T.C.

GAZİ ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

ELEKTRİK – ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

EE – 303

SAYISAL TASARIM LABORATUVARI

DENEY FÖYÜ

HAZIRLAYANLAR

Arş. Gör. Aynur KOÇAK

Arş. Gör. Kezban KOÇ

DENEY 1: Vivado Programı ve VHDL Kodlarının Kullanımı

DENEY 1.1. Vivado Programı Kurulumu

EE-303 Sayısal Tasarım Laboratuvarı dersi süresince FPGA yazılımı için **"Xilinx Vivado"** programı kullanılacaktır. Kişisel bilgisayarınıza programı yükleyebilmeniz için izleyeceğiniz adımlara aşağıdaki linkten ulaşabilirsiniz:

https://tf-eem.gazi.edu.tr/view/page/286670/ders-dokumanlari

Linkte yer alan Ders Dokümanları kısmından Sayısal Tasarım Lab. sekmesini seçerek "Vivado Kurulum Kılavuzu" bölümünde yer alan aşamaları izleyebilirsiniz.

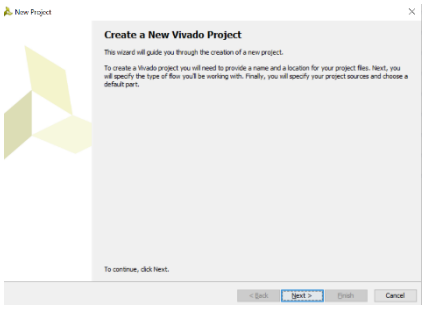
1.2. Yeni Proje Oluşturma

Buradan sonra anlatılanlar **Xilinx Vivado Design Suite** (WebPack) kullanılarak **Xilinx Vivado 2014.4** sürümü baz alınarak anlatım yapılacaktır. Ve anlatım Digilent firmasının üretmiş olduğu üretmiş olduğu **BASYS 3** FPGA kartı için yapılacaktır.

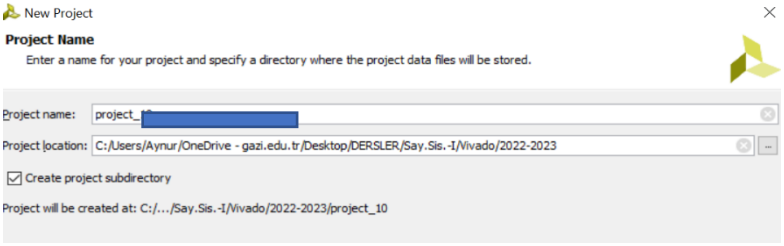
Vivado programını başlattığınızda karşınıza;

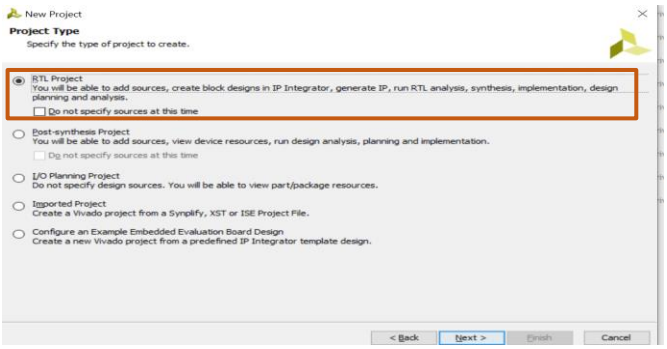
Soldaki görünümde bir ekran gelecektir. Bu ekrandan "Create New Project" sembolünü seçiyoruz.

Sonraki adımda ise sağ tarafta bulunan görselde "Next" diyerek devam ediyoruz.



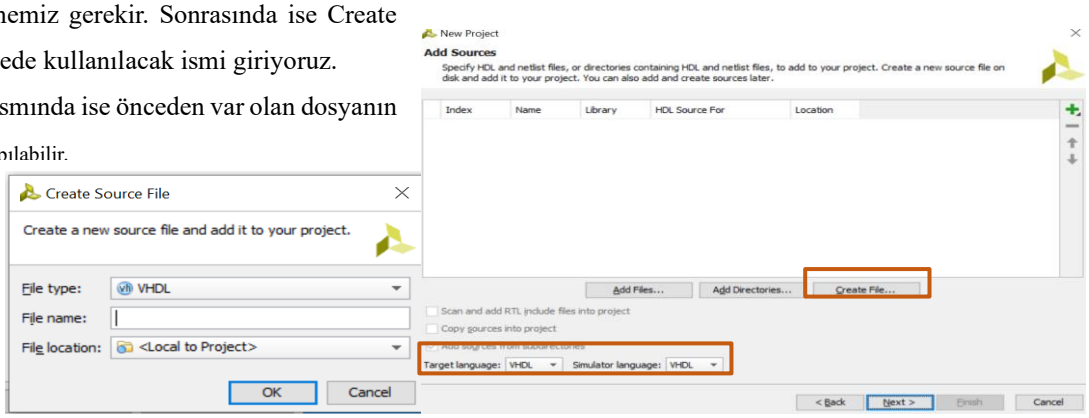
Bir sonraki adımda projenizin ismini ve yerini belirtiyoruz.

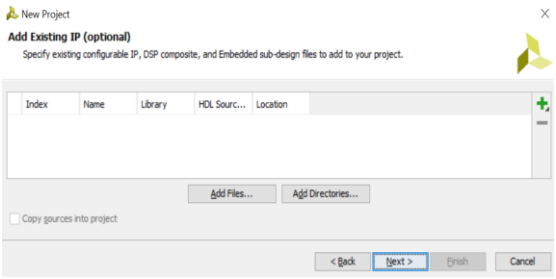


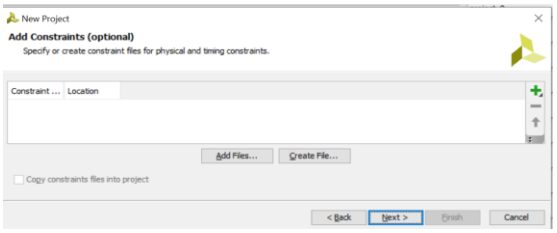


Bir sonraki adımda ise oluşturulacak projenin türü seçilir. Bu ekranda RTL Project seçeneğini seçerek next ile devam ediyoruz.

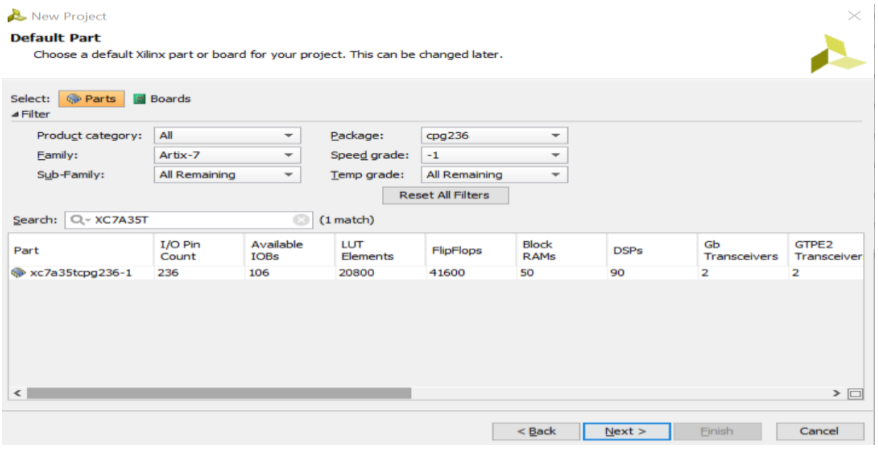
Sağdaki ekranda Target language kısmını VHDL olarak seçmemiz gerekir. Sonrasında ise Create File ile projede kullanılacak ismi giriyoruz. Add files kısmında ise önceden var olan dosyanın eklemesi yapılabilir.

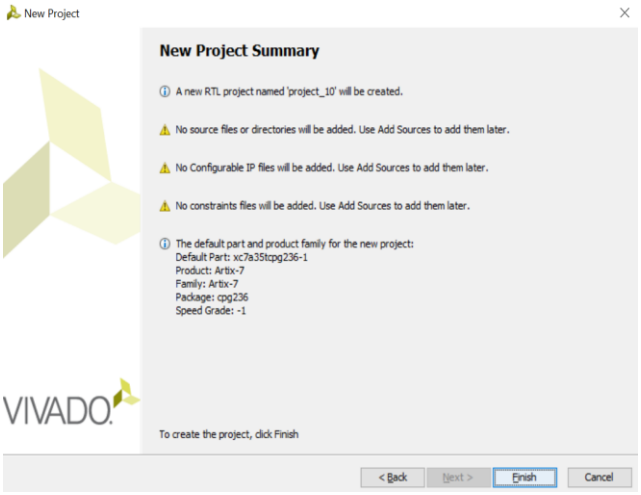






Daha sonraki iki adımda ise var olan IP'lerin eklendiği ve kısıtlamaların belirlendiği ekranlar yer almaktadır. Burada bir değişiklik yapmadan iki ekranı da next diyerek devam ediyoruz.





Bu ekranda ise proje ile ilgili yaptığımız işelmlerin özeti yer almaktadır. Ekranda görünen ünlem ifadeleri ise sırasıyla, kaynak dosyası eklenmemesi, IP eklemesinin yapılmaması ve kısıtlama dosyalarının eklenmemesi anlamına gelmektedir.

Dosyalarda I/O portları sonrasında tanımlanabilir.

DENEY 1.3. FPGA Nedir?

- **F**ield **P**rogrammable **G**ate **A**rray (Sahada Programlanabilir Kapı Dizileri)
- Programlanabilir lojik kapılar ve programlanabilir ara bağlantılardan oluşan entegre devreleri denilebilir. Yani içerisindeki transistorlar birbirinden bağımsızdır. Kullanıcının tasarımına göre bu transistorlar birbirine bağlanabilir ve istenen fonksiyonlar çalıştırılabilir.

FPGA Özellikleri

I. Paralel işlem yapma kabiliyetine sahiptir. Paralel işlem yapma; aynı anda birden fazla işlem yapabilme anlamına gelir. Bu özellik diğer entegre devrelerine göre FPGA'nın daha hızlı çalışmasını sağlar.

II. Sahada yani çalıştığı ortamda programlanabilir. FPGA'ler defalarca programlanabilir bu da tasarımcıya zaman ve maliyet açısından kolaylık sağlar.

III. FPGA'ler ile yapılan tasarım test edilebilir. Yani tasarımcı bilgisayar ortamında simülasyon yapabilmektedir. Bu özellik; tasarımcının programı FPGA'e yüklemeden önce test edip doğrulayabilme olanağı sağlar.

IV. Diğer önemli bir özellik ise FPGA'e işlemci gömülmesidir. İşlemci gerektiren tasarımlarda FPGA içerisine işlemci gömülebilmekte ve C/C++ gibi yazılım dilleri ile bu işlemciler programlanabilmektedir. Tasarımcı birden fazla işlemcide kullanabilir. Bununla birlikte içerisinde fiziksel olarak gömülü işlemciler bulunduran FPGA'ler de vardır.

Genel Yapısı

FPGA genel olarak üç birimden oluşur. Bunlar;

- Mantık blokları
- Giriş/Çıkış Birimleri
- Ara Bağlantı Yolları

FPGA üzerinde bulunan mantık blokları içerisinde LUT denilen ve gerekli mantıksal işlemleri yerine getiren küçük bellekler bulunmaktadır. Bu bellekler minimum 4 girişlidir ve giriş sayısına bağlı olarak işaret edilen bellek sayısı değişim gösterir. Kullanıcı, mantık kapılarını tasarlamak istediği sayısal devreye göre dizayn eder. Bloklar içerisinde aynı zamanda 'multiplexer' denilen çoklayıcı devreler ve bir bitlik saklayıcı görevi gören flip-floplar bulunmaktadır. Kullanıcının tasarlamak istediği devre doğrultusunda mantık blokları birbirleri ile ara bağlantı yolları vasıtasıyla bağlantı kurar ve bir sayısal devre tasarlanmış olur. Yapılarında bulunan giriş/çıkış birimleri sayesinde çevresel birimlerle haberleşme, veri alma veya tasarlanan devre doğrultusunda çıkış verme özelliklerine sahiptirler. Bu sayede FPGA'ler üzerinde sinyal veya görüntü işlenebilir, üzerinde bulunan sensörler vasıtasıyla birçok ölçüm yapabilir ve yaptığı bu ölçümler doğrultusunda programlandığı amaca hizmet edebilir. Bir FPGA'in performans ölçeği yapısında bulunan mantık bloklarının sayısına bağlıdır. Yapılarında bulunan binlerce mantık bloklarının doğru yapılandırılmasıyla çok karmaşık ve büyük tasarımlar ortaya çıkarılabilir.

Performans

FPGA'ler, kullanıcıya paralel veri işleme olanağı sağlar. Paralel çalışma kavramı, birbirinden bağımsız olarak yerine getirilmesi gereken işlemlerin aynı anda farklı bloklar üzerinden gerçekleştirilmesidir. Geleneksel bilgisayar mimarisine göre çalışan diğer sistemler aynı anda farklı işlemler yerine getiremezken, yüksek hız ve eş zamanlı veri işleme gerektiren uygulamalarda FPGA'ler tercih edilmekte. Paralel işlem yapılabilme özelliği, FPGA'lere düşük frekansta yüksek hız özelliği kazandırmakta. Normal bir işlemcinin 1 GHz seviyesinde

gösterdiği performansı, FPGA 100 MHz civarındaki bir frekansta gösterebilir. FPGA'ler aynı zamanda düşük güç tüketimine sahiptirler. Bu sayede enerji kaynaklarının sınırlı olduğu alanlarda tercih edilirler. Aynı zamanda birçok dijital devrenin bir araya gelmesiyle yapılacak işlemleri tek başına yerine getirebilir. Bu sayede küçük bir alanda yüksek işlem yapmaya olanak sağlamış olur.

Kullanım Alanları

- Havacılık ve Savunma Sistemleri: FPGA üzerinde dalga formu üretimi, sinyal ve görüntü işleme yapılabildiği için tercih edilir.
- Tüketici Elektroniği: Yakınlaştırılmış ahizeler, dijital düz panel ekranlar, bilgi cihazları, ev ağı ve konut set üstü kutuları gibi yeni nesil, tam özellikli tüketici uygulamalarında kullanılır. Uygun maliyetli çözümler üretilebilir.
- Ses: Çok çeşitli ses, iletişim ve multimedya uygulamaları için daha yüksek esneklik, daha hızlı piyasaya sürme süresi ve daha düşük tekrarlanan mühendislik maliyetleri sağlar.

Diğer kullanım alanları:

- Tıbbi Görüntüleme
- Kablolu İletişim
- Kablosuz İletişim
- Endüstri
- Otomotiv
- Hesaplama ve veri depolama

Yüksek hızı, yeniden programlanabilir olması, paralel işlem yapabilmesi, sinyal ve ya görüntü işleyebilmesi ve daha birçok özellikleri sayesinde FPGA'ler günümüzde çeşitli alanlarda kullanılmaktadır.

DENEY 1.4. VHDL Kodlarının Kullanımı

- VHDL "VHSIC (Very HighSpeed Integrated Circuit) HardwareDescription Language" Oldukça hızlı tümleşik devre donanım tanımlama dilidir.
- VHDL modülü en az 3 parçadan oluşur. Bunlar sırasıyla;
  - ✓ Kütüphane tanımı,
  - ✓ Modülün dış dünya ile iletişimini sağlayan giriş/çıkış tanımları,
  - ✓ Modülün çalışmasını belirten mimari tanımıdır.
- VHDL case sensitive **olmayan** bir dildir. Yani büyük küçük harf ayrımı yapmaz. Fakat kısıt dosyalarında büyük küçük harf duyarlılığı vardır.
- VHDL ne kadar case sensitive olmasa da kod okunabilirliği açısından,
  - ✓ VHDL komutları BÜYÜK HARF
  - ✓ Değişkenler küçük harf
  - ✓ Modül isimleri İlk harfi büyük olacak şekilde yazılması tavsiye edilir.

##LIBRARY

VHDL tasarımının ilk satırında LIBRARY ieee; satırı bulunur ve kütüphane tanımlaması olarak kullanılır. İkinci satırında ise USE ieee.std_logic_1164.ALL; kullanılır ve ieee kütüphanesinin altında bulunan std_logic_1164 paketinin tüm modüllerinin tasarıma ekleneceği anlamına gelir. Daha sonra ise modülün dış dünya ile bağlantı tanımları olan ENTITY bulunmaktadır.

##ENTITY

Entity isimlendirmesi;

-- bir harf ile başlamak zorundadır.

-- harften sonra rakam ve alt çizgi ile devam edebilir.

-- arka arkaya alt çizgi (__) kullanılamaz.

-- alt çizgi ile bitemez.

-- VHDL özel isimlerinden oluşamaz.

- ENTITY içinde eğer varsa giriş çıkışlar PORT komutu ile tanımlanır. Eğer port olmayacaksa tanımlanmayabilir.
- PORT tanımlamaları parantez içerisinde yapılır. İsmi yazıldıktan sonra iki nokta (:) kullanılır ve yön belirtilir. Daha sonra portun tipi belirtilir. Her port tanımlandıktan sonra; ile sonlandırılır. Fakat son portta; kullanılmaz çünkü parantezden sonra bulunmaktadır.

Örnek:

```
ENTITY sayısal_tasarim IS
                    PORT (  a: IN STD_LOGIC;
                            b: OUT STD_LOGIC_VECTOR(0 TO 6)
                            );
ENTITY sayisal tasarim;
```

<işaret modu> --> VHDL de işaretin yönünü belirtir ve 3 tip işaret modu vardır. bunlar;

{

IN --> Giriş için kullanılır

OUT --> Çıkış için kullanılır

INOUT --> hem giriş hem de çıkış olcaksa kullanılır.

}

<veri tipi> --> VHDL de bulunan verilerin tiplerinin nasıl belirtileceğini belirler.. Bunlardan en çok kullanılanları;

{

BIT : 2 seviyeli lojik ('0' ve '1')

BIT_VECTOR : 2 seviyeli lojik dizi (bus,vectör)

STD_LOGIC : standart lojik tanımlamasıdır ('0' , '1' ve 'Z')

STD_LOGIC_VECTOR : standart lojik vektörüdür.

--NOT: vektörler tanımlanırken;

STD_LOGIC_VECTOR (0 to 15) --> LSB TO MSB

STD_LOGIC_VECTOR (15 DOWNTO 0) --> MSB DOWNTO LSB şeklinde tanımlanır.

}

VHDL tasarımında son kısımda ise ARCHITECTURE bulunur ve devrenin mimari yapısını belirtir.

##ARCHITECTURE

Lojik fonksiyon işlevi ARCHITECTURE tarafından belirlenir. Her ENTITY için bir ARCHITECTURE tanımlanır.

{

ARCHITECTURE <architecture ismi> OF <entity ismi>
BEGIN
[lojik fonksiyonun yapacağı işler]
END <architecture ismi>

}

OPERATÖRLER

{--Atama operatörleri:

<= : operatörü Sinyal atamak için kullanılır
sinyal <= değer; şeklinde
:= : operatörü Variable/Constantlar için kullanılır
constant := değer; şeklinde

}-- Lojik operatörler:

AND : Ve
NOT : değil
OR : veya
NAND : ve değil
NOR : veya değil
XOR : özel veya
XNOR : özel veya değil

ÖRNEK:

```
y <= NOT A AND B; --> A'B
y <= NOT(A AND B); --> (A.B)'
```

-- Birleştirme operatörü

& birleştirme operatörüdür ve kullanımı;

```
x <= '1';
y <= x & "100"; --> y çıktısı y=1100 olur.
```

--OTHERS deyimi

OTHERS => '0' şeklinde kullanılır. Farklı bir durum belirtilmedikçe tüm bitler 0 olsun anlamına gelir. 32'b0 ifadesi ile aynı anlama gelir.

ÖRNEK:

{

led : OUT STD_LOGIC_VECTOR(15 DOWNTO 0); olsun ve atama

```
led <=
(
    0 => '1',
    3 => '1',
    8 DOWNTO 5 => '1',
    12 TO 15 => '1',
    OTHERS => '0'
);
```

--> ifadenin sonucunda 0,3,8,7,6,5,12,13,14,15 bitleri 1 değeri olsun geri kalan bitler 0 değeri alsın anlamına gelir.

EK – 1: DENEY RAPORUNUNUN HAZIRLANIŞI

1. Deney raporları deney föyüne ek olarak A4 boyutundaki çizgisiz kâğıda hazırlanacaktır. Kâğıdın sol kenarından 2,5 cm, diğer kenarlardan 1 cm boşluk bırakılarak çerçeve çizilecek ve rapor bu çerçeve içerisine el yazısı ile yazılacaktır.
2. Rapor içeriği aşağıda verilen konu başlıklarını içerecektir.
3. Deney raporlarında çizilecek grafiklerde milimetrik kâğıt kullanılacaktır.
4. Raporlar deney yapıldıktan bir hafta sonra teslim edilecektir. Öğrenci 1. Haftadan sonra da raporlarını teslim edebilir fakat geç teslim edenlerden gün başına %5 not kesilir. 2. Haftadan sonra kesinlikle rapor kabul edilmez.
5. Deney ve rapor birlikte değerlendirilmektedir. Deneye katılmayan rapor veremez. Deneye katılan öğrenci rapor vermez ise o haftaki deneyden not alamaz.
6. Deney raporunda bir sonraki sayfada verilen rapor kapağı kullanılacaktır.
7. Rapor düzeni “Kapak + Deney Föyü + Rapor” şeklinde sıralanacak ve plastik kapaklı dosyaya yerleştirilerek teslim edilecektir.

T. C.

GAZİ ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ BÖLÜMÜ

EE – 303

SAYISAL TASARIM LABORATUVARI

DENEY RAPORU

Deneyin yapılış tarihi :……

Deney No :……

Deneyin Adı :……

……

Hazırlayan:

Öğrenci No :……..

Adı Soyadı :……..

Sınıfı :……..