Digital IC Design

CHENCHETY ABHISHEK

EE20BTECH11012

Assignment-1

Question-4

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

Verilog code for Memory and Addition

```
module memory_h0(out,in,clk);
  input [3:0] in;
  input clk;
  output reg [7:0] out;
  integer j;
  parameter h0 = 1;
  reg [7:0] memory[0:((2**4)-1)];
  initial begin
    for (j = 0; j < ((2 ** 4)); j++) begin
      memory[j] = h0 * j;
    end
  end
  always @(posedge clk) begin
    out <= memory[in];
  end
endmodule
module memory_h1(out,in,clk);
  input [3:0] in;
  input clk;
  output reg [7:0] out;
  integer j;
  parameter h1 = 2;
  reg [7:0] memory[0:((2**4)-1)];
  initial begin
    for (j = 0; j < ((2 ** 4)); j++) begin
      memory[j] = h1 * j;
    end
  end
  always @(posedge clk) begin
    out <= memory[in];
  end
endmodule
module memory_h2(out,in,clk);
```

```
  input [3:0] in;
  input clk;
  output reg [7:0] out;
  integer j;
  parameter h2 = 3;
  reg [7:0] memory[0:((2**4)-1)];
  initial begin
    for (j = 0; j < ((2 ** 4)); j++) begin
      memory[j] = h2 * j;
    end
  end
  always @(posedge clk) begin
    out <= memory[in];
  end
endmodule
module memory_h3(out,in,clk);
  input [3:0] in;
  input clk;
  output reg [7:0] out;
  integer j;
  parameter h3 = 4;
  reg [7:0] memory[0:((2**4)-1)];
  initial begin
    for (j = 0; j < ((2 ** 4)); j++) begin
      memory[j] = h3 * j;
    end
  end
  always @(posedge clk) begin
    out <= memory[in];
  end
endmodule
module memory_h4(out,in,clk);
  input [3:0] in;
  input clk;
  output reg [7:0] out;
  integer j;
  parameter h4 = 5;
  reg [7:0] memory[0:((2**4)-1)];
  initial begin
    for (j = 0; j < ((2 ** 4)); j++) begin
      memory[j] = h4 * j;
    end
  end
  always @(posedge clk) begin
    out <= memory[in];
  end
endmodule
module memory_h5(out,in,clk);
  input [3:0] in;
  input clk;
  output reg [7:0] out;
  integer j;
  parameter h5 = 6;
  reg [7:0] memory[0:((2**4)-1)];
```

```
  initial begin
    for (j = 0; j < ((2 ** 4)); j++) begin
      memory[j] = h5 * j;
    end
  end
  always @(posedge clk) begin
    out <= memory[in];
  end
endmodule
module memory_h6(out,in,clk);
  input [3:0] in;
  input clk;
  output reg [7:0] out;
  integer j;
  parameter h6 = 7;
  reg [7:0] memory[0:((2**4)-1)];
  initial begin
    for (j = 0; j < ((2 ** 4)); j++) begin
      memory[j] = h6 * j;
    end
  end
  always @(posedge clk) begin
    out <= memory[in];
  end
endmodule
module memory_h7(out,in,clk);
  input [3:0] in;
  input clk;
  output reg [7:0] out;
  integer j;
  parameter h7 = 8;
  reg [7:0] memory[0:((2**4)-1)];
  initial begin
    for (j = 0; j < ((2 ** 4)); j++) begin
      memory[j] = h7 * j;
    end
  end
  always @(posedge clk) begin
    out <= memory[in];
  end
endmodule
module memory_h8(out,in,clk);
  input [3:0] in;
  input clk;
  output reg [7:0] out;
  integer j;
  parameter h8 = 9;
  reg [7:0] memory[0:((2**4)-1)];
  initial begin
    for (j = 0; j < ((2 ** 4)); j++) begin
      memory[j] = h8 * j;
    end
  end
  always @(posedge clk) begin
```

```
    out <= memory[in];
  end
endmodule
module memory_h9(out,in,clk);
  input [3:0] in;
  input clk;
  output reg [7:0] out;
  integer j;
  parameter h9 = 10;
  reg [7:0] memory[0:((2**4)-1)];
  initial begin
    for (j = 0; j < ((2 ** 4)); j++) begin
      memory[j] = h9 * j;
    end
  end
  always @(posedge clk) begin
    out <= memory[in];
  end
endmodule

module adder(out,m0,m1,m2,m3,m4,m5,m6,m7,m8,m9);
  input [7:0] m0, m1, m2, m3, m4, m5, m6, m7, m8, m9;
  output [10:0] out;
  assign out = m0 + m1 + m2 + m3 + m4 + m5 + m6 + m7 + m8 + m9;
endmodule
```

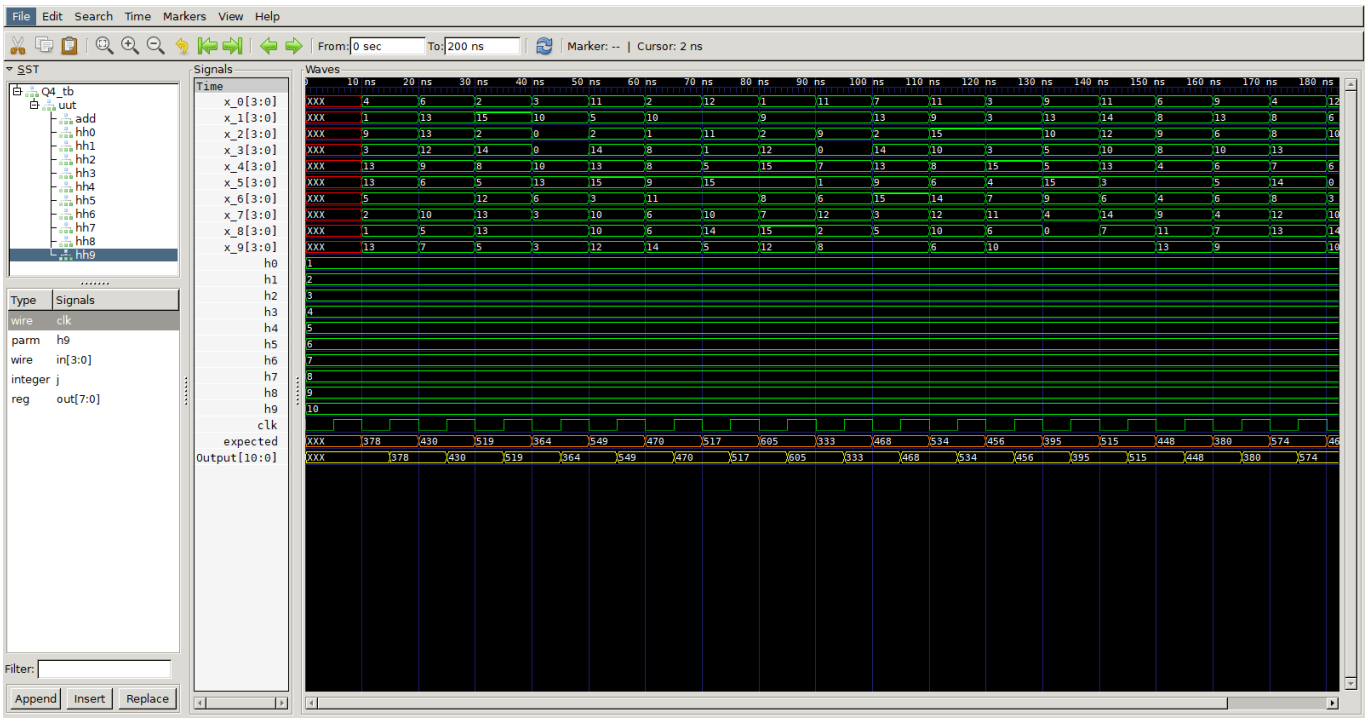
Verilog Code for Module

```
`include"memory.v"
module Q4(out,clk,x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9);
output [10:0]out;
input [3:0] x_0,x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8,x_9;
wire [7:0] m_0,m_1,m_2,m_3,m_4,m_5,m_6,m_7,m_8,m_9;
input clk;

memory_h0 hh0(m_0,x_0,clk);
memory_h1 hh1(m_1,x_1,clk);
memory_h2 hh2(m_2,x_2,clk);
memory_h3 hh3(m_3,x_3,clk);
memory_h4 hh4(m_4,x_4,clk);
memory_h5 hh5(m_5,x_5,clk);
memory_h6 hh6(m_6,x_6,clk);
memory_h7 hh7(m_7,x_7,clk);
memory_h8 hh8(m_8,x_8,clk);
memory_h9 hh9(m_9,x_9,clk);

adder add(out,m_0,m_1,m_2,m_3,m_4,m_5,m_6,m_7,m_8,m_9);
endmodule
```

Testbench Code

```
`timescale 1ns / 1ns
`include "Q4.v"
module Q4_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;
  Q4 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 #5 clk = ~clk;
  end
  initial begin
    for (i = 0; i < 19; i++) begin
      #10 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*(1)+x_1*(2)+x_2*(3)+x_3*(4)+x_4*(5)+x_5*(6)+x_6*(7)+x_7*
(8)+x_8*(9)+x_9*(10);
    end
  end
  initial begin
    $dumpfile("Q4.vcd");
    $dumpvars;
  end
  initial begin
    #200 $finish;
  end
endmodule
```

GTKWave waveforms plot


File Edit Search Time Markers View Help
From: 0 sec
To: 200 ns
Marker: -- | Cursor: 2 ns
SST
Q4_tb
uut
add
hh0
hh1
hh2
hh3
hh4
hh5
hh6
hh7
hh8
hh9
Type Signals
wire clk
parm h9
wire in[3:0]
integer j
reg out[7:0]
Filter:
Append Insert Replace
Signals
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]
h0
h1
h2
h3
h4
h5
h6
h7
h8
h9
clk
expected
Output[10:0]
Waves
10 ns 20 ns 30 ns 40 ns 50 ns 60 ns 70 ns 80 ns 90 ns 100 ns 110 ns 120 ns 130 ns 140 ns 150 ns 160 ns 170 ns 180 ns