High-Performance CMOS System Design Using Wave Pipelining

Kevin J. Nowka

Technical Report CSL-TR-96-693

January 1996

Partially supported by an ARPA Fellowship in High Performance Computing administered by the Institute for Advanced Computer Studies, University of Maryland. Additional support for this work from NSF Contract No. MIP88-22961 using facilities provided by NASA under contract NAG2-842.

High-Performance CMOS System Design Using Wave Pipelining

by

Kevin J. Nowka

Technical Report CSL-TR-96-693

January 1996

Computer Systems Laboratory
Departments of Electrical Engineering and Computer Science
Stanford University
Stanford, California 94305-4055

Abstract

Wave pipelining, or maximum rate pipelining, is a circuit design technique that allows digital synchronous systems to be clocked at rates higher than can be achieved with conventional pipelining techniques. It relies on the predictable finite signal propagation delay through combinational logic for virtual data storage. Wave pipelining of combinational circuits has been shown to achieve clock rates 2 to 7-times those possible for the same circuits with conventional pipelining.

Conventional pipelined systems allow data to propagate from a register through the combinational network to another register prior to initiating the subsequent data transfer. Thus, the maximum operating frequency is determined by the maximum propagation delay through the longest pipeline stage. Wave pipelined systems apply the subsequent data to the network as soon as it can be guaranteed that it will not interfere with the current data wave. The maximum operating frequency of a wave pipeline is therefore determined by the difference between the maximum propagation delay and the minimum propagation delay through the combinational logic.

By minimizing variations in delay, the performance of wave pipelining is maximized. Data wave interference in CMOS VLSI circuits is the result of the variation in the propagation delay due to path length differences, differences in the state of the network inputs and intermediate nodes, and difference in fabrication and environmental conditions.

To maximize the performance of wave pipelined circuits, the path length variations through the combinational logic must be minimized. A method of modifying the transistor geometries of individual static CMOS gates so as to tune their delays has been developed. This method is used by CAD tools that minimize the path length variation. These tools are used to equalize delays within a wave pipelined logic block and to synchronize separate wave pipelined units which share a common reference clock. This method has been demonstrated to limit the variation in delay of CMOS circuits to less than 20%.

Delay models have demonstrated that temperature variation, supply power variations, and noise limit the number of concurrent waves in CMOS wave pipelined systems to three or less.

Run-to-run process variation can have a significant impact on CMOS VLSI signal propagation delay. The ratio of maximum to minimum delay along the same path for seven different runs of a 0.8-micron feature size fabrication process was found to be 1.35. Unless this variation is controlled, the speedup of wave pipelining is limited to two to three to ensure that devices from any of these runs will operate. When aggregated with variations due to environmental factors, the maximum speed-up of a wave pipeline is less than two.

To counteract the effects of process variation, an adaptive supply voltage technique has been developed. An on-chip detector circuit determines when delays are faster than the nominal delays and the power supply is lowered accordingly. In this manner, ICs fabricated with fast processes are run at a lower supply voltage to ensure correct operation at the design target frequency.

To demonstrate that wave pipeline technology can be applied to VLSI system design, a CMOS wave pipelined vector unit has been developed. Extensive use of wave pipelining was employed to achieve high clock rates in the functional units. The VLSI processor consists of a wave pipelined vector register file, a wave pipelined adder, a wave pipelined multiplier, load and store units, an instruction buffer, a scoreboard, and control logic. The VLSI vector unit contains approximately 47000 transistors and occupies an area of 43 sq mm. It has been fabricated in a 0.8 micron CMOS technology. Tests indicate wave pipelined operation at a maximum rate of 303MHz.

An equivalent vector unit design using traditional latch-based pipelining was designed and simulated. The latch-based design occupied 2% more die area, operated with a 35% longer clock period, and had multiply latency 8% longer and add latency 11% longer than the wave pipelined vector unit.

This work was supported in part by an ARPA Fellowship in High Performance Computing administered by the Institute for Advanced Computer Studies, University of Maryland. Additional support for this work from NSF Contract No. MIP88-22961 using facilities provided by NASA under contract NAG2-842.

Key Words and Phrases: Wave-pipelining, pipelining, propagation delay, clocking

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

In an effort to improve the throughput of digital systems, designers have long turned to pipelining. In a pipelined system, a logic network is partitioned into pipeline stages, each of which operates upon data computed in the previous cycle by the previous pipeline stage. When a logic network is pipelined, synchronizing elements, either latches or registers, are inserted to partition the network into stages. Pipelining of a circuit into N stages can result in speedup in throughput up to a factor of N. The inserted synchronizing elements increase the area and power consumption of the logic and add additional latency and cycle time overhead.

Wave pipelining is an alternative synchronous circuit clocking technique that allows overlapped execution of multiple operations without using synchronizing elements within the logic. Rather, knowledge of the signal propagation delay characteristics of the logic network are used at design time to manage the signal delays so as to ensure that operations do not interfere with their predecessor nor successor computations.

Figure 1 is a block diagram of an nonpipelined circuit, a pipelined version of the same circuit, and a wave pipelined equivalent.

1.1.1 Pipelining and Wave Pipelining

While improving the throughput of a logic circuit, traditional pipelining of VLSI systems results in overheads in latency, cycle time, area, and power consumption. Cycle time overhead results from the time required for signals to propagate out of the synchronizing elements, from the time required for signals to set up to the synchronizing elements prior to their being stored in the synchronizing elements, and for the unintentional clock skew in the arrival of the synchronizer clock signal. Latency through the traditional pipeline is defined as the total elapsed time from the time of introduction of data at the input to the first stage of the pipeline to the time the results of computations performed on that data arrive at the output of the final stage of the pipeline. Latency overhead results from the use of pipelining due to the synchronizer overhead of each stage of the pipeline as described in the cycle time overhead. In addition, latency overhead results from pipeline partitioning overhead. In traditional pipelines with a common reference clock for all synchronizers, partitioning overhead occurs if the combinational logic cannot be divided into stages of equal maximum propagation delay. Area and power overhead results from the additional transistors and wires used to implement the synchronizing latches or registers, and from the increased clock buffer area and power needed to drive the clock inputs to the synchronizers.

Wave pipelining relies on the finite propagation delay of signals through a combinational digital circuit to store data. Rather than allowing data to propagate from a synchroniz-

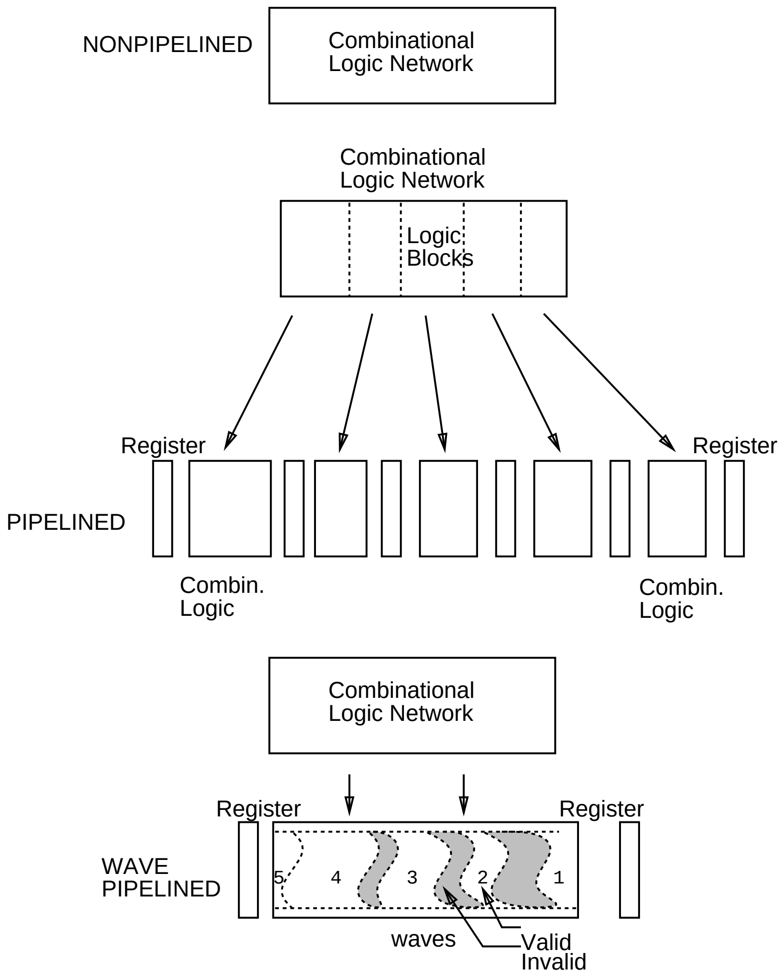


Figure 1: Circuit Model

ing element, latch or register, through the combinational network to another synchronizing element prior to initiating the subsequent data transfer, wave pipelined designs apply subsequent data to the network as soon as it can be guaranteed that it will not interfere with the current data wave. In this manner, multiple waves of data are simultaneously propagating through distinct regions of the logic network. Because waves of data are applied to the logic as fast as can be guaranteed not to interfere, the throughput of wave pipelined synchronous systems can be greater than can be achieved with conventional pipelining techniques. Wave pipelining can approach the physical switching limit of the devices [10].

Wave pipelining can improve the throughput of a logic circuit while avoiding some of the overheads of traditional pipelining. Wave pipelines avoid the cycle time overhead of traditional pipelines because there are no internal synchronizers. Instead, cycle time is determined by the variation in the propagation delay of the signals through the logic and the input and output register delays. Latency through the wave pipeline avoids the traditional pipeline overhead because the signals do not propagate through internal synchronizers. Partitioning overhead is avoided since the pipeline is not partitioned into stages separated by synchronizers. The area and power overheads of a traditional pipeline are avoided in the wave pipe since there are no internal synchronizers. Manipulation of the circuitry to maximize performance of wave pipelines can, however, introduce additional area and power overhead.

1.1.2 Prior Wave Pipeline Research

Significant research in wave pipelining has been conducted over the past thirty years in the areas of theory, tools, and VLSI designs.

Latchless pipelining techniques were first used in the development of the IBM System 360/91 floating point unit in 1967 [1]. This design was able to achieve a cycle time which was one-half the latency through the combinational logic without intermediate latches or registers.

Cotten [10] in 1969 formalized the theory of wave pipelining or maximum rate pipelining. As higher levels of integration and better CAD tools became available, a renewed interest in the theory of wave pipelining resulted. Ekroot [12] examined the optimizing of wave pipelines through the inserting of delays in the logic network. Wong [53] detailed the timing constraints for the operation of wave pipelined circuits and contrasted them to traditional clocking methods. Additional work in determination of the minimum clock period for wave pipelined operation has been performed by Joy and Ciesielski [27]. Lam, et. al. [33] suggested formal methods of analysis of clocking of wave pipelines using Timed Boolean Functions (TBFs).

Tools for wave pipelined circuit synthesis, optimization, and verification are pivotal in the success of wave pipelined design. Wong [53] developed methods of balancing delays in CML wave pipelined circuits to optimize cycle time. Additional work in standard cell placement and routing for the minimization of the clock period of wave pipelines has been performed by Joy, et. al. [26]. Chang, et. al. [6] developed a method of removing latches from a traditional pipeline when a lower clock period results and wave pipelined timing constraints can be met. Kim, et. al. [28] have developed optimization tools which restructure the wave pipeline logic to improve path length balance.

Wave pipelining has been successfully applied in several VLSI designs. Wong [54] developed a wave pipelined bipolar population counter with a latency of 10ns and a cycle time of 4ns, thus supporting 2.5 concurrent waves. Chappell, et. al. [8] applied wave pipelining techniques in the design of an SRAM that consisted of self-resetting logic blocks which were operated in wave pipelined fashion. This SRAM had a latency of 3.9ns and a cycle time of 2ns, thus supporting 1.9 concurrent waves. Fan, et. al. [14] developed a CMOS adder using wave pipelining. Simulated operation of this adder achieved 250MHz and supported greater than five waves. Lien, et. al. [35] applied wave pipelining to CMOS domino logic circuits and designed a 100MHz, 2-wave CMOS wave domino multiplier. Klass [29] developed a CMOS multiplier which operated at 350MHz and supported four waves. Additional wave pipelined VLSI designs include CMOS multipliers [41, 19, 55], CMOS static RAMs [40, 52], and several simple CMOS circuits.

In general VLSI implementations of wave pipelining have demonstrated up to 2 waves of data for memory devices and from 2 to 6 waves of data for arithmetic circuits

1.2 CMOS VLSI System Wave Pipelining

Previous research in CMOS wave pipelining has claimed and demonstrated from two to six concurrent waves of data. These studies have relied on simulations and empirical measurements to gauge the performance benefits of wave pipelining. One goal of this research has been to use CMOS device delay behavior to ascertain the limits of the performance of wave pipelining in CMOS systems. Analytical delay variation models reinforced with simulation data were used to quantify hard limits to wave pipeline performance and to suggest where optimization efforts are best suited.

Previous wave pipelining research devices have been single wave pipelines. Two basic approaches have been taken in the development of wave pipelines: Several VLSI circuits [8, 40, 54] have measured the wave pipeline behavior of the combinational logic only; the combinational logic was not embedded in a sequential system, and thus the constraints of the input and output synchronizers were ignored. In these designs the outputs of the wave pipelines were not latched.

A second class of demonstration wave pipelines surround the single block of combinational logic with input and output registers [31, 35, 14, 41]. These designs either provide the ability to set the phase relationship between the input and output register clocks, or modify the frequency of operation of the pipeline to insure valid wave pipelined operation.

When multiple, interconnected wave pipelines are integrated into a single system, additional complexity of design and operation result. First, all pipelines in the system should operate over a common range of clock frequency. Second, signals flowing between pipelines, including those in feedback paths, must meet the timing constraints for proper pipeline operation. This research effort has developed design methods for systems with multiple, interconnected wave pipelines.

Other CMOS wave pipelining has relied on manual optimization of wave pipelined performance or has relied on the addition of fixed circuit elements to assist in the performance optimization of the circuits. Automated CMOS optimization techniques for use in CMOS wave pipelined system design have been developed as part of this research.

Unlike previous wave pipelined research, where the operating frequencies could be determined and set individually for each die, in this research it was deemed necessary to design and operate all dice at a given target frequency. Techniques that ensure the correct operation of all dice at the target frequency were developed in this research effort.

Wave pipeline system design algorithmic, architectural, and circuit design issues such as wave pipeline stalling, low data-dependent CMOS circuit design, and wave pipeline / traditional pipeline interfacing were also examined in this research.

To validate the performance limits and the design techniques and tools a demonstration system was designed. A wave pipelined CMOS vector unit VLSI integrated circuit was designed, fabricated, and tested. This vector unit design operates at 300MHz. It contains a wave pipelined vector register file, a wave pipelined adder, and a wave pipelined multiplier. It demonstrates the use of multiple, synchronized wave pipelines with feedback. The performance of this system is optimized through the use of the automated balancing techniques and through process and environmental compensation techniques.

1.3 Related Research

The transistor sizing method of balancing for wave pipelining developed in this research is related to transistor sizing for performance optimization and area minimization for traditional pipelining. Fishburn, et. al. [16], in the development of TILOS, used a greedy algorithm to size the widths of transistors so as to minimize the critical path delay of CMOS circuits. Marple [37] used a nonlinear program solution to size transistor widths for minimum critical path delay. Berkelaar, et. al. [3] used a piecewise linear approximation to the delay functions used by Marple and thus solved a linear program for the widths. Sapatnekar [48] used convex programming to find a minimum area solution to the transistor widths which met critical path maximum constraints.

The compensation methods for changes in delay due to process and operating environment developed in this research are related to temperature and process compensation work for CMOS circuit testers [7], to multiple chip signal synchronization techniques [25], to power reduction for self-timed circuits [42], and to self-clocking techniques developed for use in low power circuits [23, 36].

Self-timed design techniques in which the completion detection logic is signaled with a timing reference which is guaranteed to be longer than the critical path logic [51] is somewhat akin to the wave pipelining with critical skew as presented in Chapter 5. Recent efforts in self-timed circuit design which make use of the dynamic signal propagation delay characteristics of critical logic paths to generate "dynamic clocks" [47, 11] are also so related.

Clock distribution techniques with constructive clock skew which are applied to traditional pipeline designs [15, 18] are specific cases of wave pipelining in which the intentional clock skew results in a number of waves fractionally greater than one.

1.4 Contributions

This dissertation develops constraints for wave pipeline operation which extend previously presented constraints [53, 33, 21] for environmental and process dependencies. Using these constraints, and models of static CMOS gate delays, performance limits for CMOS wave pipelines are established. A quantification of the performance implications of the delay variation for both wave pipelines and traditional pipelines is derived.

To optimize the performance of CMOS wave pipelined circuits, a method of equalizing CMOS circuit path delays is presented. The transistor sizing mechanism was developed, implemented, measured for balancing accuracy, and applied to the design of a vector unit as part of this research effort.

Optimization methods for systems of wave pipelines which are more general than examined by other researchers [12, 21, 27, 6] are developed. Methods of determining constructive clock skew and intentional delay insertion for optimization of these wave pipelined are proposed.

The strict constraints placed upon CMOS wave pipelines by fabrication and environmental variations quantified in this dissertation motivated an examination of delay compensation techniques. These techniques which have been employed for other compensation purposes, are evaluated for suitability to wave pipelined CMOS system design. Due to the range of compensation necessary and area and power benefits, a variable power supply technique is determined to be attractive. This technique was demonstrated in the vector unit IC developed in this research.

In this research, one impediment to the use of wave pipelines in processors, the inability of a wave pipeline to stall is examined. Stalling wave pipelines which employ additional transistors within the pipeline to provide stall capabilities are proposed, their clock constraints are presented, and their performance is contrasted to conventional pipelines.

While previous efforts have demonstrated operation of wave pipelined memories and arithmetic circuits, this research demonstrates that systems of CMOS wave pipelines, using the tools and techniques developed, can be designed, optimized, fabricated, and operated at clock rates above those achievable using conventional techniques.

1.5 Scope

The following chapters detail the performance limits of CMOS wave pipelining, wave pipelining design and optimization techniques, VLSI vector unit design and testing results, and architectural and circuit optimizations for wave pipelining.

Chapter 2 is an analysis of the performance limits of wave pipelining in CMOS systems. It presents the timing constraints for valid wave pipelined operation, presents an analytical model of the delay characteristics of CMOS circuits, details the causes and performance effects of delay variation in CMOS circuits, and contrasts the variation effects upon performance to those exhibited by traditional pipeline designs.

Chapter 3 presents design techniques for high performance wave pipelines. It details the path delay balancing method employed in this research and describes the procedure used to synchronize and optimize the performance of multiple wave pipeline systems. It relates techniques for process and environmental compensation to ensure correct operation of wave pipelined systems over all design operation ranges.

Chapter 4 describes the organization, design procedure, test procedure, and test results of a CMOS wave pipelined vector unit integrated circuit.

Chapter 5 is an exposition of architectural and circuit design enhancements for CMOS wave pipeline design. It describes a methods of supporting pipeline stalls in wave pipelines, latchless feedback, and low data dependent circuit techniques.

Chapter 6 summarizes the results of this research and suggests further areas of wave pipelining research.

2 Performance Limits of CMOS Wave Pipelining

This chapter is an analysis of the limits of the performance that can be achieved through the use of wave pipelining in CMOS circuits. The clock timing constraints which must be met for correct circuit operation in concert with modeled delay behavior for CMOS logic are used to derive performance limits for static CMOS wave pipelining.

2.1 Wave Pipeline Clock Constraints

The clock period of traditional pipelined synchronous circuits must meet race-through and long-path timing constraints.

Long-path constraint requires that the results from the current cycle's inputs are valid at the next synchronizing element prior to being latched. Thus, the propagation from the synchronizing element, through the data network, to the next synchronizing element is less than the time from the initiating edge of the current clock cycle to the latching edge of the next clock cycle. Figure 2 defines the initiating and latching edges for both flow latch and edge-triggered register synchronizing elements.

The race-through constraint requires that in the same clock cycle data cannot propagate out of a synchronizing element, through the combinational network, and into the next synchronizing element prior to the occurrence of the storage transition. Thus the minimum propagation time though the synchronizing element, through the network, to the next synchronizing element is less that the time from the output initiating edge to the latching edge of the same cycle. Thus the data resulting from the current input data cannot interfere with the previous results in the next synchronizing element.

Similar constraints for wave pipelines exist. The primary differences in the wave pipeline constraints result from the fact that the data initiating edge and data storage edge may be separated by several clock cycles. The long path constraint for wave pipelines requires that the propagation out of the synchronizer, through the combinational logic, and into the output synchronizer is less than the time from the initiating edge to the latching edge which occurs N cycles later. This constraint is [53, 33, 21]:

$$N * T_{clk} + cs \geq P_{max} + \Delta C + T_s + \frac{RF_{max}}{2} + T_{synch} \tag{1}$$

In Figure 3 this constraint is shown for the wave 1 data. N is the number of clock periods between the application of the input data and the subsequent latching of the results at the output. It is also the number of concurrent waves in the wave pipeline. cs is the constructive skew between the clock at the input synchronizer and output synchronizer. P_{max} is the worst case maximum propagation delay through the combinational network. It is measured from the time at which the slowest input reaches the midpoint of its switching

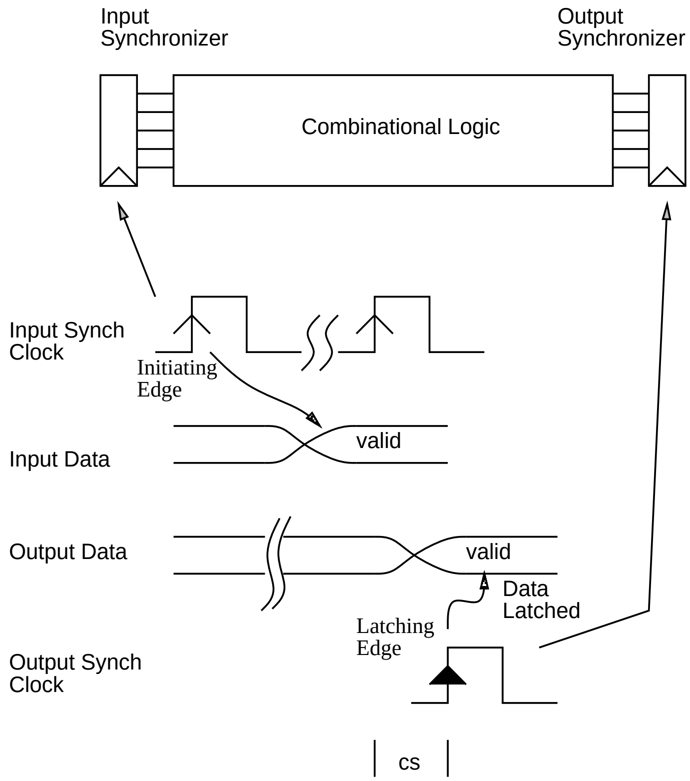


Figure 2: Synchronizer Edge Definitions

voltage to the time at which the slowest output of the logic reaches the midpoint of its switching voltage. ΔC is the unintentional clock skew between input and output clocks. T_s is the maximum setup time of the output synchronizer. RF_{max} is the maximum rise/fall time of the inputs to the output synchronizer. T_{synch} is the maximum time from the data initiating edge of the clock to valid output of the input synchronizer. This inequality ensures that the result of the slowest computation has sufficient time to propagate to the output, all outputs rise or fall to its terminal value, and all outputs meet the minimum setup time of the synchronizer prior to being latched.

In addition, the subsequent wave must not reach the synchronizer prior to the synchronizing clock edge. Thus the race through constraint for wave pipelines using edge-triggered registers as synchronizing elements is:

$$(N-1) * T_{clk} + cs \leq P_{min} - \Delta C - T_h - \frac{RF_{min}}{2} + T_{synch} \tag{2}$$

In Figure 3 this constraint is shown for the wave 2 data. This inequality ensures that the result of the fastest computation is not able to propagate through the logic fast enough

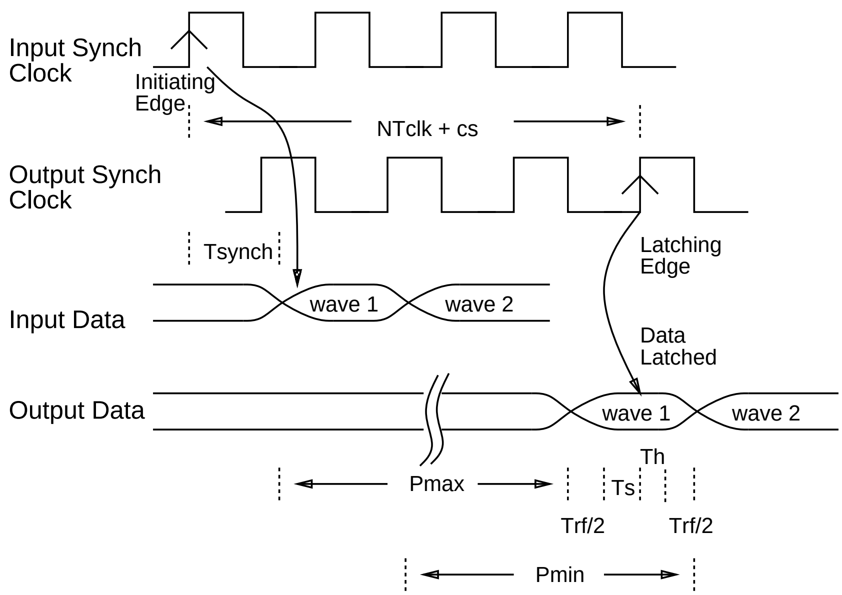


Figure 3: Wave Pipeline Timing Definitions

to change the voltage of any output in the cycle before the results will be latched. This figure is for the edge-triggered registers as synchronizing elements. For transparent latches as synchronizing elements:

$$(N-1) * T_{clk} + T_{trans} + cs \leq P_{min} - \Delta C - \frac{RF_{min}}{2} + T_{synch} \tag{3}$$

where T_{trans} is the time over which the latch is open and transparent.

In addition to meeting the race-through and long-path constraints, wave pipelined circuits require that waves of data do not interfere with each other at the output synchronizing element. This constraint result in the following inequality:

$$T_{clk} \geq P_{max} - P_{min} + 2\Delta C + T_s + T_h + \frac{RF_{min} + RF_{max}}{2} \tag{4}$$

In addition to the output constraint, wave pipelined circuits can not allow wave interference at any point in the network. This can be represented by the following:

$$T_{clk} \geq P_{max} - P_{min} + \Delta C + T_{ms} + \frac{RF_{min} + RF_{max}}{2} \tag{5}$$

where T_{ms} is the minimum amount of time a node voltage must be stable to ensure the subsequent level of logic operates correctly.

Details of the timing constraints for pipelined and wave pipelined circuits are found in [53].

To establish the performance limits of wave pipelining, constraints 1 and 2 can be combined to find the maximum number of waves which can be supported by a wave pipeline:

$$N_{max} = \frac{P_{max} - cs + \Delta C + T_s + \frac{RF_{max}}{2} + T_{synch}}{P_{max} - P_{min} + 2\Delta C + T_s + T_h + \frac{RF_{min}+RF_{max}}{2}} \tag{6}$$

N_{max}, the maximum number of waves in the wave pipeline, also represents the maximum speed up of a wave pipeline over the same circuit being operated as a traditional pipeline stage. By collecting the clock overhead factors for the long path and the race-through into a single terms H_{max}, and H_{min}, respectively, constraint 6 can be reduced to:

$$N_{max} = \frac{P_{max} - cs + H_{max}}{P_{max} - P_{min} + H_{max} + H_{min}} \tag{7}$$

The long path condition clock overhead is:

$$H_{max} = \Delta C + T_s + RF_{max}/2 + T_{synch} \tag{8}$$

The race-through condition clock overhead is:

$$H_{min} = \Delta C + T_h + RF_{min}/2 - T_{synch} \tag{9}$$

2.2 CMOS Propagation Delay

To ascertain the performance limits of CMOS wave pipelining, the propagation delays in CMOS logic and the causes of the variations in the propagation delays must be quantified. The full delay models used in this analysis are derived in Appendix B and are simply summarized in this section.

Combinational logic network delay is modeled as the sum of individual gate delays. In modeling the propagation delay of a gate, T_{pd}, the gate is treated as a single transistor, sized so as to match the current carrying capacity of the complex gate, charging or discharging a fixed load capacitance. This delay for long-channel transistors is found to be:

$$T_{pd} = \frac{2C_l V_t}{K(V_{dd} - V_t)^2} + \frac{C_l}{(V_{dd} - V_t)K[ln(\frac{3V_{dd}-4V_t}{V_{dd}})]} \tag{10}$$

where C_l is the total load capacitance, V_t is the transistor threshold, K is the transistor transconductance, and V_{dd} is the power supply voltage. Short-channel results are presented in Appendix B.

With this model of the gate delay, the maximum delay through a combinational logic network is:

$$P_{max} = \sum_{long\ path} T_{pd} \tag{11}$$

and the minimum path delay through the combinational logic network is:

$$P_{min} = \sum_{short\ path} T_{pd} \tag{12}$$

As shown in equation 6, the speedup of wave pipelines is constrained by relative differences in propagation delay rather than maximum propagation delay:

$$N_{max} \approx \frac{P_{max}/P_{min}}{P_{max}/P_{min} - 1} \tag{13}$$

Thus, the ratio of maximum to minimum propagation delays is necessary to ascertain the performance potential of wave pipelining.

2.3 Causes of Variation in CMOS VLSI

The clock rate of wave pipelined circuits is constrained by the worst-case variation of propagation delay through the network. The sources of variation in the network are:

1. Variations due to differences in propagation of signals along different paths.
2. Variations due to differences in the state of network node voltages and gate side inputs (data dependencies.)
3. Variations due to changes in operating temperature.
4. Variations due to supply voltage drift and noise.
5. Variations due to fabrication process variations.
6. Variations due to signal noise.

2.3.1 Path Length Imbalance

Each path through a circuit may have a different propagation delay. For highly regular logic structures like memories, the variation in delay between these paths may be relatively small. In more random logic structures the propagation delay along the longest path through the logic network may be many times the delay along the shortest path. Without optimization, wave pipelines of random logic are thus only capable of achieving fractional improvements in the throughput of the pipeline.

Following the theory developed by Ekroot [12], techniques have been developed to balance the path delays of bipolar circuits [53] and CMOS circuits [32, 45] through insertion of delay elements and through the manipulation of the delay characteristics of individual gates. Kim. et. al. [28] has concentrated on synthesis of circuits with more path length balance than can be achieved with traditional area and delay minimization synthesis techniques.

An automated procedure for the balancing of CMOS wave pipelined circuits is presented in Chapter 3. This balancing procedure has been demonstrated to limit delay variation to less than 20% of the maximum propagation delay for static CMOS circuits.

2.3.2 Data Dependencies

Data dependent delay variation results from two effects. First, signal propagation does not generally occur from a single input to a given output along one path. Instead, transitions occur along multiple, interacting paths from any number of inputs to the given output. Thus the delay along a given input to output path depends upon the occurrence of transitions on side inputs and their time relation to the transitions occurring along the given path. Secondly, the rate of signal propagation of individual gates may depend upon the state of internal node voltages. For instance the propagation delay through a two-input static CMOS NAND gate when both inputs are rising varies depending upon the voltage at the common node in the NMOS transistor stack. This form of data dependency results from previous input transitions.

Klass [32] has found that by implementing functions in relatively input pattern insensitive logic such as NAND2/INV static CMOS, delay variation can be limited to less than 10% for a 4-bit carry lookahead adder. The balancing procedure for static CMOS circuits presented in Chapter 3, has been found to limit the delay variation due to path imbalance and data dependencies to less than 20%.

Because the variation in propagation delay due to differences in path length and data dependencies are determined primarily by the implementation of the logic function, the performance potential of wave pipelining is presented as a function of the degree of imbalance in the network implementation. In Chapter 3, a method for minimizing this imbalance is presented and its application to several representative circuits is presented.

2.3.3 Fabrication Process

In addition to the effects of path length variation and data dependencies which depend primarily upon the implementation of the logic function, variation in the manufacture of the VLSI integrated circuit and its operating environment influence the delay of CMOS wave pipelined circuits. Fabrication process variation strongly influences the propagation delay of a circuit.

Process parameters are characterized as nominal and corner. Nominal process is the expected process. Corner processes are the limits of acceptable process parameters.

Table 1 shows the simulated propagation delay of a chain of 50 inverters for the fabrication corners of a 2 micron MOSIS process [49]. Over these limits, fabrication process variation affects propagation delay by +16% to -19%. Thus the ratio of maximum delay to minimum delay due to process is 1.43.

Process	Propagation Delay (ns)
fast	14.6
slow	21.0
typical	18.1

Table 1: Simulated Process Corner Propagation Delays

Figure 4 is a diagram of simulated propagation delay of a chain of 50 inverters for using SPICE model parameters derived from measurements of seven MOSIS 0.8 micron fabrication runs. For these runs, the maximum propagation delay is longer than the minimum by a factor of 1.35. When compared to the arithmetic average, the variation is +11% to -18%.

Fan, et. al. [14] performed fabrication process sensitivity analysis on a wave pipelined adder design. By varying the SPICE model parameters, they found simulated delay to be most sensitive to variations in channel oxide thickness, the transistor geometry parameters, and device transconductance.

2.3.4 Environmental Variation

In addition to the implementation dependent variation and the manufacturing process variations, the environmental operating conditions have significant impact on the delay of CMOS logic circuits.

Temperature The variation in propagation delay due to temperature is primarily the result of the variation of the channel current of the conducting MOS device. The variation

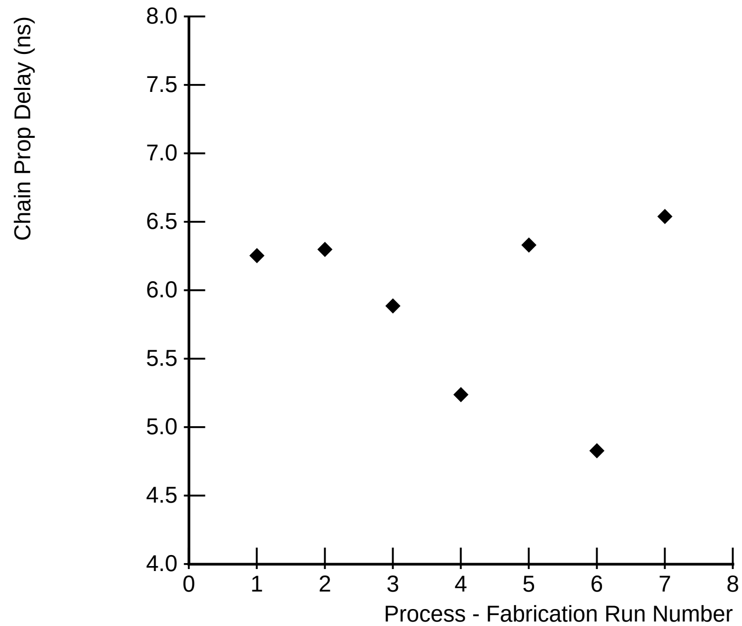


Figure 4: Inverter Chain Propagation Delay vs. Fabrication Run

of channel current with temperature is strongly related to the change in channel carrier mobility. Therefore, the variations in propagation delay are modeled as a function of variations in mobility. Secondary effects such as threshold reduction and junction capacitance variation are ignored for this analysis.

Empirical studies [46, 20] have shown that the temperature dependence of channel carriers can be represented by:

$$\mu(\tau) = \mu_0(\tau) f_v f_h \tag{14}$$

where f_v and f_h represent degradation factors in the vertical and horizontal directions, respectively.

The temperature dependence of the low-field mobility, μ_0, is;

$$\mu_0(\tau_2) = \mu_0(\tau_1) * (\tau_2/\tau_1)^{-M} \tag{15}$$

where M is an empirical constant between 1.5 and 2. HSPICE uses $M = 1.5$ for level 3 IDS MOS device modeling [38]. τ_1 and τ_2 are absolute temperatures.

Figure 5 shows the ratio of channel carrier low-field mobility at 25 C to that for temperatures from 25 C to 125 C as derived from the above mobility formula with M=1.5.

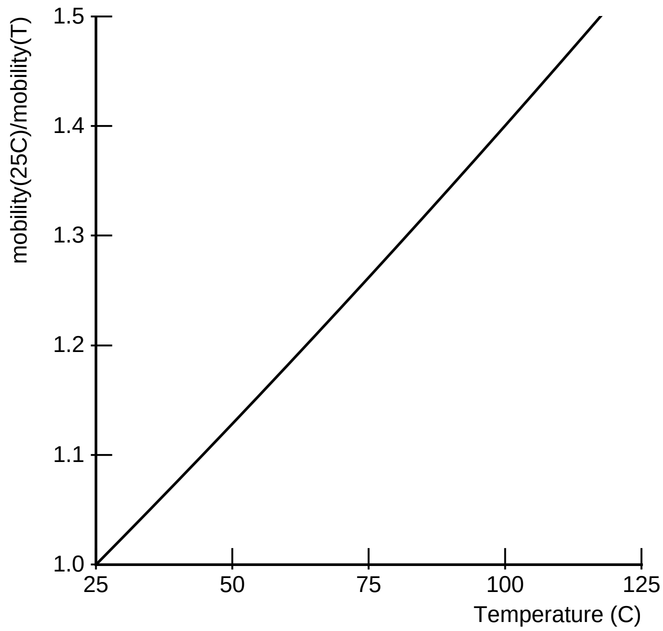


Figure 5: Relative Carrier Mobilities vs. Temperature

The variation of mobility results in a corresponding variation in channel current, and in turn, propagation delay. Ignoring the secondary temperature effects, and concentrating on the mobility variation, the propagation delay through a network of long-channel devices at a given temperature to that at the nominal temperature should be the inverse of the mobility ratio as suggested by the propagation delay equations in Section B.

Figure 5 data suggests that propagation delays of CMOS logic structures can be as much as 50 to 60% slower at 125C than at 25C due to the differences in mobility.

Figure 6 shows HSPICE simulations of propagation delay of two chains of 50 CMOS inverters over a temperature range of 25 C to 125 C. The short-channel chain consists of inverters with $1.5\mu/0.8\mu$ NMOS transistors and $3.5\mu/0.8\mu$ PMOS transistors. The long-channel chain consists of inverters with $9\mu/3\mu$ NMOS transistors and $21\mu/3\mu$ PMOS transistors.

Figure 7 shows the ratio of propagation delay of the inverter chains for temperatures from 25 C to 125 C to the propagation delay at 25 C. Superimposed on Figure 7 is the ratio of mobilities as given previously. The mobility approximation to relative propagation delay becomes less accurate as temperature is increased due to the assumption of constant thresholds.

Based upon the models of CMOS device behavior and SPICE simulations, the propagation

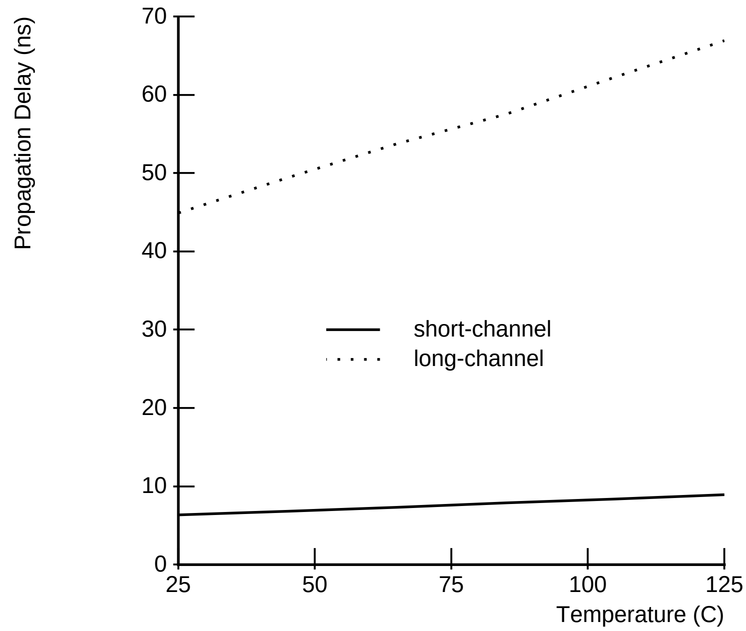


Figure 6: Inverter Chain Propagation Delay vs. Temperature

delay of a CMOS network at temperature τ_2 can be approximated by:

$$P_{max}(\tau_2) \approx P_{max}(\tau_1) * (\frac{\tau_2}{\tau_1})^{1.5} \tag{16}$$

$$P_{min}(\tau_2) \approx P_{min}(\tau_1) * (\frac{\tau_2}{\tau_1})^{1.5} \tag{17}$$

For short-channel devices, velocity saturation limits the channel current. Because temperature affects the saturation voltage, the expression for relative propagation delay is more complicated:

$$P_{max}(\tau_2) \approx P_{max}(\tau_1) * (\frac{\tau_2}{\tau_1})^{1.5} * (\frac{V_{dmax}(\tau_1)}{V_{dmax}(\tau_2)})^2 \tag{18}$$

$$P_{min}(\tau_2) \approx P_{min}(\tau_1) * (\frac{\tau_2}{\tau_1})^{1.5} * (\frac{V_{dmax}(\tau_1)}{V_{dmax}(\tau_2)})^2 \tag{19}$$

Thus, propagation along a given path for a CMOS network will be as much as 50% slower at 125 C than at room temperature.

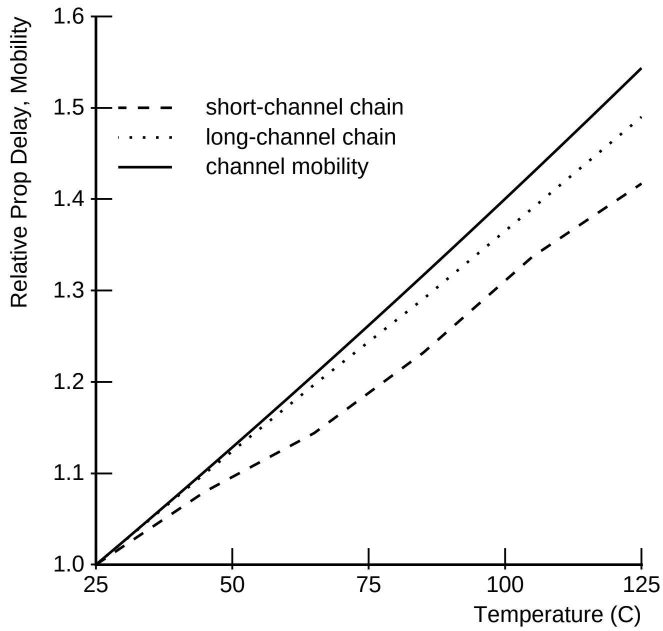


Figure 7: Relative Propagation Delay vs. Temperature

Temperature variation is both spatial and temporal. As a transistor conducts current, heat is conducted through the surrounding die area resulting in changes in local temperature. As power consumption increases, the spatial average of die temperature also increases, and thus a temporal temperature variation exists. If the consumption of power is not spatially uniform, temperature gradients exist.

Figure 8 illustrates the spatial and temporal variation in temperature for the vector unit IC developed as part of this research. The two thermal profiles are shown for the locations on the die which experience the thermal extremes. These profiles were derived using two-dimensional heat transfer models in which the die was represented as a mesh of thermal cells. Each cell is represented by a thermal resistance with each of its neighbors, a thermal resistance to the ambient, the specific heat of the cell, and the local power consumption.

At time $t = 0$ the vector unit operations which consume the greatest amount of power commence. The total power consumption at this instance increases from 0.6 W to 1.9 W. The top trace gives the temperature of the maximum temperature location. The temperature at this location rises approximately 43 C with a time constant of approximately 350 ms. The location of the minimum thermal extreme rises approximately 38 C. Thus, as shown in this figure, the maximum spatial temperature difference is 5 C and the maximum temporal temperature variation is 43 C.

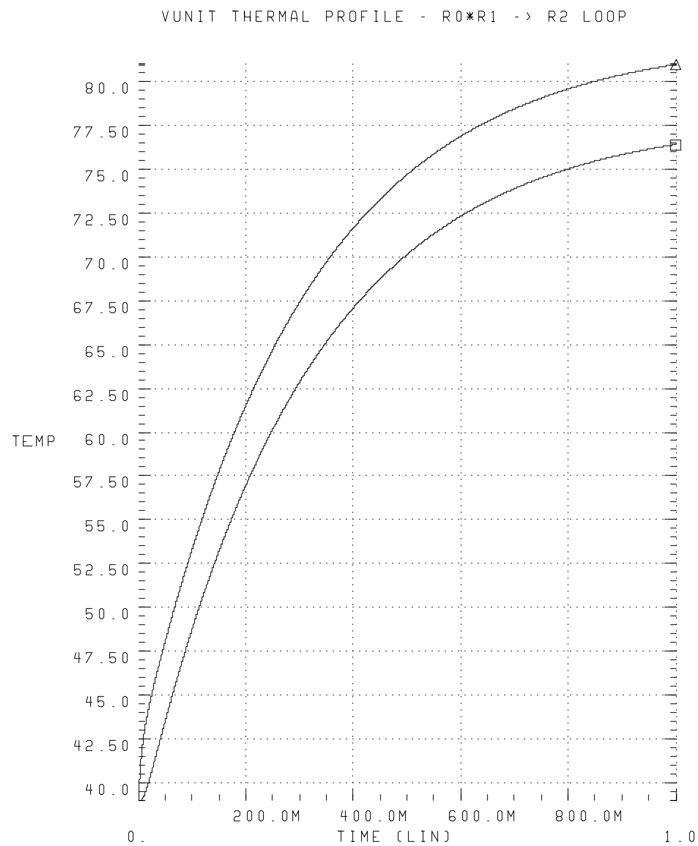


Figure 8: Vector Unit Thermal Profile

Voltage Variation Supply voltage variation affects propagation delay by altering the channel current and signal voltage swing. Using the delay expressions from Section B for the propagation delay of a capacitor discharging through an n-channel device and charging through a p-channel device, a first-order expression for the ratio of propagation delay at a given supply voltage to the propagation delay at the nominal supply voltage is derived.

For the simulated process, the model parameters are given in table 2.

Figure 9 shows the propagation delay of a capacitor being driven high and low through a PMOS and NMOS device, respectively, for a range of supply voltages, relative to the nominal 5V supply.

Since, in this model, the propagation delay through a logic network is the sum of the individual delays, the ratio of the propagation delays for the network should lie within the charging and discharging ratios.

Parameter	Value
V_{tn}	0.71V
V_{tp}	-0.90V
V_{dd}	5V
μ_{n0}	572 cm^2/Vs
μ_{p0}	178 cm^2/Vs
C_{ox}	192 nF/cm^2
v_{satn}	1980 cm/s
v_{satp}	3690 cm/s

Table 2: Simulated Process Parameters

Figure 10 shows the simulated propagation delay of a minimum-sized balanced inverter driving an identical inverter high and low versus supply voltage. This figure shows that static CMOS inverter gate delay is, to first order, inversely proportional to the supply voltage.

Figure 11 compares the computed relative propagation delay ratios for rising and falling outputs versus supply voltage. Also included in this figure is the simulated ratios for the short-channel chain of 50 inverters. Figure 12 is a plot of simulated propagation delay versus supply voltage for a nominal supply of 3.3V.

As a first-order approximation the variation in propagation delay due to supply voltage drift is linearly related to the supply voltage. Thus:

$$P_{max}(V_2) \approx P_{max}(V_1) * \frac{V_1}{V_2} \tag{20}$$

$$P_{min}(V_2) \approx P_{min}(V_1) * \frac{V_1}{V_2} \tag{21}$$

The propagation delay of a network shows a variation of 5% to 10% with respect to nominal over an operating supply voltage range of -4.5 to 5.5V. The ratio of maximum delay to minimum delay due to supply changes is thus 1.2.

In addition to dc variations, dynamic power fluctuations have an effect on the propagation delay of CMOS circuits. Supply dI/dt noise due to on-chip circuitry is small; however, driver dI/dt noise can have a significant impact on the delay of the driver [2].

With separate power distribution networks, the delay variation is isolated to the driver. Thus, the relative delay variation of a CMOS circuit path due to driver dI/dt noise can be estimated by multiplying the relative delay factor of the driver by the fraction of the nominal delay of the path due to the nominal driver delay.

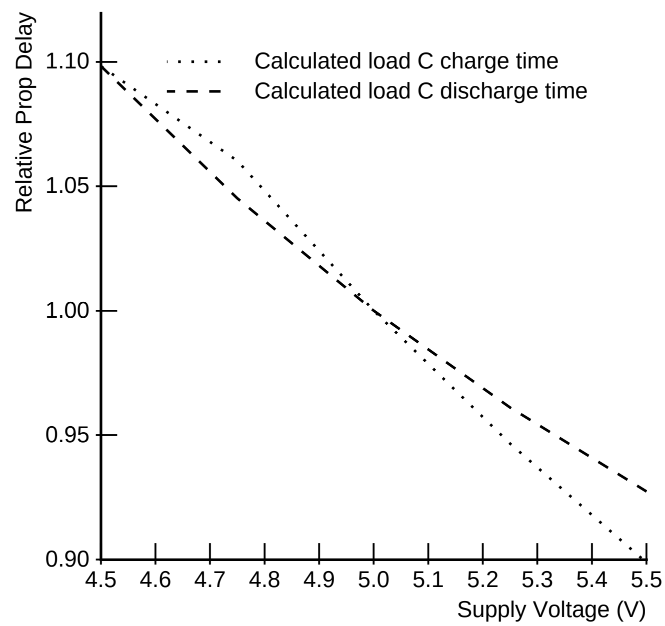


Figure 9: Relative Charge, Discharge Delay vs. Supply Voltage

Coupled Noise In addition to the power supply noise described in the previous section, noise coupled from adjacent signals to the output of a gate can have significant impact on the delay of that gate.

We model capacitively coupled noise as a change in the effective load capacitance as seen by the gate. When there is no change in the voltage on the coupled line, the effective capacitance as seen by the gate is the nominal capacitive load due to the output wire capacitance and the gate capacitance of all transistors connected to the output. When the voltage on a coupled line is moving in the same direction as the gate output, the effective capacitive load is decreased by value of the coupled capacitance. When the voltage on the coupled line is moving in the opposite direction as the gate output, the effective capacitance is increased by the value of the coupled capacitance.

The resulting delay for the gate with a capacitively coupled output is:

$$P(\nu) \approx P(\nu = 0) * [C_l - \sum d_i * C_{coupled\ i})] \tag{22}$$

$P(\nu)$ indicates the delay under coupled noise conditions. $P(\nu = 0)$ indicates the delay with all coupled wires static. d_i indicates the direction of signal switch (-1 if opposite to output and 1 if same as output). $C_{coupled\ i}$ is the mutual capacitance of the output and the signal i.

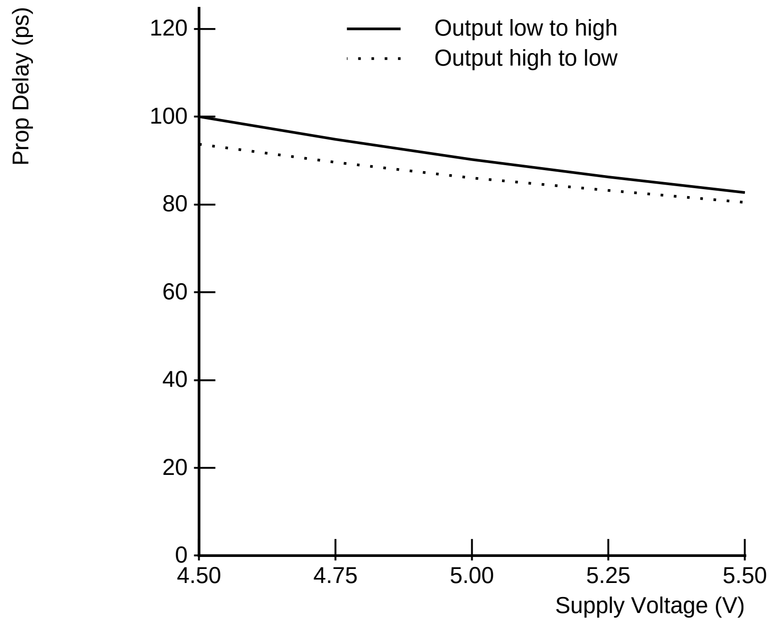


Figure 10: Inverter Propagation Delay vs. Supply Voltage

Thus the maximum to minimum delay ratio of the gate output due to coupled capacitance is:

$$\frac{P(\nu_{slow})}{P(\nu_{fast})} \approx \frac{(C_l + \sum C_{coupled\ i})}{(C_l - \sum C_{coupled\ i})} \tag{23}$$

This variation is most important for gates driving long wires. To get the effect of coupled noise on the total propagation delay the gate delay variation must be scaled by the ratio of the nominal contribution to the path delay due to the wire driver to the nominal total delay.

Process and Environmental Performance Limits This section uses the clock constraints, the delay models, and the variations in process and operating environment previously discussed to establish the limits of wave pipeline performance for fixed frequency and variable frequency CMOS systems.

Fixed Frequency Clocked Wave Pipelined Systems In an fixed frequency clocked synchronous system, a clock with fixed period T_{clk} is supplied to the device. The clock frequency is not a function of chip supply voltage, temperature, or fabrication process. Systems with external clock generation, clocks phase-locked to external fixed frequency clocks,

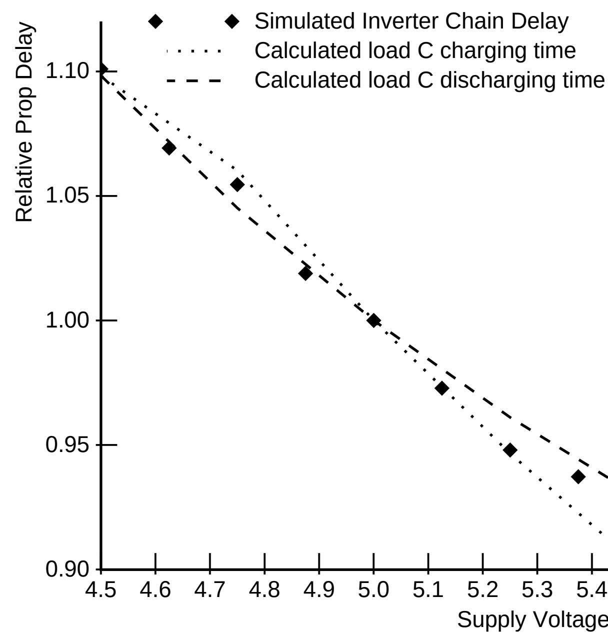


Figure 11: Relative Propagation Delay vs. Supply Voltage (5V)

and systems with temperature and supply voltage compensating on-chip fixed frequency oscillators are included in this category. Figure 13 is a block diagram of a synchronous system with an externally supplied clock.

For a fixed frequency clocked *traditional* pipelined system to operate properly, the worst case maximum propagation delay determines the clock rate:

$$P_{max} + RF_{max}/2 + T_s + \Delta C \leq T_{clk} + cs \tag{24}$$

$P_{max}, RF_{max}, T_s, \Delta C$ and cs are voltage, temperature, and process dependent. T_{clk} is voltage, temperature, and process independent.

For a fixed frequency clocked *wave pipelined* circuit to operate properly, the following two inequalities must hold for edge-triggered registers:

$$\frac{P_{max} + RF_{max}/2 + T_s + \Delta C + T_{synch} - cs}{N} \leq T_{clk} \tag{25}$$

$$\frac{P_{min} - RF_{min}/2 - T_h - \Delta C + T_{synch} - cs}{N - 1} \geq T_{clk} \tag{26}$$

For flow latches, the following inequalities must hold:

$$\frac{P_{max} + RF_{max}/2 + T_s + \Delta C + T_{synch} - cs}{N} \leq T_{clk} \tag{27}$$

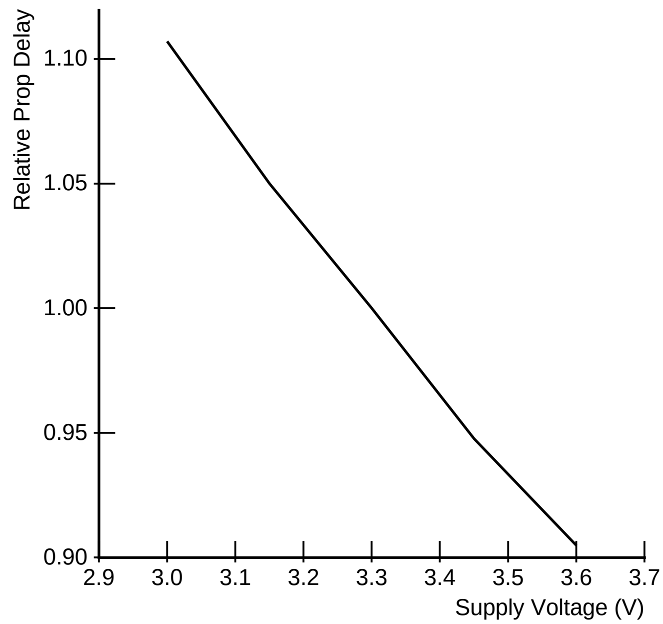


Figure 12: Relative Propagation Delay vs. Supply Voltage (3.3V)

$$\frac{P_{min} - RF_{min}/2 - T_h - \Delta C + T_{synch} - T_{trans} - cs}{N-1} \geq T_{clk} \tag{28}$$

P_{max}, P_{min}, RF, T_{synch}, T_s and T_h are voltage, temperature, and process dependent. T_{clk} and T_{trans} are voltage, temperature, and process independent.

Deviation of process parameter on a die are relatively time invariant and relatively uniform across the entire die [20]. Thus, once a device is fabricated, its T_{ox}, μ_0, V_{tn}, V_{tp}, etc. could be determined and the process characterized. This type of variation is termed static delay variation. Since the particular process parameters are not known *a priori*, wave pipelines must be designed to function over a range of expected processes.

In addition to the static variation, there is dynamic variation. Dynamic variation is due to changes in the operating environment over time, and include the temperature, voltage drift and noise, and coupled noise examined in this chapter.

For a wave pipeline to function correctly, a clock period and an integer number of waves must be specified which satisfy the above inequalities for all acceptable values of process, supply voltage, and temperature. For the first inequality, the worst condition is minimum supply voltage, maximum temperature, and slowest process. For the second inequality, the worst condition is maximum supply voltage, minimum temperature, and fastest process.

The longest path in the network is some factor greater than the shortest path in the network

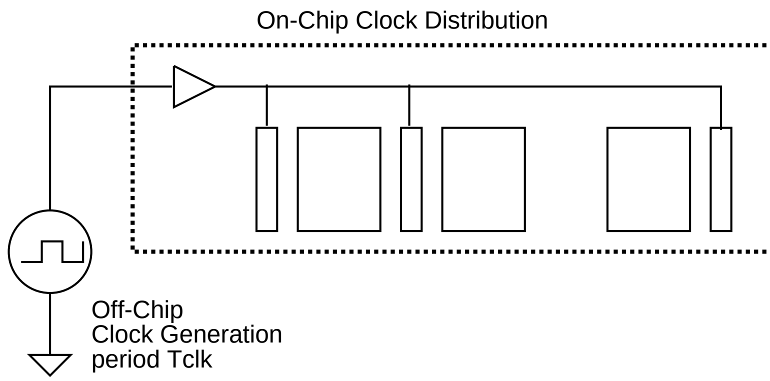


Figure 13: Externally Supplied Clocked System

for a given temperature, voltage, and process. This factor, represented by α, is due to path length differences and data dependent delay variations in the network.

$$P_{max}(V,\tau,\phi) = \alpha * P_{min}(V,\tau,\phi), \alpha \geq 1 \tag{29}$$

Because the relative variation in propagation delay due to temperature and voltage variation is to first order independent of absolute propagation delay, α is a good approximation of the relative path length difference in the network for any temperature and voltage.

The worst-case wave pipeline timing constraints become:

$$\frac{\alpha P_{min}^{slow} + RF_{max}^{slow}/2 + T_s^{slow} + \Delta C^{slow} + T_{synch}^{slow} - cs^{slow}}{N} \leq T_{clk} \tag{30}$$

$$\frac{P_{min}^{fast} - RF_{min}^{fast}/2 - T_h^{fast} - \Delta C^{fast} + T_{synch}^{fast} - cs^{fast}}{N-1} \geq T_{clk} \tag{31}$$

where *slow* signifies operating conditions $(V_{min}, \tau_{max}, \phi_{slow})$ and *fast* signifies operating conditions $(V_{max}, \tau_{min}, \phi_{fast})$.

The propagation delay at worst case operating temperature, supply voltage, and process will be some factor larger than the best case propagation delay. If β is defined as:

$$\beta = \frac{P_{min}(V_{min}, \tau_{max}, \phi_{slow})}{P_{min}(V_{max}, \tau_{min}, \phi_{fast})} \tag{32}$$

From Section 2.3.4 data:

$$\beta \approx (\frac{\tau_2}{\tau_1})^{1.5} * \frac{V_{max}}{V_{min}} * \beta_{proc} \tag{33}$$

where β_{proc} is the variation in delay due to process. If it is assumed that setup, hold, rise and fall, synchronizer delay, and skew times scale as propagation delay with temperature and voltage, the worst case timing inequalities become:

$$\frac{\alpha\beta P_{min}^{fast} + \beta RF_{max}^{fast}/2 + \beta T_s^{fast} + \beta\Delta C^{fast} - \beta cs^{fast} + \beta T_{synch}^{fast}}{N} \leq T_{clk} \quad (34)$$

$$\frac{P_{min}^{fast} - RF_{min}^{fast}/2 - T_h^{fast} - \Delta C^{fast} - cs^{fast} + T_{synch}^{fast}}{N-1} \geq T_{clk} \quad (35)$$

Combining the constraints to solve for N, the number of waves in the wave pipelined circuit:

$$N \leq \frac{\alpha\beta P_{min}^{fast} + H_{max}^{slow} - \beta cs^{fast}}{(\alpha\beta - 1)P_{min}^{fast} + H_{max}^{slow} + H_{min}^{fast} - (\beta - 1)cs^{fast}} \quad (36)$$

where,

$$H_{max}^{slow} = \beta RF_{max}^{fast}/2 + \beta T_s^{fast} + \beta\Delta C^{fast} + \beta T_{synch}^{fast} \quad (37)$$

$$H_{min}^{fast} = RF_{min}^{fast}/2 + T_h^{fast} + \Delta C^{fast} - T_{synch}^{fast} \quad (38)$$

If $P_{min}^{fast} >> RF, T_s, T_h, \Delta C, T_{synch}$ and the clocks are not intentionally skewed, $cs = 0$, then:

$$N \leq \frac{\alpha\beta}{\alpha\beta - 1} \quad (39)$$

In a perfectly balanced network $\alpha = 1$, thus:

$$N \leq \frac{\beta}{\beta - 1} \quad (40)$$

Figure 14 gives the maximum number of waves through a wave pipelined network versus the process and environmental delay variation factor, β, for several practical values of the path length variation factor, α.

Table 3 gives the simulated results for the maximum number of waves achievable for the chain of 50 inverters for a range of temperatures and voltages. It is assumed that the process parameters are nominal.

There are two important implications from constraint 39. First, based upon data from Section 2.3.4 and Section 2.3.4 values of β for temperature ranges of 25-125 C and voltage ranges of 4.5-5.5 V for CMOS circuits will be 1.4 to 1.7. Therefore, the number of waves in a static CMOS wave pipelined logic network, independent of its absolute propagation

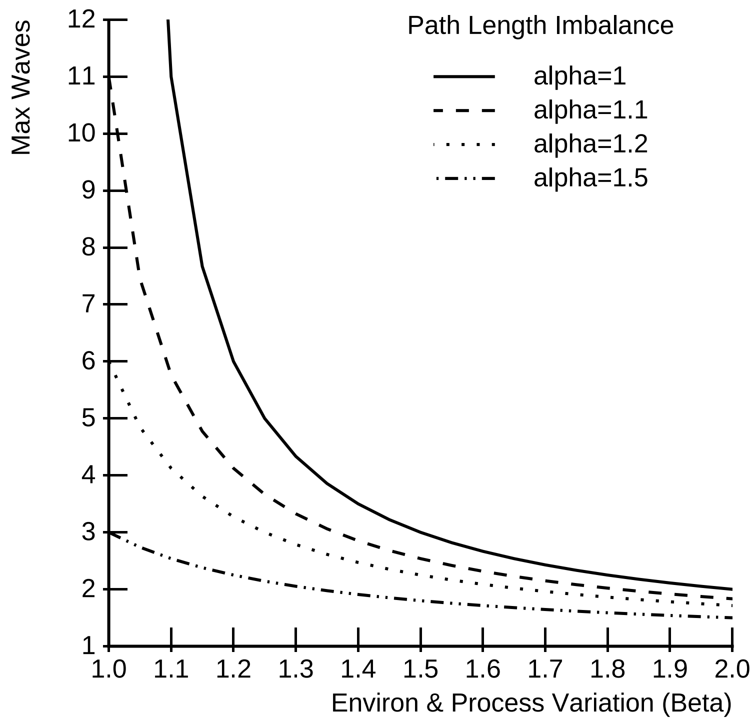


Figure 14: Maximum Waves vs. β

delay, is three or less. Process variation further reduces the number of waves which can be supported. Aggregating the above environmental variation with a process variation of $\beta_{proc} = 1.35$ the number of waves is limited to 1.6. Second, because operating environment changes result in significant changes in propagation delay, extremely accurate path-length balancing may not be necessary to achieve the maximum number of waves. For instance, if temperature and supply changes results in a relative propagation delay variation of 60%, i.e. $\beta = 1.6$, the path lengths through the network can differ by as much as 25% for two concurrent waves.

Environmental Impact Comparison In this section, the effects of environmental and process variation are contrasted for traditional pipelines and wave pipelines with fixed frequency clocking.

For a traditional pipeline, the minimum clock period over all acceptable temperatures and voltages is determined by the maximum propagation delay through the network. Thus the minimum clock period, T_{clk}^{min}, for a traditional pipeline which must operate over all possible expected supply voltages, temperatures, and process parameters is some factor, γ larger

Temp Range	Voltage Range	α	β	Max Waves
25C	5V	1	1	6
25C	4.5-5.5V	1	1.2	4
25-125C	5V	1	1.4	2
25-125C	4.5-5.5V	1	1.7	2

Table 3: Inverter Chain Simulated Maximum Number of Waves

than the clock period which could be achieved if it were expected to operate at the nominal supply, temperature, and process:

$$T_{clk}^{min}(\forall V, \forall \tau, \forall \phi) = \gamma T_{clk}(V_0, \tau_0, \phi_0) \tag{41}$$

where,

$$\gamma = \frac{P_{max}(V_{min}, \tau_{max}, \phi_{slow})}{P_{max}(V_0, \tau_0, \phi_0)} \tag{42}$$

and,

$$1 \leq \gamma \leq \beta \tag{43}$$

In these equations, the nominal voltage, temperature, and process are V_0, τ_0, and ϕ_0, respectively and all possible ranges of operation are represented by $\forall V$,$\forall \tau$, and $\forall \phi$.

$$\frac{T_{clk}^{min}(\forall V, \forall \tau, \forall \phi)}{T_{clk}(V_0, \tau_0, \phi_0)} = \gamma \tag{44}$$

This factor represents the maximum throughput lost by environmental and process variation.

For a wave pipeline,

$$T_{clk}^{min}(V_0, \tau_0, \phi_0) = P_{max}(V_0, \tau_0, \phi_0) - P_{min}(V_0, \tau_0, \phi_0) \tag{45}$$

$$T_{clk}^{min}(\forall V, \forall \tau, \forall \phi) = \alpha\beta P_{min}(V_{max}, \tau_{min}, \phi_{fast}) - P_{min}(V_{max}, \tau_{min}, \phi_{fast}) \tag{46}$$

assuming,

$$P_{max}, P_{min} >> \Delta C, T_s, T_h, RF_{min}, RF_{max} \tag{47}$$

Thus,

$$\frac{T_{clk}^{min}(\forall V, \forall \tau, \forall P)}{T_{clk}^{min}(V_0, \tau_0, \phi_0)} = \gamma \frac{\alpha\beta - 1}{\alpha\beta - \beta} \tag{48}$$

Figure 15 plots the degradation factors for both traditional and wave pipelines versus γ. It is assumed that for this figure any propagation delay through the network at the nominal environment is approximately equal to the propagation delay at maximum voltage and minimum temperature (i.e. $\gamma \approx \beta$.) Figure 15 is evidence of the need for minimization of environmental fluctuations for wave pipelined design.

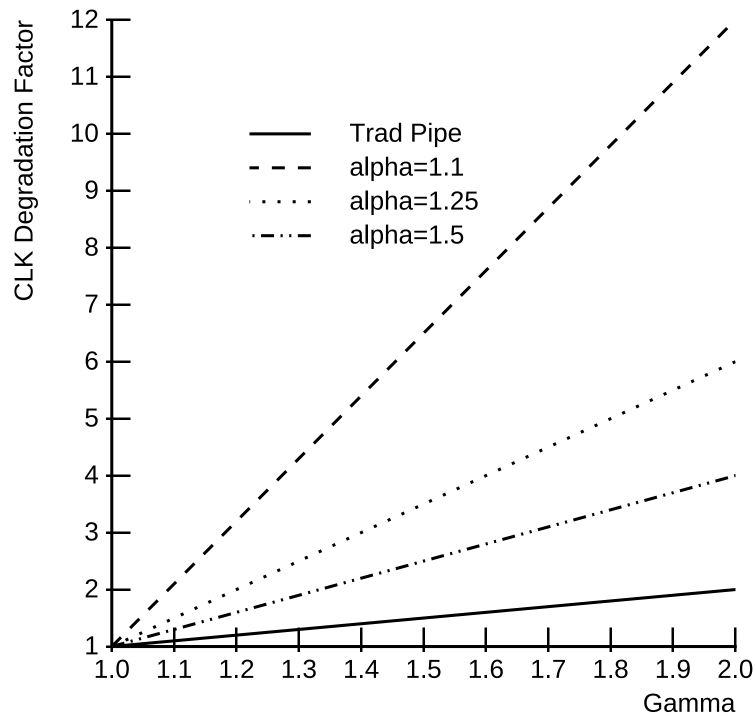


Figure 15: Environmental Degradation Factor

A strategy for maximizing the performance of externally-clocked wave pipelined circuits is tightly controlling the drift of the external power supply and minimizing Vdd and GND noise with numerous supply pins, filter capacitors on the die, and current-limiting I/O drivers. Temperature variation can be minimized by lowering the maximum junction temperature with low thermal resistivity packaging. Analysis of heat generation and flow could be used in the design process to provide tighter bounds on the expected temporal and spatial propagation delay variation. Lee [34] has suggested integrating thermal analysis in a design environment for improved reliability and performance. Temporal variation can be decreased by raising the minimum operating temperature with "warm-up" cycles.

Without tight controls on temperature and voltage, wave pipelined fixed-clock circuits are limited to 2-3 waves per stage.

For designs in which full commercial operation is required and tight environmental and process control are not practical, it is unreasonable to expect greater than two waves per wave pipelined logic block. A useful strategy in this case is to partition the logic into the smallest number of pipeline stages, k, such that constraint 2.3.4 with $N = 2$ is satisfied for each section. In this manner, each pipeline stage will be the minimum delay which holds two simultaneous waves. Therefore, the maximum speed-up over a nonpipelined circuit becomes $2 * k$ and the increase in latency will be minimized. Klass [30] analyzes pipelines in which each pipeline stage is in-turn wave pipelined.

2.3.5 Variable Frequency Clocked Systems

In an variable frequency clocked synchronous system, the clock period, $Tclk$, varies so as to match the propagation delay of the logic network.

The clock can be produced by a ring oscillator, voltage controlled ring oscillator, or external clock whose frequency is determined by on-chip delays. The clock frequency is a function of supply voltage, temperature, or fabrication process. VCOs which compensate for variations in supply voltage and temperature were analyzed with fixed frequency clocked systems. Figure 16 is a block diagram of a synchronous system with an internally generated, variable frequency clock.

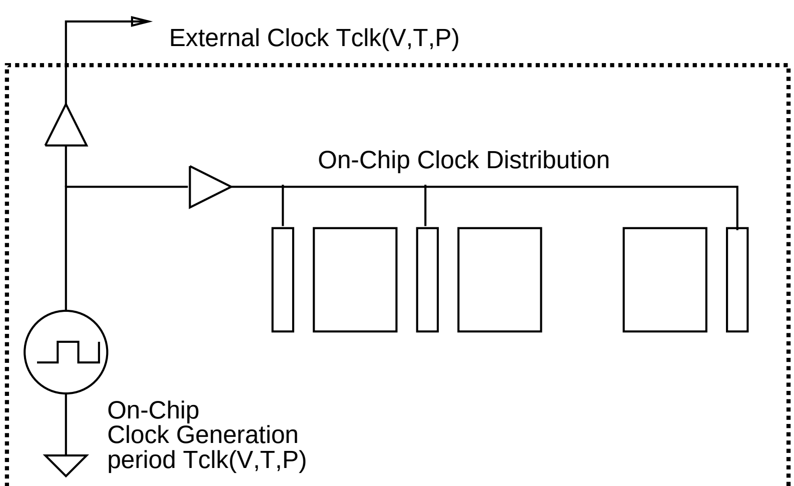


Figure 16: Internally Generated Variable Frequency Clocked System

A ring oscillator design and a voltage-controlled ring oscillator design are shown in Figure 17.

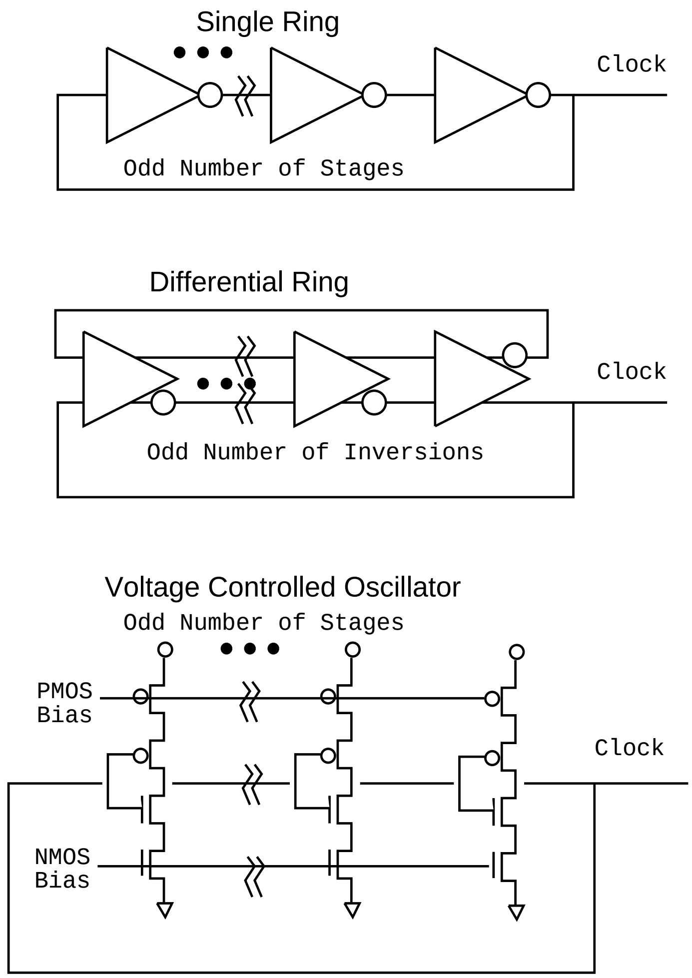


Figure 17: Internally Generated Clocks

For a variable frequency clocked traditional pipelined system to operate properly, the worst case maximum propagation delay determines the clock rate:

$$P_{max} + RF_{max} + T_s + \Delta C \leq T_{clk} \tag{49}$$

P_{max}, RF_{max}, T_s, and ΔC are voltage, temperature, and process dependent. T_{clk} is also voltage, temperature, and process dependent.

For a variable frequency clocked wave pipelined circuit to operate properly, the following two inequalities must hold for edge-triggered registers:

$$\frac{P_{max} + RF_{max}/2 + T_s + \Delta C + T_{synch} - cs}{N} \leq T_{clk} \tag{50}$$

$$\frac{P_{min} - RF_{min}/2 - T_h - \Delta C + T_{synch} - cs}{N-1} \geq T_{clk} \tag{51}$$

For flow latches, the following inequalities must hold:

$$\frac{P_{max} + RF_{max}/2 + T_s + \Delta C + T_{synch} - cs}{N} \leq T_{clk} \tag{52}$$

$$\frac{P_{min} - RF_{min}/2 - T_h - \Delta C + T_{synch} - T_{trans} - cs}{N-1} \geq T_{clk} \tag{53}$$

P_{max}, P_{min}, RF, T_{synch}, T_s and T_h are voltage, temperature, and process dependent. T_{clk} and T_{trans} are also voltage, temperature, and process dependent.

The period of oscillation of a ring oscillator is determined by the propagation delay through the ring. Thus if the temperature, voltage, and process were constant across the device, T_{clk} will vary as the combinational network propagation delay. According to Glasser [20] process parameters can be approximated as constant across a die. Surface temperature profiles of a die tend to be a superposition of a baseline temperature due to average die power dissipation and ambient temperature and hot-spots due to localized device power dissipation [22]. Thus, there is a spatially independent component and a spatially dependent component of temperature variation.

Power supply low frequency voltage variation is also time dependent due to supply drift and spatially dependent due to IR drops across the power distribution network.

Figure 18 compares the variation in propagation delay of a chain of inverters with the variation in clock period for a clock generated by an on-chip ring oscillator. This figure shows that inverter chain propagation delay and the ring oscillator period track if the temperature is spatially uniform.

Figure 19 compares the variation in propagation delay of the inverter chain with variation in period of an on-chip voltage-controlled ring-oscillator for spatially uniform temperature.

Spatial temperature variation depends upon power consumption, device placement, switching behavior, and package design. In the absence of heat flow analysis, worst case spatial temperature variation should be assumed.

With internally generated clocks, the clock frequency is a function of temperature and voltage, and is therefore not time invariant. This may present problems in interfacing a device to other devices in a system.

An additional problem for on-chip ring-oscillators is frequency jitter due to noise. Because the clocks used in wave pipelined circuits are constrained to a range of valid frequencies which becomes increasingly narrow as the number of waves through the logic increases [21],

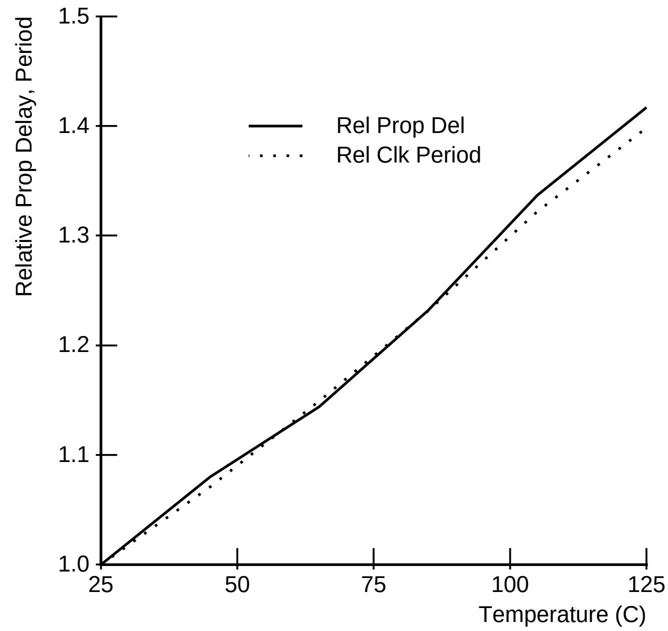


Figure 18: Inverter Chain Delay and Ring-Oscillator Period vs. Temperature

a high degree of clock frequency stability is necessary. This jitter must be included in the ΔC factor in the constraint computations. Low-jitter voltage and current-controlled oscillators minimize jitter through precise capacitance, current, and noise control. Jitter of less than 160 ppm is achievable for on-chip precision CMOS oscillator circuits [17]. They are, however, subject to frequency variation due to supply voltage and temperature changes. Further analysis of the impact of low jitter on-chip oscillators on wave pipelined designs is warranted.

2.3.6 Environmental Impact Comparison

In this section, the effects of environmental and process variation are contrasted for traditional pipelines and wave pipelines with variable frequency clocking.

For a traditional pipeline, the minimum clock period over all acceptable temperatures and voltages is determined by the worst-cast maximum propagation delay through the network. Thus:

$$T_{clk}^{min}(\forall V, \forall \tau, \forall \phi) = \gamma T_{clk}(V_0, \tau_0, \phi_0) \tag{54}$$

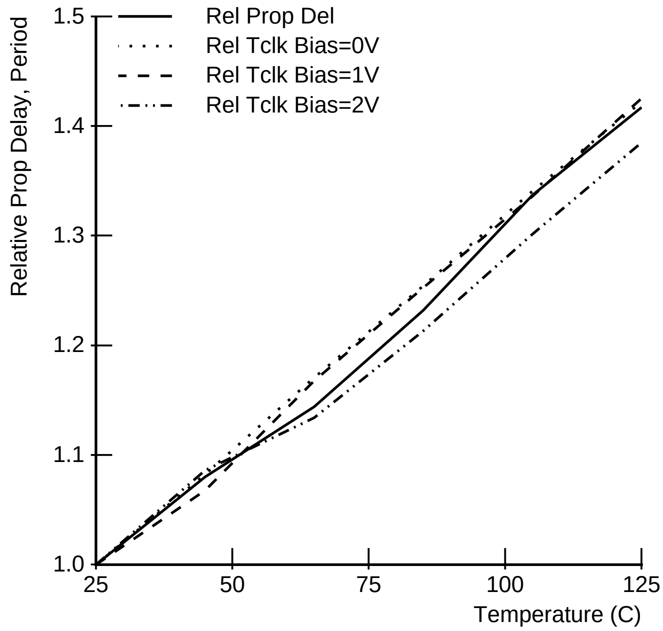


Figure 19: Inverter Chain Delay and VCO Period vs. Temperature

where,

$$\gamma = \frac{P_{max}(V_{min}, \tau_{max}, \phi_{slow})}{P_{max}(V_0, \tau_0, \phi_0)} \tag{55}$$

and,

$$1 \leq \gamma \leq \beta \tag{56}$$

or,

$$\frac{T_{clk}^{min}(\forall V, \forall \tau, \forall \phi)}{T_{clk}^{min}(V_0, \tau_0, \phi_0)} = \gamma \tag{57}$$

This factor represents the maximum throughput lost by environmental and process variation.

For a wave pipeline,

$$T_{clk}^{min}(V_0, \tau_0, \phi_0) = P_{max}(V_0, \tau_0, \phi_0) - P_{min}(V_0, \tau_0, \phi_0) \tag{58}$$

$$T_{clk}^{min}(\forall V, \forall \tau, \forall P) = \alpha\beta P_{min}(V_{max}, \tau_{min}, \phi_{fast}) - \beta P_{min}(V_{max}, \tau_{min}, \phi_{fast}) \tag{59}$$

assuming,

$$P_{max}, P_{min} >> \Delta C, T_s, T_h, RF_{min}, RF_{max} \tag{60}$$

Thus,

$$\frac{T_{clk}^{min}(\forall V, \forall \tau, \forall \phi)}{T_{clk}^{min}(V_0, \tau_0, \phi_0)} = \gamma \tag{61}$$

For variable frequency clocking with a uniform surface distribution of supply voltage and temperature, the impact of environmental and process variation affect traditional and wave pipelines equally. For these circuits, speed-ups such as those reported in [31, 14, 41, 35] of 2-7 are achievable.

For non-uniform surface temperature and supply voltage, wave pipelined circuits with variable-frequency on-chip clocks are subject to the performance constraints of Section 2.3.4 where β is due to the worst-case spatial variation of environmental conditions.

2.4 Performance Limits Conclusions

The effects of logic network path delay imbalance, changes in operating environment, and fabrication process variation on the performance of wave pipelines have been ascertained using CMOS delay models and the clock constraints which must be satisfied by wave pipelines.

Variations of propagation delays of different dice from common fabrication line are shown to have ratios of worst case delay to best case delays of 1.35 to 1.5. This variation, independent of other factors, limits the speed-up of wave pipelining to at most three to four. Changes in operating temperature may degrade delays by a factor of up to 1.4, thereby limiting the speed-up to 3.5. Supply drops and drift, supply noise, and coupled noise can result in further propagation delay variation.

Wave pipeline performance has been shown to be up to 6-times more sensitive to the effects of variation due to path imbalance, process, and environment than conventional pipelines.

This sensitivity motivates the design and optimization techniques for high performance wave pipeline design presented in the following chapter.

3 High Performance CMOS WP Design Techniques

Efficient wave pipelined system design requires tools for delay analysis and synthesis and optimization for delay balancing. As shown in chapter 2 to maximize the performance of a wave pipelined circuit, the delay variation in the combinational logic must be minimized. This variation is due to path length differences, data dependencies, and fabrication and environmental variations. In this chapter, CMOS circuit balancing and environmental delay variation compensation techniques are described.

3.1 Path Delay Balancing

Previous automated wave pipelined circuit balancing tools include technology independent cycle time optimization by delay insertion [12], balancing methods for bipolar CML circuits [53], CMOS cell placement [26], and logic restructuring [28].

A method of balancing static CMOS wave pipelined circuits through transistor sizing has been developed in this research. This method is an extension of the Wong method for balancing CML logic networks [53].

This tool uses a DAG representation of the delay of a CMOS combinational logic and formulates the balancing problem as a linear program. A piecewise linear gate delay function is found for each gate type via HSPICE simulations. This allows for flexible and accurate determination of gate delays.

The following sections detail the CMOS circuit delay model, the delay behavior of CMOS gates and networks, CMOS gate delay manipulation techniques, the linear program representation, simulated results, and technique limitations.

3.1.1 Modeling Circuit Delay Behavior

Circuit delay is represented by a polar weighted directed acyclic graph. Each node of the graph corresponds to a gate output in the CMOS logic network. A directed arc from node i to node j exists when the gate whose output is j has node i as an input. Each arc is labeled with a tuple *(rising gate delay, falling gate delay, unateness)*. The *unateness* field indicates which of *rising gate delay* or *falling gate delay* will be used for gate delay from input i to output j for a rising or falling transition at the input i. A source node is introduced into the DAG which connects to all primary inputs with zero delay, and a sink node is introduced which is connected to all circuit output nodes with zero delay arcs. An example circuit and graph are shown in Figure 20.

The delay of any path from the source node to the sink node is represented by a pair of delays *(rising sink delay, falling sink delay)*. Each of these delays is computed by summing

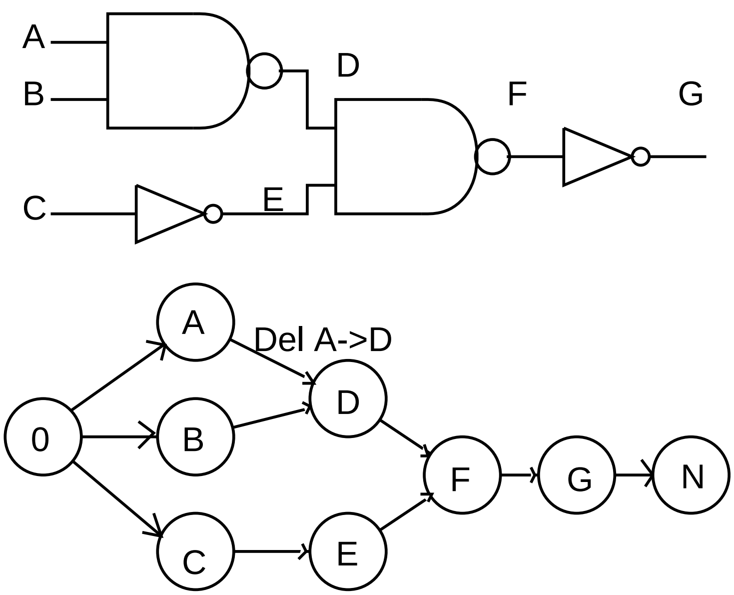


Figure 20: Example Circuit and Graph

the contributions of the individual gates along the path. The contribution from each gate is the tuple entry which results in the appropriate transition at the sink node. For positively unate gates, the input nodes inherit the output node transition direction. Negatively unate gates inherit the opposite transition direction.

DAGs with nonunate arcs cannot be represented with unique *(rising sink delay, falling sink delay)* pairs because the output of a nonunate gate can transition in a given direction as a result of a rising or falling of an input. Thus, for a given output transition direction, multiple path delays are possible. Path delays of circuits with nonunate gates are represented by the minimum and maximum bounds on the delays, *(min rising delay, max rising delay, min falling delay, max falling delay)*. The bounds at the output of a nonunate gate are computed recursively. For example the *max rising delay* to the output of nonunate gate i along a path is defined:

$$\begin{aligned} max\ rising\ delay_i &= \max(max\ rising\ delay_{i-1} + rising\ gate\ delay_i, \\ &\quad max\ falling\ delay_{i-1} + rising\ gate\ delay_i) \end{aligned} \tag{62}$$

The other bounds are similarly determined recursively, with the appropriate delay terms being substituted.

3.1.2 CMOS Gate and Network Delay

The determination of the delays associated with the circuit DAG is considered in this section. Static CMOS gate propagation delay behavior is a function of the capacitive load on the gate, the interconnection of the individual transistors in the gate, the voltage at the other inputs of the gate, the voltage at nodes internal to the gate, the rate of transition at the input, as well as the temperature, supply voltage, and signal noise at the gate.

A simple model of the propagation delay of a gate as the time required to charge or discharge a capacitance through a MOS transistor results in the following expression for gate delay [50]:

$$T_{pd} = \frac{k * C_l * V_{dd}}{I_{dsmax}} \tag{63}$$

where C_l is the total load capacitance, V_{dd} is the supply voltage, I_{dsmax} is the maximum MOS transistor source-drain current, and k is a factor to account for the difference between the maximum and average current over the switching period. The maximum current is:

$$I_{dsmax} = \frac{K}{2} * \frac{W}{L} * (V_{dd} - V_t)^2 \tag{64}$$

where K is the transconductance per unity channel width to length ratio, V_t is the device threshold, W is the effective transistor width, and L is the effective transistor length.

Thus, the delay can be approximated by:

$$T_{pd} = \frac{const * V_{dd} * C_l * L}{K * (V_{dd} - V_t)^2 * W} \tag{65}$$

The load capacitance is the sum of the wiring capacitance, C_{int}, and the gate capacitance of each transistor driven by the output:

$$C_l = C_{int} + \sum_i C_{gate_i} \tag{66}$$

The first-order gate capacitance of each transistor is related to the per unit area oxide capacitance and the transistor geometry length and width:

$$C_{gate\ i} = C_{ox} * W_i * L_i \tag{67}$$

Equation 65 can be rewritten so as to represent the driving device as an equivalent resistor:

$$T_{pd} \approx C_l * R_{eq} \tag{68}$$

Circuit path delays are the computed as the sum of the propagation delays of the gates along the path as detailed in the previous section.

3.1.3 Manipulating CMOS Delay

In equation 65 the transistor width and length and the load capacitance can be used to manipulate the delay of CMOS gates. In addition, the effective resistance of the gate (from equation 68) can be increased by introducing an additional resistor or transistor in series with the conducting device. The supply voltage and device threshold are assumed to be constant for all devices.

To efficiently balance CMOS wave pipelined circuits, it is desirable to have the manipulation of the delay of a given gate affect only that gate, not its successors or predecessors. One method of increasing the delay of a short path gates while isolating their predecessors is by increasing the load capacitance of the output nodes by adding discrete capacitors or source-drain shorted transistors [14]. This requires significant additional area to accomplish the balancing. The effective resistance of the gate can also be increased by the series addition of a poly resistor [26]. Like the added capacitance, this may result in a significant area increase if long poly wires must be introduced.

Independent manipulation of either the length or the width of the transistors does not have the desired effect. Increasing the length of a slow path transistor also increases the load capacitance of the gate which drives this gate. Decreasing the width of the device generally is not a viable option since as the width is decreased, the capacitive load on the previous stage is decreased. Thus, by increasing the delay of a given stage, the delay of its predecessor stage is decreased.

By manipulating the width and length of the devices together so as to maintain a constant gate capacitance, the delay of the gate can be adjusted without impact to the previous stage. During the delay tuning procedure, the length of the transistor is increased from L_0 to $L_0 * M_l$ and the width is decreased from decreased from W_0 to $W_0 * M_w$ ($M_w \leq 1$). From equation 67, load capacitance before delay tuning and after delay tuning is constant when:

$$W_0 * L_0 = W_0 * M_w * L_0 * M_l \tag{69}$$

This occurs when

$$M_w = 1/M_l \tag{70}$$

where M_w is the width modification factor and M_l is the length modification factor. Applying this result to the gate delay, equation 65, the transistor geometry manipulation results in a change in the delay of:

$$T_{pd}(M_w, M_l) = T_{pd}(M_w = 1, M_l = 1)\frac{M_l}{M_w} = T_{pd}(M_w = 1)\frac{1}{M_w^2} \quad (71)$$

The delay of a gate then becomes a monotonically decreasing, convex function of the modification factor.

The following delay tuning example compares the constant capacitance tuning method to the width-only and length-only tuning strategies. The delay tuning modes are applied to the simple circuit shown in Figure 21.

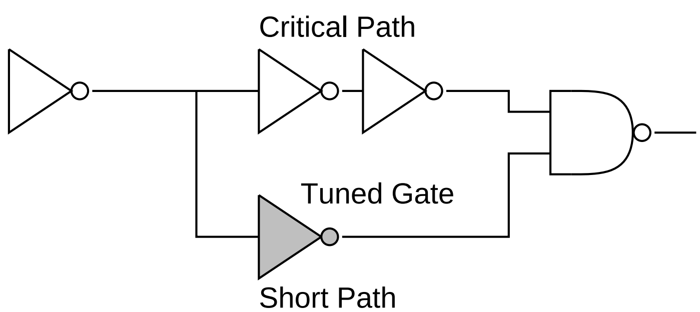


Figure 21: Delay Tuning Circuit

For this circuit, the delay through the critical path and the delay through the short path are shown in Figure 22 for the three tuning modes: length-only scaling, width-only scaling, and constant capacitance width/length scaling. The paths are balanced when the critical path line and the short path line intersect.

With length-only tuning, represented by the dashed lines, the lengths of the transistors in the tuned gate are increased from their nominal value of unity length modification factor ($M_l = 1$) to twice their nominal length ($M_l = 2$). The transistor widths are held constant. At approximately $M_l = 1.6$, the critical path delay and short path delay are balanced. However, the delay through the critical path has been increased by about 5%.

With width-only tuning, represented by the dotted lines, the widths of the transistors in the tuned gate are decreased from their nominal width ($M_w = 1$) to 2/3 of their nominal width ($M_w = 0.67$). With this range of width modification the critical and short path lines do not intersect and thus the circuit can not be balanced. In this example, this method decreases the delay of the critical path. In general, this is not desirable. The goal of delay tuning is to modify all paths to have delay equal to the critical path delay. By decreasing the delay of some paths, this method may result in additional variation between paths.

With constant capacitance tuning, represented by the solid lines, the widths of the transistors in the tuned gate are decreased from their nominal width ($M_w = 1$) to 2/3 of their nominal width ($M_w = 0.67$). Over this range, the lengths are increased so as to maintain a constant capacitance. At about $M_w = 0.7$ the critical path delay and short path delay are balanced. When balanced, the delay through the critical path has decreased by about 1%.

Length-only tuning, while providing sufficient delay tuning range, alters the critical path delay. Width-only tuning does not provide sufficient delay tuning range and alters the critical path delay. Constant capacitance scaling provides sufficient delay tuning range with minimum impact to critical path delay.

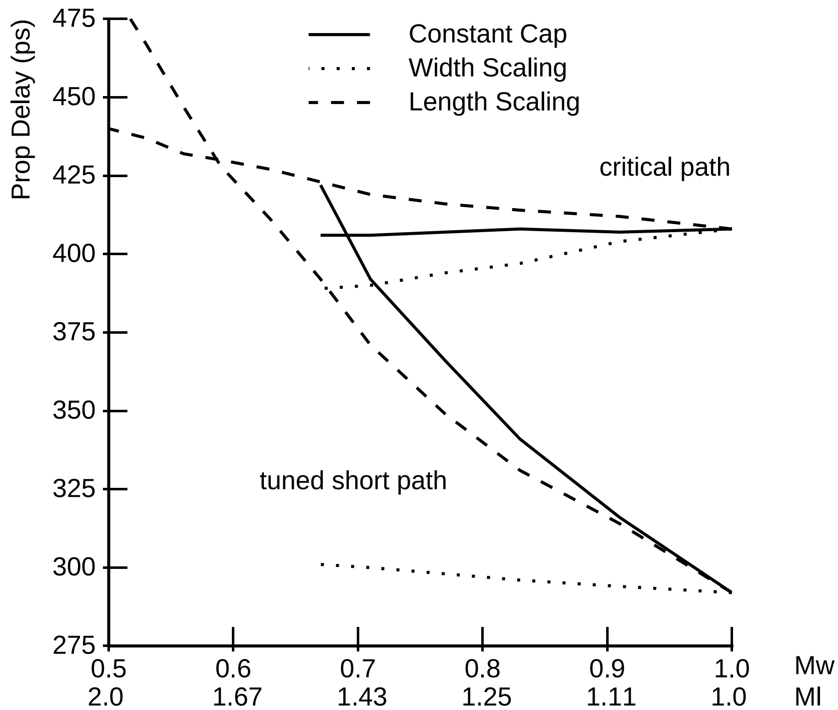


Figure 22: Delay Tuning Options

The balancing environment uses HSPICE simulations of each gate type, rather than the simple analytic models of gate delay, to accurately determine the constant capacitance transistor length to transistor width relationship. HSPICE simulations are used to ascertain gate propagation delay versus capacitive load versus transistor width modification factor. These simulations are used to build macromodels used by the balancing linear program solver.

Figure 23 shows the relationship between the propagation delay of an inverter driving a fanout of four inverters as a function of channel width modification.

The widths of all transistors in a gate are modified by this factor simultaneously in order to

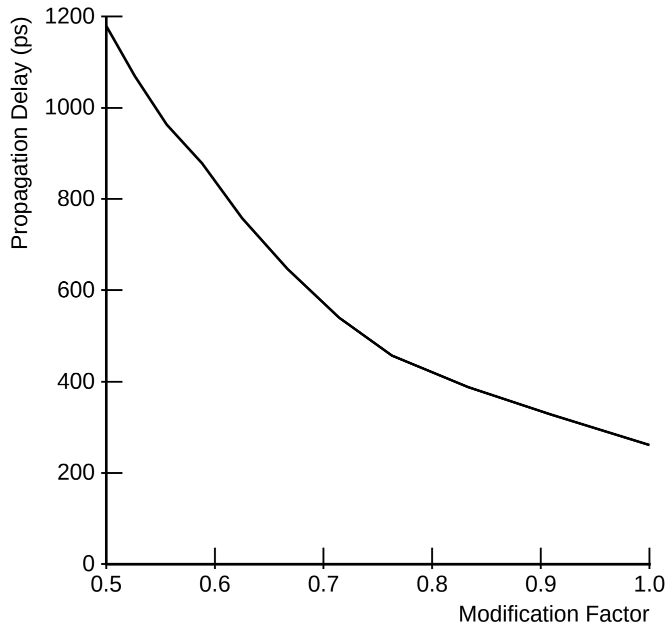


Figure 23: Inverter Propagation Delay vs. Modification Factor

modify the propagation delay of the gate. Although separate modification factors for rising gate output and falling gate output could be employed to separately modify the rising and falling delay, a single modification factor for the entire gate is employed so as to maintain equal rise and fall delays of the gates.

The automated CMOS wave pipeline balancing process consists of development of a circuit netlist and parameterizable gate library. It is assumed that the critical paths of the unbalanced netlist have been minimized. Therefore, delay balancing increases the delay along the short paths through the logic network. The unbalanced netlist is first rough balanced by the Wong rough balancing algorithm [53] which inserts buffers into paths which have local delay imbalances greater than a buffer delay.

The rough balanced netlist is converted to a linear program representation which is solved using the Simplex method [39]. The solution to the linear program is a vector of width modification factors from which the individual transistor geometries can be determined.

3.1.4 Linear Program Representation

The solution to the Fine Tuning Problem minimizes the cycle time of the wave pipelined circuit [53]. The Fine Tuning Problem is: Given a weighted acyclic DAG representation of a combinational circuit, find a vector of modification factors, M, such that all paths from the source to sink have delays which do not exceed a specified maximum propagation delay, D_{max}, and the delay of the shortest path is maximal.

The cycle time of a wave pipelined circuit is minimized when the difference between the longest and shortest path, ΔD, is zero. Gates whose delay functions are not equal for rising and falling output may result in a nonzero ΔD since the *rising sink delay* for a given path may differ from the *falling sink delay* along that same path. The relative sizes of the NMOS transistors and PMOS transistors in the gates are chosen to minimize the rising and falling delay differences. Stacked transistors and parallel conduction paths in CMOS gates limit the degree to which these delay functions can be equalized.

The Symmetric Fine Tuning Problem [53] is a restricted case of the Fine Tuning Problem in which the rising delay function of each gate equals its falling delay function. The solution to this problem is also a solution to the Fine Tuning Problem with a bounded ΔD.

A related problem, whose solution is also a solution to the Symmetric Fine Tuning Problem is solved by the delay balancing tools. The Width Minimization Problem is: Given a weighted acyclic DAG representation of a combinational circuit, find a vector of modification factors, M, such that all paths from the source to sink have delays which do not exceed D_{max} and the total transistor width is minimized. The Width Minimization Problem is:

$$Minimize\ W = \sum_i M[i] * W[i],\ Such\ that: \tag{72}$$

$$W_{min}[i] \leq M[i] * W[i] \leq W_{max}[i]$$

$$D[0,0] = D[0,1] = 0$$

$$D[N,0] \leq D_{max}$$

$$D[N,1] \leq D_{max}$$

If gate corresponding to arc (i,j) is positive unate:

$$D[i,1] + f_{lin}[i,j,1](M[x], C_l[j]) \leq D[j,1]$$

$$D[i,0] + f_{lin}[i,j,0](M[x], C_l[j]) \leq D[j,0]$$

If gate corresponding to arc (i,j) is negative unate:

$$D[i,1] + f_{lin}[i,j,0](M[x], C_l[j]) \leq D[j,0]$$

$$D[i,0] + f_{lin}[i,j,1](M[x], C_l[j]) \leq D[j,1]$$

If gate corresponding to arc (i,j) is nonunate:

$$D[i,1] + f_{lin}[i,j,1](M[x], C_l[j]) \leq D[j,1]$$

$$D[i,1] + f_{lin}[i,j,0](M[x], C_l[j]) \leq D[j,0]$$

$$D[i,0] + f_{lin}[i,j,0](M[x], C_l[j]) \leq D[j,0]$$

$$D[i,0] + f_{lin}[i,j,1](M[x], C_l[j]) \leq D[j,1]$$

where f_{lin} is a piecewise linear approximation to the monotonically decreasing, convex delay function f, $D[i,0]$ is the delay from the source node to node i falling, $D[i,1]$ is the delay from the source node to node i rising, $C_l[j]$ is the capacitive load on node j, $M[i]$ is the width modification factor of transistor i, $W[i]$ is the nominal width of transistor i, $W_{min}[i]$ and $W_{max}[i]$ are fixed limits on the width of a transistor, node 0 is the source node, and node N is the sink node .

The optimal solution to the Width Minimization Problem, M, in which all of the delay inequality constraints in the problem description are *active*, i.e. the constraints are at their limit and are thus equalities, is also a solution to the Symmetric Fine Tuning Problem in which all paths have a delay of D_{max} and $\Delta D = 0$.

The solution to the Width Minimization Problem with active constraints represents an accurately balanced circuit. In a design system using parameterizable library cells whose cell height is fixed and whose cell width is a linear function of the transistor widths the solution is area efficient.

3.1.5 Design Process and Simulated Results

The typical design process involves generation of an unbalanced netlist file with no wiring capacitances. The unbalanced netlist is first rough balanced by the Wong rough balancing algorithm [53] which inserts buffers into paths which have local delay imbalances greater than a buffer delay.

The rough balanced netlist is converted to a linear program representation which is solved using the Simplex method. The solution to the linear program is a vector of width modification factors. The solution to the linear program is then slack tested. This test identifies all gates which do not have any *active delay constraints*. These gates are at the upper limit of the delay that can be achieved through transistor sizing and, for full balance, additional delay is necessary. More tunable gate modules are substituted for these gates and the fine tuning procedure is repeated. The solution to the slack tested linear program is used by the module generators to produce layout cells. Once layout is completed, a netlist file which includes the parasitic capacitances is generated. This netlist is fine tuned for additional balance accuracy and the layout specific information is generated. Optionally, the circuit may be iteratively reextracted and fine tuned to accommodate differences in layout.

The wave pipelined balancing tool is incorporated into a design environment based on the Mentor Graphics GDT design tools. The design process for cell generation, design optimization, layout, and routing is shown in Figure 24.

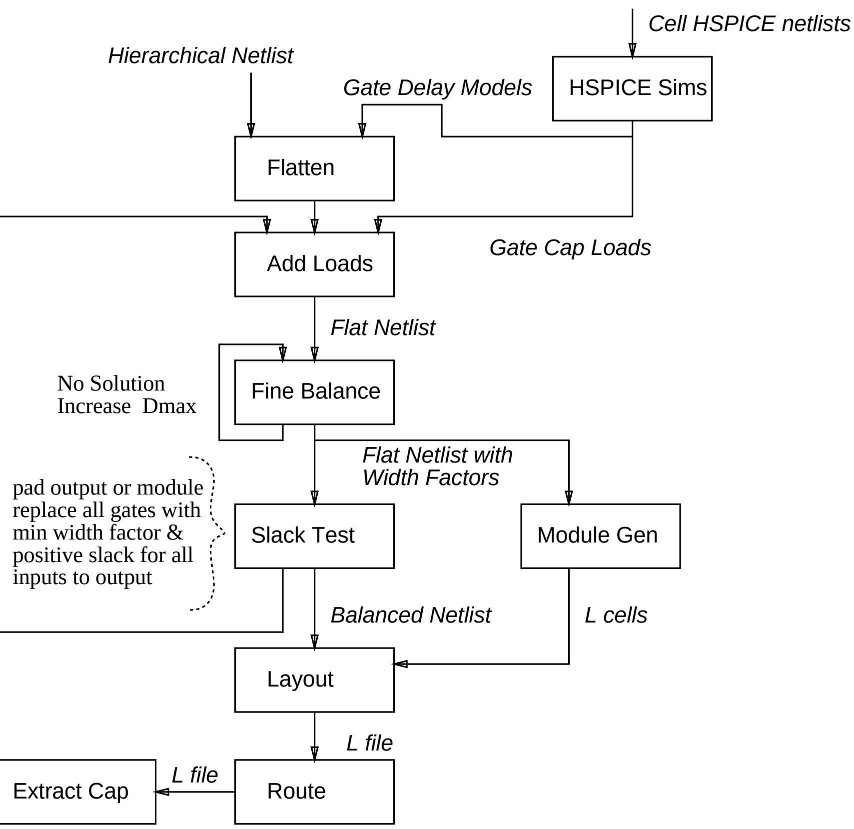


Figure 24: Design Process

To test the capabilities of the balancing tool, several demonstration circuits have been balanced and simulated with HSPICE: a pulse generator, an unbalanced carry generation circuit, a (4,2) counter circuit, a 16-bit parallel adder, and an 8-bit x 8-bit multiplier. HSPICE simulations of the circuits were for typical fabrication parameters for a 0.8-micron, 3-level metal CMOS process operating at 25 C.

The counter, adder, and multiplier circuits are typical of arithmetic designs. The carry tree is an unbalanced NAND and inverter logic structure. The pulse generator is a simple circuit which relies on differences in propagation delay in order to produce a pulse.

For these example circuits a static CMOS parameterizable gate library consisting of four sizes of inverters and three two-input NANDs was used. Balancing results for these circuits were consistent with a study by Klass [32] which showed that despite a large data-dependent delay of individual static CMOS gates, circuits designed with static CMOS inverters and 2-input NAND gates exhibit small data-dependent delay variation. These gates were also easily macromodeled.

Figure 25 is a diagram of the pulse generator circuit prior to rough tuning, following rough tuning, and after fine tuning. When this circuit is perfectly balanced by the tool, no pulse is output. Rough-tuning allows the inputs to the NAND gate to cross the Vdd/2 volt threshold within 122ps of each other, thereby inducing only a small perturbation of the NAND gate output. Fine-tuning results in 17ps difference in arrival time with no perturbation in the NAND output.

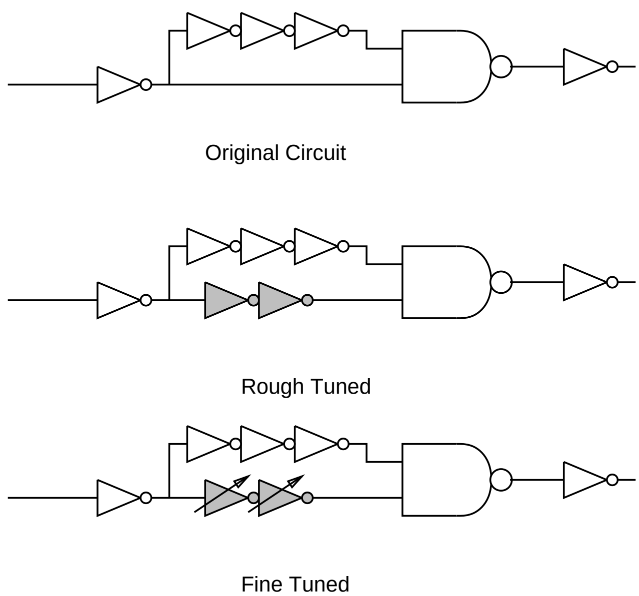


Figure 25: Pulse Circuit Balancing

A (4,2) counter circuit was rough balanced, fine balanced, and simulated with HSPICE. Figure 26 is a histogram of the delay of the carry output for all possible pairs of input vectors which cause a change in the carry output of the unbalanced circuit. The unbalanced circuit had a maximum delay of 1.64 ns and a delay variation of 970 ps. Figure 27 is a histogram of the delay of the fine balanced circuit. The maximum delay through the balanced circuit is 1.73 ns and the maximum delay variation is 370 ps. Because these counters are cascaded, the fast carry-out output was connected to the carry-in input for the balancing procedure. In the balancing procedure, the critical path delay of this circuit was increased by approximately 90ps. This increase is due to the simple delay model used by the balancing tool. The limitations of this method of fine balancing are discussed in Section 3.1.6.

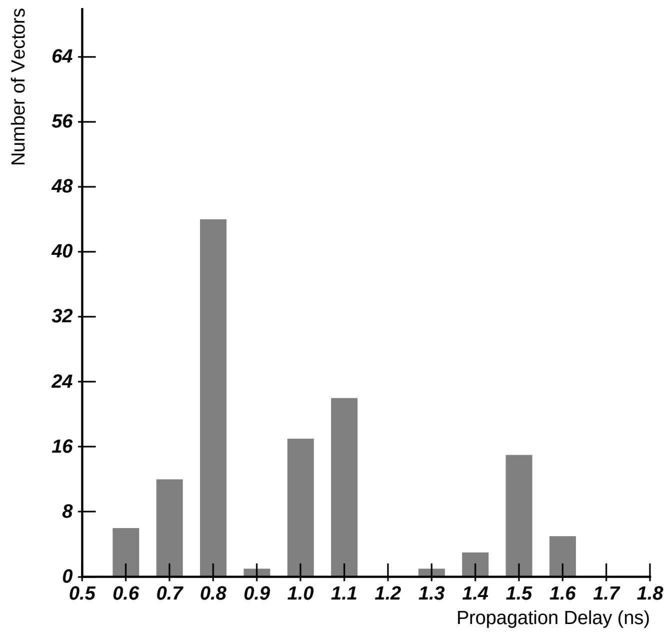


Figure 26: Unbalanced Counter Delay Histogram

The carry generation circuit is shown in Figure 28. This circuit was balanced and simulated with HSPICE. For heuristically chosen critical vectors, the original circuit exhibited a maximum delay of 1.74 ns and a delay variation of 955 ps. The fine-tuned circuit had a maximum delay of 1.74 ns and a delay variation of 250 ps.

The 8x8 multiplier and 16-bit adder used in the wave pipelined vector unit were balanced with this tool. Balancing included input and output bus capacitive loading. The balancing details are given in table 4.

3.1.6 CMOS Fine Balancing Limitations

The accuracy of the fine balancing procedure is limited by several factors: 1) use of the symmetric fine tuning problem as an approximation to the fine tuning problem; 2) the piecewise approximation of the delay function; 3) the changes in gate delay due to tuning induced changes in input slew rate; 4) the variation in the delay of a gate due to the state of other inputs and internal capacitances; and 5) the false path problem.

The false path problem results from the use of logic network topology in the determination of path delays. Topological delays, which are simply the sum the delays of interconnected gates, may include "false" paths, i.e. the function realized by the gates or the arrival time

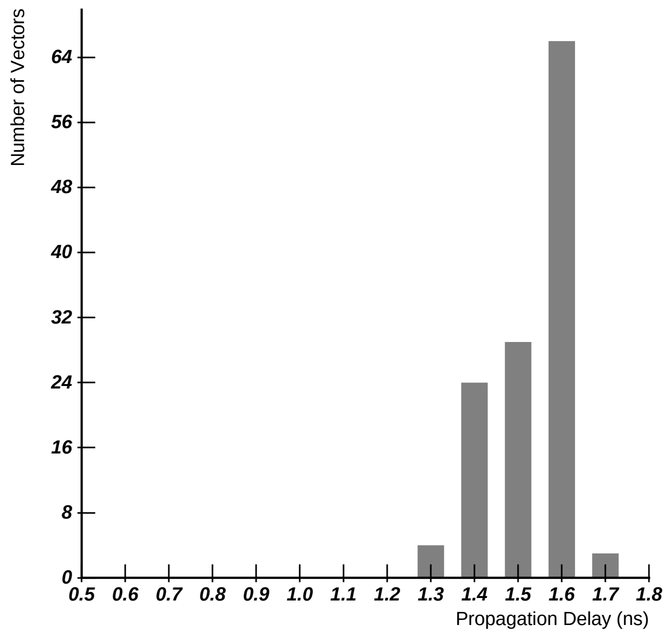


Figure 27: Fine Balanced Counter Delay Histogram

	Multiplier	**Adder**
Max Delay	10.84ns	5.52ns
Delay Variation	1.37ns	0.58ns
Unbalanced No. Trans.	5526	1322
Balanced No. Trans.	6466	1730
Unbalanced Area	1.95sqmm	0.398sqmm
Balanced Area	2.17sqmm	0.451sqmm
Balancing Time	10.94hr	0.62hr

Table 4: Vector Unit Logic Balancing Results

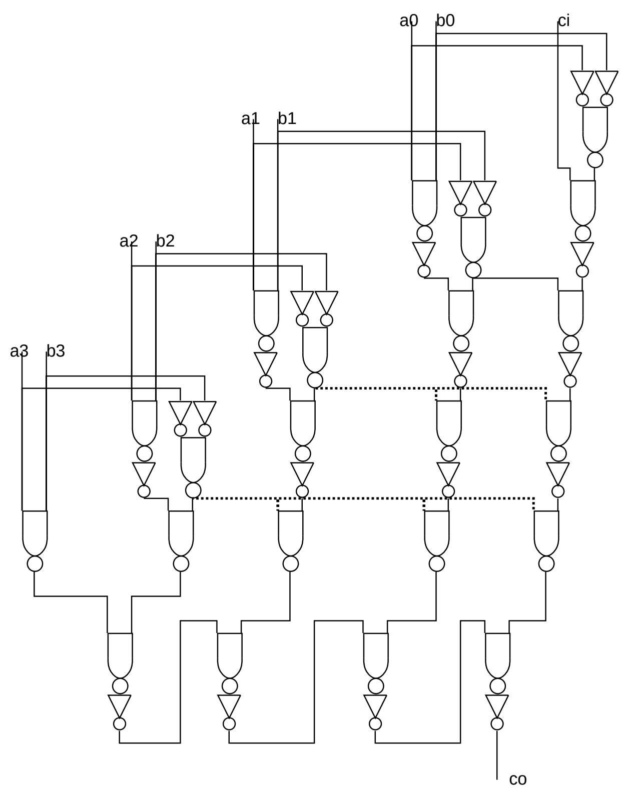


Figure 28: Carry Generation Circuit

of other inputs prevent data from propagating along the specified path. Thus, false paths tend to result in a pessimistic estimate of critical path delay.

Despite these limitations, this method provides balancing variation of 10% to 20% of maximum delay. This accuracy is consistent with the limits of manual balancing methods by Klass [31]. Higher accuracy in balancing provides diminishing returns, as the wave pipeline cycle time becomes dominated by other factors [43].

3.2 Wave Pipeline Synchronization

In this section, the synchronization of wave pipelines is examined. To make use of the performance potential made possible by the minimization of delay variation, the timing relation between input and output synchronizers must be established. The input and output clocks for each wave pipeline in a system must meet the constraints 1 and 2. The clock period and the phase relation between skewed clocks must be specified to ensure the interface and synchronization of wave pipelines and other synchronous logic in the system.

A dichotomy of wave pipelines is introduced to specify synchronization methods for each classification of wave pipelined system. Wave pipeline synchronization requires that each wave pipeline in the system is optimized such that it meets the two sided clock constraints for a clock with a period determined by the maximum delay variation of all pipelines in the system. The optimization performed to synchronize the wave pipelines is different from the optimization performed during balancing. In synchronizing, all paths through the wave pipeline are adjusted in common or the timing relation between the input and output clocks are adjusted such that all pipelines operate with the smallest valid system clock period.

The following terminology is used to characterize wave pipelines:

A *synchronizing element* is a pipeline register or latch.

A *synchronizer* is a set of one or more synchronizing elements which share a common clock signal; i.e. there is no constructive skew between synchronizing elements in a synchronizer.

A *wave pipeline* consists of a combinational logic network and two sets of synchronizers, *input synchronizers* and *output synchronizers*. The combinational logic network is a set of all paths between adjacent synchronizers. The input synchronizers are those which provide input data to the combinational logic. The output synchronizers are those which store the output of the combinational logic.

A *monoharmonic wave pipeline* is a wave pipeline in which all paths through wave pipeline i can support up to N_i waves of data. All inputs share a common reference clock and all outputs share a common reference clock. Previous research [12, 21, 27, 6] has concentrated on this classification of wave pipelines.

A *polyharmonic wave pipeline* is a wave pipeline in which the paths through wave pipeline

i can be divided into j sets of paths each of which can support up to N_i^j waves of data. Figure 29 illustrates three important examples of polyharmonic wave pipelines: parallel wave pipelines, forward receiving wave pipelines, and forward generating pipelines.

Parallel wave pipelines result when common input and output synchronizers are used to source data to and store data from separate combinational logic paths. For instance, in the vector unit described in chapter 4 the multiplier and adder share common input and output synchronizers. The propagation delays through the multiplier is approximately twice that of the adder, and thus the number of waves supported by the multiplier paths are approximately twice that of the adder paths.

The forward generating and receiving wave pipelines occur in interfacing of wave pipelines to other wave pipelines or traditional synchronous circuits. These polyharmonic wave pipeline results when inputs and/or outputs have more than one reference clock. In the forward generating case, intermediate results are output from of the combinational logic such that the number of waves in the paths to the intermediate results differs from the number of waves in the paths to the primary outputs. In the forward receiving case data is input into the wave pipeline such that the paths from the forward inputs to the primary outputs support fewer waves of data than the paths from primary inputs to primary outputs.

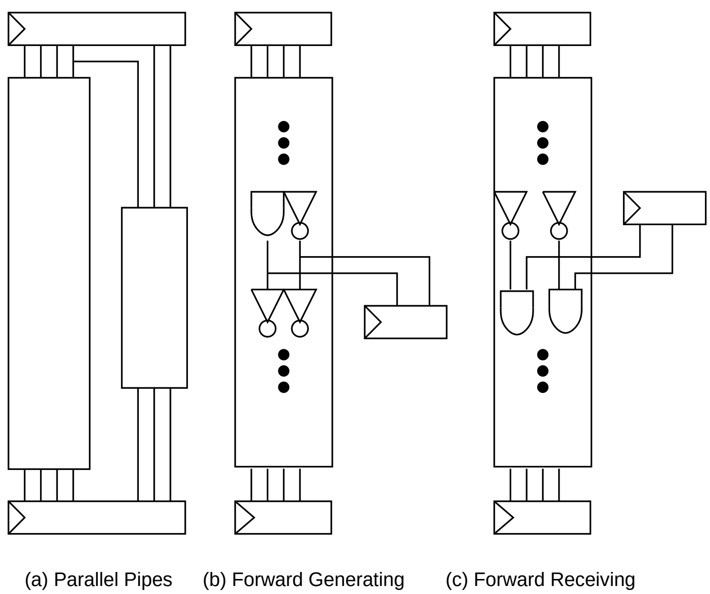


Figure 29: Example Polyharmonic Wave Pipelines

A wave pipeline with *feedback* is a wave pipeline in which there exists a path from the output of the wave pipeline through any number of synchronizers to the input of the wave pipeline.

Ekroot [12] developed a method of ensuring correct wave pipeline operation at a minimum clock period for monoharmonic wave pipelines and multistage systems of monoharmonic wave pipelines with feedback. With this method, solutions to linear program representations of the circuit optimizations specify the amount of intentional delay inserted into the combinational logic and the amount of constructive clock skew for each register which achieves the minimum clock period. This method restricts wave pipelines to using edge triggered registers and to common input registers for all inputs and outputs.

Extending the work of Fishburn [15] on clock skew optimization, Joy and Ciesielski [27] have formulated the optimization of constructive clock skew so as to minimize the clock period of register-based wave pipelined systems as a linear program. This method does not constrain the wave pipeline to a common input clock. Gray, et. al. [21] using similar formulations for skew optimization include feedback for register-based multistage wave pipelined systems. These clock optimizations to not employ intentional delay insertion.

Synchronization methods for wave pipelines of increasing complexity will be examined in the following sections. Edge triggered synchronizers are considered in this analysis.

3.2.1 Monoharmonic Wave Pipelines Lacking Feedback

For wave pipelined systems in which each wave pipeline support only one number of concurrent waves and none of the wave pipelines have feedback, two methods of optimization for wave pipeline synchronization can be used: constructive clock skew and path delay insertion.

The two sided limit on the clock period of a wave pipeline must be satisfied for each wave pipeline i in a wave pipelined system design:

$$\frac{P^i_{max} + \delta P^i_{max} + H^i_{max} - cs^i}{N_i} \leq T_{clk} \leq \frac{P^i_{min} + \delta P^i_{min} - H^i_{min} - cs^i}{N_i - 1} \tag{73}$$

H^i_{max} is the worst case maximum synchronizer overhead:

$$H^i_{max} = \Delta C + T^i_s + \frac{RF^i_{max}}{2} + T^i_{synch} \tag{74}$$

H^i_{min} is the worst case minimum synchronizer overhead:

$$H^i_{min} = \Delta C + T^i_h + \frac{RF^i_{min}}{2} - T^i_{synch} \tag{75}$$

i spans the range of all wave pipelines in the system. cs^i is the amount of constructive clock skew between the input and output synchronizers. The magnitude of the clock skew is less than the clock period, and the sign of cs^i is positive if the output clock lags the input clock and negative otherwise. δP^i_{max} is the amount of intentional path delay added to each path between input and output synchronizers under worst case operating conditions. δP^i_{min} is the same intentional path delay under best case operating conditions. The latter two quantities are always positive.

Without constructive clock skew and intentional path delay, the minimum clock period which satisfies the constraints may not be limited by the maximum propagation delay variation. The valid clock intervals are not continuous for wave pipelines in which there is delay variation in the combinational logic and/or clock overhead.

Optimization of the wave pipeline circuits, via constructive clock skew and combinational logic delay modification, can be used to achieve the minimum clock period. The minimum clock period, that which is determined by the delay variation is:

$$T_{clk}^{opt} = max(P_{max}^i + H_{max}^i - P_{min}^i + H_{min}^i) \tag{76}$$

This clock period can be achieved through constructive clock skew and intentional delay insertion when the skew and added delay have no variation.

Insertion of additional path delay into each path can be used to ensure that the output register timing constraints are met. The bounds on the additional intentional path delay, δP_{max}^i and δP_{min}^i, for each wave pipeline i can be determined:

$$\delta P_{min}^i \geq (N_i - 1)T_{clk}^{opt} - (P_{min}^i - H_{min}^i) \tag{77}$$

$$\delta P_{max}^i \leq N_i T_{clk}^{opt} - (P_{max}^i + H_{max}^i) \tag{78}$$

The number of waves, N_i, is:

$$N_i = \lceil \frac{P_{max}^i + H_{max}^i}{T_{clk}^{opt}} \rceil \tag{79}$$

For wave pipelines without feedback, constructive clock skew can also be used to ensure that the output register timing constraints are met. The constructive clock skew, cs^i, can be determined:

$$cs^i \leq (P_{min}^i - H_{min}^i) - (N_i - 1)T_{clk}^{opt} \tag{80}$$

$$cs^i \geq (P_{max}^i + H_{max}^i) - N_i T_{clk}^{opt} \tag{81}$$

The number of waves, N_i, is:

$$N_i = \lfloor \frac{P_{max}^i + H_{max}^i}{T_{clk}^{opt}} \rfloor \tag{82}$$

Figure 30(a) is an example of a system of monoharmonic wave pipelines with no feedback. Without optimization, the minimum clock period ignoring clock overhead is 3.25ns which is greater than the optimal clock period of 3ns. Figure 30(b) shows the same system which has been optimized through intentional delay insertion. This solution achieves a cycle time of 3ns at the cost of 3ns of additional latency. Figure 30(c) shows the same system which has been optimized through constructive clock skew. This solution achieves a cycle time of 3ns. The cost of this solution is a reduction of 2ns the allowable delay of any circuit which uses the output of the final synchronizer.

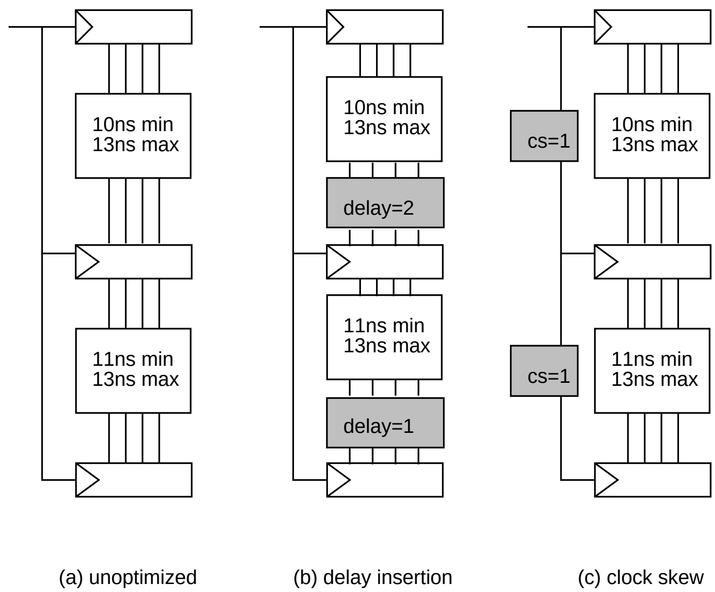


Figure 30: Monoharmonic Wave Pipeline without Feedback Optimization

For monoharmonic wave pipelines without feedback, the constructive clock skew solutions found by the linear program methods of Grey, et. al. [21] and Joy and Ciesielski [27] will satisfy the clock skew limits 81 and 80. The intentional path delay found by the linear program method of Ekroot [12] will satisfy the limits 77 and 78.

While the constructive clock skew method does not increase wave pipeline latency, it does increase the complexity of clock distribution and wave pipeline interface. The intentional path delay method increases wave pipeline latency but does not add complexity to clock generation and distribution or interface.

3.2.2 Polyharmonic Wave Pipelines Lacking Feedback

In a polyharmonic wave pipeline, separate paths between an input and output synchronizer may support different numbers of waves. The clock interval constraints must be satisfied for all paths. A polyharmonic wave pipeline can be synchronized using intentional path delay or a combination of intentional path delay and constructive clock skew.

To optimize the performance of a polyharmonic wave pipeline which does not have feedback, the following procedure is used. The maximum propagation delay from the input synchronizer to output synchronizer, denoted by P^{max}_{max}, is determined. If P^{max}_{max} is in the set of paths which can support N^i waves, in the constraint analysis, a delay term, $X_{ij}T^{opt}_{clk}$, is determined for all paths which support fewer than N^i waves. This term represents the maximum integer number of clock periods by which the maximum propagation delay through the wave pipeline exceeds the path under consideration. The difference in the number of harmonics between paths i and j, X_{ij}, is an integer bounded by:

$$\frac{P^i_{max} - P^j_{max} + H^i_{max} - H^j_{max} - \delta P^j_{max}}{T^{opt}_{clk}} - 1 \leq X_{ij} \tag{83}$$

and,

$$X_{ij} \leq \frac{P^i_{max} - P^j_{max} + H^i_{max} - H^j_{max} - \delta P^j_{max}}{T^{opt}_{clk}} \tag{84}$$

Once the difference in the number of waves supported by the path under consideration and the maximum delay path is determined, the intentional delay added to each path, δP^j_{max}, is found. Thus, for each path the bounds on the intentional delay introduced which ensures synchronization is:

$$\frac{P^j_{max} + \delta P^j_{max} + H^j_{max} - cs}{N^i - X_{ij}} \leq T^{opt}_{clk} \leq \frac{P^j_{min} + \delta P^j_{min} - H^j_{min} - cs}{N^i - 1 - X_{ij}} \tag{85}$$

Thus, for each path the bounds on the intentional delay introduced which ensures synchronization is:

$$\delta P^j_{min} \geq (N^i - X_{ij} - 1)T^{opt}_{clk} - (P^j_{min} - H^j_{min}) \tag{86}$$

$$\delta P^j_{max} \leq (N^i - X_{ij})T^{opt}_{clk} - (P^j_{max} + H^j_{max}) \tag{87}$$

Intentional clock skew can be used to reduce the amount of intentional path delay in cases where all paths in a polyharmonic wave pipeline share a common input and output synchronizer and have nonzero intentional path delay.

Figure 31 is a polyharmonic wave pipeline: two combinational logic blocks share input and output registers. The delays through these units are significantly different and it is advantageous to have a different number of concurrent waves in each block. Since they share clocks, the clock period constraints 73 must be satisfied for both pipelines. In Figure 31(a) assuming no clock skew nor register overhead, the minimum clock period is 5ns. By lengthening the delay through the long pipe so as to be a multiple of the minimum clock period of the longer pipe, as shown in 31(b), the optimal clock period of 3ns can be achieved. Figure 31(c) demonstrates the use of both intentional delay insertion and constructive clock skew to optimize performance.

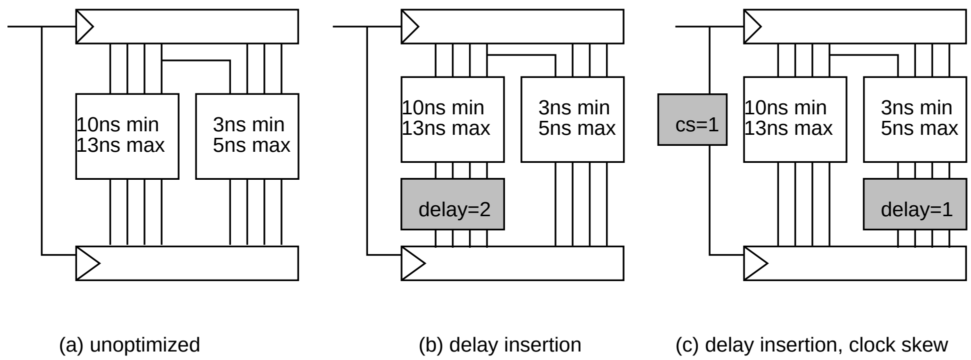


Figure 31: Polyharmonic Wave Pipeline without Feedback Optimization

The balancing method presented in Section 3.1 allows for the insertion of intentional delay for synchronization. Delay buffers are inserted and device geometries are adjusted such that all path delays are increased by the necessary delay. Since the adjusted pipelines may have a delay variation which violates the minimum clock period constraint, the delay variation of each pipe must be reverified. A polyharmonic wave pipeline exists in the vector unit: The vector register output is supplied to either the adder or multiplier, operated upon, and returned to the vector register file. The register read and write for both types of operations are performed with reference to a common clock. The delay through the multiplier is approximately double that through the adder. This optimization technique was employed to tune the delay of the demonstration vector unit's adder output buffers to ensure proper wave pipeline synchronization.

3.2.3 Monoharmonic Wave Pipelines with Feedback

Feedback in a monoharmonic wave pipeline does not affect optimization based upon intentional path delay insertion. It does, however, complicate optimization using constructive clock skew. Skew introduced between the input and output synchronizer of any sequential circuit changes the timing relation between the input synchronizer and output synchronizer and also changes the relation between the output synchronizer and the subsequent synchronizer along any path.

For monoharmonic wave pipelines with feedback, the clock period due to delay variation is:

$$T_{clk}^{opt} = max(P_{max}^i + H_{max}^i - P_{min}^i + H_{min}^i) \tag{88}$$

For wave pipelines with feedback, constructive clock skew with intentional delay insertion can be used to ensure that the output register timing constraints are met. The constructive clock skew, cs^i, can be determined:

$$cs^i \leq (P_{min}^i - H_{min}^i) - (N_i - 1)T_{clk}^{opt} \tag{89}$$

$$cs^i \geq (P_{max}^i + H_{max}^i) - N_i T_{clk}^{opt} \tag{90}$$

The number of waves, N_i, is:

$$N_i = \lfloor \frac{P_{max}^i + H_{max}^i}{T_{clk}^{opt}} \rfloor \tag{91}$$

To ensure the synchronization of the feedback signals, intentional path length delay may be required. The amount of intentional delay is found by the following procedure. Construct a directed graph G with $|V|$ vertices and $|E|$ edges. Where each vertex v_i corresponds to synchronizer i, and each edge e_{ij} corresponds to the existence of a combinational logic connection without a intervening synchronizer from synchronizer i to synchronizer j. A weight $w_{ij} = cs^i$ is attached to each arc. The amount of delay necessary at each feedback point of wave pipeline i is found by determining the amount of additional weight added to the feedback arc of the graph to make the sum of the weights around the closed path which includes that feedback path equal to zero and computing the residue of that quantity with respect to the clock period.

$$\delta P_{max}^i = (-1 * \sum_{loop} w_{ab}) \ mod \ T_{clk}^{opt} \tag{92}$$

Figure 32 demonstrates optimization of an example monoharmonic wave pipeline with feedback. In Figure 32(a), the unoptimized circuit the minimum clock period is 7ns, significantly greater than the optimal value of 3ns. Figure 32(b) shows the skew graph constructed for this circuit. Figure 32(c) shows the optimized circuit which through intentional delay insertion and constructive clock skew is able to achieve the optimal clock period of 3ns.

This method achieves the delay variation limited clock period at the expense of additional latency in the feedback paths.

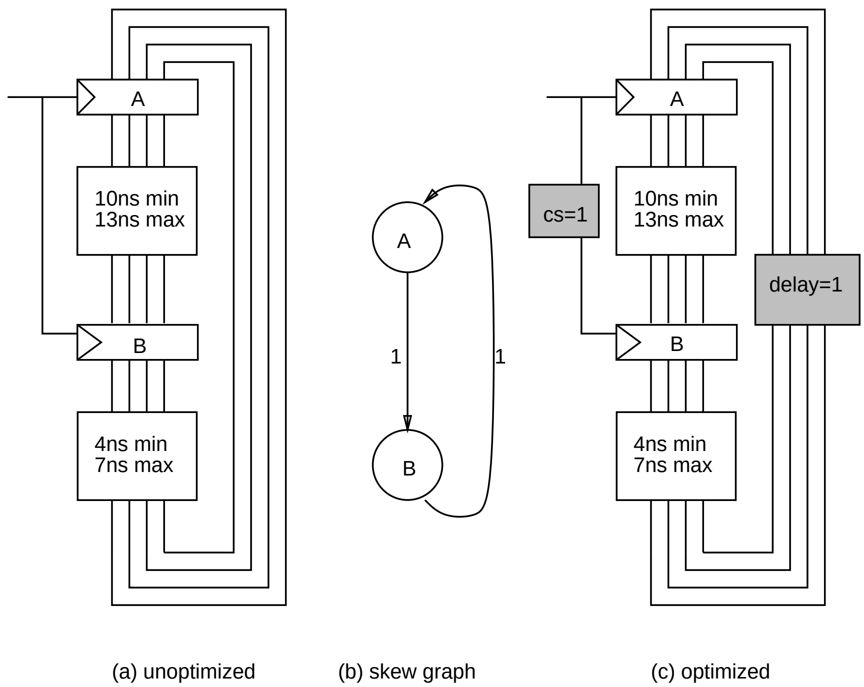


Figure 32: Monoharmonic Wave Pipeline with Feedback Optimization

3.2.4 Polyharmonic Wave Pipelines with Feedback

Feedback in a polyharmonic wave pipeline does not affect optimization based upon intentional path delay insertion. It does, however, complicate optimization using constructive clock skew. Like the monoharmonic case, this is because a skew introduced between the input and output synchronizer of any sequential circuit changes the timing relation between the input synchronizer and output synchronizer and also changes the relation between the output synchronizer and the subsequent synchronizer along any path.

For optimization using intentional propagation delay and constructive clock skew the following procedure is used for polyharmonic wave pipelines with feedback. First, the variation constrained clock period is determined:

$$T_{clk}^{opt} = max(P_{max}^{i} + H_{max}^{i} - P_{min}^{i} + H_{min}^{i}) \tag{93}$$

Next, the intentional path delay necessary to balance the maximum effective propagation delay of each path is found. If maximum propagation delay from the input synchronizer to output synchronizer, denoted by P_{max}^{max}, is in the set of paths which can support N^i waves, a delay of $X_{ij}T_{clk}^{opt} + \epsilon P_{max}^{j}$ is introduced in the constraint inequalities for each path with number of waves N^j, $j \neq i$. As in the polyharmonic wave pipeline without feedback, the term $X_{ij}T_{clk}^{opt}$ is the maximum integral number of clock periods by which the longest path through the wave pipeline exceeds the path under examination.

The difference in the number of harmonics between paths i and j, X_{ij}, is an integer bounded by:

$$\frac{P_{max}^{i} - P_{max}^{j} + H_{max}^{i} - H_{max}^{j} - \epsilon P_{max}^{j}}{T_{clk}^{opt}} - 1 \leq X_{ij} \tag{94}$$

and,

$$X_{ij} \leq \frac{P_{max}^{i} - P_{max}^{j} + H_{max}^{i} - H_{max}^{j} - \epsilon P_{max}^{j}}{T_{clk}^{opt}} \tag{95}$$

The intentional delay added to each path, ϵP^{j}, is such that the maximum effective delay of each path is equal.

$$P_{max}^{i} + H_{max}^{i} = P_{max}^{j} + H_{max}^{j} + X_{ij}T_{clk}^{opt} + \epsilon P^{j} \tag{96}$$

At this point, all paths through the polyharmonic wave pipeline have the same effective maximum delay, and the procedure for optimizing monoharmonic wave pipelined systems can be used to determine the constructive clock skew and intentional path delay necessary to achieve the optimal clock period.

Figure 33 demonstrates optimization of an example polyharmonic wave pipeline with feedback. In Figure 33(a), the unoptimized circuit the minimum clock period is 7ns, significantly greater than the optimal value of 3ns. Figure 33(b) shows the delay insertion step to equalize the polyharmonic wave pipeline. Figure 33(c) shows the optimized circuit which through intentional delay insertion and constructive clock skew is able to achieve the optimal clock period of 3ns.

As in the previous case, this method may increase latency to improve the system clock rate.

In synchronizing the wave pipelines any variation in the intentional delays added to the wave pipelines and any variation in the constructive clock skew methods will impact the cycle time of the system. Thus, the achievable clock period is increased by these variations.

3.3 Process and Environmental Delay Compensation

As shown in Sections 2.3.3 and 2.3.4 run-to-run process variation can result in a maximum to minimum delay of 1.35x, temperature variation can result in a maximum to minimum delay of 1.3x, and supply variation 1.2x. Coupled noise can result in a maximum to minimum of

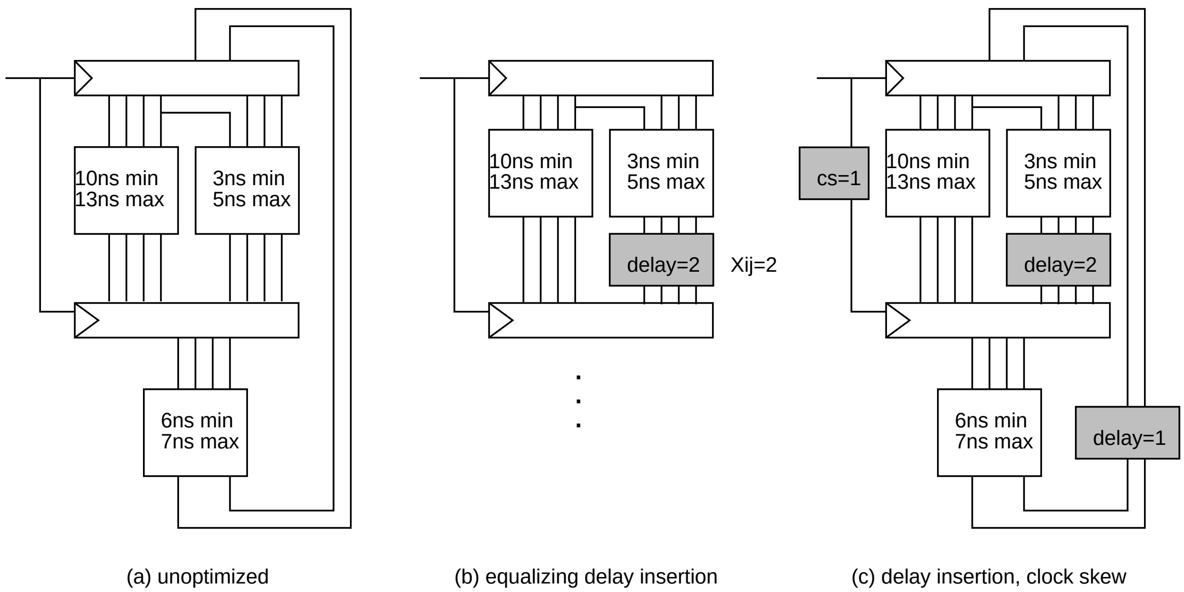


Figure 33: Polyharmonic Wave Pipeline with Feedback Optimization

as high as 1.13x if the output buffer delay is 25% of the path delay, the buffer interconnect capacitance equals the buffer load gate capacitance, and the mutual capacitance between wires is 50% of the interconnection capacitance.

Unless these variations are controlled or their effects compensated the maximum number of waves, or speedup, for a perfectly balanced wave pipeline is 1.7. This section describes methods by which the delay variations can be controlled or their effects compensated.

3.3.1 Sorting

One method of compensating for the variation in delay due to process variation is sorting. Unlike traditional synchronous circuit sorting where a bin sorted IC will run at any clock frequency below the bin upper limit, subject to the limits of any dynamic logic in the design, wave pipeline sorting is range sorting. Due to the two-sided limit on the valid clock period of a wave pipeline, sorting for wave pipelines involves determination of the valid range of clock periods given the particular device fabrication process of each VLSI device. By range sorting, the effects of process variation between dice can be minimized. Cross die spatial variations in process parameters, however, must be accounted for in the determination of the clock period.

Due to the relatively narrow range of valid clock periods for aggressive wave pipelines, range sorting is not a desirable method of accounting for delay variation due to manufacturing process.

Table 5 demonstrates the difficulties in sorting for wave pipelines. For three wave pipelines operated up to their maximum clock period the valid range of clock period for the fastest and slowest expected processes are given. The pipelines are assumed to have the maximum delay through the pipeline is much greater than the clock overheads H_{max} and H_{min}. Inclusion of the clock overheads narrows the valid ranges even more. The three wave pipelines have (a) no path length variation, (b) maximum path/minimum path = 1.1, and (c) maximum path/minimum path = 1.2. All pipelines are assumed to have environmental variation due to spatial variation in process parameters, temperature variation, voltage variation, and noise such that the process and environment variation factor, β, is 1.3. The minimum propagation delay of the wave pipeline is 20ns. In addition, on the slowest die the propagation delays of each path are 1.3-times as slow as on the fastest die, i.e. $\beta_{proc} = 1.3$. This table shows the valid operating ranges of the fastest and slowest expected dice off the line.

α	**Waves**	**valid clk interval as % of avg clk period**	**fastest die oper range**	**slowest die oper range**
1	2	42.4%	50 to 76.9 MHz	38.5 to 59.2 MHz
	3	14.3%	100 to 115.4 MHz	76.9 to 88.8 MHz
	4	2.5%	150 to 153.8 MHz	115.4 to 118.3 MHz
1.1	2	33.2%	50 to 69.9 MHz	38.5 to 53.8 MHz
	3	4.8%	100 to 104.9 MHz	76.9 to 80.7 MHz
1.2	2	24.7%	50 to 64.1 MHz	38.5 to 49.3 MHz

Table 5: Sorting Example

For each wave pipeline, as the number of waves supported is increased, the valid operating range of the pipeline is diminished. In turn, the number of divisions into which the dice must be sorted increases. For instance, if the first wave pipeline supports two waves, the operating frequency range of 50-59.2MHz is valid for all dice and no sort is necessary. If this pipeline is operated with 3 waves, the operating ranges for the fastest and slowest dice do not overlap. For this example, sorting into at least three frequency ranges would be necessary to capture the operating ranges of all dice. If this pipeline is operated with 4 waves, sorting into at least seven frequency ranges would be necessary to capture the operating ranges of all dice.

The minimum number of ranges into which wave pipelined dice must be sorted is approximately:

$$Bins \geq \frac{\frac{N}{\alpha\beta P_{min}} - \frac{N-1}{\beta_{proc} P_{min}}}{2 * (\frac{N}{\alpha\beta\beta_{proc} P_{min}} - \frac{N-1}{\beta_{proc} P_{min}}) - CLK_{interv}} \tag{97}$$

The term β_{proc} is the process degradation factor for the slowest fabricated die. The term CLK_{interv} is the width of operating clock period of each bin. This width cannot exceed the minimum valid clock frequency range of the wave pipeline. For example, if the above pipeline is operated with 4 waves, the minimum valid clock frequency range is 2.9 MHz; thus, each bin can have at most a 2.9MHz width. Assuming that each bin must have a width of 2MHz, the center frequency of the ten bins are: 117.3, 121.1, 124.9, 128.7, 132.5, 136.3, 140.1, 143.9, 147.7, and 151.5MHz.

The narrowing of the valid clock interval as wave pipeline performance is increased and the resulting increase in the number of sorting ranges makes frequency sorting impractical for high performance wave pipeline ICs.

3.3.2 Tunable Constructive Clock Skew

A second method of compensating for static delay variation in wave pipelined circuits is through the use of tunable clock skew between the input and output synchronizer clocks of wave pipeline. Fan, et. al. [14] used a laboratory tunable skew between the input and output clocks of their wave pipelined adder to counteract static process induced variation.

The constructive skew necessary to account for the static variation in delay due to process variation for wave pipeline i is:

$$cs^i = (\beta_{proc} - 1)P^i_{max} \tag{98}$$

where β_{proc} is the ratio of the delay of the critical path on a particular die to the delay of the same path fabricated with the fastest expected process.

This method, while appropriate for laboratory experiments is not practical for systems with several wave pipelines, as the clock skew mechanism for each wave pipeline must be externally accessible and controllable. In addition, wave pipelines with feedback present additional problems due to the interrelation of clocks.

3.3.3 Biased Logic

A method for compensation of static delay variation employed by Fan, et. al. [14] is biased logic. In this compensation method, pseudo-NMOS gates are used for all logic in the wave pipeline. The gate of the PMOS load transistor is driven by a bias voltage which is set so as to counteract the variation in delay due to process variations. As a result of the biased PMOS transistor, this method has the circuit problems associated with NMOS circuits: the need to ratio the sizes of the NMOS and PMOS transistors so as to be able to drive the output of the gate sufficiently low, a reduction in noise margins, and static power consumption. In addition, the routing of the bias voltage may increase VLSI area.

An alternative to the biased pseudo-NMOS logic is biased CMOS logic in which series transistors are added to the pull-up and pull-down transistor networks for each gate. The gates of the series transistors in the pull-up network and pull-down network are driven by separate bias voltages. The bias voltages are set so as to compensate for the static delay variations due to process. This method does not suffer from the ratioing, noise margin, and power consumption problems of the biased pseudo-NMOS method, but the additional transistors significantly increase the area of the logic gates and degrade the nominal speed of the gates.

Biased pseudo-NMOS and biased CMOS NAND gates are shown in Figure 34.

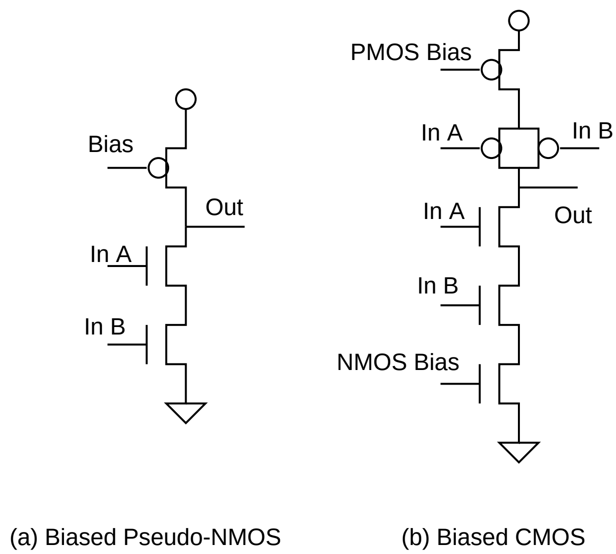


Figure 34: Biased Logic Gates

3.3.4 Driver Current Starving

A method related to the biased logic methods is driver current starving. This method relies on the tuning of the delay of a subset of the logic in the wave pipeline to counteract the process induced variation in the logic. A bias voltage is applied to current starving transistors in the output drivers of the wave pipeline. One of many examples of the use of current starving is the phase-locked clock generator circuit of Jeong, et. al. [24].

Figure 35 illustrates the use of current starved drivers.

Chapter 2 results indicate that the ratio of propagation delay between the slowest and fastest process can be at least 1.4. Because the current starving is used on a subset of the gates in the combinational logic, in order to compensate for this process-dependent

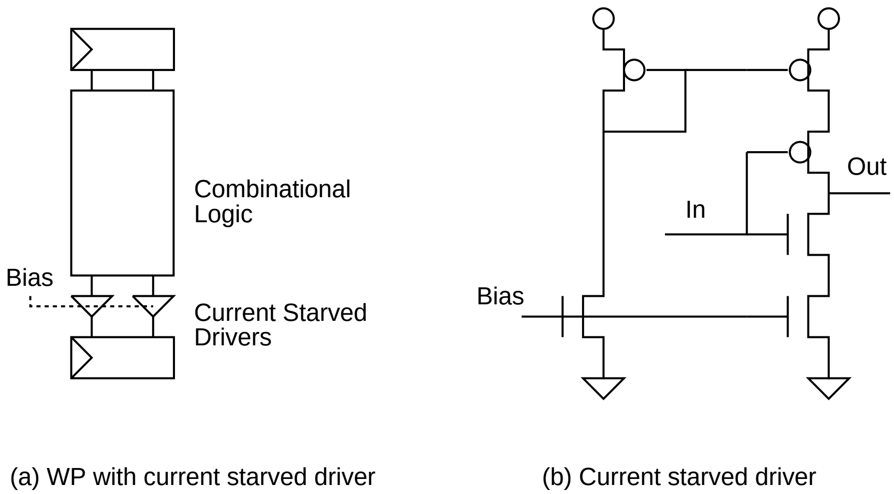


Figure 35: Compensation Using Current Starved Driver

delay variation, the current starved buffers must have a wide delay tuning range. Figure 36 illustrates a tuning range of 200ps to 1600ps. If a current starved buffer has a tuning range of P_{buf} to $r * P_{buf}$ in the fastest expected process, it can compensate for process variations for all circuits with a maximum propagation delay with the fastest expected process of:

$$P_{max} \leq \frac{r - \beta_{proc}}{\beta_{proc} - 1} P_{buf} \tag{99}$$

The buffer shown in Figure 36 with $\beta_{proc} = 1.4$ can compensate for process variation for wave pipelines with maximum propagation delay up to 3.3ns. For longer wave pipelines, current starved buffers can be cascaded.

Use of current starved buffers for compensation require the routing of a bias voltage signal and may increase critical path delay, particularly if multiple levels of buffers are required. The steep slope of the delay curve in Figure 36 indicates that this method is sensitive to noise on the bias voltage line.

3.3.5 Driver Voltage Controlled Load

A voltage controlled load can be used much as in current starving. This method changes the effective load capacitance through shunting transistors. The effective load is determined by the bias voltage applied to the gates of the shunt transistors. Like the current starving method, the range of delay modification is limited. Johnson and Hudson [25] used this method in a delay line phase-locked loop for synchronization of a cpu and coprocessor.

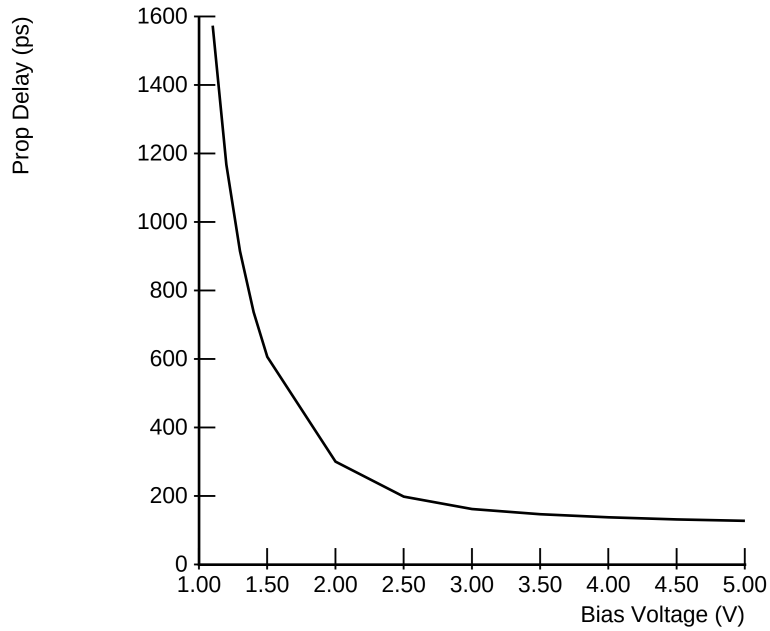


Figure 36: Delay Tuning Range of a Current Starved Driver

Figure 37 illustrates the use of a driver with a voltage controlled load. Figure 38 shows a tuning range of approximately 300 to 2000ps of a single stage of a driver with a voltage controlled load. Applying equation 99, a single level of buffers with a voltage controlled load can compensate for process for wave pipelines with a critical path of up to 4 ns.

As in the case of the current starved buffer, multiple buffers with a voltage controlled load can be cascaded. The voltage controlled load method requires the routing of a bias voltage, requires additional die area for the load transistors, is susceptible to bias noise, and may increase critical path delay.

3.3.6 Thermal Control

Intentional control over transistor temperature can be used to compensate for both static and dynamic delay variation. In this method, resistive elements are used to modify the die temperature so as to regulate the delay of the logic. Branson, et. al. [4] used this method of delay compensation in the timing vernier circuit for a CMOS circuit tester.

Figure 39 illustrates the use of a thermal control for delay compensation.

As shown in Section 2.3.4, elevating the temperature from 25 C to 100 C results in an

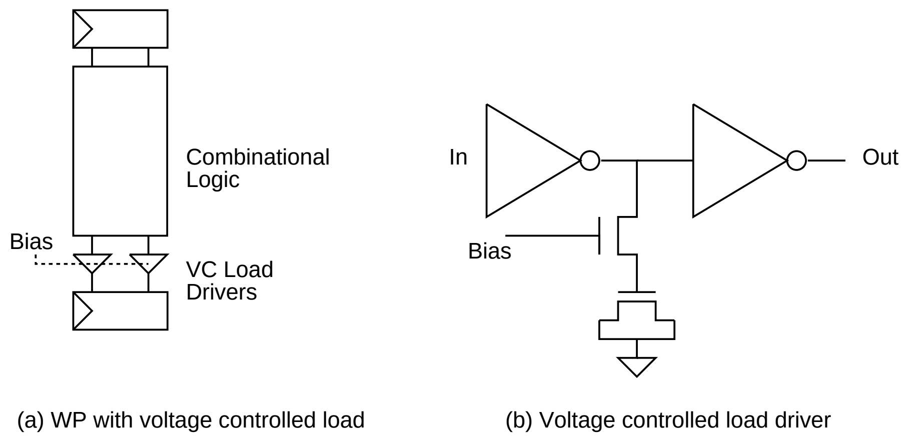


Figure 37: Driver with Voltage Controlled Load

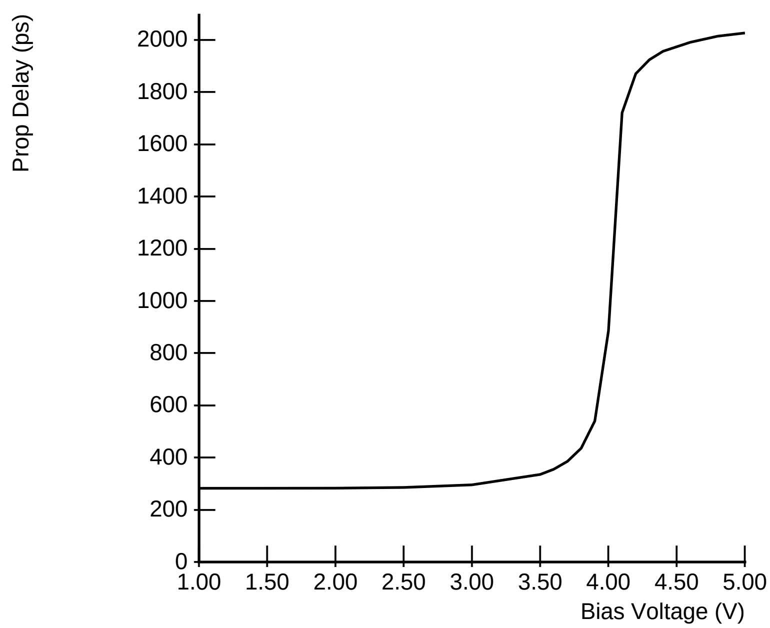


Figure 38: Delay Tuning Range of Voltage Controlled Load Driver

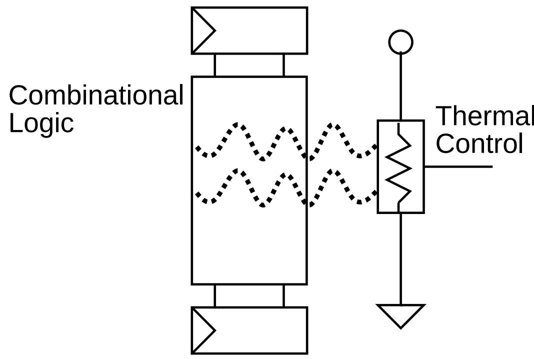


Figure 39: Thermal Controlled Delay Compensation

increase in propagation delays by a factor of 1.3. Thus, while this method can maintain a constant die temperature and thereby eliminate the variation in delay due to the temperature increases which would result from device switching, there is insufficient range of tuning to compensate for the variation in process parameters.

3.3.7 Supply Voltage Control

An attractive alternative to counteract the effects of process variation and temporal temperature variation is the adaptive supply voltage method. Figure 40 illustrates the use of power supply voltage control for delay compensation.

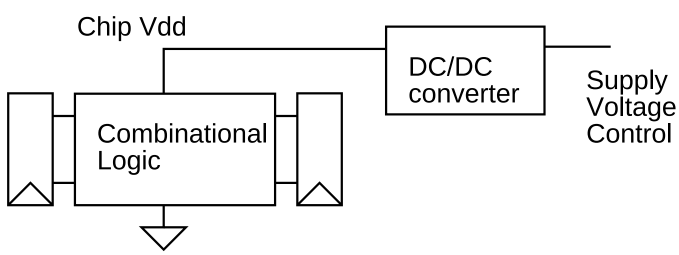


Figure 40: Power Supply Voltage Delay Compensation

To compensate for the propagation delay variation, the supply voltage is adjusted to a level which maintains the target delays. By applying equation 20, the power supply is set to:

$$V \approx \frac{V_0}{\beta} \tag{100}$$

Since β can be approximately two, the voltage supply may need to be set as low as half of the nominal supply level.

This method can be used to compensate for dynamic changes in delay, by adaptively adjusting the voltage supply of the wave pipelined logic so as to maintain circuit delays at their design targets. This adaptive method is capable of compensating for delay affecting changes for which the variation occurs with time constants greater than the closed-loop bandwidth of the adaptive circuit.

The adaptive circuit consists of a delay error detector and a supply voltage converter. Delay error detection is performed by phase comparing a signal with a voltage-controlled delayed version of the same signal. A fixed frequency source is applied to the input of an inverter chain whose delay has been set to half the source period when the chain is fabricated with the slowest anticipated fabrication. The chain input and output are phase compared. In the open loop control, if the phase difference exceeds a threshold, an external indication is toggled to indicate the device is running too fast and the power supply is lowered.

In a closed loop adaptive supply circuit, the phase error is used to charge or discharge a charge pump capacitor. In each adaptation cycle, a fixed amount of charge is added to or removed from the charge pump depending on whether the delay chain is longer or shorter than the design target.

The supply voltage converter consists of two parts, the delay chain supply, and the chip supply. A unity gain amplifier drives the charge pump voltage to the Vdd supply rail of the inverter chain. The supply rail thus is modified until the delay matches the design target. For small wave pipelined circuits, all circuitry can be driven by an on chip converter. For larger circuits the output of the error detector circuit is used to drive an off-chip dc-to-dc converter. Appendix C shows simulations for a closed-loop adaptive supply for the demonstration chip. Figure 41 details the effectiveness of the power control method for compensation of process variation. For each process run from Section 2.3.3, the simulated delays are shown with the supply at Vdd=5V and at the voltage determined by the constant delay circuit. Without compensation the maximum to minimum delay variation is 1.35x. With the compensation that ratio is 1.04x.

Because of the area and power efficiency of this method and its range of delay variation compensation, this method was employed in the wave pipelined vector unit system developed in this research. It is further described in Section 4.1.7.

Several difficulties exist in the use of an adaptive supply. Logic which is designed to switch at set voltages or which rely on voltage references and logic in which transistor threshold drops are allowed may not operate properly with a lowered power supply level. In addition, adaptive modification of power supply levels may increase the probability of CMOS latch-up and may result in static power consumption at interfaces with circuitry driven with nominal voltage power supplies.

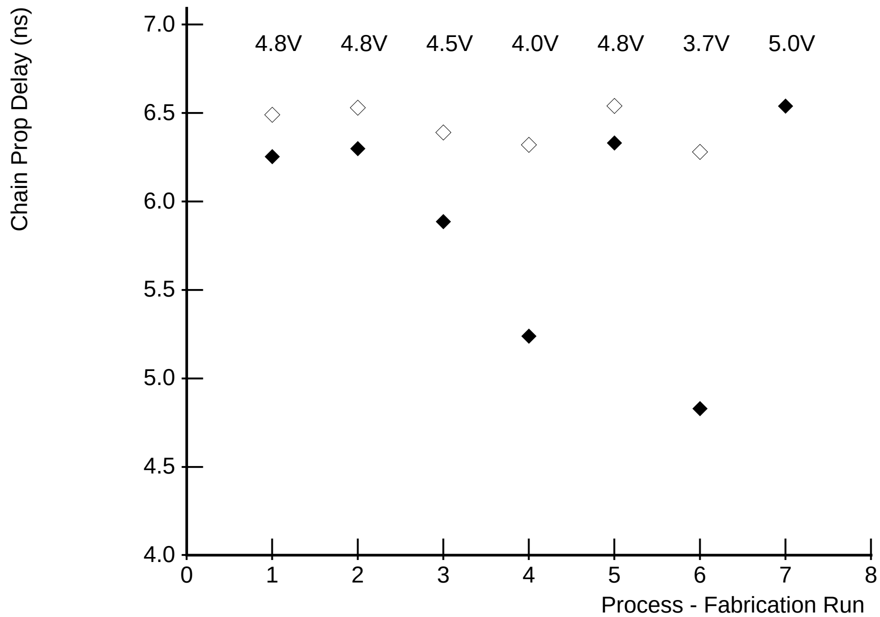


Figure 41: Power Supply Voltage Delay Compensation

3.4 Summary

This chapter detailed design techniques for the optimization of the performance of wave pipelines. A method of minimizing the path imbalance in the design of CMOS wave pipelined circuits based upon transistor sizing was developed. This method is an extension of the Wong method for balancing CML logic networks [53]. The optimization uses a topological model of the circuit and macromodels of gate delay based upon HSPICE simulations to generate a linear program representation of the transistor sizing problem. For several representative circuits this method of delay balancing has optimized the circuits such that the path delay and data dependent delay variation are limited to the 0% to 20% range. Klass [29] has shown that for similar circuits, the best manual balancing methods exhibit path delay and data dependent delay variation of up to 15% to 20%.

Further performance potential may be lost in wave pipelining due to the difficulty of operating all wave pipelines in a system at a common clock frequency. Circuit optimizations which use intentional delay insertion and/or constructive clock skew can be used to minimize the performance impact of synchronization of wave pipelines within a system.

The variations in delay due to process and operating environment impose severe limits on the performance of CMOS wave pipelined circuits. Several means of minimizing the variations or the performance effects of these variations were presented in this chapter. Frequency sorting was shown to be significantly more constrained for wave pipelines than for conventional pipelined circuits. For high performance wave pipelines sorting was demonstrated to be impractical. Delay compensation methods were evaluated. The tunable clock skew compensation method is impractical for a VLSI system in which multiple wave pipelines which are part of a synchronous system with feedback because of the need to individually skew each clock and because of the interrelation of clocks in a system with feedback. The biased logic methods suffer from area penalties and/or power consumption and noise margin problems. Methods based upon tunable delay buffers or thermal compensation suffer from limited range of delay adjustment. The adaptive power method provides sufficient range of delay adjustment, does not increase logic area, and can lower power consumption while compensating for fabrication process and temperature dependent delay variations.

The techniques for the optimization of the performance of wave pipelines allow systems of CMOS wave pipelines to be implemented in VLSI ICs. The following chapter describes the demonstration processor developed in this research.

4 CMOS WP System: VLSI Vector Unit IC

To demonstrate the techniques and tools developed as part of this research a CMOS wave pipelined VLSI integrated circuit was designed. The system implemented is a wave pipelined vector processor unit. This device demonstrates that wave pipelined CMOS VLSI systems can be designed to perform within the performance limits described in Chapter 2. This system integrates and synchronizes multiple heterogeneous wave pipelines. It includes both wave pipelined functional units and memories. It was designed with the assistance of automated CAD balancing tools. It contains adaptive power supply support for the maintenance of wave pipelined operation over a range of operating conditions and fabrication tolerances.

4.1 Vector Unit Architecture

The VLSI vector unit consists of a wave pipelined 8-bit x 8-bit unsigned multiplier, a wave pipelined 16-bit parallel adder, a wave pipelined vector register file, a scoreboard to ensure proper operation, a store buffer to interface with an external memory bus, instruction and input data buffers, and decode and issue logic. Test and power supply control support logic are also included. Figure 42 shows the organization of the VLSI vector unit. Wave pipelined logic is shaded in this figure.

4.1.1 Parallel Adder

The adder is a modified version of a Brent and Kung parallel adder [5, 14]. It operates on 16-bit unsigned operands and produces a 16-bit unsigned result. Figure 43 is a block diagram of the adder circuit. In this figure, the shaded blocks, the $GenerateDelay$ and $Propagate/GenerateDelay$, are necessary to balance delays through the adder. The open blocks, the $Propagate$ and $Propagate/Generate$, are the Brent and Kung adder computational logic blocks.

Module selection and balancing for the adder were done using the wave pipelining design tools. Wave pipelining of the adder resulted in a 15% increase in the area of the adder. The $GenerateDelay$ and $Propagate/GenerateDelay$ gates added an additional 12% to the area of the adder. Delay balancing through CMOS transistor sizing resulted in an additional 3% area penalty.

The maximum propagation delay through the adder is 5.5ns. The path length variation is 0.6ns.

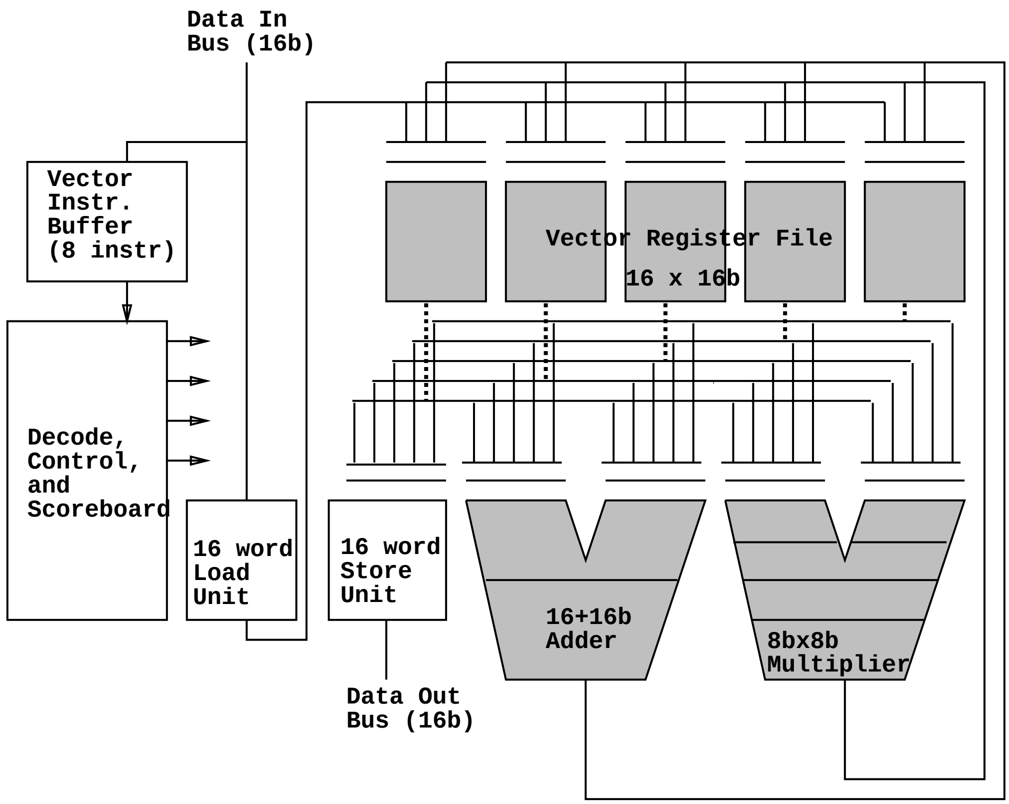


Figure 42: Vector Unit Organization

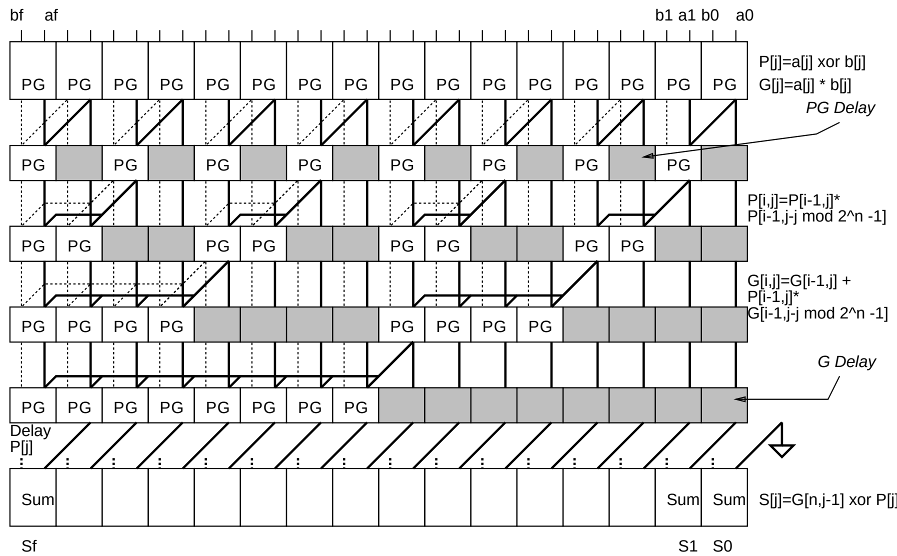


Figure 43: Parallel Adder Organization

4.1.2 Parallel Multiplier

The parallel multiplier circuit operates on 8-bit unsigned operands and produces a 16-bit unsigned product. The parallel multiplier consists of three logic blocks: the partial product generators, the partial product reduction logic, and a final parallel adder. Figure 44 details the organization of the partial product generation and reduction logic. The final parallel adder is equivalent to the parallel adder circuit used in the add functional unit. The organization of the final parallel adder is shown in Figure 43.

Each partial product is formed through the logical ANDing of a multiplier bit and a multiplicand bit. Eight rows of eight partial product generators are shown in Figure 43. Booth recoding of the partial products is not employed in this multiplier.

The partial product reduction logic transforms the array of partial product bits into a redundant binary form of the product. The redundant binary form of the product is converted to the simple binary result by the final parallel adder. The partial product reduction logic consists of two levels of (4,2) counters. In the first level, one set of eleven counters compresses from one to four bits of partial product in each of the eleven columns of binary precedence formed by the multiplication of multiplicand by the least significant four bits of the multiplier while another set of eleven counters reduce the partial product bits formed by the multiplication of the multiplicand by the most significant four bits of the multiplier.

The outputs of the two sets of counters are reduced in the second level of the reduction logic by a set of sixteen counters.

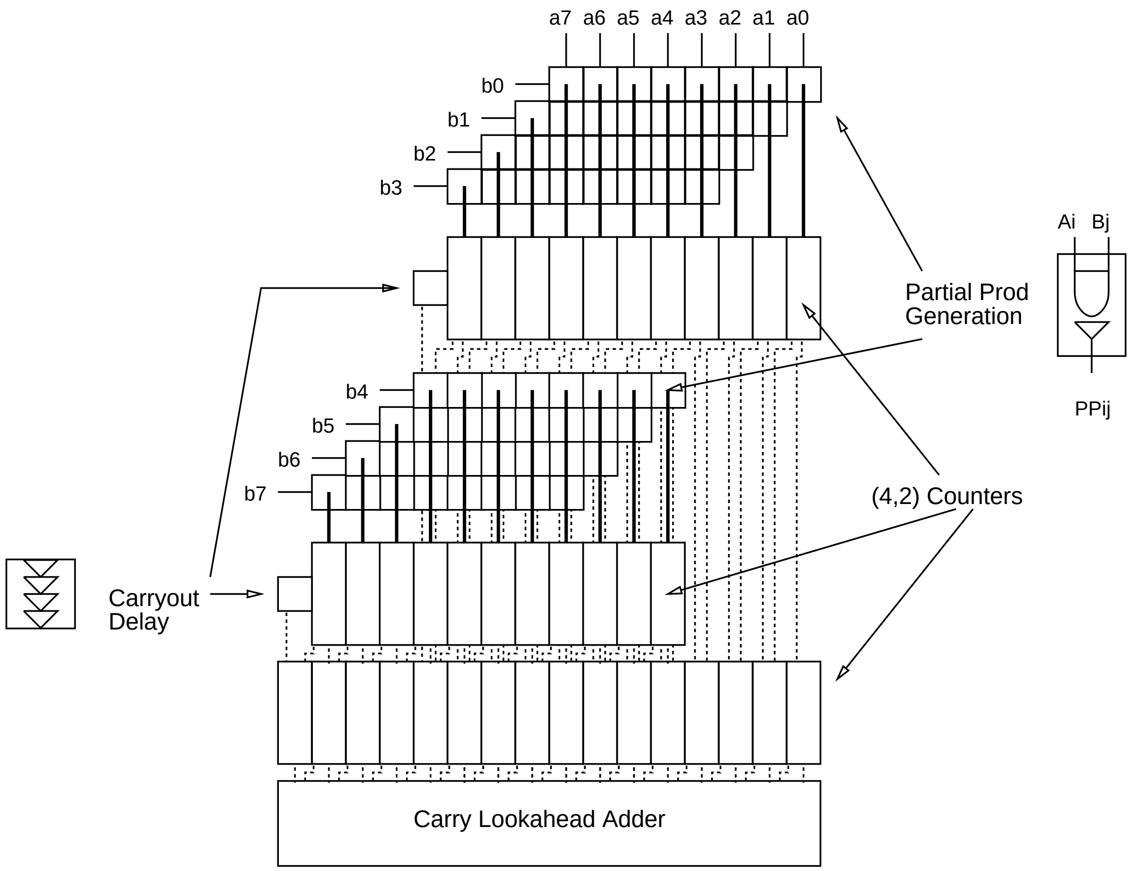


Figure 44: Parallel Multiplier Organization

Figure 45 is a schematic of the (4,2) counter used in this design. This circuit counts the number of the inputs A, B, C, D, and Cin which are asserted. This count is output in a redundant binary form in the sum output, which gives the count in the same precedence column as the input bits, and two carry outputs, which represent the portion of the count in the next higher precedence column.

The shaded inverters in Figure 45 are delay elements inserted by the rough-tuning pass of the delay balancing tool.

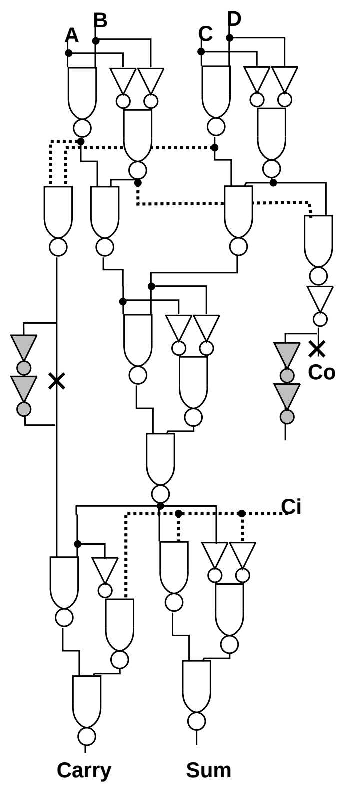


Figure 45: (4,2) Counter Implementation

Delay balancing of the multiplier for wave pipelining resulted in a 10% increase in the area of the multiplier. The delay buffers within the counter circuits and the final carry out delay shown in Figure 44 accounted for an 8% increase in area. The fine balancing transistor sizing accounted for 2% of the additional area.

The maximum propagation delay through the multiplier is 10.8ns and the path length variation is 1.37 ns.

4.1.3 Vector Register File

The vector register file contains five vector registers. Each vector register consists of sixteen 16-bit elements. Registers have one read port and one write port. Element address generation is done locally. Delayed clock signals are used for bitline equalization, bitline pullup, and sense enabling. Only stride-one addressing is supported. The register cells are cross-coupled inverters with pairs of read and write NMOS pass transistors. Clocked bitline equalization and pull-up are employed. A cross-coupled NMOS sense amp design was used. Figure 46 is a diagram of the a vector register, 46(a) shows the address generation logic, 46(b) shows the cell structure, 46(c) shows the sense amp.

Because the registers are not static CMOS structures, balancing of the registers was manually performed. A method for minimizing delay variation across the memory array was developed [44] and employed in the design of the register file.

The read access times of register cells which drive bitlines which are physically distant from the wordline drivers are longer than for cells which drive bitlines which are close to the wordline drivers. This is due to the RC delay of the wordline wire separating the cells. Similarly, the read access times of register cells which are enabled by the wordlines which are physically distant from the sense amplifiers are longer than for cells which are addressed by wordlines which are close to the sense amplifiers due to differences in bitline RC delays.

Within each register, the wordline buffers were sized so as to counteract the variation due to the wordline proximity to the sense amplifiers. The read data buffers were also sized so as to counteract the variation due to the bitline proximity to the wordline driver. This is illustrated in Figure 47. Between vector registers, delay variation of access is minimized through clock skew minimization, since read access of the registers is relative to the same clock. The balancing of the vector registers increased their area by less that 2%.

The maximum read access time of the vector registers is 3.7ns. The minimum simulated cycle time of the vector registers is 2.0ns.

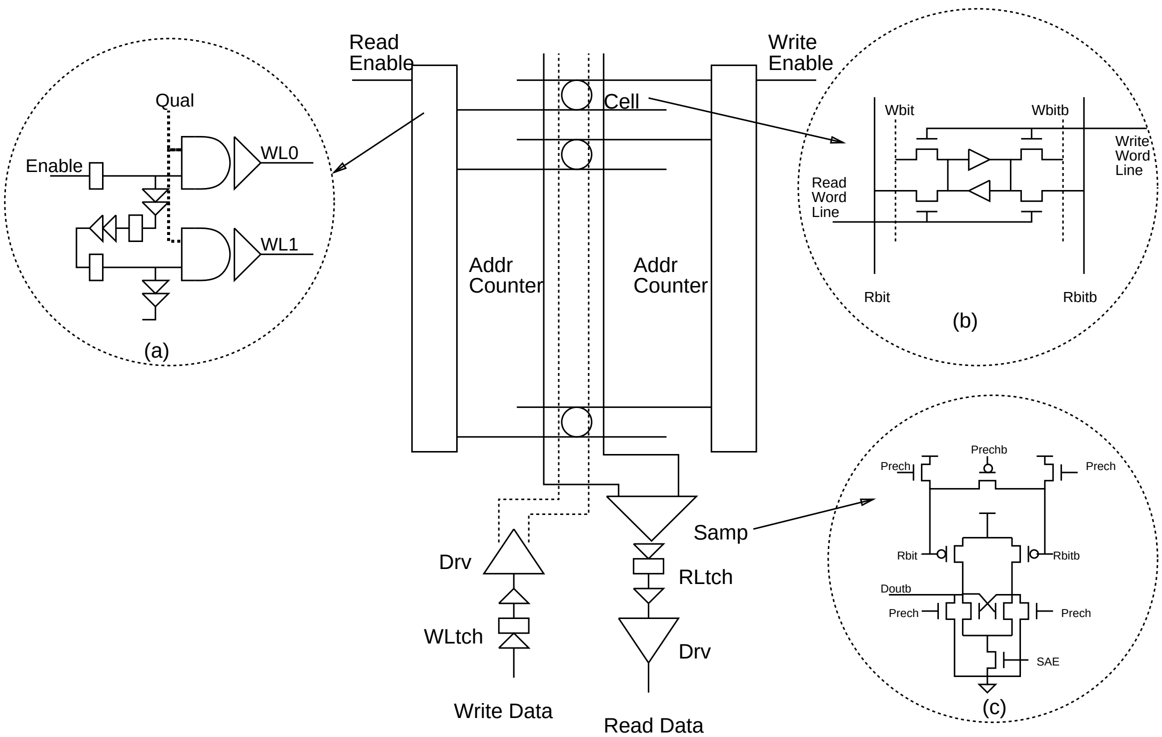


Figure 46: Vector Register Organization

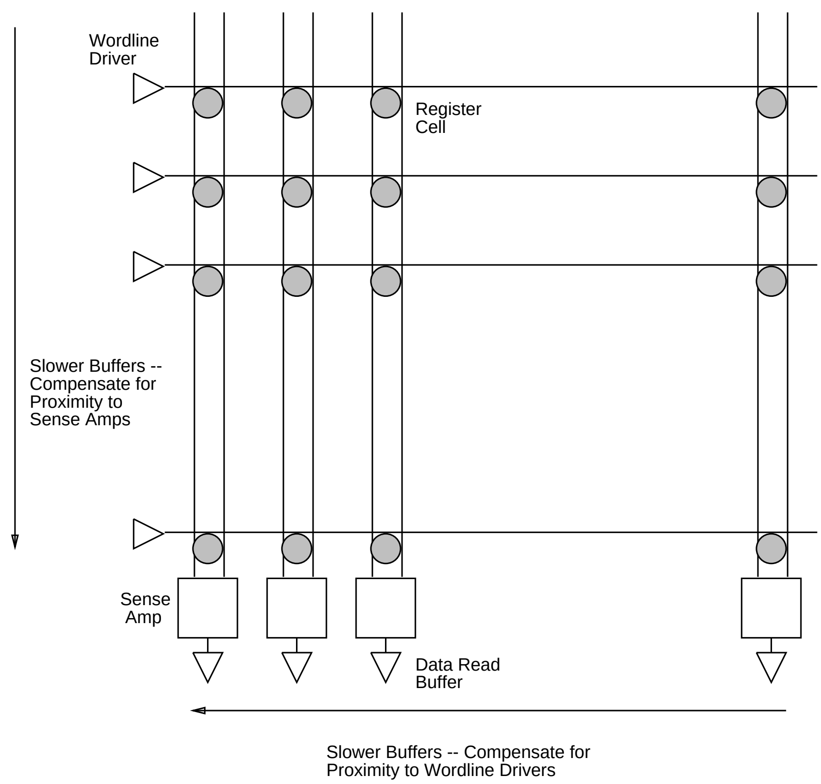


Figure 47: Vector Register Balancing

4.1.4 Load Unit, Store Unit, Instruction Buffer

The Load Unit consists of a 16 element deep fifo and fifo control logic. Each element is 16 bits wide. Entries are loaded into the fifo from the input data bus under external control at a reduced rate. The fifo is emptied through the execution of a vector load instruction.

The Store Unit contains a 16 element deep fifo and fifo control logic. Each fifo entry is 16 bits wide. Entries are loaded into the fifo through the execution of a vector store instruction. The fifo is emptied under external control at a reduced rate. The store fifo output data is placed on the output data bus.

The Instruction Buffer consists of an eight-deep queue of vector instructions. Instructions are loaded in to the buffer via the input data bus under external control at low speed. Instructions are removed from the queue by the fetch/decode logic and executed at chip core frequency.

4.1.5 Scoreboard

The vector unit scoreboard consists of timers for each functional unit, timers for each vector register for vector reads, and timers for each vector register for vector writes, and a scoreboard update state machine.

4.1.6 External Control Logic

External control of the instruction buffer, load and store units, and test circuitry was provided. External functions include:

Ibuffer Load	Low speed load of an instruction from the input data bus to the instruction buffer,
Load Unit Fifo Load	Low speed load of data from the input data bus to the load unit fifo,
Store Unit Fifo Empty	Low speed store of data from the store unit fifo to the output data bus,
Test Mode	Tristates register file data busses, enables test input and test output bus direct access to the functional units,
Run	Enables instruction fetch and execution.

External operation signals were synchronized and executed at the chip core frequency.

4.1.7 Constant Delay Power Control Logic

The constant delay adaptive power logic provides external power supply modification indications which are used to compensate for process and temporal thermal variation as described in Chapter 3. The constant delay logic consists of a delay chain of inverters which were balanced at design time to be 16.6ns at the expected slowest process operating under worst case environmental conditions. This delay chain is driven from a chip input. The input and output of the delay chain are phase compared. The phase comparator output drives a D-latch whose outputs are used to drive the power bump indications. The D-latch was designed so as to have an intentional race which acts as a threshold of phase difference which results in an external power bump indication.

4.1.8 Clock Generation and Distribution

A two phase clocking strategy is used for the vector unit. The system clock is driven from a terminated input pin. This clock signal is used to generate complimentary clock signals which are distributed to the vector unit logic which is not wave pipelined via an H-tree distribution network.

4.2 Vector Unit Operations

The instruction set of the vector unit consists of Vector Load, Vector Store, Vector Add, and Vector Multiply. Arithmetic operations occur between vector registers exclusively. Load and Store operations transfer vectors from the load unit fifo to a vector register and from a vector register to the store unit fifo, respectively. The pipeline stages for each instruction and the time spent in each functional block are shown in Figure 48.

For the vector add and vector multiply, arbitrary source and destination registers are allowed: Both sources may be the same register. The destination register can be a source register. Separate read and write ports on the registers and full connectivity between the register file and the ports on the functional units allow this flexibility.

Vector instructions are issued as soon as the *element* in the vector register is valid and the *port* in the register file is free. This allows the vector unit to chain operations between functional units with a single cycle between generation of a vector element and its subsequent use. This flexibility necessitated more complexity in the scoreboard, but provides greater throughput.

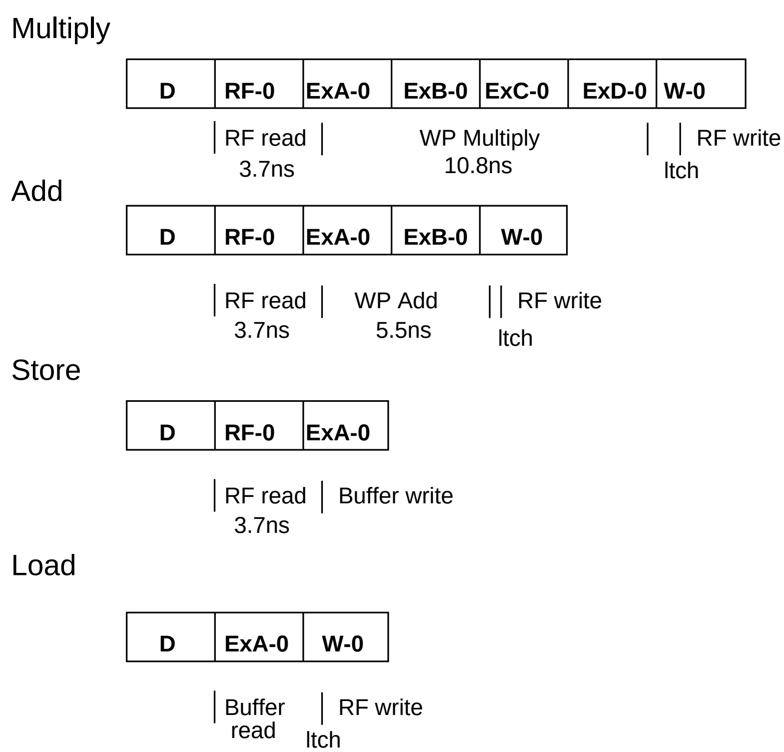


Figure 48: Vector Instruction Pipeline Stages

4.3 Balancing

This section summarizes the balancing results of the wave pipelined logic used the vector unit. Table 6 details the wave pipelined overhead in number of transistors, and functional unit area for the adder and multiplier circuits. The execution time of the balancing procedure on a Sun SPARCstation 10 are also given.

	Multiplier	**Adder**
Maximum Delay	10.84ns	5.52ns
Delay Variation	1.37ns	0.58ns
Unbalanced Transistors	5526	1322
Balanced Transistors	6466	1730
Unbalanced Area	1.95 sq mm	0.40 sq mm
Balanced Area	2.17 sq mm	0.45 sq mm
Balancing Exec. Time (Sun SS10)	10.94 hr	0.62 hr

Table 6: Vector Unit Balancing Results

4.4 Vector Unit Fabrication

The vector unit was fabricated using the Hewlett-Packard HP CMOS26b 0.8 micron CMOS process through the MOSIS service. The process provides three levels of metal and a single poly layer. Two metal levels were used for signal distribution, and M3 was used exclusively for power and ground distribution bars. The feature resolution of this process is 0.1 micron which allows a fine delay resolution for the transistor sizing delay balancing procedure. The VLSI vector unit design contains approximately 47000 transistors and occupies an area of 43 sq mm. It is packaged in a 132-pin PGA.

A die photo of the VLSI vector unit is shown in Figure 49.

4.5 Test Results

Testing of the vector unit IC consisted of several low speed functional tests and several high speed tests.

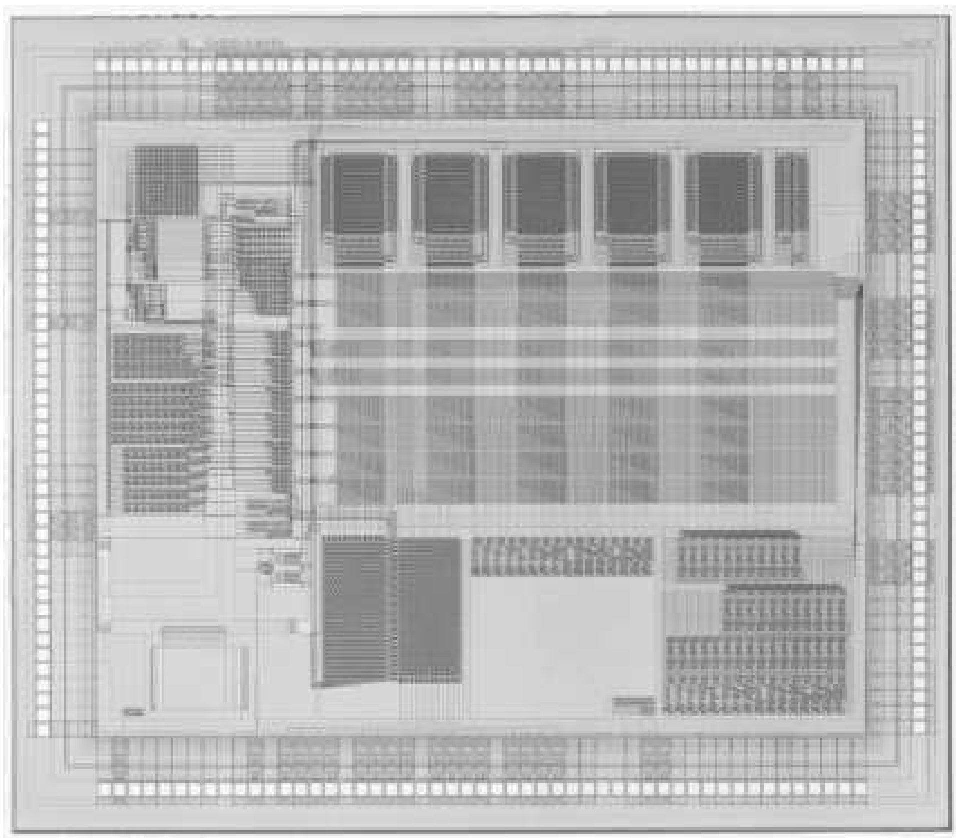


Figure 49: Vector Unit Die Photo

4.5.1 Functional Tests

Low speed functional testing consisted of tests of the on-chip ring oscillator, the test vector register, and a low-speed multiplier test.

Of thirteen vector unit ICs tested, all thirteen of the ring oscillators operated. The ring oscillator test was performed primarily as a basis for determining process variation between the dice. The oscillation frequencies ranged from 116 to 123 MHz with an arithmetic average of 119MHz at 5V power supply. The variation in ring oscillator frequency at 5V supply is indicated in Figure 50. Spice simulations predicted an oscillation frequency of 112.2 MHz for the design target process and 109 MHz for the models based upon MOSIS measurements from the run.

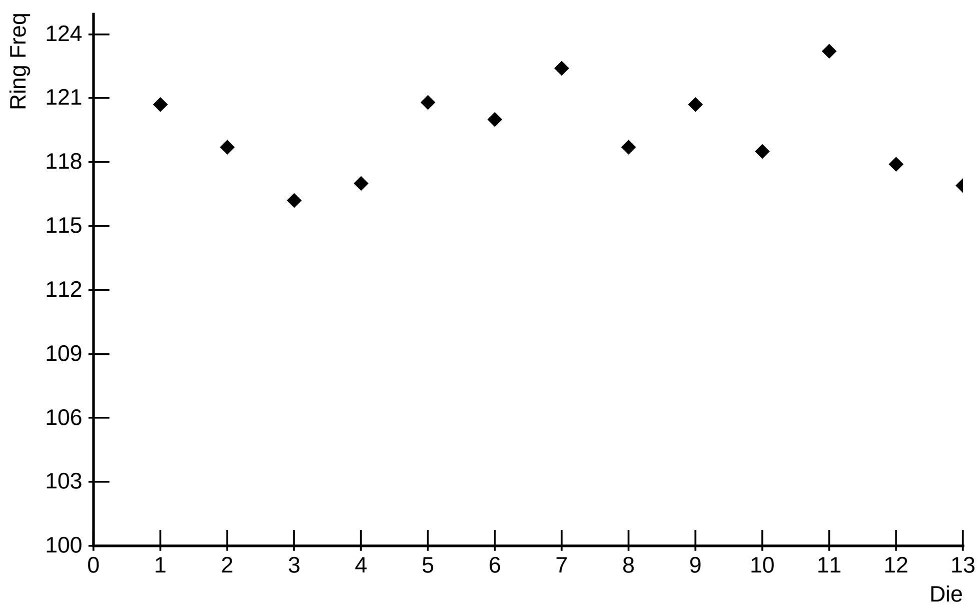


Figure 50: Dice Ring Oscillator Variation

Eleven of thirteen passed write and read verify tests of 100 random vectors at 50MHz. Vector register reads and writes at speeds up to 200MHz were performed on a single IC. A trace of a 200MHz read verify operation is shown in Figure 51. Since this test was performed from the pins, higher frequency testing exceeded the switching speed of the output pad driver design for the load imposed by the test equipment.

The functional multiplier test consisted of the application of multiplier input vectors to test inputs and product checking at product testing output pins. Ten thousand pseudorandom vectors were applied and results checked at a 10MHz rate. Eleven of the 13 ICs passed this

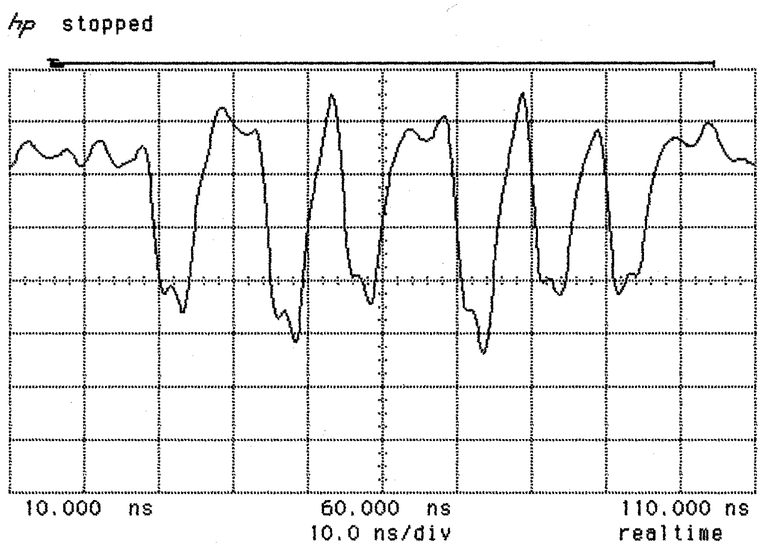


Figure 51: Vector Register Read Operation

test.

4.5.2 Wave Pipeline Speed Tests

To enable proper high-speed wave pipelined operation of each die, the constant delay power supply was adjusted so as to track the design target operating frequency. When an power bump indication was received from the die being tested, the power supply was adjusted by 0.1V. Figure 52 shows the output of the *power bump down* indication from the die # 13 at Vdd=5.0V in the top traces and at the constant delay supply of Vdd=4.8V in the bottom traces.

To indicate the effectiveness of the constant delay power method, the design target frequency of the ring oscillator was compared to the measured oscillator frequency of each die with the power supply voltage set to the value determined by the constant delay circuit. At design time, simulations were used to determine the target ring oscillation frequency of 112MHz.

Figure 53 shows the supply voltages specified by the constant delay indications for each die. For each die, the oscillator frequency at Vdd=5V is indicated with a solid diamond. These points show the delay variation of the ring oscillator circuit across the dice. The frequencies at a five volt supply were as high as 18% higher than the target frequency.

When the supply voltage for each die was set to the value indicated by the constant delay circuit, the frequencies specified by the open diamonds were measured. With the adaptive

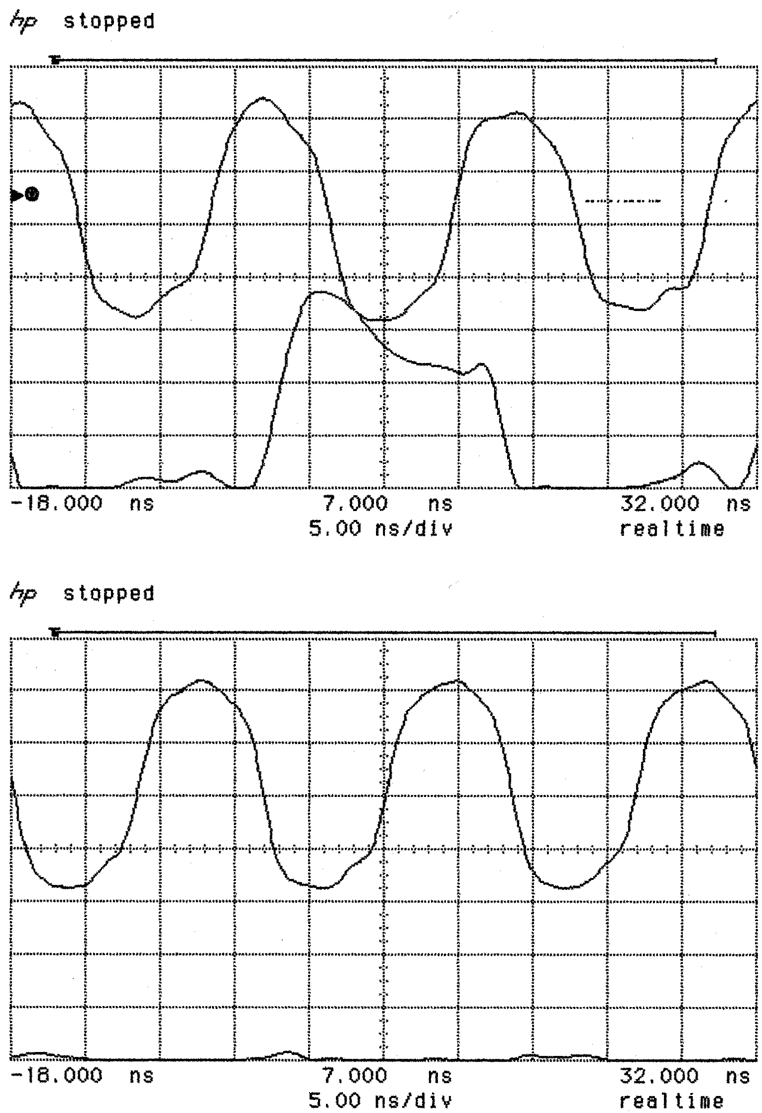


Figure 52: Constant Delay Power Bump Indications

power method the frequencies were measured at 0% to +5% faster than the target frequency.

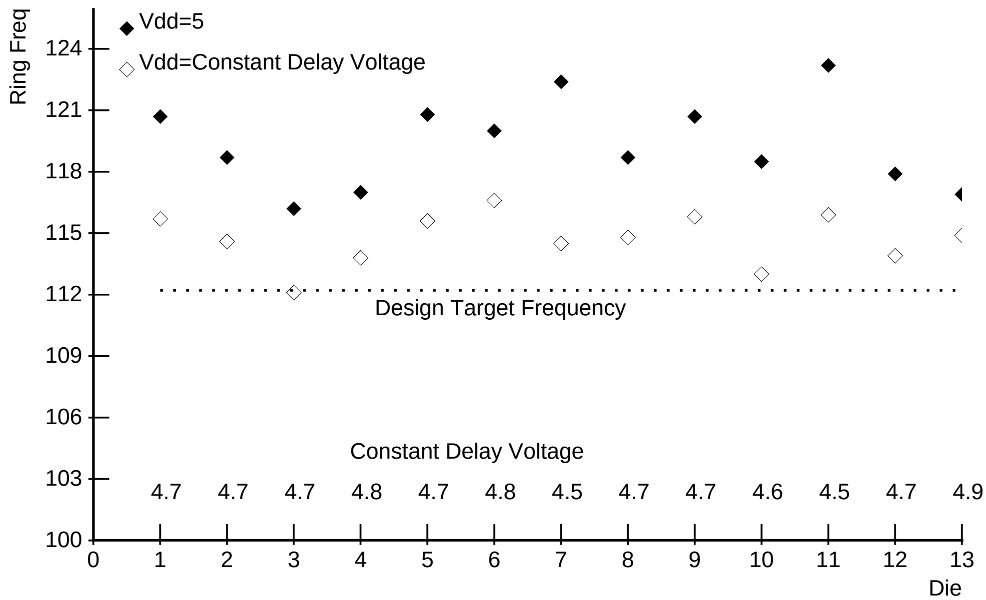


Figure 53: Die Process Variation Compensation

Once the lock voltage was determined, wave pipelined speed testing was performed by loading data and instructions into the instruction buffer and load unit fifos at low speed, performing the vector instructions at high-speed, and finally emptying store unit fifos at low speed and verifying the results. This procedure is shown in detail in Figure 54.

Using this procedure the eleven functional ICs were tested. Three of the eleven correctly performed a 16000 vector add test and a 16000 vector multiply test at 303MHz.

Eight of the eleven correctly performed the 16000 vector add test and a 16000 vector multiply test at 222MHz. At this speed, the multiply latency was decreased to four as fewer waves were held within the multiply wave pipeline.

4.6 Comparison to Traditional Design

With the performance detailed in the previous section, the wave pipelines demonstrated approximately 1.1 waves of data in the vector register file, 1.9 waves of data in the adder, and 3.7 waves of data in the multiplier.

To quantitatively determine the performance benefits of wave pipelining in this design, a vector unit was designed using traditional pipelining and the simulated performance was

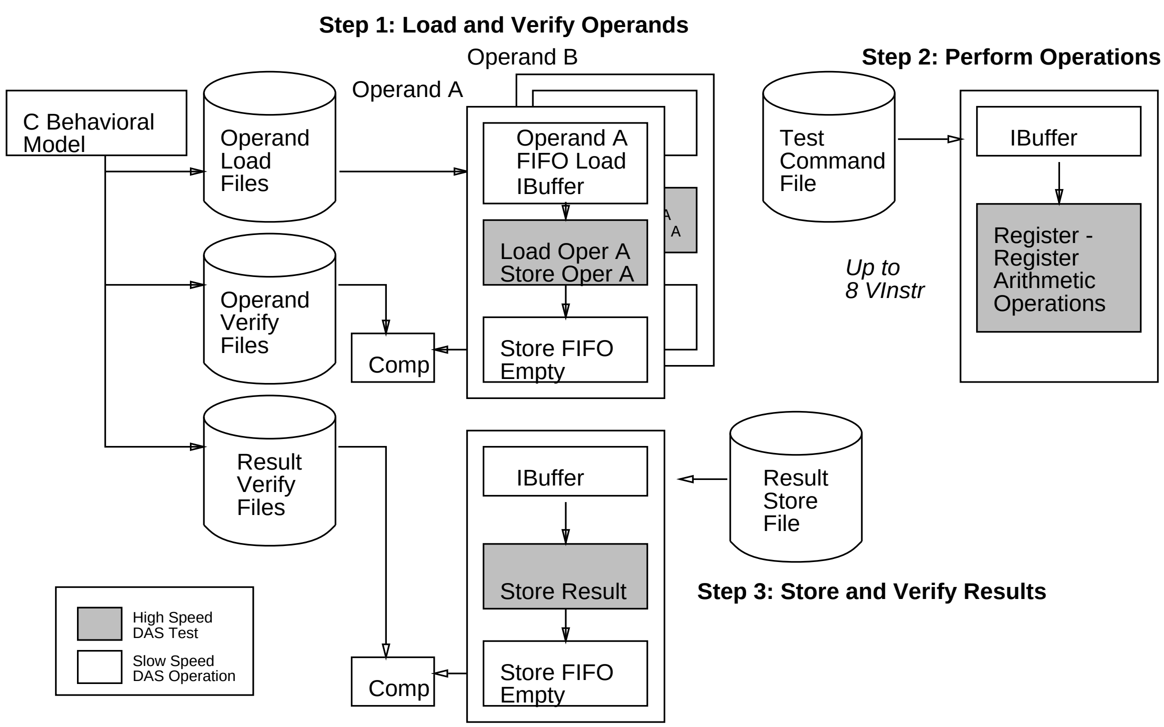


Figure 54: High Speed Wave Pipeline Testing

compared.

The traditional pipeline used two phase clocking and dynamic flow latches to achieve high performance. The adder and multiplier were redesigned using the latches. The multiplier and adder were implemented using the same cell library as the wave pipelined implementation, but delay padding elements and delay balancing transistor sizing were not used. The vector register file was operated as a single stage in the pipeline and thus was not partitioned into multiple pipeline stages.

Table 7 details simulated results for the wave pipelined vector unit and for a complete layout using latch-based pipelined units and a single cycle register access.

	Wave Pipe	**Latch Pipe**
Die Area	43.19 sq mm	44.09 sq mm
Min Clock Period	2.8ns	3.8ns
Mult Unit Latency	10.84ns	11.7ns
Add Unit Latency	5.5ns	6.1ns

Table 7: Vector Unit Results Comparison

The traditional pipeline, because of its higher cycle time, had a three cycle multiply execution rather than the four cycle execution in the wave pipelined design. The clock to the output latch of both the adder and multiplier were skewed in the traditional pipeline design. Without the use of this clock skew, the latencies would be 7.6ns and 15.2ns, respectively.

Because of the ability of the wave pipelined design to have more than a single register read operation occurring concurrently, the wave pipeline had a 35% faster cycle time. Because of the lack of intermediate latch delay and partitioning overhead, the wave pipeline had operation latencies up to 10% less than for the traditional pipeline design.

When compared to traditional pipelines designed with less aggressive clocking technologies, such as static latch or register-based pipelines, the wave pipeline performance would be even better.

4.7 Summary

This chapter has described the wave pipelined vector unit developed for the demonstration of the techniques and tools presented in this dissertation. This vector unit integrates and synchronizes multiple heterogeneous wave pipelines: Wave pipelining was employed in the design of the vector register file, the add functional unit, and the multiply functional unit.

This unit was fabricated in a 0.8 micron process and was tested at operating frequencies up to 303MHz. At 300MHz, the vector register file supported 1.1 concurrent waves of data, the adder supported 1.9 waves of data, and the multiplier supported 3.7 waves of data with no intervening latches.

An equivalent conventional pipelined design using aggressive two phase clocking and dynamic latches was developed and contrasted to the wave pipeline design. The vector unit design which used conventional pipeline clocking was approximately 2% larger. The simulated latencies through the multiplier and adder functional units using conventional pipelining were 8% and 11% longer, respectively. The maximum simulated clock rate achieved by using wave pipelining was 35% faster than that which could be achieved using conventional two phase clocking and dynamic latches.

5 Architecture and Circuit Enhancements

5.1 Architectural Enhancements

Enhancements to wave pipelines to support stalling, wave pipelining of synchronous circuits with latchless feedback, and use of self-timing techniques for wave pipelining are examined in this section.

5.1.1 Stalling in Wave Pipelined Circuits

A significant barrier to the use of the wave pipelining design technique has been the difficulty in stalling a wave pipeline. Once a wave has been launched, it precedes unimpeded through the combinational network. In the event of a stall condition, all waves in the wave pipeline propagate to the end of the pipeline, thereby overwriting previous waves. First mechanisms to ensure consistency of wave pipelined data following a stall are presented. Subsequently, a method of dynamically stalling a wave pipeline is detailed.

Wave Pipeline Circuit Model To improve throughput, a logic network can be partitioned into pipeline stages, each of which operates upon data computed in the previous cycle by the previous pipeline stage. When a logic network is pipelined, synchronizing elements, either latches or registers, are inserted to partition the network into stages. These synchronizing elements increase the network area, power, and latency.

Wave pipelining is a design style which allows overlapped execution of multiple operations without using synchronizing elements. Rather, a knowledge of the signal propagation delay through the network is used to ensure that operations do not interfere with their predecessor nor successor data values.

The effects of stalling in a wave pipeline are now examined. Figure 55 shows the propagation of waves down a wave pipeline with no stall. In Figure 56, a stall condition occurs at time 2, while wave 3 is in stage 2. Until the stall is released, no new inputs are applied to the pipeline. However, the waves already in the pipe continue to propagate. Once the stall is released, the pipeline must be restored to the condition prior to the stall.

In a traditional pipeline, the data at each stage is not allowed to propagate to the next stage during the stall. Thus, when the stall is released the data in the pipeline which succeed the stall inducing stage have not progressed.

To accomplish this in a wave pipeline the pipeline can be refilled so as to restore the placement of the waves in the wave pipeline. This restoration occurs in times k to k+3 in the figure. Wave pipeline restoration requires that enough input values are queued at the input of the pipeline to ensure that the pipeline can be restored. For an N stage wave pipeline, up to N-1 previous inputs must be reentered into the pipeline after a stall before additional computation can occur.

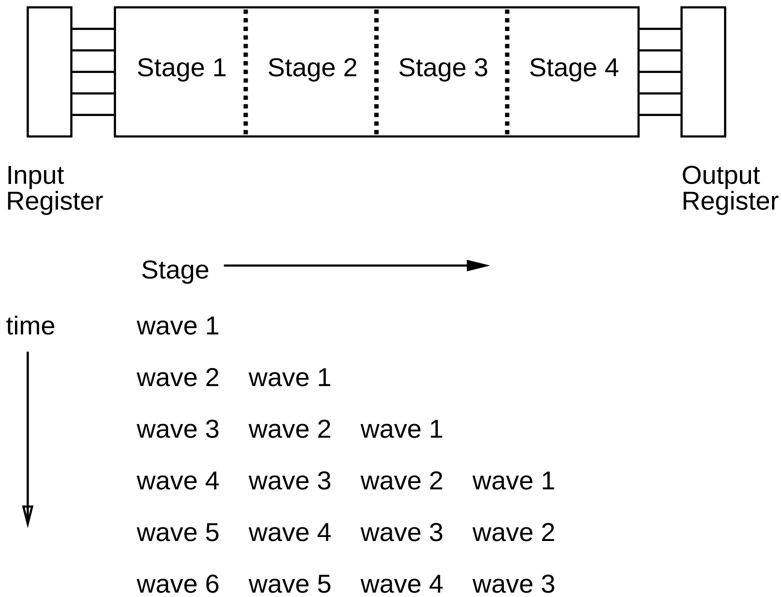


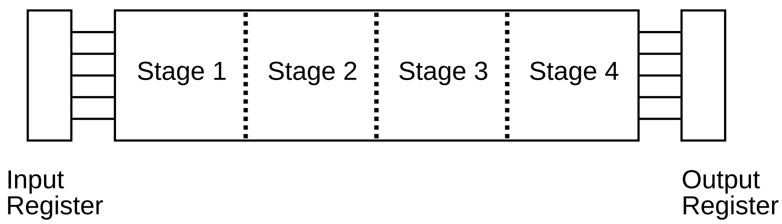
Figure 55: Propagation of Waves in Wave Pipeline

Figure 57 is a block diagram of a wave pipeline with an input register chain at the head of the pipeline. This approach requires additional area for the input register chain, sequencing logic, and input multiplexing. The cycle time of the wave pipeline may be increased by the addition of a multiplexor delay and the capacitive load of the input register chain gates. The refilling of the wave pipeline requires a number of penalty cycles which can be up to the number of stages in the wave pipeline. For long wave pipelines, the refill cycles can become prohibitively expensive.

An alternative to holding the inputs to the wave pipeline is holding the output of the pipeline. This works only if the stall condition does not alter the values produced by the wave pipeline. Figure 58 shows this condition. The results of waves 1, 2, and 3 are queued up at the output register. After the stall has been released, these results are multiplexed to the wave pipeline output. Additional logic is required to count the stall cycles and control the selection of the multiplexors on the release of the stall.

Figure 59 is a block diagram of a wave pipeline with a results queue at the tail of the pipeline. This approach requires additional area for the results queue, sequencing logic, and output multiplexing. The cycle time of the wave pipeline may be increased by the multiplexor delay and the capacitive load of the results queue gates. No penalty cycles to refill the wave pipeline are required.

Hybrid approaches can used when the wave pipeline results are influenced by the stall.



time

0	wave 1			
1	wave 2	wave 1		
2	wave 3	wave 2	wave 1	
3	wave 4	wave 3	wave 2	wave 1
4	wave 4	wave 4	wave 3	wave 2
5	wave 4	wave 4	wave 4	wave 3
6	wave 4	wave 4	wave 4	wave 4
	. . .			
	stall released			
k	wave 1	wave 4	wave 4	wave 4
k+1	wave 2	wave 1	wave 4	wave 4
k+2	wave 3	wave 2	wave 1	wave 4
k+3	wave 4	wave 3	wave 2	wave 1

Figure 56: Stall Handling in Wave Pipeline

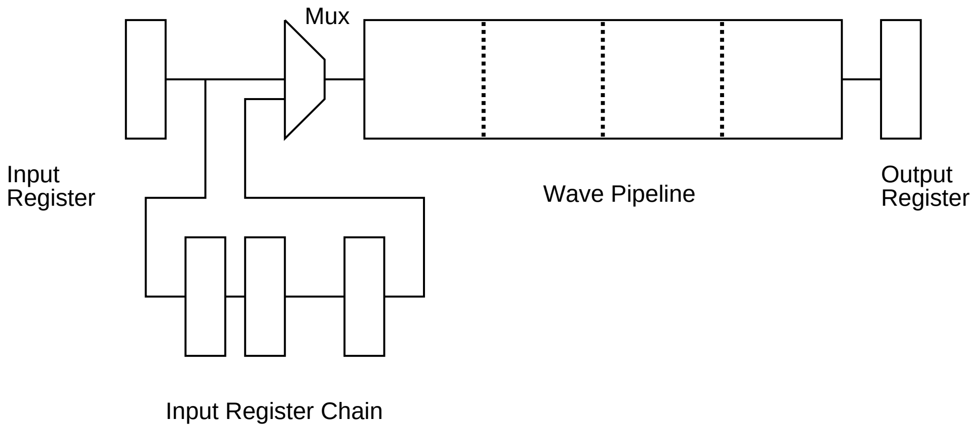


Figure 57: Wave Pipeline with Input Register Chain

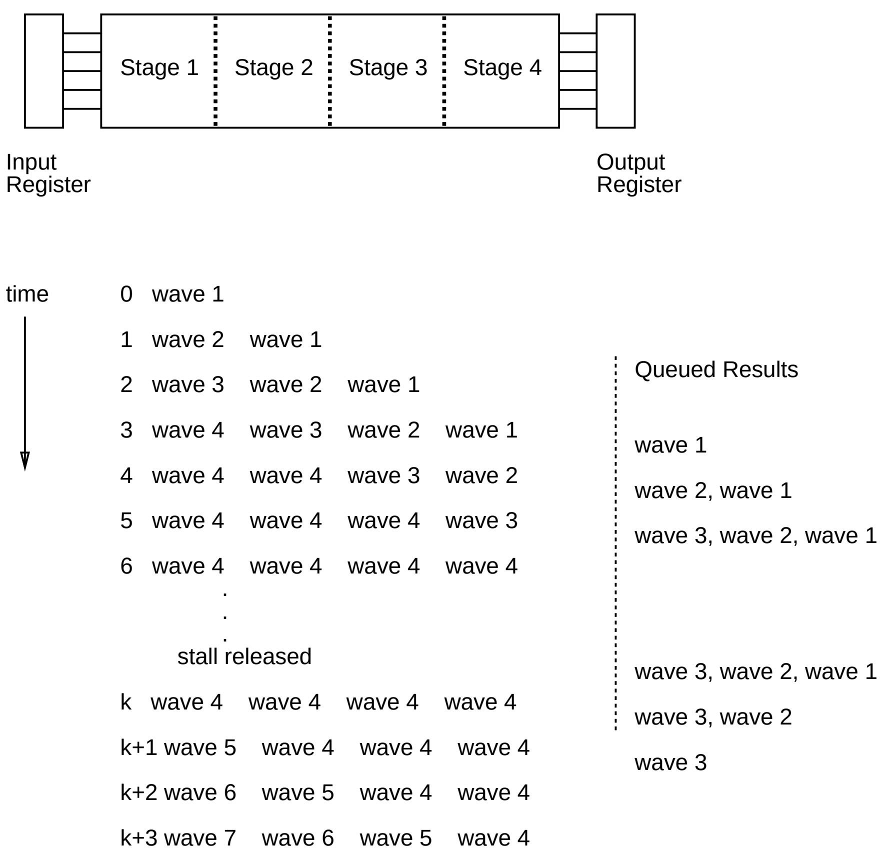


Figure 58: Stall in Wave Pipeline with Results Queue

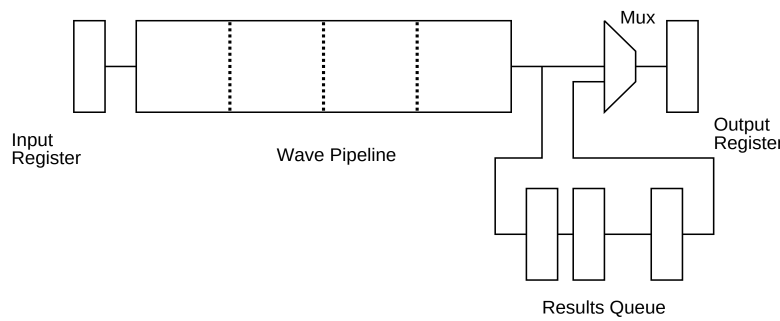


Figure 59: Wave Pipeline with Results Queue

Results of waves which precede the first stalling wave can be queued up at the output of the wave pipeline, while the stalling wave and its successors must be restarted at the head of the wave pipeline.

Dynamic Stalling of Wave Pipelines Note that the wave pipeline refilling and results queuing presented in the previous section would not be necessary if the wave pipeline could be "frozen" in time until the stall condition was resolved. Traditional pipelines "freeze" the pipeline by stalling. When a pipeline is stalled, data is not allowed to proceed to the next pipeline stage until the stall is released. Latches or registers act as barriers to the propagation of data through the pipeline. In this way, a stall condition can be resolved while the remainder of the system waits. This eliminates the need to queue results and refill pipelines.

In wave pipelining, it has not been possible to stall the pipeline since there are no latches or registers within the wave pipeline to act as barriers to signal propagation. In this section a method which allows some degree of stalling within a wave pipelined is presented.

To allow stalling of a wave pipeline, barriers to signal propagation are introduced at strategic positions within the wave pipeline. Transistors are used during the stall period to disconnect selected gate output nodes from the supply rails, thereby prohibiting changes in node states during the stall period. These transistors make the outputs of the selected gates dynamically latched.

Figure 60 is a diagram NAND gate which can be frozen during a stall. When *Stall* is inactive, the transistors driven by the *Stall* and *!Stall* signals act as closed switches. When *Stall* is active, these transistors act as open switches. Thus the output, *Out*, is unable to transition.

In a wave pipeline which supports N waves, such transistors are placed at N "freeze points" within the pipeline. These freeze points are positioned such that the maximum propagation delay between the freeze points is less than the wave pipeline clock period. At the freeze

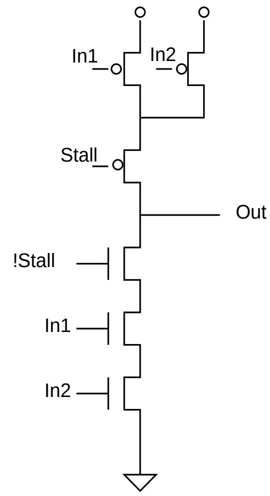


Figure 60: Freeze Points

point, the tristate transistors are disabled by stall signals which are active throughout the stall period. Figure 61 is a block diagram of a wave pipeline with freeze points to allow stalling.

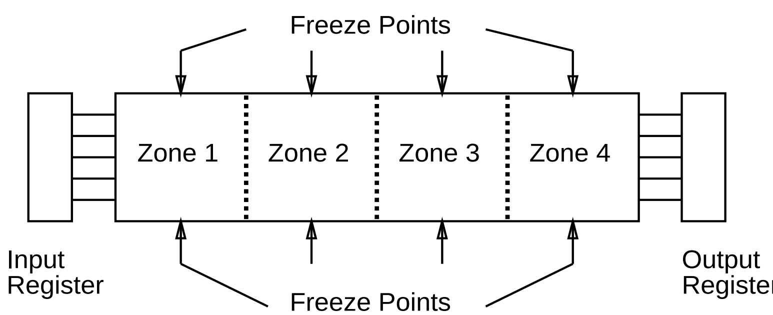


Figure 61: Wave Pipeline with Freeze Points

If activation of the stall signal is coincident with the enabling edge of the pipeline clock, the first freeze point is positioned less than one clock period from the input register or latch. This ensures that when a node is frozen, it is at its terminal voltage, not at an intermediate voltage.

The period for which the stall signals are held, T_{stall} is:

$$T_{stall} = N_{stall} * T_{clk} + (T_{freeze\ point}\ mod\ T_{clk}) \tag{101}$$

where N_{stall} is the number of required stall cycles, T_{clk} is the wave pipeline clock period, and $T_{freeze\ point}$ is the propagation delay from the input of the wave pipeline to the freeze point.

Figure 62 shows HSPICE simulated waveforms for a wave pipeline with freeze points. The wave pipeline consists of a chain of 50 CMOS inverters. This wave pipeline is has a maximum simulated propagation delay of 6.7ns. The circuit is clocked such that five simultaneous waves propagate through the inverter chain. To allow stalling, the 5th, 15th, 25th, 35th, and 45th inverters are freeze points. The top trace show a one-period pulse propagating down the pipeline at the 10th, 20th, 30th, 40th, and 50th inverters with no stalling. In the bottom trace the pipeline is stalled for two cycles. The waveforms at the same inverters are shown. Note that the stall has the effect of delaying by two cycles the edges which occur after the stall is initiated. The relative spacing of the edges is the same for the poststall edges as the prestall edges.

It must be noted that this technique uses dynamic holding of node voltages. The period for which the stall can be held is therefore limited.

The freeze points, while logically latching gates, are not pipeline latches; they remain transparent except during a stall. The following section compares the a wave pipeline with freeze points to a traditional pipeline of the same pipeline depth.

Comparison of Traditional Pipelines with Stalling Wave Pipelines In this section, implementations of a logic block for wave pipelining, wave pipelining with stalling, and traditional pipelines with dynamic and static latching are compared. A single phase clocking strategy is assumed for the traditional pipelines.

When dynamic latches are placed at the freeze points, the delay of each gate at a freeze point is increased due to the additional transistor between the gate output and the supply rail. The additional delay of a gate at the freeze point is T_{dlatch}. Thus, when N dynamic latches are placed within the combinational logic they increase the maximum propagation delay through the wave pipeline by $N * T_{dlatch}$. Thus the lower bound on the clock period becomes:

$$N * T_{clk} + cs \geq P_{max} + H_{max} + N * T_{dlatch} \tag{102}$$

or

$$T_{clk} + cs/N \geq P_{max}/N + H_{max}/N + T_{dlatch} \tag{103}$$

where T_{dlatch} is the increase in propagation delay of a freezing gate due to the dynamic latching transistors at the freeze points and H_{max} is the long path clock overhead:

$$H_{max} = \Delta C + T_s + RF_{max}/2 + T_{synch} \tag{104}$$

$$H_{min} = \Delta C + T_h + RF_{min}/2 - T_{synch} \tag{105}$$

Note that the difference between this constraint and constraint 1 is simply the additional delay through the freeze point dynamic latches. The upper bound on the wave pipeline

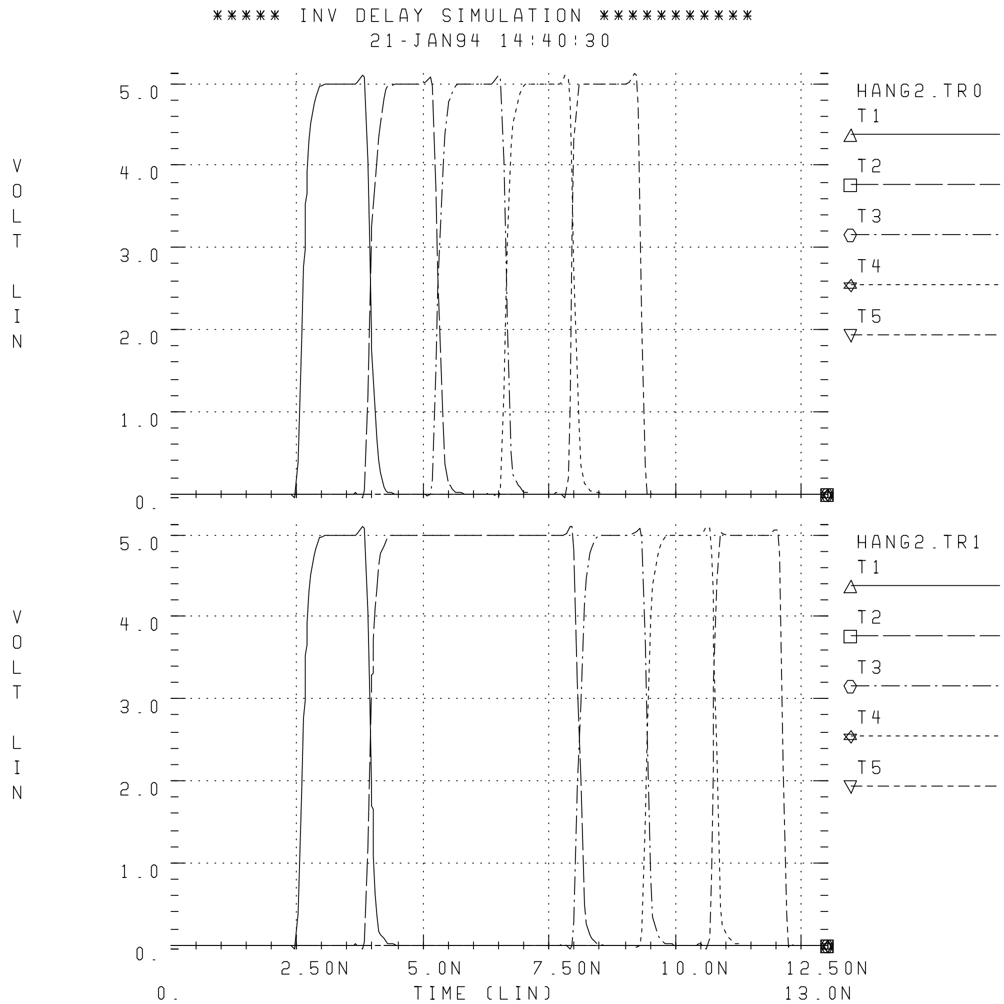


Figure 62: Stalling Wave Pipeline

clock period must also be met:

$$(N-1)*T_{clk}+cs \leq P_{min}-H_{min}+N*T_{dlatch} \tag{106}$$

where H_{min} is the race through clock overhead:

$$H_{min}=\Delta C+T_h+RF_{min}/2-T_{synch} \tag{107}$$

If the same block of logic is pipelined with traditional techniques, P_{max} is divided into N stages separated by synchronizing elements. The clock rate limit for the traditional pipeline is:

$$T_{clk} \geq P_{max}/N+H_{max} \tag{108}$$

With the traditional pipeline, it is assumed that the combinational logic can be equally partitioned into N stages. For cases where this approximation is not valid, the term P_{max}/N should be replaced with the delay of the longest stage.

Note that the stalling wave pipeline amortizes the clocking overhead over the number of waves in the wave pipeline. While the clock rate of the wave pipeline is decreased due to the freeze latches, it still outperforms the traditional pipeline of equal degree of pipelining.

Figures 63, 64, and 65 show the factor by which the maximum clock rate of a stalling wave pipeline exceeds the maximum clock rate of a traditional pipeline over a range of number of pipeline stages.

In these figures, the freeze point propagation delay, T_{dlatch}, are 10%, 20%, and 40% of the clock overhead, H_{max}, respectively. For instance, if the clock overhead, H_{max}, is 1ns, results are presented for the additional delay due to each freeze point latching gate being 100ps, 200ps, and 400ps.

In each figure, plots are given for four different maximum propagation delays. The plots vary from short to long pipelines. For instance, if the clock overhead, H_{max}, is 1ns, results are presented for pipelines whose maximum propagation delay are 2ns, 4ns, 8ns, and 16ns.

In all instances, the stalling wave pipelines are able to achieve greater performance than traditional pipelines. Clock rates up to seventy percent higher can be achieved with stalling wave pipelines. As the freeze point delays are increased, the benefits of the stalling wave pipeline diminish.

HSPICE simulations of a pipeline illustrate the performance effects of the freeze latches. The pipeline consists of fifty identical CMOS inverters. Each inverter was sized to have equal delay for rising and falling output. Since they are balanced the effects of delay imbalance are ignored. Thus the impact of the clocking and stalling mechanisms are determined.

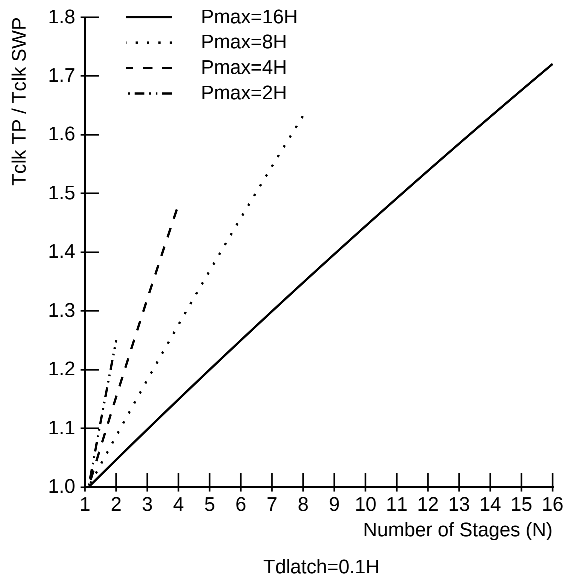


Figure 63: Relative Clock Rate (10% freeze delay)

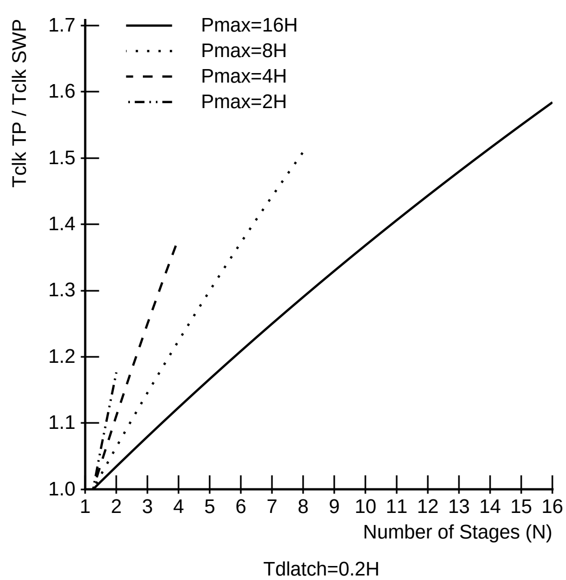


Figure 64: Relative Clock Rate (20% freeze delay)

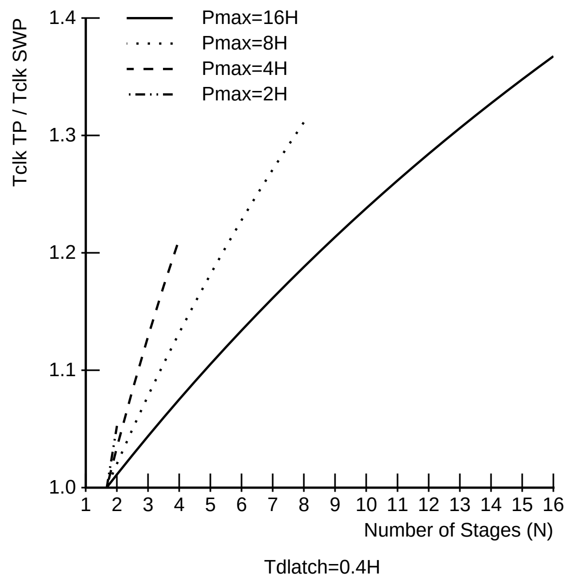


Figure 65: Relative Clock Rate (40% freeze delay)

Design	Number of Stages	Latency	Cycle Time
Wave pipeline	8	6.90	0.80
WP with stall	8	7.55	0.87
TP with dynamic latch	8	8.39	1.0
TP with static latch	8	10.09	1.2

Table 8: Performance of Pipelines

For the simulated pipelines, the clock to the output latch was skewed to ensure that a integral number of waves were held. In the traditional pipeline, this allowed the number of inverters in the final stage to be 8 as opposed to 6 in the previous stages. For this example $P_{max} \approx 13H_{max}$ and $T_{dlatch} \approx 0.2H_{max}$.

Additional performance may be realized for the stalling wave pipeline due to the decreased load on the clock driver.

The stalling wave pipeline has an increased area over the wave pipeline due to the freeze latches and additional signal routing. This area penalty is approximately the same as a traditional pipeline with dynamic latches as synchronizing elements. It is smaller than the penalty for a traditional pipeline with static latches or registers for synchronizing elements.

It has been shown that wave pipelines can be designed to ease the difficulty in stall handling.

Methods have been detailed which allow wave pipelines to be restarted with input register chains or to have results queued and sequenced out following a stall.

A method has been demonstrated by which a wave pipeline can be designed to dynamically stall. Despite the additional logic required to support stalling, a wave pipeline can outperform a traditional pipeline of equal depth.

5.1.2 Fully Latchless Feedback Circuits

The wave pipelines with feedback examined in Chapter 3 all contained at least one synchronizer in the feedback path. In this section, feedback without any intervening synchronizers are examined. Figure 66 is a wave pipeline with a feedback path without intervening synchronizers.

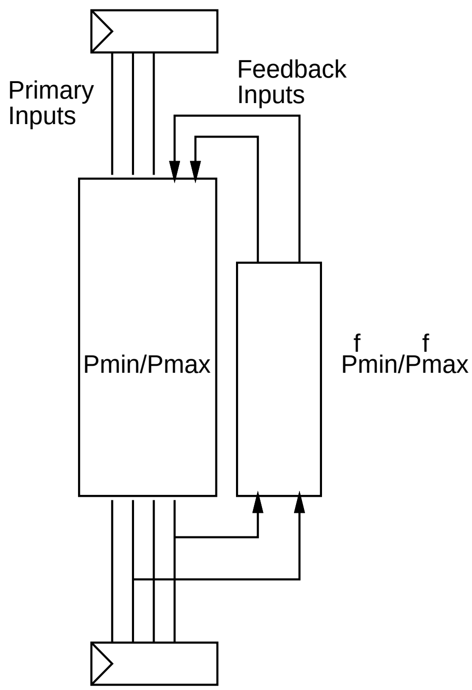


Figure 66: Wave Pipeline with Latchless Feedback

Waves interfere at the output of a wave pipeline if the time of the earliest application of the $(i+1)st$ inputs can propagate through the combinational logic so as to arrive prior to the latest possible arrival time of the data generated by the ith application of inputs. Thus the general wave interference constraint for a wave pipeline is:

$$T_i^{late} + P_{max} \leq T_{i+1}^{early} + P_{min} \tag{109}$$

Where T_i^{late} is the latest time at which input data wave i is applied to the combinational logic and T_{i+1}^{early} is the earliest time at which data input wave $i+1$ is be applied. Notice that clock distribution and synchronizer overhead are being ignored for simplicity.

For a register-based wave pipeline with latchless feedback, the time at which the primary inputs for data wave i are applied to the combinational network is $i * T_{clk}$. The feedback inputs for data wave i arrive at the inputs to the logic no earlier than:

$$j * T_{clk} + k * P_{min} + k * P_{min}^f$$

and no later than:

$$j * T_{clk} + k * P_{max} + k * P_{max}^f.$$

where $j * T_{clk}$ is the time in the past $(j < i)$ at which a change in the primary inputs resulted in the current change in the feedback. The number of cycles through the pipeline and feedback from the change in the primary inputs to the current feedback input change is k. Thus, at time $j * T_{clk}$ an input vector is applied to the logic network and after k iterations around the closed pipeline/feedback path a change in the feedback inputs occurs.

If the closed path has a maximum delay which is an integral number of clock periods, $P_{max} + P_{max}^f = N * T_{clk}$, the interference constraint becomes:

$$jT_{clk} + kNT_{clk} + P_{max} \leq (j+1)T_{clk} + kNT_{clk} - k*(P_{max} + P_{max}^f - P_{min} - P_{min}^f) + P_{min} \quad (110)$$

or,

$$T_{clk} \geq (k+1) * (P_{max} - P_{min}) + k * (P_{max}^f - P_{min}^f) \quad (111)$$

The clock constraint 111 implies that for latchless feedback, the minimum period of wave separation, T_{clk}^{opt}, is a linear function of the number of iterations, k, through the feedback loop which resulted in the current feedback inputs. Thus, in the general case, the minimum wave pipelined clock period for a wave pipeline with latchless feedback is unbounded, and thus, wave pipelines with latchless feedback will not function. There are, however, special cases for which latchless feedback will operate correctly. The first case is if there is no variation in delay through all closed feedback paths. The second case is if the delay variation of the k iterations through the feedback loop is independent of k. An example would be a feedback loop whose delay alternates between P_0 and P_1. The final case where a latchless feedback wave pipeline can operate is when the feedback inputs are periodically synchronized to the primary inputs or the primary input clock. This limits k in constraint 111 to some k_{max}. This synchronization may be through explicit qualifying, or logical ANDing, of the feedback signals with a clock signal or a primary input, or implicitly as a result of the function being performed by the logic.

5.1.3 Self-Timed Wave Pipelines

In self-timed circuit designs, data storage and synchronization is performed through the dynamic operation of the circuitry. The synchronizers time reference is provided either through use of methods of data encoding which can be used to distinguish when valid data has arrived at the output or through the use of a separate timing reference path through the logic which tracks the data propagation through the logic in a dynamic manner. Self-timed circuits, being asynchronous, use handshake protocols based upon *completion* and *ready* indications to ensure that operations do not interfere.

Several techniques which are used in self timing can be employed in wave pipelined systems to provide additional performance.

As shown in Chapter 3, the necessity of wave pipeline output synchronization can lead to cycle times which are greater than those imposed by the propagation variation constraints. In Chapter 3 intentional delay padding and intentional clock skew were shown to achieve the variation imposed optimal clock rate. Two alternative output synchronization methods based on self-timed techniques are possible: output encoding, and critical path delayed clock skew.

If data encoding techniques in which valid data and invalid data could be distinguished were employed in the wave pipeline, the output synchronizer could be clocked based upon the change in the output from invalid data to valid data. Unlike self-timed systems, this change from invalid to valid would not be used to initiate a subsequent data transfer at the wave pipeline input. This is because for a wave pipeline to have more than a single wave in the pipeline, new data must be input to the pipeline prior to the arrival at the output. The input synchronizer would be clocked by the system clock, whose period is constrained by the propagation delay variation through the pipeline.

In the critical path delayed clock skew method, the propagation delay of the critical path is replicated and placed in a path from the clock of the input synchronizer to the output synchronizer. In this manner, the interval between the input clock pulse which drives input data into the wave pipeline and the output clock which latches the results from that data tracks variations in the dynamic operation of the wave pipeline. Thus, the clock period of the wave pipeline can be guaranteed to be limited by just the propagation variation.

These techniques, while easing the synchronization of the wave pipeline output, result in significant logic overhead for CMOS wave pipelines.

5.2 Circuit Enhancements for Wave Pipelining

5.2.1 Low Variation Circuit Designs

Data dependent delay variation in static CMOS gates is primarily due to parallel conduction paths. Depending upon the number of paths from a gate output to the supply rail which are conducting at a given time, the gate delay can vary significantly. When all possible paths to the opposite rail are opened, the switching occurs much more quickly than when only a single path is opened.

The variation due to the parallel paths can be minimized by minimizing the capacitance driven by a gate with parallel supply paths. Figure 67 shows how a CMOS decoder driving a large capacitance can designed to minimize delay variation. In this figure, a wordline with a capacitance 256-times the input capacitance of a minimum-sized inverter is driven by a wordline driver which is 32-times minimum-size. In the first case, the wordline driver is driven by a NOR2NAND3 decoder. The NOR2 stage is minimum sized and the NAND3 stage is 4-times minimum size. During disable, the NOR2 can have equivalent channel impedance of R or R/2 depending upon if one or both paths are activated. The NAND3 can have an equivalent impedance of R, R/2, or R/3. Using an RC delay model the maximum delay would be:

$$R * 4C + R/4 * 32C + R/32 * 256C = 20RC \tag{112}$$

The minimum delay would be:

$$R/2 * 4C + R/(4 * 3) * 32C + R/32 * 256C = 12.67RC \tag{113}$$

Thus, the maximum delay is 1.58-times greater than the minimum delay. By minimizing the capacitance of the nets which have parallel paths to the supply, this variation can be reduced. In Figure 67 the NOR2 and NAND3 gates are minimum size. The decoder output is driven through a chain of scaled inverters. The maximum delay is:

$$R * C + R * C + R * 4C + R/4 * 32C + R/32 * 256C = 22RC \tag{114}$$

The minimum delay would be:

$$R/2 * C + R/3 * C + R * 4C + R/4 * 32C + R/32 * 256C = 20.83RC \tag{115}$$

Hence, the maximum delay is 1.056-times the minimum delay.

As illustrated in this example, to minimize delay variation due to data dependencies of CMOS gates with parallel conduction paths, all nodes driven by gates with parallel paths to the supply rails should have minimum capacitive load. Large capacitive loads should be driven by inverters with balanced pullup and pulldown times.

The intertransistor parasitic capacitance effect becomes important when the magnitude of small capacitive loads driven by gates with parallel paths approach the intertransistor parasitic capacitances. rails. This tends to limit the minimum delay variation which can be achieved by the above technique.

5.3 Summary

This chapter has exposed architectural and circuit enhancements for wave pipeline design.

Methods of providing external support for stalls through operand or output stall fifos are examined. In addition, a technique for providing stalling capabilities within wave pipelines

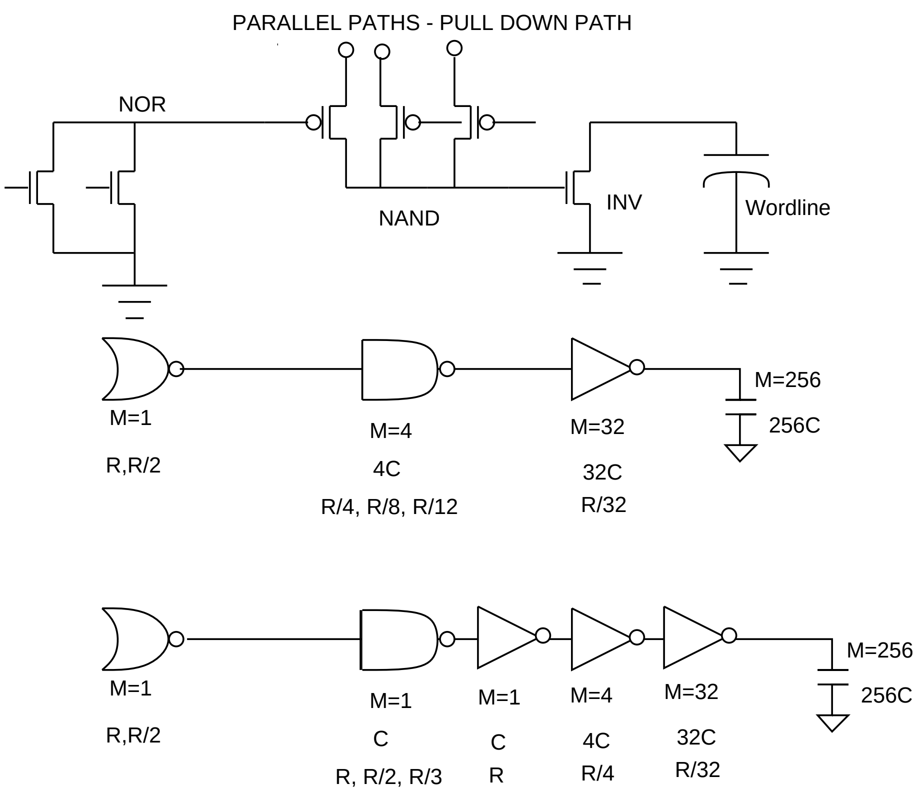


Figure 67: Decoder Delay Variation

has been developed. Stalling wave pipelines make use of dynamically latching gates within the combinational logic to impede the flow of data through the logic network during a stall condition. By not incurring the clock overhead incurred by conventional pipelines, the performance of a stalling wave pipeline in which the delay variation is sufficiently small exceeds the performance of a conventional pipeline of equal depth.

Wave pipelines with fully latchless feedback have been shown to be impractical for most circuits.

Design techniques to ease the wave pipeline constraints on the output register clocks have been explored; use of data encoding which distinguish when valid data has arrived at the output or through the use of a separate timing reference path through the logic which tracks the data propagation through the logic in a dynamic manner are suggested.

Optimizations to CMOS wave pipeline circuit designs to minimize the delay variation due to parallel conduction paths have been presented in this chapter. By minimizing the capacitance driven by gates with parallel conduction paths and driving the large capacitances with balanced drivers, this delay variation can be minimized.

6 Summary and Conclusions

6.1 Summary

Wave pipelining is a circuit design technique that allows digital synchronous systems to be clocked at rates higher than can be achieved with conventional pipelining. Rather than partitioning the combinational logic into pipeline stages separated by latches or registers, wave pipelines rely on the predictable, finite propagation delay through the logic gates to store the data. In a conventional pipeline the data synchronization function is performed by the registers or latches; in a wave pipeline, knowledge of the signal propagation delays is used at design time to ensure that synchronization failures cannot occur.

Conventional pipelined systems allow data to propagate from a register through the combinational network to another register prior to initiating the subsequent data transfer. Thus, the maximum operating frequency is determined by the maximum propagation delay through the longest pipeline stage. Wave pipelined systems apply the subsequent data to the network as soon as it can be guaranteed that it will not interfere with the current data wave. The maximum operating frequency of a wave pipeline is therefore determined by the difference between the maximum propagation delay and the minimum propagation delay through the combinational logic. Wave pipelining of combinational circuits have been shown to achieve clock rates 2 to 7-times those possible for the same circuits with conventional pipelining.

Wave pipelining concepts were introduced, wave pipelines were compared to conventional pipelines, and previous wave pipelining research and related research were presented in Chapter 1.

While previous research efforts have demonstrated performance benefits through the use of wave pipelining of CMOS circuits, quantitative determination of the potential of wave pipelining in CMOS has been primarily empirical. Models for the performance potential of CMOS wave pipelined circuits based upon CMOS delay equations and simulations were developed in this research. These models, when variations due to process and operation environment are ignored, produce performance limits with the 6 to 7-times from previous simulated results. When the variations due to process and operation environment are included, the modelled results correspond closely to measured results.

The effects of logic network path delay imbalance, changes in operating environment, and fabrication process variation on the performance of wave pipelines have been ascertained. Path length imbalance and changes in the rate of signal propagation due to dependencies upon the values of the input data have been found to result in propagation delay variations of 10 to 20% when variation minimization techniques presented in Chapters 3 and 5 are employed. This variation limits the speed-up of wave pipelined circuits to at most 11 and 6, respectively. Variations of propagation delays of different dice from common fabrication line are shown to have ratios of worst case delay to best case delays of 1.35 to 1.5. This variation, independent of other factors, limits the speed-up of wave pipelining to at most three to four. Changes in operating temperature may degrade delays by a factor of up to 1.4, thereby limiting the speed-up to 3.5. Supply drops and drift, supply noise, and coupled noise can result in further propagation delay variation. For representative CMOS circuits the aggregation of the variation causes result in worst-case delays along critical circuit paths which are 2.7-times slower than best-case delay along the fast paths. Variations of this magnitude limit the clock frequency of wave pipelines to 1.6-times the rate which can be achieved without pipelining.

The seminal clocking constraints for wave pipelining which were extended in this research to include variations due to process and environment, a model for CMOS propagation delay, the causes of delay variation in CMOS circuits, the effects of these variations on the performance of CMOS wave pipelines, and a comparison of the impact of these factors on conventional pipelines and wave pipelines were presented in Chapter 2.

Chapter 3 presented design techniques for the optimization of the performance of wave pipelines. A method of minimizing the path imbalance in the design of CMOS wave pipelined circuits based upon transistor sizing was developed. This method is an extension of the Wong method for balancing CML logic networks [53]. The optimization uses a topological model of the circuit and macromodels of gate delay based upon HSPICE simulations to generate a linear program representation of the transistor sizing problem. The solution to the linear program is used to set the lengths and widths of the CMOS transistors so as to minimize the path length variations in the logic network. This optimization method has been integrated into a wave pipelining circuit design environment.

For several representative circuits this method of delay balancing has optimized the circuits such that the path delay and data dependent delay variation are limited to the 0% to 20% range. These results are consistent with the best manual balancing methods which show variation of up to 15% to 20% [29]. Balance in this range is sufficient for high performance wave pipelines.

Further performance potential may be lost in wave pipelining due to the difficulty of operating all wave pipelines in a system at a common clock frequency. Circuit optimizations which use intentional delay insertion and/or constructive clock skew can be used to minimize the performance impact of synchronization of wave pipelines within a system.

The variations in delay due to process and operating environment impose severe limits on the performance of CMOS wave pipelined circuits. Several means of minimizing the variations or the performance effects of these variations are evaluated in Chapter 3. Frequency sorting was shown to be significantly more constrained for wave pipelines than for conventional pipelined circuits. For high performance wave pipelines sorting was demonstrated to be impractical. Methods to compensate for process and environmental delay variation so as to avoid the difficult frequency sorting procedure as well as to gain additional performance were examined. Tunable constructive clock skew, use of biased logic, use of tunable delay buffers, delay compensation through thermal control, and an adaptive power supply voltage method of minimizing delay variation were presented in Chapter 3. The tunable clock skew method is impractical for a VLSI system in which multiple wave pipelines which are part of a synchronous system with feedback because of the need to individually skew each clock and because of the interrelation of clocks in a system with feedback. The biased logic methods suffer from severe area penalties and/or power consumption and noise margin problems. Methods based upon tunable delay buffers or thermal compensation suffer from limited range of delay adjustment. The adaptive power method provides sufficient range of delay adjustment, does not increase logic area, and can lower power consumption while compensating for fabrication process and temperature dependent delay variations. This method was used in the design of the wave pipelined CMOS VLSI vector unit presented in Chapter 4.

To demonstrate the techniques and tools developed as part of this research a CMOS wave pipelined VLSI vector unit. This integrated circuit demonstrates that wave pipelined CMOS VLSI systems can be designed to perform within the performance limits described in Chapter 2. This system integrates and synchronizes multiple heterogeneous wave pipelines: Wave pipelining was employed in the design of the vector register file, the add functional unit, and the multiply functional unit. It was designed with the assistance of automated CAD balancing tools which employ the transistor sizing method presented in Chapter 3. It contains adaptive power supply support for the maintenance of wave pipelined operation over a range of operating conditions and fabrication tolerances. This unit was fabricated in a 0.8 micron process and was tested at operating frequencies up to 303MHz. At 300MHz, the vector register file supported 1.1 concurrent waves of data, the adder supported 1.9 waves of data, and the multiplier supported 3.7 waves of data with no intervening latches.

To allow a comparison of performance and costs between wave pipelining and conventional pipelining, an equivalent vector unit was designed using an aggressive traditional clocking technique. The vector unit design which used conventional pipeline clocking was approximately 2% larger due to the latches which had to be inserted into the functional units. The simulated latencies through the multiplier and adder functional units using conventional pipelining were 8% and 11% longer, respectively. The maximum simulated clock rate achieved by using wave pipelining was 35% faster than that which could be achieved using conventional two phase clocking and dynamic latches. Details of the organization, operation, balancing, and testing of the vector unit are given in Chapter 4.

Architectural and circuit enhancements for wave pipelining were detailed in Chapter 5. A significant barrier to the use of the wave pipelining design technique has been the difficulty in stalling a wave pipeline. Once a wave has been launched, it precedes unimpeded through the combinational network. Methods of providing external support for stalls through operand or output stall fifos are examined. In addition, a technique for providing stalling capabilities within wave pipelines has been developed. Stalling wave pipelines make use of dynamically latching gates within the combinational logic to impede the flow of data through the logic network during a stall condition. During normal, nonstall operation, the dynamically latching gates are not switched and thus only marginally increase the propagation delay through the logic. By not incurring the clock overhead incurred by conventional pipelines, the performance of a stalling wave pipeline in which the delay variation is sufficiently small exceeds the performance of a conventional pipeline of equal depth.

Because the need for synchronizers within the combinational logic of a wave pipeline is obviated, the latencies, clock frequencies, and die areas of wave pipelined implementations can be superior to conventional pipelines. Wave pipelines with fully latchless feedback have closed loop feedbacks of pipeline outputs to inputs. The lack of synchronizers in the feedback loop make this organization impractical for most circuits.

The constraints on the timing of the output clock of wave pipelines can be eased through the use of self-timing techniques. In self-timed circuit designs, data storage and synchronization is performed through the dynamic operation of the circuitry. The synchronizers time reference is provided either through the use of methods of data encoding which distinguish when valid data has arrived at the output or through the use of a separate timing reference path through the logic which tracks the data propagation through the logic in a dynamic manner. These same techniques can be used in wave pipeline designs to generate the output clock for the wave pipeline. The self-timed output clock eliminates the need for the intentional delay insertion or clock skewing methods presented in Chapter 3.

Optimizations to CMOS wave pipeline circuit designs to minimize the delay variation due to data dependent delays are presented in Chapter 5. The delay variation effects due to parallel conduction paths of outputs driving large capacitances are examined. By minimizing the capacitance driven by gates with parallel conduction paths and driving the large capacitances with balanced drivers, this delay variation can be minimized.

6.2 Conclusions

As a result of this research effort the following conclusions may be drawn:

Wave pipelining of CMOS VLSI systems can result in 1 to 2-times increase in the rate at which the combinational logic can be clocked when the system is required to operate over reasonable environmental conditions and with typical process variation without compensation for the changes in propagation delay.

With adaptive compensation techniques presented, the clock rates of CMOS wave pipelined systems can be 2 to 6-times those which could be achieved without pipelining.

Delay balancing based upon transistor sizing can limit the variation in path delay to less than 20% for practical circuits. This degree of accuracy is consistent with manual balancing methods.

Additional performance can be gained in CMOS wave pipelined circuits through synchronization of pipeline clocks and minimizing the capacitive loads of gates with parallel conduction paths.

Stalls can be supported in wave pipelined systems through fifos external to the wave pipeline or through the use of dynamic stalling gates. If the variation in delay of a wave pipeline is sufficiently low, a stalling wave pipeline has higher performance than a conventional pipeline.

Relatively large CMOS wave pipelined systems can be designed, tested, and operated. The vector unit developed as part of this research effort contained wave pipelined multiply and add functional units and a wave pipelined vector register file, all of which were synchronized to each other and remaining conventional synchronous system. Most circuit optimizations were automated. This design performed at up to 303MHz. The vector register file supports 1.1 concurrent waves of data, the adder supports 1.9 waves of data, and the multiplier supports 3.7 waves of data with no intervening latches. This design by using wave pipelining achieved a clock rate 35% faster than could be achieved using conventional two phase clocking and dynamic latches.

6.3 Future Wave Pipelining Research

Although this research effort contributed to the practical application of wave pipelining in CMOS VLSI system design, several areas of further wave pipelining research are evident. They are broadly classified as device models and tools, adaptation, and implementations and architectures.

6.3.1 Models and Tools

The as discussed in Section 3.1.6, the accuracy of the CMOS fine balancing tool was limited due to the simple model of delay employed. More accuracy and, thus, potentially higher performance wave pipelines could be realized with better models of gate and network delay. Significant research has been conducted in the area of delay modeling and and delay fault testing which could be leveraged to provide higher accuracy wave pipeline optimization tools. In the area of tools for wave pipelining, efficiencies could be gained in the balancing of wave pipelined circuits if more efficient solution methods for solving the linear programs were employed.

6.3.2 Adaptation

This research has shown that CMOS wave pipelines will not approach their performance potential in system applications without the ability to compensate for variations in delay due to process and operating environment. Additional research on constant propagation delay, closed loop compensation techniques and their effect on circuit performance and reliability is warranted. Additional work on variable propagation delay, self-timed output wave pipelines could further serve to increase the acceptance of wave pipelines in the design of VLSI systems.

6.3.3 Implementations and Architectures

Wave pipelining has been applied to bipolar, static CMOS, domino CMOS, and pass-logic CMOS circuits. Additional analysis of the application of wave pipelining techniques to other technologies such as BiCMOS and GaAs. Where as driving of large capacitances in CMOS leads to slow rise/fall times and large data dependent variations, selective use of bipolar drivers for these nodes within BiCMOS systems could improve the performance of wave pipelined designs. Analysis of the applicability of wave pipelining to additional logic families such as families such as dual pass transistor and CVSL are warranted.

Low power wave pipelined circuits is an area which is largely unexplored. A thorough examination of the performance benefits of wave pipelining with stringent power constraints would prove valuable for the advancement of wave pipelining.

The effects of wave pipelining on pipeline architectures presents another opportunity for future research. The impact of the wave pipeline clocking techniques on processor pipeline architectures, instruction set architectures, and exception handling could prove to be important.

A Symbols

C_l	The total load capacitance.
$C_{coupled\ i}$	The mutual capacitance of the output and the signal i.
C_{gate}	The capacitance of the gate of a transistor.
C_{int}	The capacitance of the wires connected to a gate output.
C_{ox}	The per area oxide capacitance.
cs	The constructive skew between the clock at the input
cs^i_{sys}	The time by which the clock to the output latch of wave pipeline i lags the system reference clock. synchronizer and output synchronizer.
$D[i,0]$	The propagation delay from the source node to node i with falling output.
$D[i,1]$	The propagation delay from the source node to node i with rising output.
d_i	The direction of coupled signal switch (-1 if opposite to output and 1 if same as output).
$fast$	The operating conditions $(V_{max}, \tau_{min}, \phi_{fast})$.
$f[from, to, direct](M_w, C_l)$	The propagation delay function of the gate connecting $from$ to to under the given conditions.
f_h	The horizontal mobility degradation factor.
$f_l in$	The linear approximation to the delay function f.
f_v	The vertical mobility degradation factor.
H	The clock overhead including rise/fall time, clock skew, setup or hold time, and synchronizer output time.
H^i_{max}	The worst case maximum synchronizer overhead.
H^i_{min}	The worst case minimum synchronizer overhead.
$I_{ds\ max}$	The maximum MOS transistor source-drain current.
k	A factor to account for the difference between the maximum and average current over the period during which a transistor switches.
k	The number of iterations a signal transitions makes around the feedback loop in wave pipeline with latchless feedback.
K	Transconductance per unit ratio of channel width to channel length.
K_n	The NMOS transconductance.
K_p	The PMOS transconductance.
L	The channel length.
L	The effective transistor length.
L_0	The nominal effective transistor length prior to balancing.
M	The exponential temperature dependent delay constant between 1.5 and 2.

M_l	The length modification factor of a transistor being balanced.
M_w	The width modification factor of a transistor being balanced.
N	The number of concurrent waves in the wave pipeline.
N_{max}	The maximum number of waves in the wave pipeline, also represents the maximum speed up of a wave pipeline over the same circuit being operated as a traditional pipeline stage.
N_{stall}	The number of required stall cycles.
P^f_{max}	The maximum delay of the feedback path in a wave pipeline with latchless feedback.
P_{max}	The worst case maximum propagation delay through the combinational network.
P_{min}	The best case minimum propagation delay through the combinational network.
RF_{max}	The maximum rise/fall time of the inputs to the output synchronizer.
R_{eq}	The equivalent resistance of a conducting transistor.
$slow$	The operating conditions $(V_{min}, \tau_{max}, \phi_{slow})$.
T^{early}_{i+1}	The earliest time at which data input wave $i+1$ is applied to the combinational logic.
T^{late}_i	The latest time at which input data wave i is applied to the combinational logic.
T^{opt}_{clk}	The lower limit on the clock period of a wave pipeline due to the variation in propagation delay.
T_{clk}	The clock period.
T_{dlatch}	The increase in propagation delay of a freezing gate due to the dynamic latching transistors at the freeze points.
$T_{freeze\ point}$	The propagation delay from the input of the wave pipeline to the freeze point.
T_{ms}	The minimum amount of time a node voltage must be stable to ensure the subsequent level of logic operates correctly.
T_{ox}	The device oxide thickness.
T_{pd}	The propagation delay of combinational logic.
T_{phl}	The propagation delay of a gate with output transitioning from the high to the low level.
T_{plh}	The propagation delay of a gate with output transitioning from the low to the high level.
T_{stall}	The period for which the stall signals are valid.
T_{synch}	The maximum time from the data initiating edge of the clock to valid output of the input synchronizer.

T_s	The maximum setup time of the output synchronizer.
T_{trans}	The time over which the latch is open and transparent.
V	The supply voltage.
V_{dd}	The supply voltage.
V_{dmax}	The maximum drain to source voltage during velocity saturation.
V_{sat}	The drain to source saturation voltage.
v_{sat}	The saturation velocity.
V_{tn}	NMOS threshold.
V_{tp}	PMOS threshold.
V_t	The device threshold.
W	The effective transistor width.
W_0	The nominal effective transistor width prior to balancing.
W_{max}	The maximum allowable width of a transistor.
W_{min}	The minimum allowable width of a transistor.
X_{ij}	The largest integer difference in the number of waves in the longest and shortest paths in a polyharmonic wave pipeline.
α	The ratio of the largest propagation delay through a logic network to the smallest propagation delay through the network.
β	The propagation delay degradation factor due to process and environmental variations.
ΔC	The unintentional clock skew between input and output clocks.
ΔP	The difference in propagation delay between the longest and shortest path through a combinational network.
δP^i_{max}	The worst case delay introduced in to path i during the intentional delay insertion process.
ϵP^j_{max}	The worst case delay introduced in to path j to equalize the delays in a polyharmonic wave pipeline.
γ	The worst-case process and environmental degradation factor relative to *typical* process and environmental conditions.
μ_0	The low-field channel mobility.
μ_n	The electron channel mobility.
μ_p	The hole channel mobility.
ϕ	The fabrication process.
τ	The operating temperature.

B Delay Models for CMOS Circuits

This appendix presents the delay models used in the analysis of the performance limits of CMOS wave pipelining presented in Chapter 2.

For simplicity, propagation delay is defined as the time from the controlling input reaching 50% of its terminal value to the output reaching 50% of its terminal value. The Elmore model is used for the delay of the network [13]. In this analysis, the propagation delay along a path in a logic network is the sum of the step-input delays of the individual gates along the path.

The step-input propagation delay of a CMOS gate, T_{pd}, consists of the time it takes for a load capacitance to be charged or discharged from its initial voltage to 50% of the terminal voltage. Thus,

$$T_{pd} = C_l * \int \frac{dV}{I(V, \tau, \phi)} \tag{116}$$

where C_l is the total load capacitance, τ represents operating temperature, and ϕ distinguishes the fabrication process.

To estimate T_{pd}, gates are represented as a single transistor, sized so as to match the current carrying capacity of the complex gate, charging or discharging a fixed load capacitance.

Using long-channel MOS current equations, the propagation delay equations for a high-to-low transitioning output assuming step input are: [50]

$$T_{phl} = t_1 + t_2 \tag{117}$$

$$t_1 = \frac{2C_l V_{tn}}{K_n (V_{dd} - V_{tn})^2} \tag{118}$$

$$t_2 = \frac{C_l}{(V_{dd} - V_{tn}) K_n [ln(\frac{3V_{dd} - 4V_{tn}}{V_{dd}})]} \tag{119}$$

For short-channel MOS devices, where velocity saturation limits channel current, the propagation delay for low-going outputs assuming step input is:

if $V_{dmax} > V_{dd}/2$,

$$T_{phl} = t_1 + t_2 \tag{120}$$

$$t_1 = \frac{2C_l (V_{dd} - V_{dmax})}{K_n V_{dmax}^2} \tag{121}$$

$$t_2 = \frac{C_l}{K_n [\frac{1}{V_{dd} - V_{tn}} ln(\frac{V_{dmax}(1.5V_{dd} - 2V_{tn})}{V_{dd}(V_{dd} - V_{tn} - 0.5V_{dmax})}) + (2/V_{sat}) ln(\frac{1.5V_{dd} - 2V_{tn}}{2V_{dd} - 2V_{tn} - V_{dmax}})]} \tag{122}$$

else,

$$T_{phl} = \frac{C_l V_{dd}}{K_n V_{dmax}^2} \tag{123}$$

where,

$$V_{sat} = \frac{L * v_{sat}}{\mu_n} \tag{124}$$

$$V_{dmax} = V_{sat}[(1 + \frac{2(V_{dd} - V_{tn})}{V_{sat}})^{0.5} - 1] \tag{125}$$

$$K_n = \mu_n C_{ox} W/L \tag{126}$$

and W is channel width, L is channel length, μ_n is electron channel mobility, C_{ox} is per area oxide capacitance, v_{sat} is the saturation velocity, and V_{tn} is NMOS threshold.

Corresponding equations for the propagation from low-to-high transitioning output result from the application of the same delay model, *mutatis mutandis*. Using long-channel MOS current equations, the propagation delay equations for high-going outputs assuming step input are:

$$T_{plh} = t_1 + t_2 \tag{127}$$

$$t_1 = \frac{-2C_l V_{tp}}{K_p(-V_{dd} - V_{tp})^2} \tag{128}$$

$$t_2 = \frac{-C_l}{(-V_{dd} - V_{tp})K_p[ln(\frac{-3V_{dd}-4V_{tp}}{-V_{dd}})]} \tag{129}$$

For short-channel MOS devices: If $V_{dmax} > V_{dd}/2$,

$$T_{plh} = t_1 + t_2 \tag{130}$$

$$t_1 = \frac{2C_l(-V_{dd} - V_{dmax})}{-K_p V_{dmax}^2} \tag{131}$$

$$t_2 = \frac{C_l}{K_p[\frac{1}{-V_{dd}-V_{tp}} ln(\frac{V_{dmax}(-1.5V_{dd}-2V_{tp})}{V_{dd}(-V_{dd}-V_{tp}-0.5V_{dmax})}) + (2/V_{sat}) ln(\frac{-1.5V_{dd}-2V_{tp}}{-2V_{dd}-2V_{tp}-V_{dmax}})]} \tag{132}$$

else,

$$T_{plh} = \frac{C_l V_{dd}}{K_p V_{dmax}^2} \tag{133}$$

where,

$$V_{sat} = \frac{L * v_{sat}}{\mu_p} \tag{134}$$

$$V_{dmax} = V_{sat}[(1 + \frac{2(-V_{dd} - V_{tp})}{-V_{sat}})^{0.5} - 1] \tag{135}$$

$$K_p = \mu_p C_{ox} W/L \tag{136}$$

and W is channel width, L is channel length, μ_p is hole channel mobility, C_{ox} is per area oxide capacitance, v_{sat} is the saturation velocity, and V_{tp} is PMOS threshold.

If the gate is balanced, the propagation delays for rising and falling outputs are equal. For cases where propagation delays are represented by a single number and the gates are not balanced, the arithmetic average of the rising and falling delays is used. For the analysis in Chapter 2, balanced gates are assumed. For the CMOS delay tuning via transistor sizing presented in Chapter 3, the rising and falling delays are treated separately.

C Adaptive Power Control

This appendix analyzes the performance and stability of the constant delay, adaptive power technique introduced in Chapter 3.

A closed-loop adaptive power design consists of a delay error detector circuit (phase comparator) and a power supply regulator. The supply regulator can be implemented as a dissipative circuit which lowers the power supply voltage, or a pulse-width-modulation circuit which controls the performance of a switch-mode regulator.

As the constant delay adaptive power technique was used for a relatively high power design, this analysis concentrates on the switch-mode regulation method.

A fixed frequency source is applied to an inverter chain which acts as a voltage controlled delay. The inverter chain is designed such that for the slow corner process and the worst-case operating temperature and noise, the delay through the chain with a nominal power supply is equal to 1/2 the period of the fixed frequency source.

The input to the delay chain and the output are phase compared using a master/slave latch circuit shown in Figure 68 [2]. The outputs of the comparator are exclusive. When the input to the delay chain lags the output, the delay through the chain is too short and the supply voltage to the chain must lowered. Under this situation, the chip supply is operating faster than the design target and the chip supply should be lowered. The lagging input results in the activation of the ChargeRemove signal which results in a fixed amount of charge to be removed from the capacitor on the Vctrl node. The removed charge lowers slightly the supply to the delay chain, thereby increasing the delay through the delay chain.

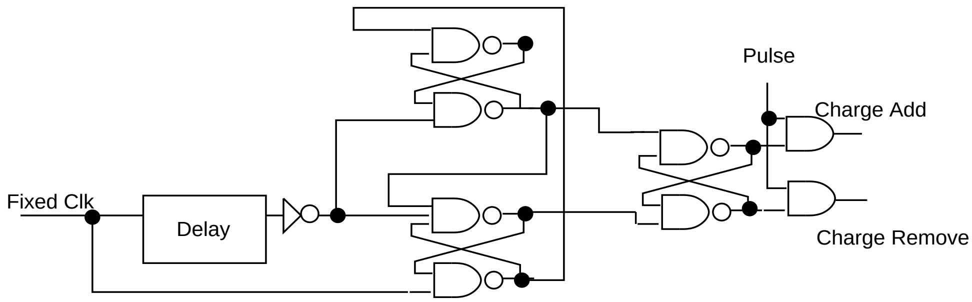


Figure 68: Phase Comparator Circuit

If the delay chain input leads the output, the supply is too low. In this case, the ChargeAdd signal is activated, a small charge is added to the charge pump capacitor, and the reference voltage is increased.

The value of the control capacitor and the amount of charge added to or removed from the capacitor are chosen so as to ensure the stability and response time of the control circuit.

The control voltage fixed by the delay loop is an approximation to the voltage at which the delay of the chain equals the design target for the realized process and the current operating conditions. If the bandwidth of the delay loop is sufficiently high when compared to the time constant of the environmental changes, this approximation is accurate.

To approach constant delay of all circuitry, the chip supply voltage should track the delay detector control voltage. The control voltage is used as the reference voltage for the buck converter in the switching power supply. The reference voltage change results in a change in the duty cycle of the switching waveform to the switching transistor. The modeling of the buck converter shown in Figure 69 is based upon work by Chetty [9].

Figure 70 shows the response of the delay detector and converter to the initial voltage locking due to a process which is faster than the design process. The top graph is the ChargeAdd indication and the middle graph is the ChargeRemove signal. The bottom graph shows two traces: The top trace in the graph, *node* 6, is the chip power supply node. The bottom trace in the graph, *node* $PULL$, is the reference voltage. This graph shows that within 50 microseconds, the chip power supply voltage reaches the value which achieves design target propagation delays.

Figure 71 shows the response of the adaptive power circuit to a rise in die temperature of the vector unit chip. At time t=0 the power consumption rises from a standby power consumption of 0.5W to a power consumption of 1.9W. The first graph in Figure 71 gives the thermal response at the location of the delay chain on the die. The temperature rise has a time constant of approximately 350ms. The temperature rise is approximately 43 C.

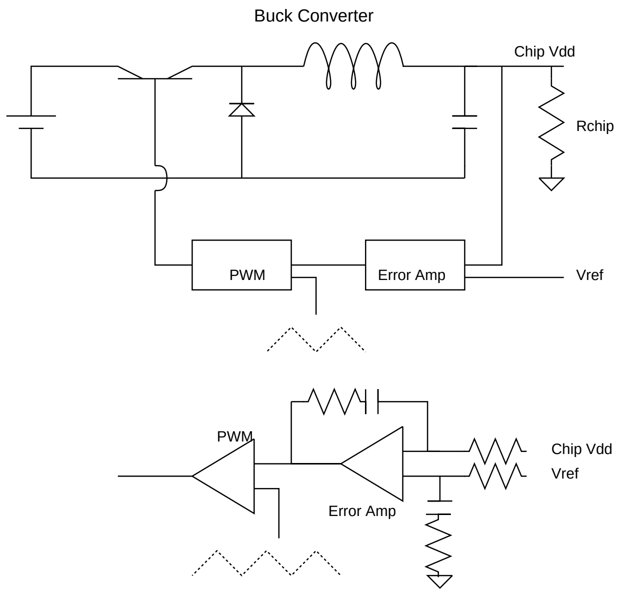


Figure 69: Power Converter Circuit

The second graph presents the response of the control voltage, *node PULL*, and the chip power supply, *node* 6. The power supply level is adjusted to track the temperature rise. The maximum difference between the control voltage and the chip power supply during the temperature increase is 420 microvolts for this simulation. The resulting deviation of the propagation delay of the inverter delay chain from its design target is negligible.

Simulation results have shown that the adaptive power supply method can compensate for process dependent delay variation and temperature dependent delay variation in CMOS wave pipelined circuits. This method cannot, however, compensate for high frequency delay affecting changes to the operating environment such as Ldi/dt noise and capacitively coupled noise, or spatial variations in the process and operating environment. Any disparity in the tracking of propagation delay between the inverter delay chain and the wave pipeline combinational logic when supply voltage is changed may also introduce some variation in circuit delay. Despite these factors, the adaptive power method is attractive for high performance CMOS wave pipelined designs.

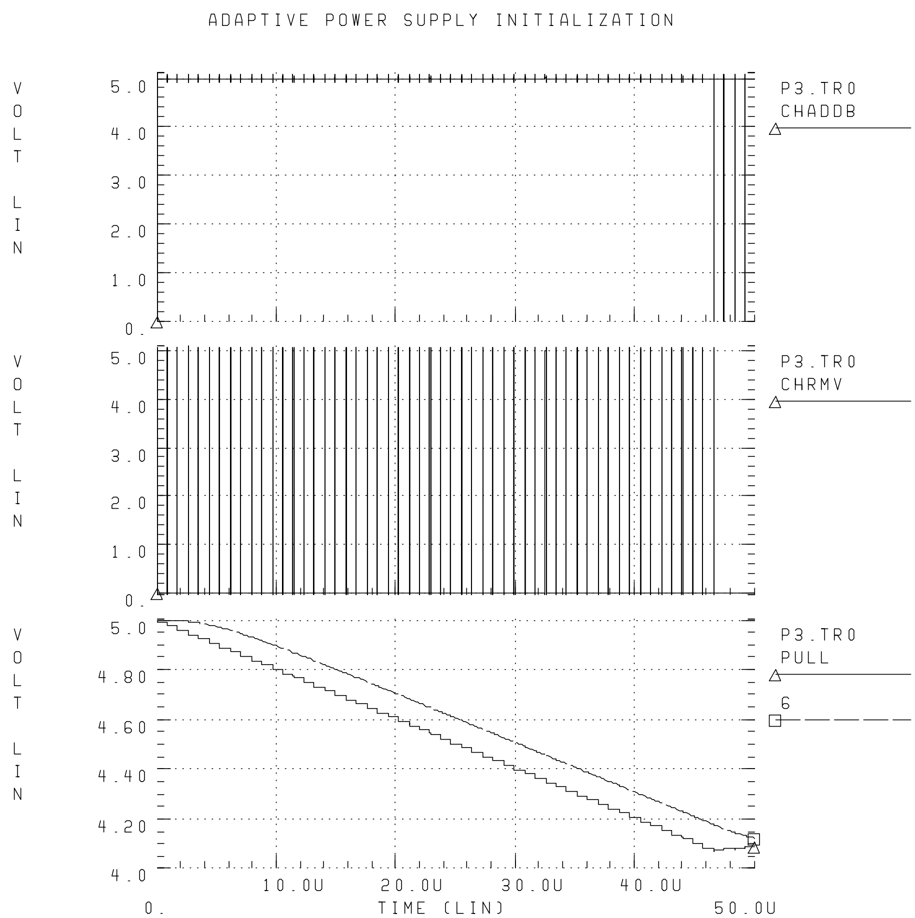


Figure 70: Adaptive Power Initialization

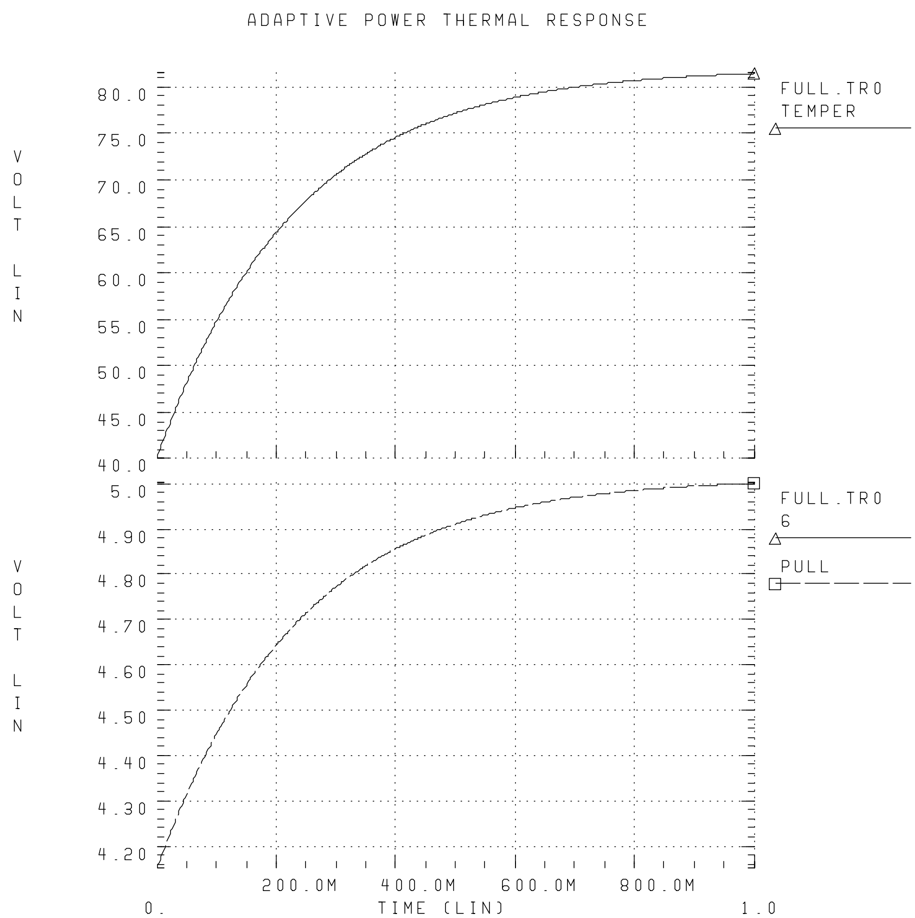


Figure 71: Adaptive Power Step Response

References

[1] S. Anderson, J. Earle, R. Goldschmidt, and D. Powers. The IBM system/360 model 91 floating point execution unit. *IBM Journal of Research and Development*, pages 34–53, January 1967.

[2] H. B. Bakoglu. *Circuits, Interconnections, and Packaging for VLSI.* Addison-Wesley, 1990.

[3] M. Berkelaar and J. Jess. Gate sizing in MOS digital circuits with linear programming. In *Proceedings of the 1990 European Design Automation Conference*, pages 217–21, Glasgow, March 1990.

[4] C. Branson, D. Murray, and S. Sullivan. Integrated pin electronics for a VLSI test system. In *International Test Conference 1988 Proceedings*, pages 23–27, Washington, D.C., September 1988.

[5] R. Brent and H. Kung. A regular layout for parallel adders. *IEEE Transactions on Computers*, C-31:260–264, 1982.

[6] C. Chang, E. Davidson, and K. Sakallah. Using constraint geometry to determine maximum rate pipeline clocking. In *Proceedings of 1992 IEEE/ACM International Conference on Computer-Aided Design*, pages 142–148, Santa Clara, California, November 1992.

[7] J. Chapman. High-performance CMOS based VLSI testers: timing control and compensation. In *Proceedings International Test Conference 1992*, pages 59–67, Baltimore, September 1992.

[8] T. Chappell, B. Chappell, S. Schuster, J. Allan, S. Klepner, R. Joshi, and R. Franch. A 2-ns cycle, 3.8-ns access 512-kb CMOS ECL SRAM with a fully pipelined architecture. *IEEE Journal of Solid-State Circuits*, November 1991.

[9] P. R. K. Chetty. *Switch-mode power supply design.* Tab Professional and Reference Books, 1986.

[10] L. Cotten. Maximum rate pipelined systems. In *Proceeding AFIPS Spring Joint Computer Conference*, pages 581–586, 1969.

[11] M. Dean. *STRiP : a self-timed RISC processor.* PhD thesis, Stanford University, Department of Electrical Engineering, 1992.

[12] B. Ekroot. *Optimization of Pipelined Processors by Insertion of Combinational Logic Delay.* PhD thesis, Stanford University, Department of Electrical Engineering, September 1987.

[13] W. Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, January 1948.

[14] D. Fan, C. Gray, W. Farlow, T. Hughes, W. Liu, and R. Cavin. A CMOS parallel adder using wave pipelining. *MIT Advanced Research in VLSI and Parallel Systems*, pages 147–164, March 1992.

[15] J. Fishburn. Clock skew optimization. *IEEE Transactions on Computers*, 39:945–51, 1990.

[16] J. Fishburn and A. Dunlop. Tilos, a polynomial programming approach to transistor sizing. In *Proceedings of the 1985 IEEE International Conference on Computer-Aided Design*, pages 326–8, 1985.

[17] M. P. Flynn and S. Lidholm. A 1.2-μm CMOS current-controlled oscillator. *IEEE Journal of Solid-State Circuits*, July 1992.

[18] E. Friedman and J. Mulligan Jr. Clock frequency and latency in synchronous systems. *International Journal of Electronics*, 70:930–4, May 1991.

[19] D. Ghosh and S. K. Nandy. a 400 MHz wave-pipelined 8x8-bit multiplier in CMOS technology. In *Proceedings of the International Conference on Computer Design*, pages 189–201, 1993. A slightly more detailed presentation is given in: D. Ghosh and S. K. Nandy. Design and realization of high-performance wave-pipelined 8x8 b multiplier in CMOS technology. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* pages 36-48, March 1995.

[20] L. Glasser and D. Dobberpuhl. *The Design and Analysis of VLSI Circuits.* Addison-Wesley, 1985.

[21] C. T. Gray, W. Liu, and R. K. Cavin III. Timing constraints for wave pipelined systems. Technical Report NCSU-VLSI-92-06, North Carolina State University, Department of Electrical Engineering, December 1992.

[22] K. Hijikata, T. Nagasaki, R. Kurazume, and W. Nakayama. Study on heat transfer from small heating elements in an integrated circuit chip. In *Proceedings of the 3rd ASME/JSME Thermal Engineering Joint Conference*, volume 4, 1991.

[23] M. Horowitz. Personal conversation. Dept. of Electrical Engineering, Stanford University, March 1993.

[24] D. Jeong, G. Borriello, D. Hodges, and R. Katz. Design of PLL-based clock generation circuits. *IEEE Journal of Solid-State Circuits*, pages 255–61, April 1987.

[25] M. Johnson and E. Hudson. A variable delay line PLL for CPU-coprocessor synchronization. *IEEE Journal of Solid-State Circuits*, pages 1218–23, October 1988.

[26] D. Joy and M. Ciesielski. Placement for clock period minimization with multiple wave propagation. In *Proceedings of the 28th Design Automation Conference*, pages 640–643, San Francisco, 1991.

[27] D. Joy and M. Ciesielski. Clock period minimization with wave pipelining. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 461–472, April 1993.

[28] T. Kim, W. Burleson, and M. Ciesielski. Logic restructuring for wave-pipelined circuits. In *International Workshop on Logic Synthesis*, 1993.

[29] E. F. Klass. *Wave Pipelining: Theoretical and Practical Issues in CMOS*. PhD thesis, Delft University of Technology, Department of Electrical Engineering, September 1994.

[30] F. Klass and M. Flynn. Comparative studies of pipelined circuits. Technical Report CSL-TR-93-579, Stanford University, Computer Systems Laboratory, Department of Electrical Engineering, July 1993.

[31] F. Klass, M. Flynn, and A. J. van de Goor. Fast multiplication in VLSI using wave pipelining techniques. *Journal of VLSI Signal Processing*, 7:233–248, 1994.

[32] F. Klass and J. Mulder. CMOS implementation of wave pipelining. Technical Report 1-68340-44(1990)02, Delft University of Technology, Department of Electrical Engineering, December 1990.

[33] W. Lam, R. Brayton, and A. Sangiovanni-Vincentelli. Valid clocking in wavepipelined circuits. In *Proceedings of IEEE Conference on Integrated Circuits Computer Aided Design*, 1992.

[34] C. Lee and A. Palisoc. Real-time thermal design of integrated circuit devices. *IEEE Transactions on Components, Hybrids and Manufacturing Technology*, December 1988.

[35] W. Lien and W. Burleson. Wave-domino logic: Timing analysis and applications. In *Proceedings of TAU92*, 1992. A short version of this work appears in: W. Lien and W. Burleson. Wave-Domino Logic: Timing Analysis and Applications. *Proceedings of IEEE International Symposium on Circuits and Systems,* pages 2949-52, 1992.

[36] P. Macken, M. Degrauwe, M. Van Paemel, and H. Oguey. A voltage reduction technique for digital systems. In *Proceedings of the 1990 IEEE International Solid-State Circuits Conference*, pages 238–9, San Francisco, CA, February 1990.

[37] D. Marple. *Performance optimization of digital VLSI circuits*. PhD thesis, Stanford University, Department of Electrical Engineering, 1987.

[38] Meta-Software. *HSPICE User's Manual: Volume 2 Elements and Models*. Meta-Software, Inc., 1992.

[39] B. Murtagh and M. Saunders. Minos 5.1 user's guide. Technical Report SOL 83-20R, Stanford University, Systems Optimization Laboratory, Dept. of Operations Research, January 1987.

[40] K. Nakamura, S. Kuhara, T. Kimura, M. Takada, H. Suzuki, H. Yoshida, and T. Yamazaki. A 220 MHz pipelined 16 Mb BiCMOS SRAM with PLL proportional self-timing generator. In *Proceedings of the 1994 IEEE International Solid-State Circuits Conference*, pages 258–9, San Francisco, California, February 1994.

[41] V. Nguyen, W. Liu, C. Gray, and R. Cavin. A CMOS signed multiplier using wave pipelining. In *Proceedings of IEEE 1993 Custom Integrated Circuits Conference*, 1993.

[42] L. S. Nielsen, C. Niessen, J. Sparso, and K. van Berkel. Low-power operation using self-timed circuits and adaptive scaling of the supply voltage. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2:391–397, 1994.

[43] K. Nowka and M. Flynn. Environmental limits on the performance of CMOS wave-pipelined circuits. Technical Report CSL-TR-94-600, Stanford University, Computer Systems Laboratory, Dept. of Electrical Engineering, January 1994.

[44] K. Nowka and M. Flynn. Wave pipelining of high performance CMOS static RAM. Technical Report CSL-TR-94-615, Stanford University, Computer Systems Laboratory, Dept. of Electrical Engineering, January 1994.

[45] K. Nowka and M. Flynn. System design using wave-pipelining: A CMOS VLSI vector unit. In *Proceedings of the 1995 IEEE International Conference on Circuits and Systems*, pages 2301–2304, 1995.

[46] A. Sabnis and J. Clemens. Characterization of electron mobility in the inverted <100> silicon surface. *IEDM Technical Digest*, 1979.

[47] M. Santoro. *Design and clocking of VLSI multipliers*. PhD thesis, Stanford University, Department of Electrical Engineering, 1990.

[48] S. Sapatnekar, V. Rao, and P. Vaidya. A convex optimization approach to transistor sizing for CMOS circuits. In *Proceedings of the 1991 IEEE International Conference on Computer-Aided Design*, pages 482–5, Santa Clara, CA, November 1991.

[49] The MOSIS Service. Mosis parametric test results. 1993.

[50] M. Shoji. *CMOS Digital Circuit Technology*. Prentice Hall, 1988.

[51] I. Sutherland. Micropipelines. *Communications of the ACM*, pages 720–38, June 1989.

[52] S. Tachibana, H. Higuchi, K. Takasugi, K. Sasaki, T. Yamanaka, and Y. Nakagome. A 2.6-ns wave-pipelined CMOS SRAM with dual-sensing-latch. In *Proceedings of the 1994 Symposium on VLSI Circuits*, pages 117–8, Honolulu, HI, June 1994.

[53] D. Wong. *Techniques for Designing High Performance Digital Circuits Using Wave Pipelining*. PhD thesis, Stanford University, Department of Electrical Engineering, 1991.

[54] D. Wong, G. De Micheli, and M. Flynn. A bipolar population counter using wave pipelining to achieve 2.5x normal clock frequency. In *Proceedings of IEEE International Solid-State Circuits Conference*, San Francisco, February 1992.

[55] X. Zhang and R. Sridhar. CMOS wave pipelining using transmission-gate logic. In *Proceedings of Seventh Annual IEEE International ASIC Conference and Exhibit*, Rochester, NY, September 1994.