Enhanced Priority Encoder

Grading Sheet

Group #:_____ Name(s):_____ .

Grading

- Section 4.1(b): VHDL code (50 points):_____.
  Attach code printout (with proper header and comment)
- Section 4.2(b) RT-level simulation waveform (20 points):_____.
  Attach simulation waveform screen capture
- Section 4.3(f): post-synthesis simulation waveform (20 points): _____.
  Attach the snapshot of simulation waveform
- VHDL code format and comments (10 points):_____.

Total points: .

Experiment
Enhanced Priority Encoder

1 Purpose

Use VHDL routing constructs to design a special priority encoder.

2 Reading

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

3 Project specification

An enhanced-priority encoder returns the codes of the highest and second-highest priority requests. The input is the 10-bit `r` signal, in which the `r(9)` has the highest priority and `r(0)` has the lowest priority. The outputs are `fst` and `snd`, which are the 4-bit binary codes of the highest and second-highest priority requests, respectively. The input and output signals are

- input:
  - `r`: 10-bit input request
- output
  - `fst`: 4-bit output, the binary code of the highest priority request
  - `snd`: 4-bit output, the binary code of the second-highest priority request

An output becomes "1111" when there is no active request.

4 Design Procedures

4.1 Encoder circuit

(a) A "normal" 10-to-4 priority encoder outputs the code of the highest priority request (can be coded by one conditional signal assignment statement). Use it as the basic building block to construct the enhanced priority encoder. Derive a top-level conceptual diagram of the enhanced priority encoder. The diagram should contain 3 to 6 blocks.

(b) Derive VHDL code according to the top-level conceptual diagram. The entity declaration of this design is:

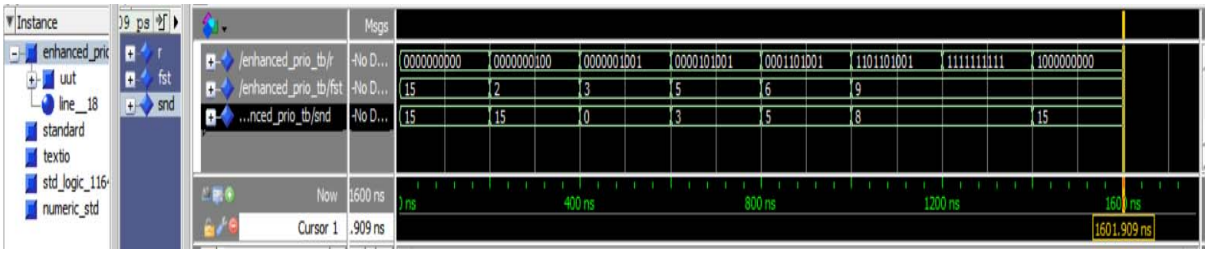
```
entity enhanced_prio is
   port(
      r: in std_logic_vector(9 downto 0);
      fst, snd: out std_logic_vector(3 downto 0)
    );
end enhanced_prio;
```

Use `rtl_arch` for the architecture name (the name is used in the testbench).

4.2 RT-level ModelSim simulation

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

(b) Develop a proper "layout" and "format" for the simulated waveform. To get full credits, the input and output signals should be properly arranged and represented in the proper format (`r` in binary and `fst` and `snd` in unsigned decimal) so that the simulation result can be easily understood, as shown below. 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) Verify the "structure" architecture body is generated.

(c) Revise the testbench (enhanced_prio_tb.vhd) 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 properly arranged and represented in the proper format (`r` in binary and `fst` and `snd` in unsigned decimal) so that the simulation result can be easily understood.

(f) Use Windows "snip tool" to capture a snapshot of the simulation result. 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 Implementation and testing of the physical circuit

5.1 (Optional) Testing with discrete LEDs

(a) Develop a top-level wrapping VHDL code to map the signals to the DE10-lite board. Make connection as follows:

- `r: sw[9..0]`
- `snd: led[3..0]`
- `fst: led[7..4]`

(b) Synthesize and implement the design.

(c) Download the configuration file to the FPGA board.

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