DIGITAL CIRCUIT AND BOARD DESIGN FOR A LOW POWER, WIDEBAND CDMA RECEIVER

by

Ian David O'Donnell

Memorandum No. UCB/ERL M96/82

20 December 1996

DIGITAL CIRCUIT AND BOARD DESIGN FOR A LOW POWER, WIDEBAND CDMA RECEIVER

by

Ian David O'Donnell

Memorandum No. UCB/ERL M96/82

20 December 1996

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Table of Contents

List of Figures

List of Tables

Acknowledgments

As with any endeavor, this research project could not have been completed without the help and support of others. Foremost, I wish to thank my research advisor Professor Robert Brodersen for his guidance throughout the course of this project. I would also like to thank Professor Jan Rabaey for reviewing this thesis and for further instruction on digital VLSI design.

I am greatly indebted to other members of this project as well. Sam Sheng has a vast wealth of knowledge that he is very willing to share and Craig Teuscher was helpful in explaining communication theory. Lapoe Lynn, Jim Peroulas, Kevin Stone, and Dennis Yee were fun to work with and also contributed good ideas and a lot of hard work to the project.

In addition, several other people outside the project provided necessary diversions and insight along the road. In no particular order they are: Anantha Chandrakasan, Tom Burd, Dennis Yee, Andy Burstein, Heather Bowers, Arthur Abnous, Shankar Narayanaswamy, Roy Sutton, and Leah Fera. For their administrative support, Peggye Brown, Tom Boot and Elise Mills receive my sincere gratitude. I would also like to thank the California MICRO program and the Advanced Research Projects Agency for their generous financial support. And, in general, I would like to thank my professors and fellow students/ colleagues here at Berkeley. They have consistently been of high caliber and it was a pleasure working with you all.

Finally, I would like to thank my family and friends for their support and encouragement.

1 Introduction

This thesis covers digital design issues relating to custom and semicustom integrated circuit and printed circuit board design associated with the high-speed, digital backend to the InfoPad's Spread-Spectrum, Direct-Sequence CDMA radio [Sheng91], [Sheng96]. The goal of this radio system is to support up to 50 users per picocell at a rate of 2 Mb/s each which requires a sampling rate of 128 MHz for the digital receiver. The digital baseband circuitry implements timing and data recovery, hand-off, and channel estimation for a battery powered, hand-held mobile unit. Hence, low power consumption was a primary issue in the design. A low power, low cost custom digital ASIC was designed and fabricated to provide a subset of this functionality in 57mm^2 in a 'pseudo' 0.8u CMOS process, dissipating 19mW in half-speed operation. Coarse and fine lock, raw data recovery (no DQPSK decoding), and some channel correlation estimation are currently implemented and have been tested. A redesign is also underway to complete the desired functionality, including DQPSK decoding, adjacent cell scan, and multipath channel and data correlation estimation. This thesis describes the design process and documents the testboard and important integrated circuit structures in the backend chip. Together with [Stone95], which covers the control circuitry and an overview of the desired functionality, the complete digital backend chip is described.

First the desired and currently implemented functionality are reviewed, then a method of empirical process characterization for digital circuits based upon a parametrized, automated SPICE file of a ring oscillator and single transistors is presented. The results of several relevant processes are presented and later used to help determine architectural trade-offs.

Following the characterization, the custom designed correlator is explored; an important datapath block that constitutes the majority of functionality and area for this chip. The design approach is outlined along with simulation results and measurements.

After the design of the first correlator, a redesign was attempted in a better process using an offset binary encoding representation to lower both the power and area. Although that redesign failed in some of its goals, it helps illuminate a better path to low power design and it implies that area can be directly traded-off for low power operation. Additionally, the redesign explores digital circuit implementation and techniques for reducing the critical path and their impact on area and power.

Moving up a level of hierarchy to the semicustom DQPSK design, the low power library cells are characterized for power, and the resulting data is used to evaluate multiplier algorithms on an area*power^2 metric. The optimal solution is a fairly small, low power iterative multiplier used in a simple DQPSK demodulator.

Finally, the issue of testing at the chip and board level is explored and the testboard design is explained along with some useful hints for board design. Some suggestions are also made regarding on-chip test structures to help simplify testing and system integration.

In the conclusion the current status of the redesign is reiterated and the work up to this date is reviewed along with a list of the tasks still left to be completed.

2 Chip Functionality

The system level specification for the CDMA radio's digital backend is covered in more detail in [Sheng91], [Sheng96] and [Stone95] but is reviewed here to familiarize the reader with the system constraints and desired behaviour of the chip.This chapter is divided into three sections: the first reviews the CDMA system and backend requirements, the second discloses the current functionality of the digital backend chip, and the third details the goals for the revised version of the chip which is still in progress.

2.1. The CDMA System

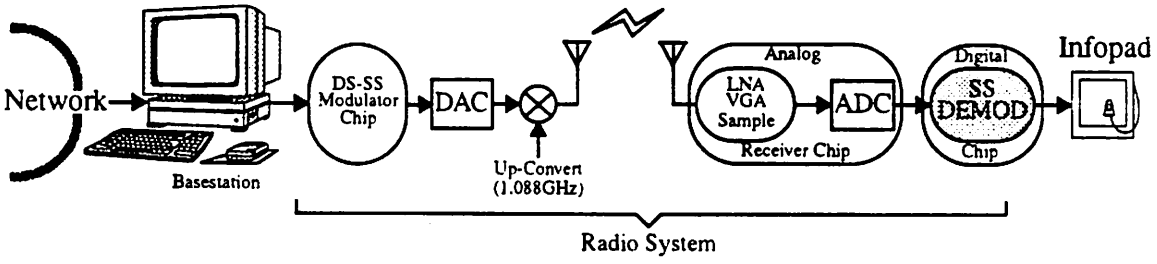


Figure 2-1. CDMA Radio System

For a more detailed description, please refer to [Sheng91], [Sheng96] for the overall system architecture and Analog Receiver chip, [Lynn95] for the ADC and VGA (Sample and Hold) in the receiver, [Stone95] for the digital demodulator chip overview and control, and [O'Donnell96] for digital demodulator design circuit specifics and testboard. On the transmitter side, see [Peroulas96] for the digital Direct Sequence Spread Spectrum (DS-SS) Modulator chip, [Yee96] for the transmitter up-conversion design.

The goal of the InfoPad CDMA radio project is to provide wireless access for 50 users per basestation at a data rate of 2 Mb/s each. The data is DQPSK encoded into a single complex 1 Mb/s stream per user for an aggregate rate of 50 Mb/s. A 1 bit PN

sequence (treated as multiplies by +/-1's) derived from the linear feedback shift register technique (length 32768) is used to provide the direct-sequence spreading at a chipping rate of 64 Mchip/s. Code division for the users is accomplished through the additional overlaying of a 6 bit Walsh code. Code 0 is reserved for a pilot tone for synchronization and channel estimation which theoretically allows for 64 orthogonal users in the system (in reality less than 50 due to interference, see [Teuscher 95]).

The radio transmitter takes in user's data modulated by the appropriate Walsh code, spreads, combines and filters the composite signal by a 30% excess bandwidth raised-cosine (resulting in a -3 dB transmit bandwidth of about 83 MHz). The filter output is then mixed up to the carrier of 1.088 GHz, a frequency chosen to place the downsampled (at 256 MHz) result of the receiver at 64 MHz to avoid DC offset and 1/f noise in the sampling switches.

On the receiving side, the 1.088 GHz signal is filtered and amplified by an LNA before being downconverted (subsampling demodulation) by a pair of sampling switches (one offset by 90 degrees at 256 MHz from the other). The sampled outputs travel through a bank of VGA's each, before finally being flash A/D converted into two 128 MHz streams (interleaved on-time and quadrature) of 4 bits in sign-magnitude format. From that point the data is input into the digital backend chip which must perform the following functions:

1. Synchronize to the pilot tone to provide coarse lock (on the order of T_{chip}).
2. After lock, activate a digital delay locked loop (DDLL) to provide fine timing recovery (on the order of $T_{chip}/4$, about 4 ns).
3. After lock, perform data recovery.
4. After lock, provide for 3 taps of channel estimation (on time plus two delays) to allow for RAKE combining.
5. After lock, scan for adjacent cells and provide support for hand-off.

A block diagram of the digital baseband receiver chip is shown below [Stone 95].

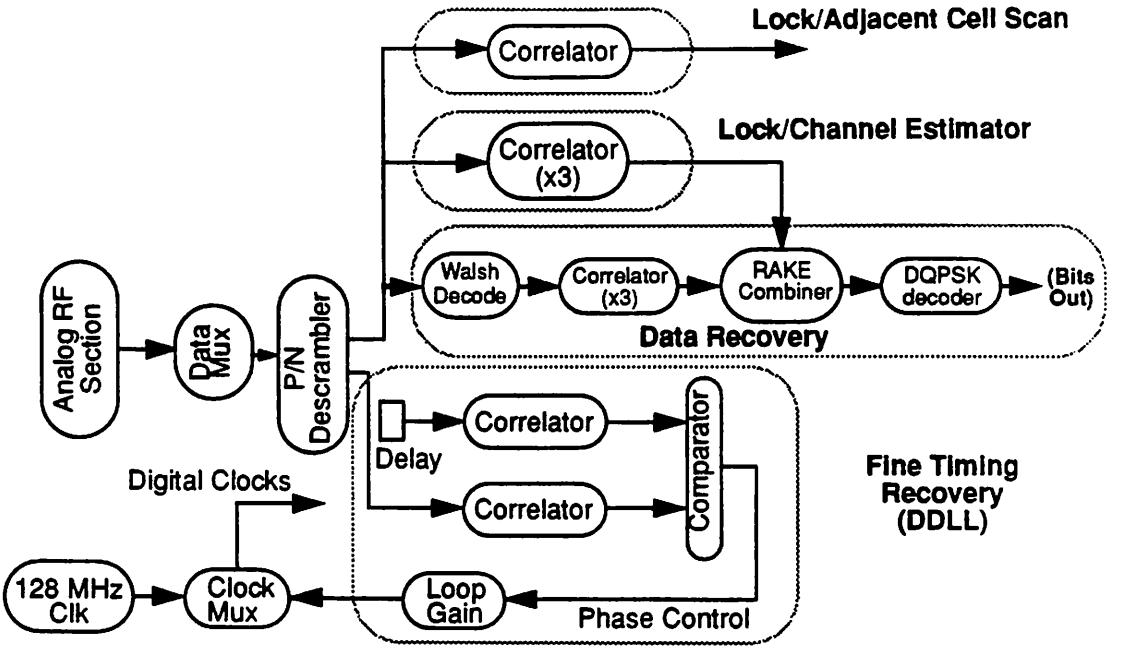


Figure 2-2. Digital Baseband Receiver Architecture

2.2. The Current Digital Backend Chip

To date a first pass at the digital backend functionality has been performed and we have a chip that implements coarse and fine timing recovery (items 1 and 2), raw data recovery (item 3 without DQPSK demodulation), and allows for observation of the multi-path channel energy estimations by viewing delayed correlation results (sort of item 4). The chip contains around 80,000 transistors and was fabricated in HP's (pseudo) 0.8μ process (all dimensions are 1.0μ drawn, except for N-MOSFET's which are mask biased to a

0.8μ channel). The size of the chip is 56.56 mm^2 (7.69 mm x 7.36 mm) and it was packaged in a standard 132 pin PGA. A die photo of the chip is shown below.

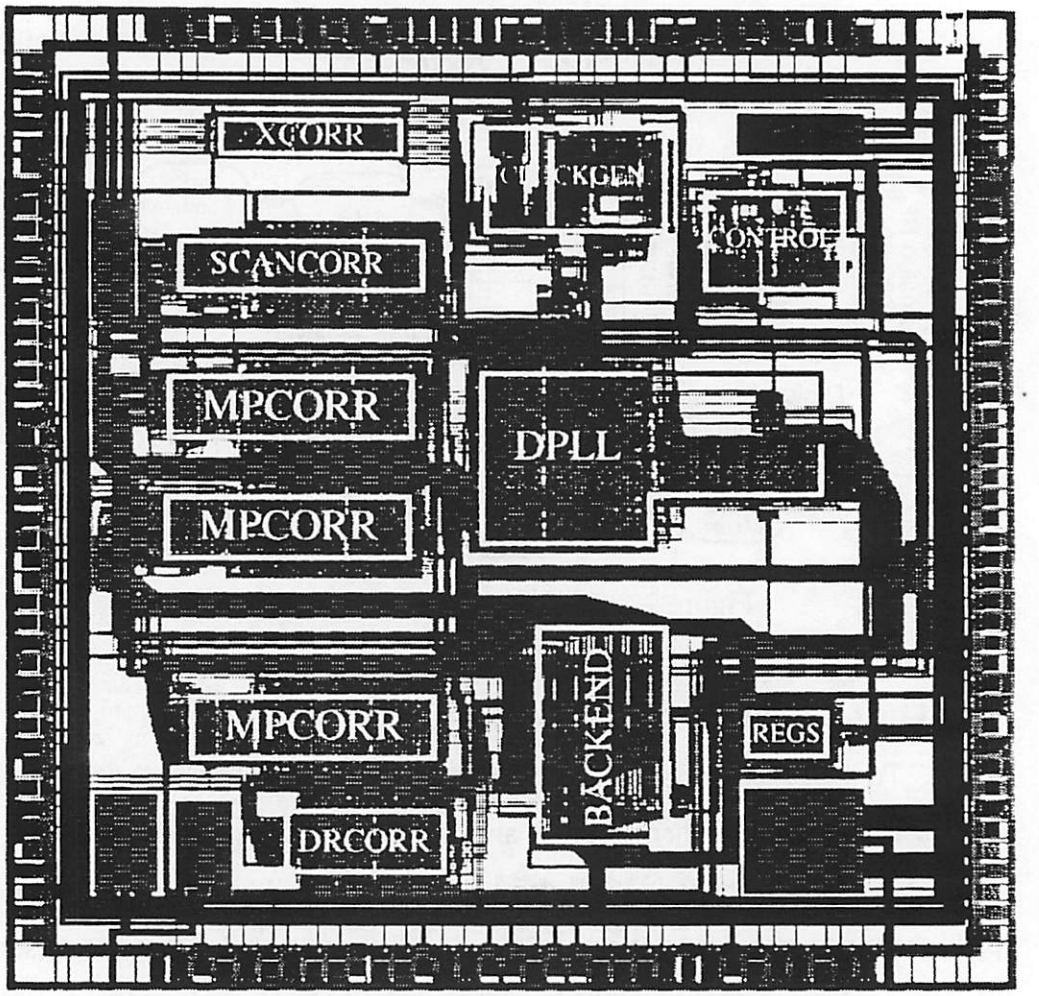


Figure 2-3. Micrograph of Digital Baseband Receiver Chip

The chip has been tested at 20 MHz (the correlator, separately, to half-speed, 32 MHz operation) using Tektronix's DAS9200 system and has been found to be functional. A description of the testboard and testing strategies can be found in a later chapter in this thesis. The correlator was measured to consume 0.6mW at 1.5V, 32 MHz and is estimated to consume 1.2 mW each at full speed. Three supplies are used for the chip and the power consumption breaks down as follows for half-speed (64 MHz clock) operation [Sheng ISSCC96]:

- 4.2 mW at 1.5V (measured)
- 7mW at 3.3V (estimated)
- 5.5mW at 5V (estimated)

The total estimated half-speed power consumption is 18.7 mW implying a full-speed power consumption around 37 mW. The chip has not yet been tested at full-speed owing to system issues and a lack proper test equipment: The DAS only pattern generates to 50 MHz (64 MHz is needed), and a full system test cannot be done until the upconversion board for the transmit side has been finished. Full verification has not been achieved, but is being addressed along with the ongoing chip revision.

Note that three supplies are used for the chip in an attempt to obtain the lowest possible power by voltage scaling. Recall from

Equation 2-1. $$P_{dyn} = C_{eff} V_{dd}^2 f$$

that dynamic power is a strong function of supply voltage. The idea behind using multiple supplies was to split the chip into voltage regions based upon performance requirements, so that each region would receive a voltage that was high enough to accommodate the necessary delay, but no higher. The correlators constituting the datapath, projected to be a dominant power consumer, were hand-designed to 1.5V, while the VHDL synthesized control logic, a relatively minor power consumer, could be placed at 3.3V to allow for extra delay margin without dominating the overall consumption. The clock (128 MHz) was run at 5V to preserve the edges, before being locally down-converted to 3.3V for the control logic. This use of multiple supplies makes life a little difficult at the system level, as the board designer needs to provide these supplies; however research by [Stratakos97] suggests the possibility of on-chip, efficient DC-DC converters which would remove this system constraint, allowing for arbitrary on-chip supplies for low power operation. Until demonstration of this functionality (anticipated within a year or so), we will probably continue to simply use multiple power supply units to drive the chip. Development solutions (separate Maxim DC-DC converters for example) exist to address the issue of multiple supplies, although they may not be as elegant as using a single supply.

2.3. The Proposed Revision of the Digital Backend Chip

The first pass of the chip is adequate to verify the operation of the CDMA radio design, but lacks a couple key features. Primarily the lack of a DQPSK decoding unit on-chip and the inability to perform hand-off prohibit the inclusion of the radio into the Info-Pad environment. The first pass was not intended to be a full-blown radio, though, it aimed to demonstrate the system functionality. Towards the end of an integrated InfoPad radio, a revision of the backend chip was undertaken to augment its functionality along with fixing some minor bugs discovered in the first pass.

The goal of the revised backend chip was to fully support the functionality mentioned in section 1.1. Namely this includes adding adjacent cell scan and hand-off ability (item 5), adding a DQPSK demodulator (finish item 3), and allowing for fast observation of channel estimations to allow for post-chip RAKE receiving (minor internal hardware correction to allow for item 4). In addition to adding functionality, the revised chip gives us the opportunity to re-examine the original design, especially in light of a now available 0.6μ CMOS process. A re-engineering attempt was made to take advantage of the new process, hoping to build a correlator with smaller area and lower power. Also, several minor bugs in the control logic will be fixed, including:

- The use of dynamic registers to hold state in the control logic (threshold values) -- These decay over time and need to refreshed in the current chip. Static registers should be used.
- The CLKRST line is phase sensitive relative to the 128 MHz clock. If CLKRST goes low after the rising edge of CLK128H instead of after CLK128L then the phases of internal clocks are inverted from what was intended and the DDLL will push the chip out of lock instead of into it. (CLKRST was added in the control logic at the last minute to try to guarantee the phase of the internal clocks after a reset, but it was not thoroughly tested before shipping.) The proposed fix is to rising-edge flop CLKRST with CLK128L to keep the internal clocks happy.

The block layout for the fast sections of the chip is planned to be hand placed, as opposed to the tool Flint, to decrease load and improve the floorplan. The availability of the new process also allows us to lower the chip power consumption by scaling the 5V supply for

the clock to 3.3V, and the control logic to 2.2V, but the lower limit for the correlators is still around 1.5V ($V_{tN} + V_{tP}$) to get decent operation.

The key point of the re-design is to analyze the trade-off between the design time and the performance for a given block. We have library cells to synthesize things like control or regular datapaths, however the performance requirement for the correlators is still strict enough in size, speed and power to require hand design. The DQPSK decoder, though, is not fast enough to warrant new cells, but needs to be hand-tiled (using low power library cells) to achieve a reasonable area. To support the multiple voltage supplies, on-chip level converters will be used to allow voltage rings to talk to one another [Chandrakasan94]. Plus, the new process will require new pads which are currently scaled versions of the pads in [Burd94].

The current state of the redesign finds us with several issues still left open. As of the writing of this document, the adjacent cell scan and hand-off circuitry have only been designed on paper. The DQPSK demodulator is nearly complete, with layout existing for the multiplier, but final layout for the backend slicing not yet complete. The correlator redesign is complete and unfortunately the new design suffers significant drawbacks that make it undesirable for actual use relative to the first design. In addition to finishing up the blocks, the overall chip will need to be built, including the aforementioned bug fixes and changes to allow for observation of correlation values for a post-chip RAKE receiving. There is still a fair amount of work necessary to realize the final, fully working version of the digital backend chip.

3 Process Characterization

The design of the digital backend chip migrated over several processes. For the first chip, the design initially targeted a 1.2μ CMOS process (through MOSIS), then moved to a 0.8μ process (MOSIS/IBM which was cancelled) before finally being fabricated in a 'pseudo' 0.8μ process (MOSIS/HP, actually the drawn width is 1.0μ, but the NMOS transistors are mask-biased to produce a 0.8μ channel). Since all of these processes were offered through MOSIS, they used the same design rules (SCMOS) allowing the same library cells to be used. However, these processes had different intrinsic delays, capacitive loading, etc. which impacted the hand-designed correlator at the architectural and circuit level. The eventual chip revision to complete the desired functionality for the radio will be fabricated in yet another different process (perhaps the 0.6μ MOSIS/HP) so characterization is still an issue.

A digital circuit designer requires some understanding of the constantly changing process parameters with which he/she is working: the empirical delay, capacitance values, and current measurements for a logic gate. A parametrized SPICE file was written to characterize a process (for a given Vdd) on the basis of delay and capacitive load versus transistor width (in lambda from widths of 6λ to 120λ with more resolution around smaller widths). A common digital metric is that of the ring-oscillator and a fairly simple SPICE file can supply some useful, rule-of-thumb approximations for delay, edge rates, and node and gate capacitances as they scale with width. Using the same SPICE file, with the appropriate models for whichever process I was characterizing, I could get some first order estimates for a given voltage and temperature that can be used for hand-analysis of circuit or architecture evaluations. Also, if there are multiple model files, this SPICE deck can be used to compare and contrast the models within a given process. For example, I discovered

that the level 13 model for the 0.6μ process was slower than the level 39 models because level 13 estimated more capacitance (as opposed to less current drive). Upon inquiry with a process engineer at MOSIS, I was told that the level 13 models were extracted from a ring oscillator with extra, inadvertent physical capacitance and that I should not use them.

As the above example illustrates the most important and first thing to do, when confronted with a new (to you) process, is to obtain the most accurate SPICE models that you can. (Hassle the process guys!) Don't accept level 2 models; demand empirically based models that are characterized over varying transistor lengths and widths, and even temperatures and voltage. And, once you get them, **don't necessarily trust them.**

3.1. Process Characterization With A Ring Oscillator

To come up with some concrete numbers for a given process we need to identify what types of measurements are interesting from a design perspective. A fairly simple model that is also accurate to a first-order involves the use of inverters for estimation of delay and load. By measuring the characteristics of an inverter as a function of size, we can extend the results to more complex gates by using an empirical 'fudge-factor' that seems to be consistent for standard CMOS processes. For example, we might expect a static CMOS NAND or NOR gate to be roughly twice (fudge-factor) the delay of an inverter since it has two stacked devices which is similar to a single device with twice the length, which halves the available charging current, doubling the delay. Also, by counting the number and size of transistors inside a gate that the input connects to, the capacitive load it can be estimated. This may seem kind of silly; to demand accurate models only to settle for approximate (within 10%) characterization data. However, this information is intended to be for high-level use; critical path simulation demands accurate models. There is always the question of how good/accurate things need to be, and the answer is usually "Good enough to work." These measurements help to give an intuitive feeling for what types of load and delays should be expected for various sizes of circuits and also dictate the upper limit of speed for a full swing signal (in practice nothing on the chip can exceed the maximum ring oscillator frequency). If you are optimizing for speed, these results also can be

used as an ideal design goal to compare against (i.e. the number of inverter delays between registers).

For the purposes of SPICE simulation each transistor needs to be modeled with its diffusion capacitance (parametrized as a function of gate width). To achieve this each transistor was treated as a parametrized subcircuit consisting of a single transistor with parametrized length and width where the area and perimeter of the source and drain are automatically calculated according to SCMOS design rules for a single, separate transistor (shown below). Note that the body is always connected to the appropriate supply, the gate

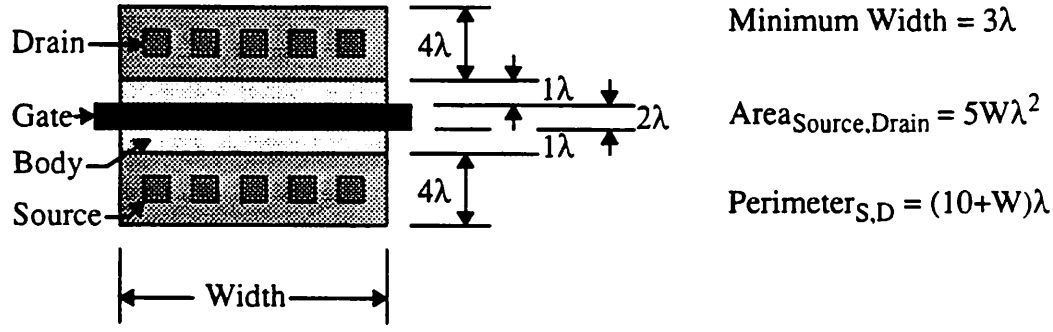


Figure 3-1. Parametrized Transistor for Ring Oscillator

is always assumed to be of minimum length, and that the source and drain are assumed to be the same size and shape.

The following circuits are used by the ring oscillator SPICE deck There are some

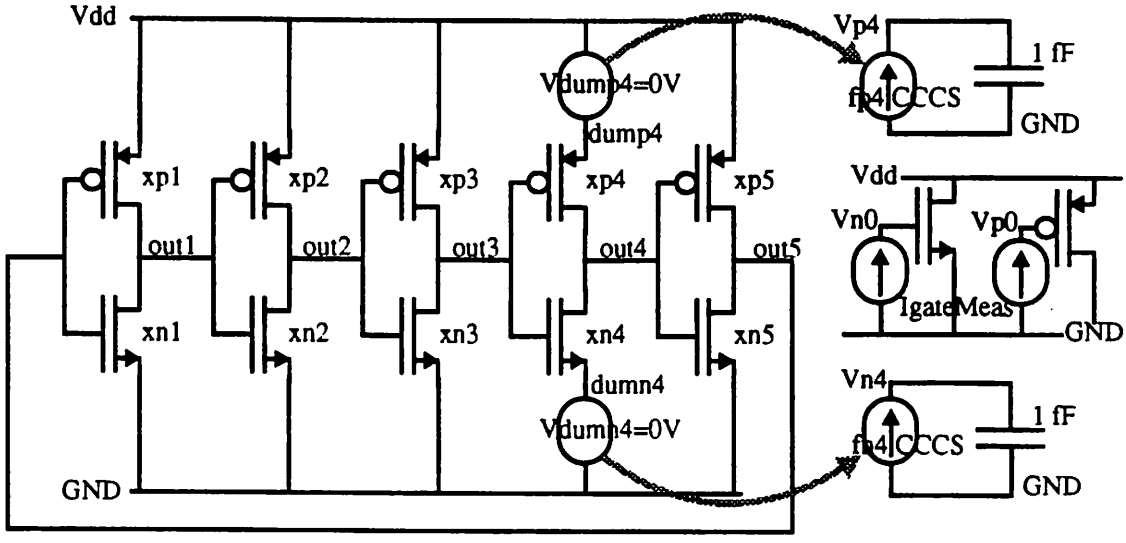


Figure 3-2. Ring Oscillator SPICE Deck

issues associated with the choice of the size and number of stages and these will be treated later in this chapter. The SPICE deck is run for a given Vdd and process. First we will examine measurements taken from these circuits.

3.1.1. Gate Capacitance Measurement

This measurement is performed for both NMOS and PMOS, although the gate cap (t_{ox}/ε_{ox}) is expected to be the same for both. (It's a good sanity check on the models anyway.) Initially nodes Vn0 and Vp0 are set to zero volts. As the transient simulation runs the nodes charge up in voltage until, if IgateMeas is properly chosen, they near Vdd at the end of the transient simulation. It's not important to be exactly Vdd at the end of the simulation, but I wanted them to finish close to Vdd to better approximate the capacitance as the amount of charge needed to provide a ΔV of Vdd. It's not desirable to estimate capacitance for positive voltages greater than Vdd or negative voltages as the MOSFET goes into capacitive modes that will not normally be seen in digital circuit operation. In the SPICE file IgateMeas is estimated

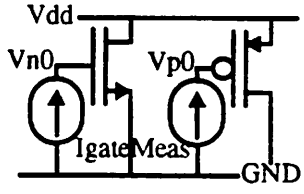


Figure 3-3. Gate Cap Measurement Circuit

based upon the expected cap t_{ox}/ε_{ox}, to make it nicely hit around Vdd at T_1 = (pTran - 2ns), but any value IgateMeas could be used. (Another approach would be to use a larger value for IgateMeas and measure when the voltage was Vdd.) The measurement is taken from noting that the estimated gate capacitance value is simply the two point average. Note that it should increase linearly with W.

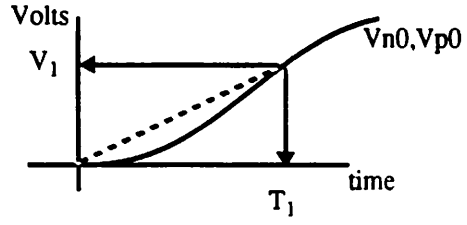


Figure 3-4. Gate Cap Estimation

Given that I is constant and known:

$$\left(I = C\left(\frac{dV}{dt}\right)\right) \approx C\left(\frac{\Delta V}{\Delta t}\right) = C\frac{V_1}{T_1}$$

Then we can approximate C_{gate} as:

Equation 3-1.

$$C_{gate} \approx \frac{IT_1}{V_1}$$

3.1.2. Node Capacitance Measurement

This measurement is similar to the gate cap measurement above except that the value of I is not a known, constant value. If we use a current controlled current source to replicate the switching current from a dummy voltage source into a known value of capacitance we can estimate the value of the nonlinear node capacitance with a simple ratio. For each transition on node out4 we expect a ΔV of $\Delta Q/C_0$ at Vp4 (for the low->high) and at Vn4 (for the high->low), where ΔQ is the charge flowing through the dummy voltage source during a transition. Just looking at the low->high for the moment.

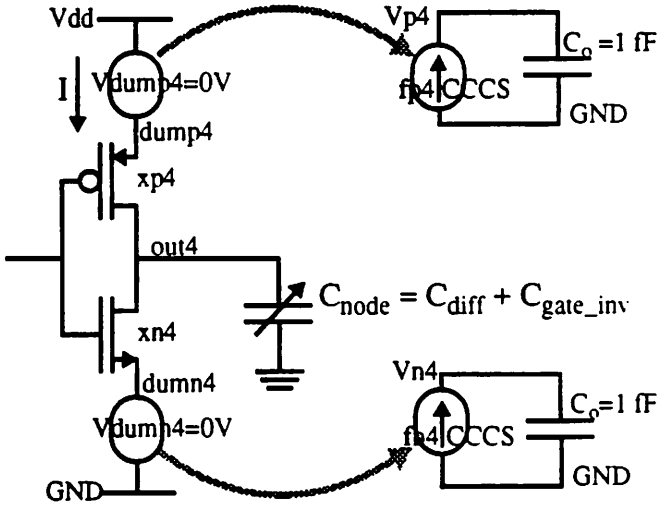


Figure 3-5. Node Cap Measurement Circuit

we would expect to see a trace like shown below. Now, if we assume that all of the current

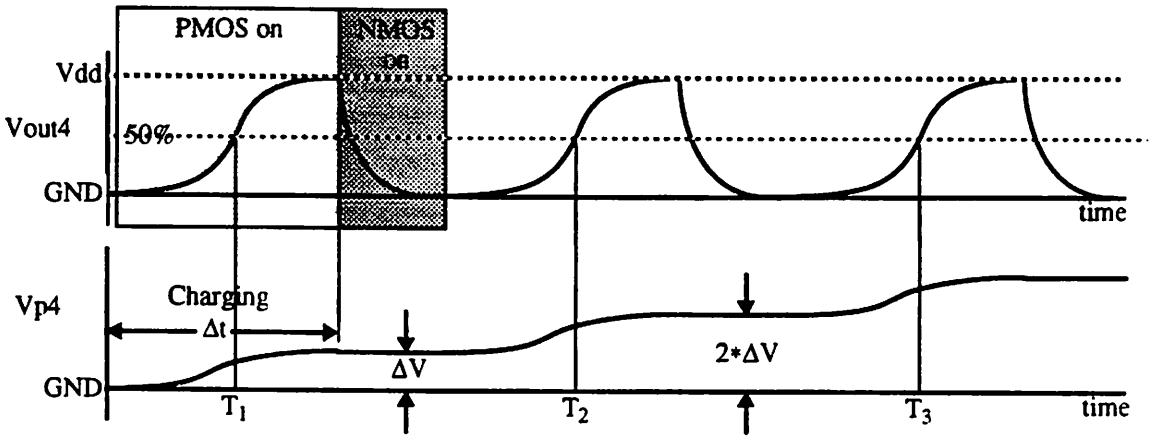


Figure 3-6. Node Capacitance Measurement Trace

flowing through the PMOS (xp4) is charging the node capacitance (ignoring short circuit current), then we can determine the average value of that node capacitance from the ΔV seen on Vp4. (The ΔV can be measured by finding V(Vp4) after the charging period is over, $(T_1+T_2)/2$ for example.) Once we know ΔV, we can derive Cnode as follows

$$I = C\frac{dV}{dt} \qquad \int_{\Delta t} I\,dt = \int_{\Delta V} C\,dV = Q$$

For the known cap, $$Q = C_o\Delta V = \int_{\Delta t} I\,dt$$

For C_{node}, $$Q = \int_{\Delta V} C_{node}\,dV = \left(\left(\int_{\Delta V} C_{node}\,dV\right)\frac{\Delta V}{\Delta V}\right) = \overline{C_{node}}Vdd$$

where ΔV = Vdd

Equating charge $$C_o\Delta V = \overline{C_{node}}Vdd$$

Equation 3-1. $$C_{node\ estimate} = \frac{\Delta V}{Vdd}C_o = \frac{\Delta V}{Vdd}\text{(in fF, since Co=1fF)}$$

This cap measurement is estimated twice, like for C_{gate} above, with one estimate derived from the current in the PMOS, the other from current in the NMOS. From the conservation of energy we would expect these values to be in close agreement unless we are not swinging the same voltage on high->low as low->high. Note that the estimate for C_{node} consists of two C_{gates} from the next inverter stage plus the diffusion capacitance for the driving inverter.

3.1.3. Energy Per Transition Measurements

The SPICE ring oscillator file also keeps track of the amount of energy per transition and reports that number. It is easily calculated from knowing $Q_{transistion}$ ($\Delta V*C_0$) above as $E_{transition} = Vdd*I_{avg}*\Delta t = Vdd*Q_{transition} = C_0*\Delta V*Vdd$. This information was intended to help estimate power, but it wound up not really being used as counting inverter transitions for power is an unusual approach. Power is simply $C*Vdd^2$, where C can be estimated from our above measurements.

3.1.4. Propagation Delay and Edge Rate Measurements

These measurements are being taken as expected. Around the fourth cycle of the ring oscillator the delays T_{dLH}, T_{dHL} and the edge rates T_{rise}, T_{fall}, are measured from the 50% fall to 50% rise, 50% rise to 50% fall, 10% rise to 90% rise, and 10% fall to 90% respectively. Four oscillation cycles are allowed prior to taking the measurement to let the initial start-up transient to die down. (Recall that, due to the odd number of inverters, one inverter is initialized with both the input and output high. Note that for some models and timesteps SPICE will crash if initialized in this way. A work-around is to initialize the first couple inverters to intermediate values as if they were already transitioning.) In addition to the normal delay and edge rate measurements, the overall period of the ring oscillator,

T_{ring}, is measured to calculate the propagation time (T_p) as $T_{ring}/10$ (5 stages x 2 trips). Note that T_p should be approximately $(T_{dLH}+T_{dHL})/2$.

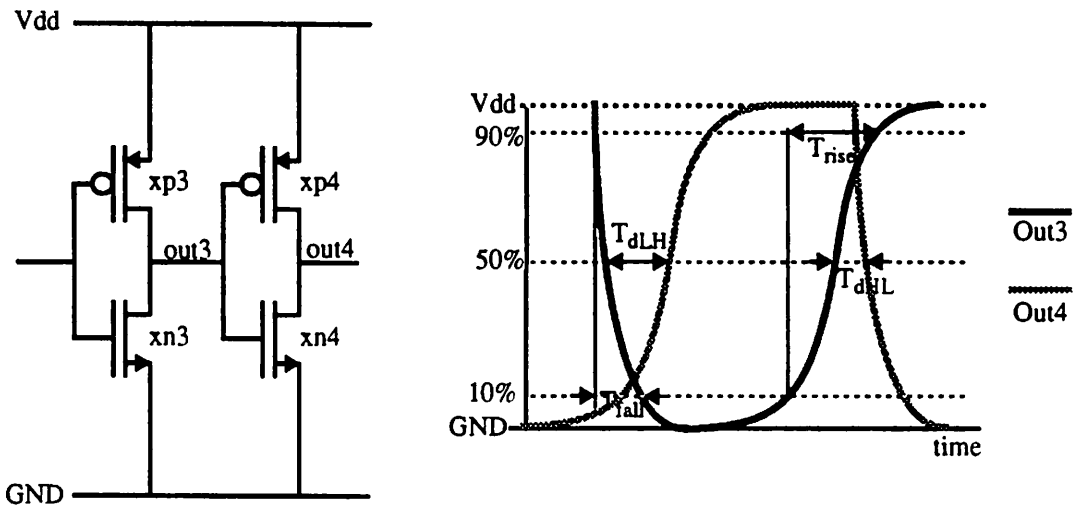


Figure 3-7. Delay and Edge Rate Measurements

3.2. Possible Issues With This Characterization Approach

While this technique is useful to help gain intuition and make approximations while designing, it is not necessarily the only or 'correct' way. Another approach to measure delay is to apply a ramp input transition through a couple inverters for pulse smoothing and then into a load inverter. Also, capacitance could be measured in a number of other ways; as an RC time constant for example. Where appropriate I have listed possible objections, caveats, and other approaches below.

1. The ring oscillator is only 5 stages. In general more stages give a more accurate measurement of the ring oscillation frequency (since that averages the delay over more transitions), however more stages means a longer simulation time (and more memory usage). Since we are only interested in an estimate good to within 10% of 'real-life', a 5 stage ring oscillator is adequate to provide the phase shift necessary without having the edge come around and change the transition before it settles down. There is a minimum number of inverters necessary, but the actual number of stages (greater than 5 in general) is arbitrary. A common number used by MOSIS is 31 and some people count the number of inverter delays in their critical path and do (2x + 1) for their ring oscillator (to get the ring oscillation time around the critical path delay).

2. There is a relaxation time associated with the ring oscillator. Since it is initialized to a state that doesn't normally exist: 1 0 1 0 1, it takes a couple cycles to settle down into its steady state oscillation. Again, this creates a slight difference in delays if they are measured too soon, but it was observed to be a small effect and is noted in case there is a concern. Also, as mentioned before, SPICE may crash if set up in this condition. A work-around is to initialize to 0.7 0.3 1 0 1 or some other values where the inverters are already 'in transition'.

3. The diffusion capacitance measurement is estimated by subtracting the gate capacitance from the node capacitance for an inverter gate. The diffusion cap for a single transistor is estimated as 1/2 the node minus the gate. In actuality since the PMOS diffusion has a different doping density from the NMOS, the value of the capacitance (from the parasitic reverse biased diode) for NMOS and PMOS may be different. However, typically they are comparable to each other. The diffusion cap may be measured in a manner similar to the gate cap by including a couple extra transistors, but this estimate matches to 15% or better in general, and provides a consistent capacitance model for C_{node} as $C_{gate} + C_{diffusion}$. It is also not uncommon to see the gate cap used as a rough value for the diffusion cap estimate if it is not known but that approximation can be off by 2x or more and probably should be avoided except for rough high-level evaluations.

4. The gate capacitance measurement is intended to model the input capacitance seen when switching from 0 to Vdd. This can be done in a number of ways such as RC delays (attach a known resistor, set Vgate to a voltage and find the time constant for the resulting time domain trace), current source input (as described above), etc. They tend to give comparable values, though. Of possibly more importance is that the drain voltage of the transistor does not move as it is attached to a supply. That is not really an accurate modeling of what the transistor is actually doing during operation. To better model circuit operation one could attach a cap to the drain with an initialized voltage (of Vdd or Vdd/2) and watch the gate cap change as the transistor charges or discharges the cap. The drain voltage will change and hence the transistor will move through a different path of operational regions. A comparison of the gate cap of this

'switching' transistor to a nonswitching one (one where the device is effectively off) may be desirable to see how large the resulting difference in estimation is. Overall this amounts to comparing the gate cap as seen by a transistor in cutoff to saturation (my method), cutoff-saturation-linear-cutoff ('switching method with Vdd initialized), cut-off (nonswitching method), etc. The difference is that an all-saturation measurement is about 2/3 the all-cutoff [Muller, Kamins]. The difference is not too large with my method vs. the 'switching' method for 1.5V, 0.7V Vt operation. It tends to average the cutoff cap with the saturation cap as (Vt+(Vdd-Vt)*2/3)/Vdd = 0.82 which is similar to the 'switching' estimate. There is a larger error for higher Vdd's and lower Vt's but it is ultimately bounded to 0.67. While this method may not model a switching gate's input as accurately as possible, it is expedient and accurate enough.

5. When sizing the inverters W_N was chosen to be the same size as W_P to provide consistency in load driving per unit width. Normally it is desirable to size an inverter to provide equal rise and fall times. However, for the purpose of characterization I was interested in how much drive is available from a device as a function of gate width. The simple, first-order hand model for a digital gate is that the PMOS will turn on (NMOS off) to charge and the NMOS turn on (PMOS off) to discharge. For this type of simplification I wanted to know what sort of T_{dLH} I could get from a PMOS versus NMOS for a given W. This way the width is conceptually consistent between devices. The PMOS ultimately cannot operate as fast as the NMOS no matter how large they are made, and my T_p estimate for an inverter with doubled W's for PMOS (for example) will be conservative, slightly overestimated as there is less NMOS diffusion and gate cap. A properly sized inverter will be faster than predicted but the effect is less pronounced as the inverters are sized up since the PMOS dominates the performance. I don't anticipate that the disparity will be larger than the margin for error designers use anyway when trading off such issues, however, it is trivial to change the SPICE parameters to have non-equal sized MOS to re-run and check.

6. What about short circuit current? Most of the design that I was interested in was at low voltages where $V_{tN}+V_{tP}$ was about Vdd, so short circuit current is non-existent or negligible. For higher voltages $I_{shortcircuit}$ winds up being around 10% of the power for a

switch assuming the slopes are being managed [Rabaey241]. Although it is not necessarily a large component, it should be acknowledged that it affects the power and delay measurements. With the proposed simple model of transistors in a digital gate as switches charging capacitance, the presence of short circuit current results in an over-estimation of node capacitance as not all of the drain current is actually going to charge the node. This is consistent from a power perspective as the node cap estimate folds the extra short circuit power into an over-estimate of C. The power is accurately modeled as the rate of energy per transition (which is how much charge moves from the supply*Vdd*f, whether it is to the node capacitance or not). In terms of delay, there is no inconsistency either: T_p still empirically measures how much time it takes to charge or discharge a node. But as the effect is usually 10% or much less, the main issue, if there is one, will be the overestimate of node capacitance which may translate to an overestimate of diffusion capacitance. Just be mindful that it is being ignored for now.

7. Some people feel that this method is fundamentally flawed; ring oscillators aren't good models for complex gate performance. Since most gates, if not all, have stacked devices with internal nodes and multiple fan-in and fan-out, how good a model can an inverter chain be? Certainly logic cells should be simulated on their own with appropriate loading to get more accurate results, but an $\alpha*T_p$ estimate, while rough, is not without its uses for high-level delay estimation. As mentioned above, for simple gates like NAND's, α is often taken to be 2. Gates with low drive ability like NOR's may be 3 and more complicated gates, i.e. an XOR, may be 4 or higher. These α 'fudge-factors' seem to be consistent across processes for CMOS designs and they compare favorably with simulated library cell delays.

3.3. Process Characterization Results

The SPICE file is coded to run a ring-oscillator characterization for the following widths (in λ): 3, 6, 9, 12, 15, 18, 21, 24, 30, 40, 60, 80, and 120. Generally speed asymptotes around 40λ and in practice larger single transistors are not made (larger widths are broken up into parallel, smaller devices). The transient timestep was generally set to 100ps and the number of simulation points kept below 1000 points to prevent grotesquely large

or long runs. (Note that this has the effect of creating 'bumps' in the delay curves, as our delay accuracy cannot be greater than our timestep granularity of 100ps.)

After SPICE is run, a simple shell script, ringpost.csh, is called to post-process the output into a matlab file to allow for easy graphing and manipulation. All of the measurements have "II_" prepended to allow for a simple grep'ing and I/O redirection to create the matlab file. (The matlab file is not in any special format, it simply lists: "variable = [value1, value2, etc.]") Once in matlab it is easy to compare processes, models, voltages, etc. Only the results for the relevant processes are included here: HP pseudo 0.8μ (drawn 1.0μ) for the first version of the backend chip, the revision in HP 0.7μ (a 0.6μ process, but with λ chosen to be 0.35μ to allow for SCMOS design rules), and HP 1.2μ which was used for specifying the initial system design.

3.3.1. HP Pseudo 0.8μ Process

There were three SPICE models available for this process, a level 3, 4, and 39. The level 39 was purported to be the most accurate and was the final authority, primarily since the level 3 and 4 models are rather simplistic (For an explanation of the differences between the levels, consult [HSPICE].) But as the graph below shows, there was a disparity

for delay prediction between the models.While the level 3 and 4 models tend to agree with

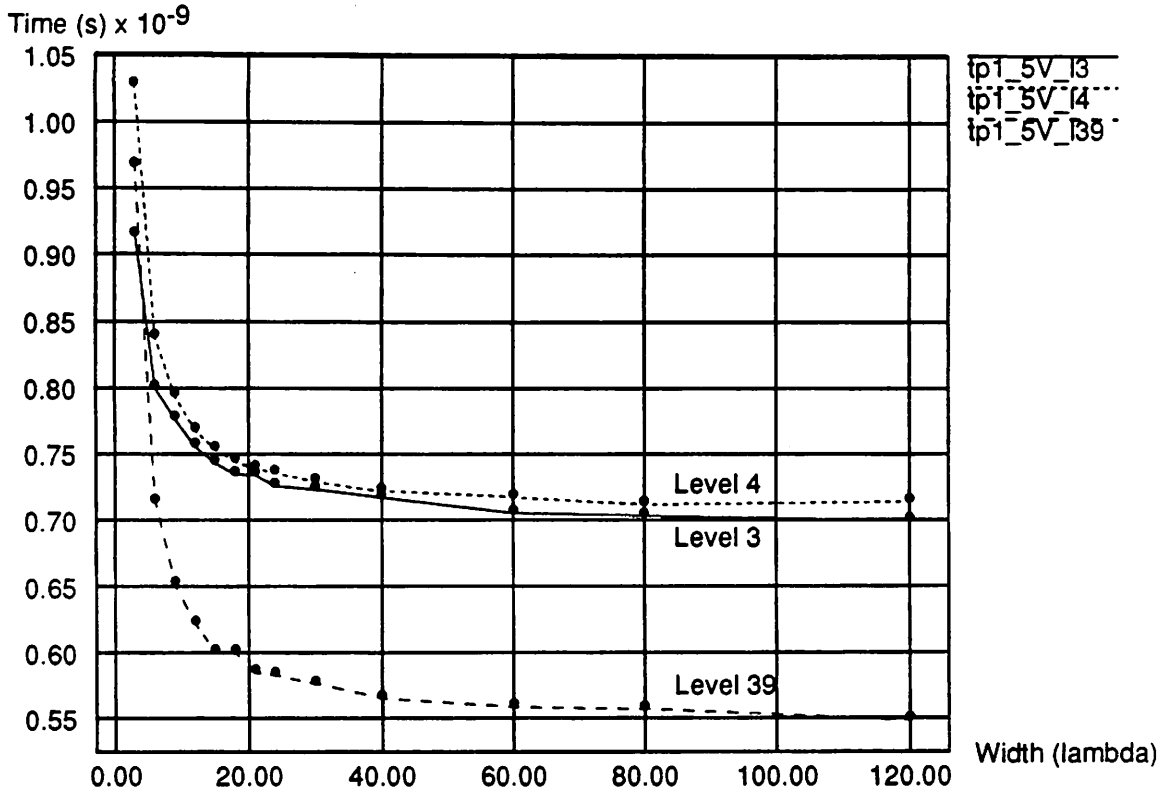


Figure 3-8. Propagation Delay from Various Pseudo 0.8μ Models, 1.5V

one another about delay, the level 39 model predicts a substantially lower propagation delay on the order of 30% lower. This may occur as the level 3 and 4 models are simpler, attempting to curve fit at a higher voltage with a smaller number of parameters, so as you move away from the region where it was characterized you get conservative (slower) estimates. The level 39 is purportedly HP's internal model and hence should be given more

credence. The delays and edge rates predicted by the level 39 model at 1.5V are shown below. Note that at 3.3V T_p tends to be about 4.3x faster and the edge rates about 3x faster.

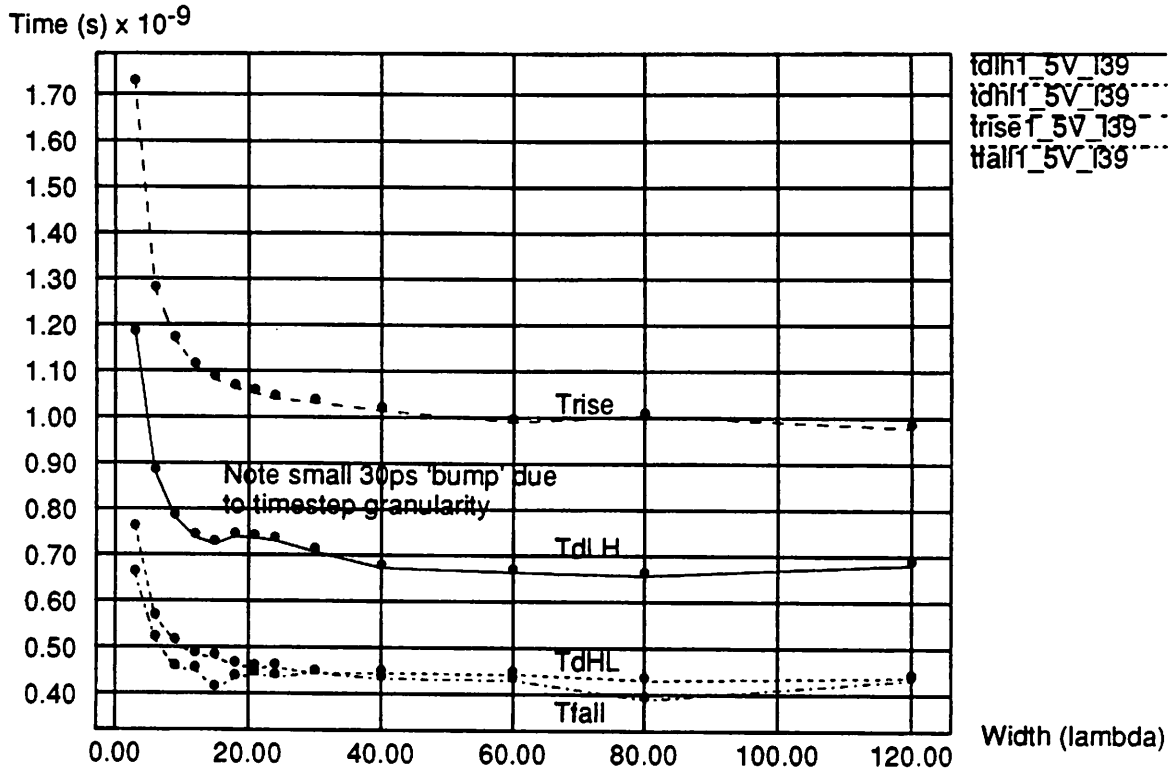


Figure 3-9. Delay and Edge Rates from Level 39 Pseudo 0.8µ Model, 1.5V

The capacitance estimates increase linearly with Width and are listed below in the following tables.Note that the node capacitance estimates for NMOS and PMOS are con-

Node Capacitance	Level 3	Level 4	Level39
NMOS Estimate	5.40 fF/λ	4.48 fF/λ	3.43 fF/λ
PMOS Estimate	5.17 fF/λ	4.65 fF/λ	3.54 fF/λ

Table 3-1. Node Capacitance Estimates: Pseudo 0.8µ Models

sistent within a model, but rather widely vary between models. The extra capacitance estimated by level's 3 and 4 may account for the larger propagation delay. Also since the estimates are not equal, that implies level 4 estimates more drive than level 3 since the propagation delays are approximately equal. The gate capacitance estimates follow a similar pattern, but note the disparity in the NMOS vs. PMOS estimate for gate cap in the level 39 model shown below. This is not an error, rather this process is a pseudo 0.8µ. actually

Gate Capacitance	Level 3	Level 4	Level 39
NMOS Estimate	0.91 fF/λ	0.75 fF/λ	0.46 fF/λ
PMOS Estimate	0.95 fF/λ	0.73 fF/λ	0.65 fF/λ

Table 3-2. Gate Capacitance Estimates: Pseudo 0.8μ Models

dimensions are extracted at 1.0μ and the NMOS transistors are mask biased to 0.8μ. The model takes care of the shrinkage for NMOS (you supply the extracted λ=0.5μ length to the model), so the estimated gate cap winds up being 80% (0.8μ/1.0μ) of that for the PMOS. Note that the capacitance results extracted at 3.3V are a little larger, but approximately the same, so they aren't mentioned here.

From these numbers we can approximate the diffusion cap as 0.5(Cnode-2*Cgate).

Diffusion Capacitance	Level 3	Level 4	Level 39
Estimate	1.72 fF/λ	1.54 fF/λ	1.19 fF/λ

Table 3-3. Diffusion Capacitance Estimates: Pseudo 0.8μ Models

Note that the value is estimated at about twice the gate cap. This makes this approximation of $C_{diffusion} = C_{gate}$ to be rather rough indeed for this process. As an estimate for a higher level evaluation it would make more sense to treat $C_{diffusion}$ as $2*C_{gate}$ when counting capacitance at a node.

From the node capacitance values we can derive the following estimates for energy used per transition.

Energy per Transition	Level 3	Level 4	Level39
NMOS Estimate	12.1 fJ/λ	10.5 fJ/λ	7.7 fJ/λ
PMOS Estimate	11.6 fJ/λ	10.1 fJ/λ	8.0 fJ/λ

Table 3-4. Energy per Transition Estimates: Pseudo 0.8μ Models, 1.5V

3.3.2. HP 0.6μ Process (0.7μ Extracted for SCMOS Design Rules)

There are two models available for this process: a level 13 and 39. However, as mentioned before, the level 13 model is inaccurate as it overestimates capacitive loading.

The characterization results for the level 39 model follow. Note that the estimate for T_p at

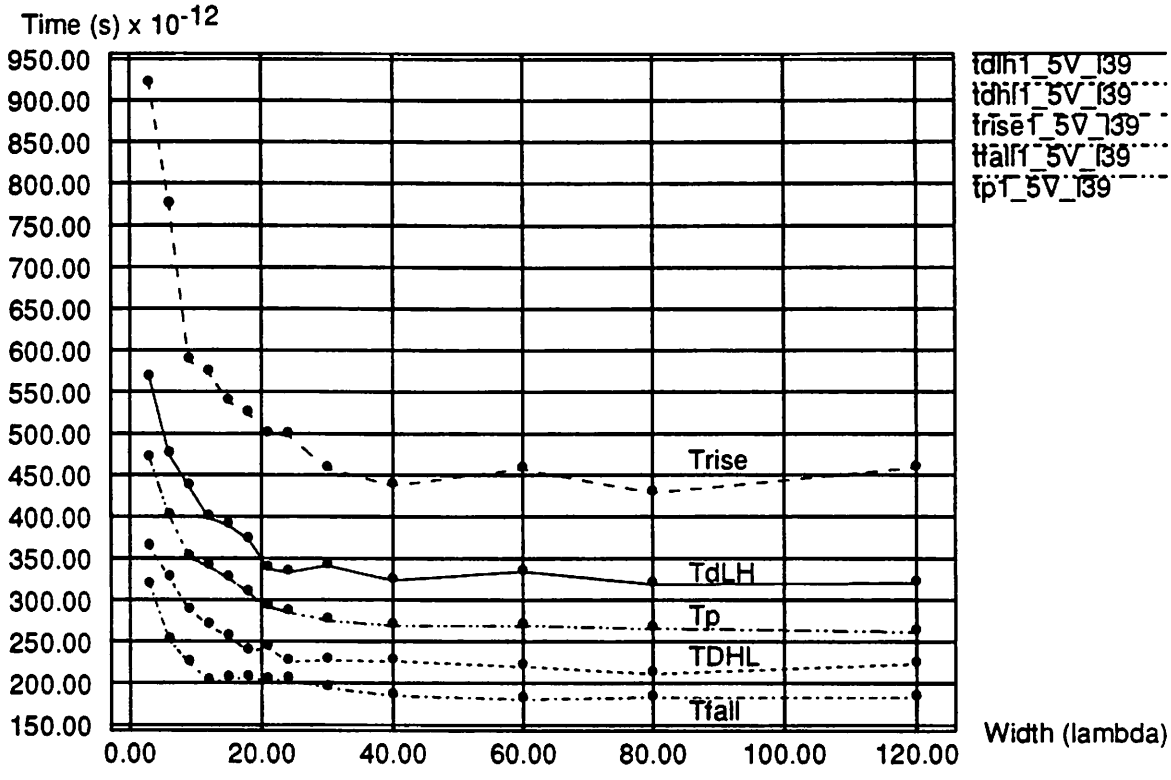


Figure 3-10. Delay and Edge Rates from Level 39 0.7µ Model, 1.5V

3.3V is 3.5x faster and the edge rates are about 2.5x faster than these numbers.

.The capacitance estimates are given below:...

Node Capacitance	Level39
NMOS Estimate	2.35 fF/λ
PMOS Estimate	2.21 fF/λ

Table 3-5. Node Capacitance Estimates: 0.7µ Models

Gate Capacitance	Level39
NMOS Estimate	0.42 fF/λ
PMOS Estimate	0.40 fF/λ

Table 3-6. Gate Capacitance Estimates: 0.7µ Models

Diffusion Capacitance	Level39
Estimate	0.70 fF/λ

Table 3-7. Diffusion Capacitance Estimates: 0.7µ Models

Again the values for capacitance estimation at 3.3V are slightly larger, but approximately the same. Note that the $C_{diffusion}$ estimates are still roughly twice the gate cap estimates for this process too.

Energy per Transition	Level39
NMOS Estimate	5.30 fJ/λ
PMOS Estimate	5.00 fJ/λ

Table 3-8. Energy per Transition Estimates: 0.7μ Models, 1.5V

3.3.3. HP 1.2μ Process

This is the process that the original design for the backend chip started out in. It is included as many early decisions about circuitry were based upon its performance. For this process there were only level 3 and 4 SPICE models. The T_p measurements tend to agree to within 10%, so only the level 4 results are shown below:

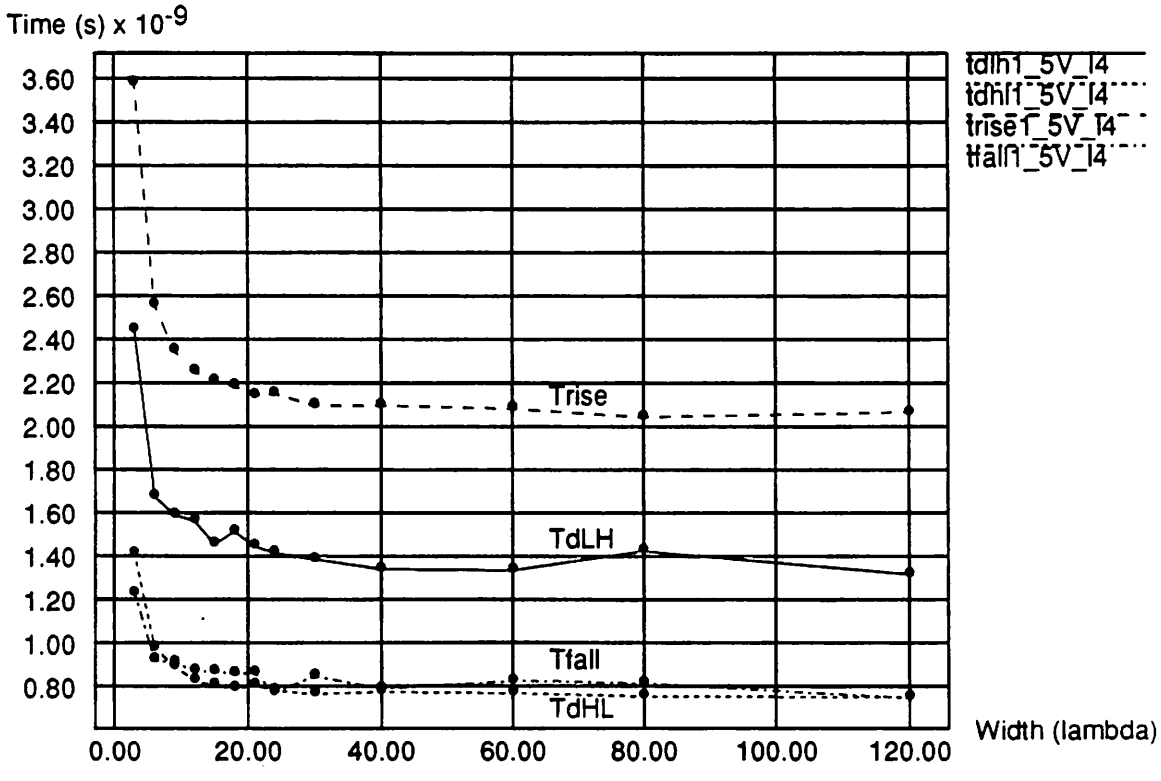


Figure 3-11. Delay and Edge Rates from Level 4 1.2μ Model, 1.5V

The capacitance estimates are given below:

Node Capacitance	Level3	Level4
NMOS Estimate	5.21 fF/λ	4.81 fF/λ
PMOS Estimate	4.95 fF/λ	4.61 fF/λ

Table 3-9. Node Capacitance Estimates: 1.2μ Models

Gate Capacitance	Level39	Level4
NMOS Estimate	0.98 fF/λ	0.92 fF/λ
PMOS Estimate	0.80 fF/λ	0.74 fF/λ

Table 3-10. Gate Capacitance Estimates: 1.2μ Models

Diffusion Capacitance	Level39	Level4
Estimate	1.60 fF/λ	1.53 fF/λ

Table 3-11. Diffusion Capacitance Estimates: 1.2μ Models

Energy per Transition	Level39	Level4
NMOS Estimate	11.7 fJ/λ	10.8 fJ/λ
PMOS Estimate	11.1 fJ/λ	10.4 fJ/λ

Table 3-12. Energy per Transition Estimates: 1.2μ Models, 1.5V

4 The Correlator Design

The correlator is an important functional block for the digital backend chip because it is replicated numerous times in the datapath. At a 64MHz clocking rate, the need for a low power, low area implementation is crucial to maintain the power and area budget in [Sheng91]. Hence a hand-design approach was taken to design the correlator.

The correlator functions basically as an accumulator of N weighted inputs where

Equation 4-1.
$$Y = \sum_{i=1}^{N} W[i] \cdot X[i]$$

X[i] is an input sample and W[i] is the weight. For the radio system the input sample is 4 bits wide, chosen in sign-magnitude format (1 bit sign, 3 bits of magnitude) to lower the power due to number representation [Chandrakasan94]. The weighting function is a 1 bit stream of +/-1's corresponding to a Walsh code overlaid on a PN sequence. N was chosen to be 64 samples corresponding to the symbol period (oversampling ratio) which is adequate for data recovery. However, for channel estimation and lock a longer correlation sequence of 1024 samples is necessary. Due to phase offset in the oscillators there is a rotation in the complex constellation [Sheng96] which precludes simply accumulating 16 symbols (16 x 64 samples) for the longer correlation sequence. In order to remove the rotation a complex multiplication would be necessary which is vastly undesirable in area and power for the correlator. An estimate of the energy for the long correlation may be performed by using the absolute value of Y and accumulating: [Sheng96]

Equation 4-2.
$$Z = \sum_{j=1}^{16} |Y[j]| = \sum_{j=1}^{16} \left| \sum_{i=1}^{64} W[i]X[i] \right|$$

Note that the incoming data is complex and that correlations must be done for the I (in-phase) and Q (quadrature phase) channels which doubles the hardware cost for the accumulation.

4.1. First Correlator Design (for 1.2μ, fabricated in pseudo-0.8μ, 1.0μ)

4.1.1. Architecture Exploration

As the weighting function is simply a sign-toggle, the correlation basically becomes an accumulation. Thus the main element of functionality is the addition/subtraction of 3 bits for 64 samples resulting in 9 bits of magnitude plus 1 bit of sign for dynamic range. (Longer correlations for 1024 samples require an additional 4 bits of magnitude for 13 bits plus sign bit = 14 bits total. These are achieved by taking the absolute value of the 64-sample correlations, and further accumulating another 16 cycles.) A simple idea for implementation is to simply add up all of the incoming data samples using a straightforward 2's complement ripple adder. Since the incoming data is sign-magnitude, it needs to

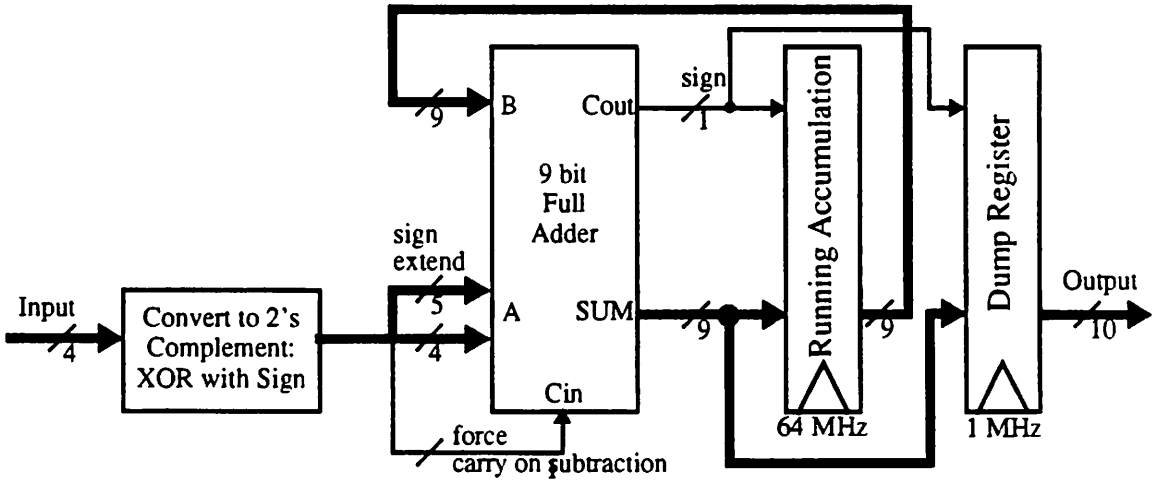


Figure 4-1. Simple Correlator Architecture

be converted to 2's complement for the above approach. This sign-extension causes significant additional power [Chandrakasan94], but it is not the worst aspect. The carry chain for a ripple adder must complete in 15.6ns (64MHz) minus the register setup and delay times. At the beginning of the correlator design we were looking at a 1.2μ process with T_p around 1.1ns which yielded 15.6/(1.1) = 15 inverter delays with a fan-out = 1. Assuming that any two-input gate will have a delay of at best 2 inverter delays, this allows for only 8 gates

between flops which isn't even enough for the carry chain. Looking at the process characterization data for the pseudo 0.8μ technology we see that this allows for 26 inverter delays, or about 13 gates. Allotting at least a gate per carry, there are only 3 gates left for the register and carry setup circuitry. Perhaps the design could be squeezed in with this very small amount of overhead, but it doesn't look promising. To be sure, there are well-known techniques to help speed up this circuit including running at a higher voltage, pipelining, carry look-ahead addition, etc., but due to the tight timing constraints and desire for low power a different approach was examined. Increasing the voltage (above 1.5V) was unacceptable from a power perspective and more complicated adders tended to drastically increase in power and area. The inability to fully ripple implies that a rather deeply pipelined scheme will need to be employed.

An architectural idea to preserve the advantage of the sign-magnitude nature of the incoming data was to break up the accumulation into two parts: an accumulation of all incoming positive data and an accumulation of all incoming negative data. The sign bit can be used to multiplex the 3 bits of magnitude to the appropriate adder and the sum can be computed after dumping at a 1MHz rate by including a subtracter after the dump register. This has the advantage that the final subtracter will take negligible power at 1MHz and has plenty of time to compute, but increases the area for the correlator by a bit. The main problem is still the critical path for the addition which is still operating at 64MHz worst case. Although, as we no longer have to sign-extend, the critical path is the carry from 3 bits of full adder plus 6 bits of half adder (for accumulation). While this is better, it is still a difficult constraint.

Another architectural idea is to cut down the critical path by pipelining the carry chain to allow for slower operation. The degree of pipelining is arguable; more registers ease the carry path design, but increase clock power and area. To examine this trade-off, SPICE results for the TSPC register and static CMOS XOR obtained. Again, originally we were in a 1.2μ process for which the clock-to-Q flop delay was around 3.5 ns ($\cong 3*T_p$) with a 2ns setup time, and the XOR delay around 3.5ns. Using these numbers, just getting out of a latch, going through two XOR's for a full adder, and getting back into a latch took (3.5+7+2) ns = 12.5ns. Since we are working with a 15.6ns period, this implies that the

carry has to be bit-pipelined -- thus a carry save architecture for the adder was necessary, the only viable choice at these speeds. This entails the use of two register banks, one to hold the current sum vector and one to hold the current carry vector. The cost of this replication is extra area for registers and adders to combine the dumped sum and carry vectors into a final result as well as in power to clock about twice as many registers. This is still less power overall than using a higher voltage to accommodate the critical path [Chandrakasan94]. A nice feature of the carry save adder is that it is fairly regular to tile since it is being bit-pipelined, and that it doesn't require a complicated design, the critical path reduces to the time for a single bitslice of a full adder cell. The choice of a carry save architecture will be reexamined for the second version of the correlator where the added speed of the new process will allow us to remove extra registers in the carry path.

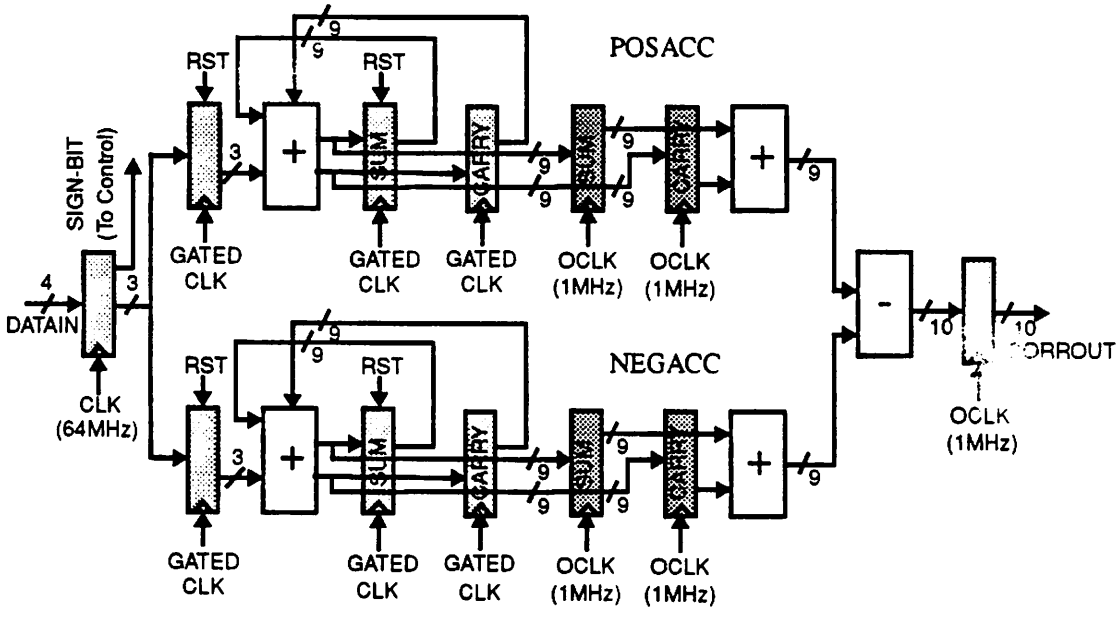


Figure 4-2. Carry-Save, Sign Magnitude Correlator Architecture

[Figure Courtesy of K. Stone [Stone95, pg. 42]

4.1.2. The Carry Save Bit Slice

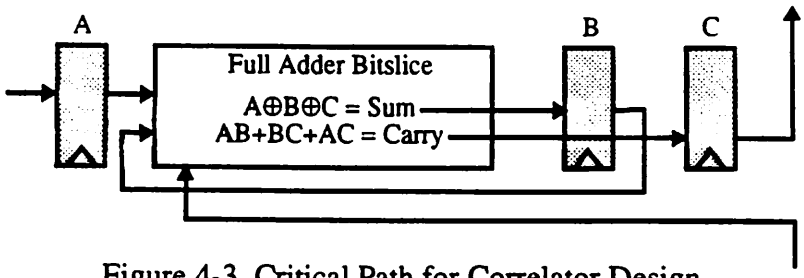


Figure 4-3. Critical Path for Correlator Design

The critical path for a carry save bit-slice, as shown in Figure 4-3 above, is given by $T_{clk2Q}+T_{fulladd}+T_{setup}$. The issues we need to look at now are cell design, tiling of the datapath, and then control and clocking. In an attempt to decrease the design time, existing library cells from the Low Power Library [Burd94] were used if appropriate. Although most of these cells were designed to run slow with a minimum of switched capacitance at 1.5V, some can be sized up to improve performance. The TSPC, or True-Single-Phase-Clocking, style of dynamic flip-flop was chosen for speed reasons and for the ease of only having to run one clock line. In general the static CMOS design style was used for the cells as it is robust, delay scales with Vdd, and it is a well-known design style.

The adder is one of the most studied digital blocks.An overview of the more common designs may be found in the references [Omondi] and [Rabaey]. While there exist many interesting complex, multi-stacked, CMOS implementations, most are intended for 5V or 3.3V operation and hence suffer from performance degradation at low voltages due to the large number of stacked transistors. (The PMOS device rapidly loses drive ability when stacked more than 2 or 3 transistors deep, in addition to providing large capacitance on internal nodes.) Also, something complicated, like look-ahead, bypass, select, etc. is unnecessary since it is only a bitslice between registers. In the interest of simplicity and ease of layout, all we need do is implement the Boolean equations. An XOR will be necessary for the Sum calculation, and the Carry Generation is simple enough to be done as a single complex gate or the cascade of a couple smaller gates. For the XOR the low-power design style of choice is pass transistor logic (with low Vt's); however, with our process the delay through a pass gate implementation was longer than that for a static XOR implementation. As two cascaded XOR's constitute the longer delay, the carry generation logic

could be implemented in a single, small, slower, static gate as opposed to several simple gates. This is also lower power as the internal node capacitance is smaller. [Rabaey241] points out that complex gates can be lower power than simpler implementations in some cases. However, as the XOR delay was the critical path, the SUM was implemented with two cascaded XOR's instead of a complex 3-input XOR gate which is too slow and unwieldy. The gates were implemented as shown below. Note that for the XOR, the inverse of A and B are delayed, and hence placed closer to OUT to improve the speed of the gate.

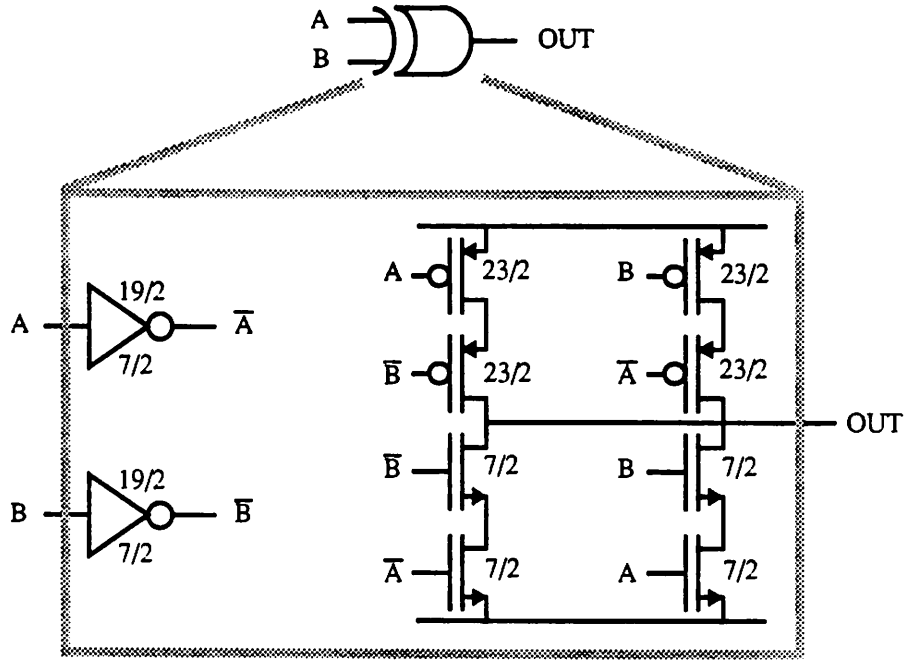


Figure 4-4. XOR Gate Implementation

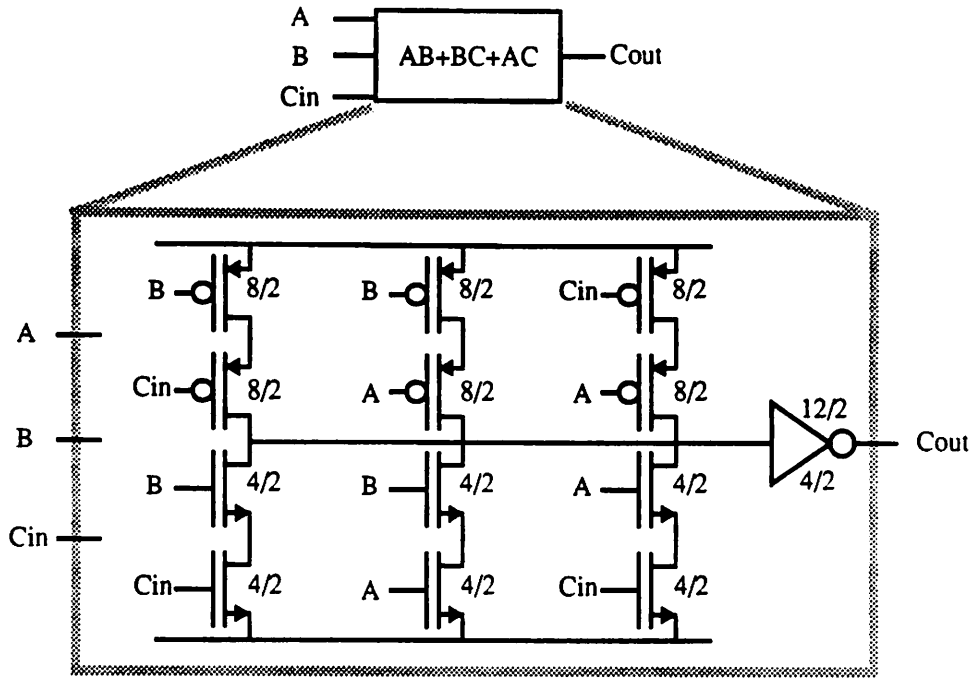


Figure 4-5. Carry Generation Gate Implementation

The TSPC register is of the same design as the library cell, [Burd94] also from [Yuan, Svensson], and is sized as below.

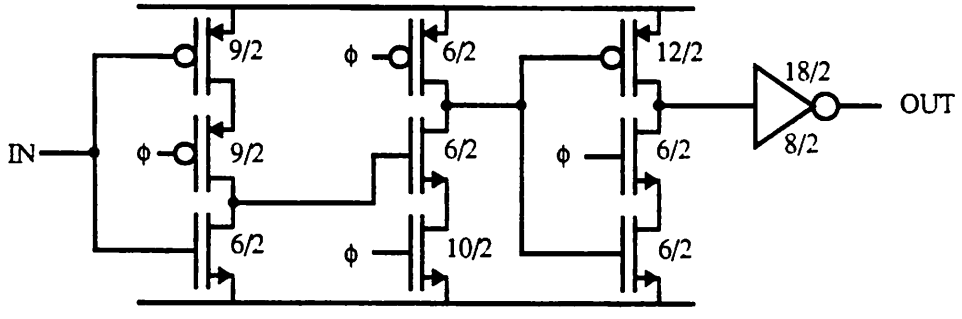


Figure 4-6. TSPC Register Implementation

The sizing for the XOR was chosen by the simple scaling technique commonly used in digital design [Rabaey]. The NMOS are scaled approximately 2x from minimum size as they are stacked two deep. The PMOS are scaled up by 4x from the NMOS to equalize the rise and fall times. As the XOR constitutes the critical path, it was sized up for faster operation, while the carry generation gate was kept near minimum size. The TSPC register

is mostly minimum size except for a slightly sized up frontend stage for quicker set-up, a large evaluation NMOS, and consequently a sized up PMOS on the next stage to speed up the slow path through the gate.

4.1.3. Tiling up the Accumulator and Correlator Datapath

The accumulator is then tiled up similar to the datapath style in [Burd94] to get tight packing, with control and power signals running vertically and data signals running horizontally as shown below. The half-adder cell is used for incrementing each carry out from

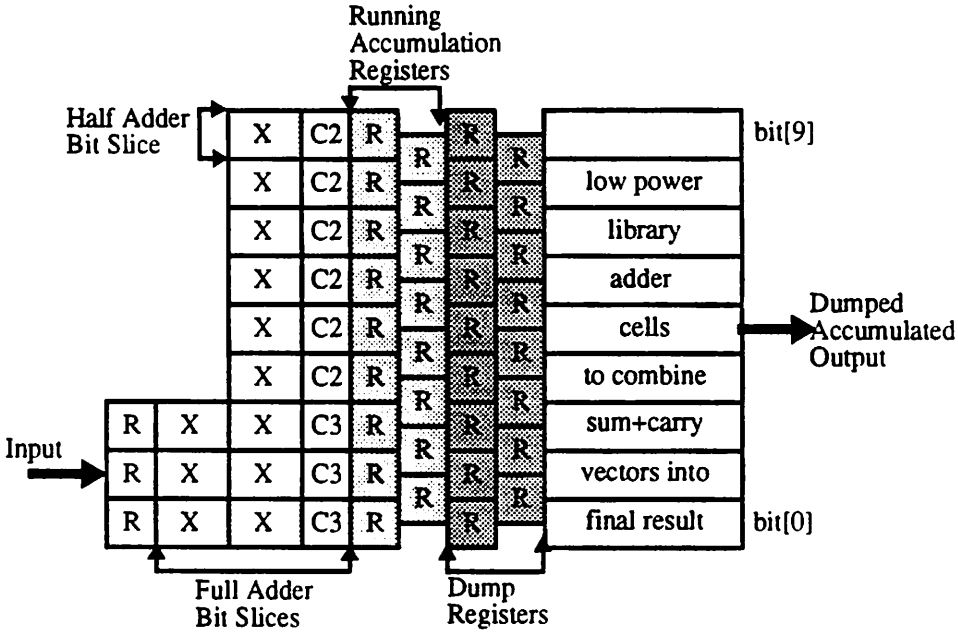


Figure 4-7. Accumulator Layout

the full adder. Also, the cells overlap sharing power and ground with adjacent cell. This change from the specification in [Burd94] was made to achieve the tightest possible layout. In addition the carry registers are shifted halfway up towards the next bitslice to ease routing.

Since there are two accumulators needed (one for positive numbers, and one for negative), a question arises of floorplanning: Should the accumulators be placed on top or one-another or side-by-side? If is often desirable for digital layout to be shaped as 'squarely' as possible, since long, thin blocks can be difficult to layout or route compactly. Since there are two correlators (for I and Q), with two accumulators each, a suggestion is to tile a correlator's accumulator's side-by-side, and tile one correlator on top of the other

in order to get a square-ish layout. It is certainly not the only way to tile up the correlator, but it was compact, and kept the high-speed, incoming data to one side, allowing the lower speed correlated outputs to come out of the other, as in the datapath style.

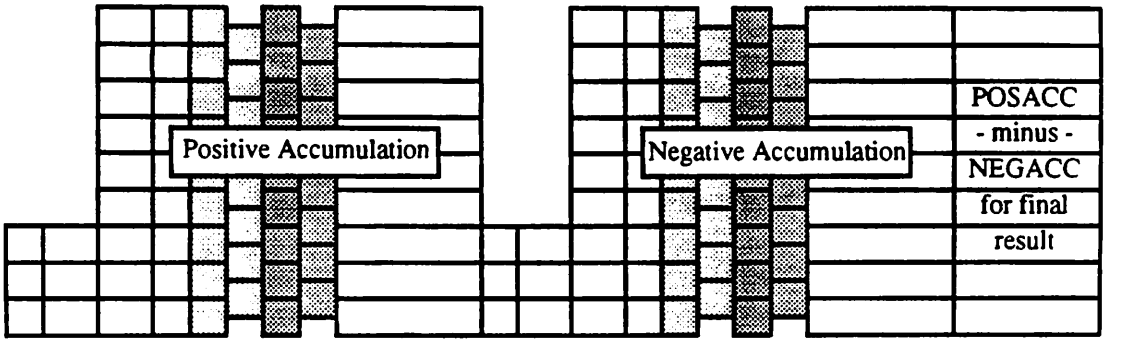


Figure 4-8. Correlator Datapath Layout

4.1.4. Performing the Weight Multiplication

The accumulators discussed thus far deal only with the summation of data samples, we still need to provide the multiply by +/-1 by the PN and Walsh codes. Since this is a trivial multiply it winds up being nothing more than the XOR of the incoming data's sign bit with the PN and Walsh bits. As we know from our experience with the carry save adder in 1.2μ, the clocking period is only able to allow safely two XOR delays between registers. Luckily that is identical to what must happen to perform the sign multiply, so we can use the same cells. It does mean, however, that another two pipeline stages will need to be added to the front of the datapath to give enough time to perform the multiply, then get the result to the control logic to multiplex the data to the proper accumulator. This increases clock power and latency, but is unavoidable.

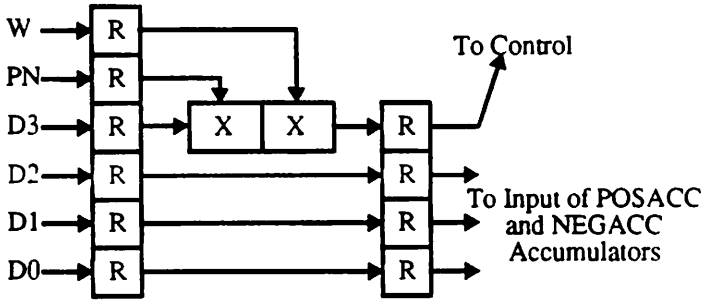


Figure 4-9. PN and Walsh Weight Multiplication

4.1.5. Correlator Control Signals

There is a minimal amount of control that needs to be designed to do the multiplexing between the positive and negative accumulators, and to accommodate a reset. The technique of gated clocks is used, even though some people consider it risky, as it is better for power to not clock sections that aren't needed. Since a reset arrives every 64 samples we don't need to worry about the fact that the registers are dynamic, as they are guaranteed to be refreshed at least at a 1MHz rate. Two control signals are added to the correlator: DUMP (an enable for latching the dump registers and resetting the running accumulation registers), and RESET_DUMP (an enable for resetting both the dump and running accumulation registers). The desired control functionality is (on rising CLOCK):

1. Dump registers take sum and carry vectors on DUMP assertion
2. Dump registers reset on RESET_DUMP assertion
3. POSACC updates running accumulation register for positive data ($\overline{\text{Sign}}$) or DUMP
4. POSACC resets running accumulation registers on DUMP
   (This seems redundant, to update and reset on DUMP, however, the reset in the TSPC registers is an enable, only evaluating to low after a clock edge.)
5. NEGACC updates running accumulation register for negative data (Sign) or DUMP
6. NEGACC resets running accumulation registers on DUMP
7. POSACC input register clocks on $\overline{\text{Sign}}$
8. POSACC input register resets on (DUMP and Sign)
   (Important to not miss the first sample of the next correlation when dumping/resetting)
9. NEGACC input register clocks on Sign
10. NEGACC input register resets on (DUMP and $\overline{\text{Sign}}$)

This is relatively easy to provide, once the Sign of the data is known. After sign bit is known, it is quickly inverted and NOR'ed appropriately to provide the needed control signals before the FALLING edge of the clock. The control signals are clocked in on the FALLING edge to give them a half cycle of the clock to be ready before the datapath clocks on the RISING edge.

Note that the control logic was laid out in the gaps at the front of the accumulators to pack the design into a rectangle. Luckily the cells fit without very much white-space. See Figure 4-14 for the complete control logic of the correlator.

4.1.6. 2's Complement to Sign-Magnitude Conversion

At this point the correlator is nearly all designed and there are only a couple issues that remain. One of them is that, post subtraction (POSACC-NEGACC), the result will be in 2's complement, as opposed to sign-magnitude. For longer correlations we want to see the absolute value which is easily accomplished by simply ignoring the sign bit. For DQPSK decoding we will preform magnitude multiplications and combine (add or subtract) afterwards to simplify the multiplier design. So, in addition to power concerns for sign-magnitude representation, which are minor at 1MHz compared to the faster circuitry on the chip, there are some strong system issues that indicate a sign-magnitude representation will be necessary.

Since the rate is low power and speed will not be much of a concern. The issue becomes one of how to do the conversion in a small amount of area. If the outcome is positive, we don't need to do anything. If the result is negative, we need to subtract 1 and invert to directly convert (or equivalently invert and add 1). A straightforward way to do this method is to run the correlation outcome into a decrementer, or half subtracter (which will subtract 1 from the data if negative, 0 otherwise -- a.k.a. subtract the value: Sign), and then run the output of that into a bank of XOR's which bit-wise invert on Sign. For the XOR and half-subtracter in this case we can use the low power library cells, which use pass tran-

sistors, as they are small with lower switched capacitance. A block diagram looks like the following.

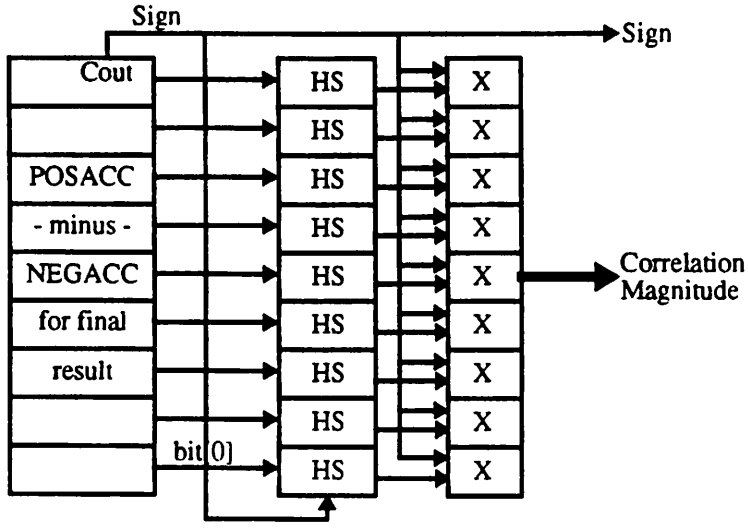


Figure 4-10. 2's Complement to Sign Magnitude Conversion

Since these cells operate slowly, there may be a concern with meeting the timing requirement after a dump: ripple adding sum and carry vectors, ripple subtracting NEGACC from POSACC, and ripple converting from 2's complement to sign-magnitude in 1000ns at 1.5V. The low power library documentation estimates a 9 bit ripple (add or subtract) at about 35ns for a 1.2μ, 0.7V Vt process [Burd94]. For a set of 3 full ripples of 9 bits, this is only ≅100ns, 10% of the 1MHz clock period.

4.1.7. Clock Buffering

As was mentioned above in Chapter 4.1.5 above, the control is achieved by gating the clock for the correlator. This is a relatively simple scheme where the global clock is gated with a NAND, then buffered with an inverter for drive. In clocking the datapath reg-

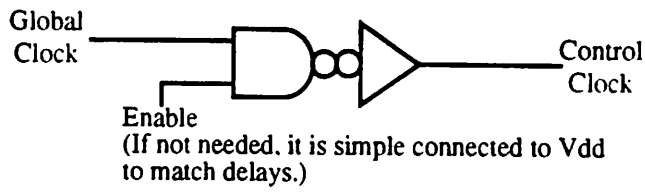


Figure 4-11. Clock Gating with a NAND

isters, the main issue we are concerned about is skew between register banks as this may

either eat into our critical path, or cause incorrect latching. Another smaller issue is that of clock edge, since the TSPC registers are sensitive to low slope rates. The inverter buffer will be sized to give fast enough edges (about 2x Trise = 4ns from the ring oscillator data for 1.2μ). The control should be set up, by clocking on the falling edge, to provide the enable for the NAND at least a couple nanoseconds before the rising edge of the global clock. This was verified with SPICE by simulating the extracted layout of the control section.

A straightforward way to break up the clock load and to ensure little skew is to try to match or balance the capacitive load seen by the inverter buffer. This can attained by grouping the registers into banks of approximately the same size. For example, the sum registers (9 bits) and carry registers (8 bits) can be separate banks. The input has 8 bits (4 data, PN, Walsh, DUMP, RESET_DUMP) and may be a bank also. The only left-out registers are: intermediate control registers (clocked by the falling edge), and the 3-bit input registers to the accumulators. Since the input registers have 1/3 the bits, we can size their driver down to compensate. Likewise, the intermediate registers (two banks of 6 bits) may be driven by an inverter 2/3 the size of the default inverter buffer. See Figure Figure 4-14 for an explicit picture of the frontal input registers, control logic, and clock gates.

Now that we have a rough idea about how to scale the inverter buffers, we need to know what the capacitive clock load of a register is. By counting the gate length (in λ) and using the process characterization info (ignoring parasitic routing cap) we find 38λ of gate cap = (0.9fF/λ*38λ) = 34fF (for 1.2μ). This closely matches the SPICE result of 35.7fF for a 1μA current source driving the clock input for the register. From the load data we can then estimate the size of the driver transistor from: 1) finding the inverter size from the process characterization data to drive a C_{node} of 2x our cap estimate (to account for load from

drain and source cap of driver), and/or 2) use the equations below: (Unfortunately this

For a non-velocity sat. MOSFET: $$I = k_p \frac{W}{L} \Delta V_{GS}^2 = \text{constant}$$

To drive C from 0 to Vdd in Δt $$Q = CVdd = I\Delta t \qquad I = \frac{CVdd}{\Delta t}$$

Equation 4-3. $$\frac{W}{L} = \frac{CVdd}{k_p \Delta V_{GS}^2 \Delta t}$$

needs k_p or some knowledge of drive versus V_{GS}. However, using level 3 or 4 data or graphing I_{DS} vs. V_{GS} can give you that knowledge. Be sure to include in 'C' the estimate of drain and source cap contributed by the inverter buffer. The following table may then be derived:

Clocking Load	Cload Est.	I for 1.5V in 4ns	Power @ 64MHz	NMOS W/L Est.
4 registers	2*(142.8) fF	113 μA	22 μW	11 (22λ/2λ)
8 registers	2*(285.6) fF	215 μA	43 μW	22 (44λ/2λ)
16 registers	2*(571.2) fF	428 μA	85 μW	44 (88λ/2λ)
32 registers	2*(1.142) pF	857 μA	170 μW	88 (176λ/2λ)

Table 4-1. W/L Estimates for Clock Drivers (1.2μ process)

From the process characterization data (1.2μ) we can see that a ring oscillator with a width of 55λ will have a 2ns rise time for a load of 2*(142.8)fF = 286fF. Doing a rough division by 2 (for a 4ns rise time) yields a width estimate of 55/2=27.5λ which is on the order of the 22λ estimate from Equation 4-3 above.

The actual sizing chosen was 26λ/2λ for NMOS, and the PMOS was sized up by roughly 3 times (to save area and power) to 80λ/2λ for driving 9 registers. Although the equations tell us that the PMOS would have to be sized up by 4 to 5 times to match edge rates, in practice this is just too large. The edge rates wound up SPICE'ing at about 4.5ns and T_p for the driver was around 3ns. The simulation was done by two methods. First a bank of NAND's driving an inverter-buffer loaded with the estimated capacitance for the number of registers it drives was simulated. Secondly, after comparing skew between

clocks in the first simulation and arriving at a sizing, the clock lines were extracted from the completed correlator layout and performance was verified. It should be noted that the NAND's were sized up to drive the inverter-buffer based upon the optimal scaling factor (e=2.78) for inverters driving a large load discussed in [Rabaey]. The PMOS in the NAND were 30λ/2λ and the NMOS 12λ/2λ.

The key question is, 'How much skew can we really tolerate?' That answer boils down to two factors depending upon whether it is positive or negative skew. On the one side, if the clock arrives at the end registers sooner than the beginning ones, this eats into your overhead for the critical path. Recall from Section 4.1.1 that we have about 3ns of overhead for the critical path, hopefully much more than we should see in any skew. If the clock to the beginning registers occurs before the end registers you could encounter a race-condition where the new data overtakes the old before it has a change to be latched up. In the design the fast path is a pair of back to back registers with no logic delay between them for the three input magnitude data bits. As the clock to Q delay for a register is on the order of 3.5ns, this implies that the most skew we could tolerate is around 1/2 that (this would barely give the output time to change) which is about 1.7ns. So in practice we have a bit of a safety margin, as long as the skew between any two registers is less than about +/-1.7ns, we should be O.K. Simulation results confirm that the observed skew due to loading was < 1ns.

4.1.8. Power Estimation for the Correlator

In general power estimation is done by running a random set of vectors through the logic and having a program count the amount of switched capacitance. In addition to this method, however, we can also come up with some rough hand-estimates to verify that the simulation results are in the same ballpark. Taking power as having two components of power to the clock and power of the data moving through the logic, we can estimate the overall power by making some back-of-the-envelope assumptions. Since the circuit is bit-pipelined with only a couple gates between registers we can assume that the power of the data will be roughly equal to the power of the clocking as they have roughly equal logic depth. (Although this ignores the switching frequency of the adders which may be less than the registers.) The clock load for an accumulator about 40 registers (at about 40fF each) for

4.1.9. Library Issues: i_frontend, i_basecorr

Once the correlator is all designed and simulated, it was made into a library cell so that, at the next level of design hierarchy, it could be treated like a leafcell. The method for doing that in OCT won't be described in detail here, especially in light of the movement towards Cadence in our design flow. Essentially all of the layout (magic files) and an SDL top level file were grouped into a library directory, and a Makefile is run to create all of the proper OCT facets and views. In addition to OCT, as Viewdraw was being used for the overall chip design, wir, sch, and sym directories needed to be made, along with the proper files. Also, a VHDL files was written that models the behavior of the correlator. Note that is it not intended to be synthesized, it is only intended to be used for system simulations of the chip. The VHDL files are included in Appendix B of [Stone95].

A word needs to be said about the naming convention for the correlations that are going on inside the digital backend. On one hand there is the symbol correlation, 64 samples long, named i_frontend, whose design has been discussed at length in the last 13 pages. In addition there is the longer, channel estimation correlation (1024 samples, 16 magnitudes of symbol correlations), named i_basecorr.

4.1.10. Layout: i_frontend and i_basecorr

The final layout of i_frontend and i_basecorr follow on the next couple pages with the control logic annotated over the layout.

about 1.6pF. If we assume this is charged linearly in 4ns, this implies a current of CV/ Δt=0.6mA. This is occurring at a 64MHz rate, so the power to drive the clock is I*(4/ 15.6)*1.5V=0.23mW. Using our bold assumption that logic power equals clocking power, we can double that power to 0.46mW. Also, as the clock has to drive its own load in addition to the gates of the registers, we may assume that the clock network's load is roughly equal to the gate load, so add another 0.23mW for a total of 0.7mW for a correlator. Since it is a complex data stream, there are two correlators in each complex correlation, yielding 1.4mW for I and Q correlations (for a 1.2μ process). We might expect the pseudo-0.8μ process power to be around 1.0/1.2 (83%) of that number (1.2mW), although W_N is 0.8μm and W_P is 1.0μm.

Using IRSIM-CAP we can count the amount of switched capacitance in response to a random input correlation and come up with a power estimate that has a little more credibility [Landman95]. Although IRSIM is a switch level simulator it has been modified to provide a reasonable power estimate based on transition frequency. For the 1.2μ process, the results indicated a correlator power of 0.8mW which are close to the above back-of-the-envelope estimate. Note, for I and Q the estimate is 1.6mW. In the pseudo-0.8μ process, we expect to see about 0.66mW, 1.32mW for I and Q.

A newer program for power measurement, PowerMill, that purports to be much more accurate, recently became available for use. PowerMill claims to have switch-level speed with SPICE-like accuracy. Running random vectors through that gives a result of 0:81mW per correlator (0.8μ).[Courtesy of Varghese George].

And finally, to compare with an actual measurement, a single correlator was measured to have a power of 0.6mW at 32MHz, implying a 64MHz power of 1.2mW (2.4mW for a complex correlation). This value is around 50% larger than predicted but of the correct order of magnitude. The error is due to several on-chip level-converters and assorted buffer circuitry running on the 1.5V rail of the chip.

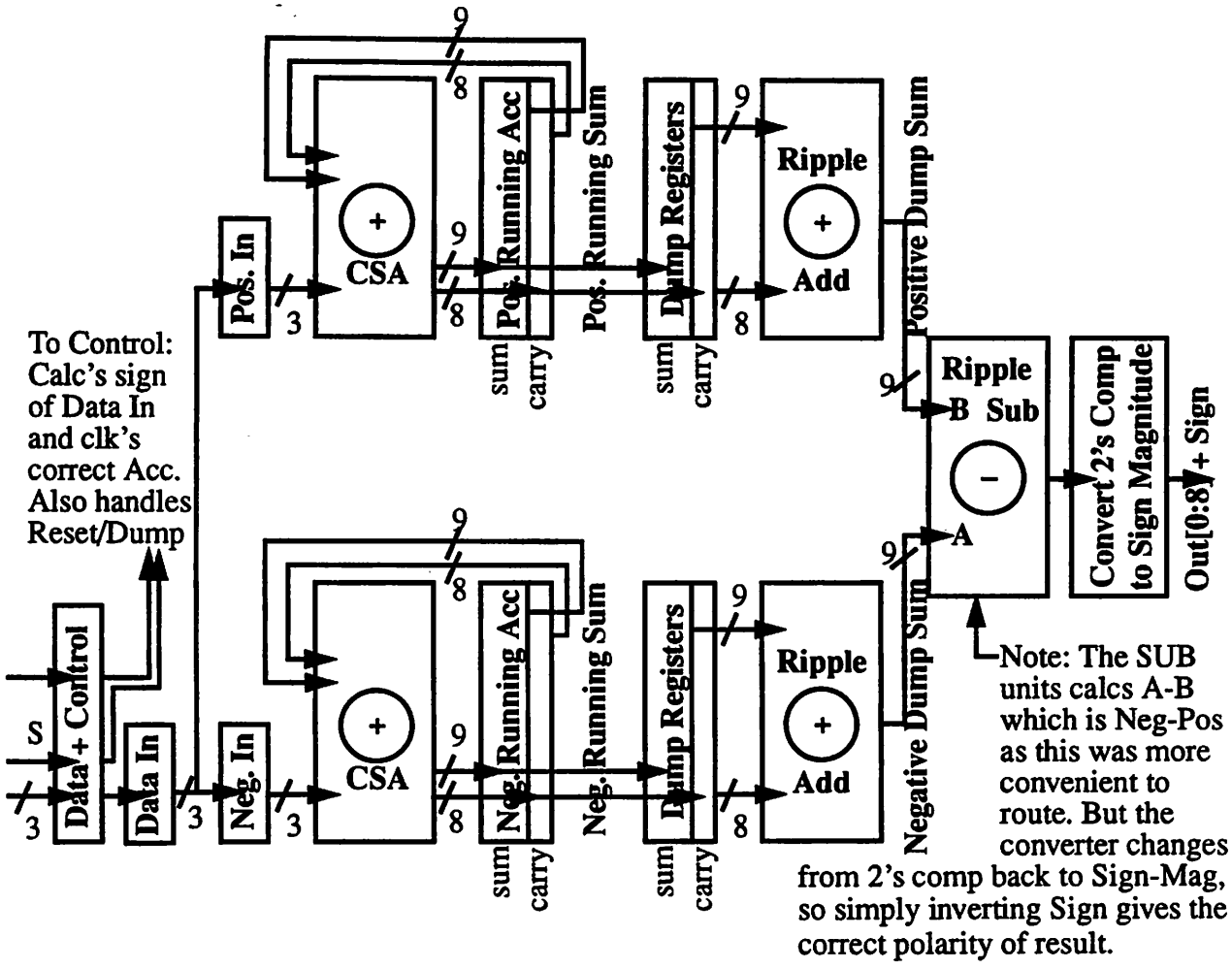


Figure 4-12. Low Level Block Diagram of i_frontend.mag

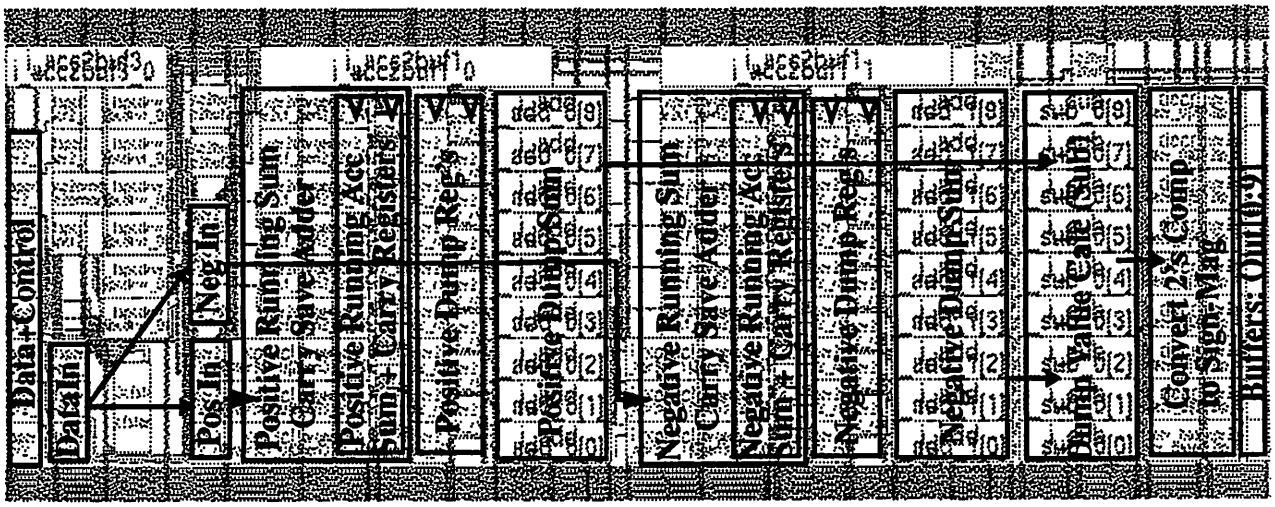


Figure 4-13. Datapath Tiling on i_frontend.mag Layout

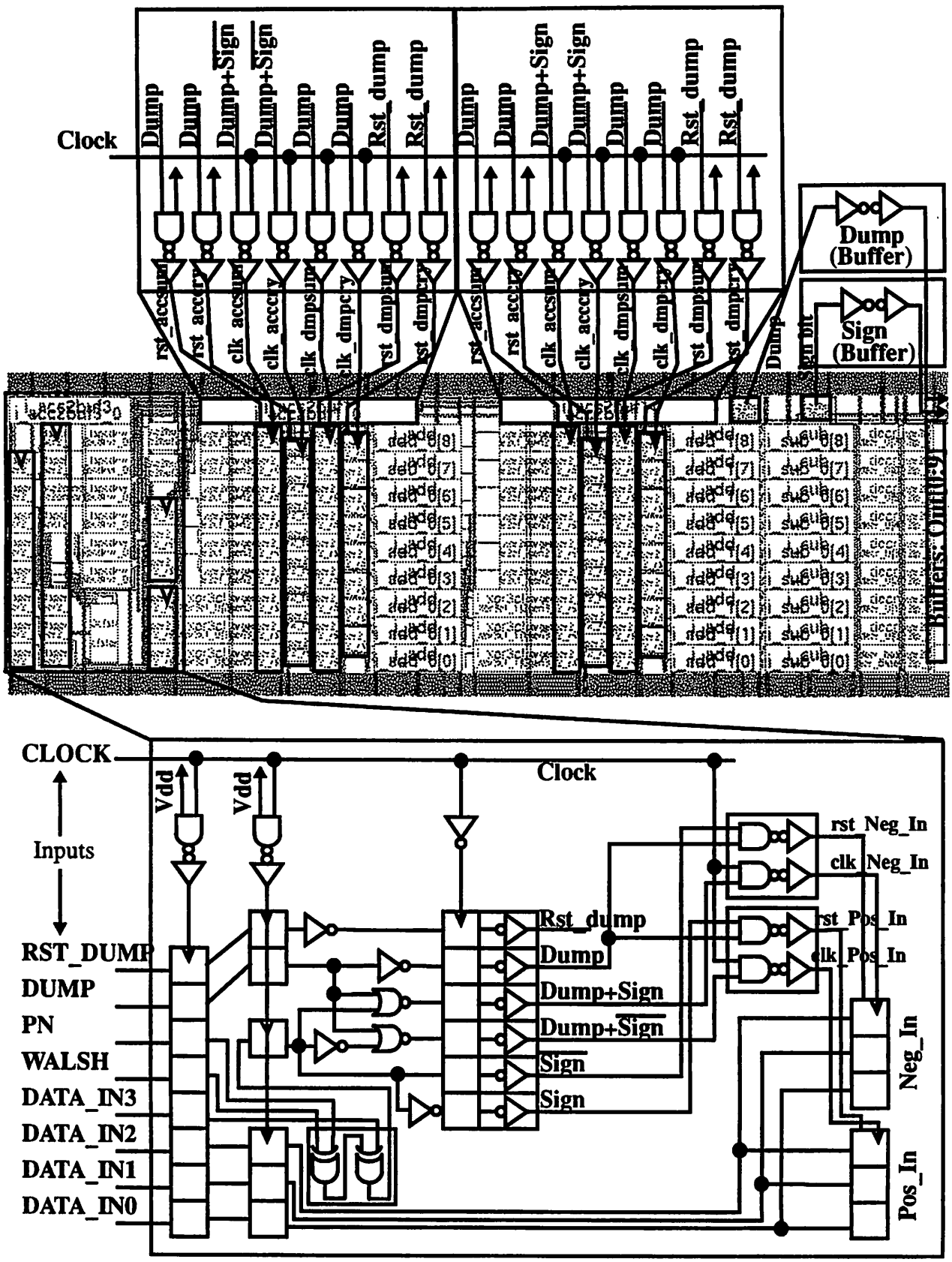


Figure 4-14. Control Logic Diagram on i_frontend.mag Layout

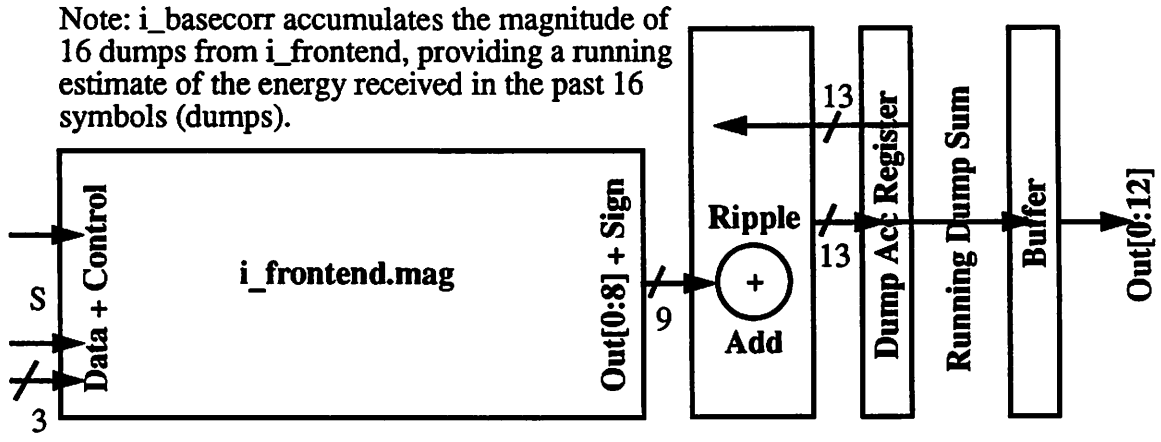
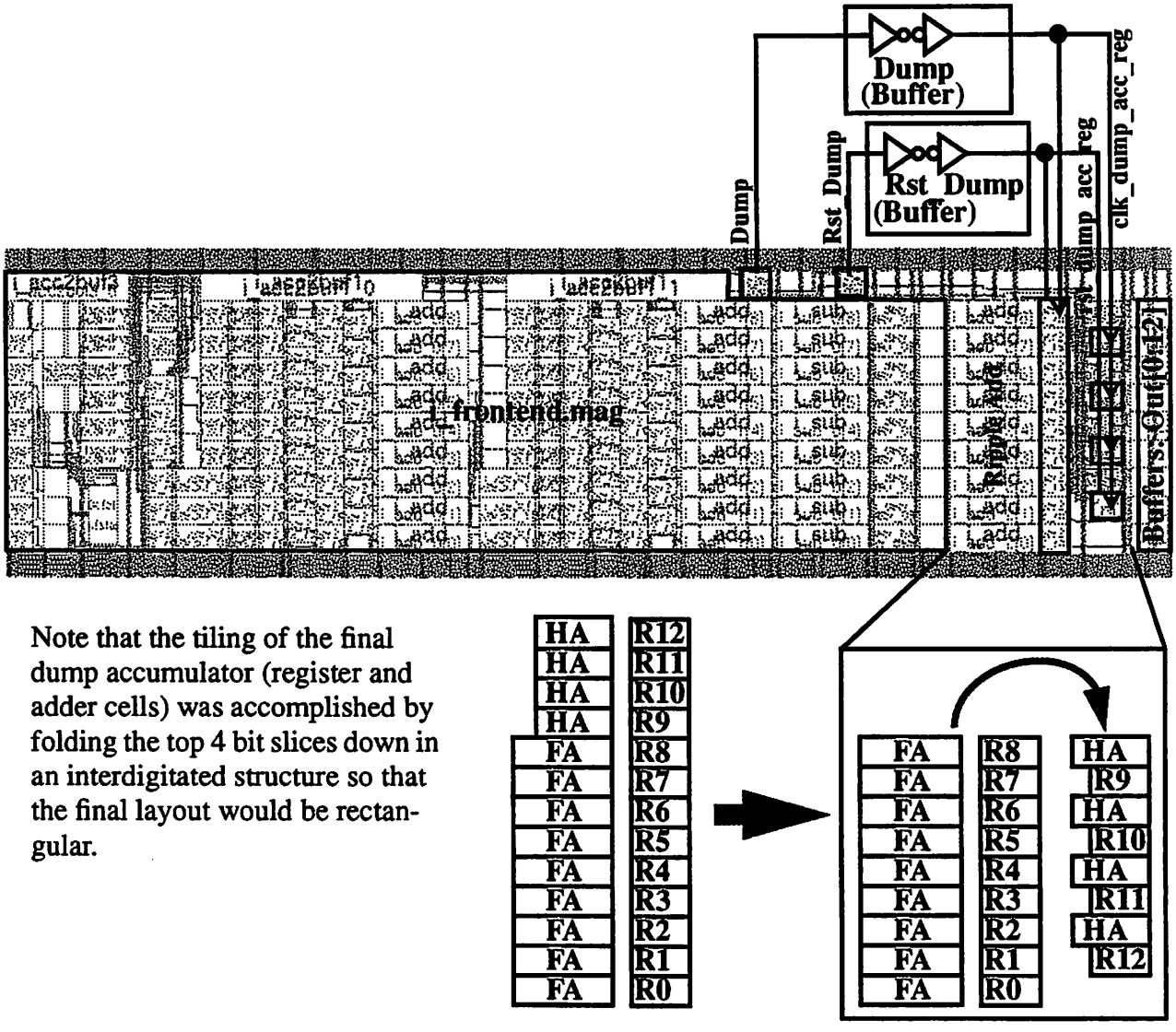


Figure 4-15. Low Level Block Diagram of i_basecorr.mag



Note that the tiling of the final dump accumulator (register and adder cells) was accomplished by folding the top 4 bit slices down in an interdigitated structure so that the final layout would be rectangular.

Figure 4-16. Datapath and Control Tiling on i_basecorr.mag Layout

5 The Revised Correlator Design

The first design of the correlator worked fine, but was designed for a 1.2μ process. After the design was finished we migrated to a better process through MOSIS (from pseudo-0.8 to true 0.8 to true 0.6) and a paper was published from UCLA suggesting the use of a biased number representation for lower power [Ercegovac, Lang]. The changes implied that a smaller, faster, and lower power correlator could be designed. The initial estimates indicated that the correlator could be roughly halved in size and power. As there are 14 correlators (7 complex correlations) on the chip, this means a halving of the correlator power (which roughly 1/3 of the chip power), in addition to the ability to lower the supply voltage for the clock and control, achieving a total power reduction for the chip of around 1/2. Since more correlators would be necessary to do RAKE receiving [Teuscher95], and since they could be conceivably used as computational units in a programmable radio, there was a strong justification for reexamining, and revising the correlator design.

[Ercegovac, Lang] suggests a biased number representation to reduce power (although they look at an older 3.3V conception of the correlator from [Chandrakasan94]). The idea is to use offset binary to reduce the number of accumulators needed to one, to save area, and to employ a slightly different adder/accumulator structure to save power. While the adder structure is not very feasible (it requires an incrementer to ripple delay of $10*T_{clk2Q\ D}$ flip flops ≅ 20ns (0.7μ, 1.5V)), and the interfacing to it a little difficult (+/-1 multiply is now in offset binary), it was not implemented. The expected power gain from it (40%) was hand-analysis, unverified with simulation, and hence a little questionable. However, in spite of this we thought that we could use the idea of a biased representation to maintain signal correlation for low power while realizing the accumulation with a single

adder. Just from being able to cut the carry save registers we can save around 40% in area and power (including the wins of having a smaller geometry). This savings is about the same as the predicted win from [Ercegovac, Lang]. A 2's complement representation, is still undesirable though, as an accumulation around zero will sign-extend toggle the entire adder length, creating a lot of extra activity. Perhaps when the data is correlated, and accumulates to a large positive or negative number, the power of 2's complement is the same as an offset-binary, or POSACC/NEGACC sign-magnitude version, but that means at its best it is the same, at its worst, it is much worse.

The revision of the correlator did not turn out quite as predicted, though. The difficulty in incorporating another number representation into the system, as well as the resizing necessary to meet the timing constraint for a non-carry-save architecture far outweighed the projected benefit. In fact, the final result, while 40% smaller, was 3x worse in power than the original correlator. Although the redesign did not realize a better correlator, the work is recorded here as it presents useful techniques in digital circuit design and it helps to illuminate the path to a low power design.

5.1. Second Correlator Design (for 0.7μ)

5.1.1. Architecture Reexamination

The 0.7μ process characterization data gives a T_p around 300ps, implying 15.6/.3=52 inverter delays! This allows for far more logic depth, implying 20 simple gates in practice. Using some simulation data for a TSPC register extracted in 0.7μ, we see that $T_{clk2Q}+T_{setup}$ is around 2.1ns, allowing around 13.4ns for carry logic or around 13 simple gates for a 9 bit ripple. A carry-save adder is no longer necessary to meet the critical path requirement. In fact, a ripple carry adder (smallest area, regular tiling, low power-delay product), may be feasible. If that doesn't meet the speed requirement, there are a host of other adders; however, we probably won't have to look farther than a simple BCLA (Block Carry Look-ahead) adder to meet the speed requirement. Area and complexity balloon with faster adder structures, such as Conditional Sum, Carry Select, Carry Bypass, etc., and such speed will probably not be necessary to meet the timing requirement. A simple ripple or

BCLA has a good cost/performance if area and power are considered. For more information of adders see [Omondi] and [Rabaey].

The proposed new architecture is shown below. Note that offset binary entails a

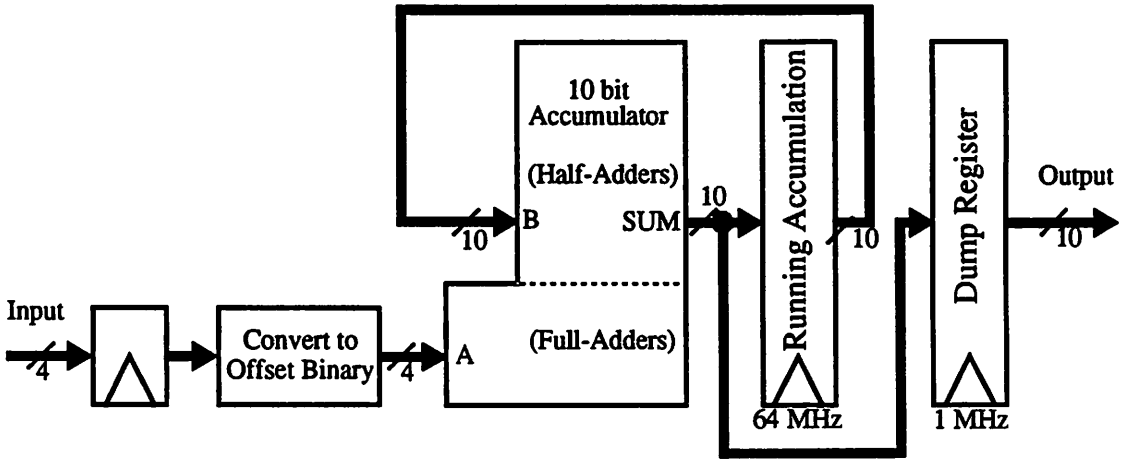


Figure 5-1. Revised Correlator Architecture

conversion from sign-magnitude and 2's complement as shown below in Table 5-1.

Accumulating 64 samples of offset binary means that the adder encodes the desired value + 8*64=512 in the dump. This representation requires a conversion at the beginning of the correlator, and at the end to return the number to sign-magnitude format. At the end the rate is low, as we were already converting from 2's complement we can simply subtract off the offset ($512=2^9$) without sacrificing much area or power. At the beginning, we need to convert 4 bits of sign-magnitude to offset binary at 64MHz. This is more of an issue, and it contributes some power and area, but not enough to nullify or equalize the gain of using the representation. See Section 5.1.8.2. on page 87 for more information on the conversion.

5.1.2. Examining the Ripple Carry Adder

In implementing the ripple adder, it is useful to look at the Boolean equations to get an idea of the critical path. For a ripple carry bit slice, the following equations hold:

$$S[i] = A[i] \oplus B[i] \oplus C[i-1]$$
$$C[i] = A[i] \bullet B[i] + (A[i] + B[i]) \bullet C[i-1]$$

Value	Sign-Magnitude	2's Complement	Offset Binary
-8	N/A	1000	0000
-7	1111	1001	0001
-6	1110	1010	0010
-5	1101	1011	0011
-4	1100	1100	0100
-3	1011	1101	0101
-2	1010	1110	0110
-1	1001	1111	0111
0	1000 0000	0000	1000
1	0001	0001	1001
2	0010	0010	1010
3	0011	0011	1011
4	0100	0100	1100
5	0101	0101	1101
6	0110	0110	1110
7	0111	0111	1111

Table 5-1. Number Representation: 4 bits

A carry-look-ahead adder can be created by defining two new subterms:

$G[i] = A[i] \bullet B[i]$ (Generate Carry Indicator)

$P[i] = A[i] \oplus B[i]$ (Propagate Carry Indicator)

Note that the ripple results, using the above subterms, are easily computed as:

$S[i] = P[i] \oplus C[i-1]$

$C[i] = G[i] + P[i] \bullet C[i-1]$

The carry look-ahead comes from unwinding the recursion at each bit stage, resulting in the following for a 4 bit carry look-ahead adder (C[-1] = 0):

$S[0] = P[0]$

$S[1] = P[1] \oplus C[0]$

$S[2] = P[2] \oplus C[1]$

$S[3] = P[3] \oplus C[2]$

$C[0] = G[0]$

$C[1] = G[1] + P[1] \bullet G[0]$

$C[2] = G[2] + P[2] \bullet G[1] + P[2] \bullet P[1] \bullet G[0]$

$C[3] = G[3] + P[3] \bullet G[2] + P[3] \bullet P[2] \bullet G[1] + P[3] \bullet P[2] \bullet P[1] \bullet G[0]$

If we look at the literal logic implementation of this (with positive-logic, and using two-input gates for speed), we find:

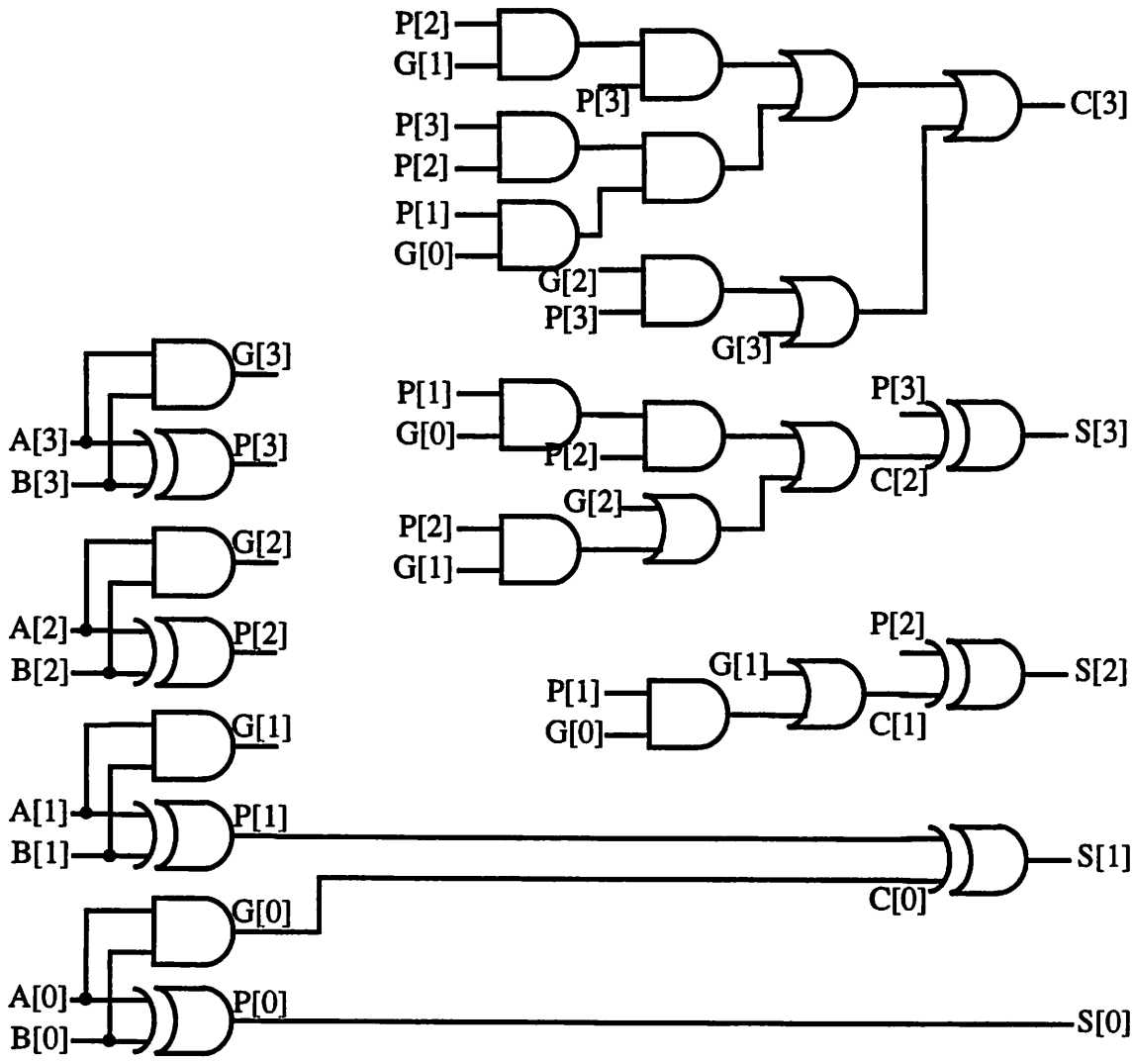


Figure 5-2. Carry Look-Ahead Adder, 4 bits

This is not how it would necessarily be implemented, but it can serve to give a minimum critical path count (5 gate delays) and to give a sense of the area and routing involved (27 gates total, with a rather irregular layout). Compared to the ripple adder positive-logic, 2-

input gate implementation, below, which only has 17 gates (with a regular tiling), but a crit-

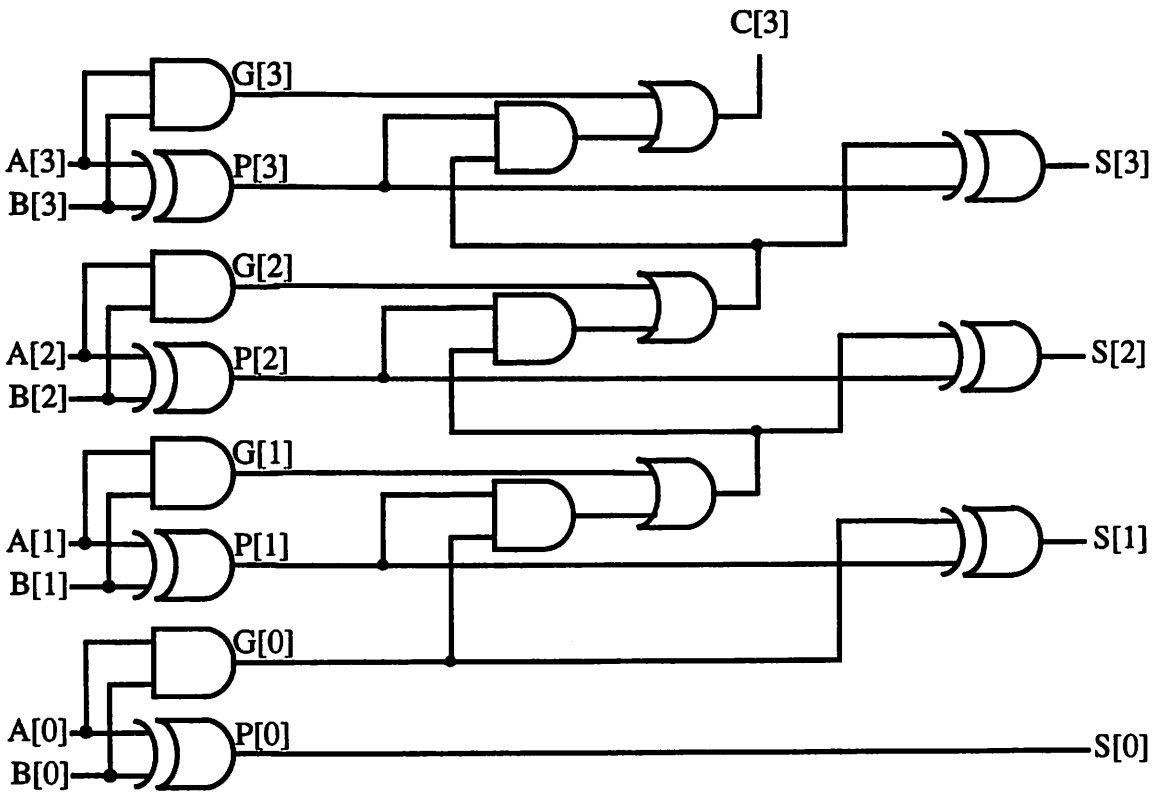


Figure 5-3. Ripple Carry Adder, 4 bits

ical path of 7 gates. Of course, not all gates have equal delays, at this level we are just getting a coarse estimate.

If we include a ripple half-adder structure to implement the top of the accumulator, the total critical path is 7 + another 6 gates for 13 total. As there is only about 13ns estimated for the carry logic, and as fairly large NAND gate has a delay of 0.8ns, this would seem to just barely fit (allowing 1ns per gate). Unfortunately it was a little tight (at the time we were looking at a true 0.8μ process with $2*T_p$ around 1ns), and hence was abandoned. On the other hand, a full carry-look-ahead adder seems to be rather unwieldy to implement. However, since S[3] is at worst around 6 gate delays, the only problem is C[3]. We can implement a Block Carry Look-ahead structure for just C[3] to speed it up, thereby not incurring the full penalty of a CLA adder. The C[3] generation in Figure 5-2 only has 5 gate delays neglecting loading. Also C[3] can be implemented relatively easily without sacrificing too much area, power, or design time. (Note that the rippling through the half-

adders winds up being simple NAND gates which are already fast and compact, so the attempt to gain headroom was not made there, although one could do BCLA there too without much expense.) The proposed BCLA adder is shown below. The critical path is 5 gates,

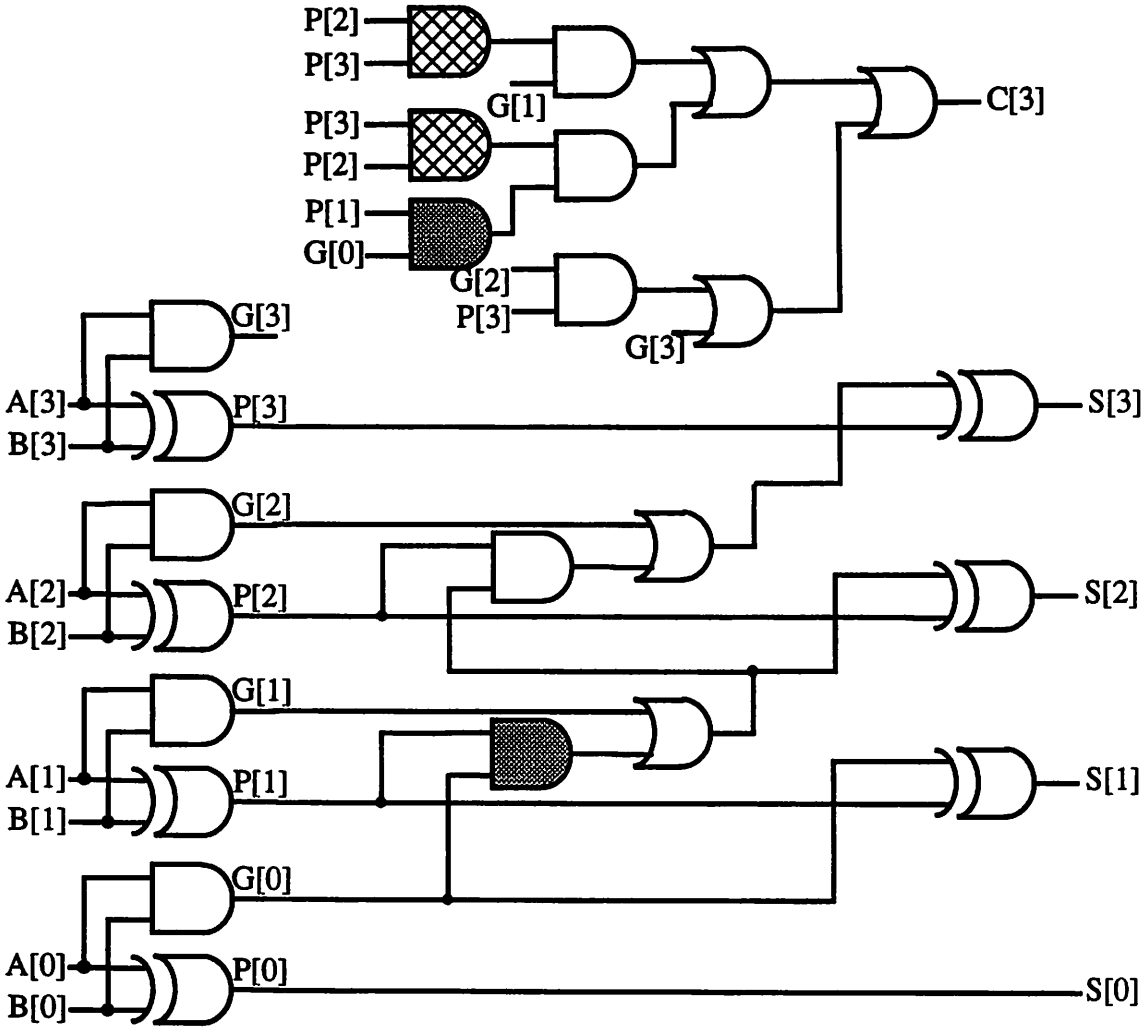


Figure 5-4. Block Carry Look-ahead Adder, 4 bits

and the total number of gates is 24 (22 if you notice that the shaded gates are duplicating functionality without increasing the critical path). For the 0.6μ process that the revised chip will probably be fabricated in, it may be fast enough to simply use a ripple carry adder, in which case the only issue is that of converting to offset binary, as the accumulator reduces to a trivial tiling of full and half adder cells.

Looking at the half-adder cells:

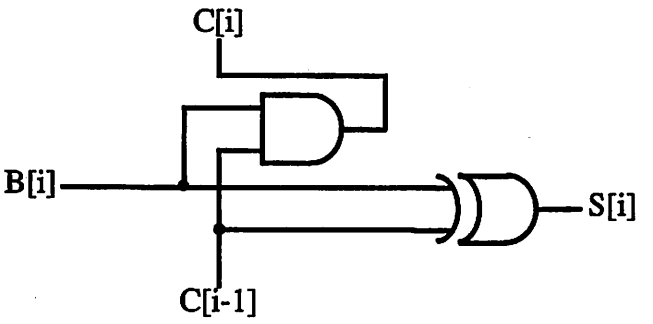


Figure 5-5. Ripple Carry Half Adder Bitslice

The overall ripple carry adder/accumulator would look like:

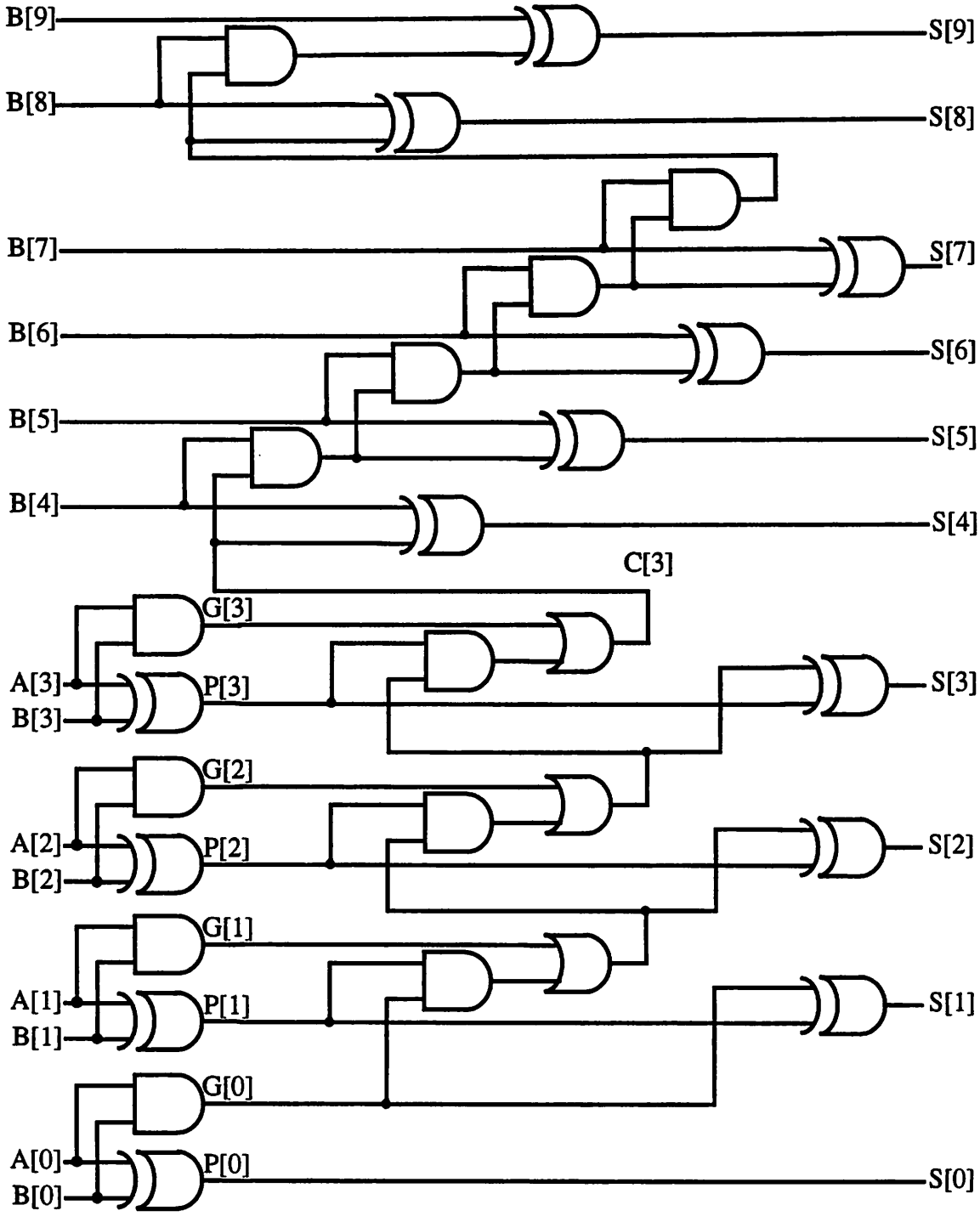


Figure 5-6. Ripple Carry Accumulator

5.1.3. Accumulator Implementation

The most promising topology looks like a ripple or BCLA-ripple adder. Before we can try implementing the Boolean logic functions, we need to say a word about circuit style. Static CMOS was chosen as it is robust, operation scales with Vdd, and is easy to design in. Dynamic logic was not chosen as it tends to stack gates, which at 1.5V slows down very quickly. (Also it is undesirable to route the clock all over the design.) Pass transistor logic was again too slow in this process relative to a static CMOS gate. No ratio-ed logic styles were used as they consume too much power.

One could do a literal implementation, making an XOR, AND and OR, however this doesn't exploit the inverting nature of static CMOS. For speed, low fan-in gates (2 input gates) will be examined, which is not a problem for the ripple adder whose outputs fan-out to no more than 2 inputs at any stage. We are not necessarily constrained to implement the Boolean functions in the topology shown in Figure 5-6, however, we can use some bubble-pushing tricks to convert the AND's and OR's into NAND's and NOR's which offer a fast implementation.

At this point a word needs to be said about this approach to the redesign of the correlator. Namely, the problem with this approach is that it will not necessarily wind up with a lower power version. Low power is achieved through lowering Vdd, using minimum size devices, and making up for speed with area by pipelining and parallelizing the circuit. There is a direct trade-off between power and area. This design style attempts to save on area and power by sizing up devices and using as few gates as possible. While saving gates does save area, sizing up to compensate for the unavoidably longer critical path does not save power. There is a question as to how much sizing up is allowable before it starts to become too much overhead. The previous correlator design required the XOR's sized up by about 2x to meet timing, however, in a faster process a more viable attempt to lower power might make all devices minimum size and compensate with the overall architecture. (One might also lower power by attempting to remove every-other carry register, keeping the same size as the previous correlator for speed. However, this approach, while lessening around 1/4 of the only the clock power, would make the layout irregular, and would not be as efficient as lowering all device sizes by 3/4 and keeping the same structure.) The moral

is: Use as minimum size and if you have to size up larger than 2x, re-examine the architecture within the constraints of allowable area usage of course.

5.1.3.1. NOR Approach:

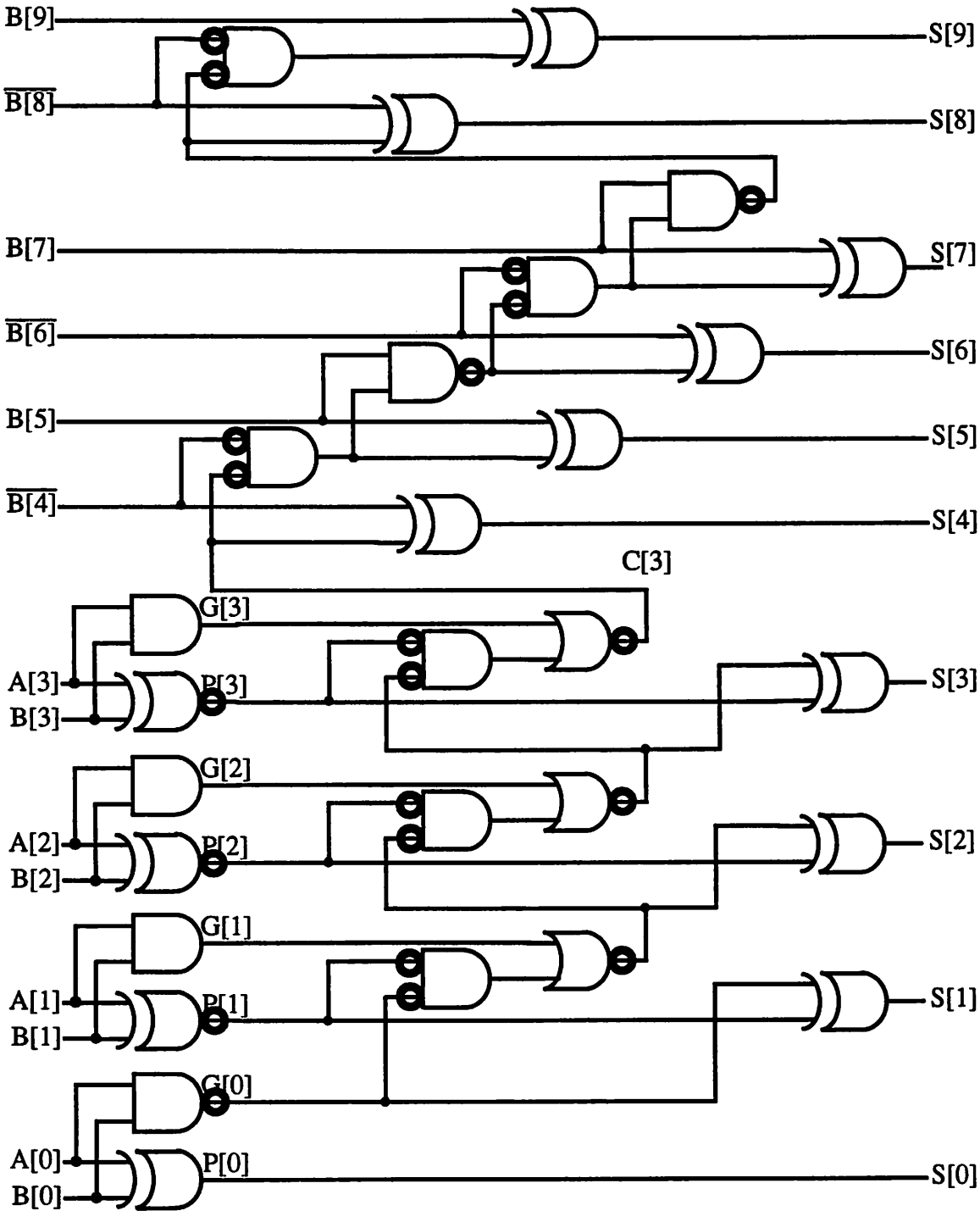


Figure 5-7. Ripple Carry Accumulator with NOR's

Note the critical path = $2*T_{XOR} + 2*T_{NAND} + 9*T_{NOR}$

5.1.3.2. NAND Approach

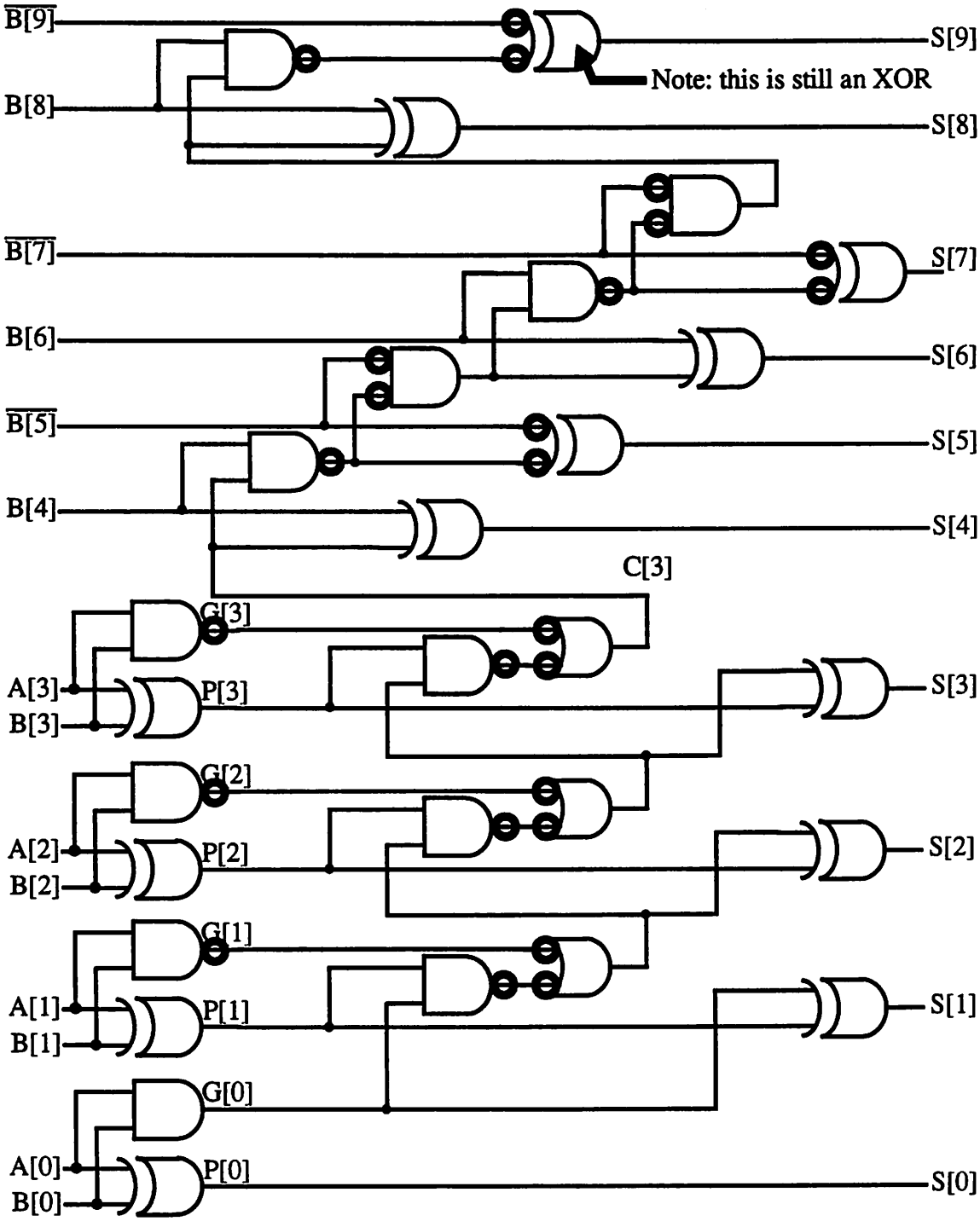


Figure 5-8. Ripple Carry Accumulator with NAND's

Note the critical path = $2*T_{XOR} + 2*T_{NOR} + 9*T_{NAND}$

Since NAND's are faster than NOR's in our process (an NWELL process), we'll go with the NAND approach.

5.1.4. Library Cells for Design

For the redesign of the correlator a semicustom design approach based upon a hand-designed library was taken. Simple cells (AND, OR's, etc.) as well as some more specific gates for carry generation and sign-magnitude to offset-binary representation were designed and pitch-matched. This approach allows for reuse, but still gives some flexibility to the design, as opposed to a standard cell design with a fixed library or a full custom implementation like the first correlator. Again a datapath style like [Burd94] was chosen for easier layout with data streaming in from right to left, and control and power flowing up and down. This approach is not optimal in the sense that extra capacitance on non-critical path's will be switched since a regular cell-based approach is being used. A back-of-the-envelope calculation of savings from cells that don't need to be as big yields around 20% of overhead in power. Also high packing density is not achieved as the cell height is determined by the worst case block size. Again, back of the envelope calculations suggest an extra 20-30% of area savings for an entirely hand done layout.

This time around, for the accumulator, I chose not to make half-adder and full-adder cells, instead simple gates were made and stacked. This was done because:

- It is more flexible and re-usable, while requiring less design time at the layout and SPICE verification level.
- Half-adder (HA) and full-adder (FA) cells are not tremendously more dense, and wouldn't allow for down-sizing on non-critical path ('small' and 'large' cells could be designed, but that goes for all library cells and was mainly not done due to time constraints.)
- The layout is not terribly regular (if tightly packed) when a BCLA. There weren't a lot of opportunities for HA or FA cells. It turned out to be easier to pack up and tile the accumulator if the blocks were smaller (the FA was split up for efficiency's sake).

Which brings me to the main reason:

- It was more expedient.

The only real change to the DPP style for the cells from [Burd94] is that they were made 66λ tall (instead of 64λ) to allow for the fact that the 0.6μ design rules require 3λ poly to poly spacing, as opposed to 2λ previously. So an extra 2λ were added to allow for the XOR gate to fit, all tiled up, into 1 column per well.

Since the redesign of the correlator involves new library cells, common library cells (such as NAND, AND, XOR, etc.) need to be characterized for delay to further evaluate the implementation proposed above. Similar to the process characterization chapter, modular SPICE files were written and all associated delays for a single loaded gate were determined automatically. This way different logic styles (i.e. CPL vs. Static CMOS) could be laid out and evaluated on their performance.

For automatic SPICE characterization two parametrized waveforms are generated for A and B inputs (as shown below) and the output of the logic gate was loaded with one

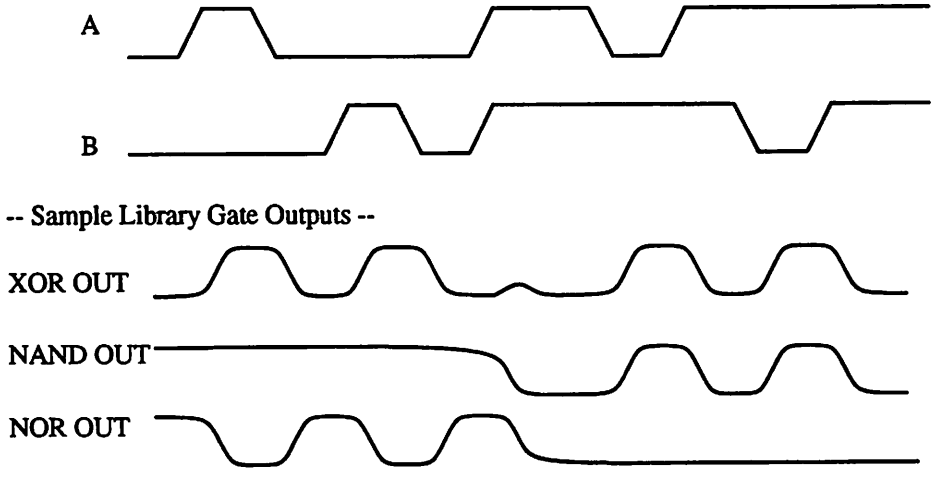


Figure 5-9. SPICE Library Input and Output Waveforms

input of a NAND gate (with it's other input grounded). This loading probably should be done for an unloaded output and a known load value (say 100fF), to get pure delay and drive components for simple linear modeling. However, only a ballpark number was needed to ensure that a comfortable margin existed for the critical path (at least a couple gate delays). The edge rates were kept at reasonable values (about 2x T_{edge} from ring oscillator data) for the process which is around 1 to 1.5ns for the 0.8μ process. From SPICE we can simply pick out the delay that we'd like to measure ($T_{OUTrise,Afall,B=1}$ for example).

We could use actual T_p's for critical path calculation, but this would ignore loading effects. Load could be taken into account without too much extra effort -- an estimate of the load cap may be plugged into a linearized model (what some CAD programs do). To be conservative, the worst case delay was taken as the estimate.

5.1.4.1. Summary of Revised Correlator Library Cells

SPICE outputs (approximate worst case delay and edge rates) are listed below for the library cells. Note that in general the rising edge is the worst case if the gate has stacked PMOS since this is an NWELL process. These numbers are for 1.5V operation in the 0.7μ (0.6μ L_{drawn} minimum, but with λ=0.35μm for SCMOS design rules) HP process and are all loaded with a 50fF capacitance (about 70 λ of gate cap). The gates are sized around the propagation delay vs. width breakpoint of Figure 3-10, "Delay and Edge Rates from Level 39 0.7m Model, 1.5V," on page 34 (which is around 12λ). Again, the idea was to size up the gates for speed (without making them excessively large) to meet the timing constraint. However, as was mentioned before, this approach does not result in a lower power design although it does meet timing.

AND (i_2and)	T_P = 1.3ns T_{edge} = 0.8ns
NAND (i_2nand)	T_P = 0.8ns T_{edge} = 0.8ns
NOR (i_2nor)	T_P = 1.2ns T_{edge} = 1.5ns
XOR (i_2xor)	T_P = 1.5ns T_{edge} = 2ns
XNOR (i_2xnor)	T_P = 1.5ns T_{edge} = 1.9ns

Also tested were some TSPC registers.

TSPCR (i_tspcr3) $T_{clk2Q} = 1.4ns$
$T_{Qedge} = 0.7ns$
$T_{setup} = 1ns$

TSPCXR (i_2xortspcr_rst) $T_{clk2Q} = 1.5ns$
$T_{Qedge} = 0.8ns$
$T_{setup} = 1ns$

Several transistors were resized in the TSPC flop to improve the speed of the design. The

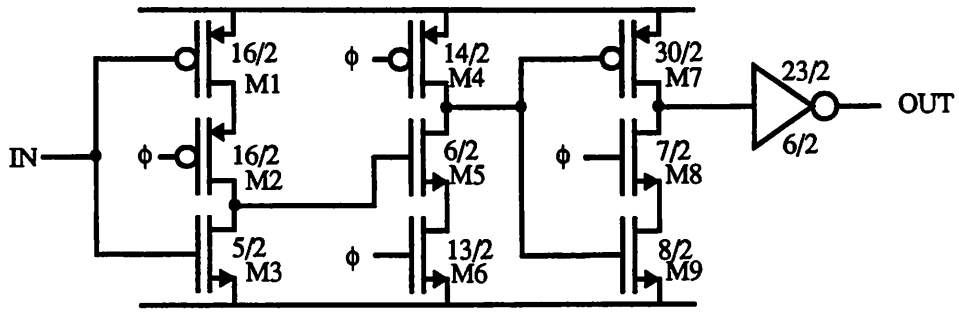


Figure 5-10. Redesigned TSPC Register (For faster operation.)

set-up time worst case was lessened by making M1 and M2 bigger. M6, the evaluation transistor was increased in size for faster evaluation. M4 needed to be increased to compensate for the larger load (M7 and M9) that precharge was seeing. As the worst case path is when M6 evaluates, M7 was made very large to speed up the delay value. The inverter was ratioed 4 to 1 to give equal rise and fall times. Adding a reset is easily accomplished by adding another transistor in parallel with M5 for synchronous reset, or in parallel with M5 and M6 (i.e. from the drain of M5 to GND) for an asynchronous reset.

5.1.4.2. Accumulator Implementation Evaluation

Using the above numbers, we can plug some numbers into out implementation's critical path and check out the results. For a ripple carry accumulator we get a critical path of Clk2Q(1.4ns)+2*XOR(1.5ns)+9*NAND(0.8ns)+2*NOR(1.2ns)+Setup(0.7ns)=14.7ns which is very close to our 15.6ns cycle time (especially when taking edge rates into account). This implies that we will have to look at speeding up the carry generation and perhaps exploring the block look-ahead structure discussed previously. After Carry[3] in

Figure 5-6 we see 3 NAND's, 2 NOR's, an XOR and a setup comprising 7ns of delay. Up to Carry[3] we see Clk2Q, an XOR and 6 NAND's for 7.7ns of delay. If we approach speeding up the critical path from a carry look-ahead approach, we see that we have about 6 NAND's (4.8ns) to compare a BCLA against (since both will have a Clk2Q+XOR). Recall that the Boolean expressions give a carry generation as shown below. If imple-

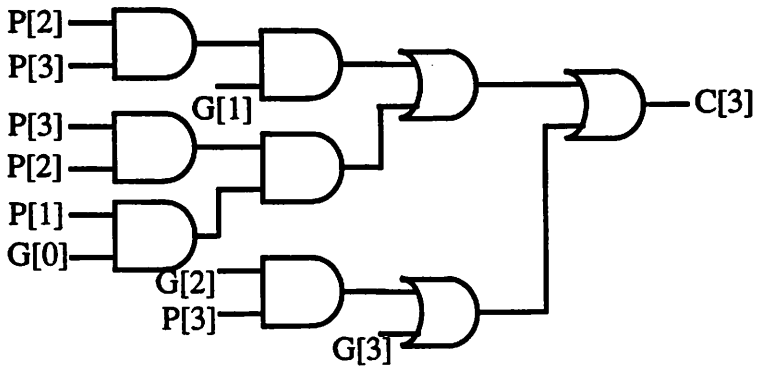


Figure 5-11. Carry[3] Generation: AND/OR

mented as shown, the delay is 2*AND(1.3ns)+2*OR(1.2ns)+2*INV(0.4ns)=5.8ns which is a full 1ns worse! Obviously there must be a more intelligent way of implementing the carry look-ahead, and by playing around with bubble pushing we can obtain the following structure. There are other possibilities for bubble-pushing, but I believe this one is the fast-

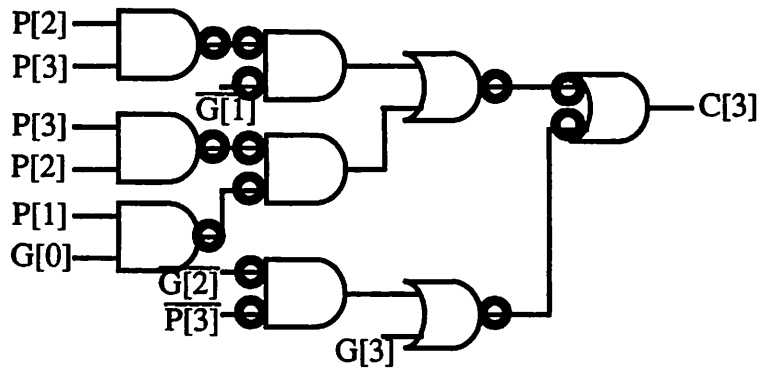


Figure 5-12. Carry[3] Generation: NAND/NOR

est for 2-input gates as it has only NAND's and NOR's, and as few NOR's as possible at that. The delay is 2 NAND's and 2 NOR's (4.0ns). That is good, we've managed to shave off a NAND delay (0.8ns) for a total overheard of around 1.8ns (about 11% of the duty cycle). But even as such, it is still rather tight, and questions could be asked as to where we might be able to shave off another nanosecond and what cost in complexity will that cause. Delay could also be attacked using block look-ahead for the carry block after Carry[3], but

this ripple chain is already simply 3 NAND's and 2 NOR's, implying that a win would be probably on the order of removing a gate, but it will involve more complex routing/area. Two other suggestions, examined in the next couple sections, are possible within the structure of this BCLA adder: move some of the logic computation into the latches, and/or explore complex gate implementations for Carry[3] (as opposed to cascaded 2 input gates -- this also may improve the routing and area minuses to doing a BCLA). The first thing we'll look at is the idea of merging logic into latches.

5.1.5. Merging Logic Into Latches

An idea to increase the speed of the critical path is to merge the latch functionality into the logic that is being computed (usually saving setup or evaluation time). Notice that all of the accumulator registers are preceded by an XOR gate which computes the sum's result. Rather than latching the output of the XOR, we can design a latch that takes A and B inputs and, on the rising edge of CLK, produces A⊕B as an output. Ideally this saves time of the XOR operation by moving it into the latching operation. In practice the win is a bit less as the extra logic either increases the setup or evaluate time. This might seem like a good idea, and indeed it saves about 1ns off of the critical path, but it winds up being an example of micro-optimization; costing more at the next level of hierarchy than it is worth. In general it is a worthy technique, and hence is discussed here, in spite of the fact that it winds up not being a big plus. To see why it is not a win in this design involves two reasons:

1. It complicates the design. If all we needed to do was latch the accumulated result, this might be fine, but we also need to dump the accumulated result from a separate set of registers (one keeping the running accumulation, the other keeping the dump result).

This means that we need two sets of XOR latches which is more routing and area than using an XOR followed by two registers. (Shown below, note that the signal OUT no longer explicitly exists with XOR latches.)

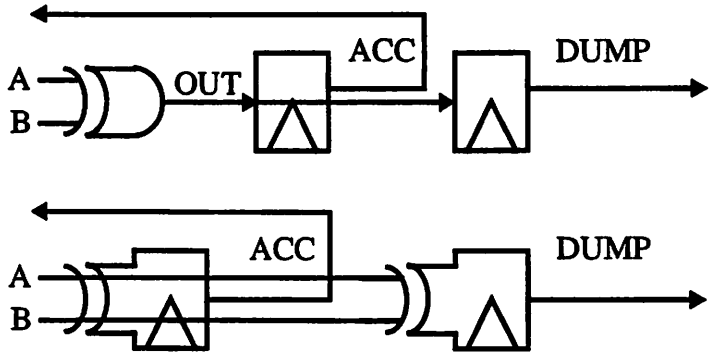


Figure 5-13. Proposed Bit-Slice with XOR Latches

2. The real reason, beyond area, is that it winds up being more power. The clock load of an XOR latch is more than for a simple latch. Merging the logic into the evaluation creates stacked gates and larger clock transistors, as well as increasing the setup circuitry complexity and power.

In spite of the result that it is not an overall win, it does win in performance (if that were the main goal) and it is interesting technique.

The main idea of merging logic into latches has already been mentioned. Basically you want to incorporate computation functionality into the latch's setup and/or evaluation stages (for a dynamic latch). The place to insert logic for those two techniques are shown below. Note that if only a high->low transition (if any transition) is guaranteed on the eval-

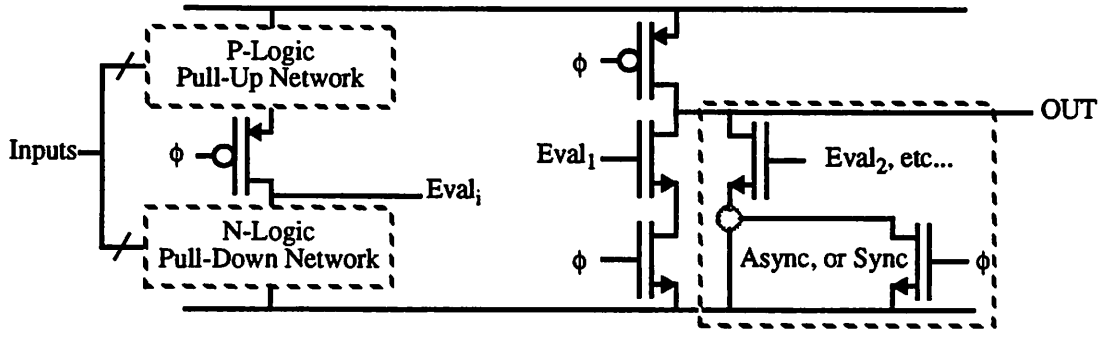


Figure 5-14. Generic Template for Incorporating Logic Into Latches

uation transistors, then logic may be included through a wired-OR (this is usually done for the reset for the low power library TSPC registers.) Note that too much logic will make the

latch extremely large, slow due to excess internal capacitance and stacked devices, and unwieldy, so use proper discretion.

The first idea for building an XOR latch is to explicitly integrate the XOR design (Figure 4-4, "XOR Gate Implementation," on page 42) into the latch.This, unfortunately,

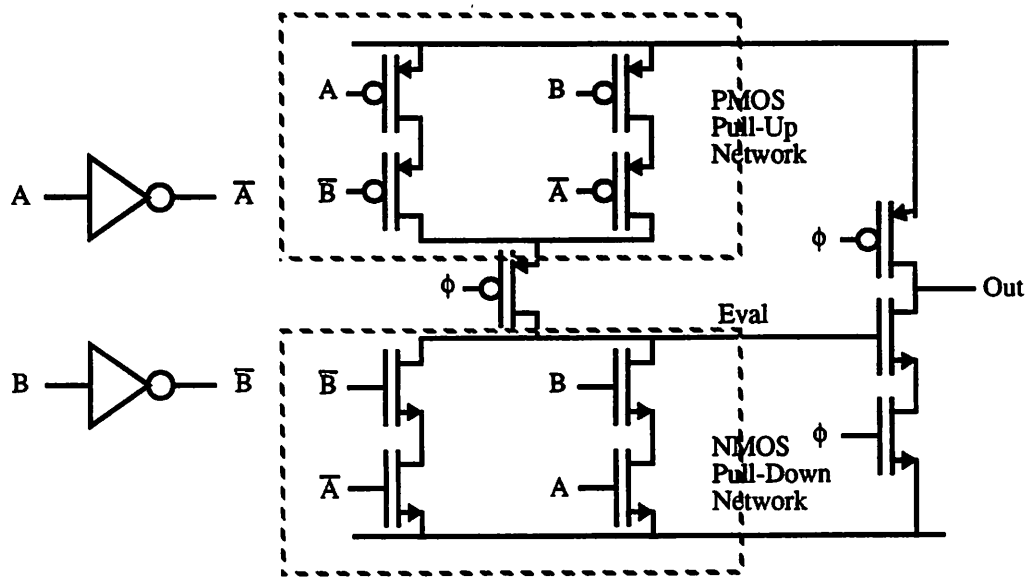


Figure 5-15. First Proposed XOR Latch Design

results in a rather poor design. The PMOS are stacked three-high which is bad for drive at 1.5V and requires very large devices to operate quickly, it is not easy to layout, and the extra inverter delay eats up about half of the prospective gain. If we try the wired-OR approach instead, moving the NMOS pull-down network from the setup stage to the dynamic inverter of the evaluation stage (shown below), we still run into problems. Now,

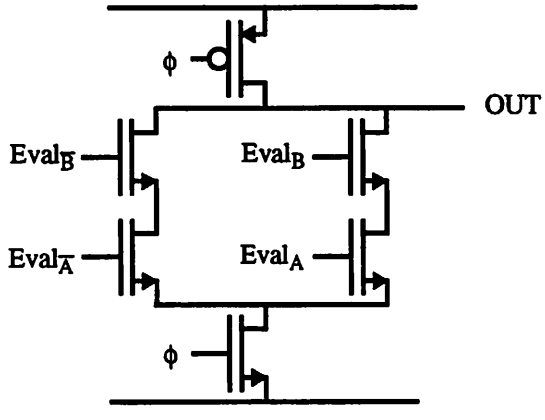


Figure 5-16. Second Proposed XOR Latch Design

four separate input stages are needed to provide A, $\overline{A}$, B, and $\overline{B}$, in addition to still needing

the extra inverter at the input. Note that we can't simply add an inverter to $Eval_A$ to get $Eval_{\overline{A}}$ as this will violate the inversion rule for a no-race condition. All is not lost; however, with a little cleverness we can design a workable XOR latch. By noting that the XOR operation (as illustrated by the NMOS pull down network above) is the OR-ing of A•B and $\overline{A}$•$\overline{B}$. By using DeMorgan's Theorem, we see that A•B = $\overline{A}$ NOR $\overline{B}$. Likewise $\overline{A}$•$\overline{B}$ = A NOR B. Incorporating a NOR into the setup stage is not too bad (there are three stacked PMOS, but they are all in a row and may be laid-out in series, reducing the internal drain/source capacitance), and the overall latch only requires two input stages. The resulting XOR latch is shown below. Two inverters are still necessary to provide the inverses of A

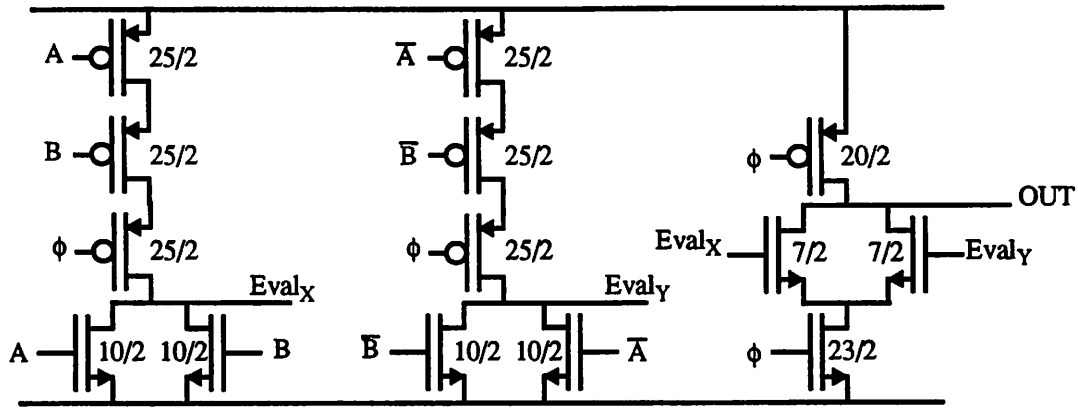


Figure 5-17. XOR Latch Design

and B, but now the circuit is reasonably sized and fairly easy to layout, although still large. The above P-block was followed by an N-block and inverter to create an XOR register

(edge triggered) library cell which is pictured below: The cell was SPICE'd and it does

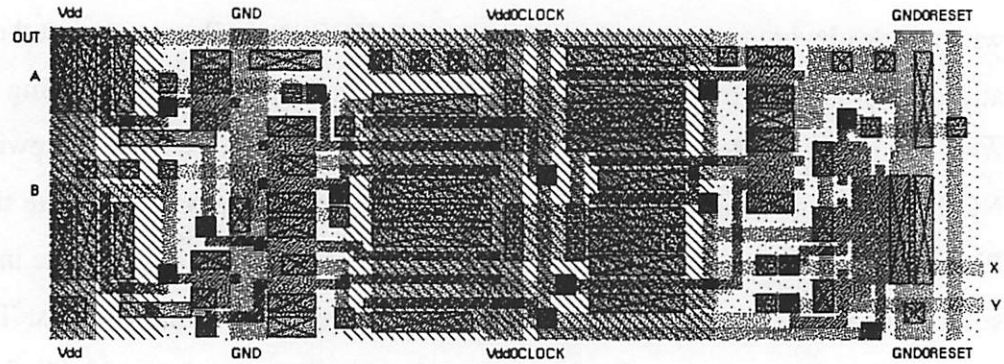


Figure 5-18. XOR TSPC Register Layout

indeed function as an XOR latch. The SPICE'd results are as follows:

TSPCXR (i_2xortspcr_rst) $T_{clk2Q} = 1.5ns$

$T_{Qedge} = 0.8ns$

$T_{setup} = 1ns$

The overall critical path decreases by 1.2ns using this XOR flop. We lose 1.5ns for the XOR, but gain an extra 0.3ns necessary for setup, so the overall win is only 1.2ns. That is good; however, when we look at area and power we see that the new latch is not necessarily good. Comparing the area of an XOR and TSPC register to the XOR-TSPCR we see that the new XOR flop is larger by 2.7x. In examining power we can compare the capacitance values and see that the XOR flop is again worse. The input gate cap (in λ, from our process info we could convert to fF: 0.42 fF/λ for 0.8μ process) is 240λ (60λ per input, goes to two flops) whereas, simply using an XOR and TSPCR gives 170λ (65λ per input, two flops with 20λ input) which is about 30% less. Also if you look at the clock capacitance, since the XOR latch has 3 stacked PMOS, the clock load is large, at 103λ of gate cap = 65fF(plus 15fF extracted routing). The TSPC register only has 33fF of clock load (less than half). In addition to the input cap, the internal cap is larger and overall the power is around 40% worse per XOR flop!

A last attempt to improve the power consumption of the XOR latch may be made by noting that we don't need two full XOR latches as in Figure 5-13 as the two input stages

(NOR's) are duplicated in each latch. The setup stage for the second flop may be removed, only duplicating the evaluate stage, to produce a sort of XOR half latch. That is, use the P-block above in Figure 5-17 and add on an additional evaluate stage (shown below). We

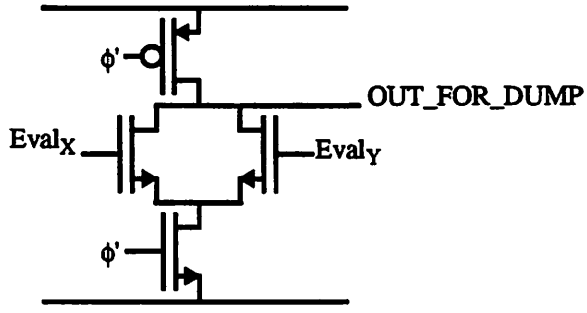


Figure 5-19. XOR Half Latch

know that the accumulator XOR latch will always be clocked, hence $Eval_X$ and $Eval_Y$ will always be valid when a DUMP is issued (using φ' as opposed to φ). This is a risky thing to do, as the skew between φ and φ' will be a critical issue, but it produces a working simulation and it reduces the clock load of the half-XORflop stage to be approximately the same as a normal flop. The skew effect can be analyzed for the following two cases:

1. If φ' is faster than φ, the skew will cut directly into the critical path, requiring the data to be stable Δ_{skew} sooner.
2. If φ' is slower than φ, we may run into a race condition. If $Eval_X$ or $Eval_Y$ can change or get to a metastable state before φ' hits, we will get incorrect or forwarded data. The only safety margin is the intrinsic hold time of the circuit (i.e., how soon it can react to a clock change) which is around 0.8ns for the NOR to pull-up enough to cause a problem. There isn't a problem pulling down, as the data won't be able to get out of the latch and back around faster to pull down $Eval_X$ or $Eval_Y$ (this would take at least 1ns).

Even assuming that the load could be balanced and hence skew matched, the result, while smaller in area and power than two XOR flops, is still larger and more power than simply using an XOR and two latches. It turns out that a more fortuitous approach is to look at a complex gate implementation for Carry[3] generation for reducing the critical path length.

5.1.6. Investigating Carry Look-ahead Generation Logic

Recall that in our critical path estimations, we only seem to be within 11% of margin (1.8ns). Another way to improve the circuit performance in speed as well as reduce its size is to investigate complex gates for carry generation. The previous estimates were

made assuming low fan-in, two-input gates for high-speed operation at low voltages. It turns out, however, that some logic functions can actually be implemented faster at low voltages with a slightly more complex gate with moderate fan-in. Another half-nano second of overhead can be gained by examining the Boolean equations, and by using an AND/OR-Invert gate (which is easy to realize in static CMOS) instead of the commonly conceived Boolean operators. In fact, this gate is frequently used by synthesis tools and is a very useful, simple gate. Although the speed up is not spectacular, it does help provide us with that last bit of safety margin without too much cost.

The typical structure for the OR-AND-Invert and AND-OR-Invert gates are shown below. Note that the Boolean equation for the OR-AND-Invert is given by:

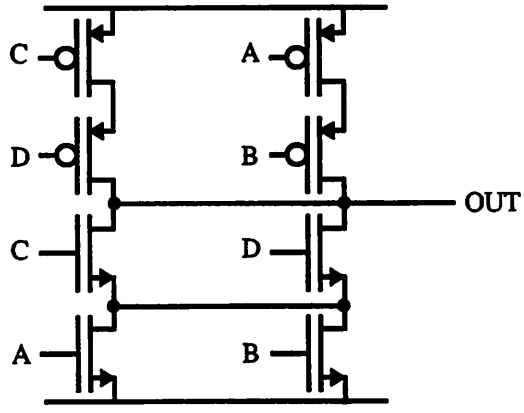


Figure 5-20. OR-AND-Invert Circuit Implementation

$OUT = \overline{(A + B)\bullet(C + D)}$. Note that this is AND-OR with inverted inputs: $\overline{A}\bullet\overline{B} + \overline{C}\bullet\overline{D}$

Whereas the inverse of this structure, the AND-OR-Invert (shown below), yields:

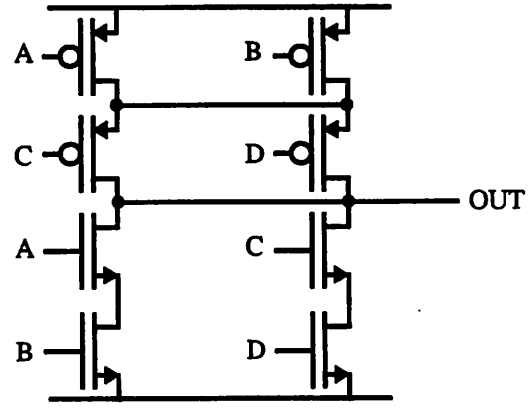


Figure 5-21. AND-OR-Invert Circuit Implementation

OUT = $\overline{A \bullet B + C \bullet D}$. Note that this is OR-AND with inverted inputs: $(\overline{A} + \overline{B}) \bullet (\overline{C} + \overline{D})$

Now recall that the carry generation for Carry[3] is given by:

$$C[3] = G[3] + P[3] \bullet G[2] + P[3] \bullet P[2] \bullet G[1] + P[3] \bullet P[2] \bullet P[1] \bullet G[0]$$
$$= (G[3] + P[3] \bullet G[2]) + P[3] \bullet P[2] \bullet (G[1] + P[1] \bullet G[0])$$

hence,

$$\overline{C[3]} = \overline{(G[3] + P[3] \bullet G[2]) + P[3] \bullet P[2] \bullet (G[1] + P[1] \bullet G[0])}$$
$$= \overline{(G[3] + P[3] \bullet G[2])} \bullet [\overline{P[3] \bullet P[2]} + \overline{(G[1] + P[1] \bullet G[0])}] = X \bullet Y,$$

where,

$$X = \overline{(G[3] + P[3] \bullet G[2])},\ Y = [\overline{P[3] \bullet P[2]} + \overline{(G[1] + P[1] \bullet G[0])}]$$

As $\overline{C[3]}$ = X AND Y, then C[3] = X NAND Y which is fast. Also note that X is an AND-OR-Invert operation. This suggests a simple, fast topology for carry generation. The X term is obtained from a cell (called i_aoitop) shown below. This cell is small, easy to

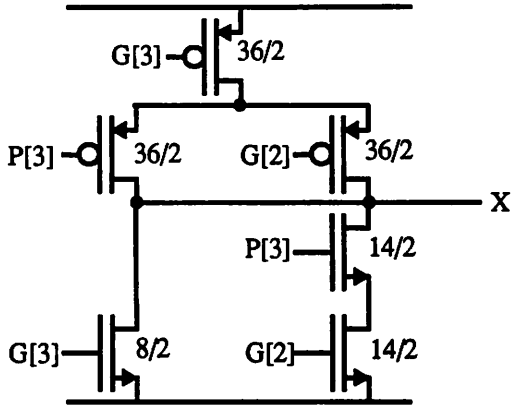


Figure 5-22. AOI Carry Generation Circuit: Term X: AOI Top

layout, and faster than doing a NAND/NAND or NAND/NOR. The delay for carry generation will be dominated by the Y term, anyway, as it is more complicated, so the devices in the X term above, although shown as large, actually can be sized smaller. The sizing was obtained by picking the NMOS propagation delay vs. width corner to be around 8λ, and then automatically scaling up from that. In reality though, as this is only a small aspect of the correlator revision which won't be fabricated anyway, size optimization was not done.

The worst case delay will be given by the case where OUT must go from 0->1 with P[3]=G[2]=0 and G[3] going from 1->0. SPICE results (loaded with 50fF, 0.7μ process) give a delay time of 1.1ns with a 10%-90% edge rate of 1.1ns. The layout is shown below.

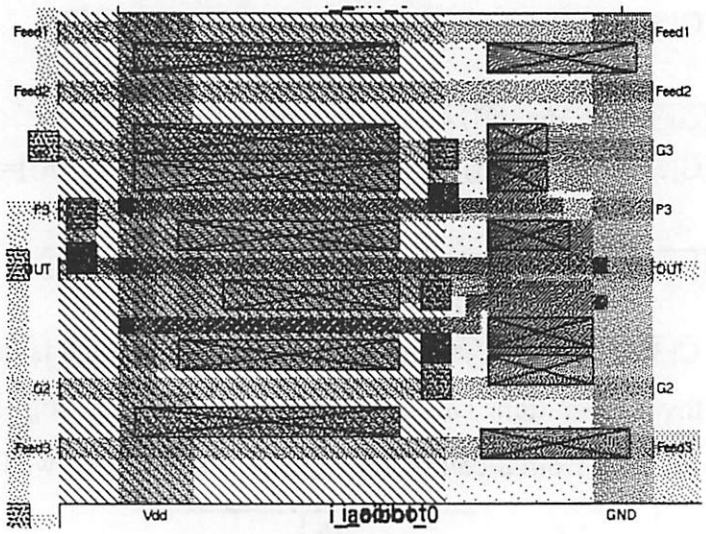


Figure 5-23. AOI Top (Term X) Layout

For the implementation of the Y term, we may be tempted to try a similar thing again, breaking it down into AOI's. By noting that:

$$Y = [\overline{P[3] \bullet P[2]} + \overline{(G[1] + P[1] \bullet G[0])}] = \overline{P[3] \bullet P[2] \bullet (G[1] + P[1] \bullet G[0])}$$
$$= \overline{(P[3] \bullet P[2]) \bullet G[1] + (P[3] \bullet P[2]) \bullet (P[1] \bullet G[0])} = \overline{A \bullet G[1] + A \bullet B}$$

where,

$A = (P[3] \bullet P[2])$, and $B = (P[1] \bullet G[0])$

Thus the Y term would become the AOI of a couple AND'ed terms. The overall delay for carry generation would then be: AND (1.2ns), AOI (1.5ns), NAND (0.8ns), or around 3.5ns (which is better than the 4ns we were trying to beat.) But, we can do better by looking at the equation for the Y term and thinking along the lines of implementing the entire Y term with one complex gate. The reason that, in general, complex gates are avoided for fast, low voltage operation is that the stacked PMOS performance degrades rather sharply for more than 3 stacked gates. However, if we look at the term:

$$Y = \overline{P[3] \bullet P[2] \bullet G[1] + P[3] \bullet P[2] \bullet P[1] \bullet G[0]}$$

Then we can note that there are only two terms in the NOR. This means that a straight-up complex gate will only have two levels of stacked PMOS. In fact, the whole Y term may be implemented with the circuit shown below (called i_aoibot). While there are 4 levels of

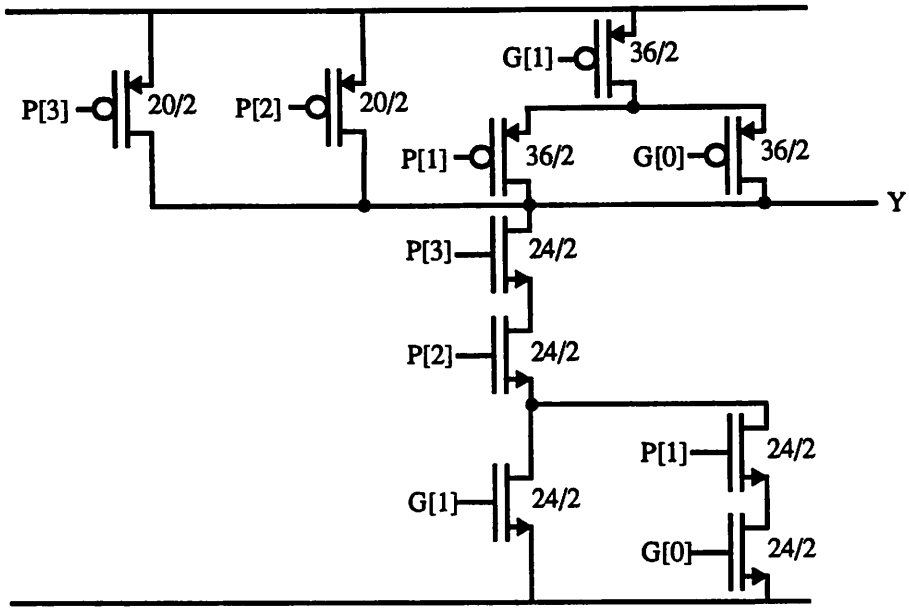


Figure 5-24. AOI Carry Generation Circuit: Term Y: AOI Bottom

stacked NMOS, their performance is as bad, if not a little better, than the PMOS's pull up. SPICE for 50fF load yields a worst case delay of about 2ns (rise time). The worst case rise is given by pulling high from G[1] with all internal nodes discharged: P[3]P[2]G[1]P[1]G[0] = 11110 -> 11010. The worst case fall is given by G[0] discharging given all of the other internal nodes are charged: 11010 -> 11011. So the overall delay for carry generation is 2ns + NAND(0.8ns) = about 3 ns! This is a full nanosecond faster than the 4ns NAND/NOR, and almost 2ns faster than a ripple carry generation (4.8ns).

While the estimation for this circuit's performance are promising, we need to simulate the worst case carry propagation to verify the results. If we look at gate loading, we see that the Fan-Out (or gate cap load) of the propagation P[i] and generation G[i] terms are a little less than the load assumed for the delay values in SPICE. Note that the load on the P[3] term is the worst, going to 60λ of gate cap (42 fF). The other terms have a similarly large load (on the order of Fan-Out=3). At the cost of adding an extra inverter (about 0.5ns) per XOR, we can convert the propagation XOR's into XNOR's (while the generation

NAND's have to be turned into AND's anyway) and buffer the output before driving the BCLA cells. This does diminish the gain from doing this complex cell approach (to be about 0.5ns), but it is more conservative. There is a good argument that it is not necessary, though, and a final evaluation was not done owing to the poor power results from the redesign. Buffering would add 0.5ns gives us about 2.3ns of estimated overhead -- several gate delays long and about 15% of the clock period, while not buffering gives 2.8ns of overhead (18%).

Thus, the over-all carry generation circuit looks like:

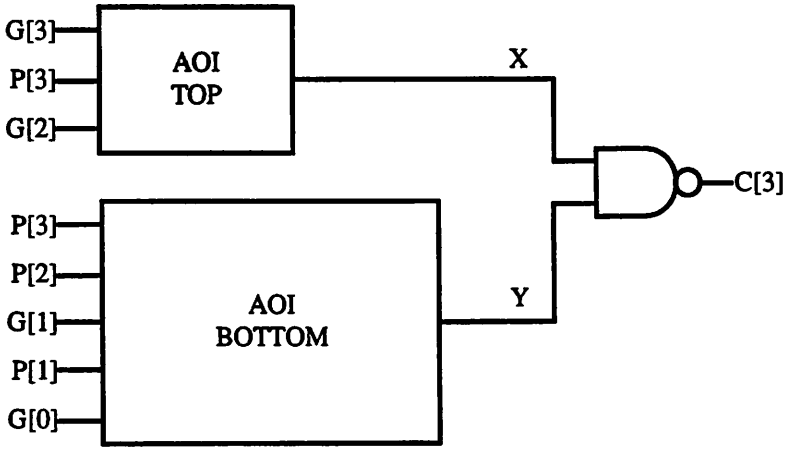


Figure 5-25. Carry[3] Generation Circuit

5.1.7. Floorplanning for Accumulator

We have designed all of the cells needed for the accumulator, now we need to lay them out in a compact fashion. Similar to the previous correlator and as mentioned in the library cell section, a DPP style (in terms of data flow) was used. The actual structure is heterogeneous and of unequal number of bitslices, so it is not truly DPP. To play around with floorplanning, I laid out the boundaries on graph paper with registers to the left and right, and popped down gates (drew boxes) to toy around with different arrangements. Perhaps not a high-tech approach, it none-the-less gave good results. The resulting tiling I came up with, after a couple attempts, is shown below. It is not very complicated and gives

tight, compact tiling without much wasted space. It is actually quite close to a literal tiling

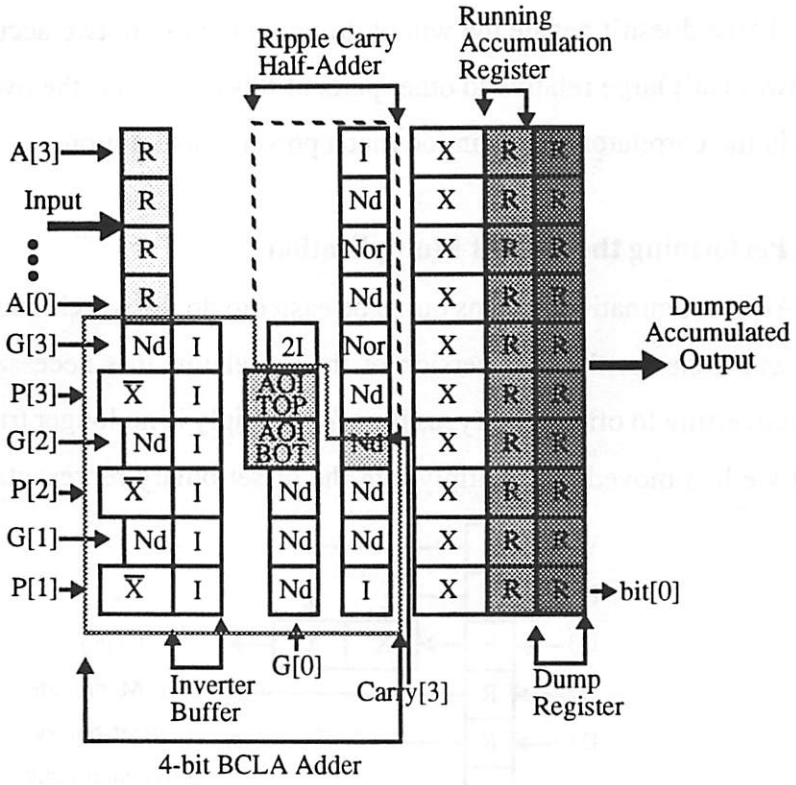


Figure 5-26. Revised Accumulator Layout

for the ripple carry adder proposed previously in Section 5.1.3.2. Note that the incoming data registers have been moved to be top-justified and that there are routing channels which weren't needed for the first correlator design, owing to its regular layout. Also, while there is a hole in the top center of the tiling (as the two inverters for B[5] and B[7] we combined into a single cell), this winds up being nicely filled by the clock buffering and control logic, creating a fairly packed rectangle -- just what you want of a digital design. If you like, you can compare the tiling to the logic diagram of Figure 5-8, "Ripple Carry Accumulator with NAND's," on page 69 (with a carry look-ahead circuit, Figure 5-25, for Carry[3]).

5.1.8. Frontend Correlator Issues

Now that we have the accumulator designed, we need to flush out the front end of the correlator. Again, we need to multiply by +/-1 for the PN and Walsh codes. In addition to that, however, we also need to convert the incoming data from sign-magnitude to offset-binary and that raises the question of which to do first, or whether to try to do them together. It also suggests, if this conversion is large or power hungry, that the offset-binary

technique may be flawed. Luckily the conversion only takes a small number of gates and the added area doesn't negate the win of dropping the use to two accumulators. While the extra power isn't large relative to other parts of this correlator, the overall sizing of all the devices in the correlator results in too much power consumption.

5.1.8.1. Performing the Weight Multiplication

After examination, it turns out to be easier to do the weight multiplication the same way as was done for the first version of the correlator. It is necessary to do the XOR's before converting to offset-binary, as the +/-1 multiply is no longer trivial in that representation.If we had moved the multiply into the offset binary representation, we could have

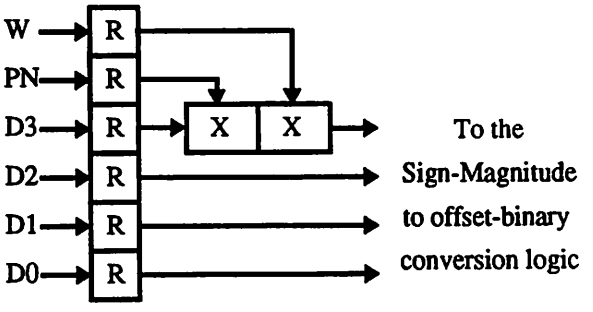


Figure 5-27. PN and Walsh Weight Multiplication

pushed the offset binary representation all the way to the A/D converter for the receiver and hence moved the conversion to a single stage at the input of the digital chip. This would be a power and complexity savings while still retaining the signal correlation, but the logic to do the multiply is of about the same size as doing the conversion itself. (See Table 5-1, "Number Representation: 4 bits," on page 60. Note that a sign-bit change in offset binary also affects the other bits and is value-dependent.) This fact nullifies the benefit of going to an all offset-binary representation. Such a change would also imply a larger redesign of the chip which, while not bad, is still more work.

5.1.8.2. Converting from Sign-Magnitude to Offset Binary

As was indicated at the beginning of the revised correlator discussion, the offset binary representation can be thought of as simply adding 2^{N-1} (for an N bit number) to the 2's complement representation, which results in an all 'positive' representation. For the 4 bit incoming data, this corresponds to an addition of $2^3 = 8$. Unfortunately the incoming

data isn't in 2's complement representation, giving us the following options (not limited to, but including):

1. Convert from Sign-Magnitude to 2's Complement, add 8, then convert to Offset Binary. Note that this is a rather in-elegant (i.e. bad) idea from a power and area perspective.
2. Design a Sign-Magnitude adder library cell; then we could add 8 to the sign-magnitude value and then convert to offset binary. Note that a simple bit-slice for such an adder doesn't seem to exist and may be rather complicated. This is a worse idea as it involves possibly much more labor for similarly bad power and area numbers.
3. Realize the Sign-Magnitude to Offset Binary conversion directly with logic; it's only 4 bits, how bad could it be? (Hint, chose this option.)

Given the Karnaugh Map for each bit, a small but glitchy, direct converter may be quickly designed. We are also able to reuse some of the accumulator logic cells, and the overall result is fast enough to fit within the same clock period as the weight multiplication, thus saving a cycle of latency over the first correlator design.The bit-by-bit conversions are given below, in subjective order of easiest to hardest to implement.

- Bit[0]

If we designate the sign-magnitude input of the conversion to be A[3:0], and the output, in offset binary, to be B[3:0], then we may note the trivial conversion for the lowest bit. Namely B[0] = A[0].

- Bit[3]

Note that, from the Karnaugh Map:

$$B[3] = \overline{A[2]}\bullet\overline{A[1]}\bullet\overline{A[0]} + \overline{A[3]} = \overline{A[3]\bullet A[2] + A[3]\bullet A[1] + A[3]\bullet A[0]}$$
$$= \overline{A[3]\bullet(A[2] + A[1] + A[0])}$$

This handles the input of +0 and -0, casting them both to 8. If we had ignored the negative 0 input (which we should theoretically never see), this would reduce $B[3]=\overline{A[3]}$. However, in the interest of robust, correct operation, we will treat the case in an intelligent manner, in case it should ever somehow come up the correlator will do the right thing. From the above equations, it is possible to realize B[3] without needing the inverse of any of the input bits A[3:0]. This is a nice feature that we will be able to preserve for all of the bits in the conversion.

There are two immediate choices for implementation: monolithic (note that this implies at most three stacked PMOS), or by using library cells (OR'ing A[2:0] and NAND'ing the result with A[3]). Note that the monolithic circuit will be smaller (it can fit into a single cell) and perhaps faster, but have worse edge rates. Also, it won't glitch due to unbalanced delay paths as the library version will. If we examine power, by assuming the input probability of change is independently 1/2 per cycle and by counting drain/source cap as approximately equal to gate cap, then the effective switched capacitance (α*Cnode) for the two version winds up being about equal neglecting glitching and lack of full swing on internal nodes. Thus, the monolithic choice was made to save on area, and the cell called "i_smag2off3" was created. SPICE results show the cell has a worst case delay (rise time)

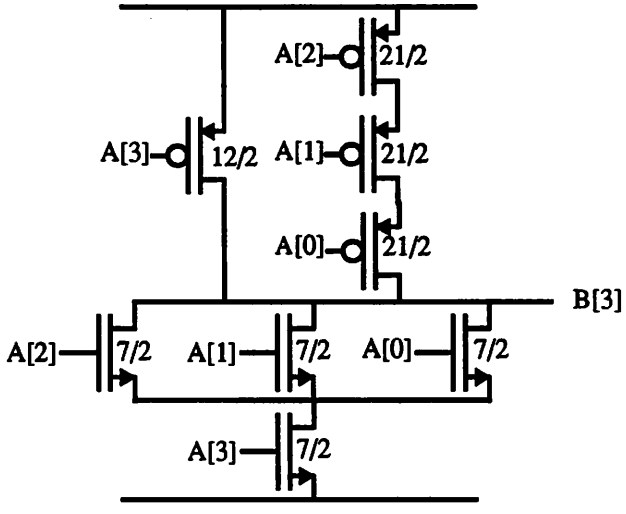


Figure 5-28. Sign-Magnitude to Offset-Binary Conversion Circuit: Bit[3]

of about 2.5ns with a 2ns edge (50fF load, 1.5V, 0.7μ).

- Bit[1]

  From the Karnaugh Map,

$$B[1] = \overline{A[3]} \bullet A[1] + A[1] \bullet \overline{A[0]} + A[3] \bullet \overline{A[1]} \bullet A[0] = A[1] \oplus (A[3] \bullet A[0])$$

Another way to interpret this bit is to think of B[1] being A[1]$\oplus$A[0] if A[3]=1 (the number is negative), otherwise B[1] is simply A[1]. Again this functionality can be built as either a collection of library cells, or as a monolithic complex gate, however, in this case the complex gate will require inverses of the inputs, will be stacked three PMOS deep (and hence have slow edges) and could be too large for comfortable routing in one cell, so it was

avoided. In contrast, a library implementation only needs two gates: Either a NAND/XNOR to directly implement the Boolean equation, or an XOR/MUX to choose B[1] based on A[3] as discussed above. Both approaches will glitch and both are simple and small, but the NAND/XNOR will be faster (although the extra speed is not critical it could be used to size down the devices), so it was chosen.

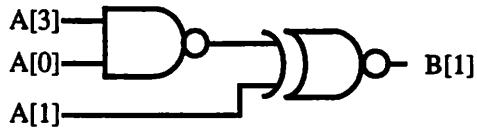


Figure 5-29. Sign-Magnitude to Offset-Binary Conversion Circuit: Bit[1]

- Bit[2]

$$B[2] = \overline{A[3]} \bullet A[2] + A[3] \bullet \overline{A[2]} \bullet A[1] + A[2] \bullet \overline{A[1]} \bullet \overline{A[0]} + A[3] \bullet \overline{A[2]} \bullet A[0]$$

which can be rearranged into:

$$B[2] = \overline{A[3]} \bullet A[2] + A[3] \bullet (\overline{A[2] \oplus Z}), \text{ where } Z = (\overline{A[1] + A[0]})$$

This is a multiplex operation again, B[2] is A[2] if the number is positive (A[3] is low), otherwise, B[2] is given by an XNOR or a NOR. Owing to the imbedded XOR, there doesn't seem to be a much better interpretation of the Karnaugh Map. A monolithic circuit is not pursued as it would require inverses and have too many stacked PMOS. To do the multiplexing operation, an AOISEL cell was designed (this is an AND-OR-Invert library cell, show below, that is easily convertible into a mux, by using a select line). SPICE shows

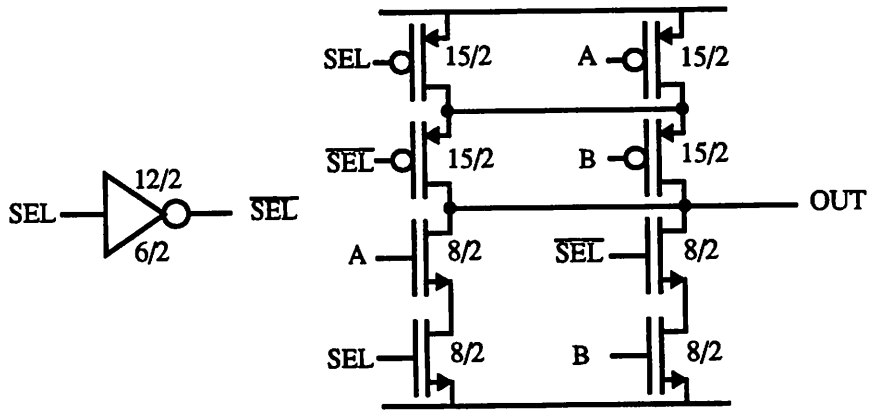


Figure 5-31. AOISEL Cell for Bit[2] Sign-Magnitude to Offset Binary Conversion

the worst case delay of the AOISEL cell to be about 1.7ns with a 3ns worst case rising edge (1.5V, 0.7μ with 50fF load).

Using the AOISEL cell, we can realize Bit[2] with the circuit below. For better

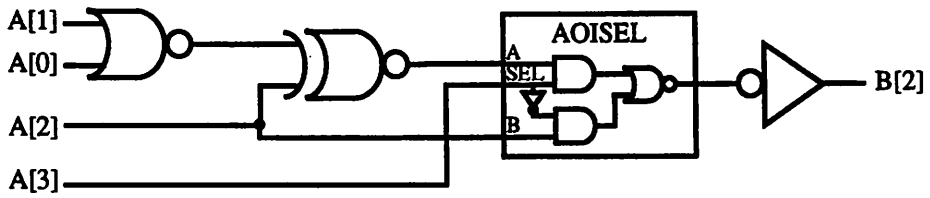


Figure 5-32. Sign-Magnitude to Offset-Binary Conversion Circuit: Bit[2]

packing efficiency, the inverter at the end was included in the NOR gate (creating a cell called i_2nor_inv) which has both independent gates.

Now with all of these designed we can tile up the front of the correlator. Note that

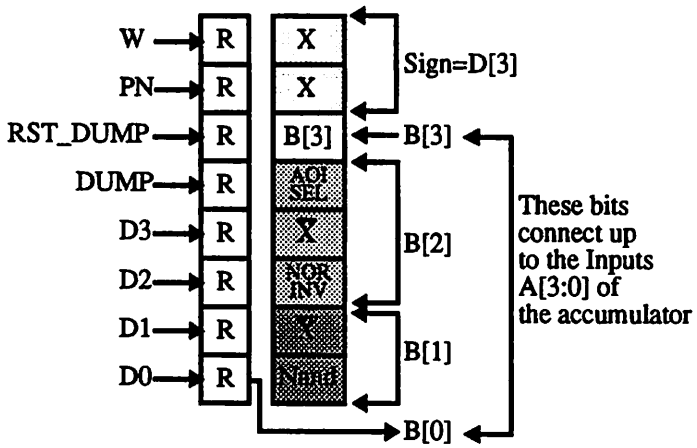


Figure 5-33. Revised Correlator Number Conversion and Weight Multiplication

the critical path for this logic is from TSPCR to Sign (2 XOR's + Clk2Q = 3.4ns), then for a NOR(1.2ns), XNOR(1.5ns), AOISEL, and INV (about 3ns for both) plus setup time (0.8ns) for a total of 10ns, which is well below the 15.6ns period. This is a plus also because is allows us to remove a pipeline delay by grouping the conversion with the weight multiplication XOR's. This extra time could also be used to size down the devices not on the critical path to save on power, but again this was not done as it would only marginally improve the overall power consumption of the redesigned correlator.

5.1.9. Clocking and Control for the Revised Correlator

With all of the work so far, the correlator is nearly redesigned. All that remain are the issues of providing clocking and control, and the backend reconversion to sign-magnitude. Thankfully, as there is only one accumulator to clock, all of the registers are always clocking and only a minimal amount of control is needed for resetting and dumping. Also, a nice feature of the design is that it turns out to be very easy to balance the loads. A block diagram of the database and control is show below. Note that registers which have the same

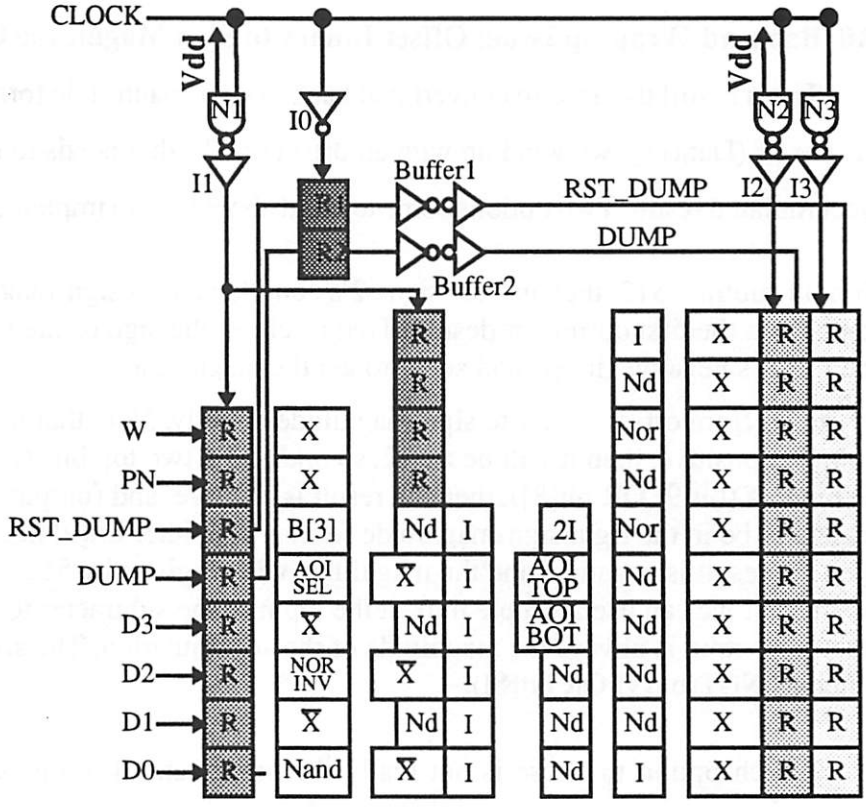


Figure 5-34. Control and Clocking for Revised Correlator

clock are the same shade. The load divides easily into three clocking regions: the first with 12 registers, and the other two with 10 registers each. At about 33fF of clock load each, this makes the loads 400fF, 330fF, and 330fF which are already nearly balanced. In terms of skew, the only expected skew will be between the first clock and the other two. The impact of this skew is to impact the critical path, but in addition there is a hard constraint to prevent a race condition. Namely, the skew: $(\Delta_1-\Delta_2)$ or $(\Delta_1-\Delta_3)$ must be less than the fastest path for the accumulator, which is at least a T_{clk2Q} (1.5ns). Guaranteeing, this condition for the above load is an easy thing to do. The inverter driver for the first clock is sized

up a bit relative to the other two, and the result SPICE'd for the estimated load to verify the skew much less than 1.5ns.

Again, gated clocks are used for control, and the reset lines are clocked on the falling edge of the clock to ensure adequate setup time before the next rising edge. As the estimated load for a reset input is about 7fF/cell, the overall load is only 70fF which is small and easily driven in time to meet this constraint.

5.1.10. Backend Wrap-up Issue: Offset-Binary to Sign Magnitude Conversion

There is still the need to convert that back to sign-magnitude format. After 64 accumulations of (Data+8), we wind up with an offset of 512 that needs to be subtracted from the accumulated result. Two options come to mind about how to implement the conversion:

1. Simply subtract 512, then use the same 2's complement to sign-magnitude conversion logic from the first correlator design. That is, check the sign of the fixed number subtract. If it's negative, invert and add 1 to get the magnitude.
2. Try to go from offset binary to sign-magnitude directly. Note that if the accumulated output is positive, then it will be > 512, so one of the two top bits (bit[9] or bit[8]) will be high. If (bit[9] OR bit[8]), then the result is positive, and (output - 512) will automatically be in the right sign-magnitude format. If neither bit[9] nor bit[8] is asserted, then the result is negative, and the magnitude will be given by 512 - output. Thus it seems that we can use a simple mux at the input of the subtracter to guarantee the subtracter's output is always the magnitude of the accumulation. The sign is easily computed as NOT(bit[9] OR bit[8]).

Which option to chose is not really that important. The clocking rate is 1MHz, which is certainly slow enough for either scheme, although option 1 has a longer critical path for rippling, it should be easily met. Power shouldn't be a big issue at this clock rate; this is not a significant contribution to the power of the correlator. In terms of area, the first approach is only a little bit larger than the second approach. One could optimize the subtract into a half-subtracter to perhaps recapture that area loss since we are always subtracting a fixed quantity, but it's hardly worth the effort. Since the overall decision is not that important, I chose option 2 because I felt it was a little more interesting than option 1. A

diagram for the circuit is shown below. (Note that the Sign and Magnitude values are buff-

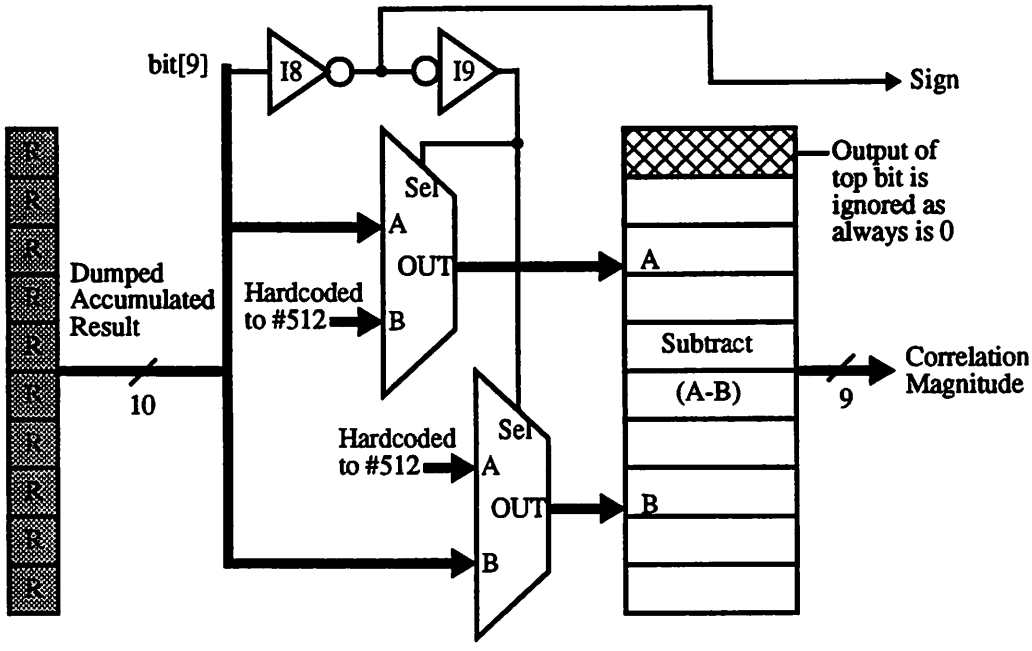


Figure 5-35. Offset Binary to Sign-Magnitude Conversion

ered as the last stage at the output of the correlator.

5.1.11. Power Estimation for the Correlator

At the beginning of this redesign we estimated that we could shave off about 40% of the correlator power due to the improvements in design and technology that are available. Now that we have the revised correlator designed, we can make some better hand estimations based on the reduction of registers, and the power savings due to process miniaturization.

Of course the correlator was simulated with random inputs (as well as constant inputs), using IRSIM-CAP and PowerMill, and the projected power was 3x worse. (Process 0.8μ, 1.5V) The actual correlator has yet to be fabricated so these results are not verified by measurements. It is, In fact, unlikely that it will ever be fabricated in its current design state owing to these results. The explanation for this outcome has been mentioned before. As many devices were sized up at least 3x (12λ/2λ minimum NMOS instead of 4λ/2λ) across the board, we would expect the power to be at least 3 times the 60% projected power, or about 2x bigger. It is lamentable that things got this far before this fact was

uncovered, but it serves to illustrate the importance of understanding the implications of an architecture's approach.

5.1.12. Final Library Issues: ir_frontend.mag (Revised Design)

All of the appropriate cells were not grouped into a library directory or made into an OCT style library because of the power simulation results. The new correlator cell has an 'r' prepended to indicate that it is the revised correlator design.

The functionality of the correlator is not the same so the same VHDL code may not be used for functional simulations. In addition to losing a pipeline delay, this correlator has a nasty properly of negatively biasing all correlations less than 64 cycles (by 8*# cycles fewer than 64) and positively biasing all correlations longer than 64 cycles (since we hard-coded the offset subtraction). This shouldn't happen in the system, but that behaviour should be represented in the VHDL. The only way to make this correlator more versatile for other schemes is to include a counter to count the number of samples currently accumulated. The counter output, or a writable register, upon a dump may be fed into the MUX for the subtract to allow for more general operation. This is extra power, area, and control, of course, and further argues against the use of this number representation.

5.1.13. Conclusion for Revised Design

This cell will probably never be used, owing to the power simulation results and the system integration issues, but at least can serve as a learning exercise. It points to a better way to do low power design. Namely, start with near minimum sized devices and pipeline/ parallelize to meet throughput. If the correlator is to be redesigned for a third time, I suggest a simple scaling of the transistors in the first design to as near minimum size as possible. If all may be scaled to minimum size, perhaps some carry registers may be removed to further lessen area and power.

5.1.14. Layout: ir_frontend.mag (Revised Design)

(Backend conversion layout in light grey, registers in dark grey.)

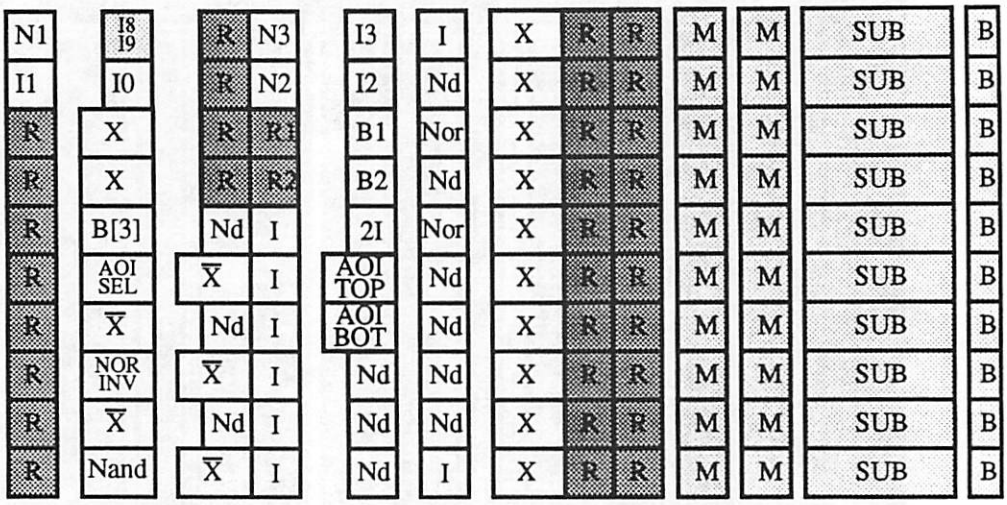


Figure 5-36. Cell Tiling of the Revised Correlator

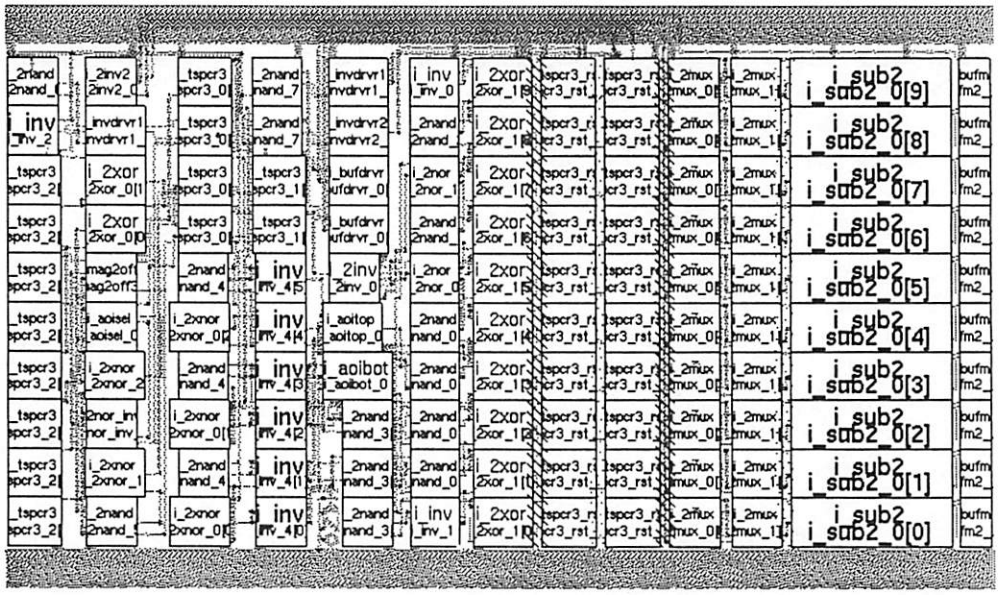


Figure 5-37. Layout of ir_frontend.mag (by cells)

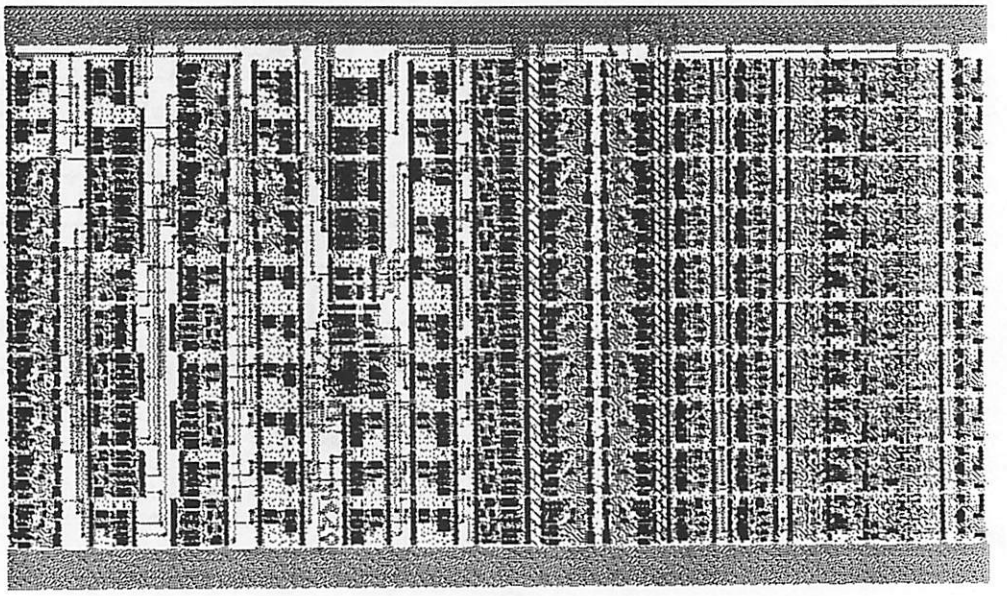


Figure 5-38. Layout of ir_frontend.mag (fully expanded)

6 DQPSK Design

The symbol stream coming out of the data correlator is encoded by Differential Quadature Phase Shift Keying to allow for incoherent demodulation [Sheng96] and must be decoded to obtain the user's information. In the first version of the digital backend chip, this decoding was not implemented due to time constraints. A decoding scheme is presented in [Stone95] that uses simple thresholding to slice the data into four quadrants for decoding. This is technique is attractive from a power and area perspective, owing to its simplicity and small amount of hardware. It was determined, though, that more information than simply the output bits would be desirable for evaluating the radio performance [Teuscher95]. Hence, the magnitude and phase information of the slice, in addition to the output bits, are provided. A straight-forward low power design to realize this functionality is presented in this chapter.

6.1. Brief Review of DQPSK Coding

Recall that DQPSK converts 2 bits of user data into a complex signal according to the following table [Proakis]:

add Δ_{PHASE} to symbol	0°	90°	180°	270°
for [Bit_{N+1} Bit_N]	0 0	0 1	1 1	1 0

Table 6-1. DQPSK Encoding

For decoding, we want to look at the Δ_{PHASE} difference (the angle) between two consecutive received symbols. As we receive the accumulated in phase and quadrature cor-

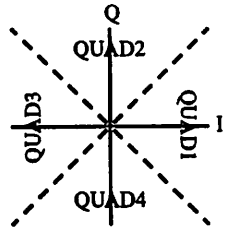
relation results, we can decode the symbol (S = I + j*Q) by looking at $\angle (S_N/S_{N-1})$. Note that as we are only interested in the angle, we can find that from:

$$\angle\left(\frac{S_N}{S_{N-1}}\right) = \tan^{-1}\left(\frac{I_{N-1}Q_N - I_N Q_{N-1}}{I_N I_{N-1} + Q_N Q_{N-1}}\right)$$

[Stone95] proposes to examine the magnitude and sign of the numerator and denominator to determine which quadrant the data is in. This necessitates the computation of four scalar multiplications at the symbol rate (1MHz) and the slicing conditions are shown below [Stone95].

:

Quadrant	Phase Difference	Check	Output (Bit_{n+1}, Bit_n)
1	$-45° < \Delta_\theta \leq 45°$	|Re| > |Img|, Re+	(0,0)
2	$45° < \Delta_\theta \leq 135°$	|Img| > |Re|, Img+	(0,1)
3	$135° < \Delta_\theta \leq 225°$	|Img| > |Re|, Img-	(1,1)
4	$-135° < \Delta_\theta \leq -45°$	|Re| > |Img|, Re-	(1,0)



Note: $Re = I_n I_{n-1} + Q_n Q_{n-1}$ and $Im = I_{n-1} Q_n - I_n Q_{n-1}$.

Table 6-2. DQPSK Slicer Decoding Conditions

Doing an actual implementation of the inverse tangent function (perhaps with a cordic algorithm and a divider) is much more work than needs to be done in hardware. It is sufficient, for observability, to provide the numerator and denominator values which may be post-processed for more accurate examination of the data constellation.

6.2. Multiplier Examination

Multiplier architectures for low power and low area implementations were examined by myself and Dennis Yee to determine the appropriate structure for the decoder [O'Donnell, Yee 241]. The input of the multiplier was taken to be two 10-bit sign-magnitude numbers: 1 bit sign, 9 bits of magnitude each for a 9x9 bit magnitude multiply with the resulting sign being an XOR computation. As the speed is very low, 1MHz, the main

focus was upon area and power as metrics and library cells were used as opposed to full custom design.

The question of low power may be attacked from many different levels of hierarchy ranging from the system level through algorithms and circuits to the process level with many viable techniques. To help bound the optimization problem we chose to assume that we are already reducing the power supply to 1.5V, using the low power library cells, and that we were building separate multiplier cells (although shared cell structures are examined). By normalizing the operational voltage, process, and library cells we can better isolate and compare the trade-offs intrinsic to the multiplier algorithms and their respective implementations. The main issue of low power design then concentrates on the amount of capacitance switched per multiply which may be analyzed by hand from a block diagram of the implementation given an estimate of the statistical power consumption per library cell.

Area was also considered to be an important parameter owing to the need to instantiate many multipliers to perform the desired parallel computation. Based on the requirements for area and power, we proposed that a sequential multiply algorithm would offer the best balance for implementation.

6.2.1. Sequential Multiplier Algorithms/Implementations

At its heart, the basic algorithm for sequential multiplication involves adding a shifted multiplicand (partial product) to a running accumulation based on an iterative scanning of the multiplier. There are two basic methods to scan the multiplier's bits: a direct method involving only addition of shifted partial products, and a modified Booth method (recoding) which also allows subtraction. One may also choose the number of simultaneous bits to scan from 1 bit per iteration to a full parallel scan, i.e. an array multiplier.

It is not sufficient to just consider the algorithm, though. The implementation of the algorithm is also necessarily as important and must be considered. For example: A 2-bit scanned direct method multiplier needs to accumulate 0, 1, 2, and 3 times the multiplicand. This may be achieved either by taking an extra clock cycle to compute 3 times, then multiplexing the correct partial product to the running sum (which is slightly more power). Or

we could apply 1 and 2 times the multiplicand in a binary weighted fashion to an extra 2-input adder whose result is the desired partial product (which is slightly more area). Of these two possibilities, it is not immediately clear which is the better choice for implementation. For each algorithm we tried to determine the 'smartest' implementation in terms of area and power. With a 1-bit scan, an overzealous designer may implement a 9x9 bit multiplier as an 18-bit 2-input adder with a shifter, while in reality all that is necessary is a 9-bit 2-input adder and a hardwired shift operation. At any given time only 9 bits of sum (plus a carry) need to be computed when adding the 9-bit partial product to the 18 bit running sum, so using 18-bits and a shifter is wasteful of area and power. In fact only 17 bits of running sum storage are needed also, as the final 18-bit sum may be latched at the end of the last addition. Another important implementation detail involves Booth recoding which requires a subtraction. Rather than sign-extending over the entire running sum length, it is possible to use a modified sign-extension technique when shifting the running sum to the right. This requires many fewer adder cells and does not have as much wasteful activity in the sign bits.

In general the 'smartest' implementation for a multiple bit scan (direct method) involves using extra 2-input adders to sum up binary weighted multiples. Each scanned bit AND's the multiplicand, contributing either 2^n times through hardwired shifting or 0 times to the partial product. This is sometimes called "redundant multiples" as you wind up adding 0 to 1 to get 1 (which is redundant). Pre-computing multiples with extra clock cycles, the other method of obtaining partial products for the direct method, becomes wasteful as the number of bits scanned increases as too many partial products which may not be used are computed. For the Booth method, the use of partial product pre-computation and modified sign-extension turned out to be the 'smartest' implementation.

6.2.2. Characterization of Library Cells

The library cells used for implementation were those of the U.C. Berkeley Lager Low Power DPP Library. This library contains minimum to near-minimum sized devices intended for low voltage operation. Having devices that are no bigger than necessary to meet the speed requirements, 10-bit ripple in 100ns for our case, ensures that the circuits

are not overly wasteful in power or area. The same cells were used for all implementations to allow for an unbiased comparison. One point to note is that while the library is good for low power, it does not contain the best optimized cells for our design. The register used was TSPC which is convenient for design, but may be more power than necessary compared to a pass-gate/inverter dynamic latch. Exploration of circuit and cell level optimizations were not performed due to time constraints, but we did wind up redesigning the DPP full adder bit slice, simplifying the design and reducing device size, to eliminate excessive power consumption from glitching and carry propagation. This did not change any relative measurements of the implementations but did reduce the cost per addition by cutting waste. The most important cells for a multiplier are generally the register and full-adder cells, as they are nearly ubiquitous.

PowerMill was chosen as the simulation tool over IRSIM and SPICE as it is more accurate than IRSIM and faster than SPICE. IRSIM tends to misjudge capacitance switching at internal nodes that do not experience a full voltage swing and exacerbates glitching by faster-than-reality signal propagation. SPICE was too slow to run many hundreds of vectors through for a statistical outcome. IRSIM did verify the relative trends of power consumption for each implementation reported by PowerMill, but predicted far more capacitive activity.

In general an empirical approach was taken to measure each cell's activity. Rather than try to compute the probability of node transition multiplied by its capacitance ratioed by its swing to full swing, the amount of switched capacitance was determined by feeding random inputs into a tiling of 9 bit slices of that cell, each loaded with an AND (with the other input grounded) at its output. The random vector files were generated from a C program and were applied at a 1MHz rate in PowerMill. Logic cells, adders, and TSPC registers were simply given random inputs on all input data terminals at a 1MHz rate, whereas the scan register was given random inputs at 1MHz rate, but scanned at 9MHz. The resulting average current from PowerMill was normalized to a per bit slice per MHz case so that the values could be easily scaled for block level estimation. Based upon examination of several sequential multipliers, this seems like a reasonable method for estimating power although it does not take uneven arrival times or glitching inputs into account.

Area estimates were calculated in units of lambda (SCMOS design rules) and measured to be the cell size, plus a 12 lambda routing channel.

Cell	Area ($k\lambda^2$)	I_{avg} (nA/bit) (1MHz)	Cell	Area ($k\lambda^2$)	I_{avg} (nA/bit) (1MHz)
Full Adder	14.4	108	Inverter	3.3	16.7
Full Adder (Redesign)	11.8	62	2-Input Mux	4.5	48
TSPC Register	4.5	39	2-Input Nand/And	4.3	17.7
Scan TSPC Reg	6.4	37.6	2-Input Nor	4.1	5
2 Input XNOR	4.5	25	2-Input XOR	4.5	26

Table 6-3. Library Cell Characterization

A design was evaluated for area by adding up all cells' area estimates. Power was estimated by adding up the contribution of each cell multiplied by: the number of cell's present multiplied by the number of times it was clocked/had an input applied during the 1MHz period.

In general this empirical characterization was in good agreement with the PowerMill simulation results as it tended to model the actual operation of a sequential multiplier. However, we discovered that glitching due to unmodeled dynamics resulted in an overly optimistic estimate of power for Booth recoding. The sign extension due to subtraction for Booth recoding resulted in a large amount of unpredicted glitching caused by uneven arrival times of the carry input versus the bit inverted adder input, and the nature of the original library adder cell. The original adder cell is faster than needed, allowing glitching to progress further, and contains both Carry and $\overline{\text{Carry}}$ which always creates activity regardless of Carry's state. By removing the dual rail carry generation path, and inserting a simple AND-OR-INVERT for carry generation (carry if any two inputs of {A, B, Cin} are high) a 40% savings in area and power were realized for the adder cell. The redesigned cell had a critical path (at 1.5V) of 8.5nS relative to the 3.3nS of the original adder cell which is adequate for our performance requirements. With the slower adder cell, excess glitching activity in the Booth was reduced by 1/3, and glitching overall due to a full gate delay between input arrivals was consistently around 20% larger than the above estimate.

If the mismatch of inputs had been much larger than a cell delay, we could have estimated the power as the activity for two sets of computations

The Full Adder cell redesign is shown below. Note that $T_{critical}$ for the bitslice is 8.5ns compared to 3.3ns for the original library cell in the 0.8μ technology.

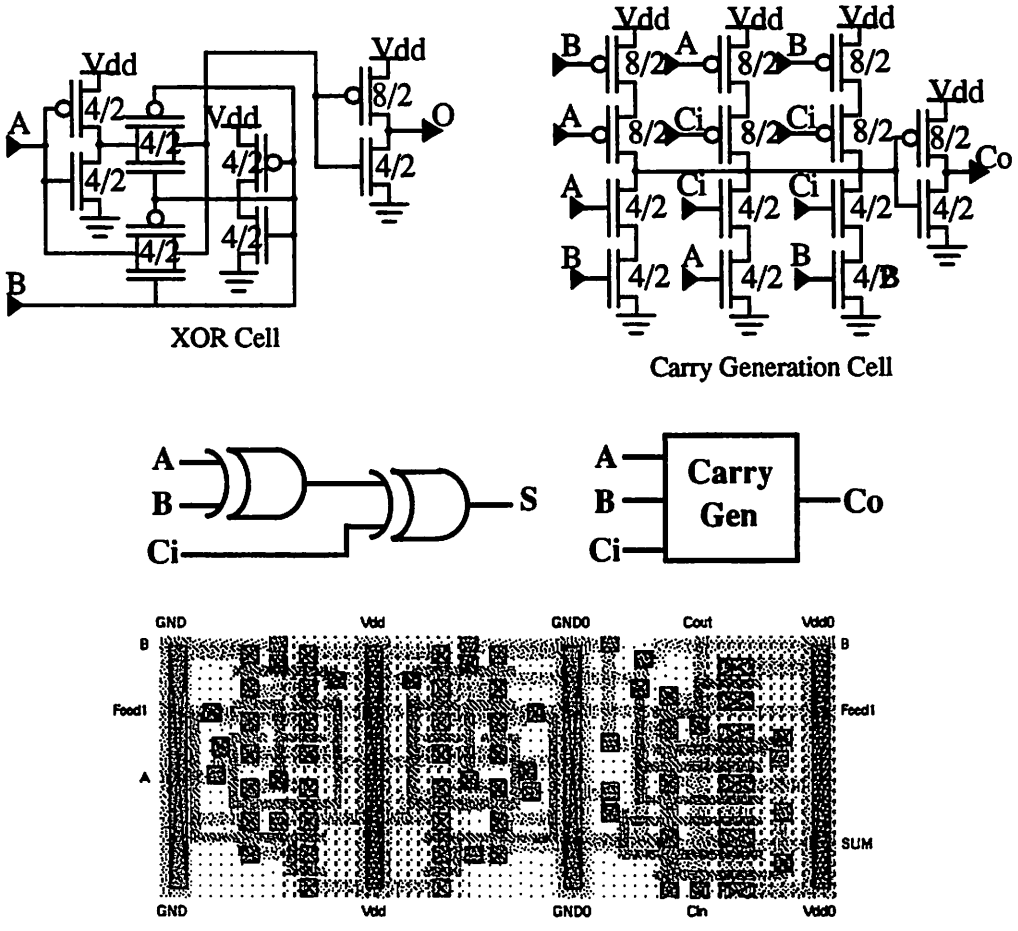


Figure 6-1. Library Full Adder Cell Redesign

6.2.3. Algorithmic Considerations for Power

The total power consumption consists of contributions from various hardware components: adders, registers, and miscellaneous logic gates. At the algorithmic level it is possible to analyze the power consumption by determining the number of required additions. A large number of additions results in greater adder activity and thus more power consumption. Therefore, a low power implementation should be based on an algorithm which minimizes the number of additions without significantly increasing the number of registers

and miscellaneous logic gates. The most basic algorithm for multiplication is the add and shift method. An implementation based on this algorithm for the multiplication of two unsigned binary numbers X and Y which are both M bits wide requires the addition of M partial products. Decreasing the number of partial products may be achieved by scanning multiple bits simultaneously. Table 6-4 summarizes the number of required additions for several cases of multiple bit scanning for multiplication of two binary numbers which are both nine bits wide.

	1-bit scan	2-bit scan	3-bit scan	4-bit scan
no recoding	9	6	6	10
Booth recoding	9	5	5	6

Table 6-4. Number of Required Additions

Although higher order bit scanning decreases the number of partial products, extra additions are required to generate multiples which are not powers of two. As seen in Table 6-4, for a nine bit multiplier the number of required additions is equal for 2-bit scan and for 3-bit scan. Also higher order bit scanning introduces increased complexity in generating the required partial products. This increased complexity results in additional area and power consumption due to the additional use of miscellaneous logic gates. Based on the number of required additions alone, simple 2-bit and 3-bit scanning and Booth recoding with 2-bit and 3-bit scanning appear to be algorithms which may result in implementations requiring the lowest power.

6.2.4. Sequential Multiplier Power and Area Estimates

Using the estimated values of power and area for the individual library cells, it is possible to approximate the total area and power consumption of a particular multiplier implementation without requiring a complete layout. Three cases for multiplication of two binary numbers which are both nine bits wide are considered: multiple bit scanning using redundant multiples; multiple bit scanning with pre-computation of partial products; and multiple bit scanning with Booth recoding. Table 6-5, Table 6-6, and Table 6-7 summarize the hardware usage and clocking rates required for the three cases.

	1-bit scan	2-bit scan	3-bit scan	n-bit scan
TSPCR registers	27 @ 1MHz 18 @ 9MHz	27 @ 1MHz 20 @ 5MHz	27 @ 1MHz 19 @ 3MHz	27 @ 1MHz 20 @ [9/n]MHz
SCAN registers	9 @ 9MHz	9 @ 5MHz	9 @ 3MHz	9 @ [9/n]MHz
full adders	9 @ 9MHz	20 @ 5MHz	30 @ 3MHz	10n @ [9/n]MHz
miscellaneous logic gates	9 @ 9MHz	18 @ 5MHz	27 @ 3MHz	9n @ [9/n]MHz
total average current	16.9μA	14.5μA	11.3μA	$\left(8.1+\frac{10.1}{n}\right)\mu A$
current due to registers	61.8%	46.0%	38.1%	$\frac{1.1+\frac{10.1}{n}}{8.1+\frac{10.1}{n}}\times 100\%$
total area	$405k\lambda^2$	$583k\lambda^2$	$735k\lambda^2$	$(269+157n)k\lambda^2$

Table 6-5. Power and Area Estimates for Multiple Bit Scanning with Redundant Multiples

	1-bit scan	2-bit scan	3-bit scan
TSPCR registers	27 @ 1MHz 18 @ 9MHz	27 @ 1MHz 20 @ 5MHz	27 @ 1MHz 19 @ 3MHz
SCAN registers	9 @ 9MHz	9 @ 5MHz	9 @ 3MHz
full adders	9 @ 9MHz	11 @ 5MHz 9 @ 1MHz	12 @ 3MHz 28 @ 1MHz
miscellaneous logic gates	9 @ 9MHz	60 @ 5MHz	180 @ 3MHz
total average current	16.9μA	15.7μA	17.3μA
current due to registers	61.8%	42.3%	24.9%
total area	$405k\lambda^2$	$761k\lambda^2$	$1498k\lambda^2$

Table 6-6. Power and Area Estimates for Multiple Bit Scanning with Pre-Calculation

For multiple bit scanning using redundant multiples, power decreases for higher orders of scanning. For the cases of multiple bit scanning with pre-computation of partial products and multiple bit scanning with Booth recoding, 2-bit scanning results in the lowest estimate of power consumption. For all three cases, area increases for higher order

	1-bit scan	2-bit scan	3-bit scan
TSPCR registers	27 @ 1MHz 18 @ 9MHz	27 @ 1MHz 21 @ 5MHz	27 @ 1MHz 24 @ 4MHz
SCAN registers	9 @ 9MHz	10 @ 5MHz	10 @ 4MHz
full adders	9 @ 9MHz	11 @ 5MHz	12 @ 4MHz 9 @ 1MHz
miscellaneous logic gates	9 @ 9MHz	60 @ 5MHz	106 @ 4MHz
total average current	16.9μA	15.1μA	17.5μA
current due to registers	61.8%	47.0%	36.2%
total area	$405k\lambda^2$	$624k\lambda^2$	$994k\lambda^2$

Table 6-7. Power and Area Estimates for Multiple Bit Scanning with Booth Recoding

scanning. In order to determine the implementation which results in the best compromise between power and area, the following rather arbitrary metric is used: $\text{AREA} \times \text{POWER}^2$

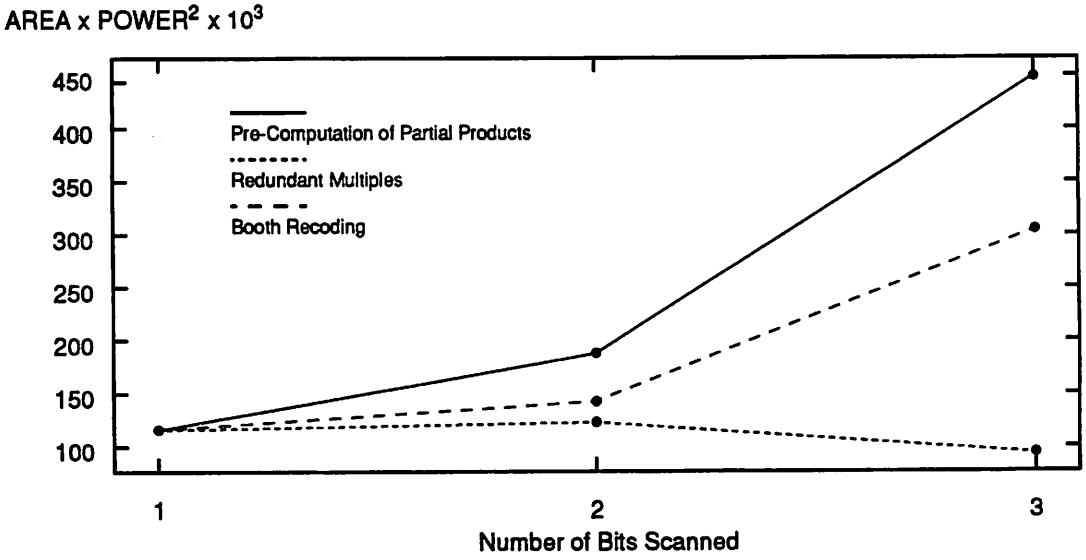


Figure 6-2. Area and Power Efficiency for Several Implementations

Figure 6-2 suggests that 3-bit scanning using redundant multiples results in the best compromise between power and area. In order to verify that this is indeed the best case, we analyzed an approximate equation for n-bit scanning to find the optimum for our metric.

$$\text{AREA} \times \text{POWER}^2 = \left(8.1 + \frac{10.1}{n}\right)^2 (269 + 157n)$$

The minimum of the above expression occurs at n=2.78. Since the number of bits scanned must be an integer, 3-bit scanning using redundant multiples is indeed the implementation which results in the best compromise between power and area.

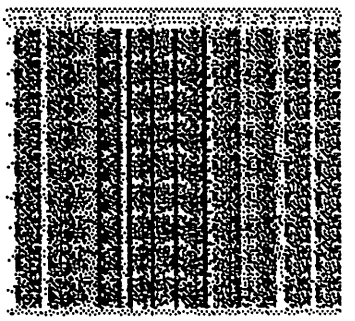
An implementation which uses 2-bit scanning with redundant multiples is actually not as power and area efficient as one which uses 1-bit scanning (the add and shift method). When performing a 2-bit scan on a 9-bit number, it is necessary to pair one of the bits with zero. Such a pairing results in inefficient hardware utilization. In order to avoid such inefficiencies, it is best to scan by a number of bits, n, such that the total number bits is a multiple of n. Thus, 1-bit and 3-bit scanning both result in better power and area efficiency for a 9x9-bit multiplier.

Of the three cases, multiple bit scanning with pre-computation of partial products gives the worst result. Although the area and power consumption of the implementation using 2-bit scanning is comparable to that of the other implementations using 2-bit scanning, the implementation using 3-bit scanning is increasingly worse due to the large area utilization. This is due mainly to the large number of full adders as well as miscellaneous logic gates required to precompute multiples of the multiplicand.

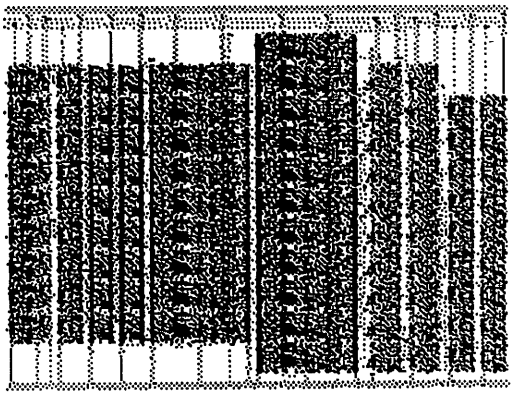
6.2.5. Sequential Multiplier Results and Discussion

In order to verify the accuracy of the above estimates several implementations were designed and laid out using Magic: simple add and shift (1-bit scanning); 2-bit and 3-bit scanning using redundant multiples; and 2-bit scanning with Booth recoding. Implementations employing multiple bit scanning with pre-computation of partial products were not

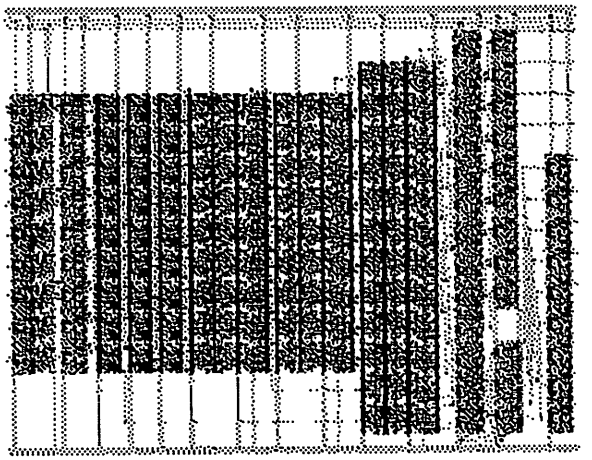
considered due to their poor performance. Magic layout of the four cases appear in Figure 6-3 below.



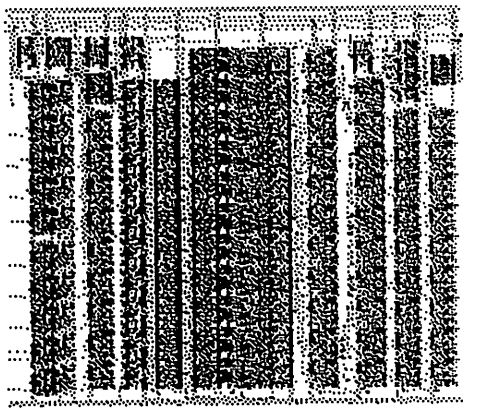
1-bit Scan (Shift and Add)



2-bit Scan (Redundant Multiples)



3-bit Scan (Redundant Multiples)



2-bit Scan (Booth)

Figure 6-3. Multiplier Layout

The values of area and power are summarized in Table 6-8.

For the add and shift implementation as well as the case using 2-bit scanning with Booth recoding, the actual areas are within 10% of the estimated values. For the two cases employing redundant multiples, the actual areas are significantly greater than the corresponding estimated values. As seen in Figure 6-3 there are significant portions of unused area in the two layouts. More efficient layouts may be achieved at the expense of more time and more complex wiring.

implementation	estimated area	actual area	estimated current	actual current
add and shift	$405k\lambda^2$	$431k\lambda^2$	16.9μA	17.2μA
2-bit scanning (redundant multiples)	$583k\lambda^2$	$738k\lambda^2$	14.5μA	14.3μA
3-bit scanning (redundant multiples)	$735k\lambda^2$	$973k\lambda^2$	11.3μA	11.2μA
2-bit scanning (Booth recoding)	$624k\lambda^2$	$698k\lambda^2$	15.1μA	17.5μA

Table 6-8. Actual and Estimated Power and Area Values

Except for the case involving Booth recoding, the actual values of power consumption agree very well with the estimated values. Simulation of the implementation using Booth recoding in PowerMill revealed a tremendous amount of glitching in the adder. An attempt to equalize the arrival times of data to the adder by introducing a set of registers resulted in little change. In this case, the decrease in power due to reduced glitching in the adder is accompanied by an equal increase in power due to the introduction of the registers. In another attempt to isolate the discrepancy between the actual and estimated values of power consumption, the XOR gates used to generate the negative multiplicands were disabled. The resulting average current consumption is 15.2μA which agrees well with the estimated value. Since Booth recoding requires subtraction of partial products, a two's complement number representation is required. In order to subtract an M-bit number, all M bits must undergo power-consuming transitions. Thus, implementations employing a two's complement number representation inherently consume greater amounts of power compared to implementations employing a sign-magnitude number representation.

AREA x POWER^2 is plotted for all four cases in Figure 6-4. The implementation using 3-bit scanning with redundant multiples gives the best compromise between area and power consumption. Of the four cases, 2-bit scanning with Booth recoding is the worst.

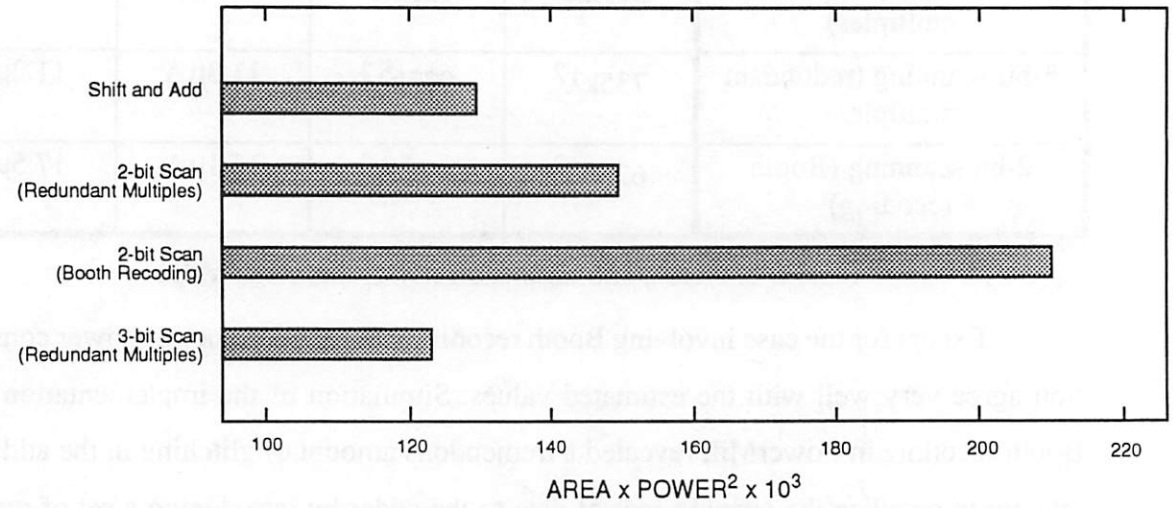


Figure 6-4. Exact Power and Area Efficiency for Several Algorithms

6.2.6. Extensions to Array Multipliers

As seen above, a sequential multiplier implementation using 3-bit scanning with redundant multiples gives the best compromise between area and power. These results obtained for sequential multipliers may be applied to array multipliers as well. An array multiplier is equivalent to a sequential multiplier using 9-bit scanning. As already noted, although a solution which employs multipliers based on 9-bit scanning uses the lowest power, such a solution also requires a very large amount of area. However, if pipeline stages are added, several multiplication operations may be processed concurrently. Thus, the effective power and area per operation decreases as long as the additional area and power consumed by the pipeline registers is not significant. The block diagram of a 9x9-bit array multiplier which scans the entire 9-bit number in sections of three bits is shown

in Figure 6-5. This implementation of the 9x9-bit array multiplier is equivalent to "unfolding" a sequential multiplier using 3-bit scanning with redundant multiples.

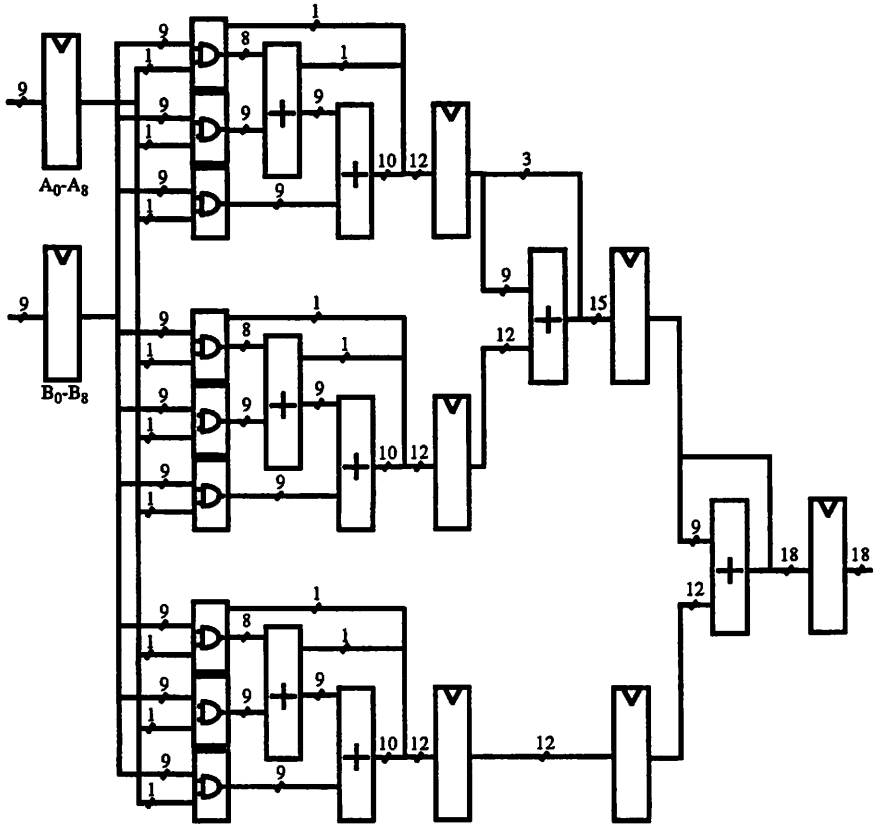


Figure 6-5. Block Diagram of a 9x9 Pipelined Array Multiplier

Since the accuracy of the power estimates have been confirmed, it is possible to proceed confidently with the analysis by just determining the number of required components for the pipelined array implementation as the intraregister logic is the same as for a sequential multiplier. This implementation requires 81 2-input AND gates, 99 TSPCR registers, and 78 full adders. If all registers are clocked at 3MHz, three multiplication operations are performed every 1μs. Thus, this implementation achieves the same throughput as an implementation which uses three parallel sequential multipliers all operating at a 1MHz rate. For the pipelined array multiplier the effective average current per operation is 9.8μA and the effective area per operation is $548k\lambda^2$. The area estimate is probably optimistic and a very careful layout is required in order to achieve the suggested benefits of such an implementation. Nevertheless, this example illustrates that the techniques used for power and

area optimization of sequential multipliers may also be applied directly to the design of pipelined array multipliers.

6.2.7. Conclusions of Multiplier Examination

For a system which is not rate constrained, the calculation of many multiplication operations requires efficient power and area utilization. A sequential multiplier architecture was selected and several multiplication algorithms were investigated for low power and area performance. Implementations using 2-bit scanning with Booth recoding are common for very high speed multipliers; however, such implementations are not optimal for low power since Booth recoding requires the use of a two's complement number representation and additional power is required for generating negative partial products. Recoding algorithms which require the use of a two's complement number representation should be avoided if low power is a primary objective.

For a 9x9-bit magnitude multiplier, the algorithm which results in the best compromise between area and power according to our metric is 3-bit scanning using redundant multiples. A sequential multiplier implementing this algorithm uses 35% less power than an implementation based on the shift and add method and 36% less power than an implementation based on 2-bit scanning with Booth recoding.

The analysis and results described for an area and power efficient implementation of a 9x9-bit sequential multiplier also apply to the implementation of a 9x9-bit pipelined array multiplier. "Unfolding" a sequential multiplier using 3-bit scanning with redundant multiples and inserting pipeline registers results in an array multiplier which is also very efficient in terms of area and power. However, in order to realize the suggested benefits, a very careful layout of the pipelined array multiplier is required. In addition, care should be taken to minimize the number of pipeline stages in order to minimize the additional area and power consumed by the extra registers.

6.3. Proposed DQPSK Design

Based on the above multiplier examination, we can tile up a low-power DQPSK decoder circuit using either four sequential multipliers (parallel implementation) or by

pipelining an array multiplier. These two options may be investigated in the same manner that as the multiplier cell itself, namely by counting the number and clocking rate of cells, and multiplying by our library characterization numbers.

6.3.1. Pipelined DQPSK with Array Multiplier

We might be tempted to think that we could obtain the best power by using a pipelined approach for the DQPSK multiplication. The projected power for a 3-bit redundant scanned array multiplier was about 10mA per multiplication, 10% lower than the iteratively scanned case due to more efficient use of the registers. However, that is not the case as the extra registers, necessarily clocked at a faster rate, eat away the margin. Hand analysis predicts a wash for overall power consumption. In addition, the control and layout of an array multiplier based DQPSK decoder are more complicated. A simple block diagram is shown, but this approach was abandoned in favor of the simpler parallel/sequential approach that has more natural, almost non-existent control.

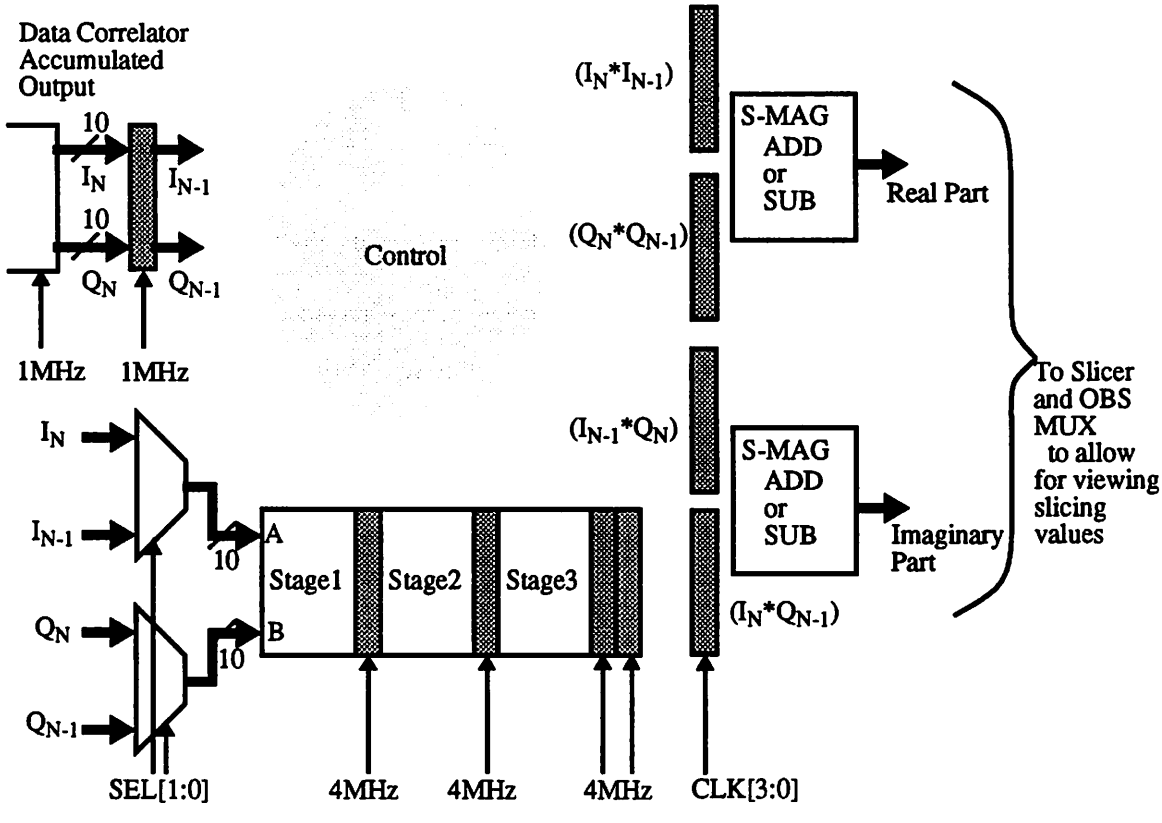


Figure 6-6. Pipelined Array Multiplier DQPSK Implementation

6.3.2. Parallel DQPSK with Sequential Multipliers

The parallel DQPSK schematic is shown below. (Note that in terms of relative area,

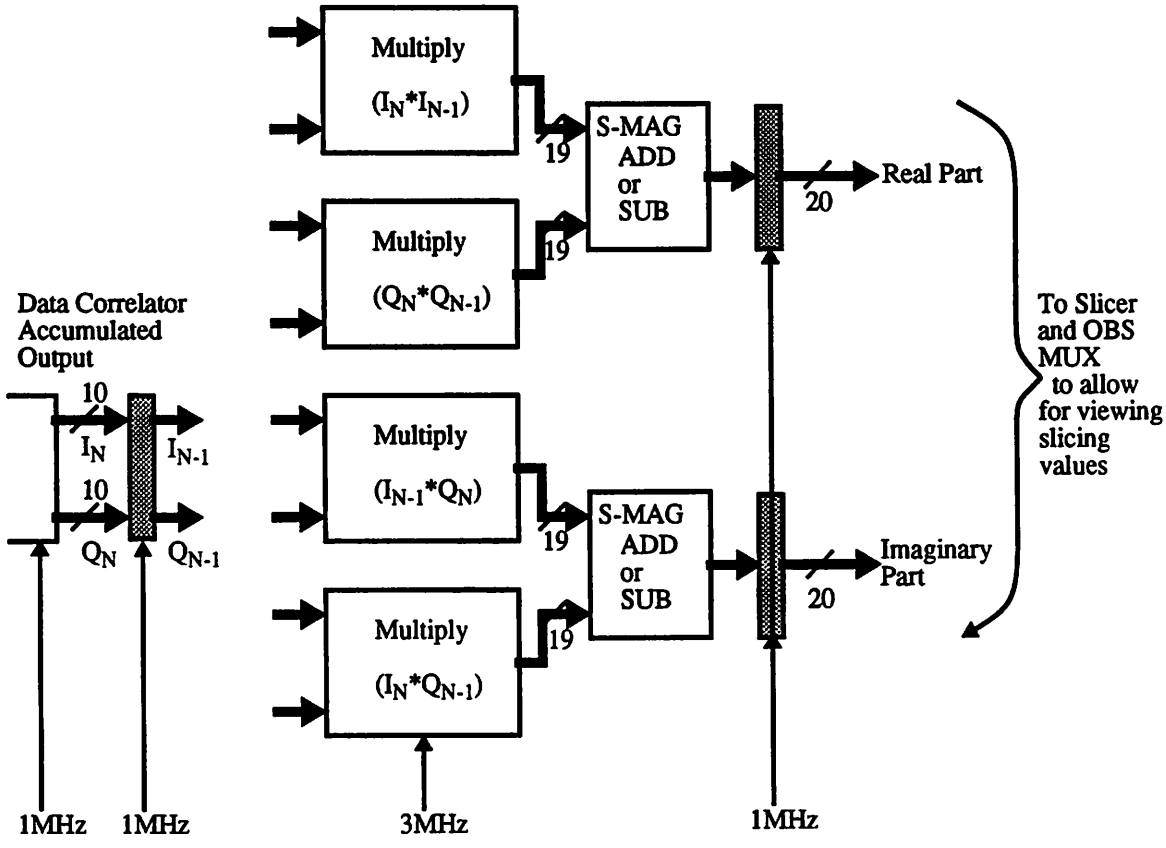


Figure 6-7. Parallel Sequential Multiplier DQPSK Implementation

the i_frontend data correlator is as tall and a little over twice as wide as a 3-bit redundant scan multiplier cell.) The Adder/Subtracter cell shown is a simple sign-magnitude arithmetic unit. Based upon the signs of the two input data samples, the values are muxed into either and addition or subtraction operation. If the operation is subtraction, a negative output is converted from 2's complement back to sign and magnitude format. All of the data buses shown are in sign-magnitude representation -- thus a width of 20 means 19 bits of magnitude plus a sign bit.

A bit slice of the Adder/Subtracter cell is shown below. The control signals are

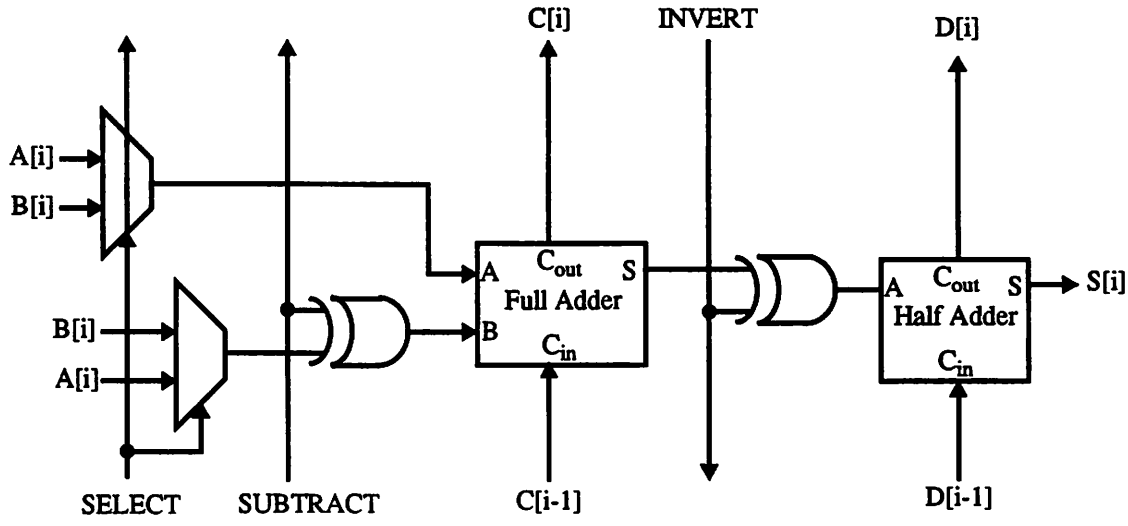


Figure 6-8. Bit Slice for Sign-Magnitude Add or Subtract ALU

fairly easy to generate based upon the sign of the incoming data and the sign of the Full Adder Output. Note that the carry in bit for full adder[0] (i.e. C[-1]) should be tied to SUBTRACT, thus if the operation is a subtraction, the carry-in will be 1. Also, the INVERT line should be tied to D[-1] and driven by the top bit of Full Adder Output (i.e. C[19]). In terms of decoding, SELECT=$\overline{\text{Sign[A]}}$•Sign[B], and SUBTRACT=Sign[A]⊕Sign[B]. The overall sign of the output is a little more complicated to design, but is given by the following equation (essentially, if the sign's are different, the subtract determines the sign, otherwise the sign is simply the sign of A):

$$\text{Sign[S]} = (\text{Sign[A]} \oplus \text{Sign[B]}) \bullet \text{C[19]} + (\overline{\text{Sign[A]} \oplus \text{Sign[B]}}) \bullet \text{Sign[A]}$$

Note that the slicer is not shown as it simply consists of 4 comparators and some simple decoding logic based that can easily worked out from the decoding conditions in Table 6-2.

The overall projected current dissipation for the DQPSK decoder is about 56μA for a 1.5V power of 84μW. The area is 5.6Mλ^2, ignoring the routing of those large buses, which is about equivalent to 3.5x the size of the first correlator design: i_frontend, or about the size of 3 i_basecorr long correlators.

6.4. Open Issues

The design is done and the multipliers have been tiled and simulated. All that is left for the DQPSK decoder is to tile it up using the 3-bit scan, redundant-multiple multiplier and low power library cells.The last set of comparators and registers for slicing need to be added too. After tiling, it should be simulated, of course, to verify functionality.

7 Testing Issues

7.1. Chip Strategy for Testing

In general we assume that all library cells are functional. Any new library cells that are added are tested individually in SPICE to verify proper operation. Also, hierarchically, new blocks are tested up to the chip core and finally pad level. The design was done in two phases: the first was a VHDL test to debug the overall chip operation prior to layout, the second was a switch-level simulation (IRSIM) after final layout to double check the extracted operation and connectivity. Unfortunately no tools were currently around to verify the critical path for the entire chip, so we rely on voltage scaling to help out with any speed problems. Race conditions won't be helped by this, of course, and should be avoided in a good design. In addition to simulation, the unique signal name alias file generated from IRSIM is examined to verify that supplies are not shorted and that clock and signal routing appear to be done correctly. This may change with the shift to CADENCE design environment which has an LVS (layout versus schematic) tool which should automatically verify the layout.

In terms of actual hardware support for testing, there is not a lot of support for the chip. No boundary scan or other techniques were employed owing to the additional work necessary and lack of support in our current tools. Rather, multiplexors were added to allow for the observation of correlation results and state bits (PN and Walsh bits, etc.). Although it is a simple and fairly low cost scheme, it requires a large number of output pins and a fair amount of wiring. From this information we would be able to determine what block was not working, but not exactly where or why. Optimistically we hoped that if the chip was well-tested prior to shipping, we wouldn't see any problems. Except for a couple minor bugs, things seemed to work as designed. An additional independent correlator was

added to the chip so that its functionality could be verified outside of the core. And, as we were using a 132 pin package, a fair number of output pins were dedicated to multiplexing the output of the dumps.

On the input side, the receiver chip was designed to operate in three different test-modes [Stone95]. Two modes connect to the A/D of the analog receiver chip through a 4-bit input which is converted internally to signed magnitude format. Another input was reserved for a half rate input from the digital transmitter chip with four paths of 2's complement to sign-magnitude conversion instead of the usual two for just I and Q. Other than those options for specifying the input format, no other testmodes exist. For the redesign, this issue should be looked at to see if supporting different formats at full and/or half rate would make sense.

In general as the chip was expected to work at or below 100mW, power supply IR drop and heating issues were not considered at all. While it is true that a chip may switch infrequently, but all at the same time yielding a large spike in current, no LdI/dt effects were considered either as this chip was not expected to exhibit that heavily correlated current consumption behaviour. The current levels are actually quite low given its 128MHz/ 64MHz operation so allocating around 8 pins per supply was deemed adequate. The supplies are: 5V core, 5V pad ring, 3.3V core, and 1.5V core. The number of 8 was arrived at by dividing up the left-over pins, after proscribing signals to the 132 pin package (chosen to allow for adequate cavity size and pin count). Note that package and chip supplies were not simulated owing to time constraints. Also, of final note, the internal power supplies are routed using wider, but not thicker, metal by Flint in no particular shape. Viz., no H-tree, grids, or other attempt to balance load, or IR drop was taken. In fact, Flint also routed the clock which tends to meander a bit over the chip. To compensate for this the high-frequency sections of the chip were treated as local islands of synchronicity, with skew-eating flip-flops placed in between where necessary.

It is a little unfortunate that the issue of reliability and testing is short-changed owing to a lack of time. In the real world, however, it is important and must be considered for a viable product. We are lucky to have the luxury to be able to ignore a lot of issues

because the design is low-power, but beware if you find yourself working on a more power-hungry design.

7.2. Testboard: Methods of Testing

A testboard was designed for the chip, using Viewlogic for schematic layout and Racal for placement and routing, that would allow for three different methods of testing: direct input, digital transmitter to digital receiver, and the whole system. The configurability of the testboard was achieved using jumper blocks to choose the input stimulus for the chip. The inputs and outputs of the chip were brought to a series of headers to allow the user of the testboard to wire-up or jumper the proper configuration for the testing being done. In addition to providing easy access to the pins, an SMA clock connector and proper biasing resistors for the input clock were added. Also, a threshold-refresh circuit and a debounced reset switch clocked by CLK128L, were added to fix the bugs discussed in Section 2.3. on page 17. An LED was added that is buffered off of the LOCK signal to allow for easy view of the chip's status.

The power supply for the chip (which requires three: 1.5V, 3V, 5V) was created by splitting the board's power plane and providing a BNC connector for each supply. In addition to the internal core supplies (1.5V, 3V, 5V), the pad supply was brought out to a separate BNC to allow for easy measurement of the core vs. pad power. Also, a 5V supply was needed for the TTL parts, so another supply was used. This does create a lot of different voltages, which is fine for a testboard in the lab with a lot of voltage generators, but bad for the final integrated radio. The issue of supply generation will have to be looked at when the integration issue arises.

Two screen captures of the board are shown below with annotation.

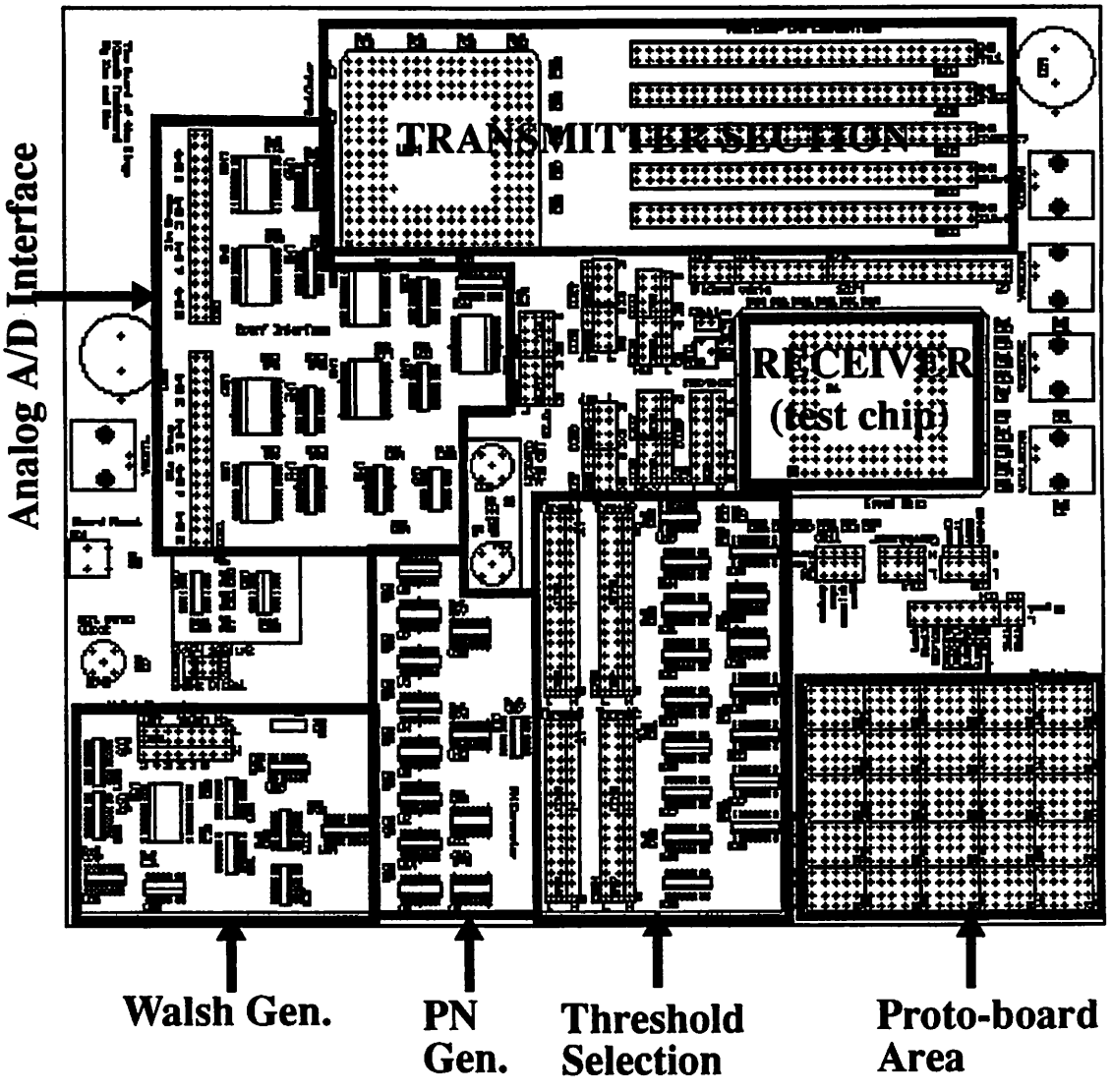


Figure 7-1. Digital Chip Test Board Layout: Part 1

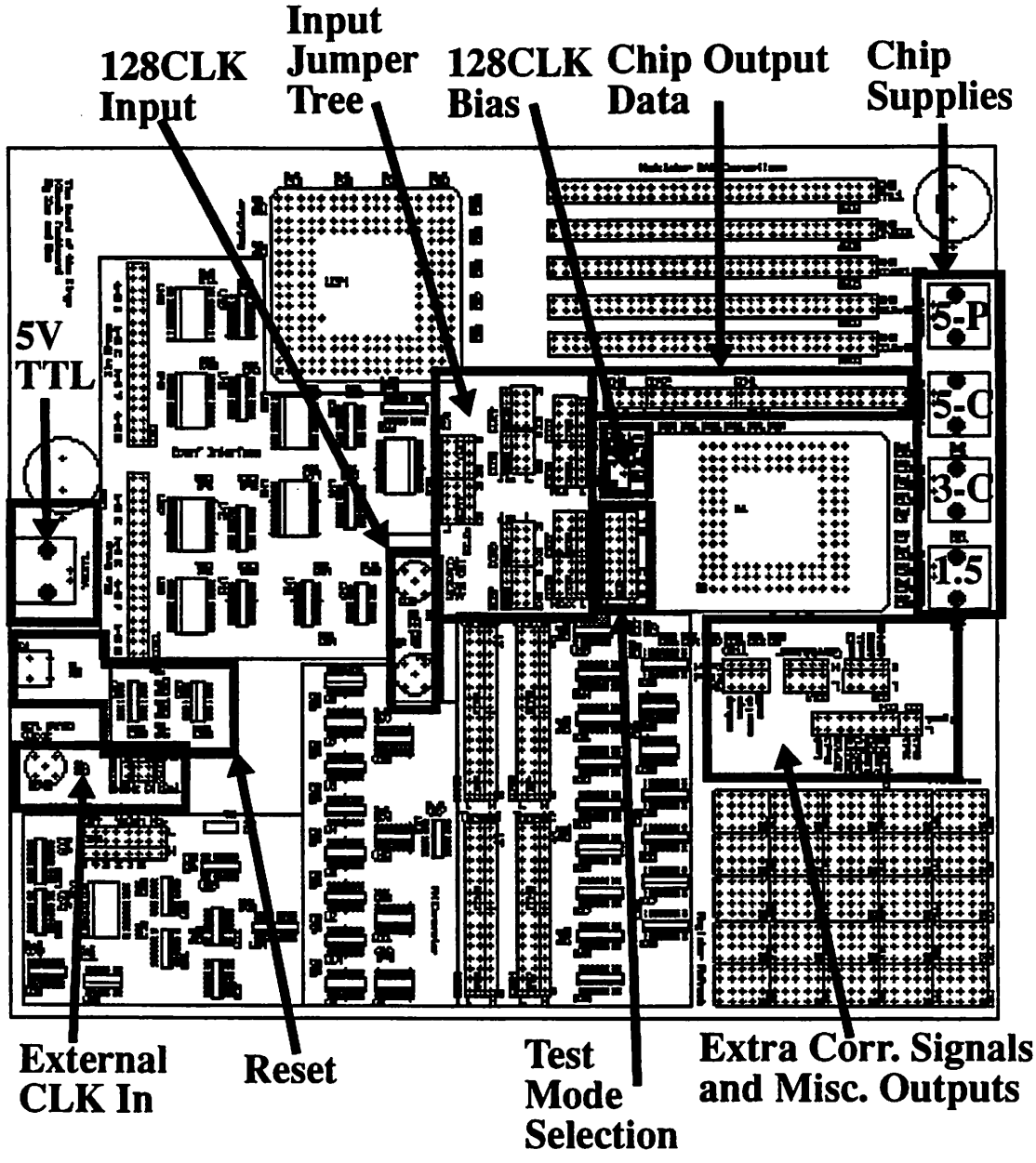


Figure 7-2. Digital Chip Test Board Layout: Part 2

Schematic for the entire board.

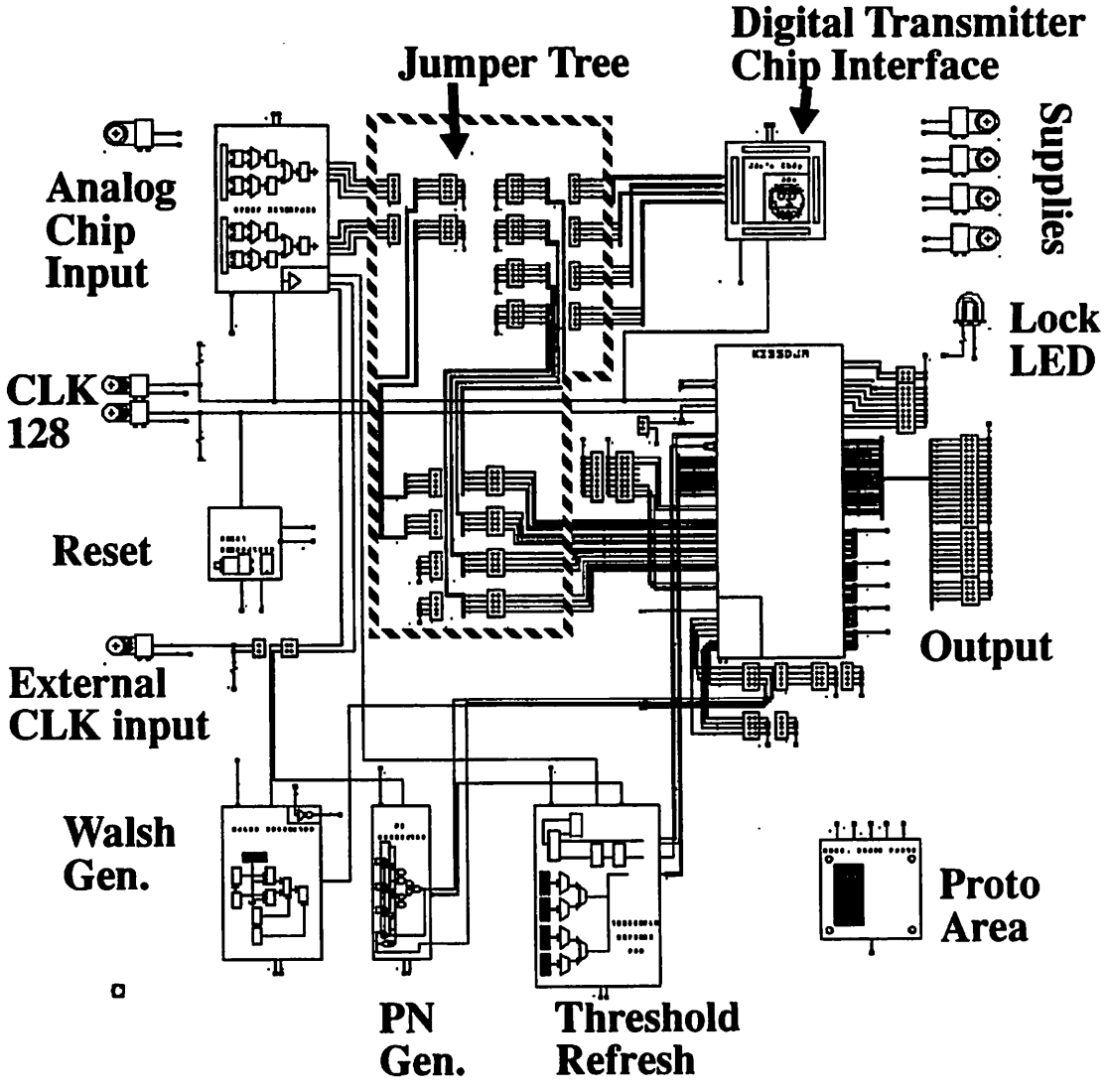


Figure 7-3. Digital Chip Test Board Schematic

7.2.1. Direct Input Testing

Known vectors were generated and fed into the digital chip by a DAS (Digital Acquisition System) to verify the correlator operation, and to test a the chip's ability to lock onto an input PN sequence. Unfortunately the memory length of the DAS precludes the ability to do a long data stream or to check for long term lock stability, so only short runs of data were done. To allow for best-case operation verification, an additional PN and Walsh code generation circuit was added to the testboard using TTL components. These could be run from a separate input clock, which would drift relative to the chip's clock, and

would allow for long correlations to verify the coarse and fine lock operation. The PN and Walsh generators were essentially TTL implementations of the same circuits described in [Stone95]. An emphasis was put on reusing the same TTL parts to simplify the design process and to help with ordering parts, as availability can be a problem. The DAS inputs (or the on-board PN/Walsh outputs) are directly connected to the jumper tree to drive the digital chip.

Below are series of schematics for the various sections of the testboard along with a brief description of their functionality.

7.2.1.1. Reset Generation

A debounced switch is flopped by the negative phase of the 128MHz clock to pro-

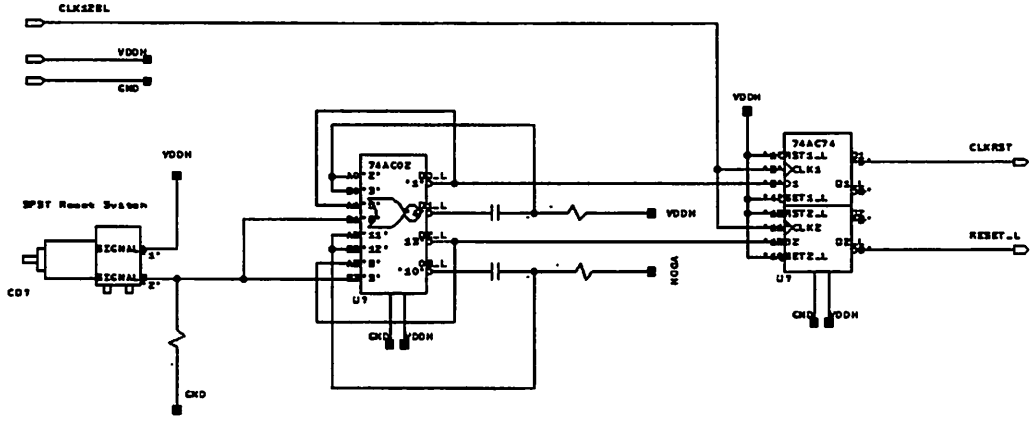


Figure 7-4. Test Board Reset Generation Schematic

vide the chip's reset. The flopping was necessary to overcome a slight internal reset bug.

7.2.1.2. Threshold Refresh

Another bug was that of the threshold registers losing their state. To overcome that,

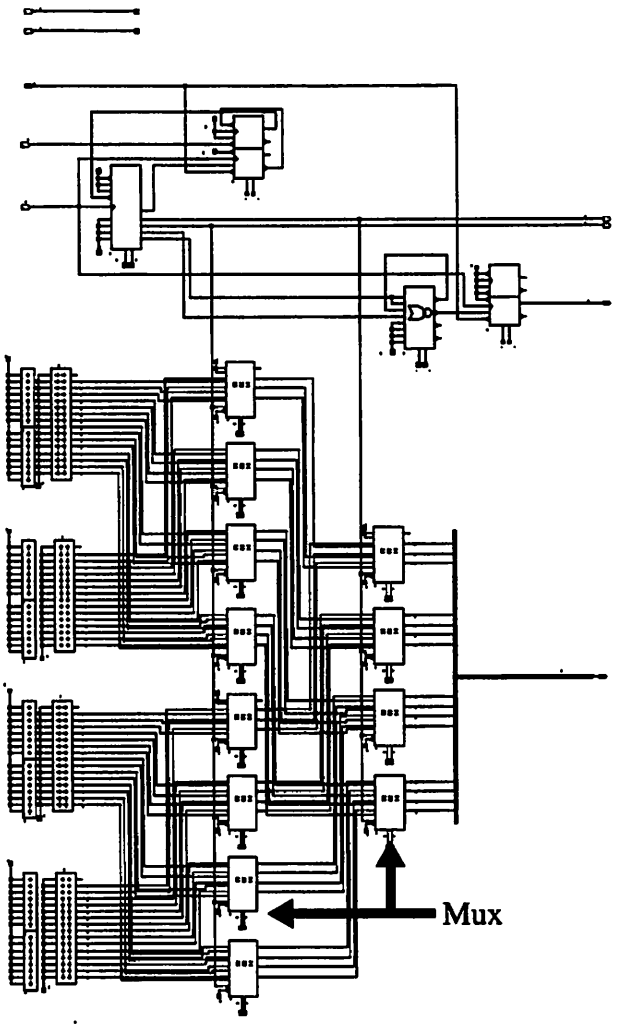


Figure 7-5. Test Board Threshold Refresh Schematic

a set of jumpers sitting between Vdd and GND allow for easy configuration of threshold values while some simple logic circuitry generates control signals for two levels of muxes which cycle between the values, writing the registers constantly. The registers are written at around the PN all-ones state (every 32768 * 15.6ns = 0.5ms) which should be often enough to avoid drift.

7.2.1.3. PN Generation

A set of eight 7474's were connected as a 16-bit shift register with their 'pre' and

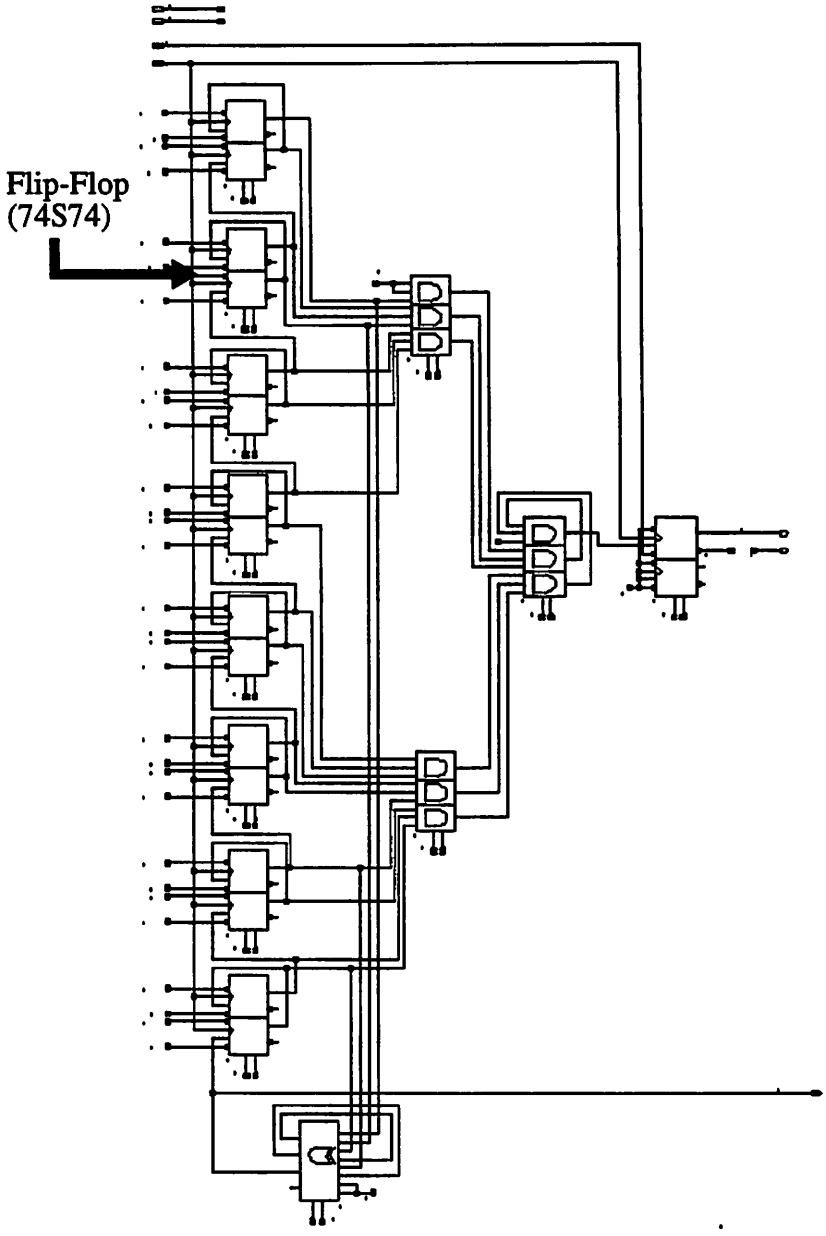


Figure 7-6. Test Board PN Generation Schematic

'clr' inputs hardwired to the PN seed value. At the all-one's state it resets itself. This is a little tricky as the TTL parts work at 64MHz and thus require some set-up (a pipe stage) prior to generating the load. Note that the worst case delays from flop to flop needed to be determined for proper operation and that not any 74XX part will work. Usually F, AS, or

S is required. Both this and the Walsh generation block optionally run off of a separate digital clock.

7.2.1.4. Walsh Generation

This circuit literally implements the Walsh circuit in [Stone95] used in the actual

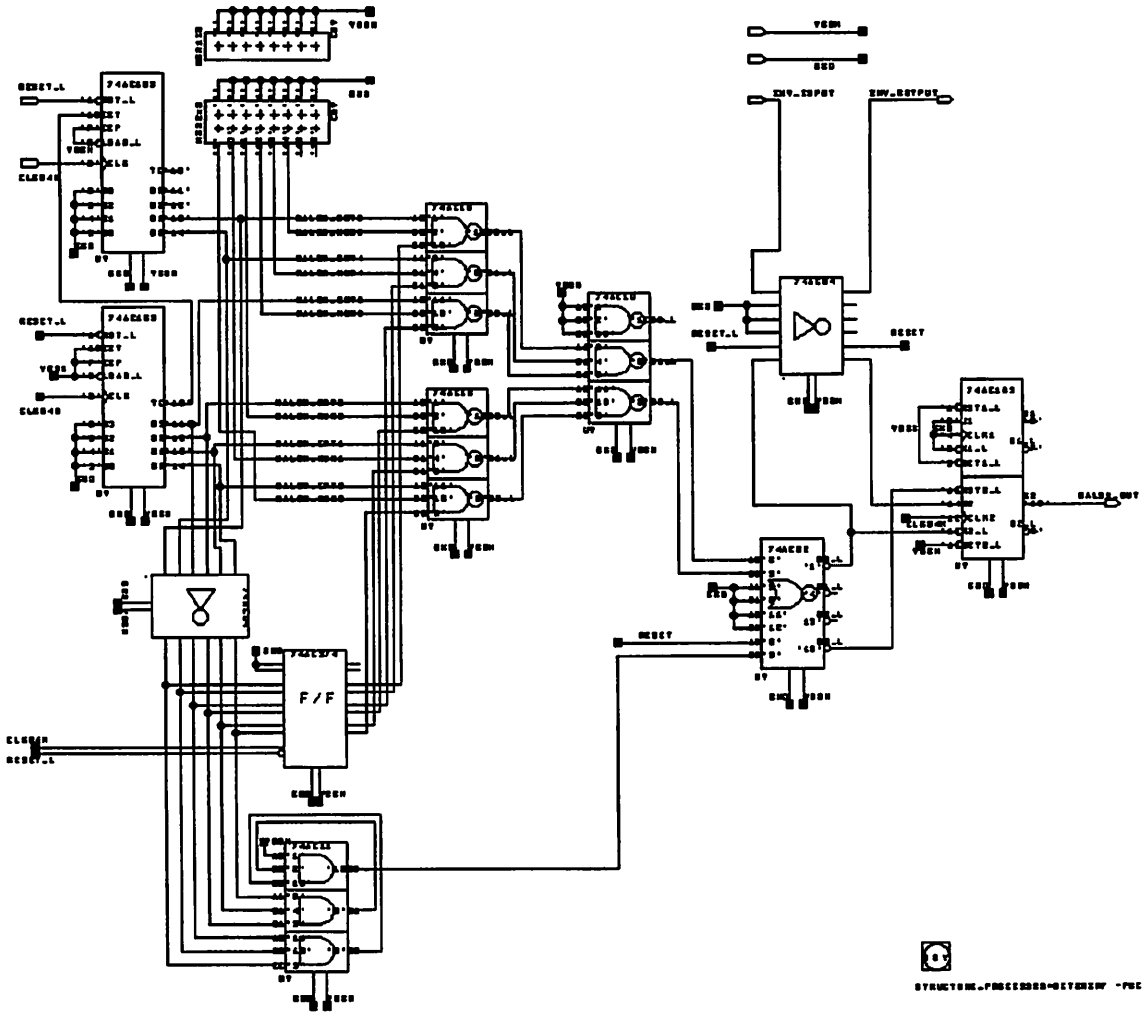


Figure 7-7. Test Board Walsh Generation Schematic

chip design. It also operates at 64MHz and carries the same caveats as the PN generation block. Both blocks run, optionally, off of a separate digital clock. A set of 6 jumpers chooses which Walsh code is desired. A good reference on Walsh functions is [Beauchamp].

7.2.2. Digital Baseband Test

The idea here is to do a partial test of the system by connecting the output of the digital baseband transmitter chip to the input of the digital receiver chip. This will verify that the digital path is working. This is discussed in more detail in [Yee96]. Basically the digital transmitter chip interface was taken exactly from the transmitter chip testboard [Peroulas96] [Yee96], and placed on the top of the receiver testboard. A set of jumpers allows the tester the option of which bits from the output of the transmitter to connect up to the input of the receiver. The receiver chip was built to have 3 test modes [Stone95], one of which is intended to receive the digital transmitter's data in 2's complement format directly. A possible issue to think about for the redesign is whether this is adequate, or if other input combinations might easily allow another variety of test input.

7.2.3. Full System Test

Just as it sounds, this involves digital transmission, to analog mix-up to air (or wire) to analog receiver and mix down, to digital receiver chip. The testboard has a series of multiplexors to take the parallel output of the analog receiver chip, and re-mux them into the expected stream. The original design was to have a single-chip solution, but this test version separates analog and digital at the A/D.

A future interface for the redesign will allow for 2's complement or sign-magnitude input of various fractional rates to allow for easy hook-up to whatever the analog front-end winds up being. Also the future chip will have the PN and Walsh Gen. on board, so the chip can help test itself and we can get rid of these TTL chips.

A diagram of the analog chip input multiplexors are shown below. Note that at 128MHz there is only enough time to go through one flop and one mux. Muxing control signals are generated from a 74163 counter. For the redesign, a half-rate or lower input would make more sense as it would relax the board frequencies and package requirements

and lessen the need for speedy, and power hungry chip I/O pads while only increasing the number of pins needed by groups of four for each 4-bit input.

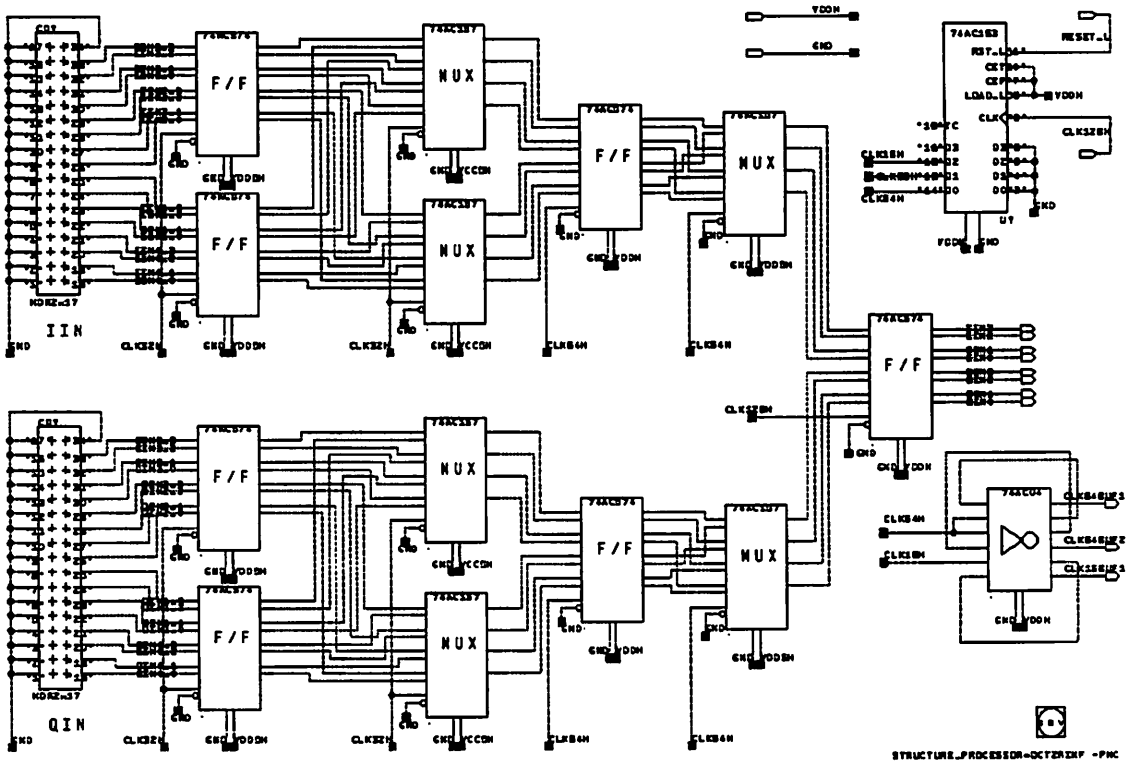


Figure 7-8. Analog Chip Input Interface

7.3. Notes About Board Design

Following are some random observations and suggestions about board design that might be helpful.

1. Clock Frequency: This testboard is a simple, standard eight-layer type from Multek with four power and four signal layers. In general it can be expected to operate up to around 64MHz without having to do anything special with Racal. Beyond 100 MHz transmission line effects, signal cross talk, packaging considerations, and current drive become issues. Beyond carefully routing, providing termination for, and sizing the 128MHz clock traces, no special care was taken with the signals on this board. For designs with more high frequency signals, see [Sheng96] and [Yee96] for examples of RF board design. (I.e. things like finding the intralayer material dielectric constant for a board given the layer spacing to calculate the trace width of a stripline for a Z_0 of 50Ω)

2. DAS Use: To help improve use with the DAS, an easy thing to do is to bring signals to headers in blocks of two lines the width of the DAS pod, one with GND, the other with signals. This allows you to simply plug the pod onto the board and moves the wiring issue to programming the software in the DAS with the correct lines. This can be a boon if you have to share the DAS or if you don't like wiring all that stuff up by hand.
3. Jumpers: Another possibly useful technique is to group jumpers in lines of three headers, with one outside line connected to Vdd, the other to GND. The signal, in the middle, may easily be jumpered to power or ground as necessary. Or, instead of connecting to supplies, other signals can be connected, allowing you the ability to multiplex which input stream is desired without much hardware overhead. Issues can arise for high frequency signals, as the inductance, capacitance and resistance of the jumpers can come into play, so use discretion.
4. On Chip Test Structures: Try to make you life easier by putting things like PN and Walsh generators on chip, area and pins allowing. A few clever additions may allow your chip to generate test vectors for itself (or another chip). This may lessen the board complexity and number of exterior parts needed.
5. Silkscreen: This is important: ALWAYS LABEL YOUR BOARD. Hopefully with at least the + and - terminals for the power supplies if not also signal names and other assorted helpful items. It takes some time, and must be edited personally in Racal, but is well worth it after 16,000 times of referring back to that tattered piece of paper which has the header pinout on it.
6. Soldering/Wirewrapping: Don't be afraid to do some rework or jumpering for the inevitable errors that show up. Practice on a scrap board. It can be fun! Lead and flux can be your toxic friends!
7. Proto Area: Always include some proto area somewhere on your board. You never know when you're going to want to add that chip or LED. Or use it to practice soldering. It can save you sometimes and isn't terribly hard to include.
8. LED's: Speaking of LED's, always put at least one LED on your board. Perhaps for the power supply. Why? LED's are neat and fun to watch. They make ordinary boards into extraordinary boards. Oh, and sometimes they can visually provide very useful info like whether the receiver is in lock, or whether a Xilinx has been programmed, etc. You don't want to have to probe all the time to find out that info unless there is a problem. But beware that sometimes they can alias quickly changing signals into a soft DC glow.
9. Debouncing Signals: Usually a good idea for anything that might involve the clock or a reset. If you don't recall how to hook up the cross-coupled NAND's as a set-reset latch with pull-up resistors, having the switch pull down an input then I'm sure you can find it in most any beginning digital design book [Wakerly].
10. Vdd and Clock Inputs: Usually it is not a bad idea to use BNC's to connect as they provide some noise immunity as they are shielded. But that is only as good as the board is, in terms of shielding.

11. Separate Power Planes: If done too much, this can turn a power plane into a power spaghetti trace which might have nasty side-effects if it is running a lot of current, but overall it is an easy way to use a single layer for multiple supplies. Also convenient for measuring the power of that supply at the terminal.

12. Decoupling Capacitors: Use at least some. Some people use a simple ratio, i.e. one decoupling cap for every 10 signals. Try to place them as near the power supply pins on the chip as possible. A rule of thumb is to use more caps if the chip uses more power, and vice versa. Note that at high-frequencies even the surface-mount decoupling caps can resonate, becoming useless, so be careful. Also, usually it is a good idea to include a large electrolytic capacitor (mF order) near the supply in addition to lots of small μF decoupling caps. These big caps sometimes leak (current), so beware when measuring small power numbers. Also, they explode if plugged in reverse polarity, by the way. Fun! (And toxic!)

13. Sockets: Sockets are very useful, especially the ZIF (zero insertion force) variety which make swapping test chips a breeze. Some sockets have a tighter fit, but require enormous physical strength to force a chip into them. Avoid these unless the frequency response of the ZIF renders it unusable; ZIF sockets don't necessarily operate much above 50MHz. In some cases, e.g. RF designs, sockets can't be used due to their insertion loss and frequency characteristics. Generally, only socket things you expect to change.

14. Ordering Parts: General rule: order the parts as soon as they are specified, even before the design has been moved to layout. Sometimes lead times can be quite long and/or parts suddenly become unavailable. Also, consult the business chip guides in the lab for the list of distributors for the company whose chip you want. Sometimes one distributor will be out while the other will have a surplus. A warning: sometimes it can take a full day of calling to find a part. Be prepared for that, and also for the possibility that the part will be unobtainable and hence a redesign will be required (better to catch early on in the design).

7.4. Redesign Test Issues

To reiterate, the future version of the chip will have some alterations to ease its testing. Namely a separate PN and Walsh generator block with an independent clock will be added to allow one chip to generate test patterns for another or itself. Also the input interface to the chip will be changed to allow for 2's complement, sign-magnitude, or unmodified data inputs at half or full rate. The chip output, correlation values and state, is planned to be multiplexed as before, but possibly with a little address decoding. This would offer a RAKE receiver the option of taking the correlations pre-DQPSK decoding. The DQPSK output bit stream will be on separate pins from the outputs mentioned above to allow for

both options independently. This does not cost much, as the output data is a single bit stream, but it impacts the ECC/Protocol chip which will be reassembling packets from the output bit stream.

8 Conclusion

Although the title 'Conclusion' is a bit of a misnomer, as the redesign of the digital receiver chip is not yet finished, this is a good time to review the progress so far, and the open issues still left to do for the design. A large amount of work has been done for the first chip design and the subsequent redesign and not all of the details are recorded in this already quite long document. I have attempted to capture the design process as well as the testboard and important integrated circuit datapath blocks in this document.

A review of the desired backend digital functionality is presented in Chapter 2 in addition to the implemented subset of the first chip version. Namely, the first version of the chip is able to achieve coarse and fine lock, raw data recovery (no DQPSK decoding), and some observation of multipath channel correlations. The goals for the redesign were to flush out the list of functionality including a DQPSK decoder, adjacent cell scan and hand-off ability, the ability to observe multipath data and channel correlations, along with a reexamination of the correlator design and some minor bug fixes. Currently the correlator has been reexamined, and determined to be adequate and a DQPSK decoder is nearly laid out. Still pending are the adjacent cell scan implementation, minor bug fixes, and the new scheme to allow for RAKE receiving. In addition the revision will include some test structures to allow for easy self-test, and it will be hand-placed at the top level to allow for tighter packing than Flint can produce.

After the review of the current state of the design, process characterization is explored as the beginning part of the design process. An automated SPICE file and script are produced that allow for high-level empirical modeling of a process based on ring oscillator and single transistor data. The results from several relevant processes are presented and later used to help determine architectural trade-offs.

Following the process characterization is the first exploration of the correlator. The rationale for its architecture, carry-save with positive and negative accumulators, is presented and a custom datapath cell based approach is taken for implementation. Clock buffer sizing and power estimation are preformed and compared to simulated results. A single long correlator (1024 samples, 'i_basecorr') is $1.54M\lambda^2$ (0.24 mm^2 in 0.8μ) and is projected to consume about 0.8mW at 64MHz, whereas the measured value is slightly higher at 1.2mW due to additional buffers and drivers.

The first correlator was designed for a 1.2μ process, when we later obtained access to a 0.6μ process, the correlator was reexamined. The idea was that an offset binary representation might be able to shrink both the area and the power of the first design. Unfortunately while we were able to shrink the area by 40%, the power increased by about three times over the first design. The culprit was a bad policy of oversizing all the devices to meet the necessary timing requirements. This helps to illuminate a better path to low power design, namely to use minimum size devices, scale the voltage down as low as it can reasonably go, and compensate for critical paths by pipelining and parallelizing the circuit. There is a direct cost of area for this approach, but it seems to produce low power results. The whole design was not wasted, though, as it also examines simple adder topologies and explores two techniques for speeding up a critical path: merging logic into latches and the use of complex cells. In addition, the use of offset binary encoding is explored and found to create its own difficulties integrating into the system. Owing to these issues and the power results, the second redesign was abandoned in favor of simply scaling the first design down to take advantage of a better process. Some cells could be reduced in size to further save on power if it does not impact the critical path.

After two chapters on custom design we move up a level to semicustom for an examination of multiplier algorithms for the DQPSK decoder. An empirical characterization process for power is performed on tilings of low power library cells and a technique for simple power estimation is determined. Several implementations of multipliers are created to verify the power estimation results. A simple 3-bit scan iterative multiplier with redundant multiples is determined to be the best candidate for a low power DQPSK

decoder, saving power by almost a factor of two over using four similar booth-encoded multipliers, and the projected final design is found to be of negligible power (84μW) and about as large as three i_basecorr long correlators ($5.6M\lambda^2$).

Finally the issue of testing at the chip and board level are explored. The rather optimistic testing strategy employed on the chip design is discussed and the three methods of chip testing at the system/board level are explained. In addition, hopefully helpful hints on board design are given and suggestions are made regarding the inclusion of test structures on chip to simplify the testboard design and system integration.

As aforementioned, the redesign is not finished as the open issue of adjacent cell scan needs an implementation, the DQPSK decoder needs a little bit more for final layout and simulation, and the whole chip will need to be rebuilt, including the bug fixes, changes to correlation observation, and floorplanning. While still a large amount of work needs to be done, it is not an overwhelming task. As of the last word, work is continuing towards a usable digital backend chip for the radio system. Hopefully within a year the CDMA system can be tested and perhaps even ultimately integrated into an InfoPad.

Bibliography

[Afghahi, Yuan] M. Afghahi, J. Yuan. "Double Edged-Triggered D-Flip-Flops for High-Speed CMOS Circuits," IEEE Journal of Solid-State, Vol. 26, No. 8, August 1991.

[Beauchamp] K. G. Beauchamp. Applications of Walsh and Related Functions, with an Introduction to Sequency Theory, Academic Press, Orlando, USA, 1984.

[Bewick92] G. Bewick and M. Flynn, Binary Multiplication Using Partially Redundant Multiples, Stanford University, Computer Systems Laboratory, Technical Report No. CSL-TR-92-528, 1992.

[Booth51] A. Booth, "A Signed Binary Multiplication Technique," Quarterly Journal of Mechanics and Applied Mathematics, pp. 236-240, 1951.

[Burd94] T. Burd. Low-Power Cell Library, M.S. Thesis, U.C. Berkeley, June 1994.

[Chandrakasan92] A. Chandrakasan, S. Sheng, R. W. Brodersen. "Low Power CMOS Digital Design," IEEE Journal of Solid-State Circuits, Vol. 27, No. 4, pp. 208-211, Feb. 1992.

[Chandrakasan94] A. Chandrakasan. Low Power Digital CMOS Design, Ph.D. Thesis, U.C. Berkeley, August 1994.

[Ercegovac, Lang] M. Ercegovac, T. Lang. "Low Power Accumulator (Correlator)," Digest of IEEE Symposium on Low Power Electronics, pp. 30-31, 1995.

[Gray, Meyer] P. Gray, R. Meyer. Analysis and Design of Analog Integrated Circuits, 3rd ed., John Wiley & Sons Inc., New York, USA, 1993.

[HSPICE] HSPICE User's Manual, Meta-Software Inc., 1991.

[Lynn95] L. Lynn. Low Power Analog Circuits for an All CMOS Integrated CDMA Receiver, M.S. Thesis, U.C. Berkeley, September 1995.

[MacSorley61] O. MacSorley, "High-Speed Arithmetic in Binary Computers," Proceedings of the IRE, pp. 67-91, 1961.

[Matsui95] M. Matsui and J. Burr, "A Low-Voltage 32x32-Bit Multiplier in Dynamic Differential Logic," Proceeding of the IEEE Symposium on Low Power Electronics, pp. 34-5, 1995.

[Moshnyaga95] V. Moshnyaga and K. Tamaru, "A Comparative Study of Switching Activity Reduction Techniques for Design of Low-Power Multipliers," IEEE International Symposium on Circuits and Systems, vol. 3, pp. 1560-1563, 1995.

[MOSIS] J. Pi. MOSIS Scalable CMOS Design Rules, Rev. 7, MOSIS Information Sciences Institute, U.S.C., 1996.

[Muller, Kamins] R. Muller, T. Kamins. Device Electronics for Integrated Circuits, 2nd ed., John Wiley & Sons Inc., New York, USA, 1986.

[Najim] F. Najim. "A Survey of Power Estimation Techniques in VLSI Circuits", IEEE Transactions on VLSI Systems, Vol. 2, No. 4, December, 1994.

[Nagendra] C. Nagendra, R. Owens, M. Irwin. "Power-Delay Characteristics of CMOS Adders," IEEE Transactions on VLSI Systems, Vol. 2, No.3 September 1994.

[O'Donnell, Yee 241] I. O'Donnell, D. Yee. Algorithmic Power and Area Considerations in Sequential Multipliers, EECS 241 Project, U.C. Berkeley, Spring 1996.

[Oklobdzija94] V. Oklobdzija, D. Villeger and T. Soulas, "An Integrated Multiplier for Complex Numbers," Journal of VLSI Signal Processing, pp. 213-222, 1994.

[Omondi] A. Omondi. Computer Arithmetic SYstems: Algorithms, Architecture, and Implementations, Prentice Hall Inc., New York, USA, 1994.

[Peroulas96] J. Peroulas. Design and Implementation of a High Speed CDMA Modulator for the INFOPAD Basestation, M.S. Thesis, U.C. Berkeley, December 96.

[Proakis] J. Proakis. Digital Communications, Prentice-Hall Inc., New Jersey, USA 1987.

[Rabaey] J. Rabaey. Digital Integrated Circuits, A Design Perspective, Prentice Hall Inc., New Jersey, USA, 1996.

[Rabaey241] J. Rabaey. EECS 241 Digital Circuit Design Class Notes. U.C. Berkeley, Spring 1996.

[Sheng91] S. Sheng. Wideband Digital Portable Communications: A System Design, M.S. Thesis, U.C. Berkeley, December 1991.

[Sheng92] S. Sheng, A. Chandrakasan, R.W. Brodersen. "A Portable MultiMedia Terminal," IEEE Communications Magazine. Vol. 30, No. 12, Dec. 1992, pp. 64-75.

[Sheng94] S. Sheng, R. Allmon, L. Lynn, I. O'Donnell, K. Stone, R.W. Brodersen. "A Monolithic CMOS Radio System for Wideband CDMA Communications," Proceedings to Wireless '94 Conference, Calgary, Canada, June 1994.

[Sheng96] S. Sheng. Wideband Digital Portable Communications, Ph.D. Thesis, U.C. Berkeley, December 1996.

[Sheng ISSCC] S. Sheng, L. Lynn, J. Peroulas, K.Stone, I. O'Donnell, R.W. Brodersen. "A Low-Power CMOS Chipset for Spread-Spectrum Communications," peoceedings of the IEEE ISSCC, pp. 346-347, 1996.

[Somasekhar] D. Somasekhar, V. Visvanathan. "A 230-MHz Half-Bit Level Pipelined Multiplier Using True Single Phase Clocking," IEEE Transactions on VLSI Systems, Vol. 1, No. 4, December 1993.

[Stone95] K. Stone. Low Power Spread Spectrum Demodulator for Wideband Wireless Communications, M.S. Thesis, U.C. Berkeley, August 1996.

[Swartzlander] E. Swartzlander, Computer Arithmetic, Parts I and II, IEEE Computer Society Press, 1990.

[Teuscher95] C. Teuscher. Software Simulation of the INFOPAD Wireless Downlink, M.S. Thesis, U.C. Berkeley, March 1996.

[Villeger93] D. Villeger and V. Oklobdzija, "Evaluation of Booth Encoding Techniques for Parallel Multiplier Implementation," Electronics Letters, vol. 29, no. 23, pp. 2016-7, 1993.

[Wakerly] J. Wakerly. Digital Design: Principles and Practices, Prentice Hall Inc., New Jersey, USA 1990.

[Wei95] B. Wei, H. Du and H. Chen, "A Complex-Number Multiplier Using Radix-4 Digits," Proceedings of the 12th Symposium on Computer Arithmetic, pp. 84-90, 1995.

[Yee96] D. Yee. The Design and Implementation of a Semi-Custom Transmitter for a CDMA Direct Sequence Spread-Spectrum Transceiver, M.S. Thesis, U.C. Berkeley, December 1996.

[Yuan, Svensson] J. Yuan, C. Svensson. "High Speed CMOS Circuit Technique," IEEE Journal of Solid-State Circuits, Vol. 24, No. 1, February 1989.

Appendix A: SPICE Files

Ring Oscillator Characterization: SPICE

```
***** top level cell is ringoscchar (5 stages)
**
** The purpose of this spice file is to obtain an estimate for the following
** parameters:  tplh, tphl, tr, tf, Cgate, Cl (= Cdrainp+Cdrainn+Cinvnextstage)
** from a ring oscillator structure (and some CCCS's and devices) through
** a transient simulation.  The objective is to parametrize Vdd and the width
** of the devices to allow for simple sweeping for a given process.  The
** results can then be used as approximations at a higher level of circuit
** design to help estimate performance.
**
** This file runs several .alters of width for the given Vdd parameter
** (it has to be re-run for different Vdd's).  Also -- all measurements
** have the prefix "II_" to allow for easy 'grep-ing' of the desired data
** from the hspice output file.
**
** Note: For higher vdd, you might want to increase the resolution of the
** .tran (and decrease the time pTran for which it runs)
**
** Note: To configure this for a different process, be sure to change the
** model file (.included below), adjust the pLambda parameter to be 1/2 the
** smallest drawn length, adjust the length of the simulation (pTran)
** to allow for at least 5 cycles for the slowest ring osc (3 lambda width)
** -- you might also change the .tran resolution appropriately also, and
** finally you might scale pIgateMeas so that it causes vng,vpg to hit pVdd
** around pTran
**
** by Ian O'Donnell Jan 16th, 1995.
*****
.options acct nomod post=1
```

```
.param pVdd=1.5 pLambda=0.35e-6 pWidthInLambda=3 pTran=40e-9
+ pIgateMeas='pWidthInLambda*pLambda*2*pLambda*3.46e-15*pVdd*.75e12/pTran'
*
** Initialization of nodes
.ic v(out1)=pVdd v(out2)=0 v(out3)=pVdd v(out4)=0 v(out5)=pVdd
+ v(vp4)=0 v(vn4)=0 v(vpg)=0 v(vng)=0
*
** Supplies
vdd vdd 0 dc pVdd
*
vdump4 dump4 vdd dc 0
vdumn4 dumn4 0 dc 0
*
isp0 0 vpg dc pIgateMeas
isn0 0 vng dc pIgateMeas
*
** Cload Calc Structure
cp4 vp4 0 1e-15
cn4 vn4 0 1e-15
*
fp4 vp4 0 CCCS vdump4 1
fn4 0 vn4 CCCS vdumn4 1
*
** Parametrized MOS subciruit
.subckt nsubcktmos d g s b pWinLambda=3
mn d g s b nmos W='pWinLambda*pLambda' L='2*pLambda'
+ PS='10*pLambda+pWinLambda*pLambda' PD='10*pLambda+pWinLambda*pLambda'
+ AS='5*pLambda*pWinLambda*pLambda' AD='5*pLambda*pWinLambda*pLambda'
.ends
*
.subckt psubcktmos d g s b pWinLambda=3
mp d g s b pmos W='pWinLambda*pLambda' L='2*pLambda'
+ PS='10*pLambda+pWinLambda*pLambda' PD='10*pLambda+pWinLambda*pLambda'
+ AS='5*pLambda*pWinLambda*pLambda' AD='5*pLambda*pWinLambda*pLambda'
.ends
*
** Ring OSC structure (5 stages)
xp1 out1 out5 vdd vdd psubcktmos pWinLambda=pWidthInLambda
xn1 out1 out5 0 0 nsubcktmos pWinLambda=pWidthInLambda
*
xp2 out2 out1 vdd vdd psubcktmos pWinLambda=pWidthInLambda
xn2 out2 out1 0 0 nsubcktmos pWinLambda=pWidthInLambda
*
```

```
xp3 out3 out2 vdd vdd psubcktmos pWinLambda=pWidthInLambda
xn3 out3 out2 0 0 nsubcktmos pWinLambda=pWidthInLambda
*
xp4 out4 out3 dump4 vdd psubcktmos pWinLambda=pWidthInLambda
xn4 out4 out3 dumn4 0 nsubcktmos pWinLambda=pWidthInLambda
*
xp5 out5 out4 vdd vdd psubcktmos pWinLambda=pWidthInLambda
xn5 out5 out4 0 0 nsubcktmos pWinLambda=pWidthInLambda
*
** Extra devices for gate-cap calc
xp0 0 vpg vdd vdd psubcktmos pWinLambda=pWidthInLambda
xn0 vdd vng 0 0 nsubcktmos pWinLambda=pWidthInLambda
*
** Models
.include 'hp_0.6um.113'
*
** Analysis
.tran 0.1n pTran
*
** Measurements
.meas TRAN II_Trise TRIG V(out3) val='0.1*pVdd' TD=0 RISE=4
+ TARG V(out3) val='0.9*pVdd' RISE=4
.meas TRAN II_Tfall TRIG V(out3) val='0.9*pVdd' TD=0 FALL=4
+ TARG V(out3) val='0.1*pVdd' FALL=4
.meas TRAN II_Tdhl TRIG V(out3) val='0.5*pVdd' TD=0 RISE=4
+ TARG V(out4) val='0.5*pVdd' FALL=4
.meas TRAN II_Tdlh TRIG V(out3) val='0.5*pVdd' TD=0 FALL=4
+ TARG V(out4) val='0.5*pVdd' RISE=4
*
.meas TRAN II_tring TRIG v(out3) val='0.5*pVdd' TD=0 RISE=4
+ TARG v(out3) val='0.5*pVdd' RISE=5
.meas TRAN II_tp PARAM='II_tring/10'
*
.meas TRAN II_vng FIND v(vng) AT='pTran-2e-9'
.meas TRAN II_Cgaten PARAM='pIgateMeas*(pTran-2e-9)/II_vng'
.meas TRAN II_vpg FIND v(vpg) AT='pTran-2e-9'
.meas TRAN II_Cgatep PARAM='pIgateMeas*(pTran-2e-9)/II_vpg'
*
.meas TRAN II_tout4c3 WHEN v(out4)='0.5*pVdd' CROSS=3
.meas TRAN II_tout4c4 WHEN v(out4)='0.5*pVdd' CROSS=4
.meas TRAN II_tout4c5 WHEN v(out4)='0.5*pVdd' CROSS=5
.meas TRAN II_tout4c6 WHEN v(out4)='0.5*pVdd' CROSS=6
.meas TRAN II_vp4a FIND v(vp4) AT='(II_tout4c4+II_tout4c5)/2'
```

```
.meas TRAN II_vp4b FIND v(vp4) AT='(II_tout4c5+II_tout4c6)/2'
.meas TRAN II_energyp PARAM='1e-15*(II_vp4b-II_vp4a)*pVdd'
.meas TRAN II_cloadp PARAM='1e-15*(II_vp4b-II_vp4a)/pVdd'
.meas TRAN II_vn4a FIND v(vn4) AT='(II_tout4c3+II_tout4c4)/2'
.meas TRAN II_vn4b FIND v(vn4) AT='(II_tout4c4+II_tout4c5)/2'
.meas TRAN II_energyn PARAM='1e-15*(II_vn4b-II_vn4a)*pVdd'
.meas TRAN II_cloadn PARAM='1e-15*(II_vn4b-II_vn4a)/pVdd'
*
.alter
.param pWidthInLambda=6
.alter
.param pWidthInLambda=9
.alter
.param pWidthInLambda=12
.alter
.param pWidthInLambda=15
.alter
.param pWidthInLambda=18
.alter
.param pWidthInLambda=21
.alter
.param pWidthInLambda=24
.alter
.param pWidthInLambda=30
.alter
.param pWidthInLambda=40
.alter
.param pWidthInLambda=60
.alter
.param pWidthInLambda=80
.alter
.param pWidthInLambda=120
*
**
.end
```

Ring Oscillator Characterization: Shell Script

```
#!/bin/csh
#
# Usage: ringpost.csh [spice_output_file].out [matlab_filename].m
```

```
#
#       Note: don't include the .out and .m suffixes, as this prog will do that
#
# Use this by running "hspice ringoscchar.sp > [spice_ouput_file].out" for
# the desired voltage (pVdd) in ringoscchar.sp first, then typing
# "ringpost [spice output file].out [matlab_filename]"
# which will write a file called [matlab_filename].m containing the
# following measurements:
#       tdlh, tdhl, trise, tfall, tp,
#       cgaten (est. gate cap of nmos), cgatep (should = cgaten),
#       energyn (est. energy used by nmos during transition),
#       energyp (same as above for pmos, should be same number),
#       cloadn (est. cap load at inverter node),
#       cloadp (same as above, but calc'ed from pmos current, like energyp)
#
# Note, it may be simplier to run ringoscchar.sp a couple times for
# the different models and voltages and simply save the [spice_output_file]
# with a different name (designating its model and vdd) and leave those
# lying around.
#
# Also note:  Current widths simulated in spice file are:
# 3, 6, 9, 12, 15, 18, 21, 24, 30, 40, 60, 80, 120  (lambda)
# with all L=2 lambda, but check ringoscchar.sp to make sure.

echo "width = [ 3 6 9 12 15 18 21 24 30 40 60 80 120 ]" >! $2.m

echo tdlh" = [ " `grep ii_tdlh $1.out | grep -v meas | awk '{print $3}'` "]" >>
$2.m
echo tdhl" = [ " `grep ii_tdhl $1.out | grep -v meas | awk '{print $3}'` "]" >>
$2.m
echo trise" = [ " `grep ii_trise $1.out | grep -v meas | awk '{print $3}'` "]"
>> $2.m
echo tfall" = [ " `grep ii_tfall $1.out | grep -v meas | awk '{print $3}'` "]"
>> $2.m
echo tp" = [ " `grep ii_tp $1.out | grep -v meas | awk '{print $3}'` "]" >> $2.m
echo cgaten" = [ " `grep ii_cgaten $1.out | grep -v meas | awk '{print $3}'` "]"
>> $2.m
echo cgatep" = [ " `grep ii_cgatep $1.out | grep -v meas | awk '{print $3}'` "]"
>> $2.m
echo energyn" = [ " `grep ii_energyn $1.out | grep -v meas | awk '{print $3}'`
"]" >> $2.m
echo energyp" = [ " `grep ii_energyp $1.out | grep -v meas | awk '{print $3}'`
"]" >> $2.m
echo cloadn" = [ " `grep ii_cloadn $1.out | grep -v meas | awk '{print $3}'` "]"
>> $2.m
```

```
echo cloadp" = [ " `grep ii_cloadp $1.out | grep -v meas | awk '{print $3}'` "]"
>> $2.m
```

Library Cell Characterization: SPICE

Note that only the SPICE for the XOR and register are shown. Other files for AND, NAND, OR, NOR, XNOR, AOI, INV, etc. cells can be easily derived by modifying this template.

XOR Auto Characteriztion File

```
***** XOR spice test file for delay and energy usage
.options nomod acct post=1
.param pVdd=1.5 EDGE=1ns DELAY=2ns THIGH=5ns
.ic v(supplyq)=0 v(inputq)=0 v(supplyqmin0)=0 v(supplyqmax0)=0
*
* Transient simulations*
vdd Vdd 0 dc pVdd

vA A 0 PWL 0 0v DELAY 0v 'DELAY+EDGE' pVdd 'DELAY+EDGE+THIGH' pVdd
+ 'DELAY+2*EDGE+THIGH' 0v 'DELAY+2*EDGE+2*THIGH' 0v
+ 'DELAY+4*EDGE+4*THIGH' 0v 'DELAY+5*EDGE+4*THIGH' pVdd
+ 'DELAY+5*EDGE+5*THIGH' pVdd '2*DELAY+5*EDGE+5*THIGH' pVdd
+ '2*DELAY+6*EDGE+5*THIGH' 0v '2*DELAY+6*EDGE+6*THIGH' 0v
+ '2*DELAY+7*EDGE+6*THIGH' pVdd
+ '2*DELAY+7*EDGE+7*THIGH' pVdd '2*DELAY+9*EDGE+9*THIGH' pVdd
+ '3*DELAY+9*EDGE+9*THIGH' pVdd

vB B 0 PWL 0 0v DELAY 0v 'DELAY+2*EDGE+2*THIGH' 0v
+ 'DELAY+3*EDGE+2*THIGH' pVdd
+ 'DELAY+3*EDGE+3*THIGH' pVdd 'DELAY+4*EDGE+3*THIGH' 0v
+ 'DELAY+4*EDGE+4*THIGH' 0v 'DELAY+5*EDGE+4*THIGH' pVdd
+ 'DELAY+5*EDGE+5*THIGH' pVdd '2*DELAY+5*EDGE+5*THIGH' pVdd
+ '2*DELAY+7*EDGE+7*THIGH' pVdd
+ '2*DELAY+8*EDGE+7*THIGH' 0v '2*DELAY+8*EDGE+8*THIGH' 0v
+ '2*DELAY+9*EDGE+8*THIGH' pVdd '2*DELAY+9*EDGE+9*THIGH' pVdd
+ '3*DELAY+9*EDGE+9*THIGH' pVdd

*
cld OUT 0 50e-15
*
```

```
rdum GND 0 0
rfd1 FEED1 0 0
rfd2 FEED2 0 0
rfd3 FEED3 0 0
rfd4 FEED4 0 0
rfd5 FEED5 0 0
*
fVdd supplyq 0 CCCS vdd 1e-3
cVdd supplyq 0 1F
*
fVdd2 supplyqmin0 0 CCCS vdd 1e-3 MIN=0
cVdd2 supplyqmin0 0 1F
*
fVdd3 supplyqmax0 0 CCCS vdd 1e-3 MAX=0
cVdd3 supplyqmax0 0 1F
*
fVA  inputq 0 CCCS vA 1e-3 max=0
fVB  inputq 0 CCCS vB 1e-3 max=0
cinq inputq 0 1F
*
.tran .05ns '3*DELAY+9*EDGE+9*THIGH'
*
.include 'i_2xor.spi'
*
.include 'hp_0.6um.139'
*
*** T out edge rise driven from A, B low.
.meas TRAN Taroredge TRIG V(out) val='0.1*pVdd' TD=DELAY RISE=1
+ TARG V(out) val='0.9*pVdd' RISE=1
*** T out edge fall driven from A, B low.
.meas TRAN Tafofedge TRIG V(out) val='0.9*pVdd' TD=DELAY FALL=1
+ TARG V(out) val='0.1*pVdd' FALL=1
*** Td A rise to out rise, B low.
.meas TRAN Taror TRIG V(A) val='0.5*pVdd' TD=DELAY RISE=1
+ TARG V(out) val='0.5*pVdd' RISE=1
*** Td A fall to out fall, B low.
.meas TRAN Tafof TRIG V(A) val='0.5*pVdd' TD=DELAY FALL=1
+ TARG V(out) val='0.5*pVdd' FALL=1
*
*
.meas TRAN nop1 FIND V(GND) AT=0
*
*
```

```
*** T out edge rise driven from B, A low.
.meas TRAN Tbroredge TRIG V(out) val='0.1*pVdd' TD=DELAY RISE=2
+ TARG V(out) val='0.9*pVdd' RISE=2
*** T out edge fall driven from B, A low.
.meas TRAN Tbfofedge TRIG V(out) val='0.9*pVdd' TD=DELAY FALL=2
+ TARG V(out) val='0.1*pVdd' FALL=2
*** Td B rise to out rise, A low.
.meas TRAN Tbror TRIG V(B) val='0.5*pVdd' TD=DELAY RISE=1
+ TARG V(out) val='0.5*pVdd' RISE=2
*** Td B fall to out fall, A low.
.meas TRAN Tbfof TRIG V(B) val='0.5*pVdd' TD=DELAY FALL=1
+ TARG V(out) val='0.5*pVdd' FALL=2
*
*
.meas TRAN nop2 FIND V(GND) AT=0
*
*
*** T out edge rise driven from A, B high.
.meas TRAN Taforedge TRIG V(out) val='0.1*pVdd' TD='DELAY+5*EDGE+5*THIGH'
+ RISE=1 TARG V(out) val='0.9*pVdd' RISE=1
*** T out edge fall driven from A, B high.
.meas TRAN Tarofedge TRIG V(out) val='0.9*pVdd' TD='DELAY+5*EDGE+5*THIGH'
+ FALL=1 TARG V(out) val='0.1*pVdd' FALL=1
*** Td A rise to out fall, B high.
.meas TRAN Tarof TRIG V(A) val='0.5*pVdd' TD='DELAY+5*EDGE+5*THIGH' RISE=1
+ TARG V(out) val='0.5*pVdd' FALL=1
*** Td A fall to out rise, B high.
.meas TRAN Tafor TRIG V(A) val='0.5*pVdd' TD='DELAY+5*EDGE+5*THIGH' FALL=1
+ TARG V(out) val='0.5*pVdd' RISE=1
*
*
.meas TRAN nop3 FIND V(GND) AT=0
*
*
*** T out edge rise driven from B, A high.
.meas TRAN Tbforedge TRIG V(out) val='0.1*pVdd' TD='DELAY+5*EDGE+5*THIGH'
+ RISE=2 TARG V(out) val='0.9*pVdd' RISE=2
*** T out edge fall driven from B, A high.
.meas TRAN Tbrofedge TRIG V(out) val='0.9*pVdd' TD='DELAY+5*EDGE+5*THIGH'
+ FALL=2 TARG V(out) val='0.1*pVdd' FALL=2
*** Td B fall to out rise, A high.
.meas TRAN Tbfor TRIG V(B) val='0.5*pVdd' TD='DELAY+5*EDGE+5*THIGH' FALL=1
+ TARG V(out) val='0.5*pVdd' RISE=2
```

```
*** Td B rise to out fall, A high.
.meas TRAN Tbrof TRIG V(B) val='0.5*pVdd' TD='DELAY+5*EDGE+5*THIGH' RISE=1
+ TARG V(out) val='0.5*pVdd' FALL=2
*

.END
```

Register Auto Characterization File

```
****** top level cell is ./i_tspcr3_rst.ext
.options acct nomod post=1
.param pVdd=1.5 EDGE=400ps T=7.8ns
.ic v(QSupply)=0
*
vdd Vdd GND dc pVdd ac 0
vCLOCK CLOCK GND PU (0 pVdd 'T/2' EDGE EDGE 'T/2 - EDGE' T)
vIN IN GND PU (0 pVdd 'T+EDGE' EDGE EDGE '2*T - EDGE' '3*T')
vRST RESET GND PU (0 pVdd 'T/2+2*EDGE' EDGE EDGE 'T/2 - EDGE' '5*T')
*
rdum GND 0 0
rf1 Feed1 0 0
rf2 Feed2 0 0
rf3 Feed3 0 0
*
cout out 0 50fF
*
fVdd QSupply 0 CCCS vdd 1
cVdd QSupply 0 1F
*
.include 'i_tspcr3_rst.spi'
*
.include 'hp_0.6um.139'
*
.tran .05ns '6.5*T'
*
.meas TRAN Trise TRIG V(out) val='0.1*pVdd' TD='1.25*T' RISE=1
+ TARG V(out) val='0.9*pVdd' RISE=1
.meas TRAN Tfall TRIG V(out) val='0.9*pVdd' TD='1.25*T' FALL=1
+ TARG V(out) val='0.1*pVdd' FALL=1
.meas TRAN Tdclk21 TRIG V(clock) val='0.5*pVdd' TD='1.25*T' RISE=3
+ TARG V(out) val='0.5*pVdd' FALL=1
```

```
.meas TRAN Tdclk2h TRIG V(clock) val='0.5*pVdd' TD='1.25*T' RISE=1
+ TARG V(out) val='0.5*pVdd' RISE=1
.meas TRAN Tdrst21 TRIG V(reset) val='0.5*pVdd' TD='1.5*T' RISE=1
+ TARG V(out) val='0.5*pVdd' FALL=2
.meas TRAN Ivddmax MIN i(Vdd) FROM=0 TO='6*T'
.meas TRAN QMoved_3T_1outcycle MAX v(QSupply) FROM='1.5*T' TO='4.5*T'
*

.END
```