Two FIFO Ring Performance Experiments

Charles E. Molnar, Ian W. Jones, William S. Coates,
Jon K. Lexau, Scott M. Fairbanks, and Ivan E. Sutherland

Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, CA 94303

Abstract

Asynchronous circuits are often perceived to operate slower than equivalent clocked circuits. We demonstrate with fabricated chips that asynchronous circuits can be every bit as fast as clocked circuits. We describe two high-speed FIFO circuits to compare the performance of asynchronous FIFOs with that of conventionally-clocked shift registers. The first FIFO circuit uses a pulse-like protocol, which we call asP, to advance data along a pipeline built out of conventional latches. Use of this protocol requires careful management of delays within its circuits. The second FIFO circuit uses a transition signaling protocol and special transition latches to store its data. These transition latches are fast, but are about 50% larger than conventional latches. Measurements obtained from chips fabricated in 0.6µm CMOS and from simulations using Spice show that the throughput of the first asynchronous FIFO design matches that of a conventional two-phase clocked shift-register design, with a maximum throughput of 1.1 Giga data items per second. The throughput of the second design exceeds the performance of the asP* design, and achieves a maximum throughput of 1.7 Giga data items per second. We have extensively tested the chips at different voltages, temperatures, and data conditions and have found them to operate reliably over a very much wider range of operating conditions than is typically specified for clocked circuits.*

1. Introduction

One might worry that an asynchronous circuit must be slower than its clocked synchronous counterpart. Indeed, we did initially share this concern, which has often led to avoidance of the use of asynchronous circuits in applications where speed is the primary design criterion. Data circuits used in clocked designs are highly tuned to exploit as much of the safe interval between clock edges as possible, and rely upon knowledge of local circuit delay relationships to achieve high speeds. Typically, in an asynchronous circuit, locally-generated control signals replace the clock signals used to operate the data latches of the synchronous circuit. Asynchronous circuits use a form of handshake to manage the safe transfer of data between blocks. Our early attempts to design fast control circuits that employ such handshake protocols produced discouraging results, because adhering to the strict handshake signaling protocol resulted in excessive timing overhead in generating the local control signals. We achieved high speed circuits only by turning our attention to control circuit designs that employ less strict signaling protocols. Our circuits, like their synchronous counterparts, exploit knowledge of local circuit delay relationships. We show that by careful design and management of local timing constraints, these asynchronous control circuits can operate every bit as fast as their clocked counterparts.

We chose as our design target the simplest form of asynchronously-controlled data path, a linear First-In-First-Out data buffer circuit, or FIFO. We compared the FIFO performance with that of a conventionally-clocked shift register. Of course, a FIFO has greater functionality than a shift register, in that the FIFO is naturally elastic and can hold varying quantities of data, while the clocked shift register must be filled completely before data appears at its output. Furthermore, an asynchronous FIFO has a much wider range of applicability. For example, unlike the clocked shift-register which is restricted to lie within a single fixed clock domain, the asynchronous FIFO can bridge separate clock domains.

Simulated and measured results for our first FIFO design which employed the asP* control protocol, demonstrate speed equivalent to the clocked shift register circuit. The asP* control circuits are very small and offer negligible area overhead compared to the clocked shift-register. Our second design, which uses special transition latches, achieves 50% faster performance but requires about 50% more chip area.

We fabricated chips for both designs in 0.6µm CMOS and have carried out extensive measurements of the chips over a wide range of operating voltages, temperatures, and data conditions. The performance of the clocked shift register was estimated from simulation to be 1 GHz. The measured maximum throughput of our asP* FIFO design, operating as a ring with Vdd of 3.3V and at ambient temperature, was in the range of 804 to 946 million data items per second. The FIFO ring contained an external interface stage which was the performance bottleneck. Estimates extrapolated from our measurements predict that the internal stages of the ring can achieve throughput rates of 930 to 1126 million data items per second. The measured maximum throughput of our transition FIFO design was in the range of 778 to 857 million data items per second. Again, the interface stage was the bottleneck, and estimates extrapolated from our measurements predict that the internal stages of the ring can achieve throughput rates of 1569 to 1731 million data items per second.

Other studies of the speed of asynchronous latch control circuits, [1], [5] and [6], have also found significant speed improvements when relaxing the strict request/acknowledge handshake protocols. However, the control cycle times reported in these other papers, even when accounting for the differences in fabrication technology, appear to be substantially longer than those for equivalent clocked circuits.

2. Goals

The primary goal of our experiments was to show that an asynchronous FIFO can operate as fast as a conventionally-clocked shift register. FIFOs are well suited to an asynchronous implementation, with the advantages over their synchronously-clocked shift register counterparts of elastic buffering, easy interfacing, and elimination of long clock paths. In the first design, we used the same data path latches as one would for a clocked design, while in the second design, we used special transition latches. These special latches provided a speed improvement of 50%, but are about 50% larger than conventional latches.

The experiments also had several secondary goals. First, we sought to exercise a path from design to fabrication. Second, we needed to develop test procedures that evaluate the performance of our circuits, where the internal circuit speeds are considerably greater than signal speeds that can be brought off-chip. Third, we wished to correlate the results of Spice simulation with measured circuit performance. Fourth, we wished to demonstrate the high performance predicted by our simulations and to test the robustness of the circuits using the “ultimate verification method” of building real circuits.

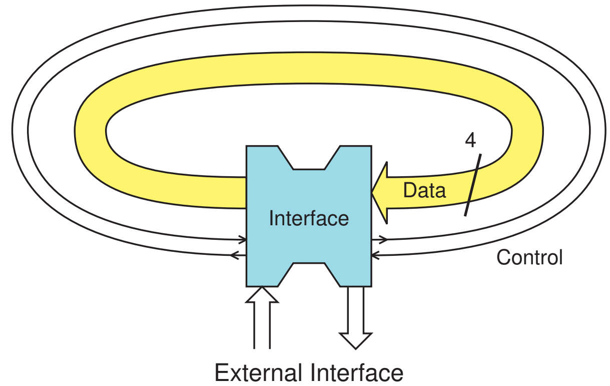

Figure 1. 17-stage FIFO ring experiment.

3. Signaling Protocols and Implementation

Many asynchronous FIFO designs employ a Delay-Insensitive request/acknowledge handshake protocol between adjacent stages. While Delay-Insensitive interfaces operate correctly without regard to circuit delays, this is often at the cost of lower speed. We aimed to achieve best speed by careful attention to local timing relationships, and have not adhered strictly to Delay-Insensitive protocols. Next, we selected the signaling protocol implementation. One common scheme is the four-phase, or return-to-zero, protocol, where a HI voltage level on a wire indicates a request or acknowledge event, and the voltage level on the wire must return to LO before a second event can be sent. Another common implementation is transition, or two-phase, signaling, in which every voltage transition, either LO to HI or HI to LO, indicates an event. Transition signaling has the potential to be faster than four-phase signaling because every transition carries an event, and there are no “inactive” transitions on the wires. However, conventional data latches require level control signals, and when used with transition-signaling control logic, require two-phase to four-phase conversion circuits which add to complexity and delay.

Another departure that we have made from common asynchronous design practice is control *kiting*. We took this name from the practice of check kiting, in which a check is drawn on an account with insufficient funds, trusting that by the time the check clears the money will be available. In our circuits, kiting means that we send control signals to announce the validity of data ahead of the data, with the knowledge that the data will arrive just in time for the control and data to interact. Control kiting imposes further constraints upon the local timing relationships between the control and data circuits.

For our first FIFO design, we chose a home grown, pulse-like signaling protocol to link stages of the FIFO named the *Asynchronous Symmetric Persistent Pulse*

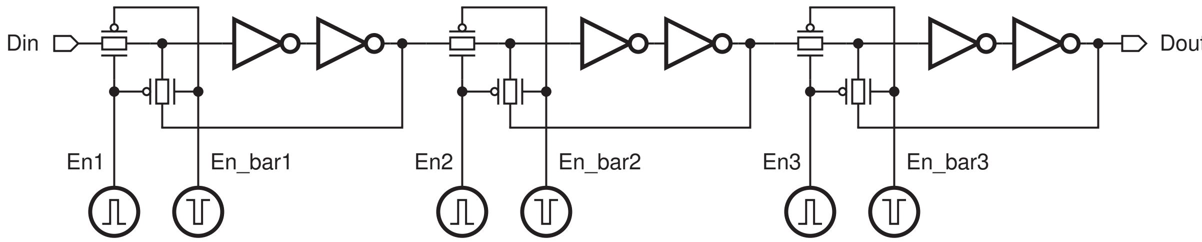

Figure 2. Data latch study - Magic Control

Protocol, or asP*. The abbreviation leaves open the option of adding further qualifiers that begin with the letter P. The asP* protocol efficiently synchronizes the movement of data from one stage in a FIFO to the next. This protocol is very much like four-phase signaling, except that a number of local delay constraints are imposed upon the control logic. In our implementation of the asP* protocol, the control signals used to communicate between the control circuits of adjacent stages can also be used to operate the data latches without requiring any form of signal conversion. The asP* protocol is not Delay-Insensitive, and requires maintenance of particular delay relationships between adjacent stages. These delay relationships are readily achievable in the FIFO design, because they involve only local delays.

Our second FIFO design uses transition signaling, and the transition dual data-path Micropipeline latch presented by Sutherland [9]. We improved the speed by implementing all the transition control signals in both true and complement, dual-rail form. In this form, one can negate a signal either with a pair of inverters, one on each of the twin wires, or simply by interchanging the wires. We call these two forms of negation “real inversion” and “swap,” respectively. In the design of the dual-rail control logic, we took care to ensure that each control loop has an odd number of “real inversions,” because this avoids undesirable stable states in the circuits.

4. Experiment Structure

Testing high-speed circuits poses difficult problems. The speed of the control and data signals internally on the chip far exceeds the speed at which the chip pad driver circuits operate. Thus a test chip with a straight FIFO using off-chip inputs and outputs means that the chip pins and associated pad circuits would limit our ability to measure the maximum performance of the internal circuits. Furthermore, even if we could generate and deliver high speed signals to the chip, we would then require very high speed test and measuring instruments.

We designed each test circuit as a ring of 17 FIFO stages with a 4-bit wide data path as illustrated in Figure 1. This ring structure provided an effective way to verify the performance and robustness of our designs. Each ring included a test interface that required only a few external input and output connections. We designed three chips, two using the asP* control circuits with conventional data latches, and a third chip design using Micropipeline control and transition data latches. The two asP* chips differed only in the external interface circuits used. We fabricated two batches of 25 chips each for the asP* design, and a third batch of 25 chips for the Micropipeline design. The asynchronous character of our interface design to the FIFO ring made it possible to exercise the test circuit at full speed while placing no performance demands on test apparatus.

5. Data Latch Study

For the asP* FIFO design, we began by examining the data path of a simple FIFO comprised of a sequence of conventional static data latches as shown in Figure 2. The figure shows three stages, with the enable inputs to the latches in each stage driven by *Magic Control*; that is, complete freedom in driving the gate inputs was permitted in order to find the control phasing and duty cycle that would produce the maximum throughput. Spice simulation was used to model independent pulse voltage sources for each data latch stage. For this study, we made no allowance for layout-dependent parasitics.

This standard data latch circuit served as our “straw man” to provide a specific target for the control studies. In simulation experiments, we adjusted the phase offset from stage to stage, but held constant the frequency and waveform of the enable signals to each stage. Symmetric waveforms with equal high and low durations worked well. Near-optimal performance required an 8-phase “clock” with a cycle time of 800 picoseconds for each pulse generator, with the enable signals into each stage lagging that of the previous stage by 300 picoseconds. However, we chose a 400-picosecond offset between stages for subsequent studies, corresponding to a simpler 2-phase clock, and offering more generous timing margins. An additional pulse voltage source, operating at half of the clock rate with adjustable phasing relative to the gate

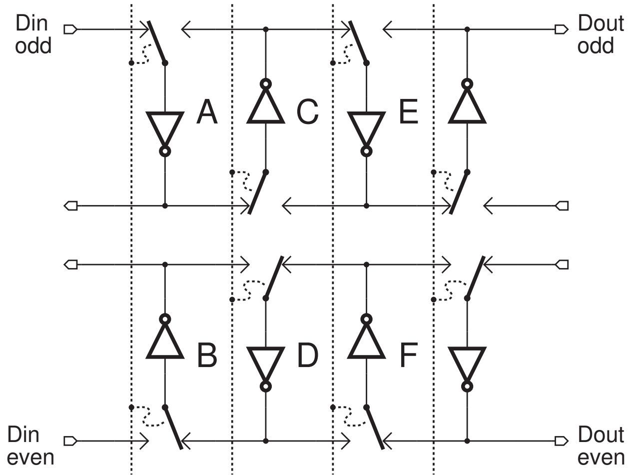

Figure 3. Four stages of Micropipeline transition data latches.

signals, provided alternating 0's and 1's as data inputs to the first stage.

The Magic Control paradigm reveals the ultimate speed of the chosen data path. Nowhere in the Magic Control study did we specify whether the enable signals would come from a source designed in the synchronous or asynchronous style. If we meet the needs of the Magic Control with an asynchronous control mechanism, we are confident that it operates as fast as a shift register with a conventional clock. Our asP* asynchronous control mechanism exploits nearly all of the potential speed of the data path.

6. Micropipeline Latch Design

Our second FIFO design employed Micropipeline transition data latches. Four stages of a 1-bit transition latch data path are shown in Figure 3. The data path has two sections, shown as the upper and lower sections in the figure. Data elements enter the upper and lower sections alternately. The switches in each stage are ganged between the upper and lower sections, as shown by the dashed lines in the figure, and are operated by transition control signals, which are not shown. The switches appear in the figure as they would in an empty FIFO, where the first data element enters the upper section of the data path. In this initial condition, the upper data path section is transparent, with the data flowing from left to right. After the left-most transition control signal flips both the left-most switches, the first data element will be stored in the loop labeled "A." The transition control signals ripple towards the right, changing the state of the switches as they go. This successively copies the data to the next stage to the right and breaks the two-inverter loop of the present stage. When all switches have changed state, the data will be stored in loop E, and the lower data path section will be transparent to the second data element.

Note that we did not carry out an equivalent Magic Control study using the transition latches to determine the performance limits of the data path. However, we expect that a multiple-phase clocked version of this design, operating as a shift register, could achieve comparable throughput.

7. asP* Control Design

We used the gate waveforms resulting from the conventional data latch study as targets for the design of an asynchronous control structure to replace the stipulated Magic Control. When the pass gates in one stage switch relative to when the pass gates of the previous stage switch, is a critical factor in performance. Any misadjustment here either adds to the cycle time or causes circuit errors. One can adjust the clock rate in a clocked design to determine the maximum safe rate of operation—a luxury denied asynchronous circuits, where the design fixes the timing relationships and must allow for variations in fabrication and operating conditions.

The basic principle of operation of a FIFO is very simple: whenever a stage that is *Full* is followed by a stage that is *Empty*, we copy the datum from the full stage to the empty stage and change the control states of both stages. We call this joint action of two stages a *Move* operation. We identify the full state of a stage as the *Opaque* condition, in which the data latch holds its present value independent of its data input, and its empty state as the *Transparent* condition, in which the data latch passes an input data value through to its output. We specifically avoid the terms *open* and *closed* as being ambiguous.

We use a simple set-reset flip-flop with states Empty and Full to hold the control state of each stage. A circuit with three NAND gates and two inverters per FIFO stage, as shown in Figure 4, is logically sufficient to construct both the set-reset gate latch and the combinational function for generating the set-empty and set-full signals needed to carry out the move actions. Outputs from the two sides of the set-reset flip-flop provide the complementary signals needed to operate the pass gates of the data latches and thus determine directly the Transparent or Opaque condition.

We used Spice simulations to adjust circuit parameters to give the latch gate signals appropriate rise and fall times and duty cycles, and to verify that data moves properly through long chains of circuits. An earlier version of the circuit, which omitted the inverters and used the opposite outputs of the flip-flops, was very sensitive to timing variations when used in long chains. The sensitivity was due to the presence of undesirable stable states in the

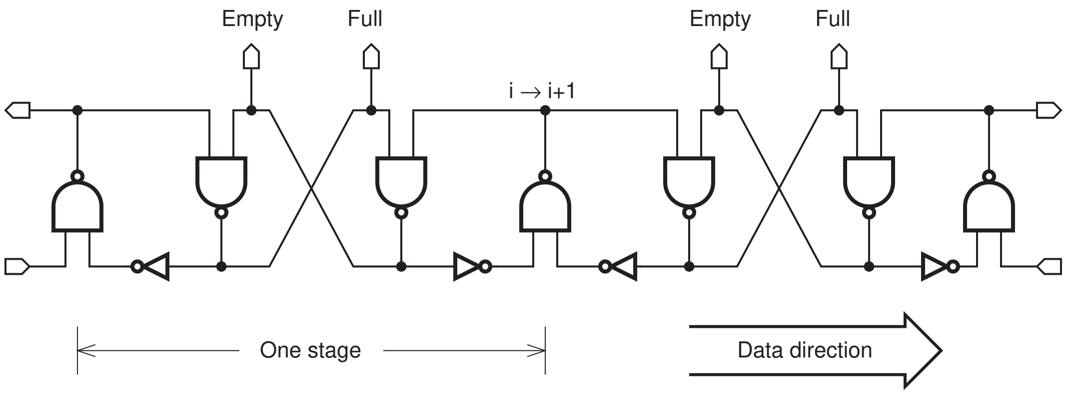

Figure 4. asP* FIFO control circuit.

control circuit loops—states that would not be reached in normal operation. This discovery led us to the circuit design shown in Figure 4, which has no undesirable stable states and is less sensitive to timing variations.

8. Micropipeline Control Circuit

The Micropipeline control circuit uses a chain of Muller C-elements, as illustrated by the sample 2-stage circuit of Figure 5. A C-element is a logic gate that carries out the AND function for transition signals; it produces a transition on its output only when all of its inputs have received a transition. The usual implementation of a 2-input C-element produces a LO output only when both its inputs are LO, and produces a HI output only when both its inputs are HI. Until both inputs have changed, the C-element maintains its previous output value.

In the FIFO control circuit, each C-element output controls a pair of switches as shown by the dashed lines in Figure 3.

9. Details of the FIFO Rings

A common approach to deal with slow chip output pads is to build a ring of N circuit stages designed such that the ring can be made to oscillate. In this way signals having one Nth of the frequency of one stage can be tapped off from the ring and brought off chip. We have used this technique in the past and it works very well for simple circuits such as chains of inverters and basic gates. However, to test a FIFO ring, we need to initialize the ring with a specific number of cells filled with certain data values, and to read the data values back out to verify correct operation. To allow initialization, loading, verifying, unloading, single-stepping, starting, and stopping of the FIFO ring to be controlled at a leisurely rate, we introduced a special interface stage into each ring design. In the interface stage, we incorporated a 1-bit rotation of the data path bit positions. This facilitated loading the data path serially and also provided a convenient way to produce low-speed signals that allowed us to observe the performance of a FIFO ring externally.

9.1. asP* Interface Stage Design

In the asP* control ring, a total of eight external input and two external output control and data connections are required. Figure 6 shows three stages of the asP* FIFO ring test structure, where stage 0 is the interface stage. The complete FIFO ring consists of a 17-stage, 4-bit wide FIFO. Stages 2 though 15, not shown in Figure 6, abut between stage 1 and stage 16 to complete the ring.

Figure 7 shows further detail of the control circuits of stages 0 and 1. Notice that very little additional control circuitry is needed to implement the required interface functionality. External PUT and TAKE control signals, in conjunction with the NFILL and NEMPTY control signals, are used to load and unload the data serially. A 1-bit MUX in the data path permits external data to be loaded into the ring.

To load data into an empty ring, the ring is first stopped by setting both TAKE and PUT to LO. The data value is presented to XD0in, with XSEL set to select the external

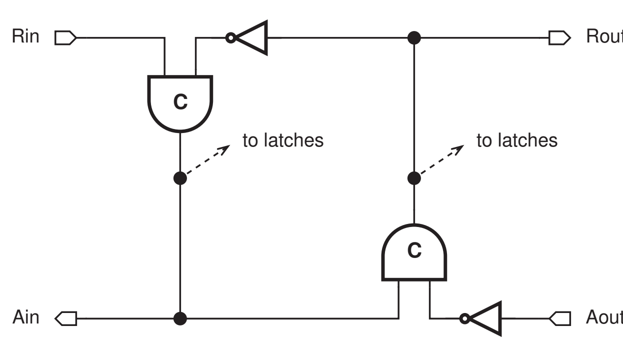

Figure 5. Micropipeline control circuit.

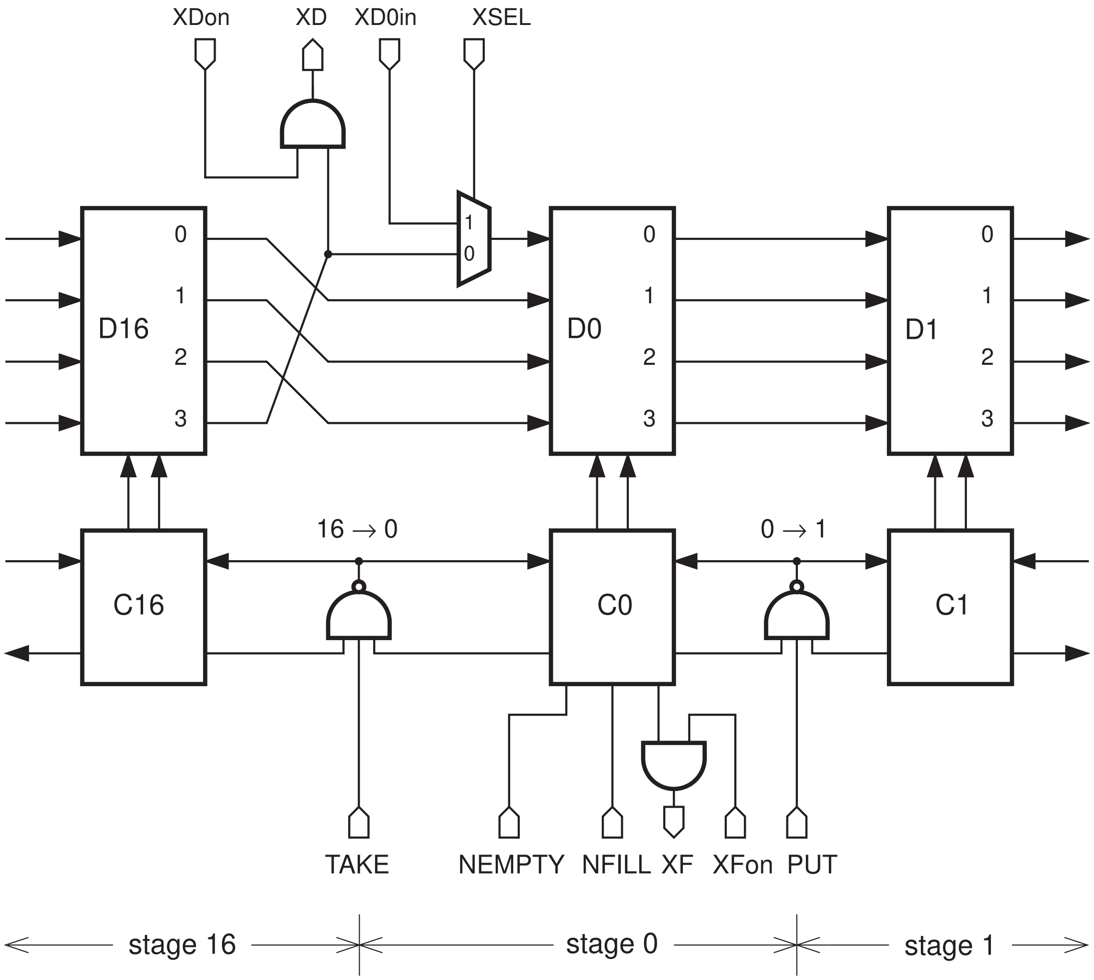

Figure 6. Three stages of the 17 stage asP* FIFO ring.

data. The data is stored in stage 0 by pulsing NFILL to LO and returning it to HI again. Next, the PUT signal is pulsed HI and then returned to LO again, thus permitting the data to move from stage 0 into stage 1, from where it will move round the ring at its own pace, stopping at stage 16. After loading the desired number of items, single-stepping them past the interface exposes the remaining 3 bits of each item for examination and modification. Unloading the ring is carried out in a similar fashion, and is achieved by deleting successive data stored in stage 0 by manipulation of the NEMPTY and TAKE signals.

To monitor the internal state of the interface stage, one control wire, XF, and a single data bit, XD, connect to output pads. These signals can be gated off to avoid high pad driver currents when running the ring.

9.2. asP* Data Path Design

The upper part of Figure 6 shows three 4-bit data latches, D16, D0, and D1. The 4-bit wide data path is arranged into a single loop 68 latches long by rotating the data bits by one position between stages 16 and 0 as shown in this figure. A MUX is provided to select either external data input via signal XD0in or closure of this loop. Data bit 3 of stage 16 is brought out as signal XD to observe the data bits in the loop. Output signal XD can be enabled or disabled by input signal XDon.

The number of data bits stored in the ring is four times the number of full cells. The data bits are loaded serially as suggested in Figure 6. For the ring to run, there must be at least one full and at least one empty stage, i.e., in a running 17-stage ring, the minimum number of full cells is 1, the maximum number of full cells is 16, and the maximum data capacity is 64 bits.

9.3. asP* Timing Conditions

To determine the delay conditions that must be satisfied for reliable operation, we used a mixture of ad hoc analysis and Spice simulation. The simulations revealed the

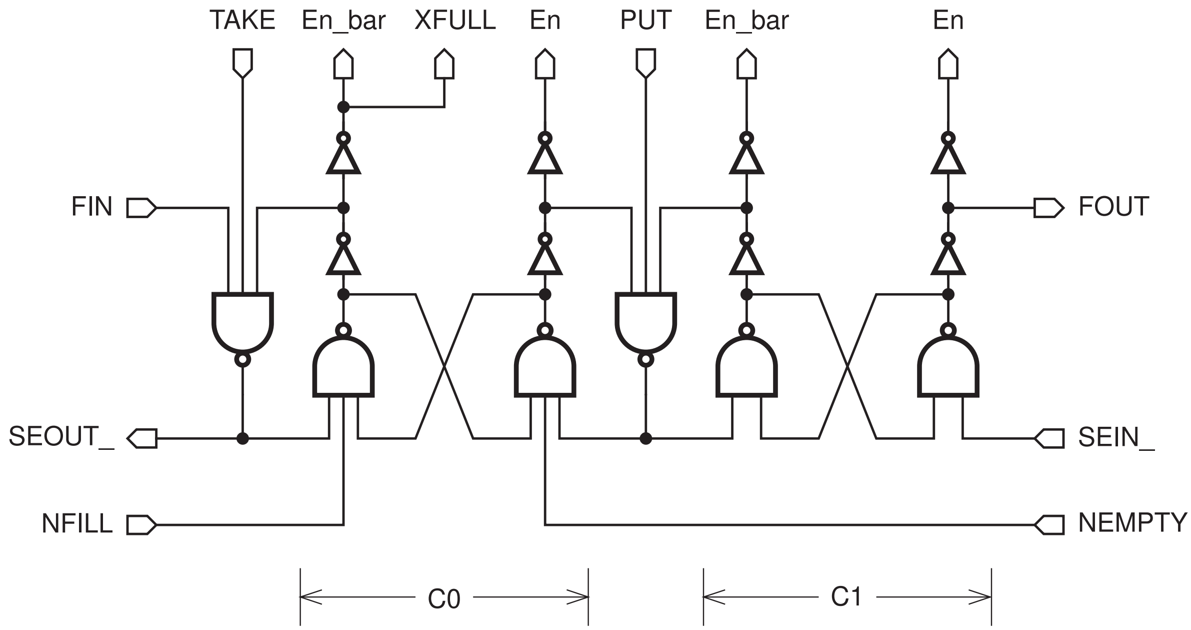

Figure 7. asP* FIFO interface control circuitry.

robustness gained by taking full and empty signals from the control latches via inverters rather than by direct connection to the "complement" side of the set-reset latch, as illustrated in Figure 4. Myers [8] has studied the systematic use of knowledge of delays to enforce orderings required by specifications. Our approach is in the same spirit but relies on continuous circuit simulation rather than gate-and-delay modeling and analysis to achieve robustness. Greenstreet [7] has illustrated an approach to determining such delay conditions using ideas from dynamical systems theory first presented by Brockett in 1989 [2].

9.4. Micropipeline Interface Design

Figure 8 shows 5 stages of the 17-stage Micropipeline ring, including the interface. In this ring the external interface is between stages 16 and 0, as compared to the asP* FIFO ring, where the interface is built into stage 0. In this figure, the dual-rail control signals are shown as single dashed lines for clarity, and the data path is shown in solid lines. The control signals are initially LO except as marked, and the data switches that they control all appear in their initial positions. Only one bit of the 4-bit data path is shown, and as with the asP* FIFO ring, the 4 data bits pass through a 1-bit rotate at the interface. Data can be loaded into the data path serially from the external input XD0in, via switches controlled by the external signal XSEL.

The external control signals used to stop, single-step, start, and delete data cells operate switches P and T shown at the top and bottom of the figure. The P switch controls the *putting* of data into stage 0, while the T switch controls the *taking* of data from stage 16. These switches perform functions similar to the PUT and TAKE signals in the asP* FIFO design. Figure 9 shows the dual-rail circuitry implied by Figure 8 for the stages surrounding the interface logic.

The P and T switches are used in three settings. The first setting is to pass the true and complement signals directly. The second setting is to swap the true and complement signals, thus effectively inverting the sense of the control signals. In the third setting, the switch acts as an opaque latch and blocks both the true and complement signals. Note that the T switch icon is shown with a crossed pair of arrows to indicate that the initial state of this switch has the signal wires exchanged. The switches can be implemented by an arrangement of four transmission gates and a keeper circuit, consisting of a pair of weak inverters, with two control signals to operate the transmission gates as shown in Figure 10.

10. Test and Measurement

10.1. Chip Initialization

The control design of the asP* chips did not require a global master clear signal, because manipulation and monitoring of the control signals of the interface stage could be used to initialize the chip.

The initialization of the Micropipeline ring is a little different. A global master clear signal is asserted to force all the control circuits into their appropriate initial state.

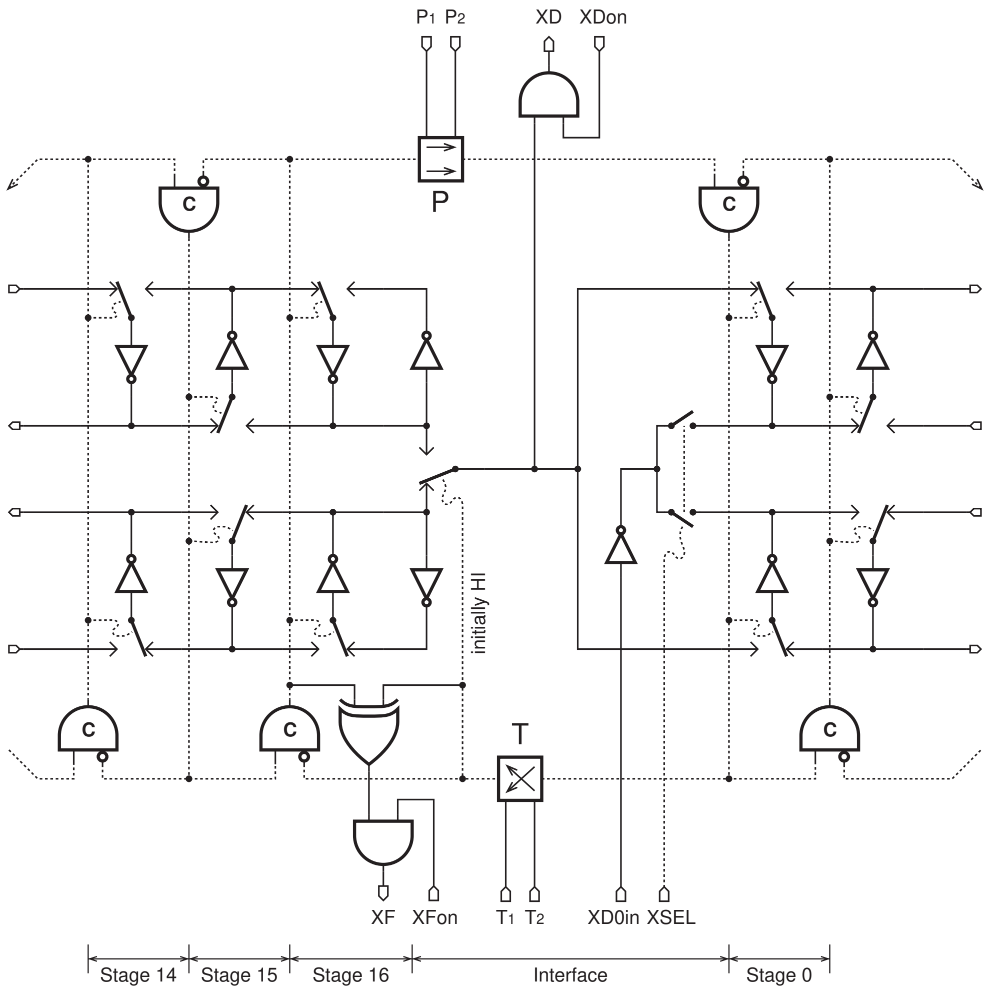

Figure 8. Micropipeline interface.

10.2. Single-stepping the FIFO Rings

The ability to single-step the FIFO rings is key to the initializing, loading, and verifying of data in the rings. In both designs, the external interface controls provided an elegant way to single-step the rings. In the asP* FIFO design, it is carried out by manipulation of the PUT and TAKE interface controls, while in the Micropipeline ring, the P and T switches are used.

10.3. Performance Testing

The speed of the FIFO rings was measured by loading the data path with a pattern of half 0's followed by half 1's to produce the lowest frequency output on the data monitoring output XD. Since the phase of the stopping position could not be controlled, test programs checked whether the data read out could be rotated to match the data pattern initially loaded. Test measurements from runs that did not allow a match were treated as error cases.

10.4. Stopping Behavior

The one task that was not elegant was stopping the FIFO rings, which was done by deasserting the PUT signal in the asP* design, and by setting the P switch to opaque in the Micropipeline design. This could cause misoperation if the external stop signal arrived at the interface just as the FIFO control was enabling a data transfer operation. We decided to use this very simple method to stop the ring for our first run of 25 asP* chips. This method can be likened to that of inserting a crowbar into the spokes of a bicycle to stop the wheel from turning, so we called this mechanism the *crowbar stopper.* We calculated that the chances of corruption when the external stop signal was applied were likely to be less than 10%. Performance measurement is still possible, since it is done before the chip is stopped. We

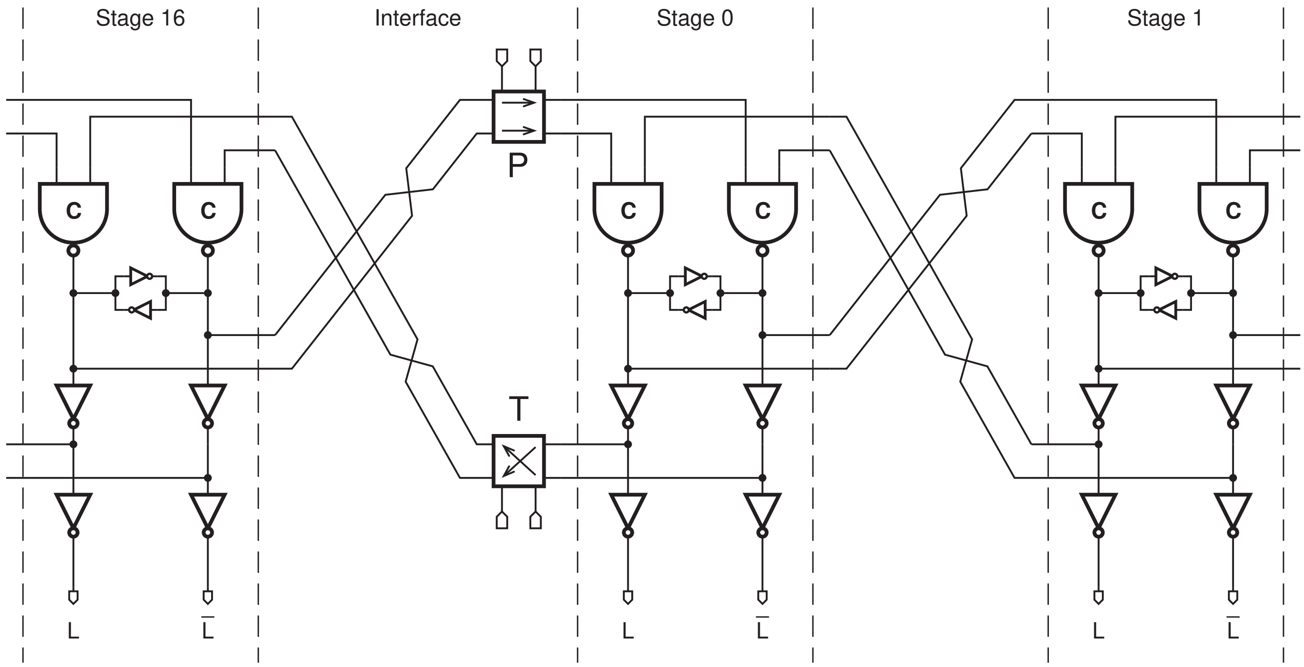

Figure 9. Two-rail implementation of Micropipeline interface logic.

recorded speed and other measurements only if the number of filled cells and the data pattern after stopping matched those loaded. For the second run of 25 chips, we used a synchronizer circuit, not described in this paper, as part of the stopping mechanism, but the FIFO ring circuit details remained the same as in the crowbar stopper design. We refer to the chips with this synchronizer circuit as the *proper stopper* chips, and as expected, we have yet to see any stopping errors with these chips.

Our third run of 25 chips used the Micropipeline design. We decided not to include any special stop synchronization circuitry in this design, and we expected stopping errors to occur in about 10% of the tests.

In the measured results for the crowbar stopper circuit version of the asP* chips, about 3% of stops resulted in mismatch, typically due to the loss of a cell. The proper stopper circuit version of the asP* chips, as expected, never showed any corruption of data or number of filled cells on stopping under normal operating conditions. Four sample proper stopper chips were tested for extended periods at nominal Vdd of 3.3V and stopped and re-examined over 307,000 times without error. In the Micropipeline chips, about 7.5% of stops resulted in mismatch. Three particular modes of failure were observed, where 6% were due to data value errors, 1.3% were control failures in which two full cells were lost entirely, and 0.2% were control failures where four full cells were lost entirely.

10.5. Chip Measurements

We extensively studied the 25 crowbar stopper asP* chips, 25 proper stopper asP* chips, and 25 Micropipeline ring chips over a wide range of operating conditions, measuring throughput and checking for stopping errors. We operated the chips at nominal Vdd of 3.3V, and at Vdd of 2.4V and 4.2V, which is about 27% above and below nominal—a much wider operating range than typically expected for conventionally-clocked parts.

All 75 tested chips operated correctly, except for the occasional stopping errors for the chips using the crowbar stopper asP* and Micropipeline designs. We used data *throughput* as the performance measure of the rings, where throughput is defined as the number of items passing through a given stage per second. We were able to measure the throughput by observing the data output XD when the rings were running. Figures 11 and 12 plot average throughput and power versus the number of filled cells in the ring for the asP* chips. As has been previously reported by Williams [10], for small numbers of filled cells, the throughput rises approximately linearly with the number of filled cells. For large numbers of filled cells, the throughput varies approximately linearly with the number of cells that remain empty. The throughput is zero, of course, if the ring is completely filled or completely empty. We interpret this to mean that throughput is limited for a near-empty FIFO by the time it takes a filled cell to make a complete cycle around the ring, multiplied by the number of filled cells that are present. Similarly, for a near-full FIFO, the

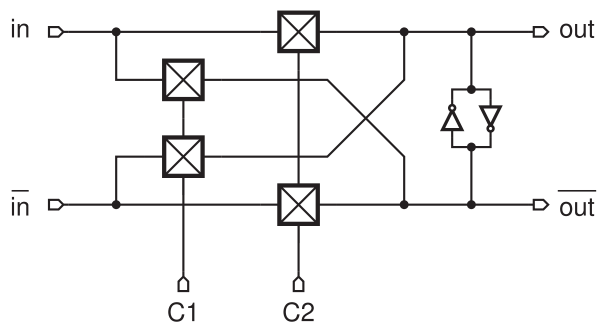

Figure 10. Switch implementation.

throughput is limited by the time it takes an empty cell to make a complete backwards cycle, multiplied by the number of empty cells in the ring.

When the ring is approximately half-filled, the throughput is less than the extrapolation of these two linear segments would indicate. The lower throughput indicates the presence of a stage or stages that cannot propagate filled or empty cells as fast as most of the stages in the ring. In particular, the interface stage is more complex and slower than the other stages of the FIFO. For cases in which the FIFO is nearly half-filled, congestion in the slow portion limits the overall throughput; the slow stage meters out full cells at its own pace thus limiting throughput. The extrapolation of the linear parts of the plot gives an estimate of the peak performance this FIFO could achieve, absent the interface stage. We have left gaps in the curves at the start and end of the flat top section because we do not have intermediate data points to indicate the slopes in these regions.

The close tracking of throughput and power consumption is not surprising, as the amount of “work” done is proportional to throughput. The power also depends somewhat on the values in the data path. While we might wish to compare the relative power consumption of the data path and the control circuits, these particular chip designs fail to offer meaningful comparisons.

Plots of measured throughput and power versus the number of full cells for the Micropipeline ring chip are shown in Figures 14 and 15. The flatter top of these results indicates that the interface circuitry creates a “narrower” bottle neck in the Micropipeline ring than the corresponding bottleneck in the interface stage of the asP* design. Without including the interface, the peak throughput of the Micropipeline stages, extrapolated from the measurements, is 1569 to 1731 million data items per second, which is roughly 50% faster than the asP* design.

The decaying hyperbolic slope of the “flat top” section of the power measurements shown in Figure 15 is due to the change in the power consumption contributed by the data path. We carried out our measurements with a data bit pattern of half 1’s followed by half 0’s. As the number of items in the ring increased, the total number of 1’s and 0’s in the ring also increased, while the throughput remained constant, and thus the frequency of data changes decreased, reducing the power consumption. This effect was not observed for the asP* measurements shown in Figure 12 because there were too few data points available in this flat top region of the plot.

10.6. Robustness Tests

Varying the Vdd supply voltage from its nominal value gives some indication of circuit robustness. The control circuitry was somewhat more robust at low voltages than was the data circuitry, but all 75 chips operated properly at voltages down to 1.67V, which is about half of the nominal supply voltage. Control circuitry generally appeared to operate properly at voltages as low as 1V, although when operated at a Vdd of 1.6V, one of the proper stopper chips gave stopping failures. The Micropipeline chips operated correctly over a voltage range from 0.9V to 8.4V, although at the high voltages we damaged some of the chips.

We have carried out extensive measurements over a large range of temperatures for several of the asP* and Micropipeline chips. Figure 13 shows the throughput variation against supply voltage for temperatures from –25° C to 100° C for one of the asP* chips. The curves end a little below 2V because of data path errors, but the control signals of this particular chip continue to operate correctly down to about 1.2V. Figure 16 shows the throughput variation against supply voltage for temperatures from –75° C to 150° C for one of the Micropipeline chips. Note the interesting crossing at 1.6V, below which some of the transistors are operating sub-threshold.

11. Conclusions

These results encourage us to believe that performance-competitive asynchronous circuits can be achieved by relying on control of delays in circuits and their interconnections as is done in clocked designs. If this can be accomplished at the detailed circuit level, one can hope to reap further performance benefits at the system level, as well as improvements in the ease of interfacing large circuit blocks and automating some parts of the system design process. As was done in the Macromodule project [3], we can hope to design highly efficient components that rely on close control of timing in their interior, while presenting an external interface to other components that can be well-modeled using the convenient concepts and formalisms of Delay-Insensitive design. One might regard

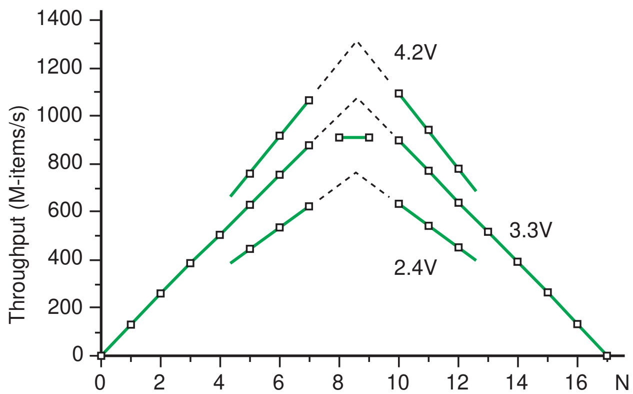

Figure 11. Throughput vs. number of full cells for the asP* chips.

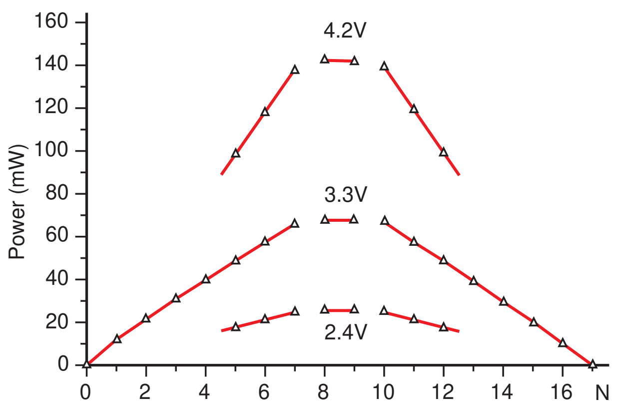

Figure 12. Power vs. number of full cells for the asP* chips.

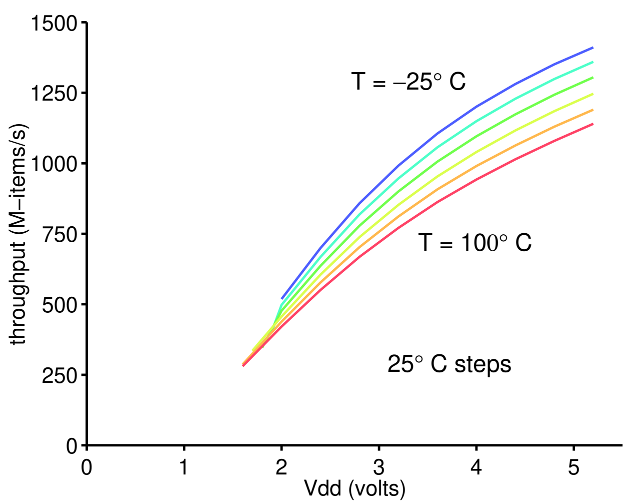

Figure 13. Throughput vs. temperature for one asP* chip.

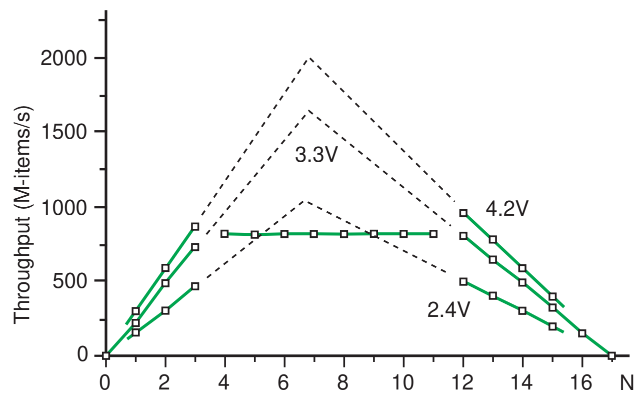

Figure 14. Throughput vs. number of full cells for the Micropipeline chips.

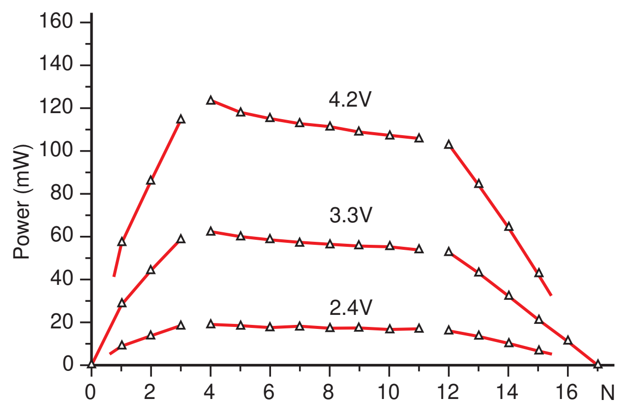

Figure 15. Power vs. number of full cells for the Micropipeline chips.

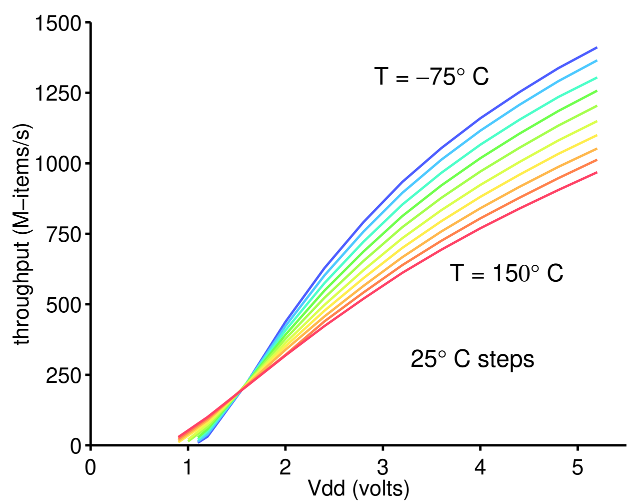

Figure 16. Throughput vs. temperature for one Micropipeline chip.

the experimental FIFO designs described here as a partial example; we have not completed the task because we did not provide a Delay-Insensitive interface to our circuits and have not explained how to find necessary and sufficient timing conditions for reliable operation. We have carried out further studies on more complex FIFO structures that perform data-dependent data flow [4], and have shown that very high performance can still be attained.

High performance is achieved by setting timing margins that are conservative enough to provide reliable operation, but tight enough to avoid wasteful sacrifice of speed. Determining the timing conditions required for robust operation is non-trivial. Process differences can be measured by comparing results from different fabrication runs. Process variations can also affect the relative delays in circuits on the same chip. Success in designing high performance circuits using the approach we have outlined depends on addressing the problem introduced by such delay variations. We require further experience in chip fabrication to determine an appropriate design policy.

12. References

[1] S. S. Appleton, S. V. Morton, and M. J. Liebelt, Two-Phase Asynchronous Pipeline Control, *Proceedings of the Third International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 12-21, April 1997.

[2] R. W. Brockett, Smooth Dynamical Systems Which Realize Arithmetical and Logical Operations, H. Nijmeijer and J. M. Schumacher, editors, Three Decades of Mathematical Systems Theory: A Collection of Surveys at the Occasion of the 50th Birthday of J. C. Willems, Vol. 135 of Lecture Notes in Control and Information Sciences, pp 19-30, Springer-Verlag, 1989.

[3] Wes A. Clark and Charles E. Molnar, Macromodular Computer System, *Computers in Biomedical Research*, Vol. IV, R. Stacy and B. Waxman editors, Academic Press, New York, 1974.

[4] W. S. Coates, J. K. Lexau, I. W. Jones, S. F. Fairbanks, and I. E. Sutherland, A FIFO Data Switch Design Experiment, *Proceedings of the Fourth International Symposium on Advanced Research in Asynchronous Circuits and Systems*, pp. 4-17, March 1998.

[5] P. Day and J. V. Woods, Investigations into Micropipeline Latch Design Styles, *IEEE Transactions on VLSI Systems*, Vol. 3, No. 2, pp. 264-272, June 1995.

[6] S. B. Furber and P. Day, Four-Phase Micropipeline Latch Control Circuits, *IEEE Transactions on VLSI Systems*, Vol. 4, No. 2, pp. 247-253, June 1996.

[7] Mark R. Greenstreet and Peter Cahoon, How Fast Will the Flip Flop? *Proceedings of the International Symposium on Advanced Research in Asynchronous Circuits and Systems*, Salt Lake City, Utah, November 3-5, 1994.

[8] C. J. Myers and T. H.-Y. Meng, Synthesis of Timed Asynchronous Circuits, *IEEE Transactions on VLSI Systems*, pp. 106-119, June 1993.

[9] Ivan E. Sutherland, “Micropipelines,” Comm ACM 32, 6(1989): 720-738.

[10] Ted E. Williams, Performance of Iterative Computation in Self-Timed Rings, *Journal of VLSI Signal Processing*, February 1994.