CSE125 Lab3: FIFO

Due Date: 05/01/2021 11:59pm

I. Register Based FIFO

In this lab, you will design a register-based FIFO as we discussed in class. In addition, you will design a test bench to test the correct behavior of your design which tests all corner cases. Finally, you will benchmark your design in terms of Fmax and resource consumption.

Prerequisites

• Read through this entire lab assignment.

• Recap the lecture slides and notes on FIFO design

• Post questions in the forum if you are uncertain about the specification!

Design Overview

Your design should implement the register based first-in-first-out (FIFO) architecture we covered in class (Do not implement an SRAM based FIFO) and which is shown in Figure 1. The FIFO should define the following module interface:

```
module fifo #(parameter WIDTH=64, parameter DEPTH=8)(
             input clk, input res_n, input shift_in, input shift_out,
             input [WIDTH-1:0] data_in,  output full, output empty,
             output [WIDTH-1:0] data_out);
            /* Your code here */
endmodule
```

The FIFO design does not need to check for the following assertions: assert(full && !shift in) and assert(empty && !shift out). Instead, your testbench needs to make sure that these scenarios cannot happen.

Your design needs to be parameterizable, in particular, you should have a parameter to define the WIDTH of the FIFO and the depth in stages. Use parameters where possible and generate loops where required. Hint: First, develop a FIFO stage module, then connect multiple FIFO stages together to form a FIFO.

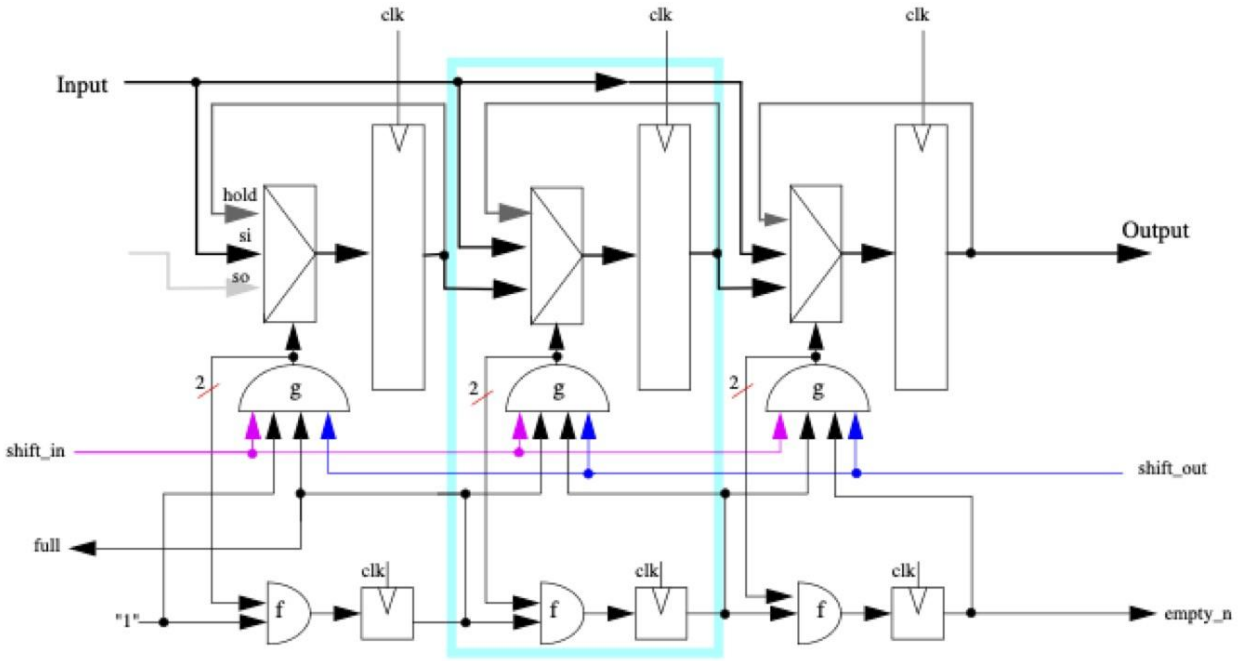

Figure 1: Register Based FIFO (only a sketch, do not assume it necessarily captures all signals/wires/logic)

Testbench

Your testbench should test all corner cases of the FIFO design, including all valid combinations of `shift_in`, `shift_out` in the case of an `empty`, `almost full` and `full` FIFO. Use $random to generate test input in a random fashion while guaranteeing that the assertions above are not violated. Read the shift-in data from a file, store it in a two-dimensional memory array in the testbench and provide it to the FIFO. Write the shift-out data into a separate file in the same format. Ensure that the FIFO is drained completely before finishing the test. Make sure that the diff of the input and output file is empty (i.e. the files have the same content).

Synthesis and Place and Route

Synthesize and Place & Route your design in the FPGA. Connect the FIFO inputs and outputs to dummy IOBs on the FPGA so the tools will not optimize them away. Make sure that the whole flow succeeds without warnings and that you can generate a bitfile.

Evaluation

Synthesize and Place & Route 5 different instances of your design by keeping a fixed WIDTH of 32 and sweep the DEPTH of the FIFO from 4,8,16,64,256.

Prepare 2 figures for your final report. One that shows the 5 FIFO configurations on the x-axis and Fmax (maximum frequency) on the y-axis. The second figure should show the 5 FIFO configurations on the x-axis and the number of required LUTs and registers on the y-axis.

Constrain the XDC/UCF file of your design to achieve the maximum achievable frequency for each of your designs.

Clocks and Clock Generation

You must generate any clocks that you use with a DCM SP module and distribute using a BUFG. In order to properly synthesize your design at the correct frequency, you need to specify the clock signal PERIOD in the XDC/UCF file. As an example,

For XDC:

create_clock -period 20.000 -name clk -waveform {0.000 10.000} [get_ports clk]

For UCF:

NET gclk PERIOD = 20000 ps;

Once this is done, Xilinx can make sure all critical paths between flip-flops satisfy this constraint. Again make sure you constrain the clock to achieve maximum frequency.

Lab Submission

Submit via Canvas the following files

a) A Verilog file named fifo.v that implements the FIFO in behavioral Verilog.
b) A testbench Verilog file directed_test.v
c) A testbench Verilog file random_test.v that implements the random stimulus testing behavior outlined above.
d) A waveform.png file for the random testbench
e) A write up that provides a high-level overview of the design and that lists the P&R results (resources and Fmax). Evaluate your design by sweeping DEPTH and WIDTH and plot your results in a diagram (e.g. Frequency vs. WIDTH & DEPTH).