Integrated Circuit Design
Homework #4
Verilog-HDL

Deadline: (1) Nov 29.

Data preparation

Unpack hw4.tar with the following command

```
tar   -xvf   hw4.tar
```

(1) 4-Bit Full Adder (50%)

Use Verilog-HDL to design the ***netlist*** of 4-bit full adder, whose circuit is shown below, with basic logic gates. Then simulate this circuit with given test bench file and the Verilog simulator to verify your design.

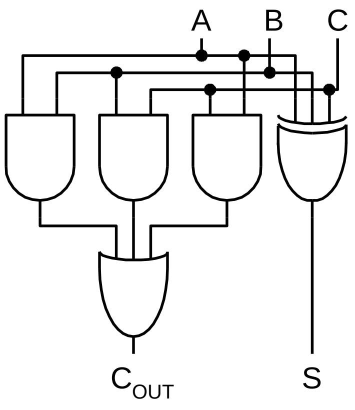


Fig. 1. 1-bit full adder.

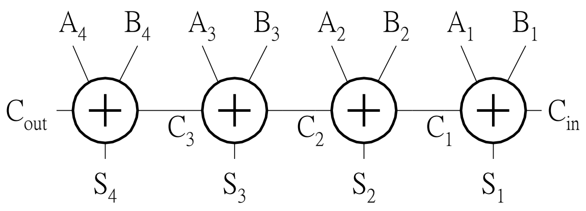


Fig. 2. 4-bit adder, which composed from four 1-bit full adders.

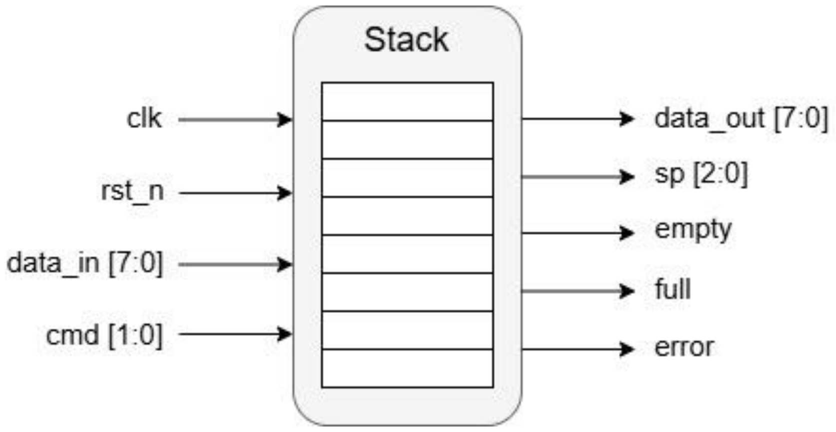
Hint:

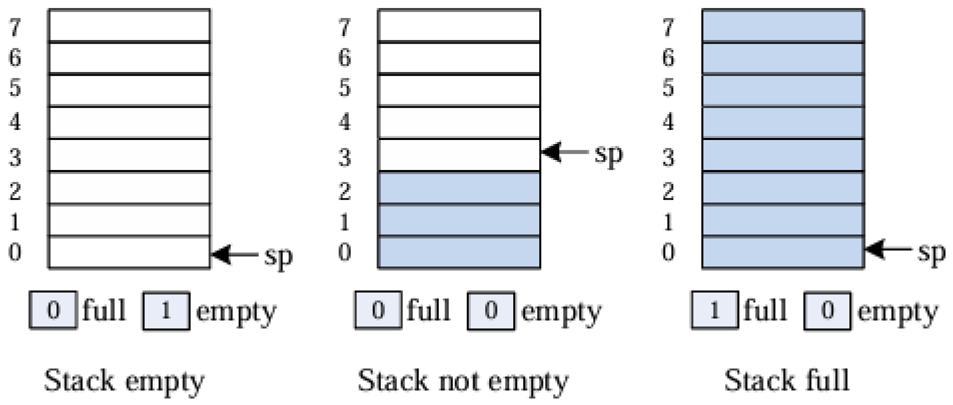
First of all, write 1-bit full adder as shown in Fig. 1 as module FA_1bit(S, Cout, A, B,

Cin). Then instance this module to design the 4-bit adder as shown in Fig. 2.
Please read the README file in the folder p1 to get more information.

(2) Stack Module Design (50%)

Use Verilog-HDL to design the ***RTL-Level*** stack module as below. The stack consists of a memory module of 8X8, a 3-bit stack pointer “sp”, and three stack flags, “full”, “empty” and “error”. “sp” always points to the next available entry for push operations. The command will be given by a 2-bit “cmd”. For push operation, your stack module will store 8-bit input data “data_in” into your memory module. For pop operation, your stack module will output 8-bit “data_out” to the testbed. Note that the output flags “full” is set to high when your stack is full, “empty” is set to high when your stack is empty, and “error” is set to high after receiving invalid push or pop operations.





Input “cmd” and “data_in” will be given by our testbed at the negative edges of the clock. The function of encoded “cmd” is as follow:

cmd	00	01	10	11
operation	No operation	clear	push	pop

Note that the input data will be set to 0 except the push operation.

Please update your outputs (“sp”, “data_out”, “empty”, “full”, “error”) at the next positive edge of the clock after receiving “data_in” and “cmd” (i.e. half cycle for you to update your internal stack status and outputs). Testbed will sample and compare your outputs with the golden data at the next negative edge.

Please set your output data to 0 except the pop operation.

Please use the positive triggered D-Flip Flops with asynchronous negative reset to design your sequential circuits.

Please read the README file in the folder p2 to get more information.

Submission

1. Create a folder named studentID_hw4 and follow the hierarchy below.

```
studentID_hw4/
    p1/FA.v
    p2/stack.v
```

Note: Use lowercase for all the letters in studentID. (e.g. b12901xxx_hw4)

2. Pack the folder studentID_hw4 into a tar file named studentID_hw4_vk.tar (k is the number of version, k =1,2,…). TA will only check the last version.

```
tar   -cvf   studentID_hw4_vk.tar   studentID_hw4
```

Note:

1. Use lowercase for all the letters in studentID. (e.g. b12901xxx_hw4_vk.tar)
2. Pack the folder on IC Design LAB server to avoid OS related problems.

3. Submit to NTU Cool.

Grading Policy

1. TA will run your code with the format of command in the README file in the folder. Make sure to run the command with no error message on IC Design LAB server.
2. In both problem 1 and problem 2, you will get 70 points if you pass all the public patterns. You will get the rest 30 points if you pass all the hidden patterns.

3. Run the following command to check your design with no latch in problem 2 (stack module design)

   First run the command dc_shell to get into design_compiler

   ```
   $ dc_shell
   ```

   Then run the command read_verilog to check your design

   ```
   dc_shell>    read_verilog    stack.v
   ```

   Please make sure that your file is read successfully and there is no any latch in your design, otherwise you will get 0 point in problem 2. If you have any concern in this part, please feel free to ask on the NTU Cool [HW4] Discussion.

   Use command exit to quit the dc_shell

   ```
   dc_shell>    exit
   ```

4. Lose 5 points for any incorrect naming or format.

5. No late submission.

6. No plagiarism