Giao Tran

Convert 1-bit ALU to 4-bit ALU and change ALU module instantiation in ALU top module from explicit name port connection to dot-star port connection

Alu_top code

```
// N-bit ALU behavioral code
`timescale 1ns/1ps
module alu_top // Module start declaration
#(parameter N=4) // Parameter declaration
( input logic clk, reset,
   input logic[N-1:0]operand1, operand2,
   input logic[1:0] operation,
   output logic[N-1:0] result
);

  // Local net declaration
  logic[N-1:0] alu_out;

  // Instantiation of module alu
  alu #(.N(N)) alu_instance(
  .opnd1(operand1),
    .opnd2(operand2),
    .operation(operation),
    .out(alu_out)
  );

  defparam alu_instance.N=4;

  // Register alu output
  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
```

alu code :

```
// 1-bit ALU behavioral code
`timescale 1ns/1ps
module alu // Module start declaration
#(parameter N=1) // Parameter declaration
(
  input logic[N-1:0] opnd1, opnd2,
  input logic[1:0] operation,
  output logic[N-1:0] out
);
  always@(opnd1 or opnd2 or operation)
  begin
    case(operation)
      2'b00: out = opnd1 + opnd2;
      2'b01: out = opnd1 - opnd2;
      2'b10: out = opnd1 & opnd2;
      2'b11: out = opnd1 | opnd2;
    endcase
  end
endmodule: alu

```

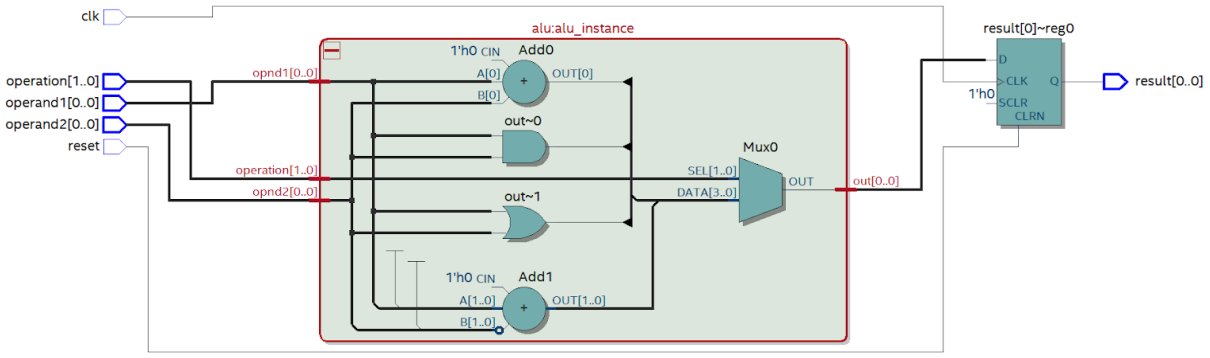
Testbench code:

```
//1-bit ALU testbench code
`timescale 1ns/1ps
module alu_top_testbench;
parameter N = 4;
logic clock, reset;
logic [N-1:0] operand1, operand2;
logic [N-1:0] result;
logic [1:0] operation;

// Instantiate design under test
alu_top #(.N(N)) design_instance(
  .clk(clock),
  .reset(reset),
  .operand1(operand1),
  .operand2(operand2),
  .operation(operation),
  .result(result)
);

initial begin
// Initialize Inputs
reset = 1;
clock = 0;
operand1 = 0;
operand2 = 0;
operation = 0;

// Wait 20 ns for global reset to finish and start counter
#20ns;
reset = 0;

#20ns
operand1 = 0;
operand2 = 1;
operation = 0;

#20ns;
operand1 = 1;
operand2 = 1;
operation = 1;

#20ns;
operand1 = 1;
operand2 = 1;
operation = 2;
```

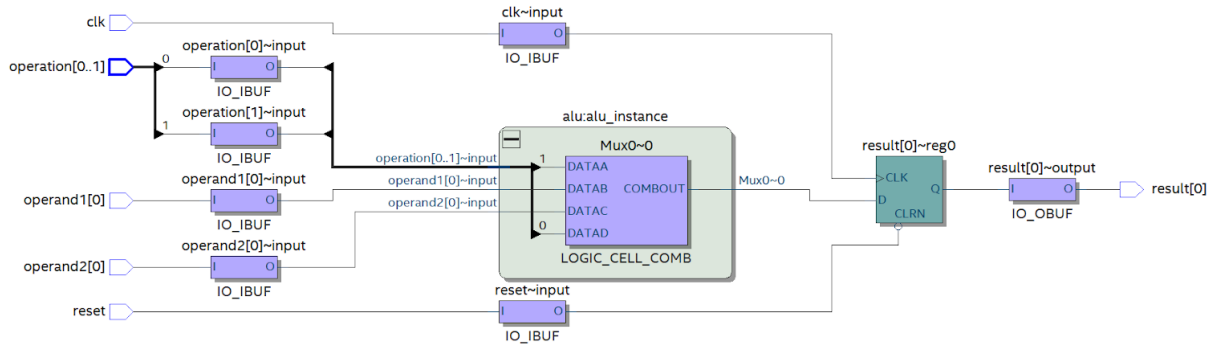
```
#20ns;
operand1 = 1;
operand2 = 0;
operation = 3;

// Wait for 10ns
#20ns;

// terminate simulation
$finish();
end

// Clock generator logic
always@(clock) begin
  #10ns clock <= !clock;
end

// Print input and output signals
initial begin
 $monitor(" time=%0t,  clk=%b  reset=%b  operation=%d, operand1=%d, operand2=%d",$time, clock, reset, operation, operand1, operand2);
end
endmodule
```

1 bit alu_top SystemVerilog design

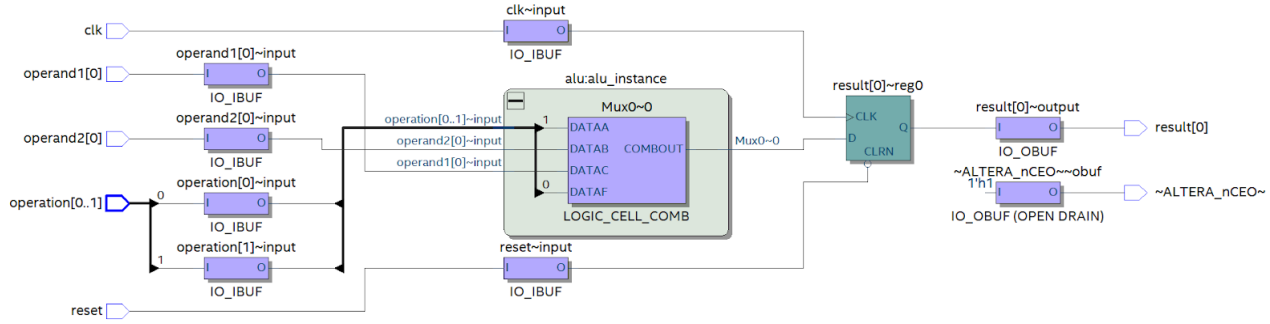
a. 1 bit alu RTL



b. 1 bit alu Post mapping



c. 1 bit alu Post fitting



d. 1 bit alu Resource usage summary

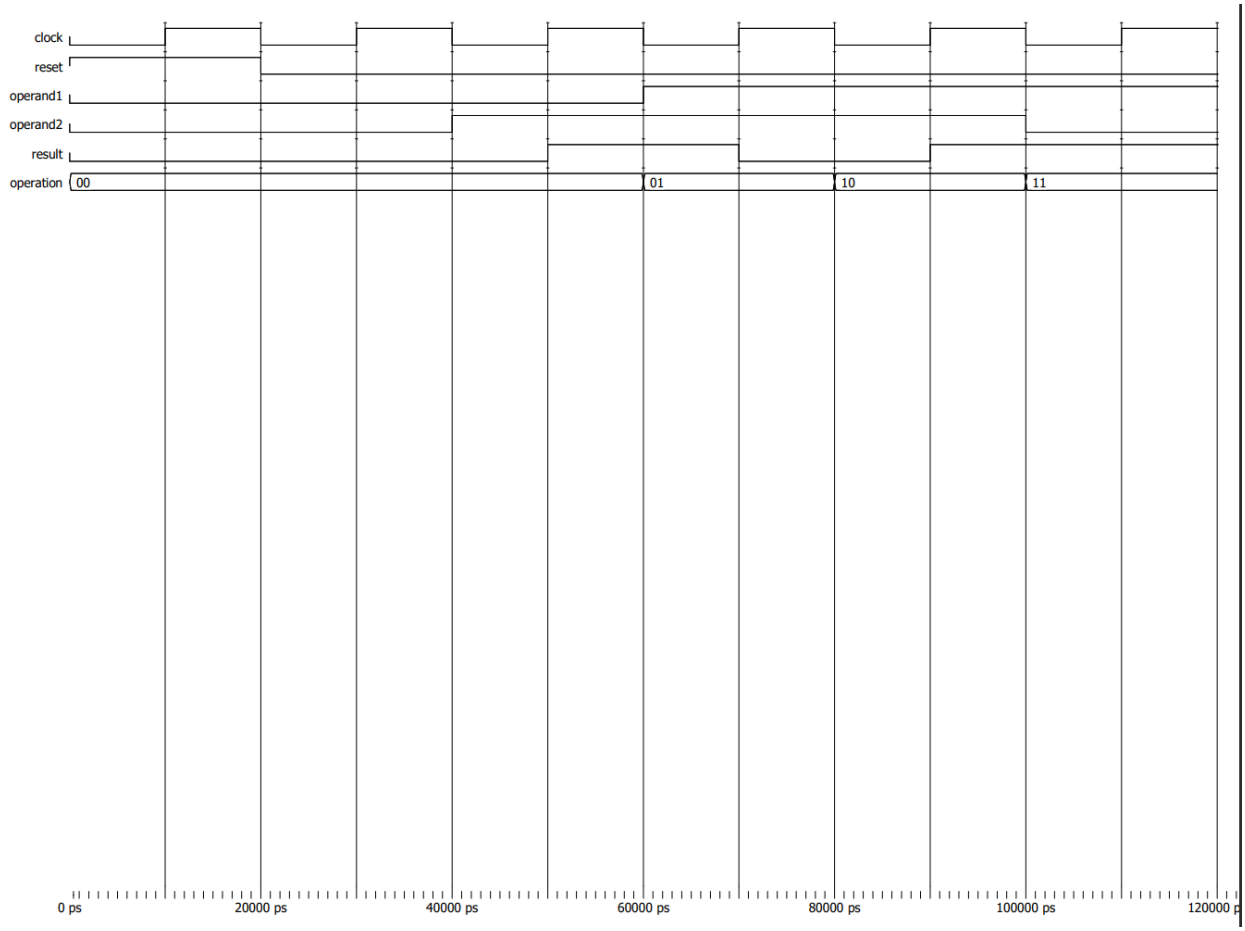
	Resource	Usage
1	Estimated ALUTs Used	1
1	-- Combinational ALUTs	1
2	-- Memory ALUTs	0
3	-- LUT_REGs	0
2	Dedicated logic registers	1
3		
4	Estimated ALUTs Unavailable	0
1	-- Due to unpartnered combinational logic	0
2	-- Due to Memory ALUTs	0
5		
6	Total combinational functions	1
7	Combinational ALUT usage by number of inputs	
1	-- 7 input functions	0
2	-- 6 input functions	0
3	-- 5 input functions	0
4	-- 4 input functions	1
5	-- <=3 input functions	0
8		
9	Combinational ALUTs by mode	
1	-- normal mode	1
2	-- extended LUT mode	0
3	-- arithmetic mode	0
4	-- shared arithmetic mode	0
10		
11	Estimated ALUT/register pairs used	1
12		
13	Total registers	1
1	-- Dedicated logic registers	1
2	-- I/O registers	0
3	-- LUT_REGs	0
14		
15		
16	I/O pins	7
17		
18	DSP block 18-bit elements	0
19		
20	Maximum fan-out node	result[0]~reg0
21	Maximum fan-out	1
22	Total fan-out	15
23	Average fan-out	0.94

Number of ALUT: 1 (7 I/O pins)

Normal mode: 1

Number of Functions: 1 (4 input function)

e. 1 bit alu Waveform simulation and transcript



Transcript

```
# Errors: 0, Warnings: 0
# vsim work.alu_top_testbench
# Start time: 16:14:52 on Jan 26,2021
# Loading sv_std.std
# Loading work.alu_top_testbench
# Loading work.alu_top
# Loading work.alu
add wave -position insertpoint sim:/alu_top_testbench/*
VSIM 10> run -all
#  time=0,  clk=0  reset=1  operation=0, operand1=0, operand2=0
#  time=10000,  clk=1  reset=1  operation=0, operand1=0, operand2=0
#  time=20000,  clk=0  reset=0  operation=0, operand1=0, operand2=0
#  time=30000,  clk=1  reset=0  operation=0, operand1=0, operand2=0
#  time=40000,  clk=0  reset=0  operation=0, operand1=0, operand2=1
#  time=50000,  clk=1  reset=0  operation=0, operand1=0, operand2=1
#  time=60000,  clk=0  reset=0  operation=1, operand1=1, operand2=1
#  time=70000,  clk=1  reset=0  operation=1, operand1=1, operand2=1
#  time=80000,  clk=0  reset=0  operation=2, operand1=1, operand2=1
#  time=90000,  clk=1  reset=0  operation=2, operand1=1, operand2=1
#  time=100000,  clk=0  reset=0  operation=3, operand1=1, operand2=0
#  time=110000,  clk=1  reset=0  operation=3, operand1=1, operand2=0
# ** Note: $finish    : C:/Users/user/Desktop/ECE 111 HW/Lab2/Lab2/alu_top/alu_top_testbench.sv(56)
#    Time: 120 ns  Iteration: 0  Instance: /alu_top_testbench
# 1
# Break in Module alu_top_testbench at C:/Users/user/Desktop/ECE 111 HW/Lab2/Lab2/alu_top/alu_top_testbench.sv line 56
```

Converting explicit name port connection to dot-star port connection

```
// N-bit ALU behavioral code
`timescale 1ns/1ps
module alu_top // Module start declaration
#(parameter N=1) // Parameter declaration
( input logic clk, reset,
   input logic[N-1:0] opnd1, opnd2,
   input logic[1:0] operation,
   output logic[N-1:0] result
);

  // Local net declaration
  logic[N-1:0] out;

  // Instantiation of module alu
  alu #(.N(N)) alu_instance(.*);

  // Register alu output
  always@(posedge clk or posedge reset) begin
    if(reset == 1) begin
      result <= 0;
    end
    else begin
      result <= out;
    end
  end
endmodule: alu_top // Module alu_top end declaration
```

We can convert to dot-star connection by matching inputs and outputs' names from the alu.sv file. Below are the changes:

- operand 1 → opnd 1
- operand 2 → opnd 2
- out_alu → out
- Anything under instantiation of a module alu is replaced by .*

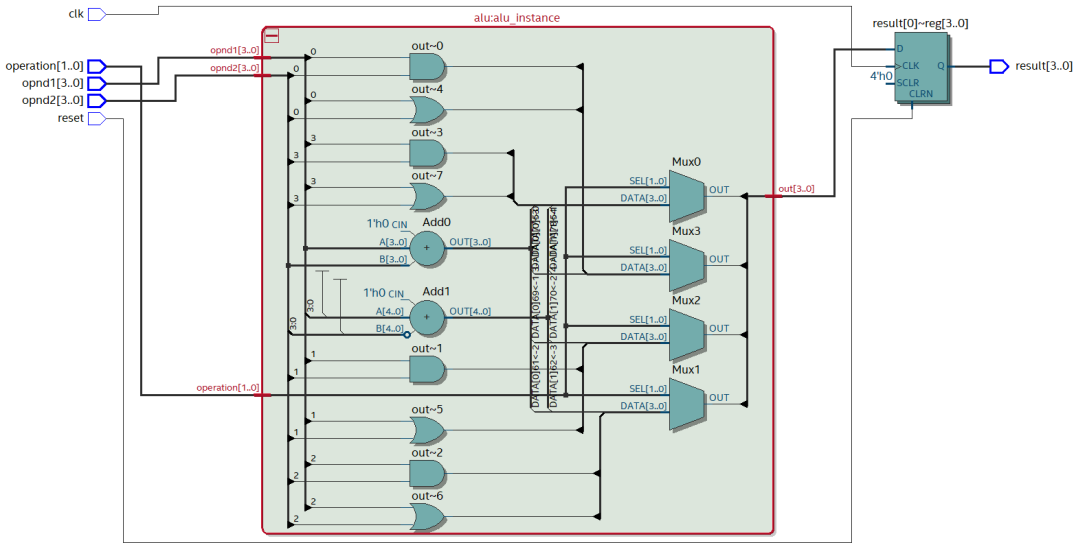
Convert 1-bit ALU to 4-bit ALU

```
// N-bit ALU behavioral code
`timescale 1ns/1ps
module alu_top // Module start declaration
#(parameter N=4) // Parameter declaration
( input logic clk, reset,
   input logic[N-1:0]operand1, operand2,
   input logic[1:0] operation,
   output logic[N-1:0] result
);

  // Local net declaration
  logic[N-1:0] alu_out;

  // Instantiation of module alu
  alu #(.N(N)) alu_instance(
    .opnd1(operand1),
    .opnd2(operand2),
    .operation(operation),
    .out(alu_out)
  );
  defparam alu_instance. N=4;
  // Register alu output
  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
```

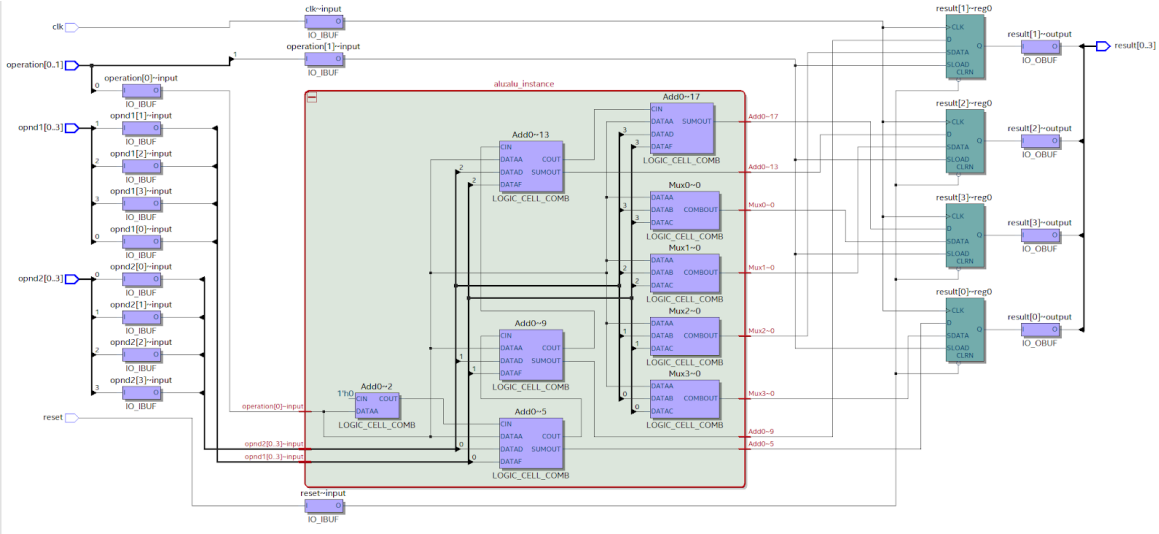
To convert 1 bit to 4 bit, change the parameter of alu_top.sv and the testbench from 1 to 4; N=4. The snapshot above is in explicit name port connection, but the code will produce the same results in dot star connection.

4 bit alu_top SystemVerilog design

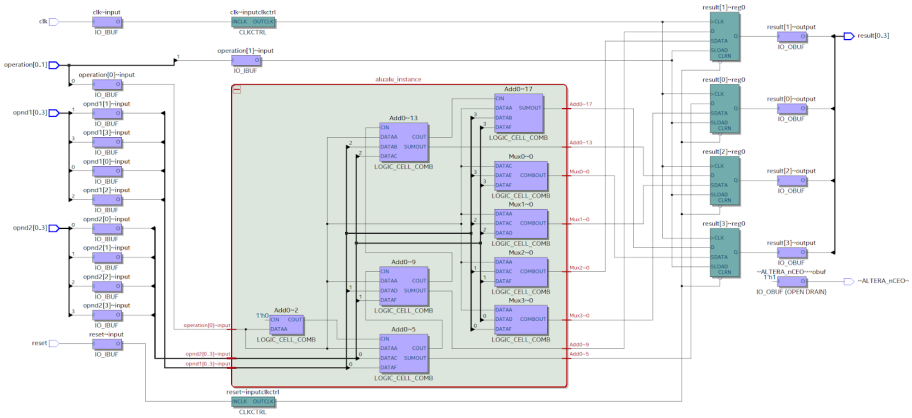
a. 4 bit alu RTL



b. 4 bit alu Post mapping



c. 4 bit alu Post fitting



d. Resource usage summary

	Resource	Usage
1	Estimated ALUTs Used	9
1	-- Combinational ALUTs	9
2	-- Memory ALUTs	0
3	-- LUT_REGs	0
2	Dedicated logic registers	4
3		
4	Estimated ALUTs Unavailable	0
1	-- Due to unpartnered combinational logic	0
2	-- Due to Memory ALUTs	0
5		
6	Total combinational functions	9
7	Combinational ALUT usage by number of inputs	
1	-- 7 input functions	0
2	-- 6 input functions	0
3	-- 5 input functions	0
4	-- 4 input functions	0
5	-- <=3 input functions	9
8		
9	Combinational ALUTs by mode	
1	-- normal mode	4
2	-- extended LUT mode	0
3	-- arithmetic mode	5
4	-- shared arithmetic mode	0
10		
11	Estimated ALUT/register pairs used	9
12		
13	Total registers	4
1	-- Dedicated logic registers	4
2	-- I/O registers	0
3	-- LUT_REGs	0
14		
15		
16	I/O pins	16
17		
18	DSP block 18-bit elements	0
19		
20	Maximum fan-out node	operation[0]~input
21	Maximum fan-out	9
22	Total fan-out	69
23	Average fan-out	1.53

Number of ALUT: 9 (16 I/O pins)

Normal mode: 4

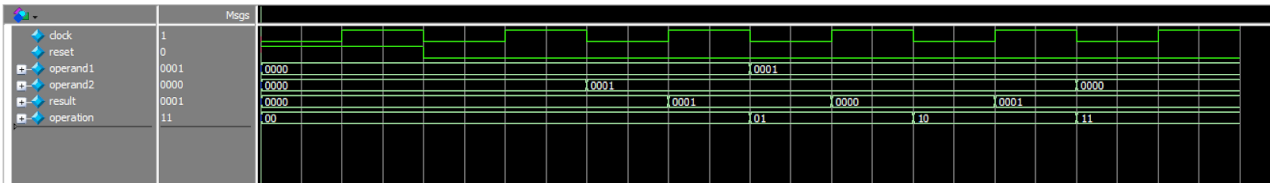
Arithmetic mode: 5

Number of Functions: 9 (3 input function)

The usage table summary shows that there are nine 3 input functions even though the post map shows that 4 ALUT have 4 inputs. The reason for this is because the only operand 1, operand 2 and operation are counted as inputs why the “4th” input the carry in is not counted. The carry in is transferred from one ALUT to anothe, first enters in CIN in Adder 0~5 and exits at COUT, then enters a different ALUT. As the 9 ALUT, the 4 normal mode ALUT are used for logic applications and combinational functions such as

the AND and OR operation, while 5 arithmetic ALUT are used for addition, subtraction, and accumulation.

e. Waveform simulation and transcript



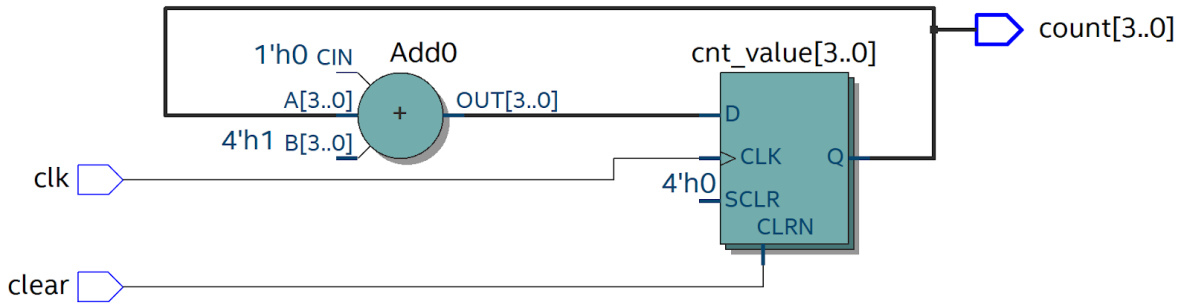
Transcript

```
ModelSim> vsim work.alu_top_testbench
# vsim work.alu_top_testbench
# Start time: 23:09:27 on Jan 26,2021
# Loading sv_std.std
# Loading work.alu_top_testbench
# Loading work.alu_top
# Loading work.alu
add wave -position insertpoint sim:/alu_top_testbench/*
VSIM 7> run -all
#  time=0,  clk=0  reset=1  operation=0, operand1= 0, operand2= 0
#  time=10000,  clk=1  reset=1  operation=0, operand1= 0, operand2= 0
#  time=20000,  clk=0  reset=0  operation=0, operand1= 0, operand2= 0
#  time=30000,  clk=1  reset=0  operation=0, operand1= 0, operand2= 0
#  time=40000,  clk=0  reset=0  operation=0, operand1= 0, operand2= 1
#  time=50000,  clk=1  reset=0  operation=0, operand1= 0, operand2= 1
#  time=60000,  clk=0  reset=0  operation=1, operand1= 1, operand2= 1
#  time=70000,  clk=1  reset=0  operation=1, operand1= 1, operand2= 1
#  time=80000,  clk=0  reset=0  operation=2, operand1= 1, operand2= 1
#  time=90000,  clk=1  reset=0  operation=2, operand1= 1, operand2= 1
#  time=100000,  clk=0  reset=0  operation=3, operand1= 1, operand2= 0
#  time=110000,  clk=1  reset=0  operation=3, operand1= 1, operand2= 0
# ** Note: $finish    : C:/Users/user/Desktop/ECE 111 HW/Lab2/Lab2/alu_top/alu_top_testbench.sv(56)
#    Time: 120 ns  Iteration: 0  Instance: /alu_top_testbench
# 1
# Break in Module alu_top_testbench at C:/Users/user/Desktop/ECE 111 HW/Lab2/Lab2/alu_top/alu_top_testbench.sv line 56
```

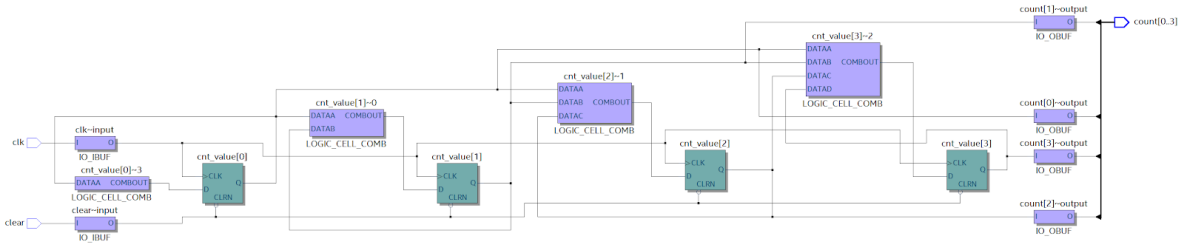
The waveform matches the expected results and operations. The 4 bit ALU decides its operation based on case commands, but this only applies when the clock is high (1). Operation 0 is addition, operation 2 is subtraction, operation 3 is AND function and operation 4 is the OR function; the results are dependent on which of these operations is performed.

Homework 2B:

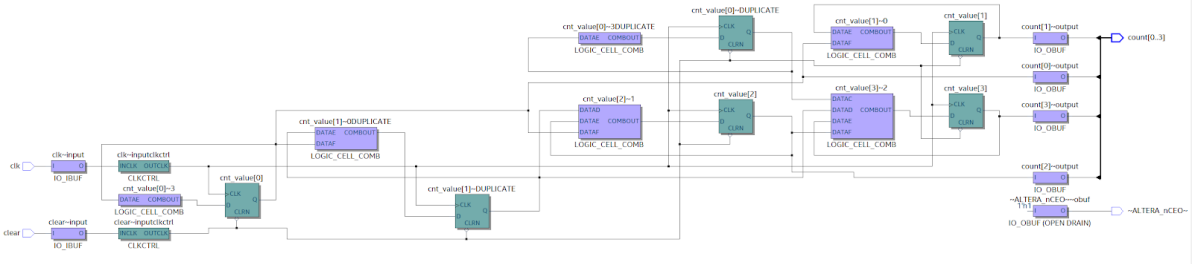
a. counter_4bit RTL



b. counter_4bit post mapping



c. counter _4bit post fitting



d. Testbench code

```
//4-bit counder testbench code
`timescale 1ns/1ns
module counter_4bit_testbench;
logic clock, reset;
logic [3:0] count_value;

// Instantiate design under test
counter_4bit #(.WIDTH(4)) design_instance(
.clk(clock),
.clear(reset),
.count(count_value)
);

initial begin
// Initialize Inputs
reset = 1;
clock = 0;

// Wait 10 ns for global reset to finish and start counter
#10;
reset = 0;

// Wait for 200ns and reset counter
#340ns;
reset=1;

// Wait for 20ns and start counter again
#20ns;
reset=0;

// Wait for 10ns
#100ns;

// terminate simulation
$finish();
end

// Clock generator logic
always@(clock) begin
  #10ns clock <= !clock;
end

// Print input and output signals
initial begin
 $monitor(" time=%0t,  clear=%b  clk=%b  count=%d",$time, reset, clock, count_value);
end
endmodule
```

No changes were made in the testbench

e. Resource usage summary

	Resource	Usage
1	Estimated ALUTs Used	4
1	-- Combinational ALUTs	4
2	-- Memory ALUTs	0
3	-- LUT_REGs	0
2	Dedicated logic registers	4
3		
4	Estimated ALUTs Unavailable	0
1	-- Due to unpartnered combinational logic	0
2	-- Due to Memory ALUTs	0
5		
6	Total combinational functions	4
7	Combinational ALUT usage by number of inputs	
1	-- 7 input functions	0
2	-- 6 input functions	0
3	-- 5 input functions	0
4	-- 4 input functions	1
5	-- <=3 input functions	3
8		
9	Combinational ALUTs by mode	
1	-- normal mode	4
2	-- extended LUT mode	0
3	-- arithmetic mode	0
4	-- shared arithmetic mode	0
10		
11	Estimated ALUT/register pairs used	4
12		
13	Total registers	4
1	-- Dedicated logic registers	4
2	-- I/O registers	0
3	-- LUT_REGs	0
14		
15		
16	I/O pins	6
17		
18	DSP block 18-bit elements	0
19		
20	Maximum fan-out node	cnt_value[0]
21	Maximum fan-out	5
22	Total fan-out	32
23	Average fan-out	1.60

Number of ALUT: 4 (6 I/O pins)

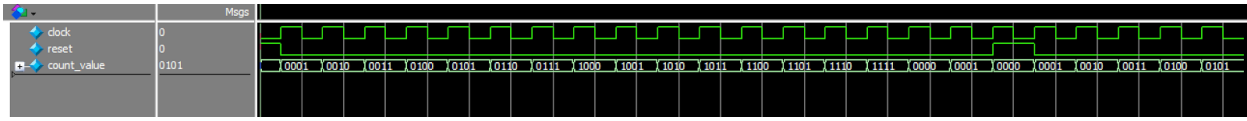
Normal mode: 4

Number of Functions: 4

4 input functions: 1

=< 3 input functions: 3

f. Waveform simulation and transcript



Transcript

```
#
# Top level modules:
#       counter_4bit_testbench
# End time: 01:10:07 on Jan 27,2021, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
ModelSim> vsim work.counter_4bit_testbench
# vsim work.counter_4bit_testbench
# Start time: 01:10:13 on Jan 27,2021
# Loading sv_std.std
# Loading work.counter_4bit_testbench
# Loading work.counter_4bit
add wave -position insertpoint sim:/counter_4bit_testbench/*
VSIM 6> run -all
#  time=0,   clear=1   clk=0   count= 0
#  time=10,  clear=0   clk=1   count= 1
#  time=20,  clear=0   clk=0   count= 1
#  time=30,  clear=0   clk=1   count= 2
#  time=40,  clear=0   clk=0   count= 2
#  time=50,  clear=0   clk=1   count= 3
#  time=60,  clear=0   clk=0   count= 3
#  time=70,  clear=0   clk=1   count= 4
#  time=80,  clear=0   clk=0   count= 4
#  time=90,  clear=0   clk=1   count= 5
#  time=100,  clear=0   clk=0   count= 5
#  time=110,  clear=0   clk=1   count= 6
#  time=120,  clear=0   clk=0   count= 6
#  time=130,  clear=0   clk=1   count= 7
#  time=140,  clear=0   clk=0   count= 7
#  time=150,  clear=0   clk=1   count= 8
#  time=160,  clear=0   clk=0   count= 8
#  time=170,  clear=0   clk=1   count= 9
#  time=180,  clear=0   clk=0   count= 9
#  time=190,  clear=0   clk=1   count=10
#  time=200,  clear=0   clk=0   count=10
#  time=210,  clear=0   clk=1   count=11
```

Counter_4bit explanation:

The output of one counter stage is connected to the clock input of the next counter stage and the counter only counts or increments when the clock is high, which is why the same number remains the same for 20 seconds.