CO200 - Computer Organization and Architecture

A1 - Hardware Modules in Verilog

Points to Note

- Design and test each of the listed hardware modules in the Verilog HDL.
- In the comments section of each Verilog code, include a short description of the program, your name, roll number, date of writing the program and other information you deem relevant.
- (a) Report: Every question can have a corresponding answer containing block diagrams/ microarchitecture, brief explanation, other relevant info. (b) Auxilary files to submit: Per question, include the following files along with the report: Verilog code, the testbench including the monitor and stimulus, execution screenshots, VCD dump, gtkwave screenshots. (c) Archive: Organize your submission into directories per question.
- This is a team assignment. One submission per team.
- (a) Submission is due October, 15, 9AM. Pack your report, code, screenshots and other files in an archive and mail to `co200.nitk@gmail.com`.
- Install iVerilog (http://iverilog.icarus.com/) and gtkwave. Ubuntu: `$ sudo apt-get install iverilog gtkwave`.
- Verilog Tutorials: http://vol.verilog.com/, http://www.asic-world.com/verilog/veritut.html, https://www.nandland.com/verilog/
- The DDS textbook *Digital Design - With an Introduction to the Verilog HDL*, 5e, Mano & Ciletti, Pearson, has some excellent Verilog introduction in Sections - 3.9, 4.12, 5.6, and 6.6.

Modules

1. **Hello World Program**. A hello world message should be printed out by a module periodically. The module should also print the current timestamp (in clock cycles). Module terminates after 100 clock cycles.
2. **Encoders and Decoders**. Implement a combinational 4-to-16 decoder and a 16-to-4 encoder.
3. **Half Adder**. Implement a combinational half adder (HA).
4. **Full Adder**. Using the HA module from the previous question, Implement a combinational full adder (FA).
5. **n-bit Adder/Subtractor**. Using the FA module from the previous question, build an n-bit Adder/Subtractor module. Value of 'n' should be configurable. Possible n values are 4,8,...,256.
6. **Single bit Sequential Memory Element**. Implement 3 versions of the 1 bit D-Flip Flop. All the three versions should work with the same Testbench code.
   (a) Use the Master-Slave D-Latch block diagram shown in Fig. 5.9, Page 198, Digital Design, 5e, Mano and Ciletti.
   (b) Code the gate-level D-FF shown in Fig. 5.10, Page 199, Digital Design, 5e, Mano and Ciletti.
   (c) Write Behavioral code for the D-FF
7. **Register**. Use the memory cell designed above to implement a 32 bit register (instantiate the 1-bit D-FF 32 times). The data input and data output are now 32 bits wide (instead of 1-bit in the case of the memory cell). The reset signal is still 1-bit signal as in the case of the memory cell. Perform repetitive Read-Write operations on the Register.