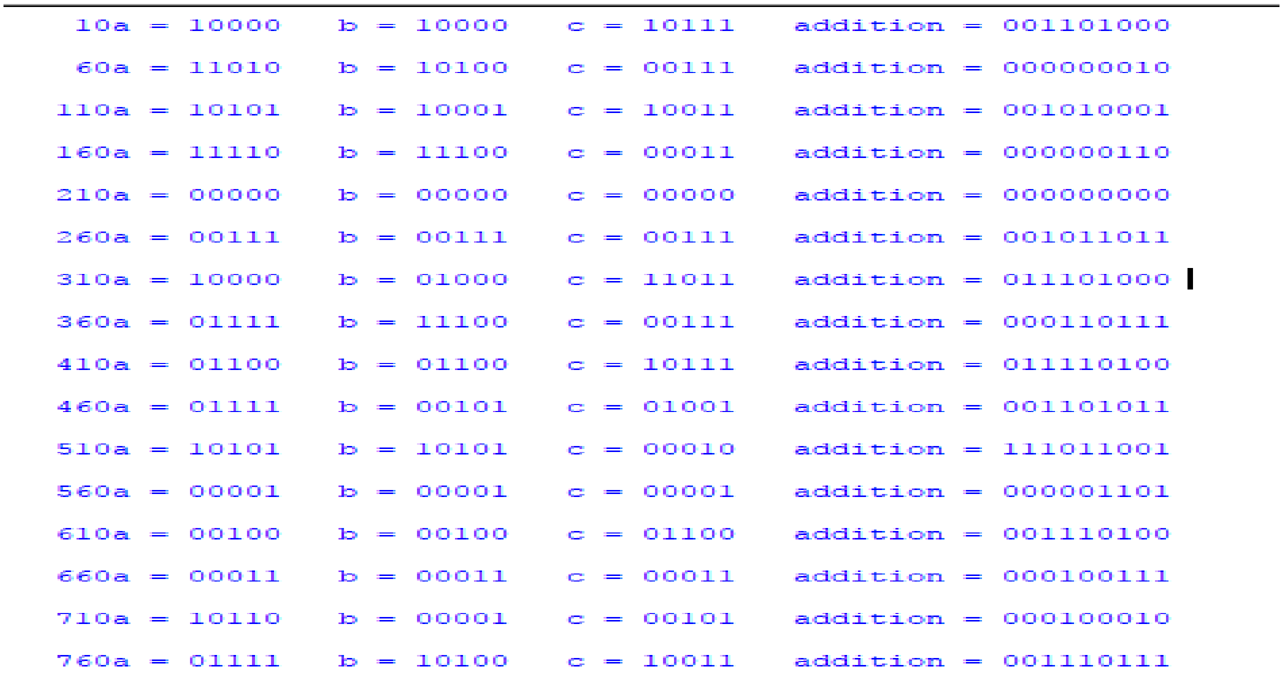
Homework 1

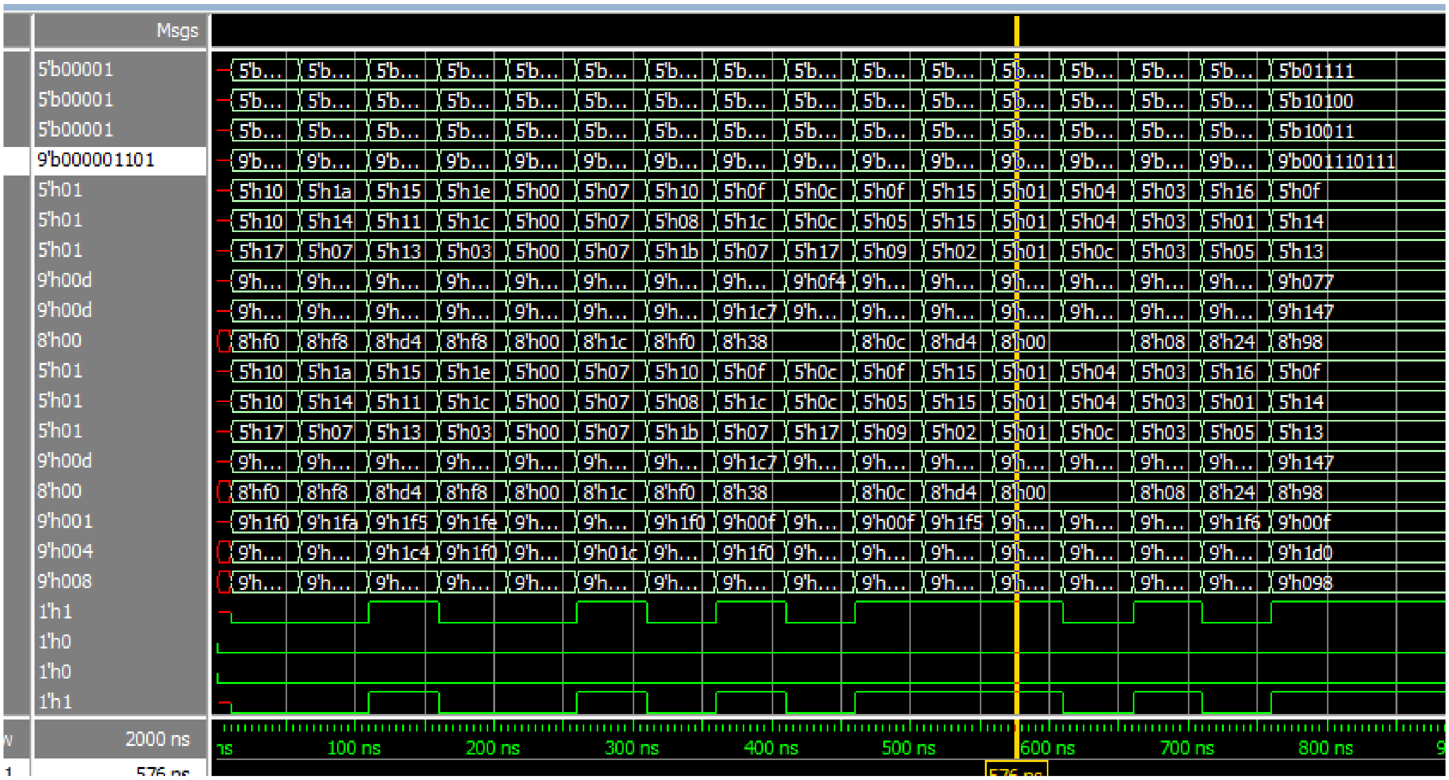
Kshitij Lakhani
EEC 281

Answer 1

OUTPUTS



```
 10a = 10000   b = 10000   c = 10111   addition = 001101000
 60a = 11010   b = 10100   c = 00111   addition = 000000010
110a = 10101   b = 10001   c = 10011   addition = 001010001
160a = 11110   b = 11100   c = 00011   addition = 000000110
210a = 00000   b = 00000   c = 00000   addition = 000000000
260a = 00111   b = 00111   c = 00111   addition = 001011011
310a = 10000   b = 01000   c = 11011   addition = 011101000
360a = 01111   b = 11100   c = 00111   addition = 000110111
410a = 01100   b = 01100   c = 10111   addition = 011110100
460a = 01111   b = 00101   c = 01001   addition = 001101011
510a = 10101   b = 10101   c = 00010   addition = 111011001
560a = 00001   b = 00001   c = 00001   addition = 000001101
610a = 00100   b = 00100   c = 01100   addition = 001110100
660a = 00011   b = 00011   c = 00011   addition = 000100111
710a = 10110   b = 00001   c = 00101   addition = 000100010
760a = 01111   b = 10100   c = 10011   addition = 001110111
```



- All test cases – correct output

Hardware Verilog Code

```verilog
module Top (
input [4:0] a,
input [4:0] b,
input [4:0] c,
output [8:0] addition
);
wire [8:0] sum;
wire [7:0] cout;
carrysave_3is2stage instantiated (a[4:0],b[4:0],c[4:0],sum[8:0],cout[7:0]);
assign addition[8:0] = sum[8:0] + ({cout[7:0],1'b0});
endmodule

module carrysave_3is2stage (
input [4:0] inp1,
input [4:0] inp2,
input [4:0] inp3,
output [8:0] sump,
output [7:0] coutp
);
wire [8:0] inp1_9bit;
wire [8:0] inp2_9bit;
wire [8:0] inp3_9bit;
assign inp1_9bit [8:0] = {inp1[4],inp1[4],inp1[4],inp1[4],inp1[4:0]};
assign inp2_9bit [8:0] = {inp2[4],inp2[4],inp2[4:0],1'b0,1'b0};
assign inp3_9bit [8:0] = {1'b0,inp3[4:0],1'b0,1'b0,1'b0};
FullAdder FA0 (inp1_9bit[0],inp2_9bit[0],inp3_9bit[0],sump[0],coutp[0]);
FullAdder FA1 (inp1_9bit[1],inp2_9bit[1],inp3_9bit[1],sump[1],coutp[1]);
FullAdder FA2 (inp1_9bit[2],inp2_9bit[2],inp3_9bit[2],sump[2],coutp[2]);
FullAdder FA3 (inp1_9bit[3],inp2_9bit[3],inp3_9bit[3],sump[3],coutp[3]);
FullAdder FA4 (inp1_9bit[4],inp2_9bit[4],inp3_9bit[4],sump[4],coutp[4]);
FullAdder FA5 (inp1_9bit[5],inp2_9bit[5],inp3_9bit[5],sump[5],coutp[5]);
FullAdder FA6 (inp1_9bit[6],inp2_9bit[6],inp3_9bit[6],sump[6],coutp[6]);
FullAdder FA7 (inp1_9bit[7],inp2_9bit[7],inp3_9bit[7],sump[7],coutp[7]);
FullAdder FA8 (inp1_9bit[8],inp2_9bit[8],inp3_9bit[8],sump[8], );
endmodule

module FullAdder(
input in1,
input in2,
input cin,
output sum,
output cout
);
assign sum = (in1 ^ in2) ^ cin;
assign cout = (in1 & in2) | (cin & (in1 ^ in2));
endmodule
```

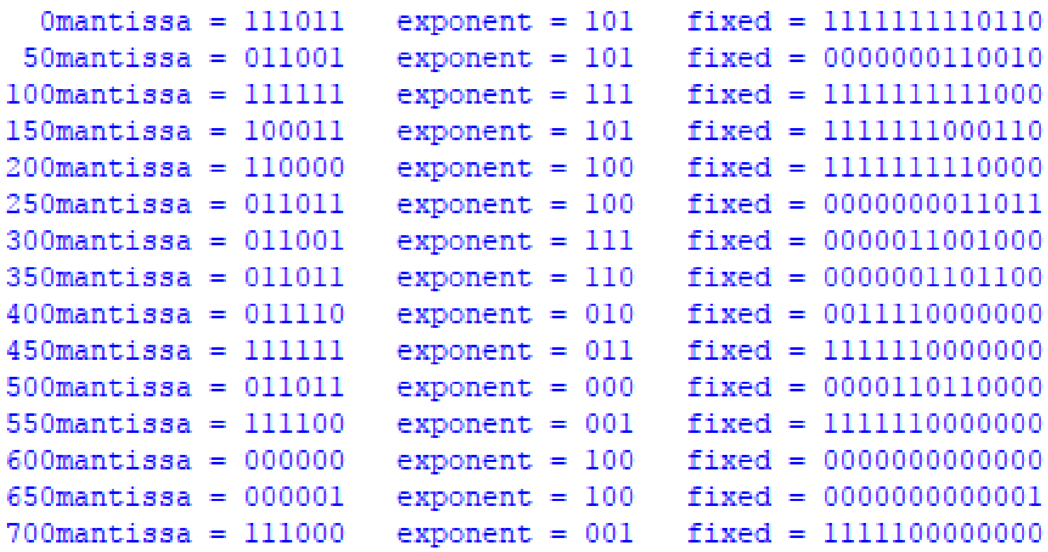
Test bench Verilog Code

```
module Top_tb;
reg [4:0] a;
reg [4:0] b;
reg [4:0] c;
wire [8:0] addition;

Top testbench_adddition (a[4:0],b[4:0],c[4:0],addition[8:0]);

initial
begin
$monitor ($time,"a = %b \t b = %b \t c = %b \t addition = %b \n",a[4:0],b[4:0],c[4:0],addition[8:0]);
#10
a[4:0] = 10000;
b[4:0] = 10000;
c[4:0] = 01111;
#50;
a[4:0] = 10010;
b[4:0] = 10100;
c[4:0] = 11111;
#50;
a[4:0] = 10101;
b[4:0] = 10001;
c[4:0] = 01011;
#50;
a[4:0] = 10110;
b[4:0] = 11100;
c[4:0] = 11011;
#50;
a[4:0] = 00000;
b[4:0] = 00000;
c[4:0] = 00000;
#50;
a[4:0] = 11111;
b[4:0] = 11111;
c[4:0] = 11111;
#50;
a[4:0] = 10000;
b[4:0] = 1000;
c[4:0] = 10011;
#50;
a[4:0] = 00111;
b[4:0] = 11100;
c[4:0] = 11111;
#50;
a[4:0] = 01100;
b[4:0] = 01100;
c[4:0] = 01111;
#50;
a[4:0] = 00111;
b[4:0] = 00101;
c[4:0] = 01001;
```

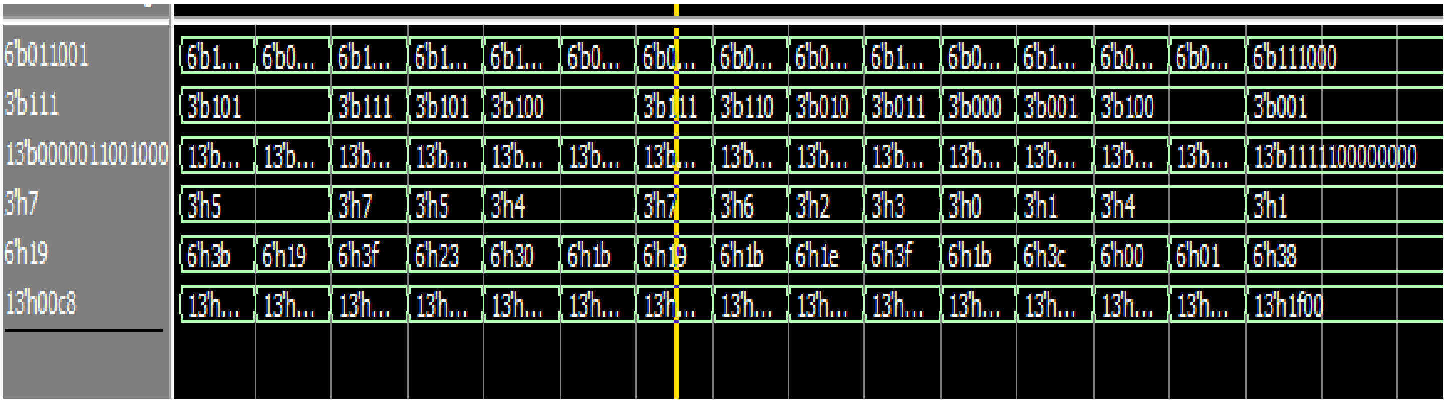
```
#50;
a[4:0] = 10101;
b[4:0] = 10101;
c[4:0] = 11010;
#50;
a[4:0] = 00001;
b[4:0] = 00001;
c[4:0] = 00001;
#50;
a[4:0] = 00100;
b[4:0] = 00100;
c[4:0] = 01100;
#50;
a[4:0] = 11011;
b[4:0] = 11011;
c[4:0] = 11011;
#50;
a[4:0] = 01110;
b[4:0] = 00001;
c[4:0] = 00101;
#50;
a[4:0] = 00111;
b[4:0] = 10100;
c[4:0] = 01011;
#50;
end
endmodule
```

Answer 2

OUPUTS



```
  0mantissa = 111011   exponent = 101   fixed = 1111111110110
 50mantissa = 011001   exponent = 101   fixed = 0000000110010
100mantissa = 111111   exponent = 111   fixed = 1111111111000
150mantissa = 100011   exponent = 101   fixed = 1111111000110
200mantissa = 110000   exponent = 100   fixed = 1111111110000
250mantissa = 011011   exponent = 100   fixed = 0000000011011
300mantissa = 011001   exponent = 111   fixed = 0000011001000
350mantissa = 011011   exponent = 110   fixed = 0000001101100
400mantissa = 011110   exponent = 010   fixed = 0011110000000
450mantissa = 111111   exponent = 011   fixed = 1111110000000
500mantissa = 011011   exponent = 000   fixed = 0000110110000
550mantissa = 111100   exponent = 001   fixed = 1111110000000
600mantissa = 000000   exponent = 100   fixed = 0000000000000
650mantissa = 000001   exponent = 100   fixed = 0000000000001
700mantissa = 111000   exponent = 001   fixed = 1111100000000
```


6'b011001 6'b1... 6'b0... 6'b1... 6'b1... 6'b1... 6'b0... 6'b0... 6'b0... 6'b0... 6'b1... 6'b0... 6'b1... 6'b0... 6'b0... 6'b111000
3'b111 3'b101 3'b111 3'b101 3'b100 3'b111 3'b110 3'b010 3'b011 3'b000 3'b001 3'b100 3'b001
13'b0000011001000 13'b... 13'b... 13'b... 13'b... 13'b... 13'b... 13'b... 13'b... 13'b... 13'b... 13'b... 13'b... 13'b... 13'b... 13'b1111100000000
3'h7 3'h5 3'h7 3'h5 3'h4 3'h7 3'h6 3'h2 3'h3 3'h0 3'h1 3'h4 3'h1
6'h19 6'h3b 6'h19 6'h3f 6'h23 6'h30 6'h1b 6'h19 6'h1b 6'h1e 6'h3f 6'h1b 6'h3c 6'h00 6'h01 6'h38
13'h00c8 13'h... 13'h... 13'h... 13'h... 13'h... 13'h... 13'h... 13'h... 13'h... 13'h... 13'h... 13'h... 13'h... 13'h... 13'h1f00

- All test cases correct output

Hardware Verilog Code

```
module float2fixed(
mantissa,
exponent,
fixed);

input [2:0] exponent;
input [5:0] mantissa;
output [12:0] fixed;

wire [5:0]  mantissa;
wire [2:0]  exponent;
reg  [12:0] fixed;


always @(*) begin
    case({exponent})

    3'b000: begin
    fixed = {mantissa[5],mantissa[5],mantissa[5],mantissa,1'b0,1'b0,1'b0,1'b0};
    end

    3'b001: begin
    fixed = {mantissa[5],mantissa[5],mantissa,1'b0,1'b0,1'b0,1'b0,1'b0};
    end

    3'b010: begin
    fixed = {mantissa[5],mantissa,1'b0,1'b0,1'b0,1'b0,1'b0,1'b0};
    end

    3'b011: begin
    fixed = {mantissa,1'b0,1'b0,1'b0,1'b0,1'b0,1'b0,1'b0};
    end

    3'b100: begin
    fixed = {mantissa[5],mantissa[5],mantissa[5],mantissa[5],mantissa[5],mantissa[5],mantissa[5], mantissa};
    end
    3'b101: begin
    fixed = {mantissa[5],mantissa[5],mantissa[5],mantissa[5],mantissa[5],mantissa[5],mantissa,1'b0};
    end

    3'b110: begin
    fixed = {mantissa[5],mantissa[5],mantissa[5],mantissa[5],mantissa[5],mantissa,1'b0,1'b0};
    end

    3'b111: begin
    fixed = {mantissa[5],mantissa[5],mantissa[5],mantissa[5],mantissa,1'b0,1'b0,1'b0};
    end
    endcase
end
endmodule
```

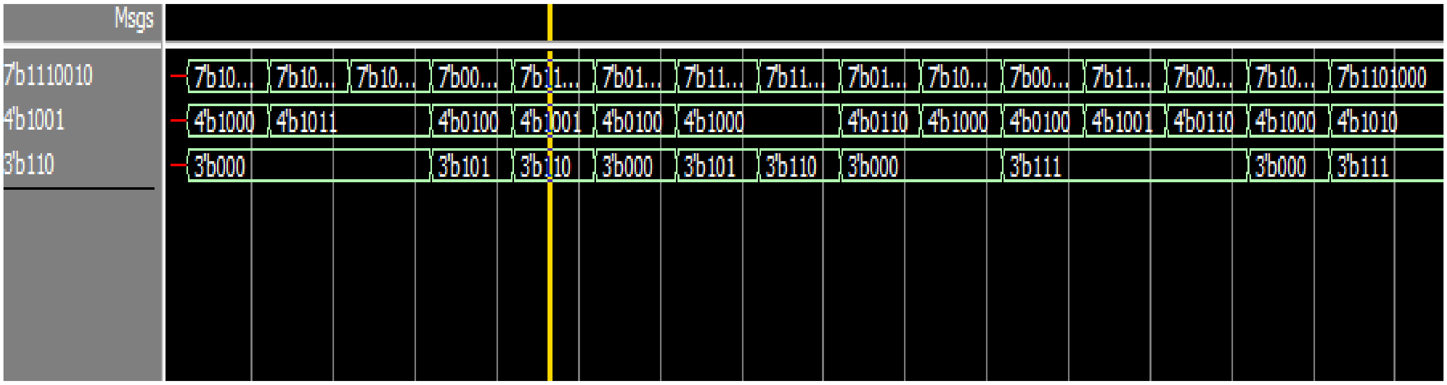
Test bench Verilog Code

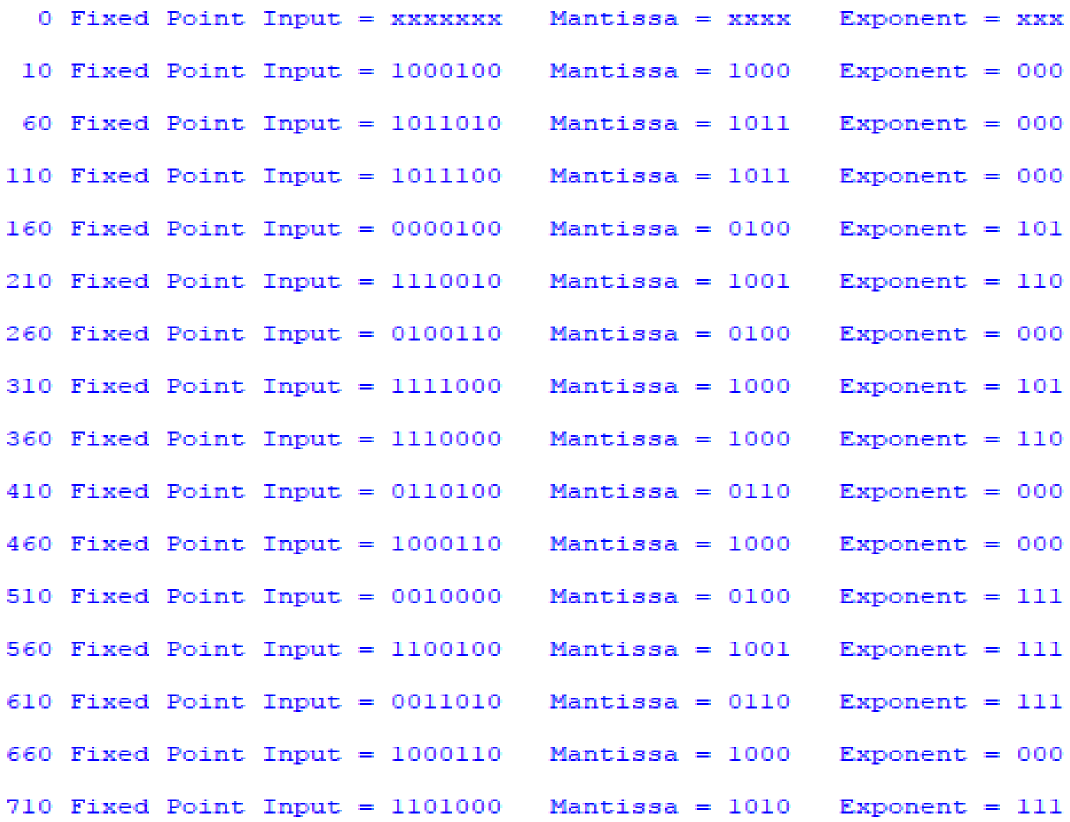
```
module float2fixed_tb;
reg   [5:0] mantissa;
reg   [2:0] exponent;
wire  [12:0] fixed;

float2fixed FF(mantissa[5:0],exponent[2:0], fixed[12:0]);

initial
begin

$monitor($time, "mantissa = %b \t exponent = %b \t fixed = %b \t",mantissa[5:0],exponent[2:0],fixed[12:0]);

mantissa[5:0] = 6'b111011;
exponent[2:0] = 3'b101;
#50;

mantissa[5:0]    = 6'b011001;
exponent[2:0]     = 3'b101;
#50;

mantissa[5:0] = 6'b111111;
exponent[2:0] = 3'b111;
#50;

mantissa[5:0] = 6'b100011;
exponent[2:0] = 3'b101;
#50;

mantissa[5:0] = 6'b110000;
exponent[2:0] = 3'b100;
#50;

mantissa[5:0] = 6'b011011;
exponent[2:0] = 3'b100;
#50;
```

```
mantissa[5:0] = 6'b011001;
exponent[2:0] = 3'b111;
#50;

mantissa[5:0] = 6'b011011;
exponent[2:0] = 3'b110;
#50;

mantissa[5:0] = 6'b011110;
exponent[2:0] = 3'b010;
#50;


mantissa[5:0] = 6'b111111;
exponent[2:0] = 3'b011;
#50;

mantissa[5:0] = 6'b011011;
exponent[2:0] = 3'b000;
#50;

mantissa[5:0] = 6'b111100;
exponent[2:0] = 3'b001;
#50;

mantissa[5:0] = 6'b000000;
exponent[2:0] = 3'b100;
#50;

mantissa[5:0] = 6'b000001;
exponent[2:0] = 3'b100;
#50;
mantissa[5:0] = 6'b111000;
exponent[2:0] = 3'b001;
end
endmodule
```

Answer 3

OUTPUTS


Msgs
7'b1110010
4'b1001
3'b110
7'b10... 7'b10... 7'b10... 7'b00... 7'b11... 7'b01... 7'b11... 7'b11... 7'b01... 7'b10... 7'b00... 7'b11... 7'b00... 7'b10... 7'b1101000
4'b1000 4'b1011 4'b0100 4'b1001 4'b0100 4'b1000 4'b0110 4'b1000 4'b0100 4'b1001 4'b0110 4'b1000 4'b1010
3'b000 3'b101 3'b110 3'b000 3'b101 3'b110 3'b000 3'b111 3'b000 3'b111



```
  0 Fixed Point Input = xxxxxxx   Mantissa = xxxx   Exponent = xxx
 10 Fixed Point Input = 1000100   Mantissa = 1000   Exponent = 000
 60 Fixed Point Input = 1011010   Mantissa = 1011   Exponent = 000
110 Fixed Point Input = 1011100   Mantissa = 1011   Exponent = 000
160 Fixed Point Input = 0000100   Mantissa = 0100   Exponent = 101
210 Fixed Point Input = 1110010   Mantissa = 1001   Exponent = 110
260 Fixed Point Input = 0100110   Mantissa = 0100   Exponent = 000
310 Fixed Point Input = 1111000   Mantissa = 1000   Exponent = 101
360 Fixed Point Input = 1110000   Mantissa = 1000   Exponent = 110
410 Fixed Point Input = 0110100   Mantissa = 0110   Exponent = 000
460 Fixed Point Input = 1000110   Mantissa = 1000   Exponent = 000
510 Fixed Point Input = 0010000   Mantissa = 0100   Exponent = 111
560 Fixed Point Input = 1100100   Mantissa = 1001   Exponent = 111
610 Fixed Point Input = 0011010   Mantissa = 0110   Exponent = 111
660 Fixed Point Input = 1000110   Mantissa = 1000   Exponent = 000
710 Fixed Point Input = 1101000   Mantissa = 1010   Exponent = 111
```

- All test cases correct output

Hardware + Test bench Verilog Code

```
module fixedtofloating;
reg [6:0] fixed;
reg [3:0] mantissa;
reg [2:0] exponent;
always @ (*) begin
if (fixed[5] != fixed[6]) begin
mantissa[3:0] = {fixed[6],fixed[5],fixed[4],fixed[3]};
exponent[2:0] = 3'b000;
end else if (fixed[4] != fixed[6]) begin
mantissa[3:0] = {fixed[5],fixed[4],fixed[3],fixed[2]};
exponent[2:0] = 3'b111;
end else if (fixed[3] != fixed[6]) begin
mantissa[3:0] = {fixed[4],fixed[3],fixed[2],fixed[1]};
exponent[2:0] = 3'b110;
end else if (fixed[2] != fixed[6]) begin
mantissa[3:0] = {fixed[3],fixed[2],fixed[1],fixed[0]};
exponent[2:0] = 3'b101;
end else begin
mantissa[3:0] = {fixed[2],fixed[1],fixed[0],1'b0};
exponent[2:0] = 3'b100;
end
end
```

```
initial
begin
$monitor($time," Fixed Point Input = %b \t Mantissa = %b \t Exponent = %b \n",fixed[6:0],mantissa[3:0],exponent[2:0]);
#10;
fixed[6:0] = 1100100;
#50;
fixed[6:0] = 1010010;
#50;
fixed[6:0] = 0011100;
#50;
fixed[6:0] = 0100100;
#50;
fixed[6:0] = 0001010;
#50;
fixed[6:0] = 1011110;
#50;
fixed[6:0] = 0011000;
#50;
fixed[6:0] = 1110000;
#50;
fixed[6:0] = 1010100;
#50;
fixed[6:0] = 1111110;
#50;
fixed[6:0] = 0010000;
#50;
fixed[6:0] = 0000100;
#50;
fixed[6:0] = 0010010;
#50;
fixed[6:0] = 1111110;
#50;
fixed[6:0] = 0001000;
#50;
end
endmodule
```

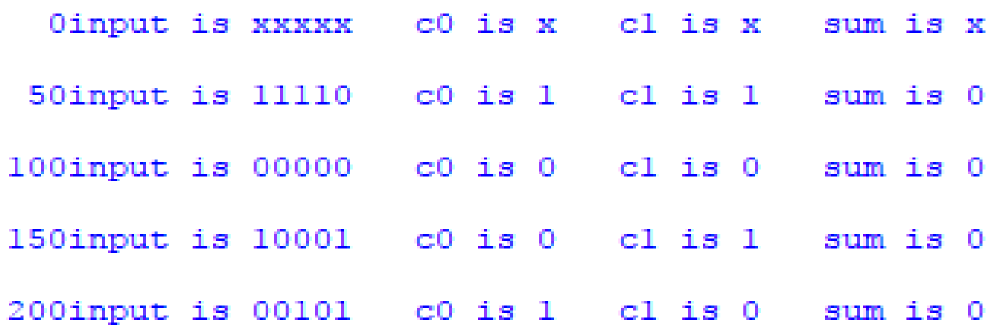
Answer 4

```
   inputs  | outputs
         c | c c
 a b c d i | o 1 s
-----------+-------
 0 0 0 0 0 | 0 0 0
 0 0 0 0 1 | 0 0 1
 0 0 0 1 0 | 0 0 1
 0 0 0 1 1 | 0 1 0
 0 0 1 0 0 | 0 0 1
 0 0 1 0 1 | 0 1 0
 0 0 1 1 0 | 0 0 1
 0 0 1 1 1 | 0 1 1
 0 1 0 0 0 | 0 0 1
 0 1 0 0 1 | 0 1 0
 0 1 0 1 0 | 0 1 0
```

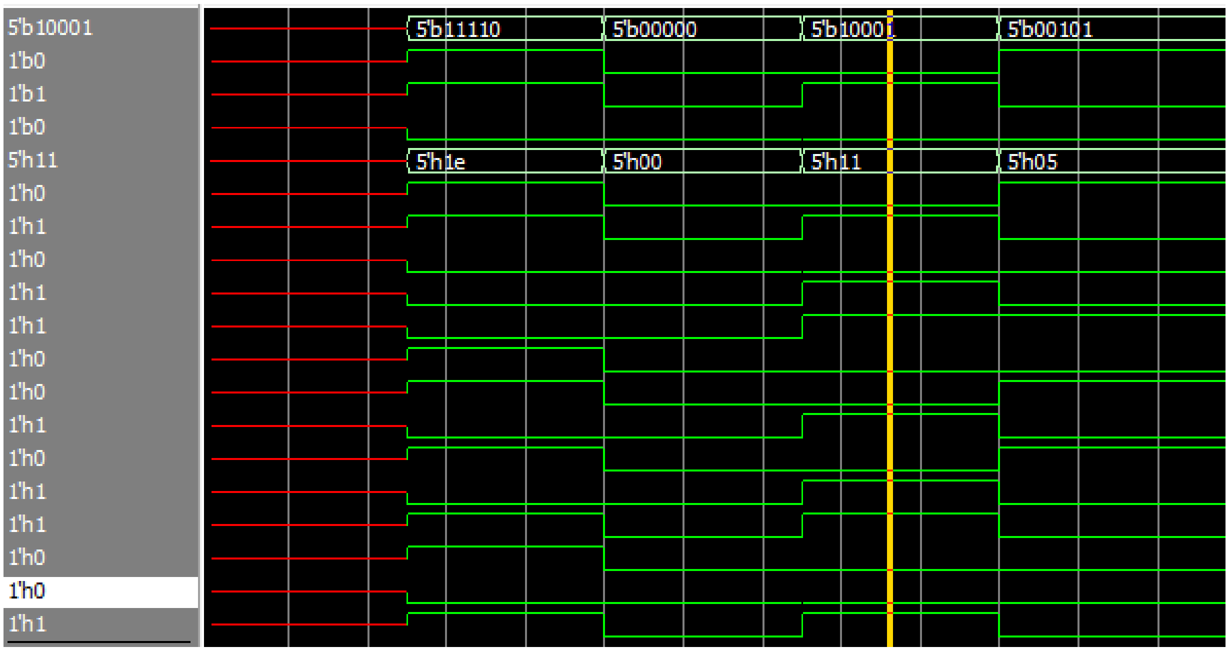
```
0 1 0 1 1 | 0 1 1
0 1 1 0 0 | 1 0 0
0 1 1 0 1 | 1 0 1
0 1 1 1 0 | 1 0 1
0 1 1 1 1 | 1 1 0
1 0 0 0 0 | 0 0 1
1 0 0 0 1 | 0 1 0
1 0 0 1 0 | 0 1 0
1 0 0 1 1 | 0 1 1
1 0 1 0 0 | 1 0 0
1 0 1 0 1 | 1 0 1
1 0 1 1 0 | 1 0 1
1 0 1 1 1 | 1 1 0
1 1 0 0 0 | 1 0 0
1 1 0 0 1 | 1 0 1
1 1 0 1 0 | 1 0 1
1 1 0 1 1 | 1 1 0
1 1 1 0 0 | 1 0 1
1 1 1 0 1 | 1 1 0
1 1 1 1 0 | 1 1 0
1 1 1 1 1 | 1 1 1
```

Answer 5

OUTPUTS



```
  0input is xxxxx   c0 is x   c1 is x   sum is x

 50input is 11110   c0 is 1   c1 is 1   sum is 0

100input is 00000   c0 is 0   c1 is 0   sum is 0

150input is 10001   c0 is 0   c1 is 1   sum is 0

200input is 00101   c0 is 1   c1 is 0   sum is 0
```



- All test cases correct output

Hardware Verilog Code

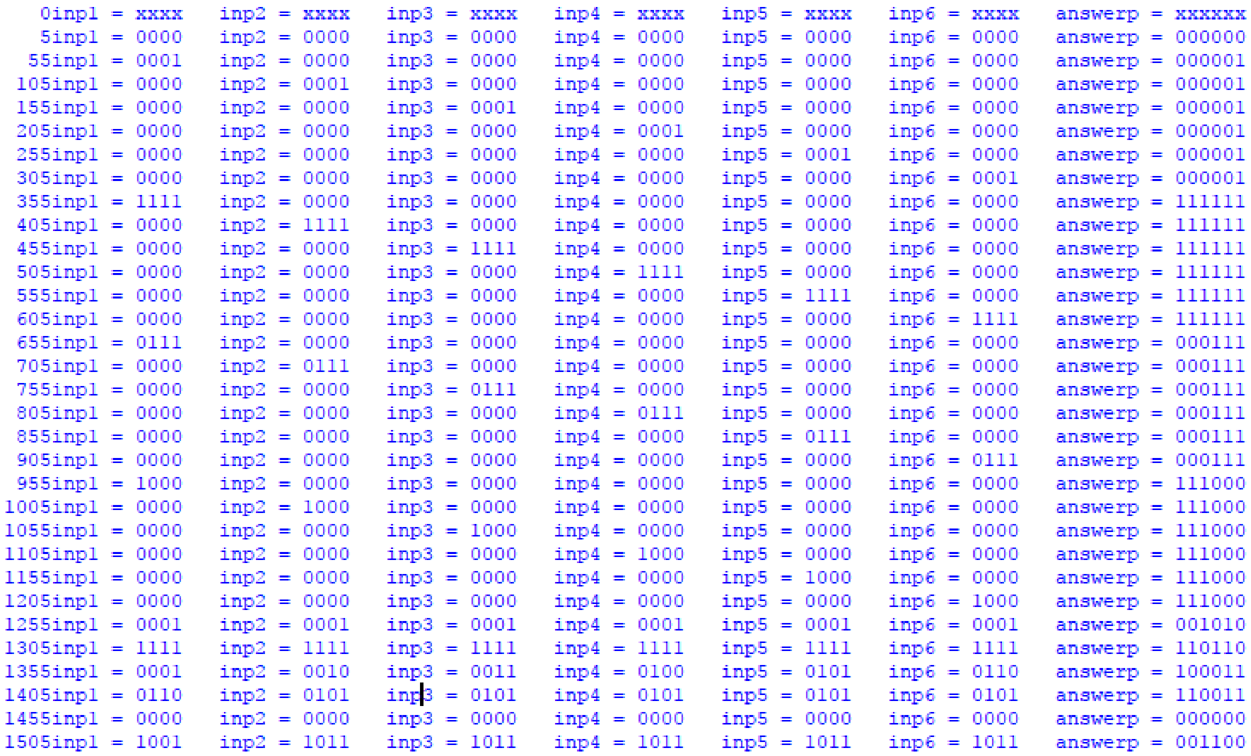
```
module carrysave_4is2 (
input [4:0] in,
output c0,
output c1,
output sum
);
wire connection;
FullAdder FA0 (in[0],in[1],in[2],connection,c0);
FullAdder FA1 (connection,in[4],in[3],sum,c1);
endmodule

module FullAdder(
input in1,
input in2,
input cin,
output sum,
output cout
);
assign sum = (in1 ^ in2) ^ cin;
assign cout = (in1 & in2) | (cin & (in1 ^ in2));
endmodule
```

Test bench Verilog Code

```
module carrysave_4is2_tb;
reg [4:0] in;
wire c0,c1,sum;
carrysave_4is2 carrysave_tb(in[4:0],c0,c1,sum);
initial
begin
$monitor($time,"input is %b \t c0 is %b \t c1 is %b \t sum is %b \n",in[4:0],c0,c1,sum);
#50;
in[4:0] = 10110;
#50
in[4:0] = 00000;
#50;
in[4:0] = 10001;
#50;
in[4:0] = 00101;
end
endmodule
```

Answer 6

OUTPUTS



```
   0inp1 = xxxx   inp2 = xxxx   inp3 = xxxx   inp4 = xxxx   inp5 = xxxx   inp6 = xxxx   answerp = xxxxxx
   5inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 000000
  55inp1 = 0001   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 000001
 105inp1 = 0000   inp2 = 0001   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 000001
 155inp1 = 0000   inp2 = 0000   inp3 = 0001   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 000001
 205inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0001   inp5 = 0000   inp6 = 0000   answerp = 000001
 255inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0001   inp6 = 0000   answerp = 000001
 305inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0001   answerp = 000001
 355inp1 = 1111   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 111111
 405inp1 = 0000   inp2 = 1111   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 111111
 455inp1 = 0000   inp2 = 0000   inp3 = 1111   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 111111
 505inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 1111   inp5 = 0000   inp6 = 0000   answerp = 111111
 555inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 1111   inp6 = 0000   answerp = 111111
 605inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 1111   answerp = 111111
 655inp1 = 0111   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 000111
 705inp1 = 0000   inp2 = 0111   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 000111
 755inp1 = 0000   inp2 = 0000   inp3 = 0111   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 000111
 805inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0111   inp5 = 0000   inp6 = 0000   answerp = 000111
 855inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0111   inp6 = 0000   answerp = 000111
 905inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0111   answerp = 000111
 955inp1 = 1000   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 111000
1005inp1 = 0000   inp2 = 1000   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 111000
1055inp1 = 0000   inp2 = 0000   inp3 = 1000   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 111000
1105inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 1000   inp5 = 0000   inp6 = 0000   answerp = 111000
1155inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 1000   inp6 = 0000   answerp = 111000
1205inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 1000   answerp = 111000
1255inp1 = 0001   inp2 = 0001   inp3 = 0001   inp4 = 0001   inp5 = 0001   inp6 = 0001   answerp = 001010
1305inp1 = 1111   inp2 = 1111   inp3 = 1111   inp4 = 1111   inp5 = 1111   inp6 = 1111   answerp = 110110
1355inp1 = 0001   inp2 = 0010   inp3 = 0011   inp4 = 0100   inp5 = 0101   inp6 = 0110   answerp = 100011
1405inp1 = 0110   inp2 = 0101   inp3 = 0101   inp4 = 0101   inp5 = 0101   inp6 = 0101   answerp = 110011
1455inp1 = 0000   inp2 = 0000   inp3 = 0000   inp4 = 0000   inp5 = 0000   inp6 = 0000   answerp = 000000
1505inp1 = 1001   inp2 = 1011   inp3 = 1011   inp4 = 1011   inp5 = 1011   inp6 = 1011   answerp = 001100
```

- All test cases, except last 5 – correct/expected output

Hardware Verilog Code

```
module stage1_3is2 (
input [5:0] in1,
input [5:0] in2,
input [5:0] in3,
output [5:0] cout_stage1,
output [5:0] sum_stage1
);
FullAdder FA2 (in1[0],in2[0],in3[0],sum_stage1[0],cout_stage1[0]);
FullAdder FA3 (in1[1],in2[1],in3[1],sum_stage1[1],cout_stage1[1]);
FullAdder FA4 (in1[2],in2[2],in3[2],sum_stage1[2],cout_stage1[2]);
FullAdder FA5 (in1[3],in2[3],in3[3],sum_stage1[3],cout_stage1[3]);
FullAdder FA12 (in1[4],in2[4],in3[4],sum_stage1[4],cout_stage1[4]);
FullAdder FA13 (in1[5],in2[5],in3[5],sum_stage1[5],cout_stage1[5]);
endmodule

module stage2_3is2 (
input [5:0] in4,
input [5:0] in5,
input [5:0] in6,
output [5:0] cout_stage2,
output [5:0] sum_stage2
);
FullAdder FA6 (in4[0],in5[0],in6[0],sum_stage2[0],cout_stage2[0]);
FullAdder FA7 (in4[1],in5[1],in6[1],sum_stage2[1],cout_stage2[1]);
FullAdder FA8 (in4[2],in5[2],in6[2],sum_stage2[2],cout_stage2[2]);
FullAdder FA9 (in4[3],in5[3],in6[3],sum_stage2[3],cout_stage2[3]);
FullAdder FA22 (in4[4],in5[4],in6[4],sum_stage2[4],cout_stage2[4]);
FullAdder FA23 (in4[5],in5[5],in6[5],sum_stage2[5],cout_stage2[5]);
```

```
endmodule

module stage3_4is2 (
input [5:0] subin1,
input [5:0] subin2,
input [5:0] subin3,
input [5:0] subin4,
output [5:0] sum_stage3,
output [5:0] cout_stage3
//output carry
);
wire connection1,connection2,connection3,connection4,connection5;
carrysave_4is2 CS1(({1'b0,1'b0,subin3[0],1'b0,subin1[0]}),connection1,cout_stage3[0],sum_stage3[0]);
carrysave_4is2 CS2(({connection1,subin4[0],subin3[1],subin2[0],subin1[1]}),connection2,cout_stage3[1],sum_stage3[1]);
carrysave_4is2 CS3(({connection2,subin4[1],subin3[2],subin2[1],subin1[2]}),connection3,cout_stage3[2],sum_stage3[2]);
carrysave_4is2 CS4(({connection3,subin4[2],subin3[3],subin2[2],subin1[3]}),connection4,cout_stage3[3],sum_stage3[3]);
carrysave_4is2 CS5(({connection4,subin4[3],subin3[4],subin2[3],subin1[4]}),connection5,cout_stage3[4],sum_stage3[4]);
carrysave_4is2 CS6(({connection5,subin4[4],subin3[5],subin2[4],subin1[5]}), , ,sum_stage3[5]);
endmodule

module Top (
input [3:0] i1,
input [3:0] i2,
input [3:0] i3,
input [3:0] i4,
input [3:0] i5,
input [3:0] i6,
output [5:0] answer
);
wire [5:0] cout_S1;
wire [5:0] sum_S1;
wire [5:0] cout_S2;
wire [5:0] sum_S2;
wire [5:0] sum_S3;
wire [5:0] cout_S3;
//wire carry_S3;
stage1_3is2 S1 ( {i1[3],i1[3],i1[3:0]} , {i2[3],i2[3],i2[3:0]} , {i3[3],i3[3],i3[3:0]} , cout_S1[5:0] , sum_S1[5:0]);
stage2_3is2 S2 ( {i4[3],i4[3],i4[3:0]} , {i5[3],i5[3],i5[3:0]} , {i6[3],i6[3],i6[3:0]} , cout_S2[5:0] , sum_S2[5:0]);
stage3_4is2 S3 (sum_S1[5:0],({cout_S1[4:0],1'b0}),sum_S2[5:0],({cout_S2[4:0],1'b0}),sum_S3[5:0],cout_S3[5:0]/*,carry_S3*/);
assign answer [5:0] = (sum_S3[5:0]) + ({cout_S3[4:0],1'b0});
endmodule

module carrysave_4is2 (
input [4:0] in,
output c0,
output c1,
output sum
);
wire connection;
FullAdder FA0 (in[0],in[1],in[2],connection,c0);
FullAdder FA1 (connection,in[4],in[3],sum,c1);
endmodule

module FullAdder(
input in1,
input in2,
input cin,
output sum,
output cout
);
assign sum = (in1 ^ in2) ^ cin;
assign cout = (in1 & in2) | (cin & (in1 ^ in2));
endmodule
```

Test bench Verilog Code

```
module Top_tb;
reg [3:0] inp1;
reg [3:0] inp2;
reg [3:0] inp3;
reg [3:0] inp4;
reg [3:0] inp5;
reg [3:0] inp6;
wire [5:0] answerp;
Top Toptestbench (inp1 [3:0],inp2 [3:0],inp3 [3:0],inp4 [3:0],inp5 [3:0],inp6 [3:0],answerp [5:0]);

initial
begin
$monitor ($time, "inp1 = %b \t inp2 = %b \t inp3 = %b \t inp4 = %b \t inp5 = %b \t inp6 = %b \t answerp = %b
\t",inp1[3:0],inp2[3:0],inp3[3:0],inp4[3:0],inp5[3:0],inp6[3:0],answerp[5:0]);
#5;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0001;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0001;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0001;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
```

```
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0001;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0001;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0001;
#50;
inp1[3:0] = 4'b1111;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b1111;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b1111;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b1111;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
```

```
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b1111;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b1111;
#50;
inp1[3:0] = 4'b0111; //7
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0111;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0111;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0111;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0111;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0111;
#50;
inp1[3:0] = 4'b1000;
```

```
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b1000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b1000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b1000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b1000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b1000;
#50;
inp1[3:0] = 4'b0001;
inp2[3:0] = 4'b0001;
inp3[3:0] = 4'b0001;
inp4[3:0] = 4'b0001;
inp5[3:0] = 4'b0001;
inp6[3:0] = 4'b0001;
#50;
inp1[3:0] = 4'b1111;
inp2[3:0] = 4'b1111;
inp3[3:0] = 4'b1111;
inp4[3:0] = 4'b1111;
inp5[3:0] = 4'b1111;
inp6[3:0] = 4'b1111;
```

```
#50;
inp1[3:0] = 4'b0001;
inp2[3:0] = 4'b0010;
inp3[3:0] = 4'b0011;
inp4[3:0] = 4'b0100;
inp5[3:0] = 4'b0101;
inp6[3:0] = 4'b0110;
#50;
inp1[3:0] = 4'b0110;
inp2[3:0] = 4'b0101;
inp3[3:0] = 4'b0101;
inp4[3:0] = 4'b0101;
inp5[3:0] = 4'b0101;
inp6[3:0] = 4'b0101;
#50;
inp1[3:0] = 4'b0000;
inp2[3:0] = 4'b0000;
inp3[3:0] = 4'b0000;
inp4[3:0] = 4'b0000;
inp5[3:0] = 4'b0000;
inp6[3:0] = 4'b0000;
#50;
inp1[3:0] = 4'b1001; //-7
inp2[3:0] = 4'b1011; //-5
inp3[3:0] = 4'b1011; //-5
inp4[3:0] = 4'b1011; //-5
inp5[3:0] = 4'b1011; //-5
inp6[3:0] = 4'b1011; //-5
#100;
end
endmodule
```

Answer 7

```
module carrysave_4is2 (
input [4:0] in,
output c0,
output c1,
output sum
);
wire connection;
FullAdder FA0 (in[0],in[1],in[2],connection,c0);
FullAdder FA1 (connection,in[4],in[3],sum,c1);
endmodule



module Top1 (
input [10:0] inp,
output cout,
output sum
);
wire [3:0] sum_word;
wire [3:0] carry_word;
wire cs1_c0;
wire cs1_c1;
wire cs1_s;
wire cs2_c0;
wire cs2_c1;
wire cs2_s;
stage1 instantiated(inp[10:0],sum_word[3:0],carry_word[3:0]);

carrysave_4is2 cs1 ({sum_word[3],carry_word[3],sum_word[2],carry_word[2]},cs1_c0,cs1_c1,cs1_s);
carrysave_4is2 cs2 ({sum_word[1],carry_word[1],sum_word[0],carry_word[0]},cs2_c0,cs2_c1,cs2_s);
carrysave_4is2 cs3 ({cs1_c1,cs1_s,cs2_c1,cs2_s},)

//assign addition[4:0] = {1'b0,sum_word[3:0]} + {carry_word[3:0],1'b0};
endmodule

module stage1 (
input [10:0] in,
output [3:0] sum,
output [3:0] cout
);

FullAdder FA0 (in[0],in[1],in[2],sum[0],cout[0]);
FullAdder FA1 (in[3],in[4],in[5],sum[1],cout[1]);
FullAdder FA2 (in[6],in[7],in[8],sum[2],cout[2]);




HalfAdder HA (in[9],in[10],sum[3],cout[3]);
endmodule

module HalfAdder(
input inp1,
input inp2,
output sum,
output cout
);
assign sum = inp1 ^ inp2;
assign cout = inp1 & inp2;
endmodule

module FullAdder(
input in1,
input in2,
input cin,
output sum,
output cout
);
assign sum = (in1 ^ in2) ^ cin;
assign cout = (in1 & in2) | (cin & (in1 ^ in2));
endmodule
```

- Works for few test cases only erratically