İhsan Doğramacı Bilkent University Electrical and Electronics Department

EEE-102-02 Lab 2 Report

Name: Emre Şaş

ID: 22102764

Date:01.10.2023

Purpose:

Aim of this lab is to learn designing and implementing a basic combinational circuit on BASYS 3 which is a complete board, ready-to-use digital circuit development platform and has high-capacity FPGA.

Features of BASYS 3:

- 33,280 logic cells in 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops)
- 1,800 Kbits of fast block RAM
- Five clock management tiles, each with a phase-locked loop
- Internal clock speeds exceeding 450MHz
- On-chip analog-to-digital converter (XADC)
- Improved collection of ports and peripherals

Methodology:

Personal garage doors are the doors that need solid security systems since the garage is the place that cars stay in, and this place also connects the outside and the inside of a house. When these two factors are considered, I become sure that this is a problem that ought to be solved by combinational circuits. My circuit will have 4 inputs (a,b,c,d). ‘a’ will represent the signal coming from remote control (when the button is pushed it sends ‘1’ if not ‘0’). ‘b’ will represent the signal coming from motion detection sensor (if sensor

detects a motion it sends ‘1’ if not ‘0’). ‘c’ will represent signal that is sent by device recognizing one’s car plate (if sensor recognizes the plate it sends ‘1’ if not ‘0’) and finally ‘d’ will represent the lock that requires password chosen by the user (if the password is correct signal ‘1’ is sent otherwise ‘0’ is sent). Basically, there are 3 ways to open the door:

1- Going towards the garage with a car whose plate saved in database.
2- Pushing the remote control’s button.
3- Going towards the garage with a random car and entering the password.

Steps:

- Before coding we should draw the truth table (Table 1). As is seen in the truth table, we included all the possible scenarios and according to that we should derive an logic function.
- Attain inputs and outputs and write the design code (Code 1) according to these variables. After designing check the schematic of the code (Figure 1.1)
    - A=remote control signal
    - B=motion detector signal
    - C= plate recognizing device signal
    - D= password signal
    - Out1= outcome signal
- After the design phase is finished, we need to create a testbench to simulate what we have done in the design section and control if the simulated outcomes match with the scenarios (Code 2). Check the waveform (Figure 1.2).
- Add constraints to determine which input will be represented by component of the board and attain a LED to out1 see the result.
- As a last step we need to create a bitstream code and program our BASYS 3. After programming we should try all combinations to be sure that our experiment was flawless. (Figure 1.3 to Figure 1.18)

a	b	c	d	out
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Table 1

This table will help us to derive the suitable logic function for a problem. Although the first equation we get from truth table is fairly long and complex: a'bc'd + a'bcd' + a'bcd+ ab'c'd'+ ab'c'd+ ab'cd'+ ab'cd+ abc'd+ abc'd+ abcd'+ abcd . This function can be simplified to a + b(c+d).

The input variables a, b, c, and d were initialized in the PORT function before the BEGIN label was used to write the VHDL code in Vivado. Both the input and output types are STD_LOGIC. The output was given to the logic element "out1". The testbench code was then created . The code for "tester.vhd" was added to code for the test bench.Before using the PORT MAP function "UUT" is used to get the base design for testing. While testing our desing used time periods of 40 nanoseconds at the start of every time period I tested different combination of a, b, c, and d (6 different combinations were tested) . Finally the constraint files were created according to pins and their codes (can be found in Vivado and BASYS3 related documents on Moodle) .

a: V17 Switch

b: V16 Switch

c: W16 Switch

d: W17 Switch

f: U16 LED

Answer To the Questions :

How does one specify the inputs and outputs of a module in VHDL?
Inputs and outputs can be specified by using a function called PORT and writing the names and types of the data (std_logic). These inputs and outputs can also be added at the very beginning of adding design sources.

• How does one use a module inside another code/module? What does PORT MAP do?
One can define a module within another module by using component code before begin and after architecture. The purpose of ports and a PORT MAP is to make it possible to create a standalone entity that acts as a design source and then use a port map to instantiate that entity as much as you'd like in any region of your design. In fact, Component function is like a LEGO piece manufacturer which can be used for this purpose.

• What is a constraint file? How does it relate your code to the pins on your FPGA?
o Constraint file is a completion of set of documents which contains information about the pins of the board. These files are used to assign virtual inputs and outputs to analog components of the FPGA board.

• What is the purpose of writing a testbench?
o The purpose of writing testbench is simulating the code on virtual environment and analyzing the possible scenarios before programming the device.

Conclusion:

The goal of the lab was to use Vivado to create a combinational circuit in VHDL. Truth tables, minimizing rules using the Boolean algebra, minimization rules, waveforms, and VHDL codes were all employed in the circuit design process. The outcomes from BASYS 3 were in line with what I had anticipated.

Appendices:

Code 1:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity tester is
   Port ( a : in STD_LOGIC;
          b : in STD_LOGIC;
          d : in std_logic;
          out1 : out STD_LOGIC;
          c : in std_logic);
end tester;

architecture Behavioral of tester is

begin
out1<= a or (b and( c or d));

end Behavioral;
```

Code 2:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;
entity testBench1 is
end testBench1;
architecture Behavioral of testBench1 is
COMPONENT tester
PORT( a : IN STD_LOGIC;
b : IN STD_LOGIC;
c : IN STD_LOGIC;
d : IN STD_LOGIC;
out1 : OUT STD_LOGIC);
END COMPONENT;
SIGNAL a : STD_LOGIC;
SIGNAL b : STD_LOGIC;
SIGNAL c : STD_LOGIC;
SIGNAL d : STD_LOGIC;
SIGNAL out1 : STD_LOGIC;
BEGIN
UUT: tester PORT MAP(
a => a,
b => b,
c => c,
```

```
d => d,
out1 => out1
);
testBench1 : PROCESS
BEGIN
wait for 40 ns;
a<='0';
b<='0';
c<='0';
d<='0';
wait for 40 ns;
a<='1';
b<='1';
c<='1';
d<='1';
wait for 40 ns;
a<='0';
b<='0';
c<='1';
d<='0';
wait for 40 ns;
a<='0';
b<='1';
c<='1';
d<='0';
wait for 40 ns;
a<='1';
b<='1';
c<='1';
d<='0';
wait for 40 ns;
a<='0';
b<='0';
c<='0';
```

```
d<='1';

END PROCESS;
end Behavioral;
```

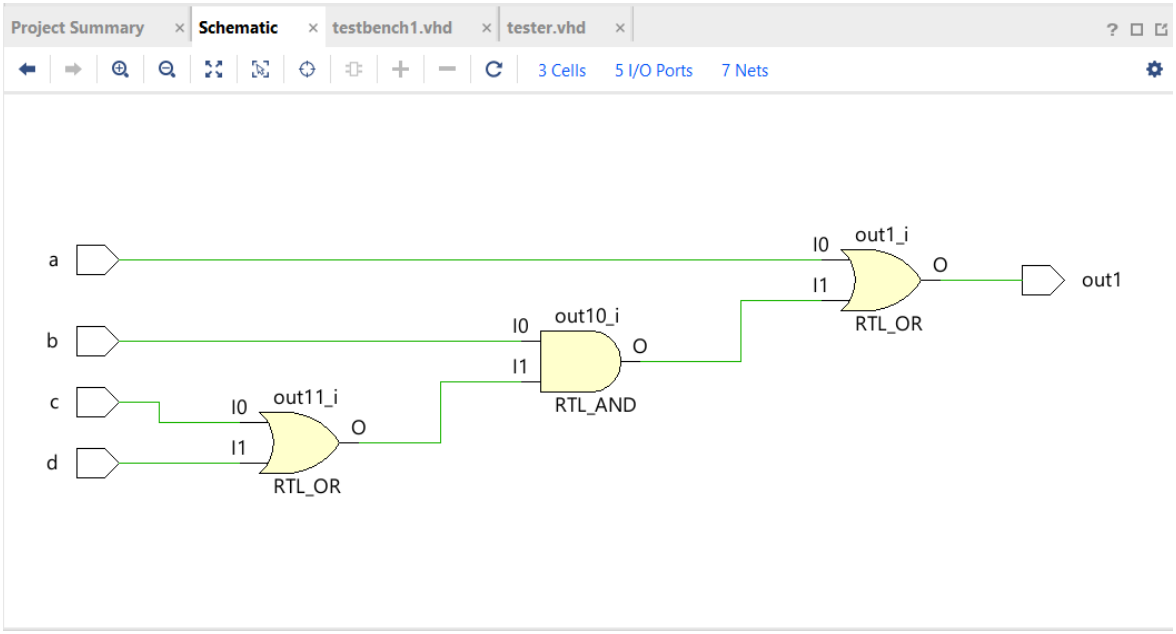


Figure 1.1

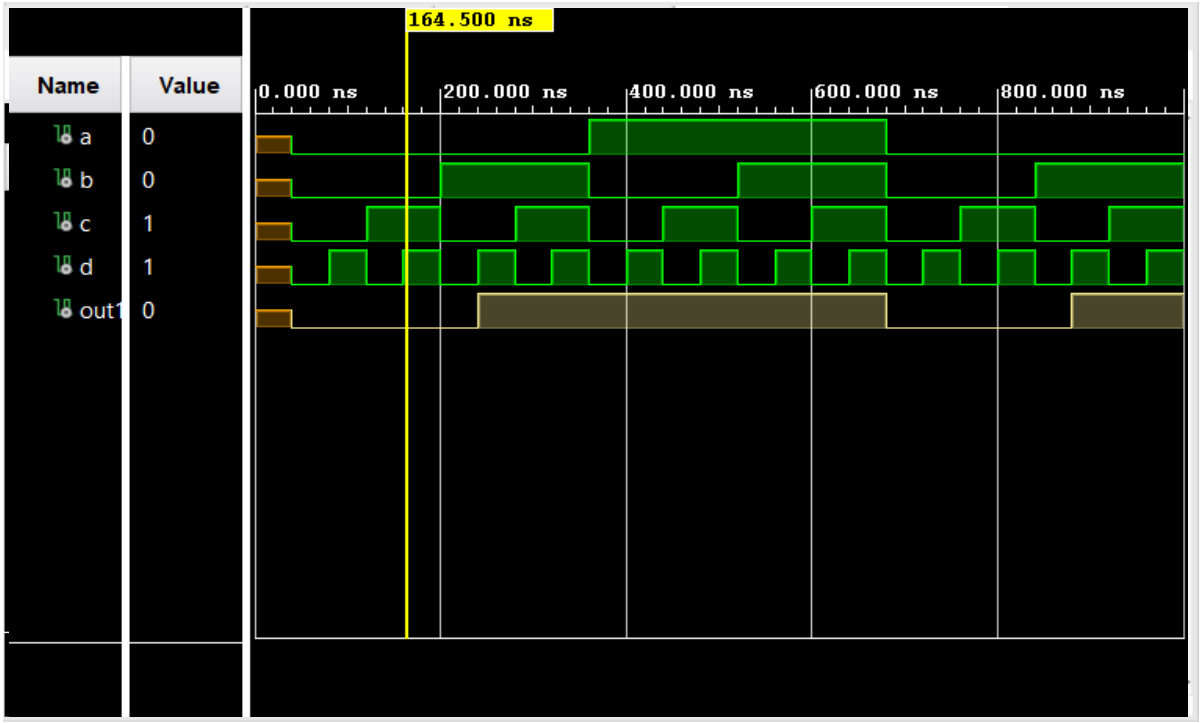
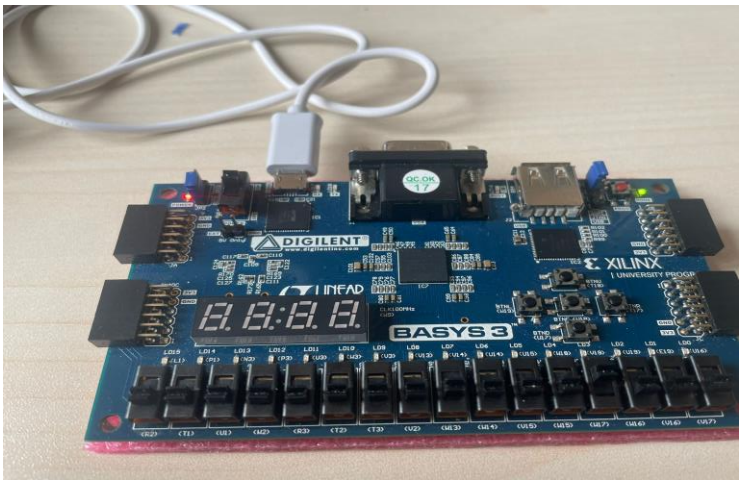
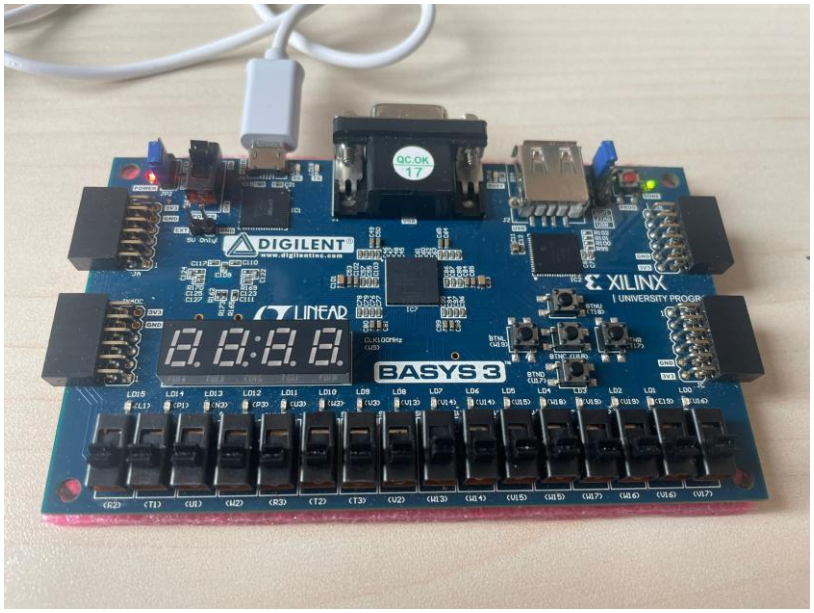


Figure 1.2



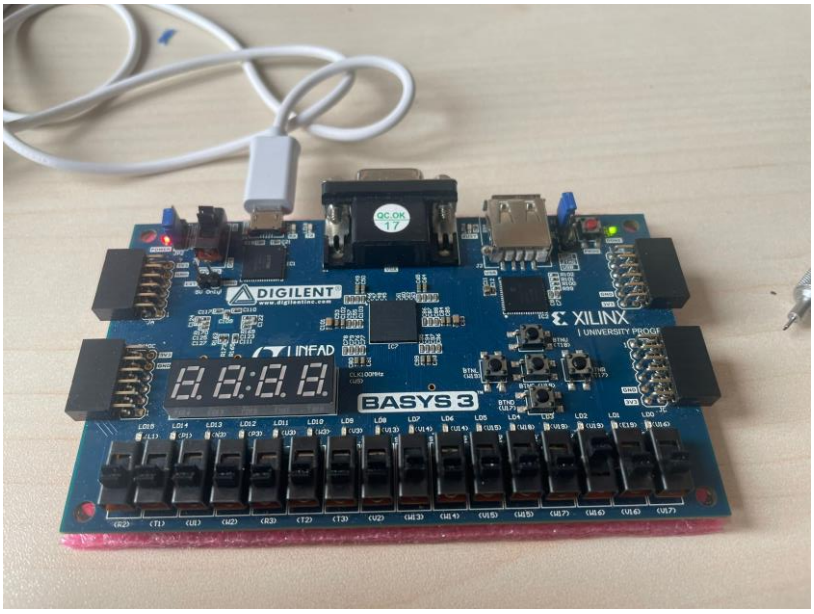
a=0 b=0 c=0 d=1

Figure 1.3



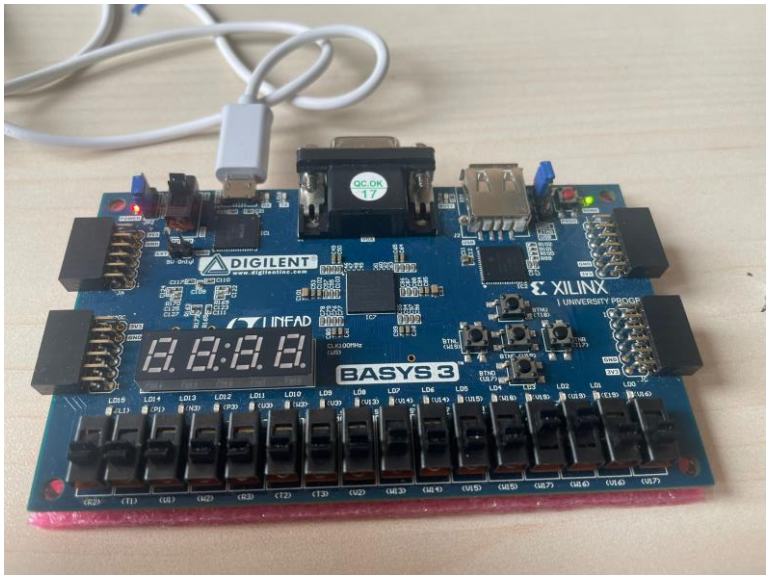
a=0 b=0 c=0 d=0

Figure 1.4



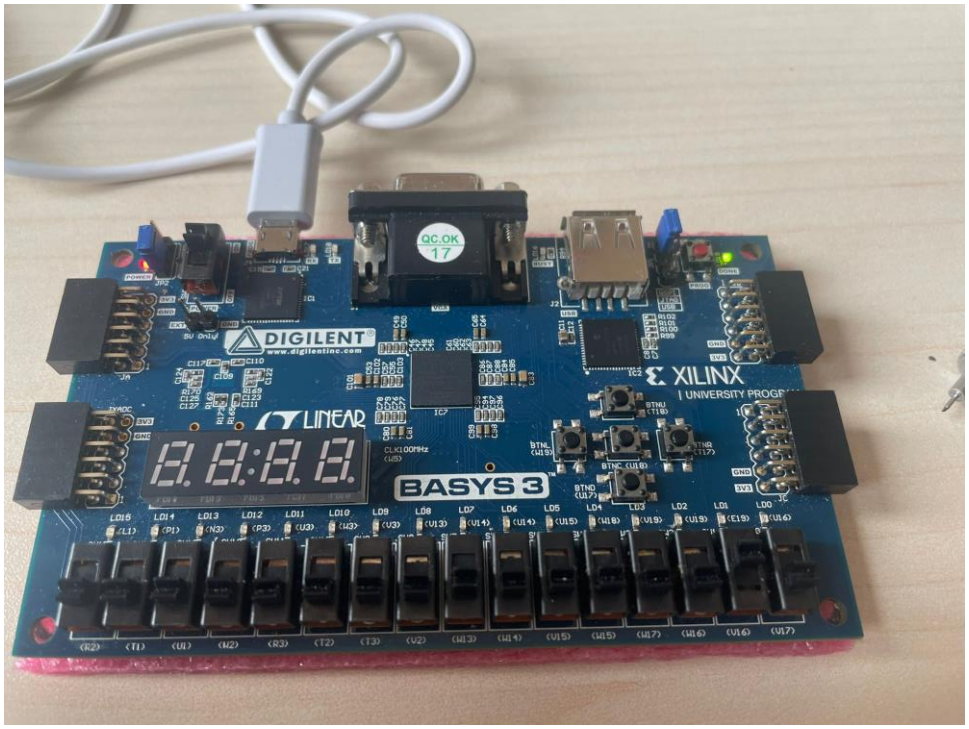
a=0 b=0 c=1 d=0

Figure 1.5



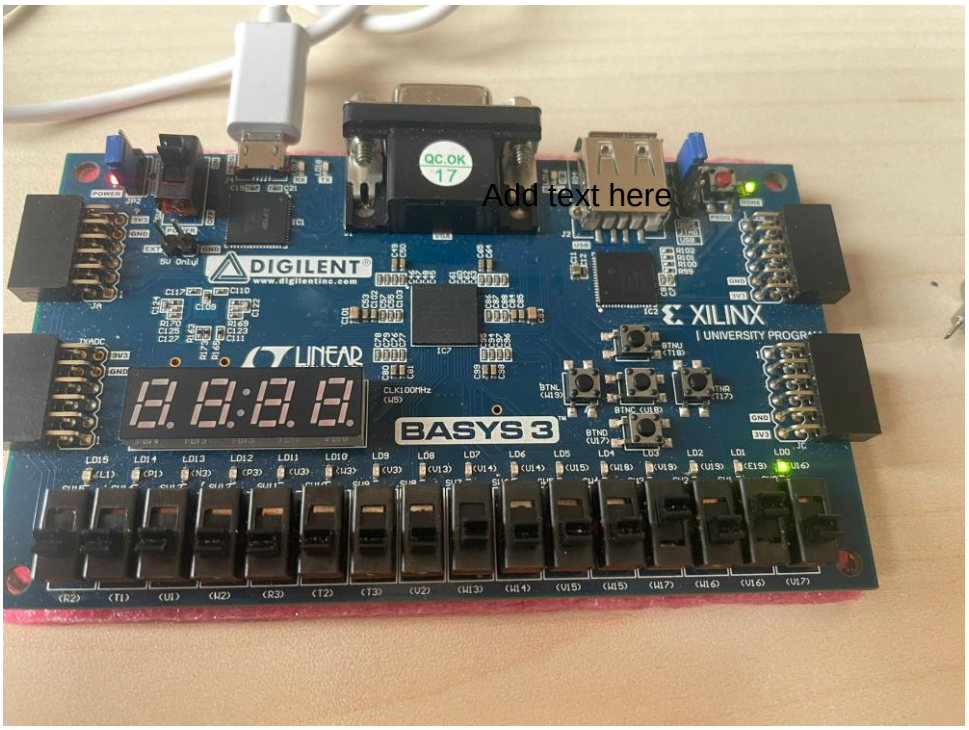
a=1 b=1 c=0 d=0

Figure 1.6



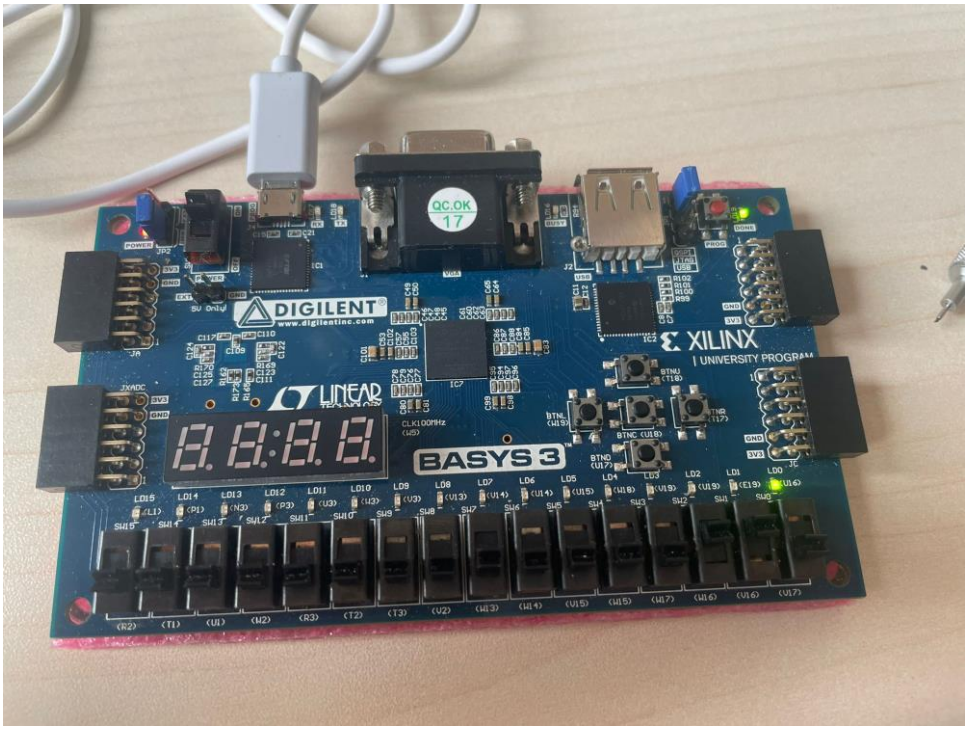
a=0 b=1 c=0 d=0

Figure 1.7



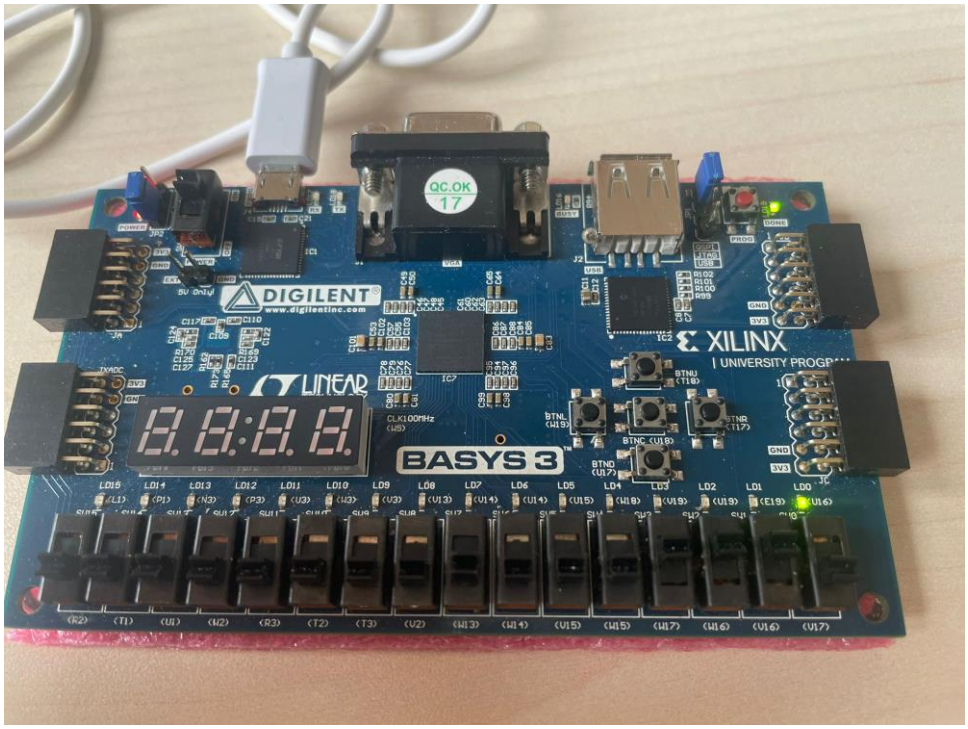
a=0 b=1 c=0 d=1

Figure 1.8



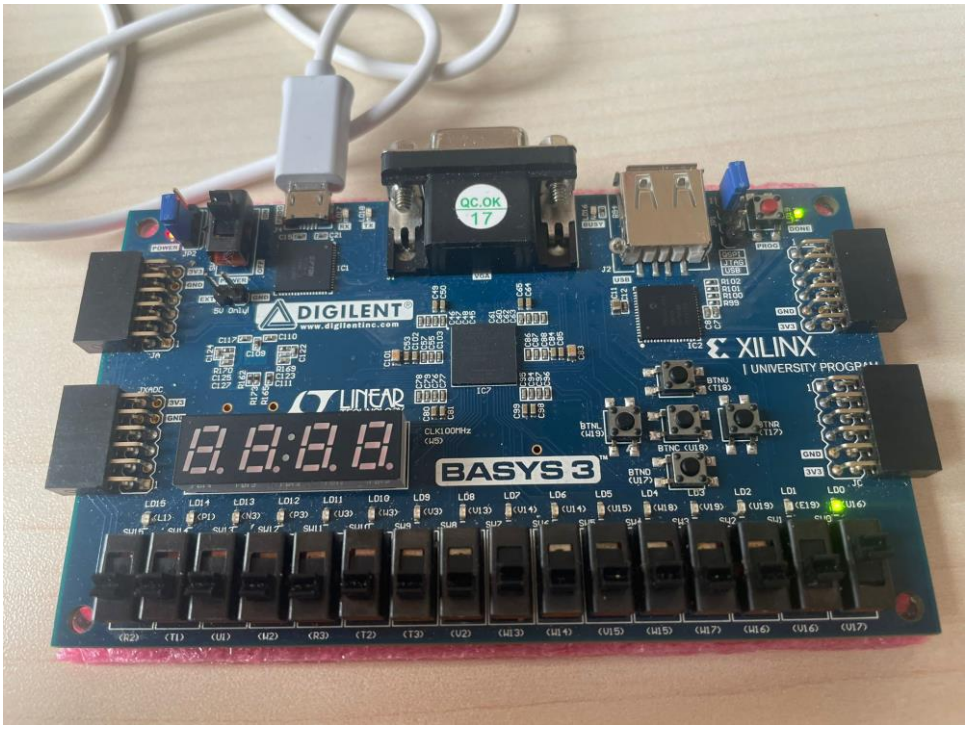
a=0 b=1 c=1 d=0

Figure 1.9



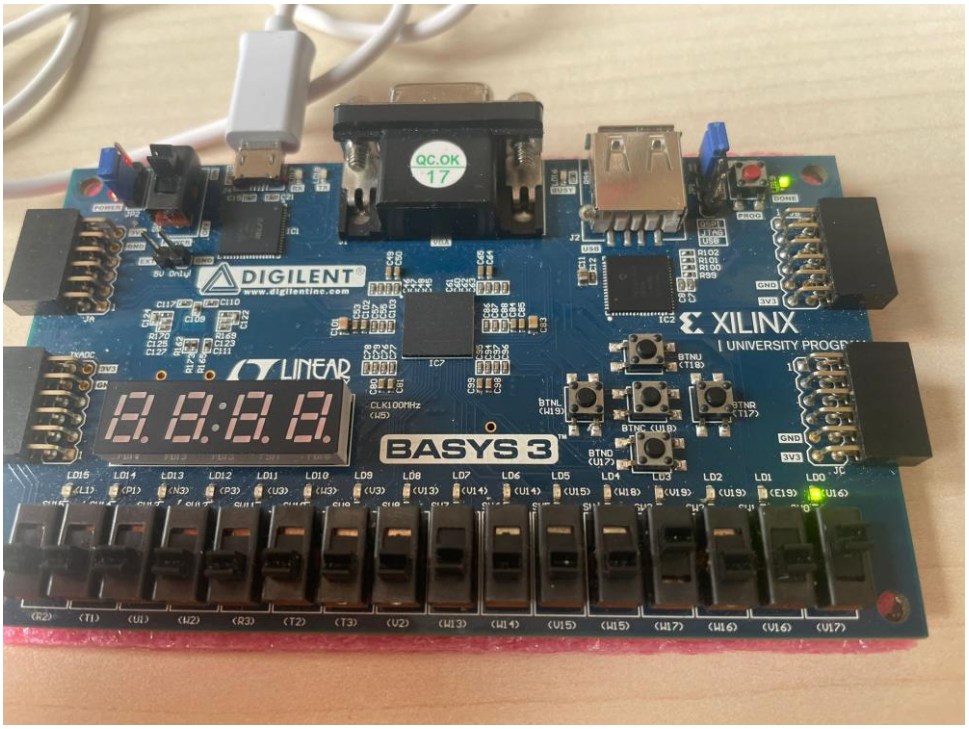
a=0 b=1 c=1 d=1

Figure 1.10



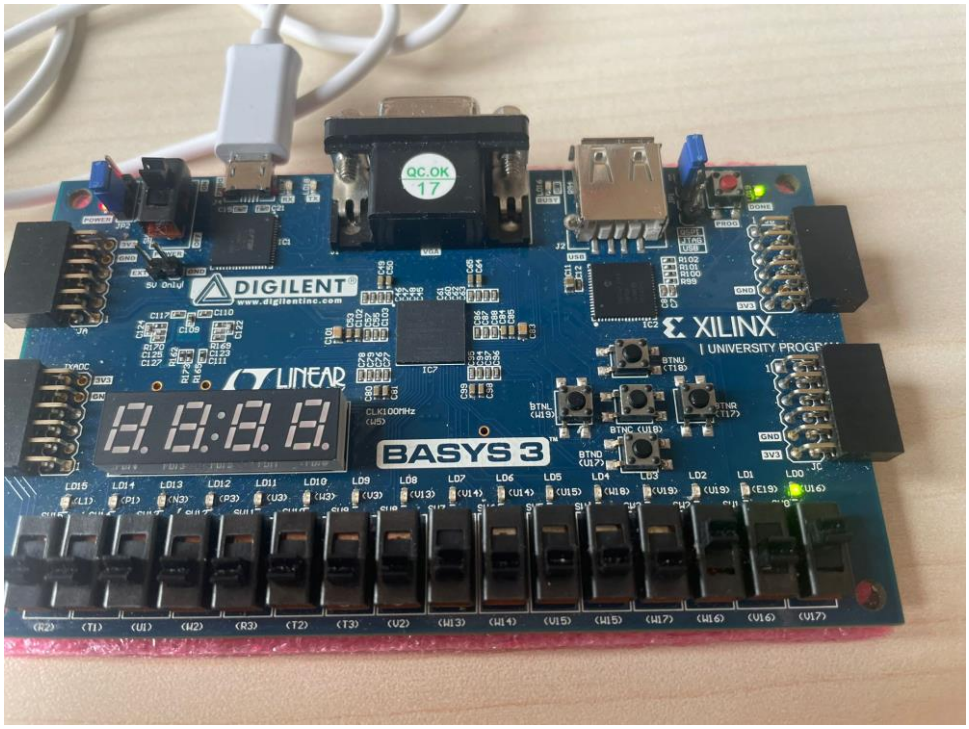
a=1 b=0 c=0 d=0

Figure 1.11



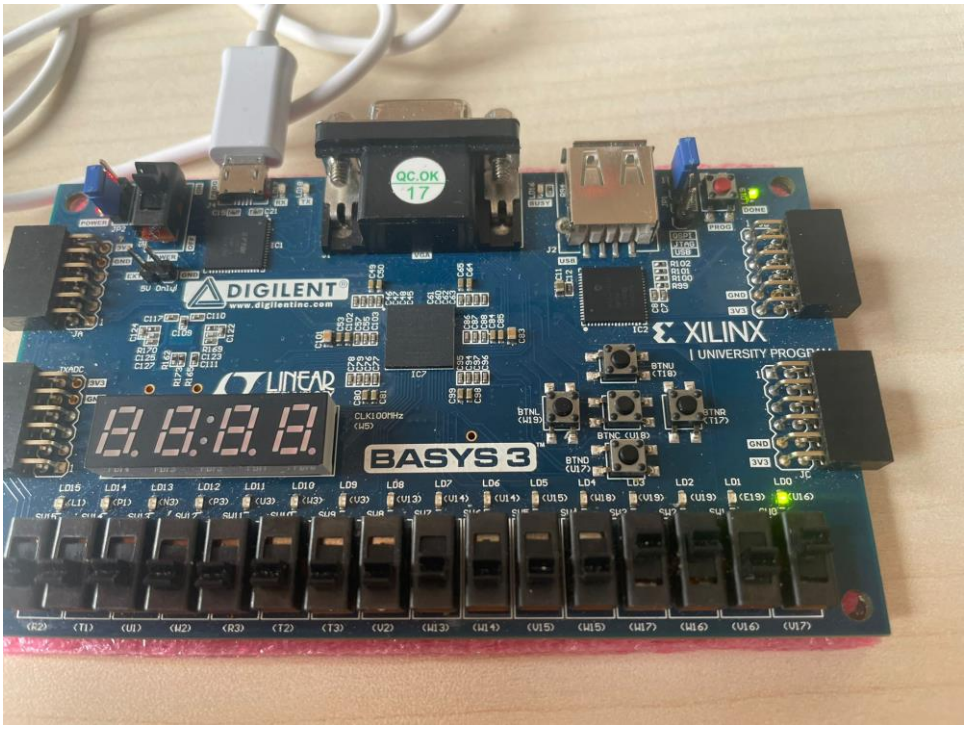
a=1 b=0 c=0 d=1

Figure 1.12



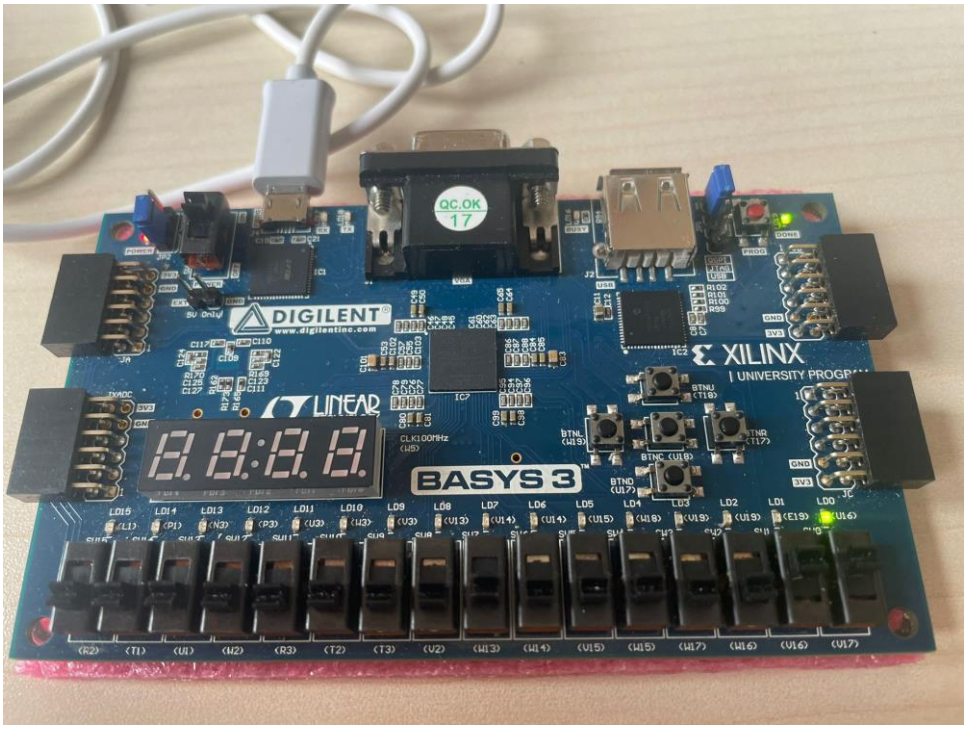
a=1 b=0 c=1 d=0

Figure 1.13



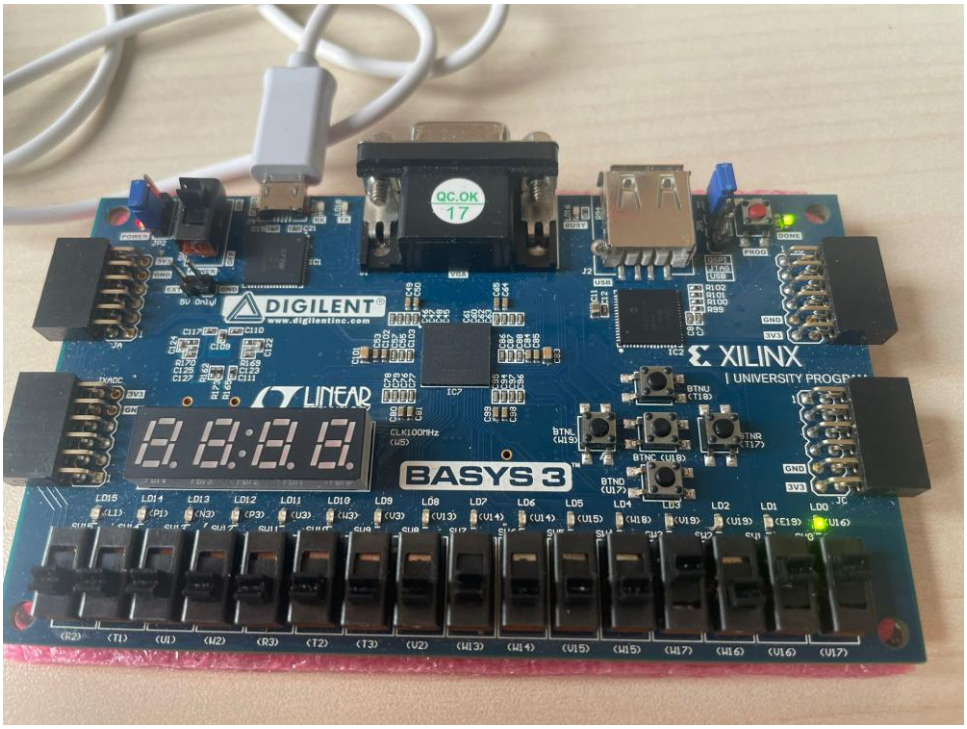
a=1 b=0 c=1 d=1

Figure 1.14



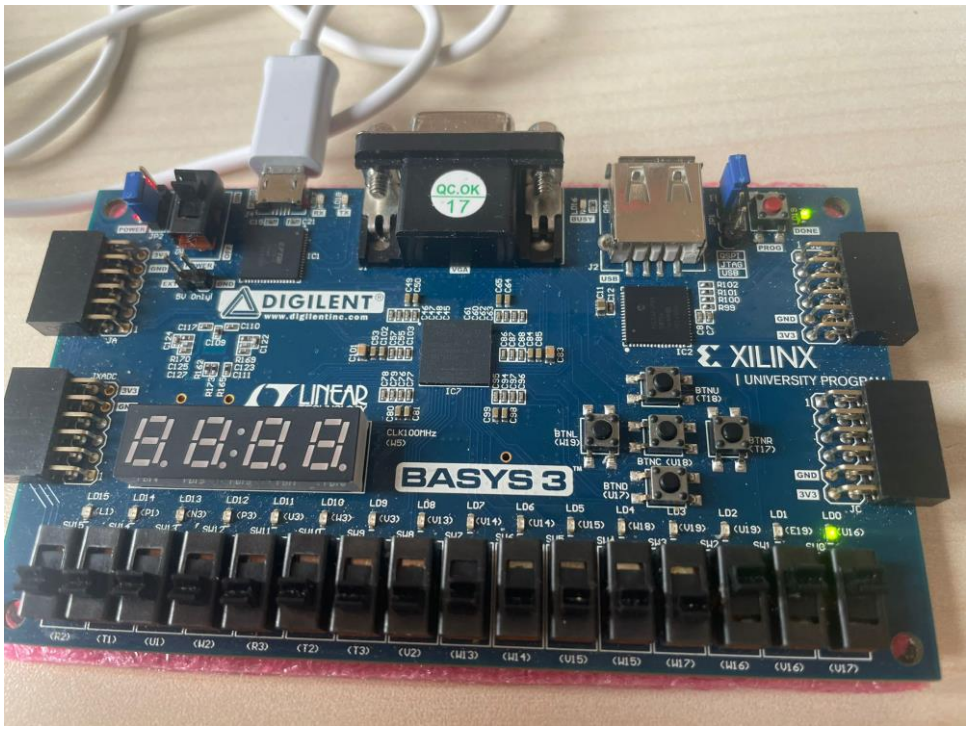
a=1 b=1 c=0 d=0

Figure 1.15



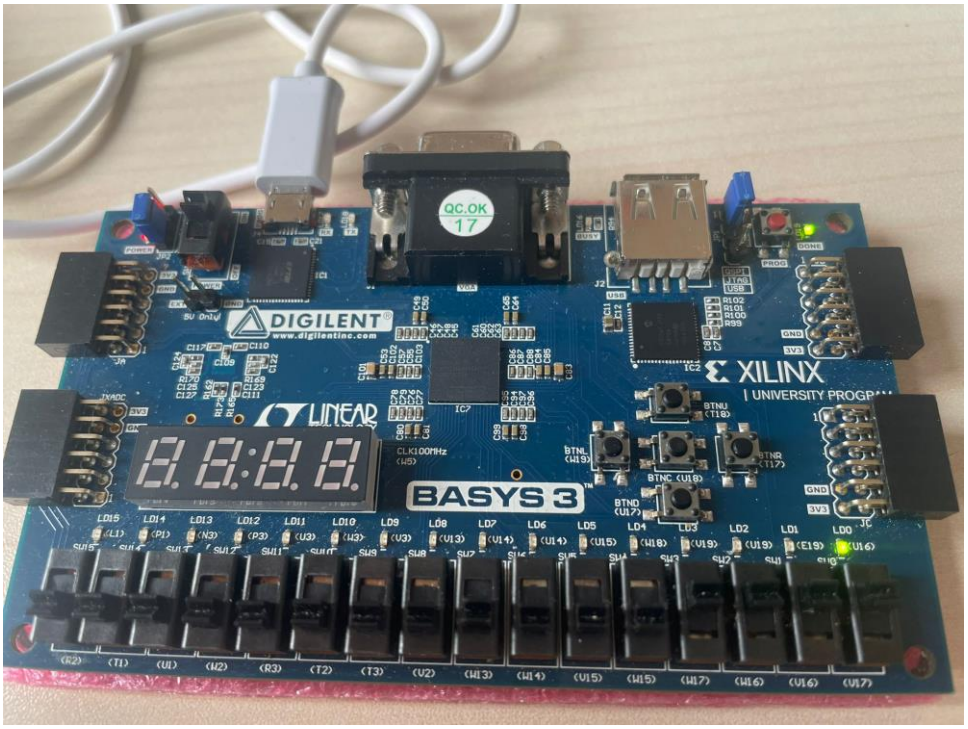
a=1 b=1 c=0 d=1

Figure 1.16



a=1 b=1 c=1 d=0

Figure 1.17



a=1 b=1 c=1 d=1

Figure 1.18