6

EXPERIMENT-6
REALIZATION OF ENCODERS AND DECODERS

1. **AIM:** To Design, simulate and implement Encoders and Decoders in 3 different modeling styles(Data Flow, Behavioral Flow Modeling, Structural Flow Modeling)

2. **SOFTWARE USED:** Xilinx Vivado 2022.2

3. **PROCEDURE:**
    - Open the Xilinx Vivado project navigator.
    - Open the Design source and go to add / create sources
    - Create new file, give appropriate name save it.
    - Open the file in the editor and write the Verilog code.
    - Open the Design source and go to add / create sources to create the test bench
    - Open the editor and write the Verilog code for test bench.
    - After completion of Verilog code set your test bench as top module in simulation settings and Run Simulation.
    - To view RTL schematic, select RTL Analysis in which select open elaborate design, select Schematic.

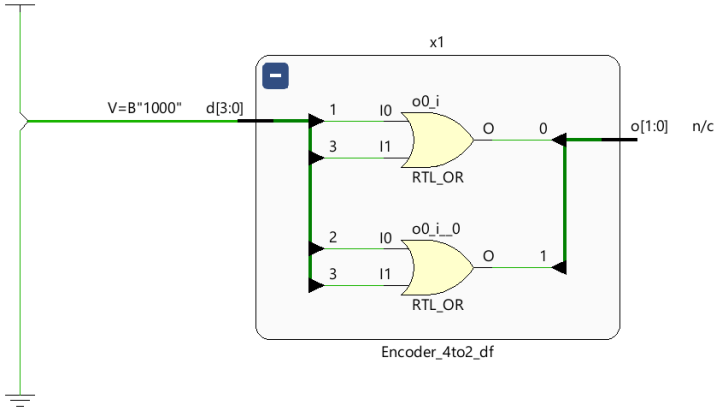
Encoders

1.1 AIM: 4to2 Encoder

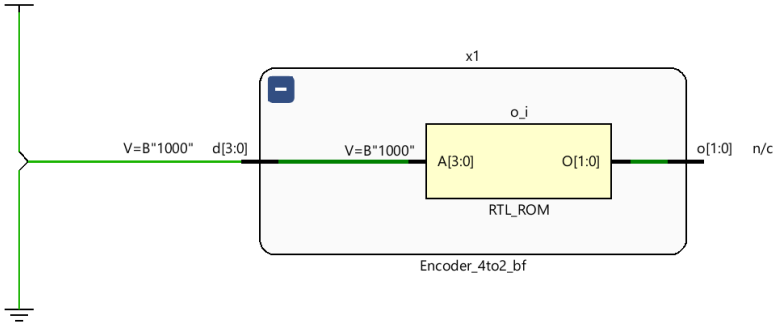
1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 RTL Schematic:

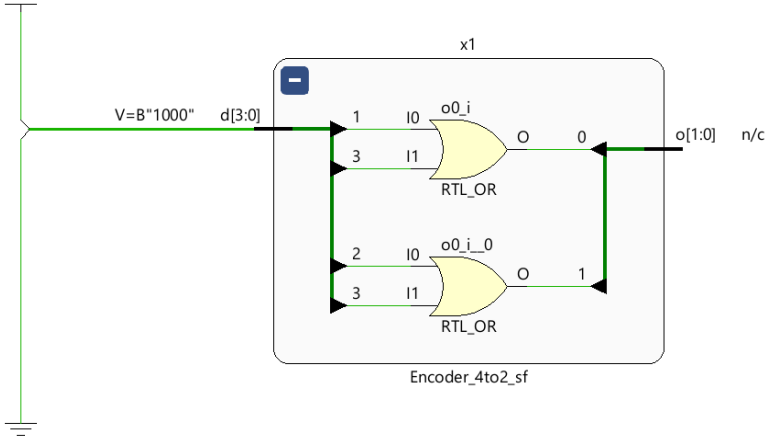
Data Flow Modeling



Behavioral Flow Modeling



Structural Flow Modeling



1.4 LOGIC EXPRESSION:

$X = D_2|D_1$

$Y = D_3|D_1$

1.5 BOOLEAN EXPRESSION:

X = D2+D1

Y = D3+D1

1.6 TRUTH TABLE:

INPUT				OUTPUT	
D_3	D_2	D_1	D_0	X	Y
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

1.7 VERILOG CODE :

Data Flow Modeling (Encoder_4to2_df.v):

```
module Encoder_4to2_df(o,d);
input [3:0]d;
output [1:0]o;
assign o[0] = d[1]|d[3];
assign o[1] = d[2]|d[3];
endmodule
```

Behavioral Flow Modeling (Encoder_4to2_bf.v):

```
module Encoder_4to2_bf(o,d);
input [3:0]d;
output [1:0]o;
reg [1:0]o;
always @ (o,d)
case(d)
4'b0001:begin o=2'b00; end
4'b0010:begin o=2'b01; end
4'b0100:begin o=2'b10; end
4'b1000:begin o=2'b11; end
default:begin o=2'b00; end
endcase
endmodule
```

Structural Flow Modeling (Encoder_4to2_sf.v):

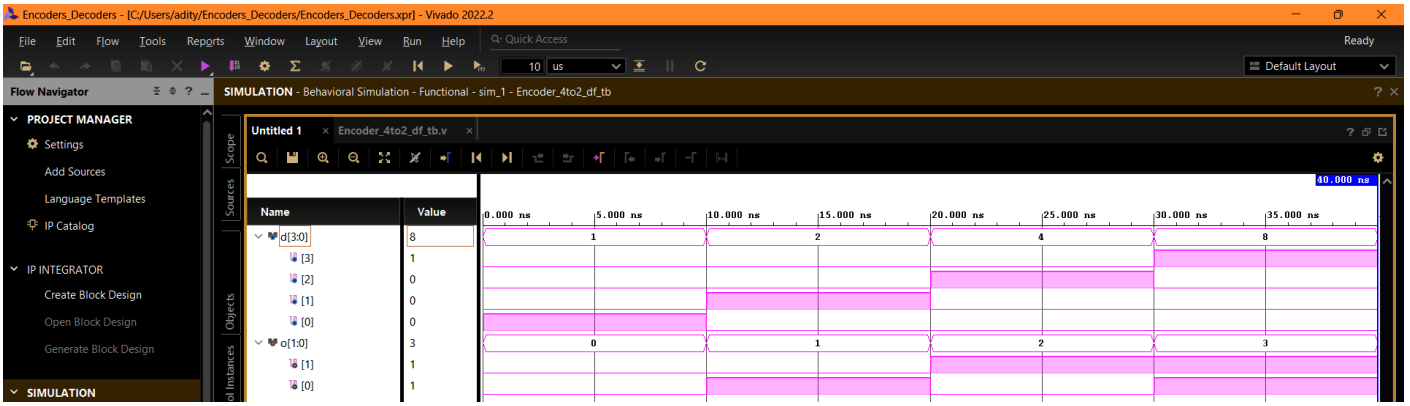
```
module Encoder_4to2_sf(o,d);
input [3:0]d;
output [1:0]o;
or (o[0],d[1],d[3]);
or(o[1],d[2],d[3]);
endmodule
```

1.8 TEST BENCH (Encoder_4to2_df_tb.v):

```
module Encoder_4to2_df_tb();
reg [3:0]d;
wire [1:0]o;
Encoder_4to2_df x1(o,d);
initial begin
d = 4'b1;
#10 d = 4'b10;
#10 d = 4'b100;
#10 d = 4'b1000;
#10 $finish;
```

```
end
endmodule
```

1.9 WAVEFORM:



1.10 RESULT:

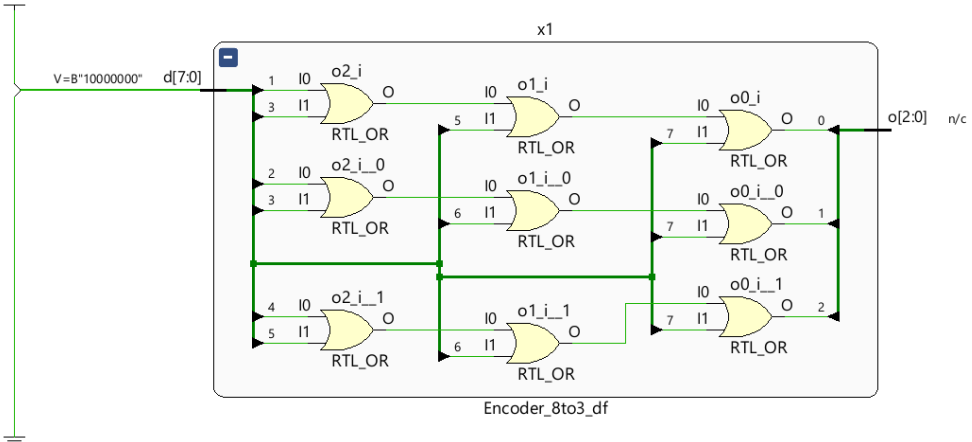
4to 2 Encoder is simulated and implemented in all 3 Modeling Styles.

1.1 AIM: 8to3 Encoder

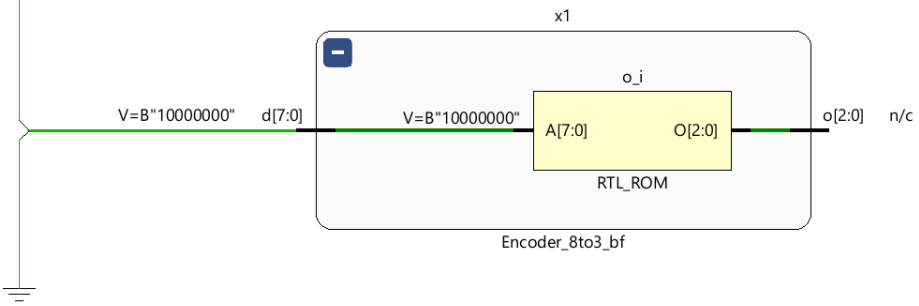
1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 RTL Schematic:

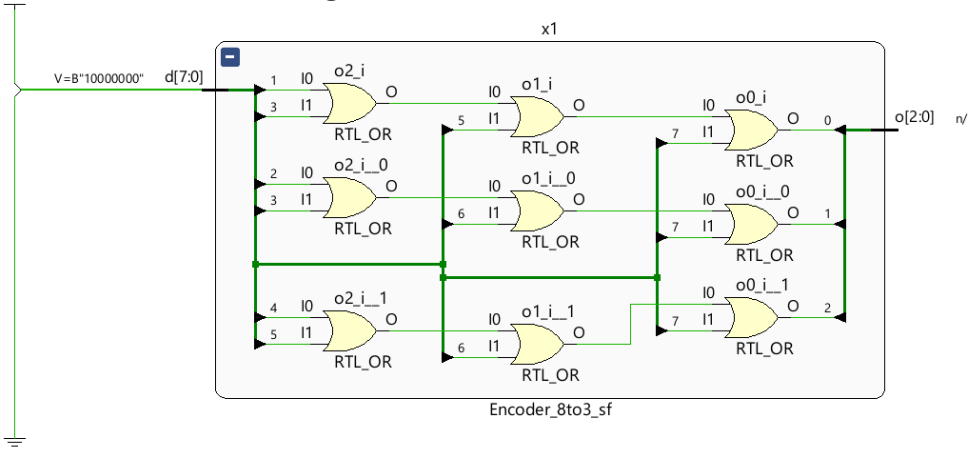
Data Flow Modeling



Behavioral Flow Modeling



Structural Flow Modeling



1.4 LOGIC EXPRESSION:

$X = D_4|D_5|D_6|D_7$

$Y = D_2|D_3|D_6|D_7$

$Z = D_1|D_3|D_5|D_7$

1.5 BOOLEAN EXPRESSION:

$X = D_4+D_5+D_6+D_7$

$Y = D_2+D_3+D_6+D_7$

$Z = D_1+D_3+D_5+D_7$

1.6 TRUTH TABLE:

INPUT								OUTPUT		
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	**X**	**Y**	**Z**
0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**
0	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**
0	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**1**
0	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
0	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
0	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**
1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**

1.7 VERILOG CODE :

Data Flow Modeling (Encoder_8to3_df.v):

```
module Encoder_8to3_df(o,d);
input [7:0]d;
output [2:0]o;
assign o[2] = d[4]|d[5]|d[6]|d[7];
assign o[1] = d[2]|d[3]|d[6]|d[7];
assign o[0] = d[1]|d[3]|d[5]|d[7];
endmodule
```

Behavioral Flow Modeling (Encoder_4to2_bf.v):

```
module Encoder_8to3_bf(o,d);
input [7:0]d;
output reg [2:0]o;
always @ (o,d)
case(d)
8'b1:begin o=3'b00; end
8'b10:begin o=3'b01; end
8'b100:begin o=3'b10; end
8'b1000:begin o=3'b11; end
8'b10000:begin o=3'b100; end
8'b100000:begin o=3'b101; end
8'b1000000:begin o=3'b110; end
8'b10000000:begin o=3'b111; end
default:begin o=3'b00; end
endcase
endmodule
```

Structural Flow Modeling (Encoder_4to2_sf.v):

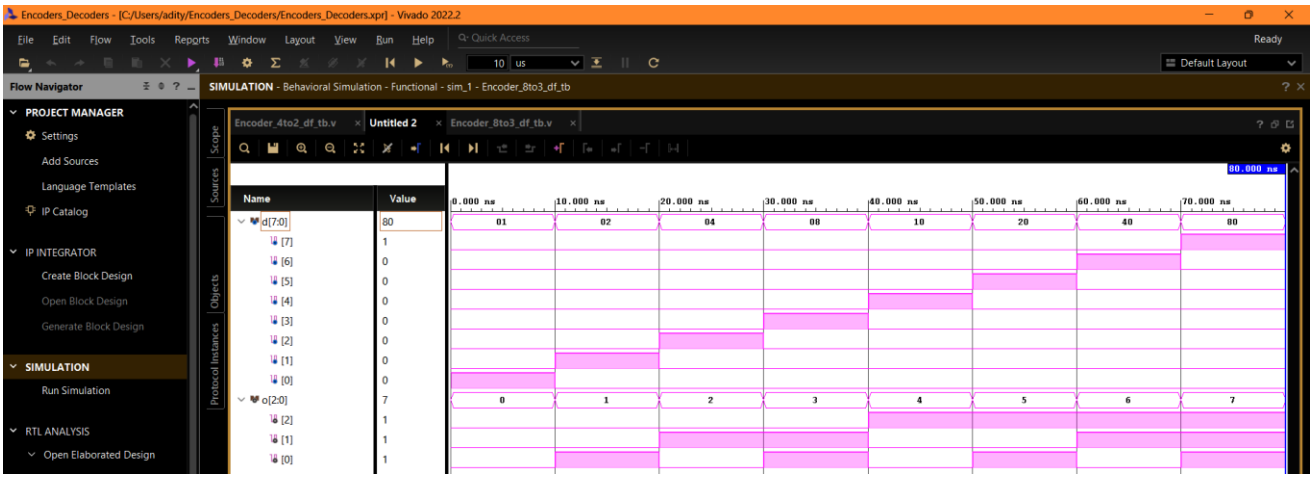
```
module Encoder_8to3_sf(o,d);
input [7:0]d;
output [2:0]o;
or (o[2],d[4],d[5],d[6],d[7]);
or (o[1],d[2],d[3],d[6],d[7]);
or (o[0],d[1],d[3],d[5],d[7]);
endmodule
```

1.8 TEST BENCH (Encoder_4to2_df_tb.v):

```
module Encoder_8to3_df_tb();
reg [7:0]d;
wire [2:0]o;
Encoder_8to3_df x1(o,d);
```

```
    initial begin
      d = 8'b1;
    #10 d = 8'b10;
    #10 d = 8'b100;
    #10 d = 8'b1000;
    #10 d = 8'b10000;
    #10 d = 8'b100000;
    #10 d = 8'b1000000;
    #10 d = 8'b10000000;
    #10 $finish;
    end
    endmodule
```

1.9 WAVEFORM:



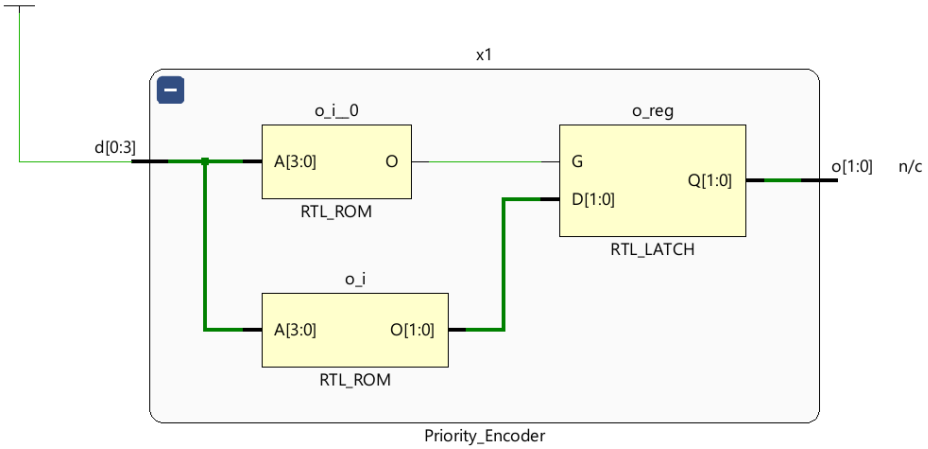
1.10 RESULT:

8to 3 Encoder is simulated and implemented in all 3 Modeling Styles.

1.1 AIM: 4to2 Priority Encoder

1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 RTL Schematic:



1.4 LOGIC EXPRESSION:

$X = D_2|D_1$

$Y = D_3|D_1$

1.5 BOOLEAN EXPRESSION:

X = D2+D1

Y = D3+D1

1.6 TRUTH TABLE:

INPUT				OUTPUT	
D_3	D_2	D_1	D_0	X	Y
0	0	0	1	0	0
0	0	1	X	0	1
0	1	X	X	1	0
1	X	X	X	1	1

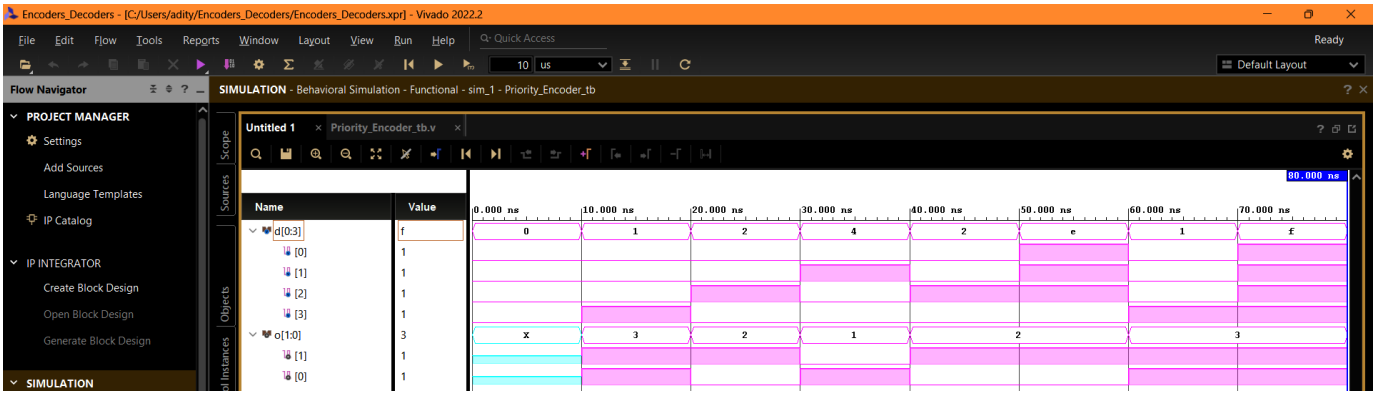
1.7 VERILOG CODE (Priority_Encoder.v):

```
module Priority_Encoder(d,o);
output [1:0]o;
input [0:3]d;
reg [1:0]o;
always@(d)
begin
case({d})
4'b0000:begin o=2'bxx;end
4'b1000:begin o=2'b00;end
4'b0100:begin o=2'b01;end
4'b1100:begin o=2'b01;end
4'b0010:begin o=2'b10;end
4'b1110:begin o=2'b10;end
4'b0001:begin o=2'b11;end
4'b1111:begin o=2'b11;end
endcase
end
endmodule
```

1.8 TEST BENCH (Priority_Encoder_tb.v):

```
module Priority_Encoder_tb();
reg [0:3]d;
wire [1:0]o;
Priority_Encoder x1(d,o);
initial
begin
{d}=4'b0000;
#10 {d}=4'b0001;
#10 {d}=4'b0010;
#10 {d}=4'b0100;
#10 {d}=4'b0010;
#10 {d}=4'b1110;
#10 {d}=4'b0001;
#10 {d}=4'b1111;
#10 $finish;
end
endmodule
```

1.9 WAVEFORM:



1.10 RESULT:

Priority Encoder is simulated and implemented.

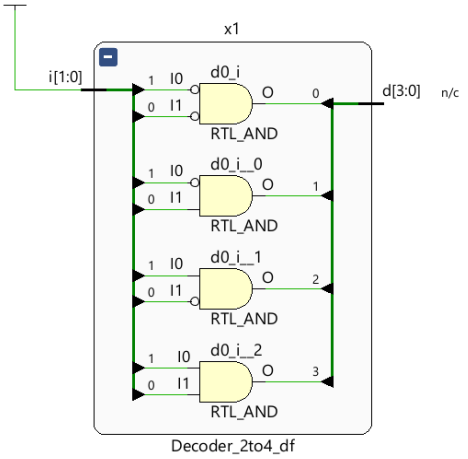
Decoders

1.1 AIM: 2to4 Decoder

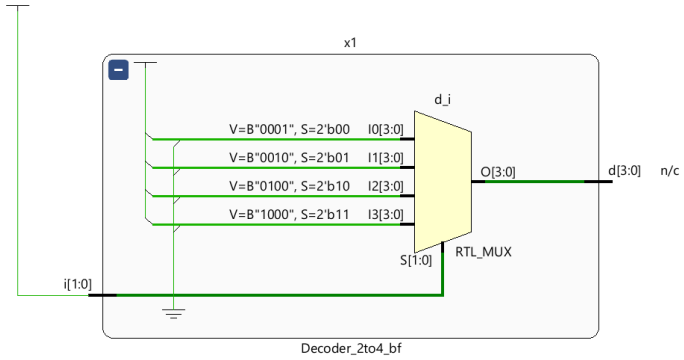
1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 RTL Schematic:

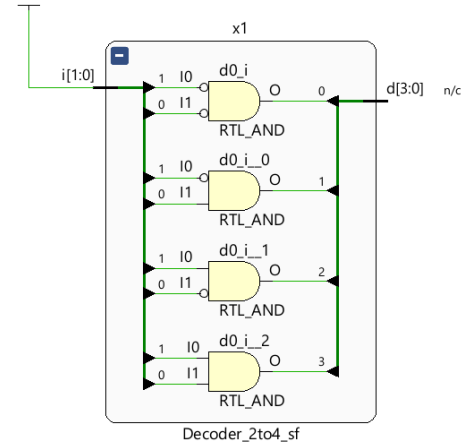
Data Flow Modeling



Behavioral Flow Modeling



Structural Flow Modeling



1.4 LOGIC EXPRESSION:

$D_0 = (\sim A)\&(\sim B)$

$D_1 = (\sim A)\&(B)$

$D_2 = (A)\&(\sim B)$

$D_3 = (A)\&(B)$

1.5 BOOLEAN EXPRESSION:

$D_0 = (\overline{A})(\overline{B})$
$D_1 = (\overline{A})(B)$
$D_2 = (A)(\overline{B})$
$D_3 = (A)(B)$

1.6 TRUTH TABLE:

INPUT		OUTPUT			
A	**B**	**D_0**	D_1	D_2	D_3
0	**0**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**0**	**0**
1	**0**	**0**	**0**	**1**	**0**
1	**1**	**0**	**0**	**0**	**1**

1.7 VERILOG CODE :

Data Flow Modeling (Decoder_2to4_df.v):

```
module Decoder_2to4_df(d,i);
input [1:0]i;
output [3:0]d;
assign d[0] = (~i[1])&(~i[0]);
assign d[1] = (~i[1])&(i[0]);
assign d[2] = (i[1])&(~i[0]);
assign d[3] = (i[1])&(i[0]);
endmodule
```

Behavioral Flow Modeling (Decoder_2to4_bf.v):

```
module Decoder_2to4_bf(d,i);
input [1:0]i;
output reg [3:0]d;
always @ (d,i)
case(i)
2'b00:begin d=0;d[0]=1;end
2'b01:begin d=0;d[1]=1;end
2'b10:begin d=0;d[2]=1;end
2'b11:begin d=0;d[3]=1;end
endcase
endmodule
```

Structural Flow Modeling (Decoder_2to4_sf.v):

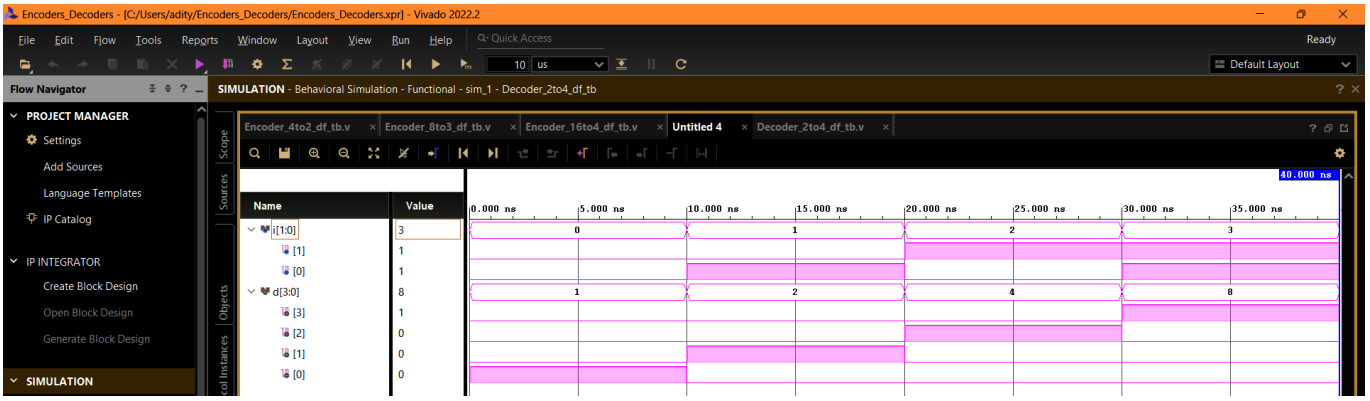
```
module Decoder_2to4_sf(d,i);
input [1:0]i;
output [3:0]d;
and (d[0],(~i[1]),(~i[0]));
and (d[1],(~i[1]),(i[0]));
and (d[2],(i[1]),(~i[0]));
and (d[3],(i[1]),(i[0]));
endmodule
```

1.8 TEST BENCH (Decoder_2to4_df_tb.v):

```
module Decoder_2to4_df_tb();
reg [1:0]i;
wire[3:0]d;
Decoder_2to4_df x1(d,i);
initial begin
```

```
        i=2'b00;
    #10 i = 2'b01;
    #10 i = 2'b10;
    #10 i = 2'b11;
    #10 $finish;
    end
    endmodule
```

1.9 WAVEFORM:



1.10 RESULT:

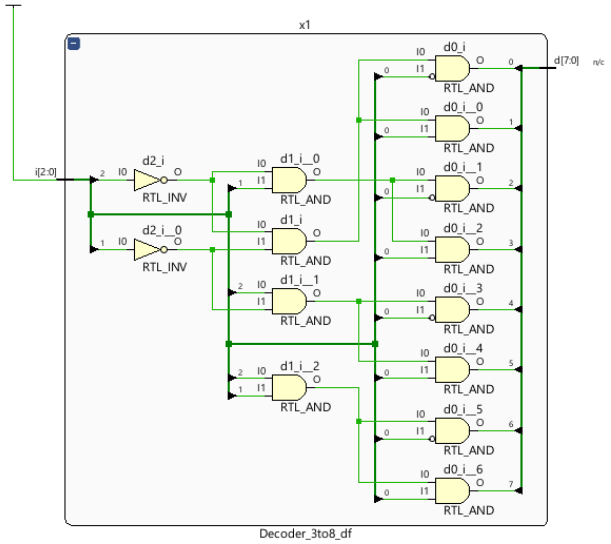
2to4 Decoder is simulated and implemented in all 3 Modeling Styles.

1.1 AIM: 3to8 Decoder

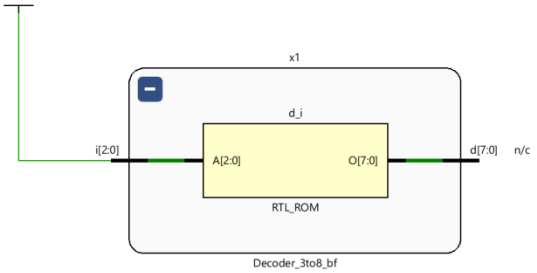
1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 RTL Schematic:

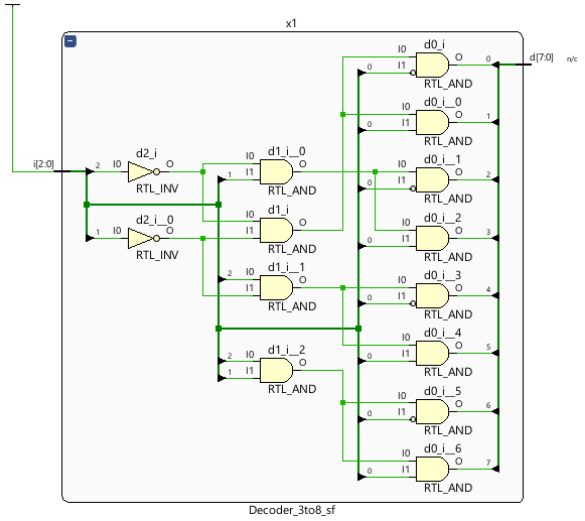
Data Flow Modeling



Behavioral Flow Modeling



Structural Flow Modeling



1.4 LOGIC EXPRESSION:

$D_0 = (\sim A)\&(\sim B)\&(\sim C)$

$D_1 = (\sim A)\&(\sim B)\&(C)$

$D_2 = (\sim A)\&(B)\&(\sim C)$

$D_3 = (\sim A)\&(B)\&(C)$

$D_4 = (A)\&(\sim B)\&(\sim C)$
$D_5 = (A)\&(\sim B)\&(C)$
$D_6 = (A)\&(B)\&(\sim C)$
$D_7 = (A)\&(B)\&(C)$

1.5 BOOLEAN EXPRESSION:

$D_0 = (\overline{A})(\overline{B})(\overline{C})$
$D_1 = (\overline{A})(\overline{B})(C)$
$D_2 = (\overline{A})(B)(\overline{C})$
$D_3 = (\overline{A})(B)(C)$
$D_4 = (A)(\overline{B})(\overline{C})$
$D_5 = (A)(\overline{B})(C)$
$D_6 = (A)(B)(\overline{C})$
$D_7 = (A)(B)(C)$

1.6 TRUTH TABLE:

INPUT			OUTPUT							
A	**B**	**C**	**D_0**	**D_1**	**D_2**	**D_3**	**D_4**	**D_5**	**D_6**	**D_7**
0	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**
0	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
0	**1**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
1	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
1	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**
1	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
1	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**

1.7 VERILOG CODE :

Data Flow Modeling (Decoder_3to8_df.v):

```
module Decoder_3to8_df(d,i);
input [2:0]i;
output [7:0]d;
assign d[0] = (~i[2])&(~i[1])&(~i[0]);
assign d[1] = (~i[2])&(~i[1])&(i[0]);
assign d[2] = (~i[2])&(i[1])&(~i[0]);
assign d[3] = (~i[2])&(i[1])&(i[0]);
assign d[4] = (i[2])&(~i[1])&(~i[0]);
assign d[5] = (i[2])&(~i[1])&(i[0]);
assign d[6] = (i[2])&(i[1])&(~i[0]);
assign d[7] = (i[2])&(i[1])&(i[0]);
endmodule
```

Behavioral Flow Modeling (Decoder_3to8_bf.v):

```
module Decoder_3to8_bf(d,i);
input [2:0]i;
output reg [7:0]d;
always @ (d,i)
case(i)
3'b00:begin d=0;d[0]=1;end
3'b01:begin d=0;d[1]=1;end
3'b10:begin d=0;d[2]=1;end
3'b11:begin d=0;d[3]=1;end
3'b100:begin d=0;d[4]=1;end
```

```
    3'b101:begin d=0;d[5]=1;end
    3'b110:begin d=0;d[6]=1;end
    3'b111:begin d=0;d[7]=1;end
   endcase
    endmodule
```

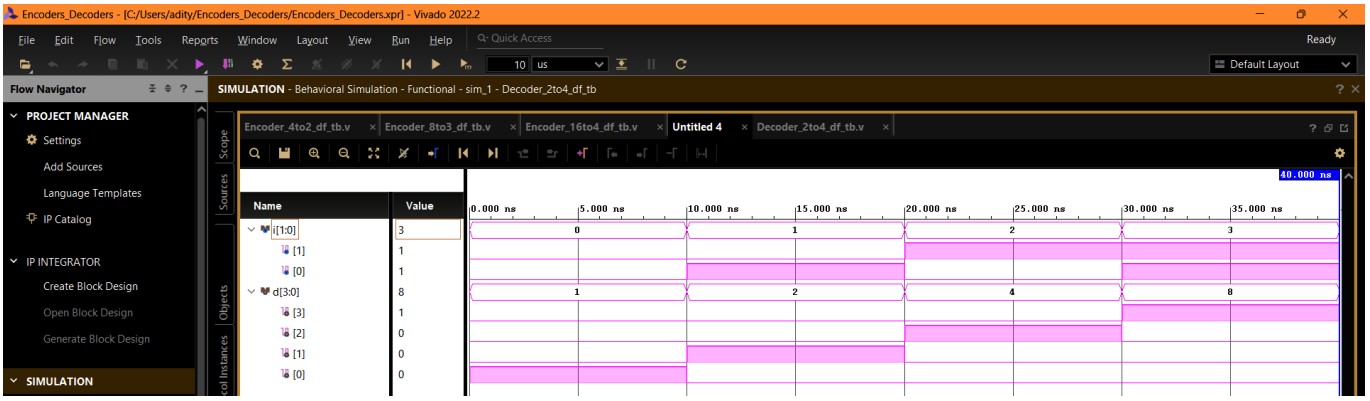
Structural Flow Modeling (Decoder_3to8_sf.v):

```
    module Decoder_3to8_sf(d,i);
    input [2:0]i;
    output [7:0]d;
    and (d[0],(~i[2]),(~i[1]),(~i[0]));
    and (d[1],(~i[2]),(~i[1]),(i[0]));
    and (d[2],(~i[2]),(i[1]),(~i[0]));
    and (d[3],(~i[2]),(i[1]),(i[0]));
    and (d[4],(i[2]),(~i[1]),(~i[0]));
    and (d[5],(i[2]),(~i[1]),(i[0]));
    and (d[6],(i[2]),(i[1]),(~i[0]));
    and (d[7],(i[2]),(i[1]),(i[0]));
    endmodule
```

1.8 TEST BENCH (Decoder_3to8_df_tb.v):

```
    module Decoder_3to8_df_tb();
    reg [2:0]i;
    wire[7:0]d;
    Decoder_3to8_df x1(d,i);
    initial begin
        i=3'b0;
    #10 i = 3'b1;
    #10 i = 3'b10;
    #10 i = 3'b11;
    #10 i = 3'b100;
    #10 i = 3'b101;
    #10 i = 3'b110;
    #10 i = 3'b111;
    #10 $finish;
    end
    endmodule
```

WAVEFORM:



1.9 RESULT:

3to8 Decoder is simulated and implemented in all 3 Modeling Styles.