City University of Hong Kong

Department of Electrical Engineering

EE 2000 – Lab Manual 3

Simple ALU design

Course Leader: CHIN Lip Ket

Objectives:

- To design combinational logic circuits
- To revise your understanding from previous labs
- To learn how to perform type conversion in VHDL Code
- To apply bit slicing and bit concatenation technique in VHDL
- To learn how to use a logic analyzer to trace digital signals

There are 4 checkpoints in total.
For each checkpoint, please take notes/photos/screenshots for your report.

Please show Checkpoints 1, 2, 4 to the lab tutor or demonstrator for marking.

Experiment 1: ALU Implementation

i. An arithmetic logic unit (ALU) is to perform arithmetic and bitwise operation on binary integer numbers. It consists of two units, including arithmetic unit (e.g. addition, subtraction, etc.) and logic unit (e.g. AND, OR, etc.). In this lab, you will implement a simple 3-bit ALU. The result is shown in the following table, which includes two 3-bit inputs **A**, **B** and the 2-bit operation code **op**.

op	Result	carry
00	A **ADD** B	It depends
01	A **AND** B	0
10	A **XOR** B	0
11	A **<< 1**	0

Here, we provide a template for you to write your own file **alu.vhd**. In this design, we add a copy of the output signals **result** and **carry**, defined as **dup_result** and **dup_carry** in the following code. For these two copies of the outputs, one is used to drive the LED and another is the output connects to PMOD connector JA. Using PMOD connector, we can use a logic analyzer to probe signals later on. In your code, **result_t** and **carry_t** used to store the required output.

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

Entity ALU is
   Port (
      A: in STD_LOGIC_VECTOR (2 DOWNTO0);
      B: in STD_LOGIC_VECTOR (2 DOWNTO0);
      op: in STD_LOGIC_VECTOR (1 DOWNTO 0);
      result: out STD_LOGIC_VECTOR (2 DOWNTO 0);
      carry: out STD_LOGIC;

      --dup_result and dup_carry used for observation when using Logic Analyzer

      dup_result: out STD_LOGIC_VECTOR (2 DOWNTO 0);

      dup_carry: out STD_LOGIC

   );
end ALU;

Architecture Behavioral of ALU is
SIGNAL result_t: STD_LOGIC_VECTOR (2 DOWNTO 0);
SIGNAL carry_t: STD_LOGIC;
--You can add your declaration here.
--(Tips: You may add a SIGNAL that temporary store the addition result of A and B.)
begin
   dup_result <= result_t;
   dup_carry <= carry_t;
   result <= result_t; carry
   <= carry_t;

   --Write your own code to implement this ALU.
end Behavioral;
```

ii. Here are some hints for you to write your own VHDL code:

(1) In VHDL, if the type is STD_LOGIC_VECTOR, you can use the following operators directly.

```
A + B        -- Addition
A AND B      -- Logical and
A OR B       -- Logical or
A XOR B      -- Logical xor
```

(2) VHDL has logical shift operators **sll** (shift left) and **srl** (shift right), but they are for BIT_VECTOR. Since A and result are both STD_LOGIC_VECTOR, type conversion is needed.

```
result <= to_stdlogicvector(to_bitvector(A) sll 1);
```

(3) You may get the sum of two 3-bit inputs by converting them into 4-bit inputs, and that is

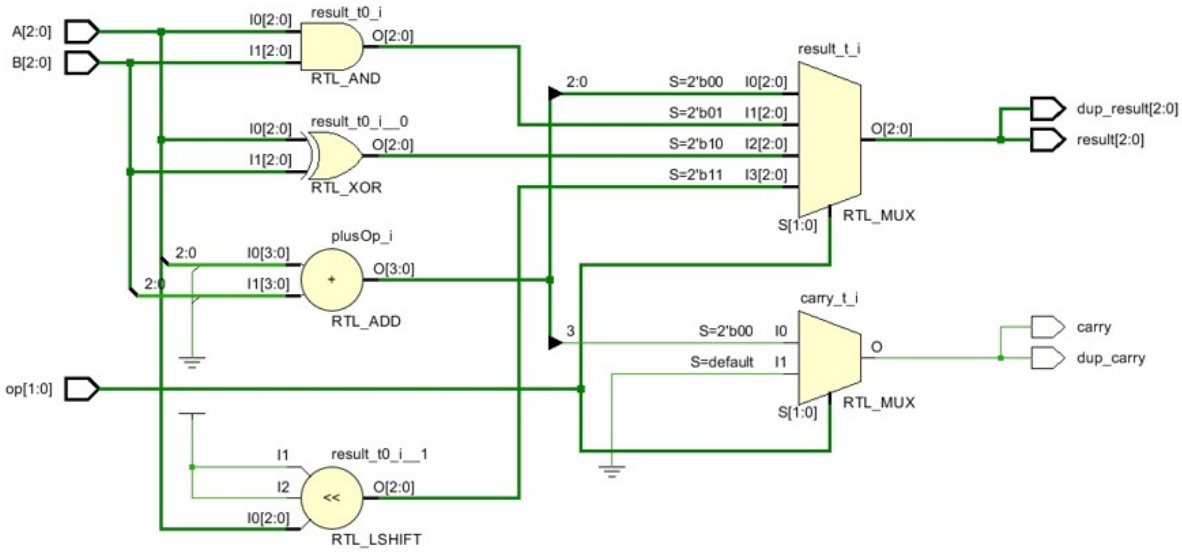
```
tmp <= ('0'&A) + ('0'&B);
-- A and B are both 3-bit inputs
-- tmp is a 4-bit result
-- tmp(2 downto 0) is the sum and tmp(3) is the carry value
```

(4) Now you can learn how to use the **WHEN** statement. Here is a simple example for you. It is a simple decoder from SIGNAL sel to SIGNAL result_out. Note the use of ',', and ';' carefully.

```
WITH sel SELECT
        result_out <= "0001" WHEN "00",
        "0010" WHEN "01",
        "0100" WHEN "10",
        "1000" WHEN "11",
        "ZZZ" WHEN others;
```

iii. **Add sources** alu.vhd to your own project.

iv. **Perform RTL analysis** on the source file. Get the schematic and see how Vivado infer logic gates according to your code. The schematic might be different from the one shown below.



v. Write an **I/O constraint** file and add it to your project. It is similar as Lab 2. In this experiment, we use SW0-SW2 as input **A**, SW3-SW5 as input **B**, SW6- SW7 as input **op**, LD0- LD2 as output **result**, LD3 as output **carry**, JA1-JA3 are connected with output port **dup_result**[0]-**dup_result**[2] and JA4 with **dup_carry**.

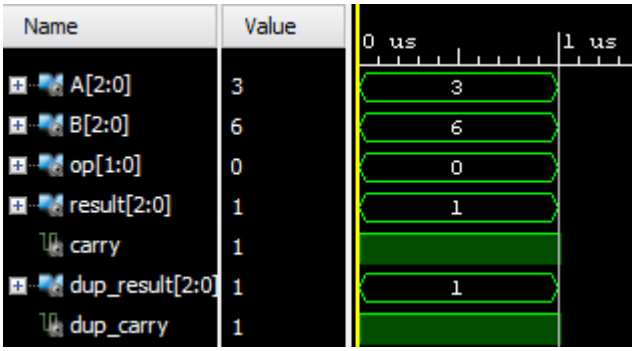
vi. **Simulate** the Design using the XSim Simulator.

To test your design, we should write a testbench to generate the input signal. In this design, input signals include operand **A** and **B**, operator **op**. In the simulation process, we should assign different values to inputs (**A**, **B** and **op**), then verify the outputs (**carry** and **result**). For testbench, we still use the same VHDL grammar to generate expected stimulations. Here we provide a simple test bench as follows.

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;

Entity ALU_tb is end ALU_tb;

Architecture Behavioral of ALU_tb is COMPONENT ALU is
Port (
A: in STD_LOGIC_VECTOR (2 DOWNTO0);
B: in STD_LOGIC_VECTOR (2 DOWNTO0);
op: in STD_LOGIC_VECTOR (1 DOWNTO 0);
result: out STD_LOGIC_VECTOR (2 DOWNTO 0);
carry: out STD_LOGIC;
dup_result:out STD_LOGIC_VECTOR (2 DOWNTO 0);
dup_carry: out STD_LOGIC
);
End COMPONENT;
SIGNALA: STD_LOGIC_VECTOR (2 DOWNTO 0);
SIGNAL B: STD_LOGIC_VECTOR (2 DOWNTO0);
SIGNAL op: STD_LOGIC_VECTOR (1 DOWNTO 0);
SIGNAL result: STD_LOGIC_VECTOR (2 DOWNTO 0);
SIGNAL carry: STD_LOGIC;
SIGNAL dup_result: STD_LOGIC_VECTOR (2 DOWNTO 0);
SIGNAL dup_carry: STD_LOGIC;
begin
uut: ALU PORT MAP (A, B, op, result, carry, dup_result, dup_carry);
A <= “011”;
B <= “110”;
op <= “00”;
end Behavioral;
```



You can also add a **PROCESS** block to generate more simulation results. Try to write a test bench alu_tb.vhd to verify your design. Variables declaration and design instantiation are provided as follows.

```
Library IEEE;
Use IEEE.STD_LOGIC_1164.ALL;

Entity ALU_tb is
end ALU_tb;

Architecture Behavioral of ALU_tb is
   COMPONENT ALU is
      Port (
         A: in STD_LOGIC_VECTOR (2 DOWNTO 0);
         B: in STD_LOGIC_VECTOR (2 DOWNTO 0);
         op: in STD_LOGIC_VECTOR (1 DOWNTO 0);
         result: out STD_LOGIC_VECTOR (2 DOWNTO 0);
         carry: out STD_LOGIC;
         dup_result: out STD_LOGIC_VECTOR (2 DOWNTO 0);
         dup_carry: out STD_LOGIC
      );
```

```
    End COMPONENT;
    SIGNALA: STD_LOGIC_VECTOR (2 DOWNTO 0);
    SIGNAL B: STD_LOGIC_VECTOR (2 DOWNTO 0);
    SIGNAL op: STD_LOGIC_VECTOR (1 DOWNTO 0);
    SIGNAL result: STD_LOGIC_VECTOR (2 DOWNTO 0);
    SIGNAL carry: STD_LOGIC;
    SIGNAL dup_result: STD_LOGIC_VECTOR (2 DOWNTO 0);
    SIGNAL dup_carry: STD_LOGIC;
begin
    uut: ALU PORT MAP (A, B, op, result, carry, dup_result, dup_carry);
        siggen: PROCESS
          begin
          -- Add your code here to set values of A, B and op.
        end PROCESS siggen;
end Behavioral;
```

Checkpoint 1: Show your testbench and the waveform window captures (that shows ADD, AND, XOR, shift operation) to your demonstrator.

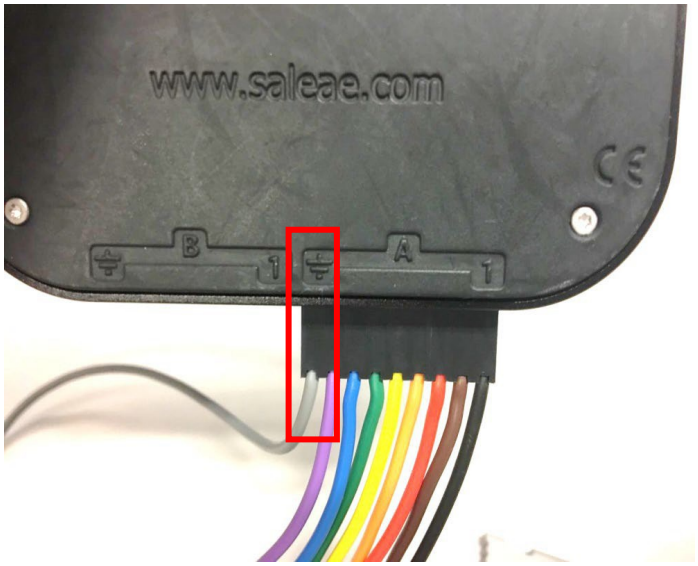
Run synthesis, **implementation and generate bitstream** as in Lab 1. Then, we can **program** the Basys 3 Board. Finally, you can verify the design by testing different inputs.

Checkpoint 2: Show your board and results to your demonstrator and verify your design properly. Please remember to take the screen capture, and photo for the lab report writing.

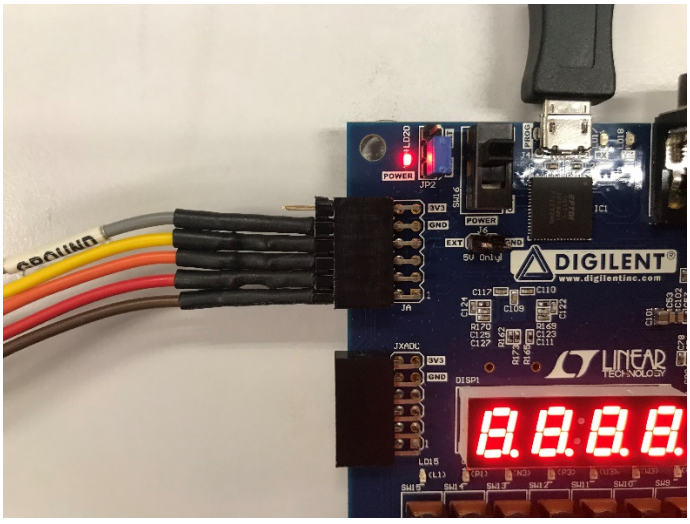
Checkpoint 3: Discuss in the lab report how to modify the VHDL source code to implement an ALU with 6 operations.

Experiment 2: Use Logic Analyzer to observe signals

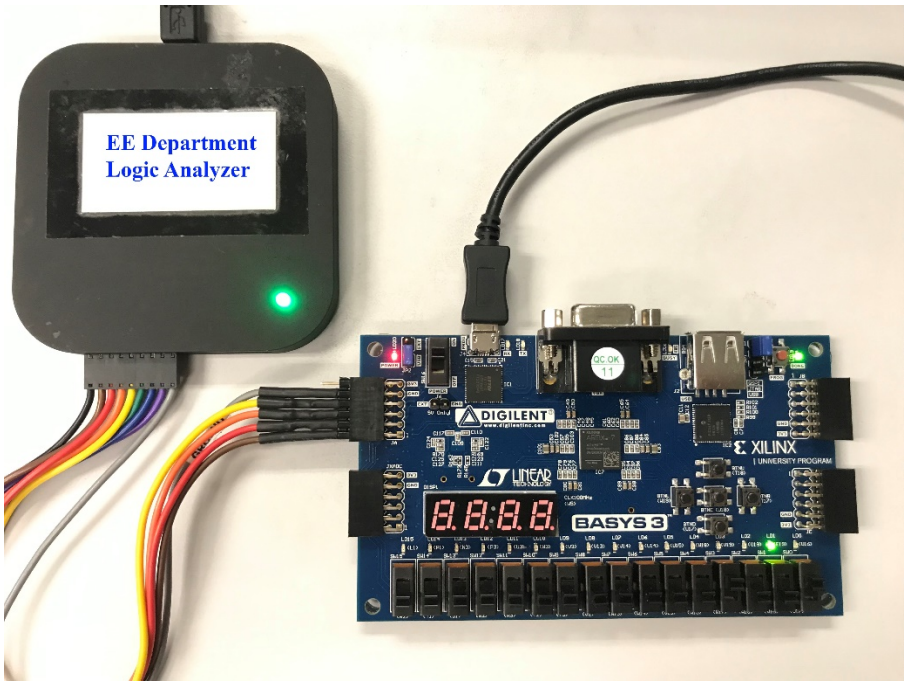
i. First, we should connect the rainbow cable to the black hub, as shown in the figure below. Ensure that the grey cable is aligned with the ground icon on the back of the logic analyzer.



ii. Then, you can connect the cable with JA pmod connector, i.e. connect the grey cable to the GND port, brown cable to JA1, red cable to JA2, orange cable to JA3 and yellow cable to JA4. The grey one should connect to the GND pin.

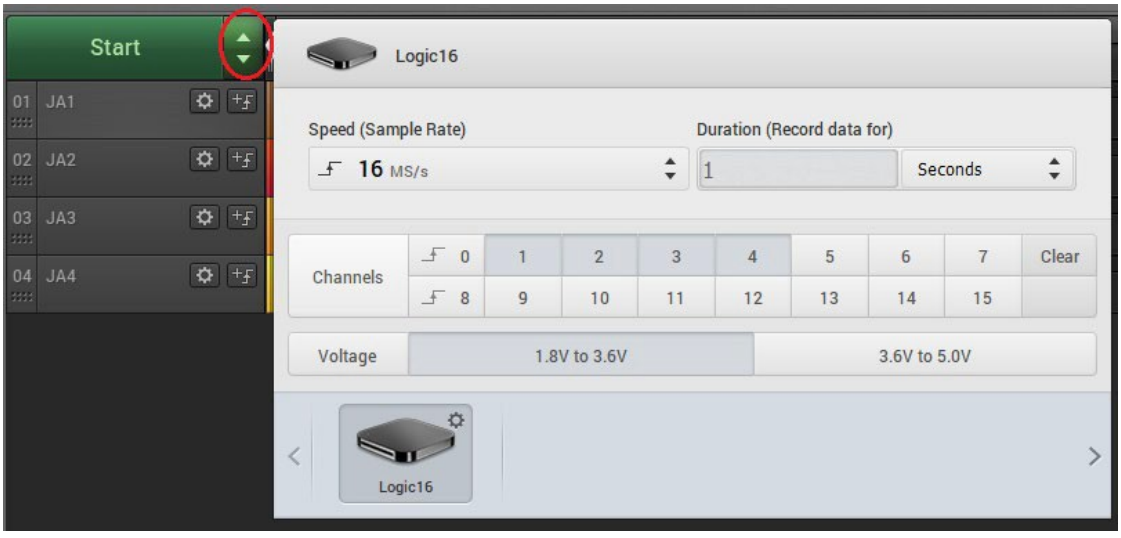


iii. Connect the logic analyzer to the computer using the micro-USB cable.



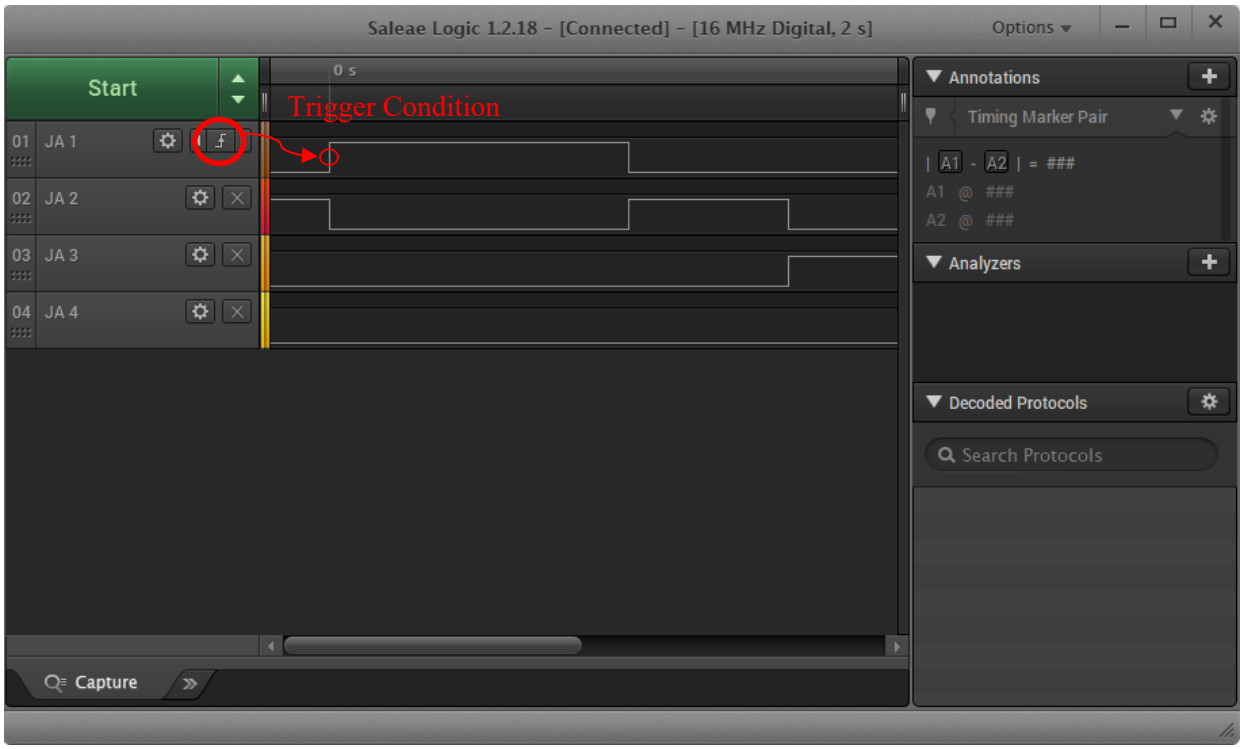
iv. Open **Logic 1.2.18** software on your computer's desktop.

v. Click the **arrow key** beside **Start**. Adjust the settings as shown in the following figure. (Logic 16: Speed 16 MS/s. Duration 1 Second. Channels 1,2,3,4. Voltage 1.8V to 3.6V)



vi. After setting all the parameters correctly, click **Start**.

vii. The waveform is transferred once you click the start button.



Checkpoint 4: Show your logic analyzer waveform result and your BASYS3 board to the demonstrator in your lab session. You should explain how the waveform is generated in your ALU design.

~ END ~