

A Distributed FIFO Scheme for System on Chip Inter-Component Communication

Ray Robert Rydberg III, Jabulani Nyathi and Jose Delgado-Frias
School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164-2752

Abstract

Interconnect delays are increasingly becoming the dominant source of performance degradation in the nano-meter regime, largely because the wires do not scale as fast as the devices. Scaling implies that complete systems can now be built on a single chip, requiring long interconnects for global signals and clock distribution networks. We propose having distributed first in first out buffers to facilitate communication between components/modules of highly integrated systems, such as system on chip. Using first in first out buffers will alleviate the clock skew, clock distribution and single clock synchronization problems; all of which are a result of global interconnect delays. In this paper, we present simple buffer circuitry and the associated control circuits that allow data transfers at a maximum frequency of 1.67 GHz in a 0.25 μm technology.

Keywords – system on chip, communication, buffer, wave-pipelining, clock distribution, interconnect

1. Introduction

Global on chip interconnects are increasingly becoming a limiting factor in the design of high performance system-on-chip (SoC). Some of the reasons cited as contributory to global interconnects being a limiting factor in SoC performance include: power supply drop variations, process variations, single clock synchronization [1], large wires with unpredictable delays [2], and interconnect power dissipation [3]. These limitations also imply that it would be difficult to achieve correct functional and reliable operations while maintaining low energy consumption within the interacting modules or components of SoC [3]. Proposed solutions to alleviate

these limitations include: separating the computation problem from the communication problem and introducing networks on chips [3], having communicating components operate asynchronously, while the computational blocks operate synchronously based on locally generated clocks [4] and optimizing repeater insertion for global interconnect [5].

Our work is based on the premise that global wires that span a significant fraction of the chip will impose signal delays that exceed the clock period [1]. Additionally, synchronizing operations on components running at different clock speeds is becoming more difficult due to clock skew and distribution [3]. With wire delays that exceed the clock period, it becomes apparent that the interacting components in a SoC design will perform computations much faster than the results could be transferred between components. This assertion can further be substantiated by the fact that transistor switching speeds are much faster than wire delays. It is shown in [1] that as the transistor switching speeds improve, the wire delays increase. A copper low- k 35-nm process technology is shown to have device switching speeds of 2.5ps, while interconnects of 1 mm in the same technology have delays of 250ps (two orders of magnitude slower). Better methods of transferring data between interacting components must be sought in order to prevent interconnects from being a performance limiting factor.

Much work has been done at the architectural, system and circuit levels to present some solutions to this problem. Some examples include: bus splitting, router based communication architectures and system level techniques such as communication based power management and adaptive supply voltage links [6]. In [3] the interacting components are viewed as a micro-network with communication taking place among the components. This “view” allows for the replacement of the global wiring on a chip with a general purpose interconnection network. There are also circuit level

techniques that have been explored to deal with the wire delay issue. The most widely used being the classical two inverter configuration with intermediate repeaters used for longer transmission distances.

2. Circuit Level Techniques

Existing literature shows that considerable work on repeater insertion at the algorithmic, as well as circuit, levels has been done [7]. The published work presents approaches of optimizing these drivers. Clock distribution networks also use the repeater insertion approach. In the nano-meter regime, repeater insertion will not cope with increased transistor switching speed. Computing components will produce results much faster than these results could be sent through the inverter chain. Synchronizing clocks will result in extensive dynamic switching even in the absence of data leading to considerable power dissipation.

The two inverter approach lacks memory and to avoid data overrun, the component producing data has to operate at the rate the signals can propagate through the inverter stages. Repeater insertion reduces wire delays. However, this reduction does not result in an increase of the transfer frequency to match that of the computing component. Figure 1 shows a diagram of the two inverter approach with each data line staggered in order to reduce inter-wire capacitive coupling.

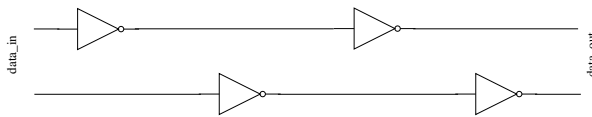


Fig. 1 Classical two inverter approach

The major disadvantages of this approach include: inability to synchronize with components that run at different frequencies from the sending component, need for multiple clock cycles for successful transfer of data implying that the sending component must accommodate this constraint to avoid data overrun, and need for multiple inverter stages to manage clock skew and distribution.

We suggest using a distributed first in first out (FIFO) buffering scheme to enable the communicating components to (i) operate at maximum frequency, (ii) provide a seamless interfacing capability for the interacting components, (iii) permit the communicating components and the FIFO to remain idle and retain their current status in the absence of data, (iv) eliminate single clock synchronization, and (v) reduce the delays associated with interconnects. Figure 2 is a block

diagram of the proposed scheme. Either one of the communicating components can signal the FIFO control circuit and initiate data transfer.

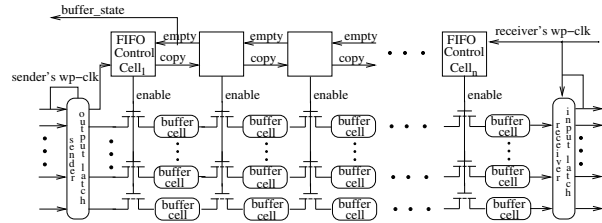


Fig. 2 The FIFO buffer array

The distributed FIFO scheme uses wave-pipelined clocks [8], [9] to trigger the start of a transfer. The hybrid wave-pipelined clocks arrive at any of the component's latches with their associated data and are thus ideal for starting the inter-component data transfers.

We realize the cost in area that will be incurred due to additional devices in the FIFO buffer. There is also a possible penalty on power dissipation, particularly under conditions that allow maximum speeds for data computation and transfers. The dynamic power component will be the most prominent in this case. Despite these drawbacks, the advantages are not mixed. Multiple clock domains can easily be synchronized with the distributed FIFO scheme. Communicating components and the FIFO buffer can lie idle in the absence of data without a need for additional circuitry to provide this functionality or to perform clock gating. Allowing the circuits to lie idle implies that there is no switching activity leading to low power dissipation (a claim yet to be verified). The distributed FIFO scheme enables for the elimination of the global clock since each stage of the control circuitry becomes self timed. The most important aspect of this approach is that the global interconnects can now allow data to propagate to the intended component at the same rate as the sender operates. Interconnects now have memory, permitting the communicating components to temporarily store the results on the data lines if necessary.

To realize the outlined advantages we use a modified GasP [10] control circuit and we add a feedback loop to the classical two inverter approach. The feedback loop gives the inverter pair capability to store data at the cost of two additional transistors. In the following we will describe how the GasP circuit provides precise control of the FIFO buffer. The GasP family of circuits is designed to provide control for simple pipelines; branching and joining pipelines and

round robin scatter and gather [10]. Figure 3 shows a schematic of a general GasP circuit and details of its operation and performance can be found in [10].

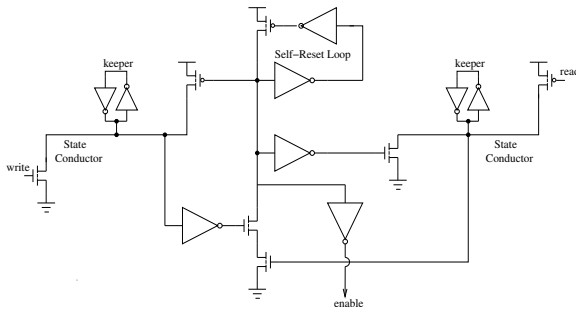


Figure 3 – The original GasP circuit schematic

The design's NAND gate is self-resetting, implying that careful transistor sizing is required. There are devices on this original design that can be eliminated without the possibility of compromising performance or functionality. To minimize the size of the GasP network, the possibility of redundant components was investigated. Figure 4 shows that three inverters in the original circuit can be eliminated, two from the self-resetting node and one from the forward path. With this reduction in devices, appropriate transistor sizing becomes critical in order for the circuit to perform as required. Our GasP design is intended for communication link control.

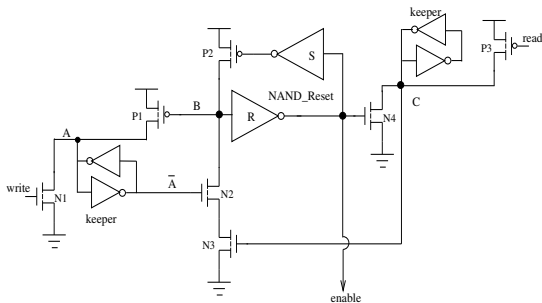


Fig. 4 The basic FIFO control circuit

The design uses a self resetting NAND gate formed by transistors N_2 , N_3 and P_2 , along with inverters r and s . The operation of the NAND gate's pull-down network depends upon the status of the current latch's neighboring latches. The *read* and *write* signals driving transistors P_3 and N_1 , are the same signals labeled as *sender's wp-clk* and *receiver's wp-clk* in the block diagram of Figure 2. These signals must be active in order for the NAND gate to allow data to be copied from the previous latch to the current latch. If the next

latch has no request or currently holds a value that is yet to be copied to its succeeding neighbor, the output of the NAND gate remains at logic 1. This results in the *enable* signal being at logic 0. In the event that the next latch is ready to receive new data the *read* signal is pulled low. However, data can only be transferred to the next stage only when the previous latch has indicated that it has data to be transferred by signaling a *write*. On the other hand, the previous stage can indicate that it has data to be transferred but if the next latch has not indicated that it is ready by raising node C to logic 1, no transfer would take place. A successful transfer occurs only when both transistors N_2 and N_3 are turned ON. The self resetting NAND gate would set its output to logic 1 while its inputs are reset to logic 0 after a successful transfer of a logic value from the current stage to the next stage. The FIFO control circuit has inverters designed to maintain any given state at nodes A and C (the keepers in the figure). This permits for proper operation (i.e. in the absence of any activity at either the sending component or the receiving component these nodes would hold logic levels that ensure that the output of the NAND gate is at logic 1). In the event that the sending component has results while the receiving component is servicing other components or inaccessible, we can fill the FIFO buffer and hold the data until the component it is intended for is ready to read the stored values.

Our design of the control circuit has a master reset that initializes nodes A , B and C to logic 1 at start-up. Initializing the design in this manner allows transistors N_2 and N_4 to be OFF while transistor N_3 is ON. This status indicates that the buffers are empty and data can be transferred from one communicating component to another. The sending component produces a pulse (the wave-pipelined clock), indicating that there is data to be transferred. Since the buffers are empty (signified by transistor N_3 being ON), the latches will be transparent momentarily, permitting the data to be copied to the receiving component in a very short time. For proper operation, the *write* signal must be at logic 1 for an inverter delay. Keeping the write pulse longer than one inverter delay would result in short circuit current being drawn since transistors P_1 and N_1 would eventually be conducting simultaneously. This would exacerbate power dissipation. Cascading the basic cell of Figure 4 would enable for progressive transfer of data along the data line. Our model requires that the receiving component indicate its readiness to accept new data and the sending component would thus set the *write* signal high, triggering a situation in which the enable signal of each stage is sequentially set to logic 1 after long enough delays to prevent data overrun.

This action enables (turns them ON) transistors on the data lines thereby passing the data from one stage to the next. The sending component is always appraised of the status of the FIFO buffer and the receiving component via the *empty* signal (Figure 2) of the first entry of the FIFO controller. We thus have a way to determine if the buffer is full or empty. If the receiving component stalls the sending component is made aware via this empty signal. In Figure 5, we show a schematic of the circuit used on the data line to break the interconnect delay. The circuit is a simple modification of the two inverter repeater with two additional pass transistors. The gates of the added transistors are driven by the enable signal and its inverse; generated by the control circuitry.

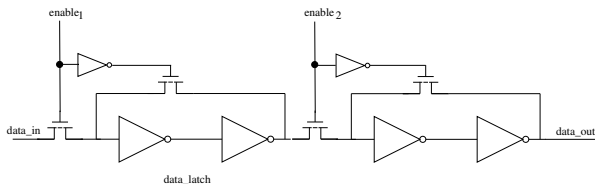


Fig. 5 Data line circuitry

The transistor on the data line allows data from the previous stage to be copied to the next stage whenever the enable signal is raised to logic 1. The period during which the latch is transparent depends primarily on how fast transistor P_1 of the control circuit can set node A to logic 1 after a brief active write signal is received. Note that this delay can be tailored to suit the required duration that would be long enough to permit data to be copied to the current latch. If the current latch contains data that has not yet been copied to its succeeding neighbor, even with the write signal asserted, there will be no data transfer. This is achieved by having the stage succeeding the current one ensure that transistor N_3 is at logic 0. The transistor in the feedback loop gives the two inverter repeater memory. Keeping the enable signal at logic 0 allows the latch to retain its current value until it can be changed when new data is received. Copying of data takes place very fast, thanks largely to improved transistor switching speeds. The approach in designing the data bus in this manner is motivated by the fact that there is a need to break the RLC delays on long global interconnects. In the nanometer range, inductance will contribute to performance degradation due to these interconnects. The rate at which the component producing data operates no longer becomes limited by the communication delays. Synchronizing multiple clock domain components can easily and efficiently be achieved using this approach. Performance issues

associated with clock distribution can be alleviated by using this approach, since it permits for globally asynchronous and locally synchronous operations.

3. Simulation Results

Figure 6 shows plots of the nodes of interest within a unit cell of the control logic. We duplicate these cells at appropriate intervals (3 mm for our experiment in a 0.25 micron technology) to form a distributed FIFO buffer for a typical SoC design.

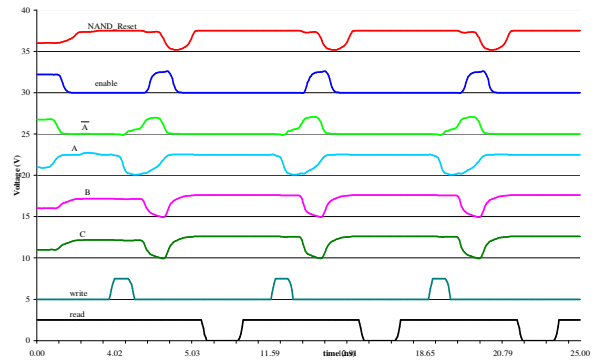


Fig. 6 Plots from FIFO control circuits

We analyze a single stage of both the control and the data line circuits and report the associated delays which can be read from the charts of Figures 6 and 7. At the onset of a write request 390.6 ps elapse before the latch is enabled for data copying to occur. This is remarkably fast at this technology node. If on the other hand we assume that a request to transfer was issued, while the next stage was full it would take 190.6 ps to generate the enable pulse once the succeeding stage issues a ready signal (read). We need 323.1 ps to reset the NAND gate and still drive the capacitance of the line enabling 32 pass transistors on the data lines. It takes 246.8 ps to turn OFF transistor N_2 after a successful transfer, 229.2 ps to turn OFF transistor N_3 after a successful transfer, and 599.1 ps to perform a succeeding transfer. These delays provide some sense of the overhead incurred from having such control circuitry to control bundled data buses. It must be noted that the reported delays take into account enabling a 32-bit bus. This implies that a single control stage is designed to gate signals on 32 data lines. The recorded values are based on the required delays for proper operation. Figure 7 shows plots of a burst transfer. The simulated case is one whereby the initial transfer has sparse data bits and no transfers are required for some time and eventually several data bits

need to be transferred in rapid succession. The generated enable signal has a maximum frequency of 1.67 GHz and this is limited by the node capacitance of the self resetting NAND gate (labeled as node B in Figure 3). This node has the largest load amongst all the nodes of the circuit designed to generate control of the data line FIFO.

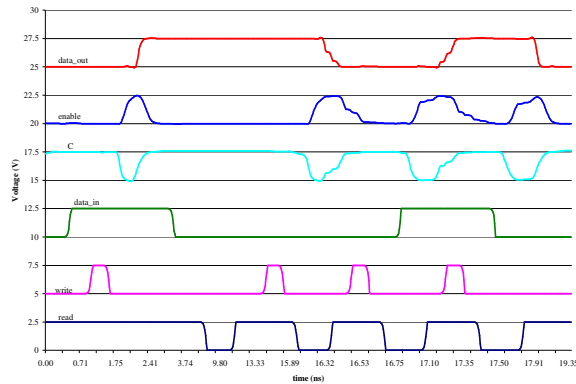


Fig. 7 FIFO burst mode operation

We have included in Figure 7 the data signal and the corresponding output signal. The traces shown are for a case in which after initialization of the control the receiving component is immediately ready to accept data. However, valid data is not immediately available. The receiving component maintains the request until the sending component has data (at approximately 0.72 ns on the chart). The receiving component sends no requests for 16 ns. In the mean time, data on the data line is held at the previous value. After 16 ns there is a burst of activity allowing successive transfer of data at 1.67 GHz with the 010₂ pattern being transferred. The signals in Figure 7 show this operation.

4. Concluding Remarks

In this paper, we have presented a distributed FIFO scheme that permits communicating components on a SoC design to perform data transfers asynchronously. The sending component needs to provide the wave-pipelined clock pulses that permit it to write data to the buffer. Successful writes to the buffer will take place only when the receiving component provides wave-pipelined clock pulses. These pulses indicate that data can be passed from one buffer to the next along the data line. The receiver can issue read pulses for as long as there are empty FIFO stages. The operation of the FIFO control circuitry ensures that in the absence of either the read or the write pulses, there is no activity; hence, no dynamic power dissipation. The distributed FIFO scheme addresses the global interconnect

problem, allowing for the replacement of global wires with a scheme that includes memory. There is also a great potential to alleviate the multiple clock domain interface problems. We also remove the need for a global clock, allowing the clock to travel with its associated data and the communicating components to have an alternative synchronization approach.

5. References

- [1] L. P. Carloni and A. L. Sangiovanni-Vincentelli, "Coping with Latency in SOC Design," *IEEE Micro*, Oct. '02, pp. 24-35.
- [2] R. Siegmund and D. Muller, "Efficient modeling and synthesis of on-chip communication protocols for network-on-chip design" *Proceedings of the 2003 International Symposium on Circuits and Systems*, Vol. 5, May 25-28, 2003, pp. 81-84.
- [3] L. Benini and G. De Micheli, "Networks on Chips: A New SoC Paradigm," *IEEE Computer*, Jan. 2002, pp. 70-78.
- [4] A. Iyer and D. Marculescu, "Power and Performance Evaluation of Globally Asynchronous Locally Synchronous Processors," *29th Annual International Symposium on Computer Architecture*, May 25-29, 2002, pp. 158-168.
- [5] V. V. Deodhar and J. A. Davis "Voltage scaling and repeater insertion for high-throughput low-power interconnects," *Proceedings of the 2003 International Symposium on Circuits and Systems*, Vol. 5, May 25-28, 2003, pp. 349-352.
- [6] V. Raghunathan, M. B. Srivastava, and R. K. Gupta, "A survey of techniques for energy efficient on-chip communication," *Design Automation Conference, Proceedings*, June 2-6, 2003, pp. 900-905.
- [7] M. L. Mui, K. Banerjee, and A. Mehrotra, "Global Interconnect Optimization Scheme for Nanometer Scale VLSI with Implications for Latency, Bandwidth, and Power Dissipation," *IEEE Transactions on Electron Devices*, Vol. 51, February 2004, pp. 195-203.
- [8] J. Nyathi and J. G. Delgado-Frias, "A Hybrid Wave-Pipelined Network Router," *IEEE Transactions on Circuits and Systems, I: Fundamental Theory and Applications*, vol. 49, no. 12, pp. 1764- 1772, December 2002.
- [9] J. G. Delgado-Frias and J. Nyathi, "A Wave-Pipelined CMOS Associative Router For Communication Switches," *2000 IEEE International Symposium on Circuits and Systems*, Geneva, Switzerland, pp. 391-394, May 2000.
- [10] I. Sutherland and S. Fairbanks, "GasP: A Minimal FIFO Control", *Proc. Of ASYNC*, 2001, pp. 46-53.