Proceedings of the IEEE International Symposium on Circuits and Systems
Espoo, Finland, pp. 805-809, June 1988.

DESIGN AND IMPLEMENTATION OF HIGH-SPEED ASYNCHRONOUS COMMUNICATION PORTS FOR VLSI MULTICOMPUTER NODES†

Yuval Tamir and Jae C. Cho

Computer Science Department
University of California, Los Angeles, California 90024
U.S.A.

ABSTRACT

A communication coprocessor that provides high-bandwidth low-latency inter-node communication is a key component of *multicomputer* systems composed of hundreds of computing nodes interconnected by point-to-point links. For high reliability, interdependency between nodes is minimized by using a separate clock at each node. Thus, the coprocessor must handle asynchronous inputs with a very low probability of synchronization failures. In order to minimize system chip count, the entire coprocessor should be implemented on a single CMOS VLSI chip. Thus, limitations on the raw speed and pin drive capabilities of this technology must be considered. This paper discusses the design and implementation of communication ports for a communication coprocessor chip. We describe several schemes for handling high-speed asynchronous communication and present a new design of a communication port that includes a synchronizer and a high-performance FIFO buffer. This design is particularly well-suited for VLSI implementation and, for a given technology, can support higher communication speeds than previous synchronizer designs.

I. INTRODUCTION

Due to advances in VLSI technology it is now feasible to implement computer systems consisting of thousands of processors. A *multicomputer* system composed of computing nodes interconnected by point-to-point dedicated links [15] can achieve both high performance and high reliability. In order to achieve high performance the system must provide for high-bandwidth low-latency communication between nodes. In order to achieve high reliability, interdependency of nodes should be minimized and correct system operation must not depend on any single resource [16]. Hence, if the nodes are based on conventional synchronous designs, each node must have its own clock.

In the UCLA ComCoBB (**Com**munication **Co**processor **B**uilding-**B**lock) project we are investigating the design and implementation of a communication coprocessor chip that will provide high-speed, reliable, message-based, point-to-point, communication between nodes in a multicomputer. One of the design goals of the ComCoBB project is to be able to fit all the needed functionality on a single CMOS VLSI chip.

† This research is supported by Rockwell International and the State of California MICRO program.

In this paper we show how the ComCoBB chip will achieve, at the link level, high-speed, low-latency communication between adjacent nodes using standard CMOS technology (available to us through the MOSIS service). In adherence with our goal of a *single* chip implementation, we will not use special line drivers, such as the ECL channel interface used on the JPL hyperswitch [4]. We focus on the problem of *synchronization failures* and the design of a high performance synchronizer for implementation using relatively slow CMOS technology. We discuss the design of a FIFO queue which can keep up with the high data rate. Our discussion also includes the problem of driving the internode links (wires) at high rates despite power dissipation limitations of the packaging of our chips.

II. DESIGN ISSUES

Since each node in the multicomputer operates with its own clock, data sent from one node to its neighbor is not synchronized with the receiver's clock. In order to process asynchronous inputs, a synchronous system must first synchronize the inputs to the local system clock by sampling the input data using a flip-flop or latch clocked with the local clock [11]. This process can lead to *synchronization failure* where the flip-flop or latch used for synchronization reaches a *metastable* state in which its output is a value between logic 0 and logic 1. The *failure* occurs when this output is interpreted differently by different parts of the system [3, 11, 14]. The probability of synchronization failure can be reduced by allowing the sampling flip-flop some ''settling'' time between the sampling of the input and the use of the flip-flop output by the rest of the system. If the sampling time is sufficiently long, then even if the flip-flop enters metastable state when the input is sampled it is likely to ''decay'' to one of two stable logic states before its output is used [11, 14]. The communication port must synchronize the input signal and achieve a low probability of synchronization failure so that this is not a major source of unreliability in the system.

In order to achieve the required bandwidth, the internode links are eight-bits wide. Thus, the port will have to sample all eight lines during every cycle. If the different bits arriving through the link are independently synchronized, small differences in delays along wires can result in ''sampling'' a byte of data that includes a combination of bits from two different bytes sent from the neighbor. Our design of the input port handles the synchronization failure problem as well as the skew between the different bits of the same byte.

The input port accepts data from the link and places it in a buffer. An output port takes data out of the buffer and sends it through the link. In both cases data is being transferred at a high rate so that the complexity of the operations on each byte must be quite limited. For a large buffer size (sixty four bytes in the present implementation), the decoding and the buffer access times may determine the system cycle time. Thus the buffer organization and access method should be carefully chosen to minimize decoding and access delays. Furthermore, the buffer must be dual-ported so that it will be possible to begin forwarding a packet to an output port without having to wait for the entire packet to arrive [10].

The chip's output port must be capable of driving the link at high speed (on the order of 20 Mbits/sec). We chose to support links up to one meter long (sufficiently long so that large multicomputers can be constructed). At these frequencies, the ''transmission line effects'' of the links cannot be ignored. Specifically, if the lines are not properly terminated reflections from the destination can cause ''ringing'' that will corrupt the signal. An additional issue related to the output port is the power requirement. For each link, eight lines have to be driven at a rate of 20 Mbits/sec. Depending on the way the links are terminated and the number of links, the total power dissipation on the chip can become a problem.

III. SYNCHRONIZER DESIGN

One technique for reducing the probability of synchronization failures for bit-serial links is based on the use of limited size packets and reliance on an upper bound on the rate of ''drift'' between the clocks of neighboring nodes. At the beginning of each packet there is a *start bit* which is used by the receiver to adjust its ''sampling clock'' so that sampling will occur in the middle of each data bit for the rest of the packet. Thus, while there is a possibility of synchronization failure for the start bit, a synchronization failure cannot occur for the rest of the packet [8, 12].

A design of a circuit for performing the sampling clock adjustment is described in [12]. It requires an internal finite state machine (FSM) which uses two non-overlapping phases of a clock operating at three times the clock rate of the communication link. At any point in time the specific VLSI fabrication technology imposes an upper bound on the clock speed at which circuits will operate reliably. The design proposed in [12] requires that the maximum clock rate of the link be one third of the maximum clock rate at which an FSM can operate reliably on chip.

Our design uses the idea of synchronizing on a start bit of a packet and adjusting the sampling clock. However, one of our major goals was to support the highest communication bandwidth possible with a given (standard) CMOS VLSI technology. Thus, we allow a link data transmission rate that is more than half the maximum internal clock rate for reliable operation. This is accomplished as follows: (1) We use a clock that is only twice the link data transmission rate for synchronizing the start bit. (2) We do not need non-overlapping phases of this fast internal clock. (3) The fast clock is used only to ''gate'' one flip-flop and one latch and there is no signal that has to pass through a PLA within the cycle time of the fast clock. (4) The synchronized start bit is used to select a clock phase of the same frequency as the link data transmission rate for ''latching in'' the rest of the bytes in the packet. Once the selection of a clock phase is made, no further operations occur using the fast clock.
Points (2) through (4) imply that the internal circuitry that must operate at the highest rate is very simple and small. Thus the frequency of our ''fast clock'' can be higher than the highest clock frequency generally usable for the given technology.

Our simulations have shown that with the current three micron CMOS technology available from the MOSIS service, we can easily achieve a link transmission rate of 20 Mbits/sec and use a fast clock of 40 MHz. Thus, we can achieve a raw communication bandwidth of 20 Mbytes per second on each link. Our synchronizer design is shown in *Fig. 1* and will be described in the rest of this section.

Since the possibility of synchronization failure cannot be entirely eliminated [11, 14], the design goal for the synchronizer is to achieve a synchronization failures rate which is negligible compared with the failure rate of the VLSI chip itself. Taking into account both permanent and transient faults, a rate of 500 per billion hours is an optimistic estimate of the failure rate of a ''typical'' VLSI chip [1, 2, 13]. To achieve a rate of synchronization failures that is an order of magnitude smaller, the mean time to synchronization failure (MTF) should be at least 7.2×10^{10} seconds.

When a flip-flop is used for synchronization, as described in the previous section, the mean time to synchronization failure (MTF) is related to system and circuit characteristics by the following formula [7]:

$$MTF = \frac{e^{T/\tau}}{f_{\phi} f_{in} \Delta}$$

In this formula, τ and Δ are parameters of the circuit and the implementation technology. The parameters f_{ϕ} and f_{in} are the frequencies of the sampling clock and of synchronization events. The parameter T is the time during which the flip-flop is allowed to settle before its output is considered valid.

If the synchronizing flip-flop enters metastable state, its output will ''decay'' exponentially to one of the two stable logic states with the time constant τ. This time constant depends on layout and fabrication parameters. The value of τ can be derived analytically or using simulation [7, 9]. Using simulation, the synchronizing flip-flop is set to an initial condition where it is close to the metastable point and allowed to decay. The value of τ is found by reading the slope of the latch's differential output voltage plotted in semi-log scale [9].
We performed this simulation using SPICE level 2 models with typical values for the three micron CMOS MOSIS fabrication process and minimum size transistors in the output latch of the synchronizing flip-flop. The value of τ was found to be approximately 0.91 nsec. The parameter Δ is the time it takes the input voltage to cross the voltage range that may cause indeterminate state at the output of the flip-flop. As a conservative estimate, we will use the input signal rise time of 3 nsec for the value of Δ.

As mentioned earlier, synchronization failure can only occur for the start bit of each packet. The frequency of the sampling clock of the synchronizer, f_ϕ, is 40 MHz, which is twice the maximum transmission rate of the link. The parameter f_{in} is the rate of transitions that must be synchronized and is therefore the rate of new packet arrivals. Since each packet will require a start bit and addressing information in addition to the ''useful'' date being transmitted, we estimate the average packet length to be at least sixteen bytes. Given a link bandwidth of 20 Mbytes/sec, the packet arrival rate will be at most 1.25×10^6 per second.

Given the values of the parameters described above, and the required minimum value for MTF, we can calculate the minimum settling time that must be allowed for the flip-flop to be 33.6 nsec. In order to be able to use a sampling clock with a cycle time of 25 nsec, this settling time must be obtained by cascading synchronizers [14].

The synchronizing start-bit detector with the mechanism for latching in the data bytes of a packet are shown in *Fig. 1*. The sampling clock and the two non-overlapping phases of the system clock are denoted $CK40$, $CK20$, and $CK20^*$, respectively. $CK20$ is derived from $CK40$ so that both clocks are in phase and there is little skew between them (see *Fig. 2*). One of the eight lines of the link is used to transmit the start bit of each packet. This line it normally held high and is driven low for 50 nsec to indicate the beginning of the packet. Each data byte of the packet is then ''transmitted'' on the eight lines of the link for 50 nsec. We denote the time between sampling of the start-bit as T_ϕ. For our design $T_\phi = 25\ nsec$.

The total required settling time in the start-bit detector is provided by two stages of synchronizers. The first stage is a negative edge-triggered flip-flop and the second stage is an S-R latch. When a packet is not being transmitted and the line that will carry the start bit is high, the $\overline{Q}$ output of FF1 (node $N1$) is low, and S-R latch $L2$ is reset so that $N2$ and $N3$ are low. At the beginning of the packet, the input to $FF1$ is driven low, causing $N1$ to be driven high with the next falling edge of $CK40$. Latch $L2$ is clocked by $DCK40$, which is $CK40$ with the rising edge delayed by a few nanoseconds (to prevent a hazard in the operation of latches $L3$ and $L4$). After node $N1$ is driven high, latch $L2$ is set with the following rising edge of $DCK40$ and is reset back to zero by the FSM controlling the input port only after the entire packet is received. Latch $L3$ requires that node $N2$ be settled before the next falling edge of $CK20$ and latch $L4$ requires that node $N2$ settle to a logic 1 before the next transition of $CK20$. Thus, the cascaded synchronizers must settle within $1.5 \times T_\phi$ time units (37.5 nsec) from the falling edge of $CK40$ which samples the start-bit line after its first transition.

The estimate of the available settling time of $1.5 \times T_\phi$ ignores clock skews, propagation delays, and latch setup time in the circuit. Taking these parameters into account, the available settling time is $(3T_\phi/2 - T_{pd} - T_{su} + T_{skew})$, where T_{pd} is the propagation delay of $FF1$, T_{su} is the setup time of $L3$ and $L4$, and T_{skew} is the delay (skew) of the edges of $CK20$ with respect to the edges of $CK40$. Using SPICE simulation of our circuit the values of these parameters (T_{pd}, T_{su}, and T_{skew}) could be determined and the result is that the total available settling time is approximately 35 nsec. Since this value is greater than the required minimum settling time calculated above, the *MTF* goal (minimum time between synchronization failures) for the synchronizer is met.

So far, we have only shown that the start-bit is properly synchronized. It remains to be shown that the appropriate phase will be chosen for latching in the rest of the packet without the possibility of synchronization failure. While a formal proof of the correctness of the scheme is available [17], it will not be presented here due to lack of space. The basic idea of the proof is to follow the signals through the circuits and produce timing bounds on the transitions at each node. We begin by denoting the instant of the high to low transition for the start bit at the beginning of the packet as **time 0**. Node $N1$ will change its value at time t_{N1}, where $0 < t_{N1} \le T_\phi$.

Given the phase relationship between $CK20$ and $CK40$, two possible cases must be considered: **(1)** the rising edge of $CK20$ is at time $t_{N1} + T_\phi / 2$, or **(2)** the rising edge of $CK20$ is at time $t_{N1} + (3/2^*T_\phi)$. In the first case, the multiplexor will be set by $L4$ so that latch $L5$ will be clocked by $CK20^*$ while in the second case $L5$ will be clocked by $CK20$. In either case, $L5$ will latch in the first data byte between time $5/2^*T_\phi$ and $7/2^*T_\phi$. Since the first data byte is asserted on the link from time $2\ T_\phi$ and time $4\ T_\phi$, the byte will be latched by $L5$ at least $T_\phi / 2$ time units (12.5 nsec, in our circuit) away from the edge (change) of the input signal.

Typical drift of crystal-controlled oscillators is less than 0.1 percent. Thus, during the transmission of one 32 byte packet (a byte in every 50 *nsec*) the drift will be less than 1.6 nsec. The skews between the bits of a byte at the sender, together with minor variation in input pad characteristics and wire lengths, may cause an additional skew of one or two nanoseconds. Thus the skews and the clock drifts together will be significantly less than the ''safety margin'' of 12.5 nsec provided by our scheme. Hence our design guarantees that there will be no synchronization failures during packet reception (assuming that the start bit is properly synchronized) and that there will be no problems arising from small skews between the bits transmitted over the link.

Two alternative solutions to the problem of communicating between the nodes of a multicomputer have been proposed in the literature. In the Mark II Hypercube [18], when neighbors communicate, the clock of the sender is sent along side the data and is used to clock in the data at the destination. In the CalTech Torus chip [5], the problem of synchronization was solved by using self-timed design [14].

Both of the above solutions require additional lines on the link. In the first case, a line for bringing in the remote clock is needed. In the second case a *request* line and an *acknowledge* line are needed for the self-timed handshake protocol. The additional lines in these two solutions require more pins on the chip and more power to drive them. Since both pins and power are critical scarce resources in high-performance VLSI chips, these are major disadvantages. An additional problem with the first scheme is that while the FSM

controlling the input port can be clocked with the sender's clock, this FSM must interact with other FSMs on the chip that are clocked by different clocks. This introduces many opportunities for synchronization failures that will lead to low system reliability.

The problem of interacting with other parts of the receiver can be solved in the second scheme if self-timed design is used throughout the chip and there is no boundary between two timing domains that must be crossed [14]. While implementing the entire node using self-timed design is clearly feasible [5], most designers, design tools, and off-the-shelf components (which might be used elsewhere in the node) are better prepared to handle synchronous design. Furthermore, with self-timed design the transmission speed is limited since each byte requires the signal to be sent from the internal circuitry of the transmitter to that of the receiver and back again to the transmitter. As VLSI devices shrink, the cost of off-chip communication increases and the need for such a ''round-trip'' for the signal is a significant disadvantage.

IV. BUFFER DESIGN

As mentioned in Section II, our design requires a buffer that can receive data at a very high rate and allow the forwarding of a packet to be initiated before the entire packet is received. In order to meet these requirements we designed a two-port buffer, shown in *Fig. 4*, which does not require an address to be decoded for each byte, but instead uses shift registers to select successive bytes during packet reception and transmission. With this scheme there is no need for a fast decoder and the maximum possible time is available for the actual read and write operations, thus allowing the use of a relatively large buffer.

The buffer (see *Fig. 4*) has been layed out and simulated. It has one write-only port, one read-only port, and sixty-four bytes of storage. Simulation of this circuit indicates that it will operate well within the timing constraints imposed by the rest of the chip for a buffer with up to ninety-six bytes. The basic timing of the buffer operation is shown in *Fig. 2*.

V. DRIVING THE INTERNODE LINKS

The problems of driving the internode links are discussed in Section II. The characteristic impedance of the wires that will be used for the internode links is likely to be less than 100 ohms. One possible way to eliminate ''ringing'' on the link is to use matched termination in parallel with the input of the destination [6]. The problem with this scheme is that the power dissipation (DC as well as dynamic) per bit will be too high for a single CMOS chip to drive four byte-wide links. Instead, we will use *source termination* where a resistor is placed in series with the output pad-driver of the transmitter making the sum of the internal impedance of the output pin plus this resistor match the characteristic impedance of the line [6].

We have simulated and measured the power dissipated by the output pad-drivers we are going to use when they drive a line at the rate of 20 Mbits/sec. The power dissipation for each link (eight bits) is approximately 0.25 watts. Thus, it will be quite possible for a single CMOS chip to drive four links.

VI. SUMMARY

We have presented a design of synchronizing communication ports for asynchronous communication between multicomputer nodes. These ports minimize the probability of synchronization failure and support communication at a very high rate relative to the limits of the implementation technology. The scheme presented does not require any lines in the link in addition to the data lines and does not require byte-per-byte handshaking. We are now in the final stages of laying out a chip for testing our design. This chip, shown in *Fig. 3*, will test the functionality of the basic input and output ports but will not contain any of the higher level features of the ComCoBB chip. The test chip includes a simple interface to a MC68000 bus, and the functional testing will be carried out using a single board computer. Scan registers are used throughout the chip in order to enhance testability. The chip measures about 2.5 *mm* by 6.0 *mm* without pads, and it will fit in a standard 40 pin package.

References

1. R. L. Budziniski, J. Linn, and S. M. Thatte, ''A Restructurable Integrated Circuit for Implementing Programmable Digital Systems,'' *Computer* **15**(3), pp. 43-54 (March 1982).
2. X. Castillo, S. R. McConnel, and D. P. Siewiorek, ''Derivation and Calibration of a Transient Error Reliability Model,'' *IEEE Trans. on Comp.* **C-31**(7), pp. 658-671 (July 1982).
3. T. J. Chaney and C. E. Molnar, ''Anomalous Behavior of Synchronizer and Arbiter Circuits,'' *IEEE Transactions on Computers* **C-22**(4), pp. 421-422 (April 1973).
4. E. Chow, H. Madan, and J. Peterson, ''A Real-Time Adaptive Message Routing Network for the Hypercube Computer,'' *Eighth Real-Time Systems Symposium* (December 1987).
5. W. J. Dally and C. L. Seitz, ''The Torus Routing Chip,'' Comp. Sci. Tech. Report 5208, CalTech (January 1986).
6. D. Del Corso, H. Kirrman, and J. D. Nicoud, *Microcomputer Buses and Links*, Academic Press (1986).
7. L. A. Glasser and D. W. Dobberpuhl, *The Design and Analysis of VLSI Circuits*, Addison-Wesley (1985).
8. INMOS, *Transputer Reference Manual*, Nov 1984.
9. T. Kacprzak and A. Albicki, ''Analysis of Metastable Operation in RS CMOS Flip-Flops,'' *IEEE Journal of Solid-State Circuits* **SC-22**(1), pp. 57-64 (February 1987).
10. P. Kermani and L. Kleinrock, ''Virtual Cut Through: A New Computer Communication Switching Technique,'' *Computer Networks* **3**(4), pp. 267-286 (September 1979).
11. L. Kleeman and A. Cantoni, ''Metastable Behavior in Digital Systems,'' *IEEE Design & Test* **4**(6), pp. 4-19 (Dec 1987).
12. C. H. Ng, ''FIFO Buffering Transceiver: A Communication Chip Set for Multiprocessor Systems,'' Comp. Sci. Tech. Report 5055, CalTech (December 1982).
13. G. Peattie, ''Quality Control for ICs,'' *IEEE Spectrum* **18**(10), pp. 93-97 (October 1981).
14. C. L. Seitz, ''System Timing,'' pp. 218-262 in *Introduction to VLSI Systems*, ed. C. Mead and L. Conway, Addison-Wesley (1980).
15. C. L. Seitz, ''The Cosmic Cube,'' *Communications of the ACM* **28**(1), pp. 22-33 (January 1985).
16. Y. Tamir and C. H. Séquin, ''Self-Checking VLSI Building Blocks for Fault-Tolerant Multicomputers,'' *Int. Conf. on Computer Design*, pp. 561-564 (November 1983).
17. Y. Tamir and J. C. Cho, ''High-Speed Synchronizing Ports for Asynchronous Communication in VLSI Multicomputers,'' Computer Science Dept., UCLA (1988). In preparation.
18. J. O. Tuazon, J. C. Peterson, M. Pniel, and D. Liberman, ''CalTech/JPL Mark II Hypercube Concurrent Processor,'' *1985 Int. Conf. on Parallel Processing*, pp. 666-673 (August 1985).

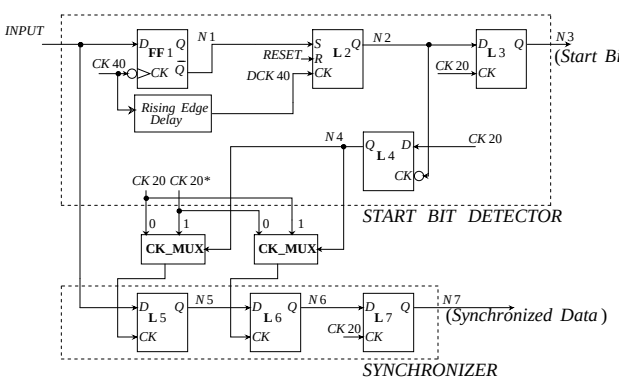

Fig. 1: Input Port Synchronizer

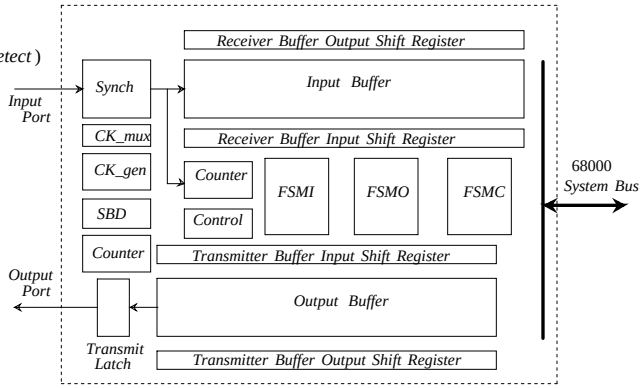

Fig. 3: Test Chip Floor Plan

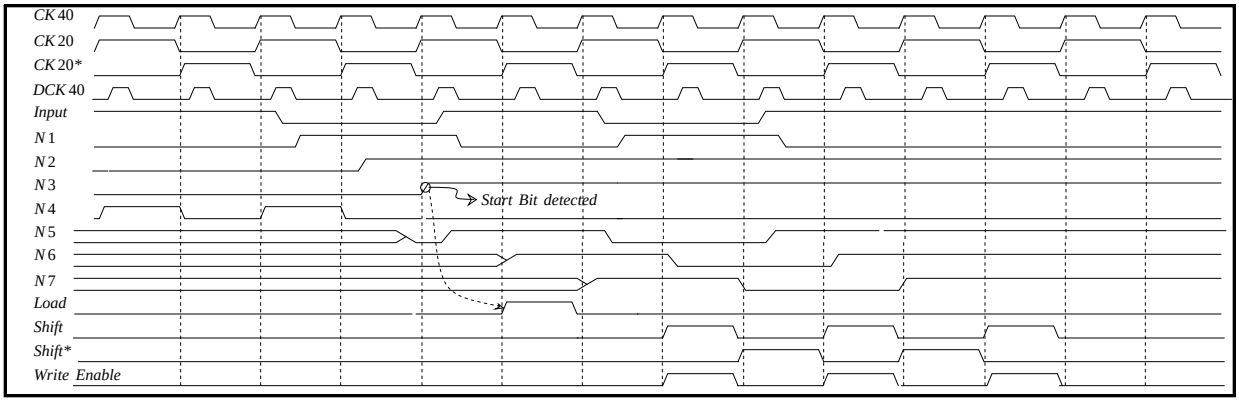

Fig. 2: A Sample Timing Diagram

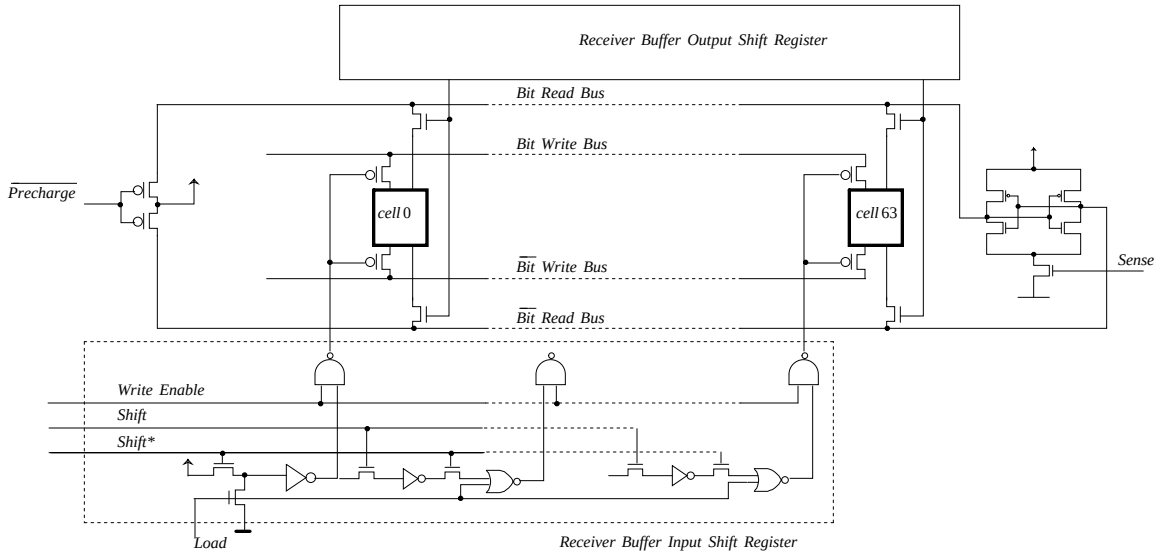

Fig. 4: FIFO Buffer Design