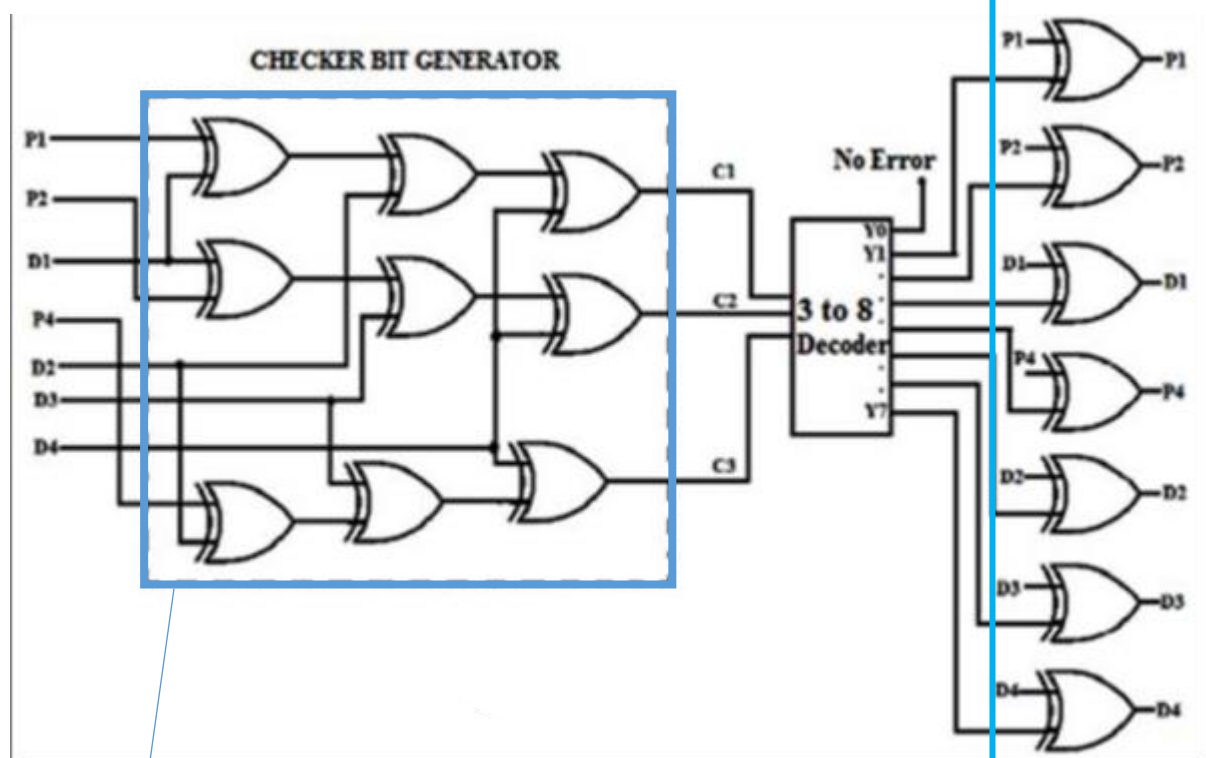


1)→ 7 BIT HAMMING CODE (ERROR AND DETECTION CIRCUIT):-



ERROR DETECTION
CIRCUIT

Correction circuit

→CODES:-

```
`timescale 1ns / 1ps
```

```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```



```
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
```

```
module hamming(O,A);  
input [6:0] A;  
output [7:0] O;  
wire[2:0] W;  
wire [7:0]D;  
xor(W[0],A[0],A[2],A[4],A[6]);  
xor(W[1],A[1],A[2],A[5],A[6]);  
xor(W[2],A[3],A[4],A[5],A[6]);  
decoder dd(D,W);  
buf(O[0],D[0]);  
xor(O[1],D[1],A[0]);  
xor(O[2],D[2],A[1]);  
xor(O[3],D[3],A[2]);  
xor(O[4],D[4],A[3]);  
xor(O[5],D[5],A[4]);  
xor(O[6],D[6],A[5]);  
xor(O[7],D[7],A[6]);  
endmodule
```

```
module decoder(O,I);
```

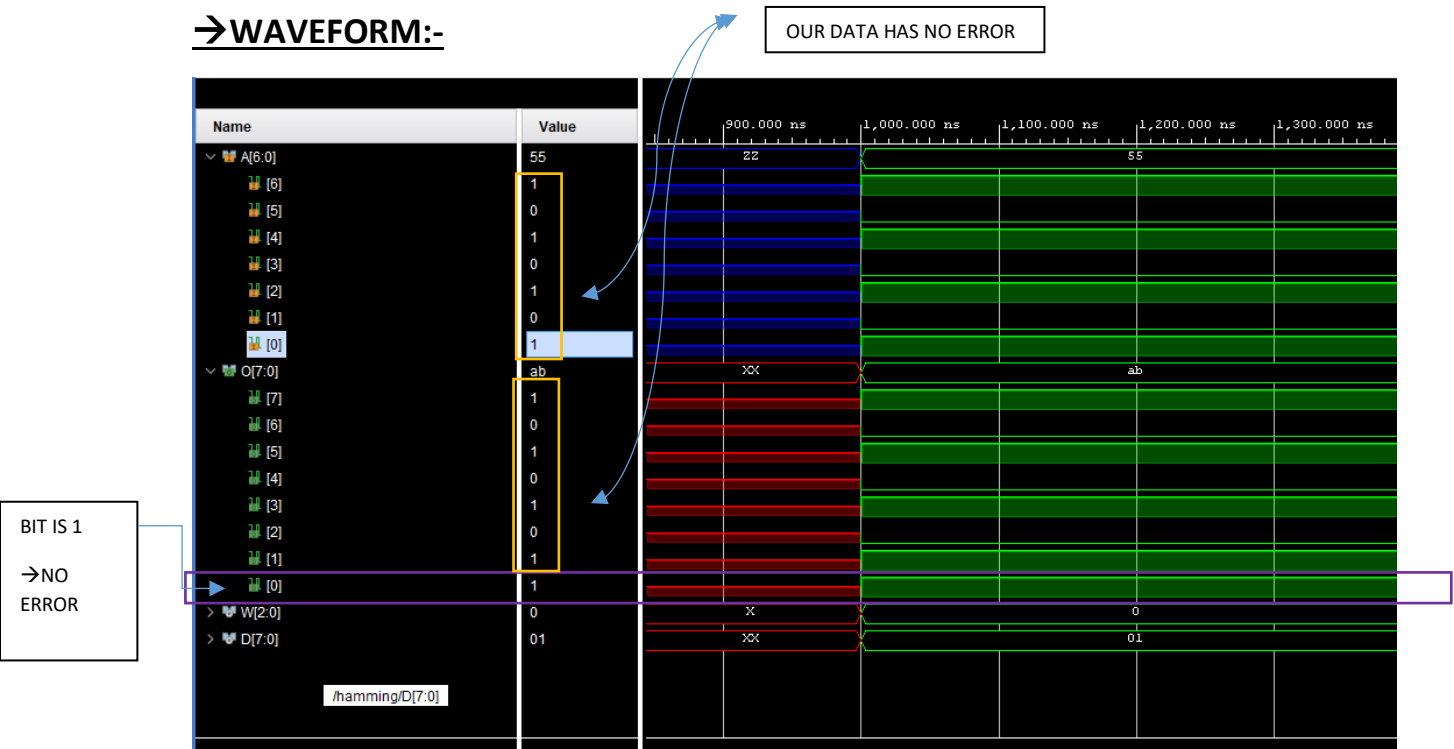
```

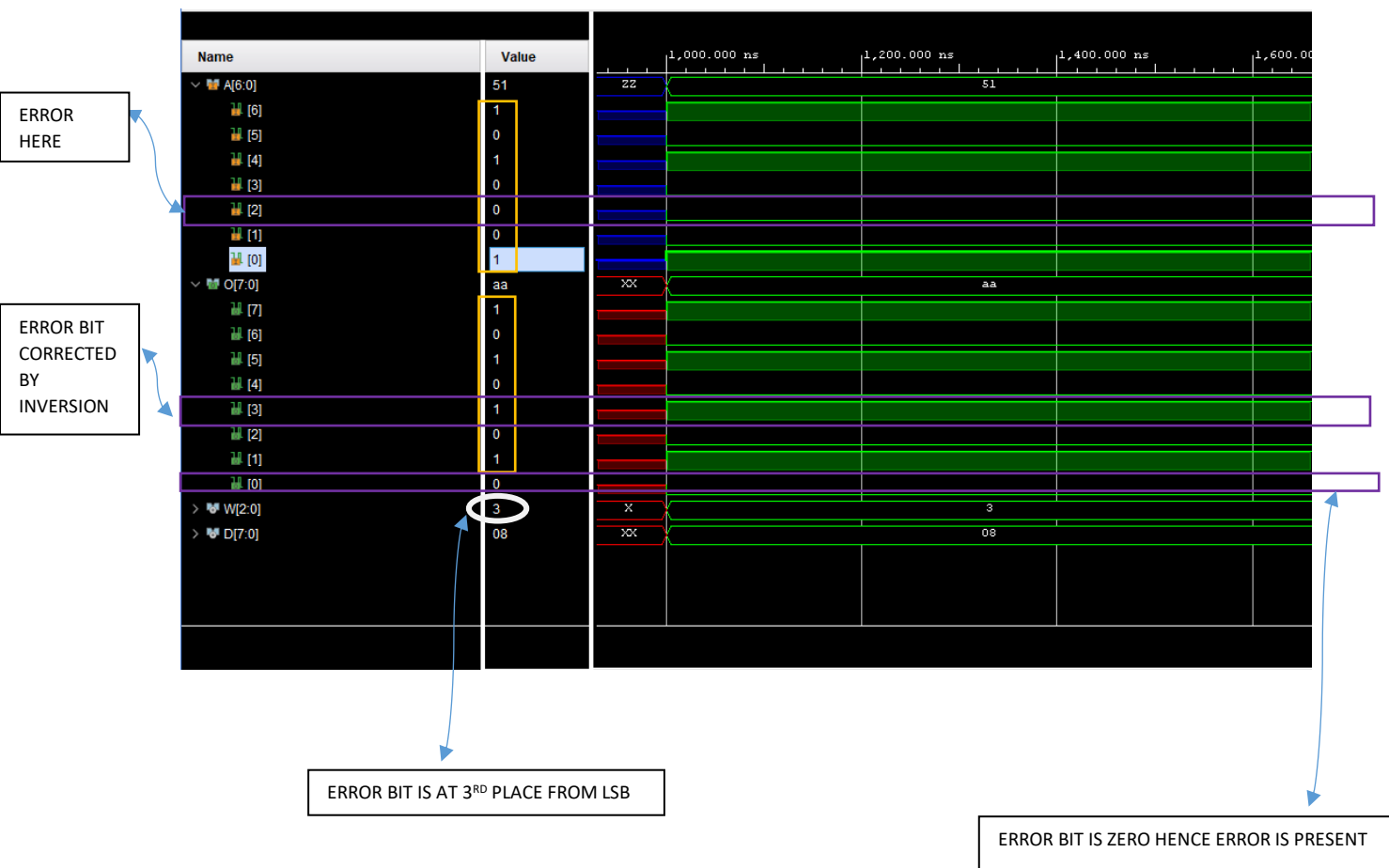
input [2:0] I;
output [7:0] O;
wire [3:0] N;
not(N[0],I[0]);
not(N[1],I[1]);
not(N[2],I[2]);
and(O[0],N[2],N[1],N[0]);
and(O[1],N[2],N[1],I[0]);
and(O[2],N[2],I[1],N[0]);
and(O[3],N[2],I[1],I[0]);
and(O[4],I[2],N[1],N[0]);
and(O[5],I[2],N[1],I[0]);
and(O[6],I[2],I[1],N[0]);
and(O[7],I[2],I[1],I[0]);

```

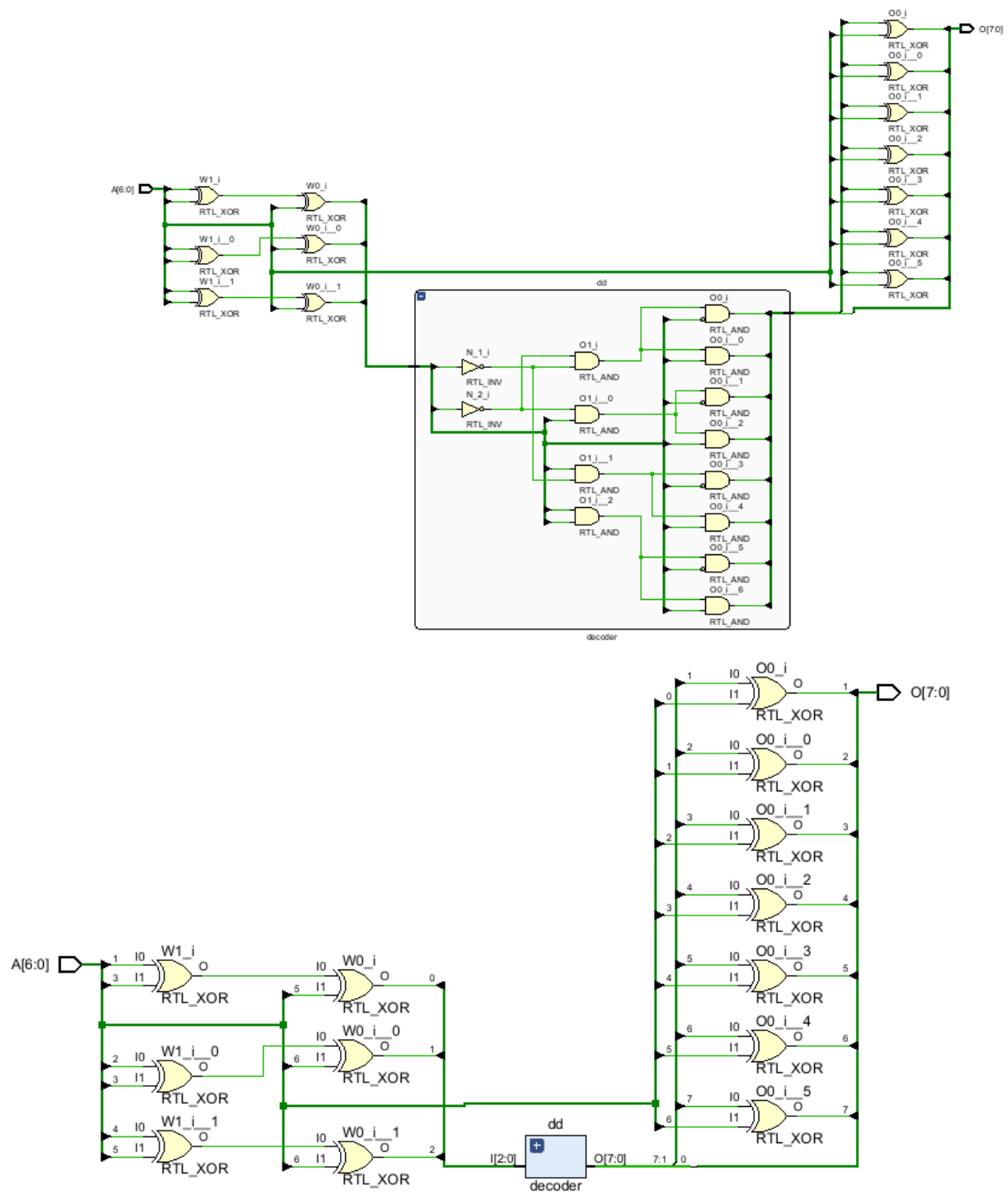
endmodule

→ **WAVEFORM:-**





→ CIRCUIT DIAGRAM:-



7 bit Hamming code testbench:-

```
module hamming_tb();
```

```
reg [6:0] a;
```

```
wire[7:0] o;
```

```
hamming f1(.O(o),.A(a));
```

```
initial
```

```
begin
```

```
$monitor($time,"a=%b,o=%b",a,o);
```

```
a=7'b1001001;
```

```
#10
```

```
a=7'b1001011;
```

```
#10
```

```
a=7'b1101001;
```

```
end
```

```
endmodule
```

The screenshot shows the EDA Playground interface. On the left, there's a sidebar with 'Languages & Libraries' and 'Tools & Simulators'. The main area is split into two code editors: 'testbench.sv' and 'design.sv'. The 'testbench.sv' editor contains the testbench code from the previous blocks. The 'design.sv' editor contains the implementation of the 'hamming' module. Below the code editors, there's a 'Log' section showing the simulation results. The results show the values of 'a' and 'o' at different time intervals.

```
module hamming_tb();
  reg [6:0] a;
  wire [7:0] o;
  hamming f1(.O(o),.A(a));
  initial
  begin
    $monitor($time,"a=%b,o=%b",a,o);
    a=7'b1001001;
    #10
    a=7'b1001011;
    #10
    a=7'b1101001;
  end
endmodule
```

```
module hamming O,A ;
  input [6:0] A;
  output [7:0] O;
  wire [2:0] W;
  wire [7:0] D;
  xor W[0],A[0],A[2],A[4],A[6];
  xor W[1],A[1],A[2],A[5],A[6];
  xor W[2],A[3],A[4],A[5],A[6];
  decoder dd D,W;
  buf O[0],D[0];
  xor O[1],D[1],A[0];
  xor O[2],D[2],A[1];
  xor O[3],D[3],A[2];
  xor O[4],D[4],A[3];
  xor O[5],D[5],A[4];
  xor O[6],D[6],A[5];
  xor O[7],D[7],A[6];
endmodule
```

Log:

```
[2023-02-25 12:24:06 EST] iverilog '-Wall' design.sv testbench.sv && unbuffer vvp a.out
0a=1001001,o=10010110
10a=1001011,o=10010111
20a=1101001,o=11000010
```

Done