FPGA Design Assignment

[redacted]

Introduction

This assignment is designed to assess your ability to design a functional and synthesizable hardware block and to acquire an understanding of your approach to problem solving. It is important that you make clear how you approached this problem and how you ultimately solved it. Please submit your functional code in Verilog (or SystemVerilog). The submitted code should be synthesizable you may also wish to design a testbench for the module. [redacted]

1 A Very Simple Switch Fabric

You will be designing a very simple network switch fabric. Your module will have multiple inputs and multiple outputs and will route data words based on a 'destination' signal associated with each input. The block contract for this module is show below.

```
/* the very simple switch fabric block contract */
module very_simple_switch
#(
    parameter DATA_WIDTH=64,
    parameter INPUT_QTY=8,
    parameter OUTPUT_QTY=8
)
(
    /* clock, reset */
    input clk,
    input reset,

    /* inputs */
    input [INPUT_QTY-1:0] data_in_valid,
    input [INPUT_QTY-1:0][DATA_WIDTH-1:0] data_in,
    input [INPUT_QTY-1:0][$clog2(OUTPUT_QTY)-1:0] data_in_destination,

    /* outputs */
    output logic [OUTPUT_QTY-1:0] data_out_valid,
    output logic [OUTPUT_QTY-1:0][DATA_WIDTH-1:0] data_out
);

/* your code here */

endmodule
```

For the purpose of this assignment, every piece of data is one single word of width DATA_WIDTH (i.e. you will not be handling "packets"). On every clock cycle, you will receive (or not receive) valid data on every input. The data_in_valid signal is active high and signifies a valid data word on the corresponding data_in bus. Each data input also has an associated value of data_in_destination. This value signifies the index of the output bus that this data is destined for.

Your module must be able to handle any combination of valid input words and destinations with the lowest possible latency. For instance, if on a given cycle a number of your inputs have valid data words destined for the same output bus, you must serialize the data on the output. Another example would be a cycle where a number of your inputs are valid, but all destined for different output buses. In this case, the data should all pass through your module in a parallel fashion.

The reset signal is active high and is synchronous to the clk signal.

2 Guidelines

2.1 Input, Output Qty

Assume that the INPUT_QTY and OUTPUT_QTY values not exceed 64.
Assume that the DATA_WIDTH value not exceed 64.

2.2 Fairness

There is no requirement that this module provides fair bandwidth to each input in the case of serialization. In the case of contention between multiple inputs, we suggest you simply give priority to the lowest indexed input.

2.3 Buffering

The module is not expected to drop data in the case of input contention. To help with this issue, you are supplied with Verilog code for a FIFO buffer (fifo.v). This FIFO has a latency of 1 cycle and the 'dout' signal will assert valid data on the same cycle as 'empty' is deasserted (showahead mode).

The only case in which the module is permitted to drop data is in the case of sustained violation of the output bandwidth.

2.4 Timeline

There is no strict timeline for completion of this assignment.

2.5 Submission

Please submit your design files as well as any testbench hdl files you have used in the design and validation of this module. [redacted]