Lab 2: Counters

Submission Due Dates:
Source Code: 2020/10/06 18:30
Report: 2020/10/11 23:59

Objective

1 Getting familiar with different counter designs in Verilog.
2 Bonus challenge: sequence generator for Tower of Hanoi disks movement.

Action Items

1 4-bit Counter (60%)

A. lab2_1.v

Design a 4-bit counter between 0 and 12 with error detection. The counter is triggered by negative clock edges.

a. IO list:
   - ✓ Inputs: clk, rst, en, dir, load, data
   - ✓ Output: out (the output of the counter value)

b. **rst**: the positive-edge-triggered reset to reset the counter value (out) to 4'b0000.

c. If **en** == 1'b0: the counter holds its current output value.

d. If **en** == 1'b1:
   - ✓ If **load**== 1'b0:
     - If **dir** == 1'b1: the output value increases by 1 at every **clk** triggered
     - If **dir** == 1'b0: the output value decreases by 1 at every **clk** triggered
     - When the output reaches 4'b1100 and **dir** == 1'b1, the next output value will remain unchanged (i.e., 4'b1100).
     - When the output reaches 4'b0000 and **dir** == 1'b0, the next output value will remain unchanged (i.e., 4'b0000).
   - ✓ If **load** == 1'b1: load the input (data) to the counter. Unfortunately, some naughty users may enter an invalid value (e.g., data > 4'b1100). Your design should detect it and set the counter output (out) as 4'b1111. It indicates an error when out==4'b1111. Out should hold at 4'b1111 until the next rst signal after the error occurred.

e. You have to use the following template for your design:

```
module lab2_1 (
  input clk,
  input rst,
  input en,
  input dir,
```

```
  input load,
  input [5:0] data,
  output [3:0] out
  );
  // add your design here
endmodule
```

B. lab2_1_t.v

a. You have to create a testbench (lab2_1_t.v) by yourself to verify the design. Your testbench must test every function and boundary condition in your design. During the demo, TA will check your result/waveform with questions about how you built your testbench.

b. You have to use the following template for your design:

```
module lab2_1_t;
  reg clk, rst, en, dir, load;
  reg [5:0] data;
  wire [3:0] out;
  lab2_1 counter (.clk(clk), .rst(rst), .en(en), .dir(dir),
  .load(load),.data(data), .out(out));
  initial clk = 0;
  always #5 clk = ~clk;
  // add your testbench here
  // add more parameters if you need
endmodule
```

2 8-bit Fibonacci counter (40%)

A. lab2_2.v

Design an 8-bit Fibonacci counter which is triggered by positive clock edges.

c. A Fibonacci sequence starts from $F_1 = 1$, $F_2 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n \geq 3$.

d. The counter counts up from **1** to **233** and then counts down from **233** to **1**, where **233** is the largest Fibonacci number that can fit in 8 bits.

Hint: the sequence of a **4-bit** Fibonacci counter with the maximum value of 13:

$$1 \to 1 \to 2 \to 3 \to 5 \to 8 \to 13 \to 8 \to 5 \to 3 \to 2 \to 1 \to 1 \ldots$$

e. You have to use the following template for your design:

```
module lab2_2 (
  input clk,
  input rst,
  output [7:0] fn
);
```

```
  // add your design here
endmodule
```

f. **rst**: the positive edge triggered reset to reset the counter value (fn) to 8'b00000001.
g. **fn**: the output of the Fibonacci counter.

B. lab2_2_t.v

a. You have to create a testbench (lab2_2_t.v) by yourself to verify the design. Your testbench must test every function and boundary condition in your design. During the demo, TA will check your result/waveform with questions about how you built your testbench.
b. You have to use the following template for your design:

```
module lab2_2_t;
  reg clk, rst;
  wire [7:0] fn;
  lab2_2 counter (.clk(clk), .rst(rst), .fn(fn));
  initial clk = 0;
  always #5 clk = ~clk;
  // add your testbench here
  // add more parameters if necessary
endmodule
```

3 Bonus: lab2_3.v, lab2_3_t.v (extra 10%)

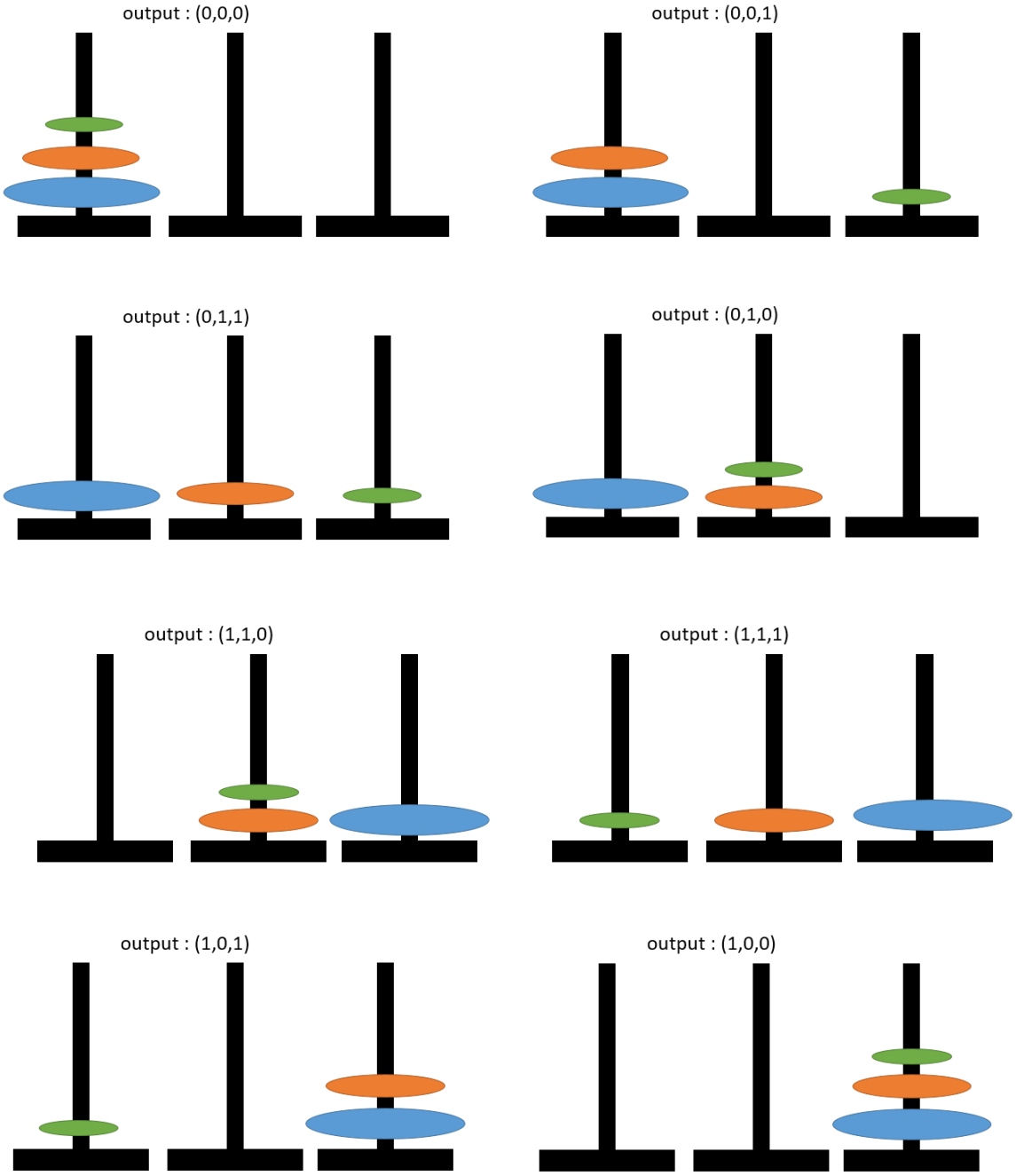
Design a positive edge triggered counter to generate the sequence that represents Tower of Hanoi disks movement, which support 8 disks. You have to create a testbench (lab2_3_t.v) by yourself to verify the design. Your testbench must test every function and boundary condition in your design. During the demo, TA will check your result/waveform with questions about how you built your testbench.

You have to use the following template for your design:

```
module lab2_3 (
  input clk,
  input rst,
  output [7:0] out);
  // add your design here
endmodule
```

```
module lab2_3_t;
  reg clk, rst;
  wire [7:0] out;
  lab2_3 counter (.clk(clk), .rst(rst), .out(out));
  initial clk = 0;
  always #5 clk = ~clk;
  // add your testbench here
  // add more parameters if you need
endmodule
```

a. This problem required you to design a module to generate a sequence showing which disk will be moved in each step. You don't have to worry about how to move the disk actually!
b. In this problem, the least-significant bit (the rightmost bit) is strictly defined as the smallest disk; the most-significant bit (the leftmost bit) is strictly defined as the largest disk.
c. Every step should not contain any redundant action, which means your module should solve the problem in $2^n - 1$ steps, where $n = Number\ of\ disks$ (not including the initial state).
d. We define switching bit as moving corresponding disks.
   For example:
   $001 \rightarrow 011$ indicates the middle disk is moved;
   $011 \rightarrow 001$ indicates the middle disk is moved as well.
e. Ex: for the Tower of Hanoi with 3 disks, to move the blue disk to the rightmost tower, we have to move the green disk first, which generates an output $(0,0,1)$ (since the green disk is the smallest disk). Then, we move the orange disk to the middle tower, generating an output $(0,1,1)$. After all, we generate an output sequence $(0,0,0) \rightarrow (0,0,1) \rightarrow (0,1,1) \rightarrow (0,1,0) \rightarrow (1,1,0) \rightarrow (1,1,1) \rightarrow (1,0,1) \rightarrow (1,0,0)$, which indicates the disk movement. Again, don't worry about how to move the disk physically.



f. **rst**: the positive edge triggered reset to reset the output to 8'b00000000.
g. **out**: the eight-bit value to indicate Tower of Hanoi disks movement
h. The disks movement sequence has to obey the rules of the Tower of Hanoi puzzles.
i. When finished, the generator should hold at the last value.
j. To get the correct simulation result you have to change the simulation runtime setting to 10000ns.
k. Hint: Wiki page of Tower of Hanoi

Attention

- ✓ You may add a **$monitor** in your testbench to show all the information of your inputs and outputs when you run the simulation.
- ✓ You may have to change your runtime (ns) in “Simulation Settings” to fit your testbench settings before you run the simulation.
- ✓ DO NOT copy-and-paste code segments from the PDF materials. Occasionally, it will also paste invisible non-ASCII characters and lead to hard-to-debug syntax errors.
- ✓ You have to hand in **lab2_1.v, lab2_1_t.v, lab2_2.v, lab2_2_t.v** (and **lab2_3.v, lab2_3_t.v**, optionally). Upload each source file directly! DO NOT hand in a compressed ZIP file!
- ✓ You should also hand in your report as **lab2_report_StudentID.pdf** (i.e., lab2_report_108456789.pdf).
- ✓ You should be able to answer questions of this lab from TA during the demo.