Zero-counting Circuit

Signature and Grading Sheet

Group #:_____ Name(s):_____ .

Grading

- Section 4.1(a)(b): pseudo algorithm and registers (10 points): .
  Attach pseudo algorithm and explanation use of registers
- Section 4.1(c): ASMD chart (20 points):_____.
  Attach detailed ASMD chart
- Section 4.1(d): VHDL code (30 points):_____.
  Attach code
- Section 4.2(b) RT-level simulation waveform (20 points):_____.
  Attach simulation waveform screen captures.
- Section 4.3(e): post-synthesis simulation waveform (10 points): _____.
  Attach simulation waveform screen captures.
- VHDL code format and comments (10 points):_____.

Total points: .

Experiment
Zero-counting Circuit

1 Purpose

To use FSMD methodology to design and implement an intermediate-sized digital circuit.

2 Reading

Chapter 6 of *FPGA Prototyping by VHDL Examples 2nd edition.*

3 Project specification

The purpose of this project is to construct a circuit that counts number of 0's of an 8-bit input word. For example, the output returns "0011" if the input is "11001110". In addition to the clock and reset signals, the input signals of this circuit are

- start: one-bit command to initialize the counting
- a: 8-bit input data word

The circuit outputs include the following:

- ready: one-bit status indicating that the circuit is ready
- count: 4-bit result showing the number of 0's in a.

The design must be synchronous or 50% will be deducted.

4 Design Procedures

4.1 FSMD design

(a) Derive a pseudo algorithm.

(b) Determine the registers needed.

(c) Derive the detailed ASMD chart according to the algorithm.

(d) Derive VHDL code according to the ASMD chart. The entity declaration of this design is

```
entity fsmd_0count is
   port(
      clk, reset: in std_logic;
      start: in std_logic;
      a: in std_logic_vector(7 downto 0);
      ready: out std_logic;
      count: out std_logic_vector(3 downto 0)
   );
end fsmd_0count;
```

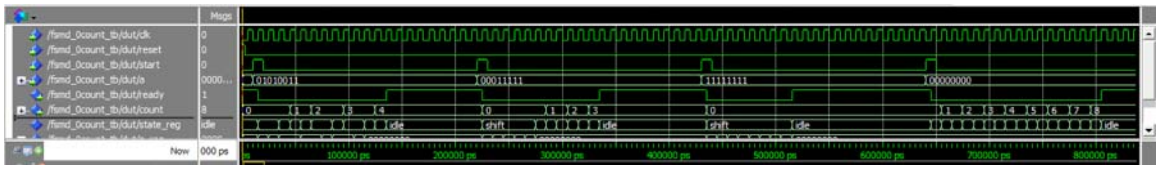
Derive the architecture body.

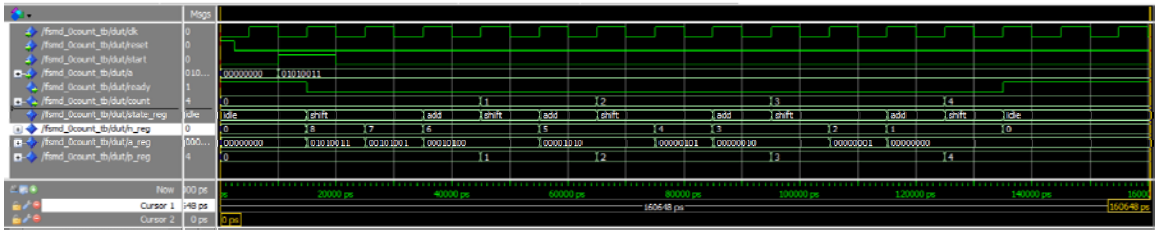
4.2 RT-level ModelSim simulation

(a) Use the testbench (fsmd_0count_tb.vhd) to simulate your VHDL code.

(b) Perform two screen captures. The first one should show all 4 test patterns and the results. The second one should expand the operation of one test pattern to make state and internal register values visible. Use proper number formats (unsigned or binary) for these signals.

(c) Develop a proper "layout" and "format" for the simulated waveform. <u>To get full credits, the input and output signals should be properly arranged and represented in a proper format so that the simulation result can be easily understood</u>.





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 testbench to use the "structure" architecture for uut.

(d) Perform post-synthesis simulation

(e) Do screen capture(s) 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. <u>No point will be given if the uut unit is not expanded in the snapshot.</u>