LAB2 - 4 to 16 Decoder

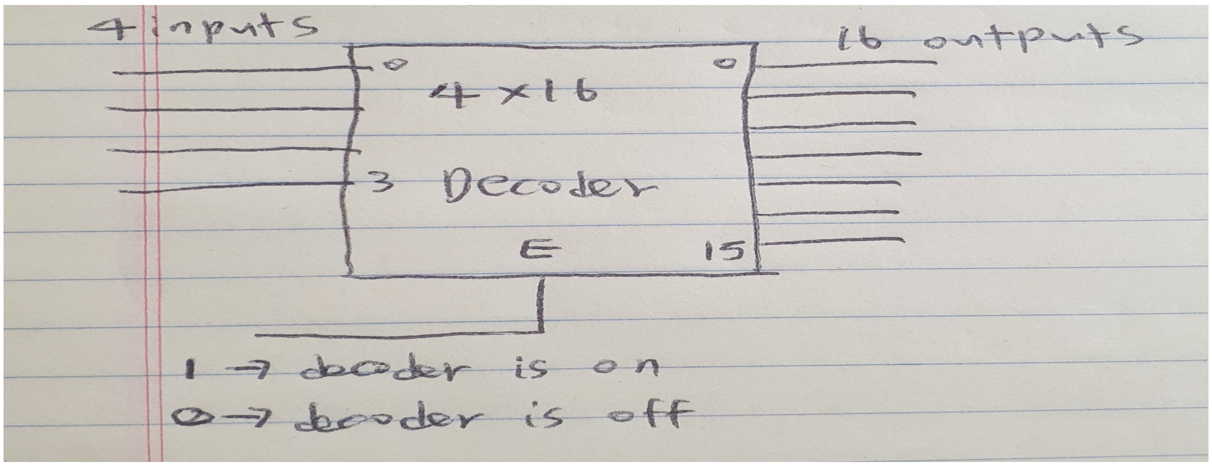
Group A :

- **Jihun Choi(Bronco ID#: 011841206)**
- **Jacky Li(Bronco ID#:014181245)**

Prof.Aly

ECE3300L.02 - Summer 2022

This lab2 is about designing a 4 to16 decoder by using vivado software and test the results with FPGA. Input 'in' is 4 bits, Input 'enable' which is an enable line is just 1 bit, and Output 'out' is 16 bits. The function of 4 to 16 decoder is pretty simple. When the enable line equals '0', the decoder is off. Even if we feed any 4 bits input, output will be simply zero(16b'0000000000000000). And when the enable line equals '1', the decoder is on.



Since it is on,

if we feed input 'in' as 0000, output 'out' equals '16b'0000000000000001'.

if we feed input 'in' as 0001, output 'out' equals '16b'0000000000000010'.

if we feed input ‘in’ as 0010, output ‘out’ equals ‘16b’0000000000000100’.

.

.

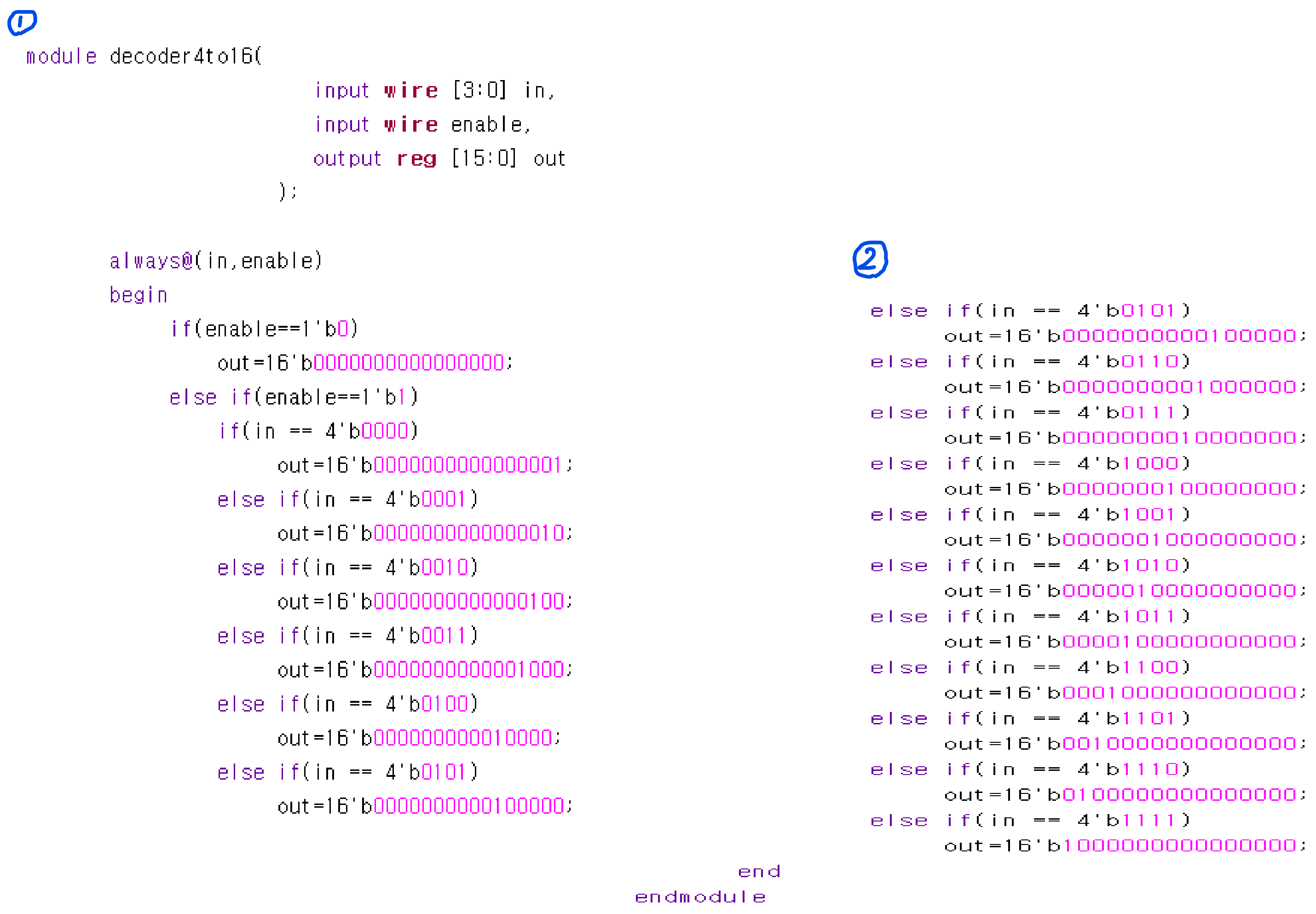
.

(continued)

if we feed input ‘in’ as 1110, output ‘out’ equals ‘16b’0100000000000000’.

if we feed input ‘in’ as 1111, output ‘out’ equals ‘16b’1000000000000000’.

To make input ‘in’’ 4 bits, ‘[3:0]’ was used. Same logic for output ‘out’, [15:0] was used for 16 bits. Once ports are set correctly, always block was used to implement if else statement.



①

```
module decoder4to16(
                        input wire [3:0] in,
                        input wire enable,
                        output reg [15:0] out
                     );

       always@(in,enable)
       begin
            if(enable==1'b0)
                out=16'b0000000000000000;
            else if(enable==1'b1)
                if(in == 4'b0000)
                     out=16'b0000000000000001;
                else if(in == 4'b0001)
                     out=16'b0000000000000010;
                else if(in == 4'b0010)
                     out=16'b0000000000000100;
                else if(in == 4'b0011)
                     out=16'b0000000000001000;
                else if(in == 4'b0100)
                     out=16'b0000000000010000;
                else if(in == 4'b0101)
                     out=16'b0000000000100000;
```

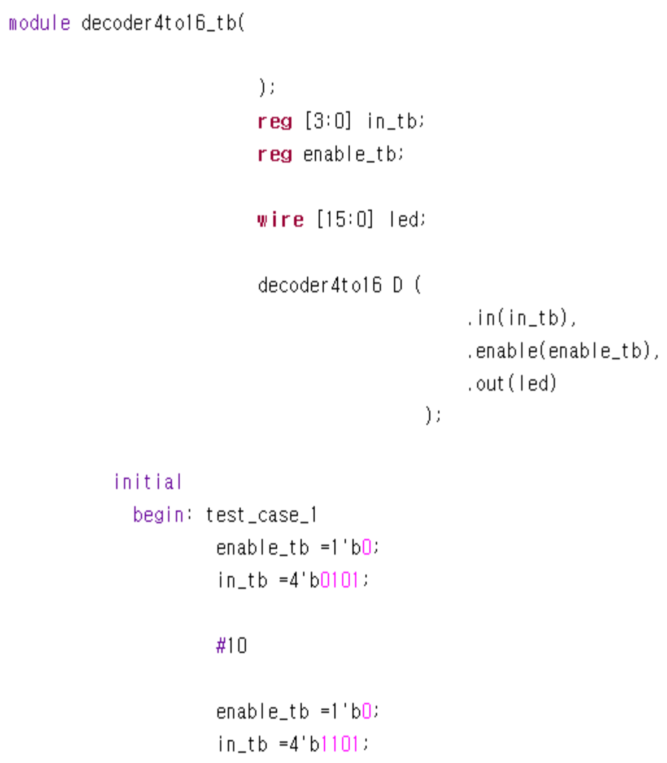
②

```
else if(in == 4'b0101)
      out=16'b0000000000100000;
else if(in == 4'b0110)
      out=16'b0000000001000000;
else if(in == 4'b0111)
      out=16'b0000000010000000;
else if(in == 4'b1000)
      out=16'b0000000100000000;
else if(in == 4'b1001)
      out=16'b0000001000000000;
else if(in == 4'b1010)
      out=16'b0000010000000000;
else if(in == 4'b1011)
      out=16'b0000100000000000;
else if(in == 4'b1100)
      out=16'b0001000000000000;
else if(in == 4'b1101)
      out=16'b0010000000000000;
else if(in == 4'b1110)
      out=16'b0100000000000000;
else if(in == 4'b1111)
      out=16'b1000000000000000;
```

```
        end
      endmodule
```

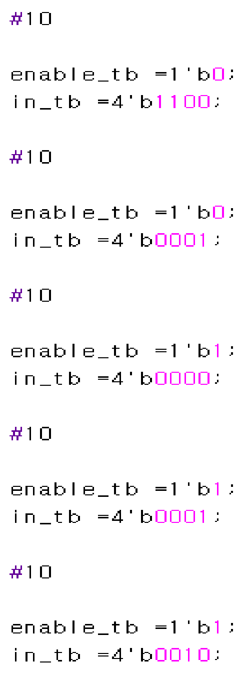
After setting up the source file, test bench was created to test, if 4 to16 decoder designed functions as it intended. First it tests to see if the decoder is off when enable equals line 0. After that it starts testing each case when the enable line equals 1.

①



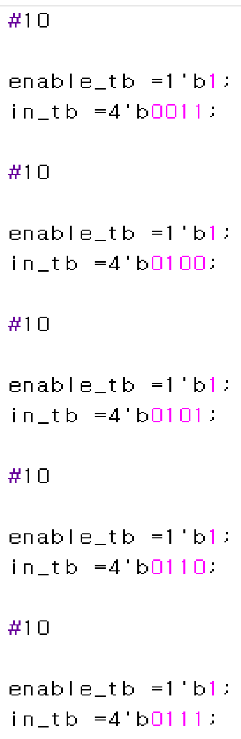
```
module decoder4to16_tb(

                    );
                    reg [3:0] in_tb;
                    reg enable_tb;

                    wire [15:0] led;

                    decoder4to16 D (
                                    .in(in_tb),
                                    .enable(enable_tb),
                                    .out(led)
                                );

        initial
          begin: test_case_1
                enable_tb =1'b0;
                in_tb =4'b0101;

                #10

                enable_tb =1'b0;
                in_tb =4'b1101;
```

②



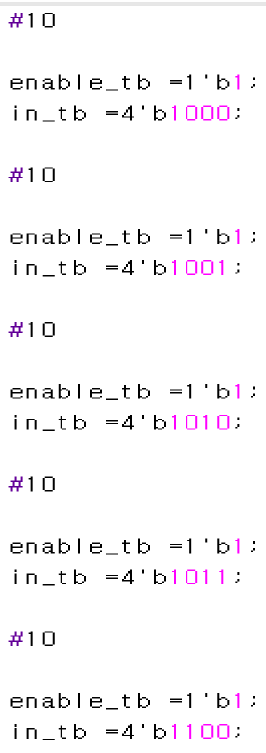
```
#10

enable_tb =1'b0;
in_tb =4'b1100;

#10

enable_tb =1'b0;
in_tb =4'b0001;

#10

enable_tb =1'b1;
in_tb =4'b0000;

#10

enable_tb =1'b1;
in_tb =4'b0001;

#10

enable_tb =1'b1;
in_tb =4'b0010;
```

③



```
#10

enable_tb =1'b1;
in_tb =4'b0011;

#10

enable_tb =1'b1;
in_tb =4'b0100;

#10

enable_tb =1'b1;
in_tb =4'b0101;

#10

enable_tb =1'b1;
in_tb =4'b0110;

#10

enable_tb =1'b1;
in_tb =4'b0111;
```

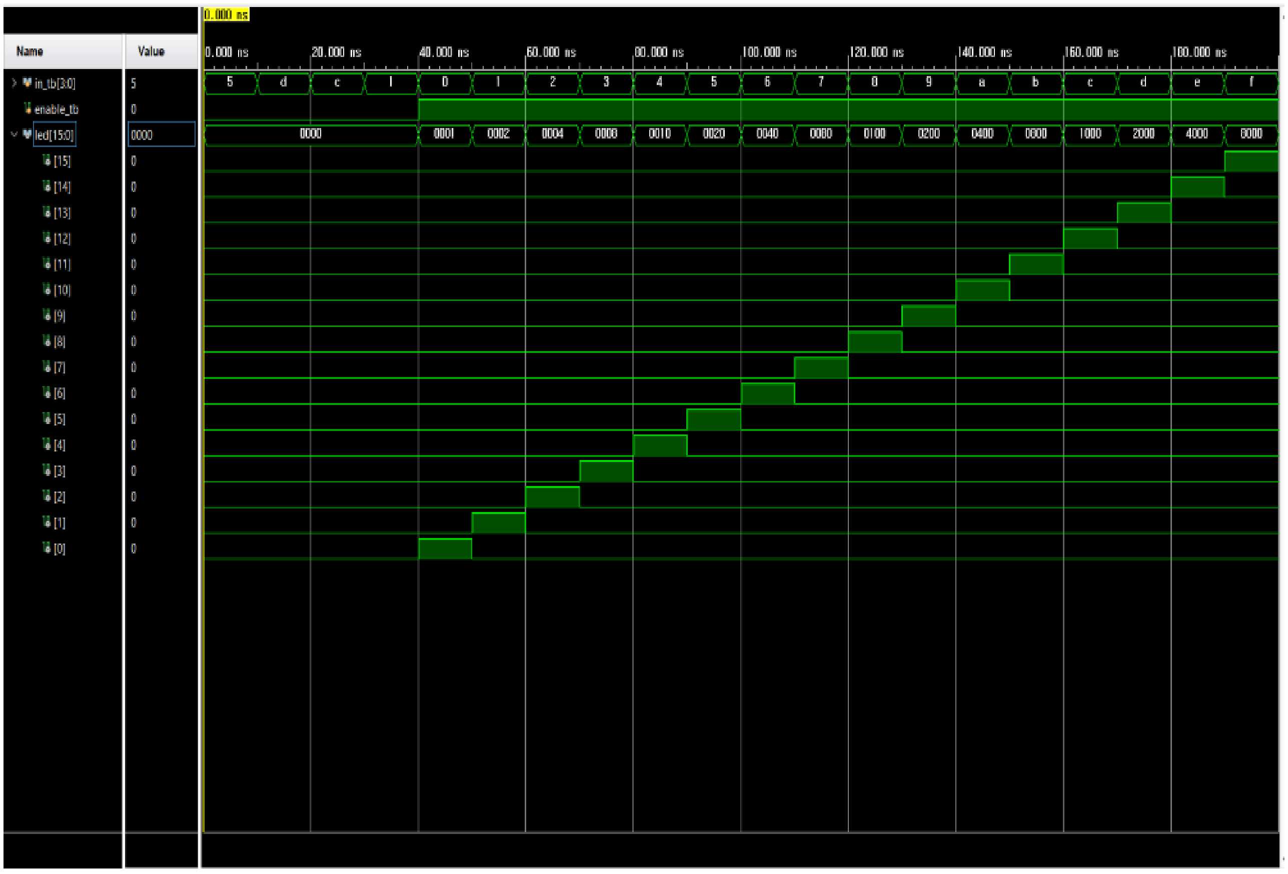
④



```
#10

enable_tb =1'b1;
in_tb =4'b1000;

#10

enable_tb =1'b1;
in_tb =4'b1001;

#10

enable_tb =1'b1;
in_tb =4'b1010;

#10

enable_tb =1'b1;
in_tb =4'b1011;

#10

enable_tb =1'b1;
in_tb =4'b1100;
```

```
                    in_tb =4'b1100;

                    #10

                    enable_tb =1'b1;
                    in_tb =4'b1101;

                    #10

                    enable_tb =1'b1;
                    in_tb =4'b1110;

                    #10

                    enable_tb =1'b1;
                    in_tb =4'b1111;

                    #10
                    $finish;

                end

endmodule
```

And simulation was performed,

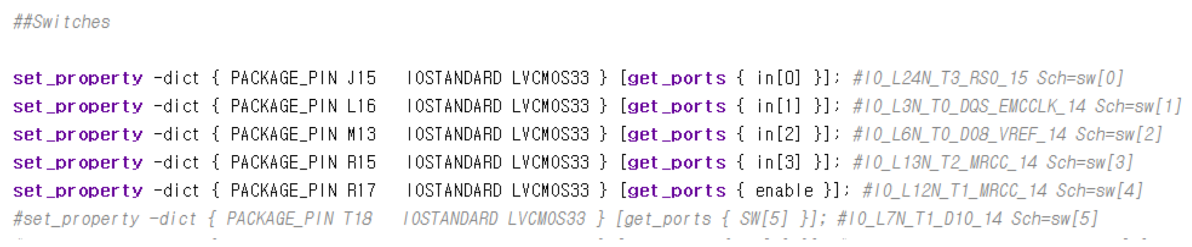


Looking at the result we can see when the enable line is 0, output is simply 0. And when the enable line is 1, each input matches with the correct output.

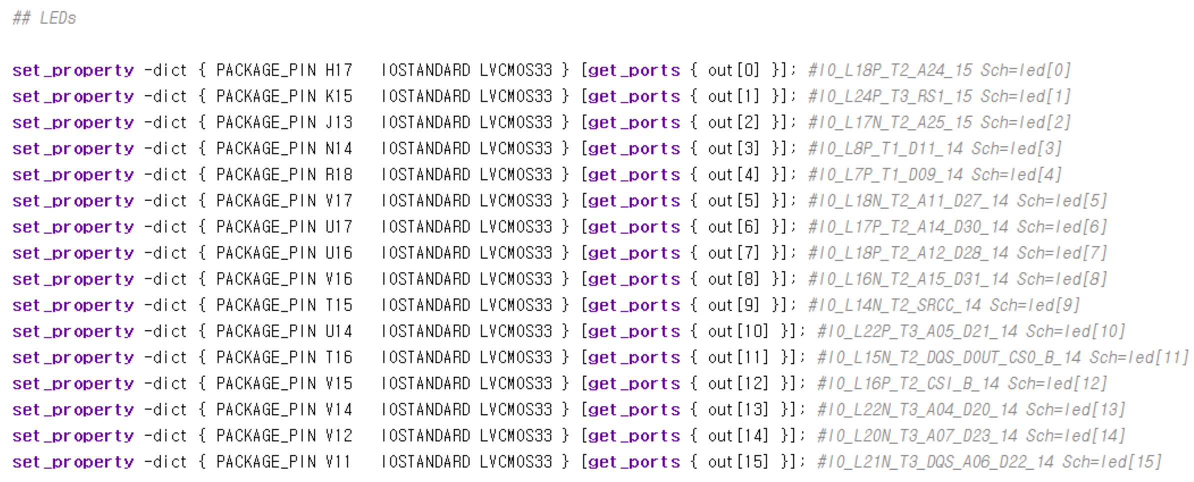
(In case, the simulation result is not readable. use the link below)

Simulation Link:
https://github.com/california-polytechnic-university/ECE3300L_Summer_2022_Group_A/blob/main/ECE3300_LAB2/lab2_simulation_result.PNG

After checking simulation results, 4 to 16 decoder is functioning correctly. xdc file was modified to properly work and the result was tested on FPGA.



```
##Switches

set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { in[0] }]; #IO_L24N_T3_RS0_15 Sch=sw[0]
set_property -dict { PACKAGE_PIN L16   IOSTANDARD LVCMOS33 } [get_ports { in[1] }]; #IO_L3N_T0_DQS_EMCCLK_14 Sch=sw[1]
set_property -dict { PACKAGE_PIN M13   IOSTANDARD LVCMOS33 } [get_ports { in[2] }]; #IO_L6N_T0_D08_VREF_14 Sch=sw[2]
set_property -dict { PACKAGE_PIN R15   IOSTANDARD LVCMOS33 } [get_ports { in[3] }]; #IO_L13N_T2_MRCC_14 Sch=sw[3]
set_property -dict { PACKAGE_PIN R17   IOSTANDARD LVCMOS33 } [get_ports { enable }]; #IO_L12N_T1_MRCC_14 Sch=sw[4]
#set_property -dict { PACKAGE_PIN T18   IOSTANDARD LVCMOS33 } [get_ports { SW[5] }]; #IO_L7N_T1_D10_14 Sch=sw[5]
```



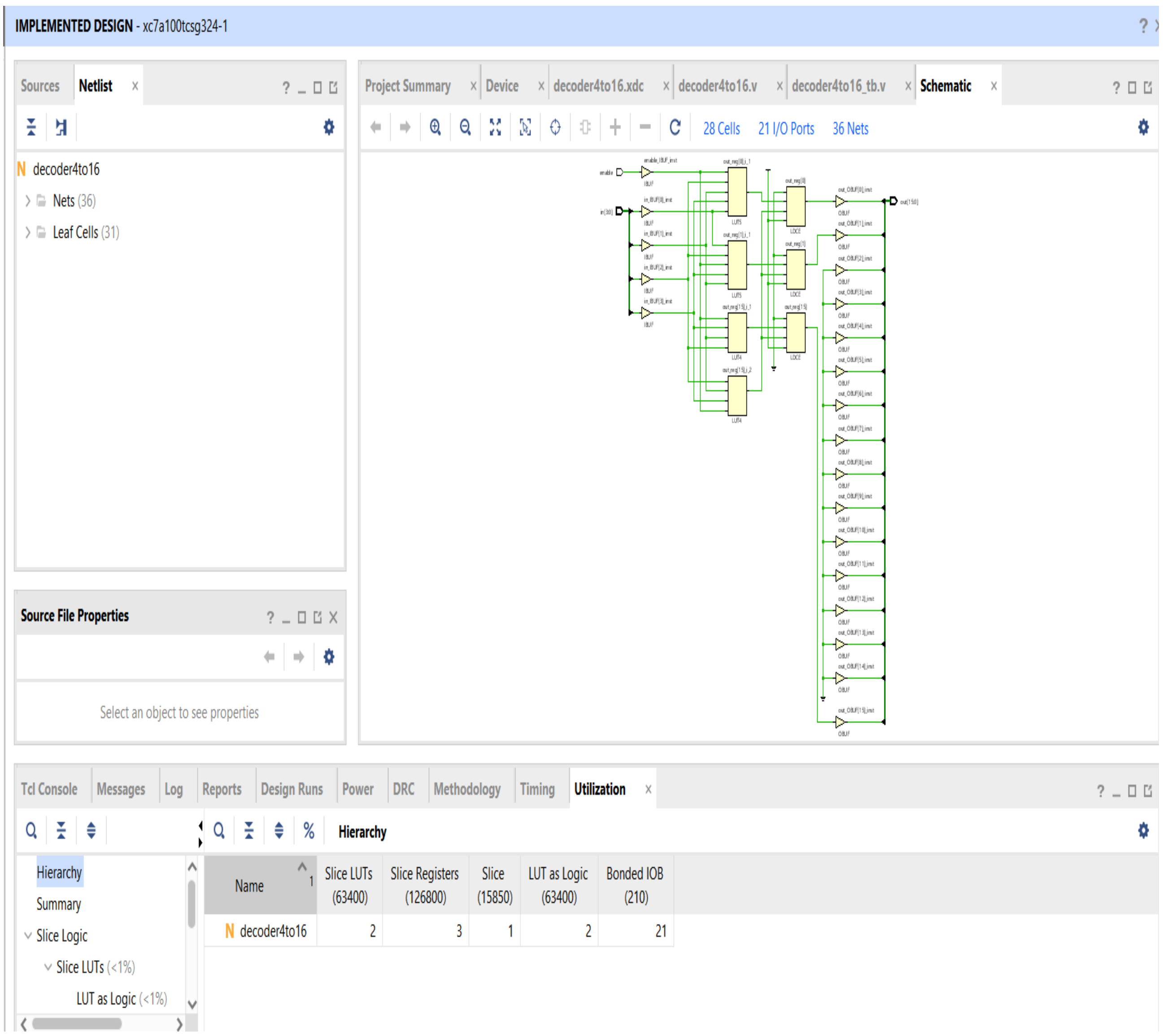
```
## LEDs

set_property -dict { PACKAGE_PIN H17   IOSTANDARD LVCMOS33 } [get_ports { out[0] }]; #IO_L18P_T2_A24_15 Sch=led[0]
set_property -dict { PACKAGE_PIN K15   IOSTANDARD LVCMOS33 } [get_ports { out[1] }]; #IO_L24P_T3_RS1_15 Sch=led[1]
set_property -dict { PACKAGE_PIN J13   IOSTANDARD LVCMOS33 } [get_ports { out[2] }]; #IO_L17N_T2_A25_15 Sch=led[2]
set_property -dict { PACKAGE_PIN N14   IOSTANDARD LVCMOS33 } [get_ports { out[3] }]; #IO_L8P_T1_D11_14 Sch=led[3]
set_property -dict { PACKAGE_PIN R18   IOSTANDARD LVCMOS33 } [get_ports { out[4] }]; #IO_L7P_T1_D09_14 Sch=led[4]
set_property -dict { PACKAGE_PIN V17   IOSTANDARD LVCMOS33 } [get_ports { out[5] }]; #IO_L18N_T2_A11_D27_14 Sch=led[5]
set_property -dict { PACKAGE_PIN U17   IOSTANDARD LVCMOS33 } [get_ports { out[6] }]; #IO_L17P_T2_A14_D30_14 Sch=led[6]
set_property -dict { PACKAGE_PIN U16   IOSTANDARD LVCMOS33 } [get_ports { out[7] }]; #IO_L18P_T2_A12_D28_14 Sch=led[7]
set_property -dict { PACKAGE_PIN V16   IOSTANDARD LVCMOS33 } [get_ports { out[8] }]; #IO_L16N_T2_A15_D31_14 Sch=led[8]
set_property -dict { PACKAGE_PIN T15   IOSTANDARD LVCMOS33 } [get_ports { out[9] }]; #IO_L14N_T2_SRCC_14 Sch=led[9]
set_property -dict { PACKAGE_PIN U14   IOSTANDARD LVCMOS33 } [get_ports { out[10] }]; #IO_L22P_T3_A05_D21_14 Sch=led[10]
set_property -dict { PACKAGE_PIN T16   IOSTANDARD LVCMOS33 } [get_ports { out[11] }]; #IO_L15N_T2_DQS_DOUT_CS0_B_14 Sch=led[11]
set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33 } [get_ports { out[12] }]; #IO_L16P_T2_CSI_B_14 Sch=led[12]
set_property -dict { PACKAGE_PIN V14   IOSTANDARD LVCMOS33 } [get_ports { out[13] }]; #IO_L22N_T3_A04_D20_14 Sch=led[13]
set_property -dict { PACKAGE_PIN V12   IOSTANDARD LVCMOS33 } [get_ports { out[14] }]; #IO_L20N_T3_A07_D23_14 Sch=led[14]
set_property -dict { PACKAGE_PIN V11   IOSTANDARD LVCMOS33 } [get_ports { out[15] }]; #IO_L21N_T3_DQS_A06_D22_14 Sch=led[15]
```

Video link:

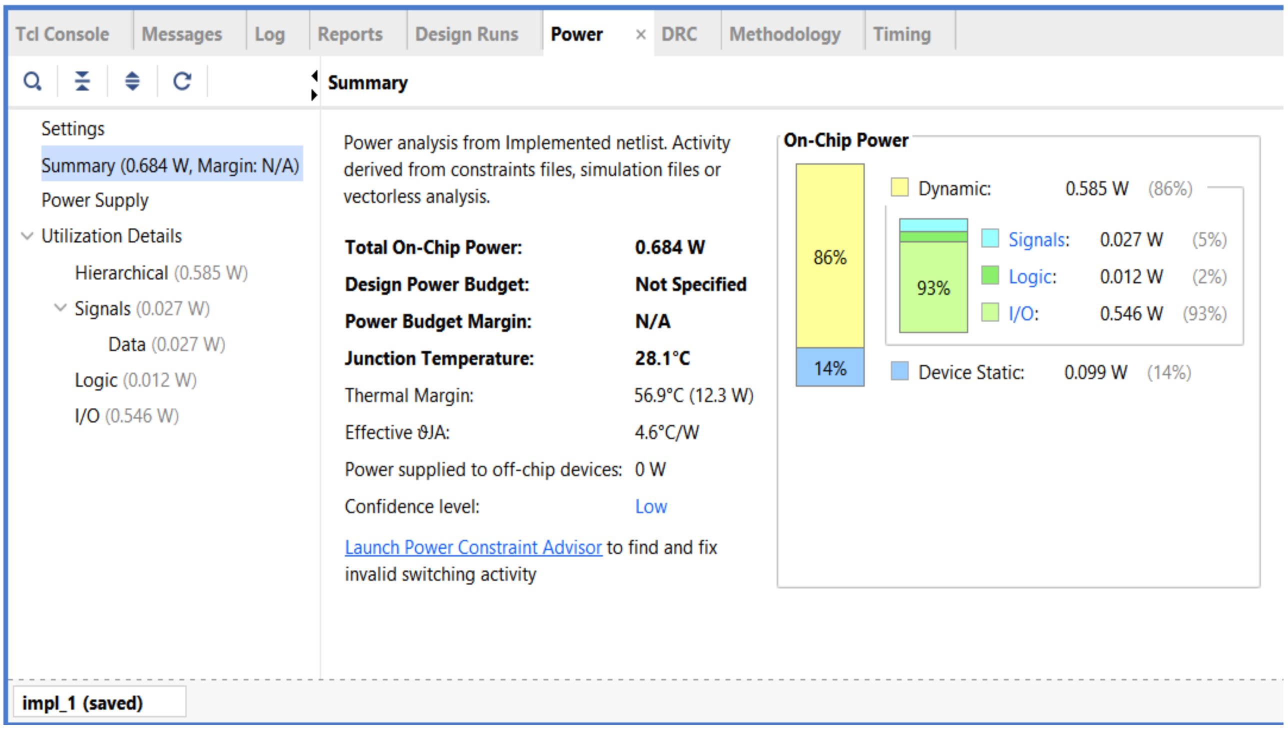
https://youtu.be/wUiby-y5p4M

or

https://drive.google.com/file/d/16Se2DlpeUQATIBwV22T7ARjVBSh67f6d/view?usp=sharing



Name	Slice LUTs (63400)	Slice Registers (126800)	Slice (15850)	LUT as Logic (63400)	Bonded IOB (210)
decoder4to16	2	3	1	2	21



Settings
Summary (0.684 W, Margin: N/A)
Power Supply
Utilization Details
Hierarchical (0.585 W)
Signals (0.027 W)
Data (0.027 W)
Logic (0.012 W)
I/O (0.546 W)

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.684 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**28.1°C**
Thermal Margin:	56.9°C (12.3 W)
Effective ϑJA:	4.6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

On-Chip Power

86%
14%
93%

	Dynamic:	0.585 W	(86%)
	Signals:	0.027 W	(5%)
	Logic:	0.012 W	(2%)
	I/O:	0.546 W	(93%)
	Device Static:	0.099 W	(14%)