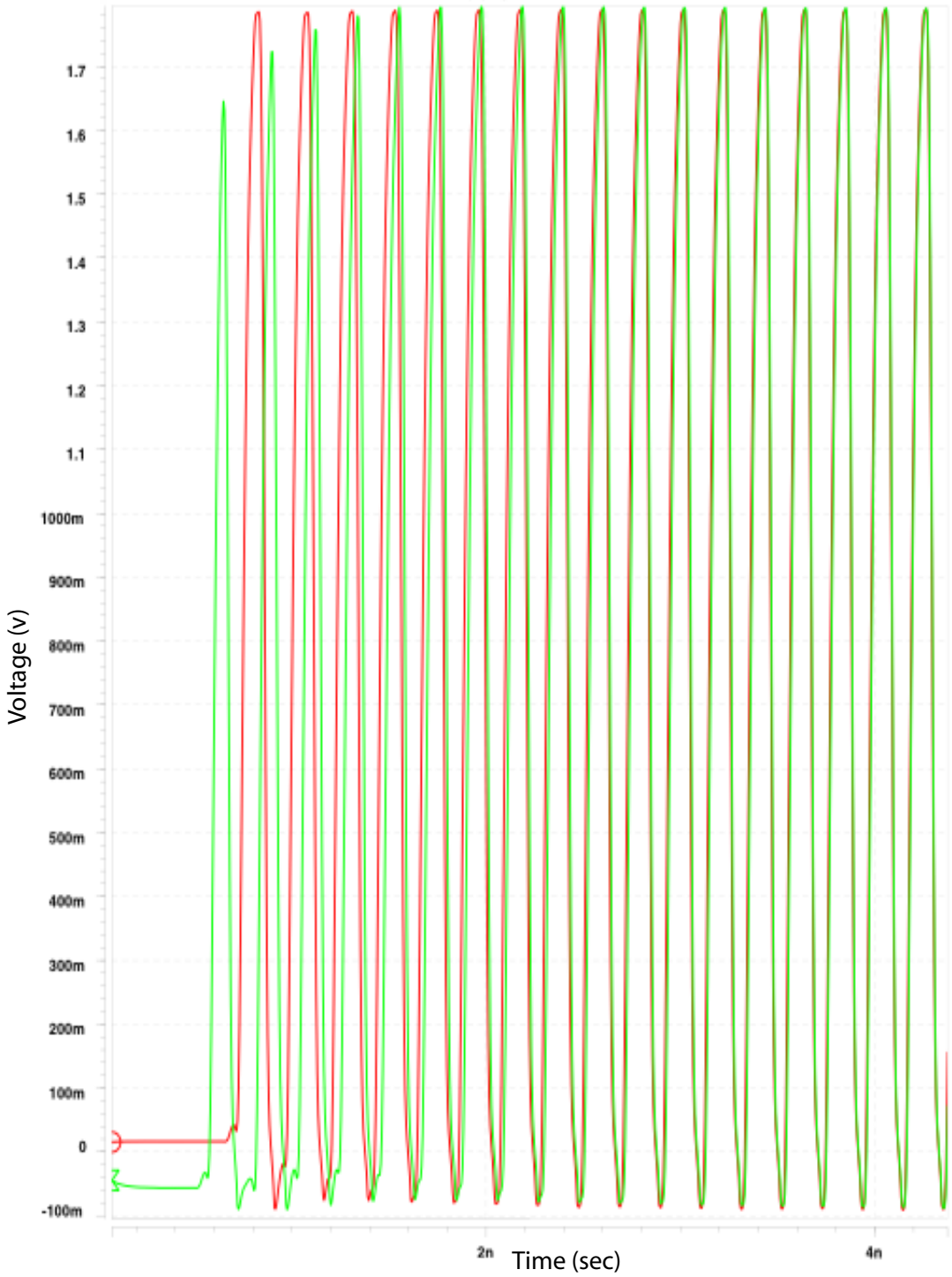


Figure 5.7: Ten tokens moving into lock in a 20 stage FIFO ring.

5.6 Performance and Power

All Hspice simulations for this thesis were done with models extracted from the same 1.8V 180nm process. The FO4 delay for an inverter with PMOS/NMOS transistor ratio of two in this process is 88ps. A ring of three unloaded inverters oscillates with a period of 158ps.

The peak to peak voltage swing for all nodes in the rings when locked was over the complete supply voltage.

The average current draw per stage when the ring is locked is within a couple percent of 0.3mA regardless of the number of stages or tokens.

5.6.1 Transistor mismatch

Table 5.1 shows the results of Monte Carlo simulations in HSpice for FIFO rings built from Dyanmic asP control. The top row in the table describes the configuration simulated. The second row is average period resulting in the ring. The third row is the ideal phase deviation in degrees and pico seconds between state conductors in every other stage. Remember that state conductors in adjacent stages have sufficiently different shapes so that they can't really be compared. I presented the results in odd and even columns. These phases are relative to two arbitrarily located state conductors in adjacent stages.

These results are not very impressive relative to the results presented in the last chapter. The expected skew is substantially greater than the expected skew in rings made from Analog Micropipelines. The source of the increased expected skew is the greater number of stages, design variables, and the non-differential implementation.

The value of the logical threshold voltage for an inverter with a PMOS/NMOS width ratio of two has a standard deviation of 15mV. The value of the voltage divided value of two inverters with a PMOS/NMOS width of two has a standard deviation of 60mV. The standard deviation for the value of the logical threshold voltage for the Analog C-element is 10mV. The physical properties of the `Dynamic asP` control has much more statistical variation than the Analog C-element. This results in a much greater range of skews.

5.6.2 Jitter

This section explores the timing robustness of `Dynamic asP` with respect to sources of jitter.I modeled the effect of power supply noise on the timing of these FIFO rings using the same method deployed for the `Micropipeline` rings in Section 4.7.2.

Two AC sources in series with the power supply model power supply noise. One AC source has a peak to peak voltage of a tenth of the supply voltage, 0.18V, and a frequency of 1GHz. This AC source models a 1GHz system clock. The other AC source is 7.4 GHz and also has a

configuration		4 Stages 2 Tokens		6 Stages 2 Tokens		10 Stages 3 Tokens	
period		208ps		241ps		260ps	
relative phase		0°		240°		216°	
	phase	odd	even	odd	even	odd	even
	0	0	0	0	0	0	0
	1	4.3ps	4.1ps	4.8ps	4.5ps	4.1ps	4.6ps
	2			5.6ps	5.8ps	6.7ps	7.8ps
	3					8.0ps	8.0ps
	4					6.7ps	5.6ps

configuration		14 Stages 5 Tokens		20 Stages 7 Tokens		20 Stages 10 Tokens	
period		208ps		207ps		207ps	
relative phase		257°		252°		0°	
	phase	odd	even	odd	even	odd	even
	0	0	0	0	0	0	0
	1	5.9ps	4.3ps	5.9ps	5.9ps	15ps	14ps
	2	8.0ps	7.1ps	7.2ps	8.3ps	28ps	23ps
	3	7.5ps	6.2ps	10.7ps	11.5ps	31ps	28ps
	4	7.5ps	7.8ps	10.7ps	11.5ps	33ps	33ps
	5	9.4ps	9.4ps	10.1ps	10.1ps	39ps	40ps
	6	7.1ps	7.4ps	11.3ps	11.6ps	42ps	39ps
	7			11.8ps	10.9ps	35ps	31ps
	8			9.0ps	8.3ps	24ps	24ps
	9			6.7ps	5.1ps	14ps	14ps

Table 5.1: Results from HSpice simulations for `Dynamic asP` rings with transistor mismatch

peak to peak voltage equivalent to a tenth of the supply voltage. This source models switching noise contributed by the logic and any environmental noise.

A ring of three inverters with a fanout of 1.994 oscillates at a frequency equivalent to a six stage ring with two tokens in it. HSpice netlists describing two rings of three inverters and two three stage FIFO rings made up my test structure. I connected one FIFO ring and one inverter ring each to a dirty supply as described above and a clean supply. I started each ring oscillating, after a few nano-seconds I turned-on the noise on the noisy supplies for a period of time equivalent to four periods of the rings when the supplies of the rings are clean. I measured the extra delay caused by the modeled jitter on all nodes and averaged.

The average delay offset between the clean and dirty version of the ring of three inverters at this loading was 28ps/cycle. The difference between the clean and dirty version of the ring of three FIFO stages was 25ps/cycle. FIFO rings provide a small amount of supply noise rejection over standard inverters.

5.7 Initialization

This section describes how `Dynamic asP` is initialized.

The output inverter of both types of control is replaced by a NAND gate. The extra input is connected to RUN. When the RUN signal is LO, the Pull-down control inserts a token on the state conductor connecting the Pull-down control with the next stage. If the RUN signal is asserted in a Pull-up stage, then tokens are prohibited from being passed. To start the FIFO ring, bring the RUN signal HI.

Two adjacent Pull-up and Pull-down stages can single step tokens into the ring. The Pull-up control is in the stage before the Pull-down control. RUN is LO in the Pull-up control, blocking tokens from being passed. The RUN signal is pulsed from HI to LO and back to HI in the Pull-down control for each token launched into the ring. The RUN signal in all other stages is left HI at all times. After the proper number of tokens are loaded into the ring, the RUN signal in the Pull-up stage is brought HI.

5.8 Conclusions

The work in this chapter further explores the space of self-timed control circuitry used for the purpose of high precision timing. Either two-phase or pulse handshaking protocols are used in the majority of self-timed systems. `Micropipelines` and `Dynamic asP`, respectively, are the self-timed circuit control topologies from each of these protocols best suited for high precision timing.

This chapter investigates whether the token repelling properties seen with Analog Micropipelines can be excited in Dynamic asP control to produce a useful high precision timing standard. The result is: yes, the token repelling properties can be excited but the resulting system lacks the precision to offer a compelling advantage over existing high precision timing systems. However the basic topology remains interesting because of the single wire used to communicate between stages.

The high precision timing circuits thus far are presented as small modules that would likely consume a small fraction of a system's total power. For systems where the self-timed circuitry timing standard is needed over a larger area, such as in a global clock, a single communication wire between the self-timed control circuits could result in significant power savings and once again make this topology compelling. The investigation as to whether Dynamic asP can be used as a competitive high precision timing system yielded a negative result. The next chapter investigates clock distribution, where timing signals must be communicated over a much larger area. Power, topology and precision are additional criteria that must considered when evaluating potential solutions. Dynamic asP is foundational to the solution presented there.

Chapter 6

Clock Distribution

6.1 Task and metrics

Clock distribution is a problem of increasing difficulty on modern high performance chips. The power-skew function, design time, and integration are three metrics used to evaluate a self-timed clock distribution apparatus that relies upon and extends the principles introduced in earlier chapters.

6.1.1 Power and skew

The task in clock distribution is ensuring that periodic events occur in many location at the same instant in time. The time-worn solution involves making many electrically equivalent paths from a single source to the many locations. Because this is a passive open-loop solution, the longer the paths are from the source, the more scattered the events are at their destinations due to inevitable fabrication mismatches.

Designs often employ two remedies to constrain the scattering of the events at the destinations. The first solution makes the electrical distance between the source and the leaves shorter using wider conductors and larger amplifiers. This reduces latency and the time over which signals can drift. The second solution is to make low resistance paths between the many places needing the clock signal, in essence, 'shorting' the leaves. A clock grid employs this strategy. This requires more wire. The enabling commodity in both solutions is power. Both require more hardware, which presents capacitance that must be charged and discharged in each cycle. Reducing power and reducing skew in the clock distribution apparatus are largely the same problem. This chapter discusses a clock distribution solution that, rather than finding an optimal operating point on this function, shifts this function so that a certain amount of skew is expected while using less power.

6.1.2 Regularity and geometry

State of the art chips are billion-plus or giga-scale systems. Their complexity necessitates the integration of the complete interconnect architecture (48). The power distribution system, clock distribution apparatus, and signal interconnect utilize the same interconnect stack. The design of one system must consider and balance the needs of the others. Charge is coupled between conductors that run over or next to each other. Currents excite fields that affect the signals around them. The route of a signal or power conductor must consider the location of others.

If we consider a microprocessor with a view to its geometry, the most salient detail is it 'rectangularity,' see Figure 6.1.

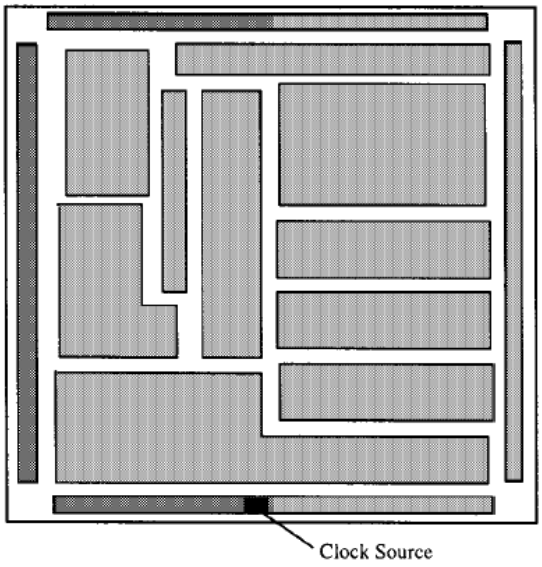


Figure 6.1: Image of Microprocessor Chip

Although recent processes have allowed 45° elbows in the routes of interconnect, the vast majority of routes are confined to run parallel to the X and Y axis of the chip and make only Manhattan or 90° turns. The functional units are typically of a rectangular shape. The power is most often distributed in a rectangular mesh whether sourced through peripheral pads or through a flip-chip bonding array. Power and ground rails are available at regular intervals along the X and Y axis of the chip. Fabrication equipment steps along one axis of the silicon, laying down masks and etching in the appropriate place. The VLSI fabrication process lends itself to rectangular topologies.

Continued progress in the art of VLSI relies upon our ability to confine materials, voltages, and currents to specific places and paths. Regular and congruent structures facilitate this goal. The clock tree structure that is employed to amplify a signal from a single source and distribute

itself over a large two dimensional area is incongruent with the power grid and the rectangular shapes of the other structures. The clock distribution techniques discussed in this chapter 'piggyback' on the existing grid topology of the power distribution system.

6.1.3 Design time

The design of the clock distribution apparatus is a time consuming challenge. The effort exerted achieving design closure must be considered. The incongruous relationship of a clock tree topology and the other systems on the chip requires making difficult concessions with respect to routing, sizing, and placement. Each alteration in the place or route of circuitry could require a complete clock tree re-balance. The cost in tools, design time, and computer cycles must be considered when evaluating the timing strategy to employ. The clock distribution technique in this chapter uses modular design principles. A small collection of gates are optimized once and then duplicated.

6.2 Micropipeline Rings for global clock distribution

Ripple FIFO rings precisely distribute timing signals and efficiently drive long wires. Both of these properties are necessary in an effective clock distribution apparatus. This section develops the ideas presented in the previous chapters of this monograph to the specific application of distributing a high precision clock signal.

6.2.1 Proposal

Figure 6.2 shows a 5×5 grid. This structure was built using an Analog Micropipeline FIFO ring. Micropipeline stages are spaced by a distance equal to some integer multiple of the pitch between supply lines. The small black dots represent locations to tap the clock signal. They are located midway between stages. Two stages are located where lines intersect in Figure 6.2. A single stage is located at the vertexes along the perimeter. Figure 6.3 magnifies the region around intersection points. Each of the lines in Figure 6.2 that connect FIFO stages represents four wires oscillating in quadrature. In both Figure 6.2 and 6.3 the red strips signify the rails carrying the supply voltage. The blue strips are ground.

The structure described is congruent with the power distribution mesh. This topology allows a designer to control strictly the environment surrounding each clock wire. The proximity of supply and ground to each clock wire is identical. The current return paths are identical. The

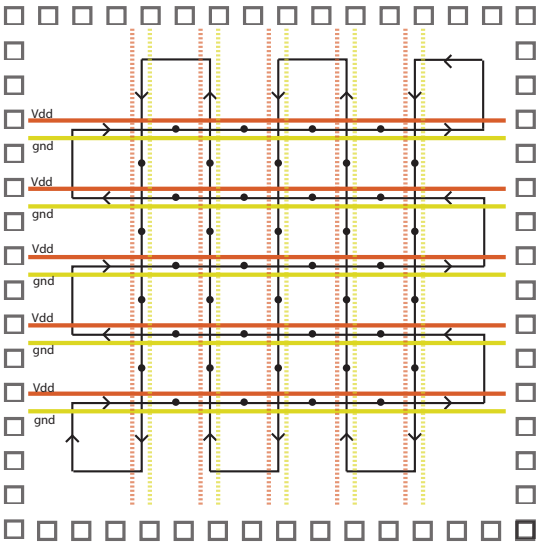


Figure 6.2: `Micropipeline` routed to form a clock grid

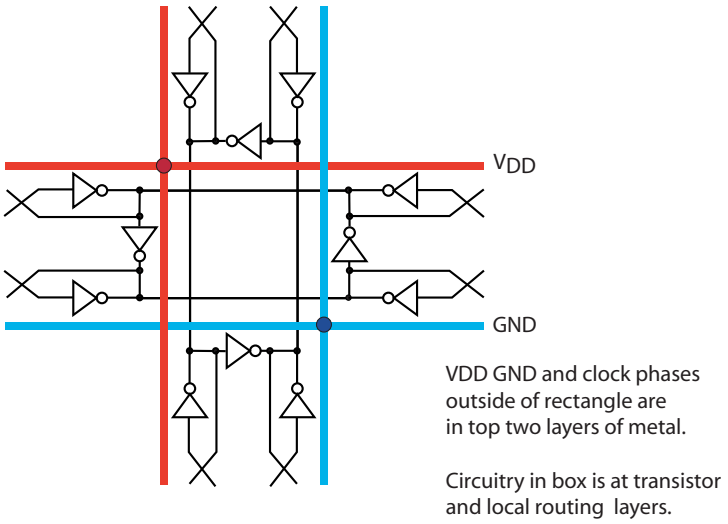


Figure 6.3: Intersection of `Micropipeline` rings for global clock distribution

clock signal is shielded from signal wires by supply and ground significantly reducing coupling between the clock and other signals (49).

6.2.2 Problems

Two fatal weaknesses limit the effectiveness of using this method to providing the system wide timing reference.

- Four phases is the minimum number of phases generated by a `Micropipeline` ring. While four phases of the clock signal is useful, the power required to distribute the four phases throughout the extent of the chip is excessive.
- Even though some of the overlapping stages are in physically adjacent regions of the silicon, they are maximally distant from each other as measured by their ring position and therefore have the most expected skew between each other, see Table 4.1.

  A FIFO ring routed in the same style as that shown in Figure 6.2 except with 11 rows and columns yields a grid with 288 clock taps. This structure approximates the 256 leaf clock tree structure described in *Digital Systems Engineering* (7). If we count stages along the route starting from the lower left corner, then for example stages 108 and 242 intersect, positionally separated by 144 stages. In fact there are seven positions in the grid where stages intersect that are separated by half the length of the ring. This is similar to the problem in a 256 leaf H-tree where there are 16 places where adjacent clock leaves are copied through maximally divergent paths.

Although `Micropipeline` FIFO rings are effective for distributing high precision timing signals, they lack the ability to effectively distribute a single phase clock over a large area.

6.3 Pulse control for global clock

6.3.1 Proposal

A ripple FIFO built from `Dynamic asP` FIFO control, could be routed between the supplies of the power distribution grid similar to the `Micropipeline` shown in Figure 6.2. Remember that when a FIFO built from `Dynamic asP` is half occupied, the signals on the wires that connect stages oscillate in phase. This feature is attractive and addresses the first problem encountered with the proposed `Micropipeline` design. If a `Dynamic asP` FIFO distributes a clock signal, it does not expend superfluous power distributing extra phases. Note that the stages handshake with each other using a single wire as opposed to four for `Micropipelines`. This is important because these wires must connect stages located far apart and charging the capacitance of the interconnect between stages consumes a large fraction of the power.

6.3.2 Problem

Unfortunately the second problem identified above while using `Micropipeline` FIFO rings to distribute a clock signal remains. Stages that are physically close are positionally distant in the ring. Worse still, results from Chapters 4 and 5 show that a FIFO ring built from `Dynamic asP` control suffers more skew between stages than `Micropipeline` control.

6.3.3 Solution

Handshaking with four stages instead of two addresses this problem. Making each stage perform handshakes with four neighboring stages instead of two drastically reduces the number of stages used to distribute the clock. Instead of performing handshakes with the control stages to the left and right, the control circuitry handshakes with stages positioned along horizontal and vertical. For simplicity I refer to these locations by the four cardinal directions North, South, East and West.

Rather than using a serpentine route of 288 stages to clock a chip in a grid, a 12×12 grid of control built in this style, shown in Figures 6.4b, could be employed. In the former case, stages are separated a maximum of 144 stages, in the later case stages are separated by a maximum of 24 stages.

This control works well as long as the phases on the state conductors on the four inputs to the control stage stay within the range over which the logic gates in the stages have a variable delay. Beyond this range the gates cannot exert a corrective force on the different phases. Stable oscillating patterns then exist where the stages remain out of phase with respect to each other.

A solution is to replace the logic gate with a phase mixing gate. The phase mixing gate asserts when the cumulative average potential of the four inputs exceeds some threshold. This last step of evolution from `Dynamic asP` control to a structure used for single phase clock distribution is shown in Figure 6.4c. Note that the inverters shown in Figure 6.4 are logical inversions and can be implemented by any odd number of inversions.

The clock distribution structure built from the control described is named the Distributed Clock Generator, or `DCG`. A 2×2 section from a `DCG` is shown in Figure 6.5. Notice the `DCG` uses two types of control stages. The Pull-up stage synchronizes the four state conductors it senses on the rising edge. The Pull-down stage synchronizes the four state conductors on the falling edge. Also notice that the `DCG` is not a number of scattered oscillators whose outputs are shorted. Rather the `DCG` is a single oscillator that is spread over the surface of the chip.

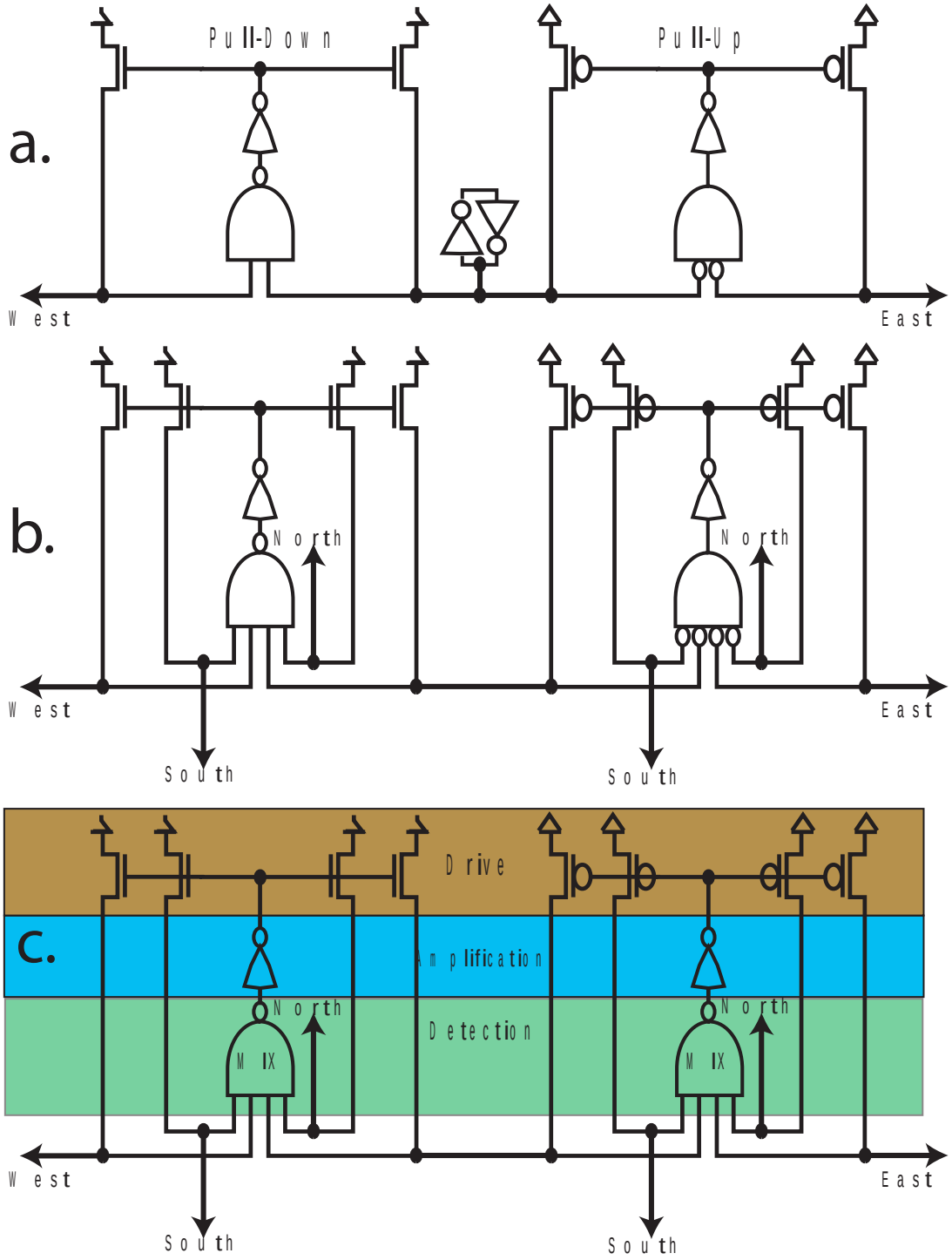


Figure 6.4: Evolution from `Dynamic asP` to the `DCG`

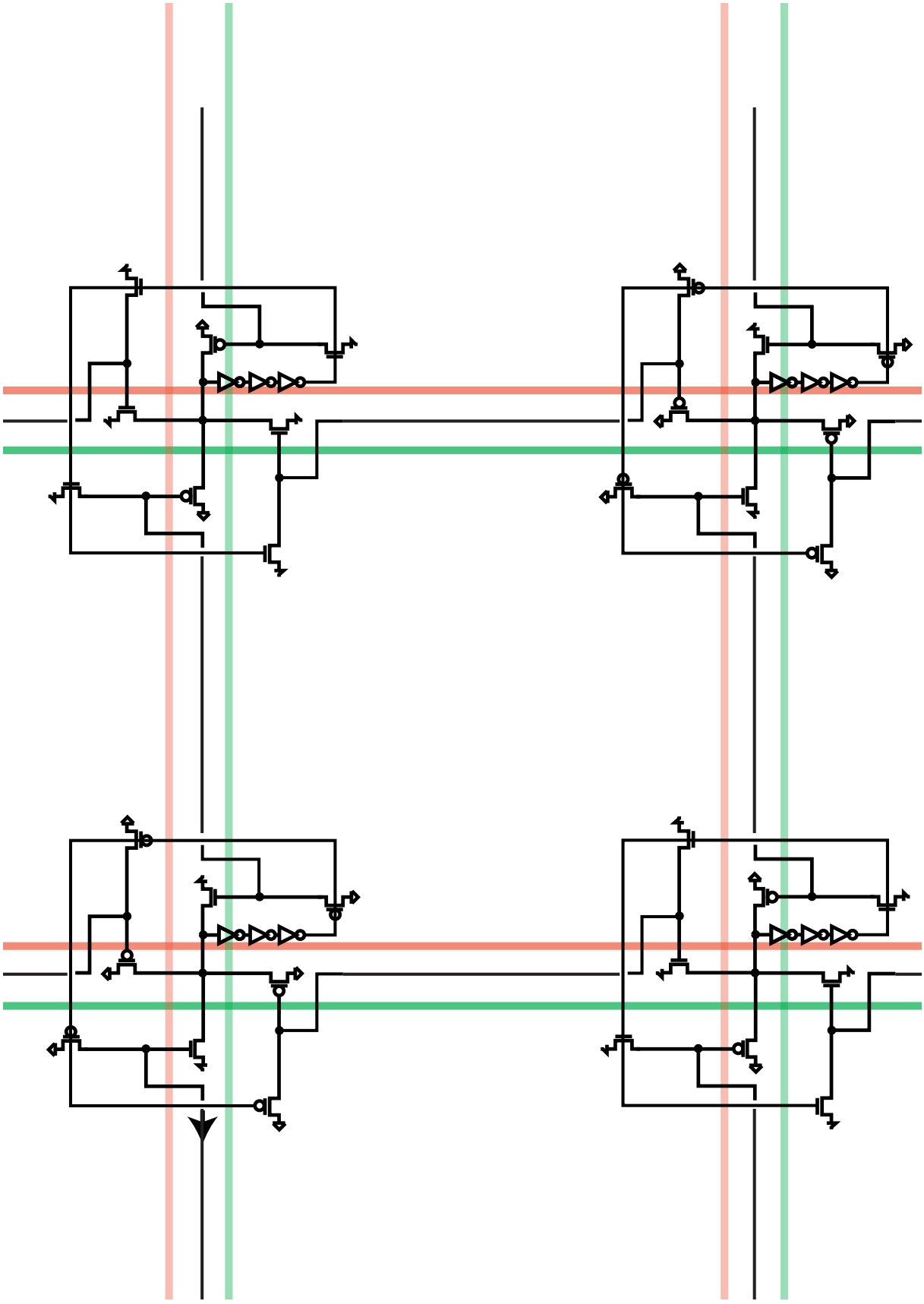


Figure 6.5: 2 x 2 section of a DCG grid

6.4 Distributed Clock Generator

6.4.1 The control stages

Each control stage can be divided into three components. These three components are distinguished in Figure 6.4c.

Detection

The detection component is the phase mixing gate. The phase mixing gate uses two PMOS and two NMOS transistors and is detailed in Figure 6.6. When used in the Pull-up stage, a falling transition signals that the cumulative state of the four state conductors are set LO and should be reset HI.

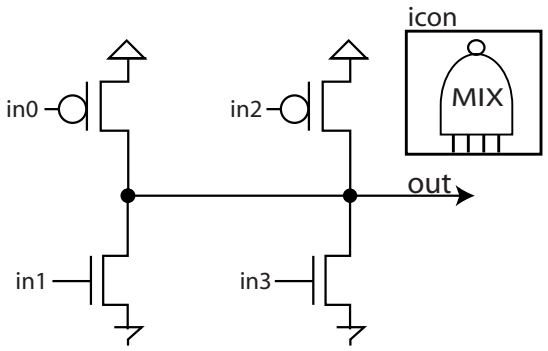


Figure 6.6: Detection component, the phase mixing gate

The falling transition is a result of two actions. When the state conductors sensed by $in1$ and $in3$ go HI, the NMOS transistors start conducting charge to the output of the phase mixing gate. When the state conductors monitored by $in0$ and $in2$ go HI, the PMOS transistors stop conducting charge to the output of the phase mixing gate and cease to resist the falling transition.

When the four state conductors oscillate in phase, the phase mixing gate is indistinguishable from two inverters connected in parallel. When the four state conductors are out of phase, the PMOS and NMOS transistors oppose each other. The delay of the mixing gate from an early arriving clock signal is long while the delay of the mixing gate from a late arriving clock signal is quick because the potential is removed from the supply voltage when the input arrives and less charge needs to be conducted to move the voltage over the threshold voltage.

Using full inverters rather than single transistors to sense the state conductors produces similar results. If inverters are employed then 8 transistor drains share the node that makes the phase mixer output. These 8 transistors should achieve the same electrical amplification as the phase mixer described by the 4 transistor mixer. The 8 transistors would be half the size of the transistors in the 4 transistor phase mixer. These transistors are the narrowest transistors in the circuit. Transistor imperfections are typically constant, not proportional errors. This being the case, transistor mismatch affects the transistors in the 4 transistor phase mixer less than in the 8 transistor mixer.

When the four state conductors are out of phase, then the transistors in the logic gate burn static power. Fortunately, in the suggested topology, these logic gates are four stages of amplification removed from the large drive transistors that charge the state conductors. Assume that the $\mathbf{EE}$ for each stage in the control elements in the clock system is chosen to be 3. This is an aggressive value. The current at the output of the four input logic gates are then about three to the fourth power, or $1/81$, smaller than the currents of the drive transistors. This static power is effectively in the noise of a sub-micron process. Less aggressive electrical step-ups result in even less significant static currents.

Amplification

The amplification component has an odd number of inverting amplifiers. The amplification component serves three purposes.

1. The amplifiers take the relatively weak assertion signal provided by the detection circuitry and amplifies it to a strength that will drive four transistors capable of driving interconnect of lengths on the order of a millimeter.

2. The amplification component is responsible for resetting the clock network. The first amplifier in the amplification component is implemented with an asymmetric NAND gate and receives the $start$ signal. One input of the NAND gate connects to the output of the Detection component. The other input connects to the $start$ signal. The $start$ signal input connects to the NMOS transistor that connects to ground. This transistor can be made large to minimize the pull-down series resistance of the NAND gate. The PMOS transistor

that connects to the $start$ signal can be made minimum width because transitions caused by this transistor are not critical and its drain typically just loads the output node.

Before the start signal asserts, the output of this NAND is HI. Pull-down stages are forced to drive all the state conductors LO while Pull-up stages disable their drivers.

3. Some number of the remaining amplifiers implement speed control functionality. A simple speed control places a series NMOS transistor in the pull-down path of one or more of the inverters in the amplification component. This transistor limits the amount of current available to discharge the gate's output node. Debilitating the ability of these inverters to source current slows the clock frequency.

   Three is the most practical number of amplification stages. One stage of amplification results in cycle times that are too aggressive. A significant amount of skew begins to accumulate in five or more amplifiers. A detailed schematic of an Amplification component is shown in Figure 6.7.

Drive component

A single transistor drives each of the four state conductor. The gates of the four drive transistors occupy the same electrical node. This ensures that the four drive transistors act in unison and synchronize the signal on the four state conductors on the rising or falling transition. The drive transistors are left unencumbered from any speed control or logic. Their sole responsibility is efficiently driving the state conductors.

Figure 6.7 details the logical circuit shown in Figure 6.4c.

6.5 Design

6.5.1 Gate sizing

Below is a six step algorithm to designing a DCG.

1. Choose a single value for the **EE** per stage. Large values result in low power designs but result in signals with much slew. Slewed signals are vulnerable to noise. Reasonable values fall in the range of 3 to 8.

2. Choose a value for the PMOS/NMOS transistor width ratio, γ. The best options are 1, 1.5 and 2. A γ of 1 results in Pull-up and Pull-down stages having the same footprint

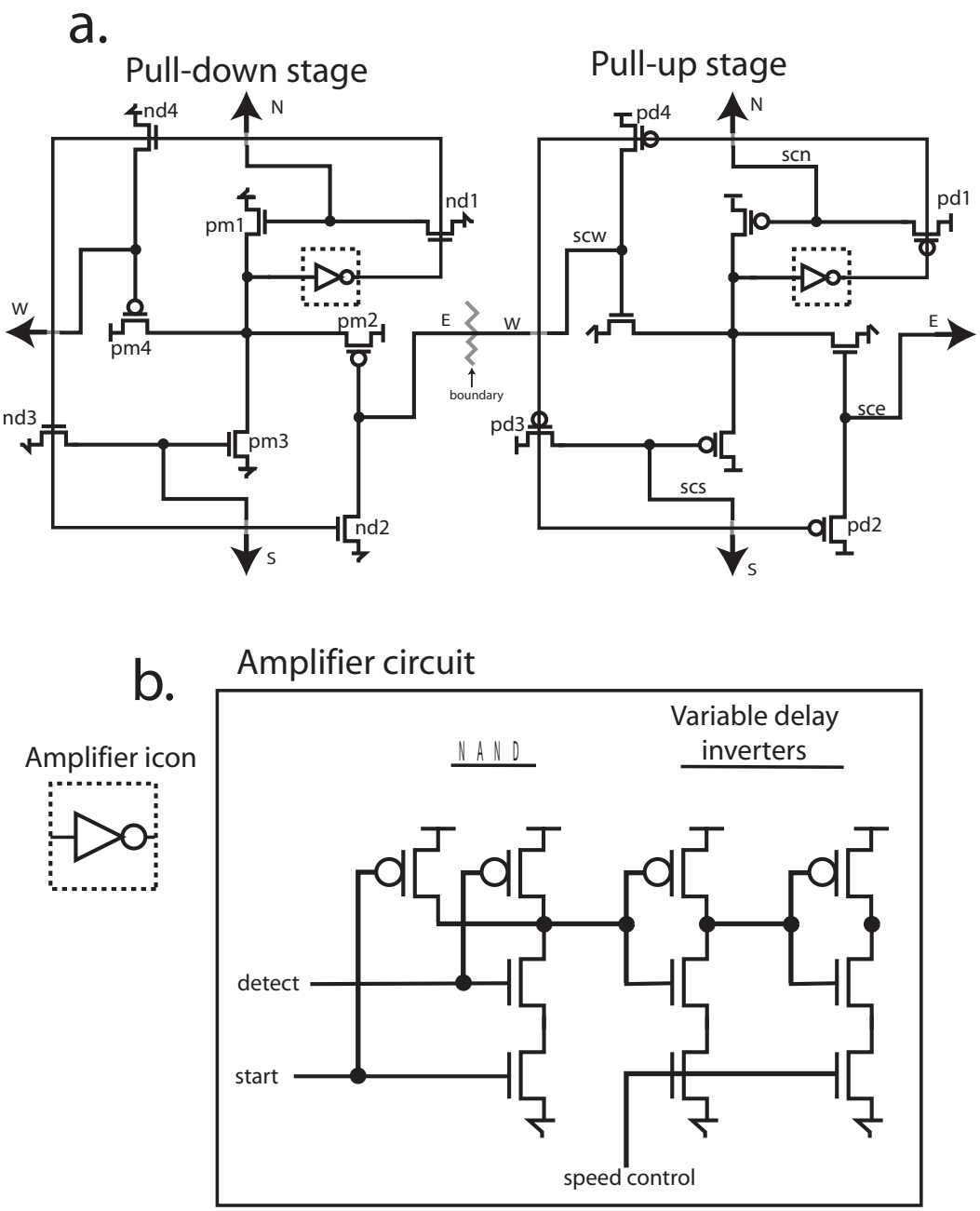


Figure 6.7: Pull-up and Pull-down control stage in detail

in the silicon. This choice is appealing from a geometric and regularity argument. Unfortunately, unless the **EE** per stage is small, the rising transitions can be anemic and vulnerable to noise.

The minimal average delay for the rising and falling transition of an inverter is achieved with a γ value equivalent to the square root of the conductivity ratio of the electron doped and hole doped silicon used to construct the transistors (39). This optimal value is about 1.6 in the process used for simulations in this thesis. 1.5 is a good approximation of this value.

A gamma of 2 balances the need for speed, healthy slew rates and circuits size.

3. Build a Spice model of the Pull-up and Pull-down control. Connect all four terminals of both control types together with a model of the interconnect of the correct length between the terminals. A simple interconnect model is discussed later.

   Each gate's size should be given a parameter in Spice to be a function of the size of the PMOS drive transistor. The last inverter in the amplification component of the Pull-up stage should have a total transistor width equivalent to the width of the four drive transistors divided by the **EE**. The next stage should be reduced by another factor of the **EE**. The transistors in the phase mixer should have a total size equal to the size of the PMOS drive transistors divided by the **EE** raised to the fourth power. The transistors in the Pull-down stage should be scaled by $1/\gamma$.

4. Sweep the size of the PMOS transistor until the `DCG` hits the target cycle time.

5. Adjust the duty cycle by either adjusting the speed control section of one type of control, or by varying the chosen **EE** in one of the control types. If the duty cycle is too long, the gates in the Pull-up stage need to be made faster relative to the gates in Pull-down stage. Either the **EE** of each stage in the Pull-up stage is reduced or slow the gates of the Pull-down stage using the speed control.

6. If the voltage swing at the clock output is insufficient, the reset time is too quick and the **EE** of each needs to be reduced.

7. Duplicate the resulting Pull-up and Pull-down control stages across the surface of the chip.

6.5.2 Interconnect optimization

The regularity of the `DCG` allows the interconnect to be optimized to a degree impractical in a clock tree. The dimensions of the straight interconnect or state conductor that connects stages

need only be designed once and then these dimensions are duplicated many times throughout the design. Tapering the width of a long interconnect along its length yields marginal but real speed and power savings (3). Optimal tapering improves delay by up to 8% (2).

The Elmore delay provides a simple and fairly accurate delay model for distributed resistances and capacitances (2). The product of a capacitance multiplied by the cumulative resistance between the node where the capacitance is located and the voltage source is a single term in the Elmore delay. The total Elmore delays takes the sum of all the terms relating the downstream resistance multiplied by the capacitance at each capacitive node up to the point where the delay is being computed. Figure 6.9 shows how the Elmore delay is computed up to the nodes $n1$ and $n2$.

A two parameter model is used to describe the capacitance of the state conductor. A single parameter describes the unit area resistance of the interconnect. Simple resistance and capacitance models exist to describe the drive transistors. Figure 6.8 illustrates how the current path between the drive transistors in a DCG is modeled for use in the Elmore delay model.

A near ideal homogenous wire width value is found using the function describing the lumped RC delay of the drive transistors and the interconnect between them. Figure 6.8 illustrates the steps necessary to find an optimized width for the interconnect assuming a uniformly sized conductor. The wire that achieves the minimum Elmore delay has a width that varies over its length.

A near ideal tapered wire is found by dividing the wire into some number of segments, see Figure 6.9. Each segment is assigned a width that is equal to the optimum uniform width wire plus or minus some small number of minimal wire width increments. The Elmore delay is then easily solved for all wire width possibilites. The combination producing the lowest power while using the same **EE** or alternatively the same chosen **EE** while using the lowest power is then used throughout the DCG.

The optimal uniform state conductor width of length 1333μm when the PMOS driver was 42μm wide and the NMOS driver was 21μm is 1μm. The algorithm above was coded into a simple C program. The program explored the Elmore delay that results with a 9 segment wire, each wire segment can assume of six widths between 0.8 and 1.4μm wide. All 282,475,249 possible wire width combinations were explored on my laptop in about a half hour. The resulting widths were 1.2, 1.1, 1.1, 1.2, 1.2,1.2,1.2 1.3,1.3. These are the widths of the state conductor starting from the side connected to the NMOS drive transistor. The replacement wire has an Elmore delay of 495 ps versus 525 ps for the wire of uniform width.

The ideal tapering tends to be 'hour-glass' shaped when interconnect is driven from both ends. Interconnect driven from a single end tapers from wide to narrow as you move from the driver.

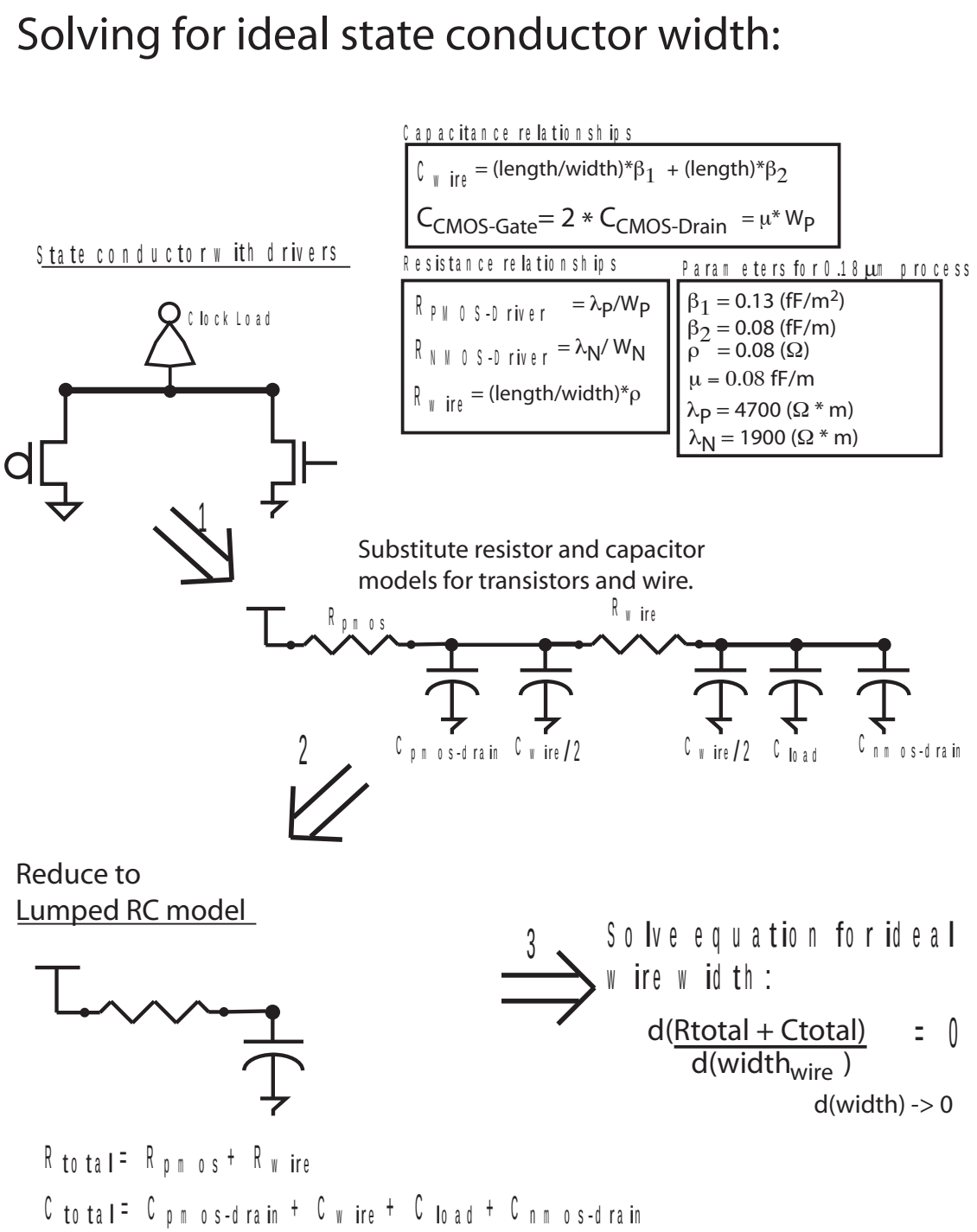


Figure 6.8: Steps for finding the ideal width for state conductors of uniform width

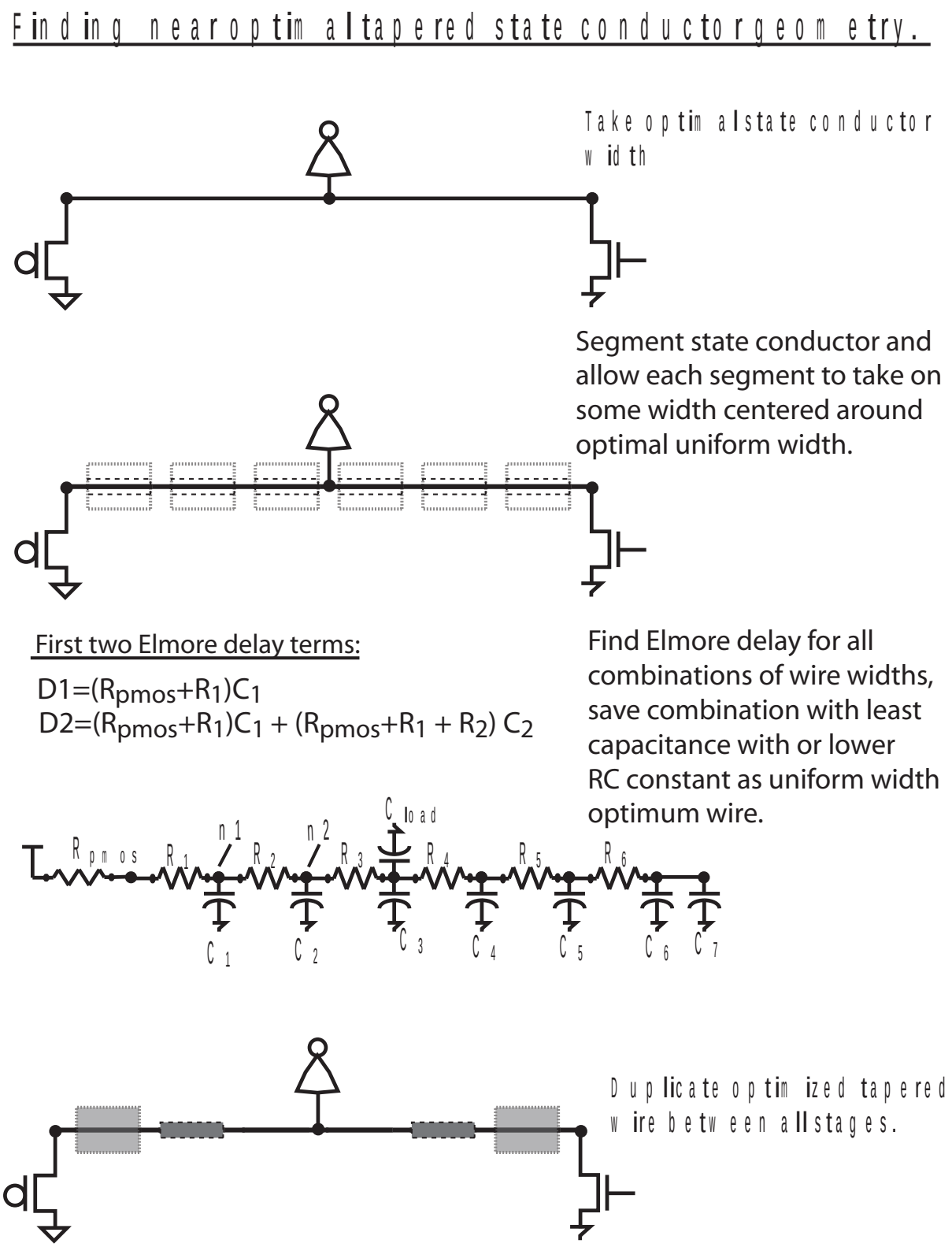


Figure 6.9: Finding a near ideal tapered width state conductor

6.6 Mechanics

6.6.1 Initialization and starting

When the $start$ signal is LO or unasserted, a HI potential is applied to the gates of the transistors in the drive component of each stage. Consequently all state conductors are driven LO. The DCG begins oscillating when the $start$ signal asserts. It is not critical that the $start$ signal arrives at each stage simultaneously. If the $start$ signal is greatly skewed, then the state conductors initially oscillate out of phase but they quickly lock into phase soon after the $start$ signal distributes to all stages. The mechanism that forces the state conductors into the same phases is the variable delay of the mixing gate in the detection component of the control.

Figure 6.10 shows two windows of a waveform viewer. The top window shows a number of start signals that are randomly chosen from a uniform distribution of a single clock period. The signals are sent to the control stages in a 12×12 DCG. Notice that the various clock signals are locked with respect to each other within 10ns.

6.6.2 Variable duty cycle on state conductor

In addition to changing the relative delays of the Pull-up and Pull-down stages, the duty cycle can be chosen by the location on the state conductor used to charge the clock driver. The duty cycle of the clock cycle varies depending on where the signal is tapped along the state conductor. The duty cycle is greatest at the point where the state conductor attaches to a Pull-up stage. This point is charged HI before any other point on the conductor and discharged LO last.

If the time of flight for the signal on the state conductor is long compared to the rise time of the signal, then there is a large range of duty cycles along the state conductor. Widening the state conductor diminishes the duty cycle variation along the length of the wire but results in greater power consumption. Having greater or lesser variation in the duty cycle doesn't hold advantage.

6.6.3 Synchronization

When the combined potential on the four state conductors reaches a potential that causes the detection component to assert, the amplification component amplifies the asserted signal and causes the drive transistors to charge the state conductors in the opposite direction. Because the gates of the four drive transistors share the same electrical node, the signals synchronize on the edges of that transition irrespective of the phases of the signals when they arrived at the stage.

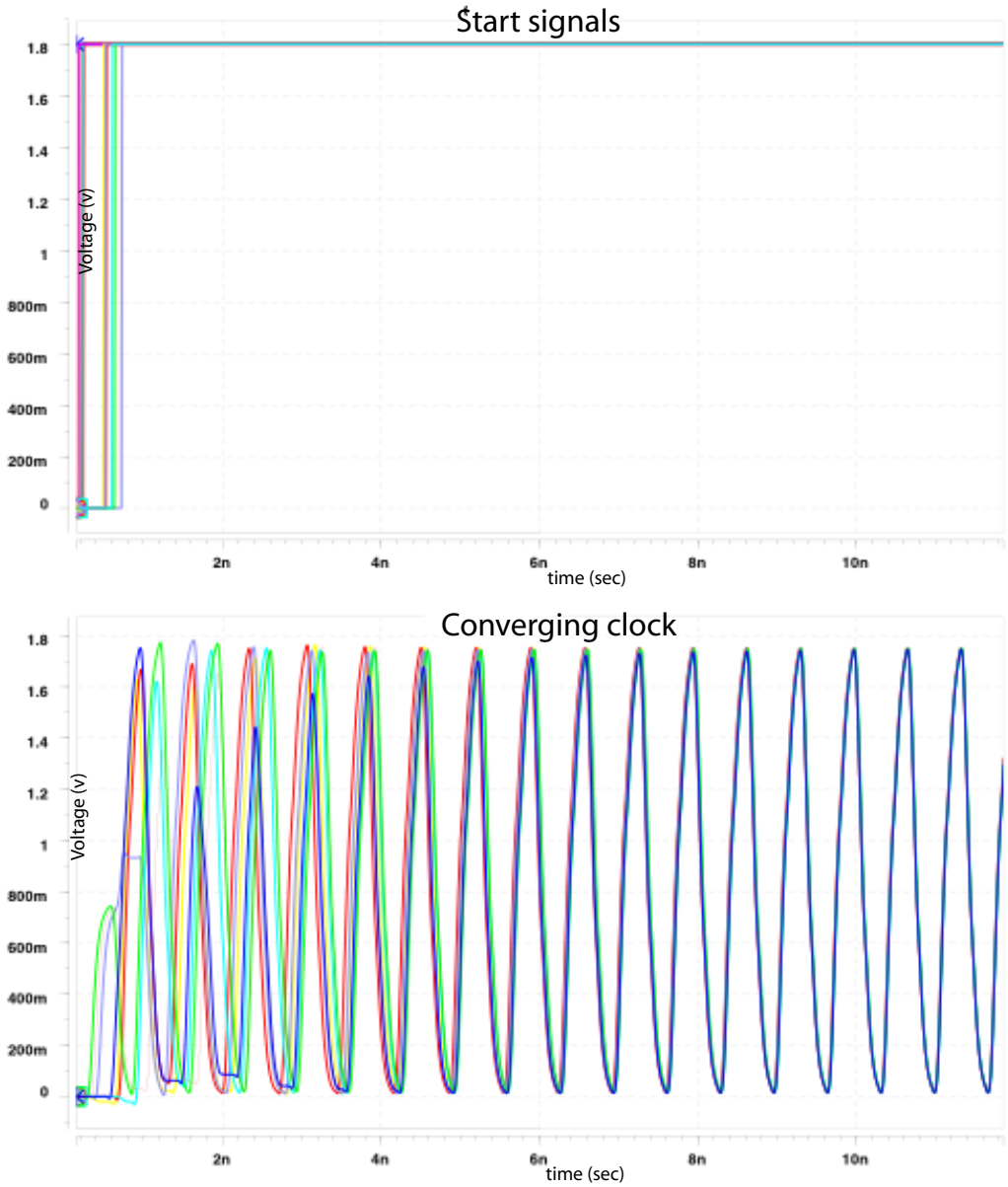


Figure 6.10: Top plot shows start signals asserting at random times, bottom plot shows clock taps converging

6.7 Hazards

6.7.1 Timing constraints

The drive from the state conductors is removed three gate delays after the detection circuitry signals that the potential on the state conductors has changed. The drive transistors first charge the state conductor wires near the detection circuitry. Then the charge spreads along the length of the wire. A small hazard exists when an insufficient amount of charge is sourced onto the state conductor to charge the whole length of the state conductor before the drive transistors are disabled. This is not a concern if the design methodology is disciplined and ensures that each node in the DCG has the same **EE**. Then the time allotted to charge the wire is always inversely proportional to the capabilities of the drive transistors.

6.7.2 Supply variations

Notice that current moves in one direction on the state conductors. A large amount of charge is sourced by the Pull-up stages and sunk by the Pull-down stages. If the power supply conductors are resistive then supply gradients exist between the two types of control. The supply voltage near the Pull-down stages is greater by an amount equivalent to the IR drop in the current return paths. As the state conductors increase in length and the frequencies increase, the width of the conductors carrying the supply and ground must increase as well.

6.7.3 Mode lock

Using a phase mixer as the detection component in the the DCG raises the concern that mode lock might exhibit itself in the DCG. Mode lock is a stable system equilibrium in which the phase averaging mechanism used to couple the oscillators settles the oscillators in a non-zero phase relationship(29). If two of the phases contributing to the average are of equal but opposite magnitude then an undesired and stable phase equilibrium occurs.

The criteria for avoiding mode lock in a two input phase mixer stipulates that the delay of the phase mixer must increases linearly for phase differences between +/-90° and decreases between 90°and 270°. This error function is easy to plot and visualize for a two input phase mixer but is substantially more difficult to formulate and visually represent with four inputs.

This thesis presents an empirical argument to show that the DCG is not susceptible to mode lock. Using HSpice 100 simulations were ran of a 4x4 DCG whose clock period was 675ps. A test setup ensured that each stage received its own $start$ signal. The individual $start$ signals asserted at a time picked from a uniform distribution between 0ps and 675ps. For this simulation transistor mismatch was not modeled. In all cases, the state conductors locked into phase by

20ns. The large number of initial phase relationships between the state conductors that did not excite the mode lock condition gives reasonable assurance that this behavior is not exhibited by the DCG.

6.8 Implementation

A table describing electrical specifications for a typical clock distribution problem is found in the text book *Digital Systems Engineering* (7). This thesis reproduces the table in Section 2.5.1. This section describes a Distributed Clock Generator designed to provide a clock signal to a chip with these parameters. The target cycle time is 9 FO4 inverter delays or 675ps/cycle.

6.8.1 Parameter choices

The PMOS to NMOS transistor width ratio, γ in the DCG is 1.5.

I chose to limit the electrical amplfication on each node in the clock network to 3.6. Values lower than this burn increasingly more power. Values greater than this become increasingly vulnerable to noise coupling. A disciplined clock distribution design methodology places a maximum electrical step-up of about five on all nodes of the circuit (30). Electrical step-up is the ratio of the current sourcing capabilities of a gate that is driven, divided by the current sourcing capabilities of the gate driving it.

6.9 Performance

The relative performance of the DCG and a canonical H-tree are compared. I assume that the clock load is spread homogeneously over the surface of the microprocessor. The DCG has 264 clock taps. The H-tree has 256. The distance between stages in the DCG is 1333μm. The final clock load is driven by two series inverters. Each inverter has a **EE** of 3.

6.9.1 Skew

Figure 6.11 is an abstract representation of a 12 x 12 DCG grid constructed to clock the chip described by Table 2.5.1. The circles represent Pull-up stages, while the squares represent Pull-down stages. Each circle and square is labeled with two numbers that refer to its row and column. Vectors going from the Pull-up to the Pull-down stages represent state conductors. The arrows indicate the direction of current flow.

The skew measurement performed was relative to the vector coming from the stage in row 5 and column 4. The skew is reported as the one sigma deviation skew. The numbers on the figure next to each vector are the absolute value of the average of the deviation of the rising edge and the absolute value of the deviation of the falling edge for the clock tap on that state conductor. The data was culled from a 30 trial Monte Carlo simulation in HSpice. The The same transistor mismatch model used for these simulation are the same used to draw Figure 2.4. The skew is measured relative to the rising edge

Notice that the skew increases the greater the distance from the reference point. Also the increase is continuous, unlike clock trees where at some point two adjacent clock taps have maximally different paths from the root.

The simulations model a microprocessor built in a 0.18μm process. Only 60% of the chip can be reached in a single clock cycle with the clock (19). Destinations beyond this distance need to be re-synchronized in a register. Only 30% of 0.13μm chip can be reached.

The worst case local skew is where two adjacent clock taps route through maximally different routes. This route is through a string of eleven different inverters. Each branch has two inverters, the first inverter drives the second inverter. The second inverter drives into a branching point and drives two copies of the input to the next driver and the interconnect between. The first driver has an **EE** of 3 while the second driver has an **EE** of about 6. The one sigma deviation skew at the adjacent clock taps derived from maximally different routes is 7ps.

6.9.2 Power

The DCG built for this task required 24,912μm of transistor width. A transistor presents about 1.9fF of capacitance for each micron of transistor. The approximate total of capacitance due to transistors is 47.3pF. The DCG had 266 state conductors each 1333μm long and 1μm wide. The interconnect contributes 74pF of capacitance.

The H-tree required 37,520μm of transistor width. This represents 71.2pF of capacitance. The H-tree required 368,000μm of interconnect length as opposed to 353,000μm of length for the DCG. All levels except for the final 'H' used 3μm wide wire. The total wire capacitance for the H-tree is 104.8pF.

The total H-tree capacitance is 176pF while the total DCG capacitance is 121pF.

6.9.3 Speed control

The simple speed control circuitry yields a wide range of clock periods. The relationship between speed control voltage and the resulting period is shown in Figure 6.12

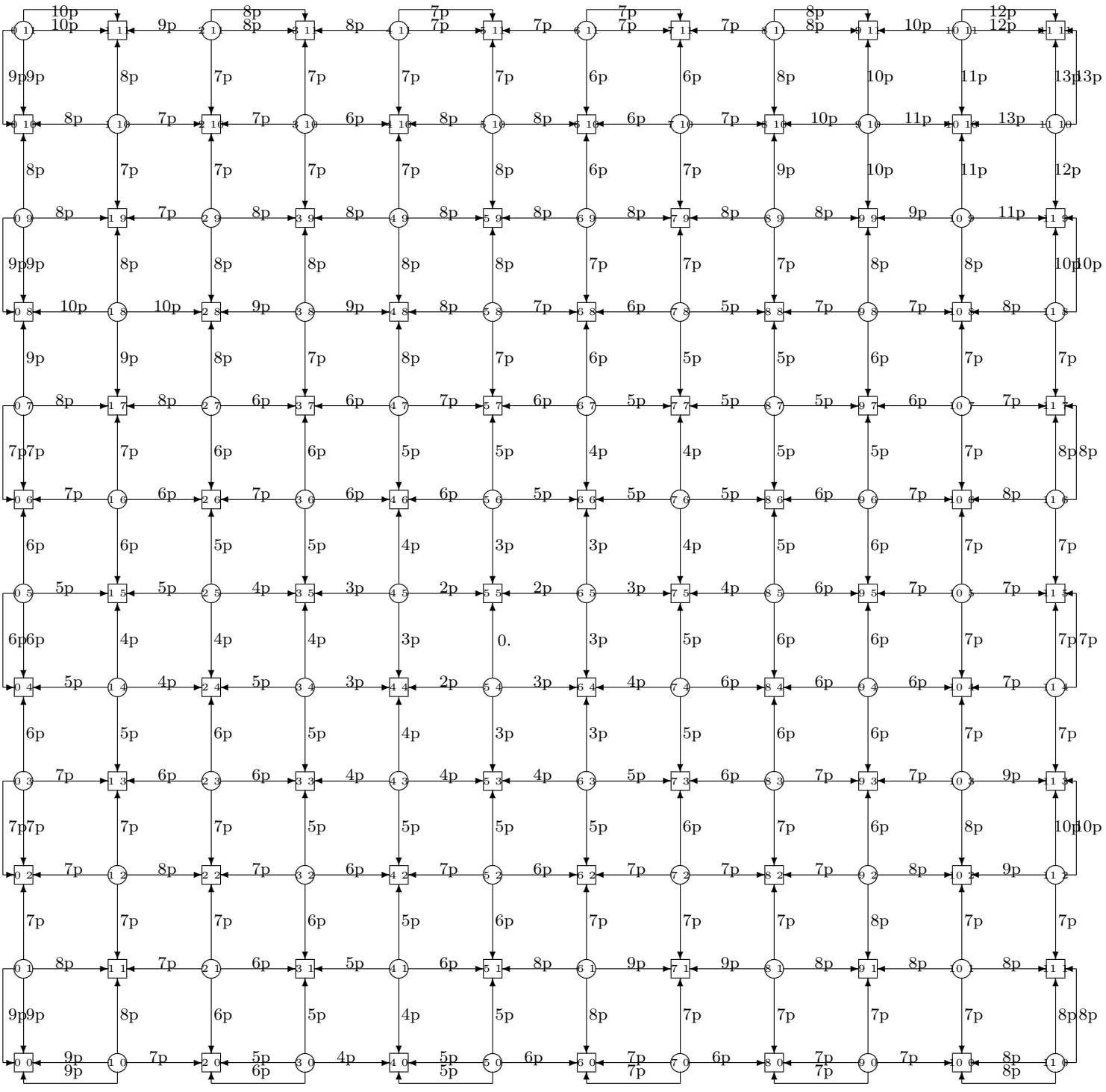


Figure 6.11: Skew expectation over the surface of microprocessor due to transistor mismatch

6.10 Jitter

Each transistor connected to the suppply voltage in the DCG was connected to a unique dirty supply. The dirty supplies were created by connecting the dirty node to two voltage sources, each through a very large transistor. The large transistor switched in the dirty source at some point in time to observe its effect. The first supply swung between 0.9×Vsupply and 1×Vsupply with a frequency of 9×FO4 delays. The second supply had the same voltage swing but the period was chosen from a uniform random distribution between 150ps and 250ps. The first supply simulates noise contributed from the output resistance of the supply while the second supply simulates higher frequency noise sources. HSpice performed a thirty trial Monte Carlo simulation and the 1 sigma expected jitter offset between clock taps found in the same row but in adjacent columns was 1.3ps.

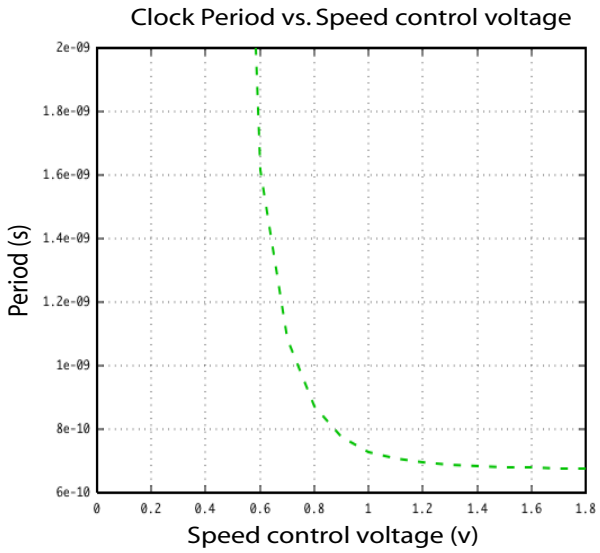


Figure 6.12: DCG period as a function of speed control voltage

For comparison the same test was run using a string of amplifiers with an electrical gain of four between stages. Interestingly, when the circuit used the dirty voltage supply the signal was filtered before the eleventh amplifier. Eleven is the number of stages of amplification from root to leaf in the comparison between the DCG and the H-tree above. When the clock was slowed from 9 to 11 FO4 periods the signal correctly amplified through the inverter string. The 1 sigma mean jitter offset here was 0.1ps. This is significantly better than the DCG but it should be noted that the noise in the H-tree would be greater because it makes greater demands on the power

supply. Despite the H-tree requiring more power and therefore likely having a noisier supply, the simulations modeled the same amount of noise in both systems.

6.11 Conclusion

`Dynamic asP` provides the foundation for an efficient low skew and low power clock distribution solution. The Distributed Clock Generator retains the drive efficiency and spartan use of hardware of `Dynamic asP`. ANDing inputs is eschewed for averaging of inputs. The control is branched into four directions instead of two. The `DCG` does not exhibit mode lock. Its topology integrates well with existing topologies. `DCG` design obeys good design principles: design once and duplicate. This allows for high optimization of the designed part.

Chapter 7

Conclusion

7.1 The Big Idea

This thesis presents novel methods for providing high precision timing references on a VLSI chip. Ironically the foundation for these methods are circuits developed by the community of asynchronous circuit designers. These circuit topologies were .built to work in systems with no notion of time. Self-timed circuits maintain ordering between data tokens with the bare minimum of logic and with the maximum electrical efficiency.

I first investigated how tokens distribute themselves throughout a free running FIFO while working at Sun Microsystems. We watched tokens sometimes gather and sometimes bunch in FIFO rings built with both `GasP` and `Micropipeline` control. Ivan Sutherland prompted me to discover the mechanics behind what was happening but the answer was elusive. Winstanley and Greenstreet's paper (44) seemed to put the question to rest but on further inspection I realized that they hadn't actually answered the question, they controlled the phenomenon but didn't really explain it.

Observing that the delay of the self-timed control circuit was a function of the separation time of its inputs and not just which input arrived last, enabled this work. The Separation curve illustrates this behavior. Essentially this thesis describes a mechanism that causes control tokens to repel each other. Understanding the phenomenon allowed for it to be controlled, amplified, and finally applied to various applications like high speed sampling and clock distribution.

The solution is a nice combination of analog and digital electronics. The circuits can be viewed as digital. The handshaking signals use digital protocols. The logic gates implement boolean logic functions. I see no reason why the circuits can not be implemented employing standard tools used in digital circuit flows. But the circuits can also be seen as being primarily analog. The wires that conduct the handshaking signals are really just feedback loops. The gates are high precision phase detectors and actuators. The delay of the gate is a variable function of

the separation time of its input. Depending on the mode of operation, the logical values may never fully transition to either rail voltage. Of course any digital circuit viewed closely enough is really analog. This thesis simply elevates and exploits the analog qualities of digital circuits.

7.2 Summary

Chapter 1 introduced the basis, task, and method for this thesis.

Chapter 2 described the sources of timing imprecision and the limits using conventional timing methods. Existing solutions were surveyed, classified, and summarized.

Chapter 3 offered a tutorial of self-timed circuits. The basic handshaking protocols were discussed. Previous research that provided the foundation to this thesis was reviewed and credited.

Chapter 4 dusted off and re-designed `Micropipeline` control circuits and put them to a new use. The `Micropipeline` handshake protocol allows for control circuits that are symmetric around many axes. Symmetry yields robust operation and simplifies understanding. The efficiency of the Analog C-element raised the performance bar significantly over existing digital methods for generating high frequency signals. Rail to rail voltage transitions at twice the frequency found in ring of three unloaded inverters and multiple phases with zero systematic skew allows for events to be localized to far smaller periods than previously possible in the digital domain.

Chapter 5 was a dutiful investigation into whether self-timed circuits that communicate using pulse-protocols can achieve the same high precision found in `Analog Micropipelines`. The investigation yielded a negative result.

Chapter 6 took the lessons learned in the previous chapters, made some alterations to the `Dynamic asP` control and developed an effective and efficient clock distribution system. The method holds promise to offer single phase signal distribution at a fraction of the power and skew of traditional clock tree distribution methods. Simulations suggest a substantial power reduction in distributing a clock signal across a chip for a certain amount of skew. Future work includes building a Distributed Clock Generator.

7.3 Future work

I see much “low hanging fruit” remaining using self-timed circuits as high precision timing circuits.

Identifying applications that use the circuits presented here in a bi-modal environment where they are either acting synchronously or asynchronously might be fruitful. These circuits deliver

synchronous timing information when advantageous and asynchronous information when allowable.

Much research remains investigating how to shape the Separation curve to excite various modes of operation. The principles developed in this thesis should easily map to alternative technologies like bi-polar or GaAs, but this remains a conjecture.

I've done preliminary research into using a separate supply for the keeper circuit in the Analog `Micropipeline` to extend the range of tokens that lock in a FIFO ring of a given size. This also yields greater frequencies than single supply voltage circuits and greater locking ranges. I have not adequately investigated the robustness or feasibility.

7.4 Final thought

On the surface, this work could be described as using asynchronous circuits to achieve synchronous solutions but this is a facile description. Synchronous design is not a description of reality but rather a statement of an assumption. All registers don't simultaneously update in a synchronous system, but if it is assumed they do, communication and the hardware necessary to communicate is drastically simplified. Safely communicating with the synchronous assumption requires spending the time necessary to confidently assume timing and data signals have performed the steps necessary to correctly progress. This thesis doesn't yield synchrony. The ideas in this thesis yield timing precision at a lower cost. Increased timing precision means spending less time gaining the confidence necessary to safely progress. Increased precision directly translates into increased performance.

The distributed self-timed circuits and the scattered buffers in a clock tree both trust that identical gates in various locations will have similar delays. The critical difference between the two is that self-timed circuits use feedback that dynamically corrects timing mismatches. The circuits presented in this thesis incorporate timing feedback with no compromise in efficiency. The self-timed circuit topologies provide a vehicle to distribute fine grained timing feedback with an efficiency honed over years by a legion of self-timed circuit designers.

Conspicuously absent from this thesis are results from real test chips. Test structures for both the `DCG` and Analog `Micropipeline` control rings are scheduled to be on a test chip for tape out before year end 2004.

APPENDIX — Publication list

This appendix contains a list of pertinent publications with me as author or co-author arranged according to chapters where the work is relevant.

Chapter 3 - Ripple FIFO basics

- **Predicting Performance of Micropipelines Using Charlie Diagrams** (9)
  Jo Ebergen, Scott Fairbanks, and Ivan Sutherland

Chapter 4 - Micropipelines

Papers

- **Analog Micropipeline Rings for High Precision Timing** (11)
  Scott Fairbanks and Simon Moore
- **A Counterflow Pipeline Experiment** (4)
  Bill Coates,Jo Ebergen, Jon Lexau, Scott Fairbanks, Ian Jones,Alex Ridgway, David Harris, and Ivan Sutherland
- **A FIFO Data Switch Design Experiment** (5)
  Bill Coates, Jon Lexau, Ian Jones, Scott Fairbanks, and Ivan Sutherland

Patents

- **One-hot Muller C-elements and Circuits Using One-hot Muller C-elements**
  US patent #6,486,700
  Scott Fairbanks and Ivan Sutherland
- **Method and Apparatus for Generating and Distributing a Clock Signal**
  US patent #6,400,230
  Scott Fairbanks

- **Two-stage Muller C-element**
  US patent #6,281,707
  Scott Fairbanks
- **High Speed Coupled Oscillator Topology**
  US patent #6,191,658
  Scott Fairbanks
- **Charge Sharing Selectors**
  US patent #6,160,422
  Scott Fairbanks
- **Charge Sharing Selectors with Added Logic**
  US patent #6,144,266
  Scott Fairbanks
- **Using Asynchronous FIFO Control Rings for Synchronous Systems**
  US patent #6,069,514
  Scott Fairbanks

Chapter 5 - Pulse protocols

Papers

- **FLEETzero: An Asynchronous Switching Experiment** (6)
  William Coates, Jon Lexau, Ian Jones, Scott Fairbanks, and Ivan Sutherland
- **GasP: A Minimal FIFO Control**[1]
  Ivan Sutherland and Scott Fairbanks
- **Two FIFO Ring Performance Experiment**(22)
  Charles E. Molnar, Ian W. Jones, Bill Coates, Jon K. Lexau, Scott Fairbanks and Ivan Sutherland

Patents

- **Asynchronous pulse bifurcator circuit with a bifurcation path coupled to control FIFO and first and second subordinate FIFO**
  US patent #6,574,690
  Scott Fairbanks and Charlie Molnar

[1] Winner of best paper award

- **Method and Apparatus for Latching Data within a Digital System**
  US patent #6,456,136
  Ivan Sutherland and Scott Fairbanks
- **Method and Apparatus for Asynchronously Controlling State Information within a Circuit**
  US patent #6,420,907
  Ivan Sutherland, Scott Fairbanks, and Jo Ebergen
- **Asynchronously Controlling Data Transfers within a Circuit**
  US patent #6,356,117
  Ivan Sutherland, Scott Fairbanks, and Jo Ebergen
- **Control Structure for a High-Speed Asynchronous Pipeline**
  US patent #5,937,177
  Charles Molnar and Scott Fairbanks

Chapter 6 - Clock Distribution

Papers

- **Clock Distribution with Self-Timed Circuits**
  11th IEEE International Symposium on Asynchronous Circuits and Systems
  Scott Fairbanks and Simon Moore

Patents

- **Method and Apparatus for a distributed clock generator**
  US patent application #20040108876
  Scott Fairbanks

Bibliography

[1] V. Agarwal, M. S. Hrishikesh, S. W. Keckler, and D. Burger. Clock rate versus IPC: the end of the road for conventional microarchitectures. In *ISCA*, pages 248–259, 2000.

[2] C.-P. Chen. Performance-driven interconnect optimization (dissertation). In *IEEE/ACM Design Automation Conference*, volume 34, 1997.

[3] C.-P. Chen, H. Zhou, and D. F. Wong. Optimal non-uniform wire-sizing under the elmore delay model. In *ICCAD*, pages 38–43, 1996.

[4] B. Coates, J. Ebergen, J. Lexau, F. Fairbanks, I. Jones, A. Ridgway, H. Harris, and I. Sutherland. A counterflow pipeline experiment. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 161–172, Apr. 1999.

[5] W. S. Coates, J. K. Lexau, I. W. Jones, F. Fairbanks, and I. E. Sutherland. A FIFO data switch design experiment. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 4–17, 1998.

[6] W. S. Coates, J. K. Lexau, I. W. Jones, S. M. Fairbanks, and I. E. Sutherland. FLEETzero: An asynchronous switch fabric chip experiment. In *Proc. International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pages 173–182. IEEE Computer Society Press, Mar. 2001.

[7] W. Dally and J. W. Poulton. *Digital Systems Engineering*. Cambridge University Press, 1998.

[8] Dehng and et al. Clock-deskew buffer using a sar-controlled delay-locked loop. *IEEE Journal of Solid State Circuits.*, 35(8):1128–1136, 2000.

[9] J. C. Ebergen, S. Fairbanks, and I. E. Sutherland. Predicting performance of micropipelines using Charlie diagrams. In *Proc. Intern. Symp. on Advanced Research in Async. Circuits and Systems*, pages 238–246, 1998.

[10] Y. Elboim, A. Kolodny, and R. Ginosar. A clock-tuning circuit for system-on-chip. *IEEE Trans. Very Large Scale Integr. Syst.*, 11(4):616–626, 2003.

[11] S. Fairbanks and S. Moore. Analog micropipeline rings for high precision timing. In *Proc. Intern. Symp. on Advanced Research in Async. Circuits and Systems*, pages 41–50. IEEE Computer Society Press, Apr. 2004.

[12] B. A. Floyd, C.-M. Hung, and K. Kenneth. Intra-chip wirless interconnect for clock distribution implemented with integrated antennas, receivers, and transmitters. In *ieejssc*, volume 37, pages 543–551, May 2002.

[13] V. Gutnik and A. Chandrakasan. Active gh clock network using distributed plls. In *IEEE Journal of Solid-State Circuits*, pages 1553–1560, 2000.

[14] D. Harris, editor. *Skew Tolerant Circuit Design*. Morgan Kaufmann Publishers, Inc., 2001.

[15] D. Harris and S. Naffziger. Statistical clock skew modeling with data delay variations. *IEEE Trans. Very Large Scale Integr. Syst.*, 9(6):888–898, 2001.

[16] I.Galton, D. Towne, J. Rosenberg, and J. Jensen. Clock distribution using coupled oscillators. In *International Symposium of Circuits and Systems*, volume 3, pages 217–220, May 1996.

[17] L.Hall, M. Clements, W.Liu, and G.Bilbro. A semidigital dual delay-locked loop. In *Conferenece on Advanced Research in VLSI*, pages 62–75, 1997.

[18] J. G. Maneatis. Precise delay generation using coupled oscillators. Technical Report CSL-TR-94-629, Stanford University, 1994.

[19] D. Matzke. Will physical scalability sabotage performance gains. In *IEEE Computer*, volume 30, 1997.

[20] C. Molnar and S. Fairbanks. Control structure for a high-speed asynchronous pipeline. *US patent*, August 10, 1999. #5,937,177.

[21] C. E. Molnar, I. W. Jones, B. Coates, and J. Lexau. A FIFO ring oscillator performance experiment. In *Proc. Intern. Symp. on Advanced Research in Async. Circuits and Systems*, pages 279–289. IEEE Computer Society Press, Apr. 1997.

[22] C. E. Molnar, I. W. Jones, C. Coates, J. K. Lexau, S. M. Fairbanks, and S. Sutherland. Two FIFO ring performance experiments. *Proceedings of the IEEE*, 87(2):297–307, Feb. 1999.

[23] D. E. Muller and W. S. Bartky. A theory of asynchronous circuits. In *Proceedings of an International Symposium on the Theory of Switching*, pages 204–243. Harvard University Press, Apr. 1959.

[24] V. Oklobdzija. Clocking and storage elements in a multi-gigahertz environment. *IBM Journal of Research and Development*, 47(5), Sept. 2003.

[25] F. O'Mahony, P. Yue, M. Horowitz, and S. Wong. Design of a 10GHz clock distribution network using coupled standing-wave oscillators. *IEEE Journal of Solid-State Circuits*, November 2003.

[26] T. On. Noise-constrained performance optimization by simultaneous gate and wire sizing based on lagrangian relaxation.

[27] M. Pelgrom, A. Duinmaijer, and A. Welbers. Matching properties of MOS transistors. In *IEEE Journal of Solid State Circuits.*, volume 24, pages 1443–1449, 1989.

[28] M. Pelgrom, J. Ruinhout, and M. Vertregt. Transistor matching in analog CMOS applications. In *International Electron Devices Meeting 1998*, pages 915–918, 1998.

[29] G. Pratt and J. Nguyen. Distributed synchronous clocking. In *IEEE Transactions on Parallel and Distributed Systems*, March 1995.

[30] Restle, P.J. et al. A clock distribution network for microprocessors. *IEEE Journal of Solid State Circuits*, pages 792–799, May 2001.

[31] M. Saint-Laurent, M. Swaminathoan, and J. Meindl. On the micro-architectural impact of clock distribution using multiple PLLs. In *Proceedings of IEEE International Conference of Computer Design*, pages 214–220, Sept. 2001.

[32] S. L. Sam, A. Chandrakasan, and D. Boning. Variation issues in on-chip optical clock distribution. In *Sixth International Workshop on statistical Mehotdologies for IC Processes, Devices, and Circuits*, June 2001.

[33] S. Schuster, W. Rehr, P. Cook, D. Heidel, and M.Immediato. Asynchronous interlocked pipelined CMOS circuits. In *IEEE Journal of Solid-State Circuits*, pages 292–293, Apr. 2000.

[34] C. Sechen. Output prediction logic: A high-performance CMOS design technique. In *Proceedings of the 2000 IEEE International Conference on Computer Design*, page 247. IEEE Computer Society, 2000.

[35] M. Shams, J. Ebergen, and M. Elmasry. A comparison of CMOS implementations of an asynchronous circuits primitive: the C-element. In *International Symposium on Low Power Electronics and Design*, pages 93–96, Aug. 1996.

[36] S. Sidiropoulos and M. Horowitz. A semi-digital dual delay locked loop. In *IEEE Journal of Solid-State Circuits*, volume 32, pages 1683–1692, Nov. 1997.

[37] J. Sparsø and S. Furber, editors. *Principles of Asynchronous Circuit Design: A Systems Perspective*. Kluwer Academic Publishers, 2001.

[38] I. Sutherland and S. Fairbanks. GasP: A minimal FIFO control. In *Proc. Intern. Symp. on Advanced Research in Async. Circuits and Systems*, pages 46–53. IEEE Computer Society Press, Mar. 2001.

[39] I. Sutherland, B. Sproull, and D. Harris. *Logical Effort: Designing Fast CMOS Circuits*. Morgan Kaufmann Publishers, Inc., 1999.

[40] I. E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, June 1989.

[41] D. Weinlader. Precision CMOS receivers for VLSI testing applications. Technical Report CSL-TR-01-3781, Stanford University, 2001.

[42] D. Weinlader, R. Ho, C. K. K. Yang, and M. Horowitz. An eight channel 36Gsample/s CMOS timing analyzer. In *International Solid State Circuits Conference*, Feb. 2000.

[43] T. Williams. Clock skew and other myths. In *Keynote for 9th IEEE Symposium of Asynchronous Circuits and Systems*, 2003.

[44] A. J. Winstanley, A. Garivier, and M. R. Greenstreet. An event spacing experiment. In *Proc. Intern. Symp. on Advanced Research in Async. Circuits and Systems*, pages 47–56, Apr. 2002.

[45] J. Wood, T. Edwards, and S. Lipa. Rotary traveling-wave oscillator arrays: a new clock technology. *IEEE Journal of Solid-State Circuits*, Nov. 2001.

[46] C.-K. K. Yang. Design of high-speed serial links in CMOS. Technical Report CSL-TR-98-775, Stanford University, 1998.

[47] C.-K. K. Yang and M. Horowitz. A 0.8um CMOS 2.5Gb/s oversampling receiver and transmitter for serial links. *IEEE Journal of Solid State Circuits*, 31(12), Dec. 1996.

[48] P. Zarkesh-Ha. Global interconnect modeling for a gigascale system-on-a-chip (GSoC). Technical report, Georgia Institute of Technology, Feb. 2001.

[49] J. Zhang and E. Friedman. Effects of shield insertion on reducing crosstalk noise between coupled interconnects. In *International Symposium on Circuits and systems*, volume 2, pages 529–532, May 2004.