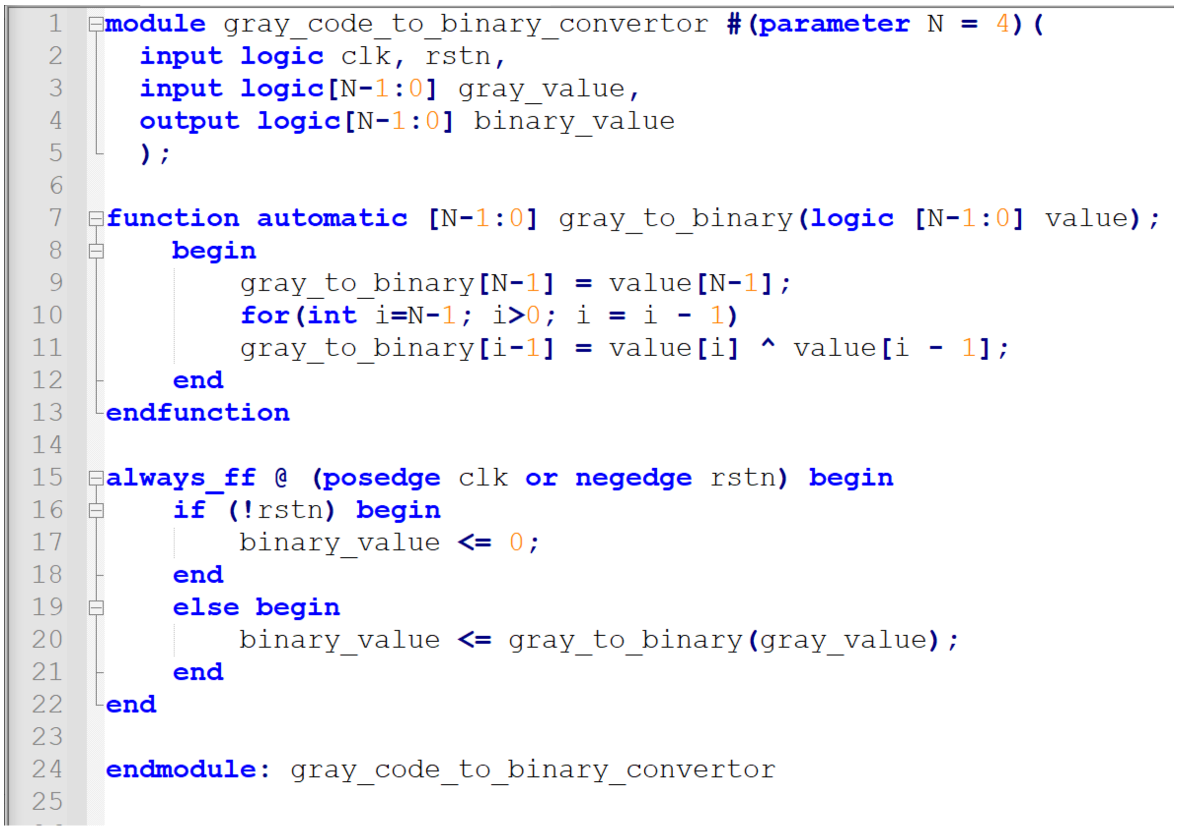
Giao Tran

ECE 111 HW 1 Report

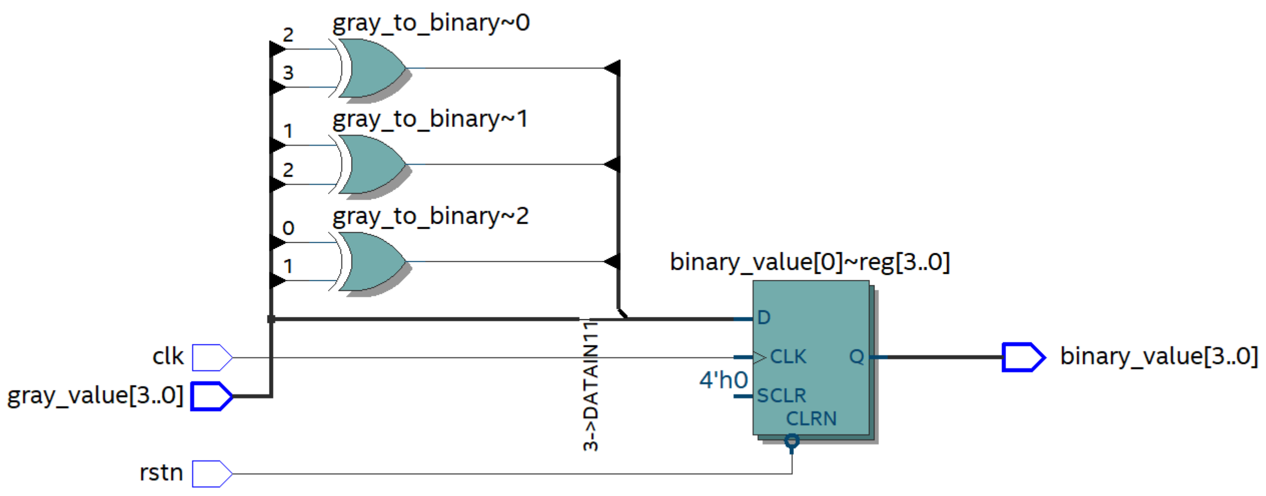
Homework 6a: gray_code_to_binary_converter

a. SystemVerilog RTL model

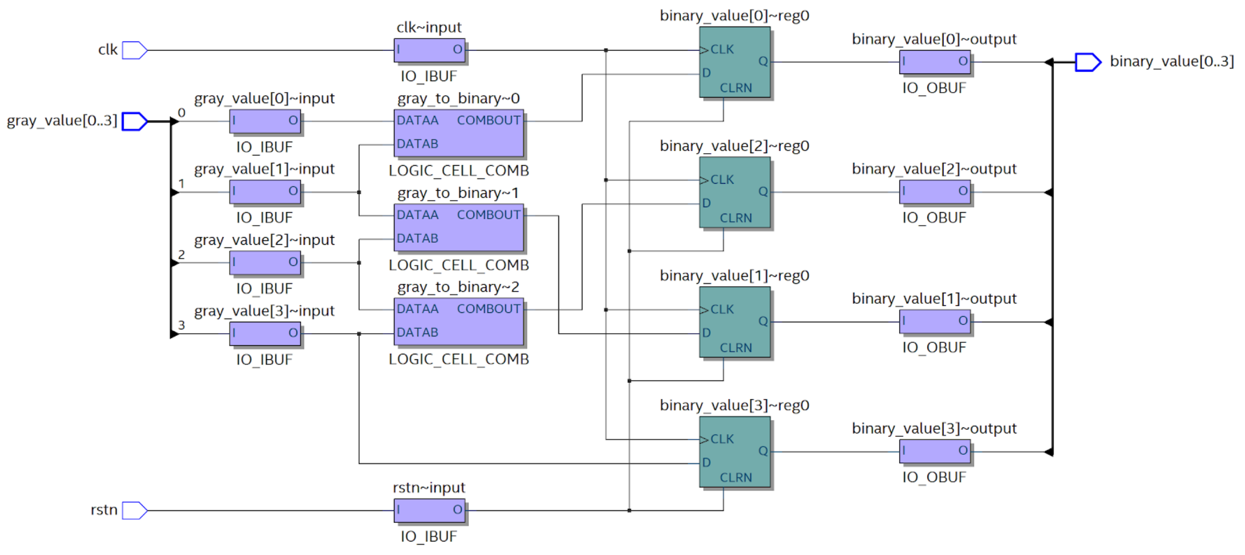


```
module gray_code_to_binary_convertor #(parameter N = 4)(
   input logic clk, rstn,
   input logic[N-1:0] gray_value,
   output logic[N-1:0] binary_value
   );

function automatic [N-1:0] gray_to_binary(logic [N-1:0] value);
     begin
         gray_to_binary[N-1] = value[N-1];
         for(int i=N-1; i>0; i = i - 1)
         gray_to_binary[i-1] = value[i] ^ value[i - 1];
     end
endfunction

always_ff @ (posedge clk or negedge rstn) begin
     if (!rstn) begin
         binary_value <= 0;
     end
     else begin
         binary_value <= gray_to_binary(gray_value);
     end
end

endmodule: gray_code_to_binary_convertor
```

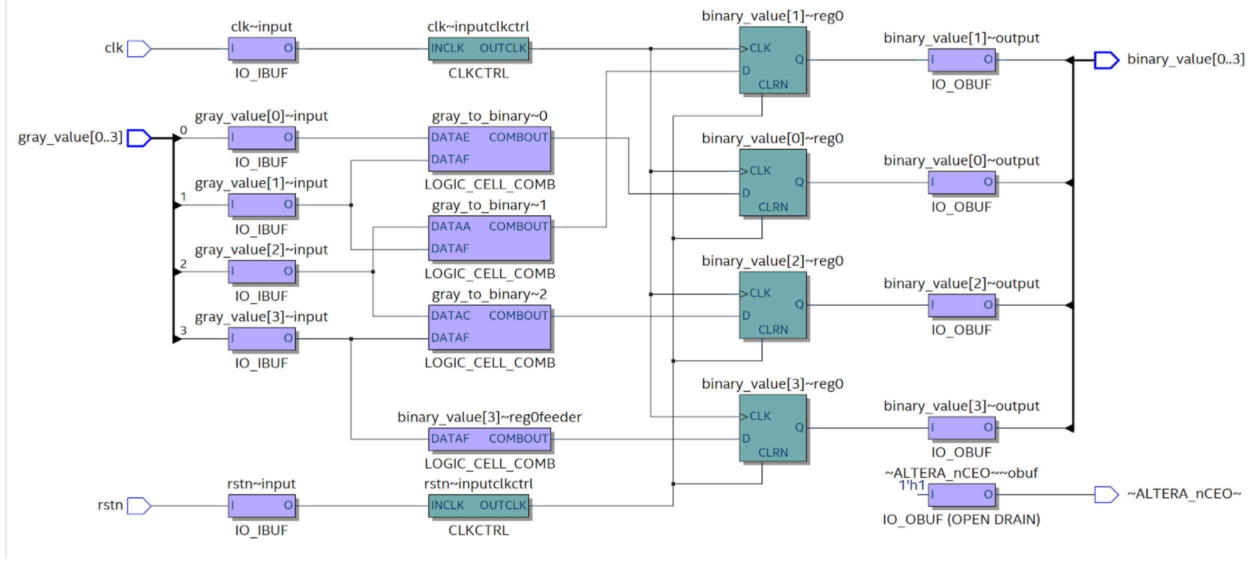
b. RTL schematic



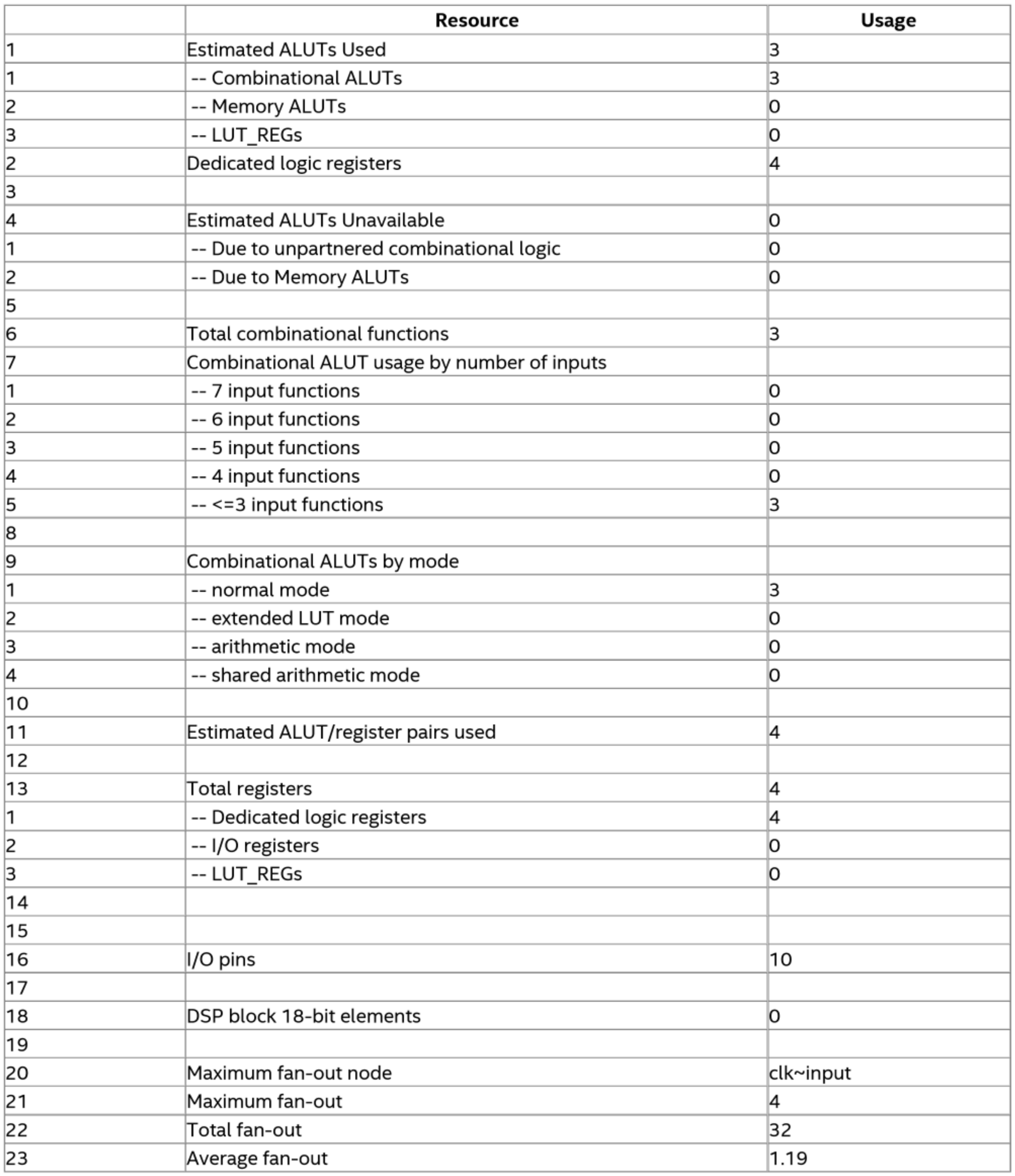
c. Post mapping schematic



d. Post fitting schematic



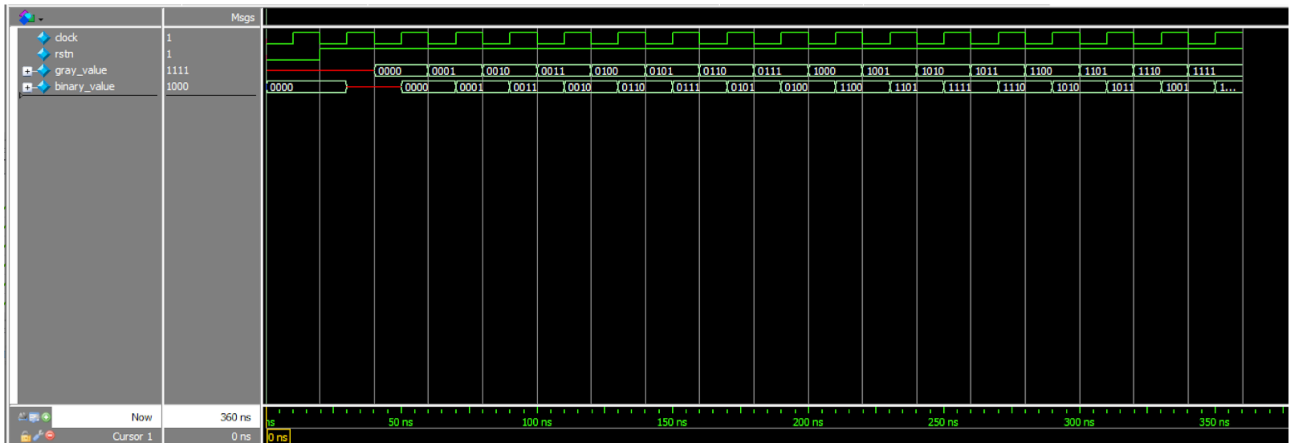
e. Resource Usage table



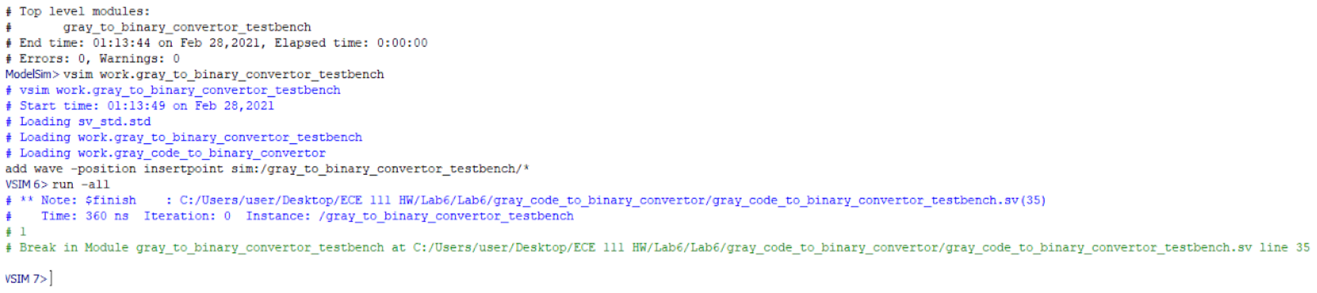
	Resource	Usage
1	Estimated ALUTs Used	3
1	-- Combinational ALUTs	3
2	-- Memory ALUTs	0
3	-- LUT_REGs	0
2	Dedicated logic registers	4
3		
4	Estimated ALUTs Unavailable	0
1	-- Due to unpartnered combinational logic	0
2	-- Due to Memory ALUTs	0
5		
6	Total combinational functions	3
7	Combinational ALUT 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	3
8		
9	Combinational ALUTs by mode	
1	-- normal mode	3
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	10
17		
18	DSP block 18-bit elements	0
19		
20	Maximum fan-out node	clk~input
21	Maximum fan-out	4
22	Total fan-out	32
23	Average fan-out	1.19

Number of ALUT: 3 (10 I/O pins)
Number of Functions: 3 (3 input functions)

f. Waveform simulation

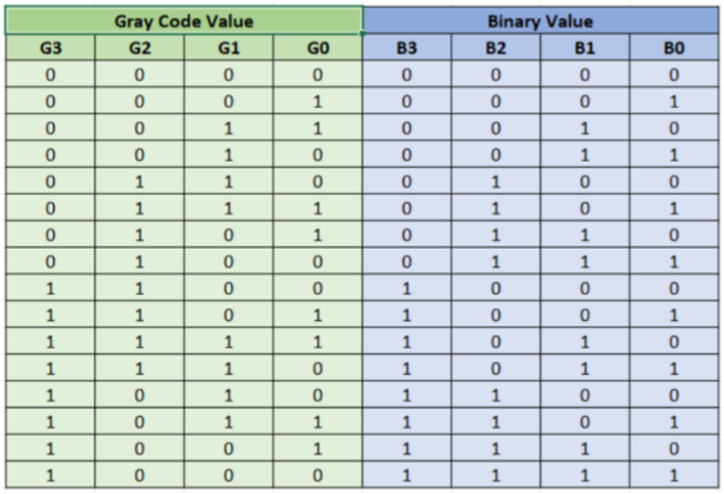


g. Transcript and explanations

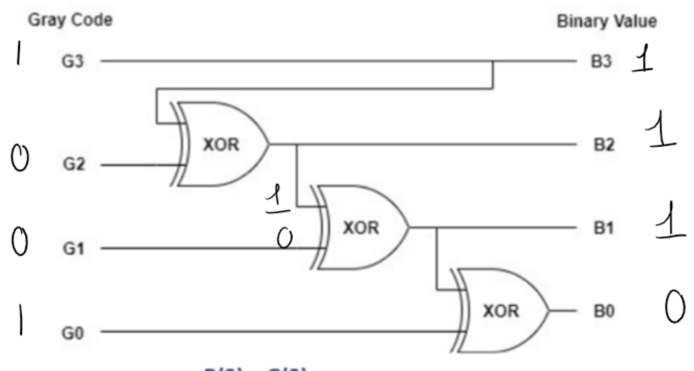


```
# Top level modules:
#       gray_to_binary_convertor_testbench
# End time: 01:13:44 on Feb 28,2021, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
ModelSim> vsim work.gray_to_binary_convertor_testbench
# vsim work.gray_to_binary_convertor_testbench
# Start time: 01:13:49 on Feb 28,2021
# Loading sv_std.std
# Loading work.gray_to_binary_convertor_testbench
# Loading work.gray_code_to_binary_convertor
add wave -position insertpoint sim:/gray_to_binary_convertor_testbench/*
VSIM 6> run -all
# ** Note: $finish    : C:/Users/user/Desktop/ECE 111 HW/Lab6/Lab6/gray_code_to_binary_convertor/gray_code_to_binary_convertor_testbench.sv(35)
#    Time: 360 ns  Iteration: 0  Instance: /gray_to_binary_convertor_testbench
# 1
# Break in Module gray_to_binary_convertor_testbench at C:/Users/user/Desktop/ECE 111 HW/Lab6/Lab6/gray_code_to_binary_convertor/gray_code_to_binary_convertor_testbench.sv line 35

VSIM 7>]
```

The waveform simulation matches the gray code value to binary value table and diagram provided below. Depending on the N value there are N-1 XOR gate, where each output, except for the N-1 output feeds into the input of the next XOR gate.

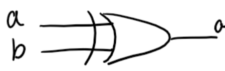


Gray Code Value				Binary Value			
G3	G2	G1	G0	B3	B2	B1	B0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	1	0	1	0
1	1	1	0	1	0	1	1
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	1
1	0	0	1	1	1	1	0
1	0	0	0	1	1	1	1



B(3) = G(3)
B(2) = G(2) xor B(3)
B(1) = G(1) xor B(2)
B(0) = G(0) xor B(1)

XOR gate



out $= a \oplus b = a\bar{b} + \bar{a}b$

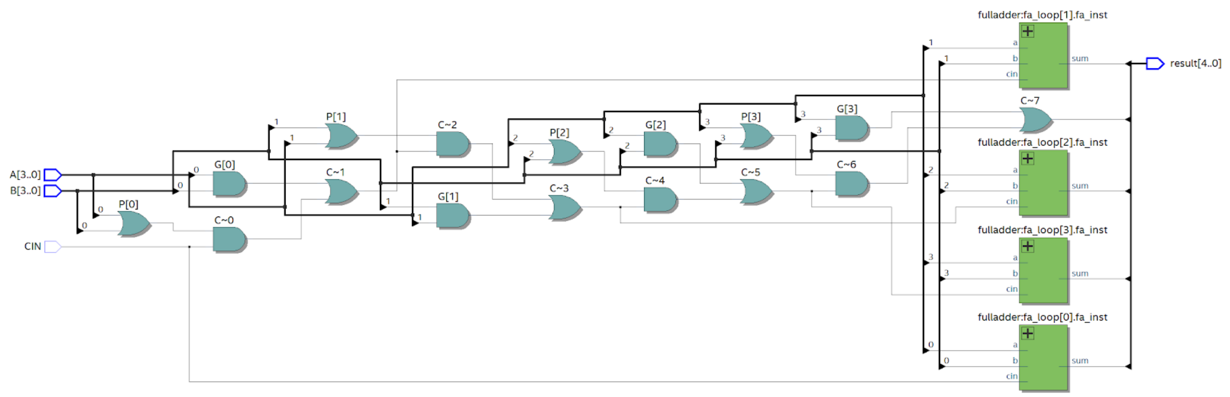
a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

Homework 6b: carry_lookahead_adder

a. SystemVerilog RTL model

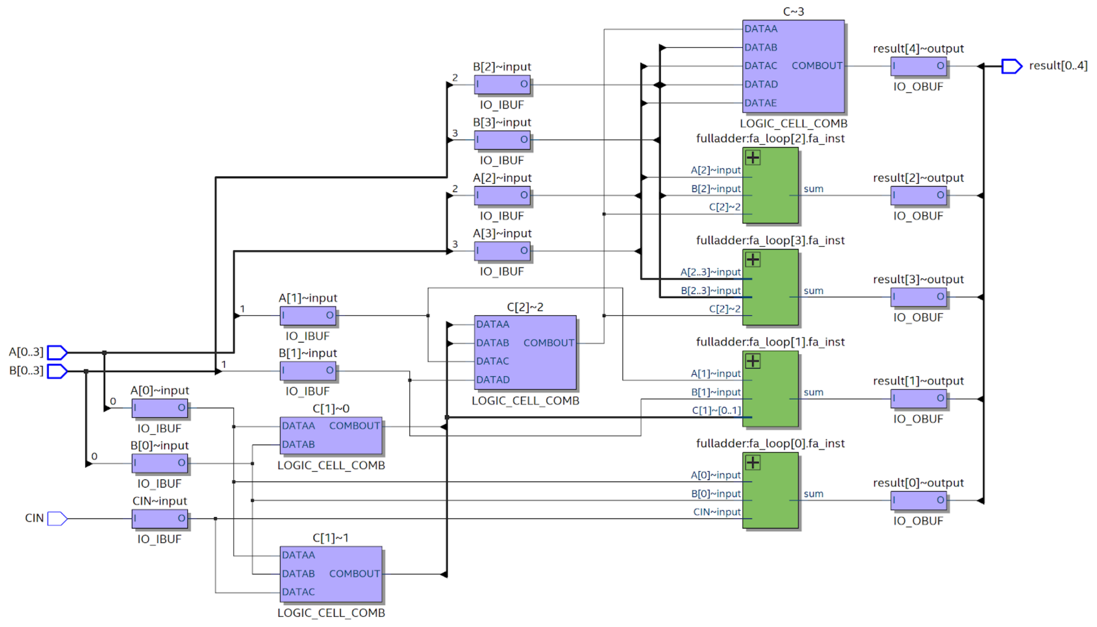
```
`include "fulladder.sv"
module carry_lookahead_adder#(parameter N=4)
(
  input logic[N-1:0] A, B,
  input logic CIN,
  output logic[N:0] result
);
logic [3:0] G, P, S;
logic [N:0] C;

assign C[0]=CIN;

genvar i;
generate
    for(i=0; i<N; i=i+1)
        begin: fa_loop
            fulladder fa_inst(.a(A[i]),.b(B[i]),.cin(C[i]),.sum(S[i]),.cout());
        end: fa_loop
endgenerate

genvar j;
generate
    for (j=0; j<N; j=j+1)
        begin: gen_and_prop_loop
            assign G[j] = A[j] & B[j];
            assign P[j] = A[j] | B[j];
            assign C[j+1] = G[j] | (P[j] & C[j]);
        end
endgenerate

assign result = {C[4], S};

endmodule:carry_lookahead_adder

/*
alternatively

fulladder inst0 (.a(A[0]),.b(B[0]),.cin(C[0]),.sum(S[0]),.cout());
fulladder inst1 (.a(A[1]),.b(B[1]),.cin(C[1]),.sum(S[1]),.cout());
fulladder inst2 (.a(A[2]),.b(B[2]),.cin(C[2]),.sum(S[2]),.cout());
fulladder inst3 (.a(A[3]),.b(B[3]),.cin(C[3]),.sum(S[3]),.cout());

  // Gi=A.B
  assign G[0] = A[0] & B[0];
  assign G[1] = A[1] & B[1];
  assign G[2] = A[2] & B[2];
  assign G[3] = A[3] & B[3];

  // Pi=A+B
  assign P[0] = A[0] | B[0];
  assign P[1] = A[1] | B[1];
  assign P[2] = A[2] | B[2];
  assign P[3] = A[3] | B[3];

  // Carry
  assign C[0] = CIN;
  assign C[1] = G[0] | (P[0] & C[0]);
  assign C[2] = G[1] | (P[1] & C[1]);
  assign C[3] = G[2] | (P[2] & C[2]);
  assign C[4] = G[3] | (P[3] & C[3]);

  assign result = {C[4], S};
 */
```

b. RTL schematic

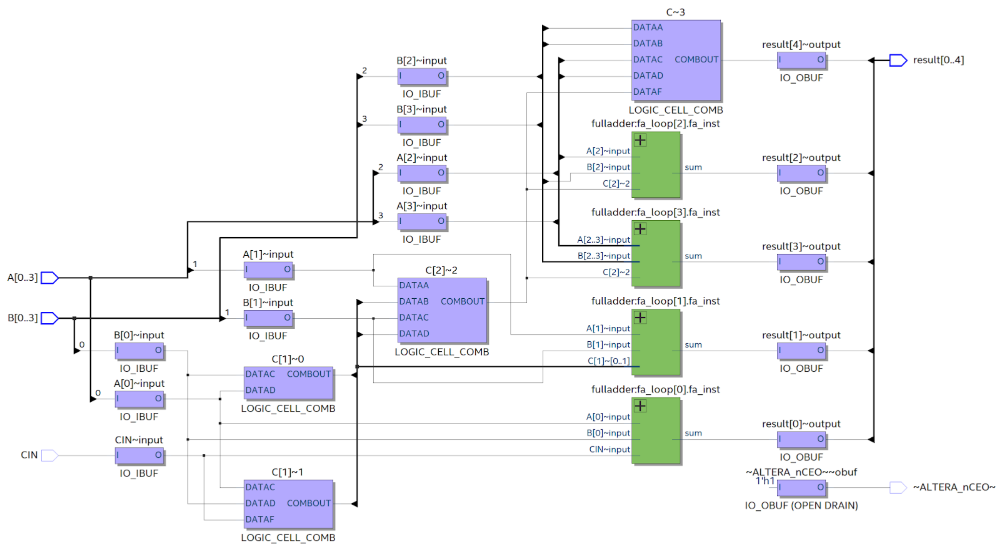


fulladder:fa_loop[1].fa_inst
fulladder:fa_loop[2].fa_inst
fulladder:fa_loop[3].fa_inst
fulladder:fa_loop[0].fa_inst
result[4..0]
A[3..0]
B[3..0]
CIN
P[0]
P[1]
P[2]
P[3]
G[0]
G[1]
G[2]
G[3]
C~0
C~1
C~2
C~3
C~4
C~5
C~6
C~7
a
b
cin
sum

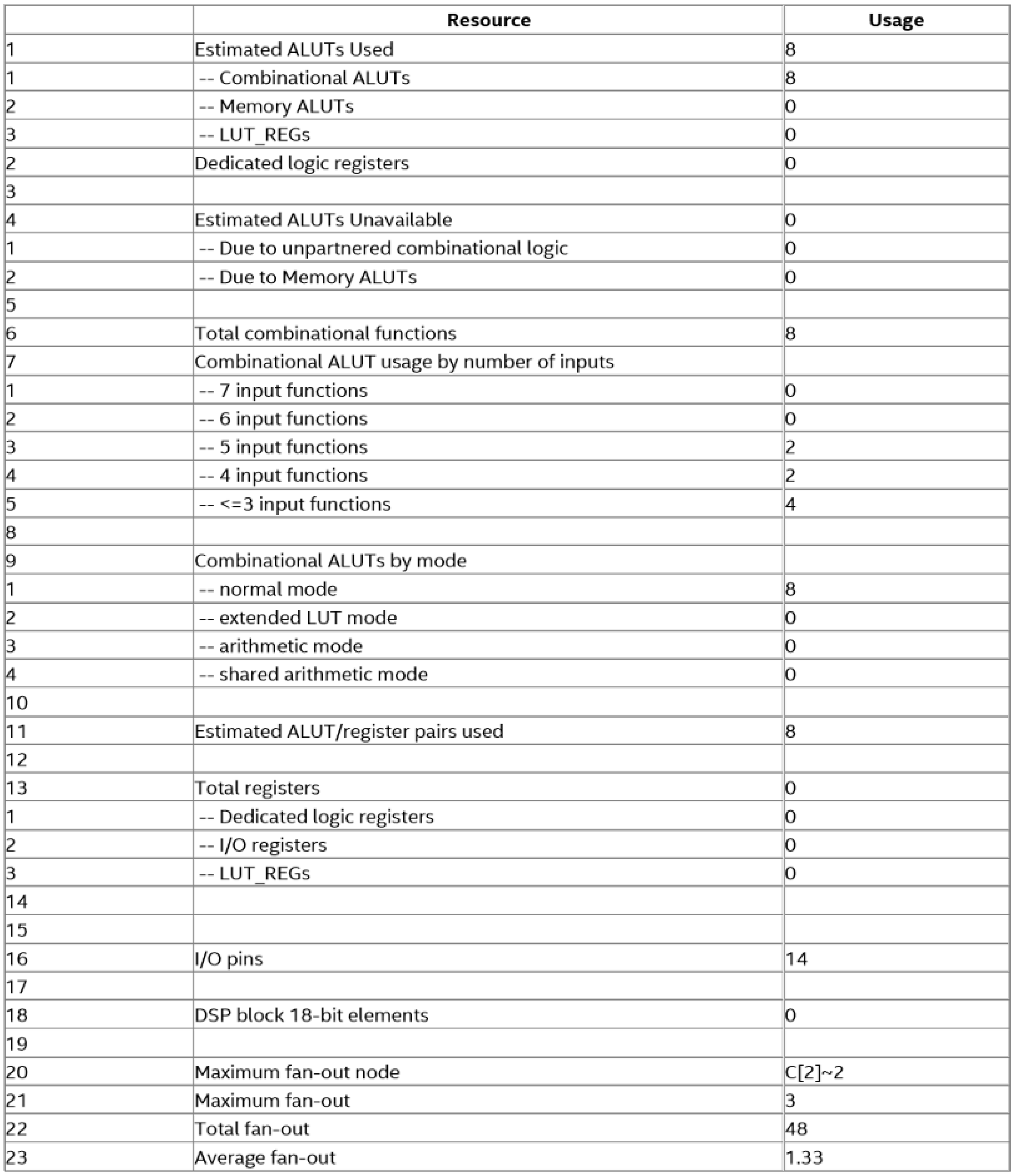
c. Post mapping schematic



d. Post fitting schematic



e. Resource Usage table

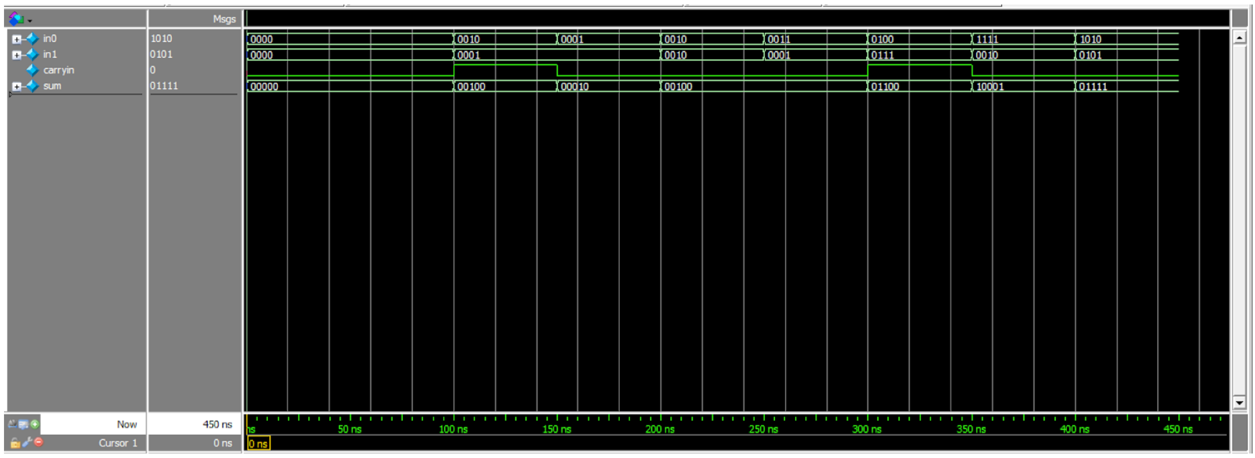


	Resource	Usage
1	Estimated ALUTs Used	8
1	-- Combinational ALUTs	8
2	-- Memory ALUTs	0
3	-- LUT_REGs	0
2	Dedicated logic registers	0
3		
4	Estimated ALUTs Unavailable	0
1	-- Due to unpartnered combinational logic	0
2	-- Due to Memory ALUTs	0
5		
6	Total combinational functions	8
7	Combinational ALUT usage by number of inputs	
1	-- 7 input functions	0
2	-- 6 input functions	0
3	-- 5 input functions	2
4	-- 4 input functions	2
5	-- <=3 input functions	4
8		
9	Combinational ALUTs by mode	
1	-- normal mode	8
2	-- extended LUT mode	0
3	-- arithmetic mode	0
4	-- shared arithmetic mode	0
10		
11	Estimated ALUT/register pairs used	8
12		
13	Total registers	0
1	-- Dedicated logic registers	0
2	-- I/O registers	0
3	-- LUT_REGs	0
14		
15		
16	I/O pins	14
17		
18	DSP block 18-bit elements	0
19		
20	Maximum fan-out node	C[2]~2
21	Maximum fan-out	3
22	Total fan-out	48
23	Average fan-out	1.33

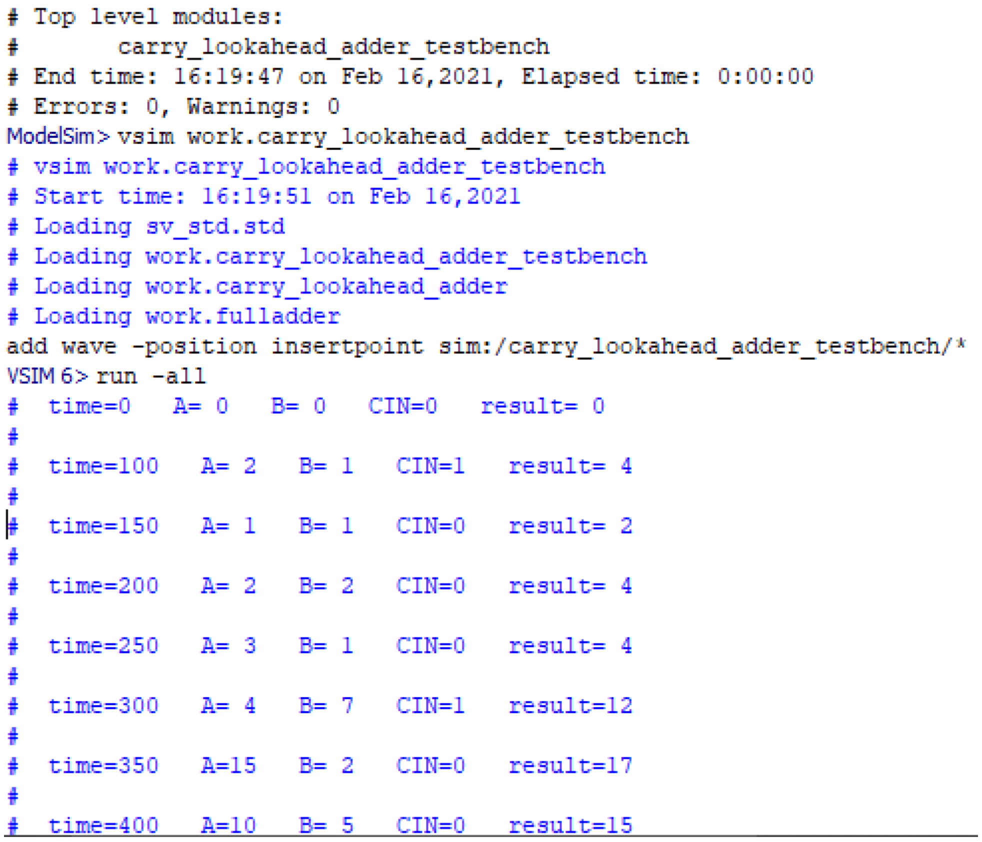
Number of ALUT: 8 (14 I/O pins)
Number of Functions: 3

2 5 input functions
2 4 input functions
4 3 input functions

f. Waveform simulation



g. Transcript and explanations



```
# Top level modules:
#       carry_lookahead_adder_testbench
# End time: 16:19:47 on Feb 16,2021, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
ModelSim> vsim work.carry_lookahead_adder_testbench
# vsim work.carry_lookahead_adder_testbench
# Start time: 16:19:51 on Feb 16,2021
# Loading sv_std.std
# Loading work.carry_lookahead_adder_testbench
# Loading work.carry_lookahead_adder
# Loading work.fulladder
add wave -position insertpoint sim:/carry_lookahead_adder_testbench/*
VSIM 6> run -all
#  time=0   A= 0   B= 0   CIN=0   result= 0
#
#  time=100   A= 2   B= 1   CIN=1   result= 4
#
#  time=150   A= 1   B= 1   CIN=0   result= 2
#
#  time=200   A= 2   B= 2   CIN=0   result= 4
#
#  time=250   A= 3   B= 1   CIN=0   result= 4
#
#  time=300   A= 4   B= 7   CIN=1   result=12
#
#  time=350   A=15   B= 2   CIN=0   result=17
#
#  time=400   A=10   B= 5   CIN=0   result=15
```

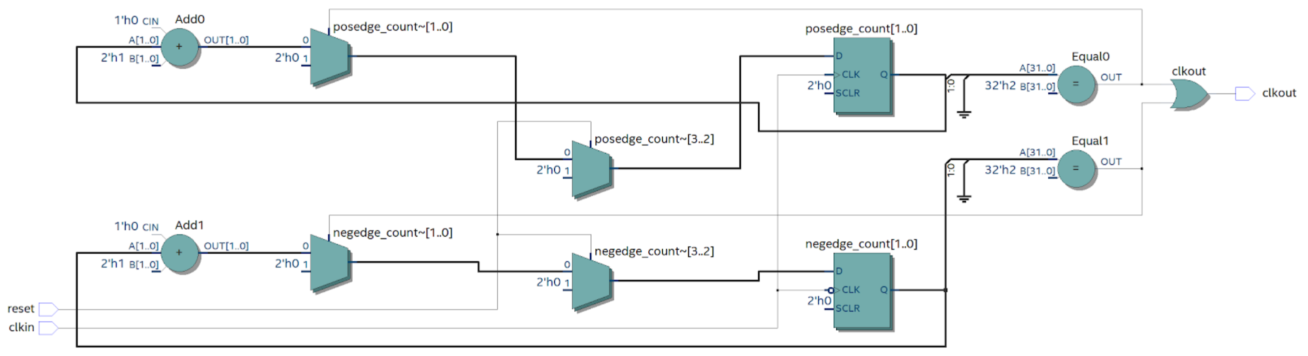
The N=4 bits carry lookahead adder implements the full adder module with procedure to generate and propagate cary. The output result is Result = A+B+carryin, which matches the values shown in the transcript above.

Homework 6c: clock_divide_by_3

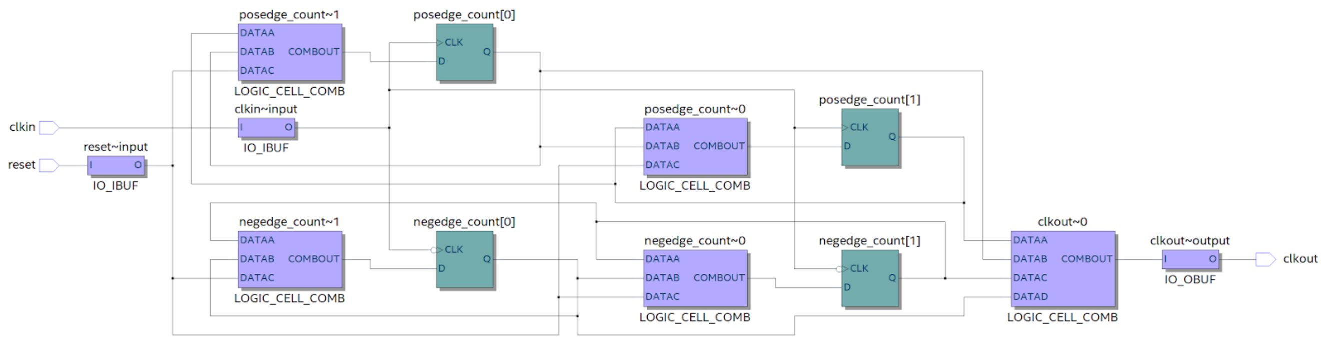
a. SystemVerilog RTL model

```
//clock divide by 3 RTL code
module clock_divide_by_3 (
 input  logic clkin, reset,
 output logic clkout);

logic [1:0] posedge_count, negedge_count;

//posedge clock counter, using 2nd implementation
always_ff @ (posedge clkin) begin
    if (reset==1)
        posedge_count<=0;

    else if (posedge_count==2)
        posedge_count<=0;
    else
        posedge_count<=posedge_count+1;
end

always_ff @ ( negedge clkin) begin
    if (reset==1)
        negedge_count<=0;

    else if (negedge_count==2)
        negedge_count<=0;
    else
        negedge_count<=negedge_count+1;
end

assign clkout=((posedge_count==2)|(negedge_count==2));

endmodule: clock_divide_by_3
```

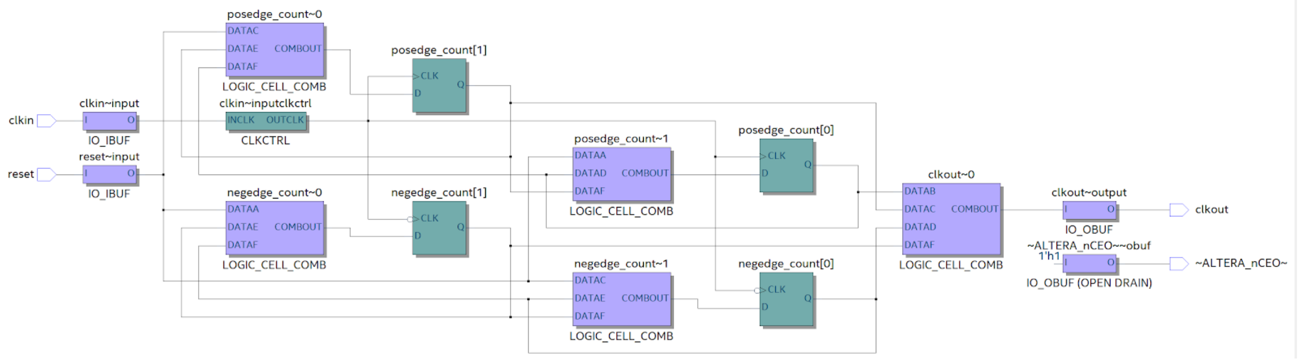
b. RTL schematic



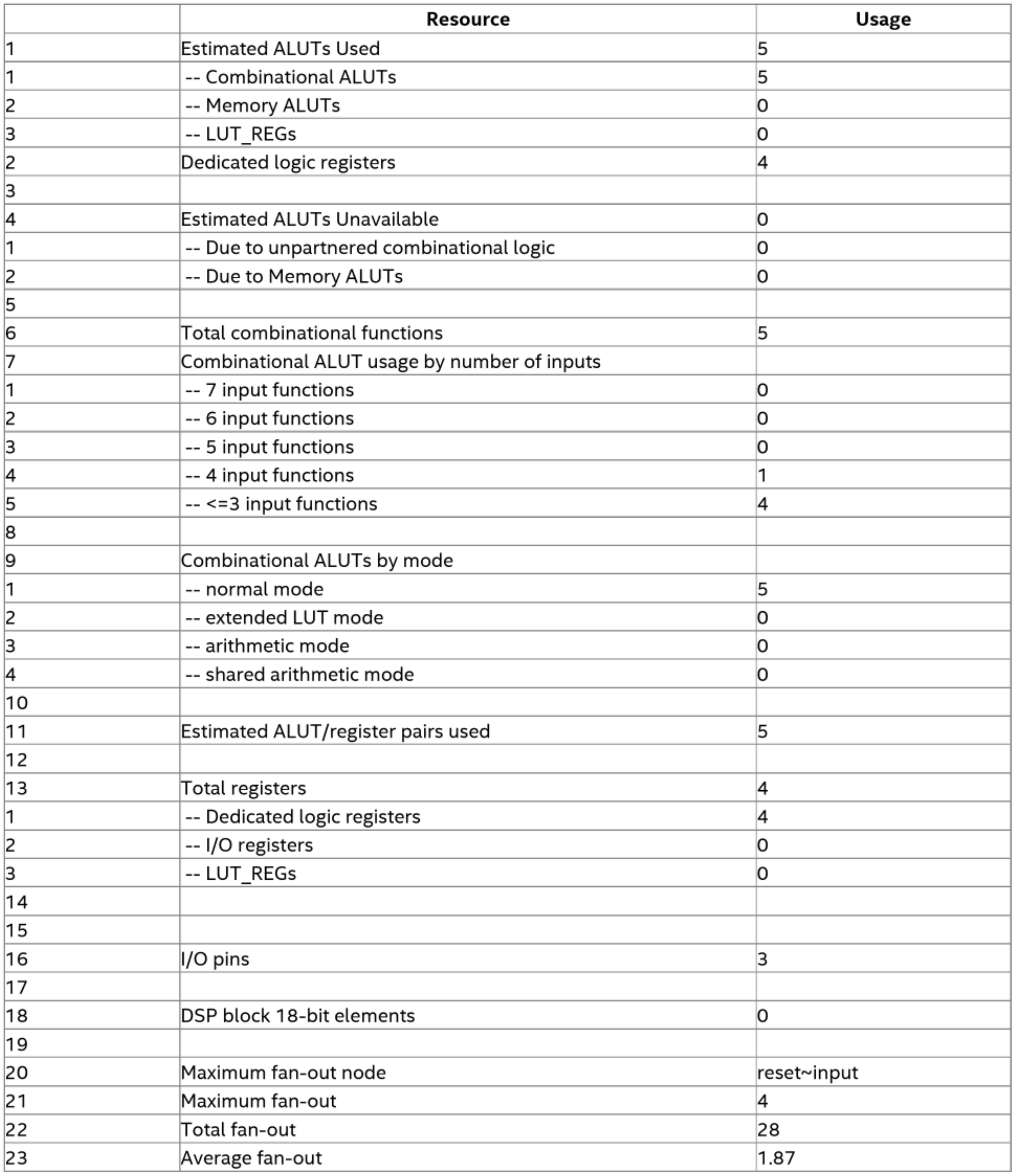
c. Post mapping schematic



d. Post fitting schematic



e. Resource Usage table

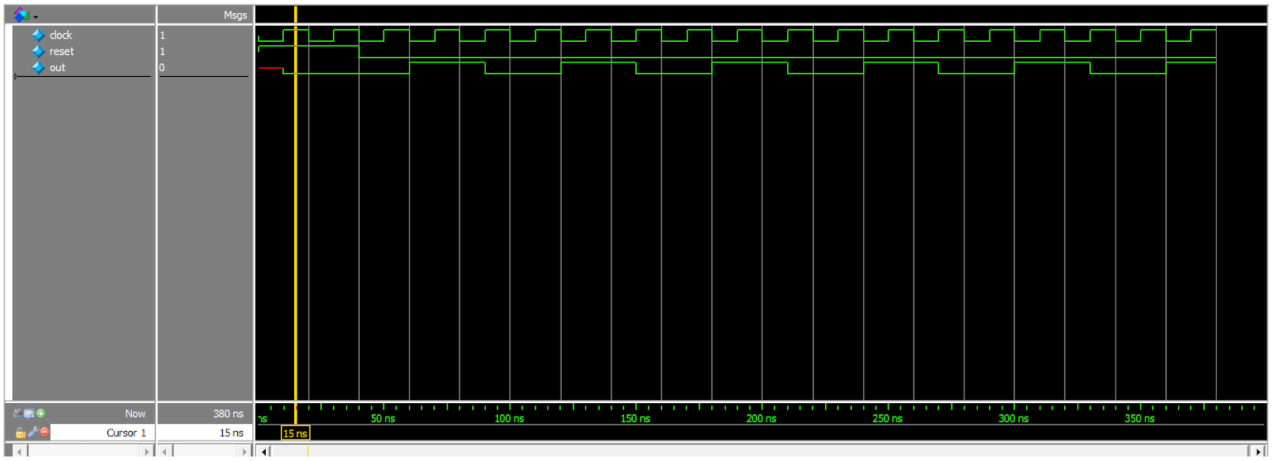


	Resource	Usage
1	Estimated ALUTs Used	5
1	-- Combinational ALUTs	5
2	-- Memory ALUTs	0
3	-- LUT_REGs	0
2	Dedicated logic registers	4
3		
4	Estimated ALUTs Unavailable	0
1	-- Due to unpartnered combinational logic	0
2	-- Due to Memory ALUTs	0
5		
6	Total combinational functions	5
7	Combinational ALUT 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	4
8		
9	Combinational ALUTs by mode	
1	-- normal mode	5
2	-- extended LUT mode	0
3	-- arithmetic mode	0
4	-- shared arithmetic mode	0
10		
11	Estimated ALUT/register pairs used	5
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	3
17		
18	DSP block 18-bit elements	0
19		
20	Maximum fan-out node	reset~input
21	Maximum fan-out	4
22	Total fan-out	28
23	Average fan-out	1.87

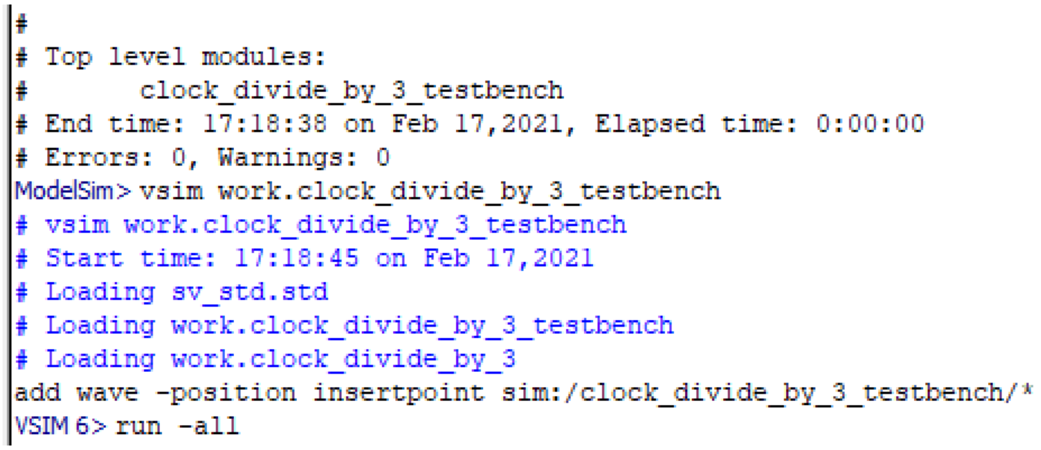
Number of ALUT: 5 (3 I/O pins)
Number of Functions: 5

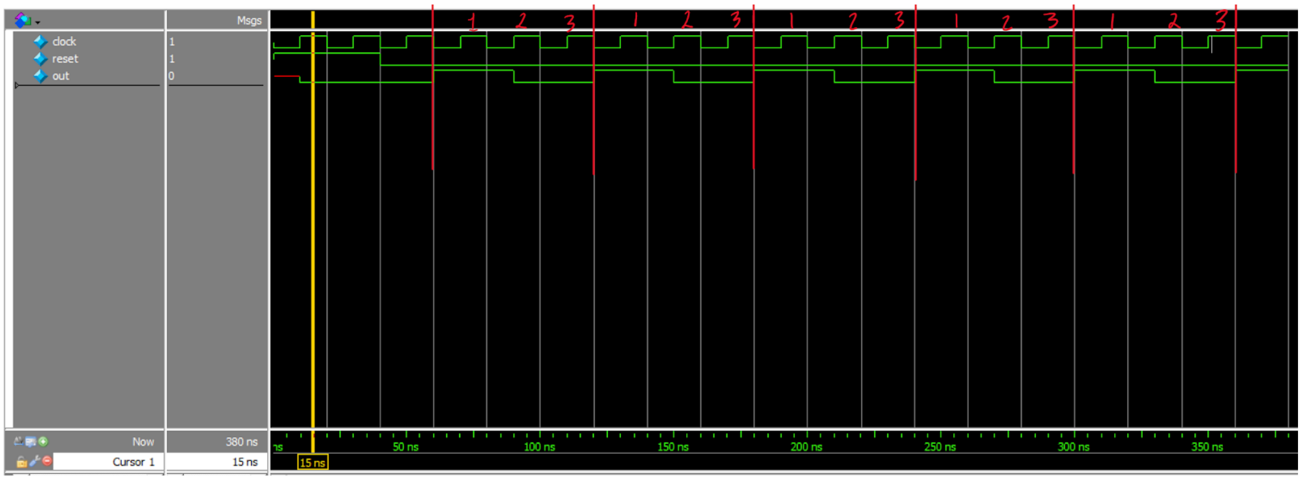
1 5 input functions
4 3 input functions

f. Simulation waveform



g. Transcript


```
#
# Top level modules:
#       clock_divide_by_3_testbench
# End time: 17:18:38 on Feb 17,2021, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
ModelSim> vsim work.clock_divide_by_3_testbench
# vsim work.clock_divide_by_3_testbench
# Start time: 17:18:45 on Feb 17,2021
# Loading sv_std.std
# Loading work.clock_divide_by_3_testbench
# Loading work.clock_divide_by_3
add wave -position insertpoint sim:/clock_divide_by_3_testbench/*
VSIM 6> run -all
```



We can see that the output is 3 times the input clock cycle, and repeats every 3 clock cycles. The clk output is from the outputs of two flipflops which were determined by the equation $2^n > 3$.