Digital IC Design

CHENCHETY ABHISHEK

EE20BTECH11012

Assignment-1

Question-2

- Implement f= x0h0 + x1h1 + x2h2 +x3h3 + x4h4 + x5h5 + x6h6 + x7h7 + x8h8 + x9h9 keeping in mind that **no two different arithmetic operations will take place in same clock edge**.

Verilog code

```
// Evaluating final result at every alternate posedge of clk

// Declaring module with the required in and out parameters
module
Q2(out,clk,adder_clk,x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,h_0,h_1,h_2,h_3,h_4,h
_5,h_6,h_7,h_8,h_9);

// Given input of 4 bit
input
[3:0]x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,h_0,h_1,h_2,h_3,h_4,h_5,h_6,h_7,h_8,h
_9;

// expected 11 bit output
output reg [10:0]out;

//Creating Registers for storing multiplied values
reg [7:0]m0,m1,m2,m3,m4,m5,m6,m7,m8,m9;

// Count var for implementation control

input clk;
input adder_clk;

always @(posedge clk ) begin
        m0<=x_0*h_0;
        m1<=x_1*h_1;
        m2<=x_2*h_2;
        m3<=x_3*h_3;
        m4<=x_4*h_4;
        m5<=x_5*h_5;
        m6<=x_6*h_6;
        m7<=x_7*h_7;
        m8<=x_8*h_8;
        m9<=x_9*h_9;
end
```

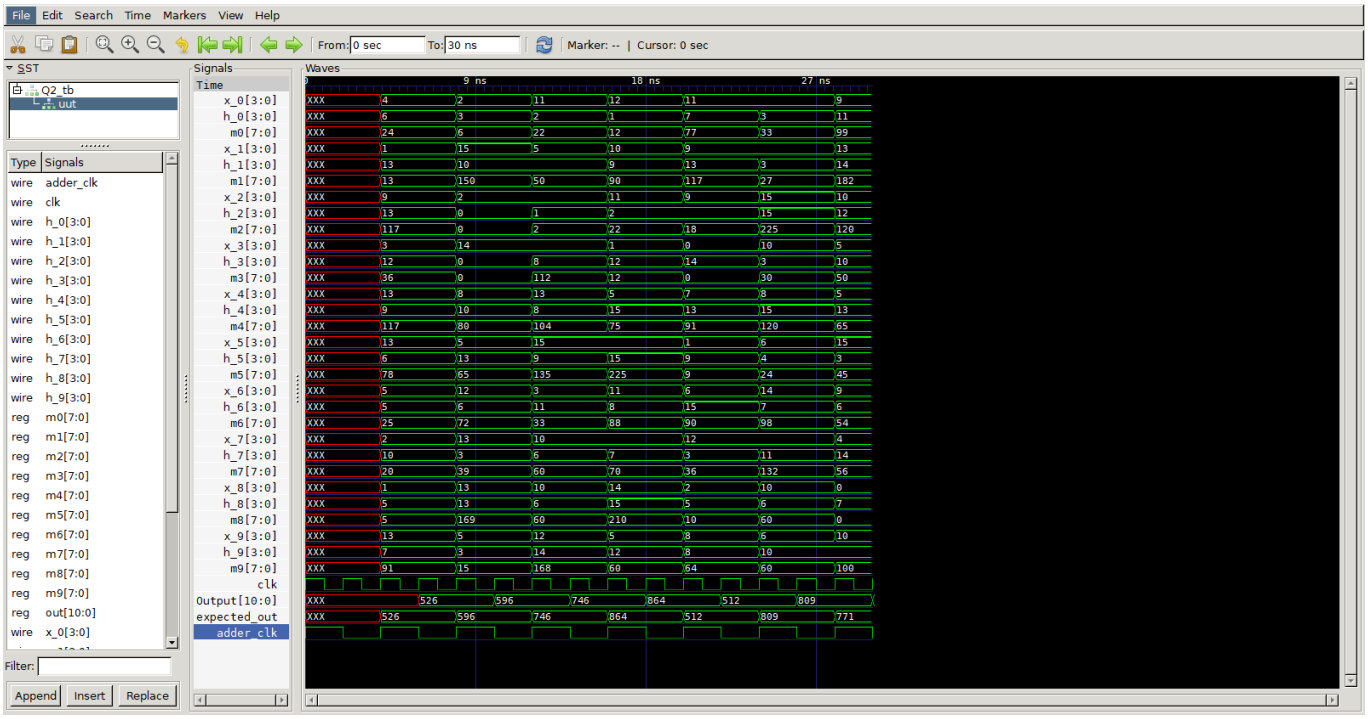
```
always @(negedge adder_clk ) begin
        out<= m0+m1+m2+m3+m4+m5+m6+m7+m8+m9;
end
endmodule
```

Testbench code

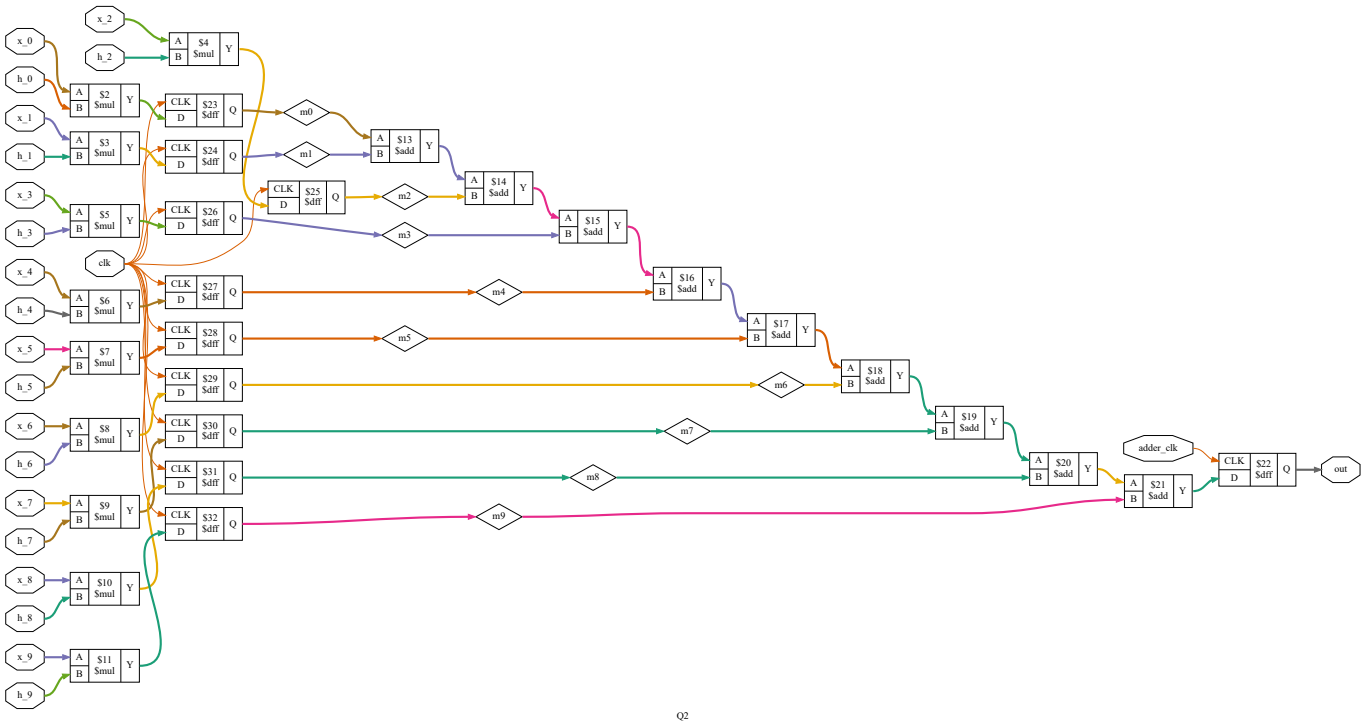
```
`timescale 1ns/1ns
`include "Q2.v"

module Q2_tb;

// Declaring Port parameters for testbench
reg
[3:0]x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,h_0,h_1,h_2,h_3,h_4,h_5,h_6,h_7,h_8,h
_9;
reg clk=1'b1;
reg adder_clk=1'b1;
wire [10:0]Output;

// Clock pulse generation
always #1 clk = ~clk;
always #2 adder_clk = ~adder_clk;

Q2
uut(Output,clk,adder_clk,x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,h_0,h_1,h_2,h_3,h
_4,h_5,h_6,h_7,h_8,h_9);

// Declaring variables for testbench
integer testcases,expected_out;
// Monitoring the input values & Output
initial begin
    $monitor($time,"x_0=%d x_1=%d x_2=%d x_3=%d x_4=%d x_5=%d x_6=%d x_7=%d x_8=%d
x_9=%d h_0=%d h_1=%d h_2=%d h_3=%d h_4=%d h_5=%d h_6=%d h_7=%d h_8=%d h_9=%d
Output=%d
expected=%d",x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9,h_0,h_1,h_2,h_3,h_4,h_5,h_6,h
_7,h_8,h_9,Output,expected_out);

    $dumpfile("Q2.vcd");
    $dumpvars;

   #30 $finish;
end

// Running testcases
initial begin
        for (testcases=0; testcases<10; testcases++) begin
            #4
x_0=$random;x_1=$random;x_2=$random;x_3=$random;x_4=$random;x_5=$random;x_6=$rando
m;x_7=$random;x_8=$random;x_9=$random;h_0=$random;h_1=$random;h_2=$random;h_3=$ran
dom;h_4=$random;h_5=$random;h_6=$random;h_7=$random;h_8=$random;h_9=$random;
```

```
            expected_out =  x_0*h_0 + x_1*h_1 + x_2*h_2 +x_3*h_3 + x_4*h_4 +
x_5*h_5 + x_6*h_6 + x_7*h_7 + x_8*h_8 + x_9*h_9;
        end
end
endmodule
```

Output on GTKWave



Verification through synthesis



Q2

- 10 Multplier modules and 9 adder modules are used in this case i.e same as discussed in class.