Digital IC Design

CHENCHETY ABHISHEK

EE20BTECH11012

Assignment-1

Question-5

- Implement f= x0h0 + x1h1 + x2h2 +x3h3 + x4h4 + x5h5 + x6h6 + x7h7 + x8h8 + x9h9. Compute f assuming there is **no multiplier module and no memory are available**. Consider the values of **h0,..., h9 are known beforehand**. You should not replace multiplier by shifted addition or repetitive additions

Verilog code for shifter

```
module h0_shift (m,x,clk);
  //we assume h0=3
  output reg [7:0] m;
  input [3:0] x;
  input clk;
  wire [6:0] k0, k1;
  wire [7:0] s1;
  assign k0 = x<< 1;
  assign k1 = x;
  assign s1 = k0 + k1;
  always @(posedge clk) begin
    m <= s1;
  end
endmodule

module h1_shift (m,x,clk);
  //we assume h1=10
  output reg [7:0] m;
  input [3:0] x;
  input clk;
  wire [6:0] k0, k1;
  wire [7:0] s, sc;
  assign k0 = x << 3;
  assign k1 = x << 1;
  assign s  = k0 + k1;
  assign sc = s;
  always @(posedge clk) begin
    m <= sc;
  end
endmodule

module h2_shift (m,x,clk);
  //we assume h2=6=4+2
  output reg [7:0] m;
```

```
  input [3:0] x;
  input clk;
  wire [6:0] k0, k1, k2, k3;
  wire [7:0] s, sc;
  assign k0 = x << 2;
  assign k1 = x << 1;
  assign s  = k0 + k1;
  always @(posedge clk) begin
    m <= s;
  end
endmodule
module h3_shift (m,x,clk);
  //we assume h3=11=8+2+1
  output reg [7:0] m;
  input [3:0] x;
  input clk;
  wire [6:0] k0, k1, k2;
  wire [7:0] s, sc;
  assign k0 = x << 3;
  assign k1 = x << 1;
  assign k2 = x;
  assign s  = k0 + k1 + k2;
  assign sc = s;
  always @(posedge clk) begin
    m <= sc;
  end
endmodule

module h4_shift (m,x,clk);
  //we assume h4=2
  output reg [7:0] m;
  input [3:0] x;
  input clk;
  wire [6:0] k0;
  wire [7:0] s, sc;
  assign k0 = x << 1;

  assign s  = k0;
  always @(posedge clk) begin
    m <= s;
  end
endmodule

module h5_shift (m,x,clk);
  //we assume h5=5=4+1
  output reg [7:0] m;
  input [3:0] x;
  input clk;
  wire [6:0] k0, k1;
  wire [7:0] s, sc;
  assign k0 = x << 2;
  assign k1 = x;

  assign s  = k0 + k1;
```

```
  assign sc = s;
  always @(posedge clk) begin
    m <= sc;
  end
endmodule

module h6_shift (m,x,clk);
  //we assume h6=9=8+1
  output reg [7:0] m;
  input [3:0] x;
  input clk;
  wire [6:0] k0, k1;
  wire [7:0] s, sc;
  assign k0 = x << 3;
  assign k1 = x;

  assign s  = k0 + k1;
  always @(posedge clk) begin
    m <= s;
  end
endmodule

module h7_shift (m,x,clk);
  //we assume h7=3=2+1
  output reg [7:0] m;
  input [3:0] x;
  input clk;
  wire [6:0] k0, k1, k2;
  wire [7:0] s, sc;
  assign k0 = x << 1;
  assign k1 = x;

  assign s  = k0 + k1;
  assign sc = s;
  always @(posedge clk) begin
    m <= sc;
  end
endmodule

module h8_shift (m,x,clk);
  //we assume h6=1
  output reg [7:0] m;
  input [3:0] x;
  input clk;
  wire [6:0] k0;
  wire [7:0] s;
  assign k0 = x;
  assign s  = k0;
  always @(posedge clk) begin
    m <= s;
  end
endmodule

module h9_shift (m,x,clk);
```

```
  //we assume h9=14=8+4+2
  output reg [7:0] m;
  input [3:0] x;
  input clk;
  wire [6:0] k0, k1, k2;
  wire [7:0] s, sc;
  assign k0 = x << 3;
  assign k1 = x << 2;
  assign k2 = x << 1;

  assign s  = k0 + k1 + k2;
  assign sc = s;
  always @(posedge clk) begin
    m <= sc;
  end
endmodule

module add_ans (y,m0,m1,m2,m3,m4,m5,m6,m7,m8,m9,clk);
  input [7:0] m0, m1, m2, m3, m4, m5, m6, m7, m8, m9;
  input clk;
  output reg [10:0] y;
  wire [10:0] y_wire;
  assign y_wire = m0 + m1 + m2 + m3 + m4 + m5 + m6 + m7 + m8 + m9;
  always @(posedge clk) begin
    y <= y_wire;
  end
endmodule
```

Verilog Code for Module

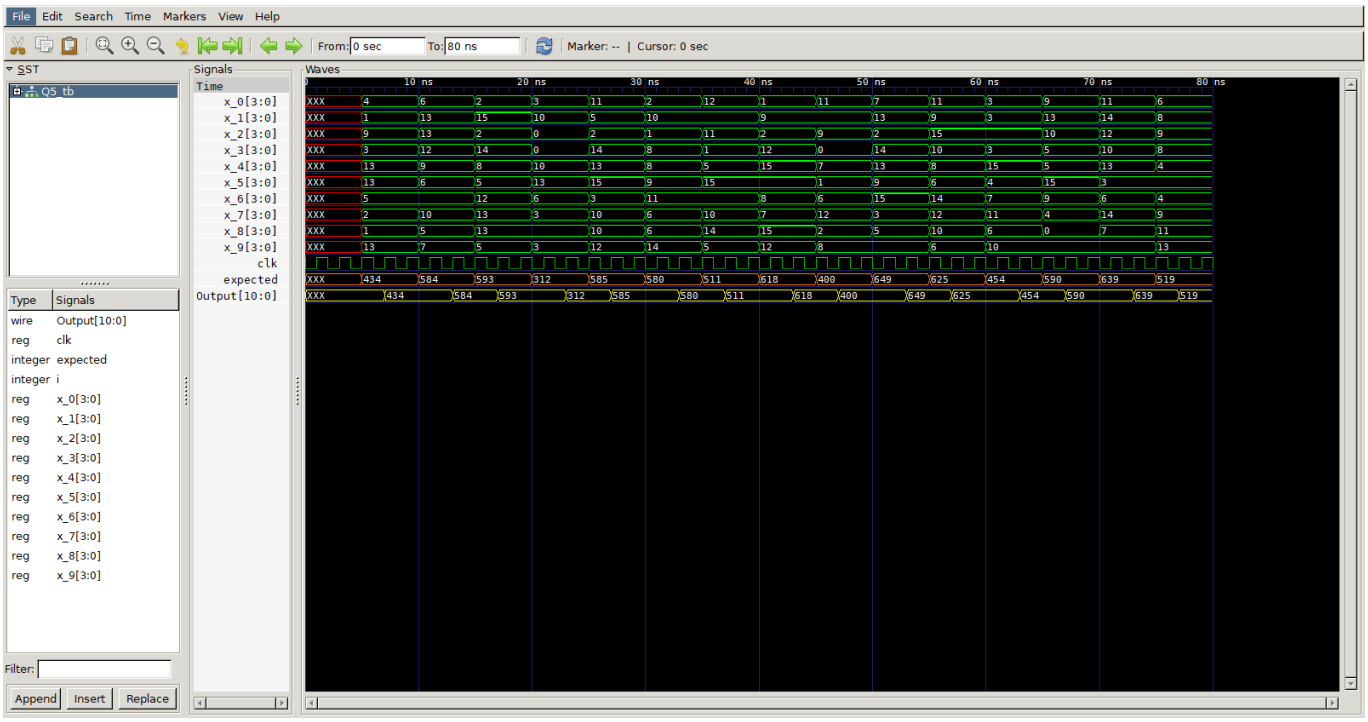
```
`include "shifter.v"
module Q5(out,clk,x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9);
input [3:0] x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9;
input clk;
output [10:0] out;
wire [7:0] m_0,m_1,m_2,m_3,m_4,m_5,m_6,m_7,m_8,m_9;
h0_shift h0(m_0,x_0,clk);
h1_shift h1(m_1,x_1,clk);
h2_shift h2(m_2,x_2,clk);
h3_shift h3(m_3,x_3,clk);
h4_shift h4(m_4,x_4,clk);
h5_shift h5(m_5,x_5,clk);
h6_shift h6(m_6,x_6,clk);
h7_shift h7(m_7,x_7,clk);
h8_shift h8(m_8,x_8,clk);
h9_shift h9(m_9,x_9,clk);
add_ans adder(out,m_0,m_1,m_2,m_3,m_4,m_5,m_6,m_7,m_8,m_9,clk);
endmodule
```

Testbench Code

```
`timescale 1ns/1ns
`include"Q5.v"
module Q5_tb;
reg [3:0] x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9;
reg clk;
wire [10:0] Output;
Q5 uut(Output,clk,x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9);
integer i,j,expected;
initial
begin
$monitor($time,"x0=%d x1=%d x2=%d x3=%d x4=%d x5=%d x6=%d x7=%d x8=%d x9=%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,Output,expected);
end
initial
begin
clk=1'b0;
forever #1 clk=~clk;
end
initial
begin

for(i=0;i<15;i++)
begin
#5 x_0=$random;x_1=$random;x_2=$random;x_3=$random;
x_4=$random;x_5=$random;x_6=$random;x_7=$random;
x_8=$random;x_9=$random;
expected=x_0*3+x_1*(10)+x_2*(6)+x_3*(11)+x_4*(2)+x_5*(5)+x_6*(9)+x_7*(3)+x_8*
(1)+x_9*(14);
end
end
initial
begin
$dumpfile("Q5.vcd");
$dumpvars;
end
initial
begin
#80 $finish;
end
endmodule
```

GTKWave waveforms plot


File Edit Search Time Markers View Help
From: 0 sec
To: 80 ns
Marker: --
Cursor: 0 sec
SST
Q5_tb
Signals
Waves
Time
x_0[3:0]
x_1[3:0]
x_2[3:0]
x_3[3:0]
x_4[3:0]
x_5[3:0]
x_6[3:0]
x_7[3:0]
x_8[3:0]
x_9[3:0]
clk
expected
Output[10:0]
Type Signals
wire Output[10:0]
reg clk
integer expected
integer i
reg x_0[3:0]
reg x_1[3:0]
reg x_2[3:0]
reg x_3[3:0]
reg x_4[3:0]
reg x_5[3:0]
reg x_6[3:0]
reg x_7[3:0]
reg x_8[3:0]
reg x_9[3:0]
Filter:
Append
Insert
Replace