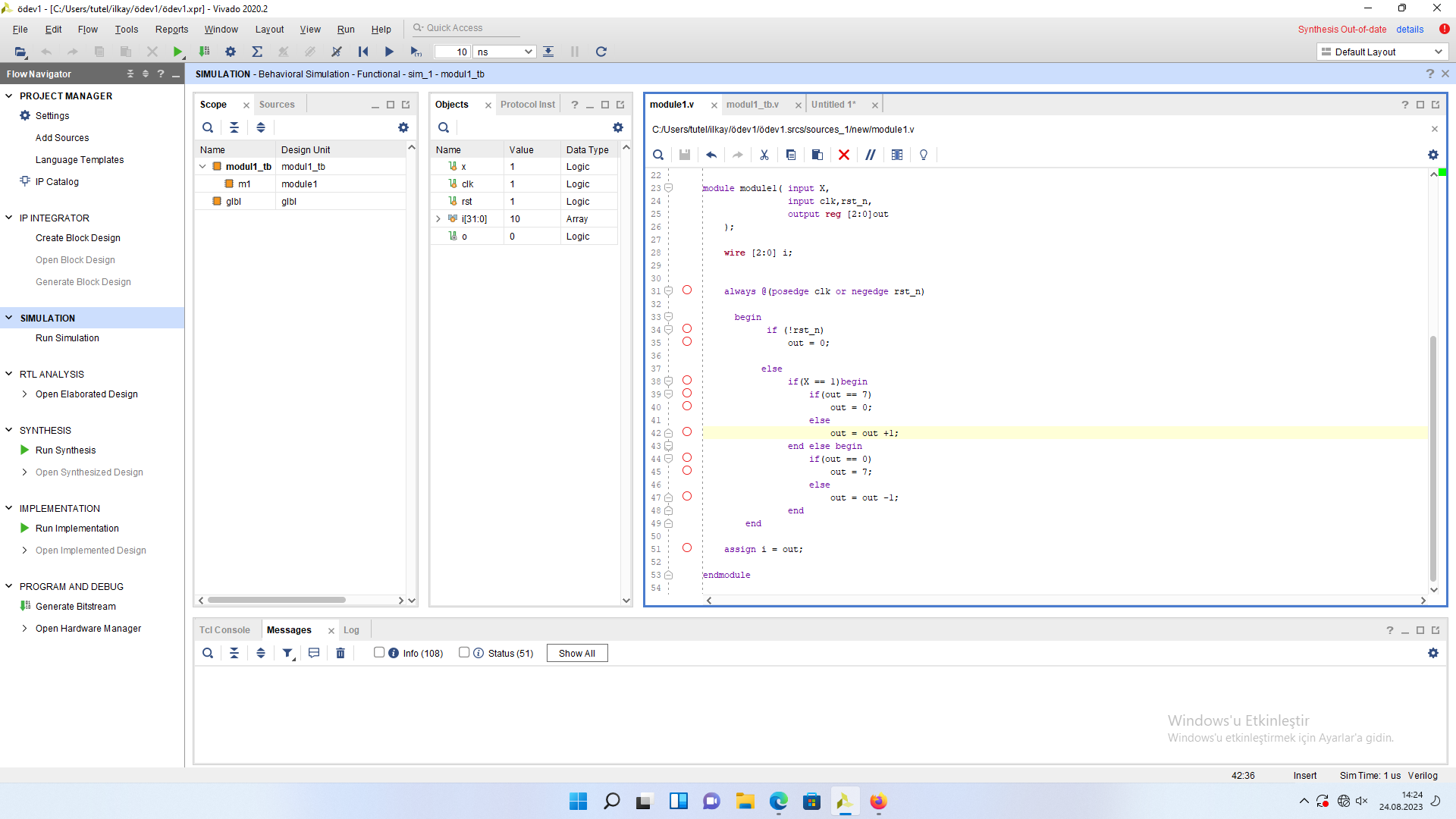
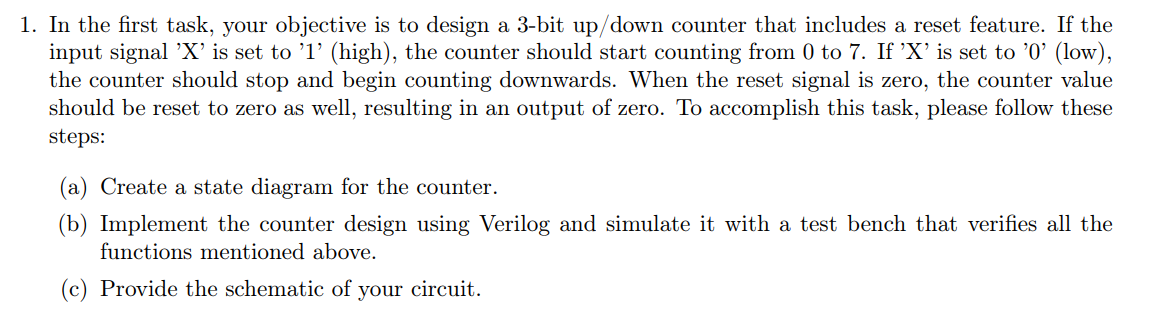
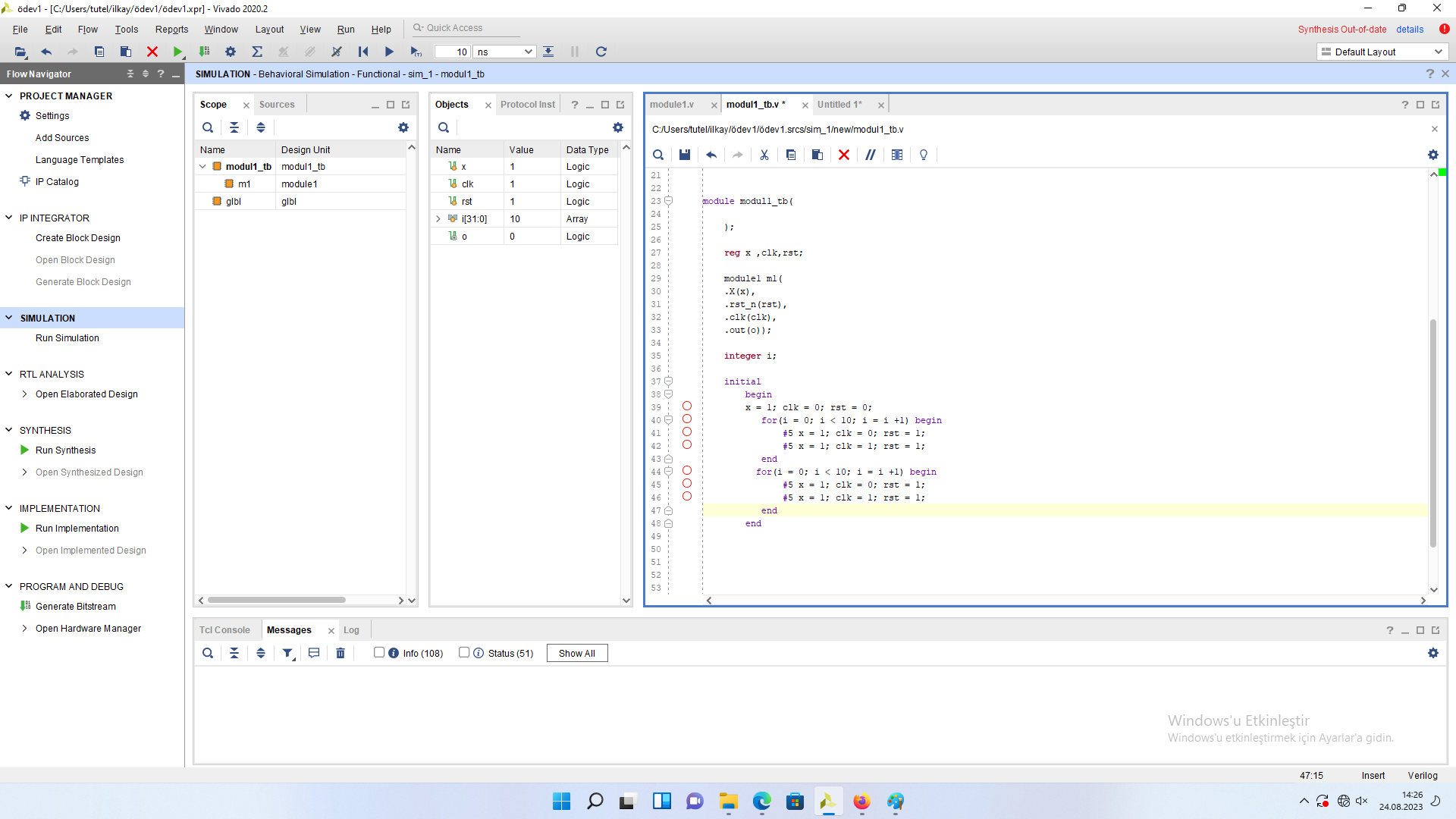
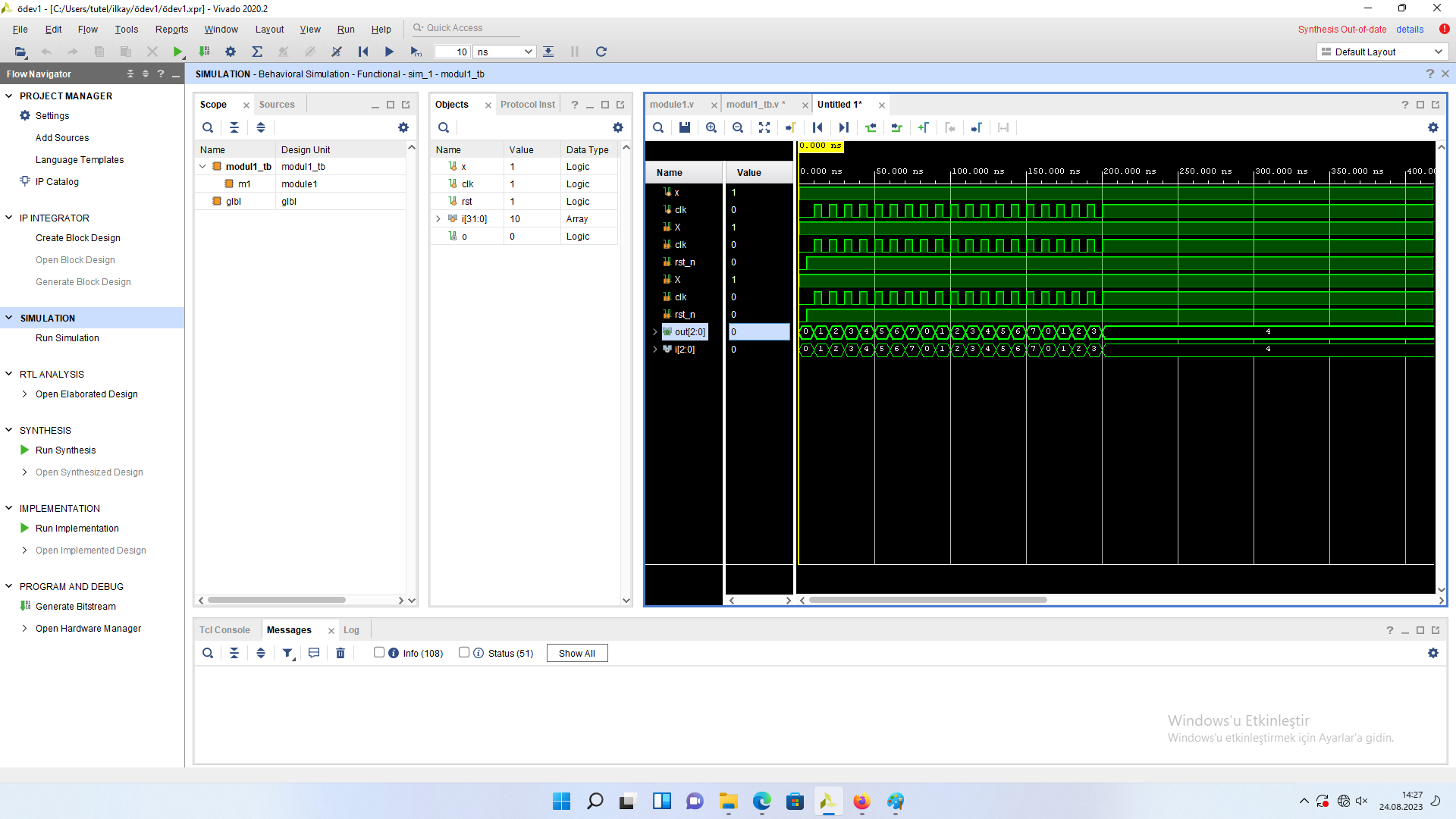
Counter – 1. Ödev



Birinci ödevin ilk aşaması 3 bitlik bir sayaç değişkeni tutup X değişkenine göre artıp azalmasını verilog ile uygulamaktı. Bu aşamada procedural block ve conditional programming construct yapısını öğrendim. Daha öncesinden Quartus kullandığım için Test Bench yazmamıştım ve Test Bench yazımnı öğrendim . Algoritma kısmına gelecek olursak always bloğu içerisine rst dalgasının negedge eklemenin çok kullanışlı olduğunu farkettim. Rst\_n ile ilk başta 0 gönderiyoruz ve tüm değişkenlere ilk değer atıyor. Eğer ilk değer atılmaz ise kodun geri kalan kısmında değişkenler kendinin değili ile değiştiği durumda sorun çıkartıyor. Sayı 111 olduğu artış durumunda 0 olması ve 000 olduğu azalış durumunda 111 olması için if blokları koydum ama ilerleyen süreçte buna ihtiyacım olmadığını farkettim

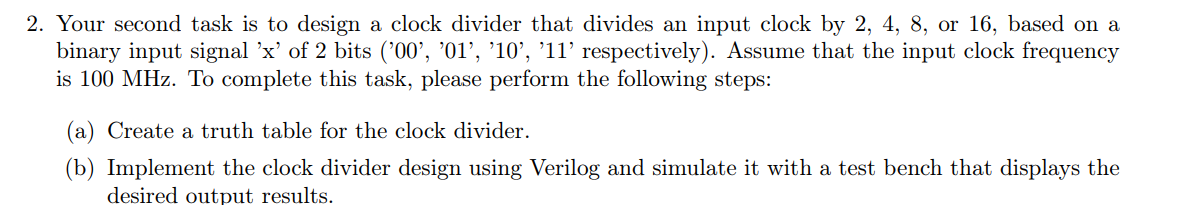


Test Bench yazımında clokc cycle gönderimini for ile yaptım

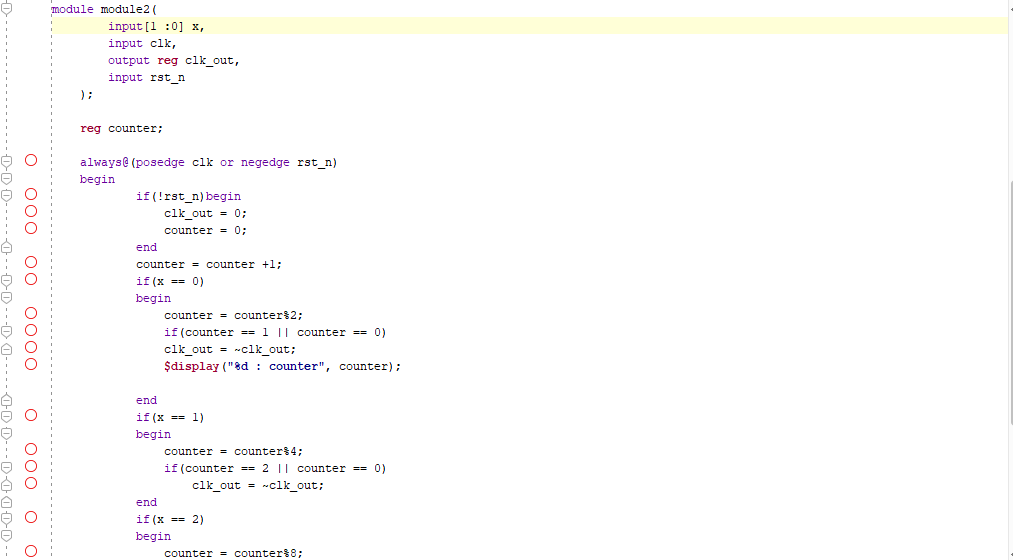


Simulasyon görüntüsü bu şekilde çıktı.

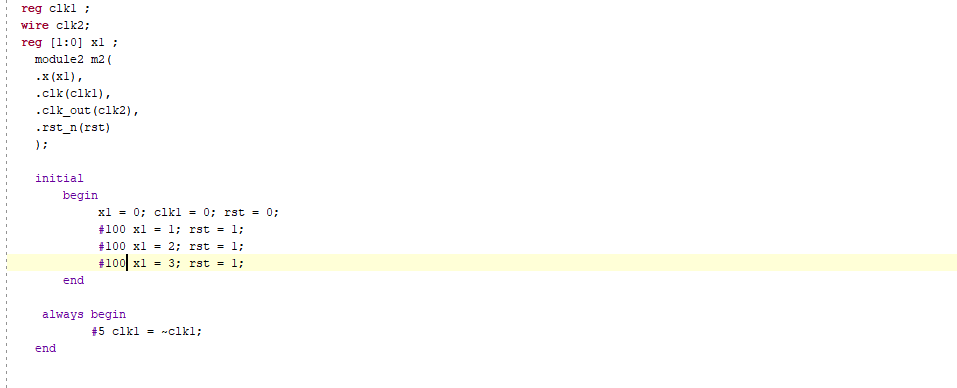
Aşama – 2



2. aşamada sistemden gelen hızlı sinyalleri bölüp başka bir sinyal üretmemiz istendi. Gelen sinyaller input değerlerine göre 2, 4 , 8 veya 16 ya bölünecekti. Sistemden gelen sinyali 100MHz olarak varsaymamız istendi bu durumda Test Bench’e #10 ile 10 nanosaniyelik bir gecikme gönderdim.

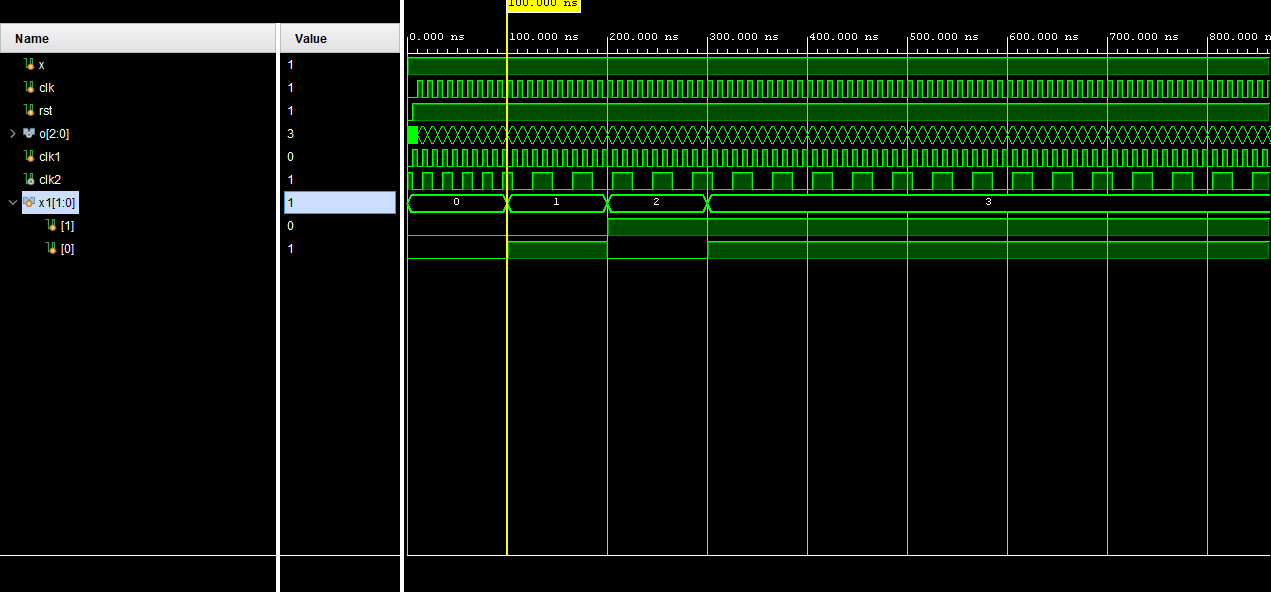


Algoritma kısmında hangi sayıya böleceğimizi belirten inputları if blokları ile ayırdım ve her clock cycle da bir artan couner tanımladım. Bölmek istenilen sayı ile modu alınan counter değeri bölünmek istenilen sayının yarısı veya 0 olduğunda yavaşlatılmış clock cycle tersine dönecek şekilde bir algoritma kurdum. Bu sayede sistemden gelen sinyaller yavaşlatılmış oldu.



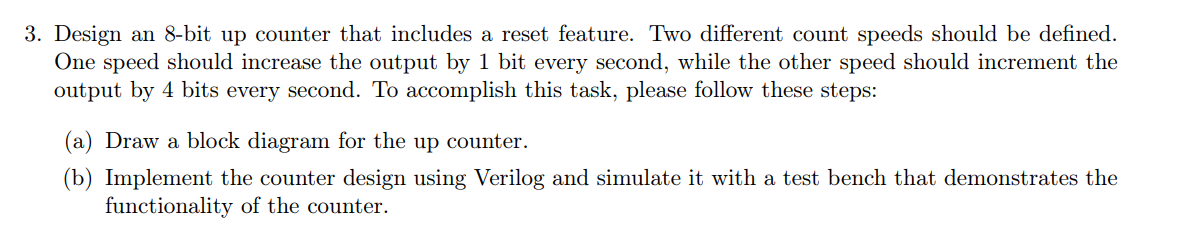
Test Bench

2. aşamya geldiğimde clock cycle’ı always bloğu ile gönderebildiğimi ve daha efektif olduğunu farkettim

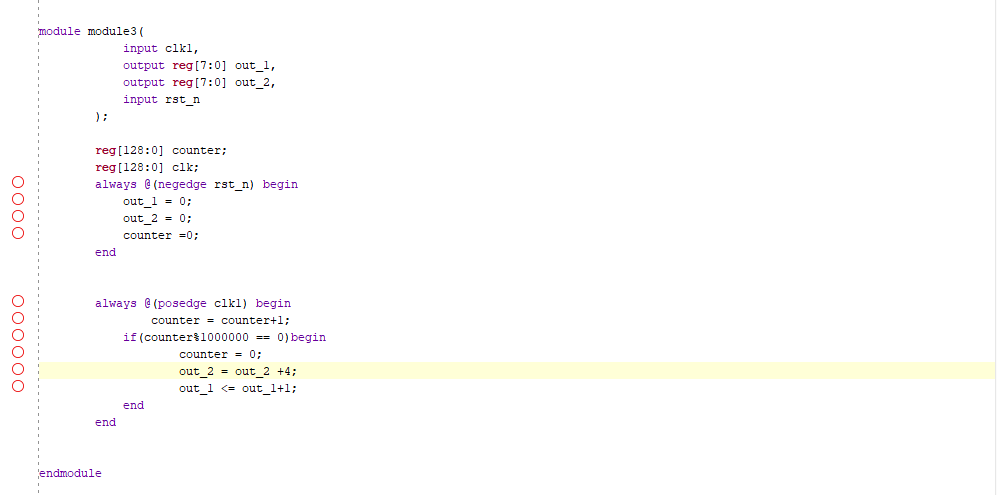


Simulasyon görüntüsü

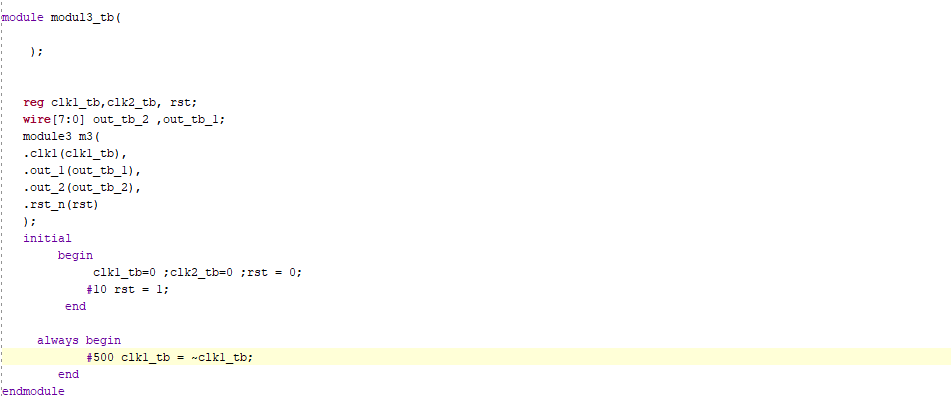
Aşama – 3



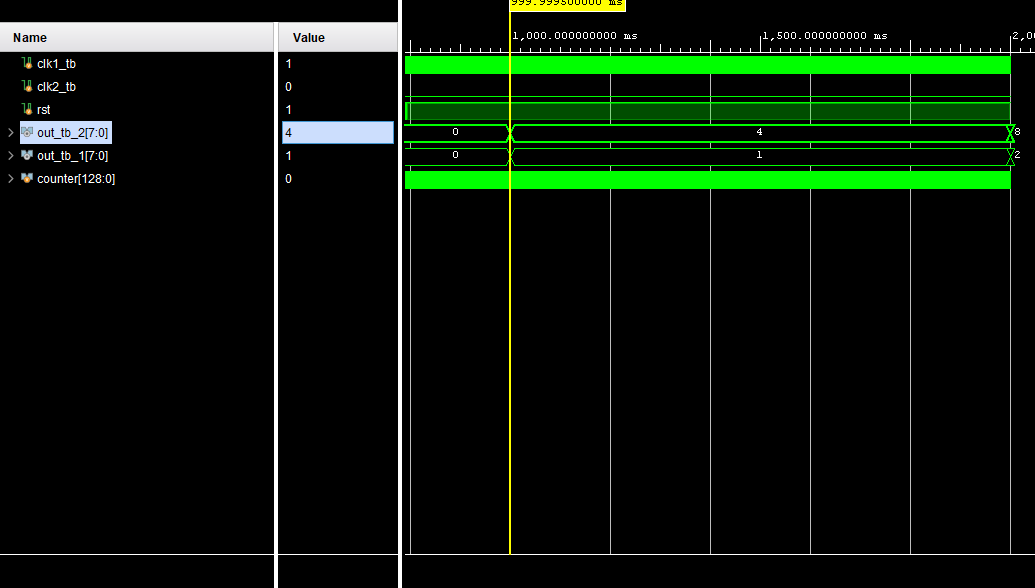
3. aşamada ilk ve ikinci aşamanın beraber bir uygulmasını yaptık. Sistem sinyalini 1 Hz e kadar düşürmemiz gerekiyordu. Bu aşamada bir önceki aşamaya göre farklı bir yol izledim. Bu yol mod almak yerine modunu almak istediğim sayı ile gelince 0’a eşitlemek oldu. Çünkü mod alma işlemi yavaşlatıyordu.



Gelen sinyal 100 MHz olduğundan dolayı counter 1 milyonda bir olacak şekilde yavaşlattım. Ve her bir 1 milyonda bir counter 0’a eşitlendi ve istenilen değerde arttırılmalar uygulandı.



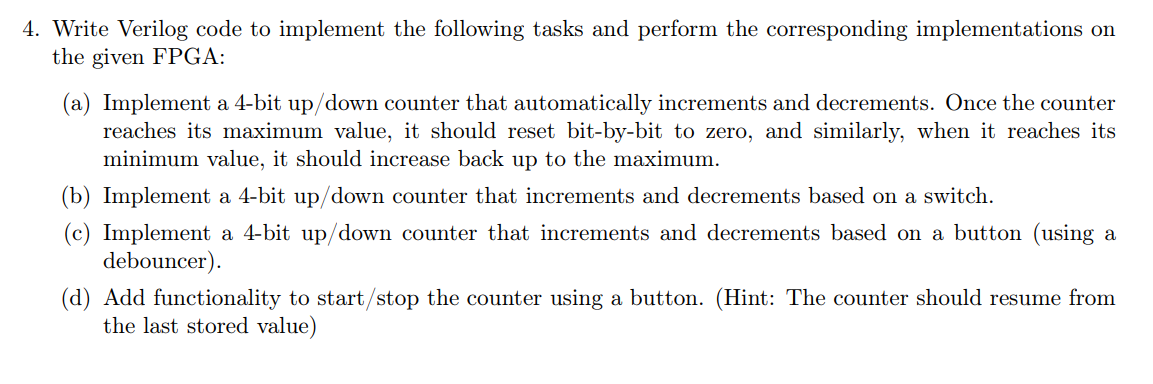
Test Bench



Simülasyon görüntüsü

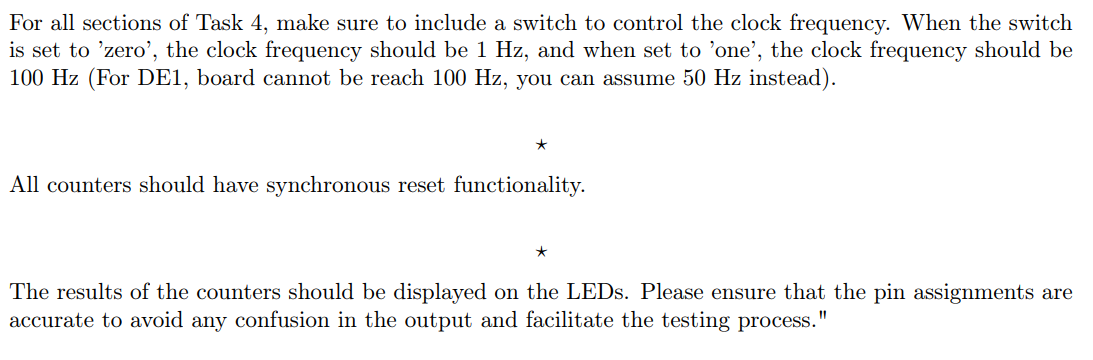
Aşama – 4

4.

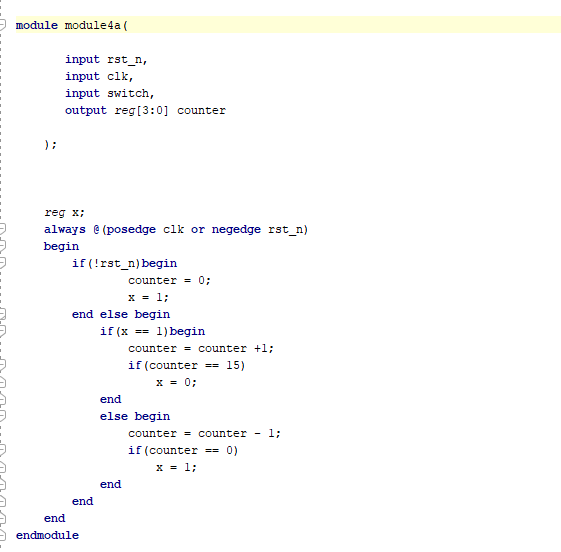


4. aşama 4 bölümden oluşuyor ve her bir bölümde FPGA üzerinde uygulanabilir olmalıydı

Ve her bölüm için istenilen bazı koşullar vardı :



Aşama - 4a



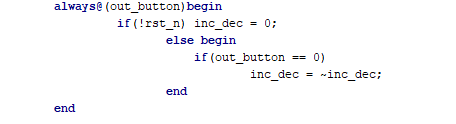
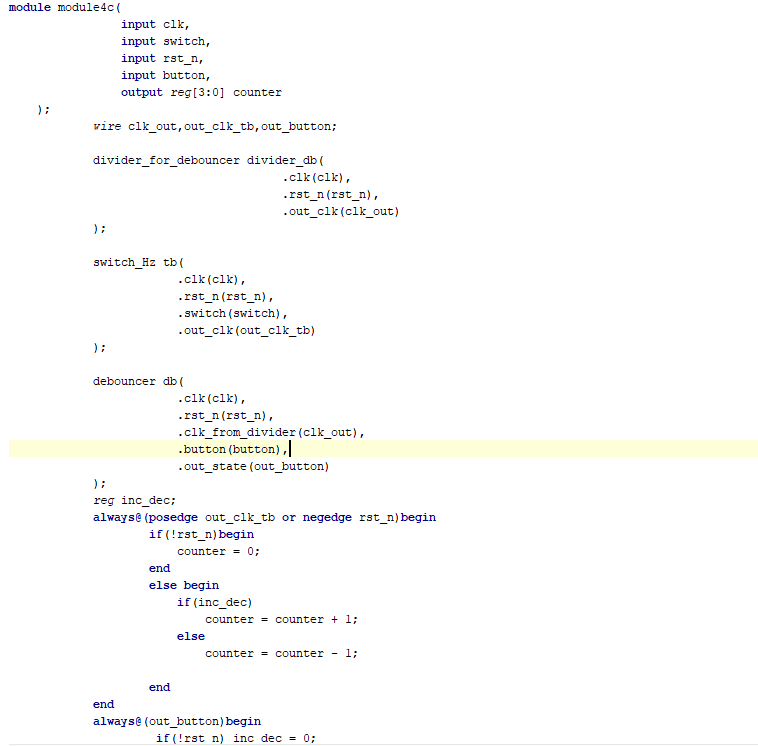
İlk bölümün algoritması yukardaki gibidir. Sayı maksimum değerine ulaştığı zaman bit bit tekrardan azalıyor ve minimum değerinde bit bit artıyor.

Aşama 4b



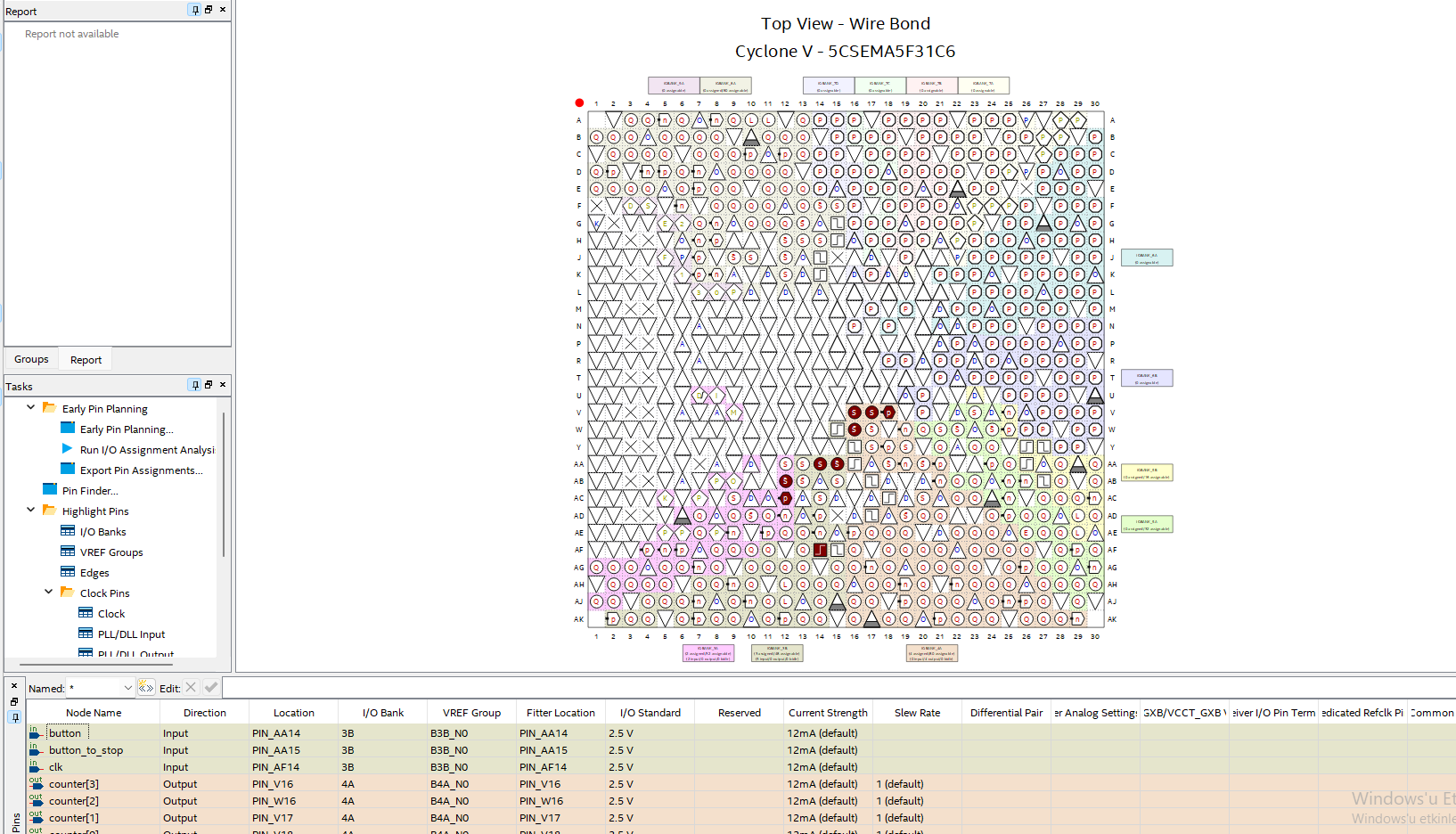
Aşama 4b de ayrı bir switch tanımlamamız ve bu switche göre sayının artıp azalması gerekiyordu ve if blokları ile uyguladım

Aşama4c



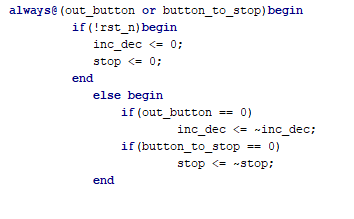
Bu bölümde bir buton tanımlamamız ve bu butona göre sayının artması ve azalması gerekiyordu FPGA üzerinde çalışabilmesi içinde kullanılan tüm modüllerin top modül içince çağırılması lazımdı.

Button algoritmasını ise button hassasiyetli bir always bloğu içinde buton 0 oldugunda art\_azal değişkenini değiştirecek şekilde ayarladım.



Sentez aşamasını geçtikten sonra pin atamalarını yaptım ve çıkan SOF dosyasını FPGA üzerinde çalıştırdım

Aşama- 4d



Son aşamada ise bir önceki koda buradaki değişikliği eklemem ile stop-start mekaniğini çalıştırdım