Performance Analysis of an Asynchronous Buffer for Crossing Clock Domains in 32nm CMOS Technology

Sam Payne, George Touloumes

Introduction

The number of cores in high-performance computing systems has continued to grow in an effort to increase computing power. However, each extra core places more stress on system networks, causing memory and inter-chip communications to become a critical bottleneck in achieving higher performance [1]. Given that the number of pins on a processor's packaging has increased more slowly than the speed of processor performance, off-chip bandwidth has become a critical obstacle to achieving high-performance in large systems. A component involved in bridging clock domains is the asynchronous buffer. For this project, we have designed an asynchronous buffer to study how a component accepting two different clocks performs and consumes power [Fig. 1]. The primary goals of this project were to create a fully functional, synthesizable Verilog design of an asynchronous FIFO and simulate it to determine functionality and power consumption in state-of-the-art 32nm CMOS component libraries. Previous work has studied the original FIFO design functionality [2], and other researchers have compared different implementations of synchronous and asynchronous FIFOs [3]. Our results provide specific analysis of how dynamic power, static power, throughput, and latency change under different read frequencies, write frequencies, and system loads.

Design

The design contains a dual port memory driven by a read and write clock. The buffer delivers messages from a write client to a read client in FIFO order. The write client provides a data signal and data valid signal, and, in return, the FIFO sends the write client a "full" signal to prevent overwriting unread values. The read client sends an enable signal, receiving a data signal and an empty signal to prevent junk values from being read. The buffer tracks the read and write locations to generate the full and empty signals. The synchronization registers are in place to mitigate the violation of setup and hold times. Since the clocks are not in sync with each other (in frequency or phase), any signal crossing the clock domain must be synchronized. The first register may go meta-stable, but the output will settle before the next clock cycle due to regenerative feedback inside the register. Thus, the second register sees a stable value (either 0 or 1).

Additionally, we must convert our read and write counters to Gray Code before crossing clock domains to ensure that only one bit goes meta-stable. Otherwise, values other than the previous count or current count might be stored. Gray Coding ensures that when incrementing a number, only one bit changes at a time, guaranteeing that only one bit can be meta-stable. The drawback of using synchronization registers is that there is a two-cycle delay when transferring counters between clock domains, making the full and empty signals conservative by two cycles.

Methodology

To test the buffer, we wrote a test bench to stimulate the device with different traffic densities (10-100% load) using read and write clients functioning at a variety of read and write frequencies (5 – 500 MHz). Each test wrote 255 randomly generated 64 bit values into a buffer of depth 16. Using vcs, we can check the functionality of the device under each of these conditions though an event-driven simulation. Design compiler tools can use 32nm library cells to implement this design. We used Primetime PX to analyze the power consumption of the design based on the events in the simulation and the 32nm design library. The results are discussed below.

Results and Discussion

In Figure 2, the average power consumed increases in a linear trend as the read frequency increases, as we could expect from CMOS technology. (the same is true of write frequency, though not shown). Figure 4 demonstrates that as the write frequency increases, static power as a percentage of total power increases rapidly due to lower throughput and longer simulation time (the longer simulation time leads to a longer time for leakage to occur between switches, which occur less often at lower frequencies). In contrast, dynamic power increases as expected. Figure 3 is an important study for this kind of device, demonstrating how varying loads affect the latency in the FIFO. In this test, write frequency is held at a constant 50 MHz. The figure clearly illustrates which loads will fill the FIFO at different read periods, as indicated by the jump in latency (occurring at different points for each load). If possible, a designer should design a system with read frequency and average load in mind to reduce latency through the FIFO. Figure 5 illustrates how throughput changes for varying read/write frequencies. At a point, throughput becomes limited by the low write frequency, as seen by the tail at the right end of the graph. On the other hand, the differing plateaus at the left of the graph show the point at which the read frequency is limiting performance. Ideally, the read and write frequency would be close for the sake of achieving high throughput without wasting power through a needlessly high frequency on the other clock. (Many systems of course do not allow for this matching of frequency – lest we defeat the purpose of having different clock domains). Figure 6 demonstrates how the energy per bit changes for different read and write frequencies; we can see that maximum efficiency is achieved when both the read and write clocks are fast. We can also note that if one clock is slow by comparison, the efficiency suffers (as demonstrated by the rising energy sections at the right and left quadrant side of the graph).

Summary

We have built and simulated a 32nm asynchronous FIFO across varying read and write clock frequencies. Overall, we characterized the trade-offs of these frequencies under varying system loads in terms of both power and latency.

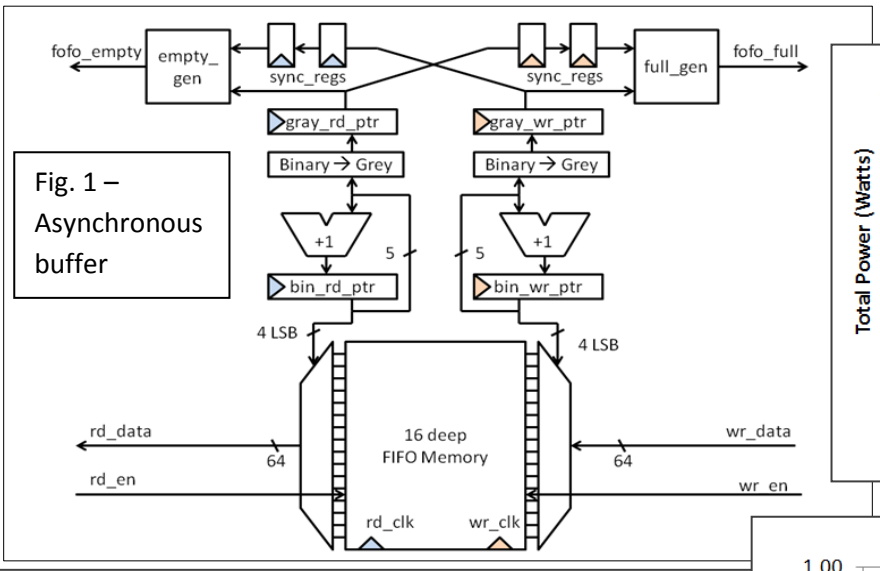
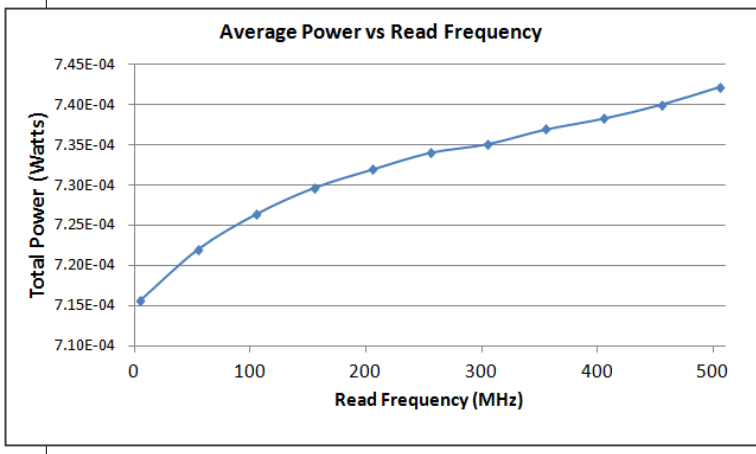
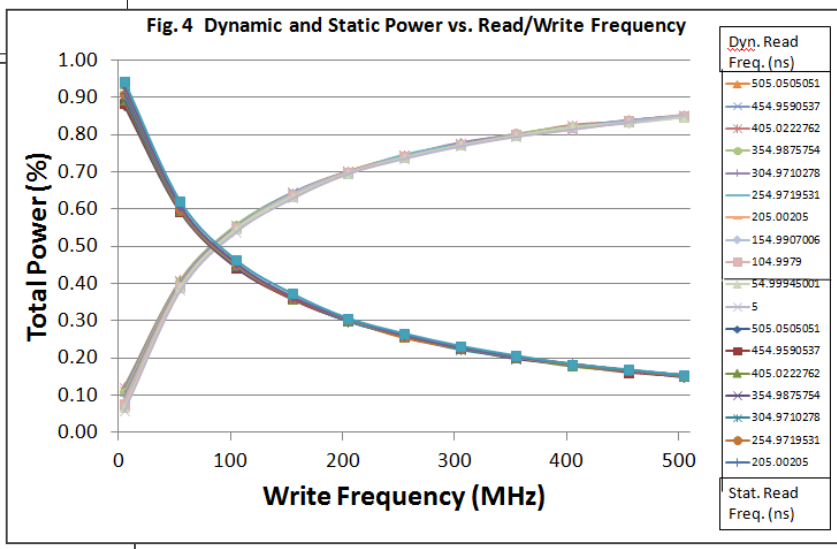
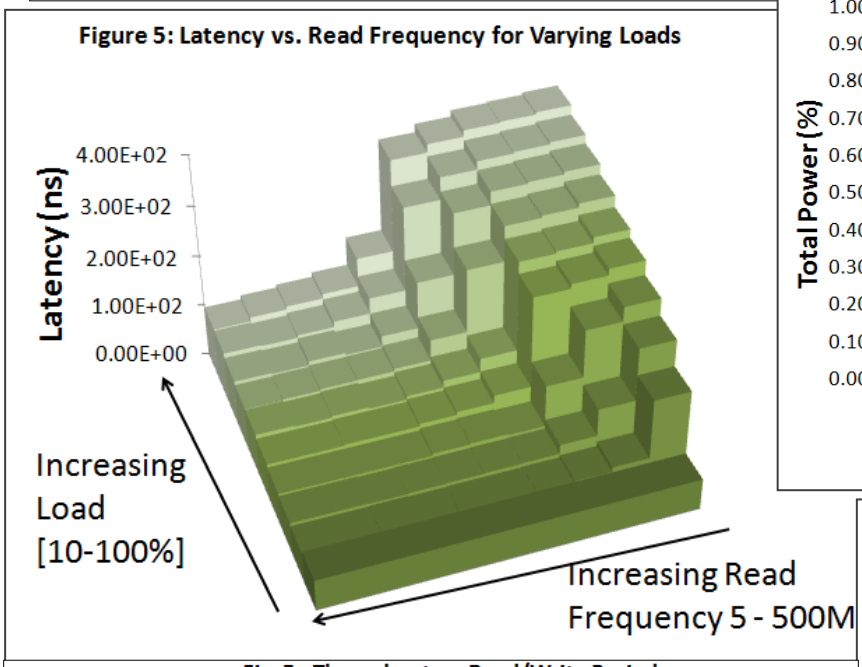
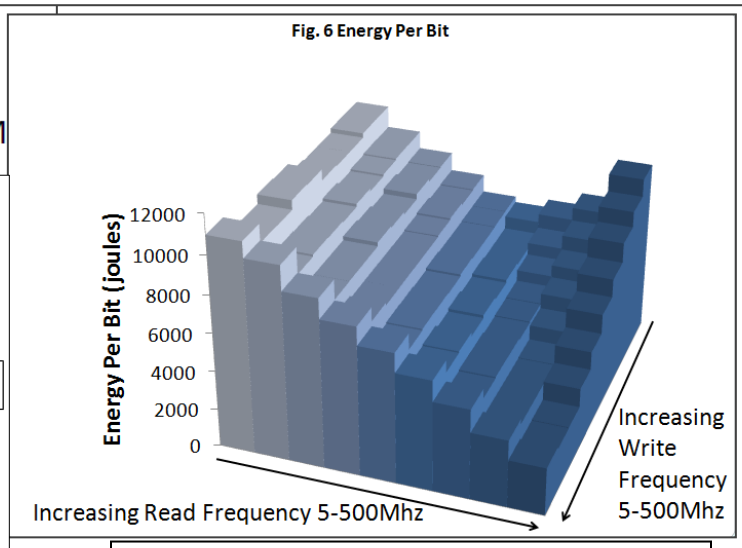


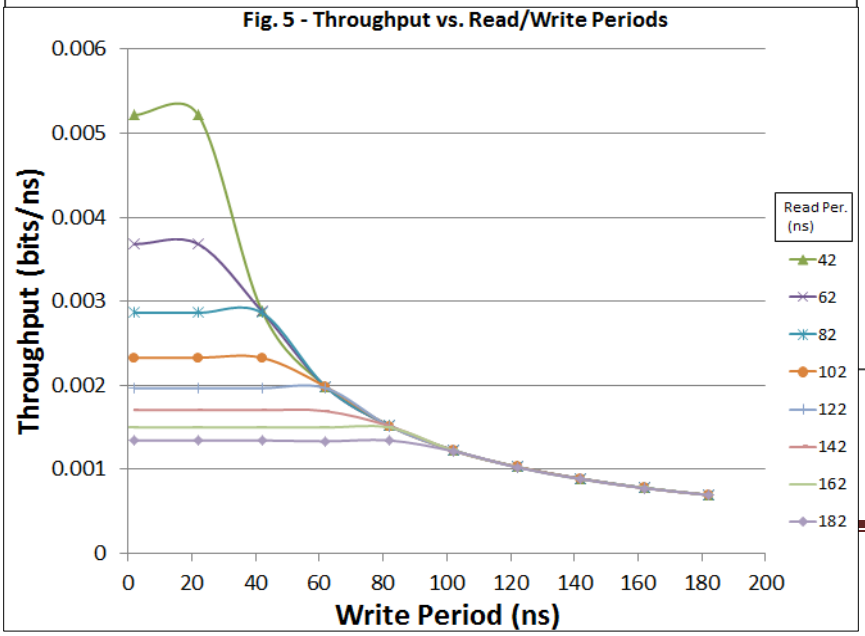
Fig. 1 – Asynchronous buffer











References

[1] Ahsan, Bushra, and Mohamed Zahran. "Cache performance, system performance, and off-chip bandwidth... pick any two." *Proceedings of INA-OCMC* (2009).

[2] Cummings, Clifford E. "Simulation and synthesis techniques for asynchronous FIFO design." *SNUG 2002 (Synopsys Users Group Conference, San Jose, CA, 2002) User Papers*. 2002.

[3] Han, HoSuk, and Kenneth S. Stevens. "Clocked and asynchronous FIFO characterization and comparison." *Very Large Scale Integration (VLSI-SoC), 2009 17th IFIP International Conference on*. IEEE, 2009.