Workshop on Programmable Logic Devices

Łukasz Sajewski

Tasks Part-2

Burak ELHAMAN

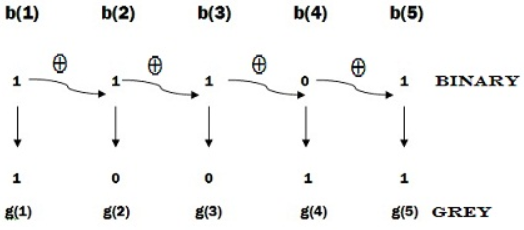
**Why We Use Gray Code**

It is important to quickly understand the error and not to fail with electronic systems. Therefore we use gray code. Since there is only 1 bit change, there is a chance of making a mistake and when there is a 2 bit change, we understand that there is a mistake. For example in binnary code when number pass from 7 to 8 (0111 to 1000) 4 bit changes however, this is not the case in gray.

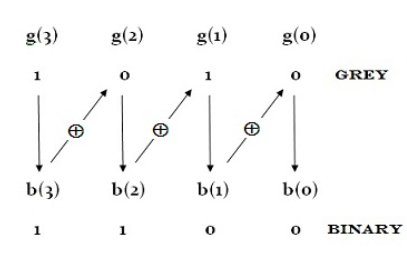
Gray codes are widely used to facilitate error correction in digital communications such as digital terrestrial television and some cable TV systems.How We Change Gray and Binnary code.

**How We Convert Gray Code And Binary Code Among Themselves**

Let's look at how to convert from gray to binary.Firstly the first bit down without changing. After that we are make XOR 2. bit with 1. bit. So if these same we write 1 but not write 0. We writing the result under 2.bit.



Gray to binary is doing like that. We bring the first bit down again. after that we make XOR 2.bit with we download bit. Than write result next to it. We repeat this process to the end.



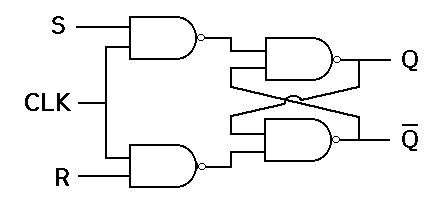
**What is Flip-Flop**

As long as the circuit is working, the structures that protect the output state are called Flip-Flops. Flip-Flops form the basic memory unit. They can store 1 bit data. There are two output in Flip-Flops (Q and Q’). The input and the trigger must change to change the output. Protects the output position if the trigger does not change.

In Flip-Flops, triggering is done by a square wave signal called clock pulse (CP). There are majorly 4 types of flip flops.

**SR Flip-Flop**

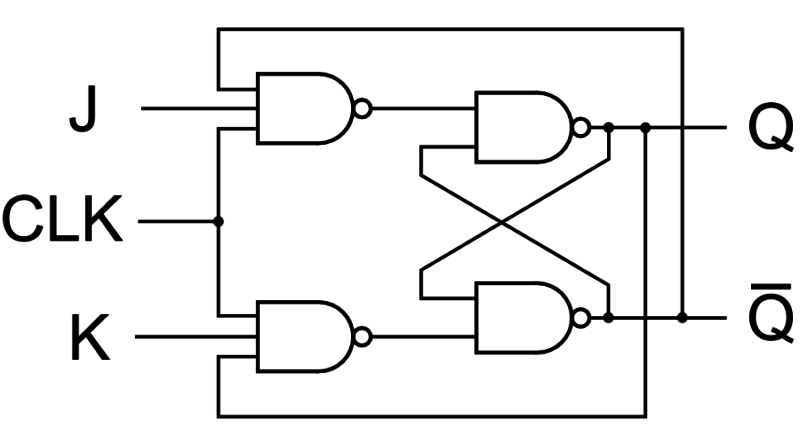
|  |  |  |
| --- | --- | --- |
| **S** | **R** | **Q(t+1)** |
| 0 | 0 | Q |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | ∞ |

****

**JK Flip-Flop**

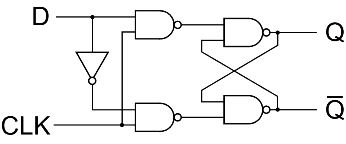
SR Flip-Flop is undefined in 1-1. Therefore another flip flop is needed. It is JK.

|  |  |  |
| --- | --- | --- |
| **J** | **K** | **Q(t+1)** |
| 0 | 0 | Q |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | Q’ |



**D Flip-Flop**

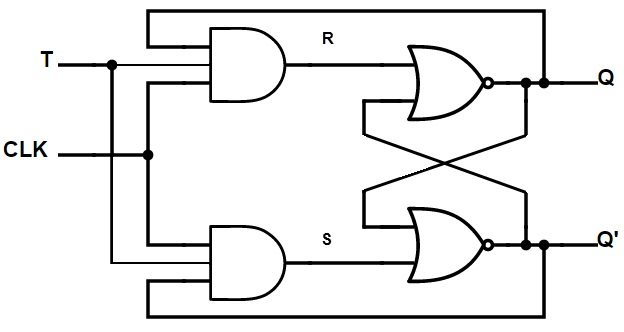
D flip flop is a better alternative that is very popular with digital electronics. They are commonly used for counters and shift-registers and input synchronisation.

****

|  |  |
| --- | --- |
| **D** | **Q(t+1)** |
| 0 | 0 |
| 1 | 1 |

**T Flip-Flop**

A T flip flop is like JK flip-flop. These are basically a single input version of JK flip flop. This modified form of JK flip-flop is obtained by connecting both inputs J and K together. This flip-flop has only one input along with the clock input. These flip-flops are called T flip-flops because of their ability to complement its state Toggle, therefore name Toggle flip-flop.

****

|  |  |
| --- | --- |
| **T** | **Q(t+1)** |
| 0 | Q |
| 1 | Q’ |

**Applications of Flip-Flops**

These are the various types of flip-flops being used in digital electronic circuits and the applications of Flip-flops are as specified below.

-Counters

-Frequency Dividers

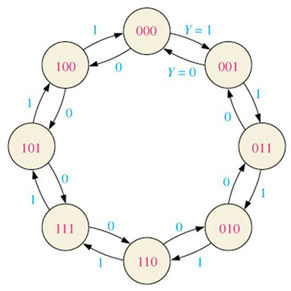
-Shift Registers

-Storage Registers

# Introduction

In this project, we will design a gray code counter which can count up/down 3 bits with flip-flops. Firstly we should understand our problem. What it want to us. It wants us to design a counter that can count with flip-flops and gray. We need to know our states. We have 8 states. We also have X input for up/down count .we need to use 3 flip-flop because we have 8 states.

**State Diagram**

The state diagram is very important at the beginning of a problem. My professors wanted us to give importance to this part.

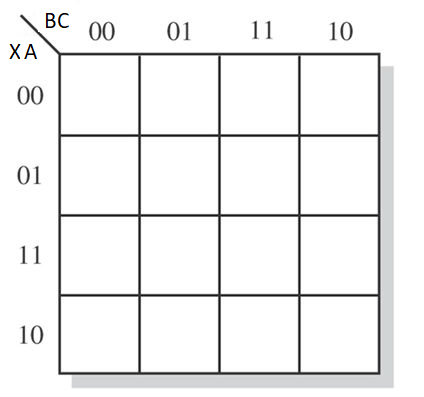
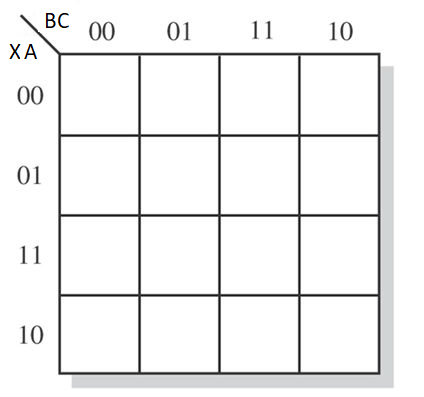
**Truth Table**

After the state diagram, we can learn our circuit by using the truth table and karnaugh map. It gives our inputs according to the desired output values.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Present State** | | | | | **Next State** | | | |
| **X** | **A** | **B** | **C** |  | | **D3** | **D2** | **D1** |
| 0 | 0 | 0 | 0 |  | | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 |  | | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |  | | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |  | | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 |  | | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 |  | | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 |  | | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 |  | | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |  | | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 |  | | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 |  | | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |  | | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |  | | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 |  | | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |  | | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |  | | 1 | 0 | 1 |

**Karnaugh Maps**

In D flip-flop the same next state and inputs. D tipi flipflopta next state ile flıp flop ınputs aynıdır. Because whatever we give to the input, we get the same on the output.



0 0 0 1

0 1 1 1

0 0 0 1

0 1 1 1

1 0 0 0

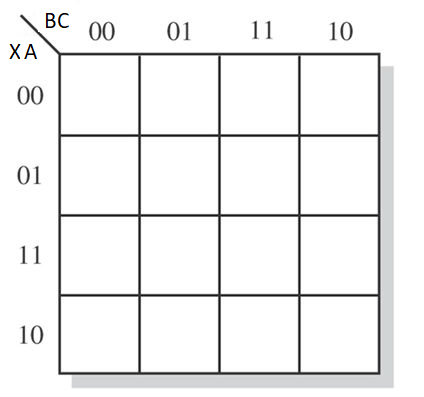
1 1 1 0

0 1 1 1

0 0 0 1

D2 = BC’ + X’AC + XA’C

D3 = X’B’C’ + AC + XBC’



0 0 1 1

1 1 0 0

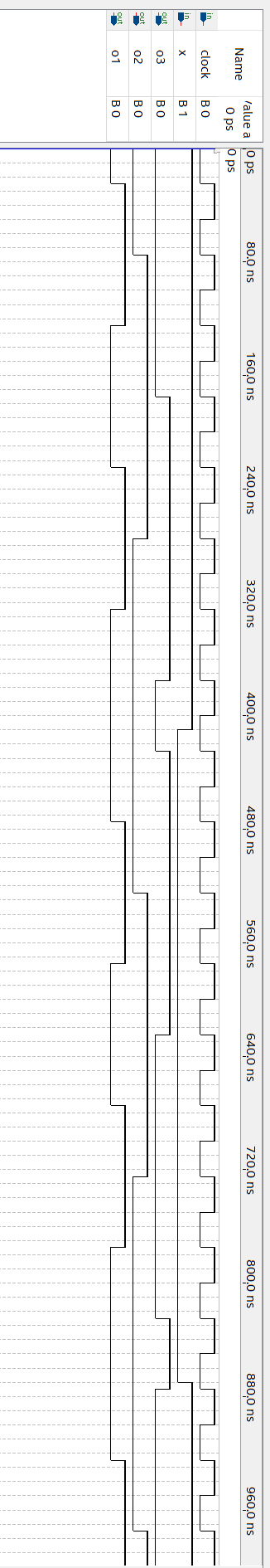
0 0 1 1

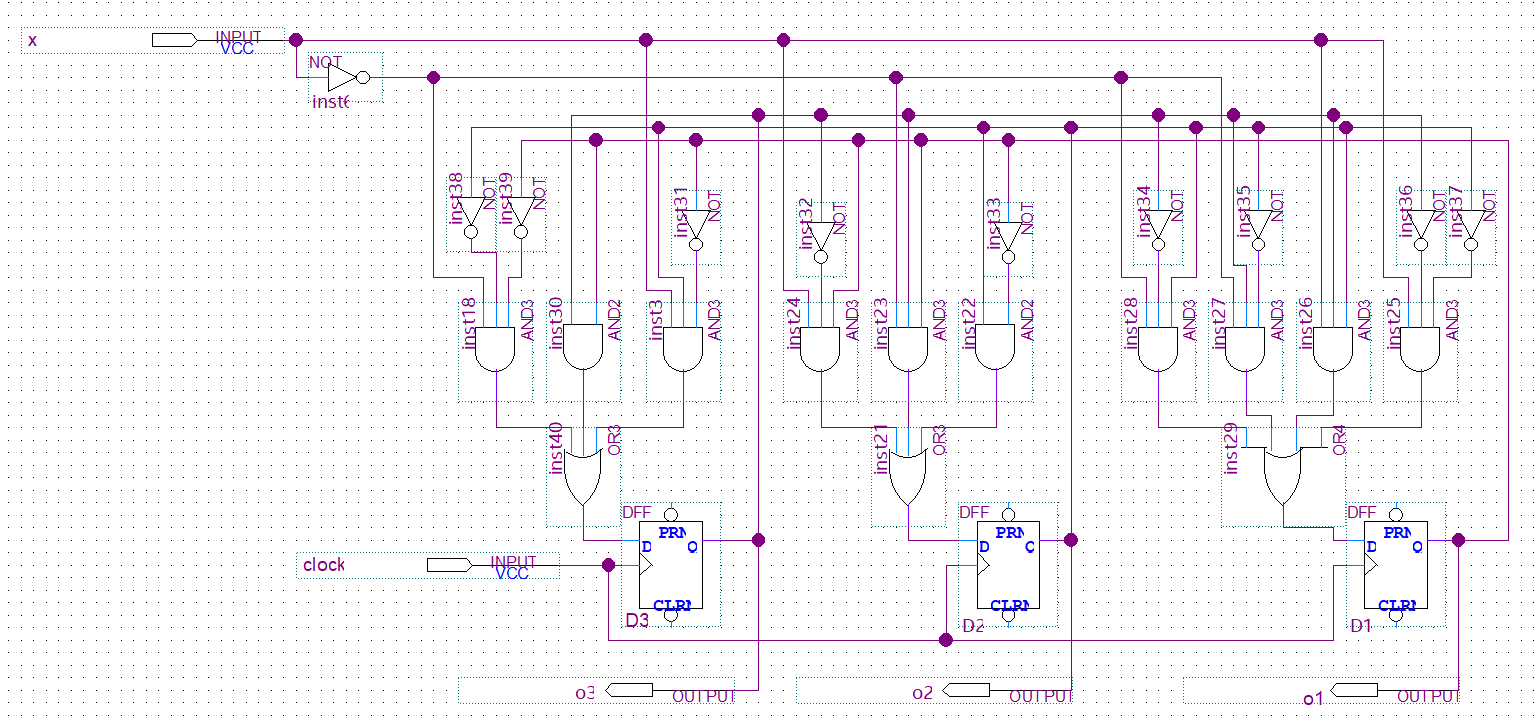
1 1 0 0

D1 = X’A’C + X’AC’ + XAC + XA’C’

Once we have found our inputs, we can draw our circuit.

GATE REALIZATION AND ANALYSIS





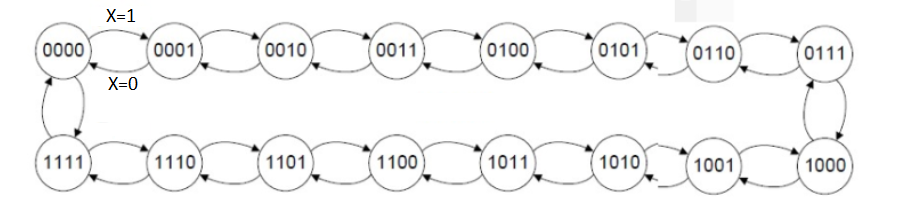
Our circuit runs smoothly. With the D flip flop, it was enough to use only 10 and gates. JK need 16 gates, and T need min. 10 gates. And Gates are common all of them. However, when I used a T type, the circuit was complex and therefore I decided to use D. As a result, only change 1 bit we can do what we want.

**Indroduction**

We will design a 4 bit counter using a flip-flop. X will use for up/down again. For Hold Function I anded clock and H input. İf H equal 1 count will stop. Because these flip-flops are synchronous. And we will use 7 segment display out of counter.

**State Diagram**

Firstly we should know our states. We have 16 states. We will use 4 flip flop. Also we have X. Therefore we will use 5 element karnaugh map.



**Truth Table**

Thanks to the Truth table, we can learn our inputs with our flip flops.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **X=0** | | | | | | | | | |
| **Present State** | | | | | **Next State** | | | | |
| **A** | **B** | **C** | **D** |  | | **D3** | **D2** | **D1** | **D0** |
| 0 | 0 | 0 | 0 |  | | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 |  | | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |  | | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |  | | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |  | | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 |  | | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |  | | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |  | | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |  | | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |  | | 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |  | | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |  | | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |  | | 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |  | | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |  | | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |  | | 1 | 1 | 1 | 0 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **X=1** | | | | | | | | | |
| **Present State** | | | | | **Next State** | | | | |
| **A** | **B** | **C** | **D** |  | | **D3** | **D2** | **D1** | **D0** |
| 0 | 0 | 0 | 0 |  | | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 |  | | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 |  | | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 |  | | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 |  | | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 |  | | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 |  | | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 |  | | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |  | | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 |  | | 1 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 |  | | 1 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 |  | | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |  | | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 |  | | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 |  | | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |  | | 0 | 0 | 0 | 0 |

**Karnaugh Map**

 **X=0 X=1**

1 0 0 0

0 0 0 0

1 1 1 1

0 1 1 1

0 0 0 0

0 0 1 0

1 1 0 1

1 1 1 1

D3 = AB’D + ABC’ + ACD’ + X’A’B’C’D’ + X’AB + XAB’ + XA’BCD



D1 = X’C’D ’ +X’CD + XC’D + XCD’

0 1 0 1

0 1 0 1

0 1 0 1

0 1 0 1

1 0 1 0

1 0 1 0

1 0 1 0

1 0 1 0

0 0 1 0

1 1 0 1

1 1 0 1

0 0 1 0

1 0 0 0

0 1 1 1

0 1 1 1

1 0 0 0

D2 = BC’D + BCD’ + X’B’C’D’ + X’BC + XB’CD + XBC’



1 0 0 1

1 0 0 1

1 0 0 1

1 0 0 1

1 0 0 1

1 0 0 1

1 0 0 1

1 0 0 1

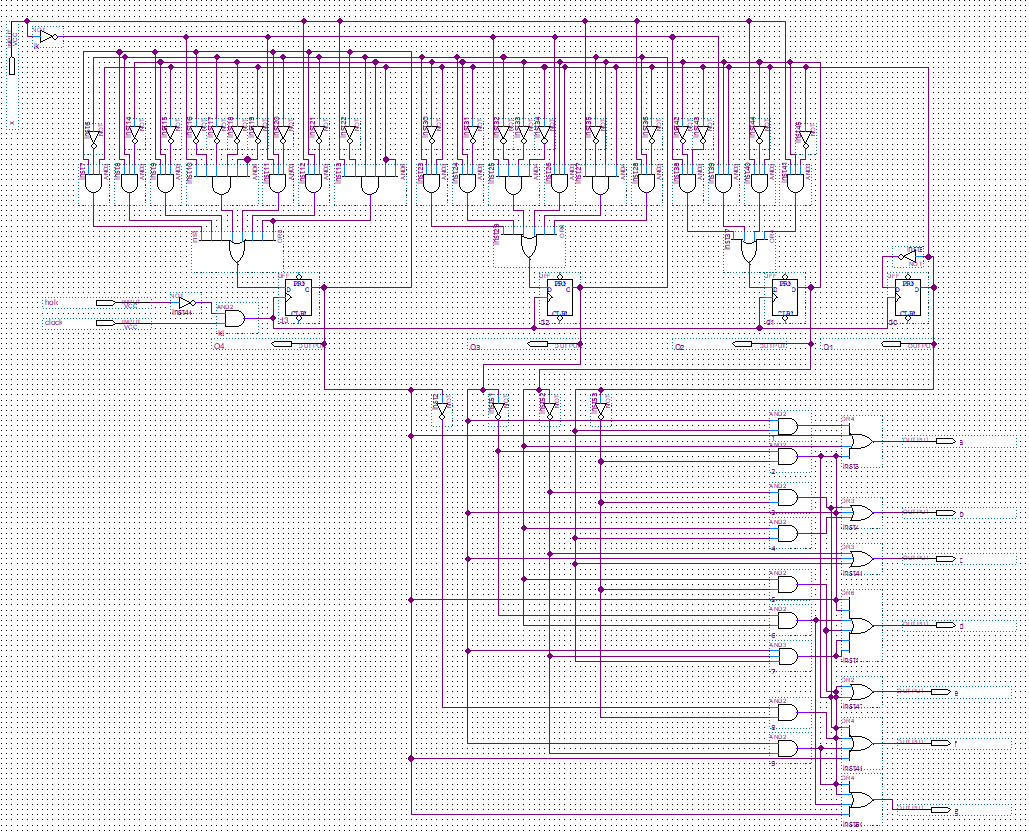
D0 = D’

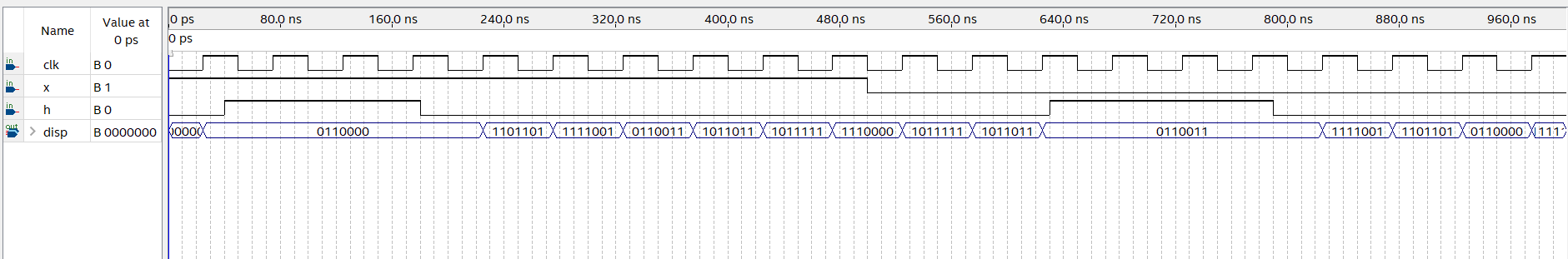
When the Karnaugh map is solved, the part that is common between X = 0 and X = 1 is written and X is eliminated. After that other parts will add.

**Verilog Code**

****

GATE REALIZATION AND ANALYSIS



****