VERİLOG İLE KAPI SEVİYESİNDE TASARIM

1 GİRİŞ

Verilog, donanım tasarlamak için geliştirilmiş bir programlama dilidir. Verilog benzeri dillere Hardware Description Language (Donanım Tanımlama Dili) denir. Verilog ile tanımlanan bir donanım, Computer-aided Design (CAD) araçları kullanılarak istenilen platform için gerçeklenebilir. CAD araçlarını (örnek: Vivado) belli bir platform için (örnek: FPGA) donanım sentezleten derleyici (compiler) gibi düşünebilirsiniz. Xilinx Vivado programı, Verilog kodunu sırasıyla "Run Synthesis", "Run Implementation" ve "Generate Bitstream" adımlarını kullanarak istenilen FPGA'e uygun bir programa dönüştürmeye yardımcı olur.

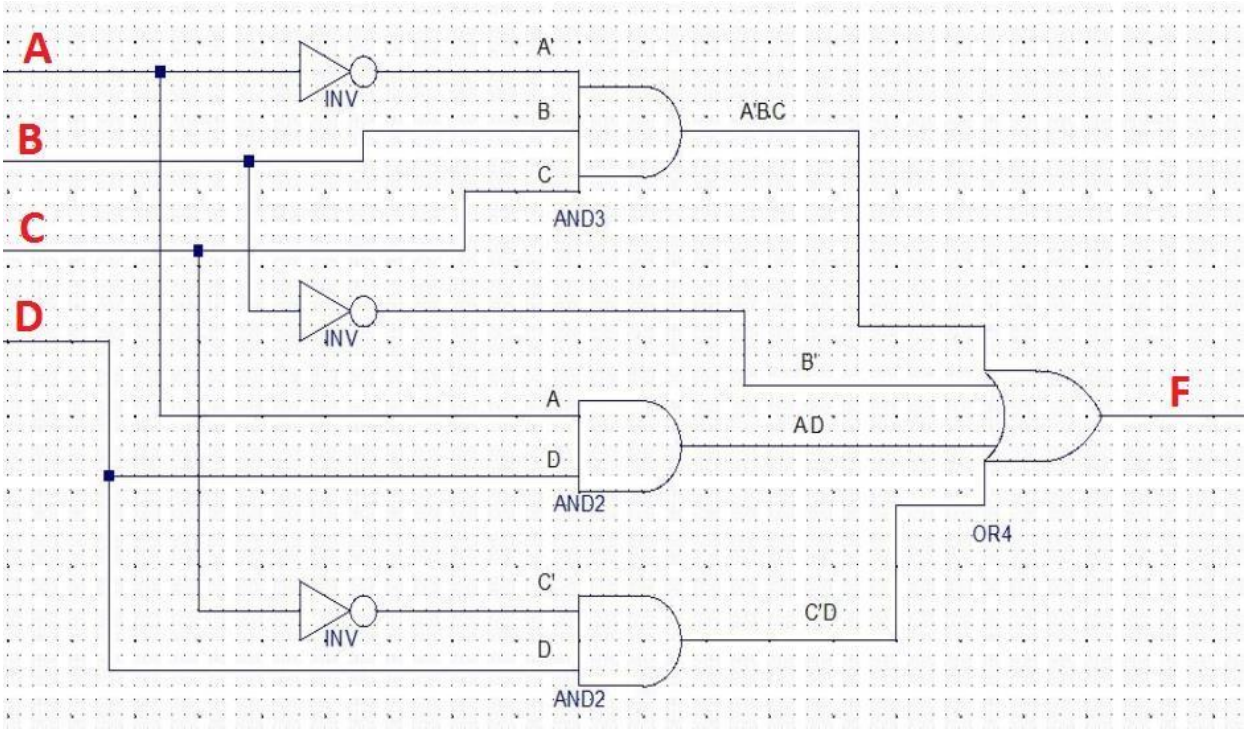
Bu dokümanda, Verilog diline basit bir başlangıç yapacağız. BİL264L/ELE263L dersleri kapsamında Verilog ile Kapı Seviyesinde (Gate-Level Modeling) ve Davranışsal Modelleme (Behavioral Modeling) yöntemlerini işleyeceğiz. Dokümanın devamında basit bir örnek ile kapı seviyesinde tasarım anlatılmıştır.

2 BİR BOOLE FONKSİYONUNUN KAPI SEVİYESİNDE VERİLOG İLE GERÇEKLEŞTİRİLMESİ

Bu uygulamada aşağıdaki boole fonksiyonunu Vivado programını kullanarak Verilog ile kapı seviyesinde gerçekleştirmeyi öğreneceğiz. Bu boole fonksiyonu **Karnaugh haritası** metodu ile bulabilirsiniz.

$$\mathbf{f\,(A, B, C, D) \;=\; A'BC \;+\; B' \;+\; AD \;+\; C'D \;=\; F}$$

A, B, C ve D tek bitlik giriş sinyallerine sahip bu fonksiyonu gerçekleyen devrenin kapı seviyesinde şematik gösterimi Şekil 1'de verilmiştir.

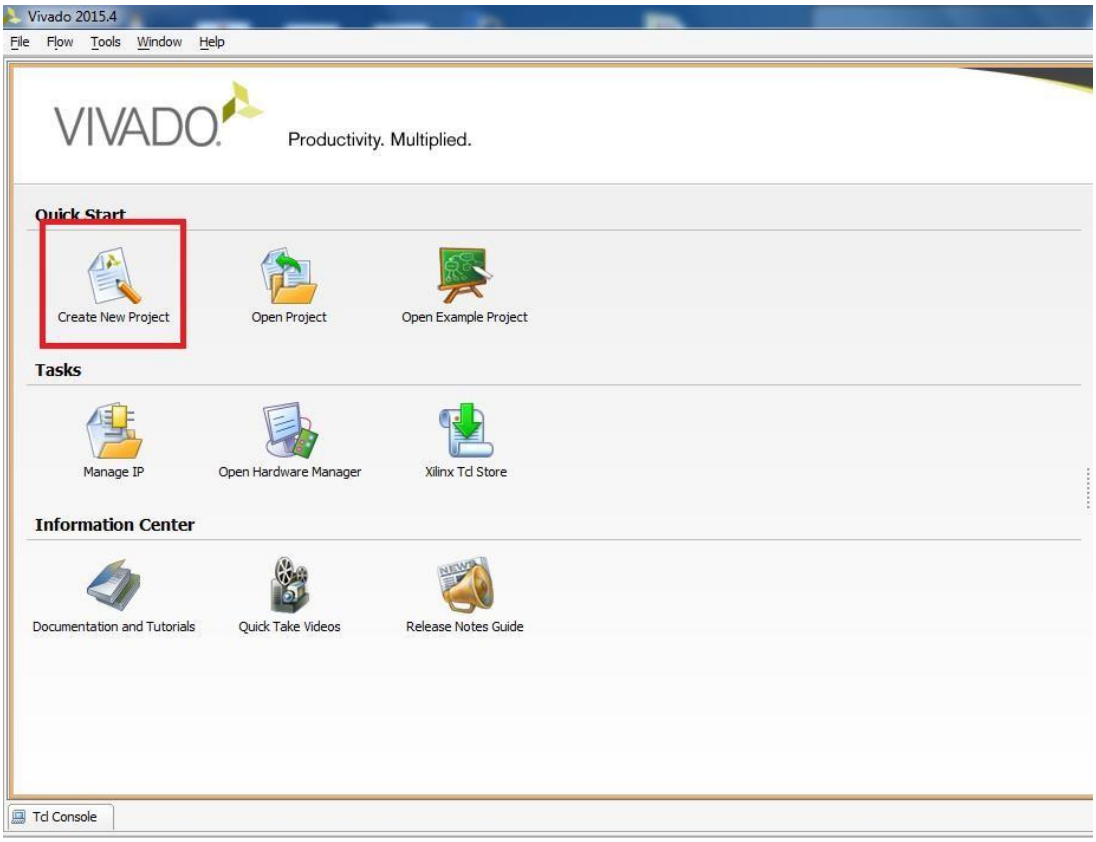


Şekil 1: A, B, C ve D tek bitlik girişlerine sahip mantık fonksiyonunu gerçekleyen devre.

Şimdi, bu devrenin Vivado programında nasıl gerçeklenebileceğini adım adım görelim.

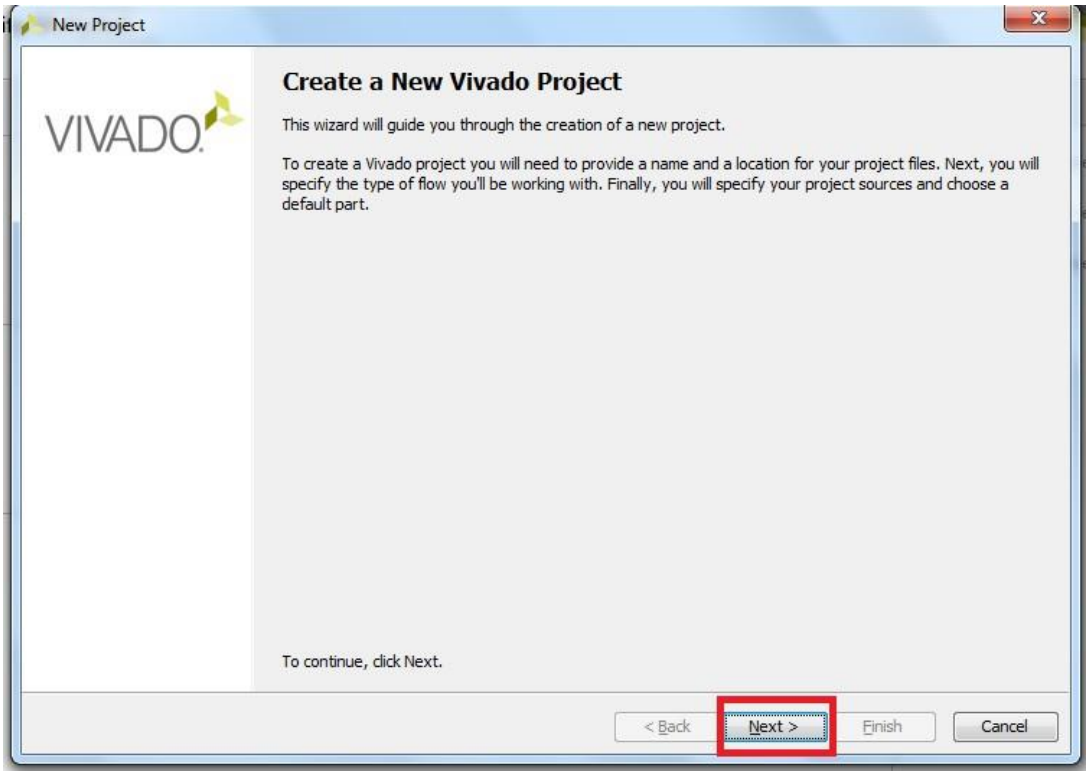
2.1 Yeni Proje Oluşturma

Yeni bir Verilog Projesi oluşturmaya başlamak için Vivado'nun açılış penceresinde bulunan "Create New Project" butonuna tıklayın. Alternatif olarak, ekranın sol üstünde bulunan menüden "File -> New Project" seçimini yaparak da yeni bir proje oluşturmaya başlayabilirsiniz.



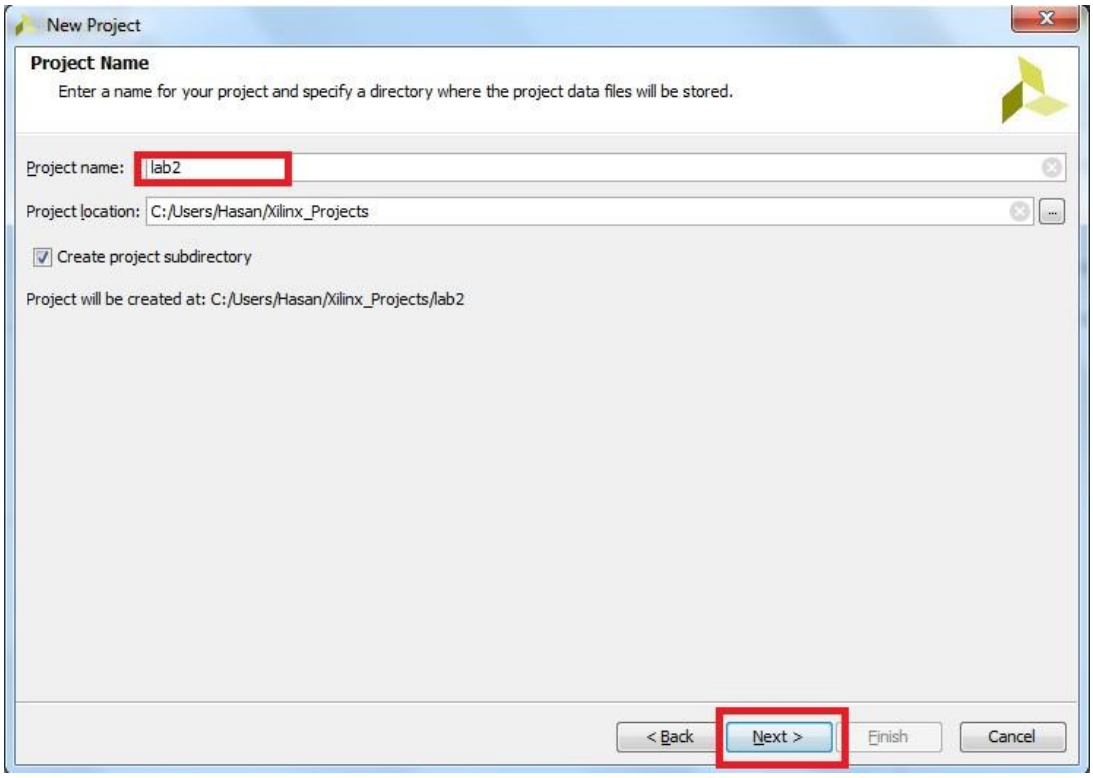
Şekil 2: Vivado'da yeni bir proje oluşturmak.

Açılan pencerede “Next” butonuna tıklayın.



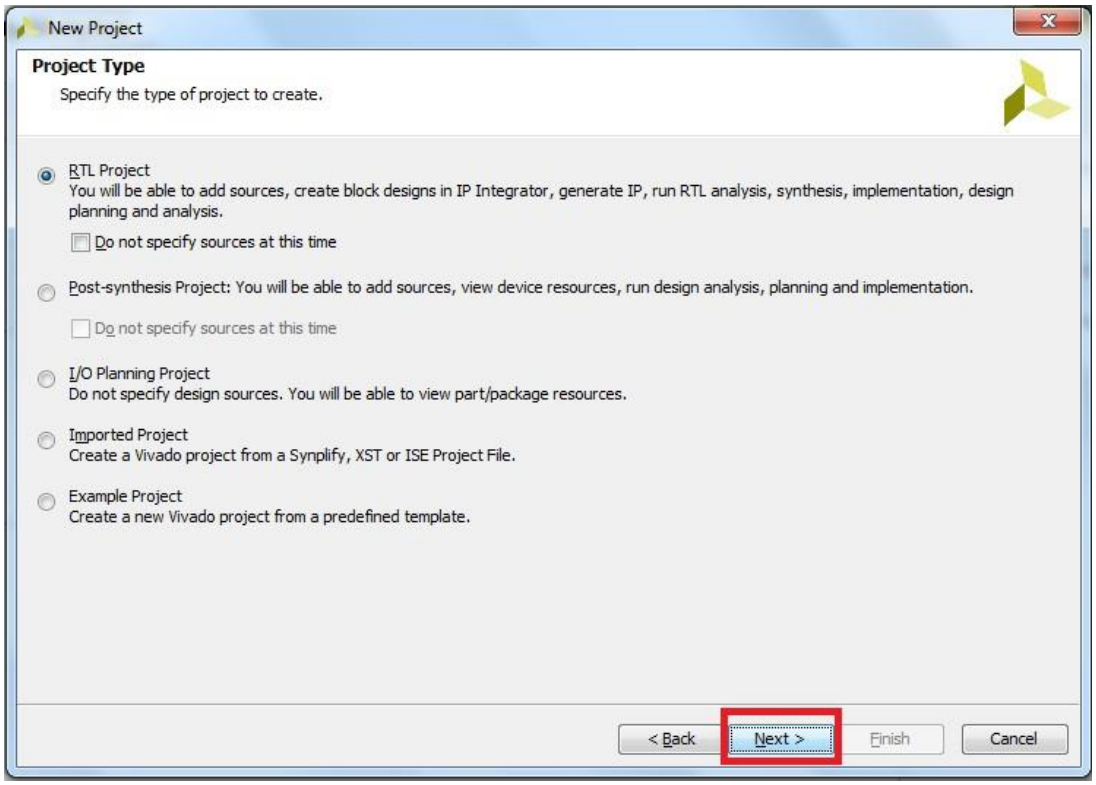
Şekil 3: Projenizi oluşturmak için “Next >” yazılı butona tıklayın.

Bir sonraki sayfada “Project Name” kısmına projeye vermek istediğiniz ismi yazın. “Project Location” ise proje dosyalarının kaydedileceği konumu göstermektedir. İsterseniz bu konumu değiştirebilirsiniz.



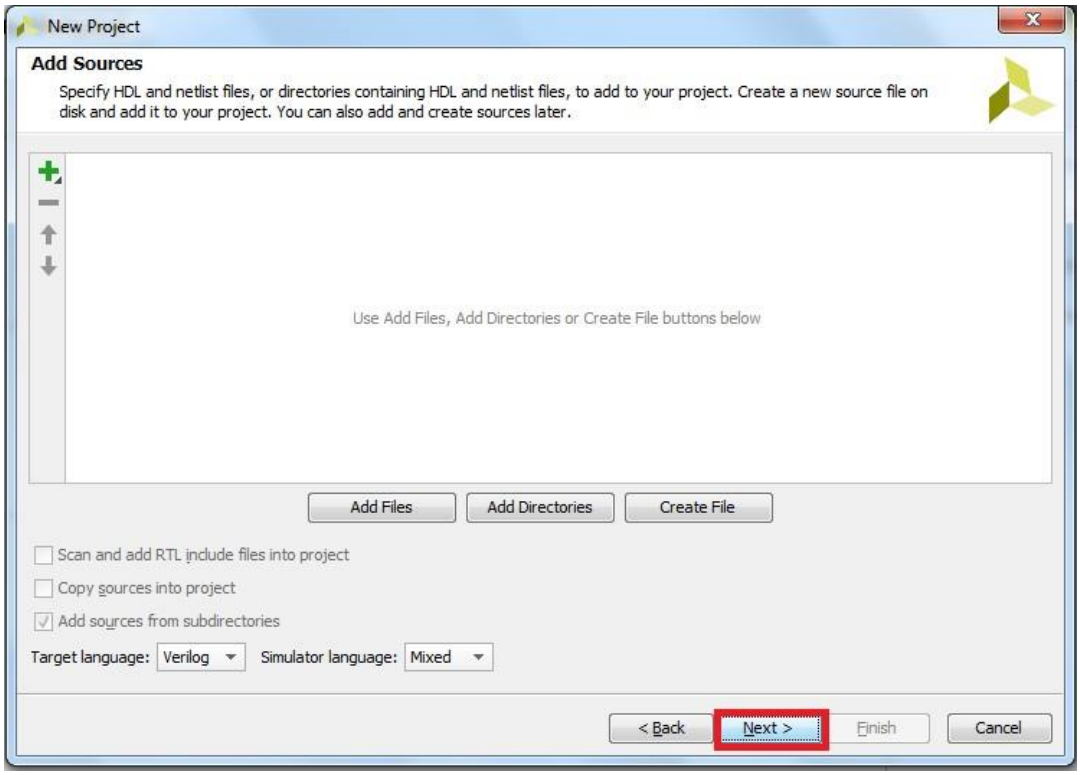
Şekil 4: Projenizin ismini ve bulunmasını istediğiniz konumu seçin.

Bir sonraki sayfada "RTL Project" seçeneğinin etkin olduğundan emin olup "Next" butonuna tıklayın.

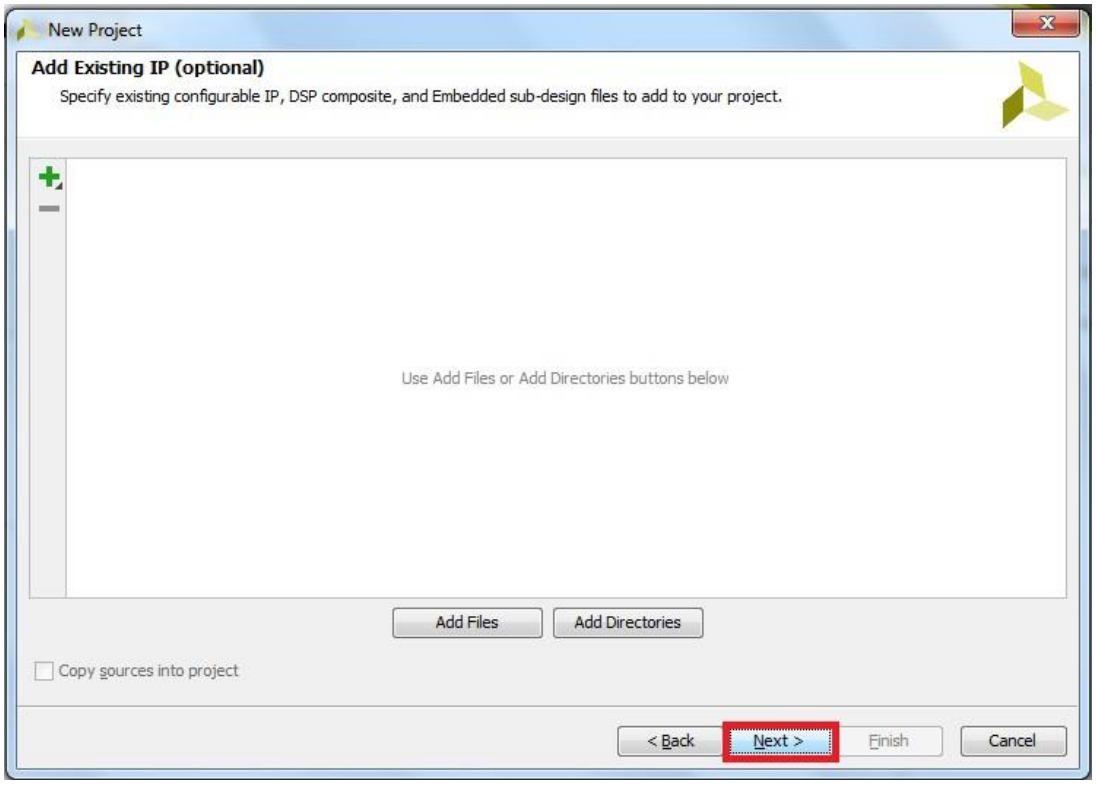


Şekil 5: "RTL Project" seçeneğinin etkin olduğundan emin olun.

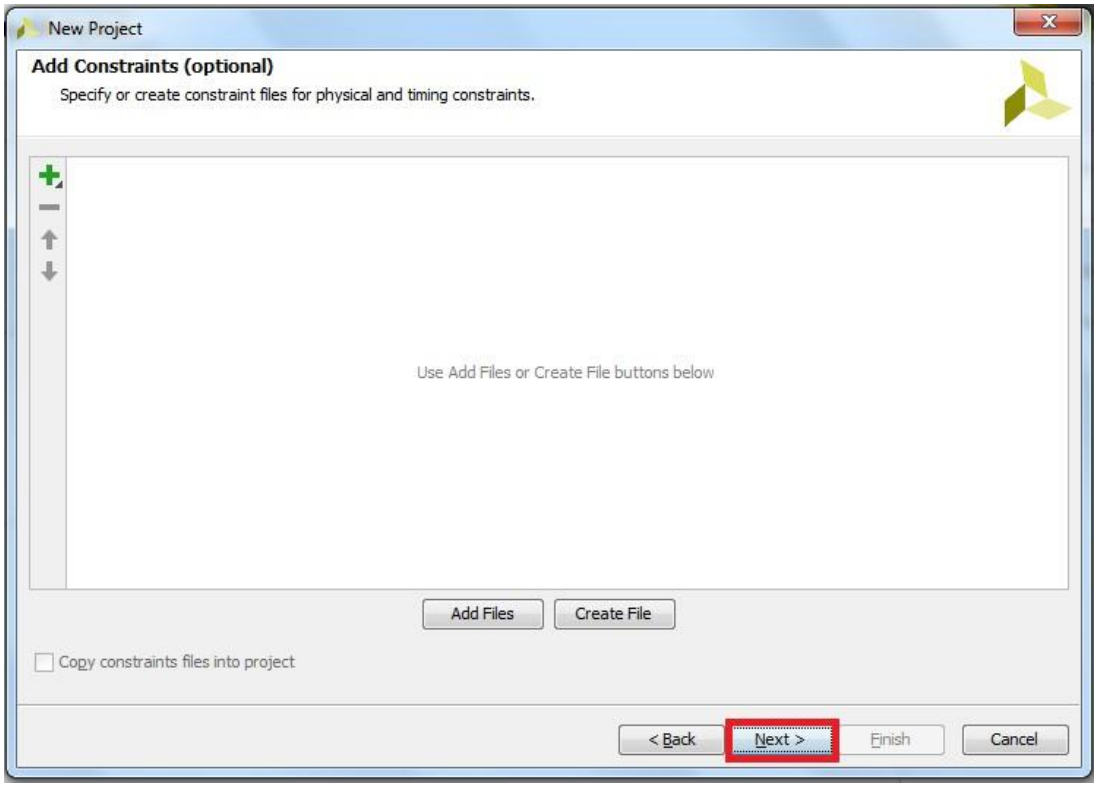
Sonraki üç sayfayı “Next” butonuna tıklayarak geçin.



Şekil 6: Projenize eklemek istediğiniz HDL dosyalarını bu pencereden seçebilirsiniz. Şimdilik bu adıma ihtiyacımız olmayacak.

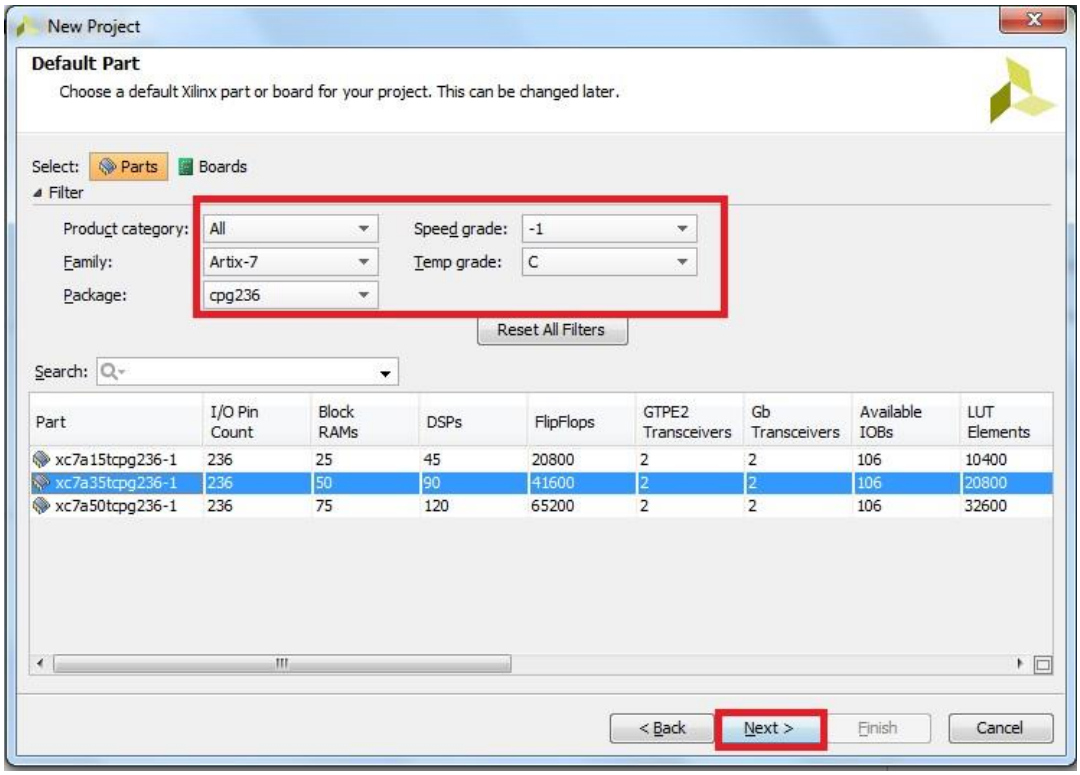


Şekil 7: Projenize hazır olarak verilen bir devre elemanı eklemenizi sağlayan pencere.
Bu deney için bu pencereyi de "Next >" butonuna basarak geçin.



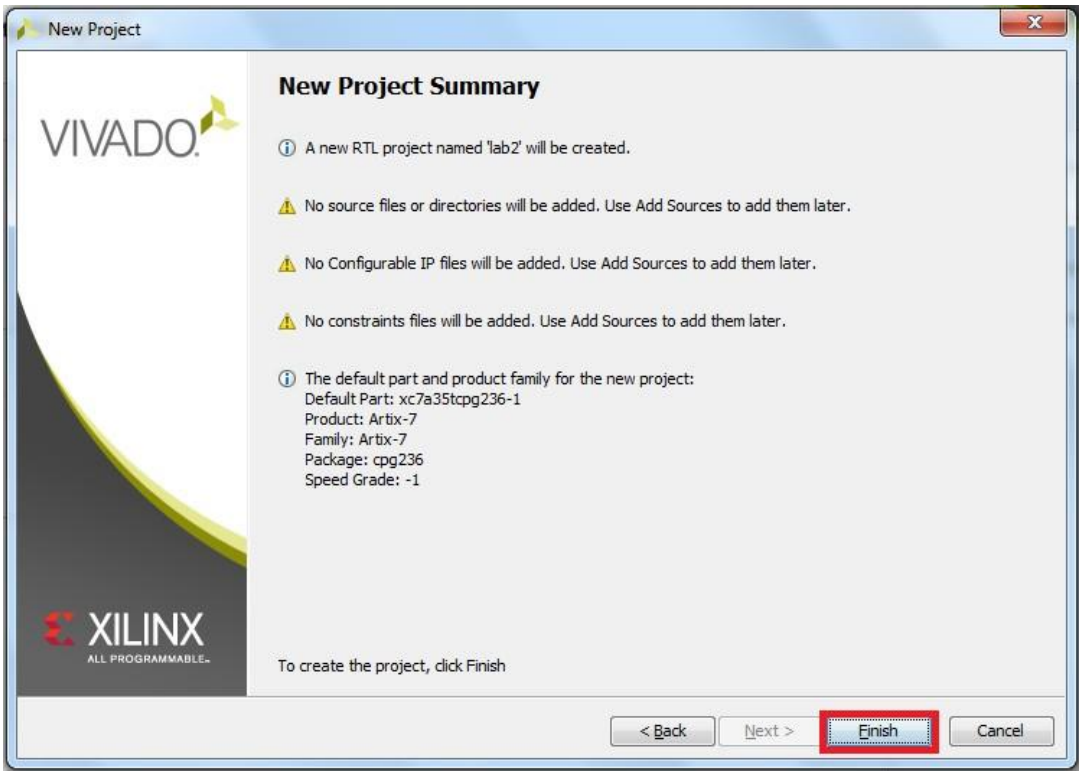
Şekil 8: Projenize kısıt dosyaları eklemek için kullanacağınız pencere.

Sonraki sayfada ise aşağıda gösterildiği gibi derste kullanacağımız FPGA'in modelini seçin. Büyük kırmızı kutucuk içerisindeki tercihleri yaptıktan sonra geriye 3 FPGA modeli kalacaktır. Bunlardan ikincisi bizim kullanacağımız FPGA modelidir. Bunu seçip "Next" butonuna basın.



Şekil 9: FPGA modeli seçimi.

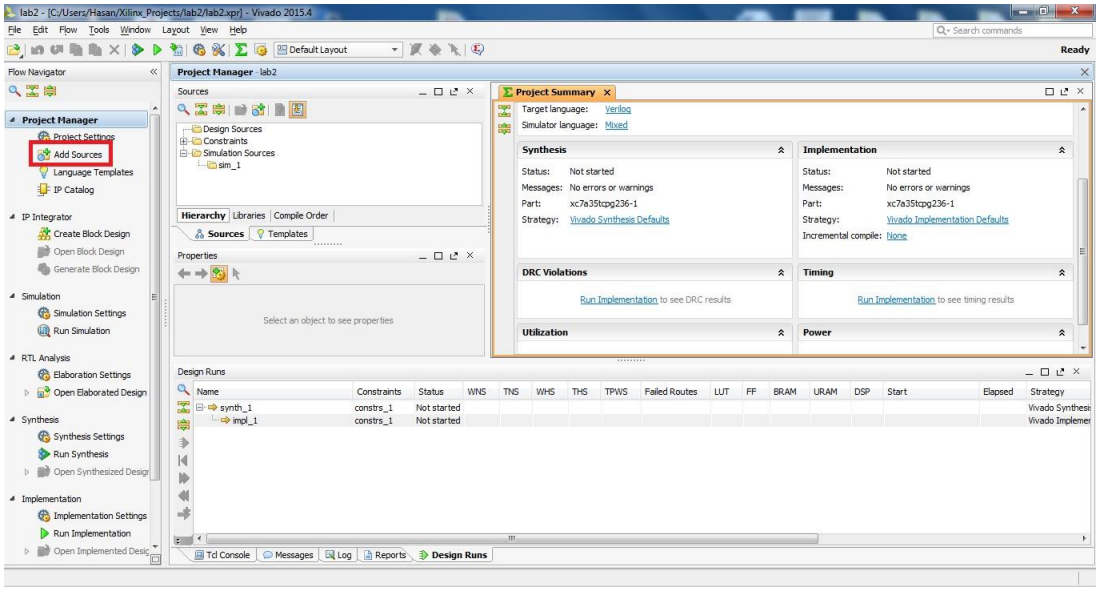
Daha sonra “Finish” butonuna basarak proje oluşturmayı tamamlayın.



Şekil 10: İlk projenizi başarıyla oluşturdunuz. Tebrikler!

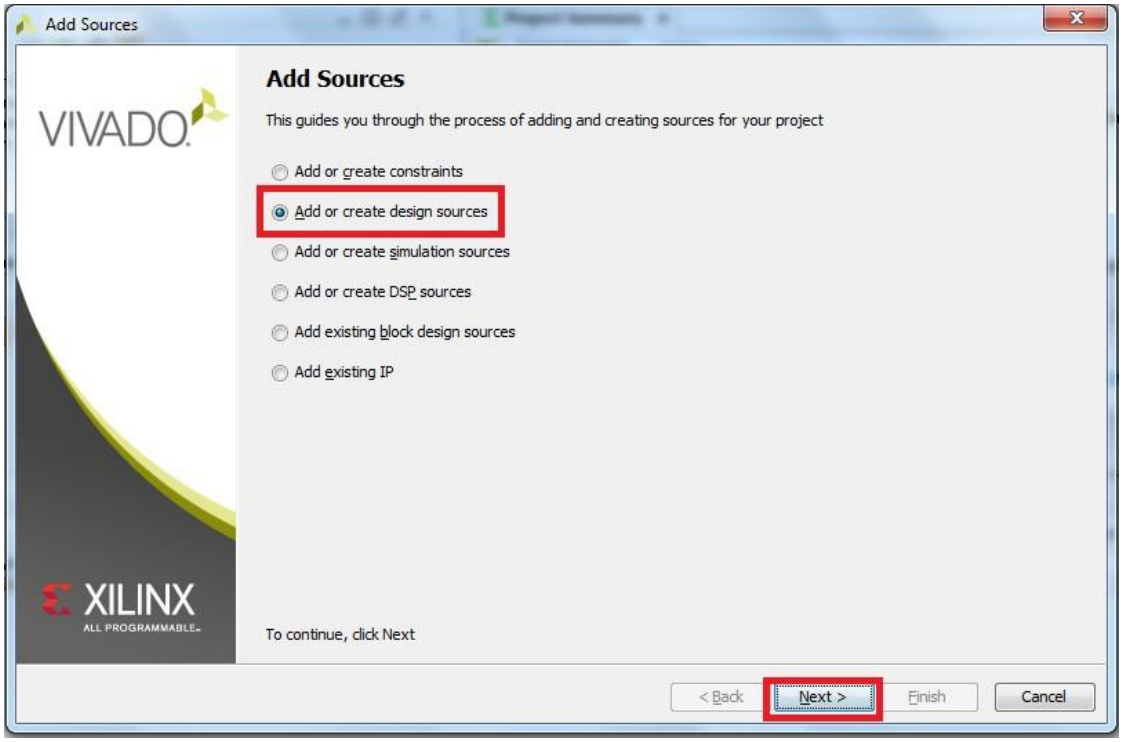
2.2 Modül Kaynak Dosyası Oluşturma

Oluşturduğunuz projede yeni bir modül oluşturmak için solda bulunan “Add Sources” butonuna basın.



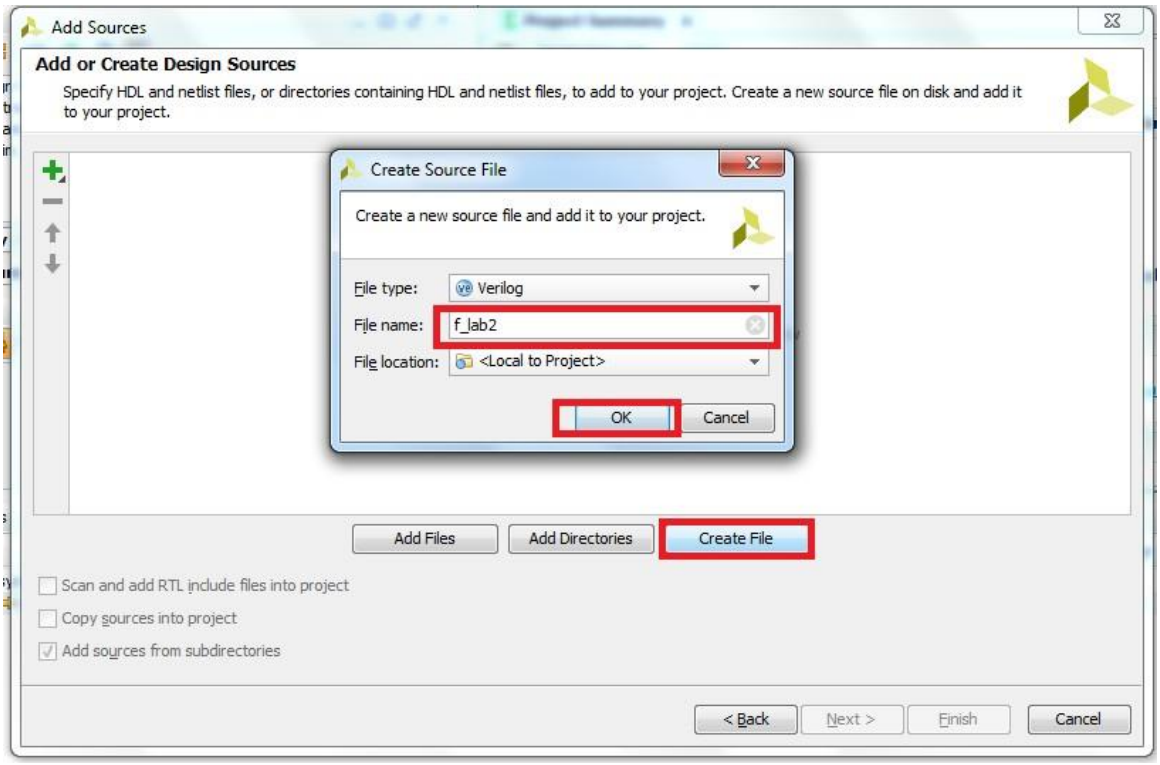
Şekil 11: Yeni bir kaynak dosyası oluşturma.

Açılan pencerede, bir tasarım dosyası oluşturmak istediğimizden, “Add or create design sources” seçeneğine tıklayın. Daha sonra “Next” ile bir sonraki sayfaya geçin.



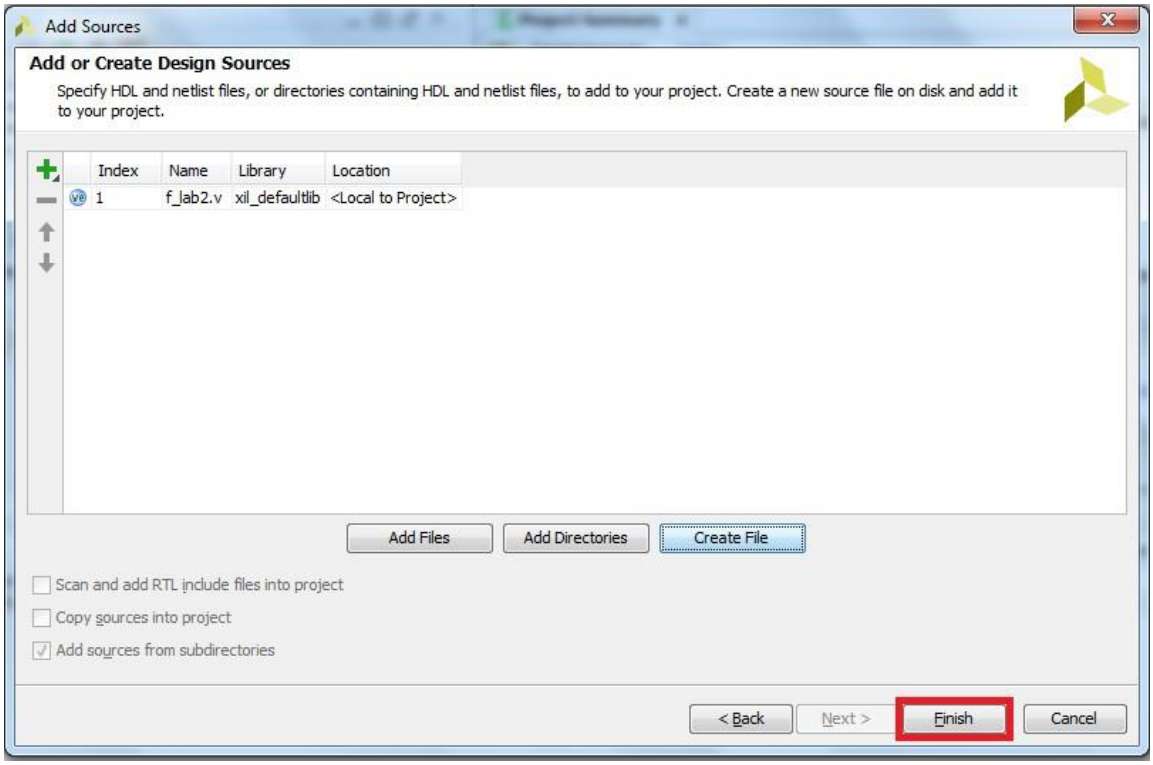
Şekil 12: Devam etmeden önce, gösterilen seçeneği işaretlediğinizden emin olun.

Bu sayfada “Create File” butonuna basarak yeni dosyayı yaratın. Bu butona bastığınızda açılan pencerede “File name” alanına oluşturmak istediğiniz dosyanın ismini girip, “OK” butonuna basın.



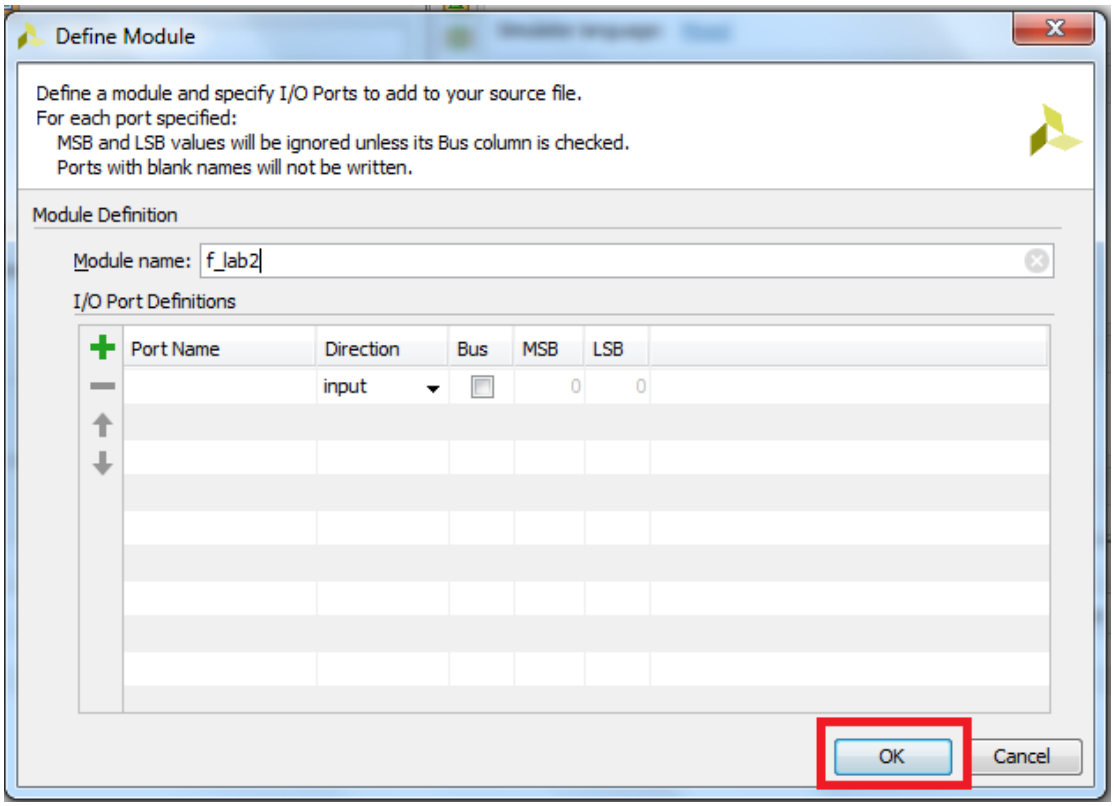
Şekil 13: Yarratmak istediğiniz dosyanın adını girin.

Dosyayı oluşturduktan sonra “Finish” butonuna basın.

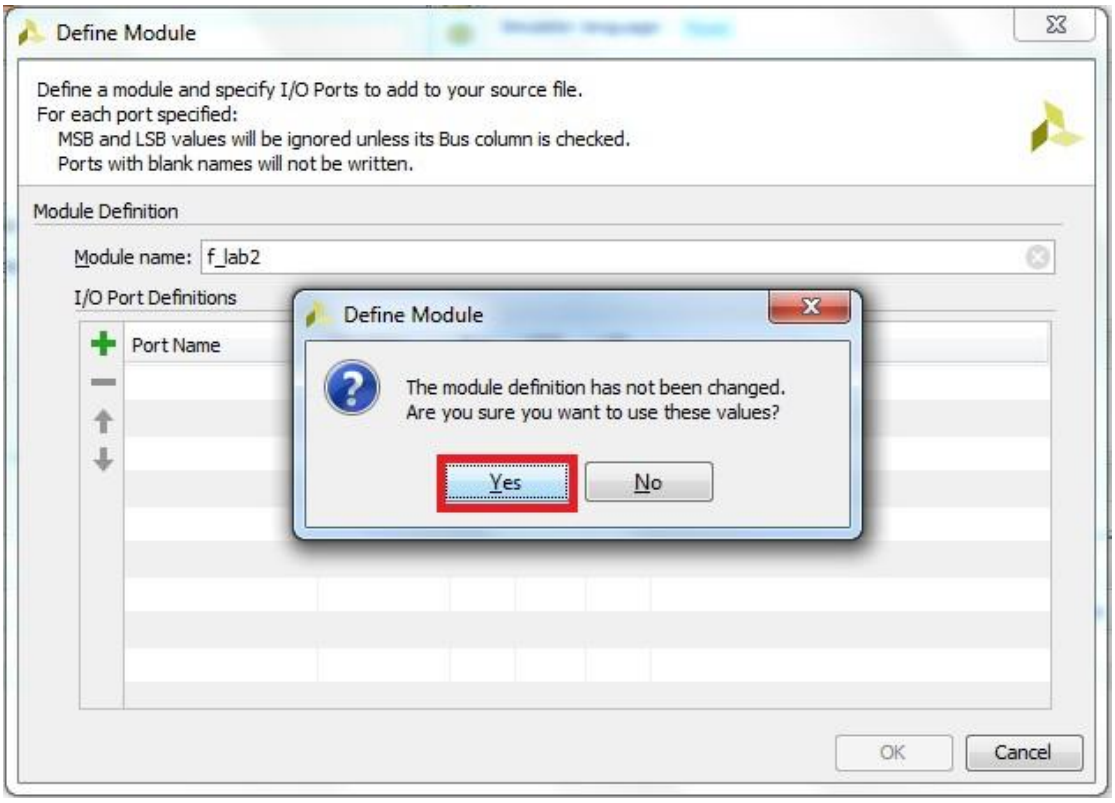


Şekil 14: Tüm adımları başarıyla tamamladığınızı düşünüyorsanız "Finish" butonuna basın.

Açılacak olan diğer iki pencerede sırasıyla “OK” ve “Yes” butonlarına basarak devam edin.

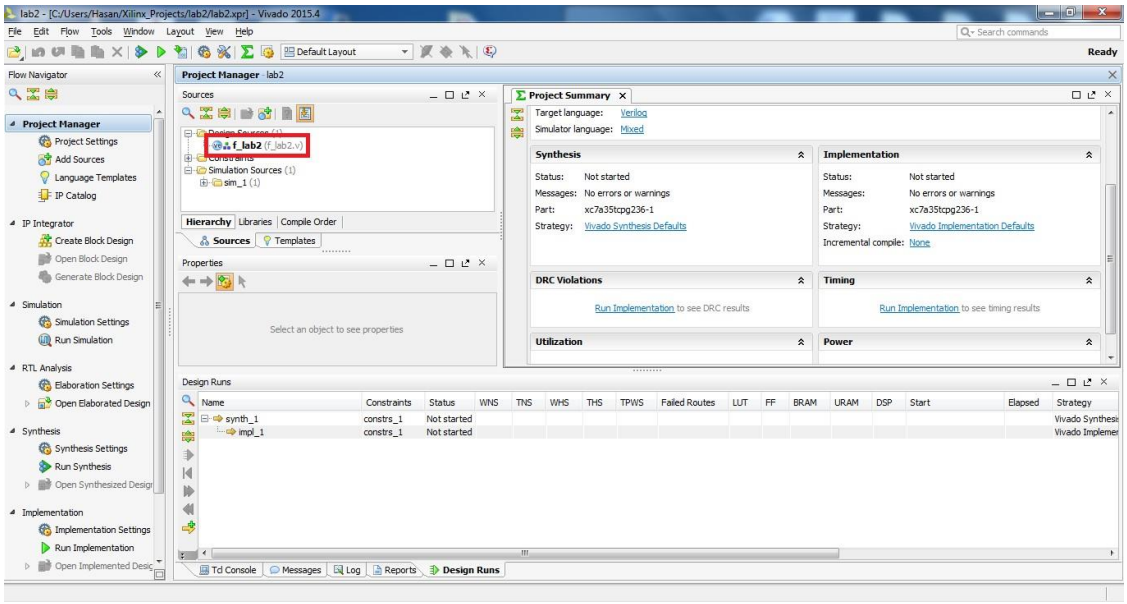


Şekil 15: Oluşturacağınız modülün giriş ve çıkışlarını bu pencereden belirleyebilirsiniz. Buna ilerleyen derslerde değineceğiz.



Şekil 16: Çıkan uyarı penceresinde "Yes" butonuna tıklayın. Bir şeyler ters gitmediyse kaynak dosyanızı oluşturmuş olmalısınız.

Oluşturulan dosya aşağıda gösterildiği gibi projenize eklenmiş olacaktır. Gösterilen yere çift tıklayarak dosyayı kod yazmak üzere açın.

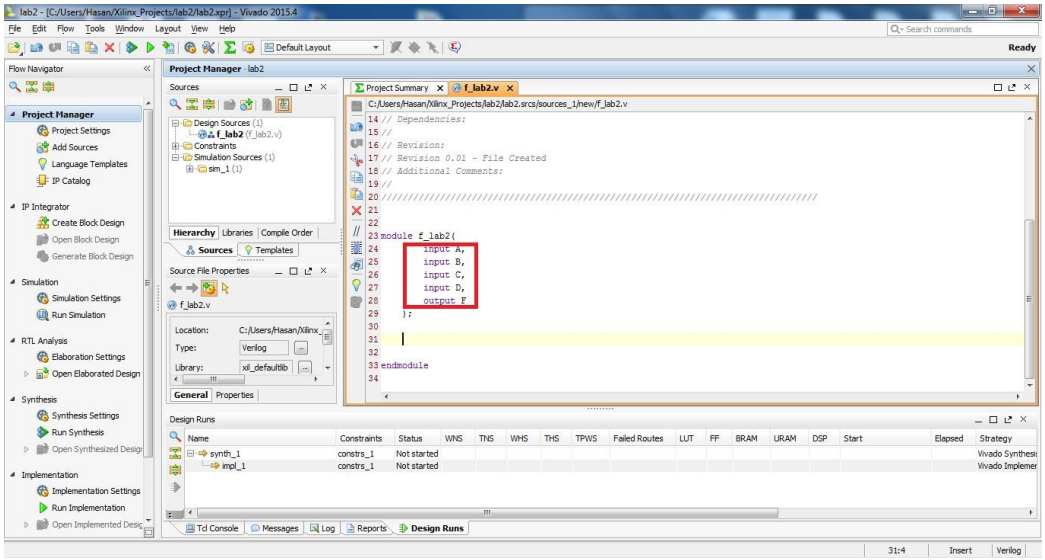


Şekil 17: Oluşturduğunuz kaynak dosyasını açın.

2.3 Kodlama

2.3.1 Giriş-Çıkış Sinyalleri

Dokümanın en başında verilen fonksiyonun A, B, C ve D isminde 4 adet tek bitlik girişi ve 1-bitlik F çıkışı vardı. Bu giriş ve çıkışlar, Verilogda aşağıdaki gibi tanımlanır.



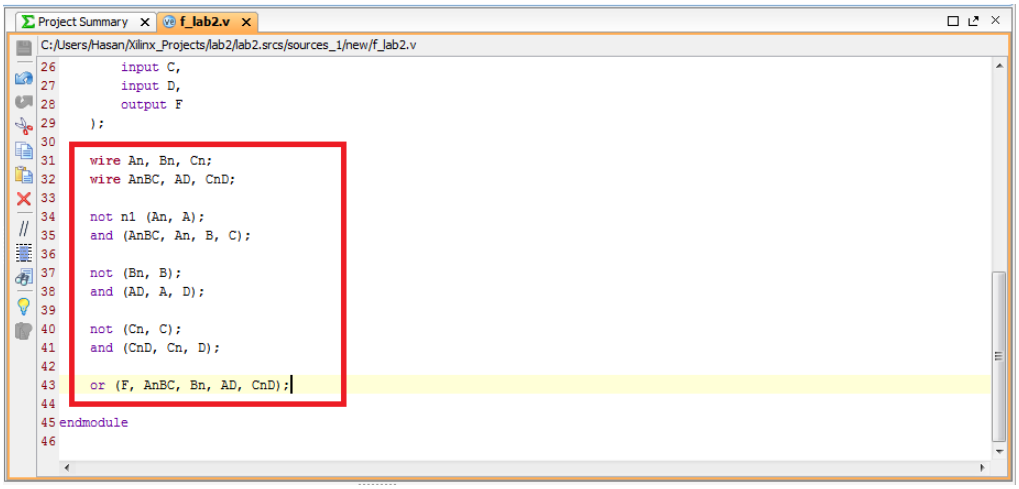
Şekil 18: Verilogda giriş-çıkış sinyallerinin gösterimi.

Giriş-çıkış sinyallerinin modül isminden hemen sonra gelen parantezlerin içine **virgül(,)** ile ayırılarak yazıldığına dikkat edin. Bu parantezlerden sonra da **noktalı virgül (;)** yer almalıdır.

Verilogda açıklama (Comment) satırları C dilindeki gibi "**//**" veya "**/***" veya "***/**" karakteriyle yazılır.

2.3.2 Devrenin Gerçeklenmesi

Devrenin çalışma biçimini tanımlayan kodu, giriş-çıkış sinyallerinden sonraki bölüme, "**endmodule**" keywordünden önce yazın.



Şekil 19: Yukarıda verilen fonksiyonu gerçekleyen Verilog HDL kodu.

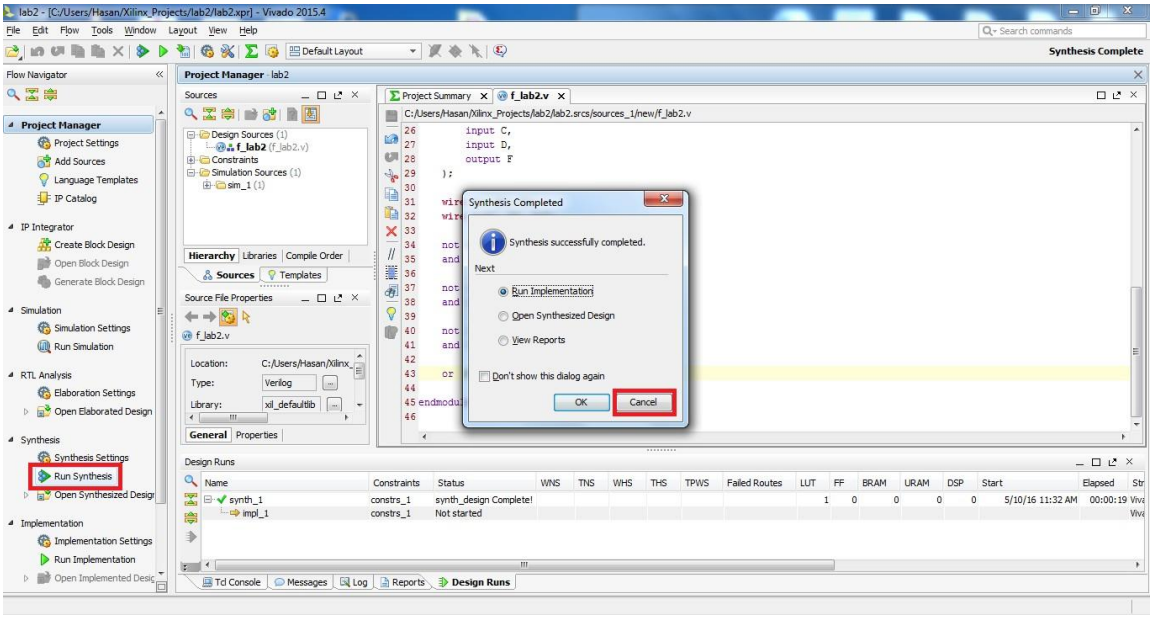
Verilog dilinde "**wire**" keywordünü kullanarak ara değerlerin tutulacağı sinyaller tanımlanabilir. Örneğin; kırmızı kutucuk içerisindeki 4. satırda yer alan (and) kapısına giriş olarak verilecek olan A'nın değilini tutmak için "**An**" isminde bir "**wire**" tanımlanmıştır.

Verilog'da temel mantık kapılarını yukarıda gösterildiği gibi isimlerini yazarak ("**not**", "**and**", "**or**", "**xor**" gibi) kullanabilirsiniz. Dilerseniz, kapı isminden sonra boşluk bırakarak bu kapıya bir isim verebilirsiniz (Örneğin, 3. satırdaki "**not**" kapısına "**n1**" ismi verilmiştir).

Satırın devamında, parantez içindeki ifadeler, kapının giriş ve çıkışlarını ifade etmektedir. Parantezin içine yazılan ilk sinyal, kapının çıkış sinyalini, diğer sinyaller ise kapının giriişlerini ifade etmektedir. Örneğin, kırmızı kutucuk içerisinde en alttaki **or** kapısı, **AnBC**, **Bn**, **AD** ve **CnD** sinyallerini giriş olarak alıp, bunların VEYA'sını **F** çıkışına vermektedir.

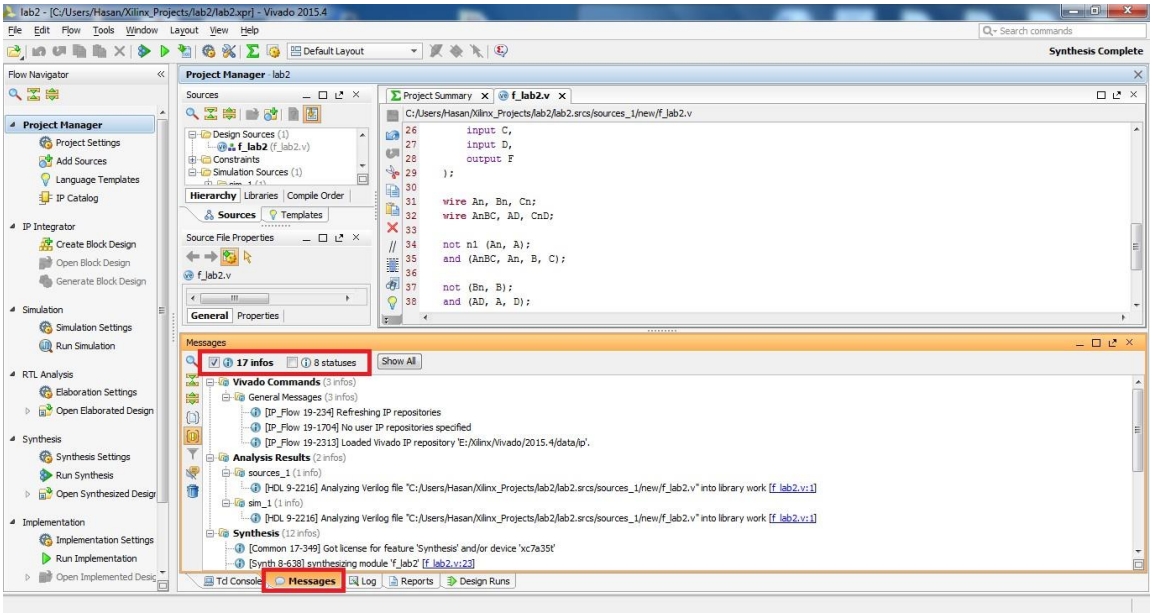
2.4 Modülün Sentezlenmesi

Gerçekleştirdiğiniz devreyi sentezlemek ve syntax hatası olmadığından emin olmak için solda yer alan "**Run Synthesis**" butonuna basın. İşlem başarılı bir şekilde tamamlandığında Şekil 20'de görülen ufak pencere açılacaktır. Bu pencereyi "**Cancel**" butonuna basarak kapatın.



Şekil 20: Devrenizin başarıyla sentezlendiğini gösteren pencere.

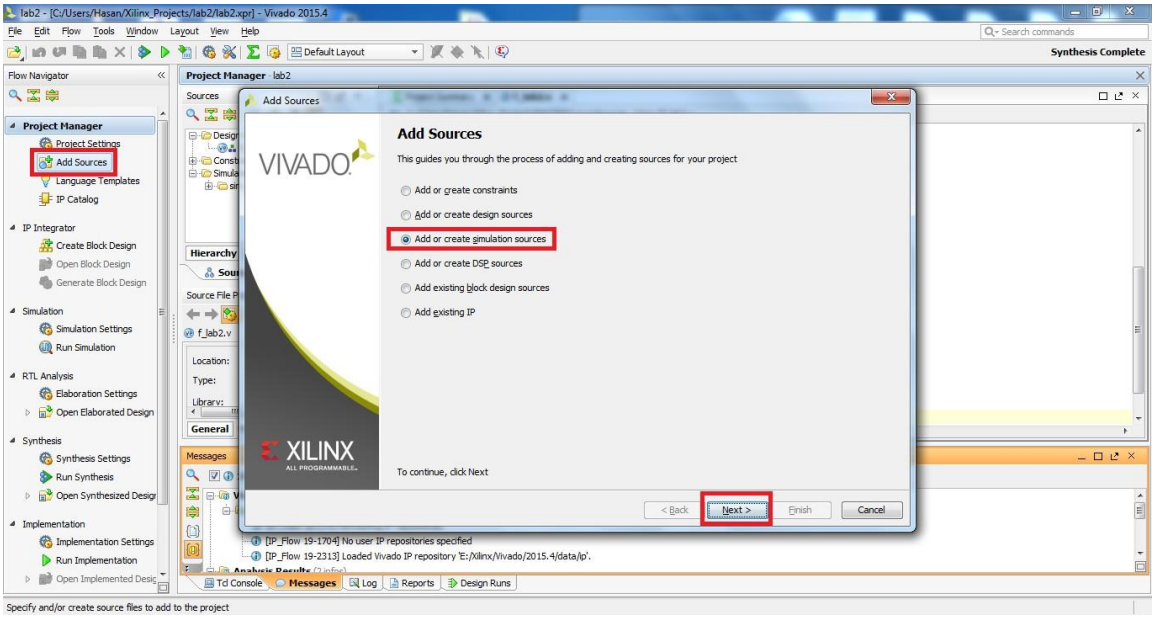
Devreyi gerçekleştirirken hata yapmadığınızdan emin olmak için uyarı (warning) mesajlarını kontrol edin. Pencerenin en altında, "**Messages**" sekmesine geldiğinizde, gerçeklediğiniz devre için herhangi bir hata veya uyarı mesajı alıp almadığınızı görebilirsiniz. Şekil 21'de gösterilen pencerede, yalnızca bilgi (info) ve durum (status) mesajları bulunuyor.



Şekil 21: Tasarlanan devre için "Messages" penceresinin görünümü.

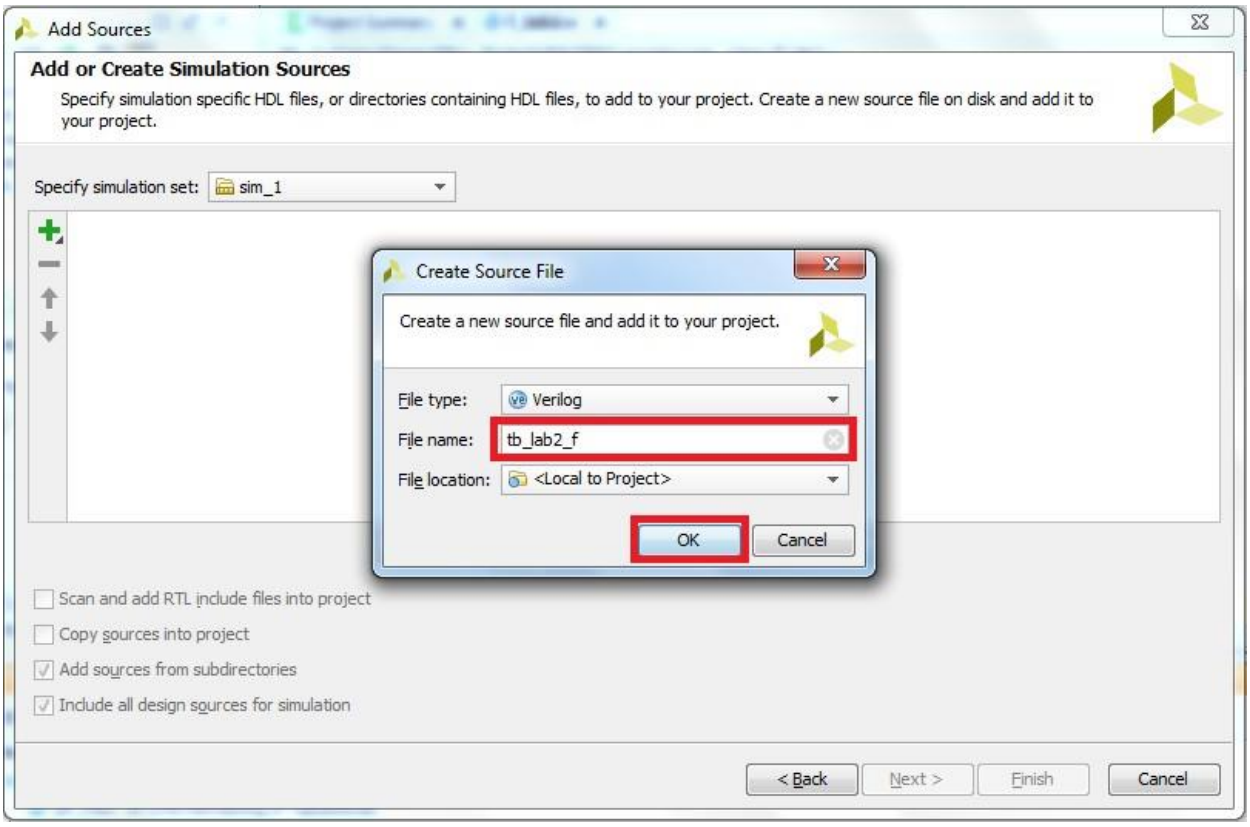
2.5 Devrenin Doğrulanması

Gerçekleştirdiğiniz devrenin istenildiği gibi çalıştığından emin olmak için bir "testbench" hazırlayıp bu devrenin simülasyonunu yapabilirsiniz. Yeni bir simülasyon dosyası oluşturmak için yine pencerenin solundan "**Add Sources**" butonuna tıklayın. Açılan pencerede bu sefer "**Add or create simulation sources**" seçeneğine tıklayın.



Şekil 22: Devreniz için bir testbench oluşturmak.

Daha sonra, tasarım modülü oluşturur gibi, "**Create File**" butonuna basarak yeni bir dosya oluşturun. Dosyanızı"tb_<modülünüzün_ismi>" şeklinde isimlendirmek, daha sonra bunun bir simülasyon dosyası olduğunu kolayca ayırt etmenize yardımcı olacaktır.

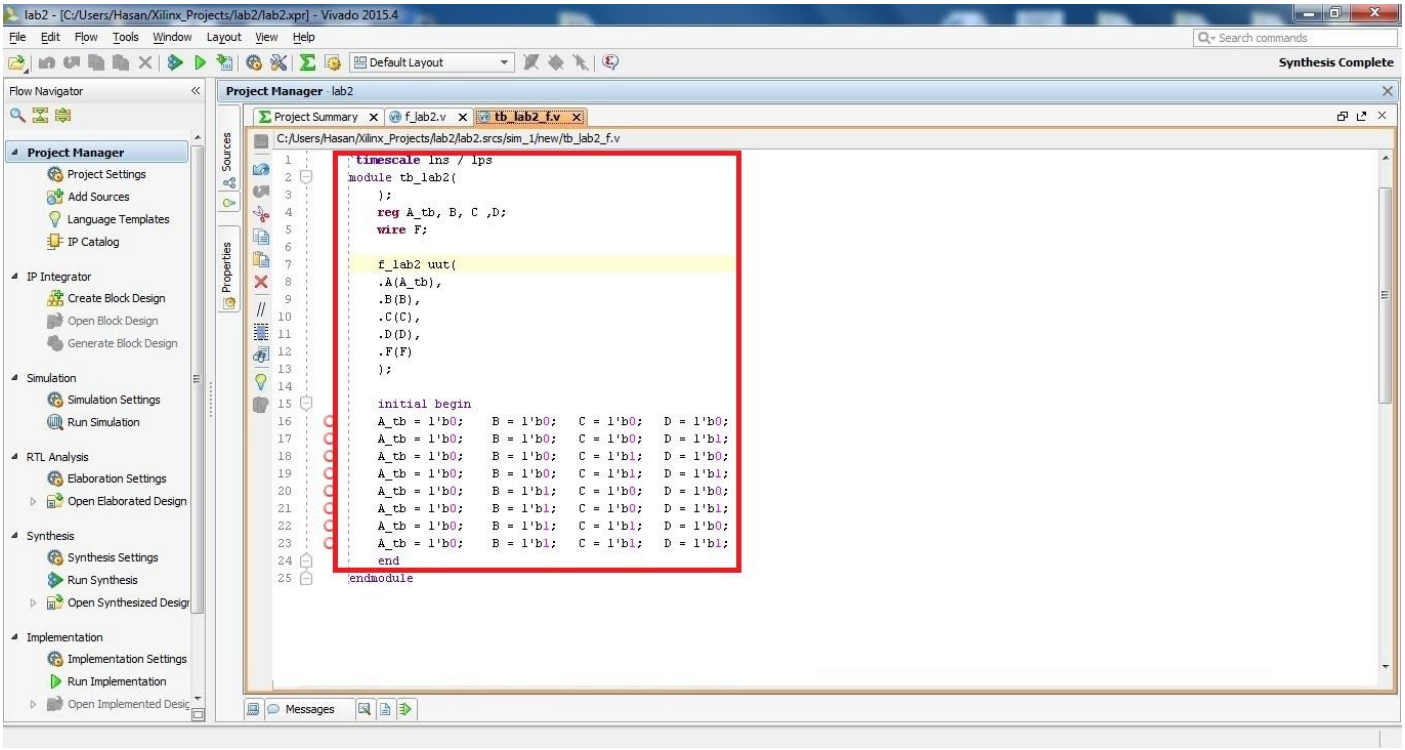


Şekil 23: Testbench dosyasını isimlendirme aşaması.

Simülasyon dosyaları da, temelde birer Verilog modülü olduğundan Verilog syntaxına uymak zorundadır. Başka bir deyişle, testbench dosyanızı, herhangi bir giriş-çıkış sinyali olmayan bir Verilog modülü gibib düüşünebilirsiniz. Bu nedenle testbenchinizin ilk satırı:

module [MODUL_ISMI] ();

şeklinde olmalıdır. "**endmodule**" keywordünden önce, aşağıda kırmızı kutucuk içerisinde gösterilen kod parçasını yazın.



Şekil 24: Verilen devre için testbench dosyası.

Bu test modülü içerisinde, doğrulamasını yapacağınız modülün (bu örnek için önceki sayfalarda gerçekleştiğiniz "f_lab2" modülü) bir örneğini (instance) oluşturun. Bu işlemi Java'da bir sınıfın nesnesini oluşturmak veya oluşturduğunuz devrenin girdi ve çıktılarına, test kablolarını bağlıyormuşsunuz gibi düşünebilirsiniz. "f_lab2" modülünün bir örneğinin nasıl oluşturulacağı Şekil 24'te 3-9. satırlar arasında gösterilmiştir. "uut" (Unit Under Test) ismini vereceğiniz bu örneğin, giriş-çıkış portlarına uygun sinyalleri bağlıyoruz. Verilogda bu bağlantılar:

.[PORT_ISMI]([BAĞLANILACAK_SINYAL])

şeklinde yapılmaktadır. Örneğin; "A" ismindeki giriş portuna "A_tb" ismindeki sinyal ".A(A_tb)" kod parçasını kullanarak bağlanmıştır. Bağlanacak sinyaller portlarla aynı isme sahip olabilir.

Simülasyon sırasında değer atanacak sinyaller "**wire**" yerine "**reg**" olarak tanımlanmalıdır. Yukarıdaki örnekte, tüm girişlere değerler atandığından, 4 giriş de "reg" olarak tanımlanmış sinyallere bağlanmıştır. "uut" modülünün çıkışı olan F'e herhangi bir değer atamak söz konusu olmadığından bunu "wire" olarak tanımlayacağınız **F** sinyaline bağlayın..

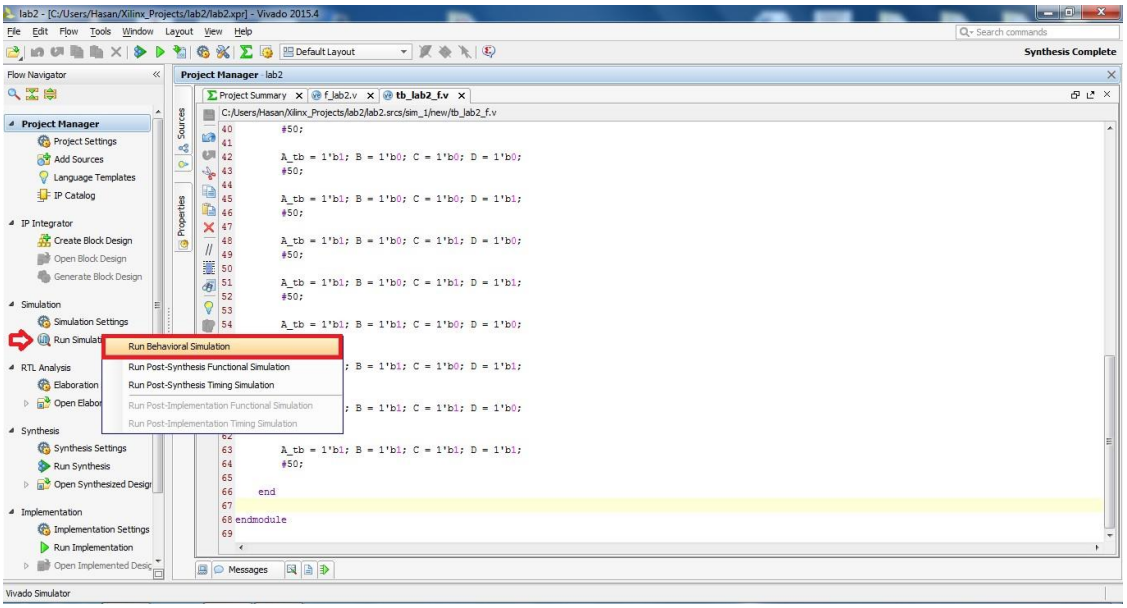
Kullanılacak sinyalleri tanımlayıp, bunları test edilecek modüle uygun bir şekilde bağladıktan sonra bir "initial" bloğu oluşturun. Initial bloğu içerisinde tanımlanan işlemler, simülasyon başladığı gibi yapılır. Verilog'da blokların (Java'da scope) başlangıç ve bitiş noktalarını belirtmek için "**begin**" ve "**end**" keywordleri kullanılır (Java'daki "{" ve "}"ifadeleri gibi).

Gerçekleştiğiniz devrenin bekelndiği gibi çalıştığından emin olmak için, giriş sinyallerinin tüm durumlarında beklenilen çıkış sinyalinin elde edilip edilmediğini kontrol etmelisiniz. 4 adet 1 bitlik giriş sinyali $2^4 = 16$ farklı durum oluşturur. Bu 16 durumu (0000, 0001, 0010, 0011, ...) tek tek "initial" bloğunun içine yazın. Durum değişimleri arasına arasına "#50;" ifadesini koyun. Bu ifade, iki durum değişimi arasında 50

nanosaniye bekleneceğini göstermektedir.

Soru: Bu beklemeyi eklemediğiniz durumda ne gibi bir sorunla karşılaşabilirsiniz? Tartışın.

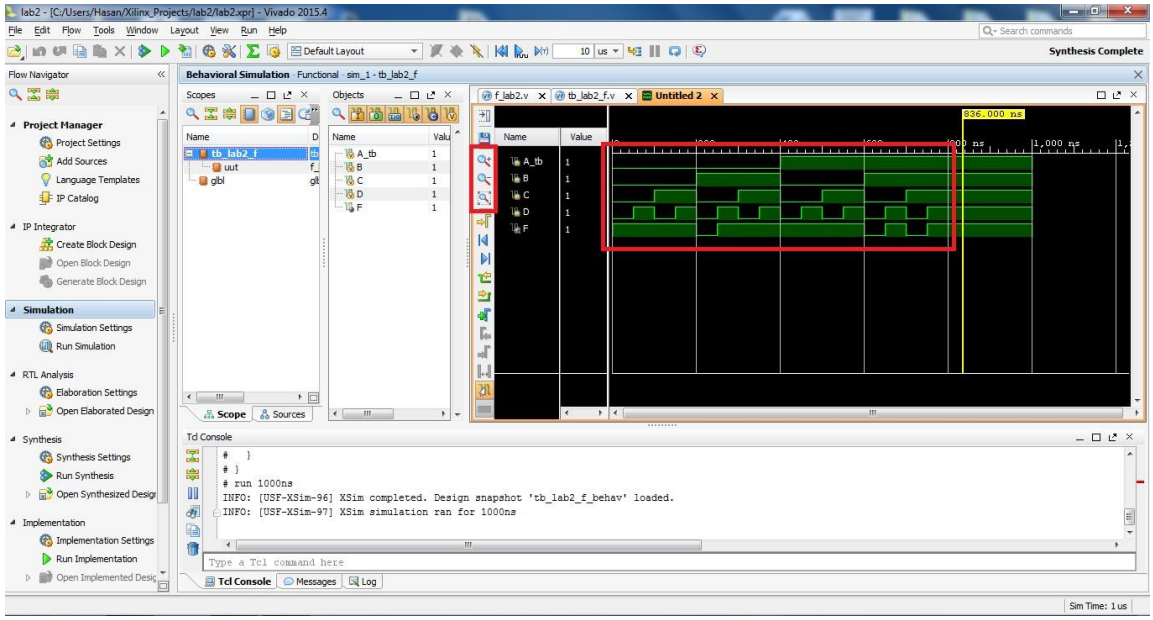
Simülasyonu başlatmak için pencerenin solunda yer alan "**Run Simulation**" butonuna basın, ardından "**Run Behavioral Simulation**" seçeneğine tıklayın.



Şekil 25: Yazdığınız testbench kodunu kullanarak devrenizin doğru çalıştığına emin olun

Simülasyon sonucu olarak aşağıdaki gibi bir kare dalga grafiği (Waveform) göreceksiniz. Bu waveform üzerinde düşey eksende, verilen girişler (A_tb, B, C, D) için çıkışın (F) doğruluğunu kontrol edin.

Not: Simülasyon tamamlandığında waveform şekildeki gibi görünmeyecektir. Farenizle "zoom out" yaparak sinyallerin uygun bir şekilde görünmesini sağlayabilirsiniz.



Şekil 26: Tasarladığınız devreye ait dalga formu.