Implementing a STARI Chip*

Mark R. Greenstreet
Department of Computer Science
University of British Columbia
Vancouver, B.C., V6T 1Z4 Canada

Abstract

STARI is a high-speed signaling technique that uses both synchronous and self-timed circuits. To demonstrate STARI, a chip has been fabricated using the MOSIS 2μ CMOS process. In a simple test fixture, it operates at data rates of 120 Mbits/sec over a pair of wires. Because STARI uses both synchronous and self-timed circuits, it provides an opportunity to compare these two design methods. The synchronous circuits of the STARI chip achieve rates of operation two to three times those of the self-timed circuits. However, the self-timed FIFO in the receiver provides robust compensation for clock skew that could not be achieved with synchronous circuitry alone. Thus, the STARI chip demonstrates advantages of combining these two design techniques.

1 Introduction

STARI (Self-Timed at Receiver's Input) is a signaling technique that uses both synchronous and self-timed circuitry to achieve high-bandwidth, chip-to-chip communication. STARI uses a global clock to establish the rate of data-transmission and does not require low-level flow control, nor does it incur the overhead of handshake circuitry. The receiver in a STARI interface uses a self-timed FIFO to compensate for clock skew and variations in the delay along the data path.

This paper describes a chip that demonstrates the STARI technique. The chip has been fabricated using the MOSIS 2μ CMOS process. It operates at data rates of 120 Mbits/sec over a pair of wires in a simple test fixture. As expected, no transmission errors are observed even in the presence of large, varying skews.

By combining self-timed and synchronous design methods, STARI provides an opportunity to compare these two approaches. For the circuits used in the chip, the synchronous designs operate at rates two to three times those that could be achieved by a self-timed FIFO. To avoid a FIFO bottleneck, the basic STARI technique was modified to accommodate the self-timed FIFO without compromising the bandwidth of the interface.

*This research was supported by operating grant OGPO 138501 from the Natural Sciences and Engineering Research Council of Canada, a fellowship from the B.C. Advanced Systems Institute, and an equipment grant from Sun Microsystems Inc., Canada.

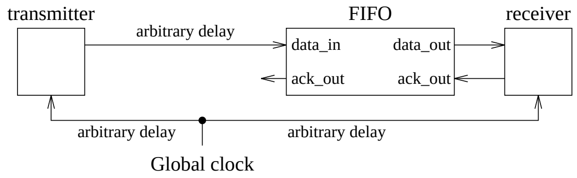

Figure 1: STARI communication

2 STARI

STARI is motivated by the observation that it is a simple matter to distribute a high frequency reference over relatively large distances. On the other hand, it is difficult to control the exact phase of high frequency signals. This uncertainty, often called *skew*, limits the performance of purely synchronous systems. Self-timed designs avoid clock skew by using handshake protocols. If no assumptions are made about the delays of components and wires, then each transmitted bit must be acknowledged before the next one is sent. This round-trip delay can severely limit the bandwidth of the channel.

Figure 1 shows a STARI interface. A global clock sets the rate of operation for the transmitter and receiver, and a self-timed FIFO compensates for skew between them. Typically, the FIFO is initialized to be roughly half-full, and during each period of the global clock, one value is inserted into the FIFO from the transmitter, and one value is removed by the receiver. This ensures that the FIFO never overflows or underflows, and STARI needs no flow control signals. The lack of flow control signals distinguishes STARI from other FIFO based designs. This eliminates the round-trip delay of a purely self-timed interface and avoids the latency required for reliable synchronization of flow control signals in a synchronous design.

For correct operation, the FIFO must complete each insert and remove operation within one cycle. When this requirement is met, the FIFO appears as a synchronous component to both the transmitter and the receiver, and both the transmitter and the receiver appear to the FIFO as well-behaved self-timed systems. These informal arguments have been formalized to prove that STARI is free of timing hazards [5, 6]. Once the interface is initialized, it operates without

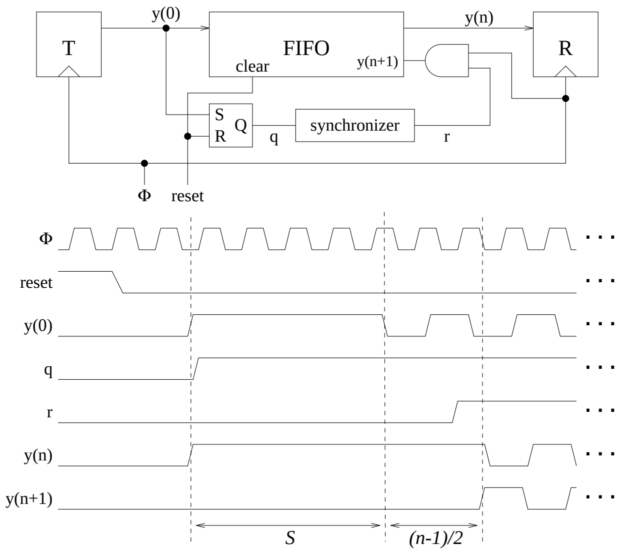

Figure 2: STARI with initialization

timing failures or metastability, even in the presence of clock skews that vary over several clock periods.

STARI interfaces compensate for varying clock skews by matching the skew with the latency of the FIFO. This paper describes an implementation of STARI that uses a ripple FIFO. Let n denote the number of stages of the FIFO, and δ be an upper bound for the time it takes a stage to perform an operation. Let π denote the clock period and λ denote the FIFO latency. We have shown that the STARI interface functions correctly if:

$$(n+1)\delta \quad < \quad \lambda \quad < \quad (n+1)(\pi - \delta) \qquad (1)$$

where λ denotes the latency of the FIFO. The lower bound corresponds to the time for a value to propagate from the first stage of the FIFO to the last stage, with δ time units of set-up time for the receiver. The upper bound corresponds to the time for an acknowledgement to propagate from the last stage of the FIFO to the first stage, with δ time units to spare (for symmetry that simplifies the analysis). The window of correct operation has a width of $(n+1)(\pi - 2\delta)$. For suitable values of n, π, and δ, this window can be much wider than the clock period.

Figure 2 shows the initialization circuit used in the STARI chip. From equation 1, it can be seen that the half-full operating point maximizes throughput and robustness to variations of skew. To reach such a state, a global reset is asserted, and the FIFO is cleared. After the reset signal is de-asserted, the transmitter starts sending. When the transmitter has sent $(n+1)/2$ values, the receiver starts to remove data from the FIFO.

Due to skew, values can arrive from the transmitter at an arbitrary point in the receiver's clock cycle, and the possibility of metastability must be considered [3]. Therefore, a chain of synchronizers is employed in the receiver to detect the *first* value sent by the transmitter. By making the chain sufficiently long, the probability of a synchronization failure can be made arbitrarily small. Note that synchronization is only performed during initialization.

Figure 2 illustrates the sequence of operations for initialization. First, the reset signal is applied, and the FIFO is cleared. After the reset is removed, the transmitter sends the first value. To accommodate the synchronization chain, the transmitter waits S cycles between sending the first and second values. The receiver detects the arrival of the first value, and after $S + (n-1)/2$ cycles, it starts removing data from the FIFO. This brings the FIFO to a half-full state and gives the receiver $S + (n-1)/2$ cycles for synchronization. Once initialized, the STARI interface provides a throughput of one data value per clock cycle, and the latency is roughly $(n+1)/2$ cycles which can be much less than S.

FIFOs are often used to interface circuits operating with independent clocks. Typically, the FIFO generates flow control signals that must be synchronized to the clocks of the transmitter and receiver. This introduces the possibility of synchronization failure [3]. Instead of using flow control signals, static timing analysis guarantees that the FIFO in STARI can neither overflow nor underflow. This simplifies the implementation of STARI, and allows it to operate at higher rates and with lower latencies than traditional FIFO based designs.

Many communication schemes that depend on close or exact frequency matching at the transmitter and receiver have been implemented [13, 7, 1]. These schemes use clock forwarding or employ phase-locked loops to regenerate the transmitter's clock at the receiver. Clock forwarding requires an acyclic network topology and phase-locked loops require mixed analog and digital circuitry on the same chip. STARI is simpler to implement than a phase-locked loop design, and it allows general network topologies. Many phase-locked designs use double buffering and synchronization is done on a per-block basis. STARI achieves lower latencies by avoiding both synchronization and double buffering.

3 Implementing a STARI chip

To demonstrate STARI, a chip has been designed, fabricated, and tested. The transmitter portion of the chip inputs parallel data at clock rates that are suitable for microprocessor based board level design and outputs high-speed serial data for transmission to a STARI receiver. Conversely, the receiver inputs high bit-rate, serialized data, and outputs parallel data. In the following, the clock that controls serial data transfer is called the **fast clock**, and the clock that controls transfers between the STARI chip and its host is called the **slow clock**.

In the implementation, the fast clock operates at five times the frequency of the slow clock. The transmitter performs a ten-to-one multiplexing operation: separate data values are output during the high and

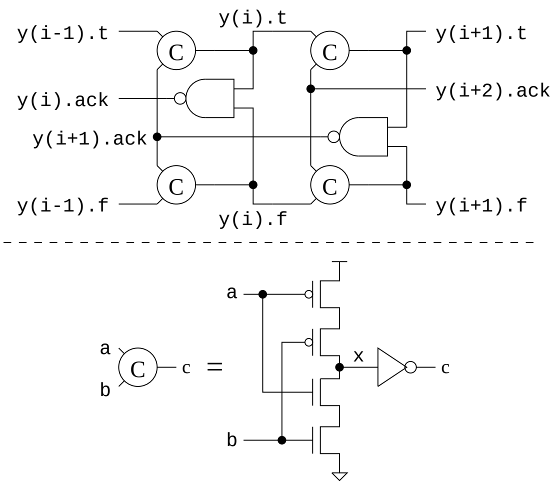

Figure 3: The self-timed FIFO

low portions of the fast clock. In the implementation, the slow clock is derived from the fast clock using a divider that is part of the STARI chip. It also would have been possible to distribute a global slow clock and derive local fast clocks using phase-locked-loop frequency multipliers.

The chip was fabricated using the MOSIS 2μ n-well CMOS process. Although more advanced processes are available, the 2μ process had several advantages: fabrication was relatively inexpensive, it is adequate for a proof-of-concept chip, and testing was simplified in that exotic test equipment would have been required for a faster chip. Likewise, a standard pin-grid-array (PGA) package was used rather than attempting to use more exotic packaging technology in the first chip. Electrical characteristics (i.e. parasitic inductances and capacitances) of this package were unavailable; so, "typical" values from [2] were used during the design process. The high-speed synchronous circuits were designed using the single-phase clocking technique of Yuan and Svensson [20], and the input and output pads for the high bandwidth signals were based on the series terminated design of Knight and Krymm [8].

3.1 The receiver

The design of the receiver was driven by the performance of the FIFO and the input amplifier. Figure 3 shows the FIFO design. The design uses an inverted dual-rail data encoding with four-phase handshake [15] and is intended to minimize the FIFO cycle time. By inverting the code, the design exploits the fact that NAND gates are generally faster than NOR gates in a CMOS implementation. Unlike many self-timed designs, the C-element in this FIFO is a purely dynamic design; this optimization is possible because every C-element drives its internal node `x` low during the "empty" phase of each handshake. The 2μ implementation of this circuit can perform a data transfer (i.e. a complete four-phase handshake) every 5.5ns (simulation using "typical" SPICE parameters).

In an earlier design, faster operation was attempted by inverting the signals of the dual-rail code for the outputs of odd-numbered stages. In this version, inverting C-elements were used at each stage, and alternating NAND and NOR gates were used to generate the acknowledge signals. By eliminating an inverter from the forward path, the time per four-phase handshake was reduced to 4.8ns. Although this optimization worked in five-corner SPICE simulation, it led to a malfunction in the actual chip. When the first datum arrived from the transmitter, the output of an inverting C-element could make a low-to-high transition and receive an acknowledgement from its successor before its output voltage exceeded the threshold voltage for a P-channel device. This causes a charge leak in the successor stage, which, during the synchronization interval, brought the FIFO into an ill-defined state. Buffering the output of each C-element with an inverter solved this problem by increasing both the cycle time and the noise margins of the FIFO stages.

Using either FIFO design, the cycle time of the FIFO is large enough to limit performance. Therefore, a design was chosen where the transmitter sent data and a forwarded clock[1] A demultiplexer driven by the forwarded clock provides data to ten, parallel FIFOs, each of which operates at the rate of the slow clock. With this approach, the self-timed FIFO provides compensation for clock skew without limiting the performance of the chip.

The synchronous demultiplexer presents a large load at its clock and data inputs. On the other hand, the biased-inverter amplifiers used in the input pads have a relatively high output impedance, even with the buffering stage proposed in [8]. A solution to this problem is to buffer the input data and clock signals using a chain of inverters where each inverter is only slightly larger than its predecessor. It is well known [11] that enlarging successive stages by a factor of e produces the minimum latency buffer. However, latency is not critical to the STARI design as long as the delays of the clock and data paths are well matched. Higher bandwidths can be achieved by using a smaller factor between stages. The STARI chip uses a four-inverter chain following the biased-inverter input amplifier, with an average growth factor of 1.25 between stages.

A challenge to this approach is that the delays of the clock and data paths must be reasonably well matched. Using the same design for the clock and data buffers helps, however mismatched rising and falling delays can still introduce skews. The design of the STARI chip addresses this by latching the incoming data to the forwarded clock after four stages of amplification for both the clock and data channels. Further amplification is provided by a shift register, in which inverters of increasing size buffer the clock from one stage to the next, and latches of increasing size buffer the data. The design requires the delay for the clock

[1]This is similar to the "bundled completion" used in many self-timed designs such as the Caltech mesh router chip [16].

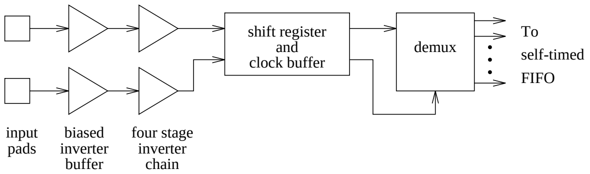

Figure 4: Input amplifier and shift-register

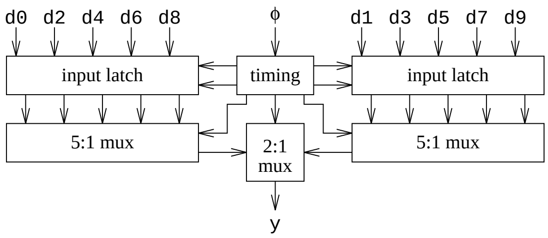

Figure 5: Transmitter channel

inverter to be less than four gate delays in the data channel, a condition that is easily satisfied. Figure 4 shows the input circuitry of the STARI chip.

3.2 The transmitter

The key components of the transmitter design are the multiplexers generating the data and clock outputs. The chip uses identical channels for the clock and data outputs; this approach allows flexible experimentation with the clock and data timing during the initialization. Figure 5 shows the block diagram for a channel. The parallel data inputs are `d0...d9`, and the serial output is labeled `y`. The channel is a synchronous design, clocked by the fast clock, ϕ. The block labeled timing produces the two-phase slow-clock with a rate one-fifth that of ϕ. The two-phase slow clock controls the input latches and is used off chip. The timing block also generates one pulse during every fifth cycle of the fast clock. This pulse controls the operation of the five-to-one multiplexers which are a simple systolic design. The two-to-one multiplexer consists of two tri-state drivers that are enabled on opposite phases of the fast clock to send different data values during the high and low portions of the fast clock. This design is simple, has a regular layout, and achieves high bandwidths.

3.3 Test results

The STARI chip was fabricated through MOSIS using the 2μ n-well process (VTI). In the current test fixture, the chip is held in a 2-level wire-wrap socket and it operates without errors at data rates up to 120 Mbits/sec. We can vary the relative skew between the transmitter and the receiver by altering the parameters of the programmable pulse generator that is used to generate the high speed clock and by modifying the initialization protocol. These tests demonstrate that the chip operates correctly over the range of skews predicted by equation 1. The transmitter operated at 350 Mbits/sec, which is the upper limit of our pulse generator. The test results strongly suggest that the top rate for the transmitter-receiver pair of 120 Mbits/sec is due to parasitic loading by the test fixture. Accordingly, we are implementing a test fixture that will allow direct, controlled impedance connections to these pins.

4 Comparing self-timed and synchronous design

By combining self-timed and synchronous design methods, STARI provides an opportunity to compare these two approaches to design. The entire chip was designed by the author; accordingly, the level of design skill is comparable for both the synchronous and asynchronous portions. Furthermore, both parts were designed using the same tools (magic [12], CAzM [4], irsim [14], and Synchronized Transitions [17]) and with the same criteria: to be fast and simple. On the other hand, the STARI chip performs little actual computation. This means that pipelining and handshaking overhead figure prominently in the comparison.

In the context of the STARI chip, synchronous circuits had smaller cycle times than their self-timed counterparts. This led to the introduction of the synchronous demultiplexer *before* the self-timed FIFO, to prevent the cycle time of the FIFO from becoming a bottleneck limiting STARI performance. In simulation, the self-timed FIFO can complete a four-phase handshake every 5.5 nanoseconds. The FIFO design was manually optimized for speed; comparable performance was obtained from a version optimized using the Caltech Asynchronous Synthesis Tools [9]. In contrast, the synchronous circuits can operate with a three nanosecond clock period. The difference in performance can be explained by examining the structures of the synchronous and self-timed designs. For the synchronous circuits used in the STARI multiplexer, the longest path for each phase (high or low) of the clock has three gate delays. Simplistically, this gives a clock period of six gate delays. In contrast, a four-phase handshake in the FIFO takes ten gate delays, (two times around a loop consisting of two C-elements and a NAND gate, where each C-element is an inverting C-element plus an inverter).

In some designs, a two-phase handshake might have less overhead. For the FIFO, two-phase handshaking did not offer an advantage. The exclusive-OR gates required for a two-phase handshake were slower than the NAND-gates used in the four-phase handshake, and the four-phase design had higher throughput.

For the STARI chip, clock distribution was not a problem. This is not surprising for such a small, simple design. We note that high frequency clocks have been used in much larger designs, such as the Alpha microprocessor [10], albeit with significant power consumption. On the other hand, broadcasting data signals can be more problematic. Here, STARI shows an advantage of self-timed design. The self-timed FIFO provides a simple way to compensate for delays in the clock and data paths, and, when controlled impedance interconnect is used, STARI allows multiple bits to be simultaneously in flight between the transmitter and receiver. Using a self-timed FIFO to compensate for clock skew is an example of a design niche where self-timed circuits excel. Other researchers have demonstrated other niches for self-timed design such as low power applications [18] or iterative computation where a suitable clock is not available [19].

The final design of the STARI chip used each design technique where it offered the greatest advantage. Synchronous circuits were used for their small cycle times in the multiplexer and demultiplexer, and a self-timed FIFO was used to compensate for skew between the transmitter and receiver. To compare optimized circuits for both approaches, we have compared a self-timed FIFO with a synchronous multiplexer and demultiplexer. The FIFO should provide a lower bound for the cycle time of a self-timed multiplexer or demultiplexer, and we conclude that a self-timed demultiplexer would be slower than our synchronous implementation. Likewise, we did not implement a synchronous FIFO to compare with our self-timed design. Although a STARI interface could use a pointer FIFO, such a FIFO requires dual-ported registers and broadcast of control signals. It seems unlikely that a pointer FIFO could be implemented with the performance and simplicity of the self-timed design.

5 Conclusions

This paper has described the STARI signaling technique and a chip that demonstrates it. STARI allows synchronous circuits to operate at high clock rates and communicate reliably, in spite of clock skews and uncertainties in data delays that may be greater than the clock period. Because, STARI does not require synchronization after initialization, it is free from timing hazards during operation. Furthermore, a STARI interface does not suffer from the latency associated with synchronization.

The STARI chip operates at 120 Mbits/sec in a simple test fixture. Test results suggest that the limiting factors are the electrical properties of the pin-grid array package for the STARI chip and the wire-wrap sockets used in the tester. Having established the correct operation of the STARI chip, a new tester is being implemented that will allow direct connections of 50Ω lines to the high frequency I/O pins of the chip.

Because STARI uses both synchronous and self-timed circuits, it provided an opportunity to compare these two design methods. In this design, the synchronous circuitry can operate at rates two to three times higher than the self-timed circuits. However, the self-timed FIFO in STARI provides compensation for skew that could not be achieved in a purely synchronous design. Using a self-timed FIFO to compensate for clock skew is an example of a design niche where self-timed circuits excel.

6 Acknowledgements

Steve Burns designed a FIFO using the Caltech tools for comparison with the manually optimized design of the STARI chip. Tom Knight provided helpful suggestions about pad design. Alex Dickinson and Bryan Gershan assisted with the testing of STARI chips.

References

[1] AMD Inc. *The SUPERNET Family for FDDI*, 1989.

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

[3] T.J. Chaney and C.E. Molnar. Anomalous behavior of synchronizer and arbiter circuits. *IEEE Transactions on Computers*, C-22(4):421–422, April 1973.

[4] D.J. Erdman, S.W. Kenkel, et al. CAzM: A numerically robust, table-based circuit simulator. Technical Report TR89-23, Microelectronics Center of North Carolina, Research Triangle Park, North Carolina, July 1989.

[5] Mark R. Greenstreet. *STARI: A Technique for High-Bandwidth Communication.* PhD thesis, Department of Computer Science, Princeton University, January 1993.

[6] Henrik Hulgaard, Steven M. Burns, et al. Practical applications of an efficient time separation of events algorithm. In *Proceedings of the 1993 International Conference on Computer Aided Design*, pages 146–151, November 1993.

[7] Mark G. Johnson and Edwin L. Hudson. A variable delay line PLL for CPU-coprocessor synchronization. *IEEE Journal of Solid-State Circuits*, 23(5):1218–1223, October 1988.

[8] Thomas F. Knight and Alexander Krymm. A self-terminating low-voltage swing CMOS output driver. *IEEE Journal of Solid-State Circuits*, 23(2):457–464, April 1988.

[9] Alain J. Martin, Drazon Borković, et al. CAST, Caltech Asynchronous Synthesis Tools: The first release. Technical Report Caltech-CS-TR-93-11, Computer Science Department, California Institute of Technology, 1993.

[10] Edward McLellan. Alpha AXP architecture and 21064 processor. *IEEE Micro*, 13(3):36–47, June 1993.

[11] Carver Mead and Lynn Conway. *Introduction to VLSI Systems.* Addison Wesley, 1979.

[12] John K. Ousterhout, Gordon T. Hamachi, et al. Magic: A VLSI layout system. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pages 152–159, Albequerque, NM, June 1984.

[13] Randall D. Rettberg and Lance A. Glasser. Digital phase adjustment. U.S. Patent no. 4,700,347, October 1985.

[14] Arturo Salz and Mark Horowitz. IRSIM: An incremental MOS switch-level simulator. In *Proceedings of the 27th ACM/IEEE Design Automation Conference*, pages 173–178, Los Vegas, NV, June 1989.

[15] Charles L. Seitz. System timing. In *Introduction to VLSI Systems* (Carver Mead and Lynn Conway), chapter 7, pages 218–262. Addison Wesley, 1979.

[16] Charles L. Seitz and Wen-King Su. A family of routing and communication chip based on the Mosaic. In Gaetano Borriello and Carl Ebeling, editors, *Research on Integrated Systems: Proceedings of the 1993 Symposium*. MIT Press, 1993.

[17] Jørgen Staunstrup. *A Formal Approach to Hardware Design*. Kluwer, 1993.

[18] C.H.˜(Kees) van Berkel, Ronan Burgess, et al. Asynchronous circuits for low power: A DCC error corrector. 11(2):22–32, 1994.

[19] Ted E. Williams. *Self-Timed Rings and their Application to Division*. PhD thesis, Stanford University, May 1991.

[20] Jiren Yuan and Christer Svensson. High-speed CMOS circuit technique. *IEEE Journal of Solid-State Circuits*, 24(1):62–70, February 1989.