High Bandwidth Interchip Communication for Regular Networks

by

Rajeevan Amirtharajah

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Electrical Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 1994

Author_____
Department of Electrical Engineering and Computer Science
May 6, 1994

Certified by_____
Thomas F. Knight, Jr.
Thesis Supervisor

Accepted by_____
Frederic R. Morgenthaler
Chairman, Department Committee on Graduate Theses

High Bandwidth Interchip Communication for Regular Networks

by

Rajeevan Amirtharajah

Submitted to the
Department of Electrical Engineering and Computer Science

May 6, 1994

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

As integrated circuit processes continue to scale downward and clock frequencies of VLSI systems increase, chip to chip communication can become a bottleneck limiting the performance of digital systems. This thesis presents circuits that achieve high performance I/O for the Abacus massively parallel SIMD computer. The Abacus is a globally synchronous machine with a two dimensional mesh network. Every Abacus processor has 64 dedicated pins for interchip communication, each pin requiring a bandwidth of 250 MBit/s. The total bandwidth requirement is 16 GBit/s. This performance is achieved by a variety of circuit techniques: impedance matching drivers, data retiming, and delay-locked loops for clock generation. The regularity of the mesh network is exploited to reduce the cost and complexity of high performance interconnect.

Thesis Supervisor: Thomas F. Knight, Jr.
Title: Principal Research Scientist, MIT AI Lab

High Bandwidth Interchip Communication for Regular Networks

by

Rajeevan Amirtharajah

Submitted to the
Department of Electrical Engineering and Computer Science

May 6, 1994

In Partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Electrical Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

Abstract

As integrated circuit processes continue to scale downward and clock frequencies of VLSI systems increase, chip to chip communication can become a bottleneck limiting the performance of digital systems. This thesis presents circuits that achieve high performance I/O for the Abacus massively parallel SIMD computer. The Abacus is a globally synchronous machine with a two dimensional mesh network. Every Abacus processor has 64 dedicated pins for interchip communication, each pin requiring a bandwidth of 250 MBit/s. The total bandwidth requirement is 16 GBit/s. This performance is achieved by a variety of circuit techniques: impedance matching drivers, data retiming, and delay-locked loops for clock generation. The regularity of the mesh network is exploited to reduce the cost and complexity of high performance interconnect.

Thesis Supervisor: Thomas F. Knight, Jr.
Title: Principal Research Scientist, MIT AI Lab

For Mom and Dad,

who did the best they could under the circumstances.

Acknowledgments

Thanks to the three people most responsible for the ideas and learning represented in this thesis: Mike Bolotski, for giving me the opportunity to work on such an exciting and challenging project as Abacus, Tom Simon, for going above and beyond in helping me with the circuit design, and Tom Knight, from whom I've learned more than from anyone else at MIT. Thanks to Tim Kutscha and Fred Drenckhahn for answering many questions.

I am grateful to all the people who've made a difference to me over the last four years: Mike Halle, Steve Benton, and the Spatial Imaging Group at the Media Lab, Kevin Wrenner and Kai Toh at the IBM T.J. Watson Research Center, Anne Hunter, the Mao family, and all my friends.

Special thanks to my friends who survived the thesis experience with me: Will, Pete, and Chris [12, 14].

Word to my boyz Fresh Billy Funkalicious [13] and Low Rider B.

I couldn't have done it without the love and support of my parents, Uma and Appiah Amirtharajah. Thanks for everything.

Thanks to my sister, Mohana Amirtharajah, for being there and being cool.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Current trends in VLSI are increasing the demand for high performance signalling between chips. As chip integration and circuit speeds increase, interchip I/O bandwidth becomes a bottleneck limiting the performance of high speed digital systems. Device switching frequency continues to increase. The number of transistors on a die increases quadratically with shrinking device dimensions. Standard bonding and packaging constrains I/O pads to the perimeter of the chip. Consequently, the number of I/O pads that can be placed on a die scales only linearly with the dimensions of the die. Pad pitch is limited by the ability of machinery to mechanically bond wires to them while on chip wire pitch is limited by processing. I/O switching speed is limited by pad and packaging parasitics, and consequently is not keeping pace with increasing clock rates. The overall effect is that increases in logic switching speeds are outpacing increases in I/O bandwidth. It becomes more important to overall performance to achieve high bandwidth I/O on each pin [2].

The Abacus computer is an example of a digital system for which efficient interchip communication is necessary for performance [1]. Achieving this performance is by no means straightforward. Tradeoffs between bandwidth, pinout, and power consumption complicate the design process. Noise and process variation must be compensated for by careful circuit design. Timing issues must be addressed to ensure reliable data transmission. Moreover, all this must be done while keeping cost and complexity at manageable levels.

This thesis explores an approach to designing high performance interconnect for networks with regularity. The approach is simple: adapt techniques developed for more complex net-

works to regular ones by taking advantage of uniformity in transmission media and synchronization in time. Regularity in networks allows the performance to be achieved with reduced cost and complexity.

Chapter 2 discusses features of the Abacus network in particular that are relevant to designing high performance interconnect. It also describes the problems involved with high performance signalling in the Abacus system. Chapter 3 discusses the design of an impedance matching bidirectional I/O pad for the Abacus system. Timing and synchronization issues are covered in Chapter 4. Chapter 5 discusses the design of circuitry that generates waveforms needed to implement the Abacus data retiming scheme. Finally, Chapter 6 sums up the important issues in designing for performance for the Abacus network.

Chapter 2

High Bandwidth Signalling in the Abacus System

This chapter describes the interchip I/O network for the Abacus computer. Issues important in designing high performance interconnect and how they are relevant to the Abacus are also discussed.

2.1 Abacus Interchip Communication Network

Abacus is a high performance computer for early vision applications [1]. It is a massively parallel single instruction, multiple data stream (SIMD) architecture. Its processing elements are connected via a two-dimensional mesh network. Each processor receives a globally distributed low-skew system clock. This regular, synchronous structure will be exploited to reduce the cost and complexity of circuit techniques for achieving good performance.

2.1.1 Architecture

Each Abacus processor chip integrates 1024 two bit processing elements. Each element on a chip is connected to its four nearest neightbors by a two dimensional mesh network. The Abacus system contains 256 processor chips, also wired in a two dimensional mesh. Part of this mesh is shown in Figure 2-1. The north edge of the mesh of chips is wired to the south edge and the west edge connects to the east edge so the entire network forms a torus. Every

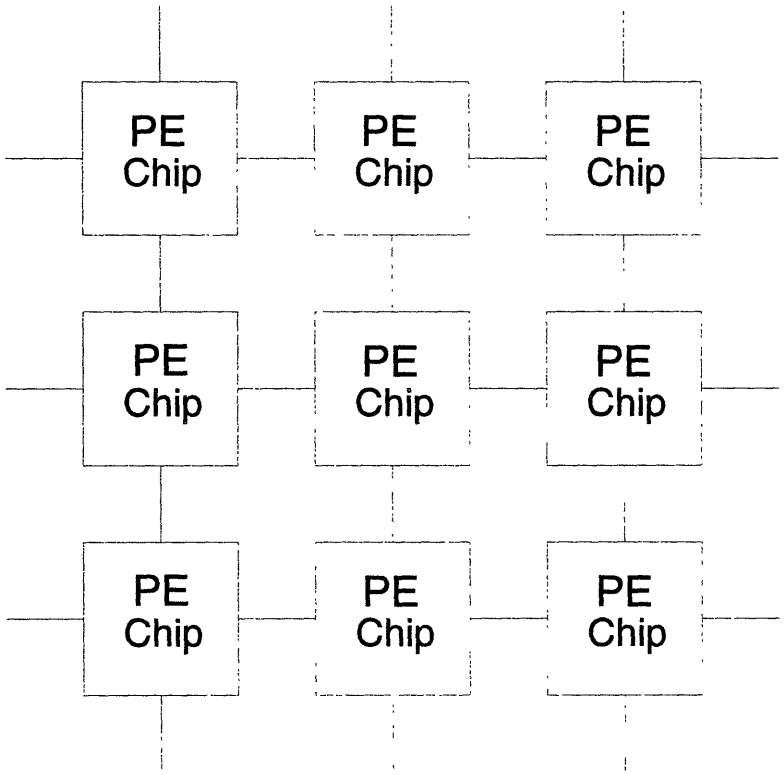

Figure 2-1: Interchip Communication Network.

two bit processing element is a node in the torus. On chip, the delays between elements are very small, and communication can take place at high data rates. However, elements on the edges of the on chip array must communicate off chip as well.

Abacus is an example of how chip to chip I/O can potentially limit the performance of a high speed digital system. The chip core is very dense since it contains so many processing elements, each with two register files, two ALUs, and networking logic. The system has a high clock frequency of 125 MHz. Abacus has high bandwidth requirements. Addressing the register files and specifying the opcodes requires a 50 bit wide instruction word. Instructions arrive every processor cycle. External memory and I/O must also connect to the chip. These ports are in addition to the communication port between chips. Even though only the processing elements on the four edges of the on chip array connect to other chips, the interface would still require 128 pins if each element had a dedicated bidirectional pin.

Meanwhile, chip I/Os are limited to the perimeter of the die since a standard bonding

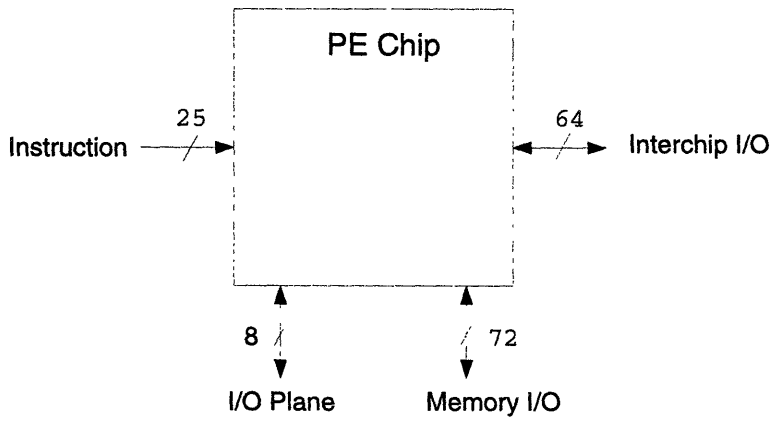

Figure 2-2: Abacus Processor Chip I/O Architecture.

and packaging process is used. This constrains the number of available I/Os. It is packaged in a land grid array, where parasitics associated with different I/Os can vary. For example, a signal that must be routed to one of the corner pins traverses a longer distance than one that goes to an inner pin. This can introduce skew between nominally in phase signals. Moreover, the increased parasitic inductance of the long line may limit the bandwidth of that I/O. These factors make some of the available I/Os unsuitable for high bandwidth communication. One must achieve high bandwidth from the remaining pins to prevent the core logic from idling while waiting for data.

The I/O architecture of the Abacus processor is shown in Figure 2-2. The number of wires for the instruction and interchip communication ports is reduced by a factor of two because of two to one multiplexing. These ports can be designed for performance since they interface to either commercially available high performance components or to custom chips. The external memory port needs many more wires. Its performance is limited by the commodity parts which must be used for economic reasons. Fortunately, pins with poor parasitics can be allocated to the low performance interfaces.

Table 2.1 lists the parameters of the Abacus interchip communication network. The important points are the high bandwidth requirement and the bounded delays. One advantage of the mesh topology of the network is that bounds can be set on the chip to chip delays. In Chapter 4, these bounds are shown to help simplify data retiming. This and other advantages

System Clock	125 MHz
Minimum Interchip Delay	2 ns
Maximum Interchip Delay	6 ns
Multiplexing	2 to 1
Interchip I/O Port Physical Width	16 wires
Interchip I/O Port Bandwidth	250 MBit/s

Table 2.1: Abacus Network Parameters.

of the Abacus network are discussed in the next section.

2.1.2 Regularity of the Network

It is interesting to contrast the Abacus network with more general digital communications networks. The Abacus network is a two dimensional mesh that limits communication to between nearest neighbors. The interchip I/O ports are several wires wide. These wires can travel in a straight line since the processor chips are mounted in a regular array on a printed circuit board. They can also travel together since they only go short distances. This means that the wires don't have to be split up and routed around other packages or signals, which would introduce skew between them. The physical environment of the printed circuit board has relatively tight manufacturing tolerances, so the characteristics of the transmission lines tend to be similar. Tightly specified connectors and cables used for board to board hops also present a uniform transmission environment. All of these factors result in a single I/O port having a uniform transmission medium for each wire and a uniform timing relationship between signals. A uniform medium makes matching impedances the same for all the wires. This matching is also pretty static over time, as the medium is physically fixed (the variation being with temperature). The synchronous operation of the system ensures that the timing relationship between the transmitter and receiver is very similar. The similar phase of the signals makes retiming cheaper since retiming circuitry for one signal can be used for all signals on the same port. Synchronous operation and bounded delays reduces retiming complexity, as shown in Chapter 4.

A much more general network, like a telecommunications system, has more complexity. In that system transmitters and receivers can be fully asynchronous and differ with each

other in frequency and phase. The timing relationship may vary with time, for example as a satellite moves in orbit. The transmission medium may vary from copper wire to fiber optic line to microwaves, making different hardware necessary for each medium. Even in digital computers, where there is much regularity in transmission media, wires often travel together, and operation is usually fully synchronous, there is frequently more complexity than in Abacus. Many multiprocessor systems have a multistage routing network that connects any processor to any other in the machine. This requires routing components and protocols. Abacus only has neighbor to neighbor communication which is implemented with direct wiring.

The regularity of the Abacus network allows high performance communication to be implemented using techniques adapted from more general networks. The adaptation reduces some cost and complexity. As an example of this reduction, the Abacus retiming circuitry is compared to the implementation in the Transit system [10].

2.2 Issues in High Performance Signalling

High performance I/O is necessary in state-of-the-art digital systems like Abacus. Communication time must be minimized to prevent processors from having to stall for lack of data. However, there are several issues that complicate designing I/Os for performance.

2.2.1 Latency

Latency is the delay required for a bit to travel from transmitter to receiver. In Abacus, this delay must be minimized to reduce communication time between processors. Latency is determined by physical parameters like the speed of light and circuit speeds. Although architectural and algorithmic techniques can help mask latency, a designer has limited options in reducing latency at the circuit or board level. Shortening the media through which data travels reduces the effect of a signal's finite propagation velocity. Since circuits also have finite delay, minimizing the number of circuit elements in the propagation path reduces latency. An example of this approach is reducing the number of inverters in an off chip buffer. One has to live with a certain amount of latency since systems cannot be made arbitrarily small.

Since the Abacus network is neighbor to neighbor, latency is minimized by packaging

processor chips physically close to each other on the same printed circuit board. The only long wires connect the edges of the torus and must be routed from board to board. Their length can be decreased by packaging the boards closely in a single cabinet. Of course, some care is also taken to minimize the delay of circuits in the critical path.

2.2.2 Bandwidth

The *bandwidth* of a communications channel is the number of bits that can be transmitted over it in a fixed time. Bandwidth can be increased by increasing the number of lines in a channel and reducing the time duration of bits (the bit cell time). Bandwidth is limited by physical effects. Finite resistances and capacitances imply a bandwidth limit in the circuits that drive and receive data on the channel. The amount of noise on a channel also reduces the available bandwidth [5]. Reducing parasitics and noise therefore improves bandwidth, but these can never be completely eliminated. The only way to increase bandwidth arbitrarily is to add more wires and drive up the cost of the system.

Abacus achieves high bandwidth by reducing the bit cell time and transmitting two bits per processor cycle. Transmitter and receiver circuits are designed with bandwidth mind. Noise is reduced by handling transmission line effects (see Section 2.2.4.

2.2.3 Power

Switching circuits rapidly is important for minimizing latency and increasing bandwidth. However, large amounts of power can be dissipated by doing this. In CMOS logic, which dissipates no static power, dynamic power increases as the frequency of operation increases [11]. Dynamic power dissipation in logic equals CV^2f. Charging and discharging the output driver's load also dissipates power as V^2/R. Power can be reduced quadratically by reducing the voltage swing of the circuits.

Lowering this swing also reduces the noise margins available to the receiver. Therefore, the swing should only be reduced until the power dissipated in the output circuitry is small compared to the power dissipated in the chip core. In Abacus, an output signalling level of 1 Volt was chosen.

2.2.4 Noise

A major obstacle in high performance signalling is noise. Most noise in digital systems is self-induced. Noise is induced in on-chip power supply networks due to the finite resistance and inductance of wires and pins. As off-chip drivers and logic switch, they draw current from the rails that generates parasitic voltages in the form of iR and Ldi/dt drops. Noise on the power supplies affects the delays of circuits and can introduce errors in receivers that are referenced to the supplies.

Power supply noise can be reduced by the addition of bypass capacitors and shaping the transitions of drivers to reduce di/dt. However, both of these alternatives are unattractive in the Abacus system. Because of the low output levels, adding bypass capacitance between the 1 Volt rails is undesirable. Two approaches can be used in implementing the on chip bypass capacitance. The capacitor could be constructed using metal, but this consumes a great deal of area because the thickness of the field oxide results in a low capacitance per unit area. Using a transistor as the capacitor is another approach. This requires less area since the transistor's thin gate oxide yields a higher unit capacitance. However, large swing noise from the 5 Volt supplies could couple into the 1 Volt supply through the gate-, source-, and drain-to-substrate parasitics of the transistor because the substrate is tied to the 5 Volt ground. This extra noise could easily disrupt chip to chip communication with its reduced noise margins. Shaping the output waveform is undesirable because it reduces the bandwidth of the channel. Instead, the 1 Volt signalling supplies are brought into the die on their own pins to decouple them from the 5 Volt noise of the rest of the chip.

At the board level, noise results from transmission line effects. When the transition times of off-chip drivers become comparable to the time-of-flight of signals along a trace, transmission line effects become significant. Impedance discontinuities along the line produce reflections that can corrupt transmitted data. Data lines that travel together couple both capactively and inductively, generating crosstalk between them. Reflections can be reduced by minimizing impedance discontinuities and adequately terminating the lines. Increasing the distance between adjacent wires decreases crosstalk at the expense of possibly introducing skew between the signals. Inserting ground lines between wires reduces crosstalk but reduces the density of signals on the board.

In Abacus, reflections are reduced using an impedance matched driver to series terminate the transmission lines (Chapter 3). Crosstalk is not a severe problem in wires that run together for short distances, like the buses that connect processor chips on the same board. Wires that go from board to board are isolated from each other by ground lines.

2.2.5 Tradeoffs

Achieving high performance communication in a digital system requires careful engineering. Noise and latency must be minimized by circuit and board design while keeping complexity at a manageable level. Power can be minimized but at the expense of reduced noise margins. There is a tradeoff between pinout and bandwidth that must be optimized for a particular set of cost and performance requirements. All of these issues must be carefully considered in designing chip to chip communication for high performance.

Chapter 3

Interchip Communication Pads

The high bandwidth interchip communication pads for Abacus are bidirectional, requiring the integration of a driver and receiver at each pad. This reduces the required pinout for a single chip to chip communication port by a factor of two over having dedicated input and output pins. A block diagram of a single pad appears in Figure 3-1. In addition to the driver and receiver, the pad incorporates two blocks of synchronization circuitry and a prebuffer for the output. This chapter reviews the idea of the impedance matching driver and describes an implementation. It also discusses the prebuffer, the receiver circuit and the configuration system for the impedance matching driver. The retiming circuitry is discussed along with the synchronization scheme in the following chapter.

3.1 Series Termination of Transmission Lines

Transmission line effects become important when the rise time of driver outputs is comparable to the time-of-flight of signals on a wire. A major component of self-induced noise in high performance I/O systems is reflections due to these effects. Reflections on a transmission line occur when an incident wave encounters an impedance that is not the characteristic impedance of the line. In point to point wiring, as in the Abacus interchip communication network, these impedance discontinuities appear at the driver and load ends of the line. On a bus, discontinuities can appear at any intermediate point where a chip connects to it.

To simplify the discussion, assume point to point wiring terminated at the source and load

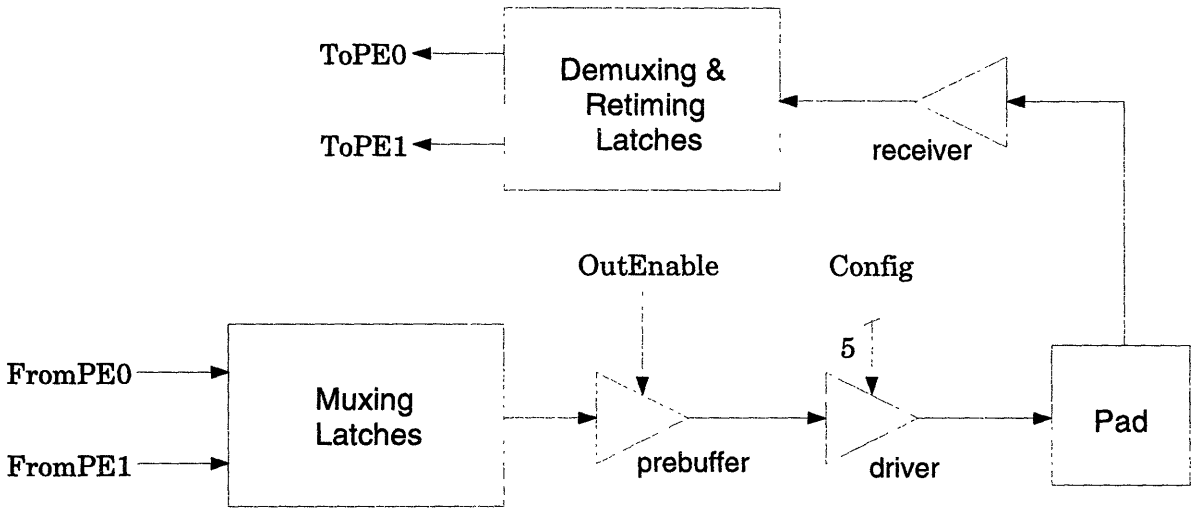

Figure 3-1: Interchip Pad Block Diagram.

ends with resistors. If these resistances differ from the charactersitic impedance of the transmission line, signals incident on them generate reflections. The reflected wave is proportional to the incident wave by a factor Γ, known as the reflection coefficient. Γ is related to the termination resistance R and the characteristic impedance of the line, Z_0, by:

$$\Gamma = \frac{R - Z_0}{R + Z_0} \tag{3.1}$$

Let the resistance at the source and load ends be R_S and R_L, respectively. The corresponding reflection coefficients are Γ_S and Γ_L. A transmission line is said to be series terminated when the resistance at the source end equals the characteristic impedance of the line and the load is an open circuit. This condition implies by Equation 3.1 that $\Gamma_S = 0$ and $\Gamma_L = 1$.

Consider what happens when a series terminated transmission line, originally at rest, is driven high. The resistance at the source and the impedance of the line form a resistive divider, and the waveform at the source end rises to half its final value. This wave propagates down the line until it reaches the load, where it is reflected at the same amplitude and polarity. The incident and reflected waves superpose, so the voltage at the load rises immediately to its final value. When the reflected wave reaches the source, the driver output also rises to the final value. No more waves are reflected at this point since $\Gamma_S = 0$.

Series termination has several aspects that make it attractive for use in a CMOS envi-

ronment. No power is dissipated holding the output line high; the only power consumption occurs dynamically as the driver switches. Abacus is designed to process images, which often have uniform regions. As these regions are communicated around the array, the drivers do not switch much. Dynamic power in the communication circuits will not necessarily be large in this system, so series termination helps reduce total power consumption. Also, the high on resistance of MOS transistors makes it area efficient to design high impedance drivers rather than low impedance ones [4]. In particular, the series terminating resistor can be "folded" into the CMOS driver by designing the driver to have an on resistance equal to Z_0. A parallel terminated line in which $R_L = Z_0$ and R_S is made small would require a very wide output driver to get the source resistance low enough.

3.2 Transmitter Design

The transmitter portion of the bidirectional pad consists of an impedance matching driver driven by a prebuffer.

3.2.1 Impedance Matching Driver

Accurate termination requires fabrication of output drivers that are closely matched to the characteristic impedance of the interchip communication lines. The straightforward approach would be to simply size the output drive transistor to match this impedance. Unfortunately, processing, power supply, and temperature variations all affect the on resistance of MOS transistors. Because of manufacturing tolerances of printed circuit boards, the characteristic impedance of the driven line may also vary. Matching impedances across these variations requires a more sophisticated approach.

Figure 3-2 shows a configurable impedance matching driver. The basic principle is simple: by turning on more transistors in the pullup or pulldown path, the impedance of the path can be varied under digital control. Note that the pullup and pulldown networks are symmetric since empirically there is not a significant impedance difference between pulling down to ground or up to the one volt power supply. Consider the pullup tree in Figure 3-2. Transistors M0 and M2 are turned on whenever the high output is driven onto the pad. The

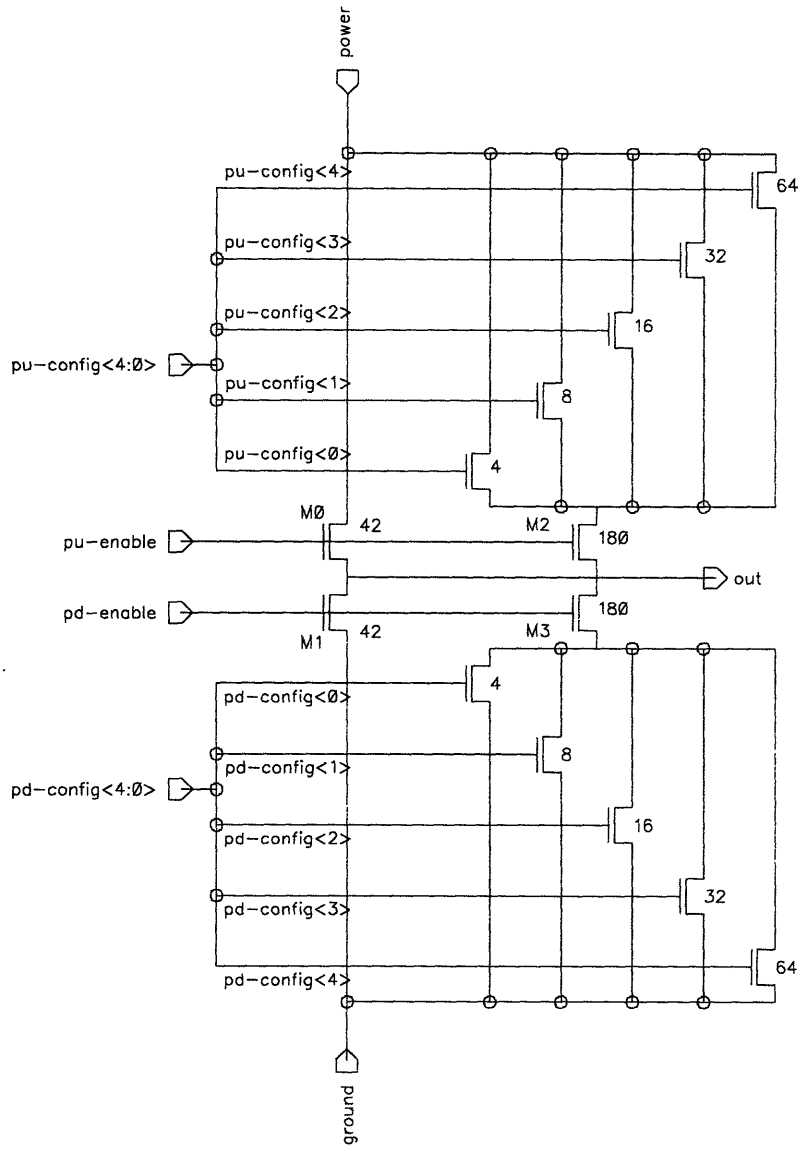

Figure 3-2: Interchip Driver.

other transistors in the pullup chain are set by configuration bits pu-config$\langle 4:0 \rangle$. As these transistors are enabled, the impedance of the pullup chain varies to match the impedance of the load [3]. The configuration bits are set by a finite state machine described below.

The transistors of both chains are sized to match the widest range of process, environment, and load variations. M0 is sized such that at the fast process and environment corner, the impedance of the driver matches the highest possible impedance of the output load (in this case, 60 ohms for the printed circuit board trace) when all of the other pullups are disabled. M2 and the configurable devices are sized such that at the slow corner the driver matches the lowest impedance of the output load (40 ohms). The number of configurable devices

is affected both by the desired resolution and the desired range. Achieving a resolution of about 5 ohms/bit between 40 and 60 ohms would require only three bits if transistors had a guaranteed resistance. However, corner changes result in a factor of two shift in the impedance of a transistor. More bits and devices are required to give accurate resolution and provide enough range over the entire process variation. In this case, five bits are necessary. In the fast corner, the widest transistors will never be turned on because their impedances are too low. In the slow corner, they will always be turned on just to lower the driver impedance into the 40 to 60 ohm range.

The exponential sizing of the configurable pullup devices does not result in constant impedance resolution, but the worst case shifts are around 5 ohms and are therefore satisfactory for this application. M2 is also sized larger than the full width of the driver when all configurable devices are enabled. This ensures that the output node "sees" the impedance shifts more clearly by providing low series impedance to the bank of configurable devices.

3.2.2 Prebuffer

The prebuffer scales up the signals from the output timing block to drive the relatively large load of the impedance matching driver. It consists of two scaled inverter chains, each of which is preceded by a NAND gate which can tristate the driver when the pad is in receiver mode. The scaleup factor (about 2.56 in this case) in the chains is chosen to be less then e, the latency optimal factor, to improve the bandwidth of the prebuffer. This ensures that the bandwidth of interchip communication is limited by package parasitics rather than circuits. The relatively large cycle time of 8 ns makes the extra latency tolerable.

3.3 Receiver Design

Using a standard CMOS inverter as a receiver is not practical with low voltage signalling. It is difficult to design an inverter with an appropriately low threshold that is robust to power supply and process variation. A differential amplifier can be used instead to amplify the input to reasonable CMOS levels. Feedback techniques also make the amplifier more robust to variations.

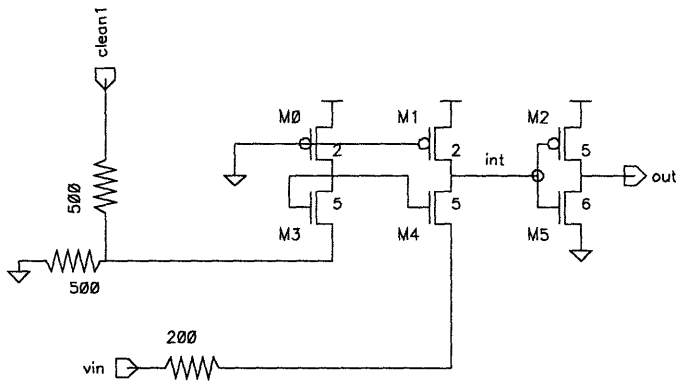

Figure 3-3: Interchip Receiver.

The interchip receiver is shown in Figure 3-3. It consists of a differential amplifier feeding a misratioed inverter. The differential amplifier in turn consists of two common gate amplifiers. A reference voltage of 500 mV is generated by the voltage divider formed by the two 500 ohm resistors. Pfet M0 has its gate tied to ground to act as a current source. Nfet M3 has its source connected to the reference voltage and its drain tied back to the gate. It is wired as a source follower with its output connected to its biasing voltage. Together, transistors M0 and M3 form a common gate amplifier with feedback. Its output is stable at the treshold of the amplifier. This output goes to bias the source follower M4 in the next stage. M0 and M3 form a replica biasing pair for M1 and M4 that biases the second stage at its threshold. The replica tracks M1 and M4 with process, so the biasing is robust to process variation. Small swings on V_{in} are amplified into large swings on the inputs of the inverter formed by M2 and M5, whose output has a full 5 V CMOS swing.

The values of the resistors forming the voltage divider were chosen such that the input impedance seen at M3 matches the input impedance seen at M4. M4's input impedance is set by the 50 ohm transmission line PCB trace in series with the 200 ohm ESD protection resistor. Matching these impedances ensures that the currents in both halves of the differential amplifier are balanced. In addition, these values will track each other somewhat across process variation.

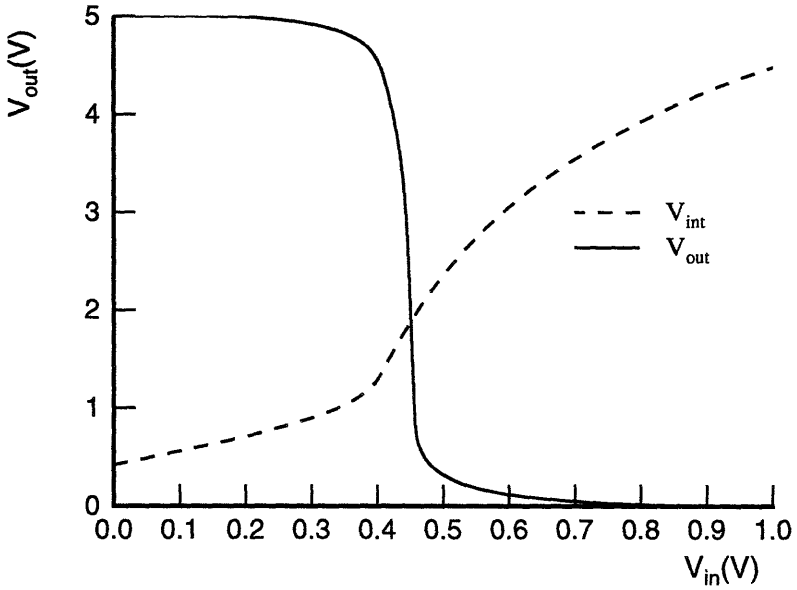

Figure 3-4: Interchip Receiver Noise Margins.

The final stage of the receiver is a misratioed inverter. M2 and M5 are sized such that the inverter treshold is about 500 mV above V_{tn}. This puts the threshold of the entire receiver slightly below 500 mV. Setting the threshold thus enables the impedance testing finite state machine to use the receiver to measure the voltage at the output of the impedance matching driver, at the cost of reduced noise margins for V_{IL}. The operation of this state machine is described in the next section.

Figure 3-4 shows the V_{in} versus V_{out} characteristic of the receiver at the nominal corner. The dashed line represents the output of the differential pair, i.e. the node labeled **int** in Figure 3-3. The solid line represents the voltage on the output of the receiver. Note that the receiver threshold is slightly under 0.5 Volts, which reduces the V_{IL} noise margin slightly to 400 mV. The worst case noise margin occurs for the slow nfet-fast pfet process corner and was simulated to be 354 mV.

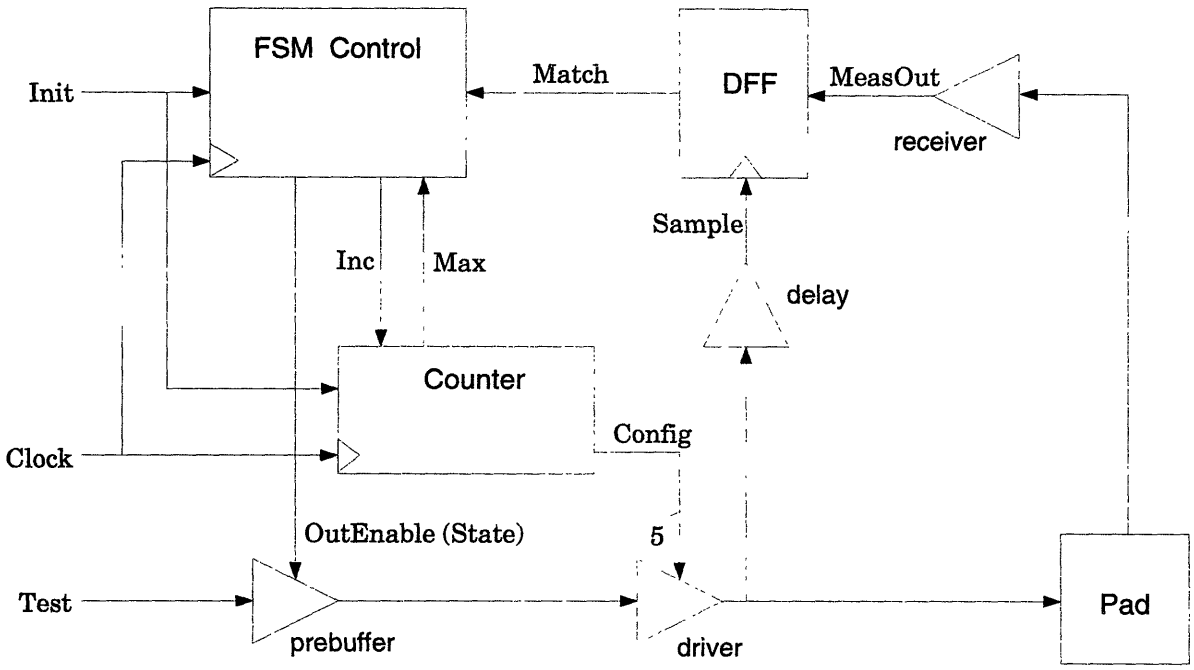

Figure 3-5: Impedance Test FSM.

3.4 Impedance Test System

To obtain a good impedance match for the drivers across tolerances and process variations, a finite state machine tests successive driver configurations until it finds a close match. This configuration is then broadcast to all the drivers on the same side of the die as the test fsm. The extra cost of the impedance testing circuitry is amortized over many interchip communication lines. Process varies little on a single die. It is adequate to have only one test structure per edge since all drivers on that edge have the same characteristics.

Figure 3-5 shows a block diagram of the impedance testing system. The operation of the FSM is straightforward. Assume it starts in an idle state, i.e. the **Inc** and **OutEnable** signals are deasserted, and the five **Config** bits are stable. When the **Init** signal is asserted, the FSM state is changed to the testing state and the configuration bits are cleared. In the testing state, the machine asserts **OutEnable** and the **Inc** signals. **OutEnable** enables the **Test** waveform to be driven out onto the pad and the test line. **Inc** allows the counter to start counting upward. However, neither the counter nor the FSM changes state until the reset is

deasserted. When **Init** falls, the controller starts paying attention to the **Match** and **Max** signals.

Suppose the pad and output transmission line has some initial voltage state, in this case either 0 or 1 Volt. The impedance of the driver matches the line when the voltage on the line changes to half its final value as the driver is switched. In this implementation, the FSM monitors the pullup waveform. **Test** is a periodic half-duty cycle waveform that is driven onto the pad when **OutEnable** is asserted. Initially the output line is low. When the rising edge of **Test** passes through the driver, the voltage on the pad rises to some intermediate voltage for the round-trip delay time of the output line. If this voltage is below the threshold of the receiver, the **MeasOut** signal is unasserted for this duration. **MeasOut** is sampled at this intermediate voltage by a positive edge-triggered flip-flop, DFF. **Match**, the sampled version of **MeasOut**, is fed to the FSM control. As long as it is unasserted, the machine remains in the testing state. More test signals are driven out and the counter increments, each time reducing the output impedance of the driver, until **Match** is finally asserted. The impedance of the driver has then been matched beacause the initial output voltage has just crossed the threshold of the receiver, which from the previous section is set just below 0.5 Volts. The FSM goes back to the idle state, deasserting **OutEnable** and **Inc**, and the new **Config** bits can be used by the other drivers.

There are several timing issues in this design. There must be sufficient delay between **Clock** and **Test** to ensure that the counter outputs are valid when the testing signal is driven. The period of **Clock** was chosen to be long enough to allow reflections on the output line to settle so that reflections from previous cycles do not affect the current measurement. This is acceptable from a performance standpoint since the FSM will only be activated at infrequent, software controlled intervals (for example at boot time). The most important timing relationship exists between **Sample** and **MeasOut**, the output of the receiver. **Sample** must latch in **MeasOut** while the pad output is at an intermediate state. Ensuring a valid measurement is latched requires **Sample** to be long enough to account for the delay through the driver and the receiver plus the setup time of the latch. **Sample** must also be shorter than the round-trip time of the delay line so that an intermediate value measurement is latched rather than a fully valid one. It is possible for the output of the flip-flop to be metastable

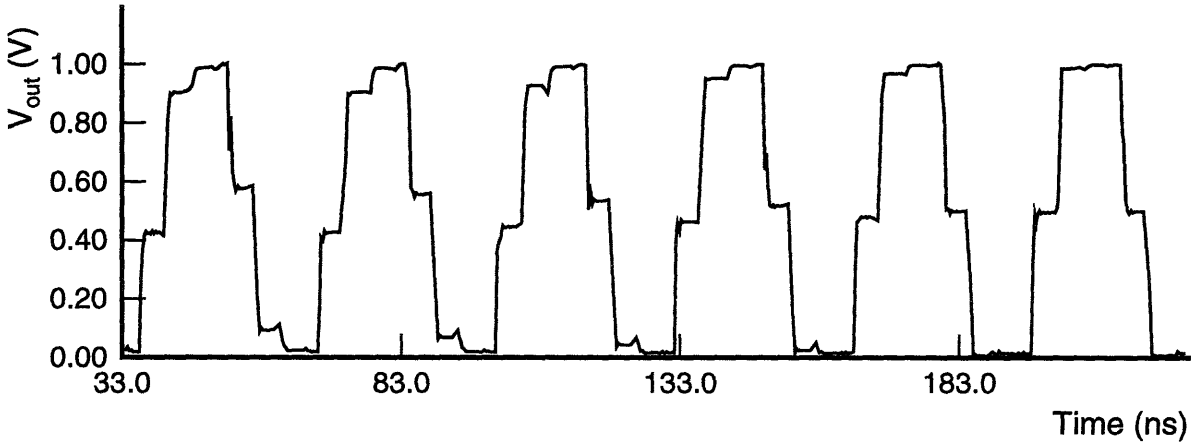

Figure 3-6: Impedance Matching.

if **MeasOut** is in the middle of its swing due to the intermediate voltage at its input. To account for this, three stages of inverters are placed after the flip-flop to amplify noise into a valid decision. The output of the third inverter is taken to be **Match**.

3.5 Simulation Results

Correct matching operation of the impedance testing FSM is demonstrated in Figure 3-6. The figure shows the waveform at the output pad as **Test** goes through several cycles. Note that the intermediate voltage rises as the **Config** bits are incremented, and that the reflections are reduced until they are no longer signficant compared to the output swing.

The impedance matching driver's range extends from at least 39.7 Ohms up to 60.6 Ohms accounting for process and power supply variation. The worst case resolution is 6 Ohms, which occurs at the high end of this range at the fast process corner. Consequently, the source impedance can be matched to the load within 10 percent.

Chapter 4

Synchronization of Interchip Communication

This chapter describes the timing scheme for the interchip communication system. The regularity of the network enables some simplifying assumptions that reduce the complexity of the timing system. Circuits that accomplish this retiming are also described.

4.1 Data Retiming in General versus Regular Networks

In high bandwidth communication networks, a situation often arises in which the duration of a transmitted bit becomes comparable to or less than the propagation delay of a bit along a communication channel. When this happens, the data arrives at the receiver with a phase offset with respect to the receiver's local clock. In order for the receiver to process the data, it must be *retimed* to the local clock. Retiming simply means that the timing relationship between transmitted data and clock is modified. The new relationship guarantees that computations performed at the receiver on the data are valid when sampled by the local clock.

Data retiming in a general, asynchronous network is complex. In such a network, a chip can be connected to many other chips over lines of unknown length. Each chip has its own locally generated clock whose frequency and phase is independent of all the other chips. Moreover, the frequency and phase relationship between transmitter and receiver may vary with time. The propagation delay between two points is unknown. In addition, each communication line

may have its own unique set of frequency, phase, and delay characteristics. A receiver in such an environment must be able to recover the phase of incoming data in a manner that is robust to all these problems. Traditionally, this would require a phase-locked loop or digital phase adjustment system at each receiver [9]. A computer network linking machines of different kinds over large distances is an example of such a system.

Matters are greatly simplified in a regular, synchronous network. Processor chips in a parallel computer often communicate through such a network. Each transmitter and receiver has a local version of a globally distributed clock. Care is taken to ensure a minimum skew between versions of this clock at different chips. Coupling chip to chip communication to this clock yields several advantages in data retiming. First, communication occurs at a known frequency (i.e. the frequency of the global clock or some multiple thereof). Second, the phase behavior of communication is identical at both transmitter and receiver. Although the phase of the global clock may jitter or drift with time, the transmitter and receiver move with it in the same manner. Regularity of interconnect allows bounds to be set on the chip to chip delay. For example, in a mesh network where chips communicate directly only with their nearest neighbors, an upper bound on delay is determined by the sum of the longest wire propagation delay and worst case circuit delays. Lastly, many lines have the same characteristics since they are driven by transmitters on the same chip, synchronized to the same clock, and travel the same distance in the same medium.

All of the above conditions help reduce the complexity of data retiming. Given certain bounds on delay, retiming may be as simple as determining the phase of the incoming data with respect to the clock and latching it appropriately, eliminating the need for frequency recovery, phase-locking, etc.

4.2 Timing and Synchronization Scheme

The Abacus system specifies interchip communication bandwidth at 250 MBits/s per pin, because the large number of processors on the die require that data from the edge elements be multiplexed two to one onto the output pins. This reduces the required pinout of the component. The bandwidth requirement implies a communication bit cell time of 4 ns. Although the network is fairly regular physically, consisting of only short wires between chips on the

same board and long wires between chips on different boards, process variation causes the i/o latencies to vary continuously between these extremes. This variation occurs because circuit delays are a significant component of the latency between close chips. Simulation shows that the delay of interchip communication ranges between 1.99 ns and 5.89 ns, accounting for both differing propagation delays and circuit speeds. Since the maximum delay is greater than a bit cell time, the receiver must retime its incoming data.

However, the regularity of the network allows some simplifying assumptions to be made. The longest delay may be larger than a cell time, but it is guaranteed to be shorter than a cycle (8 ns). This means that the first transmitted bit must arrive within one cycle of the onset of communication. The uncertainty is where in that first cycle the bit appears at the receiver. If the receiver could get a consistent source of edges from the transmitter, it could pinpoint where in the cycle the first bit appears and retime accordingly. To provide this source, a timing waveform (hereafter referred to as the escort clock) is transmitted along with the data on its own specially allocated line. Since all the processing elements on one edge of the die transmit along the same distance and since process tracks well on an individual die, only one of these special clocks is required per side of the chip. This adds only one extra line to a bus of sixteen and its addition is relatively inexpensive. Moreover, by shifting the escort clock so that it is a quarter cycle out of phase with the transmitted data, the clock can be used to latch the data at the receiver and ensure that valid data arrives.

The detailed timing diagram for interchip communication appears in Figure 4-1. **TComm** is a 50 percent duty cycle waveform with a period of 8 ns generated locally at the transmitter but carefully synchronized to thee globally distributed clock. **TQuad** is the quarter cycle delayed version of this clock, and **RComm** and **RQuad** are the corresponding clocks at the receiver. It is important that there be small phase differences between these clocks.

The timing diagram numbers the three edges of **RComm** significant for retiming. To minimize the latency of initiating interchip communication, the data must be synchronized to **RComm** by edge 3. There are only two important cases: the data and escort are either early or late. Consider the **EarlyBus-EarlyEscort** pair, which corresponds to a very short delay between transmitter and receiver. Suppose bit *A* is first latched at the receiver on the rising edge of the escort clock. As long as it is sampled by the escort sufficiently early, the data can

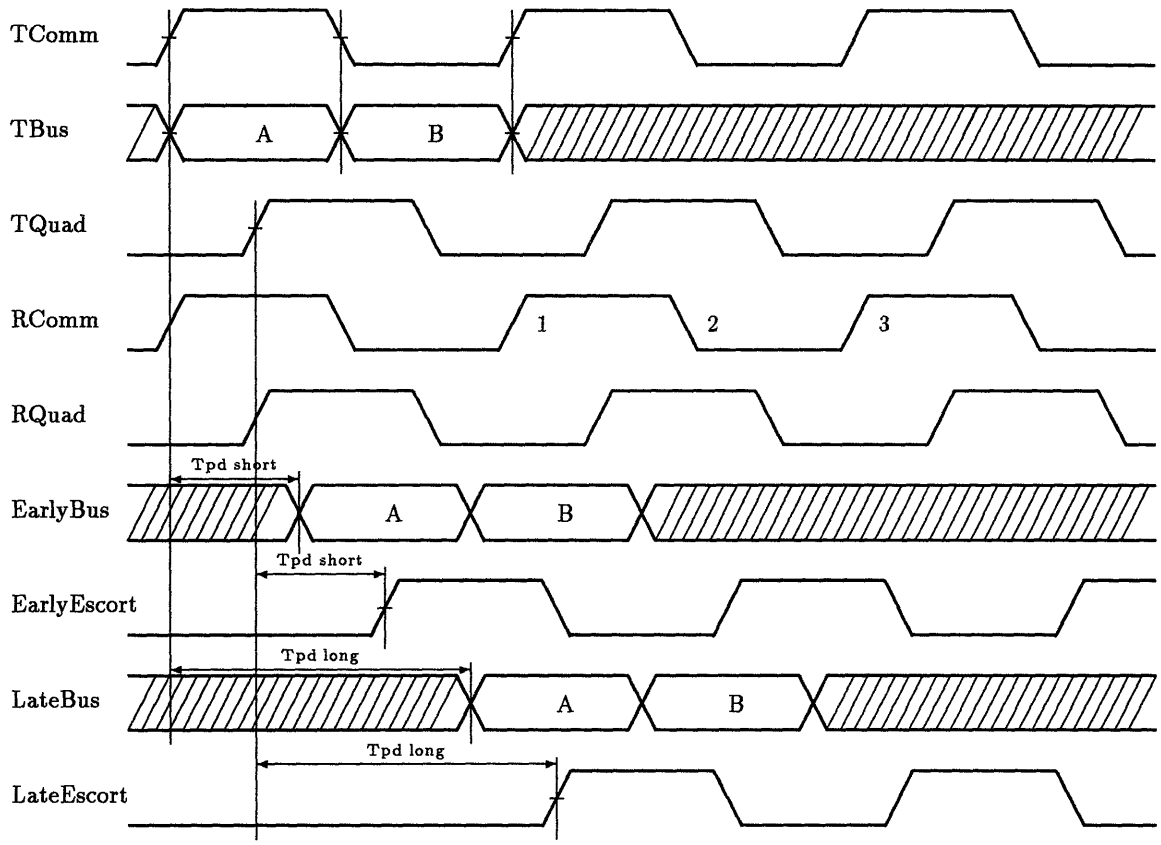

Figure 4-1: Interchip I/O Timing Diagram

be latched again edge 1 of **RComm** and a third time by edge 3. The output of the final latch is synchronized to the local clock and the data is retimed. In the early case, bit B is sampled by the falling edge of the escort, latched again by **RComm** edge 2, and a final time by edge 3.

In the second case, the communication latency is greater than a bit cell time. Assuming bit A is again first latched by the escort, it is not guaranteed to arrive early enough to satisfy the setup time of a retiming flip-flop clocked by **RComm** edge 1. Consequently, the falling **RComm** edge 2 is used to latch A after the escort. Edge 3 is again used to do the final retiming. Bit B is only latched by edge 3 after being sampled by the escort.

The receiver must only be able to distinguish between these cases and multiplex appropriately between them. This can be accomplished by comparing the escort clock with the receiver's local quadrature clock. If the rising edge of **Escort** occurs while **RQuad** is high,

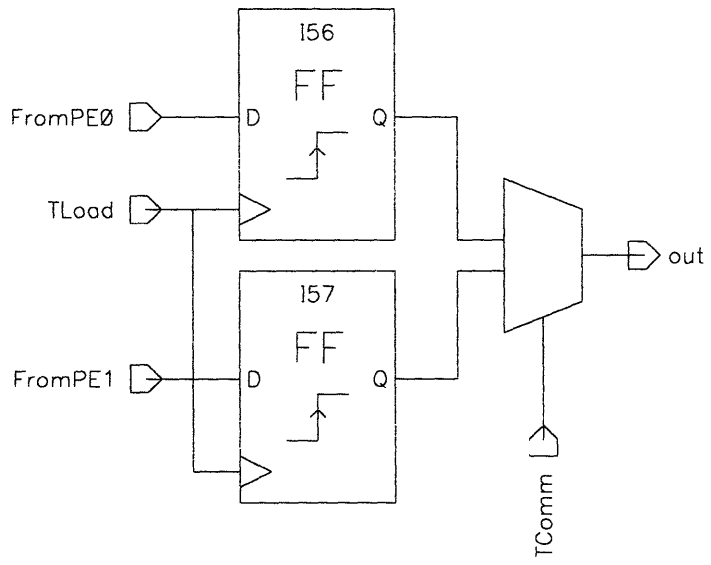

Figure 4-2: Interchip Communication Multiplexer.

the delay corresponds to the early case. If it occurs when **RQuad** is low, it corresponds to the late case. Because the delay is bounded to be greater than zero and less than one cycle, the only metastable point occurs when the escort's rising edge occurs at the same instant as **RQuad**'s falling edge. In this case, it doesn't matter which retiming path the logic chooses. The interval between the falling edge of **RQuad** and edge 1 of **RComm** is 2 ns, which satisfies the setup time requirement. If the retiming path that utilizes edge 2 is chosen, the data will again be retimed correctly. Thus it is only necessary to design the phase comparison logic to force a decision in this case.

4.3 Transmitter Circuits

Figure 4-2 shows the transmitter's multiplexing circuitry. The data bits are latched from the PE array into the two flip-flops. **TComm** is then used to time the multiplexing of the transmitter's data. On the rising edge of **TComm**, bit *A* from PE0 is transmitted through the pad's prebuffer and impedance matched driver. Bit *B* from PE1 is driven out on the falling edge. The transmitter's clock generation circuitry ensures that **TComm** and the escort clock have the appropriate phase relationship and are accurately synchronized to the

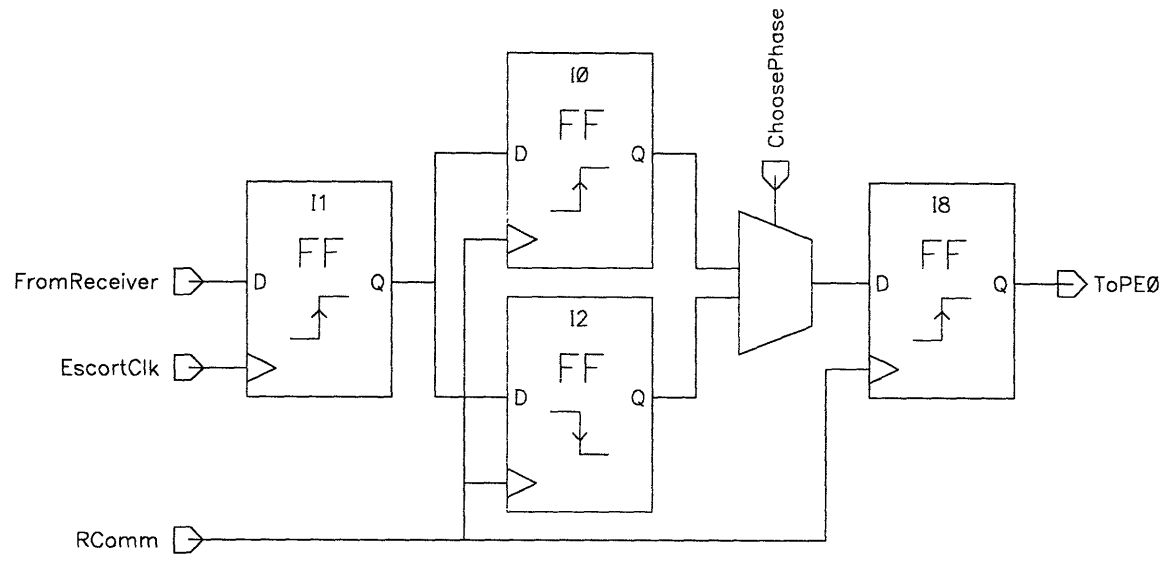

Figure 4-3: Low Order Bit Demultiplexer/Retimer.

globally distributed clock.

It is important that both transmitted bits have the same duration. If one bit had a shorter cell time, the limited bandwidth due to package parasitics may distort its shape (i.e. shorten the duration further) and make its detection impossible at the receiver. The longer duration also affords more timing margin that can absorb delay mismatches in circuits and possible skew between transmitter and receiver.

4.4 Retiming Circuits

4.4.1 Demultiplexers

Figures 4-3 and 4-4 show the demultiplexing and retiming circuits at the receiver for the first and second transmitted bits, respectively. They each consist of two cascades of flip-flops. The flip-flops are clocked with the appropriate edges described above. The multiplexer is controlled by the phase comparison circuitry.

4.4.2 Phase Detector

The phase detector used to choose the correct retiming path is shown in Figure 4-5. It

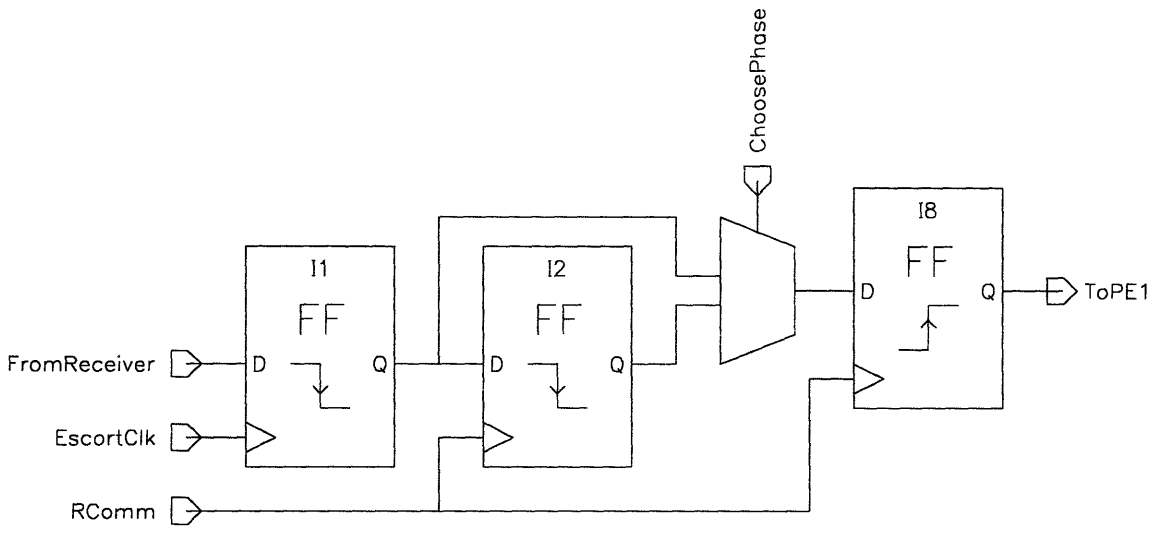

Figure 4-4: High Order Bit Demultiplexer/Retimer.

is a dynamic circuit. Suppose both inputs are low and consequently the inputs to the two edge triggered flip-flops are high. Furthermore, suppose **in** makes a positive transition. The input of the **early** flip-flop is then pulled low and the righthand pulldown chain is disabled. After a delay, the **late** flip-flop latches the high voltage on its input. Some time later, when **bufin** goes high, a low voltage will be sampled by the **early** flip-flop. The phase detector has accurately determined that **bufin** lags **in** and asserts the **late** signal. The **late** signal is used to determine the retiming path by using **RQuad** and the escort clock as the **in** and **bufin** inputs respectively. Although only one of the outputs is used, the symmetry of the comparator is preserved to ensure that both inputs are loaded equally and the phase comparison is as accurate as possible.

If the both inputs transition very near each other, the output flip-flops may beome metastable. Metastability is handled by adding extra inverters on the outputs of the flip-flops. The gain of these devices will amplify noise and force a valid logic level from a metastable state. Simulation shows that the detector accurately responds to phase differences as small as 105ps. This difference is small enough compared to the 8 ns period of the inputs that the performance of this circuit is adequate for this application.

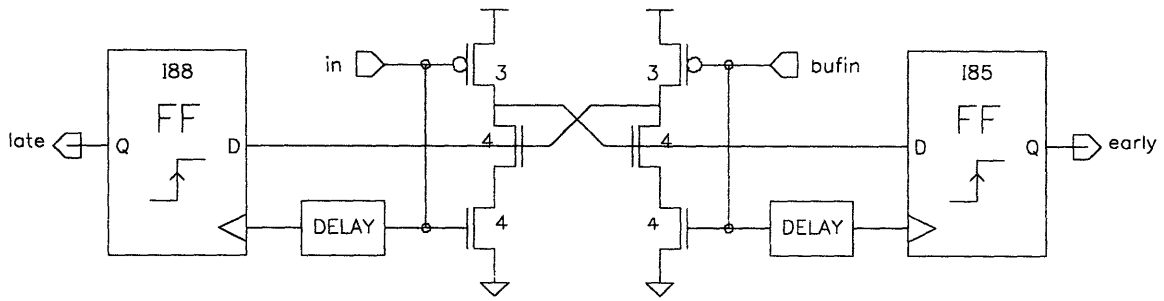

Figure 4-5: Phase Detector.

4.5 Advantages of the Abacus Network

Although many parallel computer systems have a regular, synchronous network architecture, the Abacus in particular has advantages that reduce the cost of implementing data retiming. An interesting comparison case is the Transit system [10]. Transit is a small multiprocessor system based on a multistage routing network which allows any processor in the machine to communicate with any other. Although Abacus limits interprocessor communication to nearest neighbors, the systems have some similarities. Both are globally synchronous with communication channels that are several bits wide. Both also demand high bandwidth and as a result require data retiming.

In Transit, the skew between signals in the same bus is significant. This is caused by the realities of routing long wires on printed circuit boards. Skew is introduced by the differences in wire length when leaving the package. Transit routing components use a pad grid array package. Signals that must leave through outputs in the corners therefore travel a longer distance than signals that leave directly through an edge. When a bus must make a right angle turn, the signals on the outside corner have to travel further than those on the inside. Buses must also be routed under other packages because of the high density of components on Transit boards. This complicates wiring and bus lines may jog differently to accomodate the other signals. Some traces may also have to change layers. Passing through vias adds delay to the signal. On a different layer, the distributed capacitive loading of the transmission line may be different. This also affects the delay of the trace. More skew is added as signals take different paths into the receiver's package. The higher transmission rate for Transit (200

MHz) makes these skews significant compared to a bit time.

Skew complicates retiming, especially if the different delays straddle a metastable point at the receiver. In that situation, early arriving signals may be retimed to one clock cycle earlier than the late arrivals. It is then necessary not only to retime data to the receiver's clock, but to align all the data to the same cycle of that clock. Distributing an escort clock in this situation is fruitless since the skews disrupt the phase relationship between it and the data. To account for all these skew factors, a variable delay line is incorporated at *every* output pad. Additional circuitry that measures the delay of an individual line is also integrated into the pad. The delay line is set to minimize the skew between wires on the same bus.

The Abacus network eliminates many of these difficulties. Although skews are still introduced on entering and leaving packages, the smaller number of i/o's can be routed through pins on an edge. The corner pins can be reserved for less skew sensitive signals. Since communication is purely neighbor to neighbor, buses do not make turns or run under other packages. The longer cycle time makes the skew that does appear more tolerable. Therefore, wires on the same bus in Abacus can be assumed to have the same delay. Distributing the escort clock along with them allows the receiver to easily determine the phase relationship of the incoming data. Moreover, the receiver amortizes the retiming decision circuitry over the entire bus. Minimal skew between bus signals eliminates the need for delay lines, delay measurement circuits, and retiming logic at every pad. Signals are all automatically retimed to the same cycle since the decision is made by only one set of logic.

Chapter 5

Clock Generation

From the discussion of synchronization in Chapter 4, it is clear that several timing waveforms must be generated with sufficient accuracy to ensure reliable interchip communication. Variations in integrated circuit processing pose a challenge to designing circuits that produce such signals. This chapter describes clock generation circuitry that uses delay-locked loops to achieve the necessary accuracy.

5.1 Delay-Locked Loops

The use of delay-locked loops (DLLs) is a popular circuit technique for reducing phase error between waveforms that are nominally identical. For example, process variation can cause the delays of clock buffers on different chips to be mismatched, introducing unacceptable skew in a globally synchronous system. A DLL can reduce this skew by compensating for the processing. DLLs have been used for on-chip clock generation [6], reducing chip-to-chip skew [7], and timing of high bandwidth interfaces [8]. In this particular application, a DLL is used to synthesize a fifty percent duty cycle clock from a precharge waveform which does not have such a duty cycle.

These loops have certain advantages over traditional phase-locked loops (PLLs) based on a voltage-controlled oscillator (VCO). First, the analog design issues are simplified for DLLs. Also, the noisy environment of digital circuit power supplies makes achieving acceptable jitter performance from a PLL difficult. Third, the inertia of PLLs must be overcome to achieve

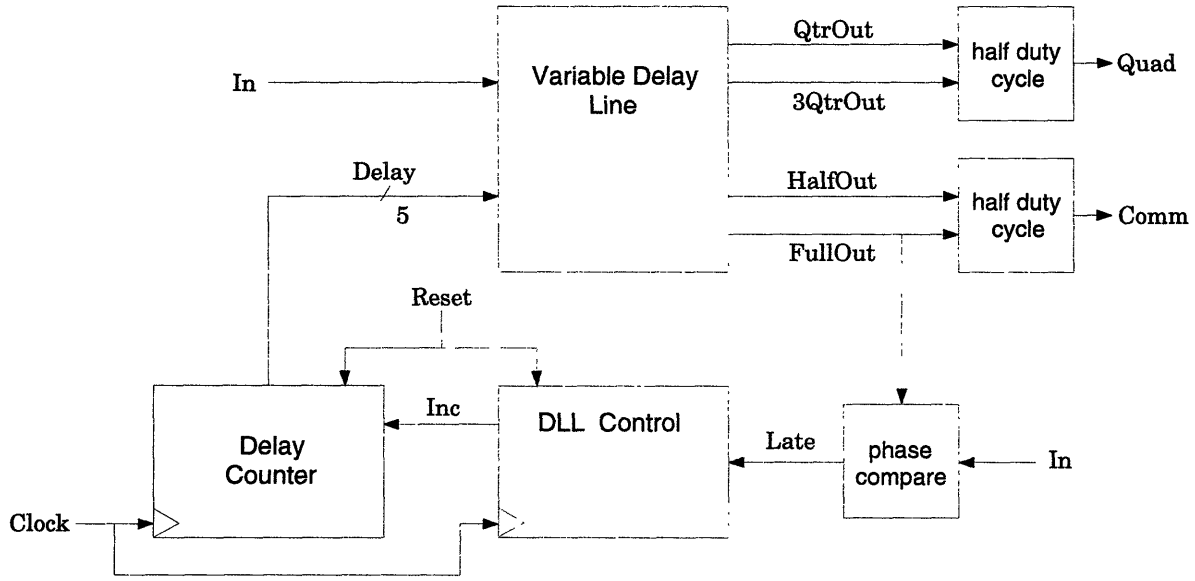

Figure 5-1: Delay-Locked Loop Block Diagram.

phase lock and the time this takes may be unacceptable for certain applications. Finally, a VCO is unnecessary in systems where the correct frequency of waveforms is provided and does not need to be synthesized.

Earlier delay-locked loops have optained very fine delay resolution by using some analog techniques. Delay lines have been constructed using current starved inverters or capacitive loads as delay elements. Analog voltages generated using charge pumps control the amount of current or capacitive loading. The same level of accuracy is not required in this application because of the relatively long cycle time. Consequently, analog techniques are unnecessary a "coarse" delay-locked loop with a digitally controlled delay is used.

5.2 Coarse Delay-Locked Loop Design

To generate accurate, fifty percent duty cycle signals with a period of 8 ns, the digital delay-locked loop shown in Figure 5-1 is used. The input signal **In** is a precharge waveform with an 8 ns period and a duty cycle of less than 50 percent. Basically, the loop is used to measure accurately the period of the input waveform and then generate copies of it with

appropriate phase relationships. The intial delay of the line (i.e. the delay when the control bits bf Delay are zero) must be less than one period. The phase of the signal **FullOut** is compared to the input. The delay of the variable delay line is by servoed incrementing **Delay** until **FullOut** is in phase with **In**. The overall delay is then matched to be exactly one period of the input. The other waveforms are generated by tapping off the delay line at the one quarter, half, and three quarter points. Since **FullOut** is delayed by a half period with respect to **HalfOut**, these waveforms can then be used to generate a fifty percent duty cycle clock. **QtrOut** and **3QtrOut** are used to generate the quadrature of this clock.

The delay line is controlled digitally through the counter bits. There are no analog control voltages in this design. Although this reduces the delay resolution of the line, hence the name "coarse" DLL, it simplifies the design by eliminating the need for a charge pump. The coarseness of the delay does affect the duty cycle, making it larger than fifty percent by half the delay resolution (500 ps) in the worst case. However, this effect is sufficiently small compared to the half period that it is acceptable. The range of the line is also important. Because it is trying to accurately measure a time interval rather than simply aligning the edges of the input and output, there is a constraint on the minimum size of the delay. The minimum delay must be less than the period to be measured, otherwise the loop may lock at twice the input's period and the generated clocks will have the correct duty cycle but at half the input frequency.

5.2.1 Delay Element

The variable delay line is constructed of four separate delay elements. This inherent division makes it easy to generate the four phases of the input described above by taking them from the outputs of the appropriate elements.

To reduce the minimum latency, each delay element is divided into two halves which reduces the number of multiplexers in the delay line. The half corresponding to the low order bits of the delay value is shown in Figure 5-2. Four inverter chains of various lengths are multiplexed in according to delay bits 1 and 2. This stage is cascaded with a capacitively loaded inverter. The low order delay bit increases the delay of this stage by switching in the capacitors. In contrast to previous work [7], this control voltage is digital rather than analog.

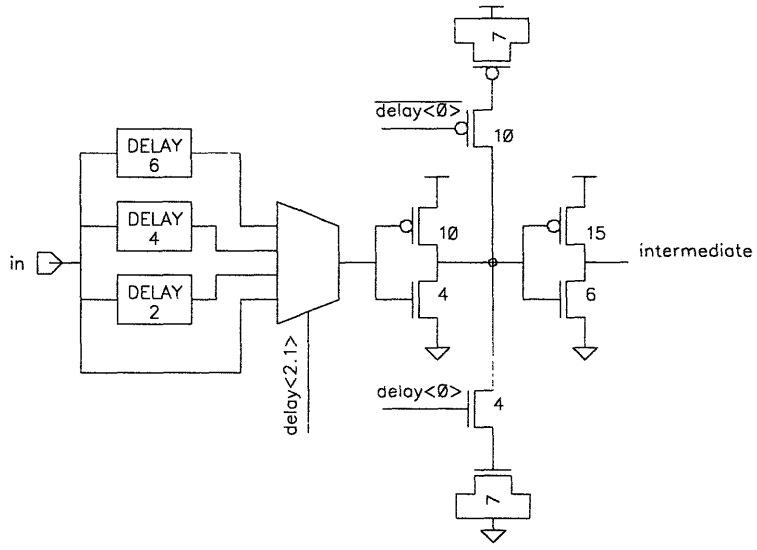

Figure 5-2: Variable Delay Element, Low Order Stage.

The high order half, shown in Figure 5-3, differs only slightly from the low order element. The inverter chains are four times longer and the output has an extra inverter. This inverter is required to equalize the propagation delay of rising and falling edges on the input. Suppose a positive edge takes time t_{pos} to propagate through one delay element consisting of a cascade of the two halves shown. Similarly, a negative edge takes t_{neg}. Without inversion, a positive edge at the input would appear at the output after a time $4t_{pos}$ while a negative edge takes $4t_{neg}$. Circuit variations like p to n transistor mismatch make it difficult to ensure that these delays are the same. However, adding the inverter means that both edges take the same time to propagate through: $2t_{pos} + 2t_{neg}$. There are an even number of inversions between the input and the half and full point outputs, so these signals have the same polarity. The quarter and three quarter outputs have opposite polarity to the input.

The total capacitive load is sized to yield a delay equal to one inverter over the entire delay element. It was split into two pieces, at the output of the low and high order stages, to increase the bandwidth of the line. The use of both p and n channel devices equalizes the load seen by both transitions.

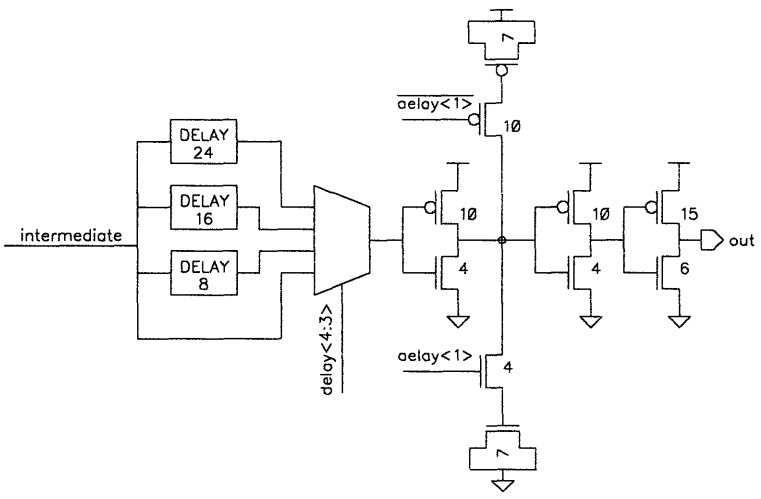

Figure 5-3: Variable Delay Element, High Order Stage.

5.2.2 FSM Controller

A finite state machine handles the setting of the delay line. In the reset state, it compares the phase of the fully delayed signal (**FullOut**) with the input. It increments the delay counter until the locked state is reached, which occurs when the delay is exactly one period of the input long. A phase detector similar to the one described in Chapter 4 is used for the comparison. To reach the locked state, the controller must first detect an early signal and then a late signal. This ensures that the loop locks at one cycle rather than before one cycle. Suppose the initial delay of the line is very short and the rising edge of the delayed signal occurs while the input is high. The phase detector will assert **Late**, even though the delay isn't long enough. The controller will ignore a **Late** assertion and continue to increment the delay until it detects a $\overline{\textbf{Late}}$ (early), which corresponds to the delayed signal rising while the input is low. Now the delay is known to be greater than the duty cycle of the input waveform and the next **Late** assertion will correspond to incrementing the delay just beyond the period of the input.

5.2.3 Half Duty Cycle Generator

The outputs of the delay line are converted into half duty cycle clocks by the dynamic circuit in Figure 5-4. The circuit is self-timed: the pullup and pulldown paths that charge

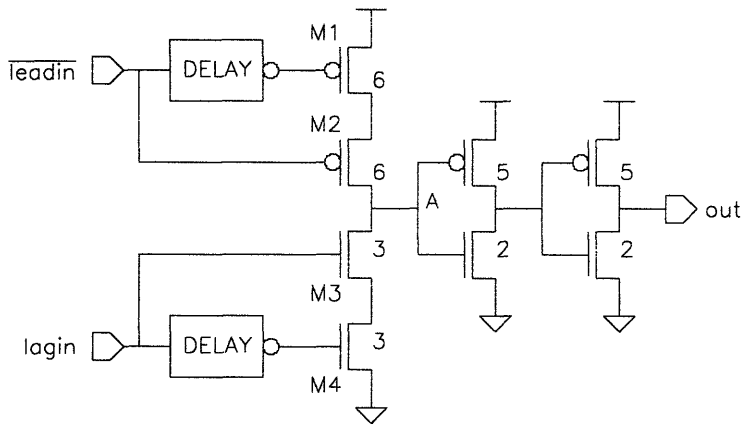

Figure 5-4: Half Duty Cycle Generator.

and discharge node **A** are enabled by transitions on the input signals and disabled some delay after these transitions. For example, suppose node **A** is precharged high and $\overline{\textbf{leadin}}$ has been high longer than the delay of the upper inverting delay line. Transistor M2 is off and M1 is on, precharging the intermediate node in the pullup stack to avoid charge sharing problems later. In addition, suppose **lagin** has been low long enough for transistor M4 to turn on, predischarging the intermediate node in the pulldown stack. Transistor M3 is of course off. When **lagin** goes high, M3 turns on and discharges **A**. Some delay later, the output of the lower delay line goes low and the pulldown path is cut off, leaving node **A** low and high impedance. Some time later, when $\overline{\textbf{leadin}}$ goes low, **A** will be pulled high and then left high impedance. If the two inputs are a half cycle out of phase and their duty cycles are long enough to charge and discharge **A**, the final output will be a fifty percent duty cycle waveform with the same period as the inputs.

5.3 Simulation Results

Results of simulating the full delay-locked loop clock generator are shown in Figure 5-5. The top waveform is the precharge clock from which the half duty cycle clock is generated. The in-phase and quadrature versions of this clock are shown in the middle and bottom traces, respectively. The duty cycles are close, but not quite half the period. The longer part of the

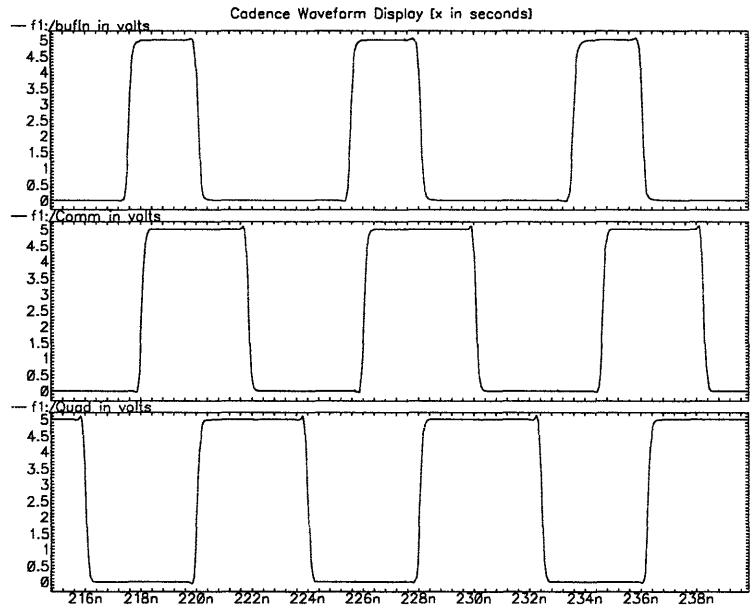

Figure 5-5: Clock Generator Waveforms.

cycle is 4.6ns long while the shorter half is 3.4 ns, the offsets being caused by the coarseness of the line. The worst case resolution of the line is 1.24 ns, and it is clear that the extra width of the outputs corresponds to half this resolution. Also, the in-phase output is delayed by this resolution with respect to the input, so the situation above corresponds to the worst case delay locking: the output is moved by one delay increment from being just early to being substantially late. As discussed above, since the loop is only trying to measure a delay rather than align the phases, this relatively poor performance can be tolerated.

The worst case range of the delay line is 14.57 ns, so it can clearly produce delay to lock one period. Improving the resolution is fairly straightforward; simply incorporate the analog techniques used in other designs. For example, the low order delay bit that couples the capacitive loads to the delay stages can be replaced by an analog control voltage produced via a charge pump.

Chapter 6

Conclusion

High performance interchip communication is becoming increasingly important in today's VLSI systems. Latency and bandwidth must be improved as circuit speeds increase to prevent the overhead associated with communication from dominating computation time. Achieving satisfactory performance requires careful engineering that takes many issues into account. Limited pinout increases the necessary bandwidth per pin. Power consumption must be kept at manageable levels. High edge rates mean that transmission line effects, crosstalk, and power supply variation must be dealt with in circuit designs. The large variation inherent in integrated circuit processing also complicates the design issues. To achieve high performance reliably, circuits must be designed to account for all these effects. The Abacus computer uses a bidirectional pad designed with these issues in mind.

Timing issues also become important to performance. As propagation delays become large compared to bit cell times, data retiming must be performed at the receiver to guarantee the validity of input data with respect to the local clock. This introduces yet more complexity into the communications system. Frequency and phase variations must be detected and compensated for by the receiver circuitry. Again, this must be done in a manner that is robust to the physical variations described above for the system to operate reliably.

Techniques that accomplish these goals exist and have been applied in general, asynchronous networks. Impedance matching drivers that account for process variation as well as manufacturing tolerances external to the chip have been used to minimize noise due to reflections. Data retiming is a common technique for synchronizing incoming data to a re-

ceiver's clock. Delay-locked loops are often used to generate accurate clocks and minimize skew between chips. These techniques can be adapted to a regular, synchronous network to achieve high performance. This adaptation reduces the cost of adding extra functionality by amortizing it over a bus of wires. Timing can be greatly simplified by the validity of certain conditions in a globally synchronous environment. In Abacus, impedance matching circuitry appears once for every sixteen wires. Retiming logic also appears once per bus. In the Transit system, skews between wires on the same bus are more severe because of routing. The appropriate solution in that instance is to retime each data bit separately.

Although the techniques presented in this paper were used in a SIMD multiprocessor, they can be applied to other systems where network regularity exists. For example, buses between processors and memory can use these ideas to achieve high performance. High performance interconnect does require extra complexity in communications circuits. However, taking advantage of inherent symmetry can achieve performance at reduced complexity and cost.

Bibliography

[1] M. Bolotski, R. Amirtharajah, W. Chen, T. F. Knight, Jr., T. Kutscha, and T. Simon, "Abacus: A High-Performance Architecture for Vision," *International Conference on Pattern Recognition*, in preparation.

[2] L. Dennsion, W. Lee, and W. Dally, "High-Performance Bidirectional Signalling in VLSI Systems", *MIT LCS Memo*, 1992.

[3] A. DeHon, T. F. Knight, Jr., and T. Simon, "Automatic Impedance Control," *ISSCC Digest of Technical Papers*, 164-5, 1993.

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

[5] B. Sklar, *Digital Communications: Fundamentals and Applications*, Prentice-Hall, Englewood Cliffs, N.J., 1988.

[6] A. Waizman, "A Delay Line Loop for Frequency Synthesis of De-Skewed Clock," *ISSCC Digest of Technical Papers*, 298-9, 1994.

[7] M. Johnson and E. Hudson, "A Variable Delay Line PLL for CPU-Coprocessor Synchronization," *IEEE J. Solid-State Circuits*, 23(5):1218-23.

[8] T. Lee et al., "A 2.5V Delay-Locked Loop for an 18Mb 500 MB/s DRAM," *ISSCC Digest of Technical Papers*, 300-1, 1994.

[9] R. Rettberg and L. Glasser, "Digital Phase Adjustment," U.S. Patent No. 4,700,347.

[10] A. DeHon, "Robust, High-Speed Network Design for Large-Scale Multiprocessing," S.M. Thesis, Dept. of Electrical Engineering and Computer Science, MIT, 1993.

[11] N. Weste and K. Eshragian, *Principles of CMOS VLSI Design: A Systems Perspective*, Addison-Wesley, Reading, Mass., 1993.

[12] W. Gump, "Investigation of y-scaling Analysis in Quasielastic A(e,e'p) Experiments at High Momentum Transfer," S.B. Thesis, Dept. of Physics, MIT, 1994.

[13] F. Funkalicious, "Fresh Billy Funkalicious and His Magic Adventures in Virtual Particle Land," S.B. Thesis, Dept. of Funk, MIT, 1994.

[14] C. Adler, "Chantefables et Chantefleurs," S.B. Thesis, Dept. of Music, MIT, 1994.