ABSTRACT

STANASKI, ANDREW JOHN. Flip Chip testing with a capacitive coupled probe chip. (Under the direction of Paul D. Franzon.)

Testing integrated circuits that employ an area array of I/O presents unique challenges because the face of the chip is not visible for probing. On chips that use perimeter bond pads the face of the chip is exposed, so signals on the wiring in the top layer metal may be probed while the chip is in operation. This is not possible when the face of the chip is hidden.

This work proposes a way to probe test points on the top layer metal of chips that use area I/O. The method works by attaching the chip to a specially designed probe chip instead of the normal packaging. Metal pads on the top layer of the probe chip correspond to lines on the top layer of the chip being tested. These points form a capacitive coupling between the chips, letting the probe chip read the signals at the test points. This leaves the original chip largely unchanged, and allows critical signals to be probed.

The geometry of the test points is examined and evaluated using a field solver for their potential to couple between the chips. A square section of metal roughly 6 μm on a side provides 1 fF coupling capacitance, enough for a receiver on the probe to reproduce the signal.

The work continues with the design of a receiver circuit to amplify the small input from the test points. The receiver employs a differential amplifier followed by an inverter to amplify the signal without excessive loading at the input. Simulations of the receiver demonstrate its ability to recreate the signal. Additional simulations measure the performance of the receiver under varying conditions, and explore the operational characteristics.

This work also describes the design of a four issue superscalar microprocessor that was used as a reference for explorations of systems design for multichip modules (MCMs). This work focused on the chip testing aspect of area array I/O chips used in an MCM. Other work investigated partitioning, routing, and other system design issues.

Finally, the work gives an outline of the CAD tool setup created for use at N. C. State University. The design kit created supports research as a vehicle for creating chips, and for integrating research CAD algorithms.

FLIP CHIP TESTING WITH A CAPACITIVE COUPLED PROBE CHIP

by

ANDREW JOHN STANASKI

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

ELECTRICAL ENGINEERING

Raleigh

2003

APPROVED BY:

_____	_____
_____	_____
	Chair of Advisory Committee

DEDICATION

This work is dedicated to Marcela, whose encouragement and support made it possible.

BIOGRAPHY

I started my program of study a number of years ago. My idea was to get a masters degree to refresh my knowledge in engineering, for I was working previously in an altogether different field. A funny thing happened on the way to that masters degree. I met some wonderful people at NCSU and decided to stay on for more. After all the work and effort you see just this document. It may not shape the world like some grand unified theory, but the experience has left its mark with me, and I hope with the people I have had the privilege to meet.

ACKNOWLEDGEMENTS

I would like to thank Paul for his patience and guidance. It has been quite an adventure since that day I walked into your office to ask about the computer architecture class. It turned out to be one of those ordinary events, on an ordinary day, that, when you look back, was a turning point in your life. Thanks for making it a point to remember.

I would also like to thank my committee. This took a little longer than I planned. Thanks for waiting.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1 Introduction

An electronic system implemented on a multichip module (MCM) creates unique and challenging test requirements. An MCM looks like a printed circuit board in miniature, but it takes bare die instead of packaged parts. Therefore, the chips must be tested as a bare die before being integrated on the MCM. Because the thoroughness of the testing has a dramatic impact on the quality of the MCM, the testing must be sufficient to deliver a high percentage of known good parts—a high chip yield.

For an MCM the yield of the module is the product of the individual chip yields.

$$\text{yield}_{MCM} = \text{yield}_{chip1} \text{ X } \text{yield}_{chip2} \text{ X } \text{yield}_{chip3} \ldots \quad (1.1)$$

If each chip has the same yield, the equation simplifies to:

$$\text{yield}_{MCM} = \text{yield}_{chip}^{\text{NumberOfChips}} \quad (1.2)$$

Clearly, more chips and lower chip yields dramatically lower the module yield. For example, a module with four chips, each with a yield of 90 percent, creates good MCMs at a rate of just 65 percent. Clearly, known good chips are vital to good first pass MCMs. Even with high yield chips, MCM test and rework may be required.

Another chip metric vital to MCM performance is individual chip speed. Chips are often tested to find their maximum operating speed, and binned into speed ratings. Microprocessors are sold by their model and speed rating. The speed rating of a multichip module is determined by the speed ratings of its constituent chips. In general, the module speed rating will be set by the slowest chip, much like the load of a chain is determined by its weakest link.

Consider a module that uses chips which are binned into three categories; slow, medium, and fast. Assume each speed rating is equally likely for all the chips on the module, and the module speed is set by the slowest chip on the module. If the chips are sorted into known speed bins before assembly you would expect one third of the modules would be fast, one third medium, and one third slow. If the chips are not binned you get mostly slow modules, as indicated in Figure 1.1. With four chips on the module there are 80 percent slow modules, 19 percent medium, and just 1 percent fast. Clearly, creating fast chips and determining the speed rating of the individual die is important.

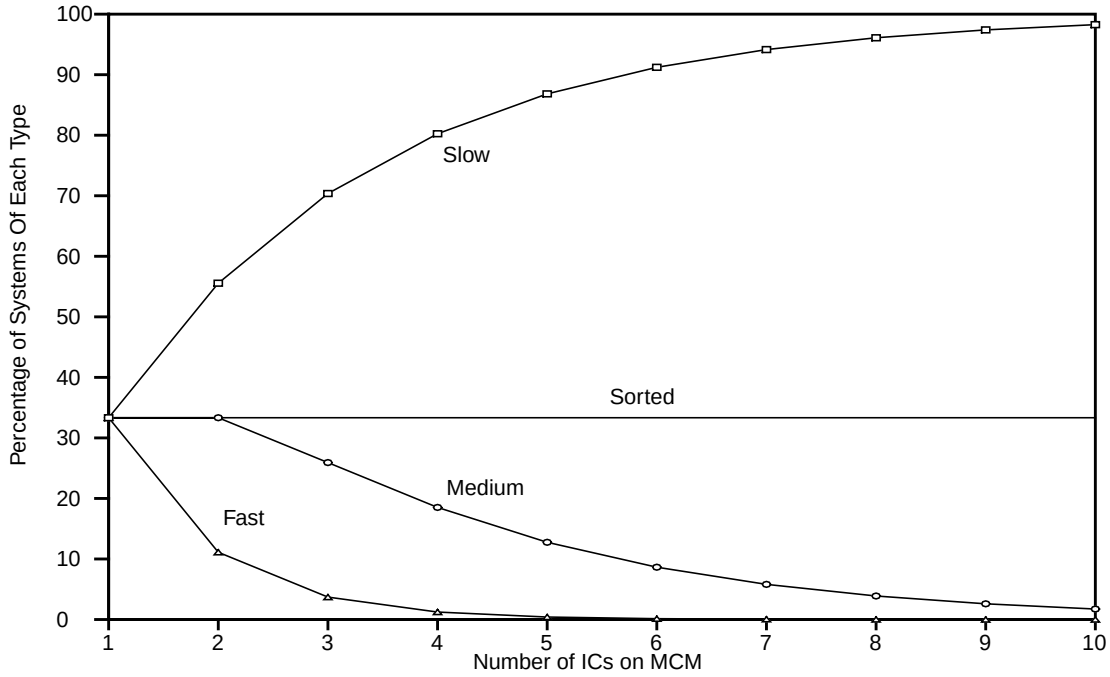

Figure 1.1 Speed sorting bare chips

One aspect of testing the chips is to make sure there are no implementation problems that cause the chip to run slower than expected. The so called timing debug examines chips early in the fabrication run to ensure they meet timing design specs. Sometimes the timing margin on certain paths, critical paths, is too small, and cause the whole chip to run slowly. If the timing modeling done during design was not adequate, the timing problems will not be detected until after early fabrication runs.

With conventional chips that use a pad ring and wire bonding to the package the face of the chip is exposed and may be probed. For timing debug an electron beam is used to detect signals on top level wiring while the chip is in operation. Analysis of these signals may lead the chip designers to the portion of the design causing the timing fault.

With chips employing an area array I/O such probing is not possible. The chip face contains the array of I/O pads. For the chip to operate the I/O points must be connected to power and signals from the test fixture. In doing so the face is covered.

This work examines a method that allows chips using area array I/O to be operated and probed. The method employes a special probe chip that attaches to the I/O of the chip being tested to provide power and signals to the chip under test. The probe uses capacitive coupling to examine signals on the top layers of the chip under test. The method works with minute levels of capacitance to couple signals to an amplifier on the probe chip. The output of the amplifier gives the design team a glimpse into the working of the chip under test.

The work was undertaken as part of a larger effort to optimize system implementation on multichip modules. Areas of research include partitioning, placement, routing, performance and testing of systems on MCMs.

The rest of this work is divided into the following sections. Chapter 2 surveys a variety of techniques used in microprocessor testing. Microprocessors were used as the system of choice because they employ a wide variety of circuitry, such as random logic and memory. They typically push the limits on clock speeds and have tight timing margins, so they represent the toughest challenges for testing. They also employ a wide variety of testability techniques. Therefore, microprocessor testability design provides a representative study.

Chapter 3 follows with a closer look at the capacitive probing method. It explains in more detail how the concept works. Following that, Chapter 4 dives into the minute detail of the method. It details the mechanical hookup, electrical modeling, probe circuit design, and probe circuit operation. This is the main part of the dissertation.

Chapter 5 presents the conclusions and outlines future work. Following that is the bibliography and appendices.

Appendix 1 contains extra, detailed data supporting the design of the probe chip.

Appendix 2 describes the design of a four issue superscalar microprocessor created as a reference for MCM design optimization. The Electronics Research Lab (ERL) at N.C. State University is examining many aspects of design implementation on MCMs. These efforts include research on system partitioning and routing as well as chip testing. The microprocessor design described was the base used to focus the MCM research work, and provided a sanity check for ideas during the research.

Appendix 3 provides an overview of the CAD tool infrastructure we built to support the MCM research. Cadence provides their tool suite to universities for a small cost. We added the fabrication process data for the MOSIS brokerage service processes, as well as customized programs and forms to create a useful tool flow. The process design kit supports a full custom design methodology for analog, mixed-signal, and digital IC design. The kit has been used successfully in classes and research at the university. Working chips have been fabricated.

Moreover, the infrastructure provides a base to do CAD tool research. By supplying a known tool flow it makes a good platform for integrating new algorithms emerging from the research. The research has a vehicle to support ideas, so it is easier to carry the idea farther than in a standalone program that has to manage its own flow.

Chapter 2 Microprocessor Testing

2.1 Introduction

Microprocessor design creates the ultimate challenge for silicon test and debug methods. The newest processors continually push the limits of silicon area, speed, and complexity. This makes design for economical test increasingly more difficult, and compensating for the fabrication variability all the more critical.

The goal of testing ICs is to verify the design, overcome the large variability inherent in the fabrication process, and ensure proper operation during service. It is currently impossible to produce chips, especially large microprocessors, with near 100 percent yields. Instead each chip must be inspected and tested to prove it operates correctly. In many industries the quality control process provides a manufacturing environment stable enough to make testing finished parts unnecessary. Chip manufacturing is not controllable enough, however, making this quality through inspection process is necessary, though costly. Furthermore, testing cannot adequately validate complex chips without the addition of specific circuitry to enhance a chip's testability. Thus, there is always a balance between the cost of adding design-for-test circuitry and its impact normal circuit performance, and the amount of test time it takes to achieve adequate fault coverage. There are a number of testing methods that are typically employed and most companies use a similar mix of these methods.

2.2 Background on test methods

This section defines some of the basic chip testing methods used today. There are numerous sources that provide detailed texts on each subject. In this paper the term flip flop is used to describe generic storage cells used in the random logic parts of designs. These cells may be level sensitive or edge triggered, but are generally distinct from the ones making up the storage in a register file or on-chip memory. The term register is used to denote a collection flip flops.

2.2.1 Scan: full, partial, boundary

Inserting scan testing involves replacing the normal flip flops in a design with ones having two different inputs. One input is used for the normal function of the flip flop while the other is used to tie the flip flops together, output of one to input of another, in a long chain. Thus, in the test mode, the internal state of the device can be set and read for all the flip flops in the scan chain. This greatly increases the observability and the controllability, and hence the testability of the design.

If a design has all of its flip flops connected by a scan chain it is said to be a full scan design. Full scan need not be one single chain. It may involve multiple scan chains, as long as all the flip flops belong to some chain. If some of the flip flops belong to chains and some do not the chip is a partial scan design. One particular type of partial scan that is widely employed is called boundary scan.

With boundary scan the inputs and outputs of a chip contain scan flip flops. These elements can separate the chip from the normal outside signals, giving it isolation from the rest of the system during testing. This is a very useful and popular employment of scan, one that has been standardized as IEEE 1149.1.

As with all hardware insertion for test, scan adds costs to a chip. For scan those costs come from the added area needed to transform the normal flip flops into scan flip flops. The transformation can also slow the flip flop speed. The area penalty is unavoidable, but the speed increase can often be accommodated. Only those registers which lie on critical paths, those that set the clock period, are of concern. The speed penalty of a well designed scan cell are a fraction on a nanosecond. In the Motorola 68060 their cell design setup time was less than 0.2ns slower. This small increase is often acceptable to gain the increase in testability.

2.2.2 Built-In Self Test (BIST)

An alternative to using an external tester to generate vectors to stimulate the chip logic, and then read and evaluate the results, is to build on-chip circuitry for those purposes. This is called built-in self test, or BIST. It is often used on specific blocks of a chip.

With BIST the block of logic to be tested is isolated from the rest of the chip, then the on-chip pattern generator feeds vectors to the block, while a signature analyzer checks the outputs. The pattern generator is usually some kind of linear feedback shift register (LFSR). The

evaluation circuit often uses a data compression unit, again usually an LFSR, before comparing the circuit output to the known good output for that test sequence. Because the pattern generation is done algorithmically BIST is best employed on circuits where algorithmic tests provide good fault coverage. Circuits with regular patterns, like memories, are good candidates.

BIST allows the test to proceed at the full clock speed of the chip, without connection to an external tester. As such it can be used while the chip is in service. It can also be buried deep in a chip where access may be difficult. Of course the cost is all the added circuitry for pattern generation and response compaction and memory to store the responses.

2.2.3 Testing Memories

The memory hierarchy of a processor has a profound impact on the overall speed of the system. Fast CPUs with wide instruction widths demand high bandwidth from the caches to keep them fed. Otherwise the CPI will drop while waiting on the memory. Because memory has such a large impact on speed, processors devote an increasing fraction of the total transistors to memory.

But in order to make the memory fast as well as large it must be carefully designed and tightly packed. Adding any extra circuitry incurs too much area and speed penalty. Therefore, memories are nearly always tested as a block with an external stimulus and response evaluation. Fortunately algorithmic tests work reasonably well, so BIST is often effectively used for testing these blocks.

2.2.4 IDDQ

Many fault models assume a go-nogo failure pattern in the circuitry. The stuck-at fault is such a model. Many test techniques employ these a models to generate test vectors and determine fault coverage. With CMOS, however, there are other failure modes. For example, typical CMOS failures include gate oxide shorts, and resistive bridges or resistive opens. Go-nogo tests may not detect these problems.

IDDQ works on the fact that static CMOS draws virtually no current after switching. Dynamic logic can also be made to draw no current after switching. The failures mentioned above may increase the power supply current when in this quiet state. Thus, measuring the

power supply current when there is no switching activity may detect these faults. In fact, this may be the only way to test for some of these faults.

In practice, testing I_{DDQ} is not a simple and straight forward task. All real devices have some leakage so setting the “bad” current threshold is a statistical process and is different for each design and fabrication technology. In addition, the logic must be put in a state where these faults lie along a path from the power supply to ground. Determining these vectors and their associated fault coverage in a large chip is not easy. Furthermore, the chip must remain in the quiet state for a relatively long period of time to get an accurate current measurement. This time can be excessive in production when testing a large number of chips.

2.2.5 Functional Testing

Ideally a chip would be partitioned for test into a large number of small clusters of transistors with each cluster easily, completely, and independently tested. Realistically a chip is divided into a much smaller number of larger blocks and each block is tested to ensure it performs the correct operation. This sort of testing is called functional testing. Often the test uses a fault model, such as the stuck-at model, together with the structure of the circuit to determine the input vectors needed and the fault coverage achieved.

However, some tests may not use a fault model for the circuitry. A test without a model generally uses an ad-hoc procedure to exercise the functionality of the block. For example, a memory block can be written a certain set of patterns, say all ones then all zeros followed by an alternating pattern. Another test might use a pseudo-exhaustive pattern of the inputs to test a large number of input combinations. Or, finally, the test might use a data storage model to determine the sequence of patterns to test that memory. All these examples attempt to verify the correct operation of the block without using the full structural design of the block.

2.2.6 Delay Fault Testing

In high speed designs it is not sufficient to just ensure correct functionality. Testing must also verify correct operation at the rated speed of the part. When a chip operates correctly at a slow speed and incorrectly at a faster speed it is said to have a delay fault. But the scope of

delay fault testing often goes beyond simply deciding if a chip operates at speed or not. It also attempts to ensure a delay fault can be isolated, robustly tested and the fault location diagnosed.

Delay fault testing uses two basic models for the cause of excessive circuit delay, the gate delay fault and the path delay fault. The gate delay model presumes the transition time of a gate causes the fault. This model looks a lot like the stuck at fault model with the correct behavior at slower speeds and faulty behavior at faster speeds. However, the accuracy of this model is not great since it assumes all the excess delay is concentrated in one gate.

The path delay model postulates that the extra delay lies along a path to the affected point. This path includes the delays of all the gates in the path as well as the nets. Thus, many pieces contribute to the overall delay time, which reflects the actual condition much more accurately. The difficulty in testing for these delayed paths is in stimulating a single path through a circuit. In order to make a path testable independent of any side path delays, also known as robustly testable, the reconvergent fanout and feedback paths must be cut. A large part of the overhead in delay fault testing goes toward ensuring the proper conditions for path setup. This can be significant.

Because of the cost of making a design delay fault testable, chips using perimeter bonding pads often employ a scanning electron microscope to probe the top layer metal of the chip in a process called voltage contrast (E-beam) testing. The E-beam probe scans a net of interest to determine the waveform on the net. This gives the timing on the net. E-beam testing, however, is not possible on flip-chip die.

2.3 Microprocessor test and debug

Overall processor execution speed depends on the product of three factors; number of instructions run, clock cycles per instruction (CPI), and the clock period. Typical microprocessor design has generally evolved from single instruction handling, to pipelined execution with several instructions being executed in a line, to superscalar machines with multiple parallel pipelines. This progression suggests that most designs have concentrated on reducing the CPI by improving the machine architecture to handle many instructions at once, which is indeed the case. Of course designers have taken advantage of fabrication technology improvements to reduce the clock period too, it's just they have spent more effort to wring out speed improve-

ments through reduced CPI than through raw clock speed. But the addition of vast quantity of transistors gained in the new technologies arranged in complex architectures has made chip testing and debug very difficult. Most designers have solved this problem in similar ways.

2.3.1 Typical structured test employment

As chips have become smaller and more complex the number of devices on-chip has grown much faster than the number of inputs and outputs (I/O). Thus, the transistor to I/O ratio has soared which means the controllability and observability of much of the logic has decreased significantly, making it very difficult to test. To access the internal logic most processors employ scan chains. For example, the UltraSPARC™-I has approximately 22,000 scannable flip-flops that give full scan to all logic blocks in the processor except one, which has partial scan [20]. In fact, the PowerPC 603™, Motorola MC68060, and HP PA7100LC all use scan as a primary means of achieving high test coverage [19, 25, 27]. Scan chains provide high controllability and observability, and are easy to implement in a design. This keeps the time to market impact to a minimum. However, they cost chip area to make the flip flops scannable, they add a small amount of delay to each flip flop, and the test time increases as the number of flip flops increases. Nevertheless, this technique is adequate for a large group of microprocessors.

In addition to an internal scan capability, all processors include IEEE 1149.1 boundary scan with its associated test access port (TAP) controller. System manufactures demand chips contain boundary scan since it makes board testing much easier. And indeed boundary scan has become ubiquitous. Most of the internal testability features, however, do not need the boundary scan features or control.

With scan chains inserted to test the random logic there remains the testing of the memory blocks. Some designs use built-in self test (BIST) for these areas. The application of BIST is nearly always limited to one memory block, however, because use of the technique proves to be expensive. The pattern generators and signature analyzers required to achieve adequate coverage consume a significant chip area. Fortunately, by testing one memory, usually the instruction cache, together with verified control logic, the remaining memory blocks can be checked with functional vectors. The PowerPC™ 603 and PA7100LC use BIST to test their instruction caches [19, 25]. The Alpha AXP 21164 and Intel i486™ and Pentium® processors also use BIST for memory testing but the they do not use scan chains [24, 28].

The expense of BIST led the designers of the 68060 and SPARC processors to a different solution [20–22, 27]. They multiplex the input and output (I/O) pins on the package and activate a special test mode to bring the memory lines to the I/O pins. This makes the chip look like a memory only chip which can be tested directly.

Thus, the structured test methodology found on many processors uses scan chains to test the random logic, either BIST or I/O pin multiplexing to test some of the memories, and functional vectors to test any remaining blocks.

2.3.2 Functional testing

The Digital Equipment Corp. (DEC) and Intel use a different strategy for test. They do not use scan chains but rely on progressive functional testing [24, 28]. They start by testing a key memory with BIST. Then the good memory feeds functional vectors to other parts of the chip gradually branching out increasing the verified area. To increase observability they use parallel-in serial-out registers that monitor key internal nets.

For Intel this test methodology has evolved through their line of X86 processors. They employ added BIST hardware to exhaustively test the programmable logic arrays (PLAs) and microcode Control ROM. The PLAs and microcode ROM then drive the test on the rest of the chip, including the other memories and the logic.

The DEC approach is similar except they start with the instruction cache. BIST hardware tests the instruction cache which then branches out testing key parts of the logic until other memories in the processors can be checked. Once enough key memories are verified, testing proceeds on the bulk of the logic.

Unlike Intel the DEC Alpha is a new architecture with no legacy of test methodology. The Alpha was designed as a replacement processor to the VAX. During the creation of the concept for this microprocessor they looked back at the history of computing and saw that processors had, over the years, realized a 1000 times improvement in speed. They decided to try to create their architecture so it too could grow and realize a similar 1000 times speedup. To do so they envisioned a CPU that could eventually operate with a 10 times faster clock rate, running 10 instructions per cycle, and have 10 processors working in parallel. In their first implementations they concentrated heavily on the clock speed factor. This lead to widespread use of dynamic logic with tight timing margins. The designers felt the use of scan chains incurred

too much area and performance penalty, and would be difficult to design. Widespread use of BIST was far too costly, so they settled on this functional testing plan.

2.3.3 IDDQ Testing

The primary testing methods then divide the chips into the two camps detailed above. I'll call these the structured approach and the functional approach. This is not the end of the story, however. The UltraSPARC™-I, PowerPC™ 603 and PA7100LC designers also planned their chips to allow IDDQ testing.

IDDQ testing is an excellent method of checking for CMOS faults that do not model well as the usual go-no go (stuck at) type of fault. It is easiest to set up when used with the structured approach because the scan chains permit the state of the machine to be more easily set to a quiet mode. And indeed the chips using IDDQ are all from the structured camp. The results from using IDDQ testing can be a dramatic drop in the failure rate during functional testing. Reference [52] gives some typical results.

Still, using IDDQ is not a simple task. All real transistors have some leakage, so setting the "bad" current threshold is a statistical process and is different for each design. And determining the quiet state vectors and their associated fault coverage in a large chip is not easy. Furthermore, the chip must remain in the quiet state for a relatively long period of time to get an accurate current measurement. This time can be excessive in production when testing a large number of chips.

Chapter 3 Chip Debug

The testability features detailed in the previous chapter are all generally targeted toward checkout of chips during volume manufacturing. During the design cycle those features must also support debug. The scan chains of the structured approach lend themselves to this task. Scan breaks the logic into many smaller blocks which helps pinpoint the location of faults. In other words, it gives high observability and controllability. The various SPARC chips have a planned methodology utilizing the full scan built into them to provide detailed debug information [20, 21, 23]. This includes control of the clock driver, use of the scan chains, and modes to dump the contents of the internal memories.

The functional approach requires more effort. It uses observability registers placed at key points to see small, critical pieces of the design. Intel call this the scanout methodology [28]. Choosing the optimum placement of these observation blocks for maximum benefit at the lowest cost takes considerable effort. Special ports that bring critical nets out to pins, or allow the memories to be read, supplement the observation blocks. This sort of design effort makes sense for a high volume producers, like Intel, who want to minimize the test circuitry per chip, and for those who want to run the clock as fast as possible.

3.1 MCM factors

On an MCM the electrical cost of interconnect is less than in a single chip package. The MCM substrate provides a high line count interconnect with lower parasitic factors than a conventional printed wiring board (PWB). This can lead to a different circuit partitioning than single chips on PWB. Chips may have many more I/O, and because off-chip signals are not as electrically costly, the system designer may choose to place circuit elements requiring high interconnect on different chips. One obvious partitioning is to separate most of the cache memory from the logic. As long as the memory hierarchy can sustain the processing rate of the logic the partitioning will be effective [11]. Bare die with area array I/O used on an MCM support such a partitioning.

The increased pin count provides greater controllability and visibility on an individual chip. There may even be multiple entry points for clocking. At the same time, the lower parasitic environment on the MCM allows the I/O drivers to be smaller.

This provides an opportunity and a dilemma for testing. Traditional VLSI testers are not geared toward running bare chips with huge I/O counts, and the smaller drivers may not be adequate to drive the tester. There are techniques that can solve this problem. To test the bare die a membrane probe is often used. Membrane probes started as a high quality probe for testing at the wafer level [41, 42]. Because they have the unique combination of small feature size that allows high bump density, and good parasitic qualities, they became a natural choice for probing bare die with area array I/O [43–47]. The latest technologies resemble the MCMs they are emulating, and the probes have shown they can operate thousands of I/O at frequencies beyond 1 GHz [44]. This allows normal production testing of bare die.

Another method uses an interposer. Connection to an interposer creates a single packaged part, also called a chip scale package [87]. This may be testable in a VLSI tester.

In combination with a membrane probe or interposer, the VLSI tester can be expanded by using a smart test head. The smart test head incorporates drivers, both to and from the tester, as well as design for test features, such as pattern generation, signal compaction and analysis, and IDDQ for the power supply lines.

A smart test head provides significant benefits. The test head can be programmed to perform some of the chip testing autonomously. This reduces the memory requirement of the VLSI tester. It may also allow one VLSI tester to drive more than one test head at the same time. In addition, placing the IDDQ measurement close to the chip provides for high quality measurement.Each power entry point can be measured individually.

There is a cost in building a smart test head. The head circuitry must operate at least as fast as the chip under test. In addition, the head design must be general enough to be used on several different chips to keep the cost reasonable. Generalized circuitry means any I/O can be treated as either a signal or power pin. Thus, the head must be configurable to connect power and ground to the appropriate pins, and measure IDDQ on those, if desired. The head must be able to drive the signal I/O with appropriate patterns and to observe the results. If testability is used, the head must be able to decide on the correctness of the results.

If a fully configurable head is not feasible for the I/O count of the chips being tested, a partially configurable head can be used. In a partially configurable head some of the pins are dedicated to a given purpose, such as power. If the chip design methodology used for a family of chips dictates the purpose of certain pins, for example, the power and ground pins are fixed to specific locations in the I/O array, a partially configurable head will be more economical than a fully configurable head. With sufficient volume of chip production on a single chip a purpose built head for that chip may be economical.

These methods provide for normal production testing. But what about pre-production debug of chips to resolve design flaws, such as timing problems?

3.2 Probing

Adding circuitry to the chip for test, costs area, design time, and complexity. The goal is to add the minimum amount of overhead that provides adequate coverage. But even with the added test circuitry that provides sufficient coverage for production checkout the engineers may not be able to pinpoint the location all of the chip faults during debug. When this happens they turn to probing. With a voltage contrast (E-beam) probe they can observe many more signals and determine their timing. While this has been successful in many cases, the method has limits. It can only read the top layer metal, and it can only run on chips with access to that metal while the chip is running. Newer fabrication processes continue to add more metal layers which limits the visibility of many signals to the E-beam probe. It may be necessary to bring critical signals to the top layer expressly for probing. More importantly, this sort of probing requires the top of the chip be visible while in operation which limits it to chips with perimeter bonding pads. New designs utilizing an area array I/O across the entire chip are mounted face down which will foil the E-beam technique.

3.3 Probing for flip-chips

For debuging chips with area array I/O, a manufacture will either have to employ more test logic on chip, or use another technique for probing. Here are some other ideas for probing.

3.3.1 Optical probing

One new probing method has been created by IBM [31]. Called the Picosecond Imaging Circuit Analysis or PICA, it works on the fact that CMOS transistors produce photons when drawing current. This light can be observed with sensitive high speed photo-detectors through the back side of the chip giving a precise record of the signal timing. The signal reveals itself when a transistor is on and drawing current. Unless a transistor has a DC current path and always draws current, usually a fault case, the signal timing reveals itself during switching

when the output is charging or discharging and the crowbar current flows. You can, essentially, see the signals propagating through a chip. It was not clear from the literature, however, the optical resolution available. Can it resolve the emissions from individual transistors on a large deep submicron chip? It would have to have this capability to be useful.

3.3.2 Capacitive probing

The approach we investigated uses capactitve coupling to probe the chip to determine the signal timing. The idea is to use a very small capactitve probe to view the signal lines on the top metal layer of the chip you want to debug. The probe is actually a chip itself that acts as a carrier for the device under test and probes selected signals by coupling between the top layer metal on each chip. This is shown in Figure 3.1.

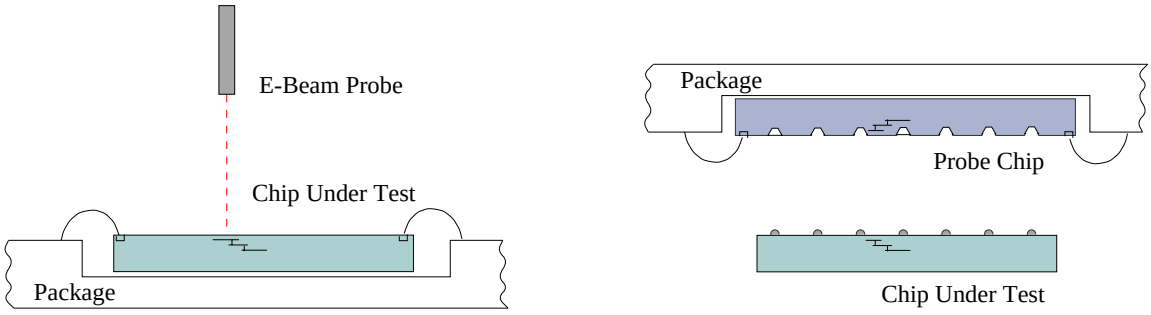

Figure 3.1 Conventional vs capacitive probing

The probe chip holds the chip under test, distributes the normal input signals to it, and probes the signals of interest. To maximize the coupling capacitance of the signal probe operating between the two chips their faces must be as close as possible. To reduce the gap, ideally to nothing, the solder bumps on the chip under test must be reduced to a very small size. We call these micro-bumps. This is not a big design change for the chip, only the overglass cut needs to change to reduce the exposed pad area and form a small solder ball. A corresponding hole is etched into the probe chip to accommodate the bump. During normal production the

overglass cut mask produces the regular opening for standard size solder balls. A close in view of the two chips together is shown in Figure 3.2.

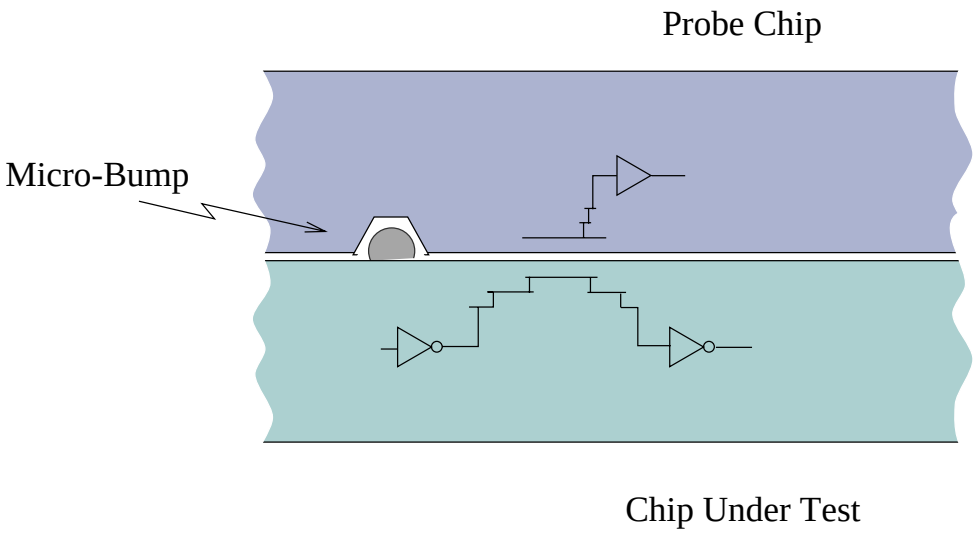

Figure 3.2 Probe chip coupled to the chip under test.

.Electrically the two chips form a circuit like that in Figure 3.3. The top metal layers on the

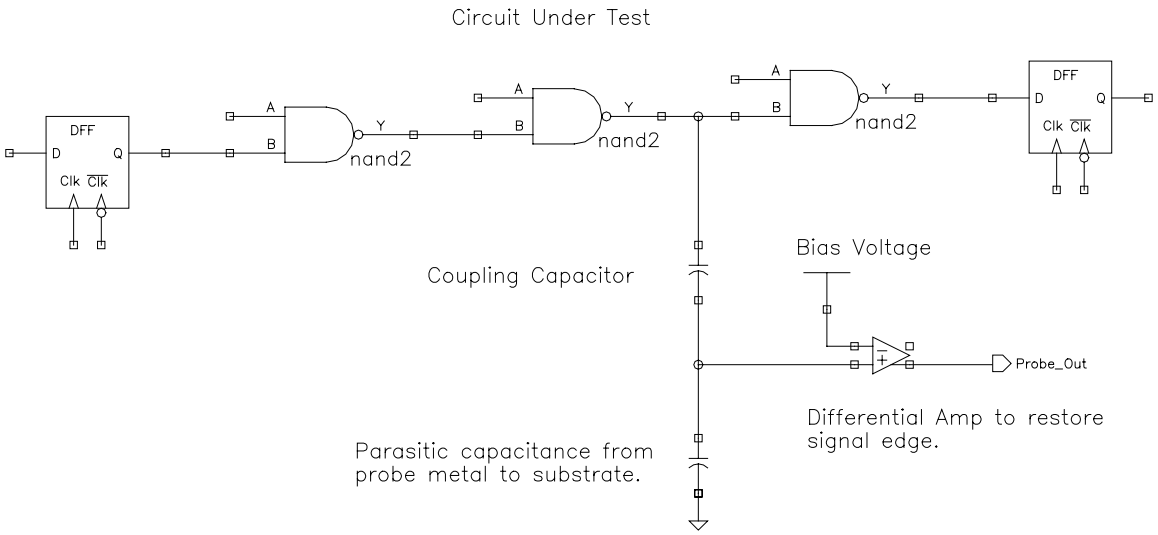

Figure 3.3 Probe electrical schematic.

chips forms a capacitor which couples the signals passing through the chip under test onto the probe chip. These signals on the probe chip are then amplified and passed to the edge of the chip where they can be observed.

Chapter 4 Capacitive Probe Chip

There are a number of design issues to consider in the probe chip. First, it has to mechanically support the chip under test and pass to it the normal signals. This involves constructing small solder balls on the chip under test and having a receptor site on the probe chip that mates to these balls such that the chip faces stay very close together. At the same time, the solder connections must still be electrically sound. Next, the nature of the capacitive coupling must be examined to ensure a viable signal path can be made between the two chips. Finally, a suitable receiver has to be constructed to regenerate the signal induced in the probe chip. This chapter explores these issues after a look at some previous uses of capacitive coupling for digital signals.

4.1 Capacitive coupling for digital signals

There have been some attempts to demonstrate capacitive coupling as a means of passing digital signals. Reference [50] argues for using capacitive coupling for all chip I/O signaling. In this scheme the normal conductive I/O array is replaced with plates in the top layer metal that form capacitors with the underlying substrate, which is assumed to be a multichip module (MCM). The signaling is then done with pulses that can pass through the capacitors. They suggest a value of about 1 pF for each pad. In their discussion of the electrical characteristics they state that this arrangement will have “substantially zero parasitics.” However, they have ignored the capacitance the I/O plate has with the chip substrate or wiring beneath the plate. This parasitic capacitance is substantial, on the same order as the coupling capacitance. In fact the entire discussion forms a high level argument for capacitive coupled signaling with little detailed analysis of any specific cases.

Reference [51] presents a detailed design of a capacitive coupled receiver circuit. This design creates the capacitor under the wire bond pads of the chip with a value of about 1.6 pF. The top layers of metal connect to the wire bond, as in a normal chip. These layers also connect to an equivalent size plate of poly lying under the entire pad. This group forms one plate of the capacitor. Sandwiched between these layers is a block of metal 1 which forms the receiver plate of the capacitor. The arrangement isolates the receiver plate from the parasitic capacitance to the substrate. Two pads form a differential pair connected to a receiver circuit that detects the signal edges and restores the normal, level based digital signals. The receiver

uses feedback to prevent the signal decay or “zero wander” of the coupled signal and holds the digital signal at the correct levels. The design is useful for connecting chips that use different DC voltage levels for bit representation since the DC component does not pass through the capacitor. But it requires differential signaling and uses a total of 3.2 pF capacitance per signal pair. The probe chip only has single ended signals available and a capacitance value that is orders of magnitude smaller than either of these approaches.

4.2 The mechanical connection

The first problem is to connect the chip under test to the probe chip and keep their faces as close together as possible, ideally touching. This was depicted previously in Figure 3.2. The chips connect through the usual solder bump interconnections the chip under test will ultimately employ for bonding to the MCM substrate. However, to achieve the second requirement pits must be etched into the probe chip to accept the bumps on the test chip and keep the faces close. If the normal size solder bumps of around 100 μm diameter are used then the craters would have to be huge, over 50 μm deep. Such large connections are not really necessary though, since they do not have to pass the normal reliability criteria as they will only be used for chip debug and not in normal service. Therefore, smaller bumps, call them micro bumps, of around 20 μm diameter can be used. This only requires a depth of something over 10 μm. These holes have insulator grown, a normal signal conductor metal and a wettable metals deposited to form the receptors for the solder bumps. The chip under test only needs smaller overglass cuts to reduce the bump size. The mask changes for the chip under test are, therefore, small.

Another factor in how close the faces of the two chips can get is how flat those faces are manufactured. Most processes use some form of chemical-mechanical planarization (CMP), a.k.a. chemical-mechanical polish, to assure flatness during manufacturing. References [94] and [95] indicate CMP can achieve a high degree of flatness. They report a half micron variation over a three inch wafer.

Normally the top layers of a chip, the overglass layers, are not flattened since there are no further layers to be deposited. In the case of the probe chip and chip under test it would be helpful to planarize these layers to allow the top layer metal traces to rest as close together as possible. However, for the capacitance simulations I assumed the chips top layers were not planarized.

With the chips flat and pushed as close together as possible one would hope for a perfect match between the chip faces where glass touches glass and there are no gaps. Air, as everyone knows, makes a poor dielectric. Capacitors with air gaps between the plates do not generate a very high value of capacitance. In practical terms it is impossible to have the chip faces perfectly touching. The resulting air gap will ruin the coupling we are trying to achieve between the chips. Thus, it is necessary to fill those gaps with a high dielectric material to promote coupling. A liquid glycol fill, with $\varepsilon_r = 80$, can do the task. It will wick into the space between the chips and provide a good coupling dielectric.

There are other chip coupling mechanisms beside solder bumps. This work focused solder bumps because they are the most common area array connection technology. There are alternatives, such as conductive polymers [1] and other technologies, that may be usable. However, these are not widely employed.

4.3 The coupling and parasitic capacitance

To investigate the values of the various coupling and parasitic capacitances in the circuit formed by the joined chips I used the fastcap field solver to simulate different stacks of conductors and dielectric. These stacks represent a cross section of the two joined chips. The fabrication process chosen has a 0.35 μm effective feature size and the conductor and dielectric thicknesses and spacings were taken from the process manual. The process was assumed to be planarized so the tops of all the dielectrics, except the overglass, were treated as flat planes. The overglass was made conformal over the top layer metal. Also, the simulation assumed the top of the substrate is a perfect conductor. In reality it is not a perfect conductor which would increase the effective distance to this plate when used as a capacitor, lowering the capacitance. Therefore, because coupling to the substrate creates a parasitic capacitance, this simplification presents the worst case.

In generating the conductor stack for analysis the idea was to construct a circuit that had at least 1 fF of coupling capacitance. Preliminary analysis indicated this was a the smallest value that could form a workable coupling signal. Most metal lines on the chip do not make very good capacitor plates. For normal operation this is a good property. However, when set-

ting out to make an intentional coupling capacitor this is a problem. Figure 4.1 shows a depic-

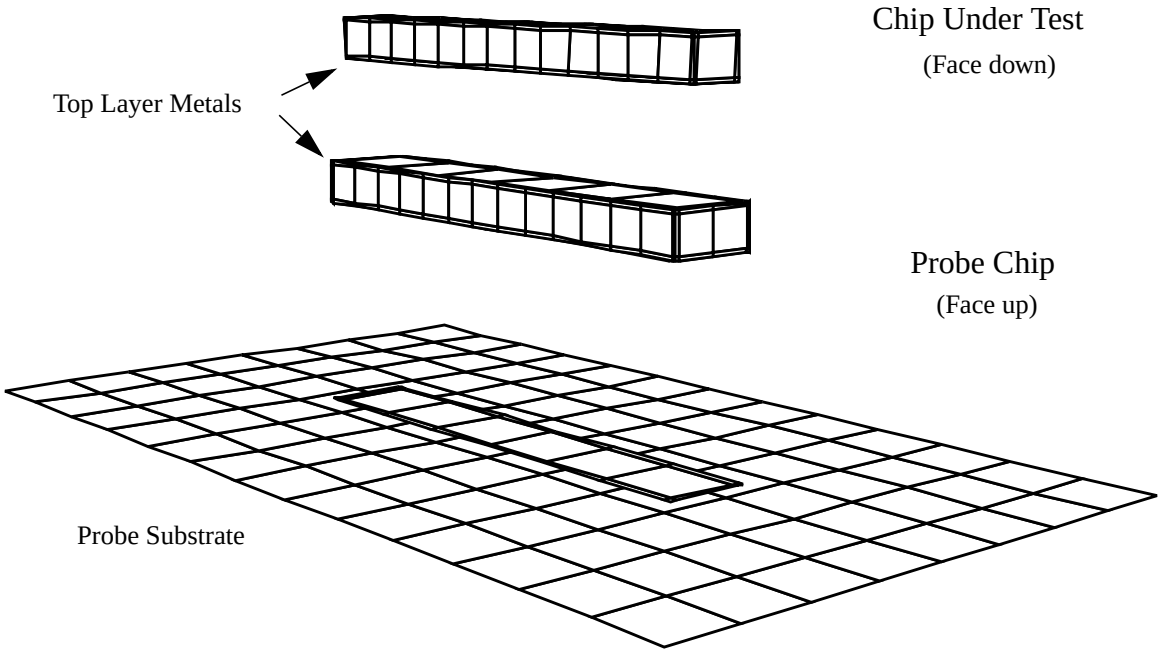

Figure 4.1 Coupled conductors over substrate

tion of a probe chip and a chip under test using normal thin metal lines for coupling. The dielectric interfaces were removed for clarity. The chip faces were assumed to not touch. Instead a gap of 1 or 2 μm was used. The metal on the probe chip was made 1 μm wider than the metal on the chip under test to increase the coupling and accommodate misalignment. This case requires a length of around 14 μm to achieve a 1 fF coupling value. And, as shown in Table 4.1, this geometry gives a large parasitic capacitance, resulting in a poor coupled to parasitic ratio. Furthermore, in the presence of a second conductor on the chip under test, parallel to the one of interest, the probe metal will pick up a large crosstalk component from this

second conductor. This is depicted in Figure 4.2 and also shown in Table 4.1. With the large

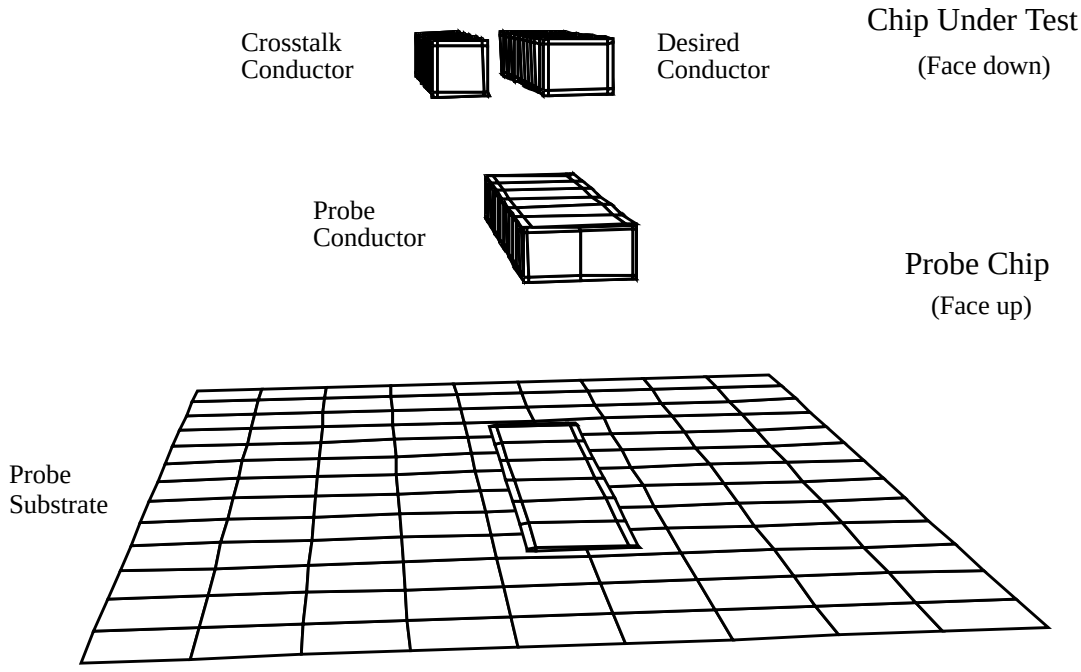

Figure 4.2 Coupled conductors with a crosstalk conductor

Table 4.1 Probe capacitance for long thin conductors

		Chip to Chip Gap	
	Capacitance Type	**1 μm**	**2 μm**
No Crosstalk Conductor	Coupling Cap	1.215	1.015
	Parasitic Cap to sub-strate	1.013	0.981
	Parasitic Cap to other sources	0.895	1.239
	Parasitic Cap total	1.918	2.221
	Coupling/Parasitic Ratio	.633	.457

Table 4.1 Probe capacitance for long thin conductors

	Capacitance Type	Chip to Chip Gap	
		1 μm	2 μm
With Crosstalk Conductor	Coupling Cap	0.853	0.700
	Crosstalk Cap	0.663	0.596
	Parasitic Cap	1.628	1.937
	Coupling/Parasitic Ratio	.523	.361

Capacitance values in fF

Desired conductor 1.5 x 14.1 μm Probe conductor 2.5 x 14.1 μm

amount of parasitic capacitance relative to the coupled capacitance the signal propagated to the receiver circuit may be too small to be of use. Also, the magnitude of the crosstalk component may overwhelm the signal.

To improve the coupling the conductors must change to a more rectangular geometry. Using a square shape 6.3 μm on a side, as shown in Figure 4.3 with a crosstalk conductor,

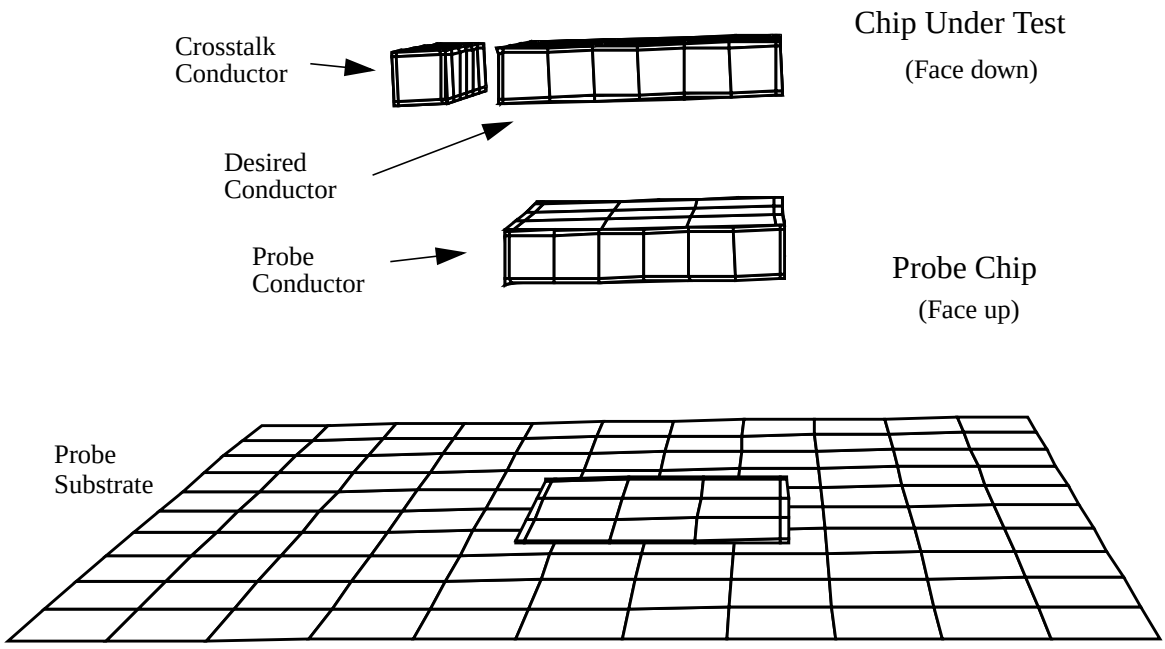

Figure 4.3 Coupled conductors having a plate structure

greatly improves the coupled to parasitic ratio. Table 4.2 shows these results. Now the cou-

Table 4.2 Probe capacitance for rectangular conductors

		Chip to Chip Gap	
	Capacitance Type	**1 μm**	**2 μm**
No Crosstalk Conductor	Coupling Cap	1.323	1.095
	Parasitic Cap	1.603	1.957
	Coupling/Parasitic Ratio	.825	.560
With Crosstalk Conductor	Coupling Cap	1.163	0.954
	Crosstalk Cap	0.295	0.272
	Parasitic Cap	1.487	1.816
	Coupling/Parasitic Ratio	.782	.525

Capacitance values in fF
Desired and probe conductors 6.3 x 6.3 μm

pling should be sufficient to provide a useful signal. Also note the crosstalk component remains relatively small.

The parasitic capacitance given in Tables 4.1 and 4.2 includes all the capacitance not coupled to either the desired conductor or the crosstalk conductor. Part of this parasitic capacitance goes to the substrate and the rest to other stray sources as indicated in Table 4.3. This

Table 4.3 Breakdown of the parasitic capacitance

		Chip to Chip Gap	
	Parasitic Capacitance	**1 μm**	**2 μm**
No Crosstalk Conductor	to substrate	0.883	0.848
	to other sources	0.719	1.108
With Crosstalk Conductor	to substrate	0.892	0.872
	to other sources	0.549	0.943

Capacitance values in fF, rectangular conductors.

stray component increases as the distance between the chips increases because the intervening space is filled with the high dielectric strength fluid. As the distance increases somewhat

fewer field lines couple between the probe and the desired conductors, but a greater number of stray field lines in the gap can terminate at the probe conductor or the desired conductor. Thus, a larger gap increases the parasitic capacitance at both the probe conductor and the desired conductor, and both the probe conductor and the desired conductor see a similar effect.

The total load on the desired conductor is equal to the normal on-chip load plus the extra load added by the test fixture (probe chip). The normal load consists of the capacitance of the devices attached to the line, which are not indicated in the fastcap simulations, and the usual parasitic capacitance for the line. The load on the desired conductor imparted by the test fixture equals the coupling capacitance plus the stray parasitic capacitance. For the coupled section the stray capacitance is about 1 fF (Table 4.3) and the coupling capacitance is at most about 1.4 fF (Table 4.2). Under normal operation the same section would have air above it and would have a stray capacitance of about 0.6 fF. This is shown in Table 4.4. Therefore, the probe chip adds approximately 1.8 fF to the line load.

Table 4.4 Parasitic capacitance of the chip under test in air

	Parasitic Capacitance	**Value**
Long thin conductor	to substrate	1.159
	to other sources	0.640
Square conductor	to substrate	1.013
	to other sources	0.578
Capacitance values in fF.		

However, the coupled section is normally attached to additional wiring. The additional wiring would connect the coupled section to the rest of the chip under test. The load capacitance for a length of (thin) conductor that could be used for such a purpose is about 2.2 fF per 14 μm of line length (Table 4.1) when used with the probe chip and 0.64 fF when alone. So the probe chip adds about 1.6 fF per 14 μm length.

For example, assume a total line length of 34 μm consists of two 14 μm sections and the 6 μm coupled section. The total additional load presented by the probe chip would be 5.0 fF.

4.4 The receiver

4.4.1 Circuit description

The receiver takes its input from middle of the coupling capacitor—parasitic capacitor series pair. The small value of the coupling capacitance drives the receiver design. It must be able to regenerate the signal from the small charge passing through the capacitor network. To do so it must have a high impedance to get the maximum voltage swing from the imparted charge. The loading is critical because the small amount of charge passed through the capacitor would be ineffective if the load impedance is too low. A differential amplifier, also called a transconductance amplifier in [12], connected as a single-ended receiver fits these requirements. The amplifier, shown in Figure 4.4, presents only one gate load, transistor M1, and a biasing resistance, R2, to the driving capacitors. Resistor R1 acts as a load to transform the output current from the differential pair into an output voltage used by the next stage. The

gain is not as high as when connected as a true differential pair with current mirror loads, but it has sufficient gain to recover the signal. When deployed with a second stage the full output swing is achieved. Details of the biasing will be discussed in a moment.

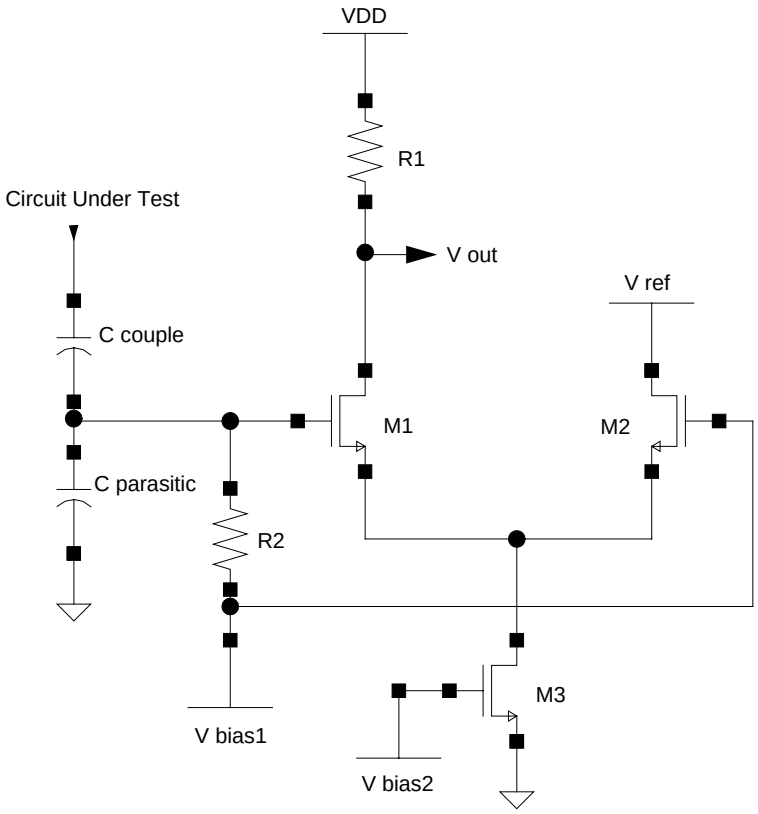

Figure 4.4 Single-ended differential receiver

Because of the way the differential pair is used it is possible to collapse the functionality of the reference leg formed by transistors M2 and M3 into a single transistor. In [91] the authors demonstrate a functional equivalence in the current response between a pair of FETs operating in the saturation region to a single FET operating in the linear region. This happens because the drain current of a saturated transistor depends only on the gate voltage, Vgs. The drain-source voltage, Vds, has little impact on the current. Thus, there is only one component to the current, the forward current I_F, as determined by the gate voltage.

$$Ids = \frac{W}{L}\frac{\beta}{2}(Vgb - Vsb - Vth)^2 = I_F \tag{4.1}$$

In the linear region both Vgs and Vds impact the drain current. So there are two components to the current, forward I_F and reverse I_R.

$$Ids = \frac{W}{L}\frac{\beta}{2}[(Vgb - Vsb - Vth^2) - (Vgb - Vdb - Vth)^2] = I_F - I_R \quad (4.2)$$

Imagine each of these two components of the drain current, I_F and I_R, in the linear transistor being replaced with an equivalent current generated by a saturated transistor. Then two transistors in saturation would be functionally equivalent to a single transistor in the linear region, as depicted in Figure 4.5.

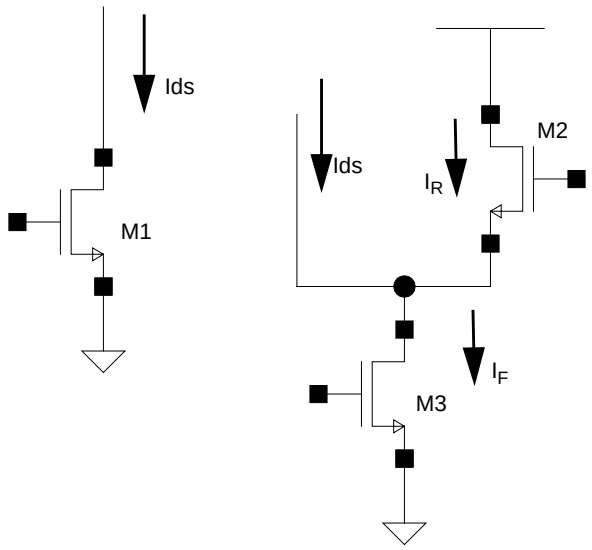

Figure 4.5 Functional equivalence of one transistor in the linear mode and two transistors in saturation

Using this equivalence we can replace the two transistors in the reference leg of the differential amplifier operating in the saturation mode with a single transistor in the linear mode.

For biasing we need to replace resistors R1 and R2 with components buildable in MOS. For the load resistor R1 we can use a PMOS transistor biased on. Changing the length and width of the channel sets the resistance. Constructing R2 is more complicated because of the high value needed to minimize the load on the input line. Unfortunately, high value resistors are difficult to build in MOS processes. The solution is to use a pair of diode connected NMOS transistors in series. With these added the complete receiver is shown in Figure 4.6.

The voltage drop across each of the diode connected bias transistors, M4 and M5, is higher than normal due to significant body effect which raises the threshold voltage. These bias transistors allow the receiver input, called “comp” in the schematic, to charge up to the low input voltage, V_{IL}, for the receiver. At that point the transistors reach cutoff giving them a very high resistance. A positive pulse from the coupling capacitor drives the bias network further into cutoff, increasing the resistance further. The gate of M2, which servers as the reference for the differential pair, maintains a higher bias voltage than M1, the signal input. This sets the baseline current through the differential amplifier. The bias and size of the reference transistor, M2, and size of the load transistor, M3, determine the output voltage operating point at net called out bar. The input and reference transistors, M1 and M2, are set at the same size so they share a common diffusion so they will be as closely matched as possible. A second stage

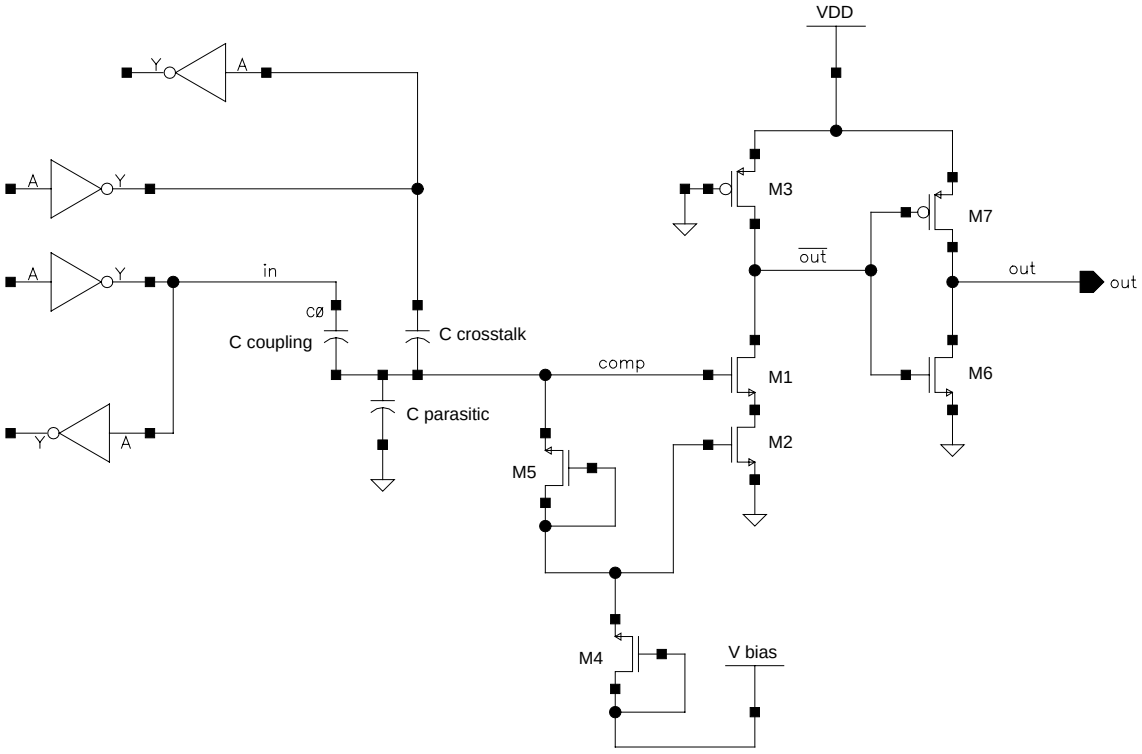

Figure 4.6 The complete receiver circuit

inverter consisting of M6 and M7 restores the output to full voltage swing and completes the receiver. The transistor sizing is shown in Table 4.5.

Table 4.5 Receiver transistor sizing

Transistor	Length	Width
M1	0.5	0.8
M2	0.5	0.8
M3	0.7	0.8
M4	0.6	0.8
M5	0.6	0.8
M6	0.4	0.8
M7	0.4	0.8
All lengths in microns. HP CMOS 10 process, MOSIS SCMOS_SUBM rules. Min Length = 0.4 µm Min Width = 0.6 µm		

Using a differential amplifier helps minimize noise interference. Because M1 and M2 are in the same diffusion, substrate noise will likely be common to both. Likewise, noise on M4 will go into both inputs. This common mode noise should have little impact on performance. Only noise into M5 will have a direct impact on the input. Using a separate clean bias supply, and substrate guard rings will minimize noise introduction.

4.4.2 Receiver performance

The receiver performance was determined using HSpice from within the Cadence Analog Artist environment. The transistor models were level 28 with binning obtained from the process manufacturer. The process had a 0.4 µm drawn minimum feature size. As shown in Figure 4.7, the input signal sources were buffered through a transmission gate selector and two levels of gates before feeding the coupling capacitance network. Also connected to this net, called “in”, are additional capacitors used to vary the loading of the drive circuit and a non-inverting buffer. The output of this buffer, net name “ckt”, served as the golden reference to

compare to the output of the receiver. In addition, the output from the inverting gate in the input buffer feeds the crosstalk capacitor in the coupling network providing the worst case crosstalk to the receiver.

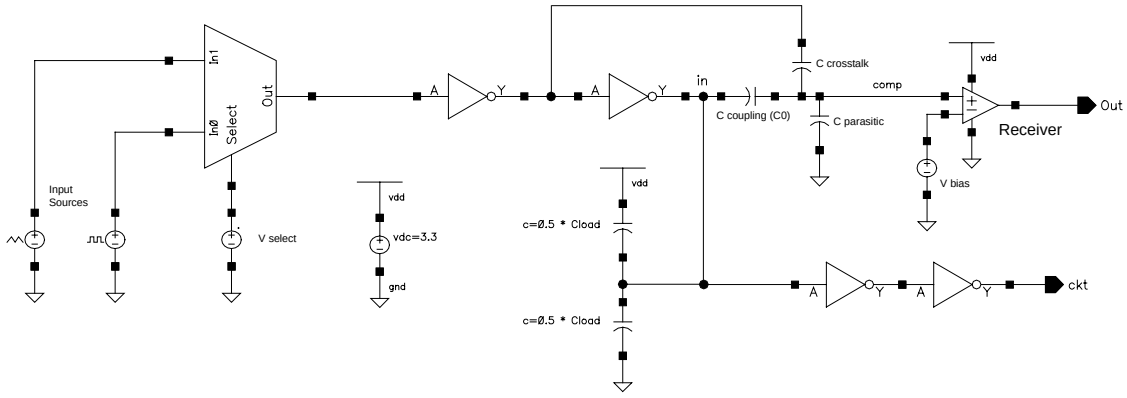

Figure 4.7 Receiver test bench

A number of runs were conducted using parametric analysis to set the transistor sizes. Most of the transistors in the receiver have a gate length longer than minimum to reduce the performance fluctuation due to process variation. This is an analog design technique that makes the receiver stable over a wide range of conditions. Reference [92], particularly Figure 1A, shows the variation of gate length in a similar size fabrication technology.

The capacitance network takes the values from Table 4.2. The nominal value for the coupling capacitance was set to 1.2 fF, the parasitic capacitance was set to 1.4 fF, and the crosstalk capacitance was set to 0.275 fF.

4.4.2.1 Nominal performance

The nominal performance is shown in Figure 4.8. Charge from the signal of interest, “in”, increases the voltage on the receiver input, “comp”, from its resting point set by the bias supply. The effect of the out of phase crosstalk net propagating charge through the crosstalk capacitor shows up as the dip in the rising edge of the comp signal (near the point where the “in” signal crosses the “comp” signal). As the comp voltage rises, current in the differential amplifier rises, dropping the voltage at the output of the amplifier, out bar. This drives the second stage inverter to swing the output from low to high. For the falling transition the input drains charge away from the comp net and the voltage falls. This reduces the current out of

the differential pair allowing the pull-up load to increase the voltage at out bar. This, of course, drives the second stage inverter high to low. The voltage gain of the differential amplifier is 3.7.

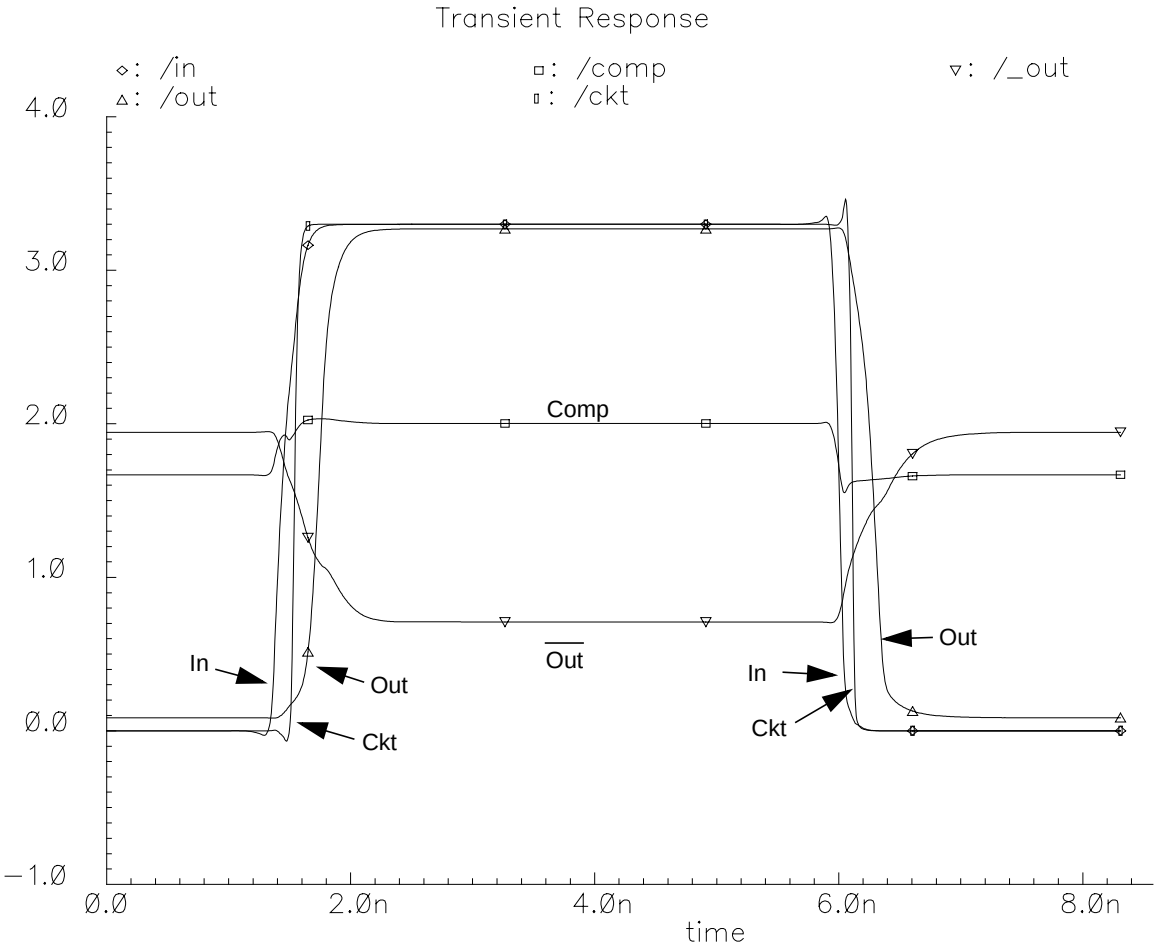

Figure 4.8 Nominal receiver performance

4.4.2.2 Effect of process variation

To investigate the effect of process variation I used the transistor corner models available for this process. The models show the type of variation that may be expected from batch to batch in a chip run. The amount of change indicated in the models is much wider than one would expect within a single chip. Reference [93] shows a distribution of threshold voltages across chips on a wafer. The variation of the voltage on a single chip is much smaller than the

total variation across all chips on the wafer. Thus the models are too wide to apply to an individual chip.

The simulation runs with the corner models, shown in Figure 4.9 for slow transistors and Figure 4.10 for fast transistors, indicate the receiver operating point changes with the model. By making a small correction to the bias voltage the receiver can be tuned to operate very close to nominal.

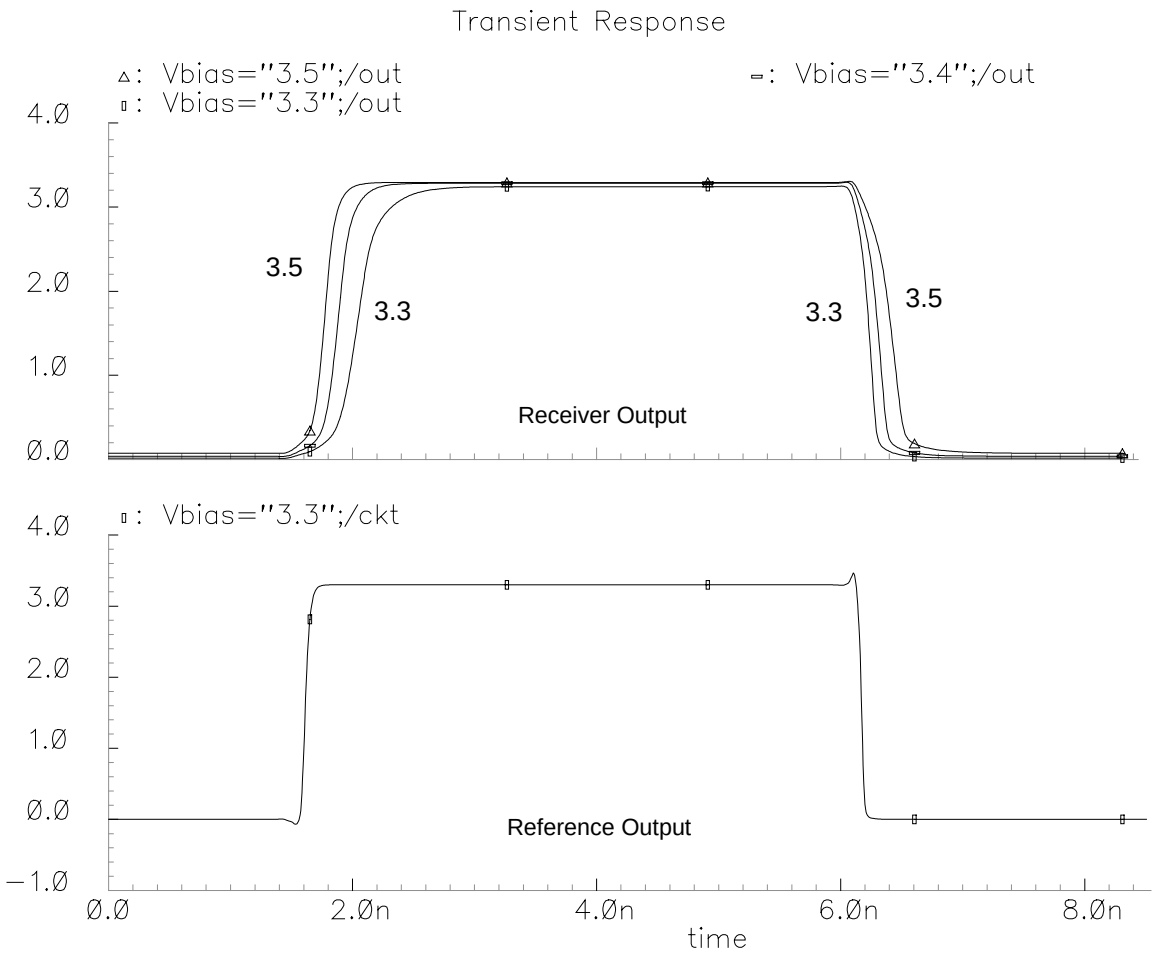

Figure 4.9 Receiver response, slow corner

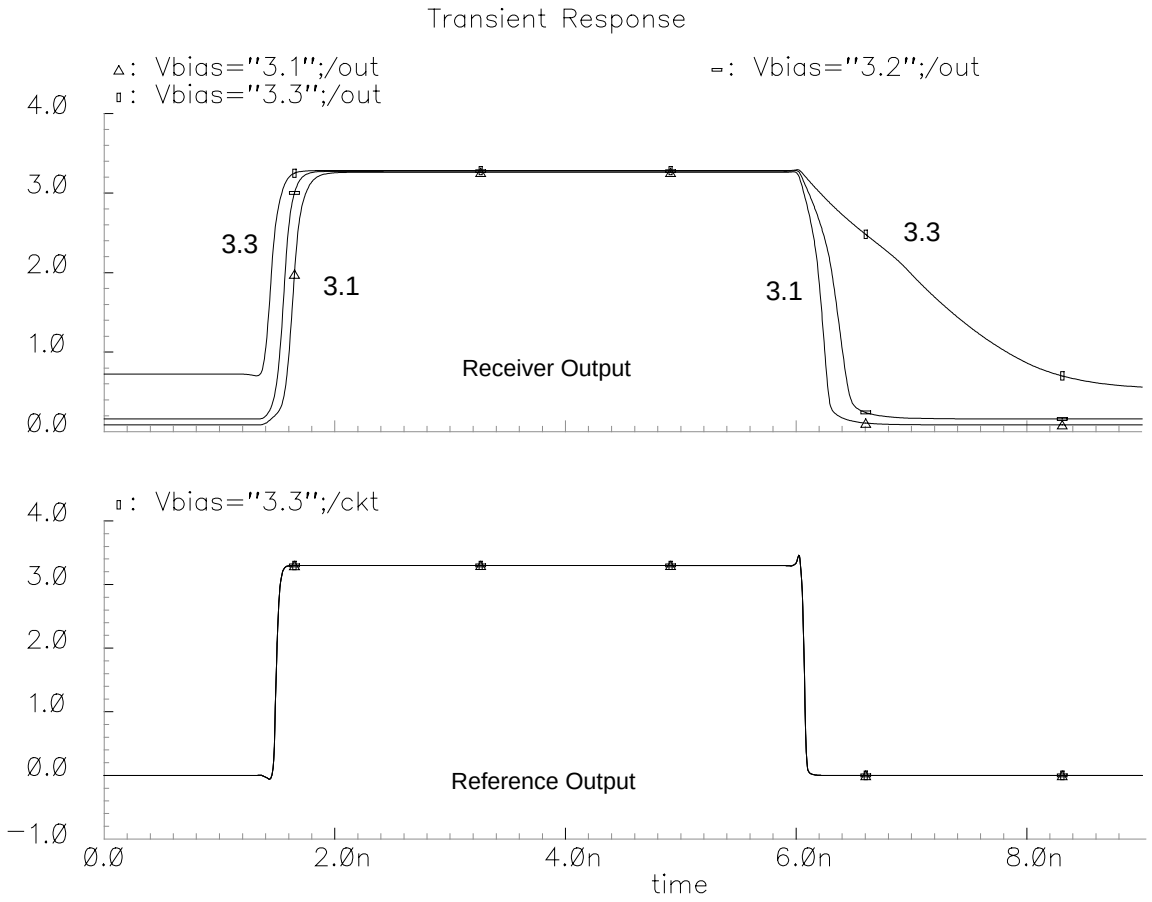

Figure 4.10 Receiver response, fast corner

4.4.2.3 Effect of coupling capacitance variation

It is also important to understand how the receiver performs if the values of the capacitors in the coupling network vary. For these experiments I varied the capacitance values over the range indicated in Table 4.2. That is, I set the coupling capacitance from 1.0 fF to 1.4 fF, and the parasitic capacitance from 1.4 fF to 2.0 fF. The crosstalk capacitance stayed at 0.275 fF since the fastcap simulations indicate it does not vary a significant amount. Each parameter was varied separately while the others were held at their nominal values, 1.2 fF for the cou-

pling capacitance and 1.4 fF for the parasitic capacitance. The effect of varying the coupling value is shown in Figure 4.11 and Figure 4.12.

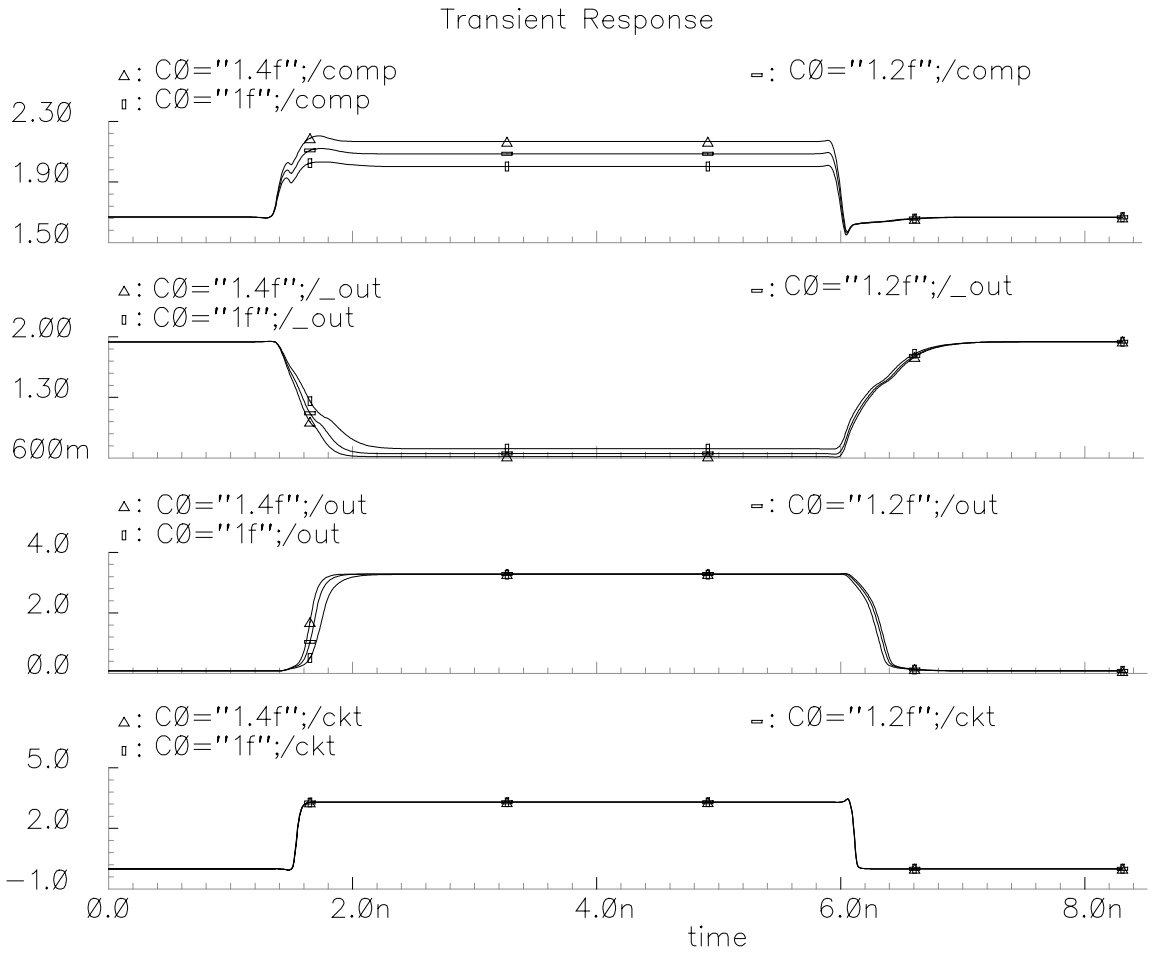

Figure 4.11 Effect of varying coupling capacitance, all nodes

As one would expect, the change in the coupling capacitance makes a marked impact on the amount of charge transferred to the comp net. Hence, the voltage range is quite reduced as the capacitance diminishes. The lower voltage at the smallest coupling value presents the most difficult challenge to the receiver. The output is still acceptable, as shown in Figure 4.12, but you can see how the slope of the probe output is affected. It is particularly evident on the rising edge where the out of phase crosstalk signal slows the rising edge of the comp net. It is no surprise that from the receiver's point of view it is always desirable to have as large a coupling capacitor as possible.

Another characteristic to note is that as charge is taken out of the comp net and the voltage approaches the bias point, if the input signal is sufficient to drive the voltage below the bias level the bias transistors will turn on to restore the voltage. This can be seen in Figure 4.11 in the small dip at the end of the falling transition of the comp node. In effect, the bias network will never let the voltage drop below the bias level on the comp net.

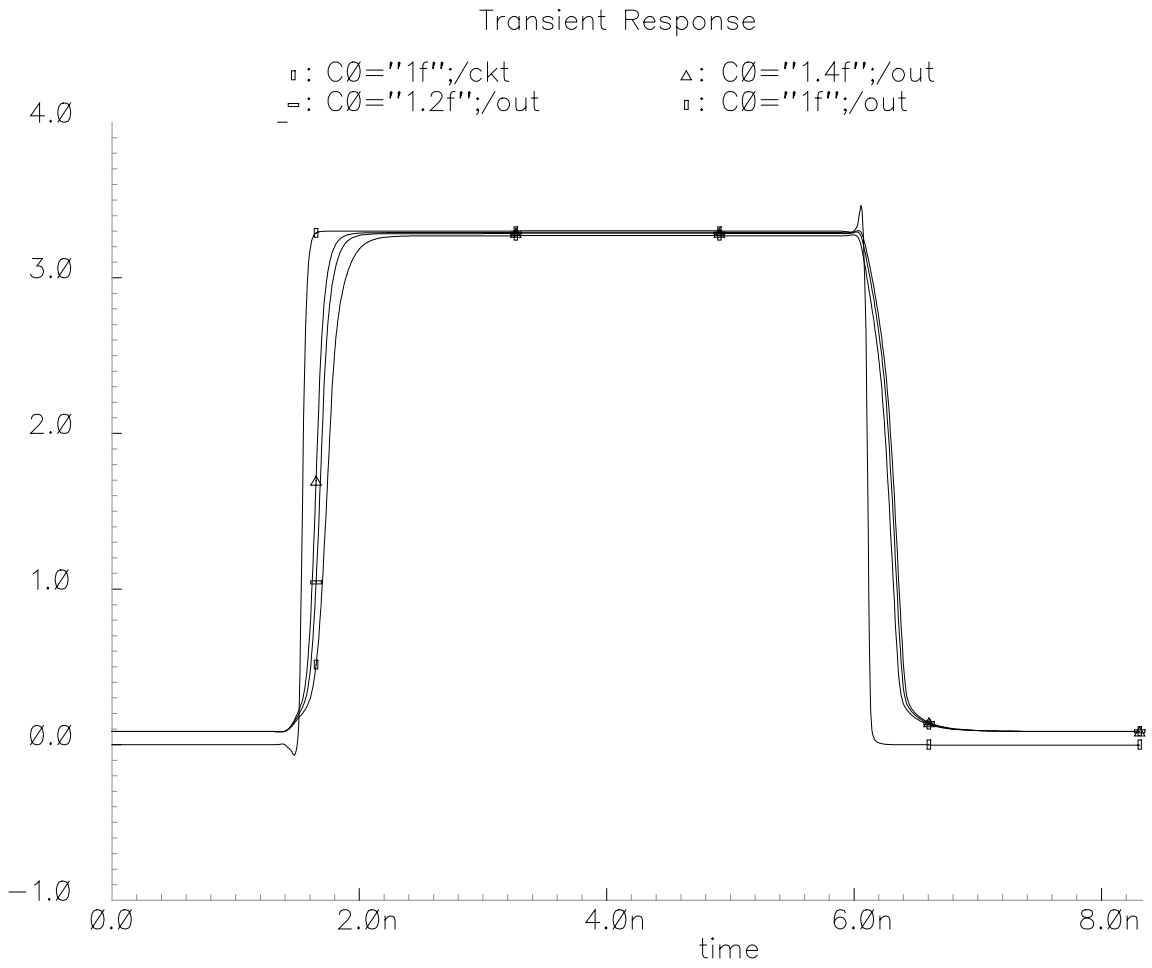

Figure 4.12 Varying coupling capacitance, circuit output vs. probe output

4.4.2.4 Effect of parasitic capacitance variation

The effect of varying the parasitic capacitance value is shown in Figure 4.13. As one would expect the parasitic variation has less effect on the circuit performance than the coupling capacitance. In fact, the entire range of parasitic values change the shape of the probe output

very little. This reinforces the notion that a larger coupling capacitance is beneficial, even if it increases the parasitic capacitance, because an increase in parasitic capacitance has little consequence, whereas an increase in the coupling capacitance strongly influences the output.

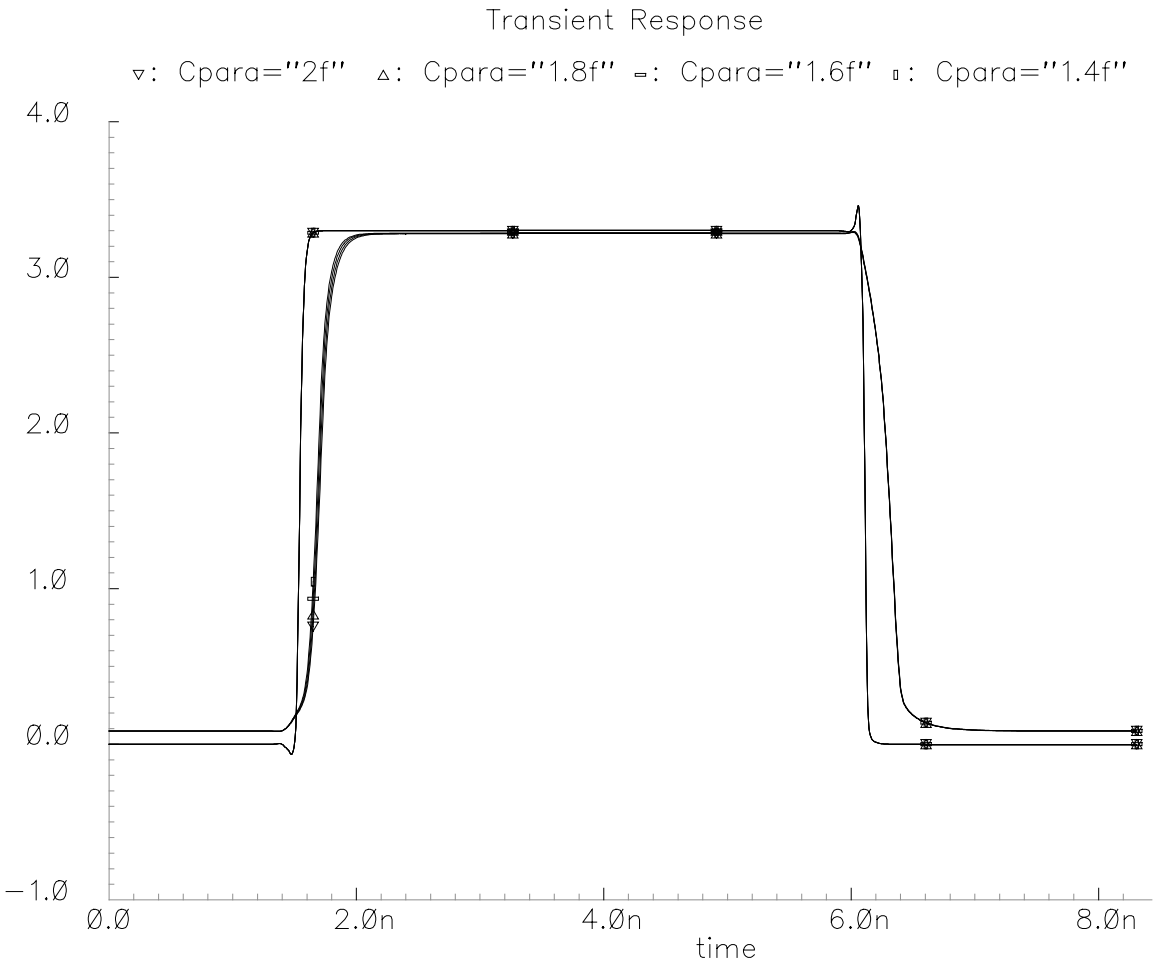

Figure 4.13 Varying parasitic capacitance, circuit output vs. probe output

4.4.2.5 Capacitive charge decay time

As with all capacitive coupled amplifiers the input bias is important. Since the bias network in this design employes no feedback, when the comp net is driven to its high input voltage, V_{IH}, the charge on the node will eventually decay back toward the bias point. In other words, when there is no signal the receiver input (comp) will always move to V_{IL}. This results in two consequences. First, the receiver must transition from low to high (rising transition)

before going from high to low (falling transition). If a falling transition were to happen before a rising transition the comp node would simply be driven to turn on the bias network and the node would be held at V_{IL}. Second, the high to low transition must occur before the signal has decayed to the point of switching the output of the receiver. Eventually the output of the rising transition will decay below the threshold required to keep the output high. Once this happens the edge a falling transition is lost. Figure 4.14 shows the receiver can hold at the high output for about 1µs. For a system running at 100MHz (10ns cycle time), a typical clock rate for this process technology, the test would have 100 cycles to trigger a falling edge. Using a more conservative 800ns to remove any doubt as to the validity of the cause of the edge the test would still have 80 cycles to trigger a falling edge.

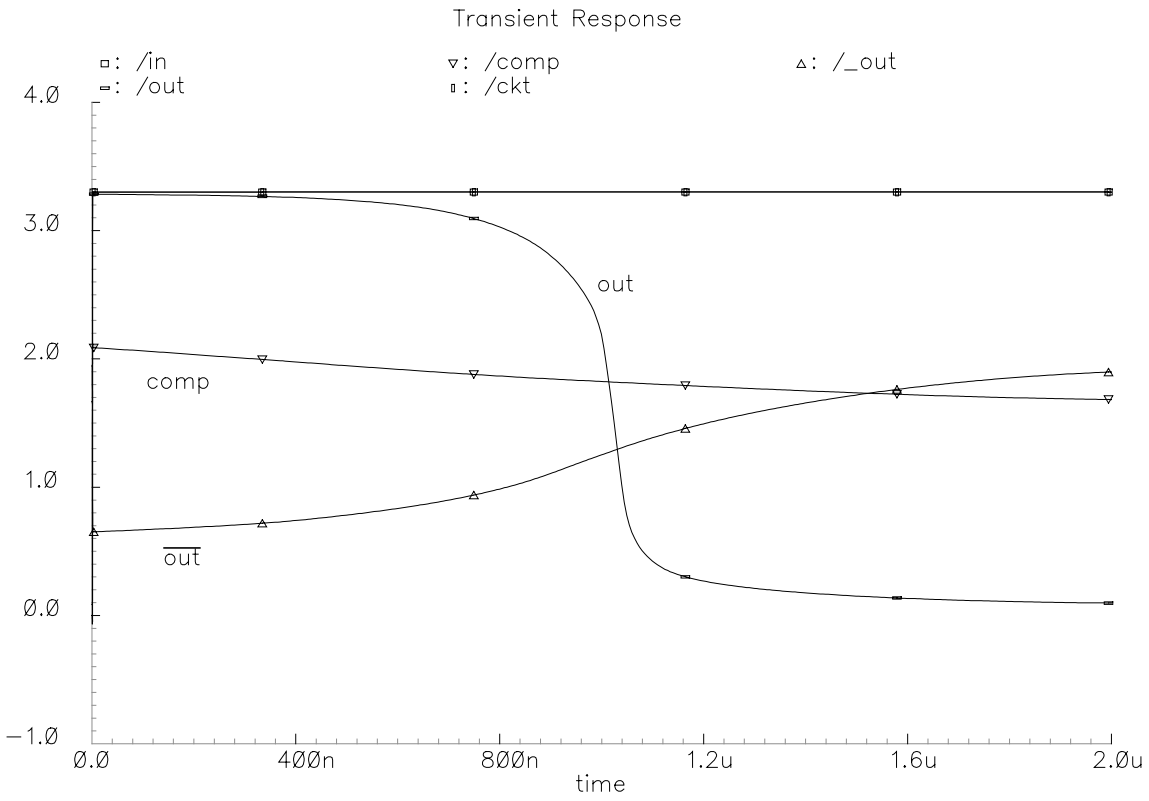

Figure 4.14 Output decay

4.4.2.6 High speed performance

At the other end of the range a designer may be concerned with the maximum rate the receiver can switch. It takes more time to propagate a signal through the coupling capacitor, charge the parasitic capacitor, and change the state of the receiver than just driving an output gate directly. Figure 4.15 shows the output of the probe versus the output of the reference circuit at several periodic input signal rates. The fastest rate shown is 700 ps, which is over 1.4 GHz. This should allow use of the probe with the fastest digital signals generally in use with this process technology.

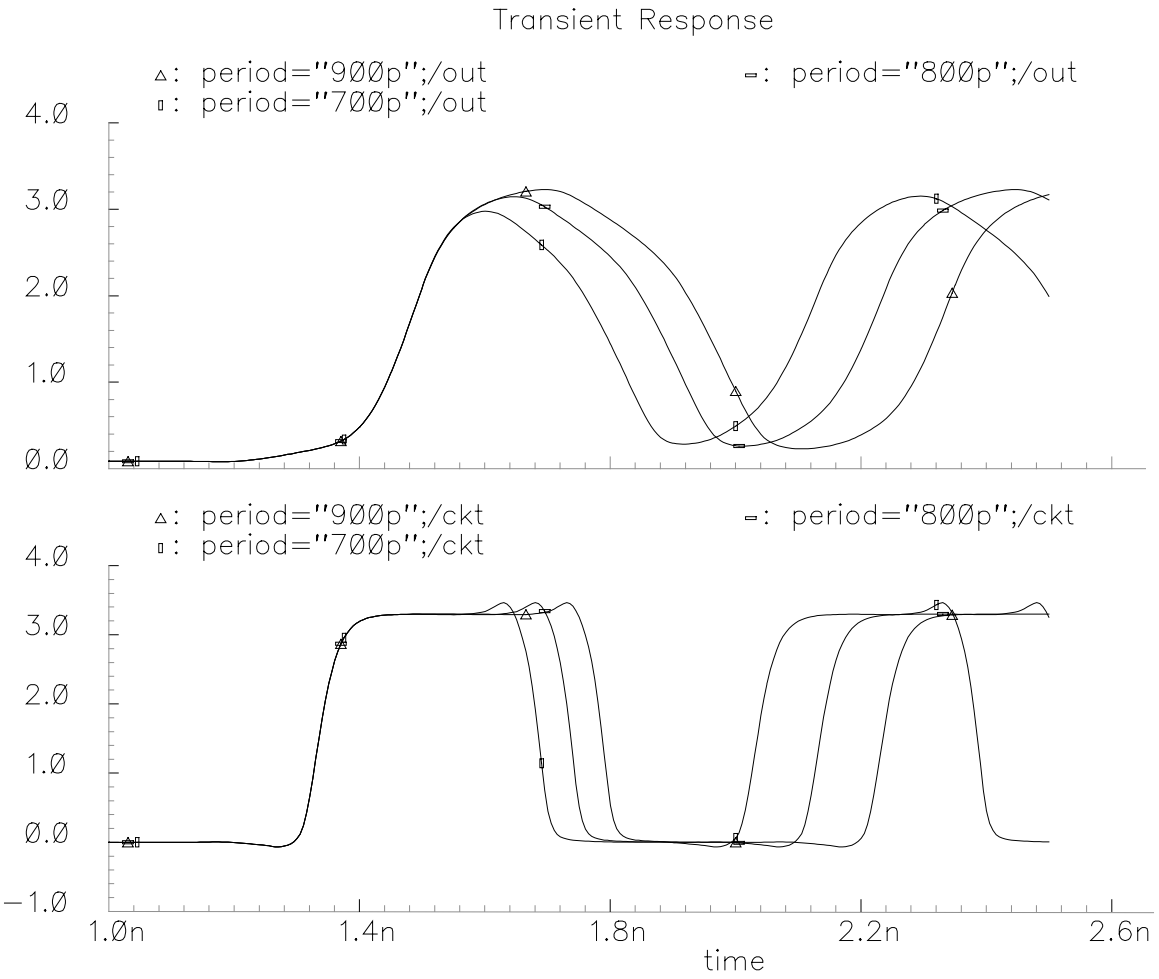

Figure 4.15 High speed operation

4.4.2.7 Edge timing

Figure 4.16 shows the response of the circuit and probe output superimposed while the input period varies from 500ps upward. At 500ps the probe cannot respond effectively to the input. Above that, however, the probe and the circuit match well. Ideally the difference in the circuit edge and the probe edge should remain constant regardless of the cycle time of the input signal. Taking the X intercept at the 50 percent crossing, Table 4.6 shows that the minimum difference for the rising transition is 127ps and the maximum difference is 141ps. The variation is 14ps. The minimum value occurs at when the input period is 1ns. At 2ns input period and above the variation is only 3ps. The falling transition variation is identical, 14ps when the input is 1ns and 3ps when the input is at or above 2ns. This holds out to a period of 2µs.

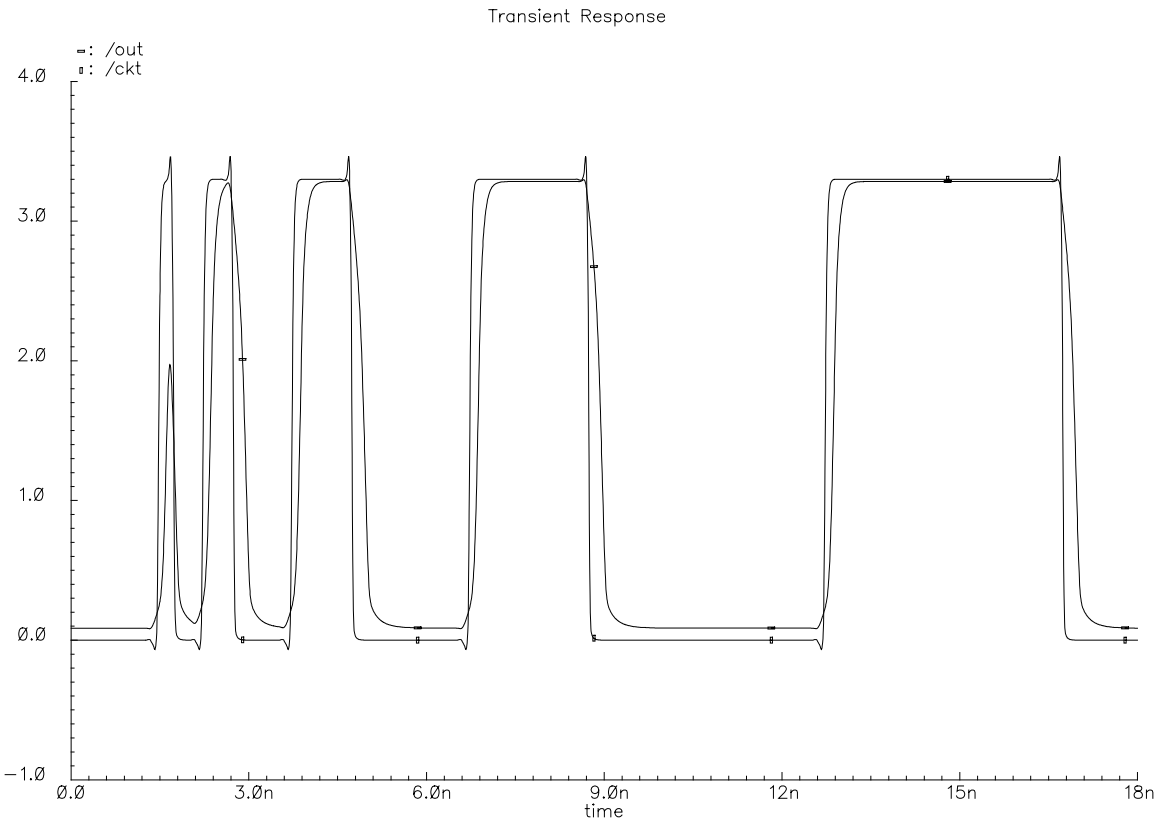

Figure 4.16 Output response for varying input periods

Table 4.6 Edge difference for varying input periods

Input period (ns)	Rising edge difference (ps)	Falling edge difference (ps)
0.5	145	-16
1	127	181
2	138	195
4	141	194
8	141	193
16	142	193
32	141	194
64	142	193
128	141	194
256	141	194
512	141	194
1024	141	195
2048	140	195

4.4.2.8 Effect of load variation and crosstalk

Finally, the response was measured as the load capacitance on the test circuit, the “in” net, varied. The capacitors, Cload in Figure 4.7, act as a variable load, simulating the driver in different fanout conditions. Figure 4.17 shows the driver signal, in, and the receivers responses. The load was varied in increments up to the point where the input edge rise and fall times were about an order of magnitude slower than the case with no additional load. The exact change in rise and fall times is 11 to 12 times slower.

As with the previous case the difference in the edge timing between the buffer circuit and the probe circuit was measured at the 50 percent voltage point and summarized in Table 3.6. What stands out is how the edge difference increases as the load increases and the driver edge flattens out. It appears as though the probe triggers at a different point along the input edge than the buffer circuit, causing the difference in timing. This could be explained by the nature of the circuitry. The capacitive coupled probe takes longer to accumulate enough charge to effect the switch in its output stage. However, the actual causes are more complicated.

First, part of the reason the rising and falling edges have a dissimilarity in the rate of increase of the delay difference is because the response of the gate driving the “in” net has a different rising response versus falling response. This happens because the transistors in the gate are the same size, and the n-channel device has an inherently greater drive than the p-channel device. The net result is unequal rise and fall times, which is especially pronounced when the load increases.

Second, by looking back at the setup of the test bench in Figure 3.7 you can see the crosstalk signal passing through Ccrosstalk is not changed by the increasing load. So the effect of the crosstalk signal is not affected by the load capacitance. And it is interfering with the edge of the driver input to the probe circuitry slowing its response. But the interference is not equal on the rising and falling edges.

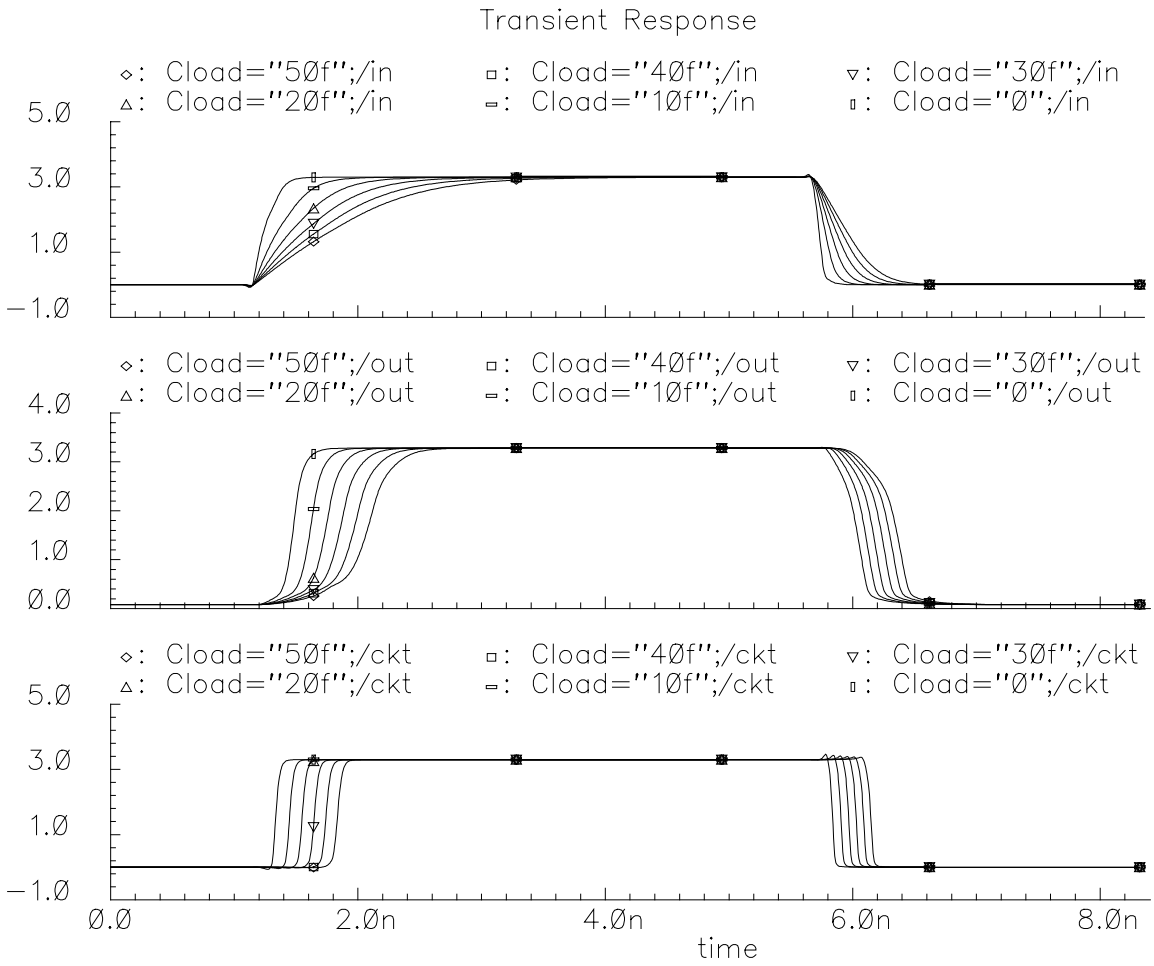

Figure 4.17 Circuit under varying load capacitance

Table 4.7 Edge difference for varying load capacitance

Load Cap (fF)	Rising edge difference (ps)	Falling edge difference (ps)
0	141	194
10	168	191
20	190	193
30	209	198
40	227	203
50	244	204

To study these effects I ran simulations with the crosstalk signal present, as just described, and with it absent. I also used a driver gate with equally sized n and p transistors and one where the p-channel device was increased in size to give equal rise and fall times. Plots of the signals at the comp net for the four cases is shown in Figure . The plots on the left

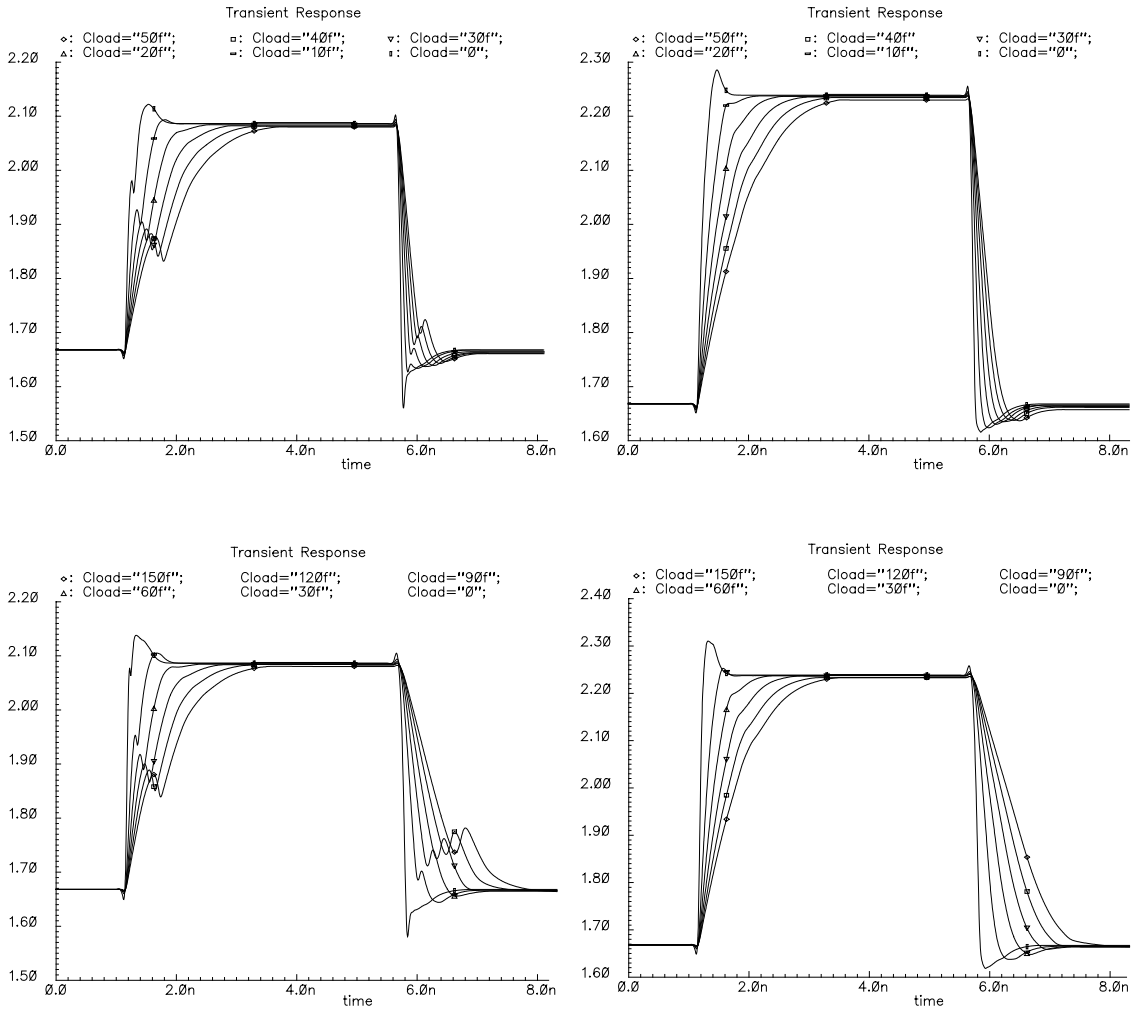

Figure 4.18 comp net signal under varying load.

side of the figure include crosstalk, while those on the right do not. On the top the driver had equal size transistors, and on the bottom equal rise and fall times (larger p-channel devices). The simulations with the larger p-channel transistors employed a larger load to give approxi-

mately the same pull up response as the smaller driver. Of course, this makes a noticeable change in the *falling* edge response. Its slope shifts closer to that shown for the rising edge.

The plots in Figure 4.18 show the crosstalk signal does indeed have a pronounced effect in the probe input as the driver load increases. With a small load the fast edge of the driving signal overwhelms any charge from the crosstalk path. The effect of the crosstalk is barely noticeable. As the driver edge slows down the steady crosstalk signal becomes more pronounced, and the timing of its effect shifts to earlier in the edge transition. This not only slows down the rise time and fall time of the edge into the probe, it also changes the slope in the critical transition region. This, in turn, changes the probe output timing.

To quantify the timing change I took 50 percent point measurements for each of the four cases at the circuit and probe outputs and computed the time differences of the edges. These differences are summarized in the graphs shown in Figure 4.19 and Figure 4.20. In these figures “MinP” is the case with equal sized transistors, and “Equal” is the case with larger p-channel devices for equal rise and fall times. The extra load is contributed by Cload. Since the n-channel devices do not change size, only the two curves for equal rise and fall times are given because they used a larger loading capacitance range. The minimum p-channel case is just a subset of these.

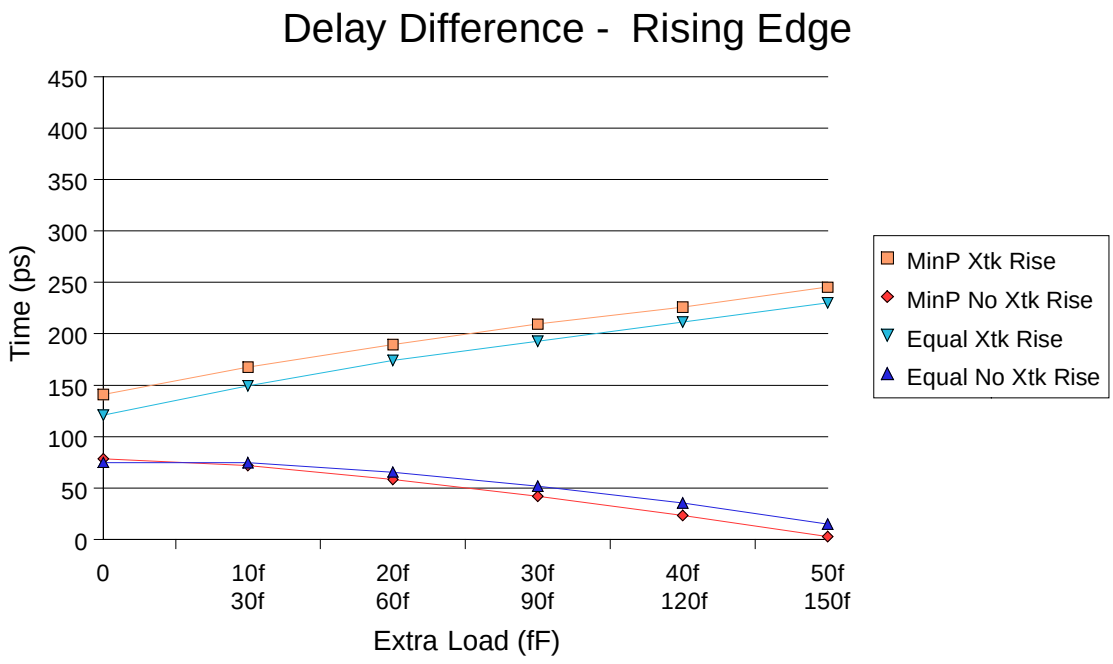

Figure 4.19 Rising edge circuit vs probe edge time difference

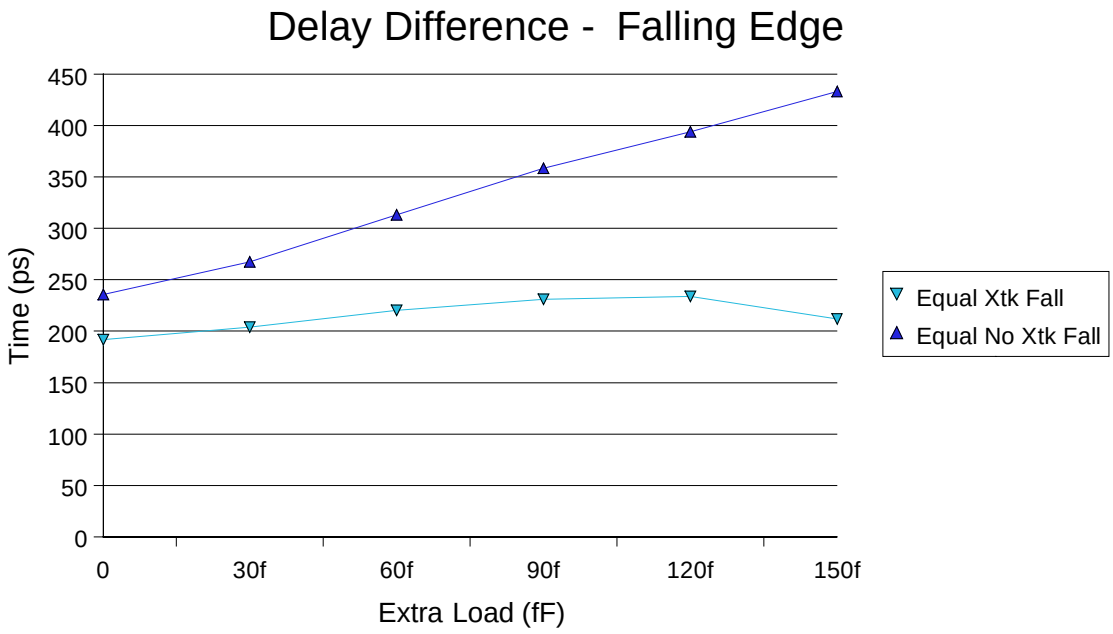

Figure 4.20 Falling edge circuit vs probe edge time difference

Ideally the difference would remain constant regardless of the load and the graphs would show horizontal lines. Obviously, the loading introduces some error in the relative timing of probe. Interestingly, for the rising edge the timing difference actually decreases when there is no crosstalk signal as shown by the down sloping curves. This is not necessarily desired. A flat line with a constant offset would be better. The absolute value of the change from the zero additional load condition is the important parameter.

Figure 4.21 and Figure 4.22 show the same data in a presentation that makes this clear. These figures plot the absolute value of the delay difference change. Here zero represents no change from the zero extra load case. Thus, zero is desired.

The worst case appears in the falling edge transition with no crosstalk. This is the top curve in Figure 4.20 and Figure 4.22. It shows a 200ps difference from zero to maximum load. The next worse cases indicate just over 100ps difference. These are rising edges with crosstalk.

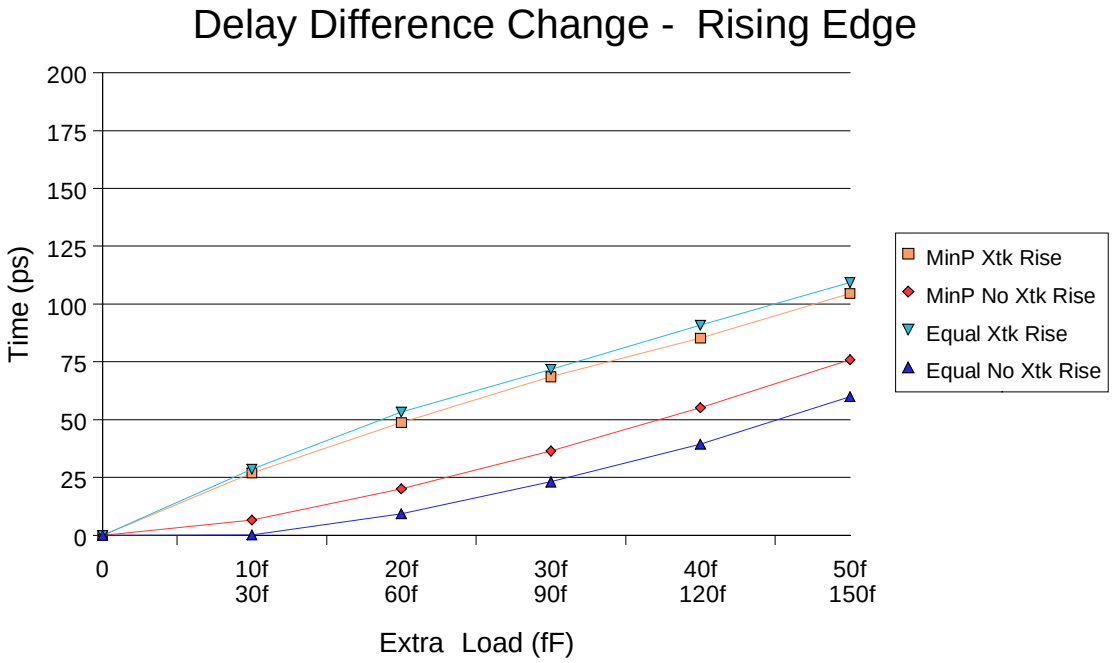

Figure 4.21 Normalized rising edge circuit vs probe edge time difference

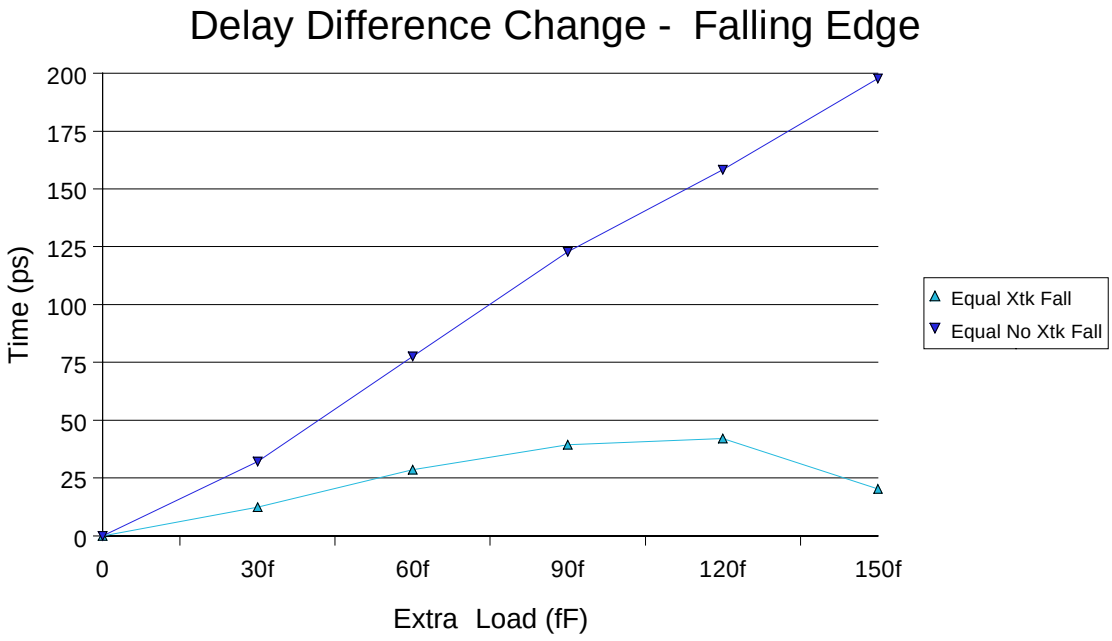

Figure 4.22 Normalized falling edge circuit vs probe edge time difference

Chapter 5 Conclusions and Future Work

5.1 Conclusions

Many aspects of the circuit performance were examined in the previous chapter. Most show the probe chip can recover the desired signal. It works in the nominal case, and at the process corners. It works when the coupling and parasitic capacitances are varied. It works at speeds that are faster than any clock period use in this process, yet it can be toggled as slow as approximately 800 ns.

These are all positive indicators the circuit is a good probe. There are a couple of parameters worth a closer look.

5.1.1 Circuit loading

The main purpose of the proposed method is for debugging signal timing. Therefore, it is important to minimize the loading added for probing. On the other hand, there must be enough signal captured to be recreated in the probe. These competing interests dictate the coupling capacitance be large enough to recover the signal, but no larger.

The nominal value of 1.2 fF used here gives around 500 mv at the probe input, reasonable edge rates from the probe, and a useful decay time of around 800 ns. Dropping the coupling capacitance down to 1 fF lowers the probe input to the neighborhood of 400 mv. The signal is recoverable, but the input is becoming tenuous. In a real chip, with some variation in the exact amount of coupling provided, and noise, it is likely that values below 1 fF will be too small to be useful.

Two factors mitigate the added capacitance. First, the probe is operated with a fill to increase the dielectric constant and promote coupling to the probe chip. In normal operation the production chip will have an air gap resulting in much less capacitance than when being probed. The operation when probed, therefore, will be pessimistic. If the chip meets timing while probed it will exceed it under normal operation.

Second, the signals of interest, ones where the timing margin is tight, will be longer lines with more capacitance. Small local interconnect using the lowest metal layers near the substrate do not usually cause timing problems. Those lines are not probed anyway. The lines of interest, for example busses, will be actuated by large drivers. Because the line capacitance is

already large, and the drivers sized for the load, the added capacitance of the probe constitutes a small percentage of the overall load. Thus, the probe will have minimal impact.

5.1.2 Timing accuracy

When probing a test point you want the probe output to reflect the timing of the net accurately, regardless of the conditions on the net. However, the capacitive coupling changes the mechanism for switching in the probe relative to on-chip circuitry, so the timing sensing between the two differs. The capacitor used in the probe makes it sensitive to the edge rate of the circuit being probed, whereas the on-chip circuitry switches strictly on voltage level. In addition, the parasitic capacitor in the probe circuit adds complexity to the response.

In the simulations the edge rate was varied over a large range by varying the load capacitance on the net. For the fabrication process used in the simulations, practical parts topped out at around 200 MHz, a 5 ns period. The edge rates (rise/fall times) in the simulations ran from 0.09 ns to 1.4 ns (see Appendix 1). The longest time takes 28 percent of the clock period for a 200 MHz clock. This is depicted graphically in Figure 4.17, which shows the output using a 10 ns period (100 MHz). When the edges of the probe output and on-chip circuit output are compared, there is an absolute timing difference of no more than 200 ps over the range of edge slopes, as indicated in Figure 4.21 and Figure 4.22. That represents an error of 4 percent of the 5 ns clock period.

From the data in Figure 4.21 and Figure 4.22, one must assume the probed timing is off by 100 ps on a rising edge and 200 ps on a falling edge. That should be acceptable for this technology. In fabrication processes with smaller feature sizes the worst edges will be sharper, assuming they stay around 25 percent of the clock period. Under these conditions the probe will have less error. Of course, there is less margin too, so the percentage error remains about the same because the error curves are nearly linear.

One way to reduce the timing error, or at least fix it to have smaller variability, is to put a buffer between the net of interest and the probe pad on the chip under test. In essence, the probe chip would be viewing the “ckt” net in the test fixture instead of the “in” net. This reduces the load going to the net seen by the probe to the capacitance of the probe pad plus the connection from the buffer to the pad. The cost is running a buffer while the chip is in production, but this should be a small cost.

5.1.3 Test setup

The nature of the probing technique imposes limits on what can be seen at test time. One significant limit is the testing must be planned during design, because the probe points must be installed and matched on both the production chip and probe chip. This does not allow for last minute unexpected signal probing requirements while at the test bench.

The maximum size of the production chip is also limited. Because the production chip and the probe chip physically join together, covering the production chip I/O array, the power and signals from the outside world must come through the probe chip. Therefore, the probe chip must be larger than the production chip to allow for a pad ring. This limits the size of chip that can be tested to the maximum die size minus the pad ring.

Furthermore, because the whole test assembly communicates via the pad ring, the number of outside connections into the test assembly is fewer than the number of I/O on the production chip. A chip with an area array has a significantly greater number of I/O than one with perimeter connection, one of the reasons for using flip chip in the first place. Therefore, the number of signals that can be controlled or observed at one time is smaller than the total number of signals.

To offset the limited number of pads the probe chip can incorporate multiplexers, drivers, or other circuitry to drive production chip inputs and select production chip outputs for observation. Also, chip scan circuitry or BIST can be used to set the state in the production chip. Still, the tests have to be carefully planned and focused on exercising and monitoring a small set of critical nets.

Process changes should not impose additional problems. The size of the smallest features continues to shrink, and the number of metal layers to grow. However, the sizing requirements of the top most metal layer remain about the same. Therefore, adding probe areas to the top metal layer is similar across processes.

5.1.4 Summary

Overall, the testing methodology proposed can be useful, but there are limitations where it can be used. The chip must be small enough to allow a pad ring. Any nets of interest must be decided in advance, presumably from design timing simulation results, and the layout altered

to include probe points. The probe must also be planned to power the requisite circuits, to pass signals to exercise the nets of interest, and to recover the desired signals.

This effort may be worthwhile during the first run of high value chips where the margins are tight. The yield and speed of chips placed on an MCM make a huge impact on the module yield and speed. Solving timing issues early in a production run can save cost from rework, and revenue lost from selling slower modules. These savings can justify the cost and effort of using this probing methodology.

5.2 Future work

To continue the development of the concept, the probe chip can be implemented and tested in steps. The first pass is to build a single chip with capacitive coupling between metal layers on the chip. The driving signal and receiver would be on the same chip. A driver would excite a metal layer which is coupled to a different metal layer, to which the receiver circuit is connected. Using values of metal to metal parameters for the process, the size of the coupling metal sections can be set to provide the same capacitance as a chip to chip connection. For this testing, there can be many receivers, each with a different geometry, to explore the capabilities with varying coupling, parasitic, and crosstalk capacitances.

With satisfactory results from a single chip experiment, a two chip arrangement can be made. A two chip setup duplicates the conditions of an actual probe as much as possible. Some accommodation to the fabrication process may be required for the experiment. For instance, depending on who is fabricating the chips and connecting them, it may not be possible to planarize the overglass. A chip fabrication broker, like MOSIS, will probably not be able to have the overglass planarized because it is a deviation from the normal chip fabrication process. In such a case some adjustment to the probe pad size may be necessary to achieve the desired coupling. After successfully probing with this experiment, actual production chips can be designed to use the probing technique.

Chapter 6 References

6.1 Books

6.1.1 MCM Design

[1] D. A. Doane and Paul D. Franzon, eds. *Multichip Module Technologies and Alternatives: The Basics*. New York, NY: Van Nostrand Reinhold, 1993.

[2] James J. Licari. *Multichip Module Design, Fabrication, and Testing*. New York, NY: McGraw-Hill, 1995.

[3] John H. Lau, ed. *Flip Chip Technologies*. New York, NY: McGraw-Hill, 1996.

6.1.2 Computer Architecture

[4] Gerry Kane. *MIPS RISC Architecture*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[5] James M. Feldman and Charles T. Retter. *Computer Architecture, A Designer's Text Based on a Generic RISC*. New York, NY: McGraw-Hill, 1994.

[6] J. L. Hennesy and D. A. Patterson. *Computer Architecture, A Quantitative Approach*. Palo Alto, CA: Morgan Kauffman, 1990.

[7] M. Johnson. *Superscalar Microprocessor Design*. Englewood Cliffs, NJ: Prentice Hall, 1991.

[8] *PowerPC 601 RISC Microprocessor User's Manual*. Somers, NY: International Business Machines, 1993.

[9] Miron Abramovici, Melvin A. Breuer, and Arthur D. Freidman. *Digital Systems Testing and Testable Design*. New York, NY: Computer Science Press, 1990.

[10] Don Anderson and Tom Shanley. *Pentium™ Processor System Architecture*. Reading, MA: Addison-Wesley, 1995.

[11] Steven A. Przybylski. *Cache and Memory Hierarchy Design: A Performance Directed Approach*. San Mateo, CA: Morgan Kaufmann, 1990.

6.1.3 Miscellaneous

[12] Carver Mead. *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[13] Satoshi Sakurai and Mohammed Ishmail. *Low-Voltage CMOS Operational Amplifiers: Theory, Design and Implementation*. Boston, MA: Kluwer Academic, 1995.

[14] Stephen F. Scheiber. *Building a Successful Board-Test Strategy.* Newton, MA: Butterworth-Heinemann, 1995.

[15] Magdy Abadir and Tony Ambler, eds. *Economics of Electronic Design, Manufacture and Test.* Boston, MA: Kluwer Academic, 1994.

6.2 Unpublished Papers

[16] Steven Wallace, Nirav Dagli, and Nader Bagherzadeh. “Design and Implementation of a 100MHz Reorder Buffer.” Department of Electrical and Computer Engineering, University of California, Irvine, 1995.

[17] Lipeng Cao and J. Peter Krusius. “Concurrent Packaging Architecture Design.” Cornell University, 1994. (Abstract published in *Proceedings of the 44th Electronic Components and Technology Conference*, pp. 75-80, 1994.)

[18] A. H. Dansky, H. H. Smith, and P. M. Williams. “Analysis and Results of Net Coupling within a High Performance Microprocessor.” IBM.

6.3 Published Papers

6.3.1 System Testing Case Studies

[19] Don D. Josephson, Daniel J. Dixon, and Barry J. Arnold. “Test Features of the HP PA7100LC Processor.” *Proceeings of the International Test Conference*, pp. 764-772, 1993.

[20] Marc E. Levitt, et. al. “Testability, Debuggability, and Manufacturability Features of the UltraSPARC™-I Microprocessor.” *Proceeings of the International Test Conference*, pp. 157–166, 1995.

[21] Hong Hao and Rick Avra. “Structured Design-for-Debug—the SuperSPARC™ II Methodology and Implementation.” *Proceeings of the International Test Conference*, pp. 175–183, 1995.

[22] Rajiv Patel and Krishna Yaragadda. “Testability Features if the SuperSPARC Microprocessor.” *Proceeings of the International Test Conference*, pp. 773–781, 1993.

[23] Kalon Holdbrook, Sunil Joshi, Samir Mitra, Joe Petolino, Renu Raman, and Michelle Wong. “microSparc™: A Case-Stydy of Scan Based Debug.” *Proceeings of the International Test Conference*, pp. 70–75, 1994.

[24] Dilip K. Bhavsar and John H. Edmondson. “Testability Strategy of the Alpha AXP 21164 Microprocessor.” *Proceeings of the International Test Conference*, pp. 50–59, 1994.

[25] Craig Hunter, E. Kofi Vida-Torku, and Johnny LeBlanc. “Balancing Structured and Ad-hoc Design for Test: Testing of the PowerPC 603™ microprocessor.” *Proceeings of the International Test Conference,* pp. 76–83, 1994.

[26] Jen-Tien Yen, Marie Sullivan, Carlos Montemayor, Pete Wilson, and Richard Evers. “Overview of PowerPC™ 620 Multiprocessor Verification Strategy.” *Proceeings of the International Test Conference,* pp. 167–174, 1995.

[27] Alfred L. Crouch, Matthew Pressly, and Joe Circello. “Testability Features of the MC68060 Microprocessor.” *Proceeings of the International Test Conference,* pp. 60–69, 1994.

[28] Wayne Needham and Naga Gollakota. “DFT Strategy for Intel Microprocessors.” *Proceeings of the International Test Conference,* pp. 396–399, 1996.

[29] H. Bonnenberg, A. Curiger, N. Felber, H. Kaeslin, R. Zimmermann, and W. Fichter. “VINCI: Secure Test of a VLSI High-Speed Encryption System.” *Proceeings of the International Test Conference,* pp. 782-790, 1993.

[30] Russell J. Wagner and Joel A. Jorgenson. “Design-For-Test Techniques Utilized in an Avionics Computer MCM.” *Proceeings of the International Test Conference,* pp. 373-382, 1993.

6.3.2 Chip Testing—Known Good Die

[31] http://www.research.ibm.com/topics/popups/serious/chip/html/pica.html

[32] D. C. Keezer. “Known Good Die for MCMs: Enabling Technologies.” *Proceeings of the International Test Conference,* pp. 266, 1993.

[33] Lynn Roszel. “MCM Foundry Test Methodology and Implementation.” *Proceedings of the International Test Conference,* pp. 369-372, 1993.

[34] Stephen Martin, Dudly Gage, Tom Powell, and Buf Slay. “A Practical Approach to Producing Known-Good Die.” *Proceedings of the International Conference and Exhibition on Multichip Modules*, pp. 139-151, 1993.

[35] Charles E. Radke, Lo-Soun Su, and Yee-Ming Ting. “Known Good Die and its EvolutionBipolar and CMOS.” *Proceedings of the International Conference and Exhibition on Multichip Modules*, pp. 152-159, 1993.

[36] Marlene J. Begay, James Van Zee, Greg Westbrook, and Bill Williams. “Getting to Know your MCM Die.” *Proceedings of the International Conference and Exhibition on Multichip Modules*, pp. 160-165, 1993.

[37] Anne E. Gattiker and Wojciech Maly. “Feasibility Study of Smart Substrate Multichip Modules.” *Proceeings of the International Test Conference,* pp. 41–49, 1994.

[38] Anne E. Gattiker, Wojciech Maly, and Michael E. Thomas. “Are There Any Alternatives to 'Known Good Die' “. *Proceedings of the IEEE Multi-Chip Module Conference*, pp. 102-107, 1994.

[39] Kaiyuaun Huang, Vinod K. Agarwal, and Laurence LaForge. “Wafer Testing with Pairwise Comparisons.” *22nd International Symposium on Fault-Tolerant Computing*, pp. 374-383, July 1992.

[40] W. Burdick and W. Daum. “High Yield Multichip Modules Based On Minimal IC Pretest.” *Proceeings of the International Test Conference*, pp. 30–40, 1994.

6.3.3 Probes for Bare Chip Test

[41] January Kister and Robert L. Franch. “Advances in Membrane Probe Technology.” *Proceeings of the International Test Conference*, pp. 927-935, 1992.

[42] Brian Leslie and Farid Matta. “Membrane Probe Card Technology (The Future for High Performance Wafer Test).” *Proceeings of the International Test Conference*, pp. 659-665, 1988.

[43] Toshiaki Ueno and You Kondoh. “Membrane Probe Technology for MCM Known-Good-Die.” *Proceeings of the International Test Conference*, pp. 22–29, 1994.

[44] Alan Barber, Ken Lee, and Hanns Obermaier. “A Bare-chip Probe for High I/O, High Speed Testing.” *Hewlett-Packard Technical Report, HPL-94-18,* March, 1994.

[45] Toshiki Hirano, Atsuo Kimura, and Shin'ichiro Mori. “Silicon Microprobing Array for Testing and Burn-In.” *Proceedings of the IEEE Multi-Chip Module Conference*, pp. 89-93, 1994.

[46] Randal Roebuck, Fariborz Agahdel, and Dr. Chung Ho. “Known Good Die: A Practical Solution.” *Proceedings of the International Conference and Exhibition on Multichip Modules*, pp. 177-182, 1993.

[47] Lina Prokopchak. “Development of a Solution for Achieving Known-Good-Die.” *Proceeings of the International Test Conference*, pp. 15–21, 1994.

[48] Thomas P. Budka and Gabriel M. Rebeiz. “A Microwave Circuit Electric Field Imager.” *Proceedings of the IEEE International Microwave Symposium*, pp. 1139-1142, 1995.

[49] Helen Congleton and Randolph E. Root. “TRB™ AssemblyA Soft-Tooled Approach to Device Pre-Test and Interconnection.” *Proceedings of the International Conference and Exhibition on Multichip Modules*, pp. 183-188, 1993.

6.3.4 Capacitive Coupling

[50] David Salzman and Thomas Knight, Jr. “Capacitive Coupling Solves the Known Good Die Problem.” *Proceedings of the IEEE Multi-Chip Module Conference*, pp. 95-100, 1994.

[51] Thaddeus J. Gabara and Wilhelm C. Fischer. “Capacitive Coupling and Quantized Feedback Applied to Conventional CMOS Technology.” *IEEE Journal of Solid-State Circuits*, pp. 419-427, March 1997.

6.3.5 Test Cost Analysis

[52] Rick Gayle. “The Cost of Quality: Reducing ASIC Defects with IDDQ, At-Speed Testing, and Increased Fault Coverage.” *Proceeings of the International Test Conference*, pp. 285-292, 1993.

[53] Wojciech Maly. “Cost of Silicon Viewed from VLSI Design Perspective.” *Proceedings of the 31st ACM/IEEE Design Automation Conference*, pp. 135-142, 1994.

[54] Linda Chao, Daren Dance, and Tom DiFloria. “Get a handle on Your Cost of Test.” *Test & Measurement World*, pp. 45–50, April 1995.

6.3.6 Scan Testing

[55] Mick M. V. Tegethoff and Kenneth P. Parker. “IEEE Std 1149.1: WhereAre We? Where From Here?” *IEEE Design and Test of Computers*, Vol.12, No. 2, pp. 53-59, Summer 1995.

[56] Kwang-Ting Cheng. “Single-Clock Partial Scan.” *IEEE Design and Test of Computers*, Vol. 12, No. 2, pp. 24-30, Summer 1995.

[57] Tom Storey. “A Test Methodology for VLSI Chips on Silicon.” *Proceeings of the International Test Conference*, pp. 359-368, 1993.

[58] Marcelo Lubaszewski, Meryem Marzouki, and Mohamed Hedi Touani. “A Pragmatic Test and Diagnosis Methodology for Partially Testable MCMs.” *Proceedings of the IEEE Multi-Chip Module Conference*, pp. 108-113, 1994.

6.3.7 IDDQ Testing

[59] M. Ruallán, C. Ferrer, J. Oliver, D. Mateo, and A. Rubio. “Analysis of IDDQ Testing Implementation and Circuit Partitioning for a CMOS Cell-Based Design.” *Journal of Microelectronic Systems Integration*, Vol. 3, No. 3, pp. 173–187, 1995.

[60] Jerry M. Soden, Charles F. Hawkins, Ravi K. Gulati, and Weiwei Mao. “IDDQ Testing: A Review.” *Journal of Electronic Testing: Theory and Applications*, Vol. 3, pp 291-303, 1992.

[61] Doug Josephson, Mark Storey, and Dan Dixon. “Microprocessor IDDQ Testing: A Case Study.” *IEEE Design and Test of Computers*, Vol. 12, No. 2, pp. 42-52, Summer 1995.

[62] Wojciech Maly and Marek Patyra. “Built-in Current Testing.” *IEEE Journal of Solid State Circuits,* Vol. 27, No. 3, pp. 425-428, March 1992.

[63] Keith Baker. “QTAG: A Standard for Test Fixture based IDDQ/ISSQ Monitors.” *Proceedings of the International Test Conference,* pp. 194-202, 1994.

[64] Jien-Chung Lo, James C. Daly, and M. Nicolaidis. “Design of Static CMOS Self-Checking Circuits using Built-In Current Sensing.” *22nd International Symposium on Fault-Tolerant Computing,* pp. 104-111, July 1992.

[65] Paul C. Wiscombe. “A Comparison of Stuck-At Fault Coverage and IDDQ Testing on Defect Levels.” *Proceeings of the International Test Conference*, pp. 293-299, 1993.

[66] J. S. Beasley, H. Ramamurthy, J. RamÍrez-Angulo, and M. DeYong. “IDD Pulse Response Testing of Analog and Digital CMOS Circuits.” *Proceeings of the International Test Conference*, pp. 626-634, 1993.

[67] Ching-Wen Hsue and Chih-Jen Lin. “Built-In Current Sensor for IDDQ Test in CMOS.” *Proceeings of the International Test Conference*, pp. 635-641, 1993.

[68] Changku Hwang, Mohammed Ishmail, and Joanne E. DeGroat. “On-Chip IDDQ Testability Schemes for Detecting Multiple Faults in CMOS IC's.” *IEEE Journal of Solid State Circuits,* Vol. 31, pp. 732–739, 1996.

[69] Kenneth M. Wallquist, Alan W. Righter, and Charles F. Hawkins. “A General Purpose IDDQ Measurement Circuit.” *Proceeings of the International Test Conference*, pp. 642-651, 1993.

[70] Robert C. Aitken. “A Comparison of Defect Models for Fault Location with IDDQ Measurements.” *Proceeings of the International Test Conference*, pp. 1051-1060, 1993.

6.3.8 Delay Testing

[71] M. Favalli and C. Metra. “Sensing Circuit for On-Line Detection of Delay Faults.” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* Vol. 4, N0. 1, pp. 130–133, 1996.

6.3.9 Pin Electronics

[72] Arnold Frisch and Thomas Almy. “Timing Analyzer for Embedded Testing.” *Proceeings of the International Test Conference,* pp. 552-555, 1993.

[73] Will Creek. “Characterization of Edge Placement Accuracy in High-Speed Digital Pin Electronics.” *Proceeings of the International Test Conference,* pp. 556-565, 1993.

[74] Kenneth D. Wagner and Bernd K. Koenemann. “Testable Programmable Digital Clock Pulse Control Elements.” *Proceeings of the International Test Conference,* pp. 902-909, 1993.

[75] James A. Gasbarro and Mark A. Horowitz. “A Single-Chip Functional Tester for VLSI Circuits.” *Proceeings of the IEEE International Solid State Circuits Conference,* pp. 84-85, 1990.

[76] Tai-ichi Otsuji. “A Picosecond-Accuracy, 700-MHz Range, Si-Bipolar Time Interval Counter LSI.” *IEEE Journal of Solid State Circuits,* Vol. 28, No. 9, pp. 941-947, September 1993.

[77] Stewart S. Taylor. “A High-Performance GaAs Pin Electronics Circuit for Automatic Test Equipment.” *IEEE Journal of Solid State Circuits,* Vol. 28, No. 10, pp. 1023-1029, October 1993.

6.3.10 Test Synthesis

[78] Robert C. Aitken. “An Overview of Test Synthesis Tools.” *IEEE Design and Test of Computers,* Vol. 12, No. 2, pp. 8-15, Summer 1995.

[79] Thomas E. Marchok, Aiman El-Maleh, Janusz Rajski, and Wojciech Maly. “Testability Implications of Performance-Driven Logic Synthesis.” *IEEE Design and Test of Computers,* Vol. 12, No. 2, pp. 32-39, Summer 1995.

[80] Henry Cox. “Synthesizing Circuits with Implicit Testability Constraints.” *IEEE Design and Test of Computers,* Vol. 12, No. 2, pp. 16-23, Summer 1995.

6.3.11 Computer Architecture

[81] Maurice P. Marks. “Future Directions in Microprocessor Technology.” *IEEE Journal of Solid-State Circuits,* Vol. 30, No. 4, pp. 371-374, April 1995.

[82] Masato Motomura, Toshaki Inoue, Hachiro Yamada, and Akihiko Konagaya. “Cache-Processor Coupling: A Fast and Wide On-Chip Data Cache Design.” *IEEE Journal of Solid-State Circuits,* Vol. 30, No. 4, pp. 375-381, April 1995.

[83] John H. Edmondson et al. “Internal Organization of the Alpha 21164, a 300-MHz 64-bit Quad-issue CMOS RISC Microprocessor.” *Digital Technical Journal,* Vol. 7, No. 1, pp. 119-133, 1995.

[84] Daniel W. Dobberpuhl et al. “A 200-MHz 64-b Dual-Issue CMOS Microprocessor.” *IEEE Journal of Solid State Circuuits,* Vol. 27, No. 11, pp. 1555-1567, Nov. 1992.

[85] Mayan Moudgill, Keshav Pingali, and Stamatis Vassiliadis. “Register Renaming and Dynamic Speculation: an Alternative Approach.” *IEEE Intenational Symposium of Micro Architecture,* pp. 202-213, Dec. 1993.

[86] Thomas M. Conte, Kishore N. Menzes, Patrick M. Mills, ans Burzin A. Patel. “Optimization of Instruction Fetch Mechanisms for High Issue Rates.” *Proceedings of the 22nd Annual International Symposium on Compter Architecture,* June 1995.

6.3.12 Miscellaneous

[87] Patrick Thompson. “Chip-scale Packaging.” *IEEE Spectrum*, pp. 36–43, Vol. 34, No. 8, August 1997.

[88] Don Cotterell. “Delay Calculation Standard and Deep Submicron Design.” *Integrated System Design,* pp. 44–52, July 1996.

[89] Masanori Izumikawa, et. al. “A 0.9V 100MHz 4mW 2mm^2 16b DSP Core.” *Proceedings of the IEEE International Solid State Circuits Conference,* pp. 84–85, 1995.

[90] “A D&T Roundtable: Deep-Submicron Test.” *IEEE Design and Test of Computers,* Vol. 13, No. 3, pp. 102–108, Fall 1996.

[91] J. Mulder, M. van de Gevel, and A. H. M. van Roermund. “A Reduced-Area Low-Power Low-Voltage Single-Ended Differential Pair.” *IEEE Journal of Solid State Circuits,* Vol. 32, No. 2, pp. 254–257, Feb. 1997.

[92] A. Appel, S. Crank, Y. Kim, et. al. “A Manufacturable 0.30mm Gate CMOS Technology for High Speed Microprocessors.” *Proceedings of the Symposium on VLSI Technology,* pp. 220–221, 1996.

[93] Mariusz Niewczas. “Characterisation of the Threshold Voltage Variation: a Test Chip and the Results.” *Proceedings of the International Conference on Microelectronic Test Structures,* pp. 169–172, 1997.

6.3.13 Chemical-Mechanical Planarization (CMP)

[94] Bob Roberts. “Chemical-Mechanical Planarization.” *Proceedings of the Advanced Semiconductor Manufacturing Conference,* pp. 206–210, 1992.

[95] Toshiroh Karaki-Doy, et.al. “Global Planarization Technique/CMP by High Precision Polishing and Its Characteristics.” *Proceedings of the International Symposium on Semiconductor Manufacturing,* pp. 214–217, 1995.

6.3.14 NCSU Papers

[96] Paul D. Franzon, Andrew Stanaski, Yusuf Tekmen, and Sanjeev Banerjia. “System Design Optimization for MCM-D/Flip-Chip.” *IEEE Transactions on Components, Packaging, and Manufacturing Technology—Part B*, pp. 620–627, Vol. 18, No. 4, Nov. 1995.

[97] Paul D. Franzon, Tom Conte, Sanjeev Banerjia, Alan Glaser, Steve Lipa, Toby Schaffer, Andrew Stanaski and Yusuf Tekmen. “Computer Design Strategy for MCM-D/Flip-Chip Technology.” *Proceedings of the Topical Meeting on Electrical Performance of Electronic Packaging*, 1997.

[98] Sanjeev Banerjia, Alan Glaser, Christoforos Harvatis, Steve Lipa, Real Pomerleau, Toby Schaffer, Andrew Stanaski, Yusuf Tekmen, Griff Bilbro, and Paul Franzon. “Issues in partitioning integrated circuits for mcm-d flip-chip technology.” *IEEE MultiChipModuleConference*, pp. 154–160, 1996.

[99] Paul Franzon, Andrew Stanaski, Yusuf Tekmen, Sanjeev Banerjia. “System Design Optimization for MCM.” *IEEE MultiChipModuleConference*, 1995.

[100] Andrew Stanaski, Optimizing Memory Design for Packagability, MS Thesis.

[101] Toby Schaffer, Andy Stanaski, Alan Glaser, Paul Franzon. “The NCSU Design Kit for IC Fabrication through MOSIS.” *Cadence User Conference*, 1998.

APPENDICIES

Appendix 1 Additional Data

This appendix contains tables of data generated from the simulations where the load and crosstalk were varied, as described in Section 4.4.2.8. The tables provide precise time points when the output curves crossed given voltage values. One of the curve sets is shown in Figure 4.17. The Y value in the tables is the voltage line. The columns corresponding to curves for various loadings. All times are nanoseconds.

The clock period used in the simulations was 10 ns (100 MHz). The parts of interest are the edges. From subtracting various times in the tables you can calculate the 10-90 percent rise/ fall times, the 20-80 percent rise/fall times, and the relative timing of the midpoint crossing.

A1.1 Net loading with min sized P transistors, crosstalk

Table A1.1 'In' node, min P, crosstalk

Y value (volts)	Load = 50 fF	Load = 40 fF	Load = 30 fF	Load = 20 fF	Load = 10 fF	Load = 0 fF
2.97	2.66455	2.40957	2.15485	1.90037	1.64602	1.39379
2.97	5.74530	5.73557	5.72522	5.71392	5.70090	5.68331
2.64	2.32458	2.12874	1.93315	1.73759	1.54205	1.34565
2.64	5.79736	5.77990	5.76198	5.74324	5.72267	5.69701
1.65	1.78359	1.67682	1.56950	1.46150	1.35253	1.23992
1.65	5.95004	5.90726	5.86431	5.82106	5.77672	5.72817
660m	1.38789	1.34752	1.30700	1.26615	1.22458	1.18087
660m	6.13602	6.06169	5.98690	5.91188	5.83645	5.75795
330m	1.27272	1.25258	1.23227	1.21164	1.19025	1.16633
330m	6.24258	6.15185	6.06029	5.96881	5.87419	5.77418

Table A1.2 'Out' node, min P, crosstalk

Y value (volts)	Load = 50 fF	Load = 40 fF	Load = 30 fF	Load = 20 fF	Load = 10 fF	Load = 0 fF
2.97	2.31756	2.17691	2.03686	1.89165	1.74077	1.58004
2.97	6.09573	6.05628	6.01463	5.97078	5.92103	5.86161
2.64	2.21131	2.08355	1.95653	1.82457	1.68646	1.53562
2.64	6.20836	6.15988	6.10618	6.05046	5.99127	5.92719
1.65	2.08193	1.96989	1.85850	1.74143	1.61758	1.47884
1.65	6.34873	6.28747	6.22287	6.15934	6.09600	6.03030
660m	1.91294	1.82873	1.74172	1.64581	1.54083	1.41936
660m	6.42404	6.35926	6.29167	6.22646	6.16288	6.09707
330m	1.70779	1.65277	1.60213	1.54788	1.47266	1.37345
330m	6.46257	6.39616	6.32730	6.26147	6.19818	6.13242

Table A1.3 'Ckt' node, min P, crosstalk

Y value (volts)	Load = 50 fF	Load = 40 fF	Load = 30 fF	Load = 20 fF	Load = 10 fF	Load = 0 fF
2.97	1.88031	1.78659	1.69088	1.59258	1.48945	1.37637
2.97	6.11062	6.05323	5.99551	5.93683	5.87679	5.81153
2.64	1.86336	1.76984	1.67452	1.57646	1.47376	1.36104
2.64	6.12425	6.06612	6.00729	5.94773	5.88663	5.82033
1.65	1.83662	1.74380	1.64914	1.55179	1.44981	1.33803
1.65	6.14565	6.08642	6.02661	5.96601	5.90363	5.83604
660m	1.81187	1.72023	1.62679	1.53092	1.43056	1.32049
660m	6.16094	6.10135	6.04105	5.97990	5.91718	5.84897
330m	1.79776	1.70718	1.61501	1.52032	1.42127	1.31287
330m	6.16890	6.10909	6.04871	5.98743	5.92429	5.85575

A1.2 Net loading with min sized P transistors, no crosstalk

Table A1.4 'In' node, min P, no crosstalk

Y value (volts)	Load = 50 fF	Load = 40 fF	Load = 30 fF	Load = 20 fF	Load = 10 fF	Load = 0 fF
2.97	2.66320	2.40833	2.15380	1.89949	1.64470	1.39202
2.97	5.74527	5.73559	5.72523	5.71391	5.70090	5.68331
2.64	2.32352	2.12773	1.93208	1.73624	1.54052	1.34389
2.64	5.79732	5.77992	5.76198	5.74322	5.72267	5.69700
1.65	1.78296	1.67628	1.56917	1.46124	1.35229	1.23987
1.65	5.95001	5.90728	5.86433	5.82106	5.77674	5.72817
660m	1.38786	1.34749	1.30698	1.26613	1.22457	1.18088
660m	6.13576	6.06168	5.98700	5.91185	5.83643	5.75794
330m	1.27271	1.25257	1.23226	1.21163	1.19026	1.16633
330m	6.24173	6.15096	6.05940	5.96809	5.87421	5.77418

Table A1.5 'Out' node, min P, no crosstalk

Y value (volts)	Load = 50 fF	Load = 40 fF	Load = 30 fF	Load = 20 fF	Load = 10 fF	Load = 0 fF
2.97	1.99551	1.90719	1.81444	1.71653	1.61146	1.49051
2.97	6.22713	6.16975	6.10690	6.04239	5.97498	5.90558
2.64	1.92518	1.84502	1.76074	1.67118	1.57412	1.46030
2.64	6.31312	6.25058	6.18208	6.11251	6.04099	5.97027
1.65	1.83877	1.76674	1.69083	1.60978	1.52132	1.41622
1.65	6.42640	6.36068	6.28835	6.21591	6.14238	6.07169
660m	1.74394	1.68039	1.61336	1.54179	1.46338	1.36931
660m	6.49456	6.42823	6.35485	6.28190	6.20805	6.13781
330m	1.66288	1.60785	1.54974	1.48724	1.41864	1.33547

Table A1.5 'Out' node, min P, no crosstalk

Y value (volts)	Load = 50 fF	Load = 40 fF	Load = 30 fF	Load = 20 fF	Load = 10 fF	Load = 0 fF
330m	6.52969	6.46331	6.38959	6.31652	6.24260	6.17276

Table A1.6 'Ckt' node, min P, no crosstalk

Y value (volts)	Load = 50 fF	Load = 40 fF	Load = 30 fF	Load = 20 fF	Load = 10 fF	Load = 0 fF
2.97	1.87996	1.78621	1.69051	1.59226	1.48918	1.37612
2.97	6.11054	6.05321	5.99548	5.93680	5.87678	5.81155
2.64	1.86299	1.76951	1.67421	1.57614	1.47350	1.36081
2.64	6.12417	6.06611	6.00727	5.94770	5.88663	5.82037
1.65	1.83627	1.74348	1.64884	1.55151	1.44957	1.33784
1.65	6.14558	6.08641	6.02659	5.96599	5.90364	5.83608
660m	1.81157	1.71994	1.62654	1.53071	1.43035	1.32034
660m	6.16085	6.10134	6.04104	5.97988	5.91719	5.84900
330m	1.79749	1.70694	1.61475	1.52008	1.42111	1.31277
330m	6.16883	6.10908	6.04870	5.98741	5.92430	5.85579

A1.3 Net loading with up sized P transistors, crosstalk

Table A1.7 'In' node, larger P, crosstalk

Y value (volts)	Load = 150 fF	Load = 120 fF	Load = 90 fF	Load = 60 fF	Load = 30 fF	Load = 0 fF
2.97	2.53766	2.28308	2.02877	1.77446	1.52078	1.27091
2.97	5.88851	5.86252	5.83571	5.80749	5.77543	5.72811
2.64	2.21747	2.02291	1.82851	1.63411	1.43959	1.23914
2.64	6.02523	5.97447	5.92335	5.87152	5.81706	5.74817
1.65	1.72356	1.61953	1.51533	1.41092	1.30590	1.19660
1.65	6.44164	6.31358	6.18530	6.05685	5.92756	5.78894
660m	1.37646	1.33693	1.29733	1.25759	1.21726	1.17094
660m	6.94204	6.72240	6.50178	6.28061	6.05709	5.82678
330m	1.27085	1.25122	1.23147	1.21141	1.19021	1.16173
330m	7.21614	6.94600	6.67601	6.40612	6.13351	5.84750

Table A1.8 'Out' node, larger P, crosstalk

Y value (volts)	Load = 150 fF	Load = 120 fF	Load = 90 fF	Load = 60 fF	Load = 30 fF	Load = 0 fF
2.97	2.23805	2.09971	1.95870	1.81625	1.66459	1.50346
2.97	6.54698	6.44032	6.32528	6.20576	6.08151	5.92395
2.64	2.13874	2.01318	1.88521	1.75552	1.61544	1.46085
2.64	6.69381	6.57900	6.46165	6.33305	6.17762	5.99028
1.65	2.01871	1.90839	1.79555	1.68010	1.55309	1.40689
1.65	7.00057	6.85214	6.67736	6.49211	6.29785	6.09370
660m	1.86906	1.78299	1.69196	1.59566	1.48604	1.35289
660m	7.14132	6.96712	6.77344	6.57343	6.36837	6.16051

Table A1.8 'Out' node, larger P, crosstalk

Y value (volts)	Load = 150 fF	Load = 120 fF	Load = 90 fF	Load = 60 fF	Load = 30 fF	Load = 0 fF
330m	1.68601	1.63271	1.58107	1.51781	1.43171	1.31597
330m	7.21228	7.02627	6.82298	6.61528	6.40505	6.19579

Table A1.9 'Ckt' node, larger P, crosstalk

Y value (volts)	Load = 150 fF	Load = 120 fF	Load = 90 fF	Load = 60 fF	Load = 30 fF	Load = 0 fF
2.97	1.83157	1.73872	1.64387	1.54584	1.44247	1.32367
2.97	6.73604	6.57070	6.40373	6.23468	6.06215	5.87706
2.64	1.81478	1.72226	1.62773	1.53006	1.42706	1.30863
2.64	6.75774	6.59027	6.42095	6.24938	6.07426	5.88607
1.65	1.78866	1.69675	1.60289	1.50598	1.40373	1.28607
1.65	6.78858	6.61841	6.44632	6.27191	6.09372	5.90206
660m	1.76494	1.67417	1.58167	1.48631	1.38570	1.26976
660m	6.80839	6.63697	6.46353	6.28791	6.10838	5.91498
330m	1.75179	1.66212	1.57086	1.47667	1.37758	1.26333
330m	6.81784	6.64570	6.47202	6.29570	6.11572	5.92216

A1.4 Net loading with up sized P transistors, no crosstalk

Table A1.10 'In' node, larger P, no crosstalk

Y value (volts)	Load = 150 fF	Load = 120 fF	Load = 90 fF	Load = 60 fF	Load = 30 fF	Load = 0 fF
2.97	2.53720	2.28269	2.02843	1.77417	1.52019	1.26970
2.97	5.88850	5.86251	5.83571	5.80747	5.77542	5.72810
2.64	2.21713	2.02257	1.82809	1.63362	1.43902	1.23934
2.64	6.02521	5.97446	5.92335	5.87151	5.81705	5.74816
1.65	1.72339	1.61942	1.51524	1.41085	1.30584	1.19661
1.65	6.44162	6.31357	6.18529	6.05684	5.92755	5.78894
660m	1.37645	1.33692	1.29732	1.25758	1.21726	1.17094
660m	6.94145	6.72178	6.50118	6.28019	6.05717	5.82676
330m	1.27085	1.25122	1.23147	1.21140	1.19021	1.16174
330m	7.21556	6.94539	6.67533	6.40534	6.13257	5.84758

Table A1.11 'Out' node, larger P, no crosstalk

Y value (volts)	Load = 150 fF	Load = 120 fF	Load = 90 fF	Load = 60 fF	Load = 30 fF	Load = 0 fF
2.97	1.94807	1.86085	1.76689	1.66781	1.55907	1.43028
2.97	6.89930	6.72413	6.55019	6.36664	6.17668	5.97107
2.64	1.88380	1.80451	1.71867	1.62738	1.52592	1.40188
2.64	7.05046	6.85701	6.66505	6.46221	6.25349	6.03611
1.65	1.80355	1.73217	1.65452	1.57143	1.47827	1.36089
1.65	7.22177	7.01215	6.80467	6.58503	6.36131	6.13766
660m	1.71511	1.65220	1.58366	1.51011	1.42712	1.31948
660m	7.31590	7.09900	6.88479	6.65793	6.42861	6.20376

Table A1.11 'Out' node, larger P, no crosstalk

Y value (volts)	Load = 150 fF	Load = 120 fF	Load = 90 fF	Load = 60 fF	Load = 30 fF	Load = 0 fF
330m	1.63999	1.58568	1.52620	1.46195	1.38924	1.29226
330m	7.36681	7.14514	6.92698	6.69586	6.46381	6.23869

Table A1.12 'Ckt' node, larger P, no crosstalk

Y value (volts)	Load = 150 fF	Load = 120 fF	Load = 90 fF	Load = 60 fF	Load = 30 fF	Load = 0 fF
2.97	1.83145	1.73862	1.64377	1.54574	1.44240	1.32364
2.97	6.73597	6.57062	6.40367	6.23461	6.06210	5.87707
2.64	1.81468	1.72215	1.62763	1.52998	1.42699	1.30861
2.64	6.75767	6.59018	6.42089	6.24931	6.07422	5.88609
1.65	1.78856	1.69665	1.60280	1.50590	1.40367	1.28605
1.65	6.78851	6.61833	6.44626	6.27186	6.09368	5.90209
660m	1.76484	1.67409	1.58160	1.48624	1.38566	1.26976
660m	6.80832	6.63688	6.46347	6.28785	6.10835	5.91500
330m	1.75171	1.66205	1.57080	1.47661	1.37754	1.26333
330m	6.81776	6.64560	6.47194	6.29563	6.11567	5.92219

Appendix 2 SAND - A Four Issue MIPS μp

A2.1 Purpose of the Project

The Electronics Research Lab is engaged in developing CAD tools and methodologies for the design of systems optimized for implementation on multichip modules with thin film interconnect (MCM-D). Areas of research include partitioning, placement, routing, performance and testing of systems on MCMs. In order to gauge the quality of the tools and methods being investigated we needed a realistic example of a system that might be built on MCM-D. Such a system would be large, require high performance, and be representative of mainstream systems. Systems with lesser qualities would never be considered for MCM implementation.

For this work we required a system with available detailed design data. Unfortunately such data was not available from commercial sources, and there were no university projects we considered adequate to meet our needs. Thus, we decided to create our own design.

We settled on the design of a superscalar RISC microprocessor since it has all the desired characteristics. We chose a peak issue rate of four instructions, and hardware scheduling and interlocking. There are many processors of this general class, all large, high performance machines, just the sort of system one might consider for an MCM.

A2.2 Design Overview

The Electronics Research Lab at North Carolina State University has developed a four issue superscalar microprocessor design using the MIPS R2000/R3000 instruction set. It is a general purpose processor that uses a reorder buffer to allow out of order issue and completion. The project was called SAND, the Superscalar Architecture Demonstrator for NCSU (North Carolina State University). The processor includes 8KB instruction and data caches, static branch target prediction, a peak issue rate of four instructions per cycle, and a reorder buffer for out-of-order issue and completion. For simplicity, the data path is limited to 32 bits, so the processor supports single precision only. Execution of instructions follows a seven stage path through the machine, with the execute stage of variable length depending on the pipeline depth of the execution unit processing the instruction.

A2.3 Constraints

Because this processor was developed with a small set of resources, limited CAD tools, and a small number of people with limited experience, we had to make trade-offs between performance and simplicity. Generally, we favored the simplicity to keep the project manageable. For example, we limited the data width to 32 bits and only support single precision floating point. In addition, we employ a static branch prediction scheme instead of a more complex branch target buffer or two level branchprediction algorithm. These sort of trade-offs allowed us to meet our objective without introducing unnecessary complexity, but make SAND somewhat less capable than a commercial grade processor.

Where possible we chose solutions that had good documentation. For instance, we used the MIPS R2000/R3000 instruction set since it is reasonably simple, and well documented. The MIPS manual became the guiding document for the operation of our processor. This gave us the added benefit of being able to use our existing compiler tools to generate example code.

A2.4 Normal Data Flow

The normal execution path for MIPS instructions in SAND follows seven stages. These are fetch, decode, issue 1–resolve, issue 2–lookup, execute, write back, and retire, as shown in Fig. 2.1.

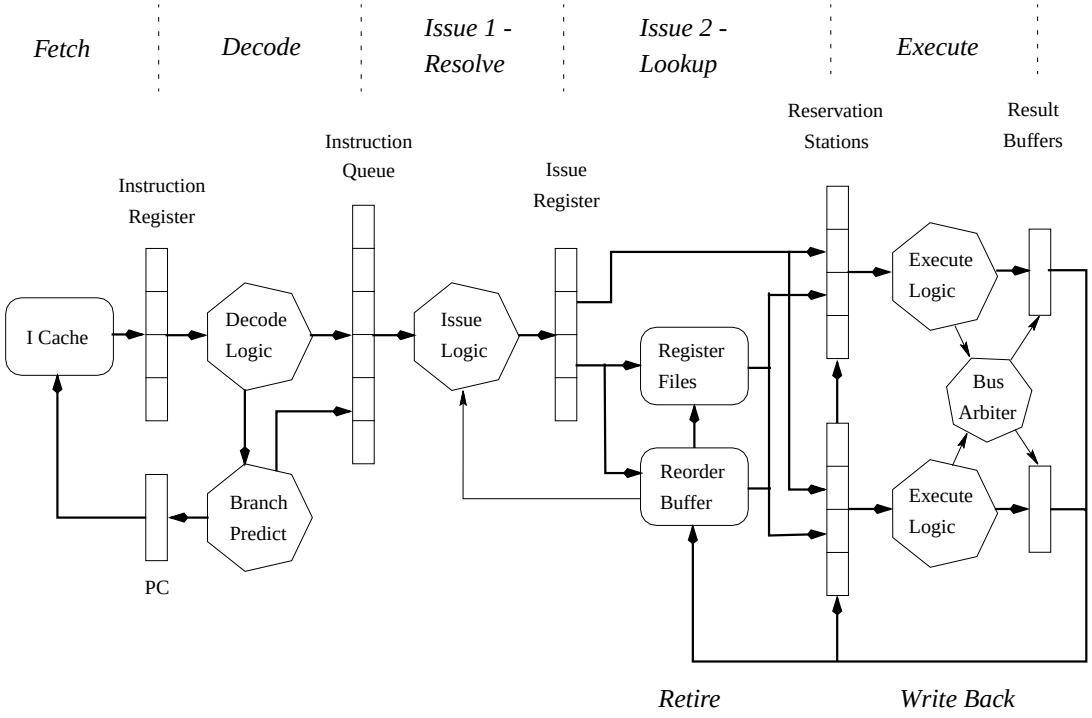

Figure A2.1 Normal Data Flow

A2.4.1 Fetch

The address of the current instruction block is latched in the program counter (PC) at the rising clock edge. The I-cache assumes this virtual address is on the same physical page as the previous one and performs the instruction lookup. At the same time the address translation takes place in the TLB to verify the page. If the address is indeed on the same page the new instructions are latched in the instruction register. If the physical page has changed, however, the instructions are discarded and another cache lookup is performed on the correct page. This creates a single cycle stall in the fetch—decode loop.

The I-cache arrangement uses one cache line four words wide. The refill lines have this same 128 bit width.

A2.4.2 Decode

The decode logic expands the four instructions in the instruction register. Also in this stage the branch predictor calculates and sets the next address of the PC based on the instructions in the register. The branch predictor uses a simple static algorithm. Forward branches are predicted not taken while backward branches are predicted taken. The prediction goes with the decoded instruction to the instruction queue to control the issue logic. Instructions coming after taken branches must be discarded by the issue 1 stage until the new stream is fetched and decoded.

The instruction queue was designed as two windows each holding four instructions. The decoder refills a window when it is empty, or stalls if both windows are full.

A2.4.3 Issue 1—Resolve

The issue logic looks at the instructions in the instruction register windows, oldest first, and picks the top four instructions. It ignores holes where a branch entered the middle of a fetch or left before the end of the fetch. The logic here is a large 8-to-4 selector.

The top four instructions are then checked for read-after-write (RAW) hazards and operand bus usage. SAND has four operand busses and allocates them to the instructions in order. When the operand busses are filled no more instructions can be issued. For example, assume the following were the top instructions.

Instruction	Operands Needed
1	1
2	1
3	2
4	1

Here five operands are needed so only the top three instructions can be resolved.

The instructions that can be resolved are deleted from the instruction queue and placed in the issue register. These instructions are matched with a tag from the reorder buffer that indicates the slot in reorder buffer where the completed instruction will write back its value.

These are called the result tags. Instructions not resolved remain in the instruction queue for processing on subsequent cycles.

A2.4.4 Issue 2—Lookup

In the second issue stage the reorder buffer and register files lookup the operands in the issue register. SAND has three register files: integer, floating point, and special. The special register file contains all the MIPS status registers. Each register file is capable of four reads and two writes. The reorder buffer is a 24 entry content addressable memory, easily expandable to 32 entries. The reorder buffer can perform four lookups per cycle.

The lookups in the register files and the reorder buffer proceed in parallel. When the reorder buffer finds a register match, it sources the most recent value on the operand bus and the register file value is ignored. If a match is found but the reorder buffer slot has not been filled with the data from a completed instruction, the tag of the slot is sent. The value from a register file is sent only if the reorder buffer contains no reference to the register.

The function codes, immediate values, and result tags from the issue register are matched with the operands and latched into the reservation stations. Each execution unit has its own reservation station sized to match the volume of incoming instructions and throughput of the execution unit. It is from these reservation stations that the instructions are issued to the execution units. Thus SAND supports out-of-order issue of instructions.

A2.4.5 Execute

The execute stage is where the processor does the work. The reservation stations hold an instruction until all its operands are available. An instruction may have come to the reservation station complete with operand values from the register files and reorder buffer, or it may have one or more tags from the reorder buffer. If an instruction operand contains a tag, the reservation station snoops the result busses for the tag. When it sees a match it grabs the returning data value and replaces the tag.

Instructions with complete operand values move to the execute logic. Execute blocks make up the integer units, floating point units, load/store unit, and branch resolution unit. Each are pipelines of varying length. For example, an integer ALU operation takes only one cycle,

whereas a floating point multiply takes four. All units except the floating point divider can start a new instruction every clock cycle.

When instruction execution completes, the execution unit requests bandwidth from the result bus arbiter. SAND has three result busses and most units need one of these to return their result. The integer multiplier and integer divider both return two words and, therefore, require two of the three busses. The arbiter follows a priority algorithm to determine bus allocation. Units with the most impact on the processor, such as the branch resolution unit, have the highest priority.

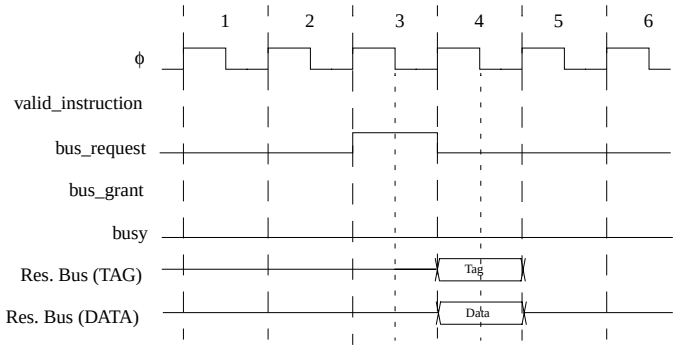

Figure A2.2 Result bus timing for a 3 cycle unit

The timing of the bus requests and allocations follows that shown in Fig. 2.2. The figure depicts a unit with a three cycle latency. On cycle one the unit receives the instruction and valid signal from the reservation station and performs the first stage of processing. Stages two and three of the pipeline occur on cycles two and three. On cycle four the result is ready for write back. However, in order for the bus arbiter to be able to allocate the unit one of the result busses on cycle four it needs the request on cycle three. Thus, execution units must request bus space during their last processing stage.

A2.4.6 Write Back

During write back the reorder buffer collects the data from the result busses and writes it into the slots corresponding to the result tags. These slots were allocated during the issue 1 stage. If a slot receiving a write back matches a register in lookup from the current issue 2 stage, the reorder buffer must forward the result to the lookup operand bus. The complete forwarding timing is shown in Fig. 2.3. If the result for slot A appears on the result bus during the same cycle that reorder buffer slot A matches a lookup from the issue register, the reorder

buffer forwards result A from the result bus to the operand bus. The reservation then latches result A as it would any operand coming from the reorder buffer of register files. If, as in case B, the reorder buffer finds no result data in the slot, it sends the slot tag to the reservation station. The reservation now snoops the result bus and catches result B when it appears on a result bus, as previously described.

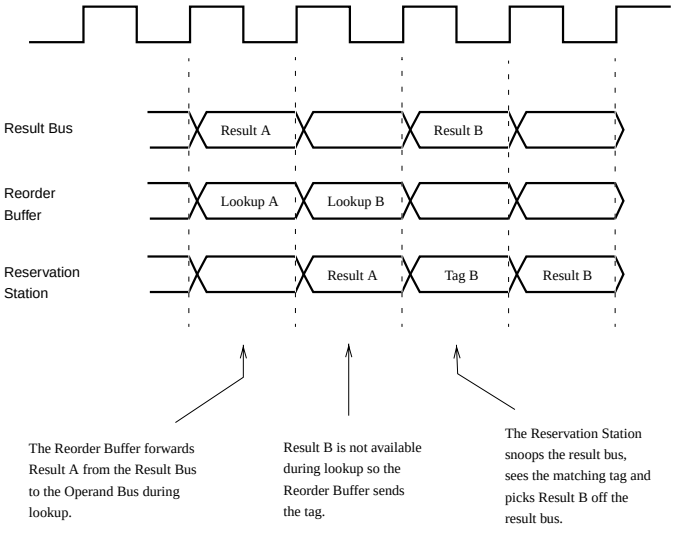

Figure A2.3 Result forwarding.

A2.4.7 Retire

The reorder buffer is organized as a queue with new instructions from the issue logic allocated slots at the end of the queue, and old, completed instructions retired from the front of the queue back to the register files. In SAND, the reorder buffer actually looks at the four instructions at the front of the queue and retires as many as it can. Thus, an instruction is retired when: 1) write back is complete, 2) it is one of the front four slots, 3) there are no incomplete instructions in the queue in front of this instruction, and 4) there is enough bandwidth to the register file for the write.

Each register file can accept up to two writes. Thus with three register files this allows two integer instructions, and two floating point instructions to be retired at the same time. However, three integer instructions cannot be retired at the same time.

The special registers contain additional circuitry to allow writes to certain addresses to actually write more than one register, and others to write only bits. This is needed, for exam-

ple, on a TLB miss where the bad virtual address, context, and entry hi registers all receive the virtual address from one instruction.

A2.4.8 Resource Sizing

A2.4.8.1 32 Bit Data Path

In order to reduce the magnitude of the circuit design, SAND employs a 32 bit data path throughout the processor. All the busses and execution units use 32 bits. Because of this the floating point units are limited to single precision. Integer multiplies and divides still return 64 bit values which are treated as two 32 bit results.

A2.4.8.2 Instruction Issue

The issue logic handles a four instruction peak rate, and a two instruction average rate. Branching out of and into the middle of fetch lines accounts for much of the reduction. The operand busses were sized to the average rate of two instructions, with two operands each, or four operand busses.

A2.4.8.3 Saving Resources with R0 and Next Cycle Operand

In MIPS the integer register R0 (zero) is always set to zero. In SAND the issue logic detects lookups of R0 and sends a signal to the reservation station instead of doing a lookup. The reservation station generates the zero value and inserts it in the operand field. This saves an operand bus.

In addition, it often happens that there are an odd number of lookups needed by the issue logic. For example, instruction 1 needs one operand, instruction 2 needs two, and instruction 3 needs two. Since there are only four busses the issue logic would have to settle on issuing two instructions, wasting one operand bus.

In order to utilize that remaining bus the issue logic sends the third instruction with one operand, and holds the second operand lookup until the next cycle. The issue logic sends a signal to the reservation station to expect the second operand on the next cycle, where it will appear on the first operand bus.

A2.4.8.4 Result Collection

To write back the average two instructions per cycle, SAND uses three result busses. This accommodates the peak created by out-of-order instruction processing, and integer multiplies and divides returning two results.

A2.5 Branches and Exceptions

In order to implement branching and exception handling, SAND incorporates signal paths from the branch resolution unit execute block and the reorder buffer to the branch predict unit. The branch resolution unit and the reorder buffer detect changes needed in the PC and the branch predict unit actually controls the PC. The block diagram is shown in Fig. 2.4.

A2.5.1 Branches

Because our processor follows the R2000/R3000 MIPS architecture, it must allow for a delay slot following branch instructions. This puts the maximum number of branch instructions in a four instruction fetch at two. The branch predict unit takes advantage of this by grouping instructions one and two, and instructions three and four together.

When a branch instruction appears in one or both of the groups the branch predict unit calculates the taken address for each branch. Because the prediction scheme is a simple static algorithm where backward branches are predicted taken, and forward branches are predicted not taken, the predict unit looks at the sign of the branch offset to determine the new PC. It sets the PC mux accordingly. The taken and not taken addresses, as well as the prediction are stored in the branch address queue for later use by the branch resolution unit.

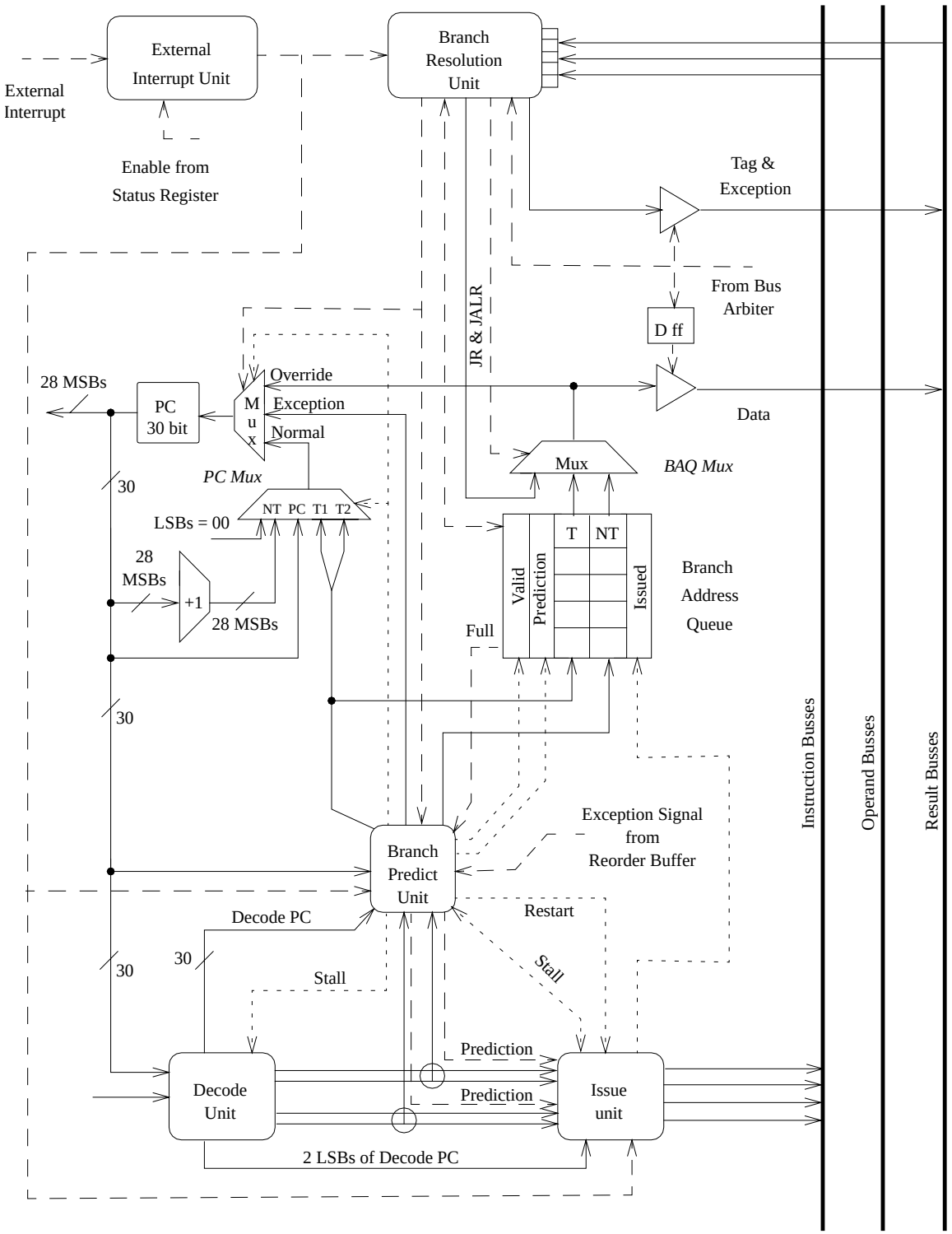

Figure A2.4 Branch unit block diagram.

The branch address queue provides two important functions. First, it reduces operand bus usage by transporting the branch addresses to the branch resolution unit by a separate mechanism. Otherwise, each branch would have to use two more operand busses for this purpose. Second, it allows the reservation station at the branch resolution unit to be identical to the reservation stations at other units. Communicating the addresses over the operand busses would have required more storage at the reservation station.

The branch resolution unit acts as an independent execute block to test the branch condition. The resolution unit decides if a branch should have been taken or not, then checks the prediction to see if it was correct. If incorrect, the resolution unit overrides the control of the PC mux and resets the PC with the correct address from the branch address queue. The true address of the branch, regardless of the prediction, is returned to the reorder buffer as the result of the operation.

The resolution unit also handles jump register instructions. In this instruction the target is not in the instruction word but must be fetched from the register file. The fetch goes to the branch resolution unit as a normal instruction. The resolution unit sends the address through the branch address queue (BAQ) mux and the override mechanism to the PC.

A2.5.2 Exceptions

SAND implements precise exception handling with the reorder buffer as the focal point of the exception process. The result busses each contain a five bit field called “exception” to communicate the status of the returning result to the reorder buffer during the write back stage. The execution units put an “OK” in this field to indicate normal instruction completion, or a code for an exception when an instruction warrants. The reorder buffer checks the code at the retire stage. If it contains an exception the reorder buffer signals the branch predict unit to load the PC with the appropriate exception vector. The reorder buffer then clears all remaining entries and signals the execution units to clear also. At the same time the status and cause special registers (defined by MIPS) are set to start exception processing.

In order to be precise, however, the processor needs the address of the excepting instruction. The PC cannot be used as it has advanced beyond this instruction, possibly across one or more branches. To handle this the reorder buffer tracks the address of the last retired instruction in a register called the reorder buffer PC or RBPC. It is the value of the RBPC that is

loaded into the exception program count (EPC) special register at the start of exception processing, and returned back to the RBPC during a restore from exception instruction.

To help the reorder buffer track addresses across branches and jumps, the branch resolution unit returns the true address of these instructions in the data field of the result bus. The exception code tells the reorder buffer that the data field contains a branch address. The reorder buffer writes this address into the RBPC during the retire stage.

The exception code from the branch resolution unit also tells whether the branch was or was not predicted correctly. The reorder buffer scans the result busses for mispredicted branch exception codes in the write back stage. When one is detected the reorder buffer immediately marks as invalid all allocated slots in the buffer that come after the branch delay slot. These instructions were all speculative, based on the prediction being correct. The invalidation must occur before any instructions from the correct program flow reach the issue stage where they will be allocated space in the reorder buffer. Because of this, the branch resolution unit is always granted a result bus when it has a result.

Note the incorrect, speculative instructions are not immediately purged from the reorder buffer and execution units. Instead, they are allowed to complete and the results are discarded at the retire stage. This saves having to make a checkpoint mechanism for each execution unit. The reorder buffer handles the whole process. Also, mispredicted branches are restarted quickly, minimizing the penalty. The new instructions simply follow the invalidated ones.

A2.6 Memory Management

The SAND memory management unit (MMU) consists of the load/store execution unit, the TLB, the instruction and data caches, and the external memory (system bus) interface. These pieces operate together during the fetch, execute, and retire stages. The external data path width is 128 bits, matching the line size in the I-cache. Each cache is 8KB in size.

During fetch the MMU provides instructions to the instruction decoder as described earlier. However, if the fetch operation generates an exception there is no instruction yet allocated in the reorder buffer to pass an exception code and start exception processing. Reorder buffer slots are not allocated until the issue stages. To overcome this problem the MMU signals the issue unit that an exception has occurred during the instruction fetch. The issue unit responds by creating an internal instruction called “I fetch exception” and sends it to the load/

store unit. The load/store unit processes this instruction by returning the exception code to the reorder buffer.

Data movement between the processor and memory flows through the load/store unit. The unit first coordinates the lookup of the physical address in the TLB. SAND uses only one lookup logic block in the TLB that the instruction and data translations must share. Normally instruction addresses are being searched in the TLB in parallel with the I-cache lookup. The last physical page of instruction addresses is stored in a register, sort of a TLB cache, so this TLB lookup is not needed as long as the instruction is on the same page. When a data lookup is required the load/store unit preempts the instruction lookup. The instruction fetch in the I-cache proceeds as normal, and most often incurs no penalty. However, if the instruction page changes at the same time the TLB is processing a data address, the instruction fetch will have an additional wait cycle. This is acceptable as load and store instructions comprise a small fraction of most programs, so the probability of penalty is low.

Within the load/store unit, store instructions are kept in a store buffer until the reorder buffer signals they can be safely retired to memory. With this handshaking the reorder buffer controls the retirement of all instructions, whether to a register file or memory. During a load instruction the store buffer is first checked for a matching address. If the buffer holds the address, the value is forwarded to the load and the memory lookup is not performed. This is called load forwarding.

A2.7 Coping with Pipelining in MIPS

The MIPS R2000/R3000 was designed as a pipelined control integer processor with a separate floating point coprocessor. Certain instructions and registers were specified for efficient use in a pipelined control processor. In a superscalar implementation like SAND, however, these optimizations become liabilities. For example, the delay slot helps the pipelined control, but hurts the superscalar processor by limiting the use of a branch target buffer. This section describes how SAND deals with these problems.

A2.7.1 Special Registers

MIPS specifies a number of status registers that are contained in its coprocessors: CP0, the system control coprocessor, and CP1, the floating point coprocessor. In SAND control is distributed throughout the processor, with the main CPU control in the branch predict unit, issue logic, and, especially, the reorder buffer. Accordingly, the status registers were consolidated into a special register file which the reorder buffer controls during lookup and retire.

However, certain exceptions require more than one of these registers to be written at the same time. For example, during a TLB miss the bad virtual address register, as well as parts of the entry hi and context registers, receive the virtual address of the instruction that caused the exception. To accommodate this, writes to certain addresses in the special register file actually write into more than one register. In this case a write to the bad virtual address register also writes into part of the entry hi and context registers.

A2.7.2 Processor State Changes

Certain instructions cause the operating state of the processor to change, or context switch. These include writes into the TLB, moving words into some status registers, and a restore from exception. When any of these instructions is issued, subsequent instructions must wait until the processor state change has completed before they start execution. The SAND issue logic detects the state changing instructions and stops issuing further instructions until in the new state. The reorder buffer facilitates the process by sending a signal to the issue logic whenever it is empty. Thus, the issue logic just waits after issuing a state change instruction until it receives the empty signal from the reorder buffer. This effectively causes serial execution of these instructions.

A2.8 Implementation

The target technology for SAND was 0.8μm CMOS. The goal was to produce a design with a clock period of 10ns (100MHz). We used the following global rules for the design.

Single phase clock.

All data moves on the positive clock edge.

Static CMOS construction.

We engineered interesting parts of the processor down to layout. The rest were designed as structural or behavioral verilog. The least interesting parts, such as the integer ALU, were implemented behaviorally.

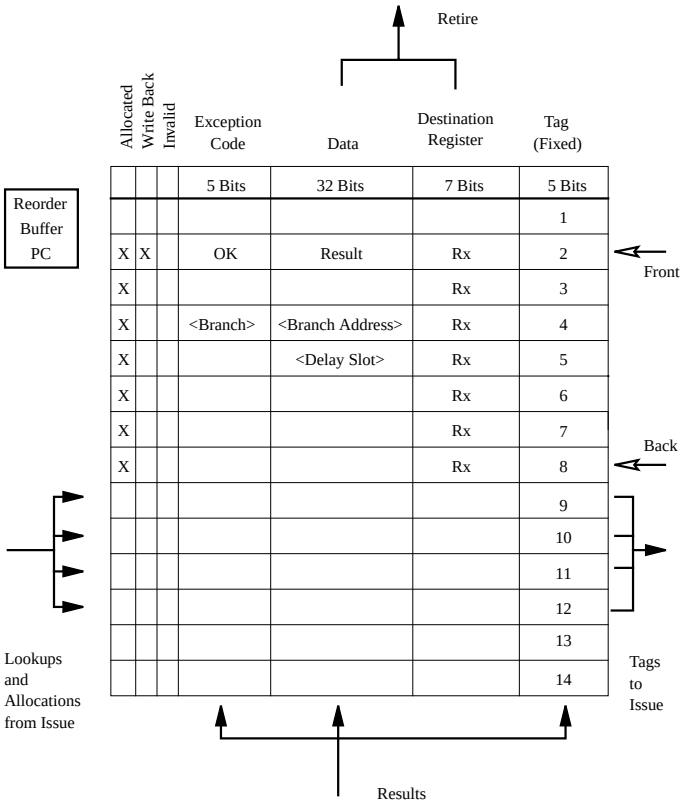

Figure A2.5 Reorder buffer

The heart of the processor contains the reorder buffer, shown in Fig. 2.5. The buffer was designed as a circular queue. In this design the data remains in a fixed location and “pointers” track the front and back of the queue. The reorder buffer accomplishes the following functions in parallel.

1. Perform a content addressed lookup of four register numbers, each seven bits. This includes the result busses as well as the buffer.
2. Allocate slots for four new instructions from issue.

3. Write back three results from the result busses. These are indexed by the destination tag.
4. Check for mispredicted branch write backs on the result busses. If one is present, invalidate all instructions following the branch delay slot.
5. Decide which instructions can be retired and send them to the correct register file. Signal the load/store unit to retire store instructions.
6. Increment the reorder buffer PC for retired instructions. If a branch or jump is retired, load the reorder buffer PC with the data from the instruction.
7. Move the front and back pointers based on the new instructions allocated and the old ones retired.

Because of the many complex functions the reorder buffer is large. Furthermore, the register lookup and delivery to the reservation stations sets the critical path. But, in spite of this complexity, the reorder buffer design is straightforward. Most of the functions apply to each cell, so once a cell is designed it need only be replicated to create the full buffer. This also allows for easy expansion of the buffer.

A2.9 Conclusion

This appendix described the design of a superscalar MIPS processor that uses a reorder buffer to allow out-of-order issue and completion of instructions. We created this design as an example to aid the development CAD tools and methodologies for system level optimization of multichip modules. While necessarily complex, our design was kept as simple as possible by eliminating unneeded functionality, and sacrificing some performance to gain simplicity.

Appendix 3 The NCSU Cadence Design Kit

A3.1 Purpose of the Project

In the course of examining ideas for implementing hardware, in our case systems on MCMs, researchers need an electronic CAD tool infrastructure for two purposes. One is to design and test prototype circuits, and two is to provide a vehicle to support new CAD algorithms that would be developed. The first purpose is clear. When researching hardware it is usually necessary to build hardware. The second purpose is not as clear cut, but makes a lot of sense. When developing CAD tool prototypes, it is much harder to build a standalone program that can function independently, than it is to build one in the context of an existing flow. The CAD infrastructure provides a vehicle for the integration of new ideas.

Cadence provides their complete tool set to universities at a small cost. The tool set is comprehensive, and it comes with a user accessible extension language, so it fulfills both needs well. However, the tools come without any fabrication process setup. They are just tools. In order to be useful we had to develop the process data, sometimes called a process design kit, or PDK, to fabricate using the MOSIS broker service. This appendix gives an overview of the kit we developed. We call it the NCSU Cadence design kit, or CDK [101].

A3.2 Functionality

The CDK is a collection of technology files, custom SKILL routines, parts libraries, and Diva rules files aimed at facilitating full-custom CMOS IC design through MOSIS. The CDK provides customizations for the Cadence Design Framework, Composer, Analog Artist, Verilog, Virtuoso and Diva tools. It allows for full custom IC analog, mixed-signal, and digital design.

A3.2.1 Design Framework

MOSIS provides access to several CMOS fabrication processes, Some processes have options associated with them. For example, there may be a different number of metals layers for each process, and the fabrication cost is predicated on the number of layers used. There

can be other special features too, such as a linear capacitor well implant. For most new users, especially undergraduates, this can be confusing. The CDK attempts to hide some of this detail by simplifying library creation, using a customized form.

The user types the name and directory of the new library. If the library is to include mask data (i.e., layout), the user also chooses the appropriate process by clearly understandable name rather than by an obscure technology code. The CDK includes a technology library for each MOSIS CMOS process; by simply clicking the appropriate button, the user either attaches the new library to one of these pre-existing technology libraries or compiles the new library as its own technology library.

The selection of a process provides access to the mask layers available in that process. For example, layouts done in HP's four-metal process CMOS10QA can use metal 4, while layouts done in HP's triple-metal CMOS14TB cannot. After the library is created, the CDK creates links in the library's directory to the site wide Diva verification rules files.

In a similar vein, the CDK replaces the stock form for attaching a technology library to a design library with one that refers to the MOSIS CMOS processes by name.

A3.2.2 Composer (schematic entry)

To facilitate schematic design and simulation, the CDK provides customized libraries. These are called NCSU Analog Parts and NCSU Digital Parts, and they contain common parts such as logic gates, transistors, and RLC components for schematic capture and Verilog and SPICE simulations. They also include some higher-level parts, such as multiplexers and flip-flops. Originating from the Cadence provided sample, analogLib, and US 8ths libraries, the NCSU part libraries were created to consolidate the commonly-used components in a way that would allow us to add parts, for example multiplexers, and modify the parts with custom Component Description Format (CDF) properties that tie into the rest of the setup.

The components in these libraries are technology-independent. There is no layout data in these libraries. SKILL callbacks triggered by the CDFs assign parameter values, for example, minimum width and model name for transistors, to the parts when they are placed in a schematic.

The CDFs:

- Enforce gridding. Transistor widths and lengths must be a multiple of one-half lambda.
- Enforce minimum transistor widths and lengths.
- Automatically select the correct SPICE model based on the library's technology.
- Use a simple heuristic to estimate the source and drain areas and perimeters.
- Set properties so that the user can take advantage of the technology library's parameterized cells when creating a layout.

A3.2.3 Analog Artist (circuit simulation)

Circuit simulation is done through Analog Artist. There are relatively few customizations with respect to this tool. The CDK does include several directories containing transistor models, and the CDK's startup files set the necessary variables so Artist can find the proper directory for the process in use. The CDK includes all the models we are allowed to distribute (HSPICE level 13 and Spectre level 4), which are obtainable from the MOSIS. The CDK supports more detailed (HSPICE level 39) models, but these must be obtained from MOSIS directly.

A3.2.4 Verilog (digital simulation)

To assist digital simulations we provide Verilog primitives with the components in the NCSU Digital Parts library. Our Verilog hierarchy consists of "functional" views as structural parts or logic gates, and "behavioral" views as more abstract blocks. The simulation setup netlists the Verilog using this hierarchy so different abstractions are possible.

A3.2.5 Virtuoso (mask layout)

This section describes the CDK components used in mask design with Virtuoso.

A3.2.5.1 Mask Layers

All mask layers for MOSIS SCMOS processes are supported, along with the optional layers that are not common to all processes, such as Orbit's layers for NPN BJTs. Adding support for

a new process is fairly straightforward, especially if the process does not introduce any new mask layers, and is described in the CDK documentation. NCSU also releases patches to bring existing CDK installations up-to-date as new MOSIS processes are announced.

The layers are defined in the technology files. These are generated from the MOSIS process manual. There is documentation included in the CDK that lists the Cadence layer name, a description, the GDSII number and CIF abbreviation of all SCMOS layers as well as the process(es) for which they are valid.

As discussed above, the CDK eliminates the temptation to use layers which do not exist in the technology used in a design library by not defining those layers in the library when it is created.

A3.2.5.2 Parameterized Cells

Parameterized cells (pcells) are a powerful way to assist full-custom layout by eliminating the need to manually draw every polygon of common structures, such as FETs and contacts. Included in each MOSIS technology library are pcells for the following constructs:

- NFET/PFET
- N/P ohmic contact
- metal1–N/P diffusion contact
- metal1–poly/poly2 contact
- metal–metal contacts between layers
- thin-oxide linear capacitor.

The FET parameters include the number of poly gates, in either serial or parallel configuration, as well as width and length. The contact pcell parameters include number of rows and columns, and all contact pitches are the minimum allowed by the library's technology. The thin-oxide capacitor, currently available only in the HP CMOS14TB process, can be described by either total capacitance desired or by width and height.

Transistors from the NCSU Analog Parts library are automatically set to use the FET pcells when creating devices in the layout editor.

A3.2.6 Diva (layout verification)

All verification, DRC, extraction, and LVS, is done with Diva. The rules files are stored under the standard filenames divaDRC.rul, divaEXT.rul, and divaLVS.rul in the local/techfile directory of each technology library.

A3.2.6.1 DRC

All rules from the MOSIS SCMOS User's Manual, revision 7.2, are checked. The value of lambda is stored in global data file, globalData.il, and used by Diva when performing DRC checks. In addition to the SCMOS rules, a few extra rules are implemented that are not in the SCMOS manual but should be followed anyway. One such rule is not allowing p-type select inside cwell.

In the library MOSIS Layout Test is a layout, based on one provided by MOSIS, which consists of a group of DRC-test structures which exercise every design rule.

A3.2.6.2 Extraction

The following circuit elements can be extracted:

- FETs
- vertical NPNs
- PN/NP diodes
- poly-metal1/thin-ox/polycap capacitors
- parasitic capacitors.

A single SKILL variable sets the threshold below which parasitic capacitances are ignored. Parasitic capacitance values are based on data published on the MOSIS Web site.

A3.2.6.3 LVS

LVS matching is supported. Switches in the LVS rules file are activated from a custom form accessible through the NCSU menu. The user can select LVS rules on a library-by-library basis.

A3.2.7 Other Functionality

The following are some of the miscellaneous features the CDK brings to Virtuoso:

- Symbolic contacts are provided to allow path-stitching with the path tool.
- The label creation process is similar to Composer's. Multiple labels can be typed into the form at one time, and array notation is allowed. For example, dataBus[7:0] will place eight separate labels.
- Elements (shapes and instances) can be selected and aligned to make their edges flush with a selected object.
- The kit includes programs that allows a user to import to layout JPEG images. This is handy for scanned-in logos or signatures, or other structures where you have picture of the structure. It also supports text strings in any available X Window font, which is quite useful for creating on-chip labels and markers.
- For teaching classes, and research projects, the kit has hooks to include individually customized setup files. These allow specialization for the needs of separate projects while still leveraging the base setup provided by the CDK.

A3.3 Conclusion

This appendix described the contents of the CDK in general terms. The CDK is used at N.C. State University in both teaching and research, and it has been used to fabricate working chips. The latest version of and patches for the NCSU CDK are available at http://www.ece.ncsu.edu/cadence/CDK.html. There you can find up-to-date information on the CDK, and download it for your site.