Report HW 2 ECE-111

Name: Jasper Huang

PID: A17796149

Homework 2a (4-bit ALU)

1. **SystemVerilog code snapshot**
   a. **alu_top**

```
// N-bit ALU TOP RTL code        You, 14 minutes ago • F
module alu_top // Module start declaration
#(parameter N=4) // Parameter declaration
(  input logic clk, reset,
   input logic[N-1:0]operand1, operand2,
   input logic[3:0] select,
   output logic[(2*N)-1:0] result
);

  // Local net declaration
  logic[(2*N)-1:0] alu_out;

  // Student to Add instantiation of module alu
  alu #(.N(N)) alu_inst (
      .operand1(operand1),
      .operand2(operand2),
      .operation(select),
      .alu_out(alu_out)
    );

  // Adding flipflop at the output of ALU
  always@(posedge clk or posedge reset) begin
    if(reset == 1) begin
      result <= 0;
    end
    else begin
      result <= alu_out;
    end
  end
endmodule: alu_top // Module alu_top end declaration
```

b. alu

```
// 1-bit ALU behavioral code
module alu // Module start declaration
#(parameter N=4) // Parameter declaration
(
  input logic[N-1:0] operand1, operand2,
  input logic[3:0] operation,
  output logic[(2*N)-1:0] alu_out
);

  // always procedural block describing alu operations
  always@(operand1 or operand2 or operation)
  begin
    // Student to add remainder part of the code
    case (operation)
      4'b0000: alu_out = operand1 + operand2;
      4'b0001: alu_out = operand1 - operand2;
      4'b0010: alu_out = operand1 * operand2;
      4'b0011: alu_out = operand2 != 0 ? operand1 % operand2 : 0;
      4'b0100: alu_out = operand2 != 0 ? operand1 / operand2 : 0;
      4'b0101: alu_out = operand1 & operand2;
      4'b0110: alu_out = operand1 | operand2;
      4'b0111: alu_out = operand1 ^ operand2;
      4'b1000: alu_out = (operand1 != 0 && operand2 != 0);
      4'b1001: alu_out = (operand1 != 0 || operand2 != 0);
      4'b1010: alu_out = operand1 << 1;
      4'b1011: alu_out = operand1 >> 1;
      4'b1100: alu_out = (operand1 == operand2);
      4'b1101: alu_out = (operand1 != operand2);
      4'b1110: alu_out = (operand1 < operand2);
      4'b1111: alu_out = (operand1 > operand2);
      default: alu_out = '0;
    endcase
  end
endmodule: alu
```

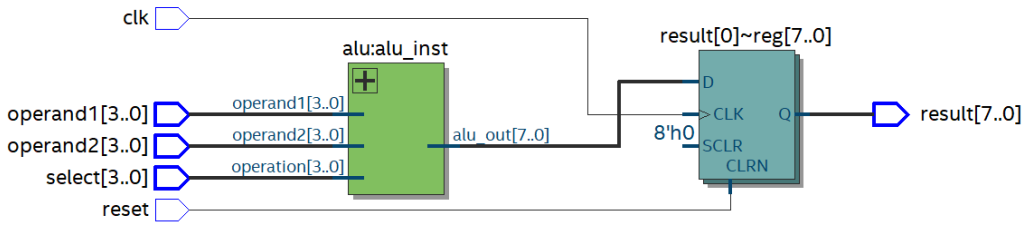
2. **Provide snapshot of FPGA resource usage generated post synthesis**

Analysis & Synthesis Resource Usage Summary

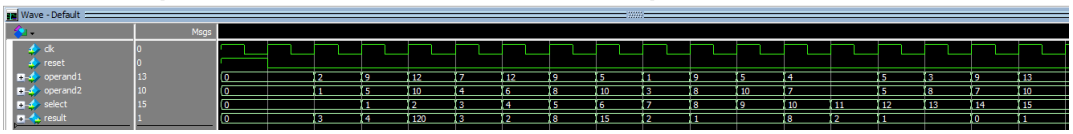
<<Filter>>

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	56
2		
3	Combinational ALUT usage for logic	97
1	-- 7 input functions	3
2	-- 6 input functions	12
3	-- 5 input functions	16
4	-- 4 input functions	22
5	-- <=3 input functions	44
4		
5	Dedicated logic registers	8
6		
7	I/O pins	22
8		
9	Total DSP Blocks	1
10		
11	Maximum fan-out node	operand2[3]~input
12	Maximum fan-out	28
13	Total fan-out	428
14	Average fan-out	2.85

3. **Provide snapshot of schematic generated from RTL netlist viewer**



4. Provide snapshot of simulation waveform and explain simulation result



The simulation graph shows the clock tick and after the reset flagged, the operand1 and operand2 began to perform based on the selected operation (determine by the select logic) and result changes based on the operation result from operand1 and operand2. For each of the select, the result matches the expected operation which for example, select 0 is addition, which 2 + 1 = 3 matches the result 3.

Homework 2b (4-bit up down binary counter)

1. **SystemVerilog code snapshot**
   a. **up_down_counter**

```
// 4-bit up and down counter RTL code        You, 40 minutes
module up_down_counter    // Module start declaration
 // Parameter declaration, count signal width set to '4'
 #(parameter WIDTH=4)
 (
    input logic clk,
    input logic clear,
    input logic select,
    output logic[WIDTH-1:0] count_value
 );

 // Local variable declaration
 logic[WIDTH-1:0] up_count_value, down_count_value;

 // Student to add code to instantiate up counter
   up_counter #(.WIDTH(WIDTH)) up_counter(
      .clk(clk),
      .clear(clear),
      .count(up_count_value)
   );

 // Student to add code to instantiate down counter
   down_counter #(.WIDTH(WIDTH)) down_counter(
      .clk(clk),
      .clear(clear),
      .count(down_count_value)
   );

 // Student to add acode to instantiate 2-to-1 multiplexer

   mux_2x1 #(.WIDTH(WIDTH)) mux_2x1(
      .in0(up_count_value),
      .in1(down_count_value),
      .sel(select),
      .out(count_value)
   );

endmodule: up_down_counter  // Module end declaration
```

b. up_counter

```
// 4-bit counter RTL behavioral code
module up_counter    // Module start declaration
 // Parameter declaration, count signal width set to '4'
 #(parameter WIDTH=4)
 (
    input logic clk,
    input logic clear,
    output logic[WIDTH-1:0] count       You, 40 minutes ago
 );

 // Local variable declaration
 logic[WIDTH-1:0] cnt_value;

 // always procedural block describing up counter behavior
 always @(posedge clk or posedge clear)
   begin
     if (clear == 1)
       cnt_value = 0;
     else
       cnt_value = cnt_value + 1;
   end

 // Counter value assigned to output port count
 assign count = cnt_value;
endmodule: up_counter  // Module end declaration
```

c. **down_counter**

```
// 4-bit counter RTL code        You, 40 minutes ago •
module down_counter    // Module start declaration
 // Parameter declaration, count signal width set to
 #(parameter WIDTH=4)
 (
    input logic clk,
    input logic clear,
    output logic[WIDTH-1:0] count
 );

 // Student to add code for down counter logic
 logic[WIDTH-1:0] cnt_value;

 always @(posedge clk or posedge clear)
   begin
     if (clear == 1)
       cnt_value = 'b1111;
     else
       cnt_value = cnt_value - 1;
   end

 // Counter value assigned to output port count
 assign count = cnt_value;

endmodule: down_counter  // Module end declaration
```

d. mux_2x1

```
// 2to1 Multiplexor RTL code
module mux_2x1 #(parameter WIDTH=4)
(
  input logic[WIDTH-1:0] in0, // Student to change in0 width to 4
  input logic[WIDTH-1:0] in1, // Student to change in1 width to 4
  input logic sel,
  output logic[WIDTH-1:0] out // Student to change out width to 4
);

  // always procedural block describing 2to1 Multiplexor behavior
  always @(sel or in0 or in1)
  begin        You, 40 minutes ago • First Commit
    if(sel == 0)
      out = in0;
    else
      out = in1;
  end
endmodule
```

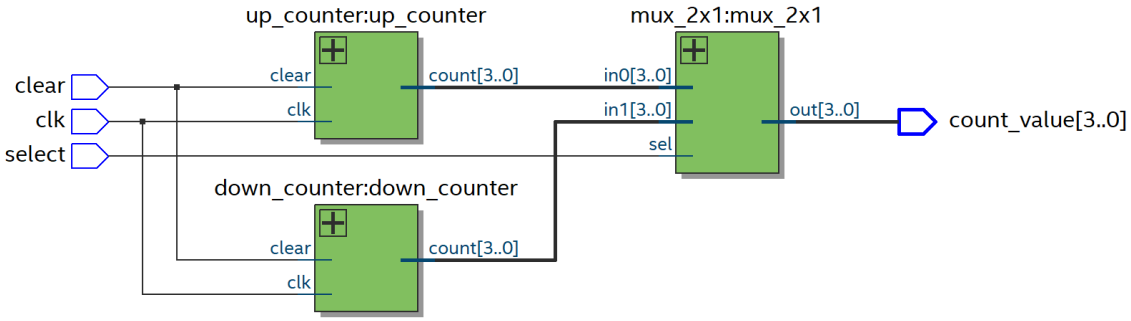
2. **Provide snapshot of FPGA resource usage generated post synthesis**

Analysis & Synthesis Resource Usage Summary

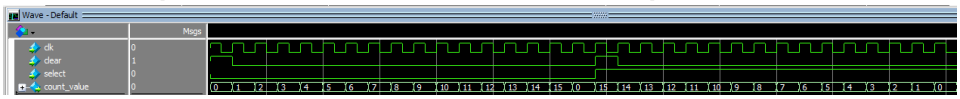
<<Filter>>

	Resource	Usage
1	Estimate of Logic utilization (ALMs needed)	5
2		
3	Combinational ALUT usage for logic	10
1	-- 7 input functions	0
2	-- 6 input functions	0
3	-- 5 input functions	0
4	-- 4 input functions	2
5	-- <=3 input functions	8
4		
5	Dedicated logic registers	6
6		
7	I/O pins	7
8		
9	Total DSP Blocks	0
10		
11	Maximum fan-out node	up_counte..._value[0]
12	Maximum fan-out	7
13	Total fan-out	56
14	Average fan-out	1.87

3. **Provide snapshot of schematic generated from RTL netlist viewer**



4. **Provide snapshot of simulation waveform and explain simulation result**



The simulation waveform shows a count_value changing at each positive edge of clk where the counting value changes depending on either select 0 which is count_up from 0 to 15 or count_down which is from 15 to 0. The presented waveform shows we first clear and select count up, which you can see 0 to 15, and then after second clear, the count down is selected which shows 15 to 0.