High Speed 64-b CMOS Datapath[1]

John Wawrzynek
Bertrand Irissou
EECS, Computer Science Division
University of California, Berkeley
Berkeley, CA 94720

Abstract

This paper reports on our investigations into the performance limits of CMOS datapaths. We have used a combination of single phase clocking, reduced voltage swing logic, moderate pipelining, and custom layout to achieve dramatic speed improvements over conventional design techniques. We have also used a novel fast adder structure and register file. To demonstrate the feasibility and effectiveness of these techniques and circuits, we have designed a test chip including a 64-bit integer datapath and a PLA-based finite state machine for testing. The chip layout was generated using MOSIS design rules and fabricated in the HP CMOS34 1.2-μm process. It has been tested and is fully functional at 180MHz.

1 High-speed circuit techniques

Industry circuit designers commonly use complementary CMOS gates and multi-phase clocking with level-sensitive latches or single phase clocking with edge-triggered flip-flops. These techniques are popular for their robustness to noise and process variations. We have found that abandoning these traditional techniques in favor of a more aggressive design style can lead to significant performance improvements.

Similar performance improvements have recently been reported independently by the DEC Alpha team[4, 3]. Their processor is a 64-bit RISC-style microprocessor that operates up to 200MHz with a 3.3 volt power supply. It was fabricated in 0.75-μm n-well CMOS with three layers of metalization. An accurate performance comparison between our chip and the the Alpha chip is not possible because of differences in the fabrication processes.

Factors under the control of circuit level designers that limit the speed of circuits include overhead due to clock distribution and latches, gate delay, capacitive loading due to busses, and the parasitics associated with transistor drains. The techniques we have used to overcome these limitations and achieve high-speed operation fall into three categories. First, we have

[1]Research supported by the National Science Foundation throught the PYI award MIP-8958568.

systematically employed reduced voltage swing circuitry through the use of precharged logic. At the circuit level, reduced voltage swings help to reduce gate delay and decrease the capacitive delays of long lines. The second technique we use is single phase clocking. The technique relies on both the high and low phases of the clock to achieve the same effect of two-phase clocking. Thirdly, parasitics that limit high-speed operation are reduced through the careful custom layout of circuitry. Large folded transistors are used to counteract the effects of gate resistance and parasitic drain capacitance. Additional care has also been taken to minimize noise due to coupling of the fast switching nodes.

1.1 Precharged Logic—Reduced Voltage Swings

The first technique that we employed is reduced-voltage-swing dynamic logic for the speed-critical nodes. Precharged logic takes advantage of the fact that precharged gates switch closer to the rails than complementary gates. The switching threshold of a normal CMOS inverter is approximately $V_{dd}/2$. This threshold may be increased or decreased by making the pullup or the pulldown transistor wider. However, because the change inverter threshold voltage is proportional to the square root of the ratio of the strengths of the pullup to pulldown, in practice only a limited change is possible. On the other hand, an n-type precharged circuit (pulldown network with clocked precharge to V_{dd}) has a switching threshold much closer to ground, near the transistor threshold for the n-type transistor. In the complementary CMOS case, an input signal that is rising from ground to V_{dd} does not cause the gate to switch until the voltage makes it to $V_{dd}/2$. However, for the precharged case, the output switches at the n-type transistor threshold voltage (around 1.0v for our process). The resulting noise margin is lower than in conventional designs, but problems due to noise can be kept to a minimum by careful layout.

A good example of how this circuit family improves performance is the carry lookahead structure of our adder. We use alternating n-type and p-type NOR gates. During precharge, the n-type and p-type gates precharge high and low, respectively. During evaluation, the n-type gate begins to switch as soon as one of its inputs is above an n-type transistor threshold voltage. In turn, the p-type gate switches as soon as one of its inputs from the n-type gate drops below a p-type transistor threshold voltage. For the technology that we are using for this project, the effective gate delay through this type of circuit is approximately 200ps, less than half of the gate delay of standard CMOS circuits.

We also use precharged logic to speed up the recovery of signals from highly capacitive busses and other nodes—as one might use a sense amplifier. For a bus receiver we use a simple precharged inverter. In the case of a bus that precharges high, we use an inverter that precharges low. Both the bus and the receiver precharge on the same phase of the clock. On the next phase, the bus driver either begins to pull the bus low or allows it to float high. If the bus is pulled low, the p-type transistor of the precharged inverter detects the voltage as it falls lower than the p-type transistor threshold voltage, allowing the inverter to switch. In this situation we make the p-type pullup very wide. The reason is that circuit operation relies on the fact that as soon as the bus voltage falls below the threshold of the p-type transistor it starts conducting. However, the conductance is normally low in that regime; a wider pullup is more effective. Because the bus is a already a highly capacitive node, the extra gate capacitance added to the bus has little effect.

1.2 Single Phase Clocking

The other technique that we use is a form of single phase clocking. The technique was first introduced as "true single phase clocking" (TSPC) clocking [6, 1]. It was demonstrated for very high-speed bit-serial processing and other simple circuits. A single clock is distributed throughout the system, and never inverted. Very simple latches result in fast circuit operation, less latch setup and delay times, and less clock load. Single clock signal distribution also simplifies the task of clock signal generation and distribution.

In the TSPC clocking methodology there are two latch types: a p-latch which is transparent when the clock is low, and latched when the clock is high; and an n-latch which is transparent when the clock is high, and latched when the clock is low. Both types are shown in Figure 1 along precharged gates. As shown, the latches are used to follow two different types of precharged combinational logic blocks. The p-block precharges low when the clock is high, and evaluates when the clock goes low; the n-block precharges high when the clock is low, and evaluates when the clock is high. Each block is followed by a latch of the same type; a p-block is followed by a p-latch and an n-block by an n-latch. If the block is precharging the latch is locked, preventing the precharged value from propagating through to the next stage. The block evaluates when the latch is transparent. Systems built from blocks are arranged in pipelines of alternating p- and n-stages. When the n-blocks are precharging the p-blocks are evaluating and vice-versa. This approach imposes a constraint that cycles must be made from an even number of blocks.

The use of TSPC allows systems with no invertered clocks. While it is possible to build modules in the TSPC methodology using no inverted clocks, in practice it is very difficult to build efficient systems this way. Consequently we have relaxed this no clock inversion rule to allow an inverted clock on precharge transistors and power and ground switches, but not on the latches. This modification retains the nice properties of TSPC and its resistance to skew while permitting simplified circuits. It is simple to show that this modification is safe; the worst potential problem with this modification is extra power consumption.

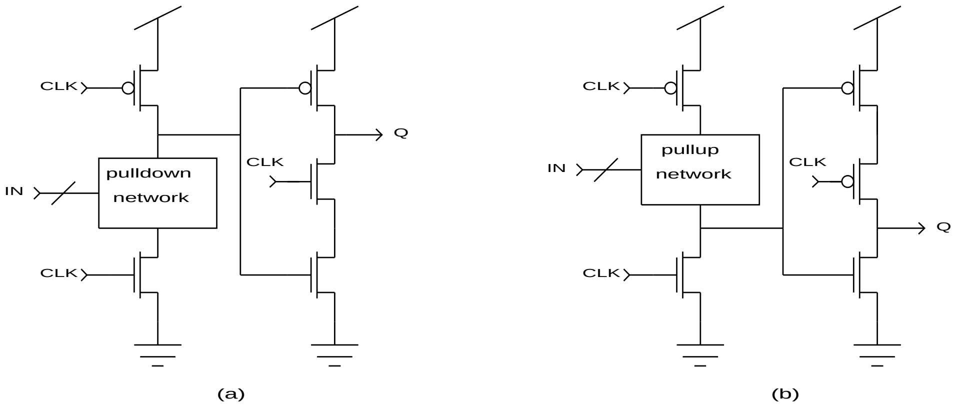

Figure 1: Precharged gates with TSPC n-type latch (a) and p-type latch (b)

2 Datapath Design

We strongly believe that system design and fabrication are imperative to understand the issues involved in successfully fabricating high speed circuits. To demonstrate that the techniques presented are feasible for CPU-sized chip design, we integrated some basic components of microprocessors into a datapath test chip. Clock distribution, power distribution, noise, input and output communication and testing are some of the issues that needed to be solved to make a functional chip.

The experimental 64-bit integer datapath consists of two of the basic building blocks of the integer core of a microprocessor: a 64-bit triple-ported register file and a 64-bit integer adder. Of course, a complete datapath is more complex that this test datapath, but we believe that these blocks are sufficient to demonstrate that our proposed techniques are viable.

Speeding up the carry chain of the 64-bit adder is a challenge for a high speed design. The register file is also difficult to design for high-speed operation because of the highly capacitive bit-lines. The width of the datapath was motivated by the fact that the popular 32-bit architectures are likely to be supplanted by 64-bit architectures. During the design phase we used several circuit-level simulators to help verify the delays in our circuits. The delays listed in this section refer to delays in simulated circuits. A later section in the paper reports on the actual running speed of the fabricated chip.

The basic structure of the test chip is shown in Figure 2. The datapath consists of the register file and the adder. Two buses are used for internal communication. The write-back bus, wbus, is used to write a result from the adder or the register file back into the register file. A bypass bus, bbus, bypasses the register file and forwards results directly into the adder.

As our clocking discipline dictates, each functional block is broken into p and n stages. The timing diagram of Figure 2 shows the operation of the pipeline. After the register addresses are decoded in the low phase, the

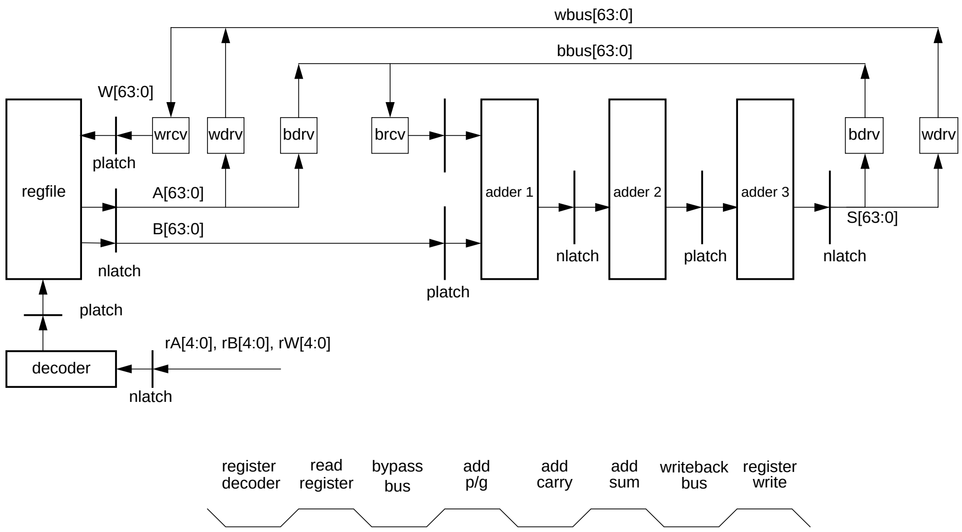

Figure 2: Fast Datapath Block Diagram

register file RAM array is read during the following high phase. The data read is latched on the falling edge of the clock. Then, the low phase is used to drive data on the bypass bus to the adder. The adder then takes three phases to evaluate, but a new addition can be started every cycle. Finally, the result from the adder is driven on the writeback bus during the low phase and the register file is written during the following high phase. All blocks precharge on the phase immediately before they evaluate.

2.1 Register File

The operation of the register file is broken down over two half-cycles. First, the low phase of the clock is used to decode the address and drive one of the 32 row-select lines. The inputs to the row select lines are then latched on the rising edge of the clock. These inputs are stable during the high phase.

For a read operation, the high phase is used by the bit-cell to drive one of the precharged bit-lines; the data is sensed by a cascaded pair of precharged inverters and latched on the falling edge of the clock. For a write operation, the data to be written must be ready by the beginning of the high phase. The write bit-lines are driven differentially during the high phase of the clock to write the cell.

The register file decoder is a p-block. It is basically a NAND decoder. The partial schematic of Figure 3 shows one branch of the decoder tree for selecting register 0. The rest of the tree can be easily completed by symmetry.

On the phase before evaluation, a high phase, the internal address lines in the decoder are grounded, causing the internal nodes of the decoder to be precharged high. Precharging the internal nodes prevents charge sharing

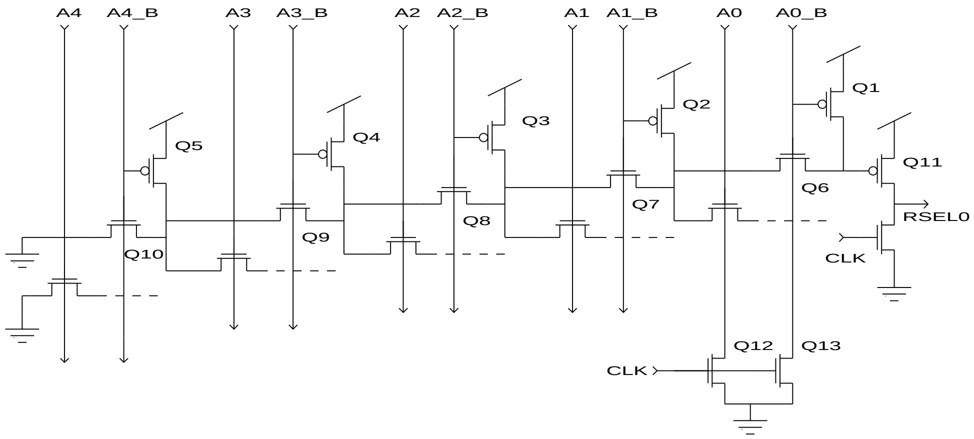

Figure 3: Register File Decoder

in the tree during evaluation. The precharge also turns off all the row select lines in the RAM array to avoid writing the bit-cells. During the low phase of the clock, the internal address lines are driven differentially, turning on only one path to ground in the decoder tree. The speed of the decoder resides in the exponentially increasing pulldown n-type transistor chain constituted by Q6–10. Each transistor is sized twice as large as the previous one in the chain. Also, the large p-type transistor Q11 at the end of the chain acts as a sensing device that turns on as soon as the output of the decoder is a threshold below V_{dd}. The layout of the decoder is very regular. Each decoded bit is made of 5 $\frac{16}{2}\lambda$ ($\frac{9.6}{1}\mu$m) n-type pass transistors and 5 $\frac{16}{2}\lambda$ p-type pullup transistors. The larger pass transistors and pullups are constructed by connecting in parallel 2, 4, 8 or 16 $\frac{16}{2}\lambda$ n-type or p-type transistors at the different stages of the chain. In simulation, the decoder evaluates in 1.7 ns from the time the addresses are asserted.

The array part of the register file is an n-block. Figure 4 shows the circuitry of one bit-cell and the sense circuitry for one port. The basic RAM bit cell is made of the static cross coupled inverters Q1–4. Two pass transistors, Q5 and Q6, provide two single ended read ports. Two additional pass transistors (not shown) are used to differentially write the cell. The bit-cell was carefully sized so that its contents are not altered on a single-sided read. The sense circuitry is precharged by Q7 and Q8 during the low phase of the clock. During the high phase, the bit-lines associated with the select row lines are pulled down or remain high depending on the content of the bit cell. The bit-line is then sensed by Q9. The data is ready on the latch 1.9 ns after the rising edge of the clock.

2.2 Adder Architecture

The adder is pipelined over three phases, each stage separated by alternating p or n latches. The latency through the adder is one and a half cycles. The low-level circuits for the adder are fairly straightforward and consists of a

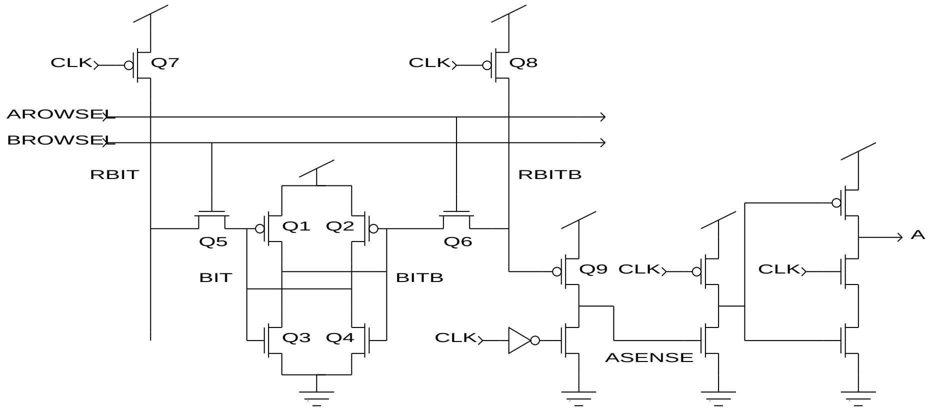

Figure 4: Register File Read Circuitry

Kogge-Stone binary lookahead tree [5]. However, a unique 2 to 1 reduction of the carry chain results in a significant reduction in area without affecting the speed.

Figure 5 shows the block diagram of the adder. The adder comprises three blocks of alternating n, p, and n stages. The first block of the adder, up to the first n-latch, has only three gate delays (in addition to the latch). We create the generate and the propagate signals. Then, the propagate and generate signals of two consecutive bits are combined in the *d-transform* block. The net result is to cut the length of the carry chain in half. Also, the first stage of the Kogge-Stone lookahead binary tree completes within this block. The second block is the last four stages of the binary lookahead tree, where each stage consists of a pair of propagate/generate blocks. In the last block of the adder, the real carry is reconstituted and the final sum calculated.

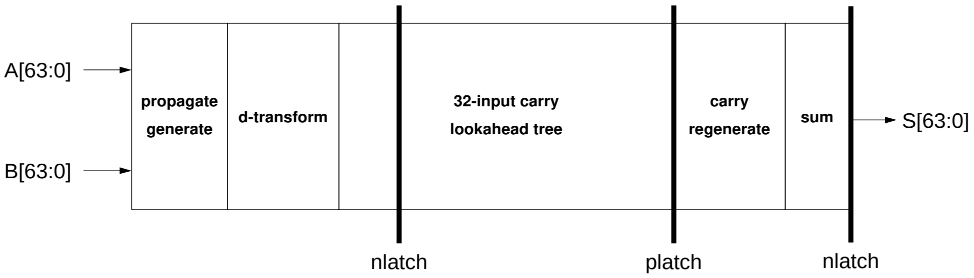

Figure 5: Adder Block Diagram

The 2 to 1 reduction in the carry chain was invented by A. Despain and extended by P. de Dood [2]. The reduction starts by rewriting the carry term of a Manchester carry chain of equation:

$$C_{2n} = G_{2n} + (P_{2n}.C_{(2n-1)}) \quad (1)$$
$$= P_{2n} \cdot (G_{2n} + C_{(2n-1)}). \quad (2)$$

By doing so on every even term, the carry chain can be modified to have adjacent OR function and adjacent AND function rather than alternating AND and OR. We can now combine these two consecutive AND or OR into one gate by creating the P'_n and G'_n terms:

$$P'_n = P_{(2n+1)} \cdot P_{2n} \quad (3)$$
$$G'_n = G_{2n} + G_{(2n-1)}. \quad (4)$$

The modified carry chain is shown in Figure 6; it is half the size of the original chain. Notice that the carry chain term are now C'_n and not C_n. The carry terms can be easily reconstructed once the carry chain has evaluated as:

$$C_{2n+1} = G_{(2n+1)} + C'_n \quad (5)$$
$$C_{2n} = P_{2n} \cdot (G'_n + C'_{(n-1)}). \quad (6)$$

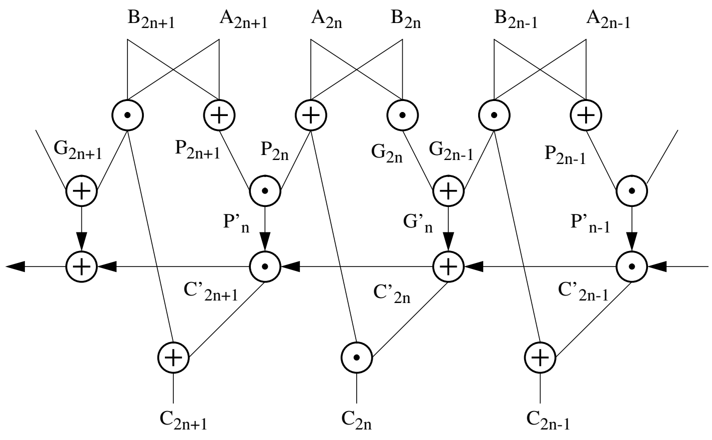

Figure 6: D-Transform

The result of applying this transformation is that, with the same gate delay, a 32-bit carry lookahead tree can be used instead of the much larger 64-bit tree.

3 Chip Layout and Implementation

Figure 8 shows a die photo of the datapath. The die is 6mm on a side. In the center the adder is $5100 \times 4000\lambda^2$ ($3060 \times 2400\mu\text{m}^2$). Pitch-matched to its left, the register file measures $5100 \times 2000\lambda^2$ ($3060 \times 1200\mu\text{m}^2$) for the RAM array and the sense and write circuitry. The decoders are to the top and bottom of the register file. The bus logic and shift registers are located to the right of the adder. To the top, a PLA state machine is used to control the operation of the datapath.

There is a triple ring of power, ground and clock circling the entire chip. It is distributed in the datapath horizontally in first layer metal interconnect (metal-1) and then down to the particular cells in metal-2. Metal-2 is used to run power lines over the tops of the cells; this organization permits us to keep the wires as wide as possible, and helps to minimize resistive effects. This is important in power and ground distribution (to prevent resistive voltage drops) and in the clock wires (to minimize clock skew). The other advantage to running power and ground perpendicular to the dataflow is that it helps in minimizing the effects due to ground noise in one stage effecting the next. Noise coupling between stages was of particular concern to us because we have dynamic nodes, surrounded by wide transistors with high currents.

Power supply inductance issues were a major challenge in the design of the test chip; our goal was to keep the bounce on the supply rail to no more than 500mV. After our first fabricated chip failed because of bounce problems, we added on chip bypass capacitors. By adding about 3nF of gate-oxide capacitance between power and ground, we were able to bring down the instantaneous clock switching current from 3.8A to 1.8A, limiting the bounce on the internal power and ground rails. Because the 84-pin PGA package used has no dedicated power or ground plane and has over 10nH of inductance per pin, all but 12 pins are used to supply power.

The chip layout was generated using MAGIC. Each block was simulated in its entirety using CaZM, which has allowed us to simulate up to 20,000 transistors, at the circuit level. We used IRSIM at the chip level to verify the function of the chip. Finally, we used HSPICE to confirm the timing of some of the critical paths of the chip.

The design process was complicated by the lack of adequate design tools. We were forced to rely on manual calculation and overly conservative design in several areas. Minimizing resistive effects in power and clock rails was a particular concern to us and, where possible, we over-sized these rails. Another layout issue of concern was in node-to-node capacitive coupling. Our layout tools did not accurately extract inter-node capacitance nor did our timing simulators efficiently estimate induced voltages. A tool that could flag trouble spots in a layout based on extracted coupling capacitance and estimated rise and fall times would be extremely useful.

The clock distribution is unique. We distribute the clock driver around the pad ring; close to the power and ground leads that supply the power to the driver and to the on-chip bypass capacitors. There is a U-shaped clock node that surrounds the core. The clock signal is brought into the core from both sides. This configuration is an attempt to minimize clock skew due to physical distance. The worst case delay is to the middle of one piece of the datapath. This represents an insignificant transmission line delay of 35ps. The more difficult problem with clock distribution is RC delays and mismatched buffer delays. We avoid mismatched buffer delays by having a single clock buffer. To minimize RC delays on the wires, we kept the clock wires very wide, throughout the clock distribution. The goal was to have one equipotential clock node. By keeping the wires wide we minimized the resistive effects, attempting to keep the clock node as one large capacitor. Of course, by making the clock node wire wide, it adds area and capacitance to the node. However, the clock node is inherently high capacitive, so adding more capacitance is not very significant. The total clock node capacitance in our case is 0.38nF.

In simulation the chip dissipates an estimated 9 Watts at 250MHz and power supply of 5V. Over a third of the 9 W is used by the clock driver (3.6 W). In comparison, the adder consumes 2.9 W, the register file, 1.8 W, and the test circuitry, 0.6 W.

4 Testing

Testing at high speed is a difficult task. We opted for a test strategy where the internal datapath is continuously clocked at high speed and controlled by an internal PLA. The *test id* and *reset* signals are applied from the pins to start the controller. When the program completes, a *done* signal is asserted. Data is shifted in and out at 15MHz via two serial shift registers, buffer_in and buffer_out. Figure 7 shows the complete chip block diagram.

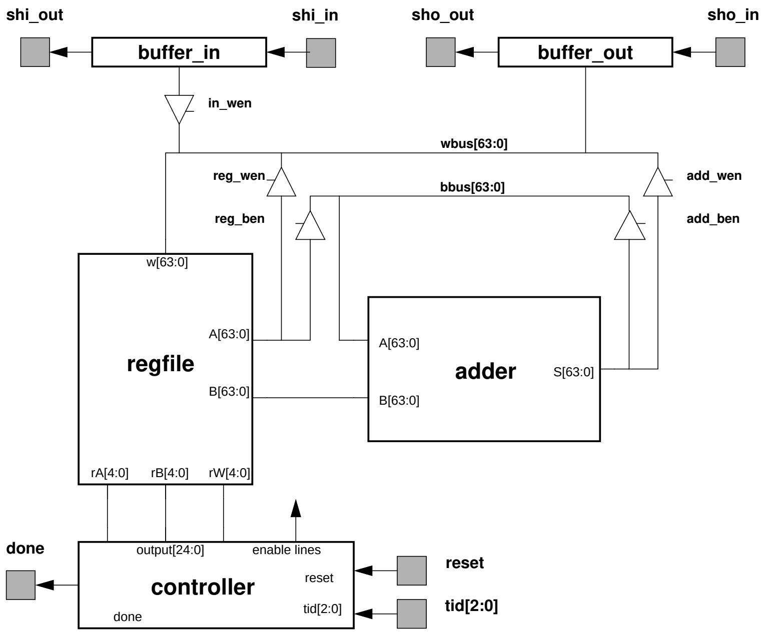

Figure 7: Fast Datapath Test Setup

5 Results

The design began as a class project in the Spring 1991. The layout was completed using MOSIS design rules and submitted for fabrication in the HP CMOS34 1.2-μm (1.0-μm gate length) in early December 1991. The first silicon was received from MOSIS in mid-January. Problems with this chip were traced to excessive V_{dd} and ground bounce. We added on-chip bypass capacitors and refabricated. The second chip returned from fabrication mid-May 1992.

The second fabrication of this design was successfully tested at 180MHz at $39°C$ and 5.0V power supply. A HP 8082A high speed signal generator was used to generate externally the high speed clock. A Tektronix DAS 9100 was used to drive the test id and reset signals, and to control the input and output shifters. The maximum operating frequency of the chip is limited by the adder. The register file works up to 210MHz and the PLA-based controller functions up to 230MHz. The power consumption is 1.4A at 180MHz.

Acknowledgments

We would like to thank all of the members of Spring 1991 CS254 class, in particular Krste Asanović, Brian Kingsbury, and Stuart Kleinfelder. This work was funded by the National Science Foundation through the PYI award, MIP-8958568.

References

[1] Morteza Afghahi and Christer Svensson. A Unified Single-Phase Clocking Scheme for VLSI Systems. *IEEE Journal of Solid-State Circuits*, 25(1):225–233, February 1990.

[2] Alvin Despain and Paul de Dood. Unpublished Notes on Fast Arithmetic, UC Berkeley. January 1989.

[3] Daniel Dobberpuhl et al. A 200-MHz 64-b Dual-Issue CMOS Microprocessor. *IEEE Journal of Solid-State Circuits*, 27(11):1555–1565, November 1992.

[4] Daniel Dobberpuhl et al. A 200mhz 64b Dual-Issue CMOS Microprocessor. In *IEEE International Solid-State Circuits Conference*, pages 106–107, February 1992.

[5] Tackdon Han, David A. Carlson, and Steven P. Levitan. Fast Area Efficient VLSI Adders. In *IEEE International Conference on Computer Design*, pages 418–422, October 1987.

[6] Jiren Yuan and Christer Svensson. High-Speed CMOS Circuit Techniques. *IEEE Journal of Solid-State Circuits*, 24(1):62–70, February 1989.

Figure 8: Test datapath chip die photograph.