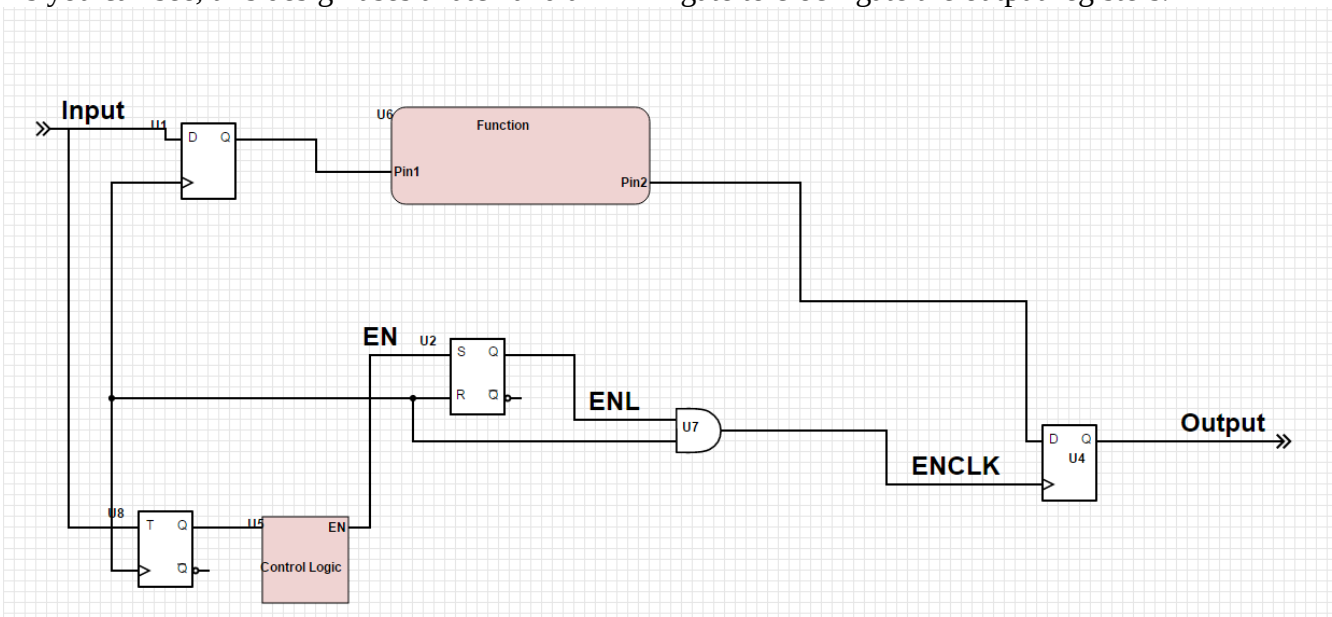
Cory Snooks
Aaron Pederson
EE 465 Lab 9
11/11/2015

Lab 9 Report

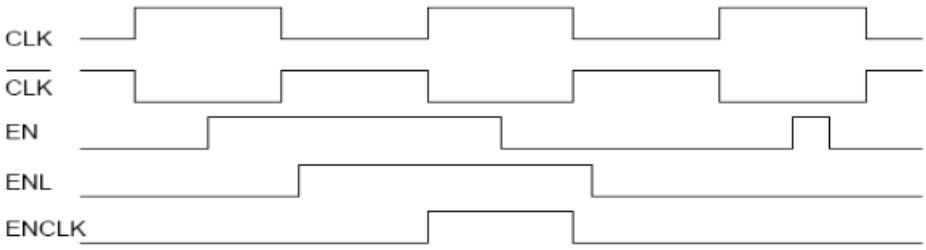
The task of this lab is to reduce the power consumption of the circuit. In order to do this, we will eliminate any unnecessary switching of registers or gates. There are many possible registers that may be clock gated, but we will only do this for the output registers for time purposes.

1 DFF-based output clock gating

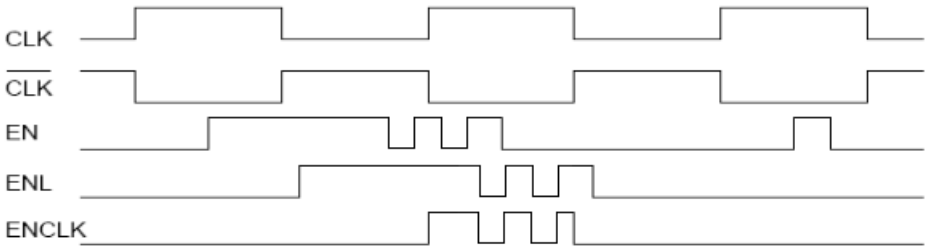
As you can see, this design uses a latch and an AND gate to clock-gate the output registers.



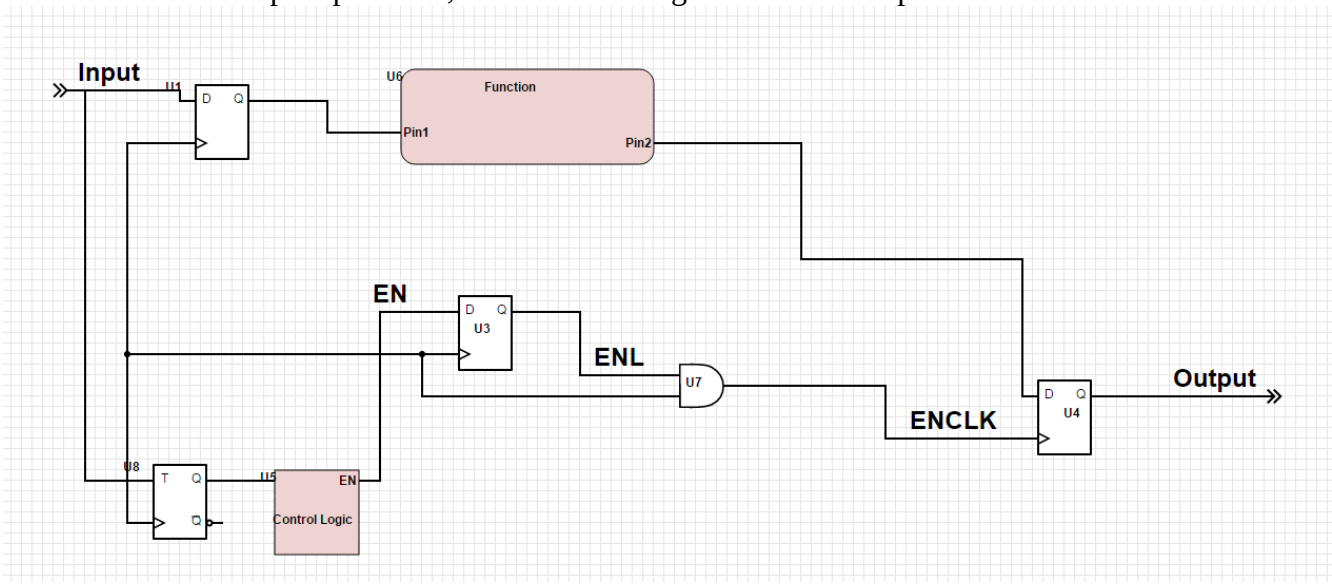
It will produce an output like this..



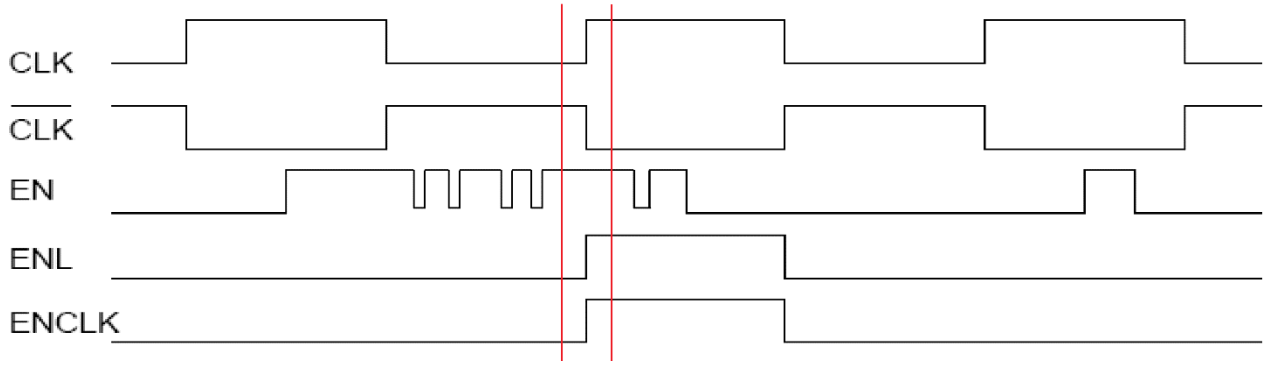
But if glitches are present in the control logic signal, there will be major problems as shown below with multiple clock edges at the output:



So.. If we use a D-flip-flop instead, there will be no glitches at the output.



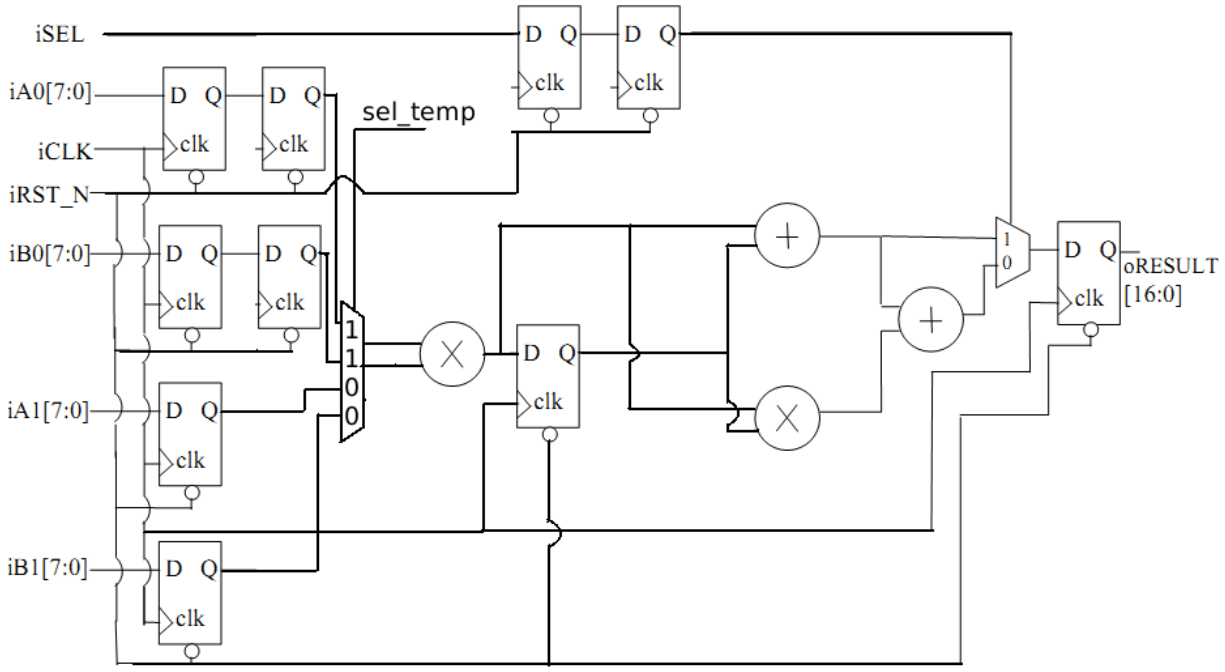
You can see that as long as the enable signal is high at the positive clock edge, the circuit will function properly.



2/3 Power-Efficient Design changes.

Before Change:

The schematic diagram for the circuit before our changes for power efficiency is below:



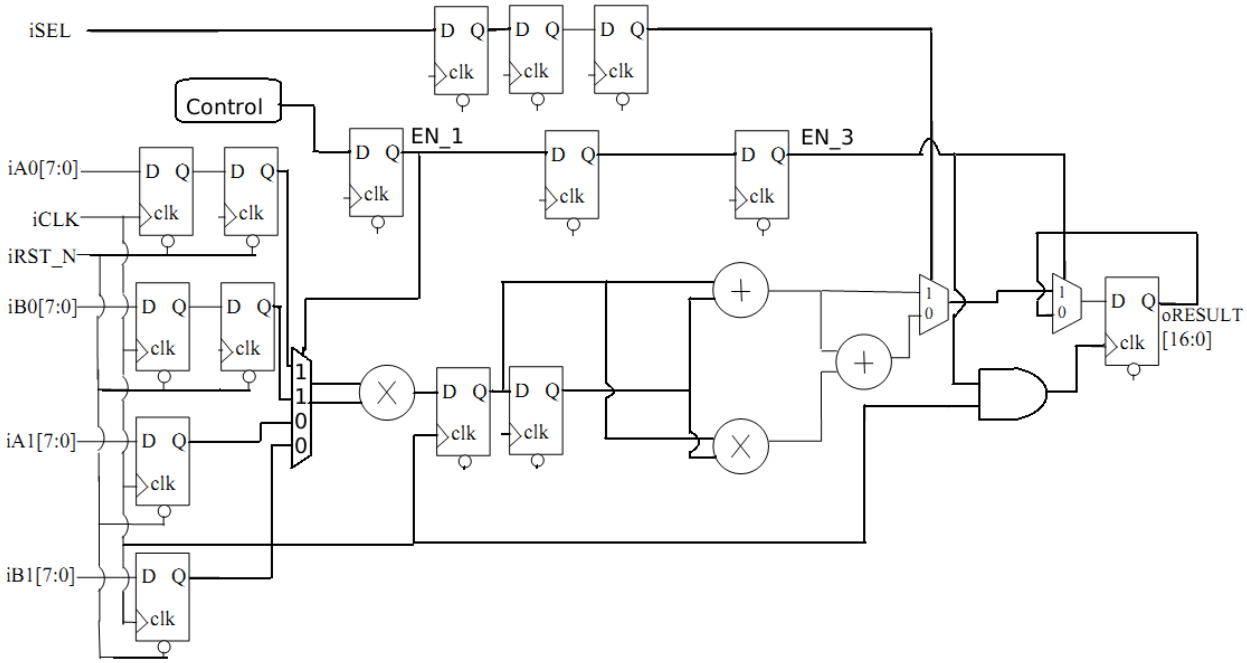

```
module ALT_MULTADD_re(iCLK, iRST_N, iSEL, iSEL2,  iA0, iA1, iB0, iB1, oR);
input iCLK, iRST_N, iSEL, iSEL2, iA0, iA1, iB0, iB1;
output oR;

wire iCLK, iRST_N, iSEL, iSEL2;
wire [7:0] iA0, iA1, iB0, iB1;
reg SEL1, SEL2, sel_tmp;
reg [7:0] A0, A1, B0, B1, A01, B01, AxB;
reg [16:0] oR;

always @ (posedge iCLK or negedge iRST_N)begin
A0 <= iA0;
B0 <= iB0;

A1 <= iA1;
B1 <= iB1;
A01 <= A0;
B01 <= B0;
SEL1 <= iSEL;
SEL2 <= SEL1;
sel_tmp <= iSEL2;

  if(iRST_N)begin
    if(~sel_tmp)begin
      AxB <= A01 * B01;
    end
    else begin
      AxB <= A1 * B1;
```

```
    end

    if(SEL2) begin
      oR <= AxB + A1 * B1;
    end
    else begin
      oR <= AxB + A1 * B1 + AxB * A1 * B1;
    end
  end
  else begin
    sel_tmp <= 0;
    oR <= 0;
  end
end


endmodule
```

With this version of my code, there is no control logic, so the iSEL_TMP signal serves as the indicator that a new input is present and must be a one clock cycle pulse. I intend to create control logic to create this signal automatically depending on the input signal edges.

After Change:

I decided to re-write the Verilog code to make it easier for the compiler to know to exactly construct the circuit.
I decided to re-introduce the internal control logic that would be triggered by edges on any of the input signals iA0,iA1,iB0,iB1 and iSEL. This would make the control logic for the enable signal for the clock gate much easier.
I also added the optional d-flip flop after the first multiplier to fully pipeline the circuit.



As you can see, I changed the clock frequency to 5MHz by setting the timescale to 100ns which is half the clock cycle. 1s/200ns = 5MHz.

```
`timescale 100ns/1ns
module ALT_MULTADD_9(iCLK, iRST_N, iSEL, iA0, iA1, iB0, iB1, oRESULT);
input iCLK, iRST_N, iSEL, iA0, iA1, iB0, iB1;
output oRESULT;

wire iCLK, iRST_N, iSEL;
wire [7:0] iA0, iA1, iB0, iB1;
reg SEL1, SEL2, SEL3, EN, EN_1, EN_2, EN_3;
reg [7:0] A0, A1, B0, B1, A01, B01, A, B;
reg [16:0] oR, AB, oRESULT, AB1_FF, AB0_FF;
reg go, no, no_FF, go_FF;

always @ (posedge iCLK)begin
if(~iRST_N)begin

  A0 <= 0;
  B0 <= 0;
  A1 <= 0;
  B1 <= 0;
  A01 <= 0;
  B01 <= 0;
```

```
    SEL1 <= 0;
    SEL2 <= 0;
    SEL3 <= 0;

    EN = 0;
    EN_1 <= 0;
    EN_2 <= 0;
    EN_3 <= 0;

    AB = 0;
    AB1_FF <= 0;
    AB0_FF <= 0;

    go_FF <= 0;
    no_FF <= 0;
    go = 0;
    no = 0;

    oR = 0;
    oRESULT <= 0;
  end
  else begin
    //Input registers
    A0 <= iA0;
    B0 <= iB0;
    A1 <= iA1;
    B1 <= iB1;
    A01 <= A0;
    B01 <= B0;

    //Multiplied signal registers
    AB1_FF <= AB;
    AB0_FF <= AB1_FF;

    //Select signal registers
    SEL1 <= iSEL;
    SEL2 <= SEL1;
    SEL3 <= SEL2;

    //enable(multiplexer select) signal registers
    EN_1 <= EN;
    EN_2 <= EN_1;
    EN_3 <= EN_2;

    //control logic stuff
    go_FF <= go;
    no_FF <= no;

    if(EN_3) begin
      oRESULT <= oR; //clock gate
    end
    //output register

  end
  end

  //Select Control Logic
  always@ (go_FF or EN_1 or no_FF)  begin  //
```

```
  if(go_FF ^ no_FF) begin   //whenever no_FF and go_FF are different, that means
there is a new input.
    EN = 1;
    no = ~no_FF;
  end
  if(EN_1)begin
    EN = 0;
  end
end


// part of control logic
always @ (iA0 or iB0 or iA1 or iB1 or iSEL) begin
    go = ~go_FF;
end

//first stage of logic (multiplier)
always@ (EN_2 or A01 or B01 or A1 or B1)  begin  //
    if(EN_2)begin  // this is the "multiplexer" for the multiplier
      A = A01;
      B = B01;
    end
    else begin
      A = A1;
      B = B1;
    end
    AB = A * B;
end


// final stage of logic
always@ (A01 or B01 or A1 or B1 or AB1_FF or AB0_FF)  begin  //
      if(SEL3) begin   // this is the "multiplexer" 1 input of interest.
        oR = AB0_FF + AB1_FF;
      end
    else begin   // this is the "multiplexer" 0 input of interest.
        oR = AB0_FF + AB1_FF + AB0_FF * AB1_FF;
      end
    end
endmodule
```

We implemented the control logic with an always block triggered by changes in the inputs. We made 3 one bit registers go, no and count. When go and no are different, which will happen when the input changes because of the always block that inverts go, count will be set to 1 as well as no being inverted as well.

4 Verify

We made this test bench which would test the throughput of the circuit and many different inputs: After the initial input, we delayed the next input by #4 which means 2 clock cycles. We were able to get correct outputs with this so this means the expected maximum throughput is verified.

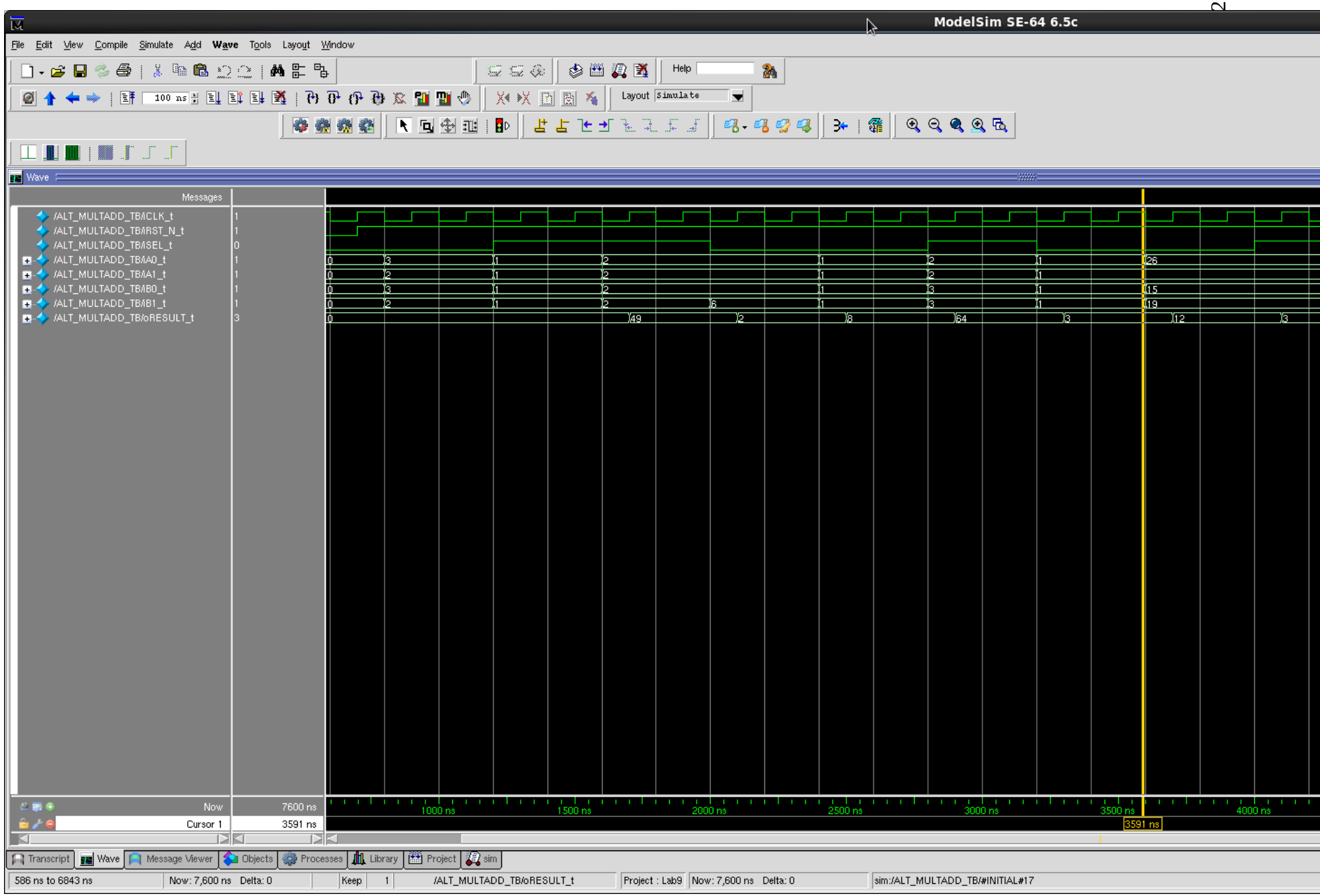
```
`timescale 100ns/1ns
module ALT_MULTADD_TB ();


  reg iCLK_t, iRST_N_t, iSEL_t;
  reg [7:0] iA0_t, iA1_t, iB0_t, iB1_t;
  wire [16:0] oRESULT_t;

  ALT_MULTADD_9 X(iCLK_t, iRST_N_t, iSEL_t, iA0_t, iA1_t, iB0_t, iB1_t, oRESULT_t);
  initial $display ("Test control");
    initial $display ("%10s %10s %10s %10s %10s %10s %10s %10s",
      "iCLK_t", "iRST_N_t", "iSEL_t", "iA0_t", "iA1_t",
      "iB0_t", "iB1_t", "oRESULT_t");
   initial $monitor ("%10b %10b %10b %10d %10d %10d %10d %10d",
      iCLK_t, iRST_N_t, iSEL_t,  iA0_t, iA1_t, iB0_t, iB1_t, oRESULT_t);

  initial begin

    iCLK_t = 0;
    iRST_N_t = 0;
    iSEL_t = 0;
    iA0_t = 0;
    iA1_t = 0;
    iB0_t = 0;
    iB1_t = 0;

    #7
    iRST_N_t = 1;
    #1
    iA0_t = 3;
    iA1_t = 2;
    iB0_t = 3;
    iB1_t = 2;

    #4
    iSEL_t = 1;
    iA0_t = 1;
    iA1_t = 1;
    iB0_t = 1;
    iB1_t = 1;


    #4
    iA0_t = 2;
    iA1_t = 2;
    iB0_t = 2;
    iB1_t = 2;



    #4
```

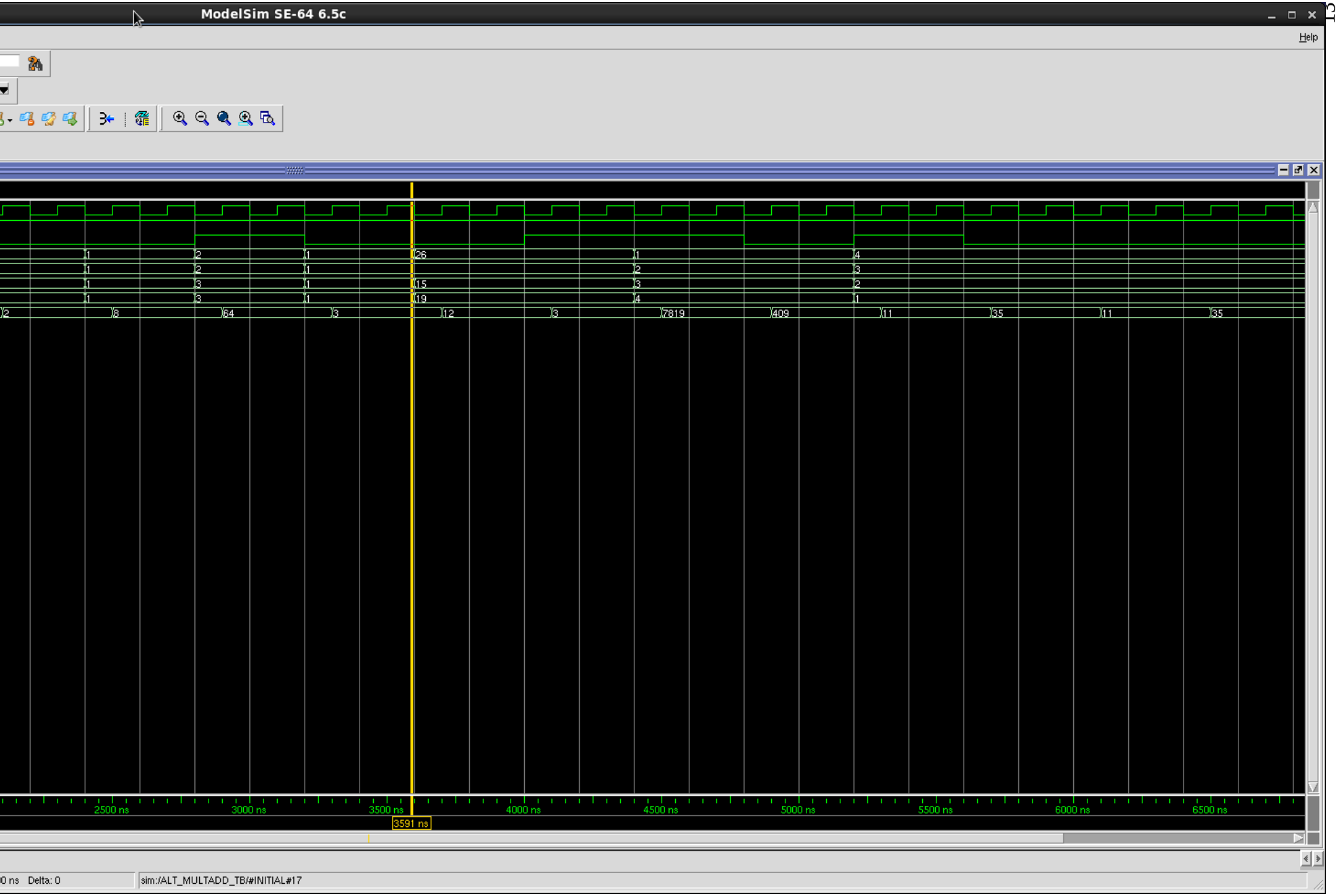
```
iSEL_t = 0;
iA0_t = 2;
iA1_t = 2;
iB0_t = 2;
iB1_t = 6;


#4
iA0_t = 1;
iA1_t = 1;
iB0_t = 1;
iB1_t = 1;

#4
iSEL_t = 1;
iA0_t = 2;
iA1_t = 2;
iB0_t = 3;
iB1_t = 3;

#4
iSEL_t = 0;
iA0_t = 1;
iA1_t = 1;
iB0_t = 1;
iB1_t = 1;


#4
iA0_t = 26;
iA1_t = 1;
iB0_t = 15;
iB1_t = 19;

#4
iSEL_t = 1;
iA0_t = 26;
iA1_t = 1;
iB0_t = 15;
iB1_t = 19;

#4
iSEL_t = 1;
iA0_t = 1;
iA1_t = 2;
iB0_t = 3;
iB1_t = 4;

#4
iSEL_t = 0;
iA0_t = 1;
iA1_t = 2;
iB0_t = 3;
iB1_t = 4;

#4
iSEL_t = 1;
iA0_t = 4;
iA1_t = 3;
```

```
    iB0_t = 2;
    iB1_t = 1;

    #4
    iSEL_t = 0;
    iA0_t = 4;
    iA1_t = 3;
    iB0_t = 2;
    iB1_t = 1;


    #20
    $stop;

  end

  always #1 iCLK_t = ~iCLK_t;

endmodule
```

```
# Test control
#     iCLK_t   iRST_N_t     iSEL_t      iA0_t      iA1_t      iB0_t      iB1_t  oRESULT_t
#          0          0          0          0          0          0          0          x
#          1          0          0          0          0          0          0          0
#          0          0          0          0          0          0          0          0
#          1          0          0          0          0          0          0          0
#          0          0          0          0          0          0          0          0
#          1          0          0          0          0          0          0          0
#          0          0          0          0          0          0          0          0
#          1          1          0          0          0          0          0          0
#          0          1          0          3          2          3          2          0
#          1          1          0          3          2          3          2          0
#          0          1          0          3          2          3          2          0
#          1          1          0          3          2          3          2          0
#          0          1          1          1          1          1          1          0
#          1          1          1          1          1          1          1          0
#          0          1          1          1          1          1          1          0
#          1          1          1          1          1          1          1          0
#          0          1          1          2          2          2          2          0
#          1          1          1          2          2          2          2         49
#          0          1          1          2          2          2          2         49
#          1          1          1          2          2          2          2         49
#          0          1          0          2          2          2          6         49
#          1          1          0          2          2          2          6          2
#          0          1          0          2          2          2          6          2
#          1          1          0          2          2          2          6          2
#          0          1          0          1          1          1          1          2
#          1          1          0          1          1          1          1          8
#          0          1          0          1          1          1          1          8
#          1          1          0          1          1          1          1          8
#          0          1          1          2          2          3          3          8
#          1          1          1          2          2          3          3         64
#          0          1          1          2          2          3          3         64
#          1          1          1          2          2          3          3         64
#          0          1          0          1          1          1          1         64
#          1          1          0          1          1          1          1          3
#          0          1          0          1          1          1          1          3
#          1          1          0          1          1          1          1          3
#          0          1          0         26          1         15         19          3
#          1          1          0         26          1         15         19         12
#          0          1          0         26          1         15         19         12
#          1          1          0         26          1         15         19         12
#          0          1          1         26          1         15         19         12
#          1          1          1         26          1         15         19          3
#          0          1          1         26          1         15         19          3
#          1          1          1         26          1         15         19          3
#          0          1          1          1          2          3          4          3
#          1          1          1          1          2          3          4       7819
#          0          1          1          1          2          3          4       7819
#          1          1          1          1          2          3          4       7819
#          0          1          0          1          2          3          4       7819
#          1          1          0          1          2          3          4        409
#          0          1          0          1          2          3          4        409
#          1          1          0          1          2          3          4        409
#          0          1          1          4          3          2          1        409
#          1          1          1          4          3          2          1         11
#          0          1          1          4          3          2          1         11
#          1          1          1          4          3          2          1         11
#          0          1          0          4          3          2          1         11
#          1          1          0          4          3          2          1         35
#          0          1          0          4          3          2          1         35
#          1          1          0          4          3          2          1         35
#          0          1          0          4          3          2          1         35
```

```
#       1       1       0       4       3       2       1       11
#       0       1       0       4       3       2       1       11
#       1       1       0       4       3       2       1       11
#       0       1       0       4       3       2       1       11
#       1       1       0       4       3       2       1       35
#       0       1       0       4       3       2       1       35
#       1       1       0       4       3       2       1       35
#       0       1       0       4       3       2       1       35
#       1       1       0       4       3       2       1       35
#       0       1       0       4       3       2       1       35
#       1       1       0       4       3       2       1       35
#       0       1       0       4       3       2       1       35
#       1       1       0       4       3       2       1       35
#       0       1       0       4       3       2       1       35
```


ModelSim SE-64 6.5c
File Edit View Compile Simulate Add Wave Tools Layout Window
Help
Layout Simulate
Wave
Messages
/ALT_MULTADD_TB/CLK_t
/ALT_MULTADD_TB/RST_N_t
/ALT_MULTADD_TB/SEL_t
/ALT_MULTADD_TB/A0_t
/ALT_MULTADD_TB/A1_t
/ALT_MULTADD_TB/B0_t
/ALT_MULTADD_TB/B1_t
/ALT_MULTADD_TB/oRESULT_t
Now 7600 ns
Cursor 1 3591 ns
1000 ns 1500 ns 2000 ns 2500 ns 3000 ns 3500 ns 4000 ns
3591 ns
Transcript Wave Message Viewer Objects Processes Library Project sim
586 ns to 6843 ns
Now: 7,600 ns Delta: 0
Keep 1
/ALT_MULTADD_TB/oRESULT_t
Project : Lab9
Now: 7,600 ns Delta: 0
sim:/ALT_MULTADD_TB/#INITIAL#17

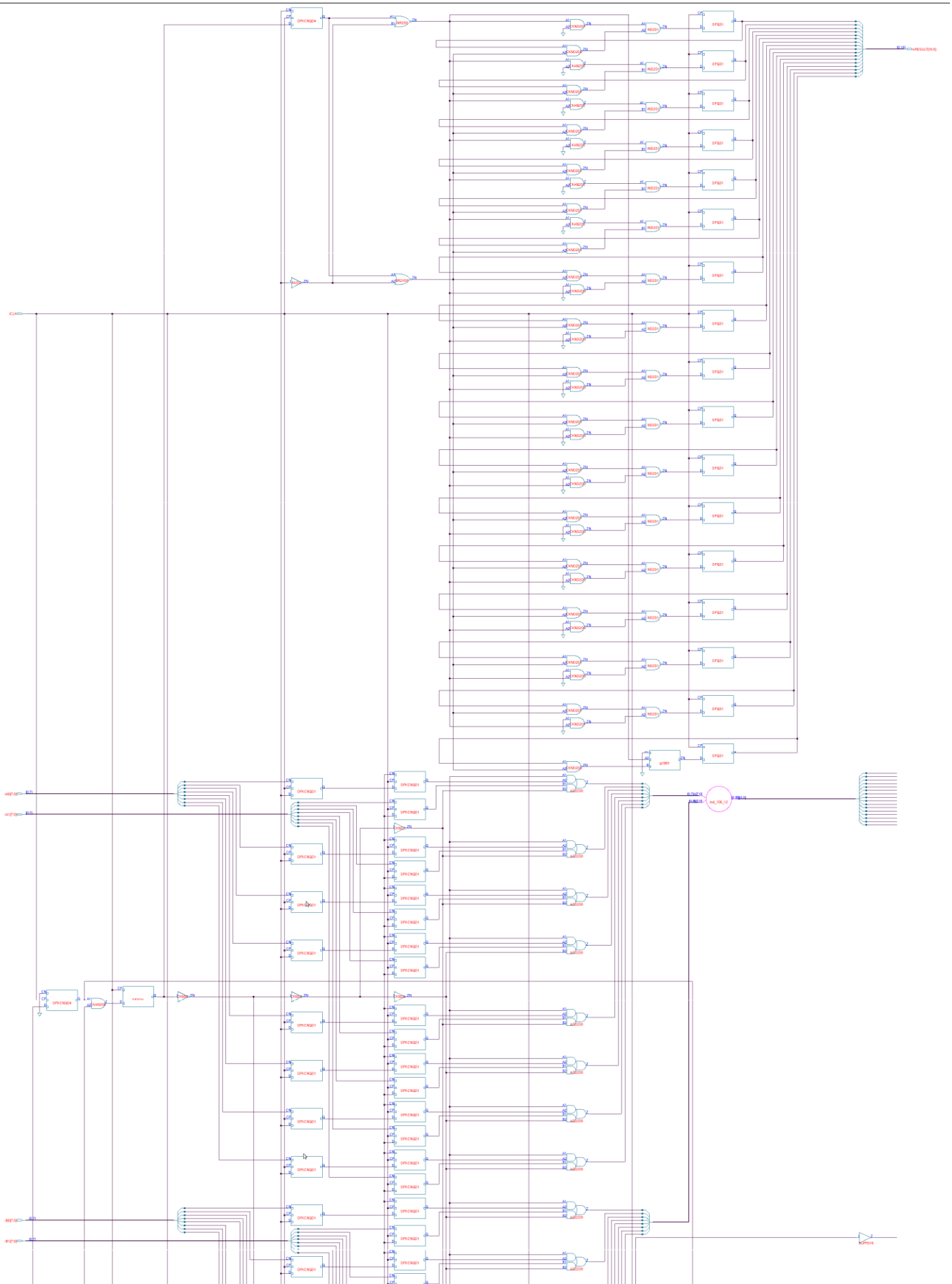

ModelSim SE-64 6.5c
Help
1
2
1
26
1
4
1
2
1
2
3
1
3
1
15
3
2
1
3
1
19
4
1
2
8
64
3
12
3
7819
409
11
35
11
35
2500 ns
3000 ns
3500 ns
4000 ns
4500 ns
5000 ns
5500 ns
6000 ns
6500 ns
3591 ns
0 ns Delta: 0
sim:/ALT_MULTADD_TB/#INITIAL#17

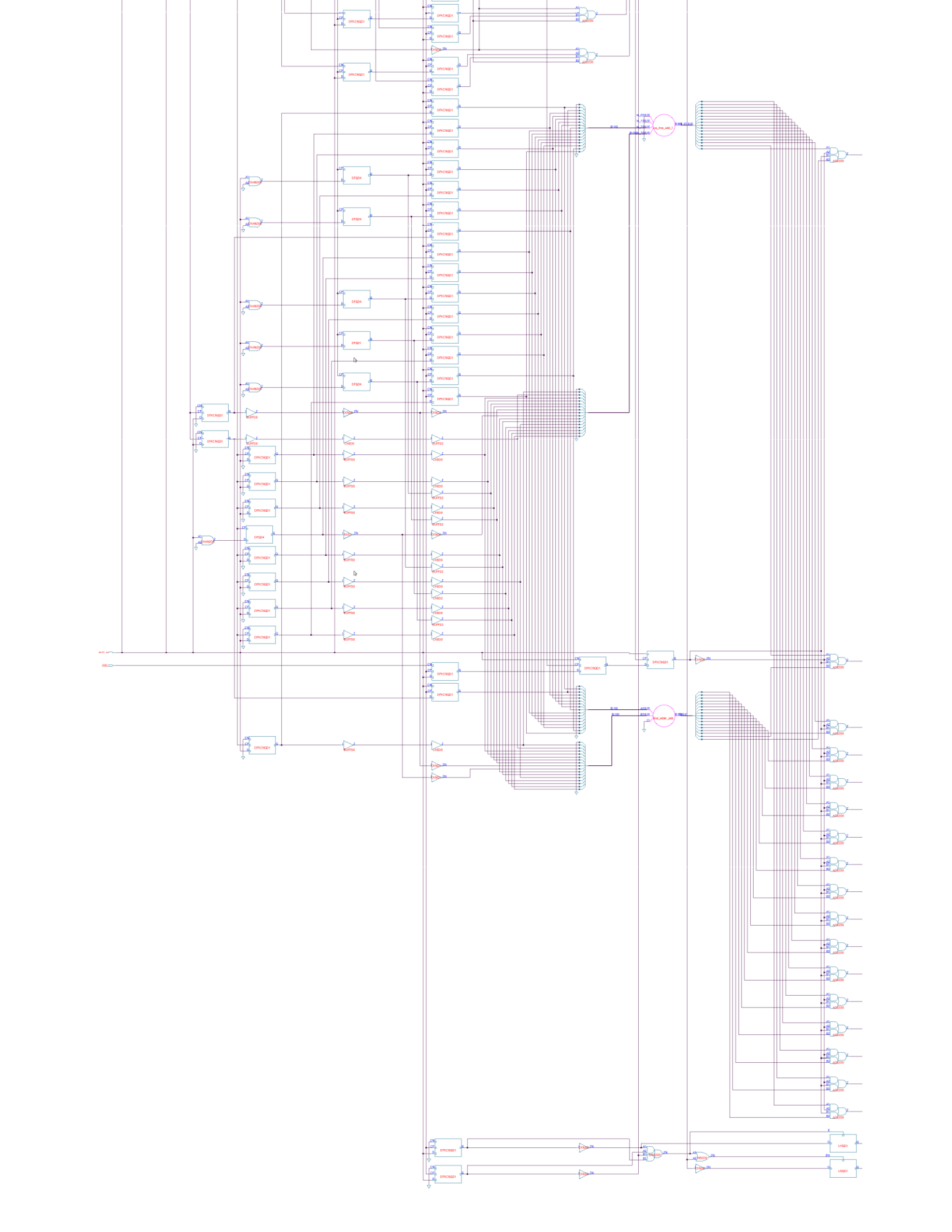
5 Synthesis:

In order to synthesize the Verilog code, I used this script.

```
## This sets the name of the directory in which area/timing/power reports
## and synthesized (mapped) netlists are stored.
set OUTPUT_DIR ./run_dir
if { ![file exists ${OUTPUT_DIR}] }  { sh mkdir ${OUTPUT_DIR} }


#### Step 1 ####
## This tells the compiler where to look for the libraries
set_attribute lib_search_path ../libdir

## This defines the libraries to use
set_attribute library {tcbn65gpluswc.lib}
##set_attribute library {tcbn65gplustc.lib}
##set_attribute library {tcbn90ghpbc_ccs.lib}
##set_attribute lp_insert_clock_gating true

#set attribute lp insert operand isolation true
#rm /designs/*


#### Step 2 ####
##This must point to your VHDL/verilog file
load -v2001 ../../ALT_MULTADD_redid.v
set_attribute lp_insert_clock_gating false


#### Step 3 ####
## This builds the general block
elaborate

dc::set_time_unit -nanoseconds
dc::set_load_unit -picofarads
define_clock -period 200 -name clk [dc::get_ports {iCLK}] -rise 5 -fall 5
set_attribute lp_power_unit {nW}
set_attribute max_leakage_power 100000 /designs/ALT_MULTADD_9


set_attribute power_optimization_effort high
synthesize -to_mapped -effort high


report area > ${OUTPUT_DIR}/area.rpt
report gates > ${OUTPUT_DIR}/gates.rpt
report timing > ${OUTPUT_DIR}/timing.rpt
report timing -lint > ${OUTPUT_DIR}/lint.rpt
report summary > ${OUTPUT_DIR}/summary.rpt
report power > ${OUTPUT_DIR}/power.rpt

write -mapped > ${OUTPUT_DIR}/jpeg_mapped.v
write_script > ${OUTPUT_DIR}/jpeg_mapped.g
```

```
write_sdc > ${OUTPUT_DIR}/jpeg_mapped.sdc
```





Power, Area and Timing reports:

Power:

```
                                    Leakage    Dynamic      Total
          Instance          Cells Power(nW)   Power(nW)   Power(nW)
-------------------------------------------------------------------
ALT_MULTADD_9                 788 25576.140 3670617.854 3696193.994
  csa_tree_a..116_30_groupi   392 10639.582       0.000   10639.582
  mul_106_12                  135  5552.124       0.000    5552.124
  final_adder_add_113_21       18  1633.508       0.000    1633.508
```

Area:

```
          Instance            Cells  Cell Area  Net Area     Wireload
-------------------------------------------------------------------------
ALT_MULTADD_9                   788       3167         0  ZeroWireload (S)
  csa_tree_add_116_30_groupi    392       1377         0  ZeroWireload (S)
  mul_106_12                    135        590         0  ZeroWireload (S)
  final_adder_add_113_21         18        144         0  ZeroWireload (S)
```

Timing:

```
      Pin             Type     Fanout Load Slew Delay Arrival
                                      (fF) (ps)  (ps)   (ps)
-------------------------------------------------------------
(clock clk)         launch                                10 R
EN_3_reg/CP                                0               10 R
EN_3_reg/Q          DFKCNQD4       2 13.6   31  +123      133 F
g1946/A1                                        +0      133
g1946/ZN            NR2XD8        17 19.4   35   +29      162 R
g1669/A2                                        +0      162
g1669/ZN            CKND2D1        1  1.2   25   +28      190 F
g1668/A1                                        +0      190
g1668/ZN            ND2D1          1  1.1   21   +20      210 R
oRESULT_reg[9]/D    DFQD1                      +0      210
oRESULT_reg[9]/CP   setup                   0   +30      240 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clk)         capture                               210 R
-------------------------------------------------------------
Timing slack :     -30ps (TIMING VIOLATION)
Start-point  : EN_3_reg/CP
End-point    : oRESULT_reg[9]/D
```

Clock gating report:

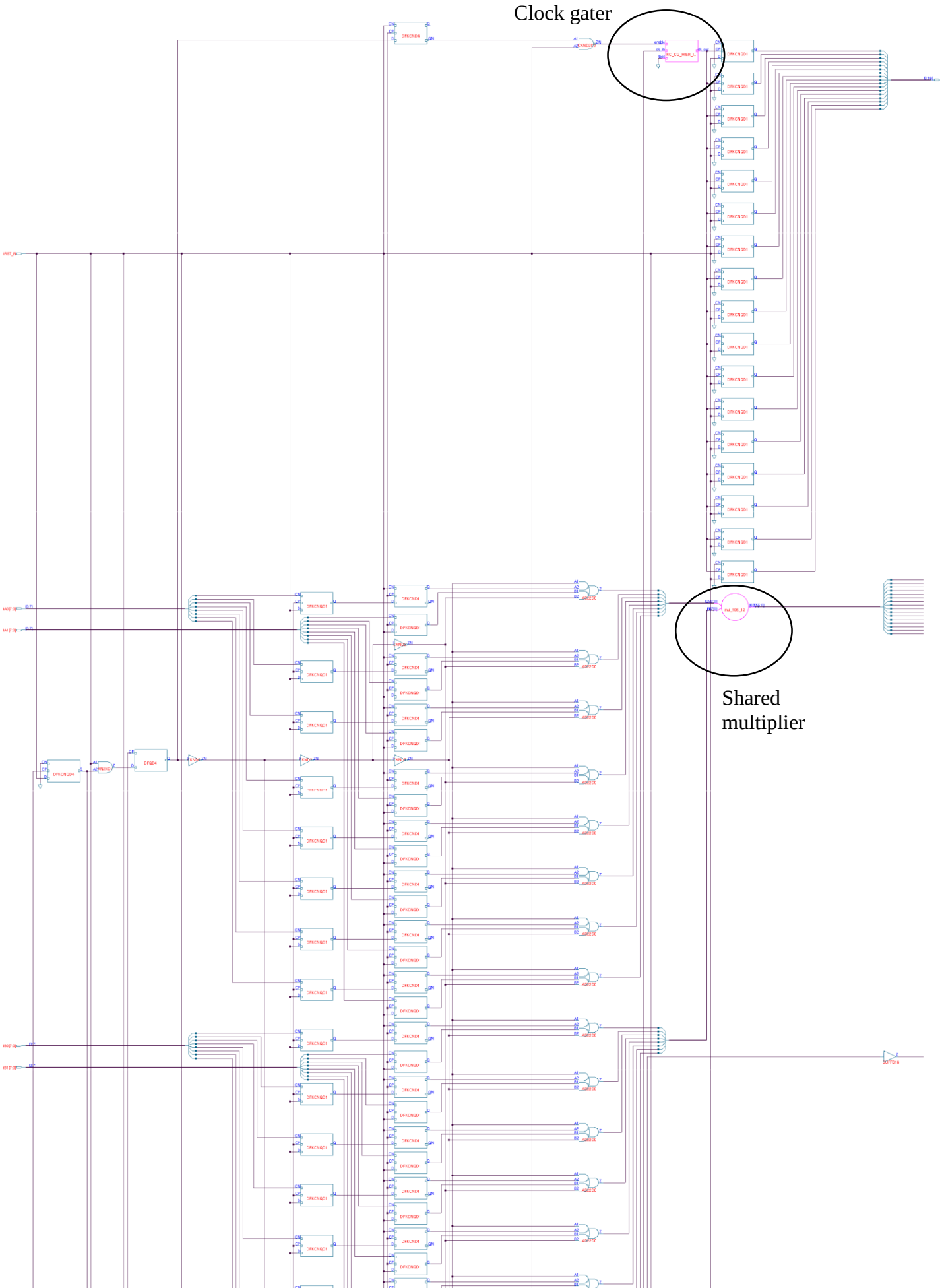
With the command “report clock_gating -summary” I got the following report:

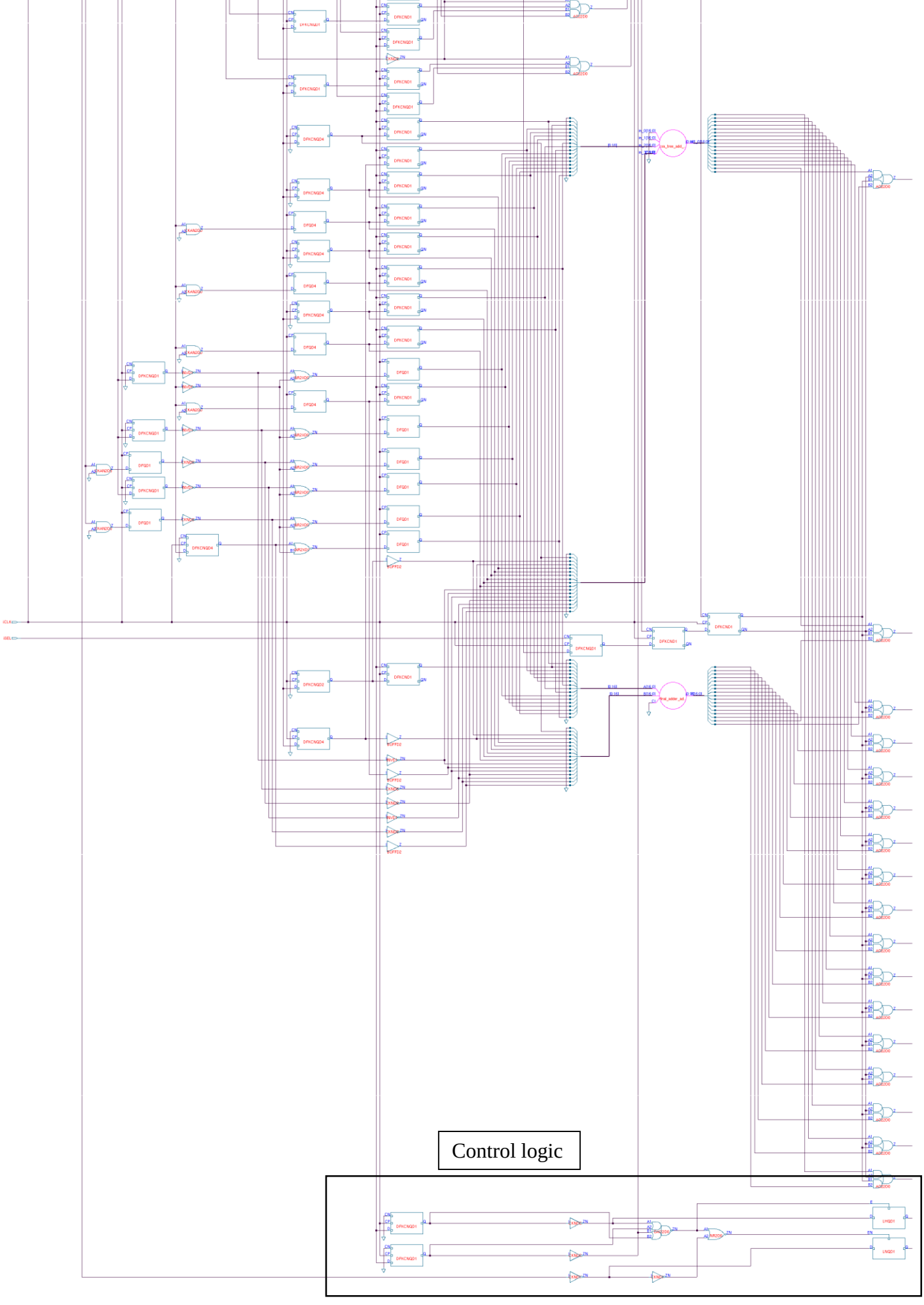
```
Summary
-------
--------------------------------------------
            Category            Number   %
--------------------------------------------
 RC Clock Gating Instances           0    -
 Non-RC Clock Gating Instances       0    -
--------------------------------------------
 RC Gated Flip-flops                 0    0
 Non-RC Gated Flip-flops             0    0
 Ungated Flip-flops                105  100
 Total Flip-flops                  105  100
--------------------------------------------
```

Obviously because I didn’t enable clock gating there were no clocks gated.

Synthesis with clock gating enabled:

```
## This sets the name of the directory in which area/timing/power reports
## and synthesized (mapped) netlists are stored.
set OUTPUT_DIR ./run_dir
if { ![file exists ${OUTPUT_DIR}] }  { sh mkdir ${OUTPUT_DIR} }


#### Step 1 ####
## This tells the compiler where to look for the libraries
set_attribute lib_search_path ../libdir

## This defines the libraries to use
set_attribute library {tcbn65gpluswc.lib}
##set_attribute library {tcbn65gplustc.lib}
##set_attribute library {tcbn90ghpbc_ccs.lib}
##set_attribute lp_insert_clock_gating true

#set_attribute lp_insert_operand_isolation true
#rm /designs/*


#### Step 2 ####
##This must point to your VHDL/verilog file
load -v2001 ../../ALT_MULTADD_redid.v

set_attribute lp_insert_clock_gating true


#### Step 3 ####
## This builds the general block
elaborate

dc::set_time_unit -nanoseconds
dc::set_load_unit -picofarads
define_clock -period 200 -name clk [dc::get_ports {iCLK}] -rise 5 -fall 5
set_attribute lp_power_unit {nW}
set_attribute max_leakage_power 100000 /designs/ALT_MULTADD_9


set_attribute power_optimization_effort high
synthesize -to_mapped -effort high


report area > ${OUTPUT_DIR}/area.rpt
report gates > ${OUTPUT_DIR}/gates.rpt
report timing > ${OUTPUT_DIR}/timing.rpt
report timing -lint > ${OUTPUT_DIR}/lint.rpt
report summary > ${OUTPUT_DIR}/summary.rpt
report power > ${OUTPUT_DIR}/power.rpt

write -mapped > ${OUTPUT_DIR}/jpeg_mapped.v
write_script > ${OUTPUT_DIR}/jpeg_mapped.g
write_sdc > ${OUTPUT_DIR}/jpeg_mapped.sdc
```


Clock gater
Shared multiplier


Control logic

Power, Area and Timing reports:

Power:

```
                                    Leakage    Dynamic      Total
          Instance          Cells Power(nW)   Power(nW)   Power(nW)
-------------------------------------------------------------------
ALT_MULTADD_9                 714 24911.327 3187026.832 3211938.160
  csa_tree_a..116_30_groupi   379 10296.895       0.000   10296.895
  mul_106_12                  135  5552.124       0.000    5552.124
  final_adder_add_113_21       18  1633.508       0.000    1633.508
  RC_CG_HIER_INST0              1    41.593   23555.208   23596.801
```

Dynamic and total power consumption decreased as expected due to the clock gating being enabled.

Area:

```
          Instance            Cells  Cell Area  Net Area     Wireload
-------------------------------------------------------------------------
ALT_MULTADD_9                   714       3080         0  ZeroWireload (S)
  csa_tree_add_116_30_groupi    379       1356         0  ZeroWireload (S)
  mul_106_12                    135        590         0  ZeroWireload (S)
  final_adder_add_113_21         18        144         0  ZeroWireload (S)
  RC_CG_HIER_INST0                1          6         0  ZeroWireload (S)
```

Area decreased due to replacement of AOI gates with the clock gate.

Timing:

```
      Pin            Type     Fanout Load Slew Delay Arrival
                                     (fF) (ps)  (ps)    (ps)
-------------------------------------------------------------
(clock clk)          launch                              10 R
EN_3_reg/CP                                 0            10 R
EN_3_reg/QN          DFKCND4       1  1.9   16  +154     164 R
g1263/A1                                          +0     164
g1263/ZN             CKND2D2       1  1.0   14   +15     178 F
RC_CG_HIER_INST0/enable
  RC_CGIC_INST/E     CKLNQD1                        +0     178
  RC_CGIC_INST/CP    setup                   0    +47     226 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock clk)          capture                            210 R
-------------------------------------------------------------
Cost Group   : 'cg_enable_group_clk' (path_group 'cg_enable_group_clk')
Timing slack :     -16ps (TIMING VIOLATION)
Start-point  : EN_3_reg/CP
End-point    : RC_CG_HIER_INST0/RC_CGIC_INST/E
```

Timing slack improved slightly.

Clock gating report:

With the command “report clock_gating -summary” I got the following report:

```
Summary
-------
------------------------------------------------
            Category             Number    %
------------------------------------------------
 RC Clock Gating Instances            1     -
 Non-RC Clock Gating Instances        0     -
------------------------------------------------
 RC Gated Flip-flops                 17    16
 Non-RC Gated Flip-flops              0     0
 Ungated Flip-flops                  88    84
 Total Flip-flops                   105   100
------------------------------------------------
```

Conclusion

As we have mentioned before, the coding style dramatically effects the synthesized circuit. We can see this clearly now that we have changed the code to be more modular and to not set registers in multiple always blocks. There are improvements in all specifications of the circuit. As for the changes caused by the clock gating, it is hard to tell if it alone improved specifications because it may be only due to the changed coding style. I had a hard time getting the clock gating to synthesize correctly but finally found that I needed to put the if statement in the always block for the registers which I was clock gating.