T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FPGA YAPILARI İLE DİJİTAL OSİLOSKOP GERÇEKLEMESİ

YÜKSEK LİSANS TEZİ

Berkant Başa

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜH.**
Tez Danışmanı	**:**	**Yrd. Doç. Dr. Murat İSKEFİYELİ**

Ocak 2014

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FPGA YAPILARI İLE DİJİTAL OSİLOSKOP GERÇEKLEMESİ

YÜKSEK LİSANS TEZİ

Berkant BAŞA

Enstitü Anabilim Dalı : BİLGİSAYAR VE BİLİŞİM MÜH.

Bu tez 13 / 01 /2014 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Yrd.Doç.Dr.	Doç.Dr.	Yrd.Doç.Dr.
Ali GÜLBAĞ	Murat ÇAKIROĞLU	Murat İSKEFİYELİ
Jüri Başkanı	Üye	Üye

ÖNSÖZ

FPGA ile Dijital Osiloskobun Gerçeklemesi Yüksek Lisans tez konusu olarak kabul edildiğinde ilk başlarda yabancılık çektim. Çünkü FPGA ile ilk defa çalışacağım için biraz heyecanlıydım. Konum belirlendikten sonra FPGA ve VHDL hakkında iyi bir çalışma yapmam gerektiğine inandım. Bu bağlamda konum ile alakalı çalışmalarım amacına ulaştı. Yorucu fakat zevkli geçen bu çalışmanın hazırlanması ve başarılı bir şekilde sona ulaşılmasında, değerli fikir ve yardımlarıyla bana destek olan değerli danışman hocam Sn. Yrd. Doç. Dr. Murat İSKEFİYELİ'ye teşekkürü bir borç bilirim.

Bu süreçte bana sabırla manevi anlamda destek olan aileme de teşekkürlerimi sunarım. Umarım bu çalışma, gelecekte FPGA ile ilgili üretilecek olan yeni proje veya çalışmalarda, temel anlamda bir kaynak işlevini görür. Bu anlamda, ortaya işe yarar bir çalışma sunmak için elimden gelenin en iyisini yapmaya özen gösterdiğime inanmaktayım.

İÇİNDEKİLER

ÖNSÖZ……..…….. i
İÇİNDEKİLER……..… ii
KISALTMALAR……...… v
ŞEKİL LİSTESİ……. viii
TABLO LİSTESİ……..... x
ÖZET…….. xi
SUMMARY……... xii

BÖLÜM.1.
GİRİŞ……. 1
1.1. Araştırmanın Amacı…… 1
1.2. Osiloskop ve İşlevi……...……... 1
1.3. Osiloskobun Kullanımı ve Yapısı…..…...…….. 2
1.4. Osiloskop Cıhazının Temel Çalışma Prensibi……..……... 3
1.4.1. Osiloskop ile yapılan ölçümler……..……... 4
1.4.2. Kömütatör anahtarların görevi……..……... 5
1.5. Analog Osiloskop Nedir?..... 7
1.6. Dijital Osiloskop Nedir?..... 9
1.7. Ekran ve Ekran Çeşitleri…….. 10
1.8. VGA (Video Graphic Adapter/Ekran Kartı Arayüzü)……. 12
1.9. Görselleştirme…….. 13

BÖLÜM.2.
FPGA ve VHDL……. 15
2.1. FPGA Tarihçesi……..……. 15
2.2. FPGA Nedir……..……... 18
2.3. FPGA Yapısı ve Pinleri……..……. 21

2.4. FPGA ile Tasarım……..…….. 24
2.5. VHDL Nedir……..……. 29
2.6. VHDL Yapısı……... 29
2.7. VHDL Veri Tipleri…….……...……..…… 31
2.7.1. Nesne tipleri (Object Types)…...…….…… 32
2.7.2. İşaretler (Signal)……..……...…….. 34
2.7.3. Değişkenler (Variable)……...…….. 35
2.7.4. Sabitler (Constant)……...…… 37
2.7.5. Tam sayılar (Integer)……... 38
2.7.6. Gerçek sayılar (Real)……..…..…… 39
2.7.7. Sıralı sayılar (Enumerated)……..……... 40
2.8. VHDL'in Avantajları……...….... 41
2.8.1. Tasarım……....…… 42
2.8.2. Benzetim……. 42
2.9. VHDL ile FPGA Arasındaki İlişki……...…….. 43

BÖLÜM.3.
TERASIC DE0 GELİŞTİRME KARTI.……..…… 44
3.1. DE0 Geliştirme Kartı FPGA Platformu..……..…..…… 44
3.2. DE0 Kurulu Blok Şeması…..……...……..... 45
3.3. Bellek Bileşenleri…….…..…...…….. 49
3.4. Quartus II Programı……...……..…...……. 50

BÖLÜM.4.
SİSTEMİN GERÇEKLEŞTİRİLMESİ.……..…… 53
4.1. Genel Şema…….……..……... 53
4.2. Akış Diyagramı…..……...……... 54
4.3. Elektronik Devre……..…..…...……..…… 55
4.4. Sinyal Okuma…….……..……... 56
4.5. Görüntünün Elde Edilmesi……..……..…….. 56
4.6. Fonksiyon Jeneratör Ayarları...…...……... 58

BÖLÜM.5.
SONUÇLAR VE ÖNERİLER…….. 62

KAYNAKLAR…….. 63
ÖZGEÇMİŞ…… 65

KISALTMALAR LİSTESİ

FPGA	: Alanda Programlanabilir Kapı dizileri (Field Programmable Gate Array)
VHDL	: Yüksek Hızlı Donanım Tanımlama Dili (Verilog Hardware Description Language)
HDL	: Donanım Tanımlama Dili (Hardware Description Language)
s.	: Sayfa
vb.	: Ve Benzeri
MUX	: Multiplexer
PLD	: Programlanabilir Mantık Dizisi (Programmable Logic Devices)
PROM	: Programlanabilen Salt Okunur Bellek (Programmable Read-Only Memory)
PLA	: Programlanabilen Mantık Dizisi (Programmable Logic Array)
PAL	: Faz Değiştiren Çizgi (Phase Alternating Line)
CPLD	: Programlanabilir Kompleks Mantık Dizisi (Complex Programmable Logic Device)
SPLD	: Programlanabilir Basit Mantık Dizisi (Simple Programmable Logic Device)
ASIC	: Uygulamaya Özel Tümleşik Devre (Application Specific Integrated Circuit)
CLB	: Yapılandırılabiliri Mantık Bloğu (Configurable Logic Block)
IOBs	: Giriş Çıkış Bloğu (Input Output Block)
I/O	: Giriş Çıkış (Input Output)
LUT	: Look Up Table
RAM	: Rastgele Erişimli Hafıza (Random Access Memory)
VCC	: Voltage Common Collector
PAR	: Place and Route

VHSIC	: Yüksek Hızlı Tümleşik Devreler İçin Donanım Tanımlama Dili (Very High Speed Integrated Circuits)
LabVIEW	: Laboratory Virtual Instrumentation Engineering Workbench
CompacRIO	: combination real-time controller, reconfigurable Modules
IEEE	: Institute of Electrical and Electronics Engineers
LRM	: Language Reference Manual
AC	: Analog Çevirici (Analog Convertor)
DC	: Dijital Çevirici (Digital Convertor)
ADC	: Analog Dijital Çevirici (Analog to Digital Converter)
CRT	: Katot Işınlı Tüp (Cathode Ray Tube)
TAP	: Test Erişim Protokolü (Test Access Port)
PCI	: Peripheral Component Interconnect
TCK	: Test Sinyali (Test Clock)
TDI	: Test Veri Girişi (Test Data Input)
TDO	: Test Veri Çıkışı (Test Data Output)
TMS	: Test Mode Select
TRST	: Test Yeniden Başlat (Test Reset)
CGA	: Renk Grafik Adaptörü (Color Graphics Adapter)
EGA	: Geliştirilmiş Grafik Adaptörü (Enhanced Graphics Adapter)
VGA	: Video Grafik Dizisi (Video Graphics Array)
SVGA	: Süper Video Grafik Dizisi (Super Video Graphics Array)
W/B	: Siyah Beyaz (White/Black)
Hz	: Hertz
msn	: Milisaniye
Msn	: Mikrosaniye
ATI	: Array Technologies Incorporated
PCI-E	: PCI Express
SLI	: Ölçeklenebilir Bağ Arayüzü (Scalable Link Interface)
AGP	: Hızlandırılmış Grafik Portu (Accelerated Graphics Port)
DVI	: Sayısal Görüntü Arayüzü (Digital Visual Interface)

ŞEKİLLER LİSTESİ

Şekil 1.1. Osiloskop……... 3
Şekil 1.2. Osiloskop ekranı……... 4
Şekil 1.3. Katot ışınlı tüp (CRT)…….. 5
Şekil 1.4. Osiloskop'un içyapısı…… 6
Şekil 1.5. Basit bir analog osiloskobun blok diyagramı…… 6
Şekil 1.6. İkisinde de analog devreler aynıdır: analog ve dijital bellekli osiloskopların blok şemaları…….……..……..……. 8
Şekil 1.7. Dijital osiloskop şemaları…… 10
Şekil 1.8. Harflerin ekranda görünümü……...… 11
Şekil 1.9. Ekran……. 11
Şekil 2.1. Farklı teknolojilerin ortaya çıkışları……... 15
Şekil 2.2. İlk PLD'ler basit PROM'lardı……... 16
Şekil 2.3. PLA yapısı…… 17
Şekil 2.4. PAL yapısı…… 17
Şekil 2.5. Yaygın CPLD yapısı…… 18
Şekil 2.6. FPGA iç bileşimi-CLB-IOBS……... 20
Şekil 2.7. Dijital osiloskop blok diyagramı……. 21
Şekil 2.8. Mantık hücresi (Logic- cell)…… 21
Şekil 2.9. CLB yapısı ve CLB giriş çıkışlarının FPGA üzerindeki görünümü……..…….. 22
Şekil 2.10. Programlanabilir ara bağlantılar…… 23
Şekil 2.11. FPGA tasarım adımları……. 25
Şekil 2.12. Nasıl programlanır? SRAM kontrollü programlanabilir anahtar…… 28
Şekil 2.13. Nasıl çalışıyor? Xilinx Xc4000 configurable logic block (Clb)….. 28
Şekil 2.14. Bir yapı tanımlama örneği…… 30
Şekil 2.15. VHDL veri tipleri diyagramı…… 34

Şekil 3.1. DE0 KİT…… 44
Şekil 3.2. DE0 blok diyagramı……. 46
Şekil 3.3. DE0 temel bilgisayar blok diyagramı…….. 49
Şekil 3.4. Quartus II ekran görüntüsü…….. 51
Şekil 3.5. Quartus II dosya türleri……... 52
Şekil 4.1. Genel şema…… 53
Şekil 4.2. Akış dıyagramı……. 54
Şekil 4.3. Elektronik devre…… 55
Şekil 4.4. FPGA DE0 sinyal okuma…… 56
Şekil 4.5. Yatay yenileme döngüsü…….. 57
Şekil 4.6. Düşey yenileme döngüsü…….. 58
Şekil 4.7. Sinüzoidal dalga…… 58
Şekil 4.8. Kaydırıcı sinüzoidal dalga……. 59
Şekil 4.9. Evirilmeyen gerilim toplayıcı…… 59
Şekil 4.10. ADC0804…… 60
Şekil 4.11. VGA 640x480 sinüzoidal dalga çıktısı……... 61

TABLO LİSTESİ

Tablo 2.1. Bir yarım toplayıcının VHDL kullanılarak tanımlanması. 42
Tablo 3.1. Quartus sürüm kıyaslama tablosu. 50

ÖZET

Anahtar kelimeler: FPGA, VHDL, Altera DE0, Osiloskop

Çalışmada eğitim kurumlarındaki laboratuarlarımızda osiloskop ihtiyacını daha ekonomik olarak karşılamak amacıyla FPGA (Alan Etkili Kapı Dizileri) yapıları ile sayısal osiloskobun gerçeklenmesi hedeflenmiştir. Maliyetleri düşük olan eğitim amaçlı FPGA kitleri ile profesyonel olmayan osiloskoplar elde edilebileceği gösterilmiştir. FPGA'ların hızlı olması yüksek frekanslı sinyallerin ölçümünde de olanak tanımaktadır. Çalışmada kit olarak Terasic DE0 kiti kullanılırken, kitin programlanmasında Quartus II yazılımı kullanılmıştır. Ölçülecek sinyal olarak bilinen bir sinyal olan sinüs dalgası fonksiyon jeneratöründen elde edilmiştir. FPGA'da örneklenen sinyalin grafiksel görünümü ise 640x480 çözünürlüğe sahip bir VGA ekranda gösterilmiştir.

REALIZATION OF DIGITAL OSCILLOSCOPE WITH FPGA

SUMMARY

Key Words: FPGA, VHDL, Altera DE0, Oscilloscope

In this study, it is aimed to implement a digital oscilloscope by FPGA architectures to correspond the requirements of laborataries in the educational institutions in an economic way. It is shown that non-professional oscilloscopes can be implemented with low-cost, educational purposed FPGAs. It is also able to measure high frequency signal by high speed FPGAs. Terasic DE0 kit is used as a hardware and Quartus II is used as a software in this application. The sine signal which is a widely known signal used to measure is generated by the signal generator. The graphical view of the sampled signal in the FPGA is shown on a 640x480 pixeled VGA monitor.

BÖLÜM 1. GİRİŞ

1.1. Araştırmanın Amacı

Bu çalışmanın amacı FPGA ile dijital osiloskobun gerçeklemesi olacaktır. Dijital osiloskobu gerçekleştirirken Terasic Altera FPGA DE0 kiti kullanılacak ve VHDL Quartus II editöründe yazılacak olan kodu FPGA'ya yüklenerek VGA monitörde çıktısı görülecektir. Günümüzde teknolojinin hızla gelişmesine paralel olarak kullanılan elektronik devreler de daha karmaşık hale gelmektedir. Geleneksel olarak kullanılan yöntemler ile bu devreleri tasarlamak oldukça zor hale gelmektedir. Bu ihtiyacı karşılamak için yeni teknolojiler geliştirilmektedir. FPGA' lar da bu teknolojilerden biridir. FPGA' lar programlanabilir sayısal bloklar ve bağlantılarını içeren yarı iletken cihazlardır. Programlanabilen bu sayısal kapılar sayesinde karmaşık tasarımlar kolay bir şekilde geliştirilmektedir. FPGA'ları programlamak için Xilinx firması tarafından geliştirilen ISEWebPack programı kullanılır. Bu program kullanıcıya üç ayrı programlama seçeneği sunmaktadır. VHDL programlama dili FPGA' ları programlama da kullanılan program seçeneklerinden sadece bir tanesidir. VHDL bir donanım tanımlama dilidir. Diğer bir programlama seçeneği şematik programlamadır. Bu programlama seçeneği bize sayısal kapılar, flip-filoplar, çoğullayıcılar (multiplexer) vb. elemanları kullanarak devre tasarımlarımızı yapmamıza olanak sağlar. Diğer bir programlama seçeneği akış diyagramı şeklinde programlamadır.

1.2. Osiloskop ve İşlevi

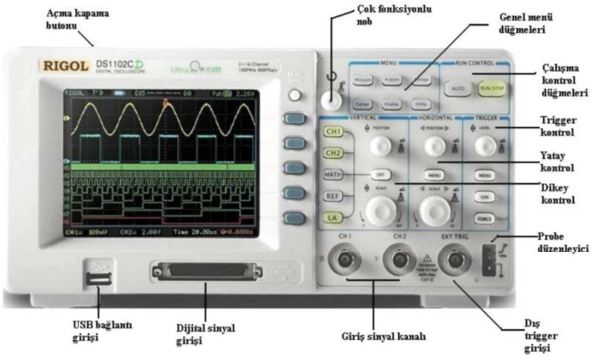
Elektriksel büyüklükleri ölçen aletleri, ölçtükleri büyüklükleri sayısal veya analog olarak ifade ederler. Osiloskoplar ölçtüğü büyüklüğün dalga şeklini göstererek maksimum değerini ölçer. Örneğin bir voltmetre ile ölçülen 12V AC gerilim osiloskop ile ölçüldüğünde yaklaşık 16,97 V gibi bir değer okunur. Bu değerlerin

farklı olmasının sebebi ölçü aletlerinin AC'de etkin değeri, osiloskobun ise AC'nin maksimum değerini ölçmesidir. Osilaskoplar, diğer ölçü aletlerine göre daha pahalı olmalarına karşılık bir sistemdeki arızanın tespiti osiloskoplar ile daha kolaydır. Çünkü televizyon veya daha karmaşık sistemlerin belirli nokta ve katlardaki çıkışları sabittir ve bu çıkışlar sisteme ait kataloglarda nokta nokta belirtilir. Osiloskop ile yapılan ölçümlerde katalogdan farklı çıkış veren katta arıza var demektir. Osiloskopların dijital ve analog çeşitleri mevcuttur.

Standart olarak iki kanallı olan bu cihazların daha fazla kanala sahip olan modelleri de bulunmaktadır. Örneğin 3 kanallı, 8 ışınlı, 200 Mhz'lik bir osiloskop ile 3 kanaldan sinyal girilip, bu sinyaller ve tabi tutulduğu işlemler sonucunda oluşan 8 değer aynı anda görüntülenebilir ve 200 Mhz kadar olan sinyaller ölçebilir. Son üretilen dijital osiloskoplar ile ölçülen büyüklük renkli olarak izlenebilmekte; ölçülen değer hafızaya alınıp, bilgisayara aktarılabilmektedir. Elektrik ve elektronik devrelerinde akım ve gerilimin değeri, frekans ve faz farkı ölçümlerini dijital veya analog ekranda grafiksel olarak gösteren aletlerdir. [1].

1.3. Osiloskobun Kullanımı ve Yapısı

Osiloskopta görüntü katot ışınlı tüp (CRT) aracılığıyla oluşturulur. Osiloskop çalıştırıldığında katot ışınlı tüpün arkasındaki filâman ısınıp elektron yaymaya başlar. Bu elektronlar, kontrol girişinden geçerler ve odaklama anodu ve hızlandırıcı anot tarafından çekilen elektronlar, elektron demeti haline getirilir. Oluşturulan elektron demeti dikey saptırma plakaları elektron demetini ekran üzerinde aşağı yukarı hareket ettirirken yatay saptırma plakaları elektronları ekran üzerinde sağa sola hareket ettirir. Elektron demeti yatay ve dikey saptırma plakalarını geçerek florasan kaplı ekranın arka yüzeyine ulaşır. Bu yüzeyde elektron demetlerinin temas ettiği yerler parlar ve sinyalleri gösteren görüntüsü bu noktaların birleşimiyle oluşturulur.



Şekil 1.1. Osiloskop

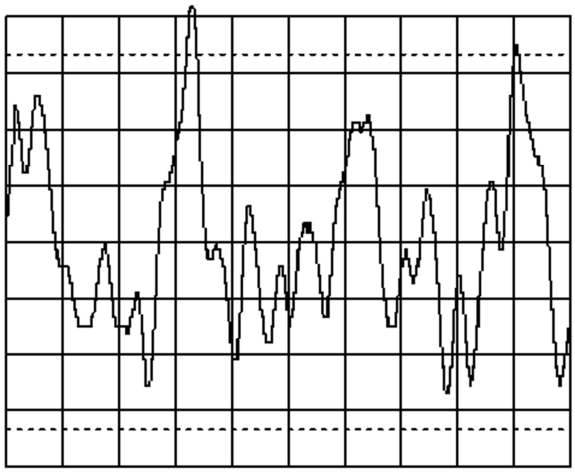
Elektriksel sinyallerin ölçülüp değerlendirilmesinde kullanılan aletler içinde en geniş ölçüm olanaklarına sahip olan osiloskop, sinyalin dalga şeklinin, frekansının ve genliğinin aynı anda belirlenebilmesini sağlar. Çalışması, hareket halindeki elektronların yörüngelerinin bir elektrik alan içerisinden geçerken sapmaları prensibine dayanır.

Katot ışın tüpündeki saptırma plakaları adı verilen düzlemsel levhalara uygun potansiyellerde gerilimler uygulanarak oluşturulan elektrik alanlar, plakalar arasından geçen elektronları (elektron demetini) saptırarak fosfor ekrana çarptığı noktanın yerini değiştirir.

Bu noktanın konumu saptırma plakalarına uygulanan gerilimin ani değeri ve dalga şekline bağlı olarak değişir ve ekranda ışıklı bir çizgi oluşur. Osiloskop devreye daima paralel bağlanır. [20].

1.4. Osiloskop Cıhazının Temel Çalışma Prensibi

Osiloskop cihazı bir elektriksel işaretin zamana göre değişimini ya da başka bir işarete göre değişimini bir ekranda gösteren cihazdır. Aşağıdaki örnek osiloskop ekranın çıktısı nasıl görüntü vermektedir gösterilmiştir.



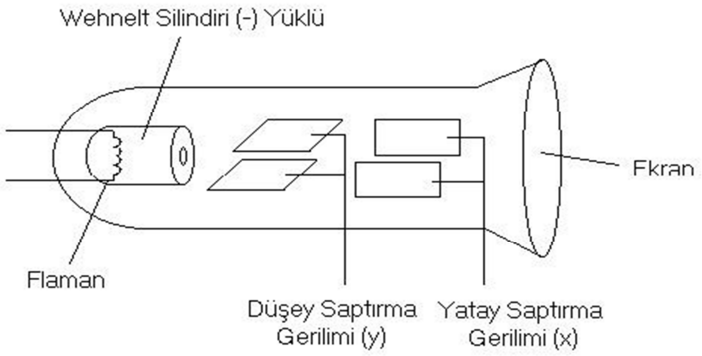
Şekil 1.2. Osiloskop ekranı

Ekranın X ve Y eksenleri olduğunu varsayarsak yatay eksen(X ekseni) zamanı, düşey eksen(Y ekseni) ise genliği göstermektedir. Şekil 1.2.'de görüldüğü gibi yatay eksen zaman ekseni ve düşey ekseninde genlik ekseni olup işaretin zamana göre değişimi rahatlıkla izlenebilmektedir.

Burada yatay ekseni de bir işaret kaynağı gibi düşünebiliriz. Şöyle ki, düşey eksen işaretin genliğiyle doğrusal olarak değişmektedir, yani bir nevi tarama yapmaktadır. Yatay eksende ise zamana bağlı doğrusal bir değişim olması gerekir. Bunu sağlayan ise yatay ekseni tarayan testere dişi dalgadır. [11].

1.4.1. Osiloskop ile yapılan ölçümler

1. AC sinyalin genlik (gerilim) ve frekansı ölçülebilir.
2. DC sinyalin gerilimi ölçülebilir.
3. Osiloskop girişine uygulanan sinyalin ani değerleri gözlenebilir.
4. Çok kanallı osiloskoplarda, birden fazla sinyalin karşılaştırılması yapılabilir ve faz farkları ölçülebilir.
5. Dolaylı olarak akım ölçümü yapılabilir.

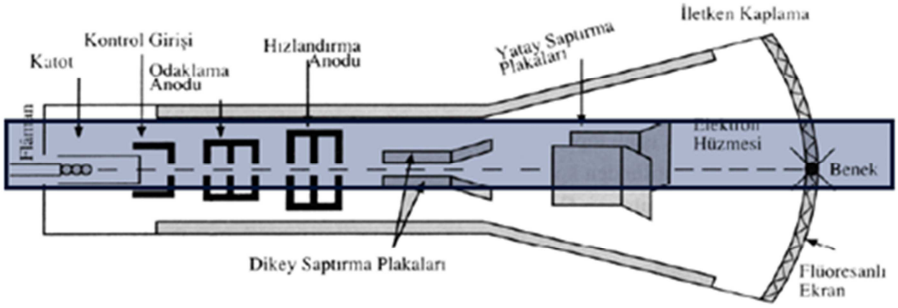


Şekil 1.3. Katot ışınlı tüp (CRT)

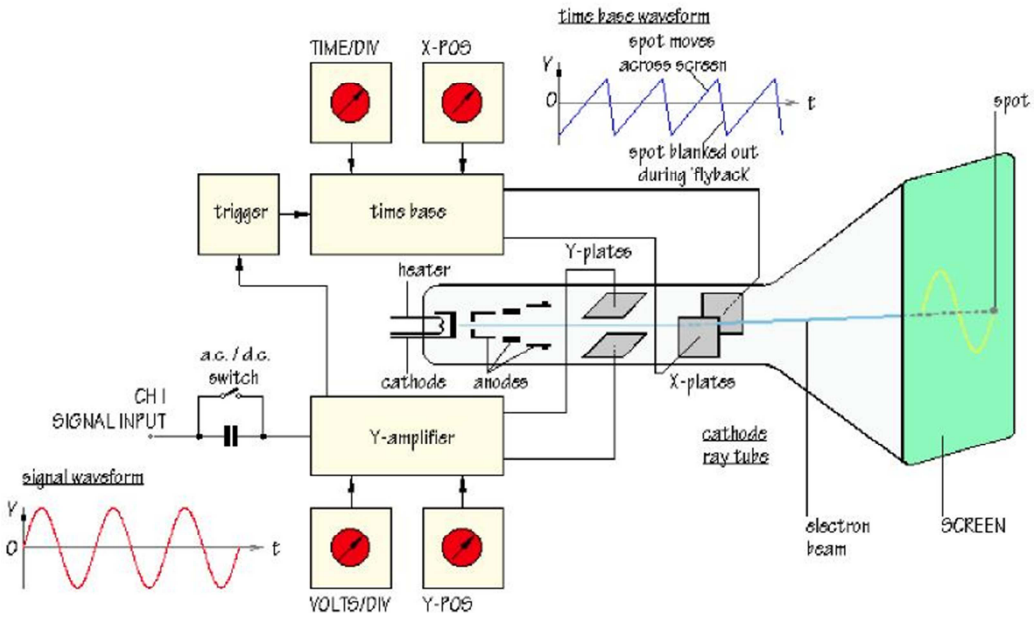
1.4.2. Komütatör anahtarların görevleri

1. Power: (On-Off) Anahtarı Osiloskobu açmak ve kapamak için kullanılır.
2. Intensity: Ekranda oluşan görüntünün (çizginin) parlaklığını ayarlamak için kullanılır.
3. Focus: Ekranda oluşan çizginin netliğini ayarlamak için kullanılır.
4. X-Position: Işıklı çizgiyi sağa kaydırmak için kullanılır.
5. Y-Position: Işıklı çizgiyi yukarı aşağı kaydırmak için kullanılır.
6. AC: Alternatif akım sinyallerini ölçmek için kullanılır.
7. DC: Doğru akım sinyallerini ölçmek için kullanılır.

Kömütator anahtar bir kasa içine konmuş iki anahtardan oluşur. Bir giriş ve iki tane çıkışı vardır Hemen hemen her yerde kullanılır. Örneğin beş lambalı bir avizeyi ekonomik kullanmak için uygun bağlantı yapıldıktan sonra birinci anahtarı açınca iki lambayı, ikinci lambayı açınca diğer üç lambayı, dolayısı ile iki lambayı açınca diğer beş lambayı kontrol edebilirsiniz. Bu anahtarlar 2 farklı lamba ya da elektrikle çalışan 2 farklı cihazları komuta etmek için kullanılır. [11].



Şekil 1.4. Osiloskobun içyapısı



Şekil 1.5. Basit bir analog osiloskobun blok diyagramı

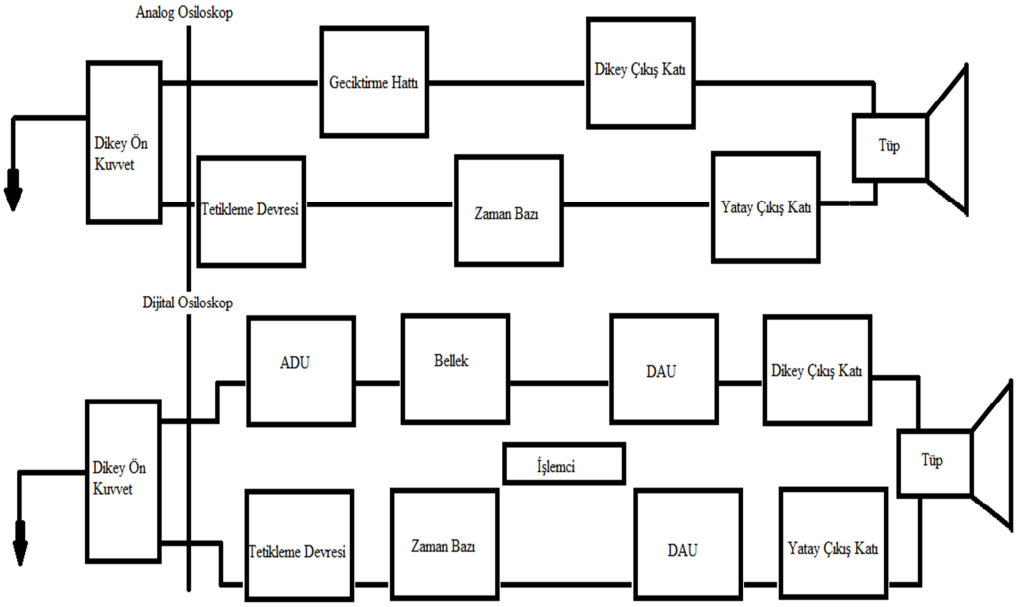
Yukarıdaki şekil incelendiğinde giriş kısmında AC ya da DC kuplajı seçmeyi sağlayan bir anahtar olduğu görülür. Girişe uygulanan işaretin DC bileşeni engellenmek isteniyorsa bu bileşeni yalıtmak için bir kuplaj kondansatörü kullanılır. Daha sonra bu işaret uygun oranlarda yükseltilir ya da zayıflatılır ve ekranda Y ekseni olarak görülür. Yine yukarıdaki şekil incelendiğinde X eksenini oluşturan işaretin bir testere dişi dalga olduğu görülür. Bu işaretin bir periyodunda ekran bir kez taranır ve sonra ekranın soluna dönülür. Ancak burada dikkat edilmesi gereken en önemli noktalardan biri gözlemlenecek işaretin periyodik olması gerekliliğidir.

Eğer gözlemlenecek işaret periyodik olmazsa her defasında ekranda görünen işaret hızlı bir biçimde değişecek ve bu değişimi insan gözü algılayamayacağı için görüntüyü kayıyormuş gibi görecektir. Osiloskobun en önemli kısımlarından biri ise tetikleme (trigger) bölümüdür. Tetikleme gözlemlenecek sinyalin her defasında aynı yerden yakalanmasını ve ekrana basılmasını sağlar. Eğer tetikleme düzeneği olmazsa işaret her defasında rastgele bölümlerden yakalanacağı için ekrandaki görüntü yine düzgün olarak gözlemlenemez. Tetikleme bölümünün yaptığı işi yükselen kenar için örnek verecek olursak, tetikleme; işaretin genliğinin tetikleme seviyesinin altında olduğu durumun, genliğin tetikleme seviyesinin üstüne çıkıp bu durumu bozmasıyla beraber işaretin ekrana basılmasıdır. Böylece işaret her defasında tetikleme seviyesinin üstüne çıkar çıkmaz ekrana basılabilecektir. [11].

1.5. Analog Osiloskop Nedir?

Analogla dijital osiloskop arasındaki en önemli farklar Şekil 1,6'daki blok şemalarda görülmektedir. Modern analog osiloskoplar mikroişlemci kontrollüdür ve kapsamlı bir iç işaret işleme sistemine sahiptir. Analog osiloskopların olumlu yanı, elektron ışığının dönüş süresi ve bununla birlikte tetiklenmeye hazır duruma gelme süresinin kısa olmasından kaynaklanan yüksek yenileme hızıdır. Böylece işaret gösteriminin yenilenmesi bir DSO'da (bellekli dijital osilloskop) olduğundan birkaç 1000 kat daha hızlıdır. Bu nedenle bir analog osiloskop sürekli ve düzensiz bileşenli işaretlerin saptanması için fevkalade bir ölçme aracıdır. Ne var ki normal bir analog osiloskopta işaret saklama olanağı yoktur ve yüksek saptırma hızlarında ya da çabuk olup biten bir defalık olaylarda elektron ışığı çok az ya da hiç görünmez.

Bu sorun, görsel yazma hızı alışılagelmiş resim tüplerininkinden yüksek olan mikro kanal plaklı bir resim tüpüyle çözülür. Mikro kanal plağı (Micro Channel Plate, MCP), paralel dizilmiş birkaç yüz bin sekonder elektron çoğaltıcısı içermektedir. Çoğaltıcı kanallar, iç duvarları elektriksel olarak iletecek şekilde buharlanmış cam borucuklardan ibarettir. [18].



Şekil 1.6. İkisinde de analog devreler aynıdır: analog ve dijital osiloskobların blok şemaları

Dolayısıyla tümleştirilmiş bir diyot ve direnç zinciri fonksiyonu meydana gelir. Cam borucuklar petek biçiminde bir plaka oluşacak şekilde birbirlerine yapıştırıldıktan sonra ön yüzeyleri dümdüz yapılır ve çoğaltma gerilimini uygulayabilmek için kontaklanır. Bu gerilimin yüksekliğine bağlı olarak mikro kanal plakasının kazanç faktörü 104'ü geçebilir. Elektronların çoğaltılmasıyla ışığın aydınlığı büyük ölçüde artırılır, bu ise özellikle çok çabuk olup biten bir defalık işaretlerin saptanmasında büyük avantajlar sağlamaktadır. Bir başka avantaj ise özel mikrokanalların, elektron ışığının yüksek yeğinlikte olduğu bölgelerde doyuma girmesi, düşük yeğinlikte olduğu bölgelerde de tam kazançla çalışmasıdır. Bu adaptif ışık aydınlığı denen normalizasyon etkisi istenen bir şeydir, çünkü bunun sayesinde tüm "yeğinlik dinamik bölgesinde" memnun edici bir aydınlık seviyesi elde edilir.

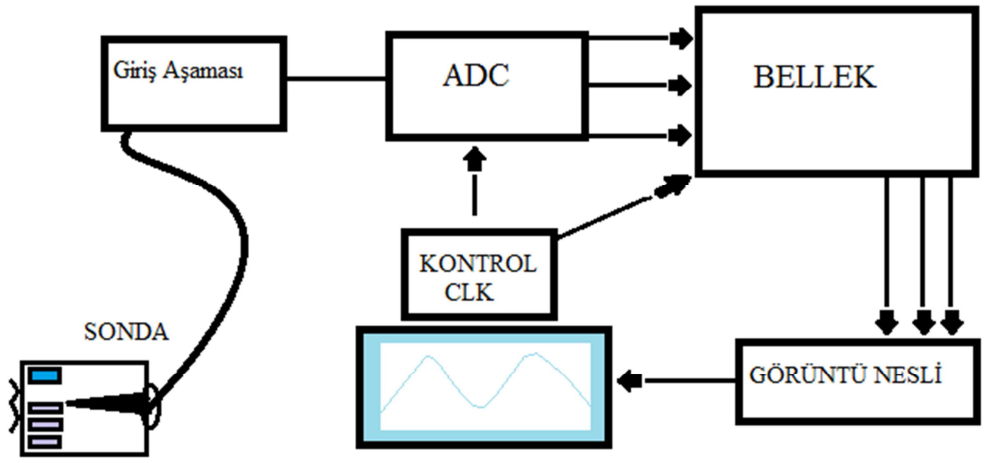
Analog osiloskobun yüksek yenileme hızının MCP resim tüpünün yüksek görsel yazma hızıyla birleştirilmesi sonucu, bazı durumlarda 106 normal işlemde sadece bir kez meydana gelebilen düzensiz olayların saptanması için oldukça verimli bir alet kombinezonu elde edilir. MCP osiloskobuna dijitalise eden bir kamera sistemi eklenirse örnekleme hızı 100 Gigasamples/s ve dikey ayırıcılığı 9 Bit olan bir "transient" dijitalisör meydana gelir (Şekil 1.6).Yarıiletken A/Çeviricileriyle bu kadar yüksek örnekleme hızları önümüzdeki yıllarda gerçekleştirilemeyecektir (internet kay).

1.6. Dijital Osiloskop Nedir?

Klasik analog ya da benzer oskiloskoplara kıyasen dijital oskiloskoplar kuvvetli bir şekilde ve pratik anlamda artık analog oskiloskopların kullanımda olmadıklarını vurgulamaktadır. Bunun en bariz veya doğal sebebi Dijital elektronik cihazların olağandışı evrim veya gelişimlerinden kaynaklanmaktadır. Dijital çip üreticileri her seferinde, ürettiklerini daha küçük boyutlarda ve daha çok sayıda tranzistörle birlikte, en düşük maliyetle, bir öncekinden daha hızlı ve sürekli çalışabilen cihazlar olarak yapmaya çalışmaktadırlar. Şekil 1,7'de dijital oskiloskobun en basit fonksiyonlu örneğini görebiliriz: İçinde her çeşit donanım araçlarından iyileştirme aşamalarını barındırmaktadır. Dolayısıyla verdiğimiz örnekte, ADC (Analog to Digital Converter) uyarı sinyallerini bildirme ve bunların değer ölçümlerinden sorumludur. Bu bölüm oskilosposun en kritik kısmıdır. Çünkü örnekleme oranı ve değiştiricinin hassas düzeyine (yani miktar ölçer kısım) bağlı olarak bu kısım oskiloskobun en temel karakteristik özelliğini tanımlamaktadır. Örneklenmiş değerler daha sonraki süreç ve görüntüleme işlemleri için bir kereliğine dijital hafızada depolanarak kayıt altına girer. Bu anlamda Dijital Oskiloskop, dijital hafızaya teşekkür eder. Çünkü bu kayıt ile dijital oskiloskop, dijital hafızada toplanmış olan bilgilerin karmaşık hesaplamalarını fark edebilme özelliğini elde ederek işlemlerin kolaylaşmasını sağlar. Bu özelliği ile dijital oskiloskop, analog oskiloskoplardan daha avantajlıdır.

Başlıca Verimliliği: Tasarımı basit ve çabuk olan, tasarımındaki farklı üretim parçalarından oluşabilme özelliği ve karakteristik yapısıyla tam olarak aranılmayan çip türlerinden biridir.

Boyutunu küçültme: İçinde gerekli olan bütün çipleri barındıran Anakart, daha fazla sınırlandırılabilecektir. [12].



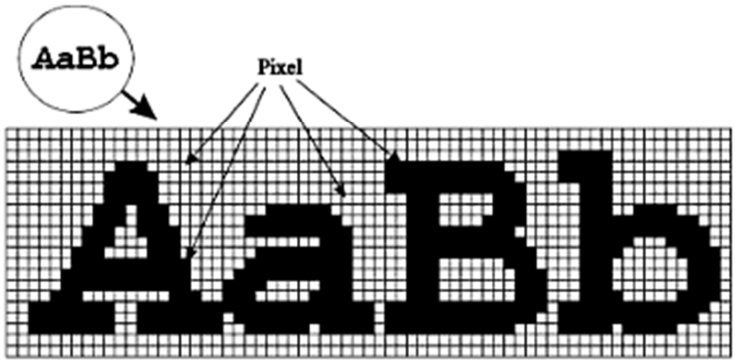
Şekil 1.7. Dijital osiloskop şemaları

1.7. Ekran ve Ekran Çeşitleri

Bilgisayar çıkış birimidir. Monitör olarak da bilinir. Bilgisayara verdiğimiz bilgileri, isteğe göre ekrandan yansıtır. Boyutu çoğunlukla 12" ya da 14"tir. 24 satır, 40-80 sütundan oluşur. 25, satır kullanımı isteğe bağlıdır. Ekranda görebileceğimiz en küçük noktaya PİKSEL denir. Pikseller çoğaldıkça daha net görüntü elde edilir. Bilgisayarda gördüğümüz harf, sembol, çizim vb. her şey noktalardan oluşur.

RENKLİ (Color) ya da RENKSİZ (Monochrome) olabilirler. Çeşitli grafik özellikleri ile grafik çizebilme özelliğine sahiptirler.

Grafik özelliği bilgisayarın grafik kartını da ilgilendirir. Grafik kartları HERCULES, CGA, EGA, VGA, SVGA vb. kartlardır. Grafik kartları, monitör ile uyumlu olmalıdır.



Şekil 1.8. Harflerin ekranda görünümü

Renkli ya da Renksiz renk anlamındadır. c (White/Black) Siyah/Beyaz ekranı belirtir. Ayrıca 50 hz 60 hz gibi, frekansları da belirtilir. Ekranların sınıflandırılması üç özelliğe göre yapılır: Ekranın saniyede kaç kez görüntülendiğine yenilenme oranı ya da yatay tarama frekansı denir. Bu hertz (Hz) ile ölçülür. Örneğin saniyede 70 Hz, 70 ekran yenilenme oranı demektir. Bir monitörün temiz görüntülü olması ve CRT üzerinde saniyede en az 60 tam ekran görüntü boyayarak titremeden çalışması gerekmektedir. Bu durum gözlerimizi de korur. Devre kartı alırken de aynı özellikte (Hz) olmasına dikkat edilmelidir.

Şekil 1.9. Ekran (Monitör)

1. Boyut: 5 ila 25 inç arasında değişir en çok kullanılanlar, 12” ve 14” olanlarıdır. 25 satır, 80 sütundan oluşur.
2. Renk: Tek renkli olan monitörler siyah-beyaz, yeşil ve amberdir. 2 renkten 16 milyon renk tona varan monitörler bulunmaktadır.
3. Çözünürlük: Monitörün görüntüsü ile kalitesi ölçülür. Ne kadar net görüntü sağlanabiliyorsa o kadar iyidir.

4. Netliğin iyi olması da piksel sayısına yani adreslenebilir nokta sayısına bağlıdır. Bu sayı 65,000'den 16.000.000'a kadar değişebilir Maksimum Yatay Nokta Sayısı X Maksimum Dikey Nokta Sayısı formülü ile hesaplanır.

Daha önce de belirttiğimiz gibi piksel ekranda en küçük nokta idi. Bu noktalar ne kadar çok olursa, görüntü o kadar net olur. (720 X 360), (640 X 350), (800 X 600), (1024 X 728) gibi…

CGA (Color Graphics Adapter): 320x200 640x200 çözünürlüktedir. 4 renk gösterir.
EGA (Enhanced Graphics Adapter): 640x350 720x350 çözünürlüktedir. 16 renk gösterir.

VGA (Video Graphics Array): 720x400 çözünürlüktedir. 16 renk ve 320x200 çözünürlükle 256 renk gösterebilir. Süper VGA kartlarla 1024x768 ve 1280x1024 çözünürlükte görüntü elde edilir. VGA kartının kendi belleği vardır. Bellek artıkça renk sayısı artar. [16].

1.8. VGA (Video Graphic Adapter/Ekran kartı arayüzü)

VGA/Ekran kartlarını inceleyelim: Ekran kartı bilgisayarın zorunlu donanımlarından biridir. Temelde ekranda görüntünün oluşması için gerekir. Basit bir işlem gibi görünse de, özellikle uç boyutlu oyunlar gibi yüksek grafik işlem gücü gerektiren uygulamalarda büyük önem kazanmaktadır. Ekran kartları yapı olarak kendi içinde bir bilgisayar olarak düşünülebilir.

Grafik uygulamaları için özelleşmiş işlemcisi, bu işlemcinin işleyeceği verilerin depolandığı RAM bölümünün olması yapı olarak bilgisayara benzemesini sağlamaktadır Ekran kartlarındaki işlemcilere 'Grafik işlem birimi(Ünitesi)' adı verilir. Yüksek performanslı kartlarda ATI ve NVIDIA'nın çekişmesi görülmektedir. [17].

Ana kartın VGA ara yüzü hangi türde ekran kartı takabileceğinizi göstermektedir. Bu özellik ana karta entegre bir ekran kartı olmaması durumunda daha çok önem kazanmaktadır.

Yeni ana kartlarda PCI-E (PCI Express) olarak gelen bu tür, daha öncekilerde AGP olarak bulunmaktaydı. PCI-E x16 olarak belirtilen ara yüz piyasadaki en yeni ekran kartlarını çalıştırabileceğini göstermektedir. Ana kart özelliklerinden birisi olan SLI, ana karta ayni anda iki tane takarak daha yüksek bir grafik performansı elde edebileceğinizi göstermektedir. Normal bir kullanıcı için yüksek fiyatlı ekran kartları almaya gerek yoktur. Bazı ana kartlara entegre olarak gelen (VGA onboard olarak belirtilir) ekran kartları normal bir kullanıcının tüm ihtiyaçlarını karşılayacaktır. Sadece çok yeni 3 boyutlu oyunlar daha yüksek grafik performansına ihtiyaç duyduğu için gelişmiş ekran kartları gerekmektedir. Ekran kartlarının çıkışları da ekran kartının kalitesi ile ilgili bilgi vermektedir. DVI çıkışlı (genelde beyaz renkli eskisine göre daha uzun bir çıkış) bir bütünleşik ekran kartı varsa yeni bir ekran kartı takma ihtiyacınız olmayacaktır. Çünkü dijital ara yüz ile monitörünüzü bağlayıp yüksek kalitede görüntüler alabilirsiniz. Eğer bütünleşik ekran kartında sadece Analog çıkış varsa (mavi renkli standart çıkış) ayrı bir ekran kartına ihtiyaç duyabilirsiniz. [17].

1.9. Görselleştirme

En başından da söylediğimiz gibi oskiloskop içerikli projemizin temel amacı maliyetleri düşürmektir. Bu yönlendirmeyle en uygun seçenek olarak görselleştirmenin kendi başına VGA gözlemleme olarak bunu yapmayacağı yönündeydi. Çünkü:

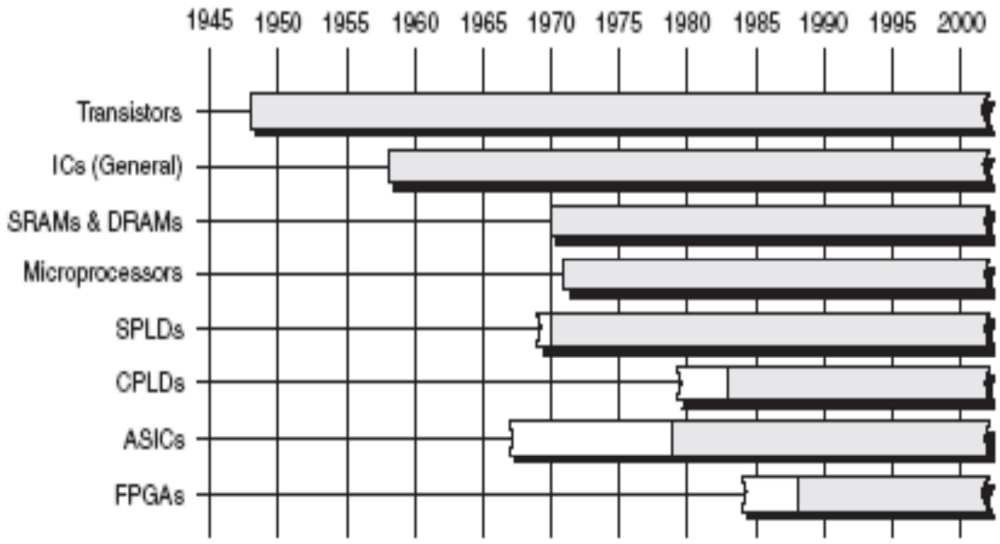
1. Daha ucuz maliyet. Eğer ekranı/elemeleri sisteme dâhil etmezsek fiyatlar şiddetli/ani bir şekilde düşüşe maruz kalır.
2. Ekranlar son derece yaygınlar VGA. Örneğin Bilgisayarlar (PC) VGA’lar ile uyumlu ekranları kullanırlar. Bizim oskiloskobumuz bu ekranların her hangi birinde görüntüleme işlemini rahatlıkla yapabilmektedir.

3. Kolay uygulama, Bu anlamda uygulama modülünün VGA sinyallerini basit bir şekilde üreten FP-GA'ın işlem gücüne teşekkür edebiliriz.
4. Eşzamanlılık, yatay ve dikey eşzamanlılık, fark edilmesi gereken her bir dikey güçlenmenin veya her bir görüntünün, ekranlarda belli sıklıkta bunu belirtilmesi yapılmaktadır.
5. RGB. 3 tane analojik/benzer sinyalleri, kırmızı bileşenlerin yoğunluğunu, Her bir Yeşil ve Mavi görüntü noktasını(pixel), sonuç itibariyle sayılan bütün bunlar belirtilmektedir. Bunu üretebilmek için, bize bütün 64 renk çeşidini sağlayan, renkler için ağ direncine bağlı olduğu düşünülen 2 bitlik ADC kullanılmaktadır. [12].

BÖLÜM 2. FPGA ve VHDL

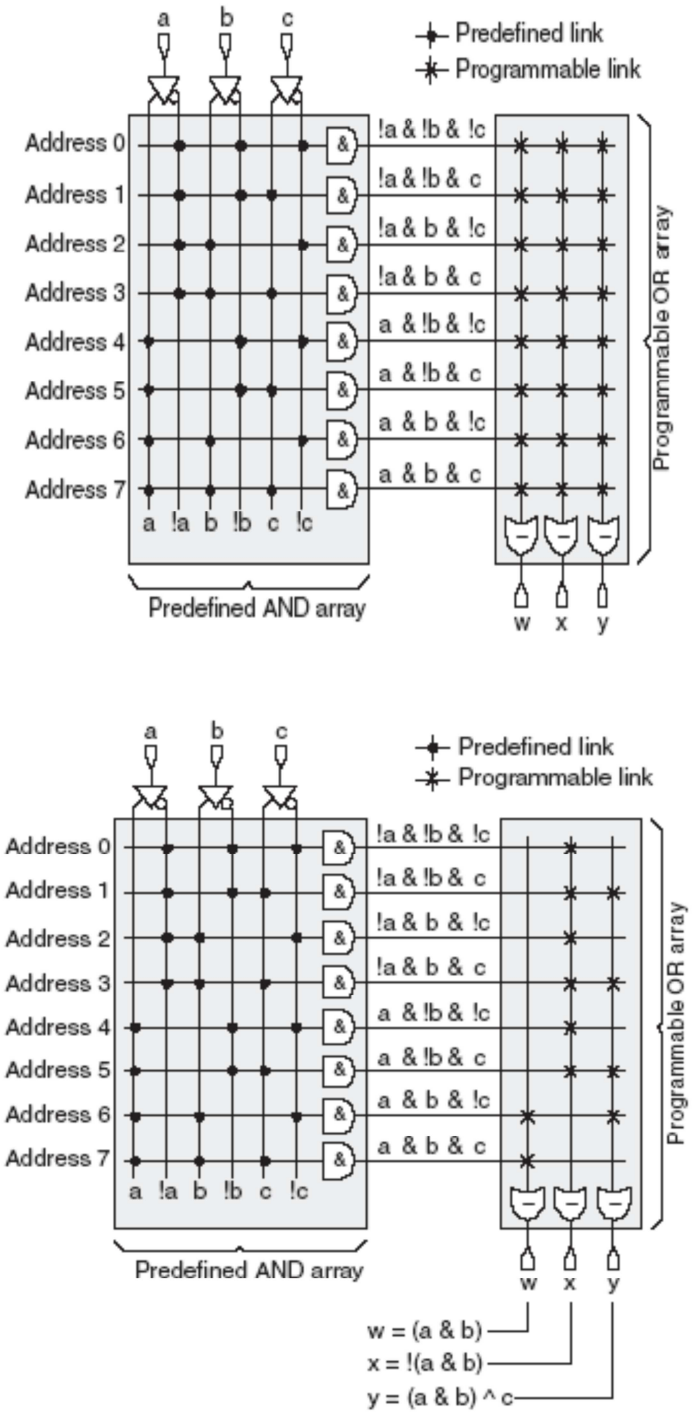
2.1. FPGA Tarihçesi

Xilinx ilk FPGA ürününü 1984 yılında duyurdu. Şekil 3.1'de teknolojilerin ortaya çıktıkları yıllar verilmiştir. Çubuklardaki beyaz kısımlar, ürünün yaygın kullanım ve kabul görmediği zamanlara işaret eder. Tasarım mühendisleri 1990'lı yılların başlarına kadar FPGA'lara rağbet etmediler.



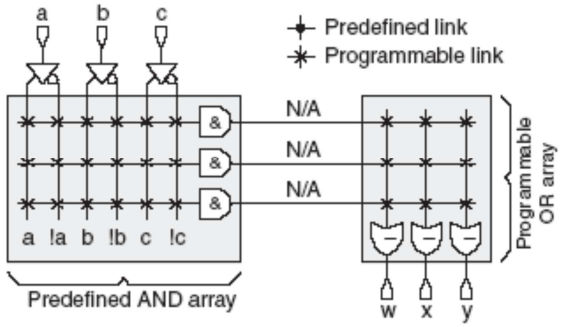
Şekil 2.1. Farklı teknolojilerin ortaya çıkışları

1990 yılından sonra FPGA kullanımının artmasına bağlı olarak fiyatlarının düşmesi, gelişen teknoloji ile birlikte FPGA'lerin kapasitelerinin artması, kullanıcıya büyük kolaylıklar sağlayan gelişmiş tasarım programları ve örnek uygulamaların kolay erişilebilir hale gelmesi FPGA'i günümüzde popüler yapan özelliklerdir. Savunma sanayi, sayısal işaret işleme, telekomünikasyon, tıbbi görüntüleme ve otomotiv sanayi FPGA'lerin uygulama alanlarından bazılarıdır.



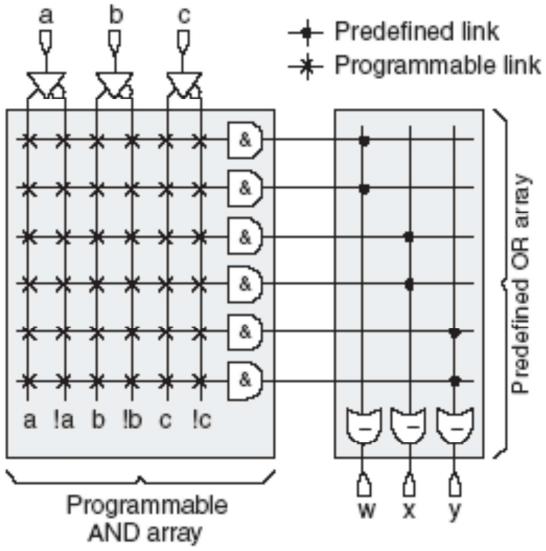
Şekil 2.2. İlk PLD'ler basit PROM'lardı

Bu yapıda VE dizileri üretimden sabit olup VEYA dizilerindeki bağlantılar kullanıcı tanımlıdır. Bir sonraki büyük adım PLA'nın ortaya çıkmasıydı:



Şekil 2.3. PLA yapısı

Görüldüğü gibi PROM'dan farklı olarak VE dizisindeki bağlantılar da PLA'larda programlanabilir bağlantılardır. İki alanın da programlanabilir oluşu hız problemini ortaya çıkardı. Bunun üzerine ortaya çıkan PAL yapılar, PROM'ların tersi yapıdaydılar ve VE dizisi tarafı programlanabilir olduğundan PROM'lardan daha esnektiler.

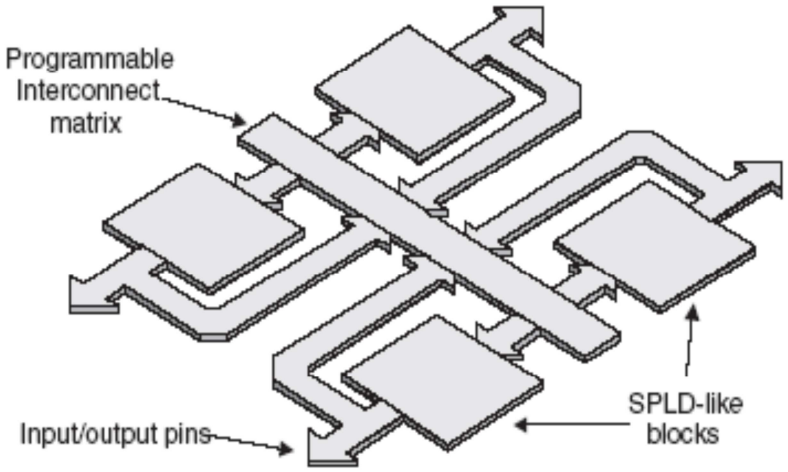


Şekil 2.4. PAL yapısı

1980'lerde CPLD denen (Complex PLD) daha karmaşık PLD yapıları ortaya çıktı. Genellikle CPLD'ler, SPLD (Simple PLD) bloklar (çoğunlukla PAL yapılar) ve bunlar arasındaki programlanabilir ara yollardan oluşur. Ara yollar 100 civarı kablo taşıyabilir. 1980'lerde sayısal tüm devre dünyasında bir boşluk hissediliyordu.

Bir tarafta SPLD ve CPLD'ler gibi konfigüre edilebilir, hızlı, ucuz, esnek ve kolay tasarım süreçlerine sahip iken, karmaşık ve zor problemlerin çözümünde kullanılamıyorlardı.

Öte yandan zor problemleri çözebilen ASIC, pahalı ve zor tasarım anlamına gelmekte ve bir kez üretildiğinde tüm devre üzerinde değişiklik yapılamamakta idi. 1984 yılında XILINX bu boşluğu dolduracak cihazı piyasaya sürdü. [6].



Şekil 2.5. Yaygın CPLD yapısı

2.2. FPGA Nedir?

FPGA alanda programlanabilir kapı dizileri olarak adlandırılırlar. Çok genel bir tanımlamayla FPGA, dijital tasarımlarda kullandığımız farklı lojik kapıların milyonlarcasının tek bir entegre üzerinde toplanmasıdır. Bu lojik kapılar istenildiği gibi programlanabilir ve istenilen her türlü dijital tasarım bu elemanlar üzerinde gerçeklenebilir.[2].

FPGA'ler istenen fonksiyona göre iç yapısı kullanıcı tarafından değiştirilebilen donanım-programlanabilir entegre devrelerdir. FPGA (Field Programmable Gate Array) programlanabilen bir yongadır (entegre veya çipdir). [5].

Tasarım sırasında büyük esneklik sağlaması ve paralel işlem yapabilme kabiliyeti sebebiyle FPGA kullanımı günümüzde oldukça yaygınlaşmıştır. [5].

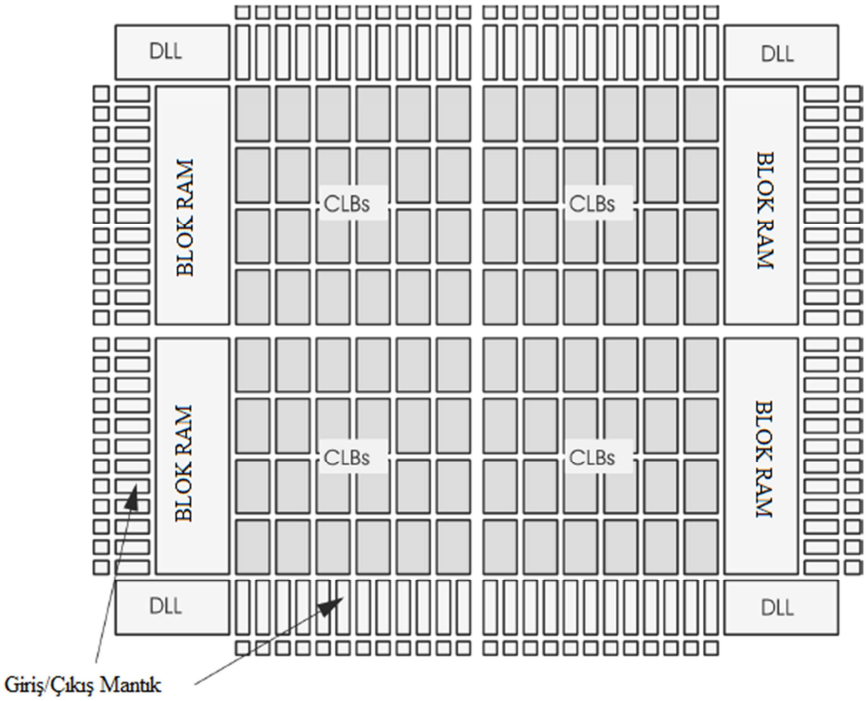
FPGA'leri programlamak için HDL (Hardware Description Language-Donanım Tanımlama Dili) kullanılır. VHDL ile Verilog en yaygın kullanılan iki HDL türüdür. FPGA, 1985 yılında Xilin'in kurucusu Ross Freeman tarafından icat edilmiştir. FPGA mimari yapısı, tamamı kullanıcı tarafından belirlenebilen lojik bloklar ve ara bağlantılarından oluşmuştur. FPGA'ler Günümüzde 10 milyon eşdeğer kapı sayısına (iki girişli AND olarak) ulaşmıştır. FPGA'ler Yapılandırılabilir Lojik Bloklar (Configurable Logic Blocks - CLBs) ve Giriş/Çıkış Blokları (I/O -Blocks- IOBs) içermektedir. CLB hem senkron hem de kombinasyonel devreler için temel lojik kaynakları içerir.[10].

1. Bu kapı diziliş ya da sıralanış içeriğine bir göz atalım. FPGA’nın organize edilmiş hali olarak:
2. CLBs (Configurable Logic Block): Düzenli bir şekilde yonga üzerinde ayarlanmış hücre düzenekleri/sıraları. CLB’ler programlanabilir.
3. IOBs (Input Output Block): Ayarlanmış yonga giriş çıkış pine’ler ile programlanabilir bir ağı oluşturur.
4. (Programmable Switch Matrix): Bağlar arası bloklar/öbekler ki bu bağlar, programlanabilir CLB üzerinden veya bilgi/veri toplama prosedürü IOB’s arasında bağlantı kurmalarına izin verir. [5].

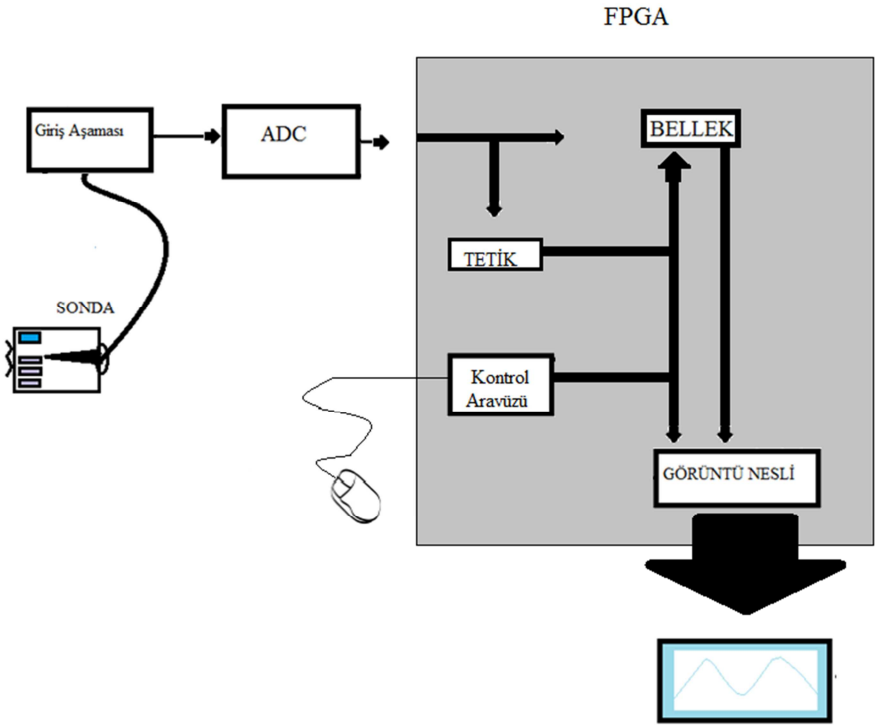
Neden FPGA: FPGA’ler bize paralel işlem kabiliyeti sunan ve içyapısını istediğimiz fonksiyon ve uygulamaya göre programlayarak değiştirebildiğimiz donanımlardır. [2].

Ayrıca başlıca faydası, klasik oskiloskopa zıt olarak görsel anlamda tekrarlanmayan ya da rastgele sinyallerin oluştuğu esnada görülür. Klasik oskiloskoplar düzenli olarak testere/bıçkın üzerinde işaretleri meydana getirmeye yönelik bir işlevselliğe sahip olduğu gibi, olmazsa olmaz doğru görüntünün oluşması için mevcut bir uygulamadan ibarettir.

Diğer taraftan “foxgloves” = yüksükotu olarak da bu gerçekleşmemektedir. Rastgele aralıklarla her hangi bir giriş kısmına kadar görselleştirmek için eğilim gösterecektir. Şekil 2.6.’ FPGA da İçsel derleme/birleşim: CLBs ve IOBs Herbir CLB “Slices” adındaki 2 tane programlanabilir hücreden derlenmiştir. Her “Slices” hücresi her hangi 5 tane değişik mantıksal işlevi içinde barındıracak yetenektedir. CLB ve Slice hücrelerinin uygun yazılımı sayesinde FPGA 5 tane olacak şekilde fakat istediği fonksiyon işlevlerini seçmede/fark etmede başarılı olma imkânını elde etmiştir. FPGA’nın temel anlamda ki karakteristik özellikleri: İçerdiği Slices hücreleri sayesinde ve tek bir (FPGA’nın) çalışmada ki hızı, FPGA’nın içinde uygulanan tasarım esnasındaki boyutunu sınırlandırmaktadır. Şekil 2,6’de FPGA’nın sağladığı bir diğer avantaj ise SoC tasarımı ile ilgili beraberinde pek çok avantajı sunmasıdır. Şekil 2,7’de görüldüğü gibi bizim oskiloskobumuzun şeması yer almaktadır. Burada, analog düzeyinin kazanım ve iyileştirme işlemlerini gözlemleme imkânını elde ederken, diğer yandan da FPGA da işlenmiş/uygulanmış dijital elektronları gözlemleme fırsatını vermektedir. [12].



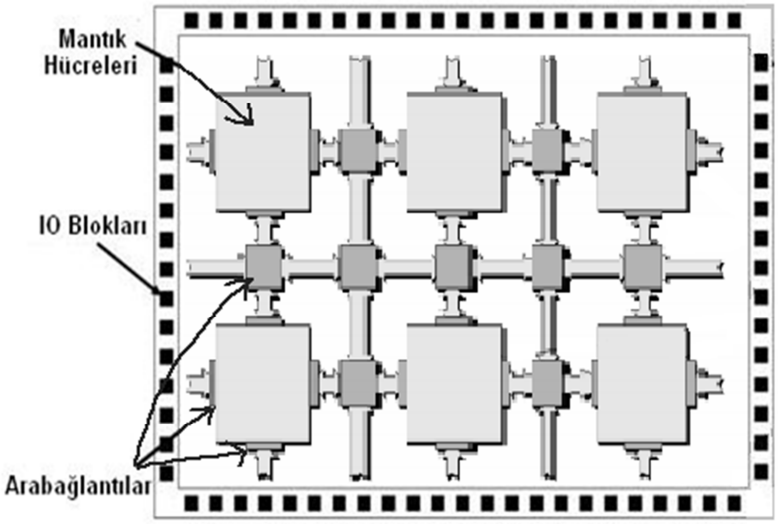
Şekil 2.6. FPGA iç bileşimi-CLB-IOBS



Şekil 2.7. Dijital osiloskop blok diyagramı

2.3. FPGA Yapısı ve Pinleri

FPGA temel olarak Mantık Hücreleri (Logic Cell), Giriş/Çıkış Blokları (IO Block) ve Ara bağlantılardan oluşur. [9].

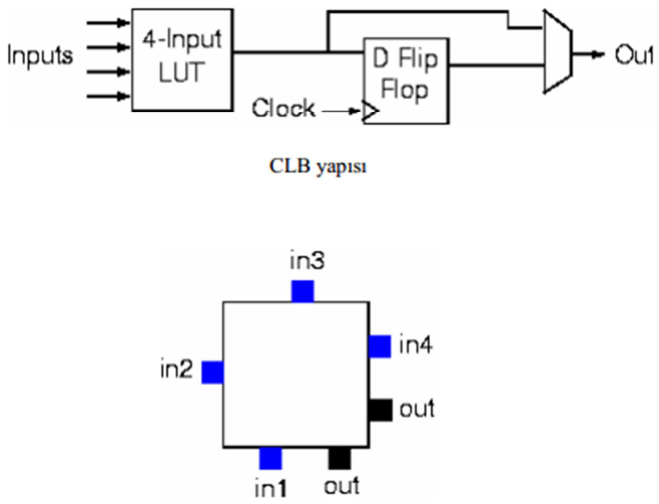


Şekil 2.8. Mantık hücresi (Logic- Cell)

FPGA, mantık hücrelerinden meydana gelir. Bunlar, Girdi/Çıktı hücrelerine ya da diğer mantık hücrelerine bağlanabilirler. Mantık hücreleri programlanabilen tümleşik elemanlara sahiptirler. Günümüz teknolojisinde sık kullanılan sayısal fonksiyonlar FPGA içerisinde toplanmıştır ve kullanıcılar bu fonksiyonlara sahip donanımı tek yonga içerisinde tasarlayabilirler. FPGA içerisinde; toplayıcı, çarpıcı, bölücü, kod çözücü, mikroişlemci, programlanabilen girdi/çıktı blokları yer almaktadır. [7].

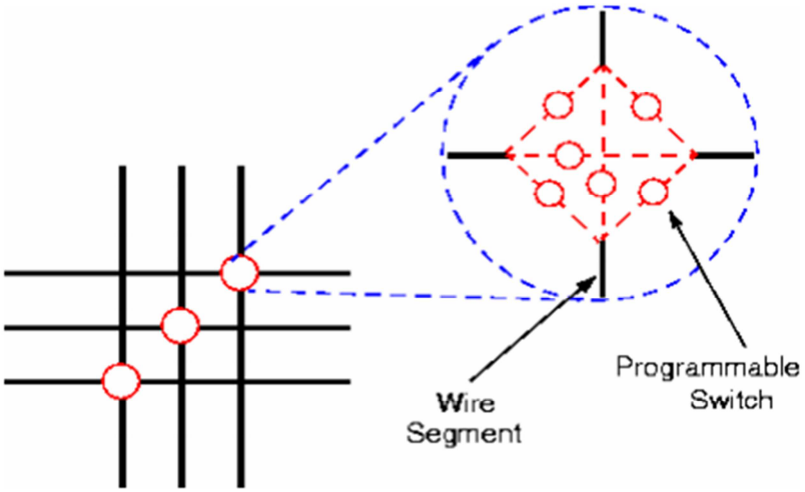
Çok genel bir tanımlamayla FPGA, dijital tasarımlarda kullandığımız farklı lojik kapıların milyonlarcasının tek bir entegre üzerinde toplanmasıdır. Bu lojik kapılar istenildiği gibi programlanabilir ve istenilen her türlü dijital tasarım bu elemanlar üzerinde gerçeklenebilir. [8].

FPGA'in ana yapısını Mantık Hücreleri oluşturur. Bir Logic-Cell 1 adet Lookup Table (LUT), 1adet D Flip-Flop ve bir adet 2 to 1 Mux' tan oluşur. CLB (LUT) yapıları genel olarak 5 girişten oluşurlar, bu girişlerden 4 ü LUT (look-up table) girişleri iken diğer giriş ise D flip flop yapısının saat girişidir. Konfigure edilebilir lojik blok yapısına ait genel blok şema, giriş ve çıkış pinlerinin FPGA programlanabilir birimi üzerindeki yerleşimi Şekil 2,9'de verildiği gibidir. [15].



Şekil 2.9. CLB yapısı ve CLB giriş çıkışlarının FPGA üzerindeki görünümü

CLB (LUT) giriş çıkışlarının FPGA üzerindeki konumlarının şemada gösterildiği gibi olmasının nedeni hat üzerinde oluşabilecek gecikmeleri, hat uzunluklarını eşit yaparak olabildiğince eşit olmasını sağlamaktır. Yatay ve düşey konumlarda bulunabilen CLB (LUT)'ler istenildiği durumlarda kak kat bağlanabilmekte ve bu bağlantılar programlanabilir ara bağlantılar yardımıyla gerçekleştirilmektedir. FPGA üzerindeki programlanabilir ara bağlantı yapısı Şekil 2,10'te verilmiştir. [15].



Şekil 2.10. Programlanabilir ara bağlantılar

Yapının sahip olmuş olduğu programlanabilir anahtarlar sayesinde istenilen tasarım gerekli anahtar bağlantılarının yapılması ile gerçekleştirilebilir. Ayrıca bu yapılara ek olarak saat dağıtımını uygun bir şekilde gerçekleştirebilmek amacıyla FPGA' ler üzerinde dağıtım hatları bulunmakta ve ardışıl birimlere saat işaretinin, tasarımın çalışmasını etkilemeyecek şekilde ulaşması sağlanır. LUT 'lar aslında bir mantık işlemi yerine getiren küçük belleklerdir (RAM). N girişli bir LUT, 2N'li bir belleğe işaret eder. Binlerce Mantık Hücreleri'nin birleşimi sonucunda kompleks ve büyük programlar oluşturulur. Mantık hücrelerinin ara bağlantıları matris şeklindeki veri yolları ve programlanabilir anahtarlarla (FPGA yüklenen programa göre) sağlanır. FPGA tasarımı, her bir mantık hücresinin uygulayacağı fonksiyonu ve programlanabilir anahtarların durumunu (açık/kapalı) belirleyerek bu mantık hücreleri arasındaki bağlantıları tanımlar. [15].

FPGA pinleri genel olarak 2 katagoriye ayrılır:

a. Ayrılmış pinler (Dedicated pins)
b. Kullanıcı pinleri (User pins)

Ayrılmış pinler: Bir FPGA'de tüm pinlerin %20 ila %30'u ayrılmış pin’lerdir. Bu pinler, FPGA'de gerçekleştirdikleri özel fonksiyonlara göre 3'e ayrılır.

1. Güç Pinleri: FPGA için gerekli olan güç ve ground (Toprak) sağlayan pinlerdir.
2. Konfigürasyon Pinleri: Oluşturulan programın FPGA yüklenmesi (download) için kullanılan pinlerdir.
3. Clock Pinleri: Clock sinyalleri için ayrılmış özel pinlerdir.

Kullanıcı pinleri: Bunlar kullanıcı tarafından konfigüre edilebilen standart I/O pinleridir. Input, Output, Input/Output olarak üç kategoriye ayrılır. Her bir I/O pini FPGA'de bir I/O hücresine bağlıdır. I/O hücrelerinin güçleri VCC - I/O tarafından sağlanır. Eski FPGA' ler birden fazla VCC - I/O pinine sahip olmalarına rağmen, bütün pinler aynı voltajla beslenirdi. [3].

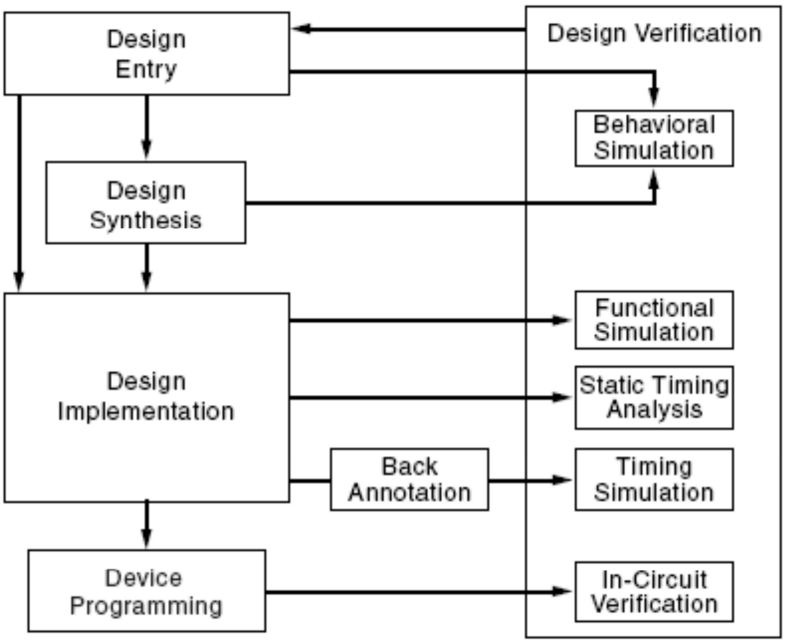
Yeni üretilen FPGA'lerde ise I/O'lar gruplara ayrılabilir ve bu gruplar farklı voltajlardan beslenebilirler. Böylelikle bir grup I/O pinleri 3,3 V ile çalışırken; diğer grup I/O pinleri de 2,5 volt ile çalışabilmektedir.

2.4. FPGA İle Tasarım

Kullanıcı, donanım tanımlama dilini (VHDL ya da Verilog) kullanarak FPGA ile uygulamasını yapmak istediği devreyi tasarlar. Bilgisayar destekli tasarım yazılımı yardımıyla, yazılan kod sentezlenir. [3]. Diğer bir bilgisayar destekli tasarım yazılımı kullanılarak sentezlenen koda karşılık uygun bağlantılar FPGA içinde gerçekleştirilir. Yazılım gerekli veri akısını oluşturup FPGA içerisine aktararak FPGA’yı programlar.

Günümüzde FPGA büyük mantık devrelerinin tek entegre olarak gerçekleştirilmesi için kullanılmaktadır. Bu kullanım pazara sunulacak ürünün tasarım zamanını, güç tüketimini ve kapladığı yeri azaltmaktadır. Şekillendirilebilir hesaplama (reconfigurable computing), FPGA genel amaçlı mantıksal eleman olduğu için yüksek performanslı hesaplama işlemleri için kullanılabilir. [3].

Geliştirilen bir algoritmanın donanım olarak gerçekleştirilmesi, paralel yapıların kullanılmasına ve mikroişlemcilerden çok daha hızlı işlem yapılmasına imkân vermektedir. FPGA'nın en önemli özelliklerinden birisi de çalışma sahasında hatta çalışırken bile tekrar programlanabilmesidir. [7].



Şekil 2.11. FPGA tasarım adımları

Tasarım Girişi: FPGA Platformunda tasarım süreci bir lojik tasarımın girişi ile başlar. Bu işlem HDL ve/veya Şematik olarak yapılabilir.

Davranış Simülasyon: Tasarım girişi yapıldıktan sonra sistemin davranışsal simülasyonu yapılır ve tasarımın doğrulaması yapılır. Eğer düzeltme gerekirse tasarım girişinde değişikliler yapılır.

Tasarım Sentez: Tasarım girişinde HDL kullanılması durumunda gerçekleme yapılabilmesi için HDL'den sentezlemesi yapılır. Şematik tasarımda ise, hedeflenen cihazda bulunan temel birimler kullanıldığından bu adıma gerek yoktur. Sentezleme sonucunda tasarlanan sistemi oluşturan lojik blokları ve bunların bağlantılarını içeren bir netlist oluşturulur.

Post-Sentez Davranış Simülasyon: HDL sentezlendikten sonra elde edilen sistemin davranışsal simülasyonu yapılır ve tasarımın doğrulaması yapılır. Eğer düzeltme gerekirse tasarım girişinde değişikliler yapılır ve tekrar sentezlenir.

Tasarım Uygulama: FPGA platformunda gerçekleme işlemi sırayla, "Translate", "Map" ve "Place&Route(PAR)" olarak üç alt işlem olarak yapılır. "Translate" alt işlemi gerçeklemeye verilen netlist ve tasarım kısıtlamalarını (örneğin giriş/çıkış pinlerini) birleşitirir. "Map" işlemi tasarımı hedeflenen cihaz üzerinde bulunan kaynaklara eşleştirir."PAR" işlemi de tasarım zamanlama kısıtlarına göre tasarımı hedef cihaz üzerindeki yerleşimini ve yolları belirler.

Fonksiyonel Simülasyon: Fonksiyonel simülasyon, "Translate" işleminden sonra, gerçeklemeden önce tasarımda bulunan lojik hataların tespit edilmesi için yapılır. Zamanlama bilgisi henüz belli olmadığından simülasyon testleri birim zaman gecikmesi ile yapılır.

Statik Zaman Analizi: Sabit Zamanlama analizi, "Map" işleminden sonra tasarımdaki lojik gecikmeleri inceleme imkânı sağlar. [3].

Her ne kadar bağlantı yol gecikmelerini içermese de, tasarımın lojik gecikmelerinden kaynaklanabilecek potansiyel hatalarının belirlenmesinde yardımcı olur.

Zamanlama Simülasyonu: Zamanlama Simülasyonu, PAR işlemi sonucunda belirlenen hedef cihaz konfigürasyonunun, standart lojik ve yol gecikmeleri ile beraber ifade edilen (Back Annotation) HDL üzerinden yapılır. Böylece tasarımın

hedef cihaz üzerinde yol ve lojik gecikmeleri ile beraber simülasyonu ve doğrulaması yapılır.

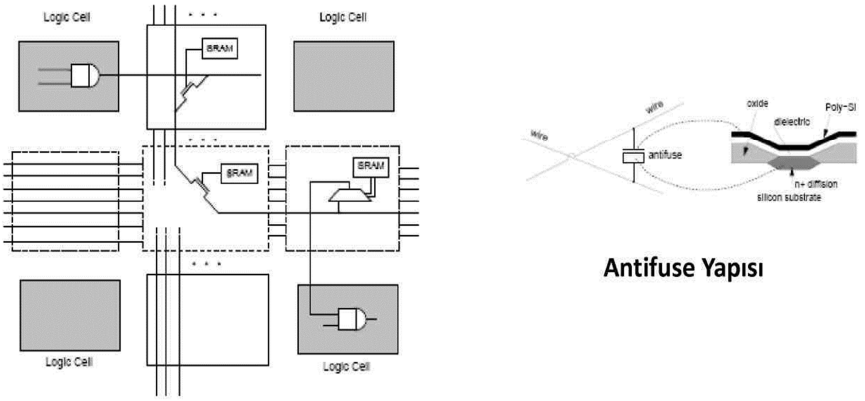
Cihaz Programlama: Gerçeklenen tasarım programlama formatına dönüştürülerek hedef cihaz'ın konfigürasyonun saklandığı PROM'a yüklenir.

Devre İçinde Doğrulama: Hedef cihaza konfigürasyon yüklendikten sonra, yazılım araçları kullanarak çalışma zamanında tasarım doğrulaması yapılabilir. [10].

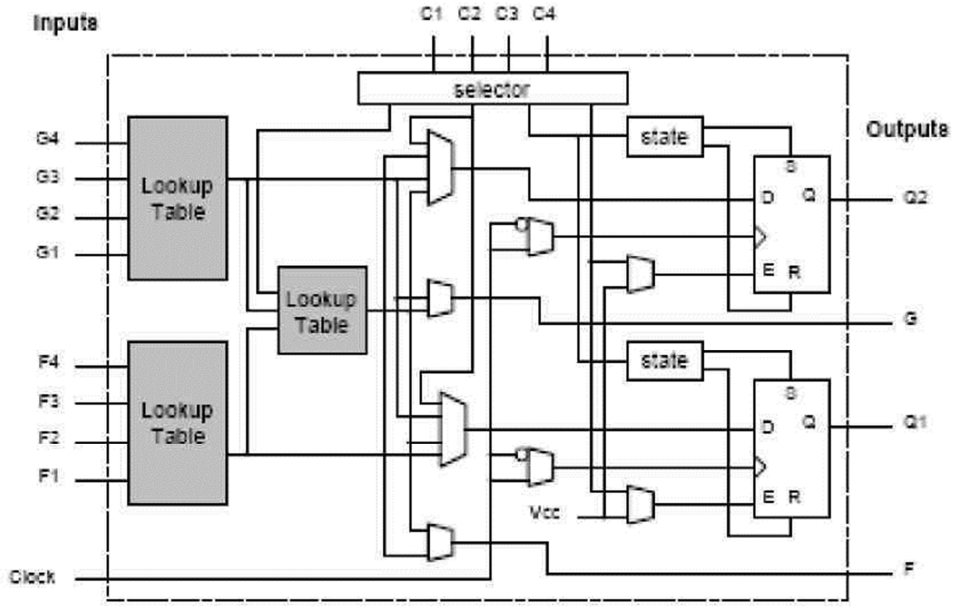
Son zamanlarda FPGA kullanımı birçok alanda hızla artmaktadır. FPGA yüksek hızda seri haberleşme ve bilgi iletimini yerine getirmek amacıyla kullanıldığı gibi bulanık mantık uygulamalarında da başarılı bir şekilde kullanılmaktadır. Yapılan bir çalışmada robot kontrolü için tasarlanan algoritma FPGA üzerinde gerçekleştirilmiştir.

Diğer bir çalışmada yüksek performanslı 3 boyutlu bir görüntüleme sistemi, FPGA'nın paralel işlem yeteneğinden faydalanılarak gerçekleştirilmiştir. Kadir tarafından yapılan çalışmada LabVIEW yazılımı ve CompactRIO donanımı kullanılarak online güç kalitesi izleme sistemi oluşturulmuştur. Bakhri. Tarafından yapılan çalışmada ise, LabVIEW yazılımı ve CompactRIO donanımı ile bir asenkron motor için gerçek zamanlı durum izleme uygulaması gerçekleştirilmiştir. Elektrik enerji sistemlerinde güç kalitesi analizi ile ilgili olarak yapılan benzer bir çalışmada da yine FPGA tabanlı işlemcisi olan donananım kullanılmıştır. [3].

FPGA tabanlı CompactRIO teknolojisi kullanılarak Tıp alanında yapılan bir çalışmada ise, hastaların kalp sesi verileri toplanarak ayrık dalgacık dönüşümü yöntemiyle erken tanı yöntemleri üzerine çalışılmıştır. Bir diğer çalışmada ise FPGA tabanlı gömülü kontrol sistemi kullanılarak bir motosiklet prototipi üzerinde yapılan çalışma sunulmuştur. [7].



Şekil 2.12. Nasıl programlanır? SRAM kontrollü programlanabilir anahtar. [9]



Şekil 2.13. Nasıl çalışıyor? Xilinx Xc4000 configurable logic block (Clb) [9]

FGPA'lerin, OTP(One Time Programmable) ve SRAM-tabanlı programlanabilen olarak iki temel tipi vardır. Bu iki FPGA tipinde lojik blokların fiziksel gerçeklenmesi ve bağlantıları oluşturmak için kullanılan mekanizma farklıdır. Daha yaygın olarak kullanımı olan SRAM-tabanlı FPGAler istenildiği kadar tekrar programlanabilir. SRAM FPGA'ler zaten özel bir çeşit hafıza gibidir ve sisteme her güç verildiğinde yeniden programlanır. Bu nedenle SRAM FPGA'leri programlamak için PROM veya benzeri bir hafıza birimine ihtiyaç duyulur. SRAM lojik blokları,

loijk kapılar yerine LUT(Lookup Table-Arama Tablosu) içerir. LUT girişlere göre çkış sonucunu belirler. [10].

2.5. VHDL Nedir?

VHDL sayısal elektronik sistemleri tanımlamak ve bir FPGA içine gömmek için kullanılan bir programlama dilidir. Bu dil Birleşik Devletler Hükümeti'nin Çok Yüksek Hızlı Tüm devreler (Very High Speed Integrated Circuits-VHSIC) programı çerçevesinde 1980 yılında başlatıldı. Program çerçevesinde geliştirme çalışmaları sürerken tüm devrelerin (ICs) yapılarını ve fonksiyonlarını tanımlamak için standart bir programlama diline ihtiyaç duyulduğu anlaşılır. Bu sebepten dolayı VHSIC Hardware Description Language (VHDL) geliştirilir. [13].

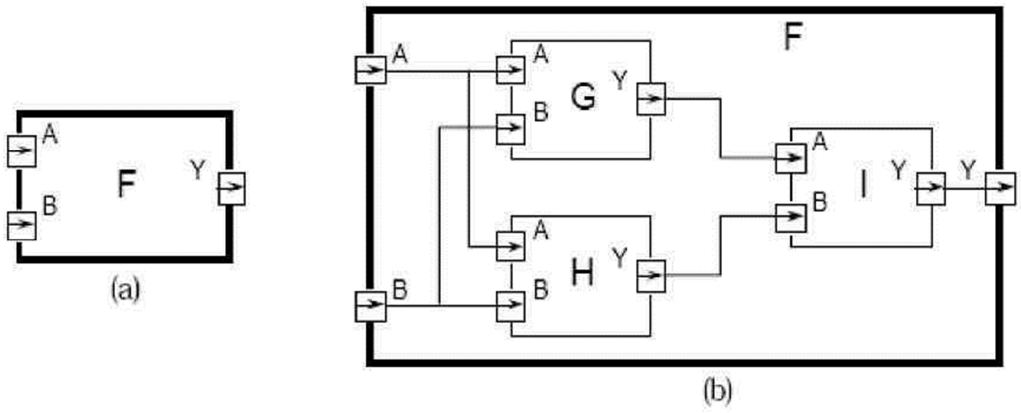
Elektronik sistemlerdeki teknolojinin ilerlemesi, tasarım yöntemlerinin de gelişmesini gerektirmiştir. Bu sebeple, geleneksel "kâğıt ve kalem kullanarak tasarımı yap" ve "devreyi kurarak denemeleri yap" yöntemlerinin yerini "tanımla ve sentezle" yöntemleri alır. Tanımlama ve sentezleme yöntemleri, donanım tanımlama dillerinin (Hardware Description Language) ortaya çıkmasına neden olur. VHDL (Very high speed integrated circuit Hardware Description Language) günümüzdeki donanım tanımlama dillerinin en popüleridir. [14].

VHDL 1986'da IEEE standardı olması için IEEE' ye teklif sunuldu. 1987'nin Aralık ayında IEEE 1076 standardı olarak kabul edilmesine kadar VHDL birçok revizyona uğradı. [13].

2.6. VHDL Yapısı

Birinci olarak bir tasarımın yapısı tamamen VHDL ile tanımlanabilir. Bu tasarımı alt tasarımlara ayrıştırır ve bu alt tasarımların birbirleriyle bağlantıları sağlanabilir. İkinci olarak bu dil yakın bir programlama dili olan dillerle hazırlanmış fonksiyonların kullanılmasına izin verir. Üçüncü olarak yapılan tasarımlar işlevi olan sistemde kullanılmadan önce simülasyonu yapılabilir. Tasarımcılar hızlı bir şekilde

alternatifleri geliştirip bir prototip olmadan bu alternatiflerin doğruluklarını sınayabilirler. [13].



Şekil 2.14. Bir yapı tanımlama örneği

Bir sayısal elektronik sistemi giriş ve/veya çıkışlarıyla beraber bir parça olarak tanımlanabilir. Çıkışın elektriksel değerleri girişin elektriksel değerlerinin bir ve birkaç fonksiyonu olarak karşımıza çıkarlar. Şekil 2,14'da bir sayısal sistemin modül halinde gösterimi mevcuttur. Görüldüğü üzere F modülünün A ve B olmak üzere iki girişi ve bir Y çıkışı vardır. VHDL terminolojisi kullanılarak F bir tasarım (entity) olarak düşünülürse girişler ve çıkışlar port diye nitelendirilebilir. Modüle işlevi kazandırmanın bir yolu modülü alt modüllere ayırmaktır. Bu modüllerin portlarını oluşturduktan sonra, sinyal tipi yapılarla birbirleriyle haberleştirmektir. Şekil 2,14'da bir bütün yapının daha küçük yapılara nasıl ayrıldığı gösterilmiştir. Burada F modülü G, H ve I alt modüllerine ayrıştırılıp bunların istenen fonksiyonu gerçekleştirecek konfigürasyonunu yapmak gerekecektir. Bu tür tanımlamaya "yapısal tanımlama" denir. F bir yapısal tanımlama olduğu gibi G, H ve I alt modülleri de yapısal tanımlama olabilir. [13].

2.7. VHDL Veri Tipleri

Öncelikle yazılacak programlarda karşımıza çıkacak olan terimleri incelememiz gerekmektedir. Bu terimler yazılan her temel programda genelde bulunur. Bunları anlamak programın yapısını anlamak ve yeni yapılar oluşturmak için önemlidir.

Entity: Kelime olarak anlamı mevcudiyet, varlık ve vücuttur. Bütün tasarımlar entity'lerle ifade edilir.

Architecture: Kelime anlamı olarak mimaridir. Yukarıda tanımlanan programın vücudu yani temel program içinde barındırdığı mimarilere göre simülasyonu yapılır. Mimari (architecture) programın yani yapının davranışını belirler. Bir temel yapı (entity) içinde birkaç adet mimari olabilir. Mesela bir mimari, davranışı belirlerken diğeri de yapıyı belirleyebilir.

Configuration: Bu deyim genel olarak programda kullanılacak konfigürasyonu belirtir. Configuration tasarım için bir parça listesi olarak tanımlanabilir. Bu hangi entity hangi özelliklerin kullanılacağını ifade eder. Genel olarak parçaların nerede ve nasıl kullanılacağını belirten bir menü gibi ifade edilebilir.

Package: Kelime anlamı olarak pakettir. Programı yazarken kullanılacak olan alt programlar ve veri tipleri bu yapıyla paketlenir. Bunu tasarım yaparken kullanacağımız bir araç kutusu (toolbox) gibi düşünebiliriz.

Driver: Kelime anlamı olarak sürücüdür. Burada sinyal kaynağı anlamında kullanılmıştır. Eğer sinyalin iki kaynağı varsa ve her iki kaynak da aktifse kaynağın iki sürücüsü var denir, Diğer bir deyimle veri sağlayıcıları.

Bus: Bus genel olarak akla veri yolu kavramını getirir. Ama VHDL de bus özel bir işarettir ki bu işaretin kaynağı sönüktür. Kullanım itibariyle yine yol özelliği taşır.

Attribute: Kelime anlamı olarak öznitelik, özelliktir. Biz VHDL de kendimiz yeni veri tipleri tanımlarız. Bunu yaparken veri tipinin özelliklerini kendimiz belirleriz. Aynı durum nesne tanımlarken de geçerlidir. Biz nesneyi oluştururken onun herhangi bir durum karsısında onun davranışını tanımlarız yani ona bir özellik kazandırırız.

Generic: Bu VHDL terimi bir entity'nin içine bilgi gönderebilmek için kullanılan bir parametredir. Bunu da tanımlarken veri tipini belirtmek gerekir. Mesela entity'nin kenar ve çıkan kenar gecikmeli bir kapı seviye modeli ise; inen kenar ve çıkan kenar arasındaki gecikmeyi biz entity'e ancak Generic tipi parametrelerle bildiririz.

Process: VHDL'in temel yürütme birimlerinden birisidir. Genelde biz bunu sıralı program (for, if kullanılarak yazılan program) yazarken kullanırız. Çünkü ilerde anlatacağımız üzere VHDL normal dillerde olduğu gibi programı satır satır yürütmez. Program satırları hepsi birden paralel olarak çalınır. FPGA işlemcilerin hızı da buradan kaynaklanır. Ama paralel çalışan bir programda if deyimi ve for döngüsü gibi yapıların kullanılması olanaksızdır. Bu yüzden bu yapıları kullanmak için biz Process yapısını kullanırız. Process ile tanımlı program parçaları normal programlar gibi satır satır çalışır. [13].

VHDL 'de birçok veri tipi bulunur. Biz öncelikle nesnel tiplerden bahsedeceğiz. Daha sonra skaler sayılardan bileşik dizilere, yapılardan dosya tiplerine kadar birçok tipin incelenmesi yapılacaktır. Bu bölüm nesne türleri ve veri tipleri adında iki başlıkla incelenecektir. VHDL'de gerçek hayattaki nesnelere benzer nesneler vardır. Bunlar bazen bir veri yolu bazen bir değişken olabilir. Birinci bölümde bunlar incelenecektir. Tabi ki bunların kullanım yerleri farklıdır. [4].

2.7.1. Nesne tipleri

Programlama dilleri bildiğimiz gibi yapısal ve nesneye yönelik olmak üzere ikiye ayrılır. Yapısal dillere örnek olarak hepimizin duyduğu ya da kullandığı C, Pascal, Ada, Fortran verilebilir. Nesnel diller, dillerde nesneler, gerçek hayattaki nesneler, model alınarak oluşturulur. Gerçek hayatta her nesnenin kendisine uygulanan etkilere göre bir davranışı veya bir işlevi vardır. Nesnel programlarda yapılan iş bundan farklı

değildir. Nesne oluşturulur, ona davranışlar ve işlevler kazandırılır. Modüler programlamaya uygun olduğu için büyük hacimli yazılımlarda çok kullanılır. Bilinen ilk nesnel dil Smalltalk'tır. Sonraları C++, Java gibi nesnel diller oluşturulmuştur. VHDL yapısal dil özelliği gösterdiği gibi içinde nesnel özellikler barındırır. Bir VHDL nesnesi ancak aşağıdaki tiplerden birisiyle veya birkaçıyla oluşturulur.

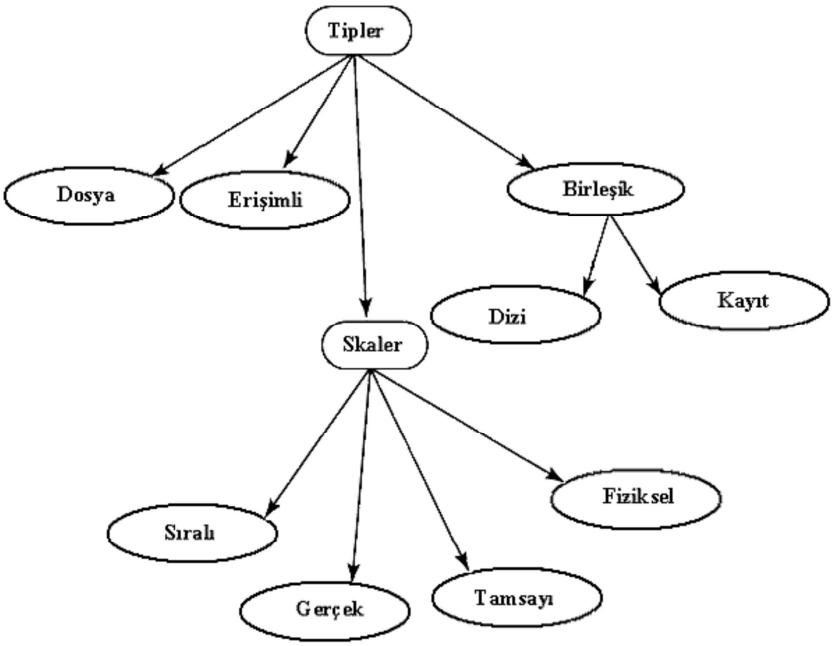
a. Signal (SİNYAL) : Bu component portları arasında karşılıklı bağlaşmayı sağlayan kablo gibi davranır.
b. Variable (DEGİŞKEN) : Değişken veriler için kullanılır. Genelde lokal olarak sadece Process'ler içinde kullanılır.
c. Constant (SABİT) : Sabit değerler için kullanılır onun değeri sistem oluşumunun sonunda hiç değişmez. [13].

VHDL çok geniş bir çerçevede birçok veri tipini destekler. En başta; yeni bir veri tipi oluşturmak istenirse bir veri tipi tanım şablonuna gerek vardır. Bir veri tipi tanım şablonu, veri tipinin adı ve tipin tanım bölgesinden oluşur. Daha ileride verilen örneklerle konu daha iyi anlaşılacaktır. Yeni bir veri tipi paket tanımlama anında, varlık (entity) tanımlama anında, mimari tanımlama anında, altprogram ve process tanımlama anlarında oluşturulabilir. Yeni bir tip tanımlama şöyle olur;

[TYPE tipin_adı IS tipin_markası]; Burada tipin_markası biraz anlamsız gelebilir. Bu kısımda normal veri tiplerinden herhangi birisi veya bunlardan oluşmuş bir karmaşık veri tipleri olabilir. Bundan sonraki birkaç örnekte bunların nasıl kullanıldığı daha iyi anlaşılacaktır. Sinyaller ve sabitler için var olan kapsama kuralları buradaki kullanımlarında da geçerlidir. Sekil 1.'de VHDL'de geçerli olan veri tipleri gösteren diyagram görülmektedir. Burada dört ana kategori göze çarpmaktadır. Bunlar; skaler tipler, birleşik tipler, erişimli ve dosyasal tiplerdir. Skaler tipler tamsayı ve gerçek sayı basit(sade) tiplerden oluşmaktadır. Birleşik tipler ise diziler ve kayıt tiplerini içerir. Erişimli tipler kasıt ise klasik programlama dillerindeki işaretçilere denktir. Son olarak dosyasal tipler ise tasarımcıya dosya nesneleri oluşturma olanağı sağlar. [13].

[Skaler Tipler (Scaler Types)]; Skaler veri bir zaman diliminde sadece bir tane değer tutabilen nesnelerdir. Bir nesne kendi kendine birden çok değer içerebilir. Bir zaman diliminde bu değerlerden sadece bir tanesini tutabilir. Skaler tipler şu dört kısımdan oluşur. [13].

1) Tamsayılar
2) Gerçek sayılar
3) Sıralı sayılar
4) Fiziksel sayılar



Şekil 2.15. VHDL veri tipleri diyagramı

2.7.2. İşaretler (Signals)

Sinyal nesneleri entity'leri birbirine ve aynı zamanda form modellerine bağlar. Sinyallerin görevi entity'leri birbirleriyle dinamik verilerle haberleştirmektir. Sinyal tanımlaması şöyle olur:

[SIGNAL sinyalin_adı: sinyalin_tipi [:=başlangıç değeri]]; SIGNAL tanımlamasının ardından sinyal ismi gelir. Sinyal isimleri ihtiyaç kadar belirtilir. Belirttiğimiz isim sayısınca sinyal üretilir. Tanımlamada da belirtildiği gibi sinyal nesnelerine ilk değer atanabilir. Son olarak sinyaller entity, architecture ve package (paket) yapılarının içinde tanımlanabilir. Paketler her yerde kullanılabileceği için burada yapılan tanımlamalar global özellik taşır. Çünkü bu tanılamalar ancak paket kullanıldığı zaman aktif olurlar. Sinyaller için bir takım kapsama (yerel, genel, ...) kuralları bulunmaktadır. Sinyaller; varlık (entity), mimari ve paket tanımlama anlarında tanımlanabilirler. Paket içi tanımlamalarda sinyaller genel bir tanımlama yapılmış gibi davranırlar. Çünkü paket bilindiği üzere birçok mimari veya varlık tarafından kullanılabilir. Ayrıca değer atamaları da tanımlandığı yerde olmayabilir.

2.7.3. Değişkenler (Variables)

Değişkenler yerel olarak sadece Process'ler veya altprogramlar içinde geçici değerleri tutmak için kullanılırlar. Sinyal tanımlamanın tersine değişken tanımlamada, değişkenin tanımlandığı yerde değer atanamayabilirler. Değer atamaları daha sonraki satırlarda olabilir. Değişkenler ortaya çok hızlı çıkıp işini yapıp ortadan kaybolurlar yani hafızada yer kaplamazlar. Bir değişkenin tanımlanması genel hatlarıyla aşağıdaki gibi olur:

[VARIABLE değişkenin_adı: değişkenin_tipi [:=başlangıç_değeri]] ; Burada her değişken adı birkaç tane değişken adı olabilir. Her isme ait bir değişken üretilir. variable_type değişkenin veri tipini tanımlar. Bu veri tipi kendi tanımladığımız bir veri tipi olabilir. Yukarıda belirtildiği gibi değişkenler ancak ve ancak altprogram ve process tanımlamalarında kullanılırlar. İlgili örneği aşağıda görebiliriz. [13].

```
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
ENTITY and5 IS
```

```
 PORT ( a, b, c, d, e : IN std_logic;
 POR q : OUT std_logic);
END and5;

ARCHITECTURE and5 OF and5 IS
BEGIN
   PROCESS(a, b, c, d, e)
   VARIABLE state : std_logic;
   VARIABLE delay : time;
BEGIN
   state := a AND b AND c AND d AND e;
    IF state = '1' THEN
   delay := 4.5 ns;
 ELSIF state = '0' THEN
   delay := 3 ns;
 ELSE
   delay := 4 ns;
END IF;
   q <= state AFTER delay;
 END PROCESS;
END and5;
```

Bu örnek 5-girişli bir AND kapısının mimarisini belirtmiştir. Burada yukarıda belirtildiği gibi sadece process içinde iki adet değişken bildirimi yapılmıştır. Bunlardan birisi durum (state), diğeri ise gecikme (delay) değişkenidir. Durum değişkeni girişlerin AND fonksiyonun geçici verileri tutmak için kullanılır. Yine aynı biçimde, çıkış değerlerini sıralarken oluşan gecikme değerlerini de bu delay değişkeni tutar. Bu iki değişkenin de tuttuğu veriler statik (static) değildir. Çünkü bunların değerleri girişlere (a,b,c,d,e) bağlı olarak değişir. Şurası unutulmamalıdır ki; verileri tutmak için değişkenler yerine sinyaller (signals) de kullanılabilirdi. Ama kullanılmadı. Bunun birçok nedeni vardır. Bunlardan başlıcalar şunlardır:

Doğal (içsel) olarak değişkenler veri tutma ve değiştirmede sinyallerden daha üstündür. Çünkü değişkenler ve yaptığı işlevler aniden oluşurlar. Fakat sinyallerin oluşması için bir plan oluşturulmaktadır.

Değişkenler daha az yer kaplarlar. Sinyaller yapısı bakımından oluşturulması için o andaki durum la ilgili daha fazla veriye ihtiyaç duyarlar ve bu veriler elde edildiğinde çok yer kaplarlar. Sinyal kullanırken veri senkronizasyonunu sağlamak için WAIT deyiminin kullanılmasına ihtiyaç vardır.

Burada tanımlanan a,b,c,d veya e girişlerinden herhangi birisinin değer değişiminde programın içinde tanımlanmış process kısmı çalışmaya başlar. Durum değişkeni olan state tüm girişlerin değerini tutmak üzere görevlendirilir. Daha sonra state değişkeni tabanına dayanılarak gecikmeleri tutmak üzere de delay değişkeni görevlendirilir. Gecikmelere dayanılarak bu değerler delay değişkenine sevk edilir. Bu değerlere dayanılarak çıkış sinyali q, durum değişkeni olan state değişkenin değerini alır ve çıkış üretilmiş olur. [13].

2.7.4. Sabitler (Constants)

Sabitler bir tasarımda çok özel verileri tutmak için kullanılırlar. Sabitler tasarımcıya çok iyi dokümantasyona sahip bir model ortaya koyma olanağı sağlar.

Eğer tasarımı yapılan modelde değişmez değerlere ihtiyaç varsa sabitler kullanılır. Tasarımcı isterse sabitlerin değerini elle değiştirip programı tekrar derleme ve optimum değeri bulma olanağına kavuşur. Bir sabitin çok iyi dokümantasyon sağlamasından kasıt şudur: Tasarlanan modelde çok sayıda pi sayısı kullanılacak olsun; her kullanışta pi sayısının direkt değeri olan 3,1414'ü kullanmaktansa bir adet sabit tanımlamalıdır. Bu tanımlama aşağıdaki gibi olur;

[CONSTANT PI: REAL := 3,1414]; Burada bu tanımlamanın yapılması programın okunurluğunu artırmıştır. Programa bir tertip düzen katmıştır. Genel sabit bildirimi aşağıdaki gibi olur;

CONSTANT sabitin_adı: veri_tipi [:= başlangıç_değeri]; Sabitlerin tanımlanmasında esneklikler söz konusudur. VHDL değer ataması için ertelemeye olanak sunmaktadır. Sabitin değeri, sabiti tanımlanan yerde atamak zorunda değildir. Bir sabit paket tanımlanırken tanımlanabilir, değeri ise paket gövdesi tanımlanırken (koşullu atama olabilir) atanabilir. Sabilerin de sinyaller gibi bir takım kapsama kuralları vardır;

Bir sabit birçok varlık (entity) tarafından kullanılan bir pakette tanımlanıyorsa, o sabit genel (global) bir sabit olmuş olur.

Bir sabit eğer bir process içinde tanımlanmışsa sadece o process içinde kullanılabilir. Bir mimaride tanımlanan bir sabit, processler dâhil hiçbir ifade tarafından kullanılmayabilir. Şimdi veri tipleri adı altında diğer veri tiplerini açıklamaya başlayabiliriz. [13].

2.7.5. Tam sayılar (Integer)

Buradaki kullanılan tamsayılar tamamen matematikte kullanılan tamsayılara benzerler. Bu tamsayılarla matematikte yaptığımız toplama, çıkarma, çarpma ve bölme işlemlerini yapabiliriz. VHDL LRM tamsayıların maksimum aralığını değil de minimum aralığı olan (-2.147.483.647)'dan +2.147.483.674'ye kadar olan kısmını destekler. Bunlar Standart Library (kütüphane)'deki Standart pakette tanımlanmıştır. Bu standart paket VHDL diline ihtiyaç olan bütün veri tipleri tanımlanmıştır. Fakat bu paket normal bir varlık veya paket tarafından özel olarak çağrılamaz. Aşağıda tamsayı ile ilgili birkaç örnek verilmiştir;

```
ARCHITECTURE test OF test IS
BEGIN
  PROCESS(X)
  VARIABLE a : INTEGER;
  VARIABLE b : int_type;
BEGIN
 a := 1; --Ok 1
```

```
 a := -1; --Ok 2
  a := 1.0; --error 3
 END PROCESS;
END test;
```

İlk iki ifadede (1 ve 2) pozitif ve negatif tamsayıların atamaları gösterilmektedir. Üçüncü ifadede ise tamsayı olarak tanımlanmış bir değişkene gerçek sayı gibi atama yapılmıştır. Bu ifade derleyicide bir sorun olarak algılanacak ve hata mesajı bildirecektir. Çünkü VHDL veri tipi bakımından oldukça güçlü bir dildir. [13].

2.7.6. Gerçek sayılar (Real number)

Gerçek sayılar matematikte bilinen gerçek sayılardan VHDL'de nesneler oluşturmak için kullanılır. Ayrıca kesirli sayılar gibi tamsayıların gösteremediği sayıları göstermek için de kullanılırlar. Tamsayılar gibi gerçek sayıların minimum değer aralıkları Standart Kütüphanedeki Standart pakette tanımlanmıştır. Bu aralık 1.0E+38'den baslar ve 1.0E+38'de son bulur. Aşağıdaki örnekteki birkaç ifadede kullanım biçimleri görülmektedir;

```
ARCHITECTURE test OF test IS
   SIGNAL a : REAL;
 BEGIN
  a <= 1.0; --Ok 1
  a <= 1; --error 2
  a <= -1.0E10; --Ok 3
  a <= 1.5E-20; --Ok 4
  a <= 5.3 ns; --error 5
END test;
```

Burada görüldüğü üzere gerçek sayı tipinden bir sinyal (signal) nesnesi tanımlanmıştır. Bütün gerçek sayılar yazılırken mutlaka bir ondalık kısmı bulunur.

Bu ondalık kısmın değeri sıfır bile olsa belirtilir. Bu derleyicinin gerçek sayıları diğer tiplerden ayırt etmesini sağlar. Yukarıdaki örnekte birinci ifade bir nesneye bir değer atamasını göstermektedir. Anlatıldığı gibi ondalık kısmın değeri sıfır olmasına rağmen bildirimi yapılmıştır. Nitekim ikinci ifadede sayısal değer aynı olmasına rağmen imla hatası yapılmıştır. Çünkü orada yapılan atama; tanımlanan sinyal tipinin bir tamsayı olduğunu bildirmektedir. Beşinci ifadede ise imla hatası yoktur fakat tip uyuşmazlığı vardır. Çünkü tanımlanan nesnenin tipi zaman (time) değil gerçek sayıdır. [13].

2.7.7. Sıralı sayılar (Enumerated)

Sıralı sayılar soyut kavramları modellemek için çok güçlü tiplerdir. Bu tip verilerin veri tiplerini tasarımcı oluşturabilir. Bu değerler karakter veya sayı olabilirler. Buna örnek bir veri x, abc, siyah gibi isimlerle tanıtılabilir. Bazen bu bir yılın ayları bazen haftanın günleri bazen rastgele isimler olabilirler. Karakteristik bir örnek vermek gerekirse dört durumlu bir simülasyon sistemi bu tip verilerle şöyle tanımlanabilir;

[TYPE fourval IS (‘X’ , ‘0’ , ‘1’ , ‘Z’)]; Görüldüğü üzere bu sistemde dört adet veri gösterilebilir. Bunların açıklaması şöyle olur;

X’—Bilinmeyen değer
‘0’—Mantıksal ‘0’
‘1’—Mantıksal ‘1’
‘Z’—Yüksek empedans durumu

Diğer bir örnek ise aşağıda olduğu gibi elemanları renk olan veri tipidir.

[TYPE renk IS (kırmızı, sarı, mavi, yeşil, turuncu)]; Bu örnekte veri tipleri tamamen soyuttur. Yani fiziksel bir değere sahip değildirler. Bu yüzden sıra dışıdırlar. Bu araştırmaya uygun bir örnek olarak bir komut seti bu çeşit bir tanımlama ile tanımlanabilir:

[TYPE instruction IS (add, sub, lda, ldb, sta, stb, outa, xfr)]; Bu çeşit veri tipleri mantıksal olarak durum makinesi modellemelerinde kullanılabilirler. Durum makineleri genel olarak ASIC veya FPGA aygıtlarının kontrol mantığını oluşturmada kullanılmaktadırlar. Bu metot işlem sırasının doğru yürütülmesinde çok büyük kolaylık sağlar. [13].

2.8. VHDL'in Avantajları

VHDL, büyük ölçekli dijital tasarımların dokümantasyonu, doğrulanması ve sentezlenmesi işlemlerini gerçekleştirmek için kullanılır. Aynı VHDL kodları ile bu üç farklı işlemin gerçekleştirebilmesi sayesinde, tasarım süresi ve kolaylığı açısından büyük kolaylıklar sağlamaktadır. [14].

Güç ve Esneklik: VHDL tasarımları yazılım tabanlı olduğundan dolayı birçok fonksiyon tek bir donanım ile gerçekleştirilebilir.

Çipten Bağımsız Tasarım: Özel işlevleri yerine getiren birçok entegrenin VHDL ile gerçeklenmesi mümkün olduğu için tasarımlarımız çipten bağımsızdır.

Taşınabilirlik: Yapılan tasarım bir dosya olduğundan mevcut başka bir tasarım dosyasının içine veya bir çipin içine eklenebilir. Böylelikle yapılan her tasarım farklı bir modül olarak düşünülebilir.

Test Edilebilirlik: Tasarımı yapılan VHDL devreleri emülatörler yardımıyla gerçek zamanlı olarak, simülatörler yardımıyla da sanal olarak test edilebilir.

Piyasaya Hızlı Çıkış ve Düşük Maliyet: Mevcut bir sistem üzerinde yapılması istenen bir değişiklik, donanım olarak değil de yazılım olarak yapıldığından tasarım süresi kısa sürmektedir ayrıca bir çok entegre, VHDL çipleri ile tasarlanabildiğinden tasarım maliyeti düşüktür. [14].

2.8.1. Tasarım

Sayısal bir sistemin VHDL tanımlaması yapılırken dört temel yapı kullanılır. Bu yapılar entity bölümü, mimari bölümü, konfigürasyon ve paketlerdir. Sayısal bir sistem, modüllerin bir hiyerarşi ile birleştirilmesiyle oluşturulur. Her modül VHDL de bir entity ile ifade edilir. Her entity ile giriş çıkışları ve fonksiyonu tamamen belirlenmiş olan bir donanım yapısı gösterilir. Her tanımlamanın entity, bildirimi ve mimari bölümleri olmak üzere iki bölümü vardır. Entity bildirimi tasarımın dış bağlantılarının, mimari bölümü ise, içeride yapılacak işlemlerin gösterilmesinde kullanılır. Pakette ise pek çok tanımlamada ortak olarak kullanılacak genel bilgiler verilir. Konfigürasyon ile yapısal tanımlamada kullanılacak alt sistemlerin giriş çıkış bilgileri tanımlanır Tablo 2.1'de entity ve mimari bölümlerinden oluşan bir yarım toplayıcının VHDL kullanılarak tanımlanması yer almaktadır. [14].

Tablo 2.2. Bir yarım toplayıcının VHDL kullanılarak tanımlanması

entity	entity half_adder is PORT(a: in std_logic; b: in std_logic; s: out std_logic; c: out std_logic); end half_adder;

mimari	architecture arch of half_adder is
	begin
	s <= a xor b;
	c <= a and b;
	end arch;

2.8.2. Benzetim

VHDL'de yazılan bir tasarım tanımının, bir VHDL benzetim programından faydalanılarak doğruluğu kontrol edilebilir. Tanımlamanın benzetimini yapabilmek için benzetim programına bir takım sayısal girişleri verilir, program daha önceden belirlenen aralıklarla bu sayısal girişleri tasarımın modeline uygular ve çıkışları üretir.

Bu sonuçlar tasarımcı tarafından gözlenerek modelin istenildiği gibi çalışıp çalışmadığına karar verilir. Benzetim, tasarımın her aşamasında kullanılabilir. Tasarımın yüksek seviyelerinde yapılan benzetim, sadece tasarımın fonksiyonel

davranışı hakkında bilgi verir. Bu aşamada benzetim çok hızlıdır, fakat benzetim sonuçlarından tasarımın gerçek devre elemanları ile çalışması ve zamanlaması konusunda çok fazla bilgi elde edilemez. Daha düşük tasarım aşamalarına gidildikçe benzetim daha uzun sürecektir, fakat benzetim sonuçlarından tasarımın çalışması ve zamanlaması konusunda daha fazla bilgi elde edilebilir. [14].

2.9. VHDL İle FPGA Arasındaki İlişki

VHDL ve FPGA arasında ilişki veya bağlantının varlığını şu şekilde açıklamaya çalışalım: VHDL IEEE standart tanımlama dilidir. VHDL ile modellenerek yani VHDL editöründe yazılan bir kod, program ondan sonra FPGA (Field programmable gate array - sahada programlanabilir kapı dizileri) modellemek istediğiniz tasarımınızın çıktısını FPGA kısmına yükleyerek üzerinde gerçekleme işlemini yapabilirsiniz. FPGA içinde, mantık kapıları ve flip-floplardan oluşan lojik birimler, kolon veya matrislerde sıralanmıştır.

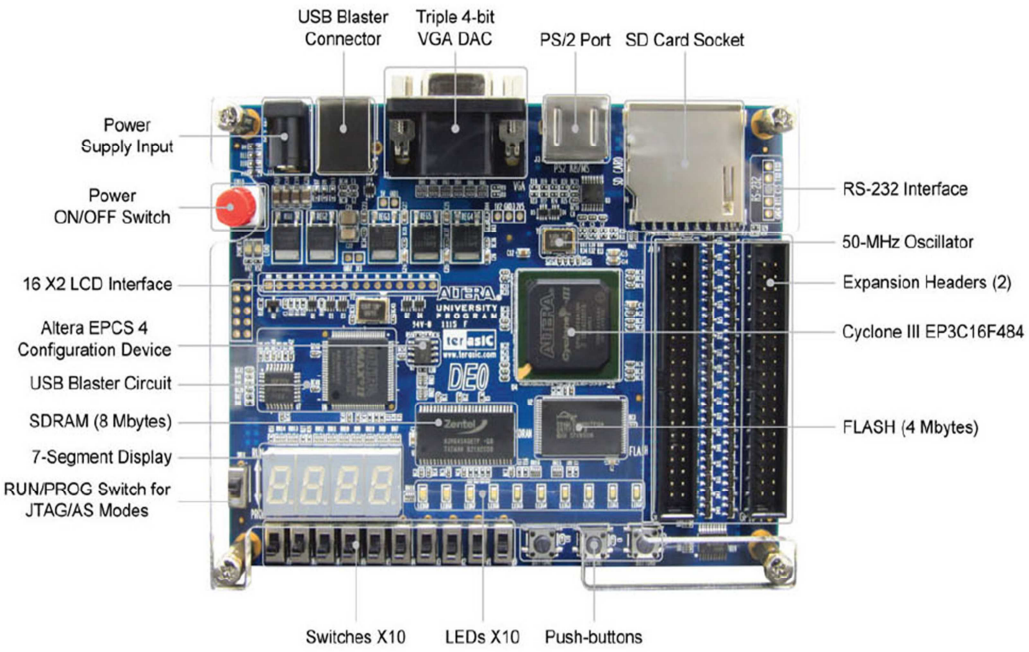
Birimler arasındaki programlanabilir bağlantı ağları sayesinde lojik birimler diğer lojik birimlerle birleşerek daha çok birim ile daha karmaşık işlemler gerçekleştirilebilir. [4].

BÖLÜM 3. TERASIC DE0 GELİŞTİRME KARTI

3.1. DE0 Geliştirme Kartı FPGA Platformu

Bu projede ilk tasarımlar ALTERA DE0 FPGA platformunda tasarlandı. Bu platformun kolay temini, RAM belleğinin yüksek olması, FPGA işlemcisinin modeli, dâhili VGA çıkışlarının olması ve kullanacağımız Osiloskop ile dolaylı da olsa bağlanabilmesi, seçilmesindeki etkenlerden bazılarıdır. DE0 FPGA platformu üzerinde birçok donanım ile birilikte gelmektedir. [2].

Burada Altera DE0 boardun düzeni ve bileşenleri hakkında bilgiler vermekteyim mesela Altera DE0 kitu aşağıdaki resimde gösterilmiştir Aşağıdaki şekilde Altera DE0 kit kurulu düzeni görülmektedir. Bu düzende DE0 kit bağlantılarını ve anahtar bileşenlerinin yerleri görülmektedir.



Şekil 3.1. DE0 kit

DE0 kit birçok özelliğe sahiptir kullanıcının basit devrelerini çeşitli mültimedya projeleri için tasarladığı, devrelerini geniş bir uygulamaya izin verir çünkü DE0 geniş bir uygulama kapasitesine sahiptir. [19].

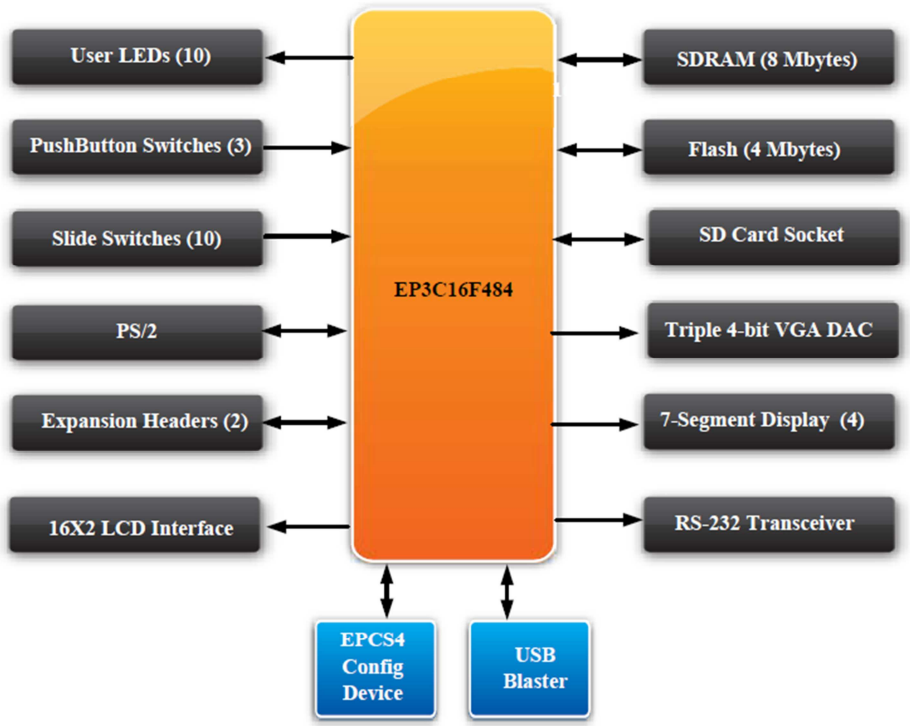
Aşağıda DE0 kit donanımları şunlardır: DE0 Boardu Cyclone III EP3C16F484C6N işlemcisini kullanmaktadır.

Altera Cyclone® III 3C16 FPGA device

1) Altera Serial Configuration device – EPCS4
2) USB Blaster (on board) for programming and user API control; both JTAG and Active Serial
3) (AS) programming modes are supported
4) 8-Mbyte SDRAM
5) 4-Mbyte Flash memory
6) SD Card socket
7) 3 pushbutton switches
8) 10 toggle switches
9) 10 green user LEDs
10) 50-MHz oscillator for clock sources
11) VGA DAC (4-bit resistor network) with VGA-out connector
12) RS-232 transceiver
13) PS/2 mouse/keyboard connector
14) Two 40-pin Expansion Headers

3.2. DE0 Kurulu Blok Şeması

Şekil 3.2.'de DE0 kurulu blok diyagramı verilmiştir. Burada kullanıcının maksimum esneklik sağlaması için, Tüm bağlantıları Cyclone IIII FPGA aygıtı üzerinden yapılır. Böylece kullanıcı herhangi bir sistem tasarımı uygulamak istediğinde FPGA aşağıdaki blok diyagramı inceleyerek yapılandırabilir. [19].



Şekil 3.2. DE0 blok diyagramı

Cyclone IIII 3C16 FPGA

1) 15,408 LEs
2) 56 M9K Embedded Memory Blocks
3) 504K total RAM bits
4) 56 embedded multipliers
5) 4 PLLs
6) 346 user I/O pins
7) FineLine BGA 484-pin package

Built-in USB Blaster circuit (Dâhili USB blaster devresi)

1) On-board USB Blaster for programming and user API (Application programming interface) control
2) Using the Altera EPM240 CPLD

SDRAM

1) One 8-Mbyte Single Data Rate Synchronous Dynamic RAM memory chip
2) Supports 16-bits data bus

Flash memory (Flash Bellek)

1) 4-Mbyte NOR Flash memory
2) Support Byte (8-bits)/Word (16-bits) mode

SD card socket (SD kart yuvası)

1) Provides both SPI and SD 1-bit mod SD Card Access

Pushbutton switches (Buton anahtarları)

1) 3 pushbutton switches
2) Normally high; generates one active-low pulse when the switch is pressed

Slide switches (Slâyt anahtarları)

1) 10 Slide switches
2) A switch causes logic 0 when in the DOWN position and logic 1 when in the UP position

General User Interfaces (Genel Kullanıcı Arayüzü)

1) 10 Green color LEDs (Active high)
2) 4 seven-segment displays (Active low)
3) 16x2 LCD Interface (Not include LCD module)

Clock inputs (Saat girişi)

1) 50-MHz oscillator

VGA output (VGA çıkışı)

1) Uses a 4-bit resistor-network DAC
2) With 15-pin high-density D-sub connector
3) Supports up to 1280x1024 at 60-Hz refresh rate

Serial ports (Serial port)

1) One RS-232 port (Without DB-9 serial connector)
2) One PS/2 port (Can be used through a PS/2 Y Cable to allow you to connect a keyboard and mouse to one port)

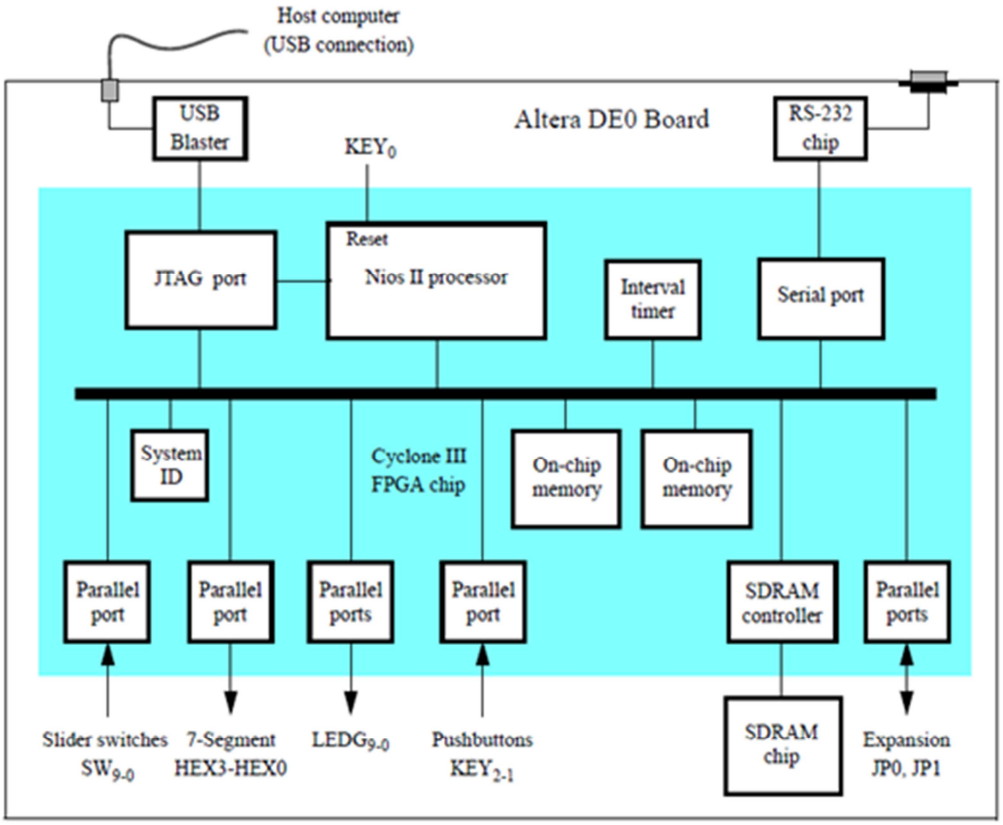
Two 40-pin expansion headers (İki 40 pinli Genişleme başlıkları)

1) 72 Cyclone III I/O pins, as well as 8 power and ground lines, are brought out to two 40-pin expansion connectors
2) 40-pin header is designed to accept a standard 40-pin ribbon cable used for IDE hard drives. [19].

Yukarıda Altera DE0 gelişimi kit'ini görsel olarak ve diyagram olarak göstermeye ve anlatmaya çalıştık. Bizim kullandığımız Altera DE0 kit'i üzerinde eğer yazılım geliştirmek isteniyorsa yada herhangi bir uygulama yapılmak isteniyorsa yine Altera'nın sunduğu Quartus II programını kullanarak ister şematik olarak ister de yazılım olarak üzerinde uygulamalar gerçekleştirebiliriz. Bu sistem, DE0 Temel Bilgisayar adı, bilgisayar organizasyonu ve gömülü sistemlerde giriş deneyler için bir platform olarak kullanılmak üzere tasarlanmıştır. Uygulama kısmında başlangıç deneyleri desteklemek için, sistem sadece birkaç bileşenleri içerir bunlar: Bir işlemci, bellek ve bazı basit I / O çevre birimleridir.

DE0 Temel Bilgisayar bir blok diyagramı Şekil 3,3 gösterilmiştir. Onun ana bileşenleri: Altera Nios II işlemci, programı ve veri depolama için bellek, paralel bağlantı noktaları, anahtarlar, ışıklara bağlı bir zamanlayıcı modülü ve bir seri bağlantı noktası. [19].

I / O cihazları için işlemci ve arayüzleri DE0 kiti Cyclone III FPGA yonga içinde uygulanmaktadır. Şekil 3,3'de gösterildiği gibi bileşenlerin her biri aşağıda tarif edilmektedir.



Şekil 3.3. DE0 temel bilgisayar blok diyagramı

3.3. Bellek Bileşenleri

DE0 Temel Bilgisayar bellek bileşenlerinin üç türü vardır: SDRAM, SRAM, FPGA yonga içerisindeki bellek ve çipin üzerindeki bellek. Bellek türleri aşağıda açıklanmıştır.

1) SDRAM
2) On-Chip Bellek
3) On-Chip Bellek
4) Paralel Bağlantı Noktaları
5) Yeşil LED Paralel Bağlantı Noktaları
6) 7-Segment Paralel Port görüntüler
7) Slâyt Anahtar Pararlel Port

3.4. Quartus II Programı

Quartus II yazılımının iki farklı versiyonu bulunmaktadır. Web-Edition Quartus II' nin ücretsiz ve kısıtlı versiyonu iken Quartus II Subscription – Edition ücretli ve full sürümdür. Alteranın sağladığı verilere göre web-edition full sürümün sağladığı özelliklerin %95 ini sağlayabilmektedir. Burada iki versiyon arasındaki farkları inceleyecek olursak;

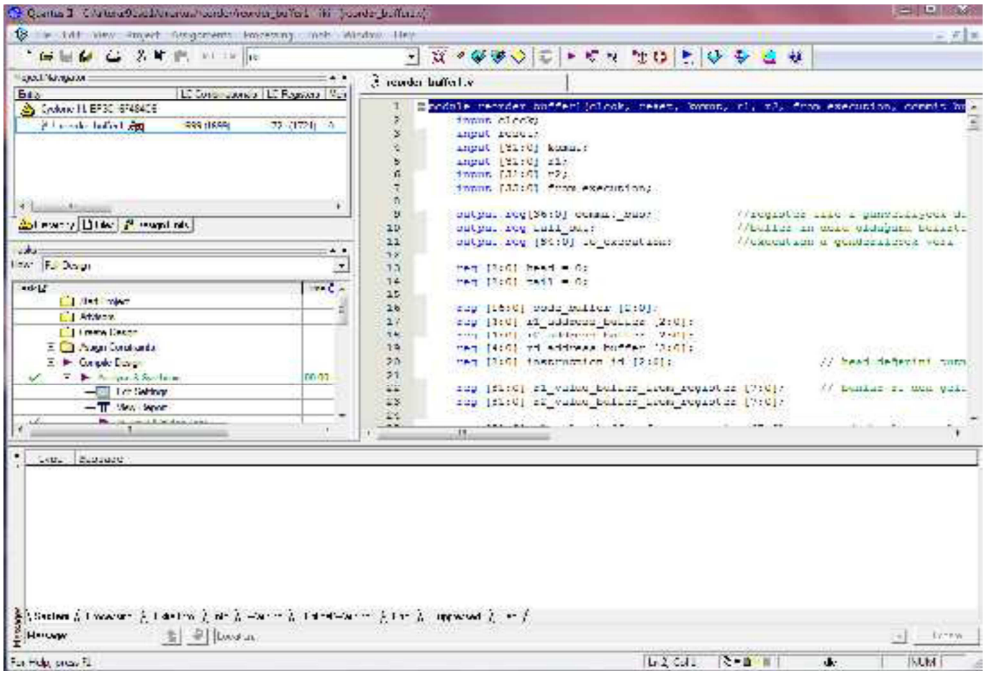
Tablo 3.3. Quartus sürüm kıyaslama programı

	Subscription – Edition	**Web – Edition**
İşletim sistemi	32-64 bit Windows, Linux	32 bit Windows
Desteklenen Aygıtlar	Tüm aygıtlara destek	Bazı aygıtlara destek
Multiprocessor desteği	Var	Yok
Rapid Compile	Var	Yok
Incremental Compilation	Var	Yok
IP desteği	Sınırlı Ücretsiz	Ücretli
Maliyet	Ücretli	Ücretsiz

Quartus II yazılımı tamamen entegre bir yazılım geliştirme ortamıdır. FPGA programlama ile uğraşan kullanıcıların ihtiyaç duyabileceği tüm birimleri içerisinde barındırmaktadır. Bir kullanıcı temelde başka bir araca ihtiyaç duymadan şematik ve HDL dilleri ile dizayn oluşturabilir, oluşturduğu dizaynı sentezleyebilir ve bu dizaynın yerleştirme ve yönlendirme (place and route) işlemlerini gerçekleştirebilir. Ayrıca projenin doğruluğu çip üzerine yüklemeden Quartus yazılımı üzerinde simülasyonu yapılarak doğrulanabilmektedir. Bu işlevlerinin yanı sıra Quartus II programı içerisinde zaman ve güç analiz işlevlerini de barındırmaktadır.

Bu sayede oluşturduğunuz devredeki sinyallerin zamanlamalarını önceden görebilir, kritik bir işlemin bulunduğu kısımda sınırlamaları belirterek Quartus II programının yerleştirme sırasında bu alana daha fazla önem vermesini sağlayabilirsiniz. Quartus II programı aynı zamanda içerisinde bir yükleme yazılımı da barındırmaktadır. Oluşturduğunuz çıktı bu yazılım yardımı ile çipe aktarılır. [2].

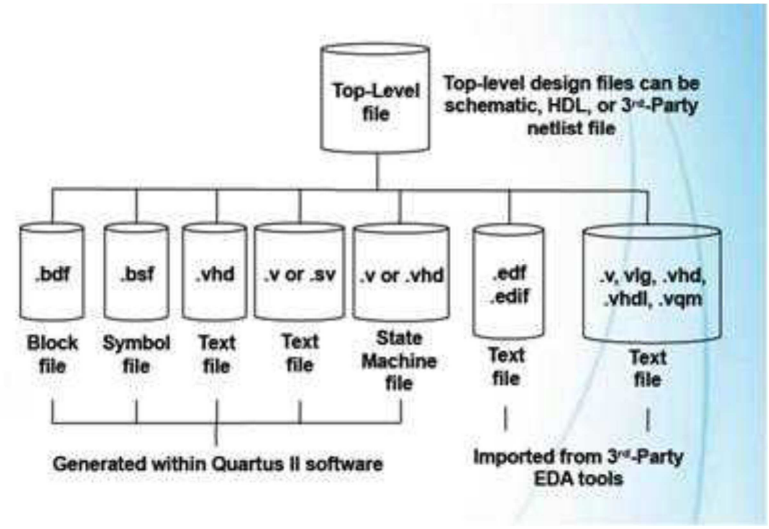
Quartus II programı kullanıcıları 3. parti yazılımlar kullanmaları konusunda da serbest bırakmıştır. Sentezleme simülasyon ve analiz işlemleri 3. parti yazılımlar yardımı ile gerçekleştirilebilirken yerleştirme ve yönlendirme işlemlerinin Quartus programı içerisinde yapılması zorunlu kılınmıştır. Bir proje oluştururken kullanmak istediğiniz araçları programa bildirmeniz yeterli olmaktadır. Baskı devre tasarımı yapacaksanız Orcad gibi kart tasarım programlarının kullanabileceği formatta çıktı üretmektedir.



Şekil 3.4. Quartus II ekran görüntüsü

Quartus II programının ana ekranını Şekil 3,4'deki gibidir. Açılış ekranı varsayılan olarak dört temel pencereden oluşur. Bu pencereler sol üstte Project navigator, altında Task ekranı, en altta mesaj ekranı ve sağ tarafta bulunan çalışma ekranıdır. Diğer ekranlar çok yer kaplamamaları için varsayılan olarak kapalı gelirler. Bu ekranların kullanımınıza göre boyutlarını ayarlayabilir genişletebilir ya da kapatabilirsiniz. Diğer ekranları eklemek ya da kapattığınız ekranları yeniden açmak için view menüsünde bulunan utility windows sekmesinden istediğiniz ekranı tıklamanız yeterlidir. [2].

Quartus programı dizayn girişi için birçok dosya formatını desteklemektedir. Farklı dosya türleri aynı proje altında da kullanılabilmektedir. Böylece HDL dillerini kullanarak tasarladığınız modülleri şematik bir top modül altında birleştirebilirsiniz. Quartus içerisinde, oluşturduğunuz dizayn dosyaları birbiri arasında da dönüştürülebilmektedir. Örneğin oluşturduğunuz bir şematik tasarımı HDL dillerinden birine dönüştürebilir ya da tam tersini yapabiliriz. Şematik tasarımdan HDL tasarıma geçerken bu oldukça eğitici olabilmektedir. File menüsünden create /update komutu ile bu dönüşümleri gerçekleştirebiliriz. [2].

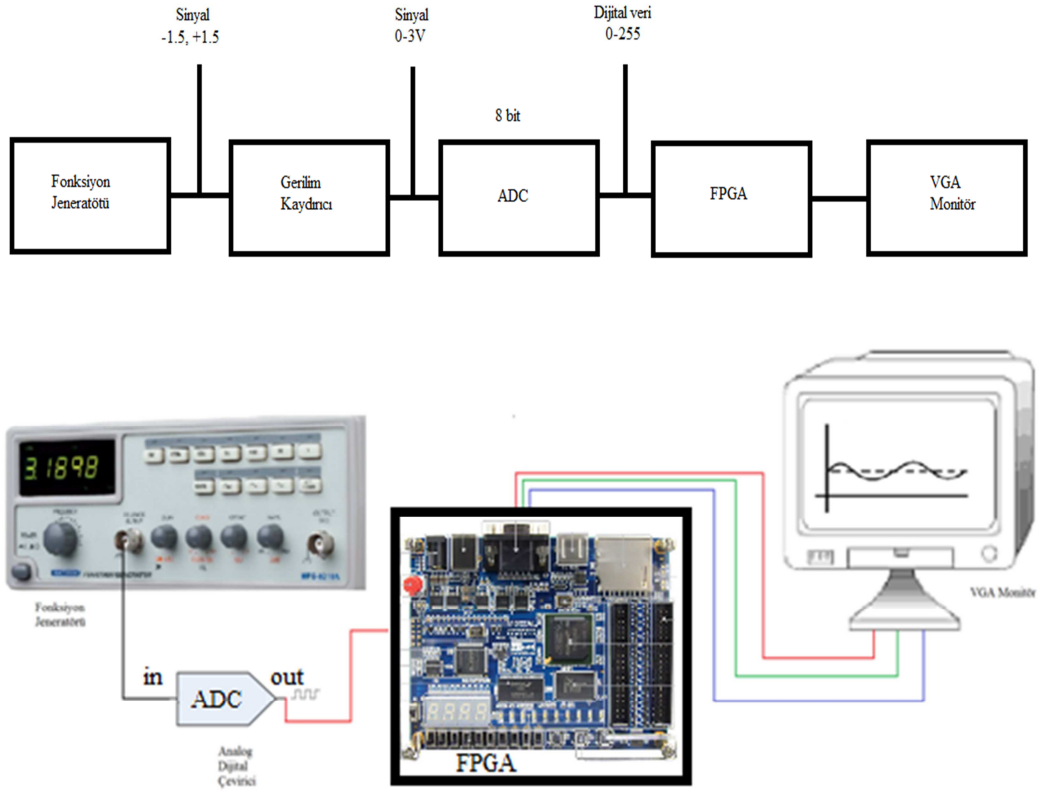


Şekil 3.5. Quartus II dosya türleri

BÖLÜM 4. SİSTEMİN GERÇEKLEŞTİRİLMESİ

4.1. Genel Şema

Aşağıdaki şemada uygulamada kullanılan bloklar görülmektedir. Ölçülecek sinyali fonksiyon jenaratörü üretmekte ve bu sinyal gerilim kaydırıcı ile 0-3 V seviyesine kaydırılmaktadır. Kaydırılmış sinyal ADC ile sayısala çevirilip FPGA'nın sayısal girişlerine uygulanmaktadır. FPGA'da yazılan program ile ölçülen sinyal VGA ekranda grafiksel olarak gösterilmiştir.

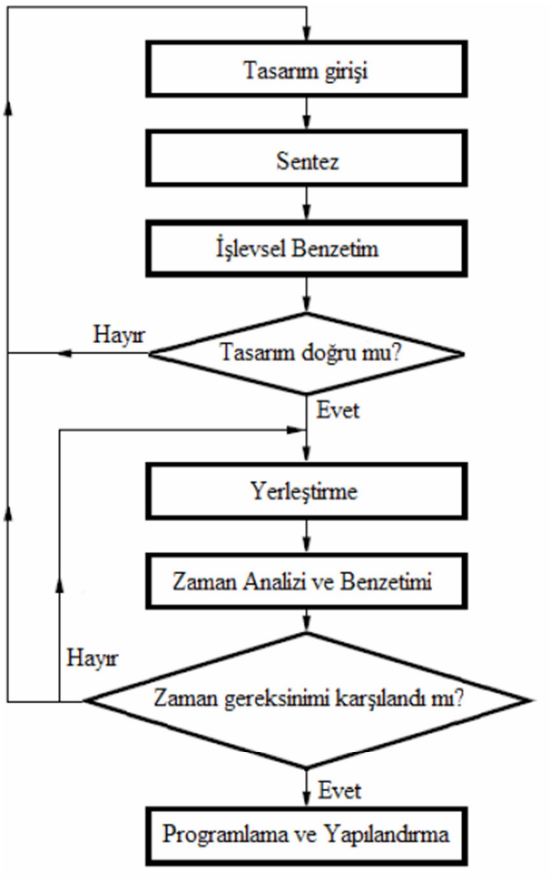


Şekil 4.1. Genel şema

Bu işlem yaparken kullanılan elemanlar fonksiyon jeneratörü, ADC, FPGA DE0 kit CycloneIII EP3C16F484C işlemcisi ve görüntüyü görmek için 640x480 ayarında VGA Monitör yukarıdaki şemada bulunmaktadır. Bu işlem yapılırken RGB, HS, VS, Clock, ADC, FJ gibi sinyal bacakları kod kısmında tanımlamak gerekir ve ayrıca hepsinin ayrı ayrı pin bağlantıları Quratus II pin tanımlama editöründen yararlanarak pin haritası üzerinden tanımlamaları gerçekleştirmektedir.

4.2. Akış Diyagramı

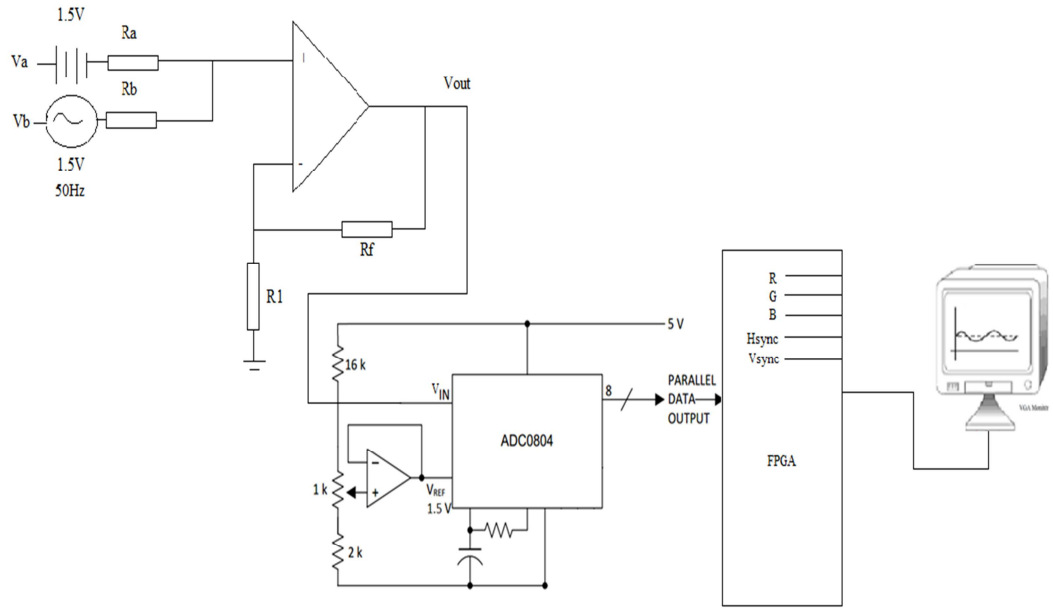
Akış diyagramında bizim FPGA ile VHDL arasında yaptığımız uygulamanın doğruluğunu kontrol etmek için kullandığımız şema bunun sonucunda Quartus II editöründe VHDL kodu yazarak uygulamamızı gerçekleştirdik.



Şekil 4.2. Akış diyagramı

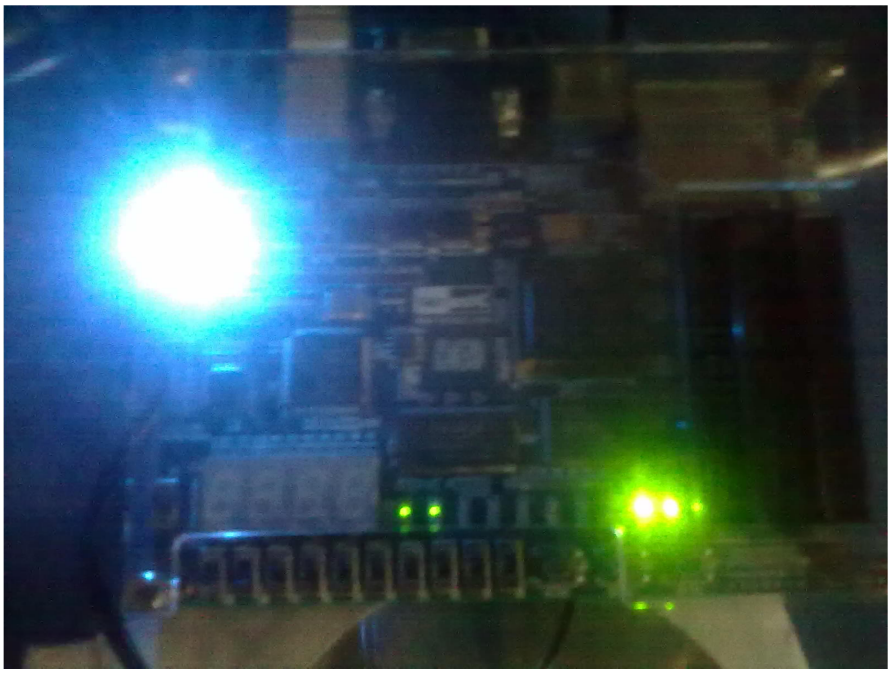
4.3. Elektronik Devre

Elektronik devrede bizim DE0 kitimizde ADC (Analog Dijital Çevirici) olmadığı için onu dışarıdan bizim devremize eklememiz gereken ekstradan bir entegre olmuştur ADC olmadan bizim devremizde istediğimiz dijital sinyali elde etmemiz mümkün olmadığında ekstradan ADC kullanmak zorundaydık. Elektronik devrede Fonksiyon jeneratörü, ADC, FPGA dışında iki tane direnç iki tane de kondansatör kullandık bunları kullanmamızın sebebi de istediğimiz sinüzoidal dalgayı elde etmemiz sağlamaktadır. Çünkü ADC ile elde edeceğimiz sonuç kare dalga olmaktadır. Bir de sinüzoidal dalga için gereken direnç ve kondansatörleri kullanarak bu sayede sinüzoidal dalgayı elde etmiş bulunmaktayız.



Şekil 4.3. Elektronik devre

4.4. Sinyal Okuma



Şekil 4.4. FPGA DE0 sinyal okuma

Sinyal okuma yukarıda şekilde örnek olarak verilmiştir. FPGA DE0 kitine istediğimiz bir bilgiyi gönderdiğimizde bizim USB Blaster bilgisayarımıza tanımlı olması gerekir bizim DE0 kitini bilgisayarımız USB blaster sayesinde DE0 kitini tanımaktadır ve biz Quartus II editöründe VHDL kodunu yazdığımızda onu Derleme (Compiler) aşamasında sorunsuz çalışması lazım ki bizim programı DE0 kitine sorunsuz yüklenmesi lazımdır. Bu aşamadan sonra bizim DE0 kitine yolladığımız kod okunur ve istediğimiz sinyali de bize göstermektedir ayrıca sinyalin okunup okunmadığını da kitin üzerindeki ışıklardan da anlamaktayız.

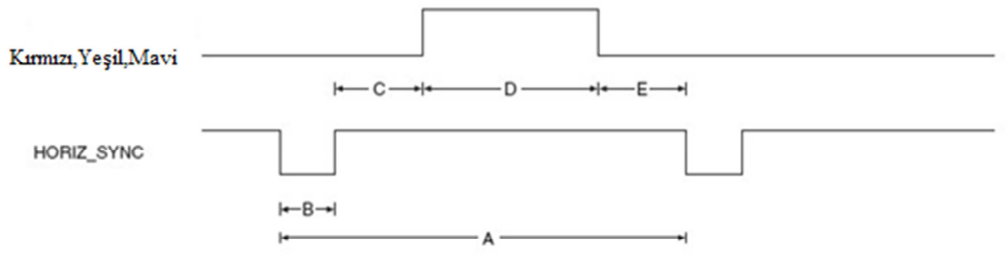
4.5. Görüntünün Elde Edilmesi

ALTERA DE0 kitindeki VGA monitör konektörü beş sinyalden oluşur. Kırmızı (4 bit), Yeşil (4 bit), Mavi (4 bit), Horiz_Sync ve Vert_Sync. Sinyaller arasında zamanlama ilişkileri Şekil 4.5'de görülmektedir.

VGA monitörde görüntüyü oluşturmadaki ızgara tarama için gerekli sinyallerin üretilmesi ALTERA DE0 kitindeki 50 MHz'lik saat frekansı, 25 MHz piksel saat

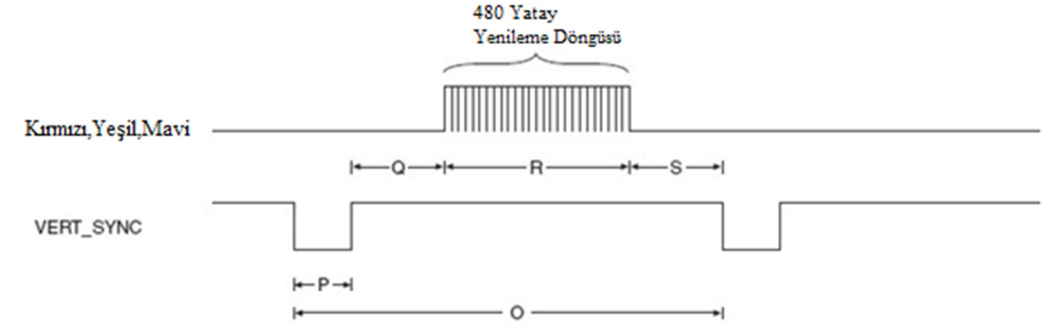
sinyaline sahip olan yatay ve dikey senkronizasyon frekanslarına bölünür. Bu piksel frekanslarını hesaplayabilmek için yatay ve düşey sayaçlara ihtiyaç vardır. Bu tasarım yatay ve dikey sayaç içeren anlamına gelir. İlgili konuma denk gelen yatay ve dikey senkronizasyon darbe sinyalleri bu sayaçlardaki değerlerden üretilebilir.

Bir video görüntüsü yatayda 640 piksel ve düşeyde de 480 pikselden oluşur. Monitör, X-Y düzleminin merkezi (0,0) olarak kabul edilen ekranın sol üst köşesindeki pikseli güncelleyerek yenileme döngüsü başlatır. İlk piksel yenilendikten sonra, monitör satırda kalan pikselleri yeniler. Monitör, HORIZ_SYNC pininde bir darbe aldığında, piksel sonraki satırı yeniler. Yatay süpürme için gerekli olan süre (süpürme periyodu) nominal olarak 31.77 mikrosaniyedir. Bu işlem ekranın en alt satırına ulaşıncaya kadar tekrarlanır. Ekranın alt kısmına ulaştığında monitörün VERT_SYNC pinine 64 mikrosaniyelik bir darbe, monitörün tekrar başlangıç noktasına gelmesi uygulanır. Şekil 4.6'de gösterildiği gibi, düşey tarama periyodunda, VERT_SYNC darbesi her 16.6 msn'de tekrarlanmalıdır. Bütün bir ekran görüntüsü, elektron ışınlarının izi ile 60 Hz'lik bir çerçeve hızı için 16.6 msn'de tamamlanacaktır. [19].



Parametre	A	B	C	D	E
Zaman	31.77 µs	3.77 µs	1.89 µs	25.17 µs	0.94 µs

Şekil 4.5. Yatay yenileme döngüsü



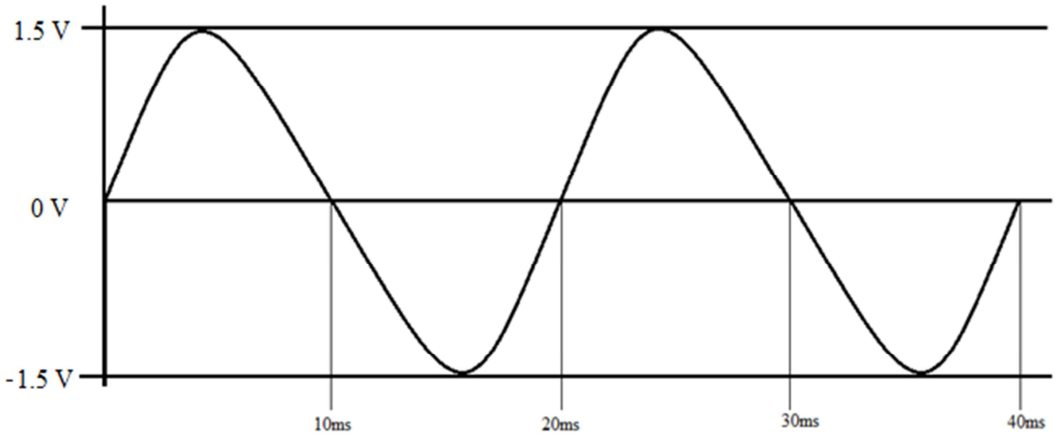
Parametre	O	P	Q	R	S
Zaman	16.6 ms	64 µs	1.02 ms	15.25 ms	0.35 ms

Şekil 4.6. Düşey yenileme döngüsü

4.6. Fonksiyon Jeneratörün Ayarları

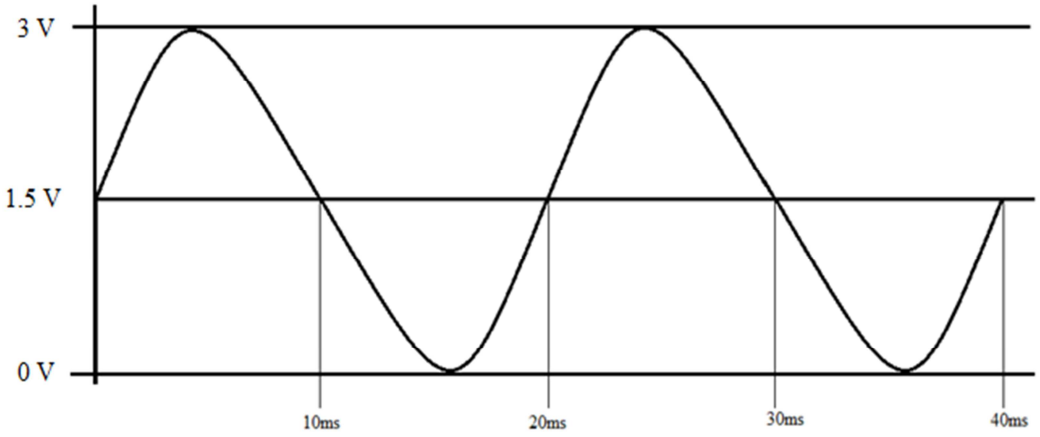
Projede, ADC'nin referans voltajı 3.0 V olarak alınmıştır. Dolayısıyla ölçülebilecek maksimum voltaj 3.0. V'tur. Referans olarak gösterilen değerden daha büyük voltaj söz konusuysa, bu durumda bu değer trafo ile dönüştürülmeli veya gerilim bölücü dirençlerle bölünerek düşürülmelidir. Bu projeyi test ederken bilinen bir sinyali üretmek için fonksiyon jeneratörü kullanılmış ve değerleri aşağıda verilmiştir:

Sinyal tipi : Sinüzoidal dalga
Genlik : 1.5 Volt
Frekans : 50 Hz



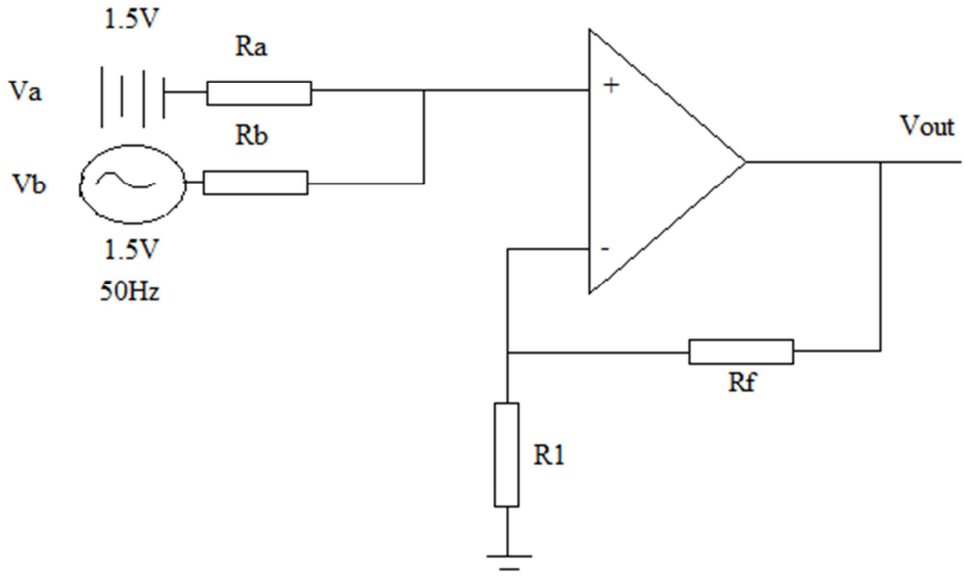
Şekil 4.7. Sinüzoidal dalga

Sinüs dalgasının bir çevriminde hem negatif, hem de pozitif kutupları vardır. Dolayısıyla maksimum değer + 1,5 V iken, minimum değer de – 1,5 V'tur. Hem negatif, hem de pozitif değerleri ölçmek için DC bir gerilim toplama devresi tasarlanmış ve sinüs dalgası genliği (-1.5 V, +1.5 V) aralığından, (0 V, +3 V) aralığına Şekil 4.7'da görüldüğü gibi kaydırılmıştır.

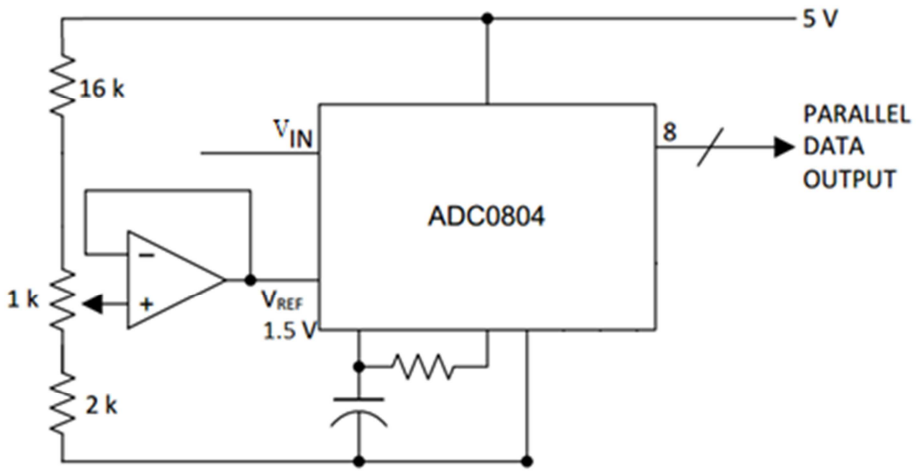


Şekil 4.8. Kaydırıcı sinüzoidal dalga

Şekil 4.8.'de, evirmeyen voltaj toplayıcı devresi görülmektedir. Devrede, Ra=Rb ve Rf=R1 kabul edildiğinde çıkış voltaj Vout=Va+Vb olmaktadır.



Şekil 4.9. Evirilmeyen gerilim toplayıcı



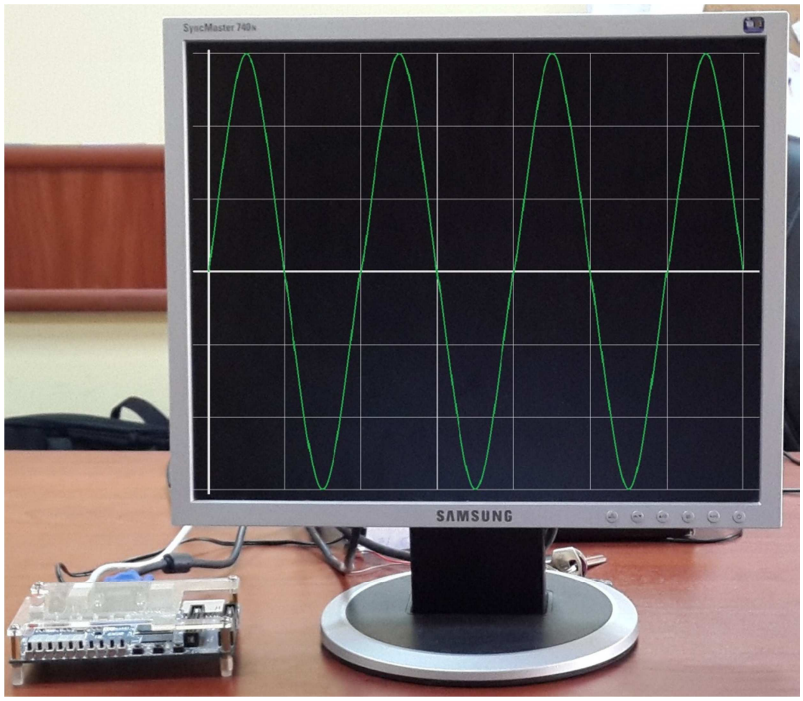
Şekil 4.10. ADC0804

Voltaj toplayıcı devresinin çıkışı ADC'nin girişine bağlanmıştır ve ADC bu analog değeri dijital değere dönüştürmektedir. ADC'nin çözünürlüğü 8 bit'tir ve 0-3 V'tu, 0-255 arası dijital duruma dönüştürür. Bu tasarıma göre -1,5 V'luk gerilim 0 durumu ile gösterilirken, +1,5 V'ta 255 durumu ile gösterilmektedir. Ölçülmek istenen voltajı hesaplamak için kullanılan denklem. (Denklem- 1)

V= (ADC_Output_Value*3/256)-1,5'dir **(1.1)**

640x480 çözünürlüğe sahip olan VGA monitör, elde edilen sinüzoidal dalgayı ekranda gösterir. Yatay eksendeki piksel sayısı 640 iken, dikey eksendeki piksel sayısı da 480'dir. Tasarıma göre yatay eksendeki her piksel 1 msn'yi, düşey eksendeki her piksel de 10 mV'u temsil etmektedir.

Bu bağlamda kullandığımız Evirilmeyen gerilim toplayıcı, ADC, Kit DE0 ve VGA 640x480 monitör elementler sayesinde gerçekleme işlemini ekrana yansıtmış bulunuyoruz. Aşağıda elde etiğimiz görüntü sayesinde sinüzoidal dalga şeklini elde etmiş bulunuyoruz.



Şekil 4.11. VGA 640x480 Sinüzoidal dalga çıktısı

Yukarıda ekrana yansıyan görüntüde evirilmeyen gerilim toplayıcı devresini kullanarak 3V aralığında yani +1.5V ve -1.5V arasında ki bunlar sinüzoidal dalganın tepe noktaları oluyor ve bu sayede (ekranda yansıyan görüntüsü ile +150pix ile -150pix aralığında) sinüzoidal dalga elde etmiş bulunuyoruz. FPGA DE0 işlem yaparken piksel bazında işlem yapmaktadır ve ekrana yansıttığı her görüntüyü pikselleri hesaplayarak yansıtmaktadır. Bizim bu görüntüyü piksel olarak görmemiz imkânsızdır çünkü FPGA'lar çok hızlı işlem yaptıkları için bizim ekrana net görüntü gelmektedir.

BÖLÜM 5. SONUÇLAR ve ÖNERİLER

Bu çalışmada FPGA DE0 ile sayısal osiloskop gerçeklemesi yapılmıştır. Maliyetleri oldukça düşük olan bu kitler ile eğitim amaçlı ya da profesyonel amacı olmayan osiloskoplar gerçeklenmiştir. Bu projenin yapımında bilinen bir sinyal olan sinüs dalgasını fonksiyon jeneratöründen elde edilmiştir. Elimizdeki geliştirme kitinin ADC modülü olmadığı için, ADC modülü dışarıdan bağlanmıştır. Sinüs dalgası hem pozitif, hem de negatif kutuba sahip olduğundan fonksiyon jeneratörünün çıkışı evirmeyen kaydırıcı devre ile pozitif kutuba kaydırılmıştır. ADC çıkışı DE0 kitine bağlanmış ve VHDL ile yazılarak sentezlenen kod FPGA'ya yüklenerek giriş sinyali DE0'nın VGA portuna bağlı monitörde izlenmiştir.

Çalışmada giriş sinyali olarak sinüs sinyali test edilmiş olsa da uygun sinyal uyumlaştırma devresi tasarlanarak tüm sinyaller okunabilir. Ayrıca okunan tüm değerler DE0 kitinde bulunan bellek modülündeki hafıza kartına kaydedilerek veriler yedeklenebilir.

Harici ADC modülü kullanmak yerine, dahili ADC modülüne sahip üst model kitler kullanılabilir. Bu sayede düşük frekanstaki sinyaller yerine daha yüksek frekanstaki sinyaller de izlenebilir.

KAYNAKLAR

[1] MEB., Doğru Akım ve Alternatif Akım Devreleri. Milli Eğitim Bakanlığı-Bilişim Teknolojileri, Ankara, 98-99, 2011.

[2] TAŞCI, M., FPGA Kontrollü Robotik Göz, Balıkesir Üniversitesi Fen Bilimleri Enstitüsü-Elektrik Elektronik Mühendisliği Anabilim Dalı, yüksek lisans tezi, Balıkesir, 23-40, 2011.

[3] ÖRENCİK B., FPGA Mimarisi Bilgisayar Mimarisinde Yeni Yaklaşımlar, İ.T.Ü. Bilişim Enstitüsü Bilgisayar Bilimleri, İstanbul, 2011.

[4] UZUN, S., CANAL, M-R., KAÇAR, M-C., VHDL Programlama Dili ve Sayısal Elektronik Devrelerin FPGA Tabanlı Uygulaması, 6th International Advanced Technologies Symposium, (IATS'11), Elazığ, 1-99, 2011.

[5] MANZAK, A., AYDIN, A., FPGA Yonga Mimarisi ve Kullanımı, Mühendislik Mimarlık Fakültesi Elektronik ve Haberleşme Mühendisliği Bölümü, Süleyman Demirel Üniversitesi, Isparta, 2005.

[6] KÜÇÜKGÜZEL. E., FPGA (Field Programmable Gate Array)'lere Giriş, Kısa Tarihçe ve Örnek Uygulama, İstanbul, 7-9, 2005.

[7] PARALI, A., TAŞKIN, S., PINAR, M., FPGA Donanımı İle Görsel Tabanlı Ölçme Sistemi, 5. Uluslararası İleri Teknolojiler Sempozyumu (IATS'09), Karabük, 1-2, 2009.

[8] SELİM, E., ULUCAN, A., FPGA Donanımı ve FPGA Üzerinde İşlemci Tasarımı: İvme – Momentum, FPGA Topluluğu-Ege Üniversitesi, İzmir, 6, 2008.

[9] KOÇBERBER, Y-O., FPGA ile Kablosuz Görüntü Aktarımı, Seminer, 4-6, 2009.

[10] BAŞAK, S., FGPA ile Gömülü Sistem Tasarımına Giriş, Bilgisayar Mühendisliği Bölümü, Yıldız Teknik Üniversitesi, İstanbul, 2-6, 2008.

[11] SEDEF, H., SADIÇ, F., Mikrodenetleyici Dijital Osiloskop ve Bode Diyagramı Çizici, Yıldız Teknik Üniversitesi Elektrik-Elektronik Fakültesi-Elektronik ve Haberleşme Mühendisliği Bölümü, Bitirme Projesi, İstanbul, 8-10, 2009.

[12] FELIP, VICEDO, F., CANET, M-J., GARCIA, F., GREDIAGA, A., ALCAREZ, S., GARRIDO, P., LLAMAZARES, K-G., Low cost Digital Oscilloscope, Universitat Politecnica de Valencia Ctra. Nazaret s/n 46730, Dpto. Fis y Arq. de Computadores Universidad Miguel Hernandez Avda. Ferrocarril s/n 03202, Tecnolog´ıa Informatica y Computacio´n Universidad de Alicante Alicante, Elche/ Gandia, Valencia, Spain, 1-4.

[13] YAKUT, M., KALE, A., FPGA ile PCI Kart Tasarımı, Elektronik ve Haberleşme Mühendisliği, Lisans Bitirme Tezi, Kocaeli, 76-91, 2007.

[14] RAMEZANI, SH., YALÇIN, B., Bilgisayar Aritmetik Algoritmalarının FPGA Üzerinde Gerçeklenmesi, İstanbul Teknik Üniversitesi Elektrik – Elektronik Fakültesi, Bitirme Ödevi, İstanbul, 3-5, 2011.

[15] DOĞAN, A-Y., YALÇIN, B-Ö., SFINKS Dizi Şifreleme Algoritmasının VHDL İle Yazılımı ve FPGA Üzerinde Gerçeklenmesi, Elektronik Mühendisliği, İstanbul Teknik Üniversitesi, İstanbul, 14-15, 2006.

[16] TANIŞ, G., Açıklamalı Uygulama Örnekleri Bilgisayar-Bilgisayara Giriş MS-DOS, PCTOOLS, PW, BASIC, Antalya, 33-35, 1994.

[17] Emmungil, L., Bilgisayar Donanımı, Ankara, 2008.

[18] http://www.frmtr.com, Erişim Tarihi, 22.04.2013.

[19] Altera University program, DE0 user manual-Development and education board, 1-7, 2012.

[20] GÜDER, H., ÇETİNKAYA, A., ŞAHİN, O., ÇAKMAK, M., KAHRAMAN, S., Fizik Laboratuarı II (Elektrik ve Manyetizma) Deney Kılavuzu, Mustafa Kemal Ünversitesi Fen-Edebiyat Fakültesi, Fizik Bölümü, Hatay, 9, 2008.

ÖZGEÇMİŞ

1982 yılında Kosova Prizren şehrinde doğan Berkant BAŞA, İlkokul, Ortaokul eğitimlerini Prizren'de tamamladı. Devamında lise eğitimini Prizren I. Teknik lisesinde okudu ve 1999 yılında mezun olduktan sonra 2000 yılında lisans eğitimi için Türkiye'ye geldi. Bir yıl Ankara'da TÖMER'de (Türkçe Öğrenim Merkezinde) Türkçe eğitimi aldıktan sonra, Konya Selçuk Üniversitesi Bilgisayar Mühendisliği bölümünü kazandı. Burada da bir yıl İngilizce eğitimi aldıktan sonra 2002 yılında Bilgisayar Mühendisliği Bölümünde okumaya başladı ve 2009 yılında mezun oldu. 2011 yılında Sakarya Üniversitesi Bilgisayar ve Bilişim Mühendisliğinde yüksek lisans eğitim programına başladı.