BCD Incrementor Circuit

Grading Sheet

Group #:________ Name(s):_____________________________________ .

Grading

- Section 4.1(a): VHDL code (50 points):_____.
  Attach code printout (with proper header and comment)
- Section 4.1 (b) (15 points):_____.
  Attach one-page schematic diagram (can be drawn by hand and then photoed).
- Section 4.2(b) RT-level simulation waveform (15 points):_____.
  Attach simulation waveform screen capture
- Section 4.3(f): post-synthesis simulation waveform (10 points): _____.
  Attach the snapshot of simulation waveform
- VHDL code format and comments (10 points):_____.

Total points: .

Experiment
BCD Incrementor Circuit

1 Purpose

To design an intermediate-sized combinational circuit

2 Reading

- Chapter 3 of *FPGA Prototyping by VHDL Examples 2nd edition.*

3 Project specification

The binary-coded-decimal (BCD) format uses 4 bits to represent 10 decimal digits. For example, 259 is represented as "0010 0101 1001" in BCD format. A BCD incrementor adds 1 to a number in BCD format. For example, after incrementing, "0010 0101 1001" (i.e., 259) becomes "0010 0110 0000" (i.e., 260). We want to design the circuit and display the results on three 7-segment LED displays. No VHDL process is allowed.

The input and output of the incrementor are

- input:
  - `b2, b1, b0`: three 4-bit inputs representing 3 BCD digits and `b2` is the most significant digit.
- output
  - `y2, y1, y0`: three 4-bit outputs representing 3 incremented BCD digits and `y2` is the most significant digit.

4 Design and simulation

4.1 BCD incrementor design

(a) Design a three-digit BCD incrementor. The entity declaration of this design is

```
entity bcd_inc is
    port(
         b2, b1, b0: in std_logic_vector(3 downto 0);
         y2, y1, y0: out std_logic_vector(3 downto 0)
    );
end bcd_inc;
```

Derive the architecture body. No VHDL process is allowed. The VHDL code must be properly documented.

(b) Derive the conceptual diagram of your VHDL code (similar to Figure 3.3 in book).

4.2 RTL ModelSim simulation

(a) Use the testbech (bcd_inc_tb.vhd) to simulate your VHDL code. The testbench includes the test patterns of 000, 123, 678, 679, 699, and 999.

(b) Develop a proper "layout" and "format" for the simulated waveform. To get full credits, the input and output signals should be arranged as six 4-bit bus and represented in proper format so that the simulation result can be easily understood. Do a screen capture of the simulated result.

4.3 Post-synthesis ModelSim simulation

(a) Perform compiling (synthesis/placement and routing) and obtain the .vho file.

(b) Follow the format guideline to add a header to the file and verify the "structure" architecture body is generated.

(c) Revise the testbech to use the "structure" architecture for uut.

(d) Perform post-synthesis simulation

(e) Develop a proper "layout" and "format" for the simulated waveform. To get full credits, the input and output signals should be arranged as six 4-bit bus and represented in proper format so that the simulation result can be easily understood. Do a screen capture of the simulated result.

(f) Since the waveform is the same as the RT-level simulation, the snapshot should include the expanded uut on the left panel. No point will be given if the uut unit is not expanded in the snapshot.

5 (Optional) Implementation and testing of the physical circuit

5.1 Testing with discrete LEDs

(a) Develop a top-level wrapping VHDL code following the discussion in Section 1.4. Make connection as follows:

- b0: sw[3..0]
- b1: sw[7..4]
- b2: "1001"
- y0: led[3..0]
- y1: led[7..4]
- y2: unconnected ("open" in VHDL)

(b) Synthesize and implement the design.

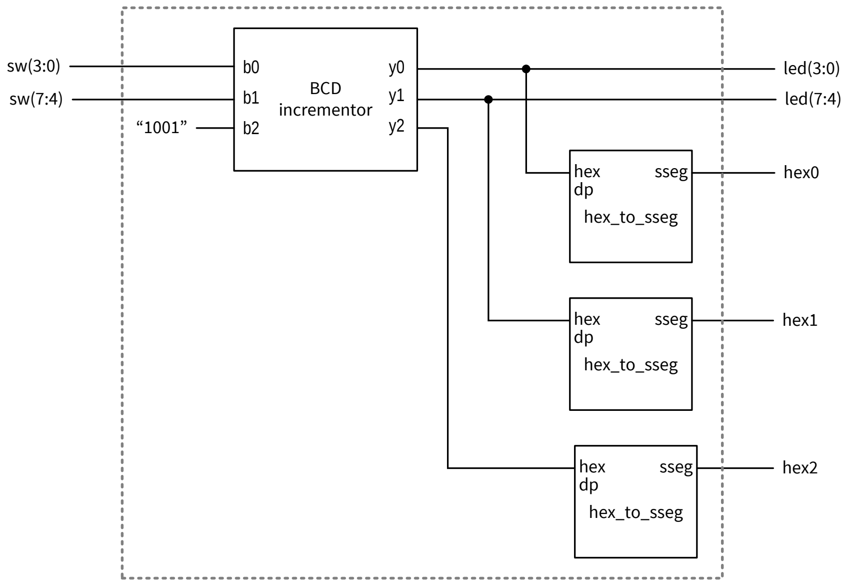
(c) Download the configuration file to FPGA board.

(d) Use the switches and LEDs to verify the operation of physical circuit.

5.2 Testing with 4-digit 7-segment LED display

(a) Read the hex-to-seven-segment decoder design/code.

(b) In addition to discrete LEDs, we want to display the 3-digit result in 7-segment LED display as well. The top-level block diagram is shown below. Develop a top-level VHDL code.



(c) Synthesize and implement the design.

(d) Download the configuration file to FPGA board.

(e) Verify the operation of physical circuit.