Digital Design (CS F215) **09-09-2024**

Lab 4: Multiplexer

Lecturer: Harikrishnan N B *Student:*

READ THE FOLLOWING CAREFULLY:

Honour Code for Students: I shall be honest in my efforts and will make my parents proud. Write the oath and sign it on the assignment.
For the lab, you need a dedicated fresh notebook. Keep this notebook dedicated for solving the questions pertaining to lab.

There will be design questions in the lab, the students are supposed to theoretically show the correctness of their design function. Following that, the students are encouraged to use iverilog software to implement the same.

The lab evaluation consists of two parts.

1) Step 1 (Theory): The students are supposed to solve the design question in their dedicated lab notebook. They are supposed to get it checked and validated by TA's/ instructors. (**Deadline for theory evaluation: 10:15 AM**)

2) Step 2 (Lab): Once Step 1 is completed, they can use iverilog to implement their design. Once you have completed the design, inform the TA's/ instructors and get it verified. (**Deadline for lab file submission in local quanta: 10:40 AM**)

3) Step 3: Upload your solution to quanta.bits-goa.ac.in (local quanta). Create a zip file CampusID.zip which contains all the files.

All questions must be solved using structural modeling or gate-level modeling techniques only. The use of `assign`, `if`, or `else` statements is strictly prohibited. Only solutions implemented using structural modeling will be accepted for evaluation.

Multibit-Multiplexer

Multibit Multiplexer: Unlike a single-bit multiplexer that works on a single bit at a time, a multibit multiplexer deals with multiple bits (e.g., 4 bits, 8 bits, etc.) in parallel.

1. Consider two bit streams $A = A_3A_2A_1A_0$ and $B = B_3B_2B_1B_0$. Now construct a 2 to 1 MUX that selects between A_i (if select = 0) and B_i (if select =1). The number of MUX you have to construct is equal to number of bits in a single bit stream (for eg. A), where each MUX corresponds to the i-th location in a bitstream. Answer the following:

   (a) **Theory(2 Marks):** Consider there are four bits in each bitstream. Given this, draw the circuit diagram of the digital circuit that does the operation mentioned in question 1. The output of the MUX (`out[3:0]`) is determined by the value of the select line. If `select` = 0, then `out[i]` = A_i; if `select` = 1, then `out` = B_i for every bit of bitstream($0 <= i <= 3$).

(a) **Lab(4 Marks):** Complete the module `mux_2to1_4bit.v` in Verilog using a structural model to implement a 4-bit 2-to-1 MUX. The module takes two 4-bit inputs (`in0`, `in1`), a 1-bit select line (`select`), and outputs a 4-bit result (`out`). The output should be `in0` when `select` is 0, and `in1` when `select` is 1.

2. A multibit 4 x 1 MUX takes inputs as four bit streams $A = A_3A_2A_1A_0$, $B = B_3B_2B_1B_0$, $C = C_3C_2C_1C_0$, and $D = D_3D_2D_1D_0$ and outputs a bitstream. The *ith* bit of output bitstream is A_i(if select=00), B_i(if select=01), C_i(if select=10) and D_i(if select=11).

   (a) **Theory(2 Marks):** Using the above 2 X 1 MUX in question 1a,implement a 4 X 1 MUX with inputs as four bit streams(`in0[3:0]`, `in1[3:0]`, `in2[3:0]`, `in3[3:0]`) and two select lines (`select[1:0]`). The output of the MUX (`out[3:0]`) is determined by the values of the select lines. If `select` = 00, then `out` = `in0`; if `select` = 01, then `out` = `in1`; if `select` = 10, then `out` = `in2`; if `select` = 11, then `out` = `in3`.

   (a) **Lab(4 Marks):** Complete the module `mux_4to1_4bit.v` in Verilog using a structural model to implement a 4-bit 4-to-1 MUX. Use 4-bit 2-to-1 MUX modules to build this 4-to-1 MUX. The module should take four 4-bit inputs (`in0`, `in1`, `in2`, `in3`), two select lines (`select[1:0]`), and output a 4-bit result (`out`).

3. A multibit 8 x 1 MUX takes inputs as eight bit streams $A = A_3A_2A_1A_0$, $B = B_3B_2B_1B_0$, $C = C_3C_2C_1C_0$, $D = D_3D_2D_1D_0$, $E = E_3E_2E_1E_0$, $F = F_3F_2F_1F_0$, $G = G_3G_2G_1G_0$, and $H = H_3H_2H_1H_0$ and outputs a bitstream. The *ith* bit of the output bitstream is A_i (if select = 000), B_i (if select = 001), C_i (if select = 010), D_i (if select = 011), E_i (if select = 100), F_i (if select = 101), G_i (if select = 110), and H_i (if select = 111).

   (a) **Theory(4 marks):** Using the above 2 X 1 MUX 1a and 4 X 1 MUX 2a, implement a 8 X 1 MUX that has inputs as eight bitstreams (`in0[3:0]`, `in1[3:0]`, `in2[3:0]`, `in3[3:0]`, `in4[3:0]`, `in5[3:0]`, `in6[3:0]`, `in7[3:0]`) and three select lines (`select[2:0]`). The output of the MUX (`out[3:0]`) is determined by the values of the select lines. If `select` = 000, then `out` = `in0`; if `select` = 001, then `out` = `in1`; and so on up to `select` = 111, then `out` = `in7`.

   (a) **Lab(4 marks):** Complete the module `mux_8to1_4bit.v` in Verilog using a structural model to implement an 4-bit 8-to-1 MUX. Use 4-bit 4-to-1 MUX modules and 4-bit 2-to-1 MUX modules to build this 8-to-1 MUX. The module should take eight 8-bit inputs (`in0`, `in1`, `in2`, `in3`, `in4`, `in5`, `in6`, `in7`), three select lines (`select[2:0]`), and output an 4-bit result (`out`).

4. **Theory(4 Marks):** Draw the circuit diagram of an 8-bit modulo 8 circuit using 8-to-1 multiplexers.

   An **8-bit modulo 8 circuit** takes an 8-bit binary input (`number[7:0]`) and produces a 4-bit output (`out[3:0]`). The output is determined based on the remainder when the input number is divided by 8. The circuit maps each possible remainder (from 0 to 7) to a specific 4-bit output value according to a predefined output table.

   The output table for the circuit is as follows:

   The circuit is designed to use an 8-to-1 multiplexer to determine the output based on the value of the input modulo 8.

5. **Lab(6 Marks):** Complete the module `mod_8bit.v` in Verilog using a structural model to implement an 8-bit modulo 8 circuit.Use the table above to set the correct output for each modulo value.

Remainder (number mod 8)	Output (out[3:0])
0	10 (in decimal)
1	11 (in decimal)
2	12 (in decimal)
3	13 (in decimal)
4	3 (in decimal)
5	7 (in decimal)
6	9 (in decimal)
7	4 (in decimal)

Instructions for Lab 4

Do not use the assign operator; finish all the modules using inbuilt gates and other modules.
Make sure to **not modify any test bench**, and keep the **module name** and **input/output variable names unchanged**.

1. Complete the `mux_2to1_4bit.v` module.

   How to compile:

   ```
   iverilog -o testbench_2to1.vvp testbench_2to1.v
   vvp testbench_2to1.vvp
   ```

   How to test:

   ```
   python3 autograder.py
   ```

2. Complete the `mux_4to1_4bit.v` module using the `mux_2to1_4bit.v` module.

   How to compile:

   ```
   iverilog -o testbench_4to1.vvp testbench_4to1.v
   vvp testbench_4to1.vvp
   ```

   How to test:

   ```
   python3 autograder.py
   ```

3. Complete the `mux_8to1_4bit.v` module using the `mux_4to1_4bit.v` module and the `mux_2to1_4bit.v` module.

   How to compile:

   ```
   iverilog -o testbench_8to1.vvp testbench_8to1.v
   vvp testbench_8to1.vvp
   ```

   How to test:

```
python3 autograder.py
```

4. Complete the `mod_8bit.v` module using the `mux_8to1_4bit.v` module.

   How to compile:

   ```
   iverilog -o testbench_mod_8bit.vvp testbench_mod_8bit.v
   vvp testbench_mod_8bit.vvp
   ```

   How to test:

   ```
   python3 autograder.py
   ```