Workshop on Programmable Logic Devices

Łukasz Sajewski

Tasks Part-2

Burak ELHAMAN

# Introduction

Bu projemizde 3 bitlik aşağı yukarı sayabilen bir gray code sayıcıyı flip-floplar yardımı ile tasarlayacağız.

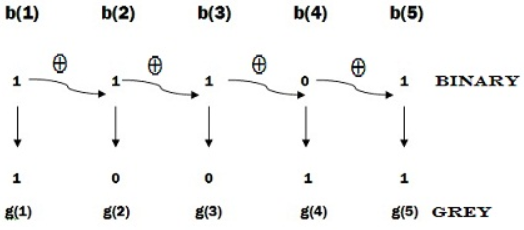
**Why We Use Gray Code**

Elektronik sistemlerde hata yaşamamak ve hatayı çabucak anlamak önemlidir. Bu nedenle gray code kullanırız. Sadece 1 bit değişimi olduğu için hata yaşama olasılığı düşürülür ve 2 bit değişimi görüldüğünde bir hatamızın var olduğunu anlarız. Örneğin binnaryde 7 den 8 e geçerken (0111 den 1000) 4 bit birden değişir ancak grayde bu sözkonusu değildir.

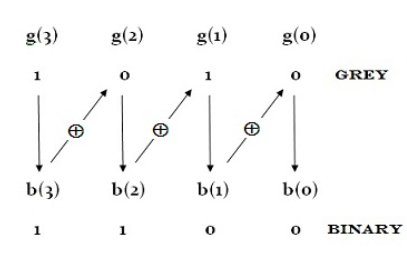
Gray codes are widely used to facilitate error correction in digital communications such as digital terrestrial television and some cable TV systems.How We Change Gray and Binnary code.

**How We Convert Gray code And Binnary code each other**

İlk önce binaryden graye dönüşüme bakalım. İlk olarak ilk biti değiştirmeden aşağı indiriyoruz. Ardından 2. Bit ile 1. Biti XOR luyoruz. Yanı aynı ise 1 farklı ise 0 yazıyoruz. Sonucu 2. Bitin altına yazıyoruz.



Grayden binarye ise şöyle yapılır. İlk biti tekrar aşağı indiriyoruz. Ardından 2. Bit ile indirdiğimiz biti XOR luyoruz sonra sonucu yanına yazıyoruz. Bu işlemi sonuna kadar tekrarlıyoruz.



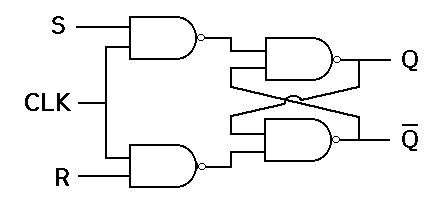
**What is flip flop**

Devre çalıştığı sürece çıkış durumunu koruyan yapılara Flip-Flop denir. Flip-Floplar temel hafıza birimini oluştururlar. 1 bitlik bilgiyi saklayabilirler. Flip-Flop’larda bu bilgiyi ve bu bilginin tümleyenini gösteren (kendisi ve değili) iki çıkışı bulunur (Q ve Q’). Bir Flip-Flop’nin çıkışı dendiğinde bu çıkışın kendisidir (Q). Çıkışın değişmesi için girişin değişmesi ve tetiklemenin değişmesi gerekmektedir. Tetikleme değişmediği sürece çıkış konumunu korur.

Flip-Flop’larda tetikleme clock palse (CP) denilen kare dalga sinyal ile yapılır. Devrenin niteliğine göre kare dalganın yükselen kenarıyla ya da düşen kenarıyla tetikleme yapılabilir. There are majorly 4 types of flip flops

**SR Flip-Flop**

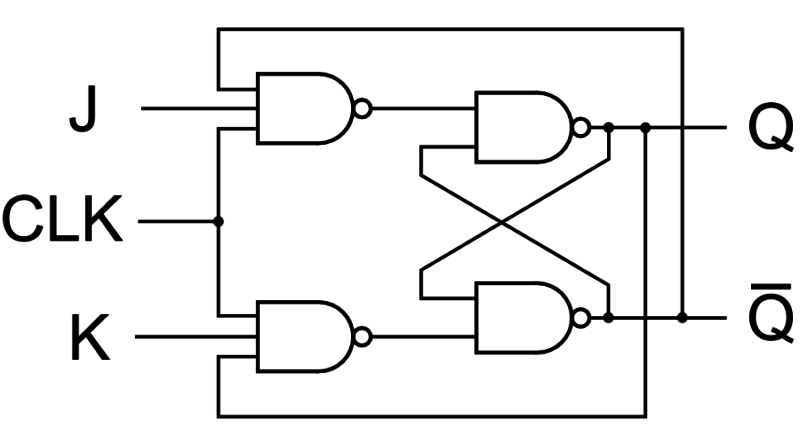
|  |  |  |
| --- | --- | --- |
| **S** | **R** | **Q(t+1)** |
| 0 | 0 | Q |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | ∞ |

****

**JK Flip-Flop**

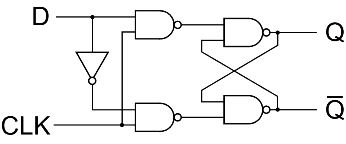
SR flip flop u 1-1 durumunda tanımsız olmakta. Bu nedenle başka bir flip flopa ihtiyaç duyuluştur. Bu da JK dır.

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



**D flip-Flop**

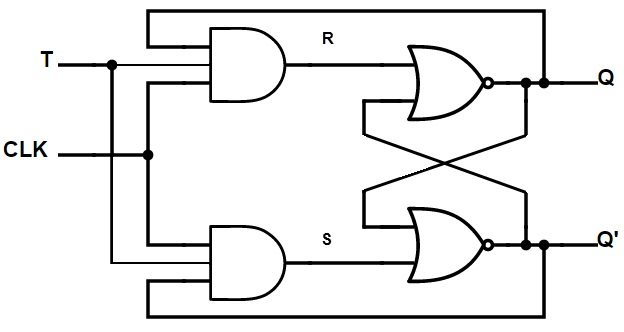
D flip flop is a better alternative that is very popular with digital electronics. They are commonly used for counters and shift-registers and input synchronisation.

****

|  |  |
| --- | --- |
| **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 (i.e.) Toggle, hence the 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

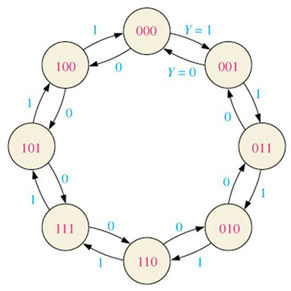
-Shift Registers

-Storage Registers

**State Diagram**

İlk olarak problemimizi anlamalıyız. Ne istiyor bizden. Aşaği yukarı gray şeklinde flip-floplar ile sayabilen bir sayıcı tasarlamamızı istıyor. Bunun için durumlarımızı bilmeliyiz. 8 adet durumumuz var. Ayrıca aşağı yukarı saymak için x inputumuz var.

8 adet durumumuz olduğundan 3 adet flip-flop kullanmamız gerekiyor. State diagram bir problem başlangıcında çok önemlidir. Hocalarım bu kısma çok önem vermemizi isterdi.



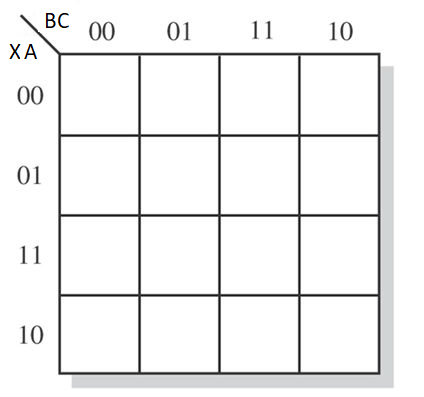
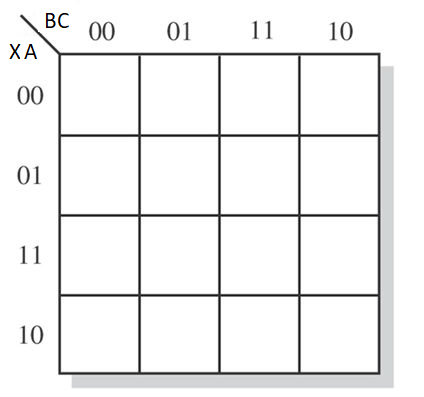
**Truth Table**

State diagramından sonra doğruluk tablosu ve karnaugh map kullanarak devre şeklimizi öğrenebiliriz. İstediğimiz çıkış değerlerine göre girişlerimiz ne olmalı sorusunun yanıtını arıyoruz.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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**

D tipi flipflopta next state ile flıp flop ınputs aynıdır. Çünkü girişe ne veririsek d tipi flip flopta çıkışta da aynısını alırız.



0 0 0 1

0 1 1 1

0 0 0 1

0 1 1 1

1 0 0 0

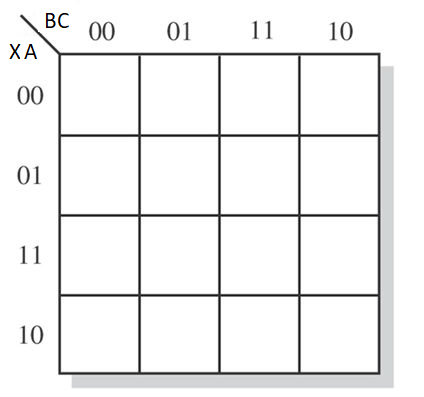
1 1 1 0

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

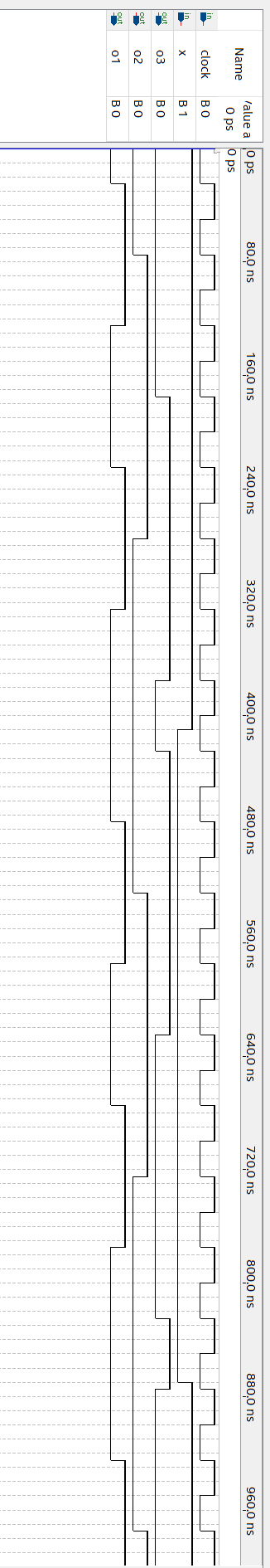
0 0 1 1

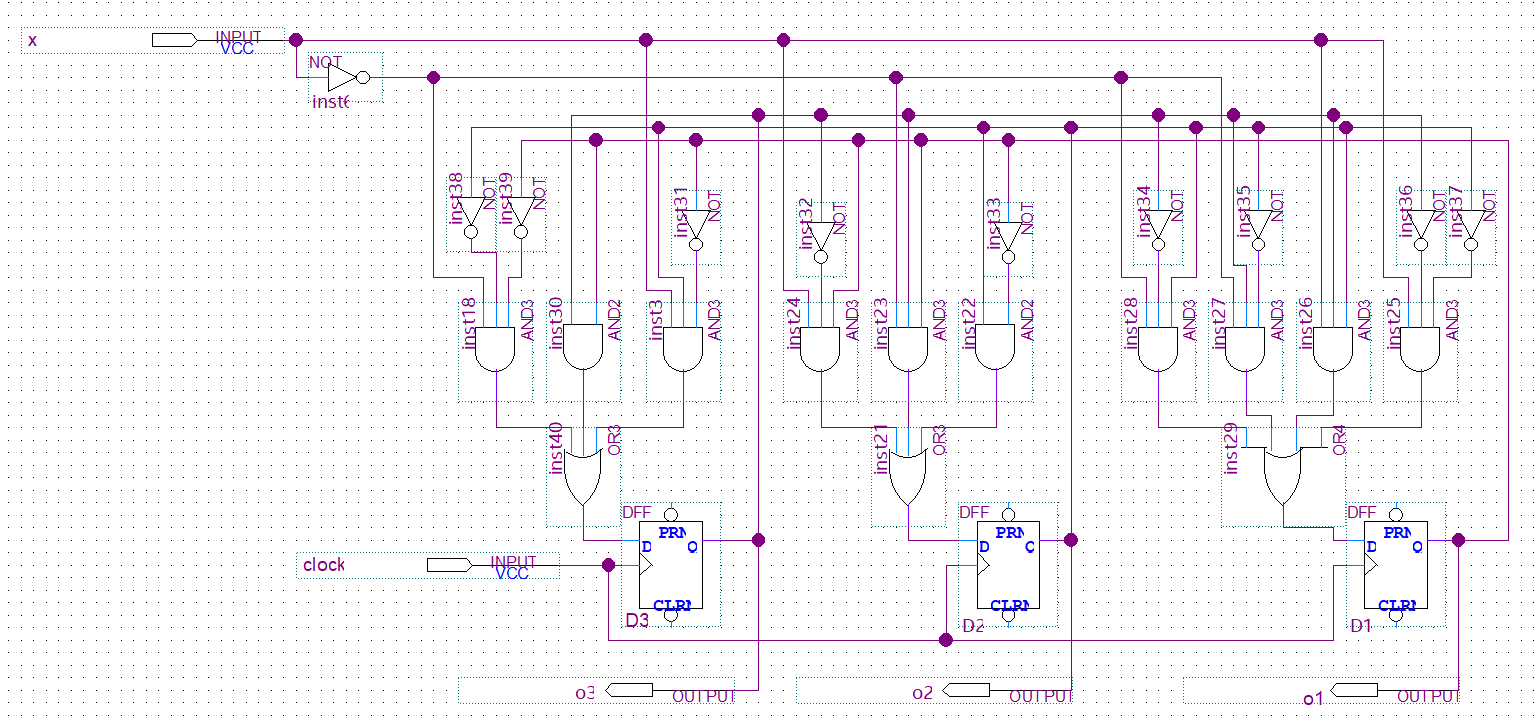
1 1 0 0

D1 = X’A’C + X’AC’ + XAC + XA’C’

Giriş değerlerimizi bulduktan sonra devremizi çizebiliriz.

GATE SİMULATİON AND ANALYSIS





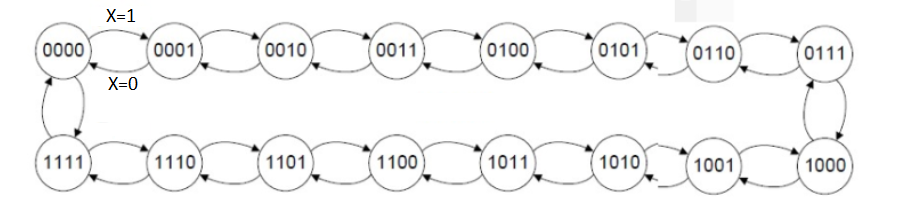
Devremiz sorunsuz olarak çalışmakta. D flip flopu ile diğerlerinden farklı olarak sadece 10 tane and gate kulanmak yeterli oldu. Jk 16 tane, t tipi yalaşık 10 tane kapı istemekte. Ve kapıları 3 tane olduğu için hepsinde ortak. Ancak t tipi kullandığım zaman devre karmaşık oldu ve bu nedenle d kulanmaya karar vedım. Sonuç olarak sadece 1 biti değiştirerek istediğimiz şeyi yaptırabildik.

**Indroduction**

Flıp-flop kullanarak 4 bit sayıcı tasarlayacağız. X yine up/down için kullanılacak. Hold için ise clock ve H and landi. Eğer H 1 olursa sayma duracak. çünkü senkronize flip flop kullanıyorum. Ve flip flop çıkışında 7 segment display kullanacağız. 0 dan 15(F)’e kadar sayacağız.

**State Diagram**

İlk olarak durumlarımızı bilmeliyiz. 16 adet durumumuz var. 4 adet flip flop kullanacağız. Ayrıca x’e da sahibiz. Bu nedenle 5’li karnaugh map kullanacağız.



**Truth Table**

Truth table sayesinde flip flop girişlerimizin ne olası gerektiğini öğrenebiliyoruz.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **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



D0 = D’

D1 = X’C’D ’ +X’CD + XC’D + XCD’

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

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

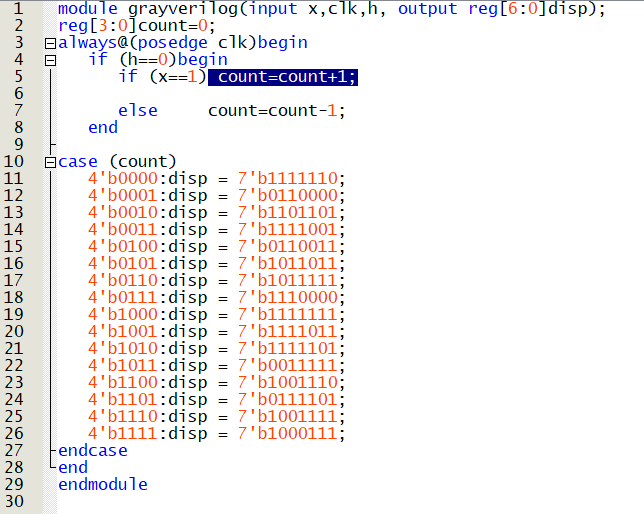
0 1 1 1

1 0 0 0

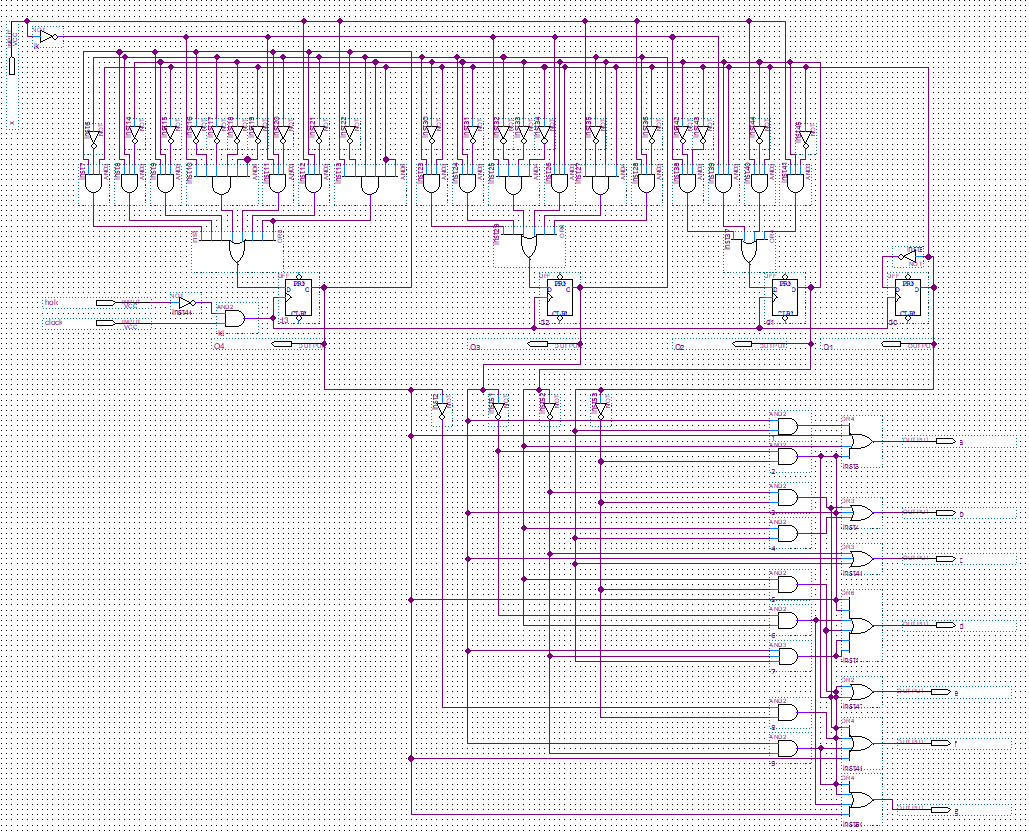
D2 = BC’D + BCD’ + X’B’C’D’ + X’BC + XB’CD + XBC’

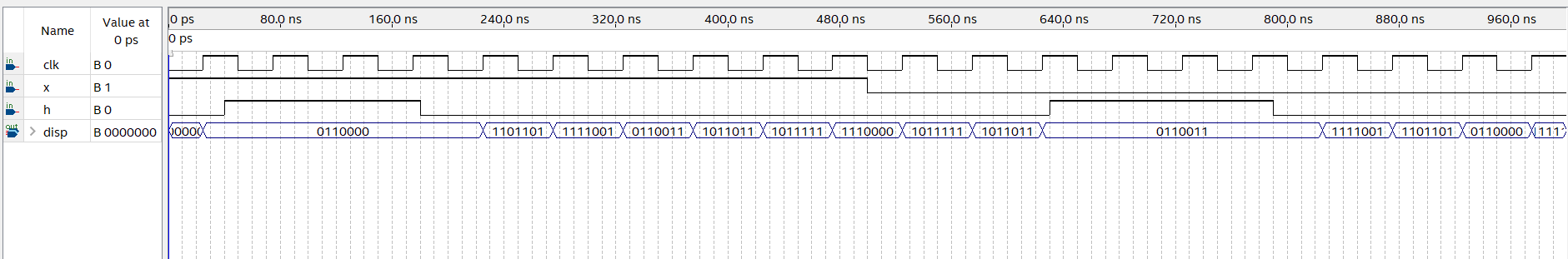
Karnaugh map çözülürken ilk olarak X=0 ile X=1 arasında ortak olan bölümler yazılır ve X yok olur. Ardından kalanlar direk eklenir.

**Verilog Code**

****

**Gate Simulation and Analysis**



****