4

EXPERIMENT-4
REALIZATION OF MULTIPLEXERS AND DEMULTIPLEXERS

1. **AIM:** To Design, simulate and implement Multiplexers and Demultiplexers in 2 different modeling styles(Data Flow, Structural Flow Modeling)

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

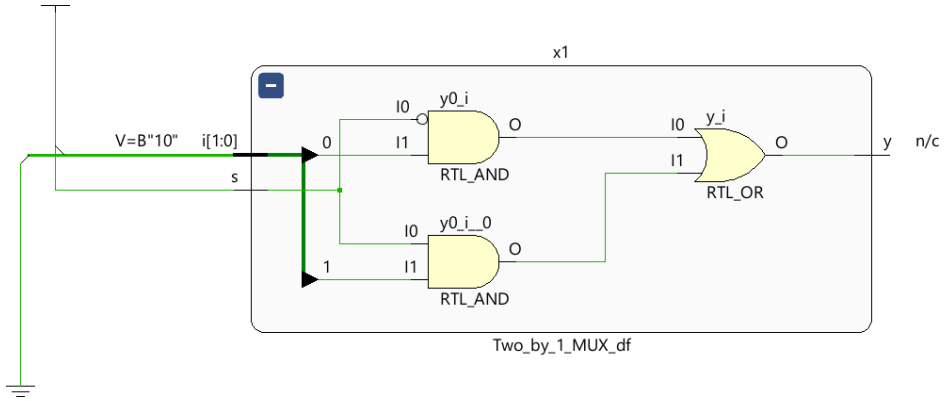
3. **PROCEDURE:**
    - Open the Xilinx Vivado project navigator.
    - Open the Design source and go to add / create sources
    - Create new file, give appropriate name save it.
    - Open the file in the editor and write the Verilog code.
    - Open the Design source and go to add / create sources 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.

Multiplexers (Data Flow Modeling)

1.1 AIM: 2x1 Multiplexer [MUX]

1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 SYMBOL:



1.4 LOGIC EXPRESSION:

$$Y = ((\sim S)\&I_0) \mid (S)\&I_1))$$

1.5 BOOLEAN EXPRESSION:

$$Y = \bar{S}I_0 + SI_1$$

1.6 TRUTH TABLE:

INPUT	OUTPUT
S	**Y**
0	**I_0**
1	**I_1**

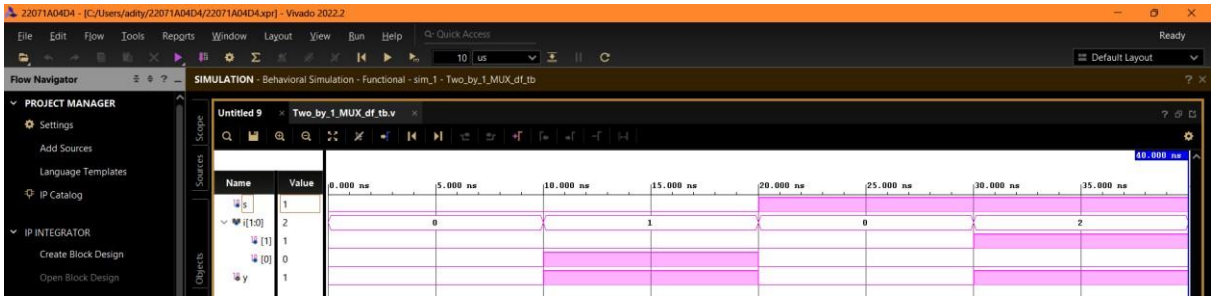
1.7 VERILOG CODE (Two_by_1_MUX_df.v):

```
module Two_by_1_MUX_df(y,s,i);
output y;
input s;
input[1:0]i;
assign y = (((~s)&i[0])|((s)&i[1]));
endmodule
```

1.8 TEST BENCH (Two_by_1_MUX_df_tb.v):

```
module Two_by_1_MUX_df_tb();
reg s;
reg[1:0]i;
wire y;
Two_by_1_MUX_df x1(y,s,i);
initial
begin
s  =  1'b0;i=2'b00;
#10 s = 1'b0;i=2'b01;

#10 s = 1'b1;i=2'b00;
#10 s = 1'b1;i=2'b10;
#10 $finish;
end
endmodule
```

1.9 WAVEFORM:



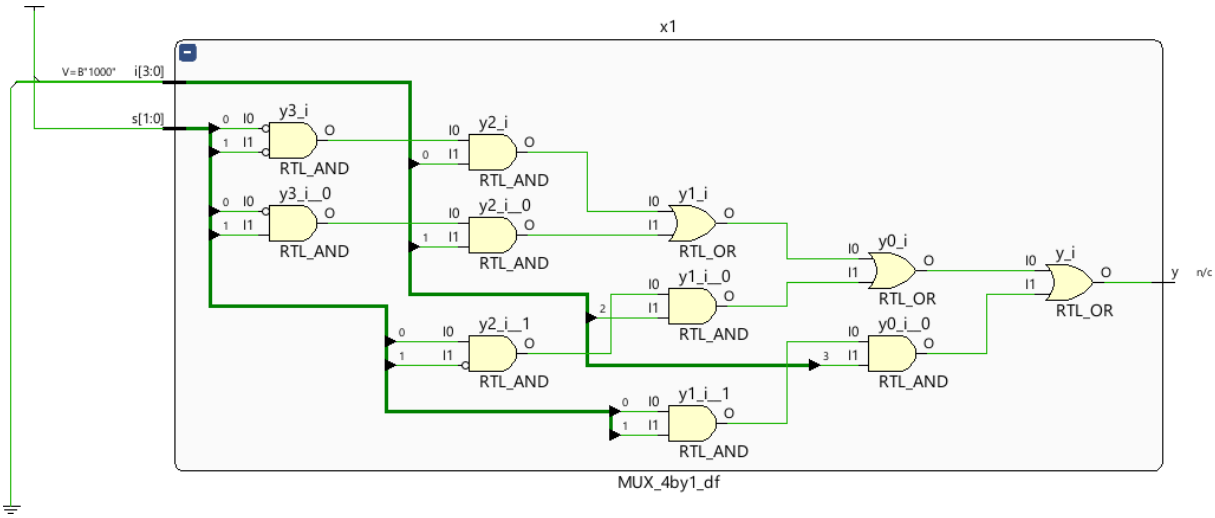
1.10 RESULT:

2x1 Multiplexer is simulated and implemented in Data Flow Modeling.

1.1 AIM: 4x1 Multiplexer [MUX]

1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 SYMBOL:



1.4 LOGIC EXPRESSION:

$$Y = ((\sim S_0)\&(\sim S_1)\&I_0) \mid ((\sim S_0)\&(S_1)\&I_1) \mid ((S_0)\&(\sim S_1)\&I_2) \mid ((S_0)\&(S_1)\&I_3)$$

1.5 BOOLEAN EXPRESSION:

$$Y = \overline{S_0}\,\overline{S_1}I_0 + \overline{S_0}S_1I_1 + S_0\overline{S_1}I_2 + S_0S_1I_3$$

1.6 TRUTH TABLE:

INPUT		OUTPUT
S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

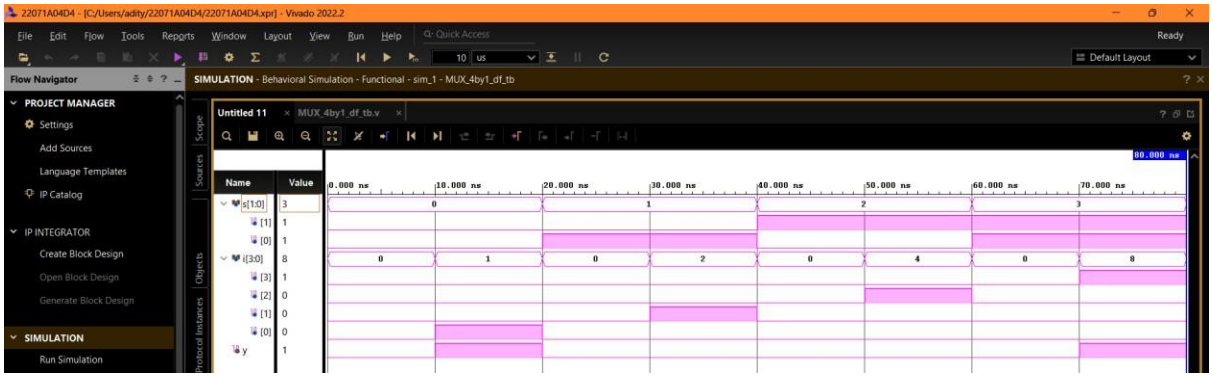
1.7 VERILOG CODE (MUX_4by1_df.v):

```
module MUX_4by1_df(y,s,i);
output y;
input [1:0]s;
input[3:0]i;

assign y = (
((~s[0])&(~s[1])&(i[0]))|
((~s[0])&(s[1])&(i[1]))|
((s[0])&(~s[1])&(i[2]))|
((s[0])&(s[1])&(i[3]))
);
endmodule
```

1.8 TEST BENCH (MUX_4by1_df_tb.v):

```
module MUX_4by1_df_tb();
reg [1:0]s;
reg[3:0]i;
wire y;

MUX_4by1_df x1(y,s,i);

initial
begin
s  =  2'b00;i=4'b0000;
#10 s = 2'b00;i=4'b0001;

#10 s = 2'b01;i=4'b0000;
#10 s = 2'b01;i=4'b0010;

#10 s = 2'b10;i=4'b0000;
#10 s = 2'b10;i=4'b0100;

#10 s = 2'b11;i=4'b0000;
#10 s = 2'b11;i=4'b1000;

#10 $finish;

end
endmodule
```

1.9 WAVEFORM:



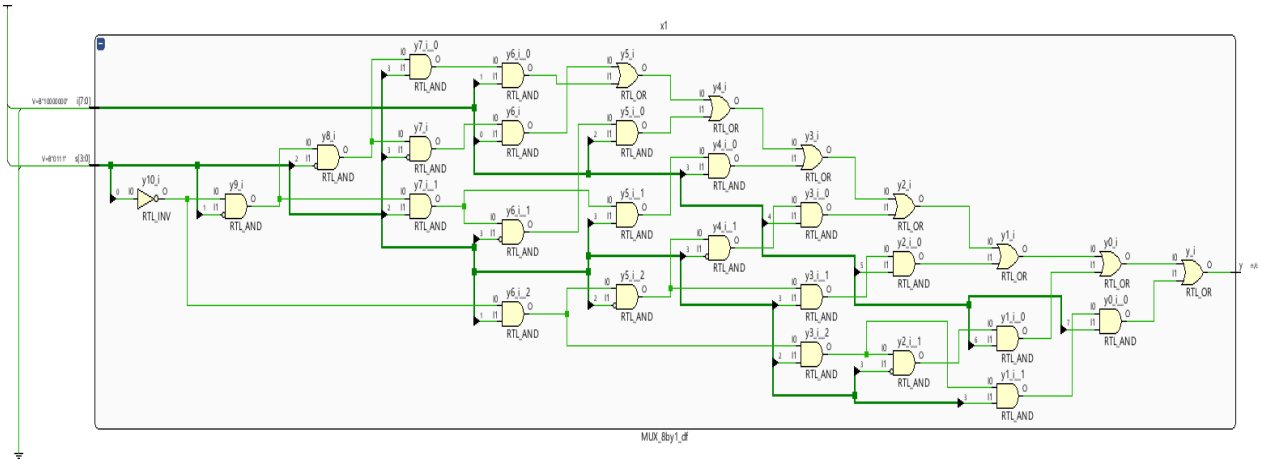
1.10 RESULT:

4x1 Multiplexer is simulated and implemented in Data Flow Modeling.

1.1 AIM: 8x1 Multiplexer [MUX]

1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 SYMBOL:



1.4 LOGIC EXPRESSION:

$$Y = ((\sim S_0) \,\&\, (\sim S_1) \,\&\, (\sim S_2) \,\&\, (\sim S_3) \,\&\, I_0) \,|$$
$$((\sim S_0) \,\&\, (\sim S_1) \,\&\, (\sim S_2) \,\&\, (S_3) \,\&\, I_1) \,|$$
$$((\sim S_0) \,\&\, (\sim S_1) \,\&\, (S_2) \,\&\, (\sim S_3) \,\&I_2) \,|$$
$$((\sim S_0) \,\&\, (\sim S_1) \,\&\, (S_2) \,\&\, (S_3) \,\&I_3) \,|$$
$$((\sim S_0) \,\&\, (S_1) \,\&\, (\sim S_2) \,\&\, (\sim S_3) \,\&I_4) \,|$$
$$((\sim S_0) \,\&\, (S_1) \,\&\, (\sim S_2) \,\&\, (S_3) \,\&I_5) \,|$$
$$((\sim S_0) \,\&\, (S_1) \,\&\, (S_2) \,\&\, (\sim S_3) \,\&I_6) \,|$$
$$((\sim S_0) \,\&\, (S_1) \,\&\, (S_2) \,\&\, (S_3) \,\&I_7)$$

1.5 BOOLEAN EXPRESSION:

$$Y = \overline{S_0}\overline{S_1}\overline{S_2}\overline{S_3}I_0 + \overline{S_0}\overline{S_1}\overline{S_2}S_3I_1 + \overline{S_0}\overline{S_1}S_2\overline{S_3}I_2 + \overline{S_0}\overline{S_1}S_2S_3I_3 + \overline{S_0}S_1\overline{S_2}\overline{S_3}I_4 + \overline{S_0}S_1\overline{S_2}S_3I_5 + \overline{S_0}S_1S_2\overline{S_3}I_6 + \overline{S_0}S_1S_2S_3I_7$$

1.6 TRUTH TABLE:

INPUT				**OUTPUT**
S_0	**S_1**	**S_2**	**S_3**	**I**
0	**0**	**0**	**0**	**I_0**
0	**0**	**0**	**1**	**I_1**
0	**0**	**1**	**0**	**I_2**
0	**0**	**1**	**1**	**I_3**
0	**1**	**0**	**0**	**I_4**
0	**1**	**0**	**1**	**I_5**
0	**1**	**1**	**0**	**I_6**
0	**1**	**1**	**1**	**I_7**

1.7 VERILOG CODE (MUX_8by1_df.v):

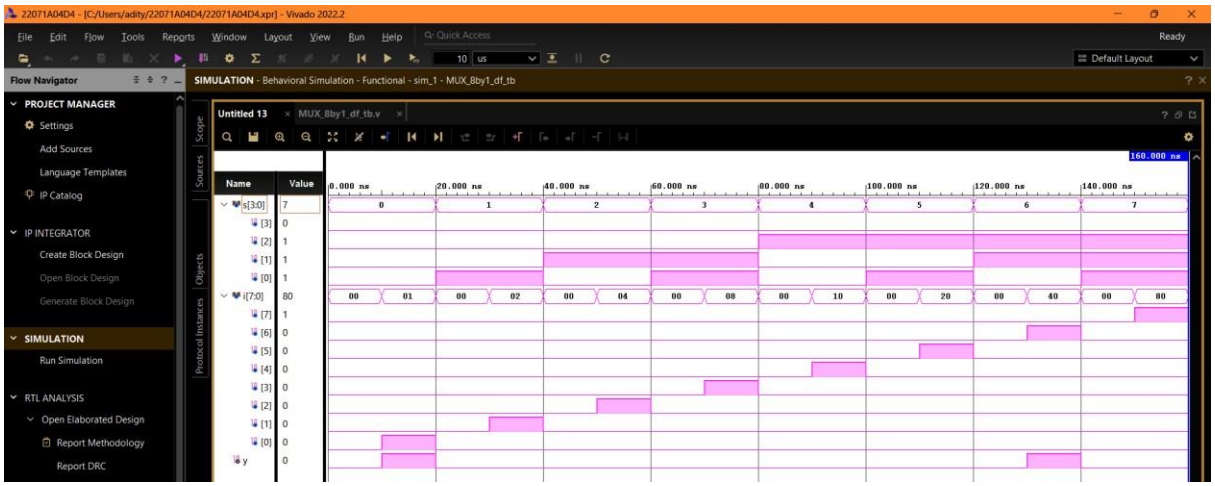
```
module MUX_8by1_df(y,s,i);
output y;
input [3:0]s;
input[7:0]i;
assign y = (
((~s[0])&(~s[1])&(~s[2])&(~s[3])&(i[0]))|
((~s[0])&(~s[1])&(~s[2])&(s[3])&(i[1]))|
((~s[0])&(~s[1])&(s[2])&(~s[3])&(i[2]))|
((~s[0])&(~s[1])&(s[2])&(s[3])&(i[3]))|
((~s[0])&(s[1])&(~s[2])&(~s[3])&(i[4]))|
((~s[0])&(s[1])&(~s[2])&(s[3])&(i[5]))|
((~s[0])&(s[1])&(s[2])&(~s[3])&(i[6]))|
((~s[0])&(s[1])&(s[2])&(s[3])&(i[7]))
);
endmodule.
```

1.8 TEST BENCH (MUX_8by1_df_tb.v):

```
module MUX_8by1_df_tb();
reg [3:0]s;
reg[7:0]i;
wire y;

MUX_8by1_df x1(y,s,i);

initial
begin
s  =  4'b0000;i=8'b00000000;
#10 s = 4'b0000;i=8'b00000001;

#10 s = 4'b0001;i=8'b00000000;
#10 s = 4'b0001;i=8'b00000010;

#10 s = 4'b0010;i=8'b00000000;
#10 s = 4'b0010;i=8'b00000100;

#10 s = 4'b0011;i=8'b00000000;
#10 s = 4'b0011;i=8'b00001000;

#10 s = 4'b0100;i=8'b00000000;
#10 s = 4'b0100;i=8'b00010000;

#10 s = 4'b0101;i=8'b00000000;
#10 s = 4'b0101;i=8'b00100000;

#10 s = 4'b0110;i=8'b00000000;
#10 s = 4'b0110;i=8'b01000000;
```

```
#10 s = 4'b0111;i=8'b00000000;
#10 s = 4'b0111;i=8'b10000000;


#10 $finish;

end
endmodule
```

1.9 WAVEFORM:



1.10 RESULT:

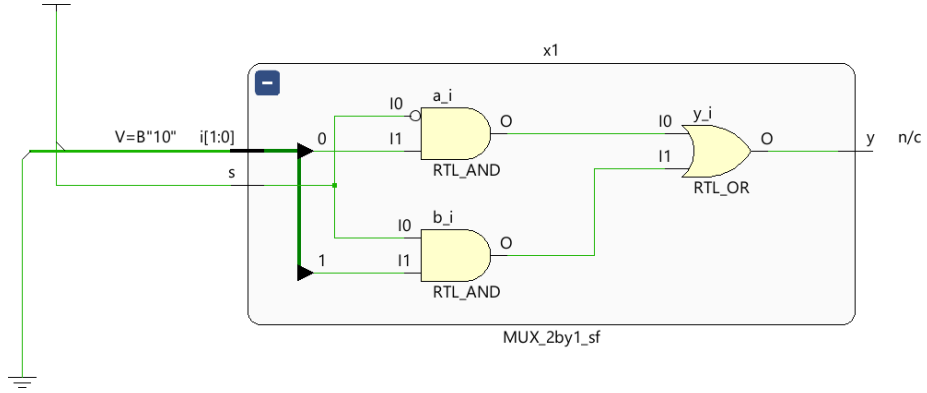
8x1 Multiplexer is simulated and implemented in Data Flow Modeling.

Multiplexers (Structural Flow Modeling)

1.1 AIM: 2x1 Multiplexer [MUX]

1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 SYMBOL:



1.4 LOGIC EXPRESSION:

$$Y = ((\sim S)\&I_0) \mid (S)\&I_1))$$

1.5 BOOLEAN EXPRESSION:

$$Y = \bar{S}I_0 + SI_1$$

1.6 TRUTH TABLE:

INPUT	OUTPUT
S	**Y**
0	**I_0**
1	**I_1**

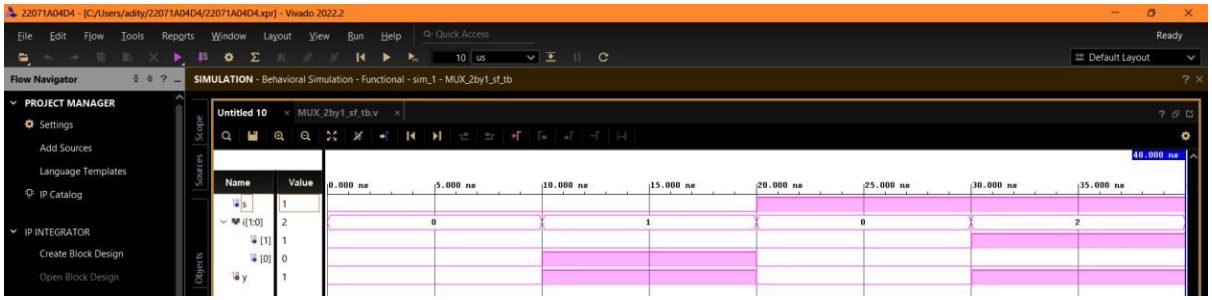
1.7 VERILOG CODE (MUX_2by1_sf.v):

```
module MUX_2by1_sf(y,s,i);
output y;

input s;
input[1:0]i;

wire a,b,c;

and (a,~s,i[0]);

and (b,s,i[1]);

or(y,a,b);

endmodule
```

1.8 TEST BENCH (MUX_2by1_sf_tb.v):

```
module MUX_2by1_sf_tb();
reg s;
reg[1:0]i;
wire y;

MUX_2by1_sf x1(y,s,i);

initial
begin
s  =  1'b0;i=2'b00;
#10 s = 1'b0;i=2'b01;

#10 s = 1'b1;i=2'b00;
#10 s = 1'b1;i=2'b10;

#10 $finish;

end
endmodule
```

1.9 WAVEFORM:



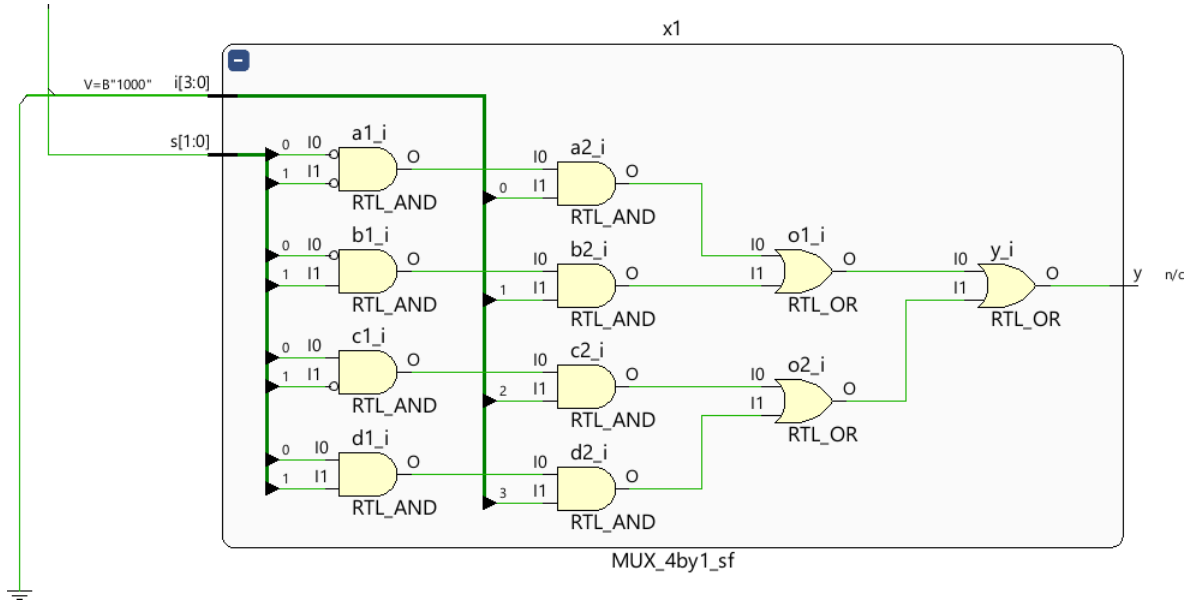
1.10 RESULT:

2x1 Multiplexer is simulated and implemented in Structural Flow Modeling.

1.1 AIM: 4x1 Multiplexer [MUX]

1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 SYMBOL:



1.4 LOGIC EXPRESSION:

$$Y = ((\sim S_0)\&(\sim S_1)\&I_0) \mid ((\sim S_0)\&(S_1)\&I_1) \mid ((S_0)\&(\sim S_1)\&I_2) \mid ((S_0)\&(S_1)\&I_3)$$

1.5 BOOLEAN EXPRESSION:

$$Y = \overline{S_0}\,\overline{S_1}I_0 + \overline{S_0}S_1I_1 + S_0\overline{S_1}I_2 + S_0S_1I_3$$

1.6 TRUTH TABLE:

INPUT		**OUTPUT**
S_0	**S_1**	**Y**
0	**0**	**I_0**
0	**1**	**I_1**
1	**0**	**I_2**
1	**1**	**I_3**

1.7 VERILOG CODE (MUX_4by1_sf.v):

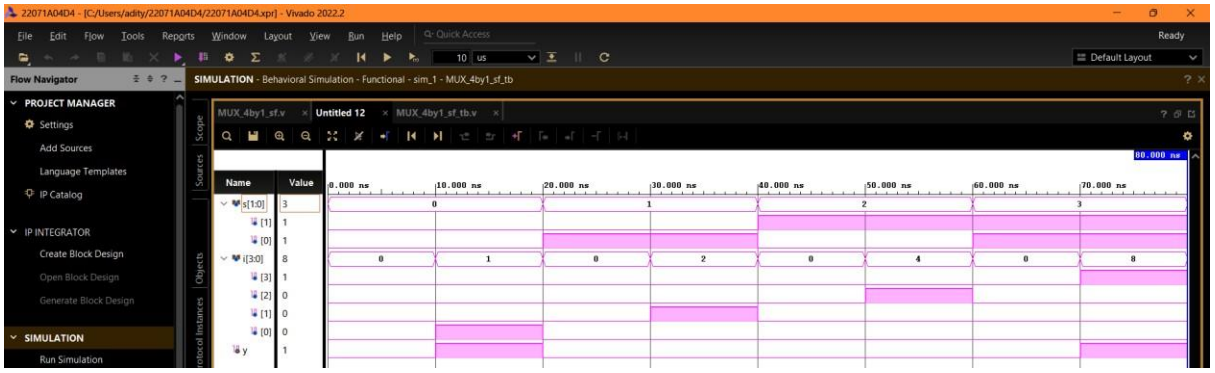
```
module MUX_4by1_sf(y,s,i);
output y;
input [1:0]s;
input[3:0]i;

wire a1,a2,b1,b2,c1,c2,d1,d2,o1,o2;

and (a1,~s[0],~s[1]);
and (a2,a1,i[0]);

and (b1,~s[0],s[1]);
```

```
and (b2,b1,i[1]);

and (c1,s[0],~s[1]);
and (c2,c1,i[2]);

and (d1,s[0],s[1]);
and (d2,d1,i[3]);

or (o1,a2,b2);

or (o2.c2,d2);

or (y,o1,o2);

endmodule
```

1.8 TEST BENCH (MUX_4by1_sf_tb.v):

```
module MUX_4by1_sf_tb();
reg [1:0]s;
reg[3:0]i;
wire y;

MUX_4by1_sf x1(y,s,i);

initial
begin
s  =  2'b00;i=4'b0000;
#10 s = 2'b00;i=4'b0001;

#10 s = 2'b01;i=4'b0000;
#10 s = 2'b01;i=4'b0010;

#10 s = 2'b10;i=4'b0000;
#10 s = 2'b10;i=4'b0100;

#10 s = 2'b11;i=4'b0000;
#10 s = 2'b11;i=4'b1000;

#10 $finish;

end
endmodule
```

1.9 WAVEFORM:



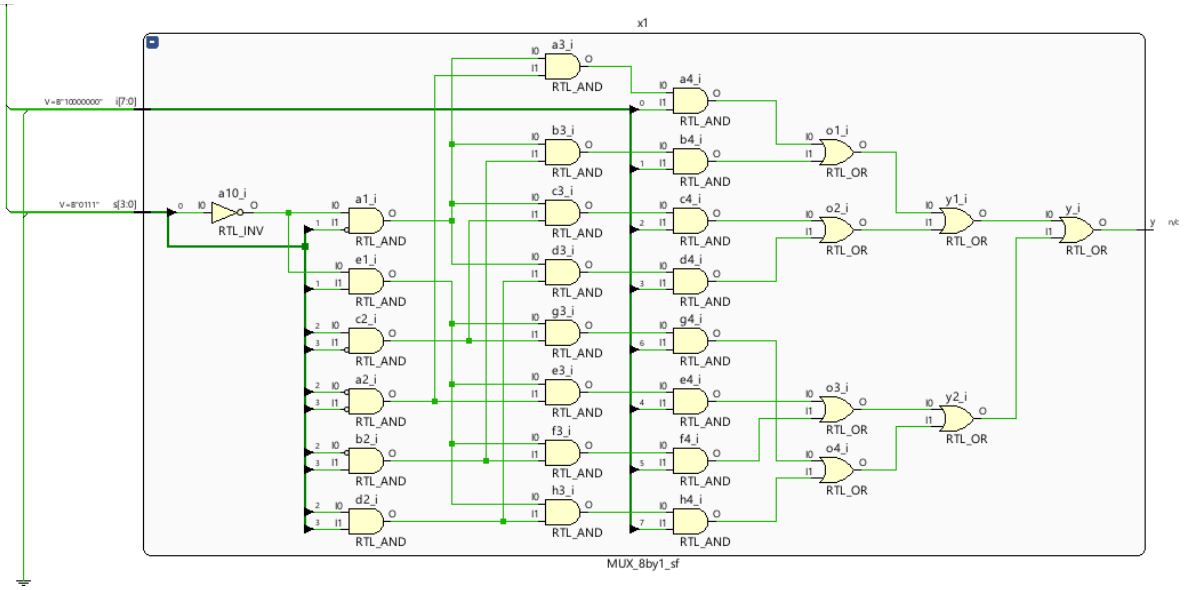
1.10 RESULT:

4x1 Multiplexer is simulated and implemented in Structural Flow Modeling.

1.1 AIM: 8x1 Multiplexer [MUX]

1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 SYMBOL:



1.4 LOGIC EXPRESSION:

$$Y = ((\sim S_0) \,\&\, (\sim S_1) \,\&\, (\sim S_2) \,\&\, (\sim S_3) \,\&\, I_0) \,|$$
$$((\sim S_0) \,\&\, (\sim S_1) \,\&\, (\sim S_2) \,\&\, (S_3) \,\&\, I_1) \,|$$
$$((\sim S_0) \,\&\, (\sim S_1) \,\&\, (S_2) \,\&\, (\sim S_3) \,\&I_2) \,|$$
$$((\sim S_0) \,\&\, (\sim S_1) \,\&\, (S_2) \,\&\, (S_3) \,\&I_3) \,|$$
$$((\sim S_0) \,\&\, (S_1) \,\&\, (\sim S_2) \,\&\, (\sim S_3) \,\&I_4) \,|$$
$$((\sim S_0) \,\&\, (S_1) \,\&\, (\sim S_2) \,\&\, (S_3) \,\&I_5) \,|$$
$$((\sim S_0) \,\&\, (S_1) \,\&\, (S_2) \,\&\, (\sim S_3) \,\&I_6) \,|$$
$$((\sim S_0) \,\&\, (S_1) \,\&\, (S_2) \,\&\, (S_3) \,\&I_7)$$

1.5 BOOLEAN EXPRESSION:

$$Y = \overline{S_0}S_1\overline{S_2}\overline{S_3}I_0 + \overline{S_0}\overline{S_1}\overline{S_2}S_3I_1 + \overline{S_0}\overline{S_1}S_2\overline{S_3}I_2 + \overline{S_0}\overline{S_1}S_2S_3I_3 + \overline{S_0}S_1\overline{S_2}\overline{S_3}I_4 + \overline{S_0}S_1\overline{S_2}S_3I_5 + \overline{S_0}S_1S_2\overline{S_3}I_6 + \overline{S_0}S_1S_2S_3I_7$$

1.6 TRUTH TABLE:

INPUT				**OUTPUT**
S_0	**S_1**	**S_2**	**S_3**	**I**
0	**0**	**0**	**0**	**I_0**
0	**0**	**0**	**1**	**I_1**
0	**0**	**1**	**0**	**I_2**
0	**0**	**1**	**1**	**I_3**
0	**1**	**0**	**0**	**I_4**
0	**1**	**0**	**1**	**I_5**
0	**1**	**1**	**0**	**I_6**
0	**1**	**1**	**1**	**I_7**

1.7 VERILOG CODE (MUX_8by1_sf.v):

```
module MUX_8by1_sf(y,s,i);
output y;
input [3:0]s;
input [7:0]i;

wire a1,a2,a3,a4;
wire b1,b2,b3,b4;
wire c1,c2,c3,c4;
wire d1,d2,d3,d4;
wire e1,e2,e3,e4;
wire f1,f2,f3,f4;
wire g1,g2,g3,g4;
wire h1,h2,h3,h4;
wire o1,o2,o3,o4;
wire y1,y2;

and (a1,~s[0],~s[1]);
and (a2,~s[2],~s[3]);
and(a3,a1,a2);
and(a4,a3,i[0]);

and (b1,~s[0],~s[1]);
and (b2,~s[2],s[3]);
and (b3,b1,b2);
and (b4,b3,i[1]);

and (c1,~s[0],~s[1]);
and (c2,s[2],~s[3]);
and (c3,c1,c2);
and (c4,c3,i[2]);

and (d1,~s[0],~s[1]);
and (d2,s[2],s[3]);
and (d3,d1,d2);
and (d4,d3,i[3]);

and (e1,~s[0],s[1]);
and (e2,~s[2],~s[3]);
and (e3,e1,e2);
and (e4,e3,i[4]);

and (f1,~s[0],s[1]);
and (f2,~s[2],s[3]);
and (f3,f1,f2);
and (f4,f3,i[5]);

and (g1,~s[0],s[1]);
```

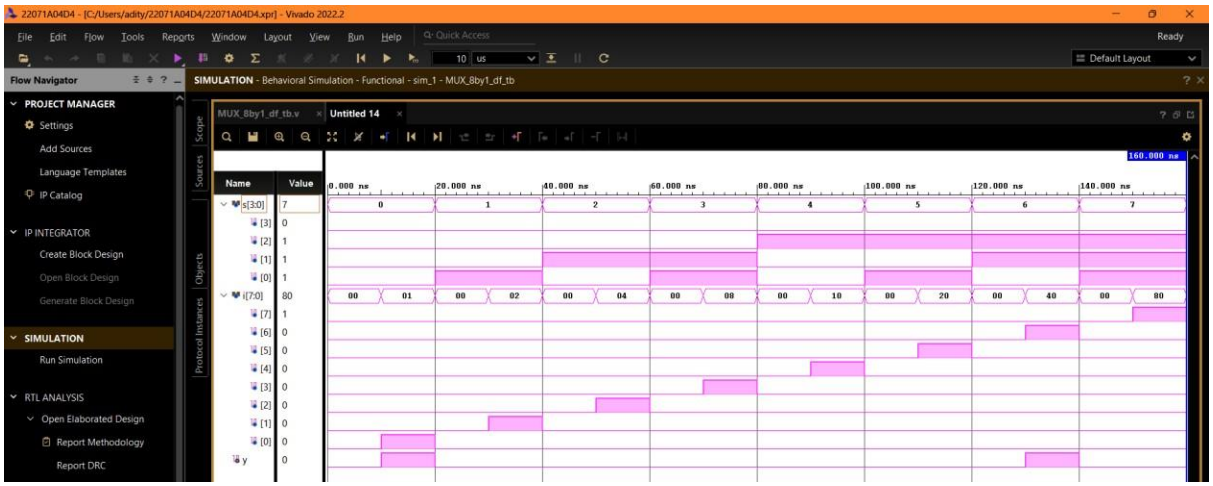
```
and (g2,s[2],~s[3]);
and (g3,g1,g2);
and (g4,g3,i[6]);

and (h1,~s[0],s[1]);
and (h2,s[2],s[3]);
and (h3,h1,h2);
and (h4,h3,i[7]);

or (o1,a4,b4);
or (o2,c4,d4);
or (o3,e4,f4);
or (o4,g4,h4);

or(y1,o1,o2);
or(y2,o3,o4);

or(y,y1,y2);

endmodule
```

1.8 TEST BENCH (MUX_8by1_sf_tb.v):

```
module MUX_8by1_sf_tb();
reg [3:0]s;
reg[7:0]i;
wire y;

MUX_8by1_sf x1(y,s,i);

initial
begin
s  =  4'b0000;i=8'b00000000;
#10 s = 4'b0000;i=8'b00000001;

#10 s = 4'b0001;i=8'b00000000;
#10 s = 4'b0001;i=8'b00000010;

#10 s = 4'b0010;i=8'b00000000;
#10 s = 4'b0010;i=8'b00000100;

#10 s = 4'b0011;i=8'b00000000;
#10 s = 4'b0011;i=8'b00001000;

#10 s = 4'b0100;i=8'b00000000;
#10 s = 4'b0100;i=8'b00010000;

#10 s = 4'b0101;i=8'b00000000;
#10 s = 4'b0101;i=8'b00100000;
```

```
#10 s = 4'b0110;i=8'b00000000;
#10 s = 4'b0110;i=8'b01000000;

#10 s = 4'b0111;i=8'b00000000;
#10 s = 4'b0111;i=8'b10000000;


#10 $finish;

end
endmodule
```

1.9 WAVEFORM:



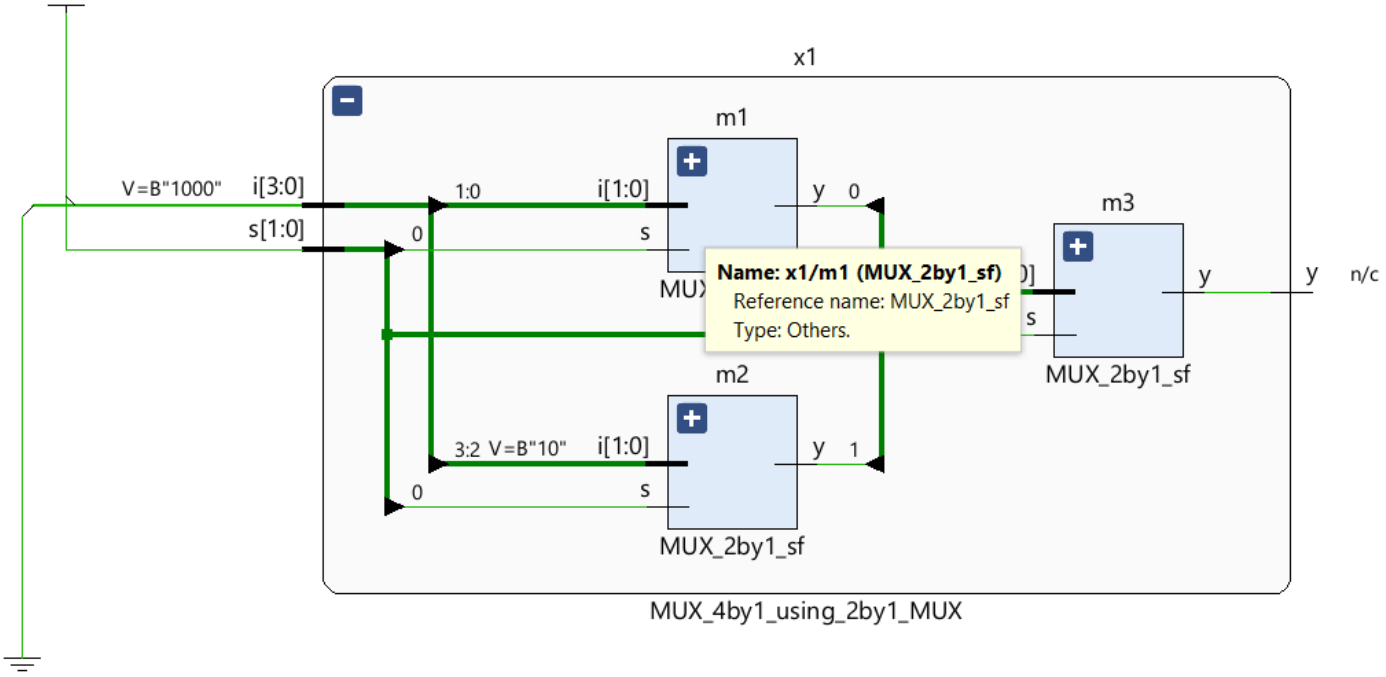
1.10 RESULT:

8x1 Multiplexer is simulated and implemented in Structural Flow Modeling.

1.1 AIM: 4x1 MUX using 2x1 MUX

1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 SYMBOL:



1.4 LOGIC EXPRESSION:

$$Y = ((\sim S_0)\&(\sim S_1)\&I_0) \mid ((\sim S_0)\&(S_1)\&I_1) \mid ((S_0)\&(\sim S_1)\&I_2) \mid ((S_0)\&(S_1)\&I_3)$$

1.5 BOOLEAN EXPRESSION:

$$Y = \overline{S_0}\,\overline{S_1}I_0 + \overline{S_0}S_1I_1 + S_0\overline{S_1}I_2 + S_0S_1I_3$$

1.6 TRUTH TABLE:

INPUT		OUTPUT
S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

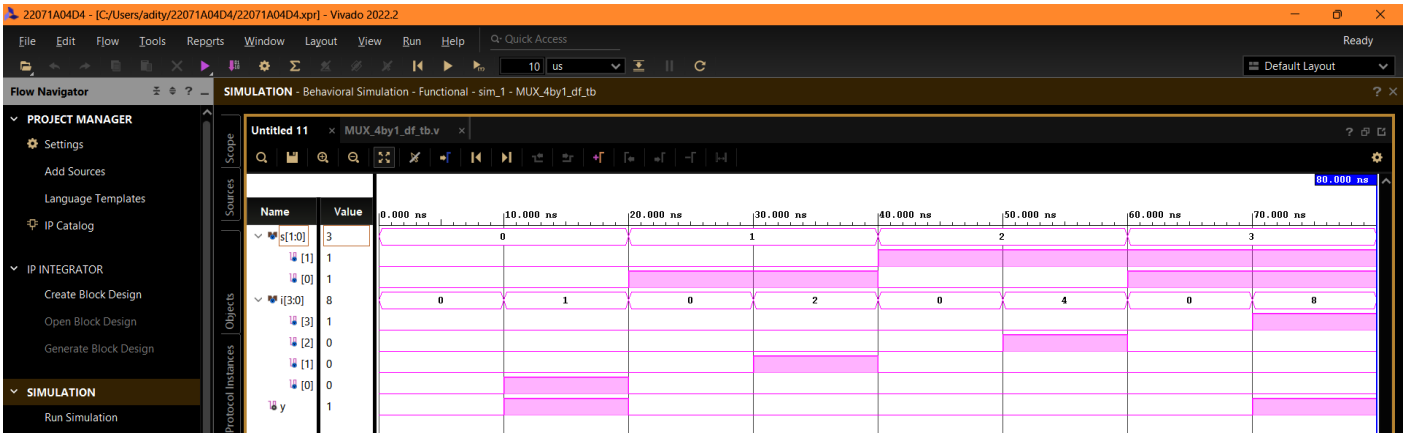
1.7 VERILOG CODE (MUX_4by1_using_2by1_MUX.v):

```
module MUX_4by1_using_2by1_MUX(y,s,i);
input [3:0]i;
input [1:0]s;
output y;
wire [1:0]w;

MUX_2by1_sf m1(w[0],s[0],i[1:0]);
MUX_2by1_sf m2(w[1],s[0],i[3:2]);
MUX_2by1_sf m3(y,s[1],w[1:0]);
endmodule
```

1.8 TEST BENCH (MUX_4by1_using_2by1_MUX _tb.v):

```
module MUX_4by1_using_2by1_MUX_tb();
reg [1:0]s;
reg[3:0]i;
wire y;

MUX_4by1_using_2by1_MUX x1(y,s,i);

initial
begin
s = 2'b00;i=4'b0000;
#10 s = 2'b00;i=4'b0001;

#10 s = 2'b01;i=4'b0000;
#10 s = 2'b01;i=4'b0010;

#10 s = 2'b10;i=4'b0000;
#10 s = 2'b10;i=4'b0100;

#10 s = 2'b11;i=4'b0000;
#10 s = 2'b11;i=4'b1000;

#10 $finish;

end
endmodule
```

1.9 WAVEFORM:



1.10 RESULT:

4x1 Multiplexer is simulated and implemented using 2x1 Multiplexer.