Amany El Gouhary
ECE 6770
Spring 2008
Final Report

Linear Asynchronous and Synchronous FIFOs in Different Scaled Libraries

Abstract

In this project we examine how power and area of asynchronous and synchronous linear FIFOs scale in comparison to ideal scaling. The libraries used for this purpose are the UofU digital library, which implements a 500nm technology and the IBM Artisan library which implements a 130nm technology. Both FIFOs are implemented using static logic.

Introduction

As MOSFETs scale down, the supply voltage cannot keep scaling down with the same rate. The need for higher performance and the constraints on threshold voltage limit how much we can go down with the supply voltage. That makes power consumption a huge issue in scaling. Asynchronous design might be a good choice in the future as it provides some potential for saving power consumption and also space. In some designs such as the FIFO, there might be some lots of idle cycles where power will be lost anyway in the usual clocked synchronous scheme. In this paper we show the results of implementing asynchronous and synchronous FIFOs in 500nm technology and 130nm technology. Both FIFOs will use controllers that facilitate interfacing to other asynchronous or synchronous parts. The asynchronous controller employs a handshake protocol while the synchronous one employs a phased elastic channel protocol.

Linear Synchronous FIFO Overview

In order to be able to interface with other asynchronous parts, we will implement a phased elastic channel protocol. The state machine illustrating this protocol is shown in figure 1. The stall signal travels backward while the valid signal travels forward. Figure 2 shows an implementation of the phased elastic channel protocol control logic. [1] The one implemented in this paper utilized flip-flops instead of the latches for simplicity.

vl-sr	vr-current	vr-next	sl-current	sl-next	en
00	x	0	x	0	0
01	x	1	x	0	0
10	x	1	x	0	1
11	x	1	x	1	0

Table1: Synchronous FIFO Controller Logic Table

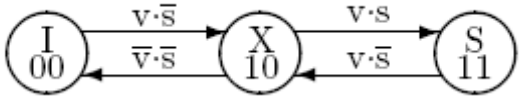

Figure 1: Phased Elastic Channel Protocol state machine.[1]

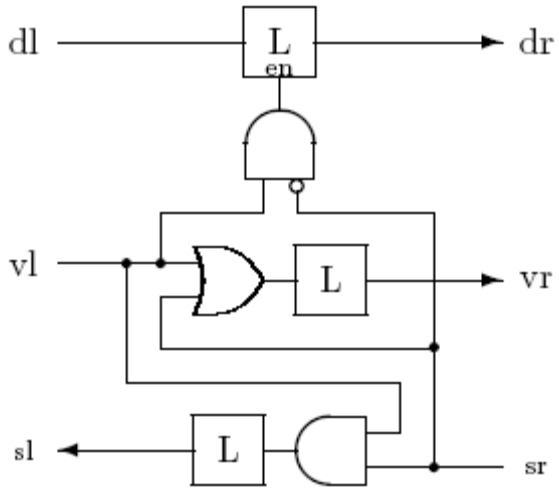

Figure 2: Phased Elastic Half Buffer Logic[1]

Linear Asynchronous FIFO Overview:

We use a handshake protocol controller that could be specified in CCS format as follows[2]:

$$\mathbf{L} = li \uparrow c \; \overline{lo} \uparrow li \downarrow \overline{lo} \downarrow \mathbf{L}$$

$$\mathbf{R} = \overline{c} \; \overline{ro} \uparrow ri \uparrow \overline{ro} \downarrow ri \downarrow \mathbf{R}$$

$$\mathbf{FIFO} = (\mathbf{L}|\mathbf{R})/\{c\}$$

In brief, when the left input (li) is on, the left output (lo) and right output (ro) are on at the same time. When left acknowledge goes on left input can go off, then left output can go off. Right output is not supposed to go off until right input goes on. The waveforms in the figure below illustrate the different transitions.

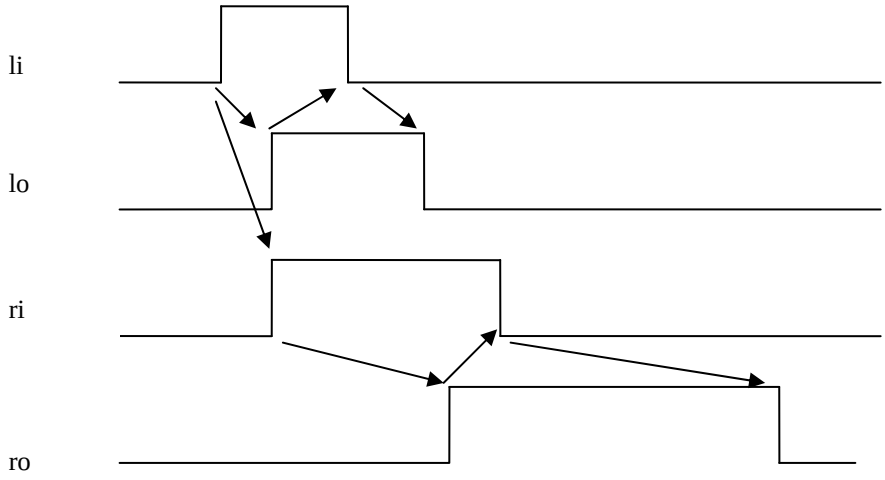

Figure 3: Wave form for the handshake protocol used in the Asynchronous FIFO design

Using petrify a static implementation of this protocol was extracted. There are several alternative circuits; one of them is shown in the figure 4. The four input NAND gate might not be a good choice and will probably be replaced.

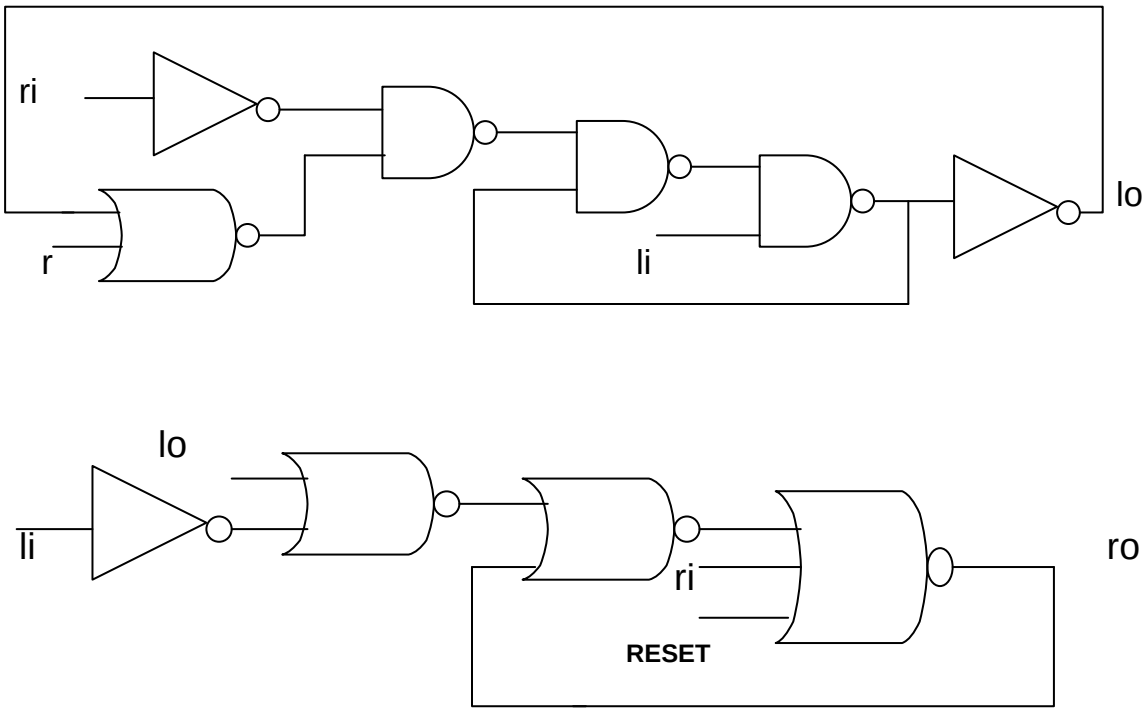

Figure 4: Petrify Implementation of Asynchronous FIFO Controller using Static Logic

Testing and Verification

Both FIFOs are going to pass 32 bit Data and each will have a length of 8.

Golden Model for the Synchronous FIFO:

C++ program is used to generate golden model results to verify that the controller signals are correct and data is correctly propagating after exactly 8 clock cycles. The program generates random 32 bit data and writes them as input and after 8 clock cycles, the same data is written as output.

Golden Model for the Asynchronous FIFO:

The same program used for the synchronous FIFO is modified to be used in asynchronous except that we don't expect a delay of 8 clock cycle to read the data. So after certain delay time, we expect the data to be ready for output.

Synthesis

Synchronous FIFO Synthesis

Design compiler from synopsis was used to create the RTL verilog. Since speed is not the major concern here, the clock constraints were relaxed a little bit so we end with a reasonable area estimates. The clock for the UofU was set to be 15 ns and for the Artisan 100ps.

Asynchronous FIFO Synthesis

By creating a virtual clock, we were able to use design compiler as well. Since the asynchronous controller is delay sensitive, we used the command set_size_only to keep the compiler from optimizing it.

Results

The area and power estimates are all summarized in tables 2 and 3 below.

Simulation

Synchronous FIFO Simulation

The script reads the input data from the goldenmodel results file and inserts them into the fifo then tries to read the next cycle and so on till all data are read. With the UofU library, there were no problems. The minimum clock period required was 10ns. With the Artisan library, there was a problem; the enable signal arrived later than the data flip-flops were clocked. Adding delay buffers to the clock of the data flip-flops solved the issue. The minimum clock period was around 15000ps.

Asynchronous FIFO Simulation

Same as before we read data from the goldenmodel results file and then read it. For both the UofU and the Artisan library when evaluating the 'ro' signal the compiler decided to take the 'lo' signal from an earlier wire which resulted in 'ro' being evaluated using the next 'lo' instead of the current one. Adding buffers before 'lo' goes to the ro circuit solved the issue. The 'ro' part of the aynchronous controller is now as shown below in figure 5.

Furthermore, in the Artisan library, the output signals were not staying on enough for the data to propagate through the latches. Buffers added to the controller again to delay the ri and li.

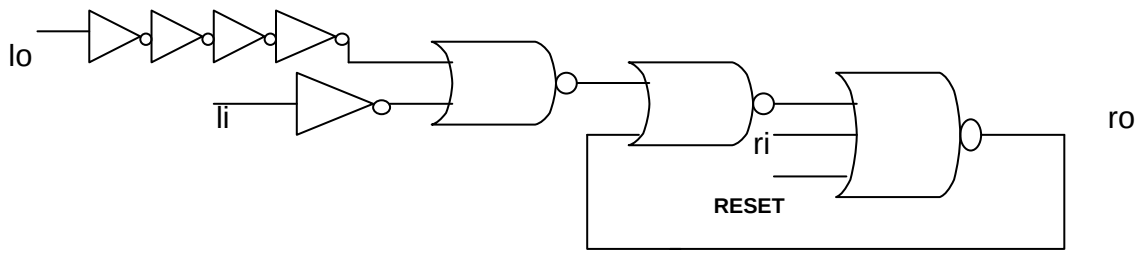

Figure 5: Asynchronous controller after adding buffers to solve timing issue.

Layouts

For the layout, we tried to achieve high densities. In general the specified densities were around 75%. The resulting densities were around 85%.

Results

Table 2 shows the area estimates from design compiler and the actual area measured from the layout generated by soc-encounter. Table 3 below summarizes the power consumption as approximated by design compiler and soc encounter.

In order to get more accurate power results, the timing file (sdf) generated by soc-encounter which accounts for interconnects was used to back annotate the cells. Then, a vcd file was generated from the simulation mentioned earlier in modelsim and loaded back into soc-encounter. The vcd file allows us to add the activity to soc-encounter power consumptions estimates. That was done only for the Artisan library as the UofU flip-flops and latches did not have "specify" statements. Since we are writing and reading from the FIFOs all the time, the activity factor is about 100%.

	Design Compiler		Soc-encounter	
	Asyn.(um^2)	Syn. (um^2)	Asyn.(um^2)	Syn.(um^2)
UofU(500nm)	275,853.6	494,164.8	382,900	870,064
Artisan(130nm)	8,519	49,141.44	16,234.8	71,346
Ideal Scaling	18,650	33,410	25,880	58,820

Table1: Area estimates from Design Compiler and Actual area measured of the final layout generated by Soc-encounter.

Discussion

Area Results

We notice that in case of the asynchronous FIFO, the Artisan library actually provides better than ideal area scaling while the synchronous FIFO scales worse than ideal. That might be because of the clock overhead that is not expected to scale as well.

We also notice that the synchronous FIFO is almost double the asynchronous one in area. This is probably due to the fact that we use flip-flops in the synchronous design and latches in the synchronous one.

	Dynamic Power(mW)		Leakage Power(mW)		Total Power (mW)		
	Asyn.	Syn	Asyn	Syn	Asyn	Syn	Source
UofU (500nm)	28.53	63.44	59.84e-6	152.3e-6	29.37	63.592	Design Compiler
Artisan (130nm)	8.15	232.4e-3	7.47e-6	37.12e-6	8.15	232.36e-3	Design Compiler
Ideally	1.93	4.29	15.56e-6	39.52e-6	1.99	4.3	
UofU (500nm)	83.8	293.3	6.31e-5	1.44e-4	83.8	293.3	Soc (no vcd)
Artisan (130nm)	6.76	0.78	8.59e-6	1.75e-5	6.77	0.778	Soc (no vcd)
Ideally	5.66	19.83	16.4e-6	3.74e-5	5.66	19.83	
UofU (500nm)	N/A	N/A	N/A	N/A	N/A	N/A	Soc (vcd)
Artisan (130nm)	14.28	11.4	9.5e-6	1.76e-5	14.3	11.4	Soc (vcd)

Table3: Power estimates from Design Compiler, Soc Encounter with no activity and after adding activity using modelsim generated vcd file

Power Results

From table 3 above, we notice that in case of the 500nm library, the early estimates from design compiler are much smaller than the one obtained after the interconnect, except for the Artisan asynchronous FIFO. That means that the interconnect RC consumes as much power as switching.

Dynamic Power: The synchronous design seems to be doing better than the asynchronous one and also better than ideal scaling. But when we add the activity factor to our evaluation the switching power of the synchronous FIFO gets very close the asynchronous one.

Leakage Power: obviously the asynchronous one is doing better in the leakage power area. The leakage power is better than ideal scaling and also much better than the synchronous one. That means that asynchronous design might have potential for further scaling.

We should note that the artisan library is a commercial one that has been optimized while the UofU is simple library that was designed for learning purposes in the University of Utah. Therefore the comparison is not conclusive.

Conclusion

In conclusion, we have implemented a synchronous linear FIFO and an asynchronous linear FIFO in a 500nm technology and 130nm technology. The asynchronous design employed a handshake protocol while the synchronous one employed a phase elastic channel protocol. We used design compiler for both designs to generate the RTL verilog and also power and area estimates. Modelsim was used to simulate both designs and generate the vcd activity file. We used soc-encounter to generate the layout and also acquire better estimates for area and power consumption with and without the activity file. The results show that artisan library (130nm) in general did better than ideal scaling except for the switching power of the asynchronous design. In addition, synchronous FIFO had less dynamic power and more leakage power than the asynchronous FIFO which shows potential for more efficient scaling in the asynchronous area as leakage power is a major concern.

References:

[1] You, J, Xu Yang, Husak Han, K. S. Stevens. *Performance Evaluation of Elastic GALS Interfaces and Network Fabric.* Electronic Notes in Theoretical Computer Science, URL: www.elsevier.nl/locate/entcs.

[2] Han, Husak. *Performance and Energy models for Synchronous and Asynchronous FIFOs.* Thesis Proposal, Advisor Kenneth Stevens.