

Assignment 1

(under supervision of Eng : Kareem Waseem)

Q1] figure (1) as shown represent the Verilog code Snippet for this part . In addition to that to check The code performance we simulate this code and Produce it's schematic as shown in figure 2.

```
module question1(A,B,C,D,E,F,sel,out,out_bar);  
  
input A,B,C,D,E,F,sel;  
output out,out_bar;  
wire w1,w2;  
  
assign w1=A&B&C;  
assign w2=~(D^E^F);  
  
assign out=(sel==0)?w1:w2;  
assign out_bar=~out;  
  
endmodule
```

Figure (1) : Verilog code

Figure (2) : schematic

Q2] in this part we have three graphs fig (3)&(4) &(5)
which are Represent the code and waveform
and schematic respectively.

```
module question2(a,b,c);
input [3:0] a,b;
output [3:0] c;
assign c= a+b;
endmodule
```

Figure (3) : verilog code

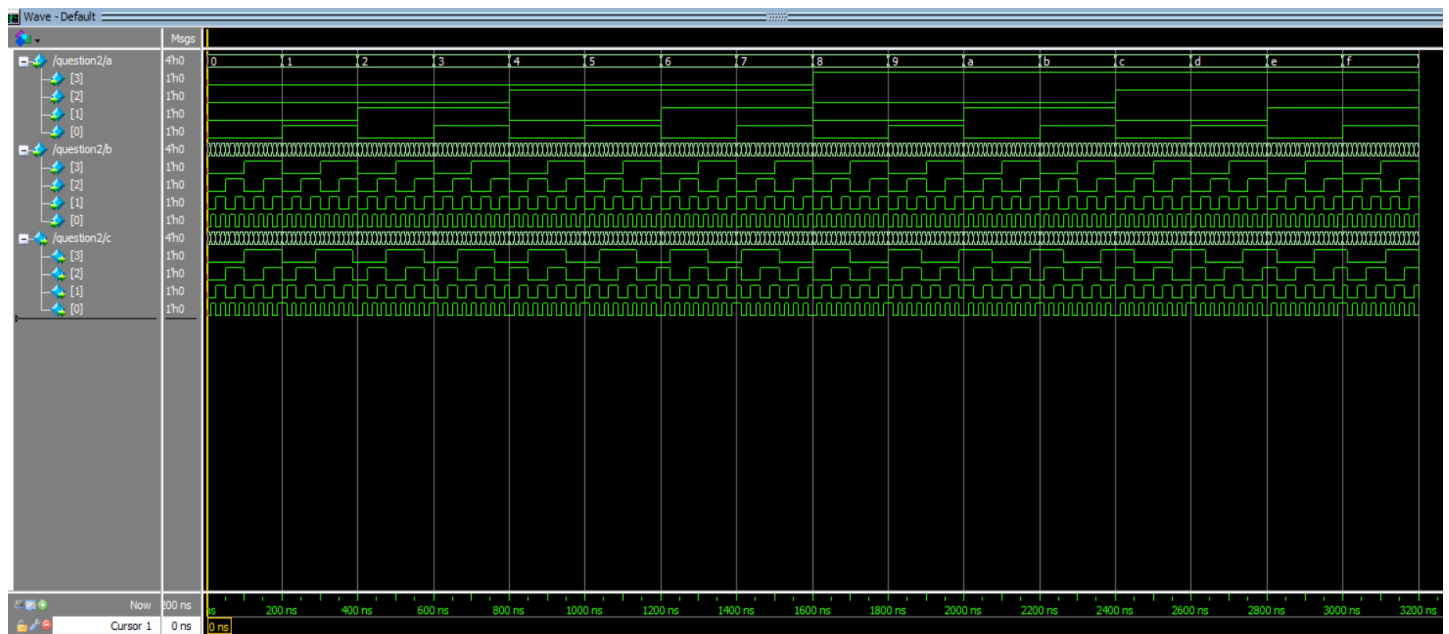


Figure (4) : waveform

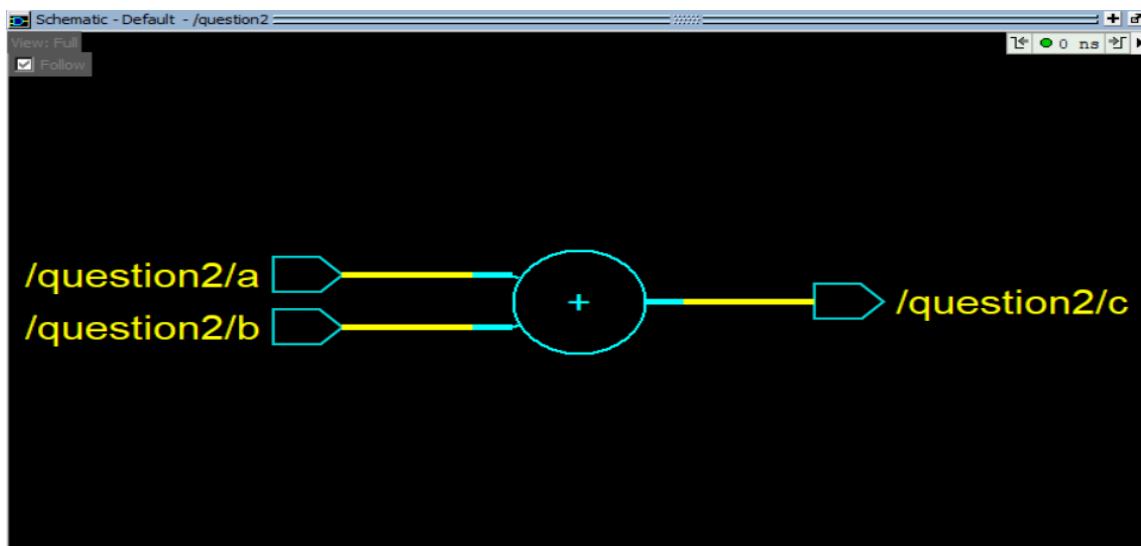


Figure (5) : schematic

Q3] this circuit operates as full adder which take three inputs each of 1-bit and produce the sum and output_carry. Here is the implementation of the code and to check the validity of the code we create the waveform and the schematic for this circuit.

```
module question3(A,B,Cin,S,Cout);
input A,B,Cin;
output S,Cout;
wire w1,w2,w3;
assign w1= A^B;
assign S=w1^Cin;
assign w2=A&B;
assign Cout=w2|w3;
endmodule
```

Figure (6) : Verilog code

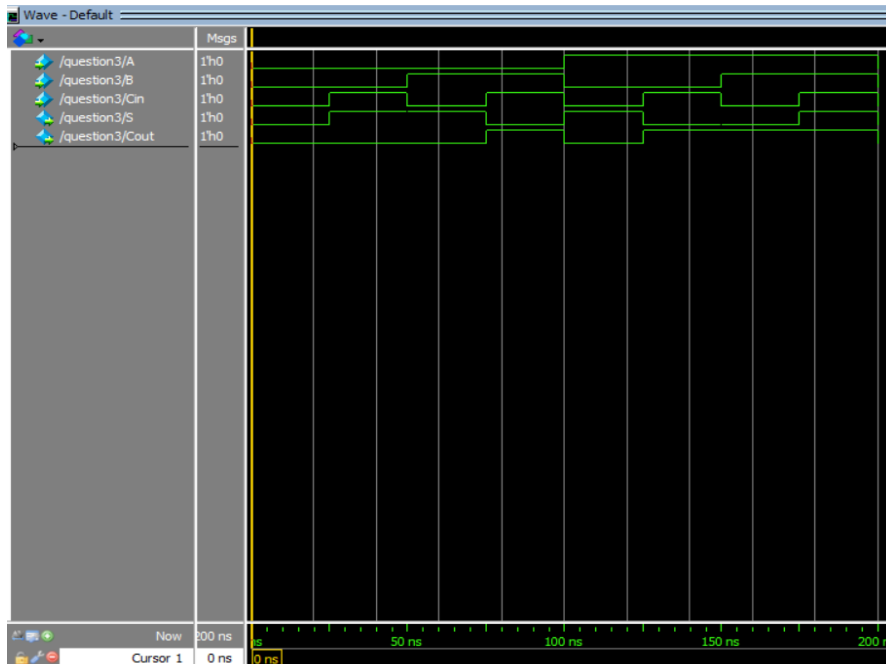


Figure (7) : waveform

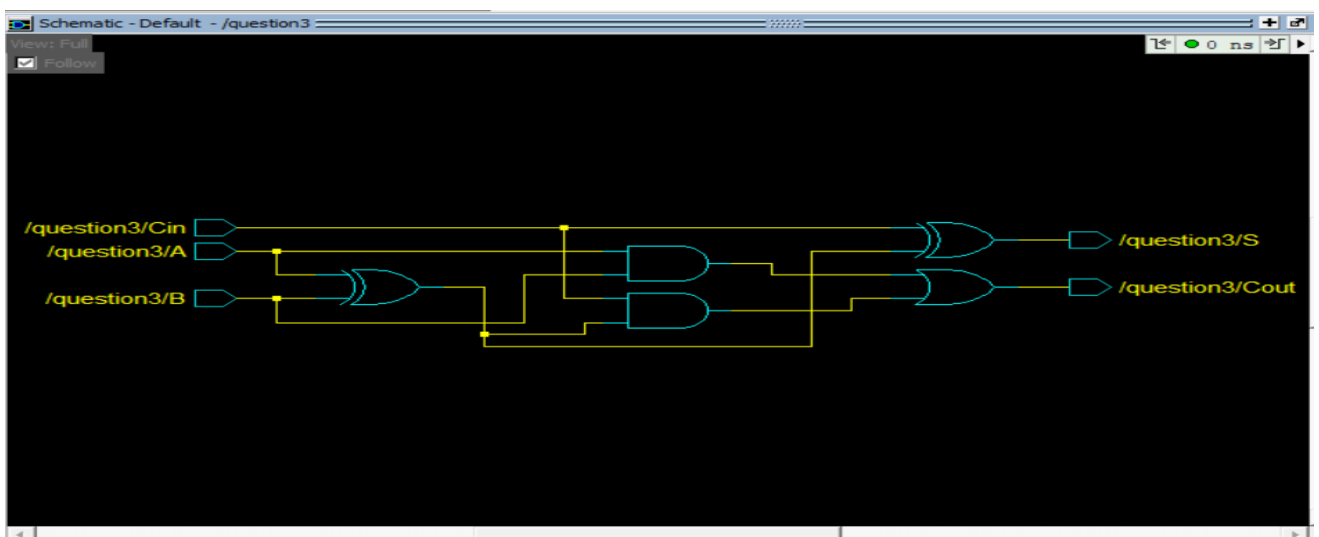


Figure (8) : schematic

Q4] this circuit represent 2*4 Decoder and here are its code with waveform and schematic.

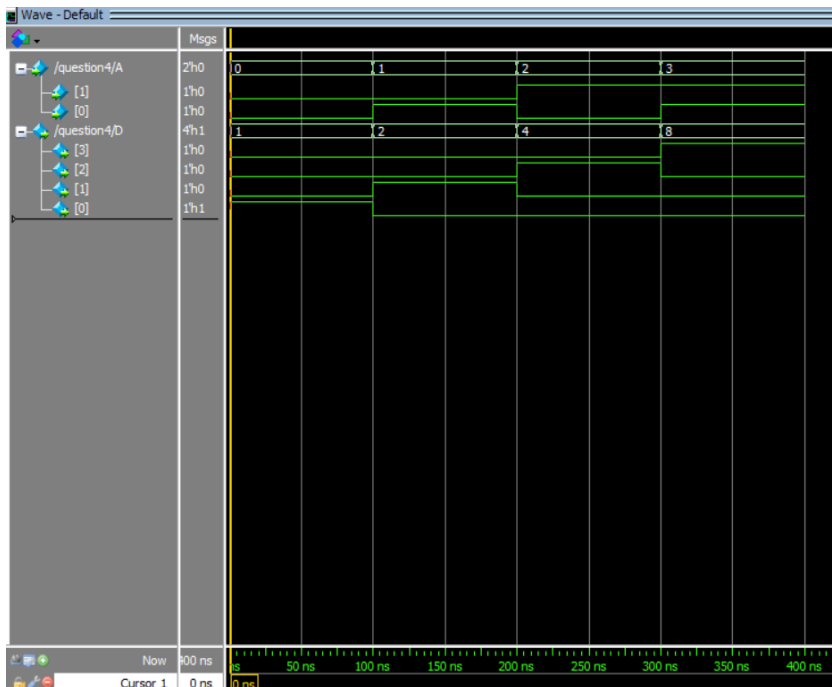


Figure (10) : waveform

```

module question4(A,D);
input [1:0] A ;
output [3:0] D ;

assign D[0]=((A[1]==0)&&(A[0]==0))?1:0 ;
assign D[1]=((A[1]==0)&&(A[0]==1))?1:0 ;
assign D[2]=((A[1]==1)&&(A[0]==0))?1:0 ;
assign D[3]=((A[1]==1)&&(A[0]==1))?1:0 ;

endmodule

```

Figure (9) : Verilog code

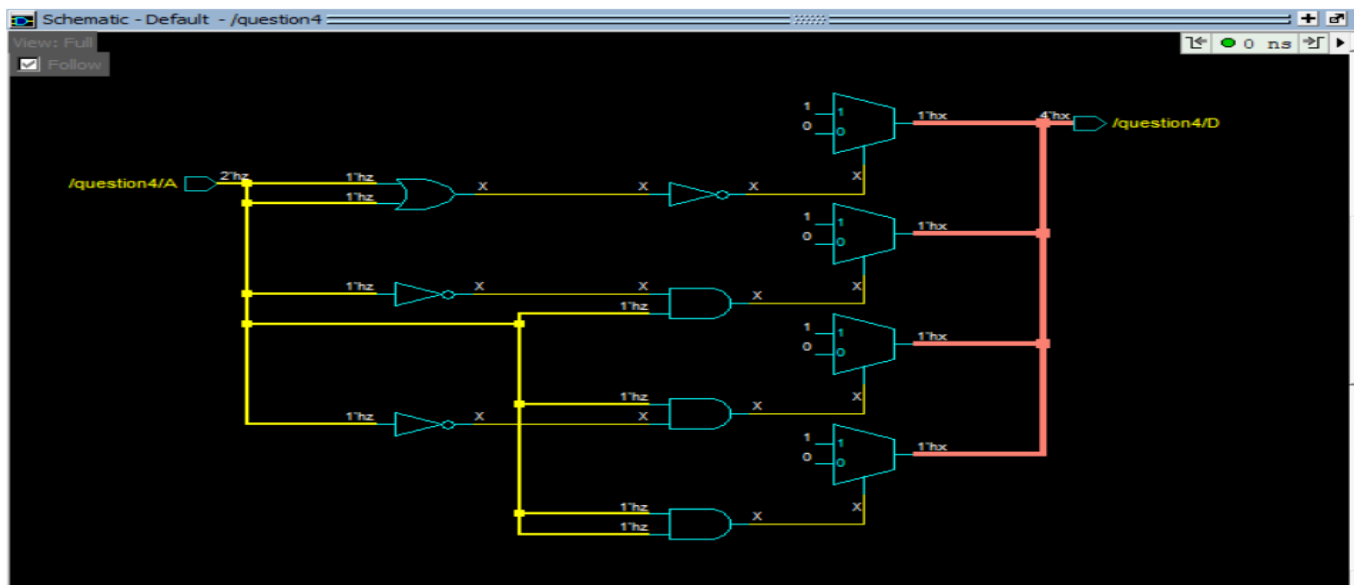




Figure (11) : schematic

Q5] This circuit takes only one input (8 bits) and produces an output = {input, parity_bit}

Where the even parity means that the total number of ones must be even, so there are two cases

First case  input has odd ones therefore in this case the parity_bit = 1 to make the overall output signal has even number of ones.

Second case  input has even ones therefore in this case the parity_bit = 0 to make the overall output signal has even number of ones.

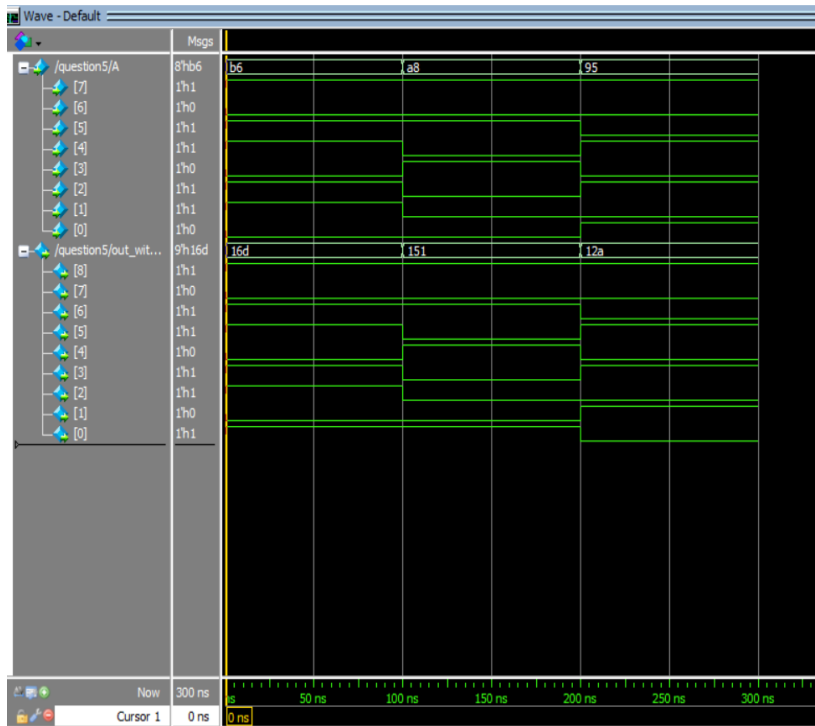


Figure (13) : waveform

```

module question5(A,out_with_parity);

    input [7:0] A;
    output [8:0] out_with_parity;

    wire parity_bit;

    assign parity_bit=^A;    //xor reduction operator

    assign out_with_parity={A,parity_bit};

endmodule

```

Figure (12) : Verilog code

Clock cycle	Input A	Out_with_parity	Parity_bit
First cycle	10110110	101101101	1
Second cycle	10101000	101010001	1
Third cycle	10010101	100101010	0

This table is obtained from the simulated waveform above in figure (13)

-Where the parity bit is the least significant bit (LSB) in the output (Out_with_parity).

Q6] this circuit called by comparator which compare 4-bit inputs (A,B) and tell the relation between them so, we use the relational operators to implement this circuit.

```
module question6(A,B,A_greaterthan_B,A_equals_B,A_lessthan_B);
input [3:0] A,B;
output A_greaterthan_B,A_equals_B,A_lessthan_B;
wire [3:0] w1;
assign w1=~(A^B);
assign A_equals_B =(&w1==1)?1:0 ;    // there is another method to check equality which is assign A_equals_B =(A==B)?1:0 ;
assign A_greaterthan_B = (A>B)?1:0 ;
assign A_lessthan_B = (A<B)?1:0 ;
endmodule
```

Figure (14) : Verilog code

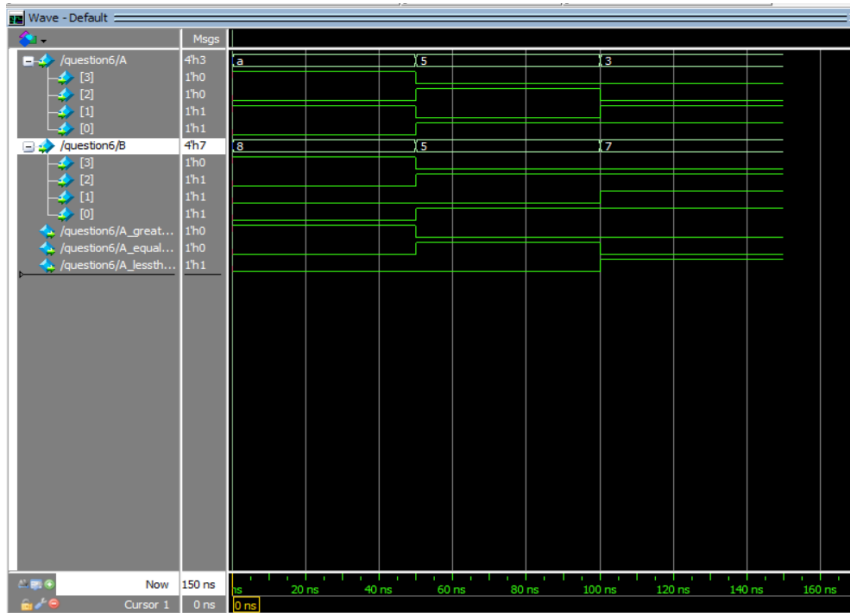


Figure (15) : waveform

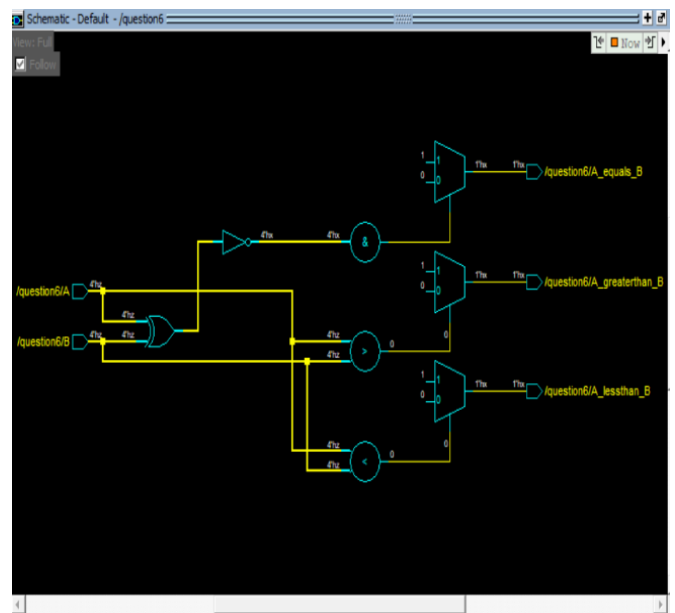


Figure (16) : schematic

Clock cycle	Input A	Input B	A_equals_B	A_greaterthan_B	A_lessthan_B
First cycle	1010	1000	0	1	0
Second cycle	0101	0101	1	0	0
Third cycle	0011	0111	0	0	1

This table is obtained from the above simulated waveform