EHB436E

DIGITAL SYSTEM DESIGN APPLICATIONS

Muhammed Erkmen

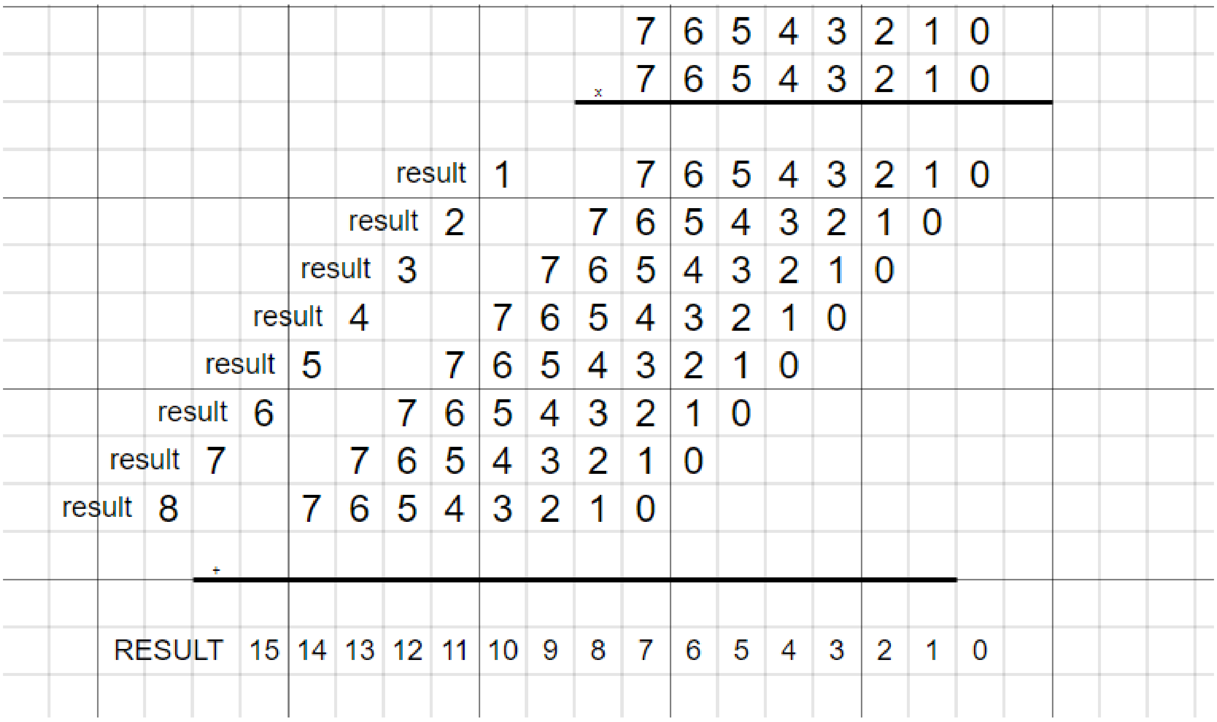
040170049

EXPERIMENT-7 REPORT

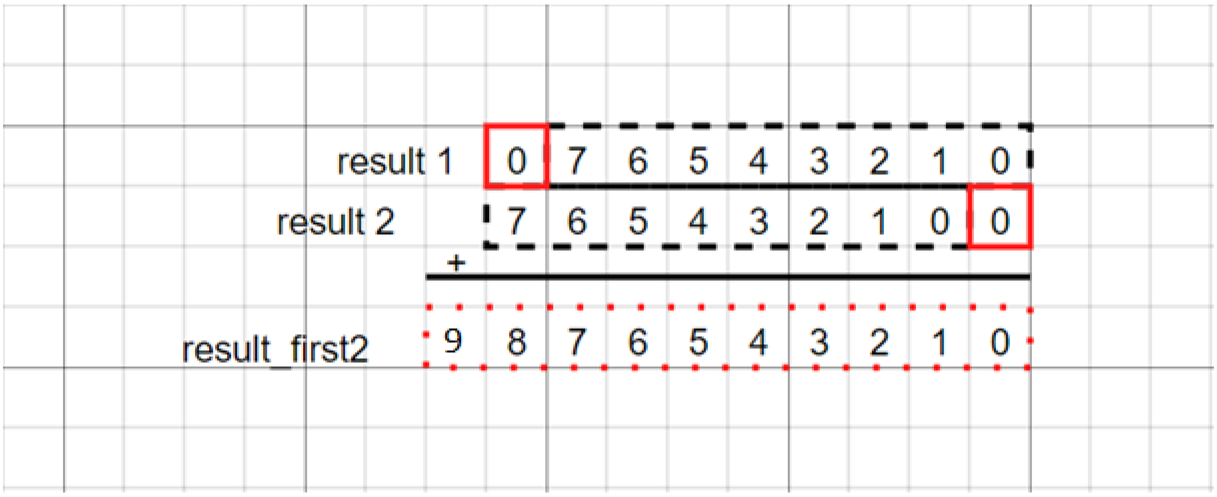
1) Structural Multiplier – Unsigned

In this part, i designed a structural multiplier with another technic other than experiment sheet. Because in the experiment sheet, multiply operator is used to get rid of shifters. But if we are able to use multiply operator, we can design our multiplier by just writing result = A*X. So I used my parametric ripple carry adders and assigned binary 0s to results to get rid of shifting. I explained what i did below.

This is what happens when we multiply 2-8 binary numbers.

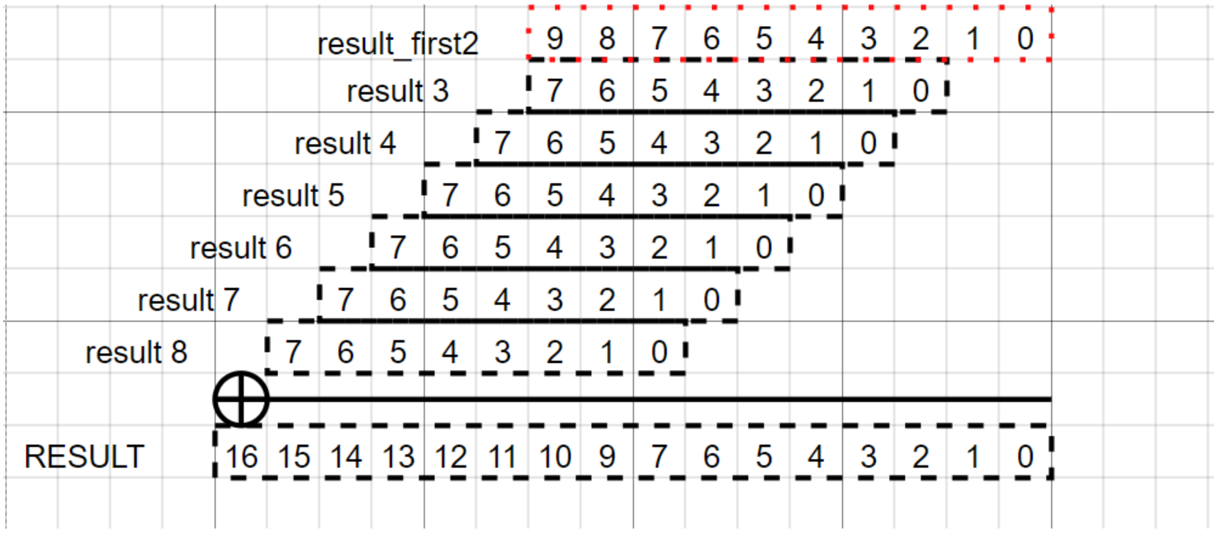


Added 1’b0 to result 1’s MSB and result 2’s LSB to make calculations clear.(Shifted but without always and without *). So the result will be come 9 bit. So in everystep, we’ll need another sized result wire to calculation. With different size wires, we are sure about we’ll never get X.

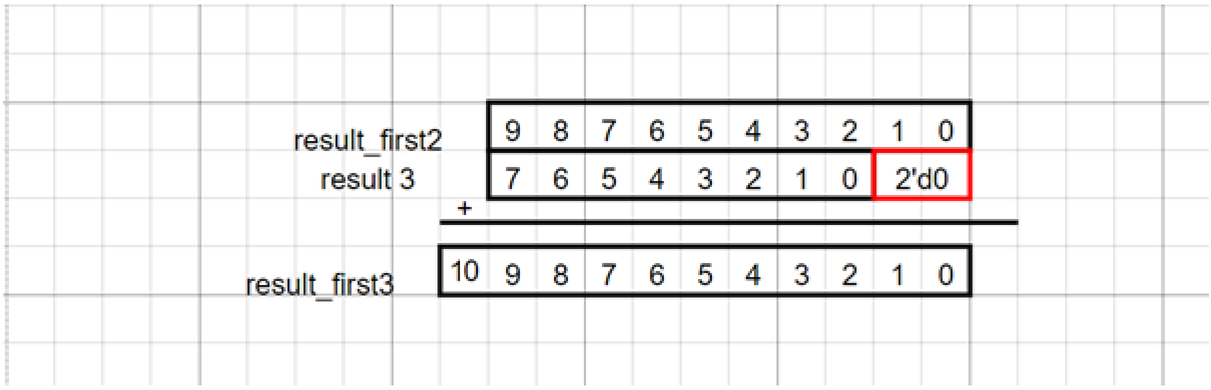

result 1 0 7 6 5 4 3 2 1 0
result 2 7 6 5 4 3 2 1 0 0
+
result_first2 9 8 7 6 5 4 3 2 1 0

I used this method for adding calculations, because when we add more than 3 bits, it is impossible to know cout with just a wire by using full and half adders.

Let me show the next step:



Now i need to make result 3's size 9 10 bit and need 11 bit result wire. That is how the process goes on and on.



That will go through to result_first7.
Adding result_first7 with 8th result will give us 16 bit result.

To get results, i used AND gate between every bit of X and every bit of A. So with doing that, **i didn't use any * operator in anywhere in my code** and made it configurable easily by adding new result wires and new parametric RCA.

Source Code:

```
`timescale 1ns / 1ps

module MULTS(
    input [7:0] A,
    input [7:0] X,
    output [15:0] result
    );


wire [8:0] first_row;
wire [8:0] second_row;
wire [9:0] result_first2;
wire [9:0] third_row;
wire [10:0] result_first3;
wire [10:0] fourth_row;
wire [11:0] result_first4;
wire [11:0] fifth_row;
wire [12:0] result_first5;
wire [12:0] sixth_row;
wire [13:0] result_first6;
wire [13:0] seventh_row;
wire [14:0] result_first7;
wire [14:0] eighth_row;


assign first_row = {1'b0,X[0]&A[7],X[0]&A[6],X[0]&A[5],X[0]&A[4],X[0]&A[3],X[0]&A[2],X[0]&A[1],X[0]&A[0]};
assign second_row = {X[1]&A[7],X[1]&A[6],X[1]&A[5],X[1]&A[4],X[1]&A[3],X[1]&A[2],X[1]&A[1],X[1]&A[0],1'b0};
assign third_row = {X[2]&A[7],X[2]&A[6],X[2]&A[5],X[2]&A[4],X[2]&A[3],X[2]&A[2],X[2]&A[1],X[2]&A[0],2'd0};
assign fourth_row = {X[3]&A[7],X[3]&A[6],X[3]&A[5],X[3]&A[4],X[3]&A[3],X[3]&A[2],X[3]&A[1],X[3]&A[0],3'd0};
assign fifth_row = {X[4]&A[7],X[4]&A[6],X[4]&A[5],X[4]&A[4],X[4]&A[3],X[4]&A[2],X[4]&A[1],X[4]&A[0],4'd0};
assign sixth_row = {X[5]&A[7],X[5]&A[6],X[5]&A[5],X[5]&A[4],X[5]&A[3],X[5]&A[2],X[5]&A[1],X[5]&A[0],5'd0};
assign seventh_row = {X[6]&A[7],X[6]&A[6],X[6]&A[5],X[6]&A[4],X[6]&A[3],X[6]&A[2],X[6]&A[1],X[6]&A[0],6'd0};
assign eighth_row = {X[7]&A[7],X[7]&A[6],X[7]&A[5],X[7]&A[4],X[7]&A[3],X[7]&A[2],X[7]&A[1],X[7]&A[0],7'd0};
assign result[0] = first_row[0];
parametric_RCA #(9) RCA1
(.x(first_row),.y(second_row),.cin(1'b0),.cout(result_first2[9]),.sum(result_first2[8:0]));
parametric_RCA #(10) RCA2
(.x(result_first2),.y(third_row),.cin(1'b0),.cout(result_first3[10]),.sum(result_first3[9:0]));
parametric_RCA #(11) RCA3
(.x(result_first3),.y(fourth_row),.cin(1'b0),.cout(result_first4[11]),.sum(result_first4[10:0]));
parametric_RCA #(12) RCA4
(.x(result_first4),.y(fifth_row),.cin(1'b0),.cout(result_first5[12]),.sum(result_first5[11:0]));
parametric_RCA #(13) RCA5
(.x(result_first5),.y(sixth_row),.cin(1'b0),.cout(result_first6[13]),.sum(result_first6[12:0]));
parametric_RCA #(14) RCA6
(.x(result_first6),.y(seventh_row),.cin(1'b0),.cout(result_first7[14]),.sum(result_first7[13:0]));
parametric_RCA #(15) RCA7
(.x(result_first7),.y(eighth_row),.cin(1'b0),.cout(result[15]),.sum(result[14:0]));
endmodule
```

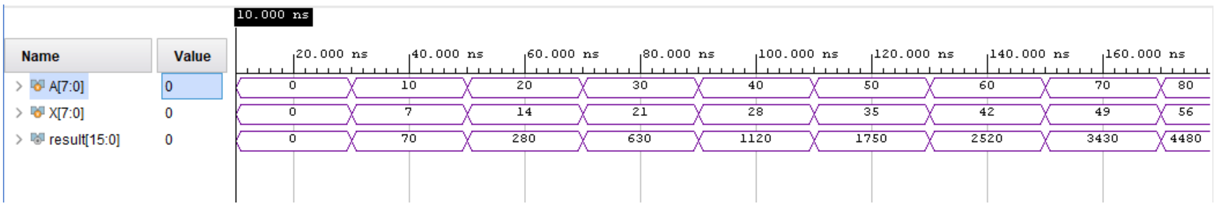
Old Parametric RCA Code:

```
`timescale 1ns / 1ps
module parametric_RCA
#(parameter SIZE = 9)(
input [(SIZE-1):0] x,
input [(SIZE-1):0] y,
input cin,
output cout,
output [(SIZE-1):0] sum);

genvar i;
wire [(SIZE-1):0] cout_inside;
assign cout = cout_inside[SIZE-1];
generate
for (i=0;i<SIZE;i=i+1)
    begin
    if(i==0)
    FA u0 (.x(x[i]),.y(y[i]),.cin(cin),.sum(sum[i]),.cout(cout_inside[i]));
    else
    FA u1 (.x(x[i]),.y(y[i]),.cin(cout_inside[i-1]),.sum(sum[i]),.cout(cout_inside[i]));
    end
endgenerate
endmodule

module HA (
input x,
input y,
output cout,
output sum
);
assign sum = ((~x)&&y) || (x&&(~y));
assign cout = x&&y;
endmodule

module FA (
input x,
input y,
input cin,
output cout,
output sum);
wire sum1;
wire cout1;
wire sum2;
wire cout2;
HA HA1 (.x(x),.y(y),.sum(sum1),.cout(cout1));
HA HA2 (.x(sum1),.y(cin),.sum(sum2),.cout(cout2));
assign sum = sum2;
assign cout = cout2 || cout1;
endmodule
```

Testbench Code:

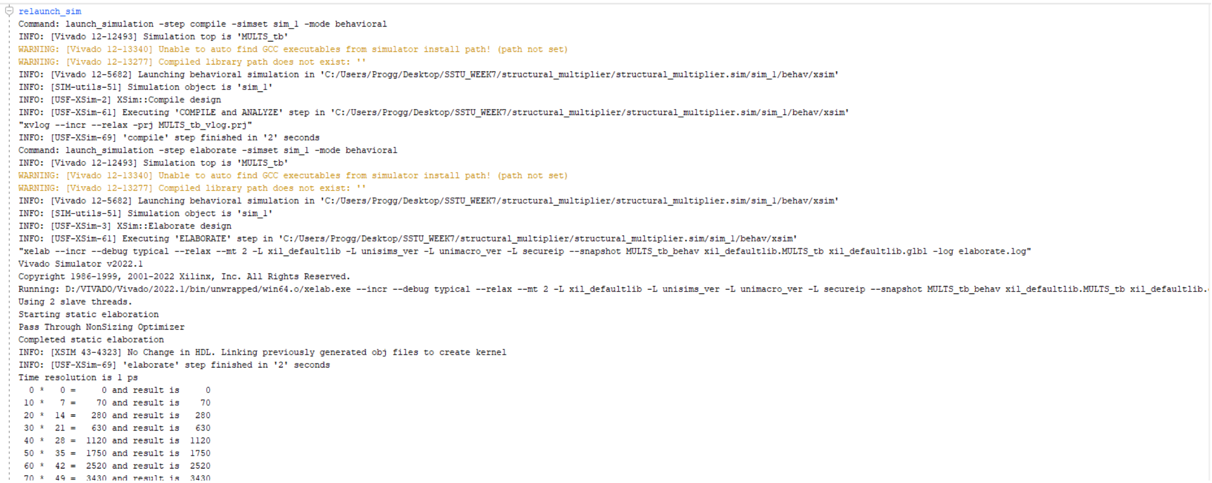
```
`timescale 1ns / 1ps


module MULTS_tb;

reg [7:0] A;
reg [7:0] X;
reg[15:0] real_result;
wire [15:0] result;

MULTS dut (.A(A),.X(X),.result(result));
integer i;
initial begin
#10;

for (i=0;i<=20;i=i+1)
    begin
        A=10*i;
        X=7*i;
        #10;
        real_result = A*X;
        #10;
        if(A*X == result)
        $display("%d * %d = %d ",A,X,real_result,"and result is %d ",result);
        else
        $display("Operation is wrong");

    end
end
endmodule
```

Behavioral Simulation Results:

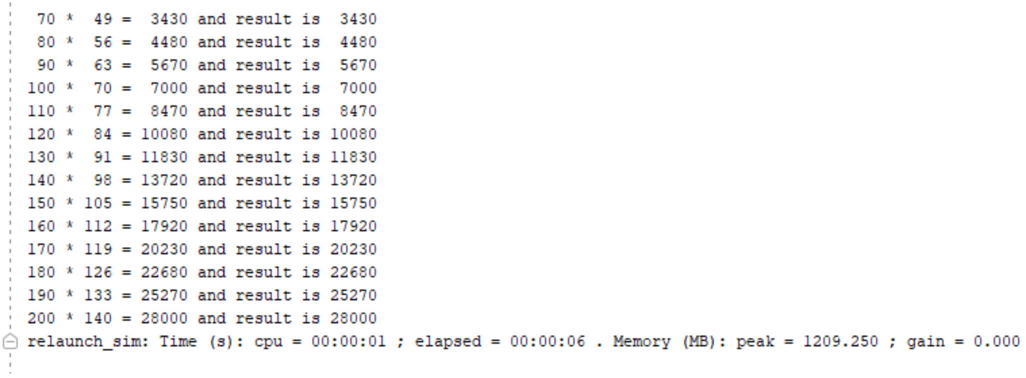


As we can see, every one of results that multiplication operation gives is true.

TCL Console Output:



```
relaunch_sim
Command: launch_simulation -step compile -simset sim_1 -mode behavioral
INFO: [Vivado 12-12493] Simulation top is 'MULTS_tb'
WARNING: [Vivado 12-13340] Unable to auto find GCC executables from simulator install path! (path not set)
WARNING: [Vivado 12-13277] Compiled library path does not exist: ''
INFO: [Vivado 12-5682] Launching behavioral simulation in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier/structural_multiplier.sim/sim_1/behav/xsim'
INFO: [SIM-utils-51] Simulation object is 'sim_1'
INFO: [USF-XSim-2] XSim::Compile design
INFO: [USF-XSim-61] Executing 'COMPILE and ANALYZE' step in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier/structural_multiplier.sim/sim_1/behav/xsim'
"xvlog --incr --relax -prj MULTS_tb_vlog.prj"
INFO: [USF-XSim-69] 'compile' step finished in '2' seconds
Command: launch_simulation -step elaborate -simset sim_1 -mode behavioral
INFO: [Vivado 12-12493] Simulation top is 'MULTS_tb'
WARNING: [Vivado 12-13340] Unable to auto find GCC executables from simulator install path! (path not set)
WARNING: [Vivado 12-13277] Compiled library path does not exist: ''
INFO: [Vivado 12-5682] Launching behavioral simulation in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier/structural_multiplier.sim/sim_1/behav/xsim'
INFO: [SIM-utils-51] Simulation object is 'sim_1'
INFO: [USF-XSim-3] XSim::Elaborate design
INFO: [USF-XSim-61] Executing 'ELABORATE' step in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier/structural_multiplier.sim/sim_1/behav/xsim'
"xelab --incr --debug typical --relax --mt 2 -L xil_defaultlib -L unisims_ver -L unimacro_ver -L secureip --snapshot MULTS_tb_behav xil_defaultlib.MULTS_tb xil_defaultlib.glbl -log elaborate.log"
Vivado Simulator v2022.1
Copyright 1986-1999, 2001-2022 Xilinx, Inc. All Rights Reserved.
Running: D:/VIVADO/Vivado/2022.1/bin/unwrapped/win64.o/xelab.exe --incr --debug typical --relax --mt 2 -L xil_defaultlib -L unisims_ver -L unimacro_ver -L secureip --snapshot MULTS_tb_behav xil_defaultlib.MULTS_tb xil_defaultlib.
Using 2 slave threads.
Starting static elaboration
Pass Through NonSizing Optimizer
Completed static elaboration
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '2' seconds
Time resolution is 1 ps
  0 *   0 =     0 and result is     0
 10 *   7 =    70 and result is    70
 20 *  14 =   280 and result is   280
 30 *  21 =   630 and result is   630
 40 *  28 =  1120 and result is  1120
 50 *  35 =  1750 and result is  1750
 60 *  42 =  2520 and result is  2520
 70 *  49 =  3430 and result is  3430
```



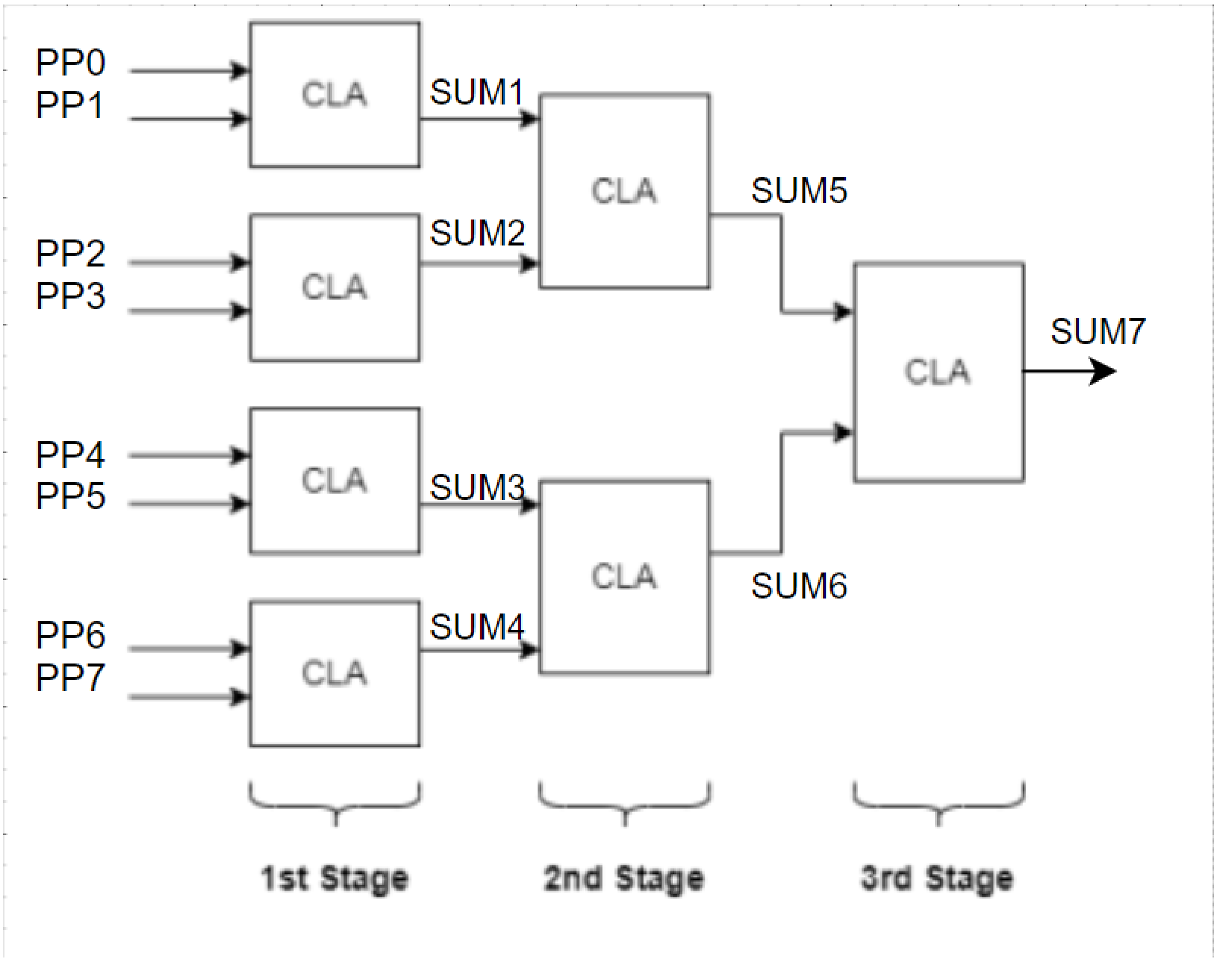
```
  70 *  49 =  3430 and result is  3430
  80 *  56 =  4480 and result is  4480
  90 *  63 =  5670 and result is  5670
 100 *  70 =  7000 and result is  7000
 110 *  77 =  8470 and result is  8470
 120 *  84 = 10080 and result is 10080
 130 *  91 = 11830 and result is 11830
 140 *  98 = 13720 and result is 13720
 150 * 105 = 15750 and result is 15750
 160 * 112 = 17920 and result is 17920
 170 * 119 = 20230 and result is 20230
 180 * 126 = 22680 and result is 22680
 190 * 133 = 25270 and result is 25270
 200 * 140 = 28000 and result is 28000
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:06 . Memory (MB): peak = 1209.250 ; gain = 0.000
```

In my testbench, i compare my result output with another A*X register in testbench which works as a calculator. And if they are equal, i print the values. Else i print the operation is wrong. As we can see there is nothing wrong, everything works excatly.

1) Structural Multiplier – Unsigned (by using method in sheet)

This is what sheet wants me to.

Sheet tells that sum6 should goes to result but we should make sum7 to go result for getting true results. Altough, we should make last SUM7 16 bit.



First, i coded 16 bit CLA module. Than i did these calculations and assigned them to PP variables.

$$
\begin{aligned}
PP0 &= x_0 * A * 2^0 \\
PP1 &= x_1 * A * 2^1 \\
PP2 &= x_2 * A * 2^2 \\
PP3 &= x_3 * A * 2^3 \\
PP4 &= x_4 * A * 2^4 \\
PP5 &= x_5 * A * 2^5 \\
PP6 &= x_6 * A * 2^6 \\
PP7 &= x_7 * A * 2^7
\end{aligned}
$$

16-bit CLA module:

```
`timescale 1ns / 1ps
module CLA(
input [15:0] x,
input [15:0] y,
input cin,
output cout,
output [15:0] sum
    );

// P = x^y
// G = x&y
// Digital Design book expressions:
// Si = Pi ^^ Ci
// Ci+1 = Gi + PiCi

// C1 = G0 + P0Cin
// C2 = G1 + P1(G0+P0Cin)
// C3 = G2 + P2(G1+(P1(G0+P0Cin))
// C4 = G3 + P3(G2 + P2(G1+(P1(G0+P0Cin))))
wire [15:0] p,g,c;
assign p = x^y;
assign g = x&y;
assign c[0] = cin;
assign c[1] = g[0] | (p[0]&cin);
assign c[2] = g[1] | (p[1]&(g[0] | (p[0]&cin)));
assign c[3] = g[2] | (p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))));
assign c[4] = g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] |
(p[0]&cin)))))));
assign c[5] = g[4] | (p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] |
(p[0]&cin)))))))));
assign c[6] = g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1]
| (p[1]&(g[0] | (p[0]&cin)))))))))));
assign c[7] = g[6] | (p[6] & (g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]&
(g[2] | (p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))))));
assign c[8] = g[7] | (p[7] & (g[6] | (p[6] & (g[5] | (p[5] & (g[4] | (p[4] &
(g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))))))));
assign c[9] = g[8] | (p[8] & (g[7] | (p[7] & (g[6] | (p[6] & (g[5] | (p[5] &
(g[4] | (p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] |
(p[0]&cin)))))))))))))))));
assign c[10] = g[9] | (p[9] & (g[8] | (p[8] & (g[7] | (p[7] & (g[6] | (p[6] &
(g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0]
| (p[0]&cin)))))))))))))))))));
assign c[11] = g[10] | (p[10] & (g[9] | (p[9] & (g[8] | (p[8] & (g[7] | (p[7] &
(g[6] | (p[6] & (g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]& (g[2] |
(p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))))))))))))));
assign c[12] = g[11] | (p[11] & (g[10] | (p[10] & (g[9] | (p[9] & (g[8] | (p[8]
& (g[7] | (p[7] & (g[6] | (p[6] & (g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]&
(g[2] | (p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))))))))))))))));
assign c[13] = g[12] | (p[12] & (g[11] | (p[11] & (g[10] | (p[10] & (g[9] |
(p[9] & (g[8] | (p[8] & (g[7] | (p[7] & (g[6] | (p[6] & (g[5] | (p[5] & (g[4] |
(p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] |
(p[0]&cin)))))))))))))))))))))))));
assign c[14] = g[13] | (p[13] & (g[12] | (p[12] & (g[11] | (p[11] & (g[10] |
(p[10] & (g[9] | (p[9] & (g[8] | (p[8] & (g[7] | (p[7] & (g[6] | (p[6] & (g[5] |
(p[5] & (g[4] | (p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] |
(p[0]&cin)))))))))))))))))))))))))));
```

```
assign c[15] = g[14] | (p[14] &(g[13] | (p[13] & (g[12] | (p[12] & (g[11] |
(p[11] & (g[10] | (p[10] & (g[9] | (p[9] & (g[8] | (p[8] & (g[7] | (p[7] & (g[6]
| (p[6] & (g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1] |
(p[1]&(g[0] | (p[0]&cin)))))))))))))))))))))))))))));
assign cout = g[15] | (p[15] & (g[14] | (p[14] &(g[13] | (p[13] & (g[12] |
(p[12] & (g[11] | (p[11] & (g[10] | (p[10] & (g[9] | (p[9] & (g[8] | (p[8] &
(g[7] | (p[7] & (g[6] | (p[6] & (g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]&
(g[2] | (p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))))))))))))))))))))))));

assign sum = p^c;

endmodule
```

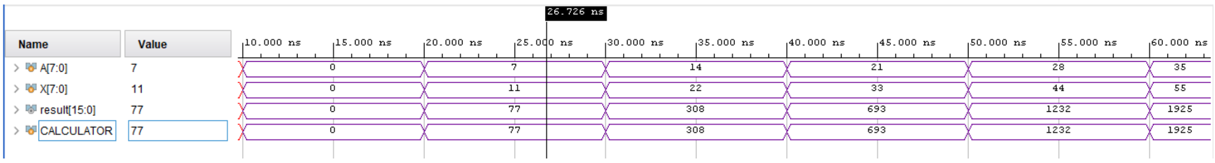
Source code:

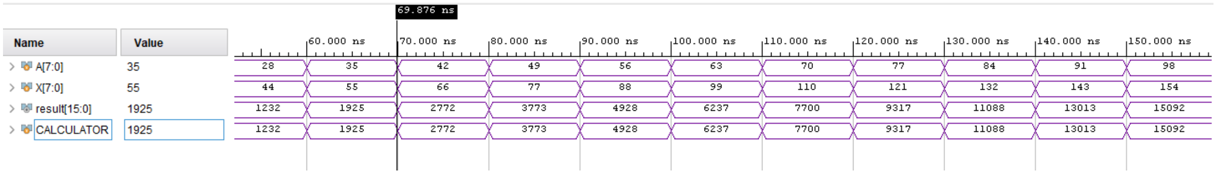
```
`timescale 1ns / 1ps

module MULTS(
    input [7:0] A,
    input [7:0] X,
    output [15:0] result
    );


wire [15:0] PP [7:0];

wire [15:0] sum1,sum2,sum3,sum4,sum5,sum6;
wire [15:0] sum7;


genvar i;

generate
    for (i = 0; i <= 7; i = i + 1) begin
        assign PP[i] = ({8{X[i]}} & A) << i;
    end
endgenerate

assign result = sum7;

//Stage1
CLA Stg1_CLA1 (.x(PP[0]),.y(PP[1]),.cin(1'b0),.cout(),.sum(sum1));
CLA Stg1_CLA2 (.x(PP[2]),.y(PP[3]),.cin(1'b0),.cout(),.sum(sum2));
CLA Stg1_CLA3 (.x(PP[4]),.y(PP[5]),.cin(1'b0),.cout(),.sum(sum3));
CLA Stg1_CLA4 (.x(PP[6]),.y(PP[7]),.cin(1'b0),.cout(),.sum(sum4));

// Stage2
CLA Stg2_CLA1 (.x(sum1),.y(sum2),.cin(1'b0),.cout(),.sum(sum5));
CLA Stg2_CLA2 (.x(sum3),.y(sum4),.cin(1'b0),.cout(),.sum(sum6));

//Stage3
CLA Stg3_CLA1 (.x(sum5),.y(sum6),.cin(1'b0),.cout(),.sum(sum7));
endmodule
```

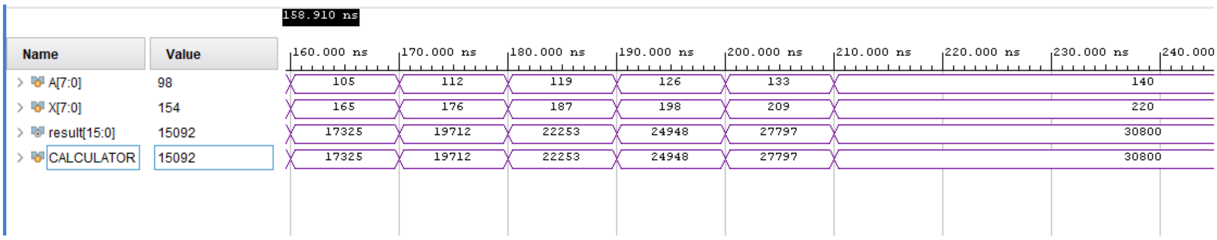
Testbench Code:

```
`timescale 1ns / 1ps


module MULTS_tb;

reg [7:0] A;
reg [7:0] X;
reg[15:0] real_result;
wire [15:0] result;

MULTS dut (.A(A),.X(X),.result(result));
integer i;
initial begin
#10;

for (i=0;i<=20;i=i+1)
    begin
        A=7*i;
        X=11*i;
        real_result = A*X;
        #10;
        if(A*X == result)
        $display("%d * %d = %d ",A,X,real_result,"and result is %d ",result);
        else
        $display("Operation is wrong");

    end
end

endmodule
```

Behavioral Simulation Results:







Result of every operation is true.

TCL Console output:

```
close_sim

INFO: [Simtcl 6-16] Simulation closed

launch_simulation

Command: launch_simulation

INFO: [Vivado 12-12493] Simulation top is 'MULTS_tb'

WARNING: [Vivado 12-13340] Unable to auto find GCC executables from simulator install path! (path not set)

WARNING: [Vivado 12-13277] Compiled library path does not exist: ''

INFO: [Vivado 12-5682] Launching behavioral simulation in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier/structural_multiplier.sim/sim_1/behav/xsim'

INFO: [SIM-utils-51] Simulation object is 'sim_1'

INFO: [SIM-utils-72] Using boost library from 'D:/VIVADO/Vivado/2022.1/tps/boost_1_72_0'

INFO: [SIM-utils-54] Inspecting design source files for 'MULTS_tb' in fileset 'sim_1'...

INFO: [USF-XSim-97] Finding global include files...

INFO: [USF-XSim-98] Fetching design files from 'sim_1'...

INFO: [USF-XSim-2] XSim::Compile design

INFO: [USF-XSim-61] Executing 'COMPILE and ANALYZE' step in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier/structural_multiplier.sim/sim_1/behav/xsim'

"xvlog --incr --relax -prj MULTS_tb_vlog.prj"

INFO: [USF-XSim-69] 'compile' step finished in '2' seconds

INFO: [USF-XSim-3] XSim::Elaborate design

INFO: [USF-XSim-61] Executing 'ELABORATE' step in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier/structural_multiplier.sim/sim_1/behav/xsim'

"xelab --incr --debug typical --relax --mt 2 -L xil_defaultlib -L unisims_ver -L unimacro_ver -L secureip --snapshot MULTS_tb_behav xil_defaultlib.MULTS_tb xil_defaultlib.glbl -log elaborate.log"

Vivado Simulator v2022.1
```

```
Copyright 1986-1999, 2001-2022 Xilinx, Inc. All Rights Reserved.
Running: D:/VIVADO/Vivado/2022.1/bin/unwrapped/win64.o/xelab.exe --incr --debug typical --relax --mt 2 -L xil_defaultlib -L unisims_ver -L unimacro_ver -L secureip --snapshot MULTS_tb_behav xil_defaultlib.MULTS_tb xil_defaultlib.glbl -log elaborate.log
Using 2 slave threads.
Starting static elaboration
Pass Through NonSizing Optimizer
Completed static elaboration
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '2' seconds
INFO: [USF-XSim-4] XSim::Simulate design
INFO: [USF-XSim-61] Executing 'SIMULATE' step in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier/structural_multiplier.sim/sim_1/behav/xsim'
INFO: [USF-XSim-98] *** Running xsim
   with args "MULTS_tb_behav -key {Behavioral:sim_1:Functional:MULTS_tb} -tclbatch {MULTS_tb.tcl} -log {simulate.log}"
INFO: [USF-XSim-8] Loading simulator feature
Time resolution is 1 ps
source MULTS_tb.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration' or type 'create_wave_config' in the TCL console."
#   }
# }
# run 1000ns
  0 *   0 =     0 and result is     0
  7 *  11 =    77 and result is    77
 14 *  22 =   308 and result is   308
 21 *  33 =   693 and result is   693
 28 *  44 =  1232 and result is  1232
 35 *  55 =  1925 and result is  1925
 42 *  66 =  2772 and result is  2772
 49 *  77 =  3773 and result is  3773
 56 *  88 =  4928 and result is  4928
 63 *  99 =  6237 and result is  6237
 70 * 110 =  7700 and result is  7700
 77 * 121 =  9317 and result is  9317
 84 * 132 = 11088 and result is 11088
 91 * 143 = 13013 and result is 13013
 98 * 154 = 15092 and result is 15092
105 * 165 = 17325 and result is 17325
112 * 176 = 19712 and result is 19712
119 * 187 = 22253 and result is 22253
126 * 198 = 24948 and result is 24948
133 * 209 = 27797 and result is 27797
140 * 220 = 30800 and result is 30800
INFO: [USF-XSim-96] XSim completed. Design snapshot 'MULTS_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:04 ; elapsed = 00:00:06 . Memory (MB): peak = 2143.816 ; gain = 9.938
```

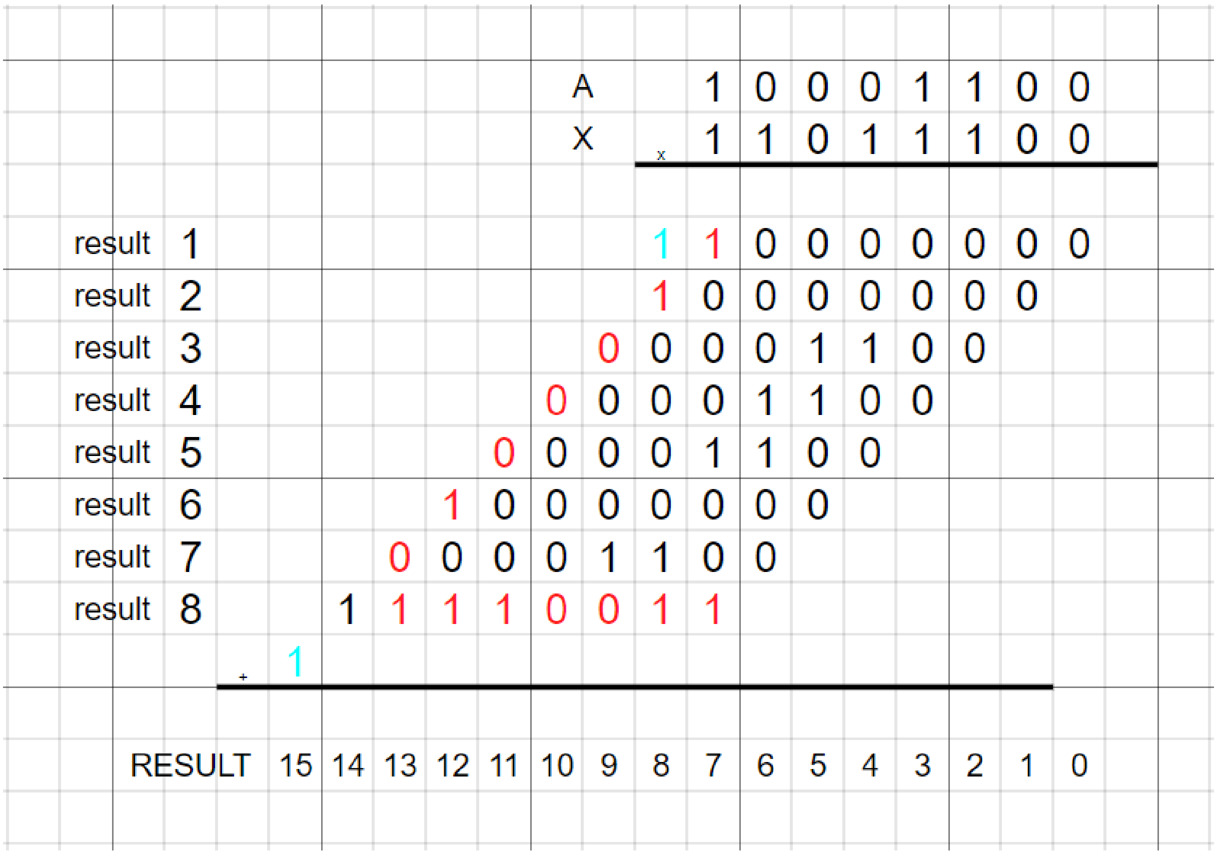
My testbench is coded for these random states and all of them came true. Nothing is wrong.

2) STRUCTURAL MULTIPLIER – SIGNED

In this part of experiment, i designed 8-bit structural multiplier by using Baugh-Wooley Method.

• Explain how many number of adder stages are necessary

In my implementation, number of adding stage didn't change.
Because, while we are generating Ppi * (2**i) = PPshifted[i] wires, i added an if block to there. We know in Baugh-Wooley method, there will be 2 1'b1 will be added to 9th index and 15th index. Everyone of us are sure that, first result of multiplication is will be first 8 bit of the wire. So i assigned 1'b1 to 9th bit of result1. In that case, we won't need any other addition. Because we get the 15th index wire with 1'b1 AND sum7, and assign this value to result outputs 15th index. So still 3 stage is enough.



						A			1	0	0	0	1	1	0	0
						X		x	1	1	0	1	1	1	0	0
result 1								1	1	0	0	0	0	0	0	0
result 2								1	0	0	0	0	0	0	0	
result 3							0	0	0	0	1	1	0	0		
result 4						0	0	0	0	1	1	0	0			
result 5					0	0	0	0	1	1	0	0				
result 6				1	0	0	0	0	0	0	0					
result 7			0	0	0	0	1	1	0	0						
result 8		1	1	1	1	0	0	1	1							
+	1															
RESULT	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

• Explanation of Baugh-Wooley Method

Baugh-Wooley Method is simply a method to make signed multiplication. This method is the one of the most effective methods that effective to handle most significant bits.

$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i$ $\qquad$ $X = -x_{n-1}2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$

a_{n-1} and x_{n-1} are most significant(sign) bits. With calculations, Result = A*X can be done by this algorithm. By making this calculations clear just look the figure above. The red numbers are the 2's complement of the result bits. Blacks are the normal results. And the turquoise ones are coming with the algorithm simplification. We can implement by just doing that .

Source Code:

```
`timescale 1ns / 1ps

module MULTS_signed(
    input [7:0] A,
    input [7:0] X,
    output [15:0] result
    );

wire [7:0] PP [7:0];
wire [15:0] PPshifted [7:0];
genvar i;
generate
for (i=0;i<=7;i=i+1)
begin
if(i<7)
    begin
    assign PP[i][6:0] = X[i]*A[6:0];
    assign PP[i][7] = ~(X[i]*A[7]);
    end
else
    begin
    assign PP[i][6:0] = ~(X[i]*A[6:0]);
    assign PP[i][7] = X[i]*A[7];
    end
end
endgenerate

generate
for(i=0;i<=7;i=i+1)
begin
if(i==0)
begin
assign PPshifted[i][8] =1'b1;
assign PPshifted[i][7:0] = PP[i][7:0];
assign PPshifted[i][15:9] = 6'd0;
end
else
assign PPshifted[i]= PP[i] << i;
end
endgenerate

wire [15:0] sum1,sum2,sum3,sum4,sum5,sum6,sum7;


//Stage1
CLA Stg1_CLA1
(.x(PPshifted[0]),.y(PPshifted[1]),.cin(1'b0),.cout(),.sum(sum1));
CLA Stg1_CLA2
(.x(PPshifted[2]),.y(PPshifted[3]),.cin(1'b0),.cout(),.sum(sum2));
CLA Stg1_CLA3
(.x(PPshifted[4]),.y(PPshifted[5]),.cin(1'b0),.cout(),.sum(sum3));
CLA Stg1_CLA4
(.x(PPshifted[6]),.y(PPshifted[7]),.cin(1'b0),.cout(),.sum(sum4));

// Stage2
CLA Stg2_CLA1 (.x(sum1),.y(sum2),.cin(1'b0),.cout(),.sum(sum5));
CLA Stg2_CLA2 (.x(sum3),.y(sum4),.cin(1'b0),.cout(),.sum(sum6));

//Stage3
```

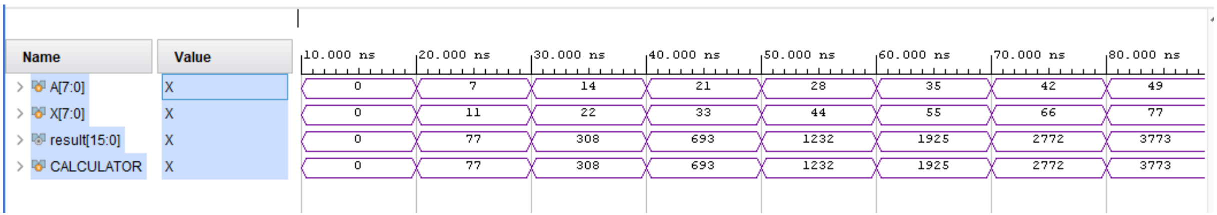
```
CLA Stg3_CLA1 (.x(sum5),.y(sum6),.cin(1'b0),.cout(),.sum(sum7));
assign result[15] = 1'b1^sum7[15];
assign result[14:0] = sum7[14:0];
endmodule

module CLA(
input [15:0] x,
input [15:0] y,
input cin,
output cout,
output [15:0] sum
    );

// P = x^y
// G = x&y
// Digital Design book expressions:
// Si = Pi ^^ Ci
// Ci+1 = Gi + PiCi

// C1 = G0 + P0Cin
// C2 = G1 + P1(G0+P0Cin)
// C3 = G2 + P2(G1+(P1(G0+P0Cin))
// C4 = G3 + P3(G2 + P2(G1+(P1(G0+P0Cin))))
wire [15:0] p,g,c;
assign p = x^y;
assign g = x&y;
assign c[0] = cin;
assign c[1] = g[0] | (p[0]&cin);
assign c[2] = g[1] | (p[1]&(g[0] | (p[0]&cin)));
assign c[3] = g[2] | (p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))));
assign c[4] = g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] |
(p[0]&cin)))))));
assign c[5] = g[4] | (p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0]
| (p[0]&cin)))))))));
assign c[6] = g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]& (g[2] |
(p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))));
assign c[7] = g[6] | (p[6] & (g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]&
(g[2] | (p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))))));
assign c[8] = g[7] | (p[7] & (g[6] | (p[6] & (g[5] | (p[5] & (g[4] | (p[4] &
(g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))))))));
assign c[9] = g[8] | (p[8] & (g[7] | (p[7] & (g[6] | (p[6] & (g[5] | (p[5] &
(g[4] | (p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] |
(p[0]&cin)))))))))))))))));
assign c[10] = g[9] | (p[9] & (g[8] | (p[8] & (g[7] | (p[7] & (g[6] | (p[6] &
(g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1] |
(p[1]&(g[0] | (p[0]&cin)))))))))))))))))));
assign c[11] = g[10] | (p[10] & (g[9] | (p[9] & (g[8] | (p[8] & (g[7] | (p[7]
& (g[6] | (p[6] & (g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]& (g[2] |
(p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))))))))))))));
assign c[12] = g[11] | (p[11] & (g[10] | (p[10] & (g[9] | (p[9] & (g[8] |
(p[8] & (g[7] | (p[7] & (g[6] | (p[6] & (g[5] | (p[5] & (g[4] | (p[4] & (g[3]
| (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))))))))))))))));
assign c[13] = g[12] | (p[12] & (g[11] | (p[11] & (g[10] | (p[10] & (g[9] |
(p[9] & (g[8] | (p[8] & (g[7] | (p[7] & (g[6] | (p[6] & (g[5] | (p[5] & (g[4]
| (p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] |
(p[0]&cin)))))))))))))))))))))))));
assign c[14] = g[13] | (p[13] & (g[12] | (p[12] & (g[11] | (p[11] & (g[10] |
(p[10] & (g[9] | (p[9] & (g[8] | (p[8] & (g[7] | (p[7] & (g[6] | (p[6] & (g[5]
| (p[5] & (g[4] | (p[4] & (g[3] | (p[3]& (g[2] | (p[2]&(g[1] | (p[1]&(g[0] |
(p[0]&cin)))))))))))))))))))))))))));
```

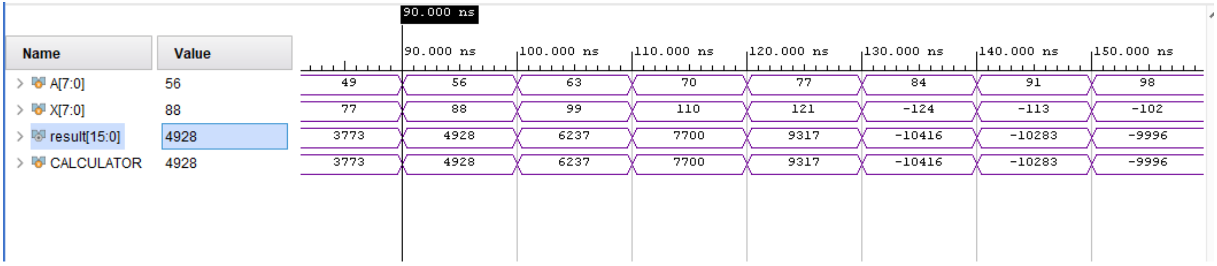
```
assign c[15] = g[14] | (p[14] &(g[13] | (p[13] & (g[12] | (p[12] & (g[11] |
(p[11] & (g[10] | (p[10] & (g[9] | (p[9] & (g[8] | (p[8] & (g[7] | (p[7] &
(g[6] | (p[6] & (g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]& (g[2] |
(p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))))))))))))))))))))));
assign cout = g[15] | (p[15] & (g[14] | (p[14] &(g[13] | (p[13] & (g[12] |
(p[12] & (g[11] | (p[11] & (g[10] | (p[10] & (g[9] | (p[9] & (g[8] | (p[8] &
(g[7] | (p[7] & (g[6] | (p[6] & (g[5] | (p[5] & (g[4] | (p[4] & (g[3] | (p[3]&
(g[2] | (p[2]&(g[1] | (p[1]&(g[0] | (p[0]&cin)))))))))))))))))))))))))))))));

assign sum = p^c;

endmodule
```

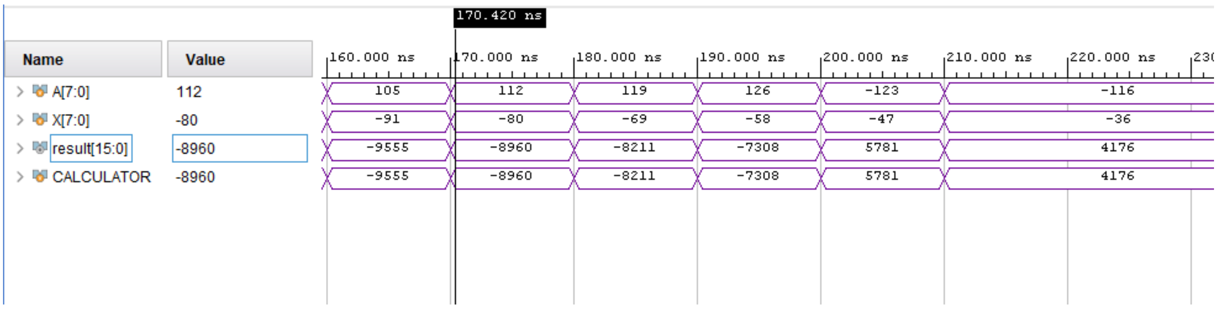
Testbench Code:

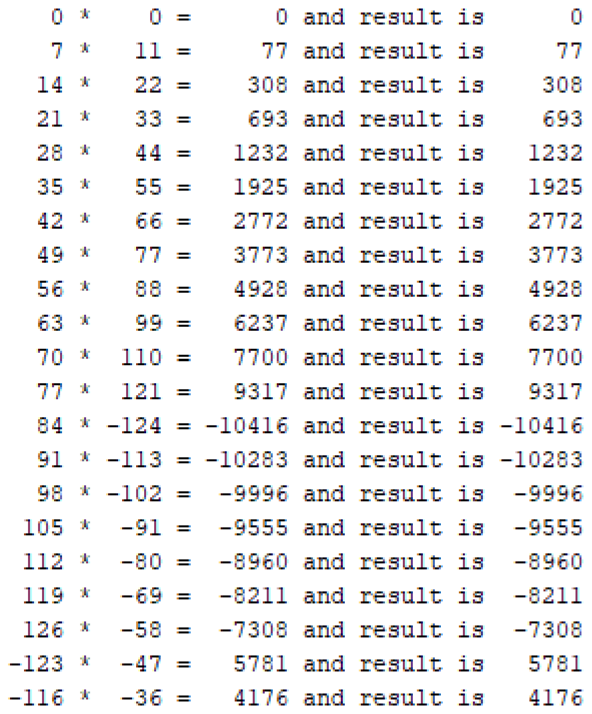
```
`timescale 1ns / 1ps


module MULTS_tb;

reg signed [7:0] A;
reg signed [7:0] X;
reg signed [15:0] real_result;
wire signed [15:0] result;

MULTS_signed dut (.A(A),.X(X),.result(result));
integer i;
initial begin
#10;

for (i=0;i<=20;i=i+1)
    begin
        A=7*i;
        X=11*i;
        real_result = A*X;
        #10;
        if((A*X) == (result))
        $display("%d * %d = %d
",$signed(A),$signed(X),$signed(real_result),"and result is %d
",$signed(result));
        else
        $display("Operation is wrong");

    end
end

endmodule
```

Behavioral Simulation Results:





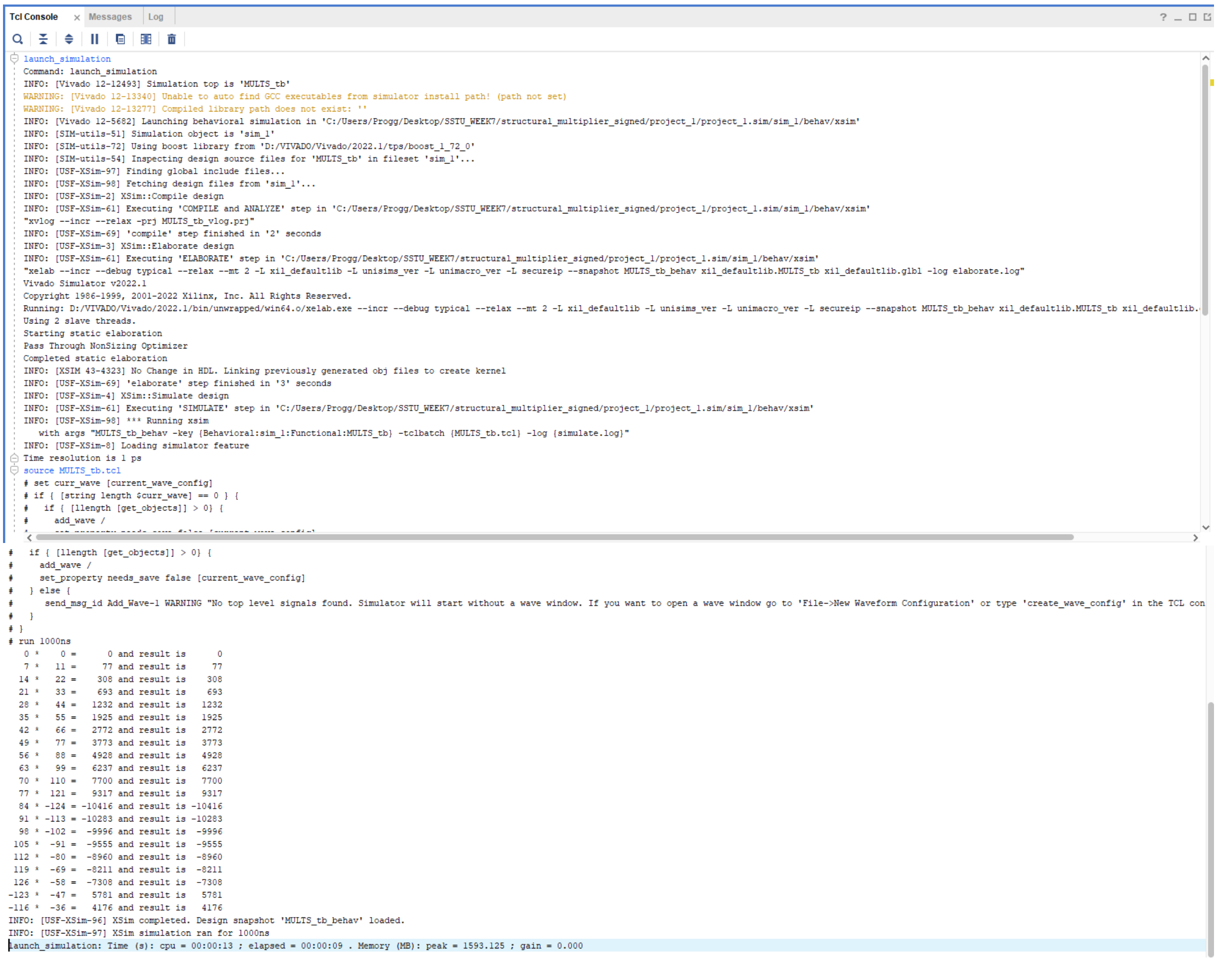




```
   0 *    0 =      0 and result is      0
   7 *   11 =     77 and result is     77
  14 *   22 =    308 and result is    308
  21 *   33 =    693 and result is    693
  28 *   44 =   1232 and result is   1232
  35 *   55 =   1925 and result is   1925
  42 *   66 =   2772 and result is   2772
  49 *   77 =   3773 and result is   3773
  56 *   88 =   4928 and result is   4928
  63 *   99 =   6237 and result is   6237
  70 *  110 =   7700 and result is   7700
  77 *  121 =   9317 and result is   9317
  84 * -124 = -10416 and result is -10416
  91 * -113 = -10283 and result is -10283
  98 * -102 =  -9996 and result is  -9996
 105 *  -91 =  -9555 and result is  -9555
 112 *  -80 =  -8960 and result is  -8960
 119 *  -69 =  -8211 and result is  -8211
 126 *  -58 =  -7308 and result is  -7308
-123 *  -47 =   5781 and result is   5781
-116 *  -36 =   4176 and result is   4176
```

While implementing, i added A to first 8 switch, and X to last 8 switch, result to all LEDs respectively by indexes of result.

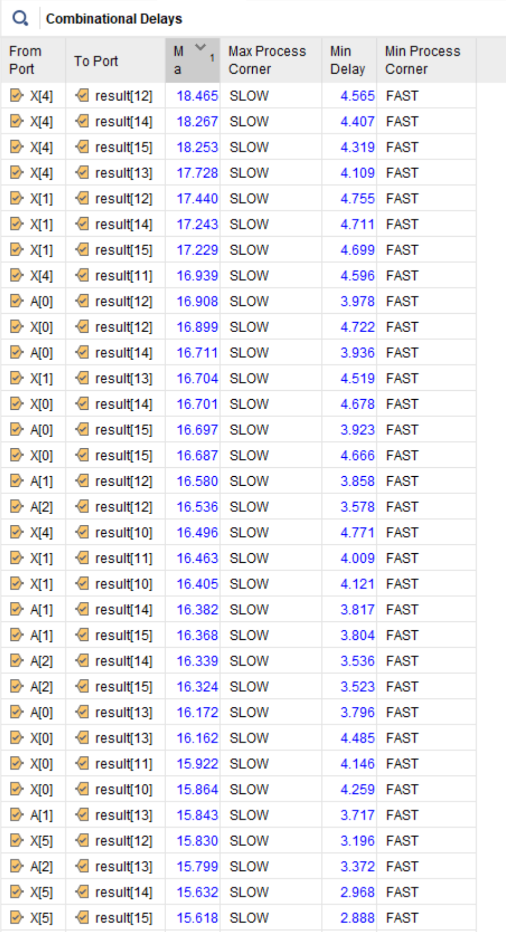
TCL Console Output:



```
launch_simulation
Command: launch_simulation
INFO: [Vivado 12-12493] Simulation top is 'MULTS_tb'
WARNING: [Vivado 12-13340] Unable to auto find GCC executables from simulator install path! (path not set)
WARNING: [Vivado 12-13277] Compiled library path does not exist: ''
INFO: [Vivado 12-5682] Launching behavioral simulation in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier_signed/project_1/project_1.sim/sim_1/behav/xsim'
INFO: [SIM-utils-51] Simulation object is 'sim_1'
INFO: [SIM-utils-72] Using boost library from 'D:/VIVADO/Vivado/2022.1/tps/boost_1_72_0'
INFO: [SIM-utils-54] Inspecting design source files for 'MULTS_tb' in fileset 'sim_1'...
INFO: [USF-XSim-97] Finding global include files...
INFO: [USF-XSim-98] Fetching design files from 'sim_1'...
INFO: [USF-XSim-2] XSim::Compile design
INFO: [USF-XSim-61] Executing 'COMPILE and ANALYZE' step in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier_signed/project_1/project_1.sim/sim_1/behav/xsim'
"xvlog --incr --relax -prj MULTS_tb_vlog.prj"
INFO: [USF-XSim-69] 'compile' step finished in '2' seconds
INFO: [USF-XSim-3] XSim::Elaborate design
INFO: [USF-XSim-61] Executing 'ELABORATE' step in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier_signed/project_1/project_1.sim/sim_1/behav/xsim'
"xelab --incr --debug typical --relax --mt 2 -L xil_defaultlib -L unisims_ver -L unimacro_ver -L secureip --snapshot MULTS_tb_behav xil_defaultlib.MULTS_tb xil_defaultlib.glbl -log elaborate.log"
Vivado Simulator v2022.1
Copyright 1986-1999, 2001-2022 Xilinx, Inc. All Rights Reserved.
Running: D:/VIVADO/Vivado/2022.1/bin/unwrapped/win64.o/xelab.exe --incr --debug typical --relax --mt 2 -L xil_defaultlib -L unisims_ver -L unimacro_ver -L secureip --snapshot MULTS_tb_behav xil_defaultlib.MULTS_tb xil_defaultlib.
Using 2 slave threads.
Starting static elaboration
Pass Through NonSizing Optimizer
Completed static elaboration
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '3' seconds
INFO: [USF-XSim-4] XSim::Simulate design
INFO: [USF-XSim-61] Executing 'SIMULATE' step in 'C:/Users/Progg/Desktop/SSTU_WEEK7/structural_multiplier_signed/project_1/project_1.sim/sim_1/behav/xsim'
INFO: [USF-XSim-98] *** Running xsim
   with args "MULTS_tb_behav -key {Behavioral:sim_1:Functional:MULTS_tb} -tclbatch {MULTS_tb.tcl} -log {simulate.log}"
INFO: [USF-XSim-8] Loading simulator feature
Time resolution is 1 ps
source MULTS_tb.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {
#     add_wave /
```

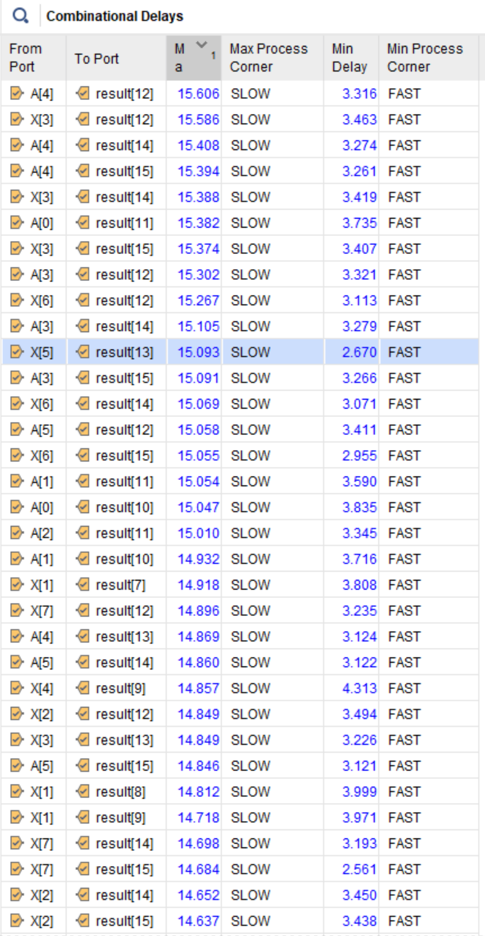
```
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration' or type 'create_wave_config' in the TCL con
#   }
# }
# run 1000ns
   0 *    0 =      0 and result is      0
   7 *   11 =     77 and result is     77
  14 *   22 =    308 and result is    308
  21 *   33 =    693 and result is    693
  28 *   44 =   1232 and result is   1232
  35 *   55 =   1925 and result is   1925
  42 *   66 =   2772 and result is   2772
  49 *   77 =   3773 and result is   3773
  56 *   88 =   4928 and result is   4928
  63 *   99 =   6237 and result is   6237
  70 *  110 =   7700 and result is   7700
  77 *  121 =   9317 and result is   9317
  84 * -124 = -10416 and result is -10416
  91 * -113 = -10283 and result is -10283
  98 * -102 =  -9996 and result is  -9996
 105 *  -91 =  -9555 and result is  -9555
 112 *  -80 =  -8960 and result is  -8960
 119 *  -69 =  -8211 and result is  -8211
 126 *  -58 =  -7308 and result is  -7308
-123 *  -47 =   5781 and result is   5781
-116 *  -36 =   4176 and result is   4176
INFO: [USF-XSim-96] XSim completed. Design snapshot 'MULTS_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:13 ; elapsed = 00:00:09 . Memory (MB): peak = 1593.125 ; gain = 0.000
```

Timing Report:



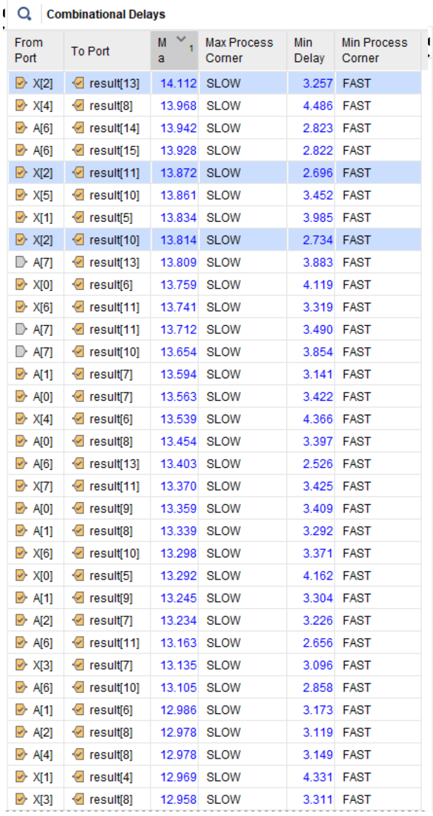
Combinational Delays

From Port	To Port	Ma	Max Process Corner	Min Delay	Min Process Corner
X[4]	result[12]	18.465	SLOW	4.565	FAST
X[4]	result[14]	18.267	SLOW	4.407	FAST
X[4]	result[15]	18.253	SLOW	4.319	FAST
X[4]	result[13]	17.728	SLOW	4.109	FAST
X[1]	result[12]	17.440	SLOW	4.755	FAST
X[1]	result[14]	17.243	SLOW	4.711	FAST
X[1]	result[15]	17.229	SLOW	4.699	FAST
X[4]	result[11]	16.939	SLOW	4.596	FAST
A[0]	result[12]	16.908	SLOW	3.978	FAST
X[0]	result[12]	16.899	SLOW	4.722	FAST
A[0]	result[14]	16.711	SLOW	3.936	FAST
X[1]	result[13]	16.704	SLOW	4.519	FAST
X[0]	result[14]	16.701	SLOW	4.678	FAST
A[0]	result[15]	16.697	SLOW	3.923	FAST
X[0]	result[15]	16.687	SLOW	4.666	FAST
A[1]	result[12]	16.580	SLOW	3.858	FAST
A[2]	result[12]	16.536	SLOW	3.578	FAST
X[4]	result[10]	16.496	SLOW	4.771	FAST
X[1]	result[11]	16.463	SLOW	4.009	FAST
X[1]	result[10]	16.405	SLOW	4.121	FAST
A[1]	result[14]	16.382	SLOW	3.817	FAST
A[1]	result[15]	16.368	SLOW	3.804	FAST
A[2]	result[14]	16.339	SLOW	3.536	FAST
A[2]	result[15]	16.324	SLOW	3.523	FAST
A[0]	result[13]	16.172	SLOW	3.796	FAST
X[0]	result[13]	16.162	SLOW	4.485	FAST
X[0]	result[11]	15.922	SLOW	4.146	FAST
X[0]	result[10]	15.864	SLOW	4.259	FAST
A[1]	result[13]	15.843	SLOW	3.717	FAST
X[5]	result[12]	15.830	SLOW	3.196	FAST
A[2]	result[13]	15.799	SLOW	3.372	FAST
X[5]	result[14]	15.632	SLOW	2.968	FAST
X[5]	result[15]	15.618	SLOW	2.888	FAST



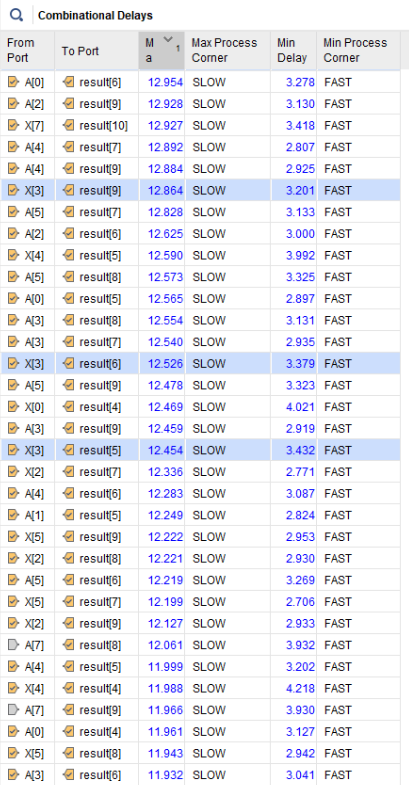
Combinational Delays

From Port	To Port	Ma	Max Process Corner	Min Delay	Min Process Corner
A[4]	result[12]	15.606	SLOW	3.316	FAST
X[3]	result[12]	15.586	SLOW	3.463	FAST
A[4]	result[14]	15.408	SLOW	3.274	FAST
A[4]	result[15]	15.394	SLOW	3.261	FAST
X[3]	result[14]	15.388	SLOW	3.419	FAST
A[0]	result[11]	15.382	SLOW	3.735	FAST
X[3]	result[15]	15.374	SLOW	3.407	FAST
A[3]	result[12]	15.302	SLOW	3.321	FAST
X[6]	result[12]	15.267	SLOW	3.113	FAST
A[3]	result[14]	15.105	SLOW	3.279	FAST
X[5]	result[13]	15.093	SLOW	2.670	FAST
A[3]	result[15]	15.091	SLOW	3.266	FAST
X[6]	result[14]	15.069	SLOW	3.071	FAST
A[5]	result[12]	15.058	SLOW	3.411	FAST
X[6]	result[15]	15.055	SLOW	2.955	FAST
A[1]	result[11]	15.054	SLOW	3.590	FAST
A[0]	result[10]	15.047	SLOW	3.835	FAST
A[2]	result[11]	15.010	SLOW	3.345	FAST
A[1]	result[10]	14.932	SLOW	3.716	FAST
X[1]	result[7]	14.918	SLOW	3.808	FAST
X[7]	result[12]	14.896	SLOW	3.235	FAST
A[4]	result[13]	14.869	SLOW	3.124	FAST
A[5]	result[14]	14.860	SLOW	3.122	FAST
X[4]	result[9]	14.857	SLOW	4.313	FAST
X[2]	result[12]	14.849	SLOW	3.494	FAST
X[3]	result[13]	14.849	SLOW	3.226	FAST
A[5]	result[15]	14.846	SLOW	3.121	FAST
X[1]	result[8]	14.812	SLOW	3.999	FAST
X[1]	result[9]	14.718	SLOW	3.971	FAST
X[7]	result[14]	14.698	SLOW	3.193	FAST
X[7]	result[15]	14.684	SLOW	2.561	FAST
X[2]	result[14]	14.652	SLOW	3.450	FAST
X[2]	result[15]	14.637	SLOW	3.438	FAST



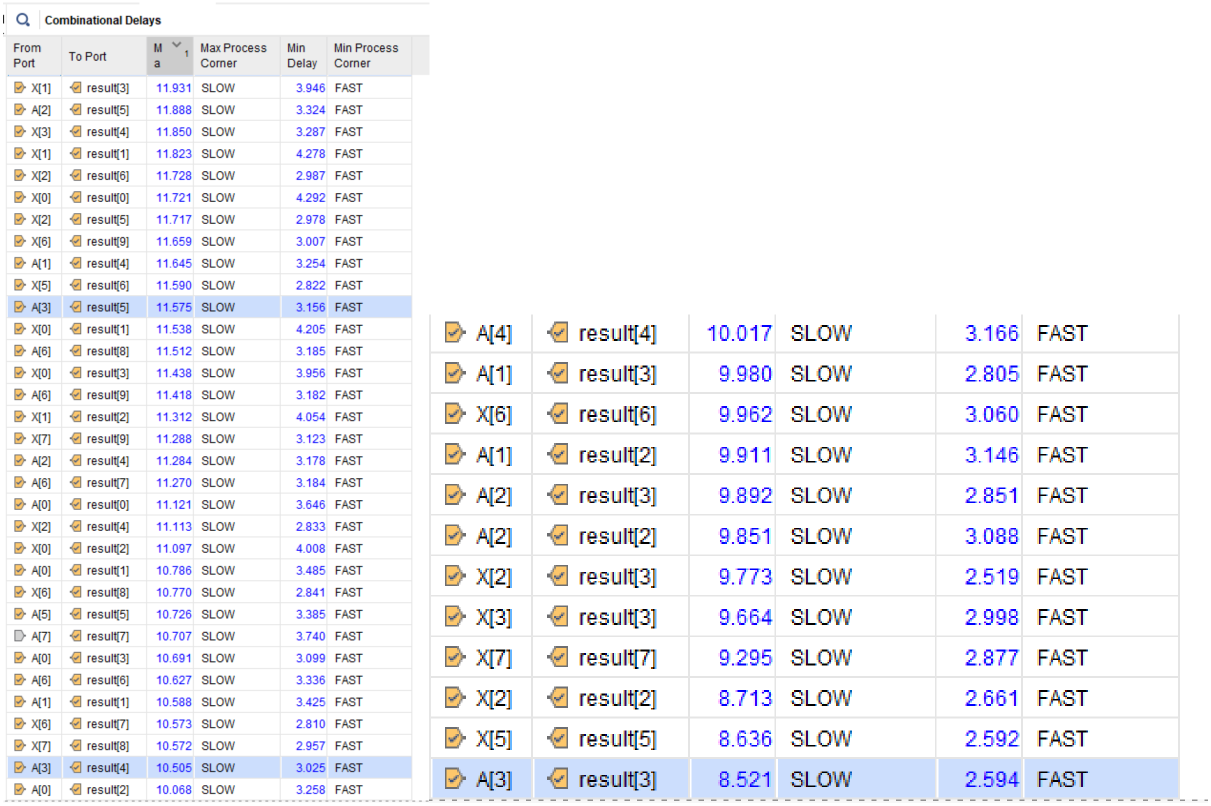
Combinational Delays

From Port	To Port	Ma	Max Process Corner	Min Delay	Min Process Corner
X[2]	result[13]	14.112	SLOW	3.257	FAST
X[4]	result[8]	13.968	SLOW	4.486	FAST
A[6]	result[14]	13.942	SLOW	2.823	FAST
A[6]	result[15]	13.928	SLOW	2.822	FAST
X[2]	result[11]	13.872	SLOW	2.696	FAST
X[5]	result[10]	13.861	SLOW	3.452	FAST
X[1]	result[5]	13.834	SLOW	3.985	FAST
X[2]	result[10]	13.814	SLOW	2.734	FAST
A[7]	result[13]	13.809	SLOW	3.883	FAST
X[0]	result[6]	13.759	SLOW	4.119	FAST
X[6]	result[11]	13.741	SLOW	3.319	FAST
A[7]	result[11]	13.712	SLOW	3.490	FAST
A[7]	result[10]	13.654	SLOW	3.854	FAST
A[1]	result[7]	13.594	SLOW	3.141	FAST
A[0]	result[7]	13.563	SLOW	3.422	FAST
X[4]	result[6]	13.539	SLOW	4.366	FAST
A[0]	result[8]	13.454	SLOW	3.397	FAST
A[6]	result[13]	13.403	SLOW	2.526	FAST
X[7]	result[11]	13.370	SLOW	3.425	FAST
A[0]	result[9]	13.359	SLOW	3.409	FAST
A[1]	result[8]	13.339	SLOW	3.292	FAST
X[6]	result[10]	13.298	SLOW	3.371	FAST
X[0]	result[5]	13.292	SLOW	4.162	FAST
A[1]	result[9]	13.245	SLOW	3.304	FAST
A[2]	result[7]	13.234	SLOW	3.226	FAST
A[6]	result[11]	13.163	SLOW	2.656	FAST
X[3]	result[7]	13.135	SLOW	3.096	FAST
A[6]	result[10]	13.105	SLOW	2.858	FAST
A[1]	result[6]	12.986	SLOW	3.173	FAST
A[2]	result[8]	12.978	SLOW	3.119	FAST
A[4]	result[8]	12.978	SLOW	3.149	FAST
X[1]	result[4]	12.969	SLOW	4.331	FAST
X[3]	result[8]	12.958	SLOW	3.311	FAST



Combinational Delays

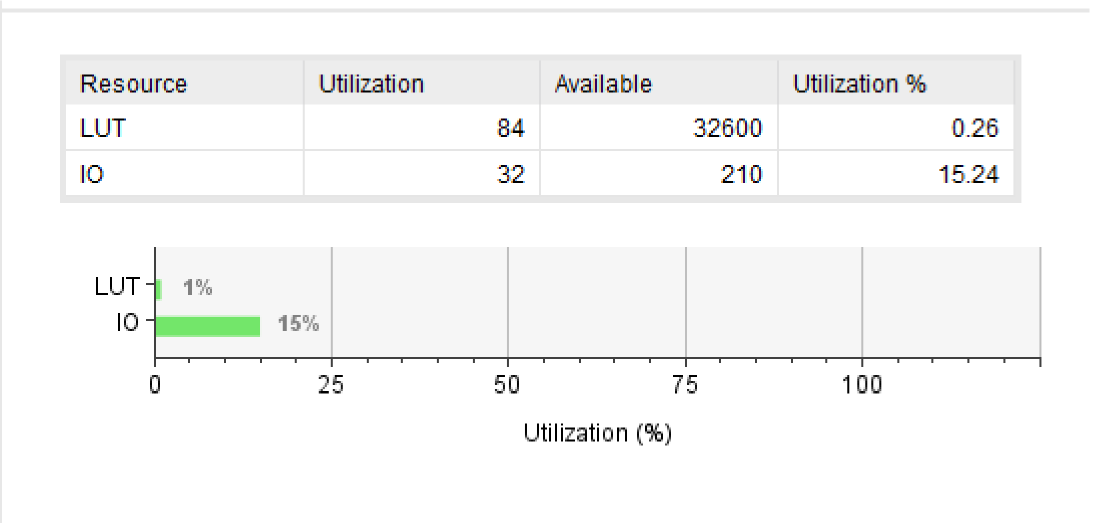
From Port	To Port	Ma	Max Process Corner	Min Delay	Min Process Corner
A[0]	result[6]	12.954	SLOW	3.278	FAST
A[2]	result[9]	12.928	SLOW	3.130	FAST
X[7]	result[10]	12.927	SLOW	3.418	FAST
A[4]	result[7]	12.892	SLOW	2.807	FAST
A[4]	result[9]	12.884	SLOW	2.925	FAST
X[3]	result[9]	12.864	SLOW	3.201	FAST
A[5]	result[7]	12.828	SLOW	3.133	FAST
A[2]	result[6]	12.625	SLOW	3.000	FAST
X[4]	result[5]	12.590	SLOW	3.992	FAST
A[5]	result[8]	12.573	SLOW	3.325	FAST
A[0]	result[5]	12.565	SLOW	2.897	FAST
A[3]	result[8]	12.554	SLOW	3.131	FAST
A[3]	result[7]	12.540	SLOW	2.935	FAST
X[3]	result[6]	12.526	SLOW	3.379	FAST
A[5]	result[9]	12.478	SLOW	3.323	FAST
X[0]	result[4]	12.469	SLOW	4.021	FAST
A[3]	result[9]	12.459	SLOW	2.919	FAST
X[3]	result[5]	12.454	SLOW	3.432	FAST
X[2]	result[7]	12.336	SLOW	2.771	FAST
A[4]	result[6]	12.283	SLOW	3.087	FAST
A[1]	result[5]	12.249	SLOW	2.824	FAST
X[5]	result[9]	12.222	SLOW	2.953	FAST
X[2]	result[8]	12.221	SLOW	2.930	FAST
A[5]	result[6]	12.219	SLOW	3.269	FAST
X[5]	result[7]	12.199	SLOW	2.706	FAST
X[2]	result[9]	12.127	SLOW	2.933	FAST
A[7]	result[8]	12.061	SLOW	3.932	FAST
A[4]	result[5]	11.999	SLOW	3.202	FAST
X[4]	result[4]	11.988	SLOW	4.218	FAST
A[7]	result[9]	11.966	SLOW	3.930	FAST
A[0]	result[4]	11.961	SLOW	3.127	FAST
X[5]	result[8]	11.943	SLOW	2.942	FAST
A[3]	result[6]	11.932	SLOW	3.041	FAST



Combinational Delays

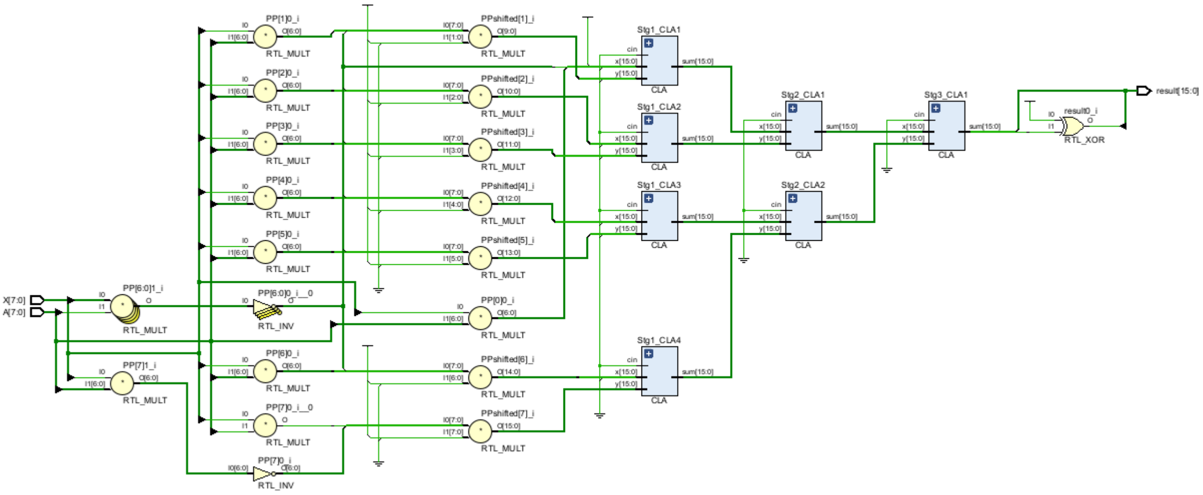
From Port	To Port	Max	Max Process Corner	Min Delay	Min Process Corner
X[1]	result[3]	11.931	SLOW	3.946	FAST
A[2]	result[5]	11.888	SLOW	3.324	FAST
X[3]	result[4]	11.850	SLOW	3.287	FAST
X[1]	result[1]	11.823	SLOW	4.278	FAST
X[2]	result[6]	11.728	SLOW	2.987	FAST
X[0]	result[0]	11.721	SLOW	4.292	FAST
X[2]	result[5]	11.717	SLOW	2.978	FAST
X[6]	result[9]	11.659	SLOW	3.007	FAST
A[1]	result[4]	11.645	SLOW	3.254	FAST
X[5]	result[6]	11.590	SLOW	2.822	FAST
A[3]	result[5]	11.575	SLOW	3.156	FAST
X[0]	result[1]	11.538	SLOW	4.205	FAST
A[6]	result[8]	11.512	SLOW	3.185	FAST
X[0]	result[3]	11.438	SLOW	3.956	FAST
A[6]	result[9]	11.418	SLOW	3.182	FAST
X[1]	result[2]	11.312	SLOW	4.054	FAST
X[7]	result[9]	11.288	SLOW	3.123	FAST
A[2]	result[4]	11.284	SLOW	3.178	FAST
A[6]	result[7]	11.270	SLOW	3.184	FAST
A[0]	result[0]	11.121	SLOW	3.646	FAST
X[2]	result[4]	11.113	SLOW	2.833	FAST
X[0]	result[2]	11.097	SLOW	4.008	FAST
A[0]	result[1]	10.786	SLOW	3.485	FAST
X[6]	result[8]	10.770	SLOW	2.841	FAST
A[5]	result[5]	10.726	SLOW	3.385	FAST
A[7]	result[7]	10.707	SLOW	3.740	FAST
A[0]	result[3]	10.691	SLOW	3.099	FAST
A[6]	result[6]	10.627	SLOW	3.336	FAST
A[1]	result[1]	10.588	SLOW	3.425	FAST
X[6]	result[7]	10.573	SLOW	2.810	FAST
X[7]	result[8]	10.572	SLOW	2.957	FAST
A[3]	result[4]	10.505	SLOW	3.025	FAST
A[0]	result[2]	10.068	SLOW	3.258	FAST
A[4]	result[4]	10.017	SLOW	3.166	FAST
A[1]	result[3]	9.980	SLOW	2.805	FAST
X[6]	result[6]	9.962	SLOW	3.060	FAST
A[1]	result[2]	9.911	SLOW	3.146	FAST
A[2]	result[3]	9.892	SLOW	2.851	FAST
A[2]	result[2]	9.851	SLOW	3.088	FAST
X[2]	result[3]	9.773	SLOW	2.519	FAST
X[3]	result[3]	9.664	SLOW	2.998	FAST
X[7]	result[7]	9.295	SLOW	2.877	FAST
X[2]	result[2]	8.713	SLOW	2.661	FAST
X[5]	result[5]	8.636	SLOW	2.592	FAST
A[3]	result[3]	8.521	SLOW	2.594	FAST

Utilization Report:

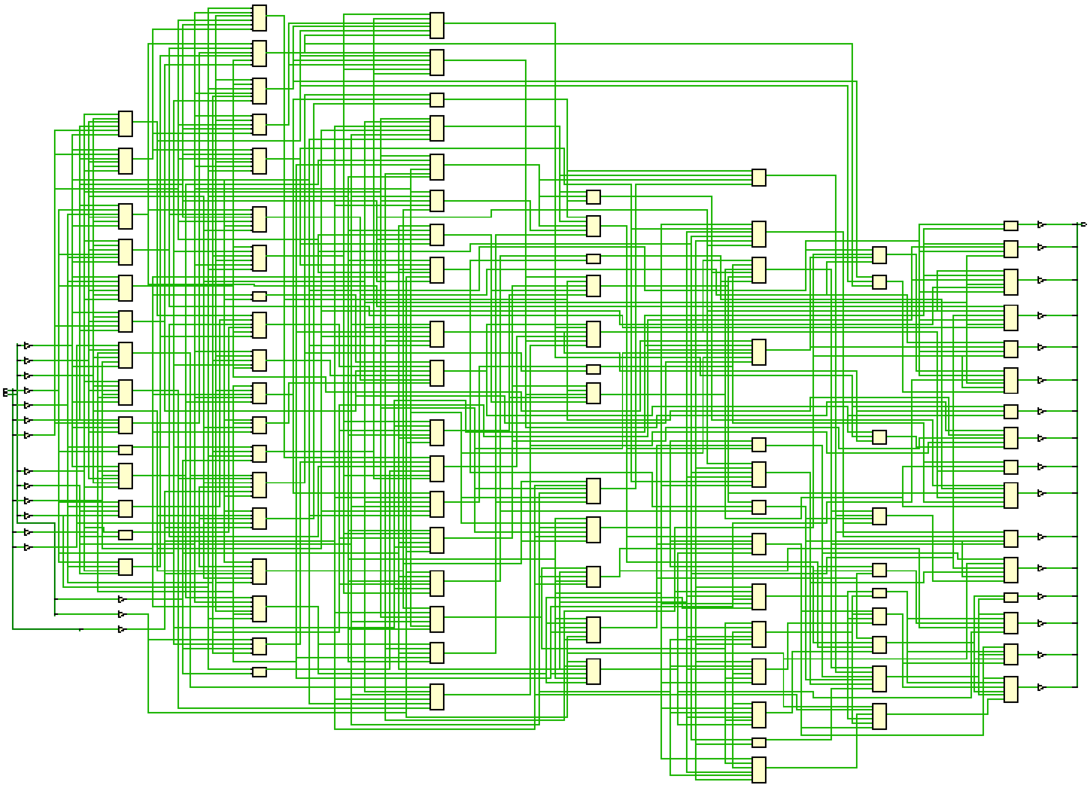


Resource	Utilization	Available	Utilization %
LUT	84	32600	0.26
IO	32	210	15.24

RTL Schematic:



Technology Schematic:



3) BEHAVIORAL MULTIPLIER

I designed behavioral multiplier in this part by using always begin structure. Result output goes to A*B in every change of A or B.

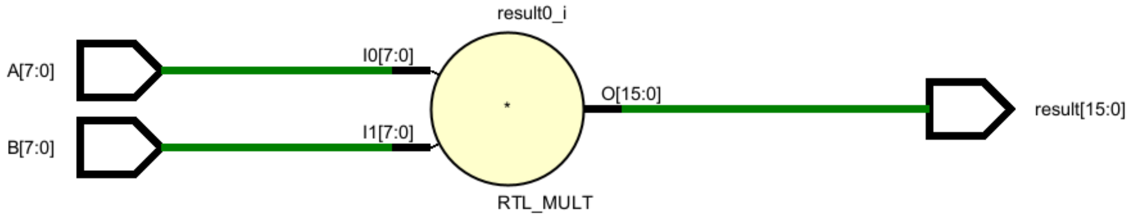
Source code:

```
`timescale 1ns / 1ps

module MULTB(
    input signed [7:0] A,
    input signed [7:0] B,
    output reg signed [15:0] result

    );

always @(*) begin

result <= A*B;

end
endmodule
```

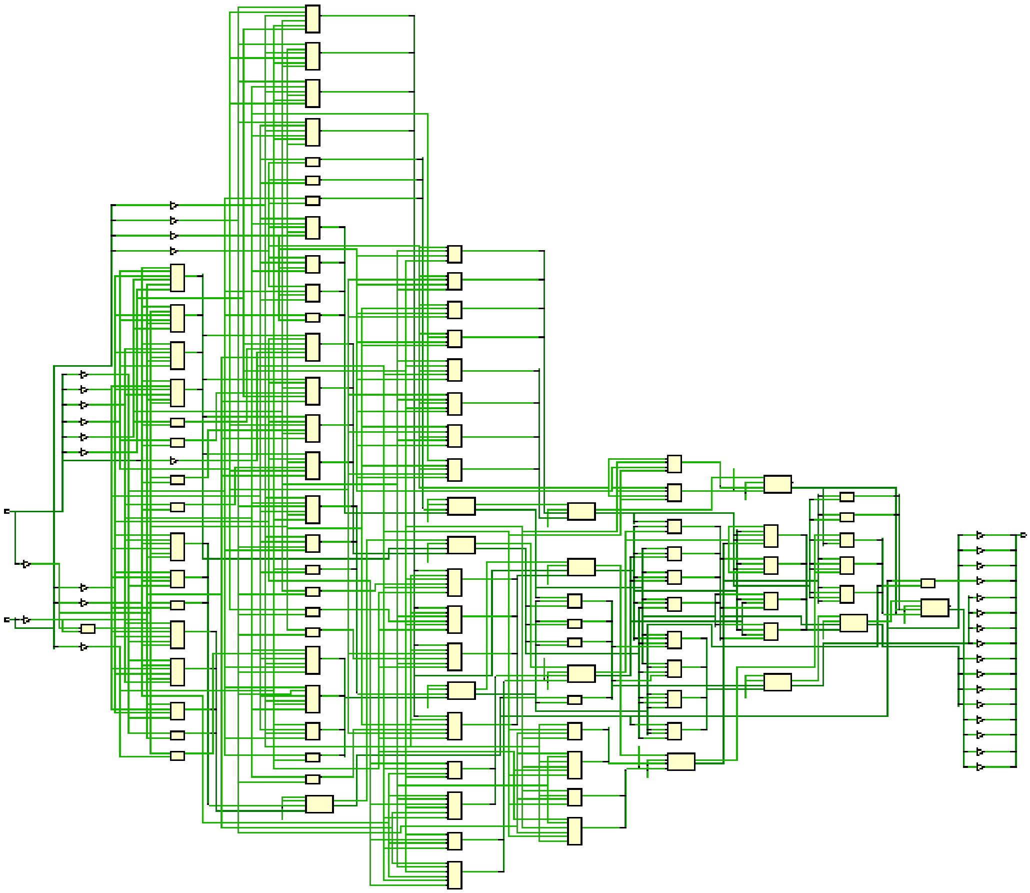
Testbench code:

```
`timescale 1ns / 1ps

module MULTS_tb;

reg signed [7:0] A;
reg signed [7:0] X;
reg signed [15:0] real_result;
wire signed [15:0] result;

MULTB dut (.A(A),.B(X),.result(result));
integer i;
initial begin
#10;

for (i=0;i<=20;i=i+1)
    begin
        A=7*i;
        X=11*i;
        real_result = A*X;
        #10;
        if((A*X) == (result))
        $display("%d * %d = %d
",$signed(A),$signed(X),$signed(real_result),"and result is %d
",$signed(result));
        else
        $display("Operation is wrong");
    end
end

endmodule
```

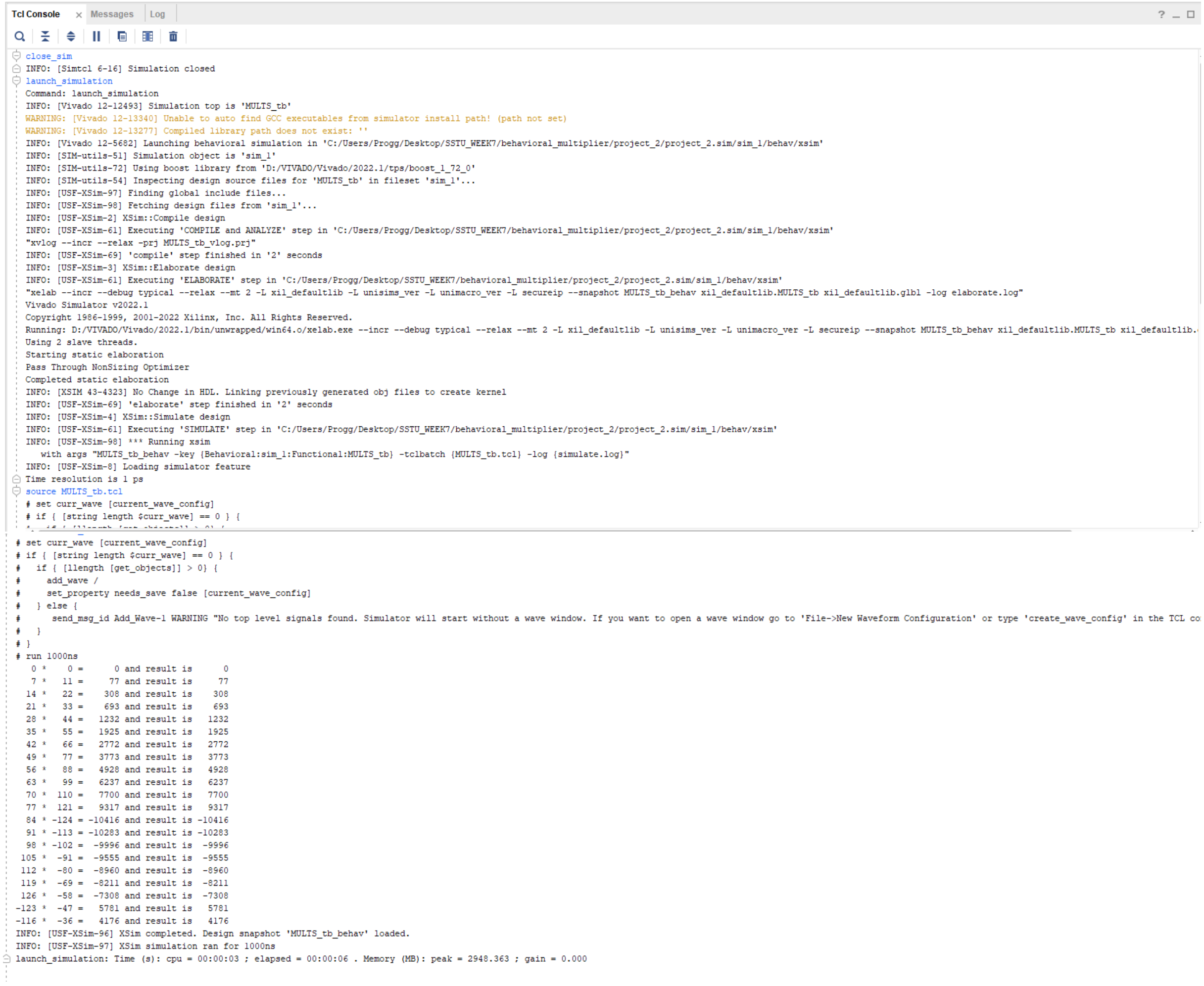
RTL Schematic:



Technology Schematic:



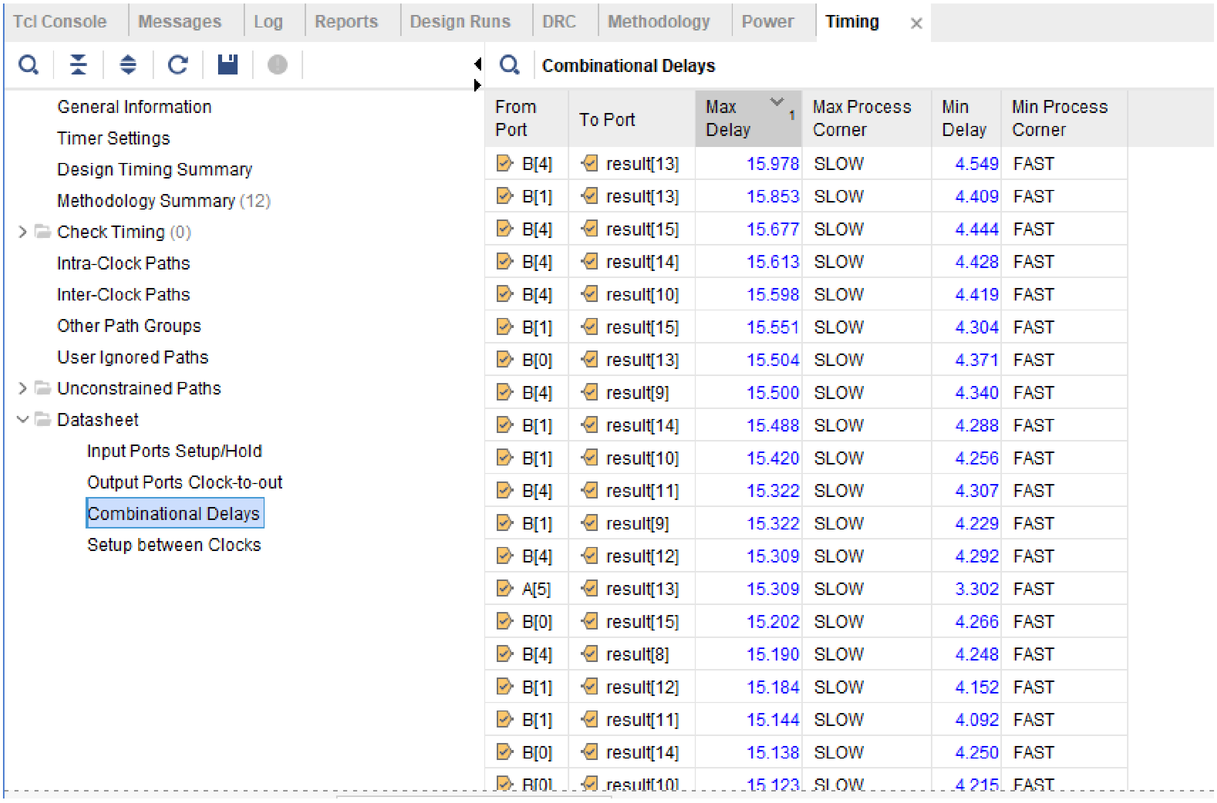
TCL Console output:



```
close_sim
INFO: [Simtcl 6-16] Simulation closed
launch_simulation
Command: launch_simulation
INFO: [Vivado 12-12493] Simulation top is 'MULTS_tb'
WARNING: [Vivado 12-13340] Unable to auto find GCC executables from simulator install path! (path not set)
WARNING: [Vivado 12-13277] Compiled library path does not exist: ''
INFO: [Vivado 12-5682] Launching behavioral simulation in 'C:/Users/Progg/Desktop/SSTU_WEEK7/behavioral_multiplier/project_2/project_2.sim/sim_1/behav/xsim'
INFO: [SIM-utils-51] Simulation object is 'sim_1'
INFO: [SIM-utils-72] Using boost library from 'D:/VIVADO/Vivado/2022.1/tps/boost_1_72_0'
INFO: [SIM-utils-54] Inspecting design source files for 'MULTS_tb' in fileset 'sim_1'...
INFO: [USF-XSim-97] Finding global include files...
INFO: [USF-XSim-98] Fetching design files from 'sim_1'...
INFO: [USF-XSim-2] XSim::Compile design
INFO: [USF-XSim-61] Executing 'COMPILE and ANALYZE' step in 'C:/Users/Progg/Desktop/SSTU_WEEK7/behavioral_multiplier/project_2/project_2.sim/sim_1/behav/xsim'
"xvlog --incr --relax -prj MULTS_tb_vlog.prj"
INFO: [USF-XSim-69] 'compile' step finished in '2' seconds
INFO: [USF-XSim-3] XSim::Elaborate design
INFO: [USF-XSim-61] Executing 'ELABORATE' step in 'C:/Users/Progg/Desktop/SSTU_WEEK7/behavioral_multiplier/project_2/project_2.sim/sim_1/behav/xsim'
"xelab --incr --debug typical --relax --mt 2 -L xil_defaultlib -L unisims_ver -L unimacro_ver -L secureip --snapshot MULTS_tb_behav xil_defaultlib.MULTS_tb xil_defaultlib.glbl -log elaborate.log"
Vivado Simulator v2022.1
Copyright 1986-1999, 2001-2022 Xilinx, Inc. All Rights Reserved.
Running: D:/VIVADO/Vivado/2022.1/bin/unwrapped/win64.o/xelab.exe --incr --debug typical --relax --mt 2 -L xil_defaultlib -L unisims_ver -L unimacro_ver -L secureip --snapshot MULTS_tb_behav xil_defaultlib.MULTS_tb xil_defaultlib.
Using 2 slave threads.
Starting static elaboration
Pass Through NonSizing Optimizer
Completed static elaboration
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '2' seconds
INFO: [USF-XSim-4] XSim::Simulate design
INFO: [USF-XSim-61] Executing 'SIMULATE' step in 'C:/Users/Progg/Desktop/SSTU_WEEK7/behavioral_multiplier/project_2/project_2.sim/sim_1/behav/xsim'
INFO: [USF-XSim-98] *** Running xsim
   with args "MULTS_tb_behav -key {Behavioral:sim_1:Functional:MULTS_tb} -tclbatch {MULTS_tb.tcl} -log {simulate.log}"
INFO: [USF-XSim-8] Loading simulator feature
Time resolution is 1 ps
source MULTS_tb.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
```

```
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration' or type 'create_wave_config' in the TCL co
#   }
# }
# run 1000ns
   0 *    0 =      0 and result is      0
   7 *   11 =     77 and result is     77
  14 *   22 =    308 and result is    308
  21 *   33 =    693 and result is    693
  28 *   44 =   1232 and result is   1232
  35 *   55 =   1925 and result is   1925
  42 *   66 =   2772 and result is   2772
  49 *   77 =   3773 and result is   3773
  56 *   88 =   4928 and result is   4928
  63 *   99 =   6237 and result is   6237
  70 *  110 =   7700 and result is   7700
  77 *  121 =   9317 and result is   9317
  84 * -124 = -10416 and result is -10416
  91 * -113 = -10283 and result is -10283
  98 * -102 =  -9996 and result is  -9996
 105 *  -91 =  -9555 and result is  -9555
 112 *  -80 =  -8960 and result is  -8960
 119 *  -69 =  -8211 and result is  -8211
 126 *  -58 =  -7308 and result is  -7308
-123 *  -47 =   5781 and result is   5781
-116 *  -36 =   4176 and result is   4176
INFO: [USF-XSim-96] XSim completed. Design snapshot 'MULTS_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:06 . Memory (MB): peak = 2948.363 ; gain = 0.000
```

Timing Report:



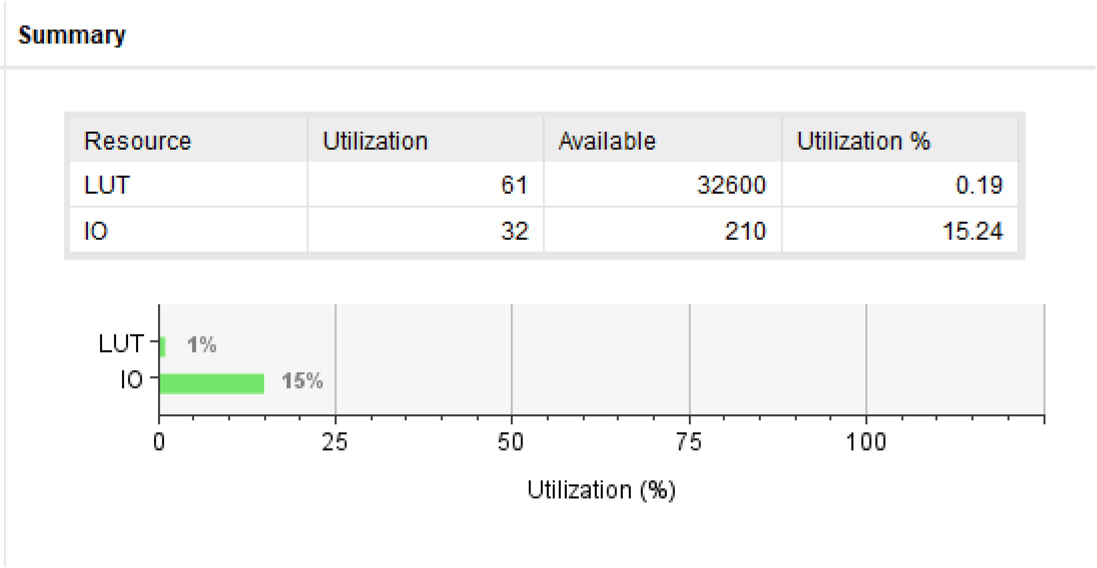
Combinational Delays

- General Information
- Timer Settings
- Design Timing Summary
- Methodology Summary (12)
- Check Timing (0)
- Intra-Clock Paths
- Inter-Clock Paths
- Other Path Groups
- User Ignored Paths
- Unconstrained Paths
- Datasheet
  - Input Ports Setup/Hold
  - Output Ports Clock-to-out
  - Combinational Delays
  - Setup between Clocks

From Port	To Port	Max Delay	Max Process Corner	Min Delay	Min Process Corner
B[4]	result[13]	15.978	SLOW	4.549	FAST
B[1]	result[13]	15.853	SLOW	4.409	FAST
B[4]	result[15]	15.677	SLOW	4.444	FAST
B[4]	result[14]	15.613	SLOW	4.428	FAST
B[4]	result[10]	15.598	SLOW	4.419	FAST
B[1]	result[15]	15.551	SLOW	4.304	FAST
B[0]	result[13]	15.504	SLOW	4.371	FAST
B[4]	result[9]	15.500	SLOW	4.340	FAST
B[1]	result[14]	15.488	SLOW	4.288	FAST
B[1]	result[10]	15.420	SLOW	4.256	FAST
B[4]	result[11]	15.322	SLOW	4.307	FAST
B[1]	result[9]	15.322	SLOW	4.229	FAST
B[4]	result[12]	15.309	SLOW	4.292	FAST
A[5]	result[13]	15.309	SLOW	3.302	FAST
B[0]	result[15]	15.202	SLOW	4.266	FAST
B[4]	result[8]	15.190	SLOW	4.248	FAST
B[1]	result[12]	15.184	SLOW	4.152	FAST
B[1]	result[11]	15.144	SLOW	4.092	FAST
B[0]	result[14]	15.138	SLOW	4.250	FAST
B[0]	result[10]	15.123	SLOW	4.215	FAST

Max Delay is 15.978 nanoseconds, which is less than my older implementations.

Utilization Report:

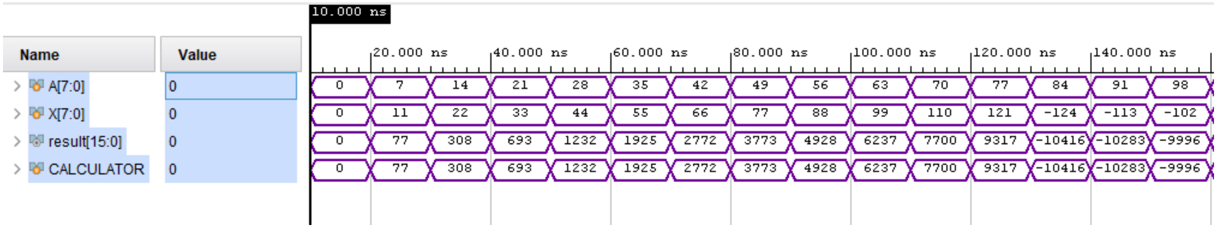


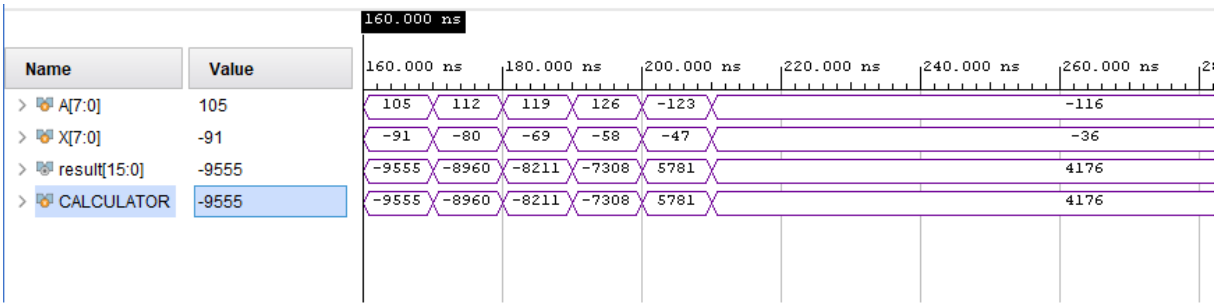
Summary

Resource	Utilization	Available	Utilization %
LUT	61	32600	0.19
IO	32	210	15.24

Utilization percentage is better than my old implementation too.

Behavioral Simulation Results:





Comparison:

Behavioral design is much easier to code, faster and requires less space.

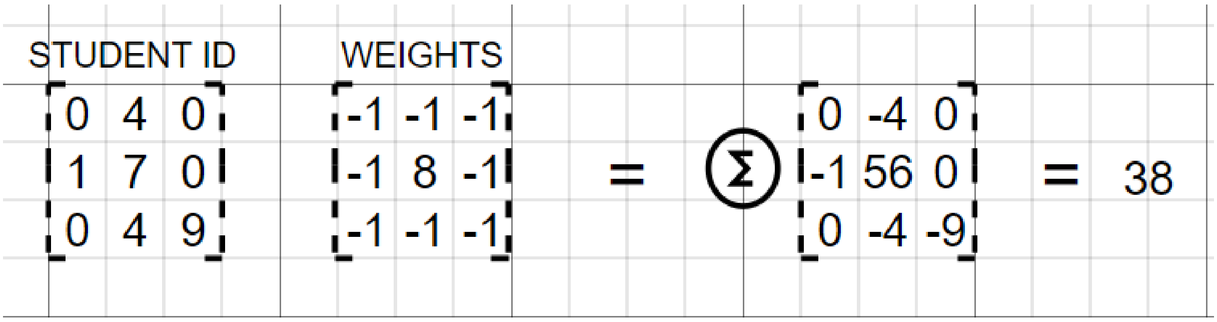
	Behavioral	Structural
Timing (Worse Scenerio)	15.978 nS	18.46 nS
Utilization	61 LUT	84 LUT

In my parametrical RCA design 130 LUTs used and my design has 21 nS worse scenerio delay. Behaviorals max clock frequency is 62.58 MHz. Structural was almost 54.15 MHz.

4) Multiply and Accumulate (MAC)

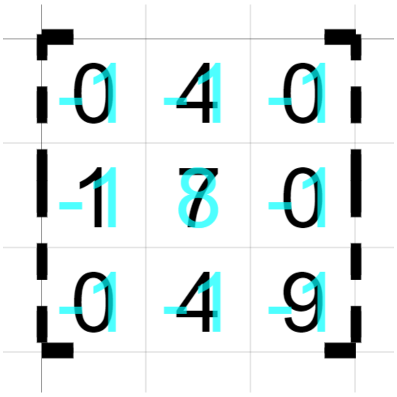
In this part, i designed a MAC. First part of the circuit is combinational, second part is sequantial.

Here is the result should be taken:



$$\begin{array}{cc} \text{STUDENT ID} & \text{WEIGHTS} \\ \begin{bmatrix} 0 & 4 & 0 \\ 1 & 7 & 0 \\ 0 & 4 & 9 \end{bmatrix} & \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \end{array} = \Sigma \begin{bmatrix} 0 & -4 & 0 \\ -1 & 56 & 0 \\ 0 & -4 & -9 \end{bmatrix} = 38$$

We are simply doing this multiplication and getting sum of them:



We got a counter in the circuit shows that how many accumulate we did. Because of our input is 3x3 and kernel matrix is 3x3, output will be a number. But according to there is no array input allowence in verilog, we'll make these operations as 3x1 to 3x1 then get sum of all of them.

When the counter goes 3, we'll get the right result.

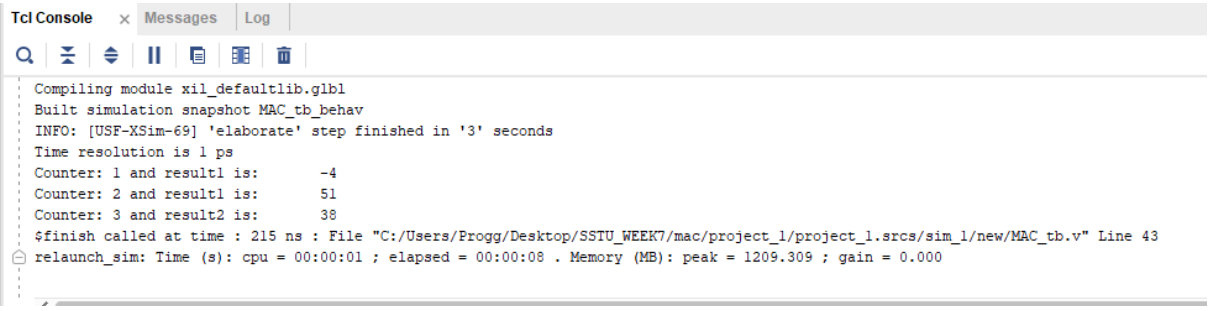
Source Code:

```
`timescale 1ns / 1ps


module MAC(
        input [23:0] data,
        input clk,
        input rst,
        input [23:0] weight,
        output reg signed [19:0] result
    );

wire signed [15:0] product0,product1;
wire signed [16:0] product2;
wire signed [16:0] sum0;
wire signed [17:0] sum1;
assign product2[16] = product2[15]? 1'b1:1'b0;
reg [1:0] count;
MULTB m1 (.A(data[7:0]),.B(weight[7:0]),.result(product0));
MULTB m2 (.A(data[15:8]),.B(weight[15:8]),.result(product1));
MULTB m3 (.A(data[23:16]),.B(weight[23:16]),.result(product2));

ADDB #(16) add1 (.A(product0),.B(product1),.result(sum0));
ADDB #(17) add2 (.A(sum0),.B(product2),.result(sum1));
always @(posedge clk or posedge rst) begin
    if(rst)
    begin
        count <=2'd0;
        result <=20'd0;
    end
    else
        begin
        result <= result + sum1;
        count <= count+1;
        end
    end
endmodule

`timescale 1ns / 1ps

module ADDB#(parameter SIZE=16)(
    input signed [(SIZE-1):0] A,
    input signed [(SIZE-1):0] B,
    output reg signed [SIZE:0] result
    );
always @(*) begin
result <= A+B;
end
endmodule


module MULTB(
    input signed [7:0] A,
    input signed [7:0] B,
    output reg signed [15:0] result
    );
always @(*) begin
result <= A*B;
end
endmodule
```

Testbench Code:

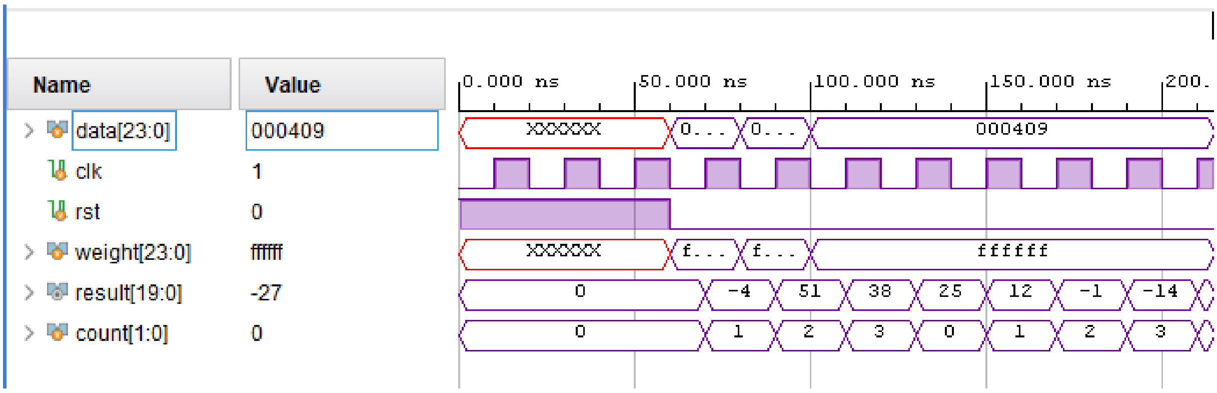
```
`timescale 1ns / 1ps


module MAC_tb;

reg [23:0] data;
reg clk=0;
reg rst=1;
reg signed [23:0] weight;
wire signed [19:0] result;

MAC dut
(.clk(clk),.data(data),.rst(rst),.weight(weight),.result(result));

always begin
#10;
clk = ~clk;
end

reg [1:0] counter =0;
initial begin
rst=1;
#60;
rst=0;
data = 24'b0000_0000__0000_0100__0000_0000; // 040
weight = 24'b11111111__11111111__11111111; // -1 -1 -1
counter = counter+1;
#15;
$display("Counter: ", counter, " and result1 is: ", result);
#5;
data = 24'b0000_0001__0000_0111__0000_0000; // 170
weight = 24'b11111111__0000_1000__11111111; // -1 8 -1
counter = counter+1;
#15;
$display("Counter: ", counter, " and result1 is: ", result);
#5;
data = 24'b0000_0000__0000_0100__0000_1001; // 049
weight = 24'b11111111__11111111__11111111; // -1 -1 -1
counter = counter+1;
#15;
$display("Counter: ", counter, " and result2 is: ", result);
#100;
$finish;
end
endmodule
```

TCL Console output:



```
Compiling module xil_defaultlib.glbl
Built simulation snapshot MAC_tb_behav
INFO: [USF-XSim-69] 'elaborate' step finished in '3' seconds
Time resolution is 1 ps
Counter: 1 and result1 is:       -4
Counter: 2 and result1 is:       51
Counter: 3 and result2 is:       38
$finish called at time : 215 ns : File "C:/Users/Progg/Desktop/SSTU_WEEK7/mac/project_1/project_1.srcs/sim_1/new/MAC_tb.v" Line 43
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:08 . Memory (MB): peak = 1209.309 ; gain = 0.000
```

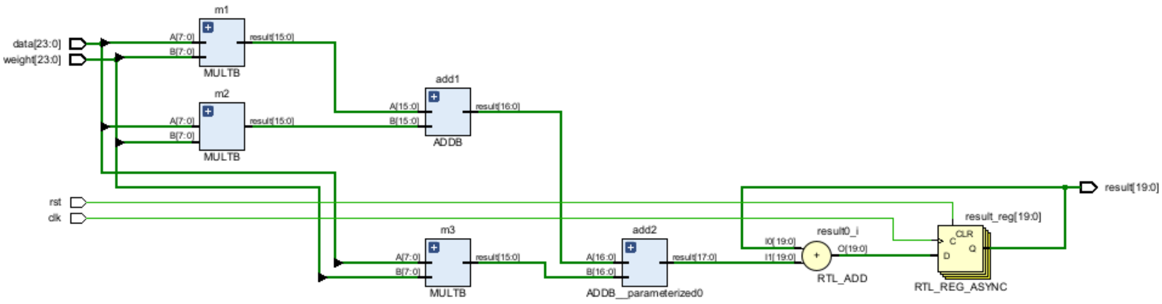
Behavioral Simulation Results:



Name	Value
data[23:0]	000409
clk	1
rst	0
weight[23:0]	ffffff
result[19:0]	-27
count[1:0]	0

As we can see above, the true result 38 comes in 3rd positive edge of clock when rst=0.

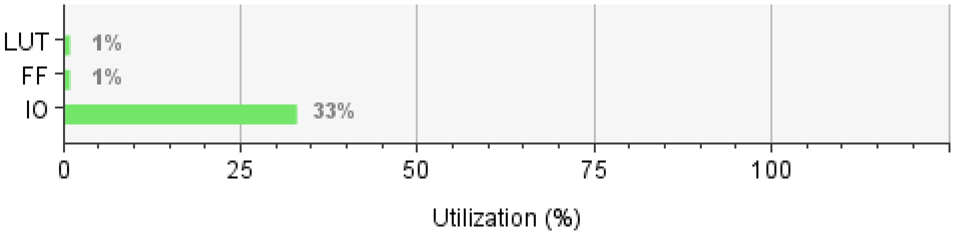
RTL Schematic:



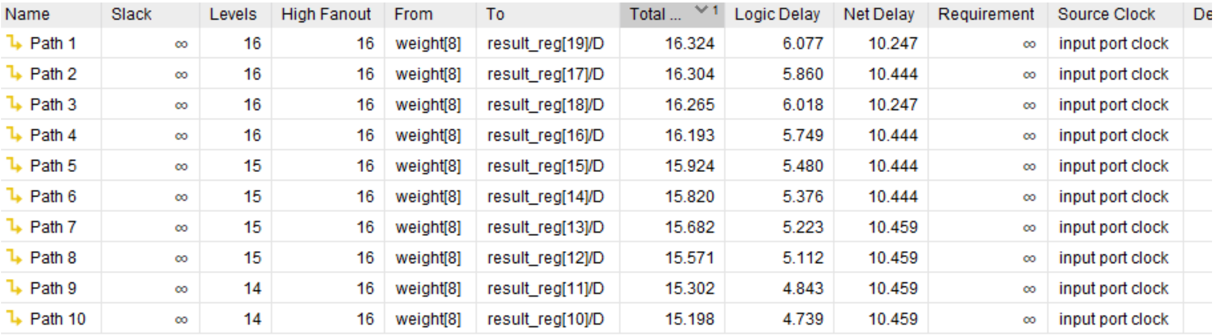
Utilization Report:

Summary

Resource	Utilization	Available	Utilization %
LUT	232	32600	0.71
FF	20	65200	0.03
IO	70	210	33.33



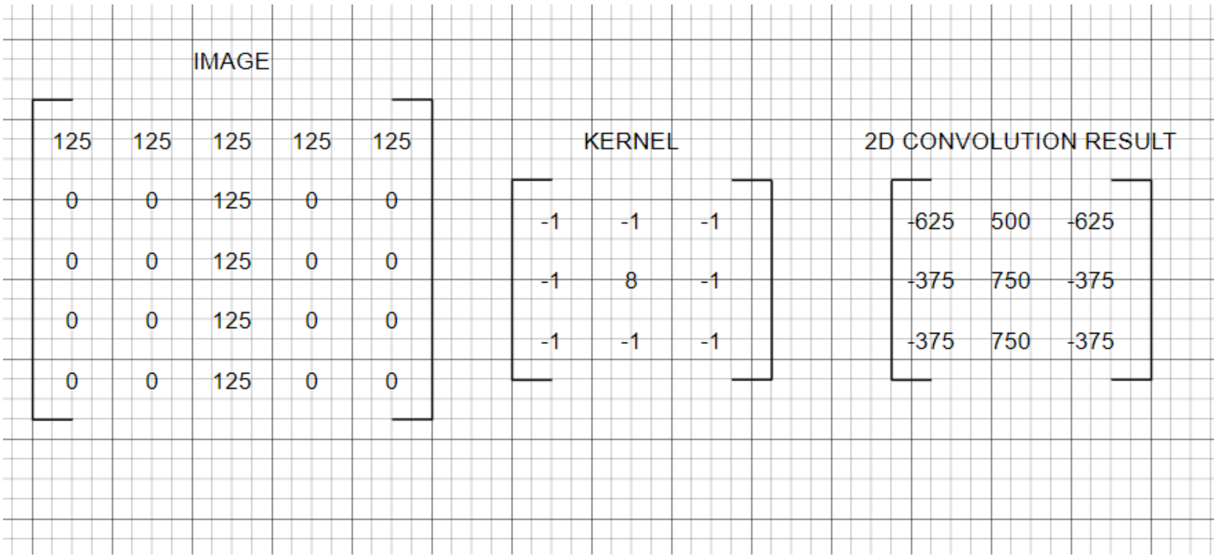
Timing Report:



Name	Slack	Levels	High Fanout	From	To	Total ...	Logic Delay	Net Delay	Requirement	Source Clock	De
Path 1	∞	16	16	weight[8]	result_reg[19]/D	16.324	6.077	10.247	∞	input port clock	
Path 2	∞	16	16	weight[8]	result_reg[17]/D	16.304	5.860	10.444	∞	input port clock	
Path 3	∞	16	16	weight[8]	result_reg[18]/D	16.265	6.018	10.247	∞	input port clock	
Path 4	∞	16	16	weight[8]	result_reg[16]/D	16.193	5.749	10.444	∞	input port clock	
Path 5	∞	15	16	weight[8]	result_reg[15]/D	15.924	5.480	10.444	∞	input port clock	
Path 6	∞	15	16	weight[8]	result_reg[14]/D	15.820	5.376	10.444	∞	input port clock	
Path 7	∞	15	16	weight[8]	result_reg[13]/D	15.682	5.223	10.459	∞	input port clock	
Path 8	∞	15	16	weight[8]	result_reg[12]/D	15.571	5.112	10.459	∞	input port clock	
Path 9	∞	14	16	weight[8]	result_reg[11]/D	15.302	4.843	10.459	∞	input port clock	
Path 10	∞	14	16	weight[8]	result_reg[10]/D	15.198	4.739	10.459	∞	input port clock	

5) 2D CONVOLUTION

We will make kernel move in the image and calculate 3x3 multiply and accumulate operation



IMAGE

$$\begin{bmatrix} 125 & 125 & 125 & 125 & 125 \\ 0 & 0 & 125 & 0 & 0 \\ 0 & 0 & 125 & 0 & 0 \\ 0 & 0 & 125 & 0 & 0 \\ 0 & 0 & 125 & 0 & 0 \end{bmatrix}$$

KERNEL

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

2D CONVOLUTION RESULT

$$\begin{bmatrix} -625 & 500 & -625 \\ -375 & 750 & -375 \\ -375 & 750 & -375 \end{bmatrix}$$

I did not understand that will we give datas for one time, or we'll give needed data everytime. So i designed that module for GETTING DATAS ONCE, save them in a image register, then calculate the 2D convolution of it. For this operation, i changed my previous designs, because there were posedge delays in add, multiply and in the MAC.

What we doing is basicly is shifting weights matrix (kernel) inside of the image matrix.

Mathematical formula is:

$$g(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} w(i,j) f(x-i, y-j)$$

Simply i changed the modules by this:

Adder and Multiplier:

```
module ADDB#(parameter SIZE=16)(
    input signed [(SIZE-1):0] A,
    input signed [(SIZE-1):0] B,
    output signed [SIZE:0] result
    );
assign result = A+B;
endmodule


module MULTB(
    input signed [7:0] A,
    input signed [7:0] B,
    output signed [15:0] result
    );
assign result = A*B;
endmodule
```

MAC:

```
module MAC(
        input [23:0] data,
        input clk,
        input rst,
        input [23:0] weight,
        output reg signed [19:0] resultout
    );
wire signed [15:0] product0,product1;
wire signed [16:0] product2;
wire signed [16:0] sum0;
wire signed [17:0] sum1;
assign product2[16] = product2[15]? 1'b1:1'b0;
reg signed [19:0] result=0;
reg [1:0] count;
MULTB m1 (.A(data[7:0]),.B(weight[7:0]),.result(product0));
MULTB m2 (.A(data[15:8]),.B(weight[15:8]),.result(product1));
MULTB m3 (.A(data[23:16]),.B(weight[23:16]),.result(product2));

ADDB #(16) add1 (.A(product0),.B(product1),.result(sum0));
ADDB #(17) add2 (.A(sum0),.B(product2),.result(sum1));
always @(posedge clk or posedge rst) begin
    if(rst)
    begin
        count <=2'd0;
        result <=20'd0;
    end
    else
        begin
        count <= count+1;
        if(count == 1 || count == 2)
        result <= result + sum1;
        if(count == 3)
        begin
        resultout<=result + sum1;
        result<=20'd0;
        end
        end
    end
endmodule
```

2D Convolution Module:

```
`timescale 1ns / 1ps

module dconv(
input [39:0] data,
input signed [23:0] KERNEL,
input rst,
input clk

    );
// Image will be : [255 255 255 255 255]
//                 [0   0   255 0   0  ]
//                 [0   0   255 0   0  ]
//                 [0   0   255 0   0  ]
//                 [0   0   255 0   0  ]

reg [39:0] IMAGE [4:0];
reg signed [59:0] CONV2D [2:0];
wire signed [19:0] result1;
reg [23:0] IMAGESHIFT;
reg signed [23:0] KERNELIN [2:0];
reg signed [23:0] KERNELSHIFT;
reg [2:0] row=0;
reg [4:0] process=0;
reg [1:0] writerow=0;
reg [2:0] process_counter=0;
reg loaded=0;
reg finish=0;


MAC MAC1
(.data(IMAGESHIFT),.weight(KERNELSHIFT),.clk(clk),.rst(!loaded),.result
out(result1));

always @(posedge clk) begin
    if(rst)
    begin
    IMAGESHIFT <=0;
    KERNELSHIFT <=0;
    row <=0;
    process <=0;
    writerow <=0;
    finish<=0;
    end
    else
    begin
    if(!loaded & !finish)
    begin
        IMAGE[row] <=data;
        KERNELIN[row] <= KERNEL;

            if(row == 4)
                begin
                loaded<=1'b1;
                row<=0;
                end
            else
                row <=row+1;
    end
    if(loaded & !finish)
        begin
        case(process)
        0:
            begin
            if(row<=2)
            begin
            IMAGESHIFT <= IMAGE[row][39:16];
            KERNELSHIFT <= KERNELIN[row];
            row <= row+1;
            end
            process_counter <=process_counter+1;
            if(process_counter==4)
            begin
                process <=1;
                process_counter <=1;
                CONV2D[writerow][59:40]<=result1;
                row<=0;
            end
            end

         1:
            begin
            if(row<=2)
            begin
            IMAGESHIFT <= IMAGE[row][31:8];
            KERNELSHIFT <= KERNELIN[row];
            row <= row+1;
            end
            process_counter <=process_counter+1;
            if(process_counter==4)
            begin
                process <=2;
                CONV2D[writerow][39:20]<=result1;
                process_counter<=1;
                row<=0;
            end
            end

         2:begin
            if(row<=2)
            begin
            IMAGESHIFT <= IMAGE[row][23:0];
            KERNELSHIFT <= KERNELIN[row];
            row <= row+1;
            end
            process_counter <=process_counter+1;
            if(process_counter==4)
            begin
                process <=3;
                CONV2D[writerow][19:0]<=result1;
                process_counter<=1;
                row<=1;
                writerow<=writerow+1;
            end
            end

         3:begin
            if(row<=3)
            begin
            IMAGESHIFT <= IMAGE[row][39:16];
            KERNELSHIFT <= KERNELIN[row-1];
            row <= row+1;
            end
            process_counter <=process_counter+1;
            if(process_counter==4)
            begin
                process <=4;
                CONV2D[writerow][59:40]<=result1;
                process_counter<=1;
                row<=1;
            end
            end
         4:begin
            if(row<=3)
            begin
            IMAGESHIFT <= IMAGE[row][31:8];
            KERNELSHIFT <= KERNELIN[row-1];
            row <= row+1;
            end
            process_counter <=process_counter+1;
            if(process_counter==4)
            begin
                process <=5;
                CONV2D[writerow][39:20]<=result1;
                process_counter<=1;
                row<=1;
            end
            end
          5:begin
            if(row<=3)
            begin
            IMAGESHIFT <= IMAGE[row][23:0];
            KERNELSHIFT <= KERNELIN[row-1];
            row <= row+1;
            end
            process_counter <=process_counter+1;
            if(process_counter==4)
            begin
                process <=6;
                CONV2D[writerow][19:0]<=result1;
                process_counter<=1;
                row<=2;
                writerow<=writerow+1;
            end
            end
          6:begin
            if(row<=4)
            begin
            IMAGESHIFT <= IMAGE[row][39:16];
            KERNELSHIFT <= KERNELIN[row-2];
            row <= row+1;
            end
            process_counter <=process_counter+1;
            if(process_counter==4)
            begin
                process <=7;
                CONV2D[writerow][59:40]<=result1;
                process_counter<=1;
                row<=2;
            end
            end
```

```
           7:begin
            if(row<=4)
            begin
            IMAGESHIFT <= IMAGE[row][31:8];
            KERNELSHIFT <= KERNELIN[row-2];
            row <= row+1;
            end
            process_counter <=process_counter+1;
            if(process_counter==4)
            begin
                process <=8;
                CONV2D[writerow][39:20]<=result1;
                process_counter<=1;
                row<=2;
            end
            end

           8: begin
            if(row<=4)
            begin
            IMAGESHIFT <= IMAGE[row][23:0];
            KERNELSHIFT <= KERNELIN[row-2];
            row <= row+1;
            end
            process_counter <=process_counter+1;
            if(process_counter==4)
            begin
                process <=0;
                finish<=1;
                CONV2D[writerow][19:0]<=result1;
                process_counter<=1;
                row<=2;
                writerow<=0;
            end
            end

        endcase


        end

end
end
endmodule
```

Testbench Code:

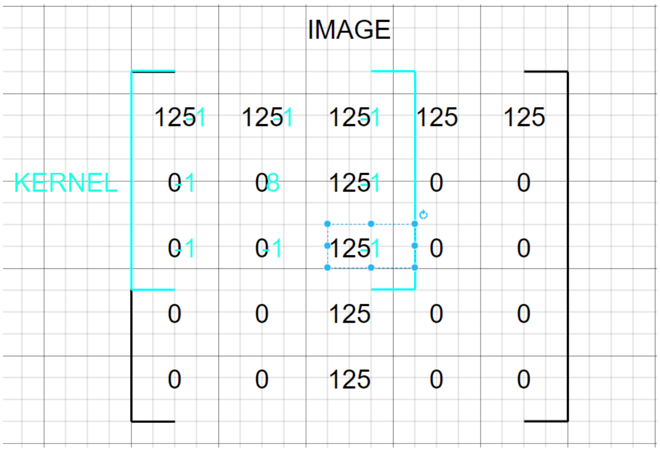
```
`timescale 1ns / 1ps


module dconv_tb;

reg clk=0;
reg [39:0] data;
reg signed [23:0] KERNEL;
reg reset = 0;
always begin
#10;
clk = ~clk;
end

dconv CONV2D (.data(data),.rst(reset),.KERNEL(KERNEL),.clk(clk));


initial begin
reset = 1;
#100;
KERNEL = 24'b1111_1111_1111_1111_1111_1111;
data = 40'h5E5E5E5E5E;
#100;
reset = 0;
#15;
KERNEL = 24'b1111_1111_0000_1000_1111_1111;
data = 40'h00005E0000;
#5;
#15;
KERNEL = 24'b1111_1111_1111_1111_1111_1111;
data = 40'h00005E0000;
#20;
data = 40'h00005E0000;
#20;
#5;
data = 40'h00005E0000;
#100;
#10;



end
endmodule
```

Testbench Results:

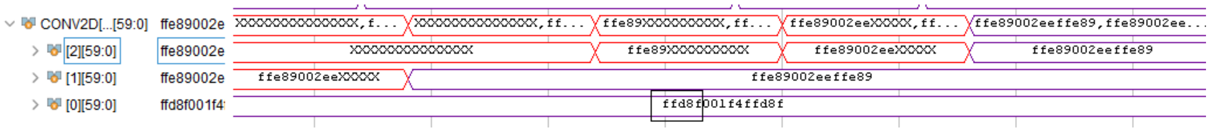
```
`timescale 1ns / 1ps


module dconv_tb;

reg clk=0;
reg [39:0] data;
reg signed [23:0] KERNEL;
reg reset = 0;
always begin
#10;
clk = ~clk;
end

dconv CONV2D (.data(data),.rst(reset),.KERNEL(KERNEL),.clk(clk));


initial begin
reset = 1;
#100;
KERNEL = 24'b1111_1111_1111_1111_1111_1111;
data = 40'h5E5E5E5E5E;
#100;
reset = 0;
#15;
KERNEL = 24'b1111_1111_0000_1000_1111_1111;
data = 40'h00005E0000;
#5;
#15;
KERNEL = 24'b1111_1111_1111_1111_1111_1111;
data = 40'h00005E0000;
#20;
data = 40'h00005E0000;
#20;
#5;
data = 40'h00005E0000;
#100;
#10;



end
endmodule
```

Explain how 2D-Convolution can be calculated by sliding kernel over the image:

1st step:

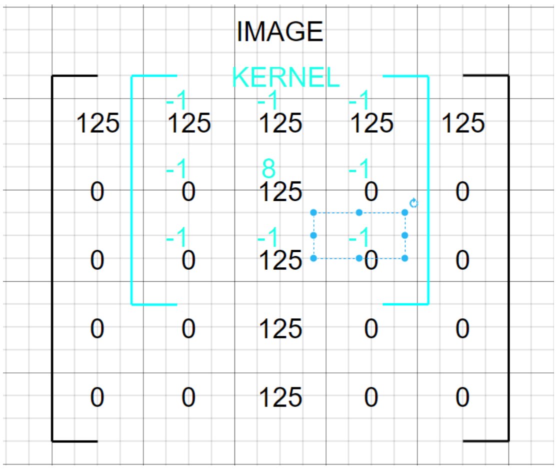


This is the first operation. We multiply kernels and images values one by one as that and we get sum of product of them. This result is the value of the 1st row 1st column of result matrix. The result should be -125-125-125-125-125 = -625

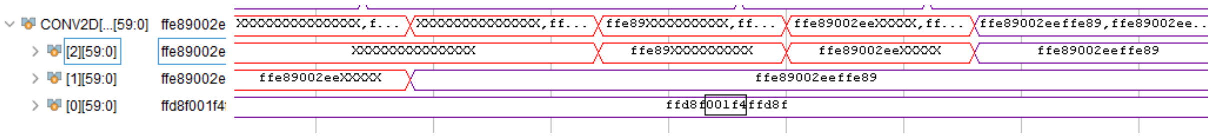

CONV2D[...[59:0] ffe89002e
[2][59:0] ffe89002e
[1][59:0] ffe89002e
[0][59:0] ffd8f001f4
XXXXXXXXXXXXXXX,f... XXXXXXXXXXXXXXX,ff... ffe89XXXXXXXXXX,ff... ffe89002eeXXXXX,ff... ffe89002eeffe89,ffe89002ee...
XXXXXXXXXXXXXXX ffe89XXXXXXXXXX ffe89002eeXXXXX ffe89002eeffe89
ffe89002eeXXXXX ffe89002eeffe89
ffd8f001f4ffd8f

and my result is -625 too. (FFD8F is equal to -625 signed decimal)

2nd step:

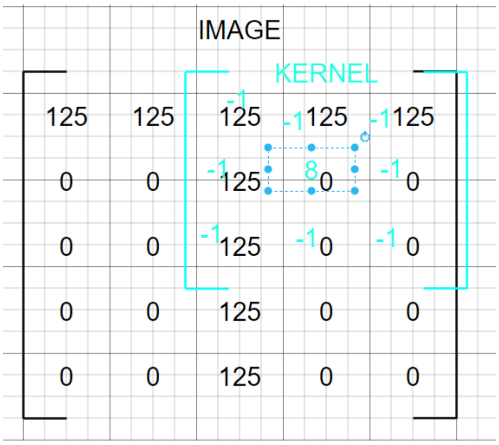


We shifted the kernel in image and the result here is -125-125-125+1000-125 = 500

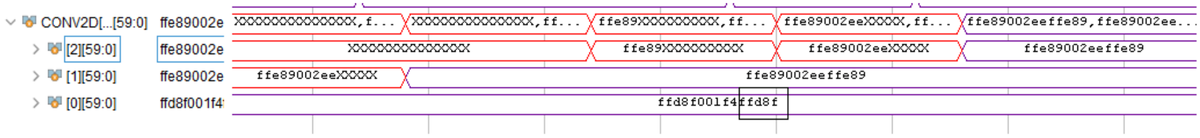

CONV2D[...[59:0] ffe89002e
[2][59:0] ffe89002e
[1][59:0] ffe89002e
[0][59:0] ffd8f001f4
XXXXXXXXXXXXXXX,f... XXXXXXXXXXXXXXX,ff... ffe89XXXXXXXXXX,ff... ffe89002eeXXXXX,ff... ffe89002eeffe89,ffe89002ee...
XXXXXXXXXXXXXXX ffe89XXXXXXXXXX ffe89002eeXXXXX ffe89002eeffe89
ffe89002eeXXXXX ffe89002eeffe89
ffd8f001f4ffd8f

and my result is 500. (001F4 is equal to 500 signed decimal)

3rd step:

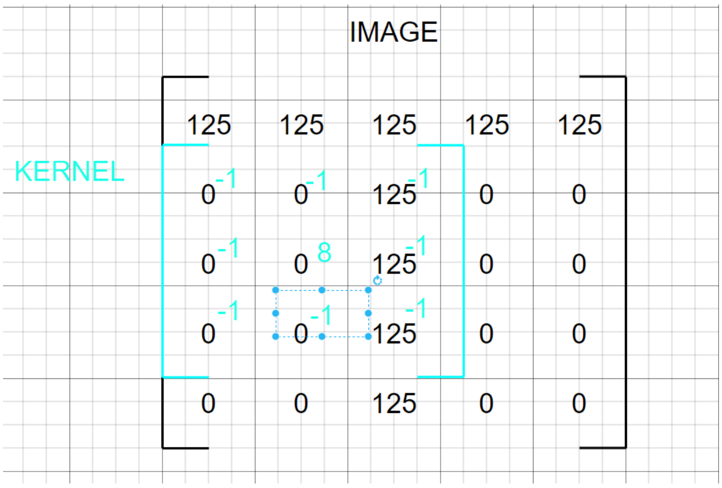


In this step kernel is shifted again. Result should be -125-125-125-125-125 = -625

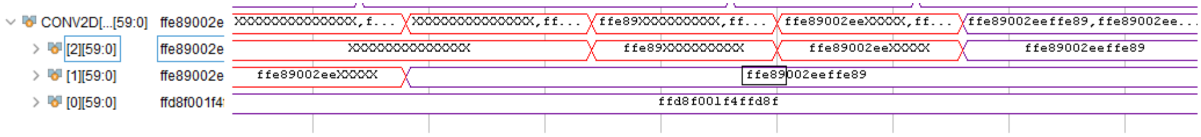

```
CONV2D[...[59:0]  ffe89002e  XXXXXXXXXXXXXXX,f...  XXXXXXXXXXXXXXX,ff...  ffe89XXXXXXXXXX,ff...  ffe89002eeXXXXX,ff...  ffe89002eeffe89,ffe89002ee...
  [2][59:0]       ffe89002e  XXXXXXXXXXXXXXX  ffe89XXXXXXXXXX  ffe89002eeXXXXX  ffe89002eeffe89
  [1][59:0]       ffe89002e  ffe89002eeXXXXX  ffe89002eeffe89
  [0][59:0]       ffd8f001f4  ffd8f001f4ffd8f
```

and my result is -625 (FFD8F is equal to -625 signed decimal)

4th step:

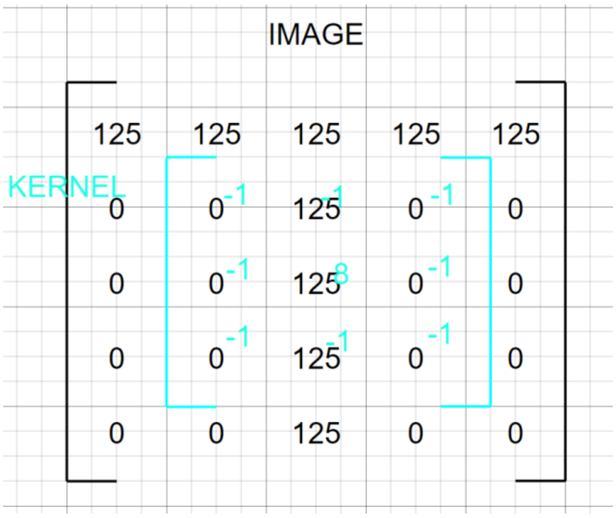


In this step kernel is shifted one row below. In here result should be -375.

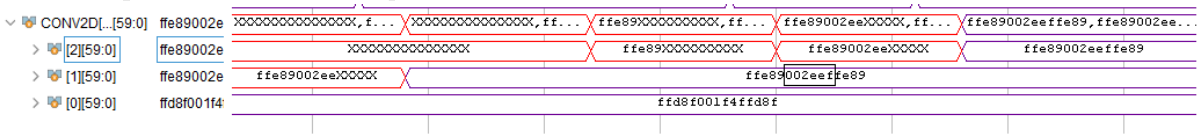

```
CONV2D[...[59:0]  ffe89002e  XXXXXXXXXXXXXXX,f...  XXXXXXXXXXXXXXX,ff...  ffe89XXXXXXXXXX,ff...  ffe89002eeXXXXX,ff...  ffe89002eeffe89,ffe89002ee...
  [2][59:0]       ffe89002e  XXXXXXXXXXXXXXX  ffe89XXXXXXXXXX  ffe89002eeXXXXX  ffe89002eeffe89
  [1][59:0]       ffe89002e  ffe89002eeXXXXX  ffe89002eeffe89
  [0][59:0]       ffd8f001f4  ffd8f001f4ffd8f
```

And my result is -375 (FFE89 is equal to -375 signed decimal)

5th step:



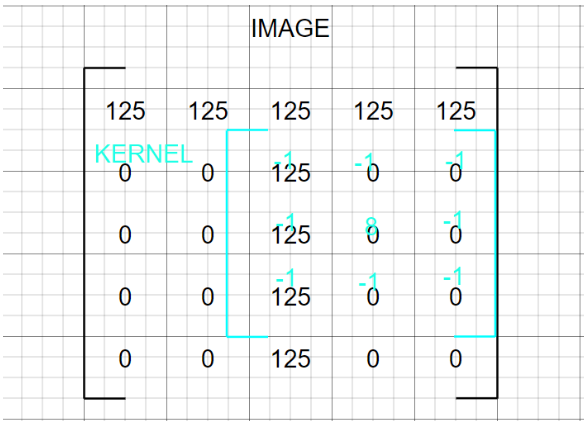
In this step, result should be equal to -125-125+1000 = 750



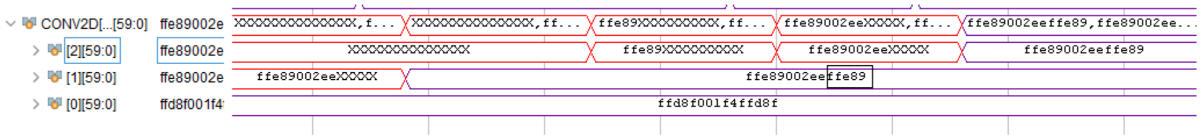
```
CONV2D[...[59:0]  ffe89002e  XXXXXXXXXXXXXXX,f...  XXXXXXXXXXXXXXX,ff...  ffe89XXXXXXXXXX,ff...  ffe89002eeXXXXX,ff...  ffe89002eeffe89,ffe89002ee...
  [2][59:0]       ffe89002e  XXXXXXXXXXXXXXX  ffe89XXXXXXXXXX  ffe89002eeXXXXX  ffe89002eeffe89
  [1][59:0]       ffe89002e  ffe89002eeXXXXX  ffe89002eeffe89
  [0][59:0]       ffd8f001f4  ffd8f001f4ffd8f
```

And my result is 750.(002EEF is equal to 750 signed decimal)

6th step:



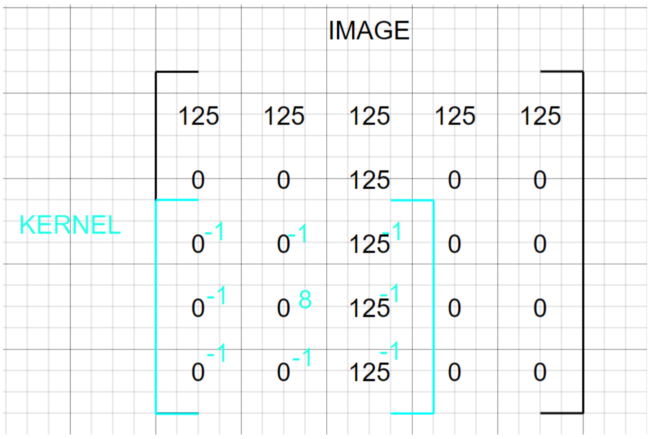
In this step result should be equal to -375



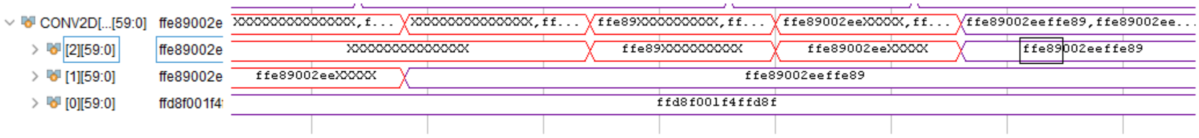
```
CONV2D[...[59:0]  ffe89002e  XXXXXXXXXXXXXXX,f...  XXXXXXXXXXXXXXX,ff...  ffe89XXXXXXXXXX,ff...  ffe89002eeXXXXX,ff...  ffe89002eeffe89,ffe89002ee...
  [2][59:0]       ffe89002e  XXXXXXXXXXXXXXX  ffe89XXXXXXXXXX  ffe89002eeXXXXX  ffe89002eeffe89
  [1][59:0]       ffe89002e  ffe89002eeXXXXX  ffe89002eeffe89
  [0][59:0]       ffd8f001f4  ffd8f001f4ffd8f
```

And my result is -375.(FFE89 is equal to -375 signed decimal)

7th step:



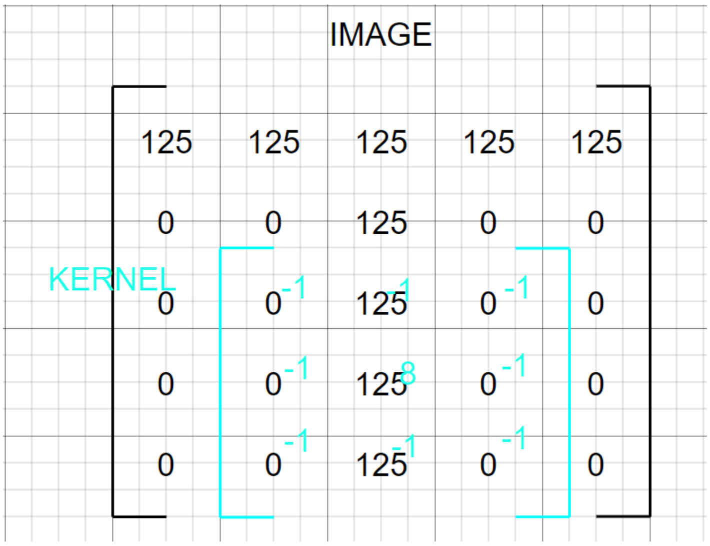
In this step result should be equal to -375.



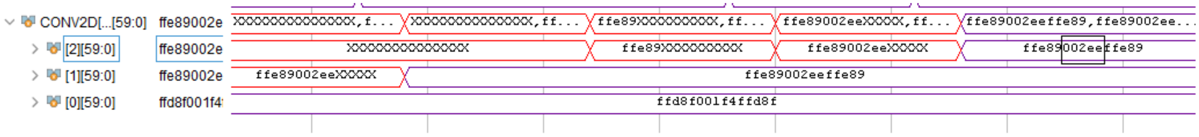
```
CONV2D[...[59:0]  ffe89002e  XXXXXXXXXXXXXXX,f...  XXXXXXXXXXXXXXX,ff...  ffe89XXXXXXXXXX,ff...  ffe89002eeXXXXX,ff...  ffe89002eeffe89,ffe89002ee...
  [2][59:0]       ffe89002e  XXXXXXXXXXXXXXX  ffe89XXXXXXXXXX  ffe89002eeXXXXX  ffe89002eeffe89
  [1][59:0]       ffe89002e  ffe89002eeXXXXX  ffe89002eeffe89
  [0][59:0]       ffd8f001f4  ffd8f001f4ffd8f
```

And my result is equal to -375. (FFE89 is equal to -375 signed decimal)

8th step:



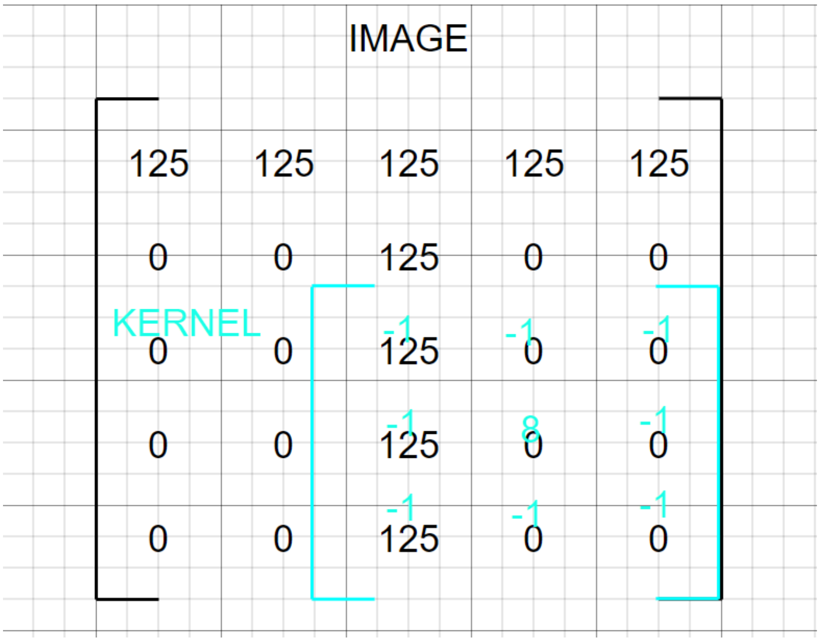
In this step result should be equal to 1000-250=750.



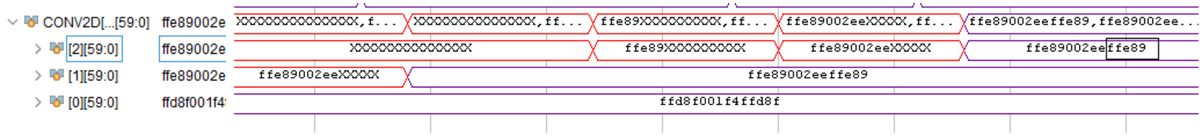
```
CONV2D[...[59:0]  ffe89002e  XXXXXXXXXXXXXXX,f...  XXXXXXXXXXXXXXX,ff...  ffe89XXXXXXXXXX,ff...  ffe89002eeXXXXX,ff...  ffe89002eeffe89,ffe89002ee...
  [2][59:0]       ffe89002e  XXXXXXXXXXXXXXX  ffe89XXXXXXXXXX  ffe89002eeXXXXX  ffe89002eeffe89
  [1][59:0]       ffe89002e  ffe89002eeXXXXX  ffe89002eeffe89
  [0][59:0]       ffd8f001f4  ffd8f001f4ffd8f
```

And my result is equal to 750 (002EE is equal to 750 signed decimal)

9th step:



In this step, result should be equal to -375.



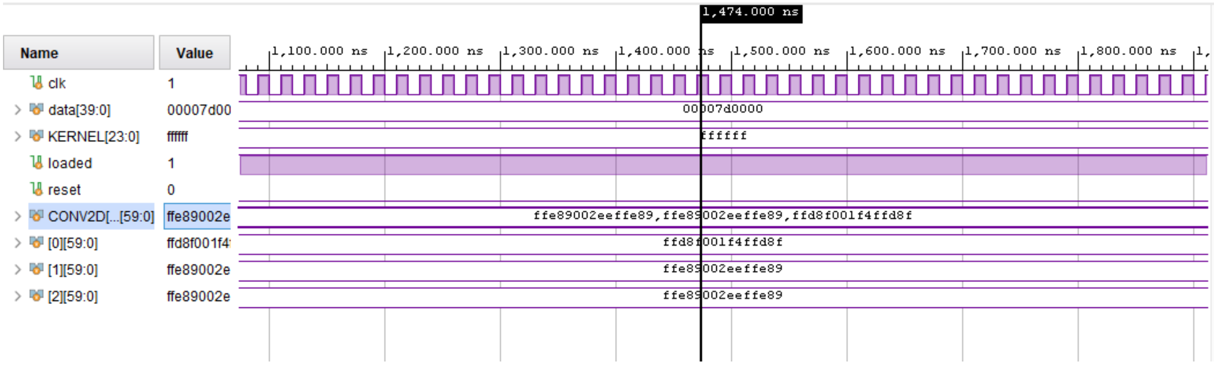
```
CONV2D[...[59:0]  ffe89002e  XXXXXXXXXXXXXXX,f...  XXXXXXXXXXXXXXX,ff...  ffe89XXXXXXXXXX,ff...  ffe89002eeXXXXX,ff...  ffe89002eeffe89,ffe89002ee...
  [2][59:0]       ffe89002e  XXXXXXXXXXXXXXX  ffe89XXXXXXXXXX  ffe89002eeXXXXX  ffe89002eeffe89
  [1][59:0]       ffe89002e  ffe89002eeXXXXX  ffe89002eeffe89
  [0][59:0]       ffd8f001f4  ffd8f001f4ffd8f
```

And my result is -375(FFE89 is equal to -375 signed decimal).

So steps are these. To explain my code, i made the clock delay calculations and give the datas of image just once. Because i don't give anything else but image data in testbench, i don't have any TCL command output too. By just looking from waveform it can be seen everything is clearly true.

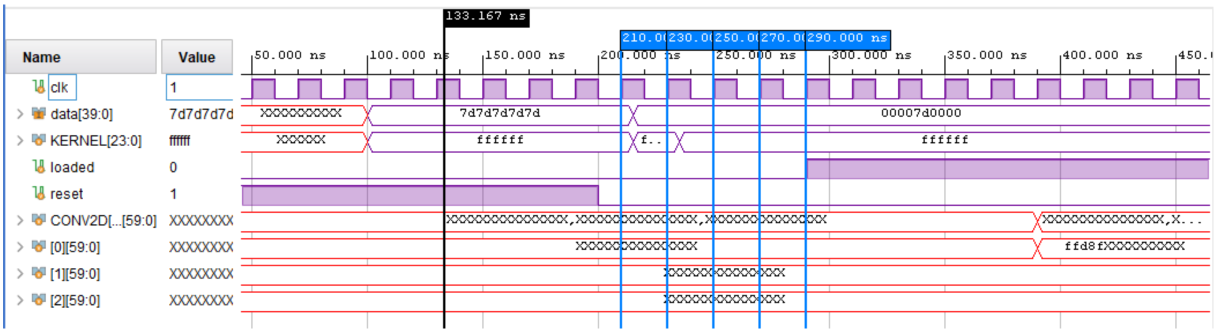
Behavioral Simulation Results:

Output when calculation ends.

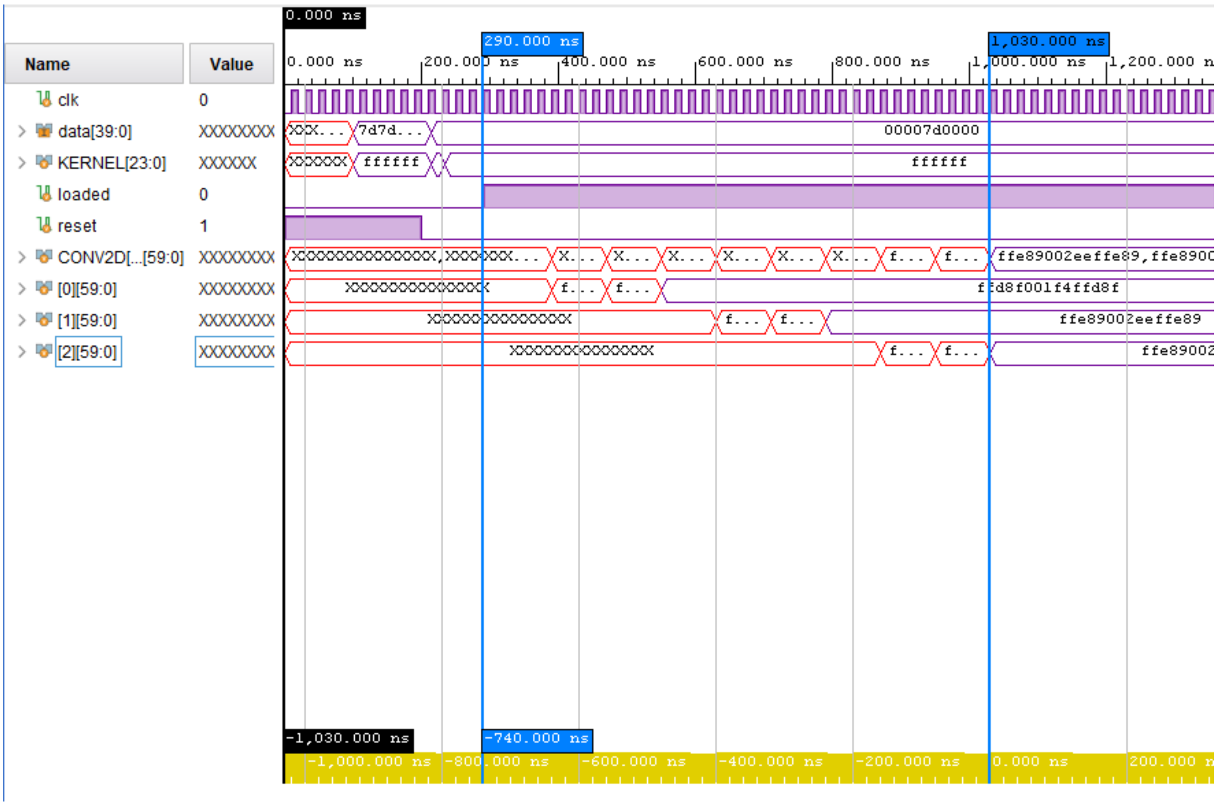


```
Name               Value
clk                1
data[39:0]         00007d00     00007d0000
KERNEL[23:0]       ffffff       ffffff
loaded             1
reset              0
CONV2D[...[59:0]   ffe89002e    ffe89002eeffe89,ffe89002eeffe89,ffd8f001f4ffd8f
  [0][59:0]        ffd8f001f4   ffd8f001f4ffd8f
  [1][59:0]        ffe89002e    ffe89002eeffe89
  [2][59:0]        ffe89002e    ffe89002eeffe89
```

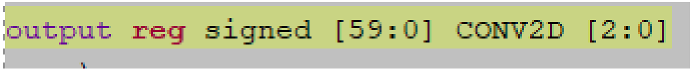
Here we can see 5 data is loaded than loaded register goes 1. Image is readen correct.



Calculation is ended in 740 nanoseconds after image loaded.



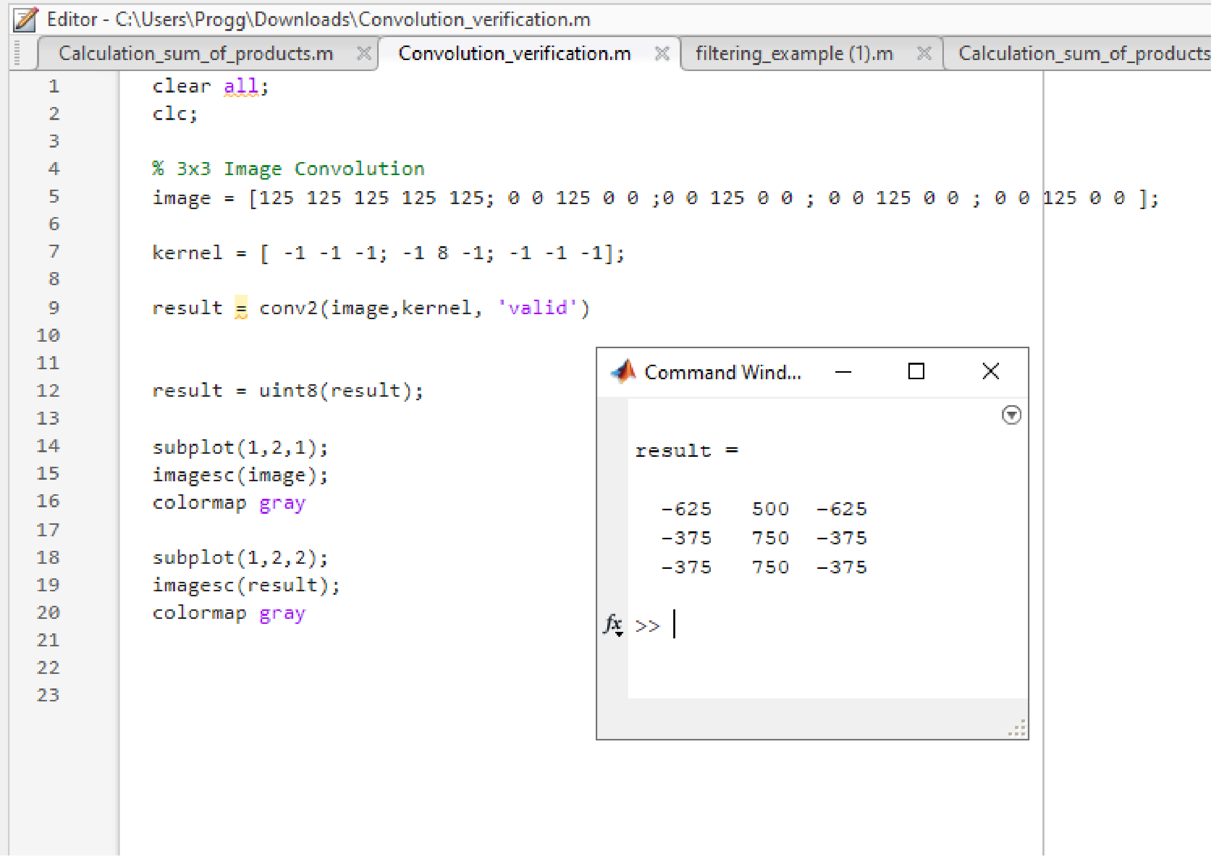
As i declared, i have no TCL console output because **i give the data rows of image just once. I just have to find the times of changes and write $display, but that has no meaning because everything can be seen and it is not necessary because i get the results in a conv2d array. I need the data of it, but i can't get it because**



```
output reg signed [59:0] CONV2D [2:0]
```

i can't define an array as an output.

Matlab 2-D Convolution calculator results:


Editor - C:\Users\Progg\Downloads\Convolution_verification.m
Calculation_sum_of_products.m
Convolution_verification.m
filtering_example (1).m
Calculation_sum_of_products
Command Wind...

```
clear all;
clc;

% 3x3 Image Convolution
image = [125 125 125 125 125; 0 0 125 0 0 ;0 0 125 0 0 ; 0 0 125 0 0 ; 0 0 125 0 0 ];

kernel = [ -1 -1 -1; -1 8 -1; -1 -1 -1];

result = conv2(image,kernel, 'valid')


result = uint8(result);

subplot(1,2,1);
imagesc(image);
colormap gray

subplot(1,2,2);
imagesc(result);
colormap gray
```

```
result =

  -625   500  -625
  -375   750  -375
  -375   750  -375

>>
```

As i declared, everything is same and true.