3

EXPERIMENT-3
REALIZATION OF CODE CONVERTERS

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

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

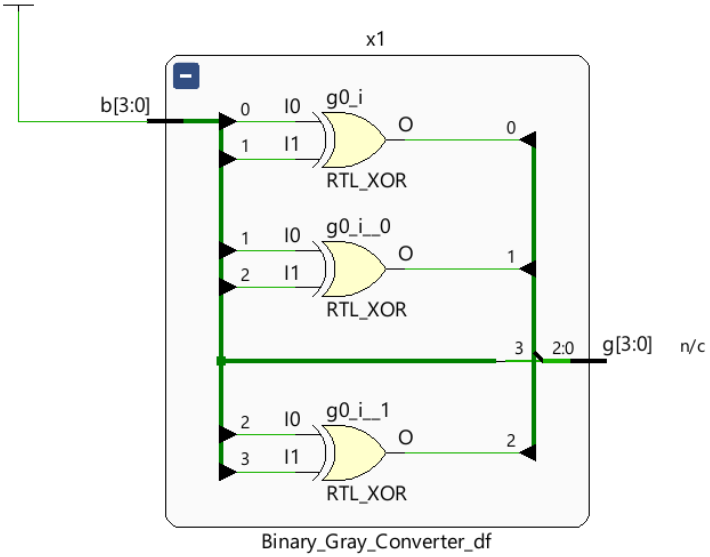
3. **PROCEDURE:**
    - Open the Xilinx Vivado project navigator.
    - Open the Design source and go to add / create sources
    - Create new file, give appropriate name save it.
    - Open the file in the editor and write the Verilog code.
    - Open the Design source and go to add / create sources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

Code Converters (Data Flow Modeling)

1.1 AIM: Binary to Gray Code Conversion

1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 SYMBOL:



1.4 LOGIC EXPRESSION:

$G_3 = B_3$

$G_2 = B_3 \wedge B_2$

$G_1 = B_2 \wedge B_1$

$G_0 = B_1 \wedge B_0$

1.5 BOOLEAN EXPRESSION:

$G_3 = B_3$

$G_2 = B_3 \oplus B_2$

$G_1 = B_2 \oplus B_1$

$G_0 = B_1 \oplus B_0$

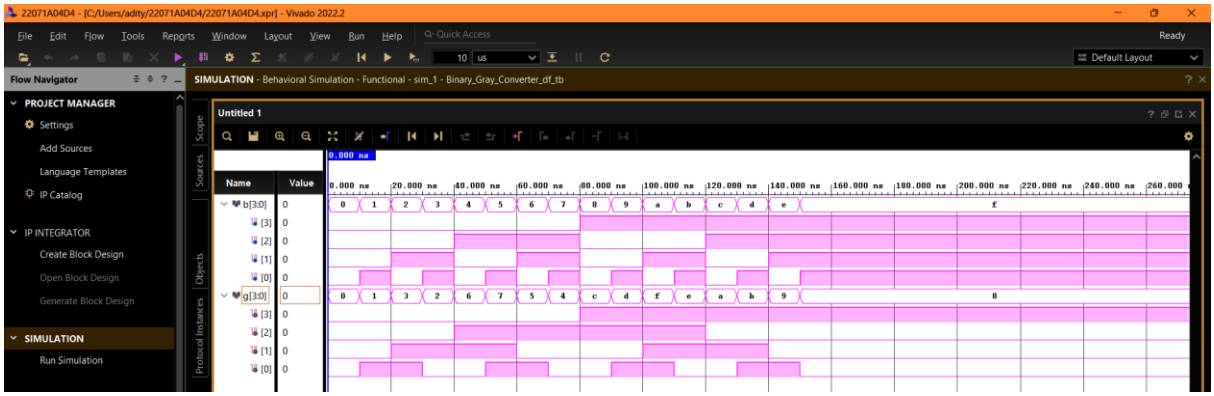
1.6 VERILOG CODE (Binary_Gray_Converter_df.v):

```
module Binary_Gray_Converter_df(g,b);
output [3:0]g;
input [3:0]b;

assign g[3] = b[3];
assign g[2] = b[2]^b[3];
assign g[1] = b[1]^b[2];
assign g[0] = b[0]^b[1];
endmodule
```

1.7 TEST BENCH (Binary_Gray_Converter_df_tb.v):

```
module Binary_Gray_Converter_df_tb();
reg [3:0]b;
wire [3:0]g;
Binary_Gray_Converter_df x1(g,b);
initial
begin
{b[3],b[2],b[1],b[0]}=4'b0000;
#10 {b[3],b[2],b[1],b[0]}=4'b0001;
#10 {b[3],b[2],b[1],b[0]}=4'b0010;
#10 {b[3],b[2],b[1],b[0]}=4'b0011;
#10 {b[3],b[2],b[1],b[0]}=4'b0100;
#10 {b[3],b[2],b[1],b[0]}=4'b0101;
#10 {b[3],b[2],b[1],b[0]}=4'b0110;
#10 {b[3],b[2],b[1],b[0]}=4'b0111;
#10 {b[3],b[2],b[1],b[0]}=4'b1000;
#10 {b[3],b[2],b[1],b[0]}=4'b1001;
#10 {b[3],b[2],b[1],b[0]}=4'b1010;
#10 {b[3],b[2],b[1],b[0]}=4'b1011;
#10 {b[3],b[2],b[1],b[0]}=4'b1100;
#10 {b[3],b[2],b[1],b[0]}=4'b1101;
#10 {b[3],b[2],b[1],b[0]}=4'b1110;
#10 {b[3],b[2],b[1],b[0]}=4'b1111;

end
endmodule
```

1.8 WAVEFORM:



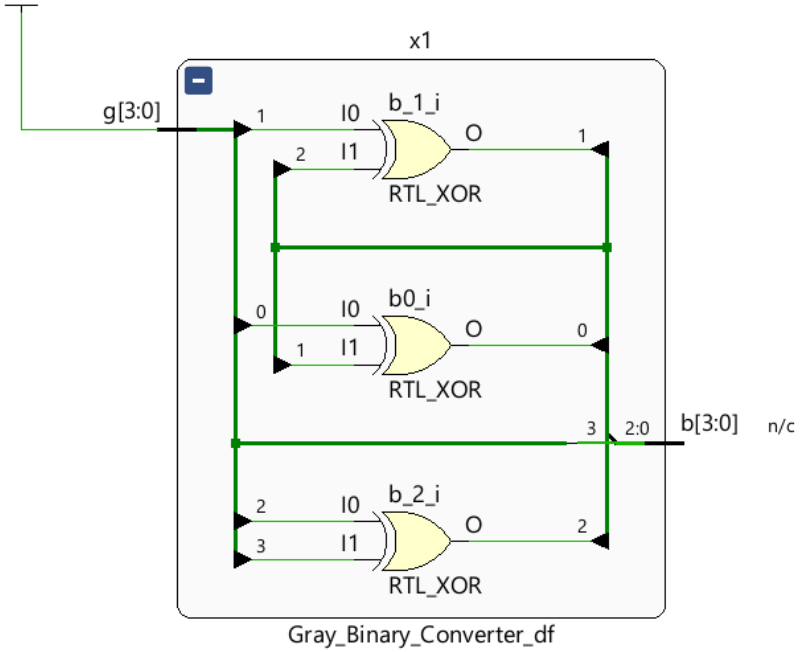
1.9 RESULT:

Binary to Gray Code Conversion is simulated and implemented in Data Flow Modeling.

2.1 AIM: Gray to Binary Code Conversion

2.2 SOFTWARE USED: Xilinx Vivado 2022.2

2.3 SYMBOL:



2.4 LOGIC EXPRESSION:

$B_3 = G_3$

$B_2 = B_3 \wedge G_2$

$B_1 = B_2 \wedge G_1$

$B_0 = B_1 \wedge G_0$

2.5 BOOLEAN EXPRESSION:

$B_3 = G_3$

$B_2 = B_3 \oplus G_2$

$B_1 = B_2 \oplus G_1$

$B_0 = B_1 \oplus G_0$

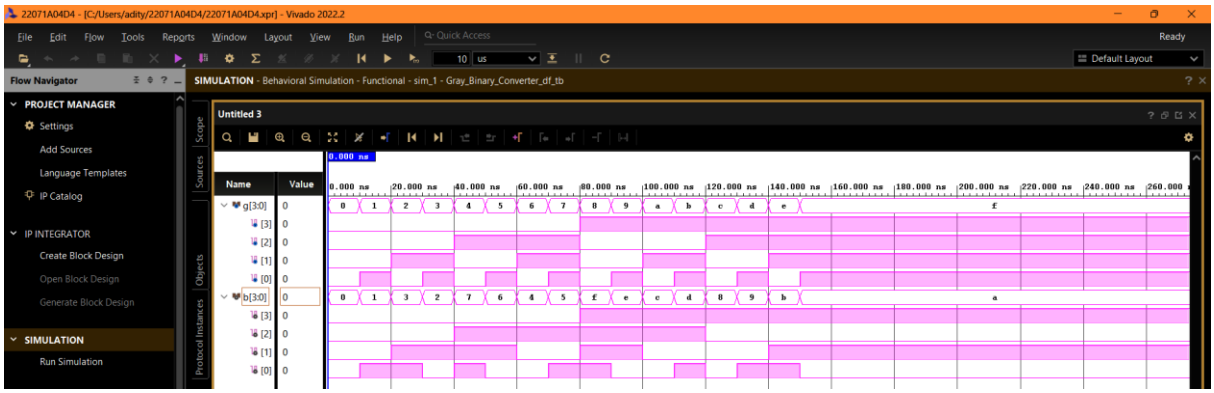
2.6 VERILOG CODE (Gray_Binary_Converter_df.v):

```
module Gray_Binary_Converter_df(b,g);
output [3:0]b;
input [3:0]g;

assign b[3] = g[3];
assign b[2] = g[2]^b[3];
assign b[1] = g[1]^b[2];
assign b[0] = g[0]^b[1];
endmodule
```

2.7 TEST BENCH (Gray_Binary_Converter_df_tb.v):

```
module Gray_Binary_Converter_df_tb();
reg [3:0]g;
wire [3:0]b;
Gray_Binary_Converter_df x1(b,g);
initial
begin
{g[3],g[2],g[1],g[0]}=4'b0000;
#10 {g[3],g[2],g[1],g[0]}=4'b0001;
#10 {g[3],g[2],g[1],g[0]}=4'b0010;
#10 {g[3],g[2],g[1],g[0]}=4'b0011;
#10 {g[3],g[2],g[1],g[0]}=4'b0100;
#10 {g[3],g[2],g[1],g[0]}=4'b0101;
#10 {g[3],g[2],g[1],g[0]}=4'b0110;
#10 {g[3],g[2],g[1],g[0]}=4'b0111;
#10 {g[3],g[2],g[1],g[0]}=4'b1000;
#10 {g[3],g[2],g[1],g[0]}=4'b1001;
#10 {g[3],g[2],g[1],g[0]}=4'b1010;
#10 {g[3],g[2],g[1],g[0]}=4'b1011;
#10 {g[3],g[2],g[1],g[0]}=4'b1100;
#10 {g[3],g[2],g[1],g[0]}=4'b1101;
#10 {g[3],g[2],g[1],g[0]}=4'b1110;
#10 {g[3],g[2],g[1],g[0]}=4'b1111;

end
endmodule
```

2.8 WAVEFORM:



2.9 RESULT:

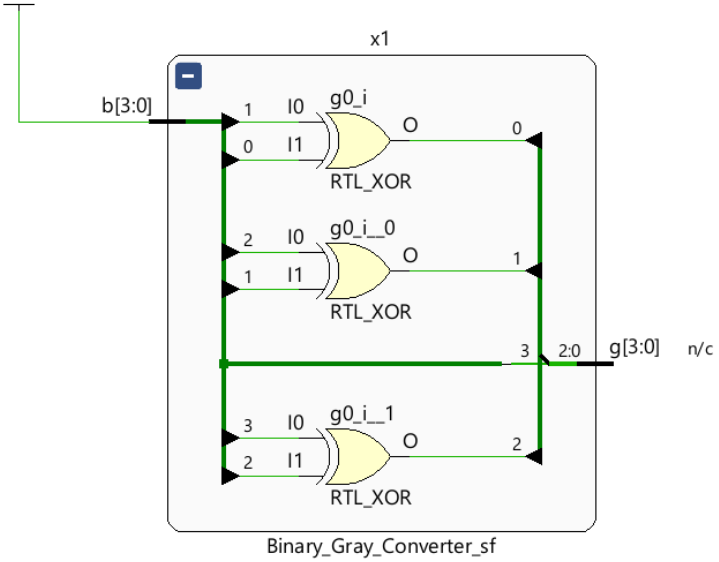
Gray to Binary Code Conversion is simulated and implemented in Data Flow Modeling.

Code Converters (Structural Flow Modeling)

1.1 AIM: Binary to Gray Code Conversion

1.2 SOFTWARE USED: Xilinx Vivado 2022.2

1.3 SYMBOL:



1.4 LOGIC EXPRESSION:

$G_3 = B_3$

$G_2 = B_3 \wedge B_2$

$G_1 = B_2 \wedge B_1$

$G_0 = B_1 \wedge B_0$

1.5 BOOLEAN EXPRESSION:

$G_3 = B_3$

$G_2 = B_3 \oplus B_2$

$G_1 = B_2 \oplus B_1$

$G_0 = B_1 \oplus B_0$

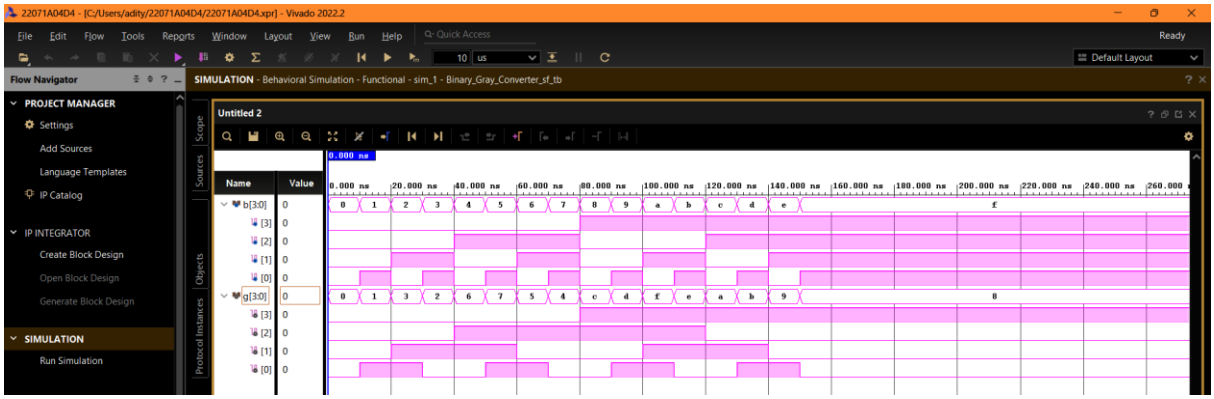
1.6 VERILOG CODE (Binary_Gray_Converter_sf.v):

```
module Binary_Gray_Converter_sf(g,b);
output [3:0]g;
input [3:0]b;

assign g[3] = b[3];
xor(g[2],b[3],b[2]);
xor(g[1],b[2],b[1]);
xor(g[0],b[1],b[0]);

endmodule
```

1.7 TEST BENCH (Binary_Gray_Converter_sf_tb.v):

```
module Binary_Gray_Converter_sf_tb();
reg [3:0]b;
wire [3:0]g;
Binary_Gray_Converter_sf x1(g,b);
initial
begin
{b[3],b[2],b[1],b[0]}=4'b0000;
#10 {b[3],b[2],b[1],b[0]}=4'b0001;
#10 {b[3],b[2],b[1],b[0]}=4'b0010;
#10 {b[3],b[2],b[1],b[0]}=4'b0011;
#10 {b[3],b[2],b[1],b[0]}=4'b0100;
#10 {b[3],b[2],b[1],b[0]}=4'b0101;
#10 {b[3],b[2],b[1],b[0]}=4'b0110;
#10 {b[3],b[2],b[1],b[0]}=4'b0111;
#10 {b[3],b[2],b[1],b[0]}=4'b1000;
#10 {b[3],b[2],b[1],b[0]}=4'b1001;
#10 {b[3],b[2],b[1],b[0]}=4'b1010;
#10 {b[3],b[2],b[1],b[0]}=4'b1011;
#10 {b[3],b[2],b[1],b[0]}=4'b1100;
#10 {b[3],b[2],b[1],b[0]}=4'b1101;
#10 {b[3],b[2],b[1],b[0]}=4'b1110;
#10 {b[3],b[2],b[1],b[0]}=4'b1111;

end
endmodule
```

1.8 WAVEFORM:



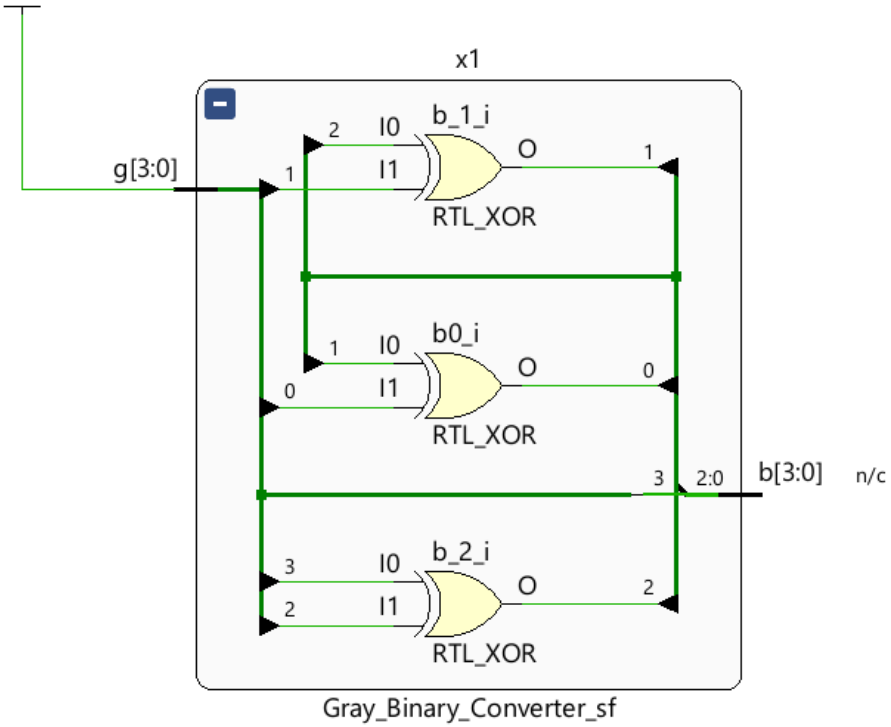
1.9 RESULT:

Binary to Gray Code Conversion is simulated and implemented in Structural Flow Modeling.

2.1 AIM: Gray to Binary Code Conversion

2.2 SOFTWARE USED: Xilinx Vivado 2022.2

2.3 SYMBOL:



2.4 LOGIC EXPRESSION:

$B_3 = G_3$

$B_2 = B_3 \wedge G_2$

$B_1 = B_2 \wedge G_1$

$B_0 = B_1 \wedge G_0$

2.5 BOOLEAN EXPRESSION:

$B_3 = G_3$

$B_2 = B_3 \oplus G_2$

$B_1 = B_2 \oplus G_1$

$B_0 = B_1 \oplus G_0$

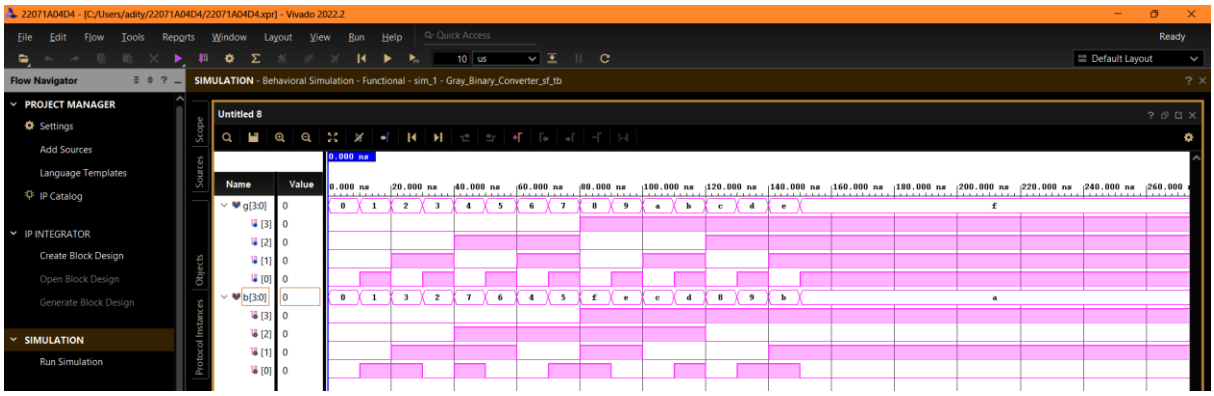
2.6 VERILOG CODE (Gray_Binary_Converter_sf.v):

```
module Gray_Binary_Converter_sf(b,g);
output [3:0]b;
input [3:0]g;

assign b[3] = g[3];
xor(b[2],b[3],g[2]);
xor(b[1],b[2],g[1]);
xor(b[0],b[1],g[0]);
endmodule
```

2.7 TEST BENCH (Gray_Binary_Converter_sf_tb.v):

```
module Gray_Binary_Converter_sf_tb();
reg [3:0]g;
wire [3:0]b;
Gray_Binary_Converter_sf x1(b,g);
initial
begin
{g[3],g[2],g[1],g[0]}=4'b0000;
#10 {g[3],g[2],g[1],g[0]}=4'b0001;
#10 {g[3],g[2],g[1],g[0]}=4'b0010;
#10 {g[3],g[2],g[1],g[0]}=4'b0011;
#10 {g[3],g[2],g[1],g[0]}=4'b0100;
#10 {g[3],g[2],g[1],g[0]}=4'b0101;
#10 {g[3],g[2],g[1],g[0]}=4'b0110;
#10 {g[3],g[2],g[1],g[0]}=4'b0111;
#10 {g[3],g[2],g[1],g[0]}=4'b1000;
#10 {g[3],g[2],g[1],g[0]}=4'b1001;
#10 {g[3],g[2],g[1],g[0]}=4'b1010;
#10 {g[3],g[2],g[1],g[0]}=4'b1011;
#10 {g[3],g[2],g[1],g[0]}=4'b1100;
#10 {g[3],g[2],g[1],g[0]}=4'b1101;
#10 {g[3],g[2],g[1],g[0]}=4'b1110;
#10 {g[3],g[2],g[1],g[0]}=4'b1111;

end
endmodule
```

2.8 WAVEFORM:



2.9 RESULT:

Gray to Binary Code Conversion is simulated and implemented in Structural Flow Modeling.