TECY - Projekt 2

Małgorzata Pszczółkowska - 311423
Anastasiya Ronskaya - 317058
Konrad Kotlicki - 310958
Sebastian Skrzek - 311442

2 kwietnia 2021

Spis treści

1 Funkcja

Funkcja F(x5,x4,x3,x2,x1,x0) podana jest w postaci tabeli:

x5x4x3x2x1x0	y1y0
100100	00
101110	01
110111	01
101100	00
001110	10
110100	10
100111	10
001011	11
001010	11

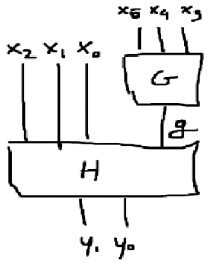
2 Dekompozycja 1

Istnieje dekompozycja F = H(x2,x1,x0), G(x5,x4,x3)
Dekompozycja jest dla G=(x5,x4,x3) i funkcja G ma postać:

x5x4x3	g
000	-
001	1
010	-
011	-
100	0
101	0
110	1
111	-

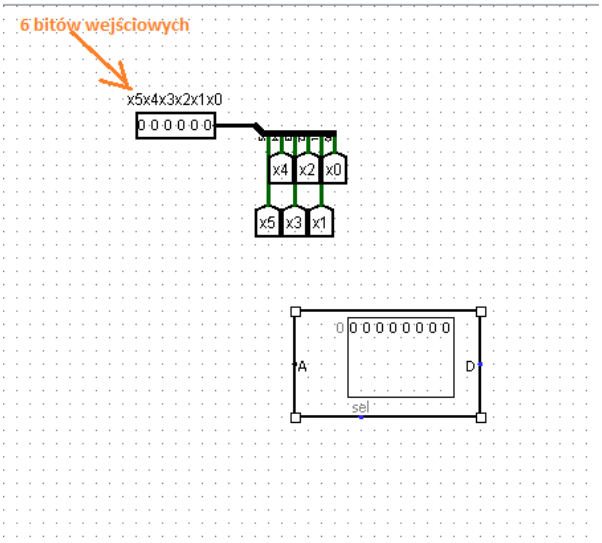
Funkcja H ma postać H =(x2,x1,x0,g):

x2x1x0/g	0	1
000	–	–
001	–	–
010	–	11
011	–	11
100	00	10
101	–	–
110	01	10
111	10	01

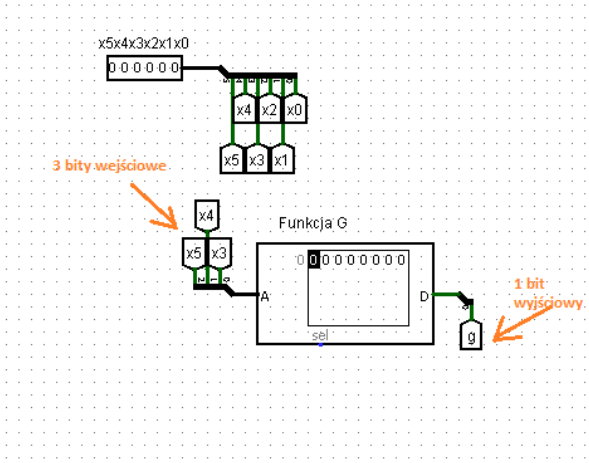


2.1 Realizacja w Logisim

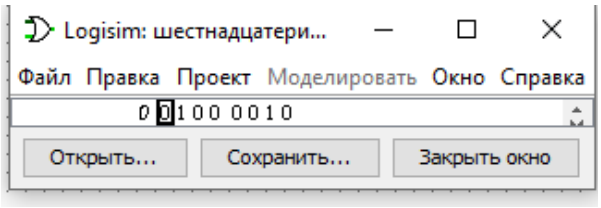
Funkcja F ma 6 bitów wejściowych i 2 wyjściowe



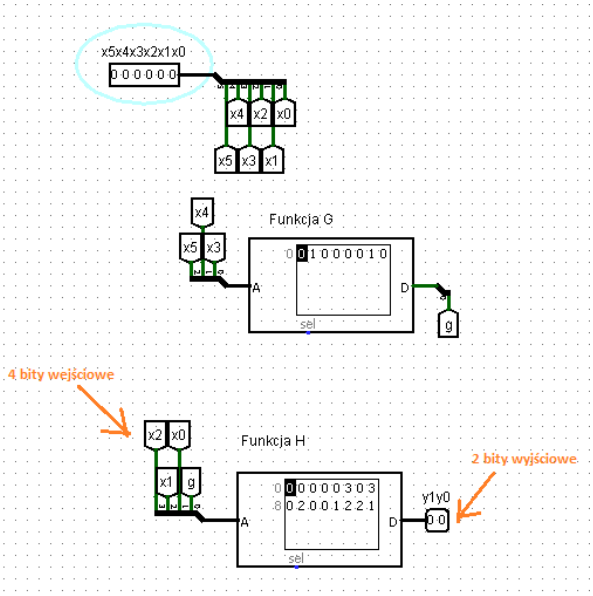
Funkcja G ma 3 bity wejściowe i 1 bit wyjściowy



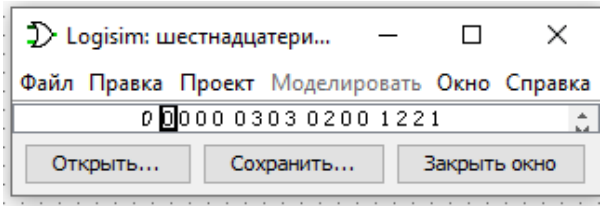
Zawartość pamięci ROM dla funkcji G



Funkcja H ma 4 bity wejściowe i 2 wyjściowe

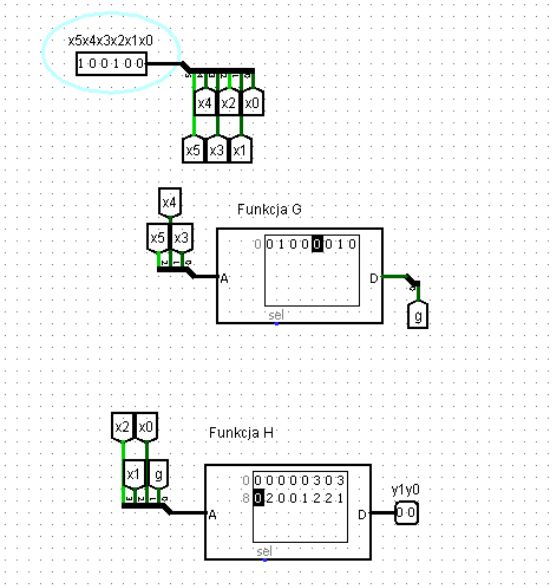


Zawartość pamięci ROM dla funkcji H

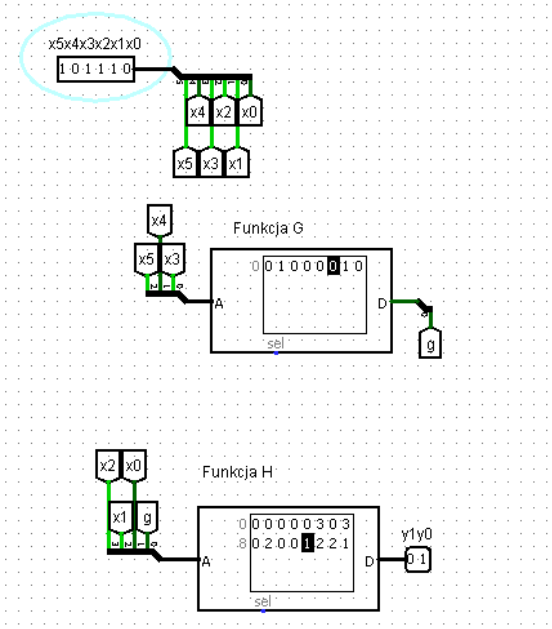


TESTY

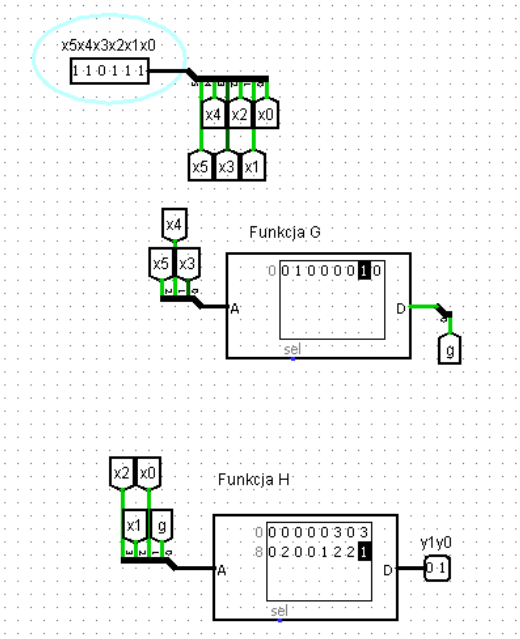
x5x4x3x2x1x1=100100



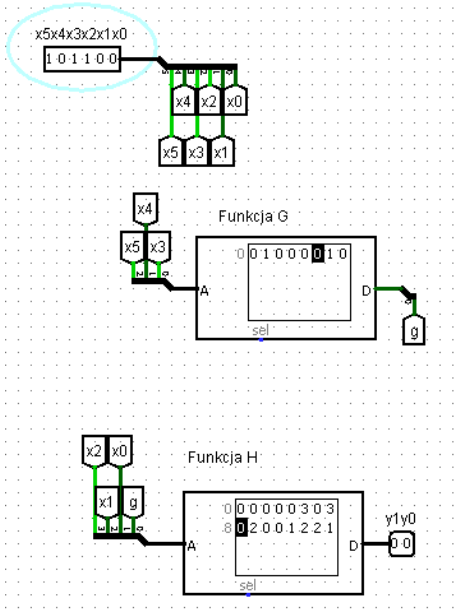
x5x4x3x2x1x1=101110



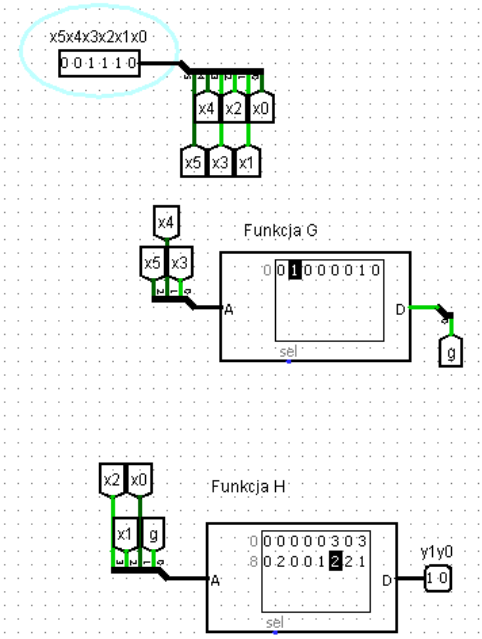
x5x4x3x2x1x1=110111



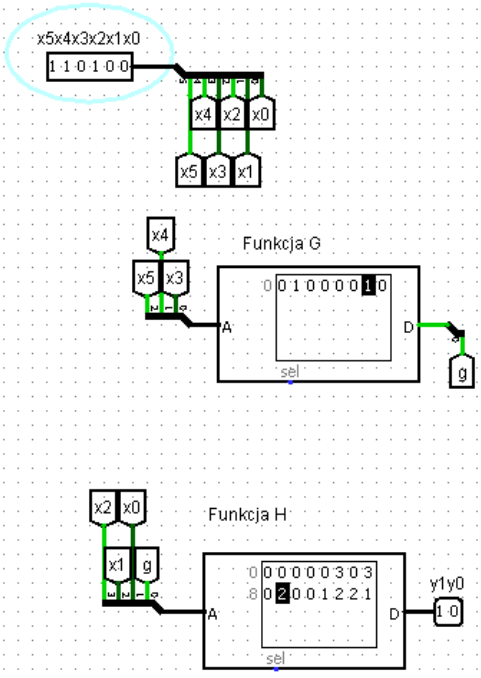
x5x4x3x2x1x1=101100



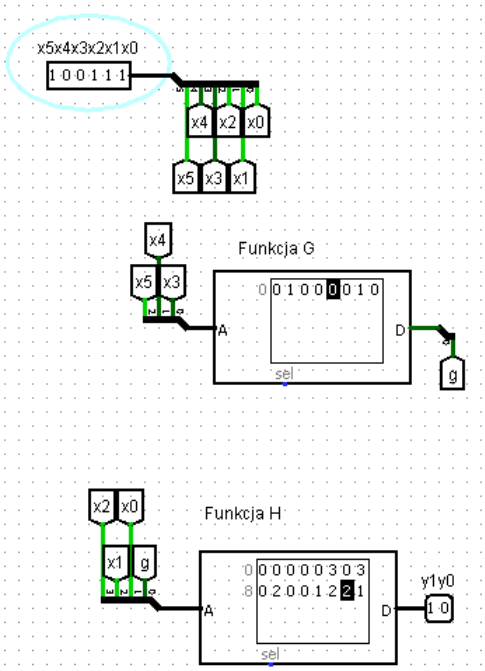
x5x4x3x2x1x1=001110



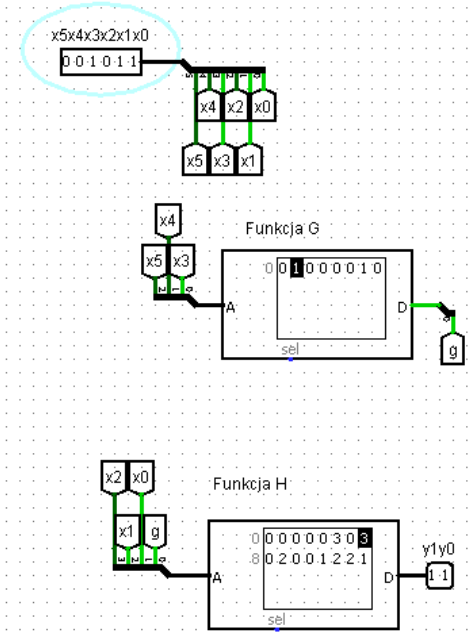
x5x4x3x2x1x1=110100



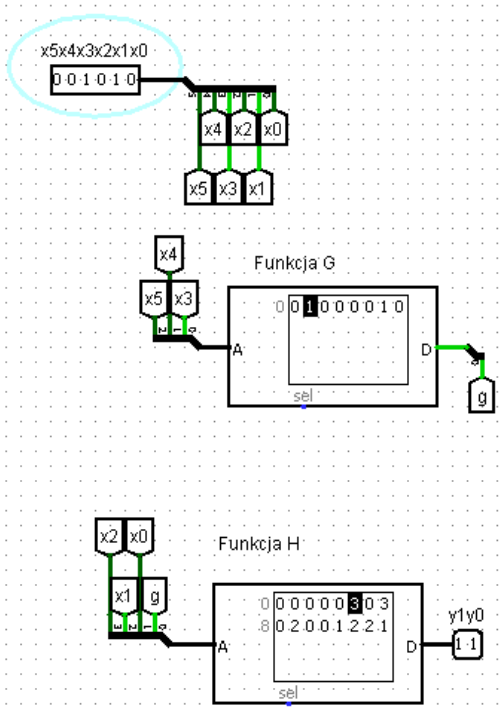
x5x4x3x2x1x1=100111



x5x4x3x2x1x1=001011



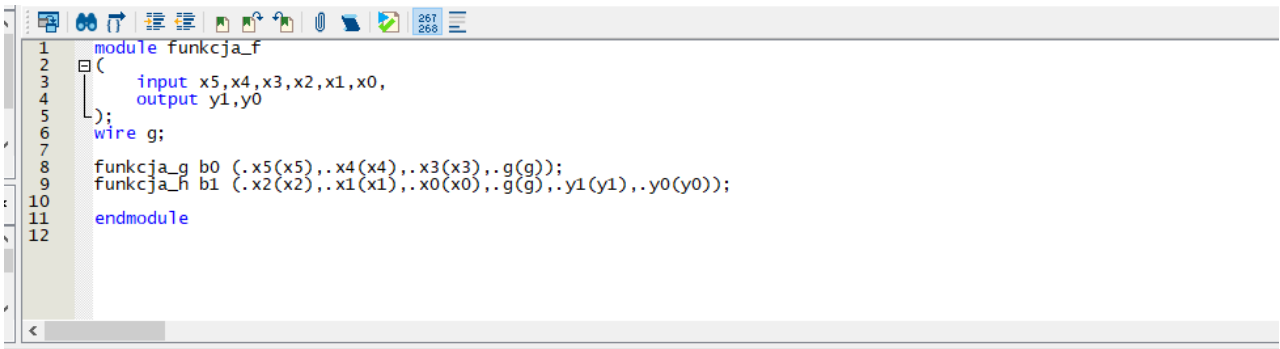
x5x4x3x2x1x1=001010



2.2 Realizacja w Quartus

Tworzymy plik Verilog HDL File o nazwie funkcja_f. Zakodujemy funkcję F. Zawartość pliku funkcja_f:

```
1  module funkcja_f
2  (
3      input x5,x4,x3,x2,x1,x0,
4       output y1,y0
5  );
6  wire g;
7
8  funkcja_g b0 (.x5(x5),.x4(x4),.x3(x3),.g(g));
9  funkcja_h b1 (.x2(x2),.x1(x1),.x0(x0),.g(g),.y1(y1),.y0(y0));
10
11 endmodule
```



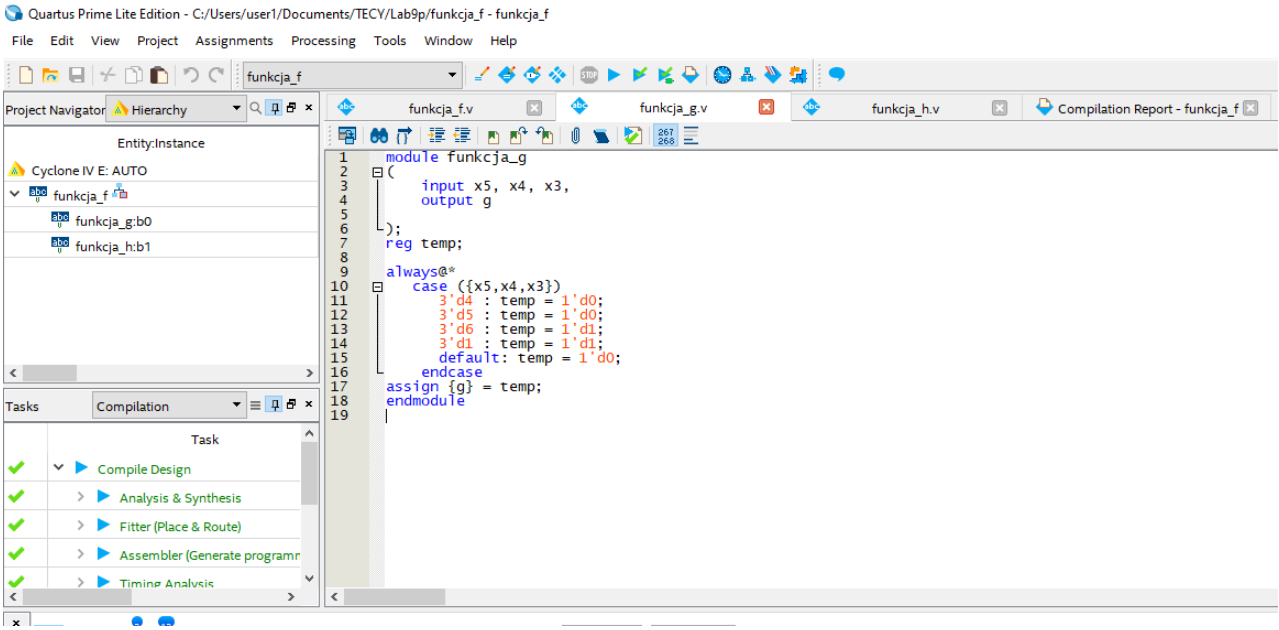
```
module funkcja_f
(
    input x5,x4,x3,x2,x1,x0,
    output y1,y0
);
wire g;

funkcja_g b0 (.x5(x5),.x4(x4),.x3(x3),.g(g));
funkcja_h b1 (.x2(x2),.x1(x1),.x0(x0),.g(g),.y1(y1),.y0(y0));

endmodule
```

Find... Find Next

```
**************************************************
ime EDA Netlist Writer
da --read_settings_files=off --write_settings_files=off funkcja_f -c funkcja_f
rs has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSO
kcja_f.vo in folder "C:/Users/user1/Documents/TECY/Lab9p/simulation/modelsim/" for EDA simulation tool
Netlist Writer was successful. 0 errors, 1 warning
 Compilation was successful. 0 errors, 14 warnings
**************************************************
ime Netlist Viewers Preprocess
pp funkcja_f -c funkcja_f --netlist_type=sgate
```

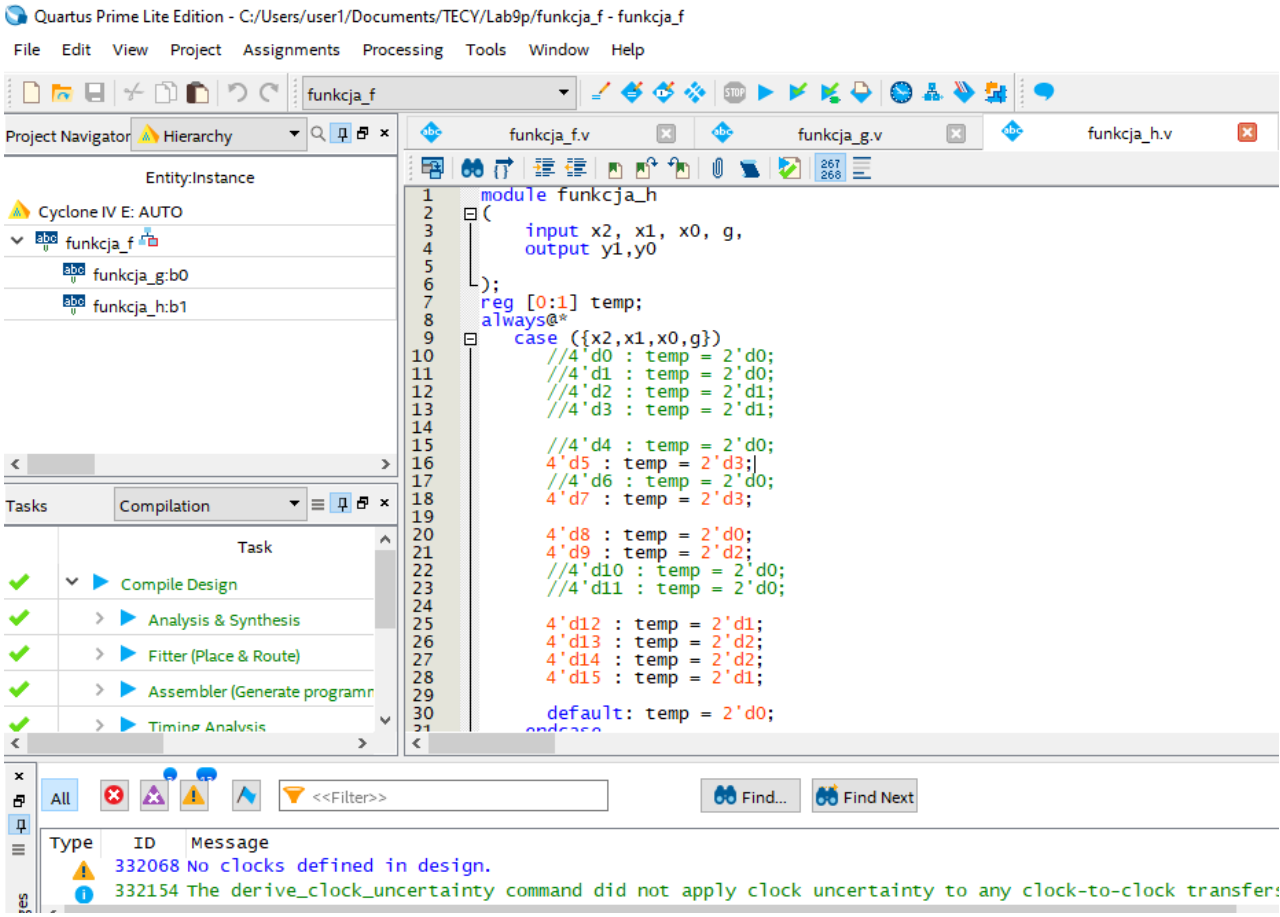
Tworzymy nowy plik Verilog HDL File o nazwie funkcja_g. Zakodujemy funkcję G.
Zawartość pliku funkcja_g:

```
module funkcja_g
(
    input x5, x4, x3,
     output g

);
reg temp;

always@*
    case ({x5,x4,x3})
        3'd4 : temp = 1'd0;
        3'd5 : temp = 1'd0;
        3'd6 : temp = 1'd1;
        3'd1 : temp = 1'd1;
       default: temp = 1'd0;
    endcase
assign {g} = temp;
endmodule
```

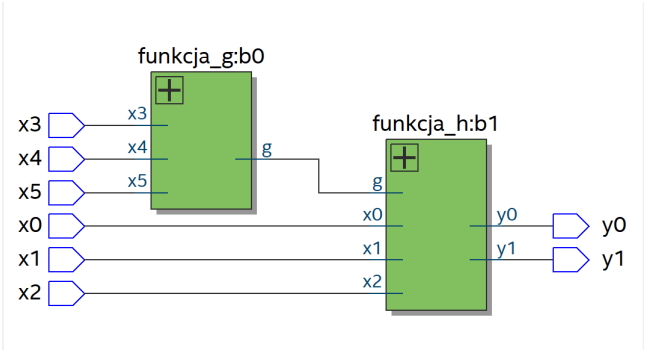


Tworzymy nowy plik Verilog HDL File o nazwie funkcja_h. Zakodujemy funkcję H.
Zawartość pliku funkcja_h:

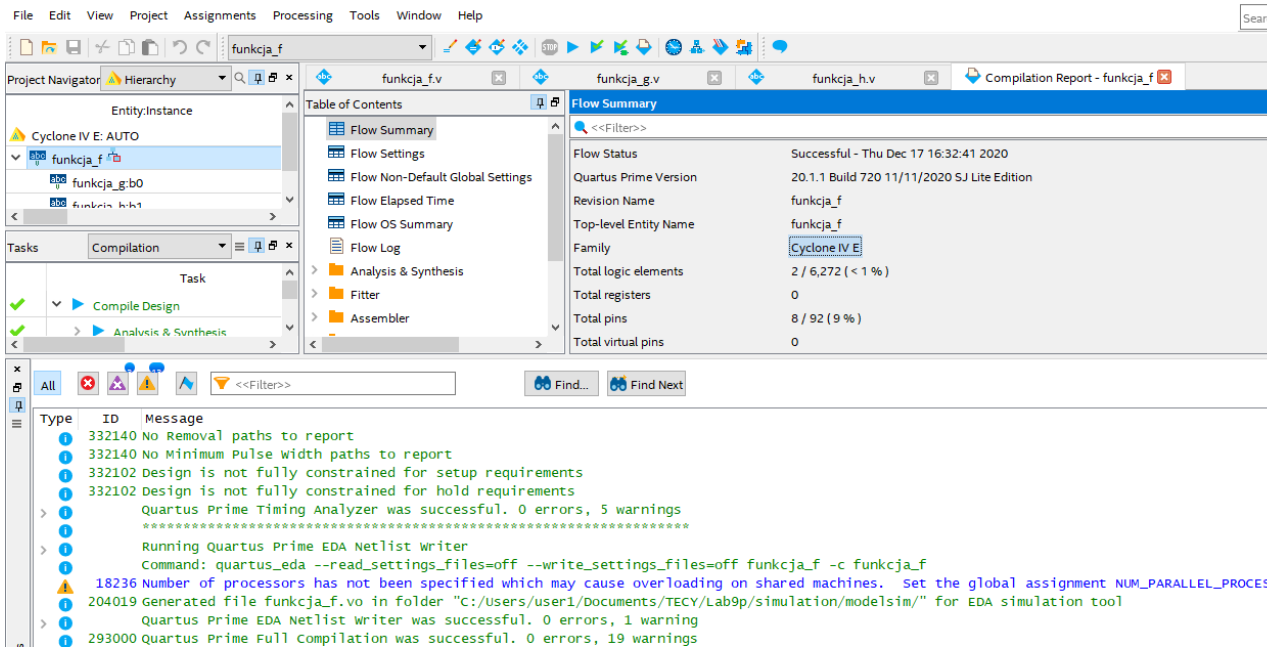
```
module funkcja_h
(
    input x2, x1, x0, g,
     output y1,y0

);
reg [0:1] temp;
always@*
    case ({x2,x1,x0,g})
        //4'd0 : temp = 2'd0;
        //4'd1 : temp = 2'd0;
        //4'd2 : temp = 2'd1;
        //4'd3 : temp = 2'd1;

        //4'd4 : temp = 2'd0;
        4'd5 : temp = 2'd3;
        //4'd6 : temp = 2'd0;
        4'd7 : temp = 2'd3;

        4'd8 : temp = 2'd0;
        4'd9 : temp = 2'd2;
        //4'd10 : temp = 2'd0;
        //4'd11 : temp = 2'd0;

        4'd12 : temp = 2'd1;
        4'd13 : temp = 2'd2;
        4'd14 : temp = 2'd2;
        4'd15 : temp = 2'd1;
       default: temp = 2'd0;
    endcase
assign {y1,y0} = temp;
endmodule
```



STRUKTURA



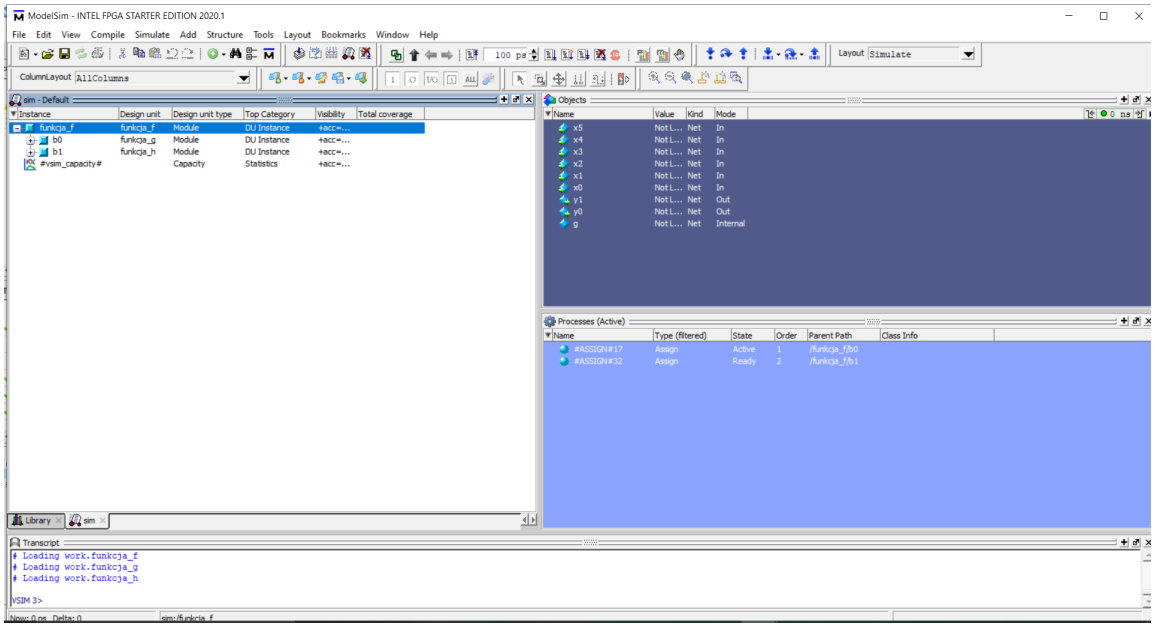
Compilation Report


File Edit View Project Assignments Processing Tools Window Help
funkcja_f
Project Navigator Hierarchy
Entity:Instance
Cyclone IV E: AUTO
funkcja_f
funkcja_g:b0
Tasks Compilation
Task
Compile Design
funkcja_f.v
funkcja_g.v
funkcja_h.v
Compilation Report - funkcja_f
Table of Contents
Flow Summary
Flow Settings
Flow Non-Default Global Settings
Flow Elapsed Time
Flow OS Summary
Flow Log
Analysis & Synthesis
Fitter
Assembler
Flow Summary
<<Filter>>
Flow Status Successful - Thu Dec 17 16:32:41 2020
Quartus Prime Version 20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name funkcja_f
Top-level Entity Name funkcja_f
Family Cyclone IV E
Total logic elements 2 / 6,272 (< 1 %)
Total registers 0
Total pins 8 / 92 (9 %)
Total virtual pins 0
All
<<Filter>>
Find...
Find Next
Type ID Message
332140 No Removal paths to report
332140 No Minimum Pulse Width paths to report
332102 Design is not fully constrained for setup requirements
332102 Design is not fully constrained for hold requirements
Quartus Prime Timing Analyzer was successful. 0 errors, 5 warnings
Running Quartus Prime EDA Netlist Writer
Command: quartus_eda --read_settings_files=off --write_settings_files=off funkcja_f -c funkcja_f
18236 Number of processors has not been specified which may cause overloading on shared machines. Set the global assignment NUM_PARALLEL_PROCES
204019 Generated file funkcja_f.vo in folder "C:/Users/user1/Documents/TECY/Lab9p/simulation/modelsim/" for EDA simulation tool
Quartus Prime EDA Netlist Writer was successful. 0 errors, 1 warning
293000 Quartus Prime Full Compilation was successful. 0 errors, 19 warnings

1

2.3 Weryfikacja w ModelSim

Przeprowadzamy symulację w ModelSim.



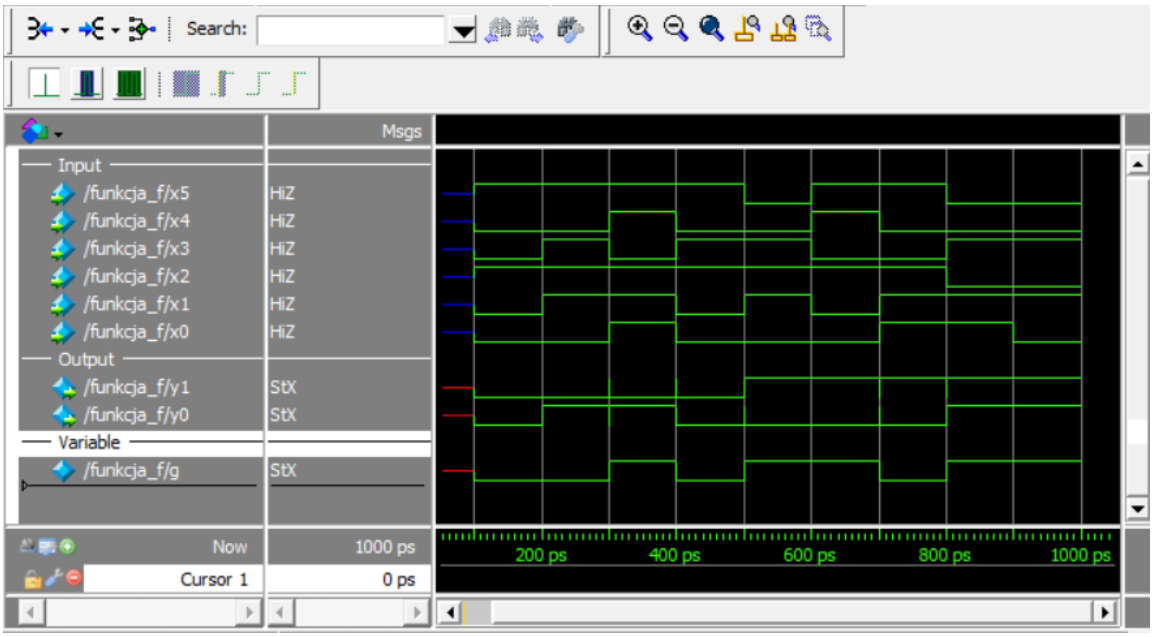
Rysunek 1: Widok okna ModelSim

Sprawdzamy poprawność wyników dla wejść podanych w tabeli prawdy funkcji F.

Zawartość pliku testującego test_f.do:

```
restart -nowave -force
add wave *
run
# 36: 100100
force x5 1 0
force x4 0 0
force x3 0 0
force x2 1 0
force x1 0 0
force x0 0 0
run
# 46: 101110
force x5 1 0
force x4 0 0
force x3 1 0
force x2 1 0
force x1 1 0
force x0 0 0
run
# 55: 110111
force x5 1 0
force x4 1 0
force x3 0 0
force x2 1 0
force x1 1 0
force x0 1 0
run
#44: 101100
force x5 1 0
force x4 0 0
force x3 1 0
force x2 1 0
force x1 0 0
force x0 0 0
run
# 14: 001110
force x5 0 0
force x4 0 0
force x3 1 0
force x2 1 0
force x1 1 0
force x0 0 0
run
# 52: 110100
force x5 1 0
force x4 1 0
force x3 0 0
force x2 1 0
force x1 0 0
force x0 0 0
run
# 39: 100111
force x5 1 0
force x4 0 0
force x3 0 0
force x2 1 0
force x1 1 0
force x0 1 0
run
# 11: 001011
force x5 0 0
force x4 0 0
force x3 1 0
force x2 0 0
force x1 1 0
force x0 1 0
run
# 10: 001010
force x5 0 0
force x4 0 0
force x3 1 0
force x2 0 0
force x1 1 0
force x0 0 0
run
```

Widok okna Wave po wykonaniu przez ModelSim polecenia >do test_f.do:



Na podstawie otrzymanych sygnałów zapiszemy tabelę:

x_5	x_4	x_3	x_2	x_1	x_0	y_1	y_0
1	0	0	1	0	0	0	0
1	0	1	1	1	0	0	1
1	1	0	1	1	1	0	1
1	0	1	1	0	0	0	0
0	0	1	1	0	0	1	0
1	1	0	1	0	0	1	0
1	0	0	1	1	1	1	0
0	0	1	0	1	1	1	1
0	0	1	0	1	0	1	1

Powstała tabela zgadza się z podaną w treści zadania tabelą funkcji F.

3 Dekompozycja 3

Istneje też dekompozycja F= H(G1(x4,x2,x1), G(x5,x4,x3), x0)
funkcja G ma postać:

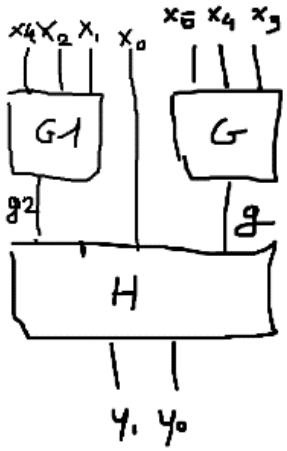
x5x4x3	g
000	-
001	1
010	-
011	-
100	0
101	0
110	1
111	-

Funkcja G1 ma postać:

x4x2x1	g2
000	-
001	0
010	0
011	1
100	-
101	-
110	1
111	1

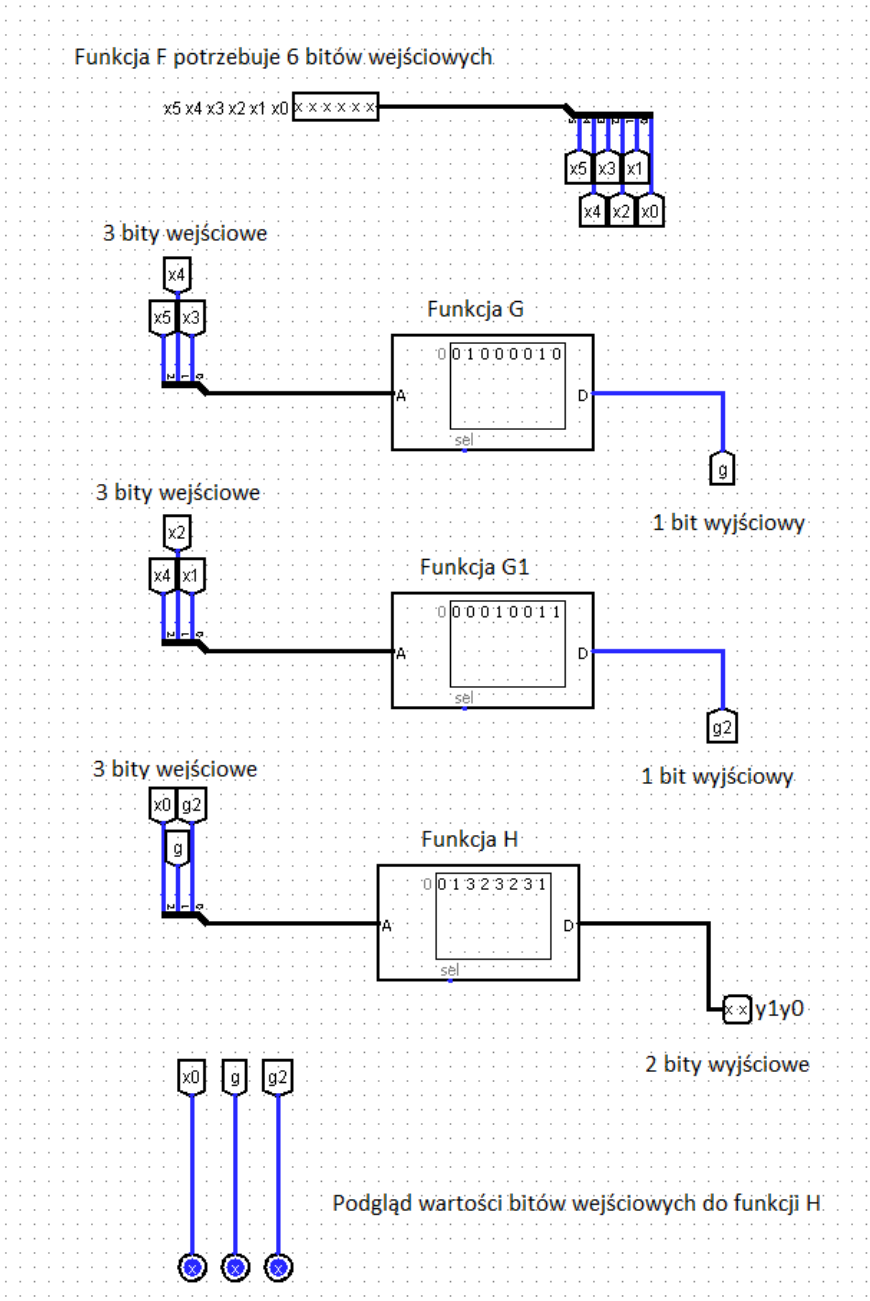
Funkcja H ma postać:

x0 g/g2	0	1
00	00	01
01	11	10
10	11	10
11	11	01

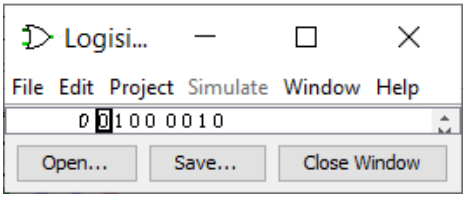


3.1 Realizacja w Logisim

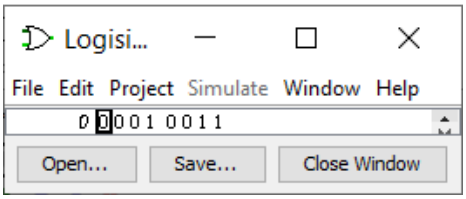
Funkcja F ma 6 bitów wejściowych i 2 wyjściowe



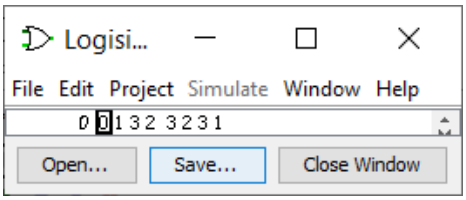
Zawartość pamięci ROM dla funkcji G



Zawartość pamięci ROM dla funkcji G1

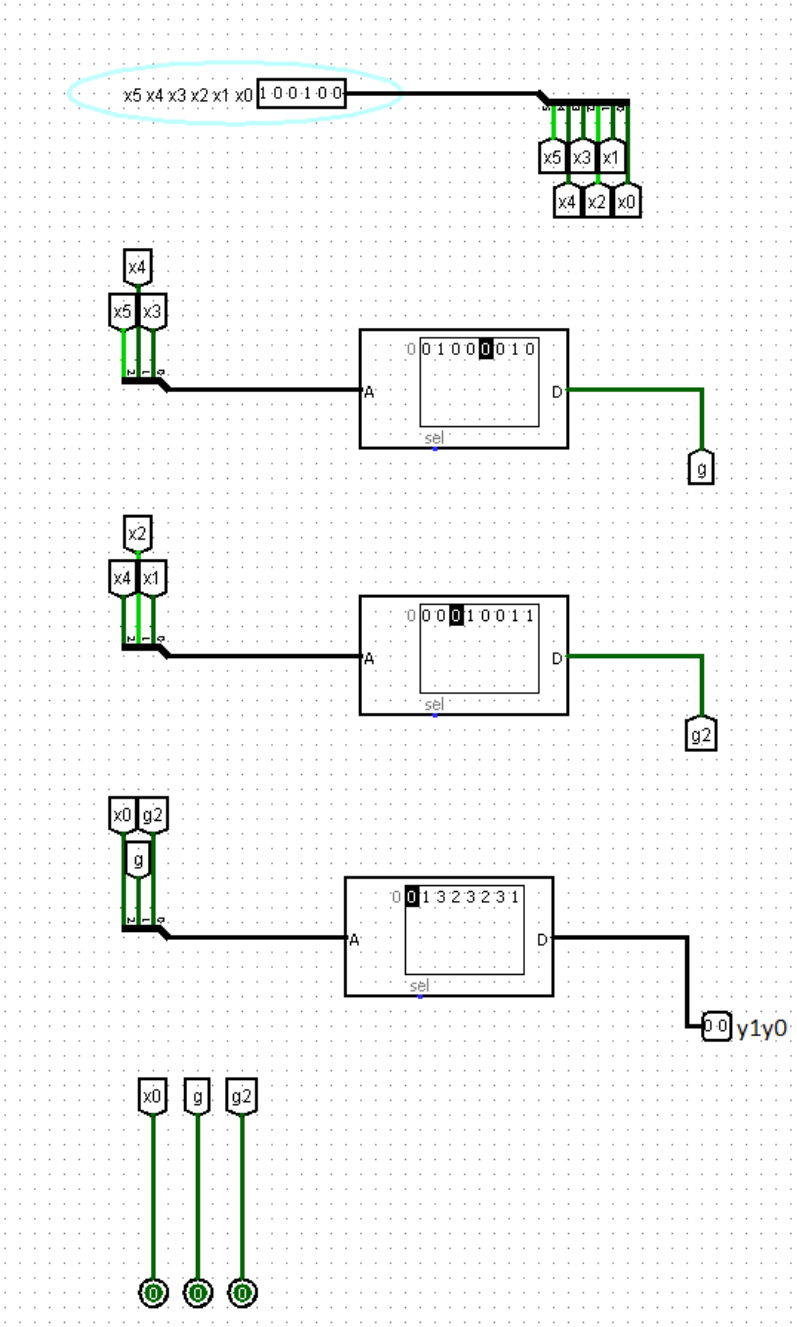


Zawartość pamięci ROM dla funkcji H

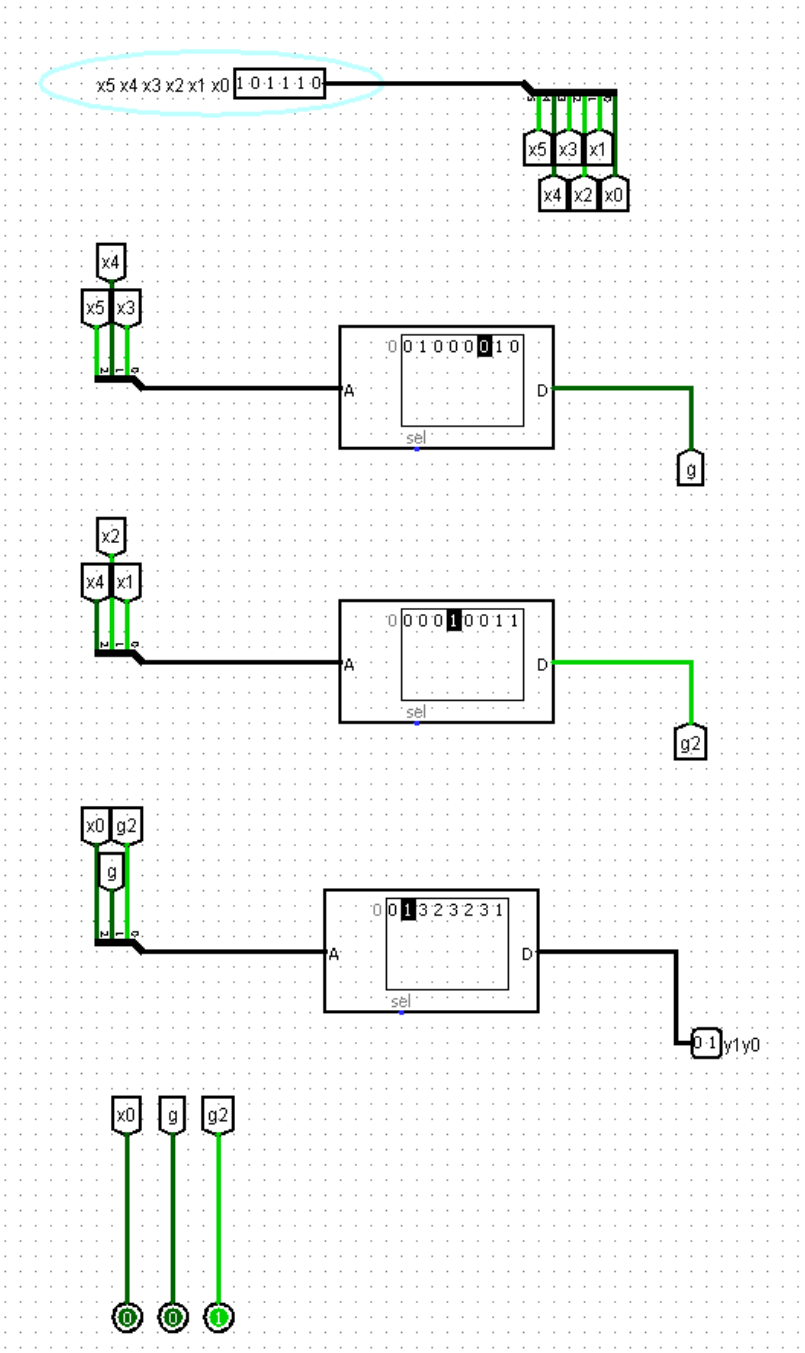


TESTY

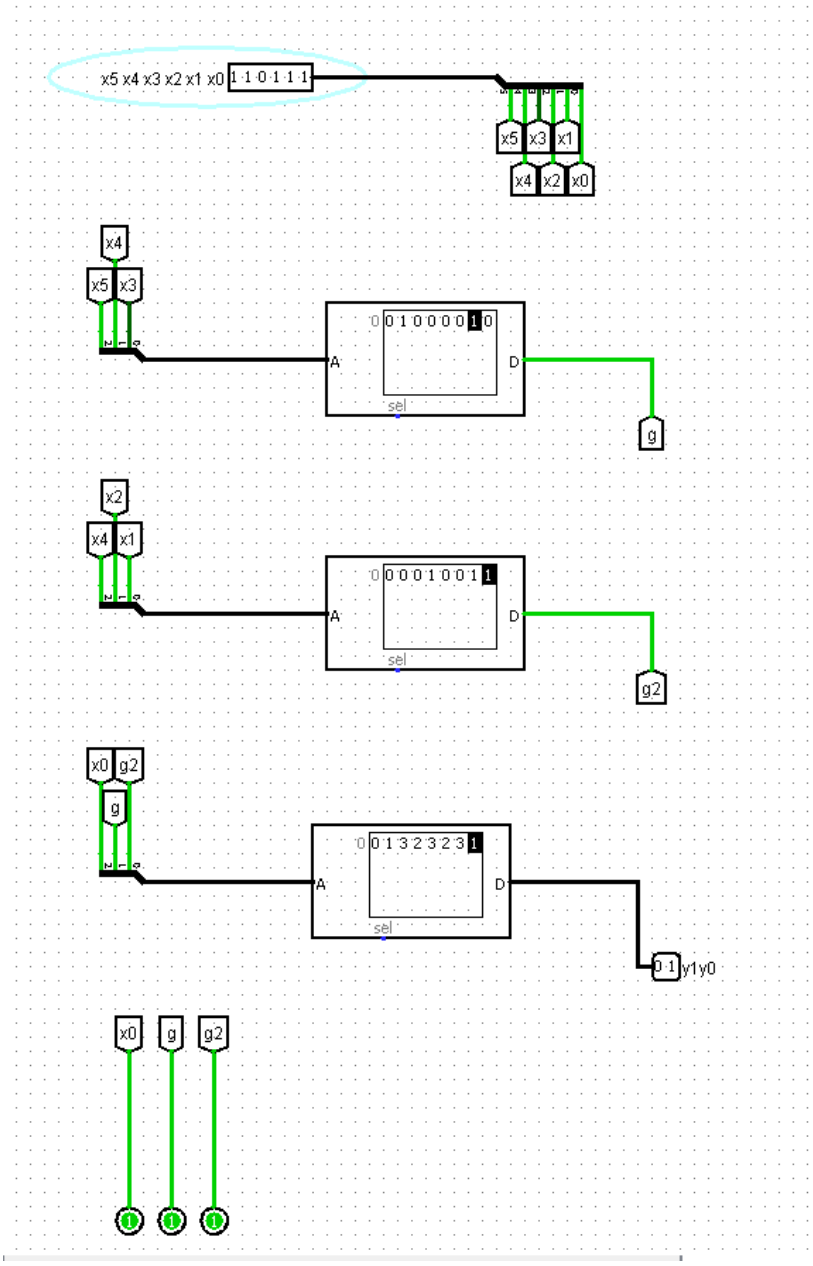
5x4x3x2x1=100100



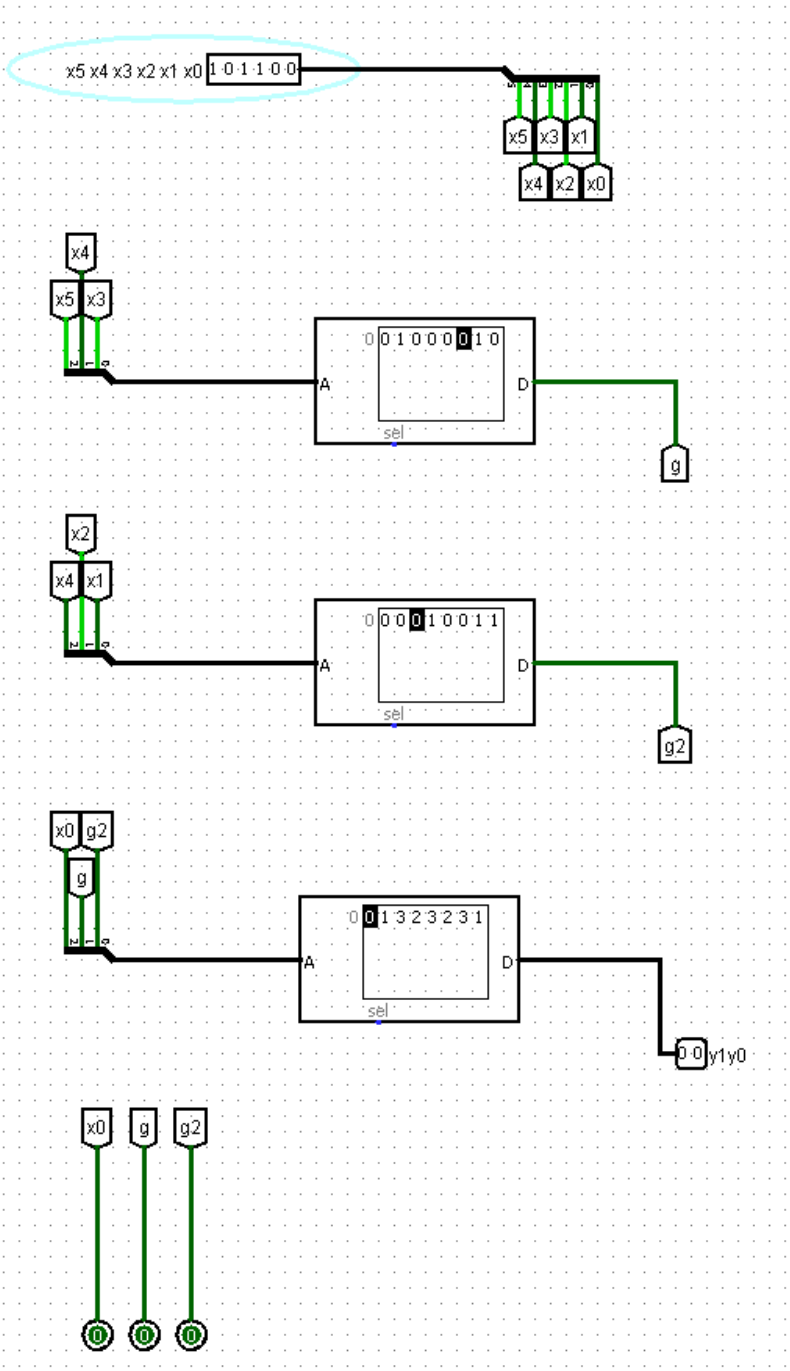
x5x4x3x2x1=101110



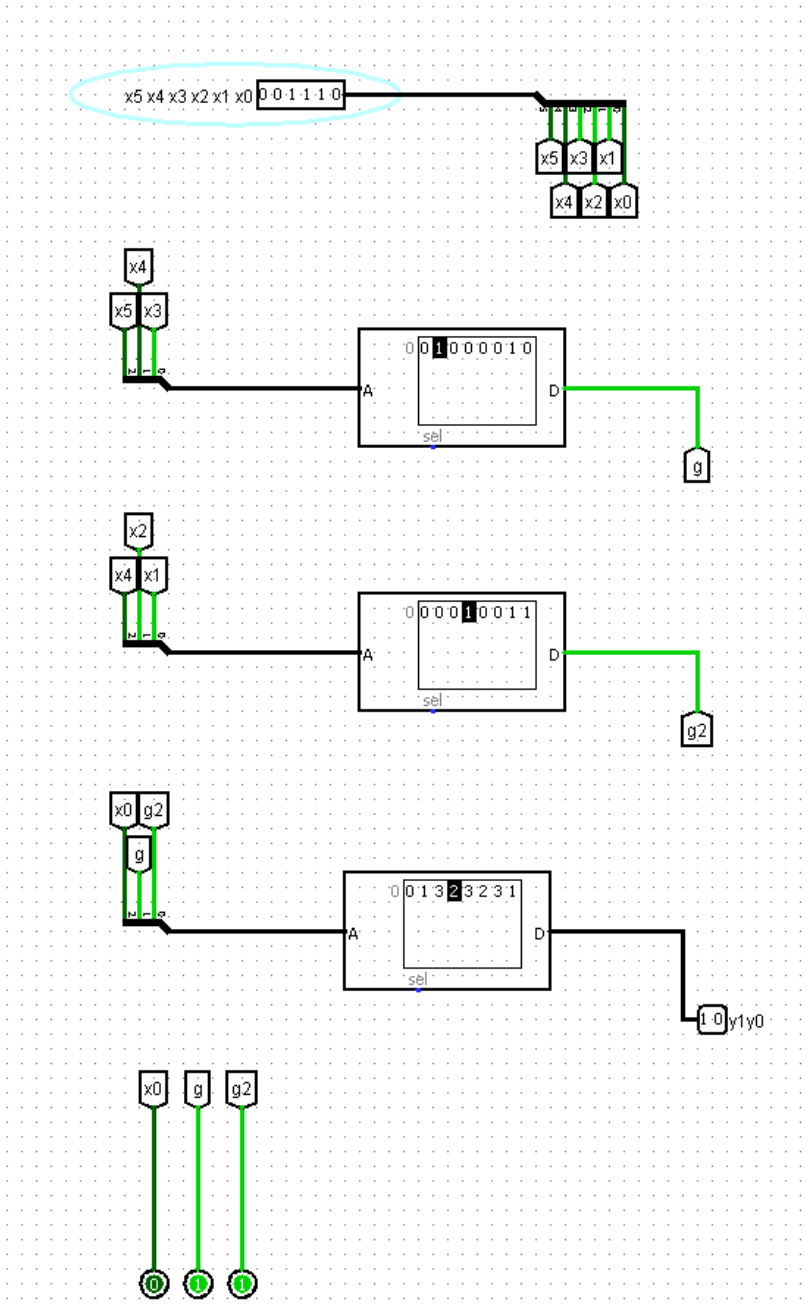
x5x4x3x2x1=110111



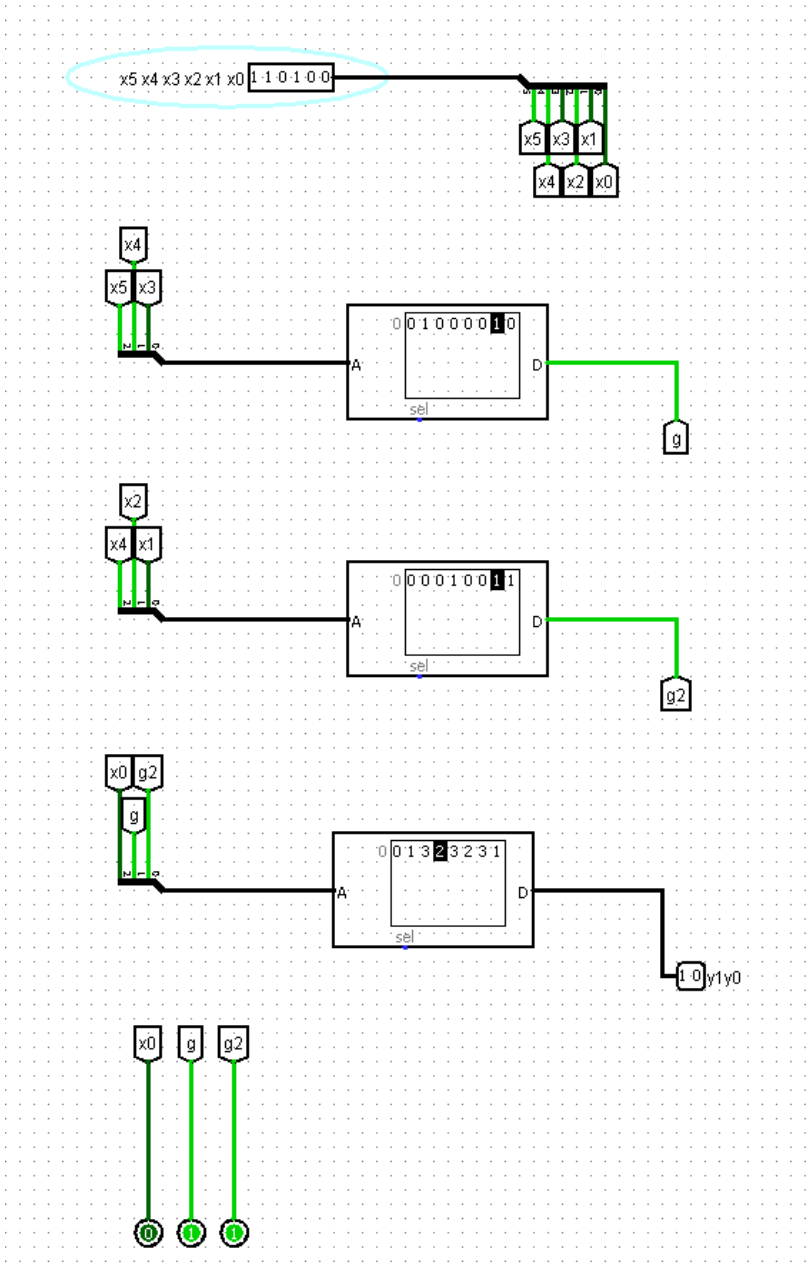
x5x4x3x2x1=101100



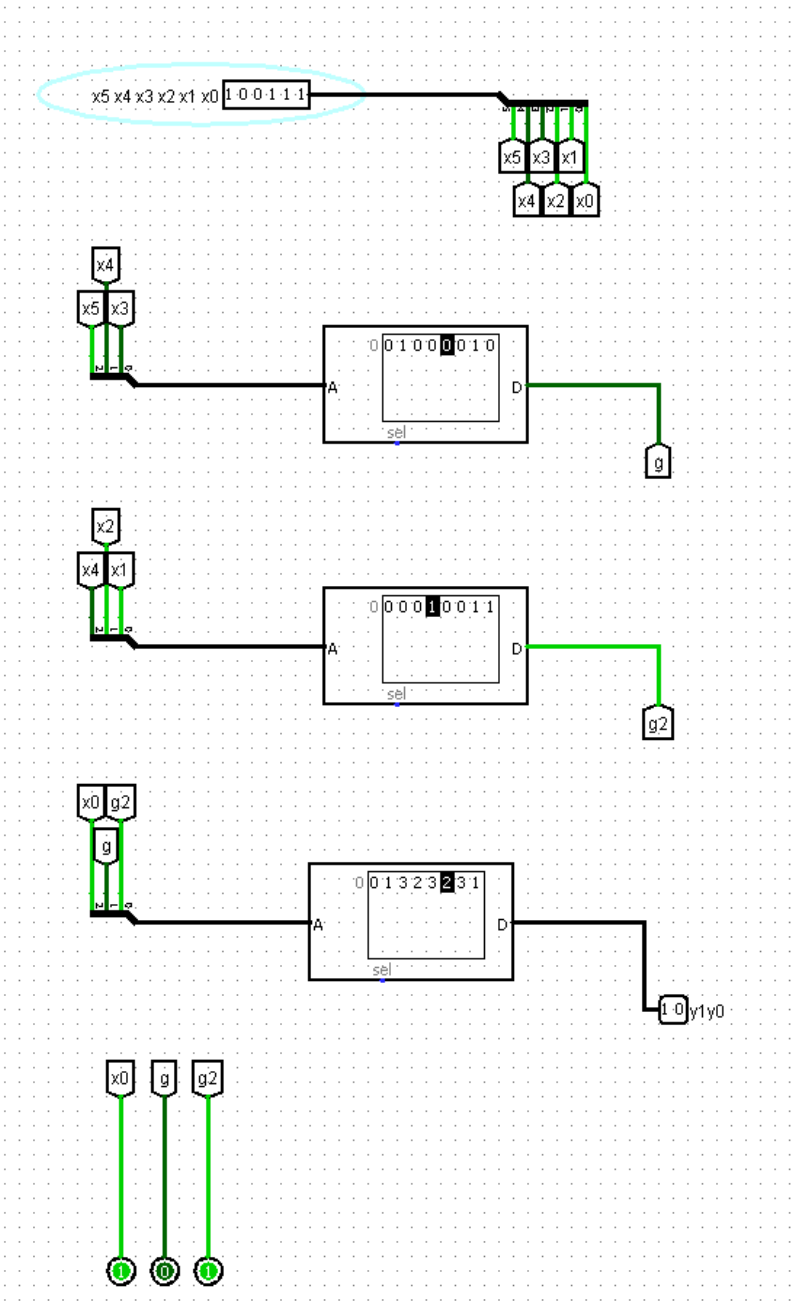
x5x4x3x2x1=001110



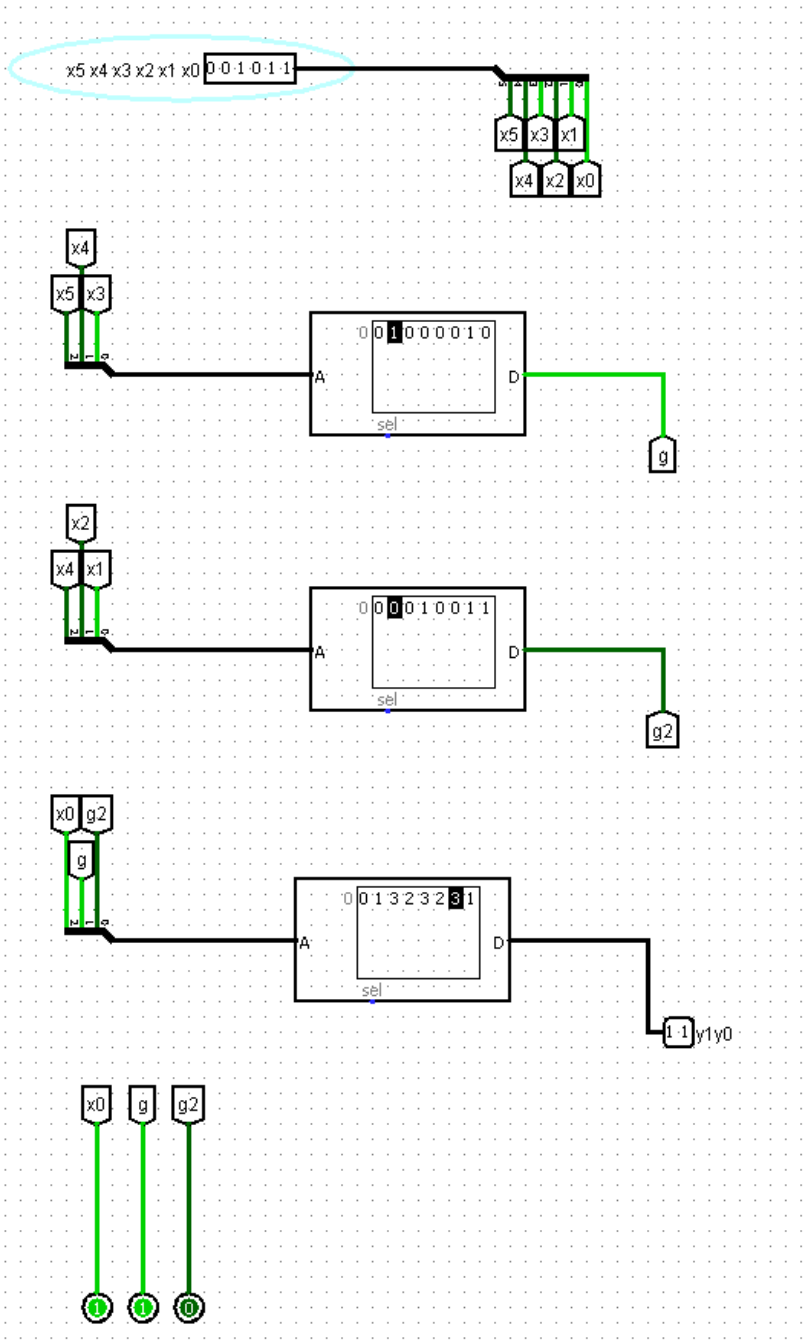
x5x4x3x2x1=110100



x5x4x3x2x1=100111



x5x4x3x2x1=001011



3.2 Realizacja w Quartus

Tworzymy plik Verilog HDL File o nazwie funkcja_F. Zakodujemy funkcję F.
Zawartość pliku funkcja_f1:

```
module funkcja_F
(
    input x5,x4,x3,x2,x1,x0,
     output y1,y0
);
wire g,g2;

funkcja_G b0 (.x5(x5),.x4(x4),.x3(x3),.g(g));
funkcja_G1 b1 (.x4(x4),.x2(x2),.x1(x1),.g2(g2));
funkcja_H b2 (.x0(x0),.g(g),.g2(g2),.y1(y1),.y0(y0));

endmodule
```

Tworzymy nowy plik Verilog HDL File o nazwie funkcja_G. Zakodujemy funkcję G
Zawartość pliku funkcja_G:

```
module funkcja_G
(
    input x5, x4, x3,
     output g

);
reg temp;

always@*
    case ({x5,x4,x3})
        3'd4 : temp = 1'd0;
        3'd5 : temp = 1'd0;
        3'd6 : temp = 1'd1;
        3'd1 : temp = 1'd1;
       default: temp = 1'd0;
    endcase
assign {g} = temp;
endmodule
```

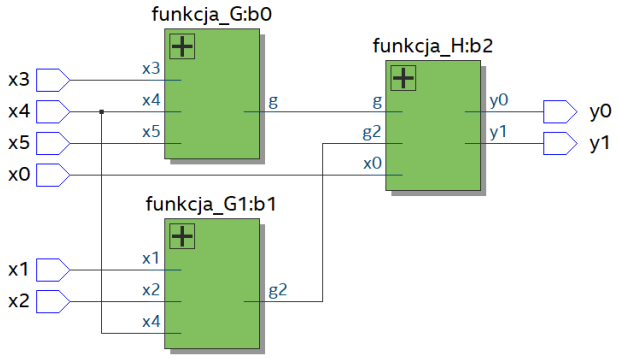
Tworzymy nowy plik Verilog HDL File o nazwie funkcja_G1. Zakodujemy funkcję G1.
Zawartość pliku funkcja_G1:

```
module funkcja_G1
(
    input x4, x2, x1,
     output g2

);
reg temp;

always@*
    case ({x4,x2,x1})
        3'd1 : temp = 1'd0;
        3'd2 : temp = 1'd0;
        3'd3 : temp = 1'd1;
        3'd6 : temp = 1'd1;
        3'd7 : temp = 1'd1;
       default: temp = 1'd0;
    endcase
assign {g2} = temp;
endmodule
```

Tworzymy nowy plik Verilog HDL File o nazwie funkcja_H. Zakodujemy funkcję H.
Zawartość pliku funkcja_H:

```
1  module funkcja_H
2  (
3      input x0, g, g2,
4       output y1,y0
5
6  );
7  reg [0:1] temp;
8  always@*
9      case ({x0, g, g2})
10         3'd0 : temp = 2'd0;
11         3'd1 : temp = 2'd1;
12         3'd2 : temp = 2'd3;
13         3'd3 : temp = 2'd2;
14
15         3'd4 : temp = 2'd3;
16         3'd5 : temp = 2'd2;
17         3'd6 : temp = 2'd3;
18         3'd7 : temp = 2'd1;
19         default: temp = 2'd0;
20     endcase
21 assign {y1,y0} = temp;
22 endmodule
```

STRUKTURA

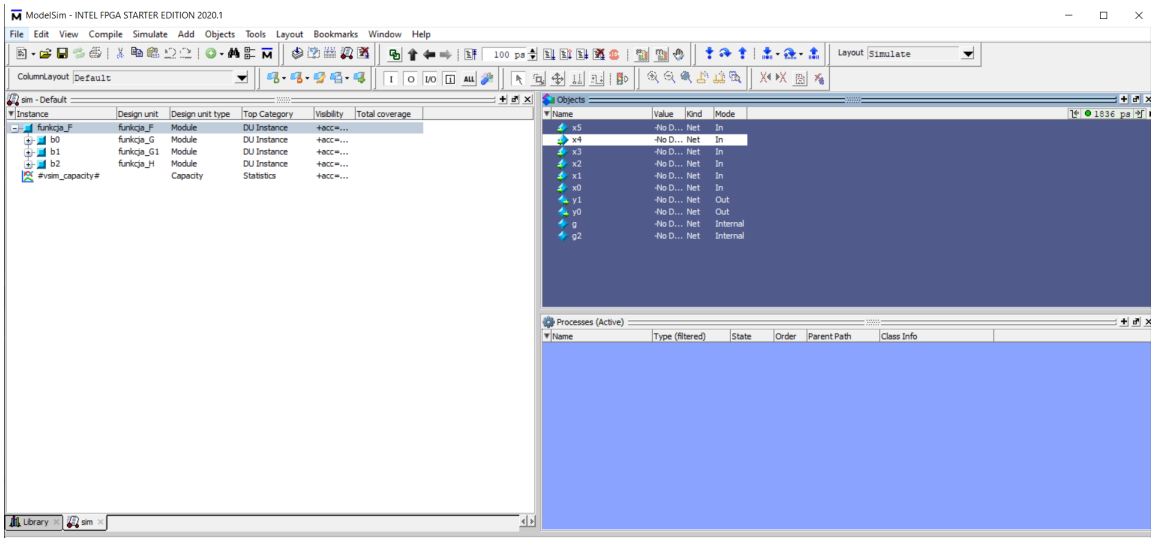


Compilation Report

Flow Status	Successful - Sun Jan 03 11:24:18 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 SJ Lite Edition
Revision Name	funkcja_F
Top-level Entity Name	funkcja_F
Family	Cyclone IV E
Device	EP4CE115F29C7
Timing Models	Final
Total logic elements	4 / 114,480 (< 1 %)
Total registers	0
Total pins	8 / 529 (2 %)
Total virtual pins	0
Total memory bits	0 / 3,981,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 532 (0 %)
Total PLLs	0 / 4 (0 %)

3.3 Weryfikacja w ModelSim

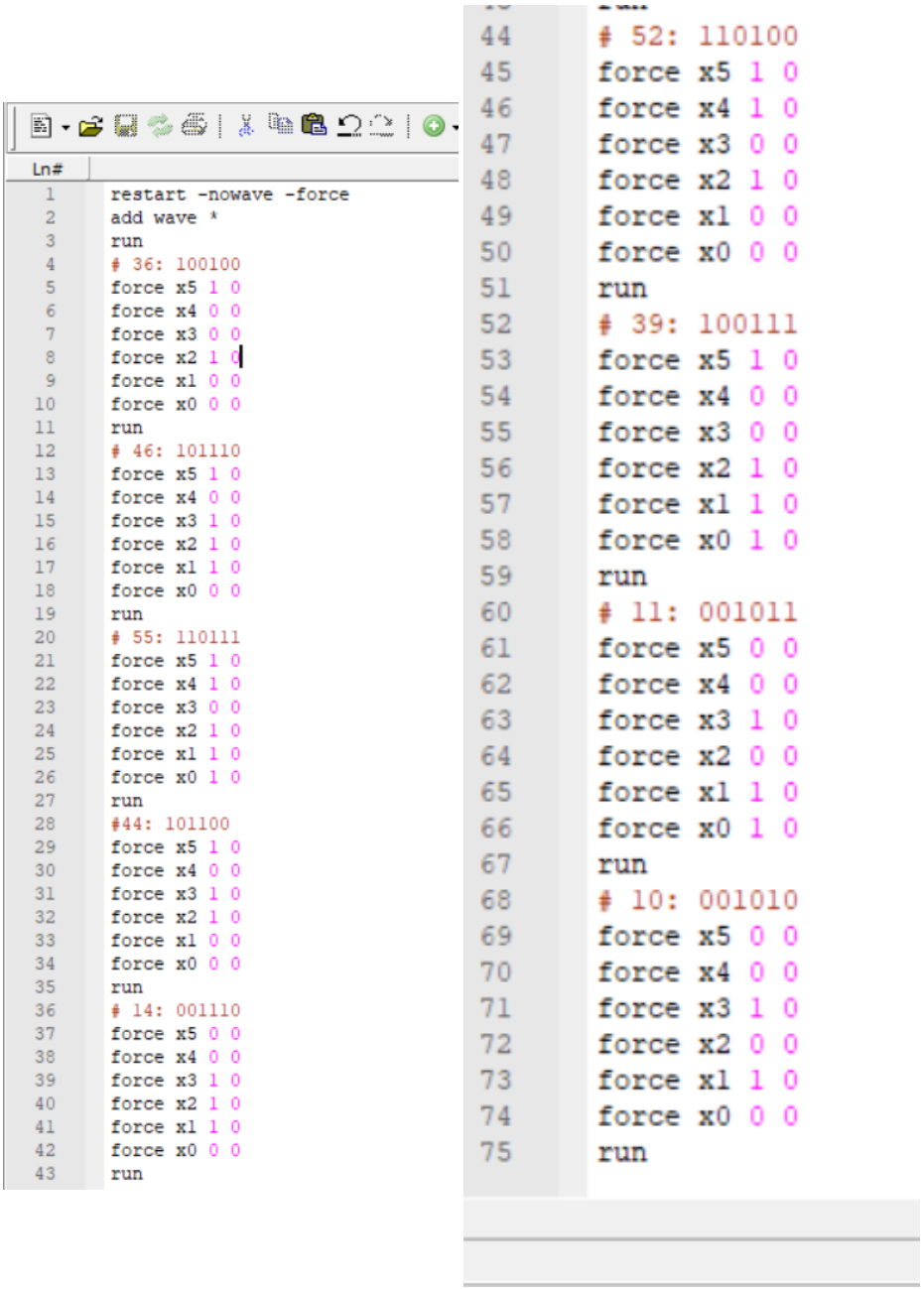
Przeprowadzamy symulację w ModelSim.



Rysunek 2: Widok okna ModelSim

Sprawdzamy poprawność wyników dla wejść podanych w tabeli prawdy funkcji F.

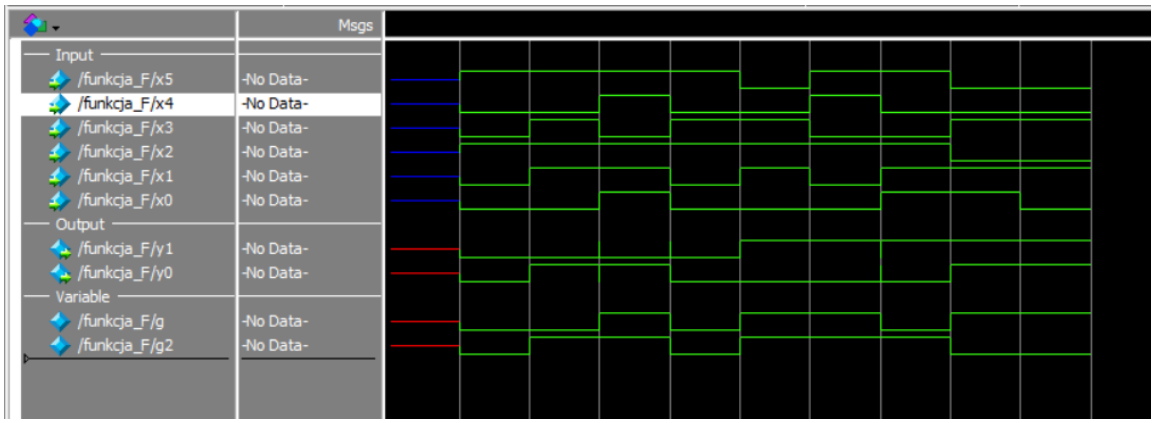
Zawartość pliku testującego test_f.do:



```
Ln#
1   restart -nowave -force
2   add wave *
3   run
4   # 36: 100100
5   force x5 1 0
6   force x4 0 0
7   force x3 0 0
8   force x2 1 0
9   force x1 0 0
10  force x0 0 0
11  run
12  # 46: 101110
13  force x5 1 0
14  force x4 0 0
15  force x3 1 0
16  force x2 1 0
17  force x1 1 0
18  force x0 0 0
19  run
20  # 55: 110111
21  force x5 1 0
22  force x4 1 0
23  force x3 0 0
24  force x2 1 0
25  force x1 1 0
26  force x0 1 0
27  run
28  #44: 101100
29  force x5 1 0
30  force x4 0 0
31  force x3 1 0
32  force x2 1 0
33  force x1 0 0
34  force x0 0 0
35  run
36  # 14: 001110
37  force x5 0 0
38  force x4 0 0
39  force x3 1 0
40  force x2 1 0
41  force x1 1 0
42  force x0 0 0
43  run
```

```
44  # 52: 110100
45  force x5 1 0
46  force x4 1 0
47  force x3 0 0
48  force x2 1 0
49  force x1 0 0
50  force x0 0 0
51  run
52  # 39: 100111
53  force x5 1 0
54  force x4 0 0
55  force x3 0 0
56  force x2 1 0
57  force x1 1 0
58  force x0 1 0
59  run
60  # 11: 001011
61  force x5 0 0
62  force x4 0 0
63  force x3 1 0
64  force x2 0 0
65  force x1 1 0
66  force x0 1 0
67  run
68  # 10: 001010
69  force x5 0 0
70  force x4 0 0
71  force x3 1 0
72  force x2 0 0
73  force x1 1 0
74  force x0 0 0
75  run
```

Widok okna Wave po wykonaniu przez ModelSim polecenia >do test_f.do:



Na podstawie otrzymanych sygnałów zapiszemy tabelę:

x_5	x_4	x_3	x_2	x_1	x_0	y_1	y_0
1	0	0	1	0	0	0	0
1	0	1	1	1	0	0	1
1	1	0	1	1	1	0	1
1	0	1	1	0	0	0	0
0	0	1	1	0	0	1	0
1	1	0	1	0	0	1	0
1	0	0	1	1	1	1	0
0	0	1	0	1	1	1	1
0	0	1	0	1	0	1	1

Powstała tabela zgadza się z podaną w treści zadania tabelą funkcji F.