Homework 5

Matt Hartnett, ECEN 5863

Q1)

A) I implemented the synchronizer in Verilog, as included in my project files. I was able to verify that my Verilog implementation matched the description by utilizing the Libero RTL viewer, as shown in figure 1.
After confirming that the implementation was correct, I ran place and route, then simulated. Instead of creating a stimulus file, I manually controlled the input stimulus to the simulation, which resulted in the screenshot shown in figure 2. As can be seen in the simulation, a change to the input data takes a long time to propagate, with the output data toggling at approximately 1/9th the speed.
As shown in figures 3 and 4, SmartTime did not have any errors. While it was able to report Fmaxes for both clocks, it wasn't able to report slack since the design was not constrained at this point. No errors were generated either, although there were many warnings about unconstrained clocks and data paths.
After applying clock, input, and output constraints, SmartTime reported the paths shown in figures 5 and 6.
The initial slack time was not calculated, as the required data time could not be found without a clock frequency constraint. However, after constraining both clocks, I found a lowest slack time of 18.964ns, as shown in figure 5.

B) I created a new Libero project and imported the same Verilog file for the implementation of the 3 FF synchronizer.
Figure 7 shows a screenshot of Modelsim with a simulation where Bclk is the fast clock and Aclk is slow. Since the data generation is clocked off of Aclk, the toggling occurred much slower than part A. Since the signal was being generated slower, the Dout signal was running at approximately the same rate as the data generation, but there was a fair amount of swing in the high and low time, resulting in pulse duration variation.
Similar to part A, without constraints there wasn't any slack calculation in SmartTime, as shown as figure 8 and 9.
After constraint application, the screenshots in figure 10 and 11 were taken, which indicate the slack calculations once the clocks have the correct constraints.
The lowest slack found was 19.139ns, as shown in figure 11.

Q2)

A) I created the SlowFast design in VHDL (variety is the spice of life), starting with a single bit toggle FF that fed into the shift register. The shift register worked by advancing bits [2:0] up to [3:1] and then adding the most recent data value from the toggle signal. For the sync signal, I created a FF that clocked in the value of Bclk on the edge of Aclk to generate a syncSignal output. This syncSignal controlled a mux that switched the data input into the sync FF, which was clocked on Aclk. When the syncSignal was high, it clocked the shiftRegister, and when syncSignal was low, it looped its own input back into itself. The output of the sync FF was then fed into a final output FF that was clocked on the edge of Bclk. This design can be viewed in the RTL viewer, as shown in figure 12.
The fMax for this design was 609.756MHz for Aclk, and 505.051MHz for Bclk (as reported by SmartTime, shown in figure 13).
The simulation is shown in figure 14, with 5 total data transfers.
As shown in the screenshot, there are 5/5 correct data transfers. It's difficult to tell, but a data transfer is occurring every time there is a rising edge of Bclk. Since the shift register is being fed by a toggle, it only has two valid states: 0x5 and 0xA. With the sync signal lining up with the shift register in a consistent manner, the Dout value stays the same. If the shift register was changing in a more unique fashion, such as a random number generator, then the data transfers would be much more obvious. However, we can still see that the syncSignal is working its magic, as the data on the output FF is stable before and after the rising edge of the Bclk, meaning that the data is stable. Adding constraints won't improve the accuracy of the synchronization, as it's already working as intended.

B) In order to switch the clock frequencies, several design changes had to be made. The sync signal needs to be generated by registering the slow clock on the fast clock, meaning that I had to switch Aclk and Bclk for the synchronization circuits. The implementation of this is shown in figure 15, utilizing the RTL viewer built into Libero.
The fMax for this design was 609.756MHz for Aclk, and 505.051MHz for Bclk (as reported by SmartTime, shown in figure 16). This indicates that the slowest path wasn't the synchronization circuit, but rather the shift register. As shown in figure 17, this indication is confirmed by the lowest slack occurring in the data path of shiftRegister[1].
The simulation is shown in figure 18.
From the simulation, we can see that 5/5 data transfers are successful. In this situation, with a slow clock generating the data and the fast clock reading the data in, it's a little harder to make a full determination of the exact accuracy, but it does make sense upon further review. Coming out of reset, the shift register would start at 0x1, then 0x2, then 0x5, then 0xA, and then back to 0x5 in a loop. In a perfect world, we would expect that a faster data sink would see this full data progression. Luckily, this synchronizer helps us achieve a more perfect world, as the data is stably transferred in a way that follows this exact progression, without any glitching or unexpected behavior. There is even a relatively small delay between the shift register and the value of Dout, only ~34ns.
Similarly to A, constraints won't help the design reach a higher accuracy, but they would allow us to fully evaluate the stability of this design.

Appendix

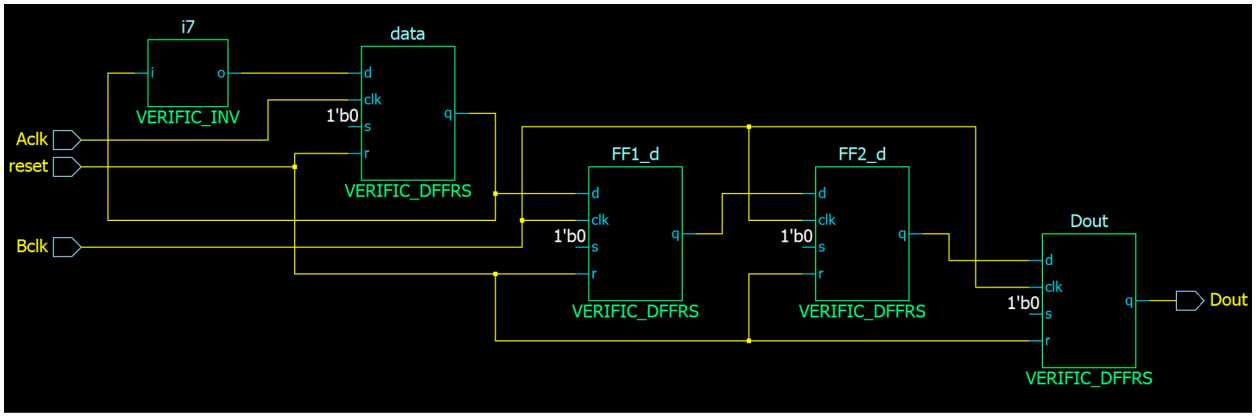

Fig 1. A screenshot of the RTL viewer in Libero showing my 3FF synchronizer.

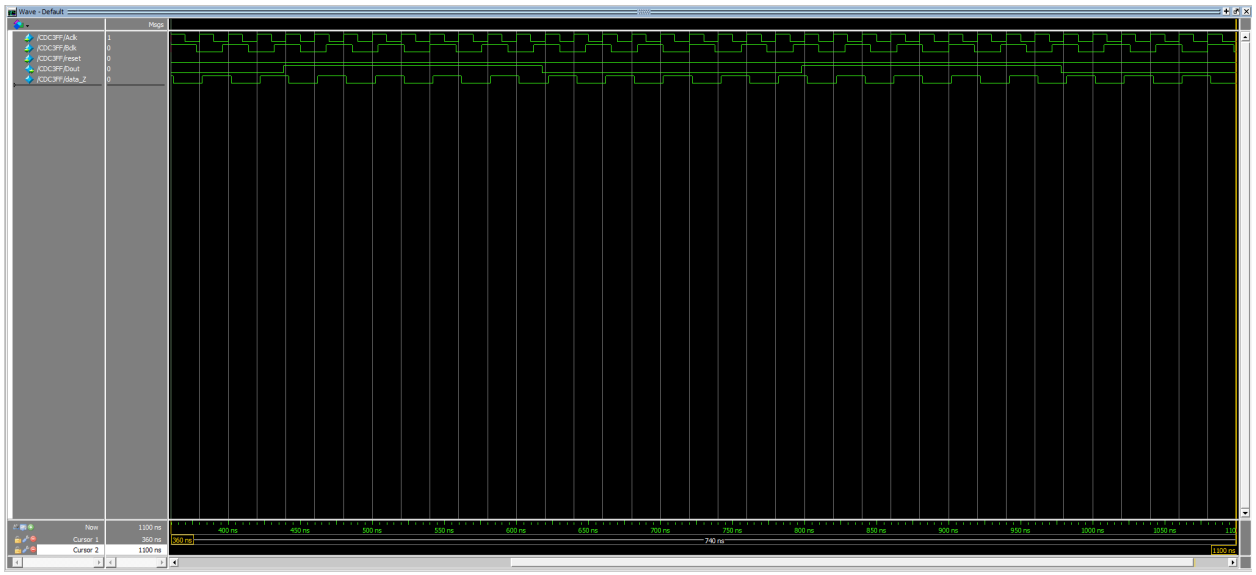

Fig 2. A screenshot of Modelsim running a gate-level simulation of the CDC3FF for 20 cycles of Bclk.

	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)	Minimum Period (ns)	Skew (ns)
1	data:CLK	data:D	0.765		3.120		0.298	1.063	0.000

Fig 3. SmartTime maximum analysis of the Aclk register to register analysis (pre-constraint).

	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)	Minimum Period (ns)	Skew (ns)
1	FF2_d:CLK	Dout_Z:D	0.565		3.416		0.298	0.891	0.028
2	FF1_d:CLK	FF2_d:D	0.568		3.419		0.298	0.883	0.017

Fig 4. SmartTime maximum analysis of the Bclk register to register analysis (pre-constraint).

	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)	Minimum Period (ns)	Skew (ns)
1	data:CLK	data:D	0.738	18.964	3.501	22.465	0.298	1.036	0.000

Fig 5. SmartTime maximum analysis of the Aclk register to register analysis (post-constraint).

	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)	Minimum Period (ns)	Skew (ns)
1	FF2_d:CLK	Dout_Z:Dn	1.123	34.839	3.956	38.795	0.187	1.262	-0.048
2	FF1_d:CLK	FF2_d:D	0.561	35.226	3.394	38.620	0.298	0.875	0.016

Fig 6. SmartTime maximum analysis of the Bclk register to register analysis (post-constraint).

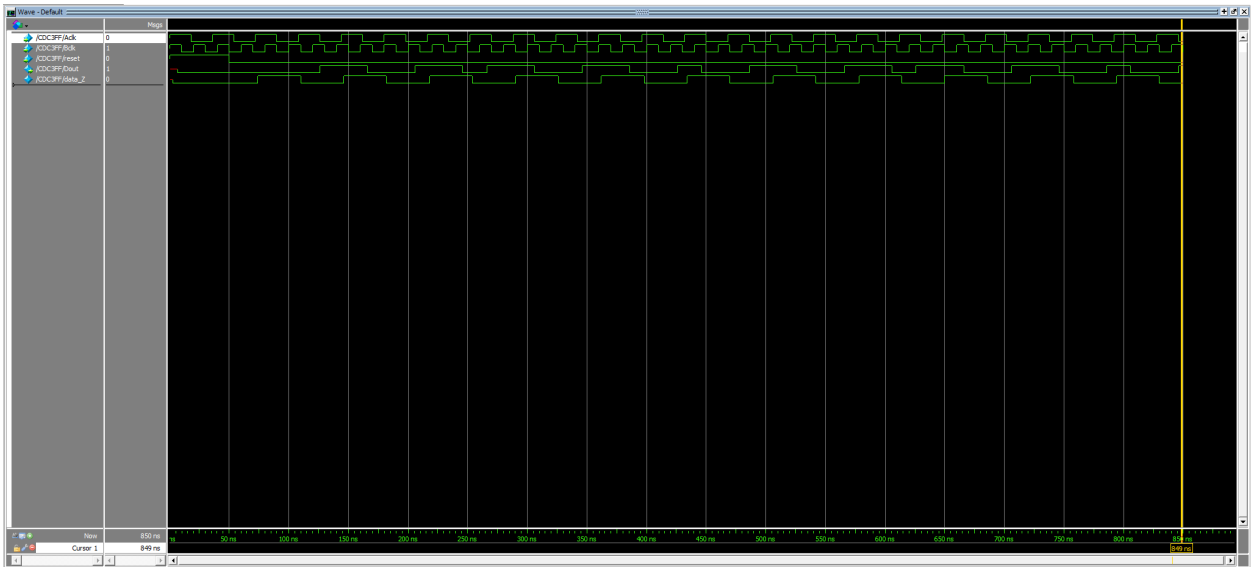

Fig 7. Modelsim screenshot showing the 3FFCDC with Bclk = 50 MHz, Aclk = 27.7MHz.

	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)	Minimum Period (ns)	Skew (ns)
1	data:CLK	data:D	0.765		3.120		0.298	1.063	0.000

Fig 8. Part B SmartTime maximum analysis of the Aclk register to register analysis (pre-constraint).

	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)	Minimum Period (ns)	Skew (ns)
1	FF2_d:CLK	Dout_Z:D	0.565		3.416		0.298	0.891	0.028
2	FF1_d:CLK	FF2_d:D	0.568		3.419		0.298	0.883	0.017

Fig 9. Part B SmartTime maximum analysis of the Bclk register to register analysis (pre-constraint).

	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)	Minimum Period (ns)	Skew (ns)
1	data:CLK	data:D	0.756	35.047	3.350	38.397	0.298	1.054	0.000

Fig 10. Part B SmartTime maximum analysis of the Aclk register to register analysis (post-constraint).

	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)	Minimum Period (ns)	Skew (ns)
1	FF2_d:CLK	Dout_Z:D	0.547	19.139	3.402	22.541	0.298	0.861	0.016
2	FF1_d:CLK	FF2_d:D	0.555	19.140	3.400	22.540	0.298	0.860	0.007

Fig 11. Part B SmartTime maximum analysis of the Bclk register to register analysis (post-constraint).

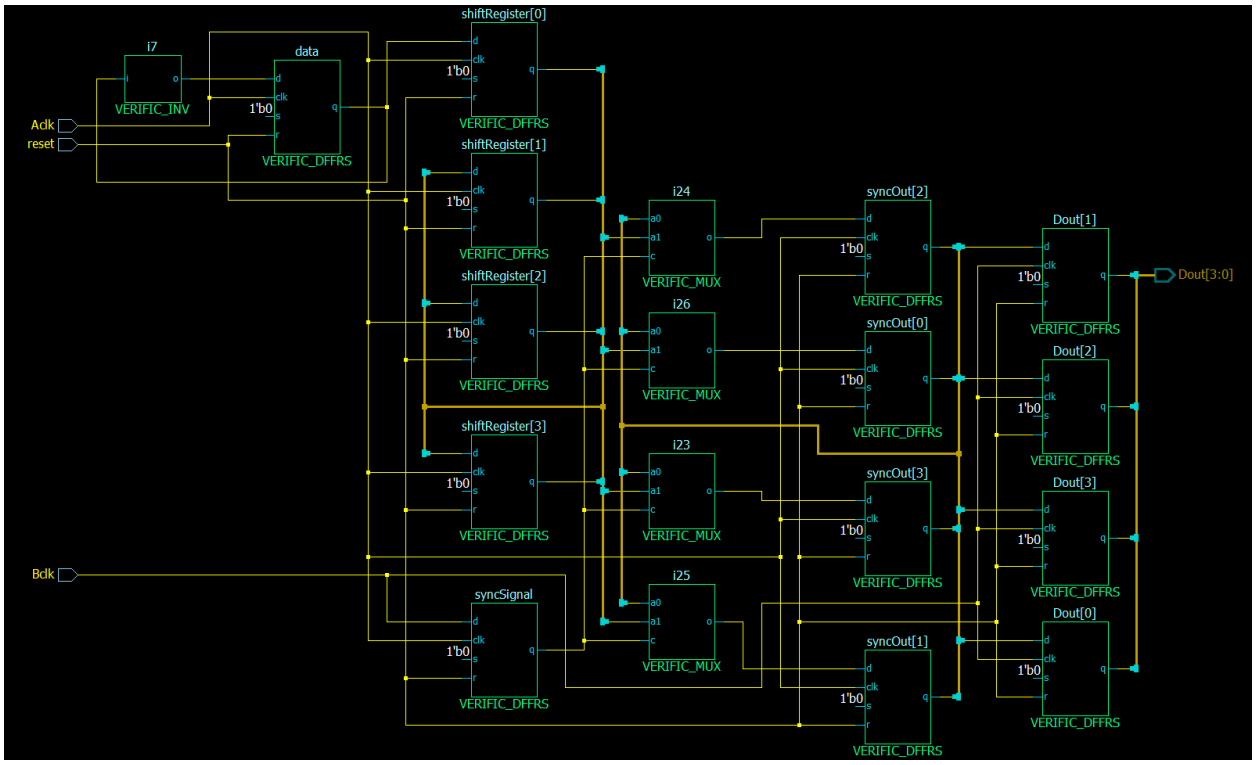

Fig 12. The completed SlowFast design in Libero's RTL viewer, with a fast Aclk and a slow Bclk.

Clock Domain	Period (ns)	Frequency (MHz)
Aclk	1.640	609.756
Bclk	1.980	505.051

Fig 13. The fMax for the SlowFast design with a fast Aclk and a slow Bclk.

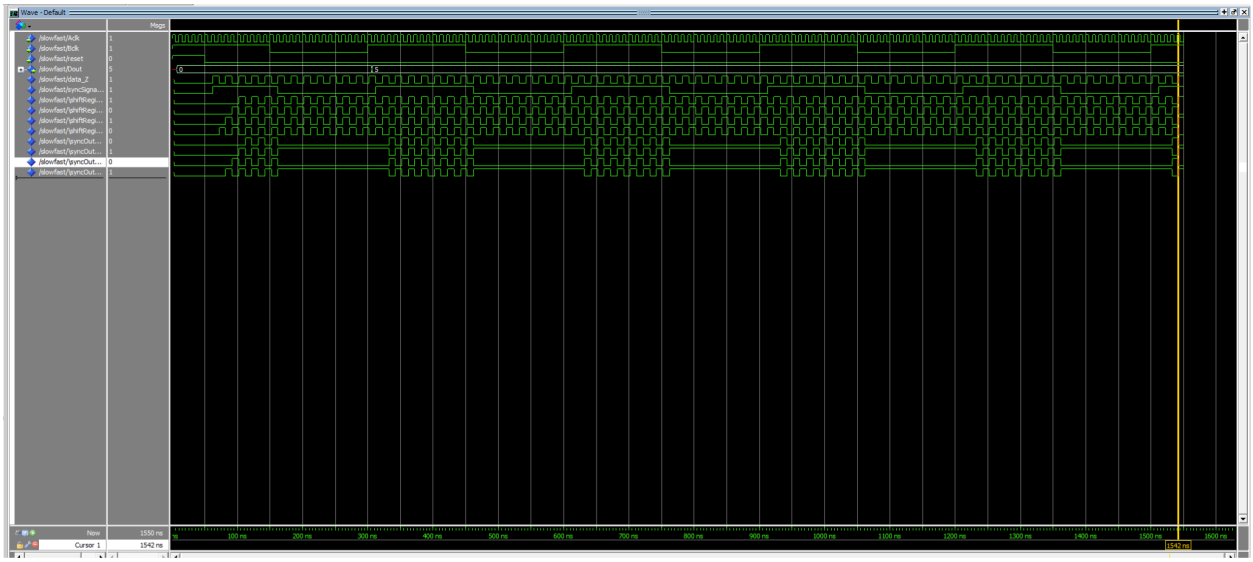

Fig 14. A ModelSim screenshot showing 5 data transfers of the SlowFast design with a fast Aclk.

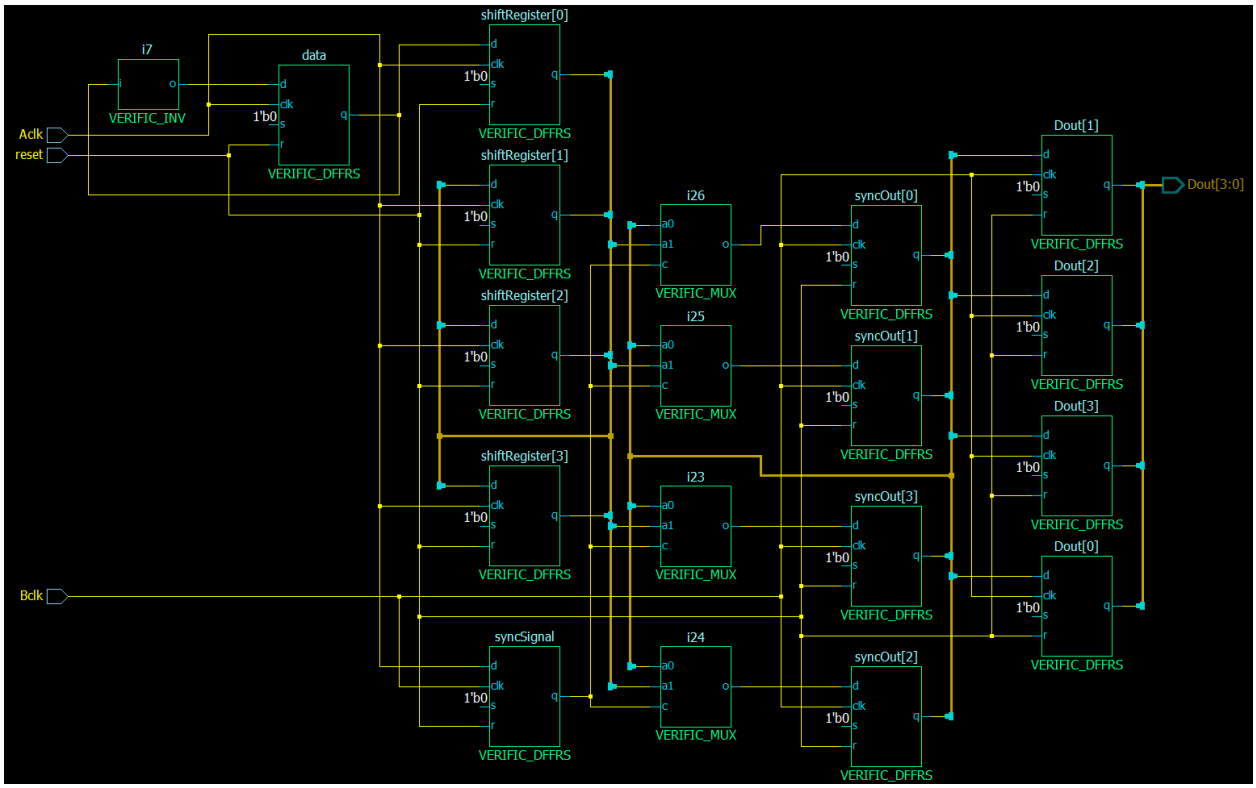

Fig 15. The completed SlowFast design in Libero's RTL viewer, with a slow Aclk and a fast Bclk.

Clock Domain	Period (ns)	Frequency (MHz)
Aclk	1.640	609.756
Bclk	1.980	505.051

Fig 16. The fMax for the SlowFast design with a slow Aclk and a fast Bclk.

	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)	Minimum Period (ns)	Skew (ns)
1	data:CLK	data:D	0.749		3.205		0.298	1.047	0.000
2	data:CLK	shiftRegister[0]:D	0.607		3.063		0.298	0.922	0.017
3	shiftRegister[0]:CLK	shiftRegister[1]:D	0.575		3.031		0.298	0.890	0.017
4	shiftRegister[2]:CLK	shiftRegister[3]:D	0.563		3.031		0.298	0.890	0.029
5	shiftRegister[1]:CLK	shiftRegister[2]:D	0.581		3.037		0.298	0.884	0.005

Fig 17. The register to register timing analysis of Aclk on the SlowFast design with a slow Aclk and a fast Bclk.

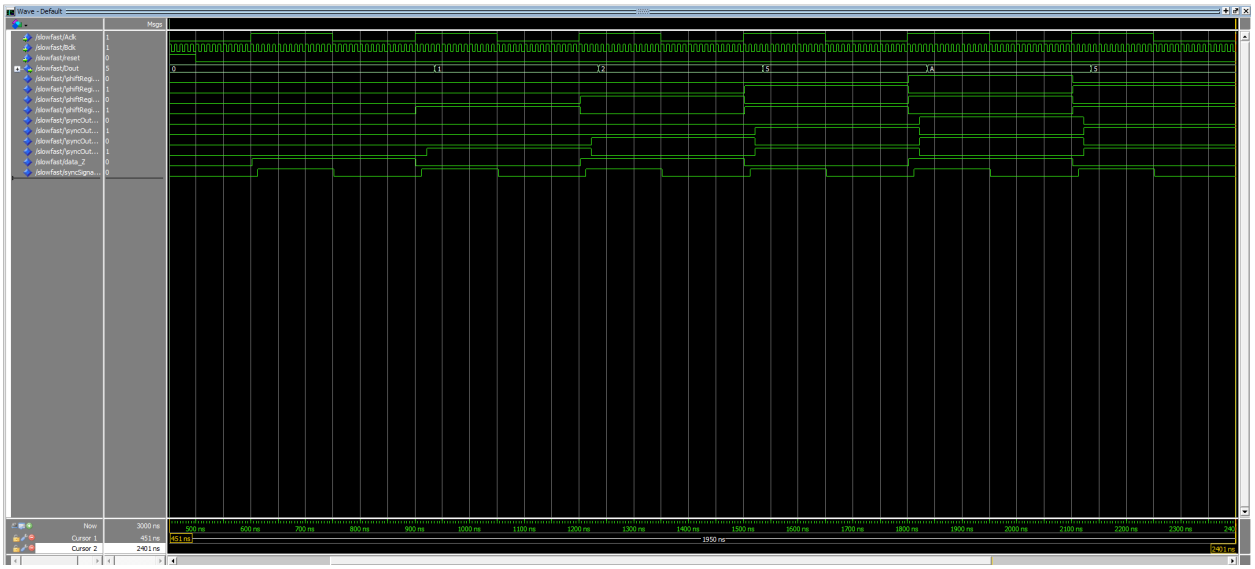

Fig 18. A ModelSim screenshot showing 5 data transfers of the SlowFast design with a slow Aclk.