CSCE 230 – Lab 9: Register File and ALU

Due: 11:59PM, Nov 7 (Tuesday) (two-week lab)

Objectives

- Task 1: Build a register file and understand how it works.
- Task 2: Build an ALU and understand how it works.

Useful References on Canvas

- Lecture notes for Chapter 5

Lab Task 1

In this task, we will design a register file with a total of 16 registers (registers 0, 1, 2, .., 15), each with 4 bits. We will use only ModelSim to check the correctness of your VHDL design, as the DE10 Lite board testing involves too many signals (bits) to test using limited number of slider switches and push buttons. Also, this register file will not be used in our project.

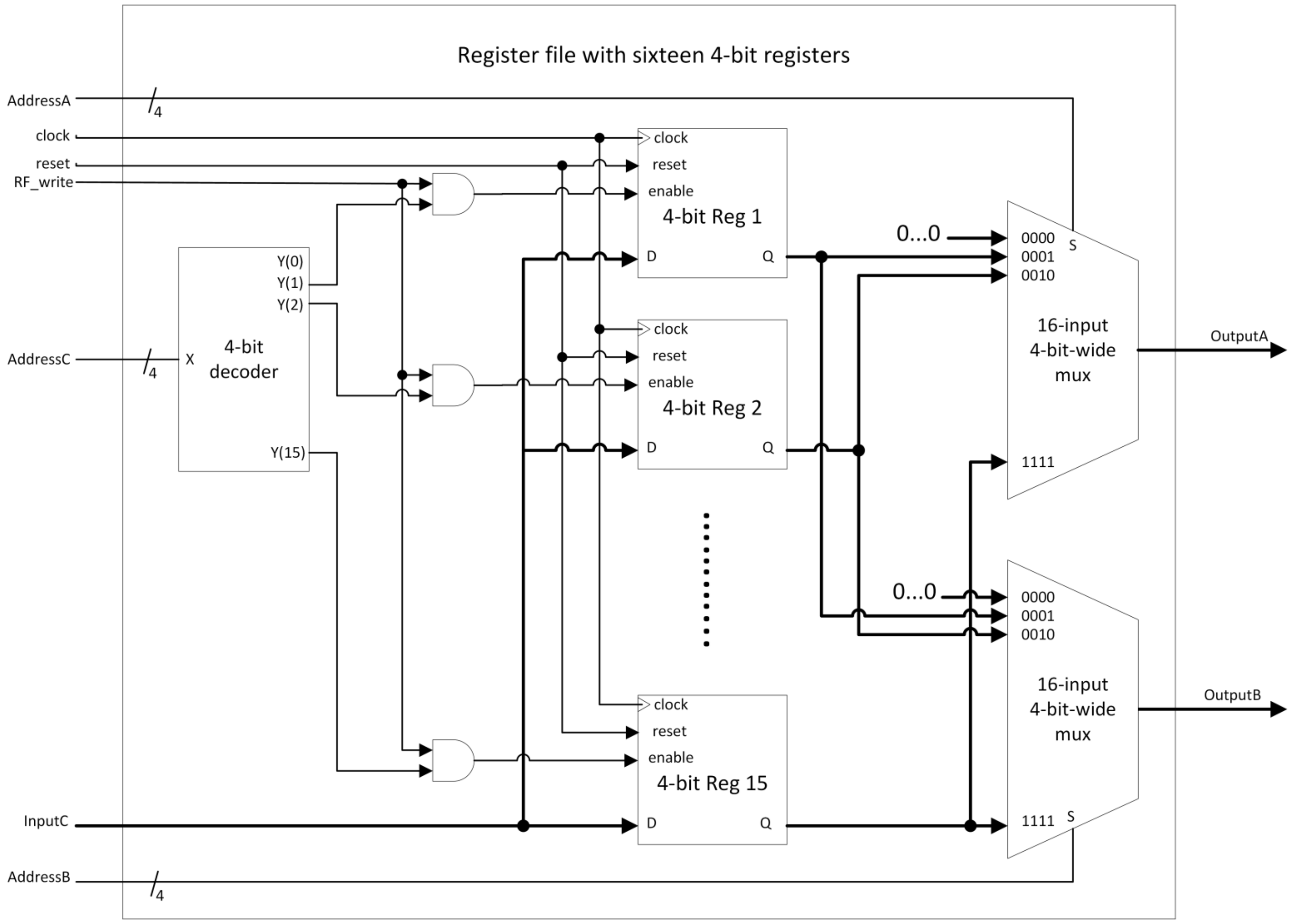
Step 1 – VHDL Design

- Please create a new Quartus project.
  - Top-level entity is "RegisterFile16by4Bit".
  - Device family is "**Max 10**" (same as before, just for your convenience).
  - Available device is "**10M50DAF484C7G**" (same as before, just for your convenience).
- Please download the following VHDL design files from Canvas, and then add them to your project (Quartus menu "Project", then "Add/Remove Files in Project").
  - Reg1Bit.vhd: A 1-bit register with reset and enable.
  - Reg4Bit.vhd: A 4-bit register with reset and enable.
  - Decoder4Bit.vhd: A decoder that changes the input from 4-bit binary encoding to 16-bit one-hot encoding.
  - Mux16Input4Bit.vhd: A multiplexer that selects one data input from a total of 16 data inputs, each with 4 bits.
- Please design your register file according to the following requirements
  - You must use the following VHDL entity definition; otherwise the simulation script will not work.

```
entity RegisterFile16by4Bit is
      port(
            clock, reset, RF_write :in std_logic;
            AddressA, AddressB, AddressC :in std_logic_vector(3 downto 0);
            InputC :in std_logic_vector(3 downto 0);
            OutputA, OutputB :out std_logic_vector(3 downto 0) );
end RegisterFile16by4Bit;
```

  - Register 0 is always zero and read-only. Do not use any DFF to implement register 0.

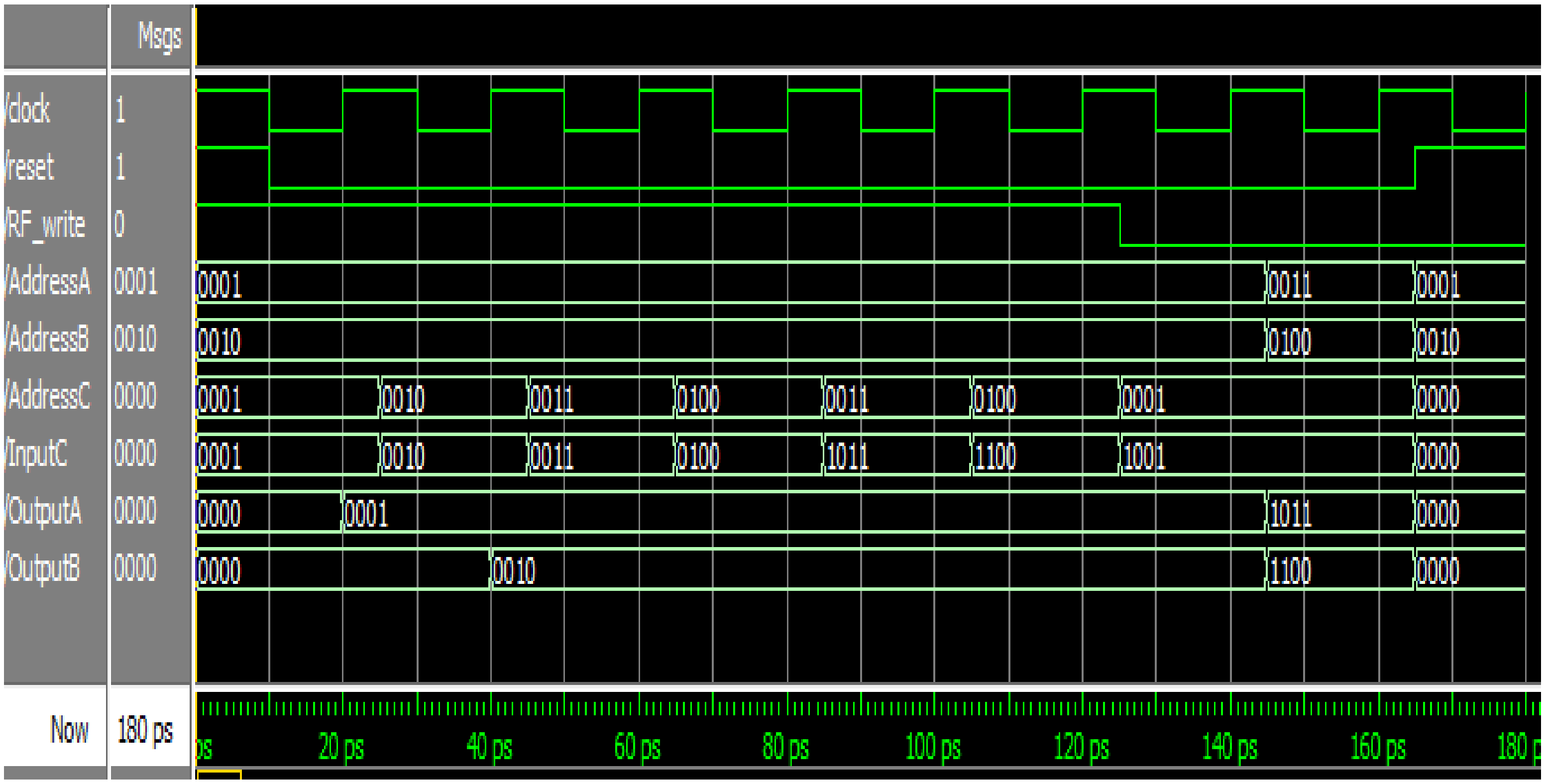
- You must use the following components to design your register file: Reg4Bit, Decoder4Bit, and Mux16Input4Bit, according to the diagram below.



Step 2 – ModelSim Simulation

- Please download simulation script task.do from Canvas, and then add it to your project (Quartus menu “Project”, then “Add/Remove Files in Project”).
- Compile your project and then run Modelsim simulation
- If you see error messages like “Use of non globally static actual (infix expression) of formal. This is caused by attempting to add an expression to a implementation port map. This requires VHDL 2008 instead of VHDL 1993”, you can fix it by the following steps.
  - close ModelSim
  - choose menu “Assignment” of Quartus
  - select “Setting” to open the Setting window

  - click “VHDL input” on the left side of the Setting window, and then select “VHDL 2008”.
  - re-compile your project and then re-run ModelSim
- If your register file is designed correctly, you should get the following waves.
  - Please carefully check the values of outputA and outputB



- Answer the questions on the Canvas
  - Hint: To find the total number of 1-bit DFFs in your register file, you can
    - either directly analyze your VHDL design,
    - or you can use “Technology Map Viewer” to show the generated circuit and then double-click on each component until you see these DFFs (a lot though).
  - Hint: To find the value of a register at a time, you can
    - either directly read and analyze simulation script task.do,
    - or you can add debugging signals to the VHDL design files and simulation script in order to show the value of each register
  - Requirement for the screenshot of your waveform.
    - Your screenshot must clearly show the signal names, signal values, and simulation times from 0 to 200ps (the above screenshot is a good example).

Lab Task 2

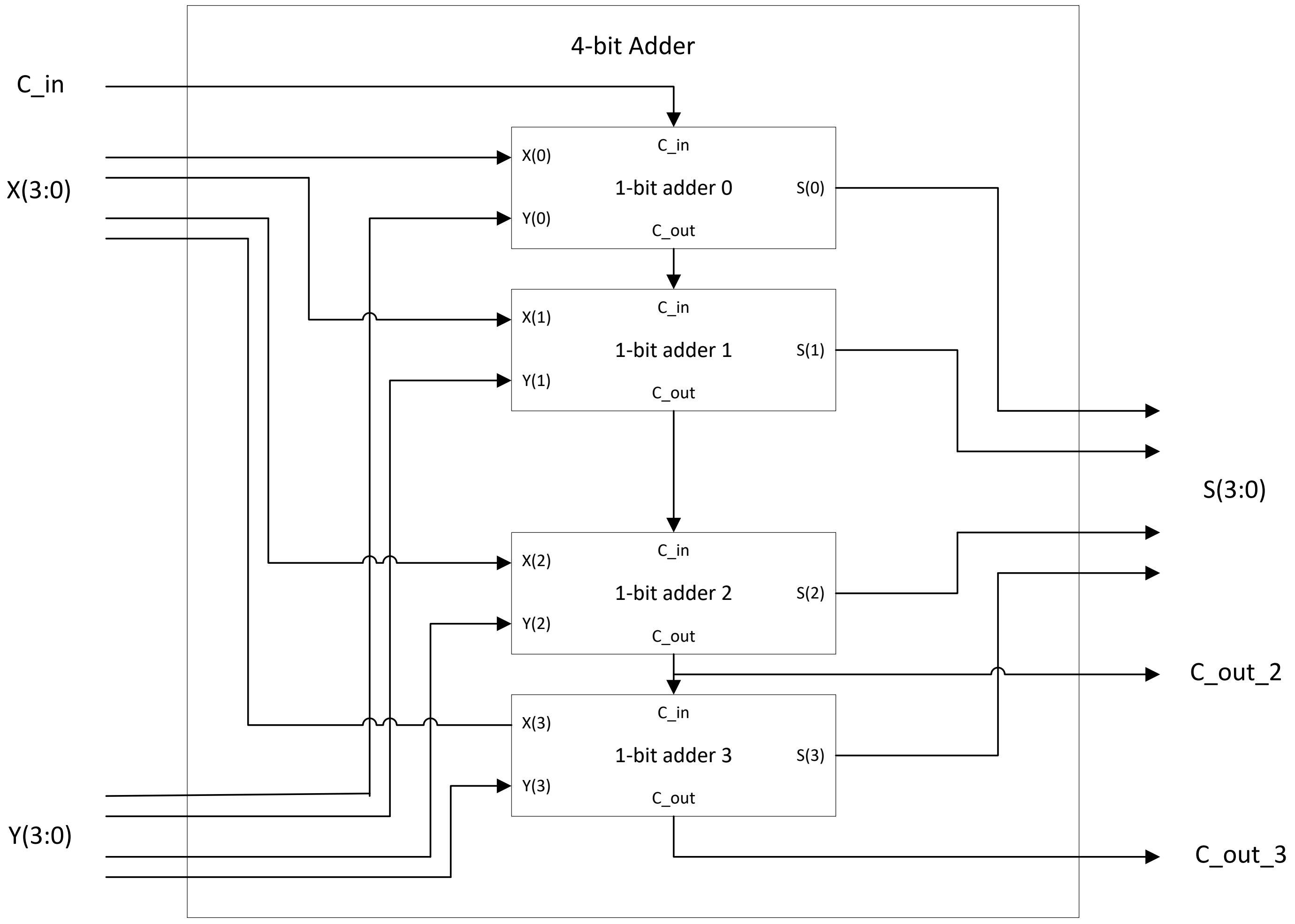
In this task, we will design a 4-bit ALU that performs arithmetic and logic operations on 4-bit binary numbers. Also, this ALU will not be used in our project.

Step 1 – VHDL Design

- Please create a new Quartus project.
  - Top-level entity is "ALU".
- Please download the following VHDL design files from Canvas, and then add them to your project (Quartus menu "Project", then "Add/Remove Files in Project").
  - Adder1Bit.vhd: A 1-bit adder
  - Mux2Input4Bit.vhd: A multiplexer that selects one data input from a total of 2 data inputs, each with 4 bits.
  - Mux4Input4Bit.vhd: A multiplexer that selects one data input from a total of 4 data inputs, each with 4 bits.
- Please first design a 4-bit adder according to the following requirements
  - You must use the following VHDL entity definition.

```
entity Adder4Bit is
       port(
              X, Y   : in std_logic_vector(3 downto 0);
              C_in   : in std_logic;
              S      : out std_logic_vector(3 downto 0);
              C_out_2, C_out_3 : out std_logic);
end Adder4Bit;
```

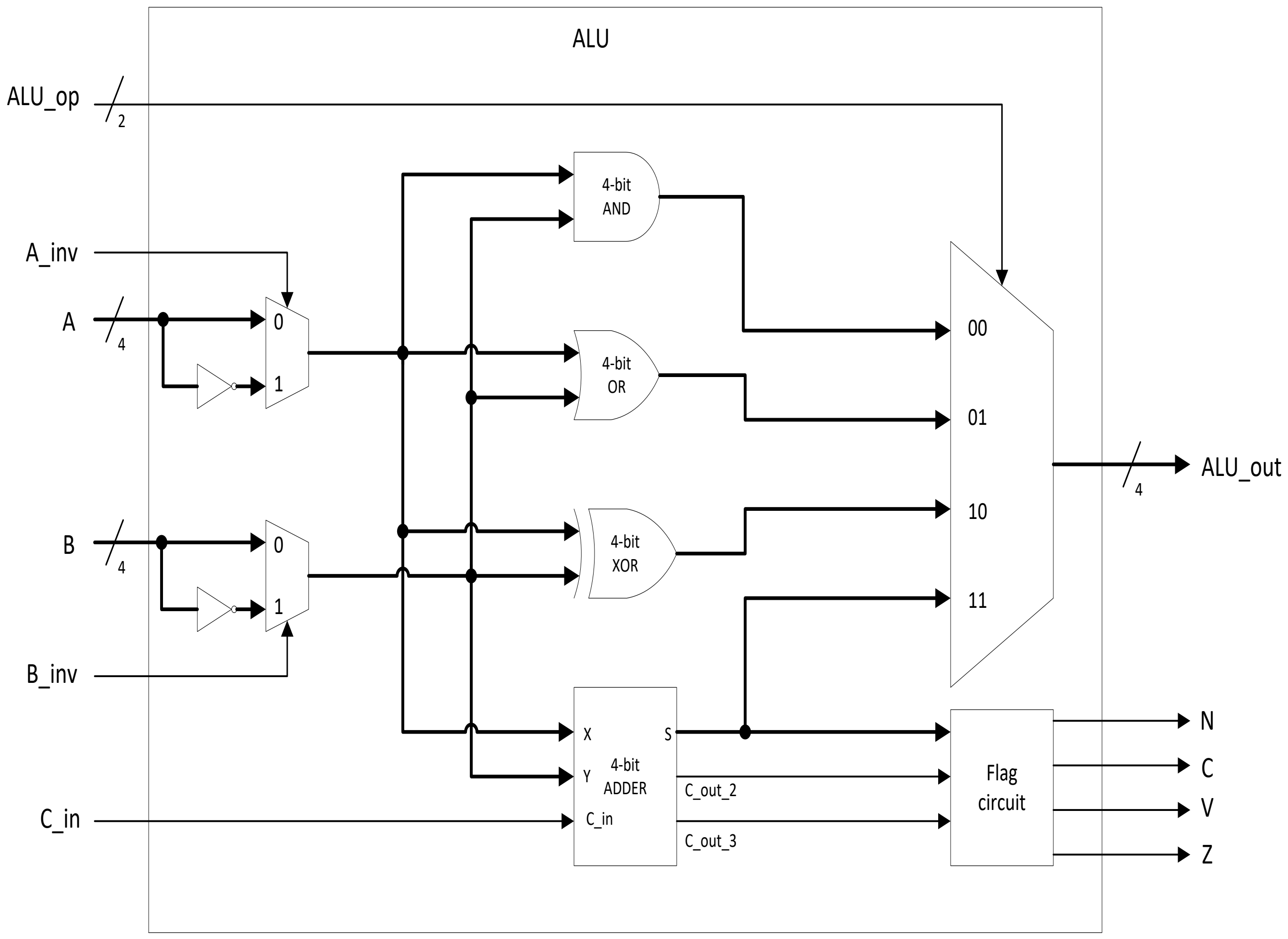
  - You may use Adder1Bit as components to design your Adder4bit according to the diagram below, or you can directly design Adder4Bit without using any components.



- Then please design the ALU according to the following requirements
  - You must use the following VHDL entity definition.

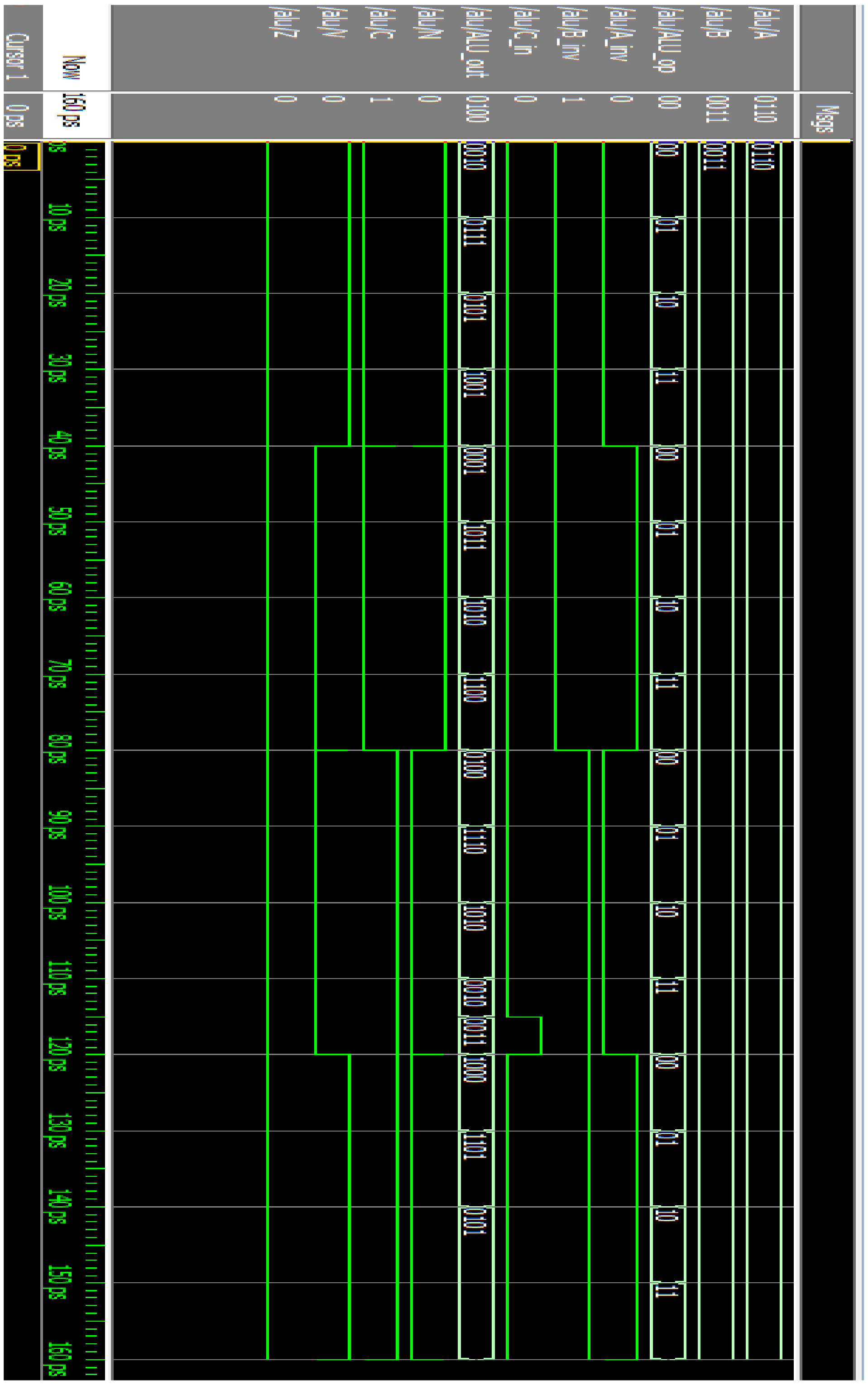
```
entity ALU is
      port(
      A, B          : in std_logic_vector(3 downto 0);
      ALU_op        : in std_logic_vector(1 downto 0);
      A_inv, B_inv  : in std_logic;
      C_in          : in std_logic;
      ALU_out       : out std_logic_vector(3 downto 0);
      N, C, V, Z    : out std_logic);
end ALU;
```

  - You must use the following components to design your ALU: Adder4Bit, Mux2Input4Bit, and Mux4Input4Bit, according to the diagram below.
    - Hints: 4-bit AND can be easily implemented using just “and” for two 4-bit signals.
    - Hints: The flag circuit is very similar to the flag circuit that we studied in the class for 32-bit ALU.


ALU
ALU_op
2
A_inv
A
4
0
1
B
4
0
1
B_inv
C_in
4-bit
AND
4-bit
OR
4-bit
XOR
X
Y
C_in
S
4-bit
ADDER
C_out_2
C_out_3
00
01
10
11
ALU_out
4
Flag
circuit
N
C
V
Z

Step 2 – ModelSim Simulation

- Please download simulation script task.do from Canvas, and then add it to your project
- Compile your project and then run Modelsim simulation
- If your ALU is designed correctly, you should get the following waves.



Step 3 – DE10 Lite board testing

- In order to test your circuit on your DE10 Lite board, you need to do the pin assignment according to the following table. You can do it either manually as before, or automatically as follows (recommended).
  - Download pin assignment file pin_assignments.csv from Canvas
  - Click Quartus menu “Assignments”
  - Select “Import Assignments”
  - Select file “pin_assignments.csv” that you just downloaded
  - Click “OK”

Node Name	Location	I/O Standard	Corresponding components on DE10 Lite Board
A_inv		3.3-V LVTTL	Not Connected (i.e. 0V)
B_inv	PIN_A7	3.3 V SCHMITT TRIGGER	Push button 1 (KEY1)
C_in	PIN_B8	3.3 V SCHMITT TRIGGER	Push button 0 (KEY0)
ALU_op[1]	PIN_F15	3.3-V LVTTL	Slider Switch 9 (SW9)
ALU_op[0]	PIN_B14	3.3-V LVTTL	Slider Switch 8 (SW8)
A[3]	PIN_A14	3.3-V LVTTL	Slider Switch 7 (SW7)
A[2]	PIN_A13	3.3-V LVTTL	Slider Switch 6 (SW6)
A[1]	PIN_B12	3.3-V LVTTL	Slider Switch 5 (SW5)
A[0]	PIN_A12	3.3-V LVTTL	Slider Switch 4 (SW4)
B[3]	PIN_C12	3.3-V LVTTL	Slider Switch 3 (SW3)
B[2]	PIN_D12	3.3-V LVTTL	Slider Switch 2 (SW2)
B[1]	PIN_C11	3.3-V LVTTL	Slider Switch 1 (SW1)
B[0]	PIN_C10	3.3-V LVTTL	Slider Switch 0 (SW0)
N	PIN_D14	3.3-V LVTTL	LED Red 7
C	PIN_E14	3.3-V LVTTL	LED Red 6
V	PIN_C13	3.3-V LVTTL	LED Red 5
Z	PIN_D13	3.3-V LVTTL	LED Red 4
ALU_out[3]	PIN_B10	3.3-V LVTTL	LED Red 3
ALU_out[2]	PIN_A10	3.3-V LVTTL	LED Red 2
ALU_out[1]	PIN_A9	3.3-V LVTTL	LED Red 1
ALU_out[0]	PIN_A8	3.3-V LVTTL	LED Red 0

- After the pin assignment, recompile your project and then burn your FPGA as before.
- Answer the questions on Canvas using your DE10 Lite board.
  - **Hint:** Signal C_in is connected to KEY0. Be careful that when KEY0 is released, C_in =1, and when KEY0 is being pressed, C_in=0. Similar for B_inv connected to KEY1, and A_inv not connected so A_inv=0.