

FPGA (Field Programmable Gate Array) lere Giriş, Kısa Tarihçe ve Örnek Uygulama

Erhan Küçükğzel - 2005

İçindekiler :

1. Giriş.....	3
2. FPGA ve Benzer Yapılar.....	3
3. Kısa FPGA Tarihçesi.....	7
4. Kendi İçinde Farklı FPGA Mimarileri.....	9
5. ACTEL ProASICPLUS FPGA lar.....	10
6. Donanım Tanımlama Dilleri.....	11
7. VHDL Programlama Dili.....	12
8. Örnek Devre Gerçekleme.....	14
9. ACTEL Libero Ortamı.....	18
10. Sonuç ve Yorumlar.....	19
Kaynakça :.....	19

1. Giriş

Bu raporda FPGA ler daha yakından incelenmiş, gelecek hakkındaki değerlendirmelere yer verilmiştir. Bu konunun irdelenmesindeki gerekçe, projenin geleceğinde *hem teknik hem de ticari* yönden FPGA ortamının gerçekçi ve sağlam bir yol olduğunu göstermektir. Bir başka gerekçe de ilerlenecek yol olan FPGA lerin daha yakından tanınması gerekliliğidir. Raporun başlarında önemli olduğunu düşündüğüm kısa tarihçeden başlayarak, geleceğe dair tahminlere kadar uzanan özet bilgiler sunmaya çalıştım. Daha sonra teknik anlamda FPGA lerin benzer diğer yapılardan farklarına değindikten sonra basit bir devreyi ACTEL geliştirme ortamında gerçekleyerek tasarım ve gerçekleştirme sürecine bir örnek sundum. Çalışmalarımızda FPGA dünyasına bu ilk adımın çok önemli olduğunu düşünmekteyim.

2. FPGA ve Benzer Yapılar

Sahada Programlanabilir Kapı Dizileri (SPKD, ing : FPGA) programlanabilen lojik (mantık) devre blokları ve ayarlanabilir ara bağlantıları içeren sayısal tümdevrelerdir. “Sahada” söylemi, programlamanın üreticide değil müşteride yani sahada yapılmasındandır. Programlanabilir cihazlarda zaman içinde ortaya çıkan yapılar :

- PLD (Programmable Logic Device),
- ASIC (Application-Specific Integrated Circuit),
- ASSP (Application-Specific Standard Parts)
- FPGA (Field Programmable Gate Arrays)

PLD

Mimarileri üretici tarafından belirlenmiş ancak gene de üzerinde farklı fonksiyonlar gerçekleştirilebilecek yapılardır (Yani temel yapı FPGA den karışık). FPGA lere göre daha az sayıda lojik devre kapısı içerirler, daha basit ve küçük işlevler yüklenebilirler.

ASIC ve ASSP

ASIC ve ASSP devreler yüzlerce milyon kapı içerebilen ve oldukça karmaşık işlevleri yerine getirebilen tümdevrelerdir. Aynı yapıdadırlar ve aynı teknoloji ile üretilirler. Tek farkları, ASIC üretici tarafından tek bir müşteriye üretilirken, ASSP ler birden çok müşteri için hazırlanırlar.

FPGA lar bu anlamda PLD ler ve ASIC ler arasında bir yerde düşünülebilir. İşlevleri PLD ler gibi özelleştirilebilirken, içerebildikleri milyonlarca kapı kapasitesi ile daha önceleri sadece ASIC tasarımlarla gerçekleştirilebilen karmaşık yapıları mümkün kılarlar. Öte yandan ASIC ile karşılaştırıldığında tasarım süreci çok daha “kolay, hızlı ve ucuz” dur. ASIC tasarımlar sadece büyük şirketlerin göze alabilecekleri derece pahalıdır. FPGA lar sayesinde birçok küçük girişimci şirket ayakta kalabilmiştir.

Bu avantajları ile 2003 yılında tahminen 1'500-4'000 arası ASIC projesi ve 5'000 ASSP projesi başlarken, tahmini 450'000 FPGA tasarımı başlamıştır.

FPGA lar ve Kullanım Alanları

- | | |
|-------------|---|
| 1980 ler | bağlantı lojik devreleri (glue-logic)
orta karmaşıklıkta durum makineleri
az miktarda veri işleme |
| 1990 lar | haberleşme devreleri
haberleşme ağları |
| 90 lar sonu | tüketici, otomotiv, endüstri |

Bir kullanım alanı da ASIC prototipi üretimidir.

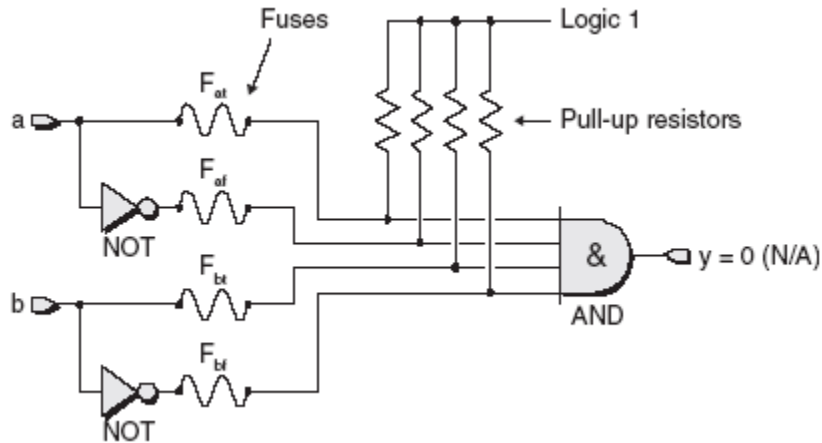
Artık 2000 lerdeyiz. Bazılarına gömülü işlemci çekirdekleri yükleniyor. Bugün kısacası her sayısal işlevi FPGA lara yükleyebilirsiniz. Yapı SoC (System on Chip=Tümdevre Dahili Sistem) denen kavrama oturmuştur. Sonuç olarak FPGA şu dört pazarda alternatiflerinin yerine göz dikmiştir :

- ASIC ve özel yapım tümdevre
- DSP
- Gömülü Mikrodenetleyici uygulamaları
- Fiziksel katmanda yer alan haberleşme tümdevreleri

Yeni bir gelişme ile örnek vermek gerekirse; ACTEL firması geçtiğimiz aylarda, oldukça popüler olan 32 bitlik ARM işlemcisinin haklarını elinde bulunduran firmayla bir anlaşmaya varmıştır. ACTEL, işlemciyi kendi FPGA ailelerinde gerçeklemek isteyen müşterilerine ARM işlemciyi gerçekleyecek kodları (Software Intellectual Property = Akıl ürünü yazılım hakkı) ücretsiz bir lisans ile vereceğini açıklamıştır. Böylece ARM işlemcide tecrübesi ve hazır yazılımı bulunan tasarımcılar, tasarımlarını koruyarak FPGA üzerine transfer edebilecekler, yeni becerilerle daha da tümleştirilmiş hale getirebileceklerdir. Bu, ARM işlemci üreticileri için kötü bir haber olarak yorumlanabilir.

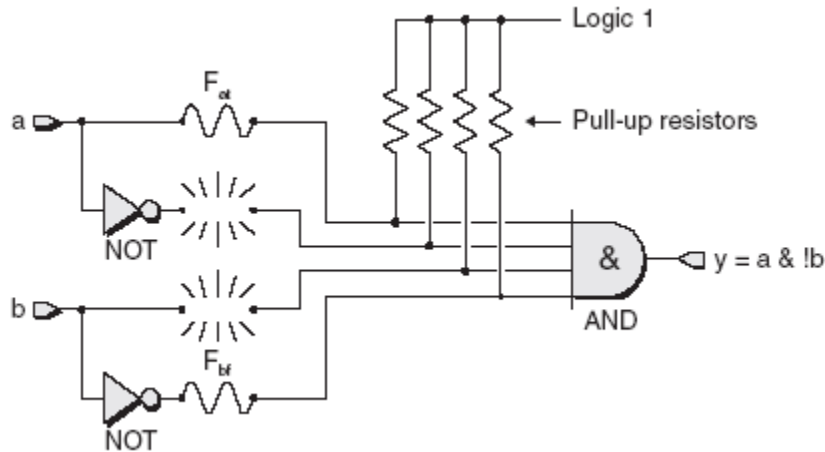
Temel Kavramlar

Programlanabilir devre kavramına basit bir örnek verelim. Şekilde görülen sigortalar (fuses) seçeneğe bağlı olarak yakılabilir olsun. Tümü kısadevrede iken VE kapısı çıkışı "0" olacaktır.



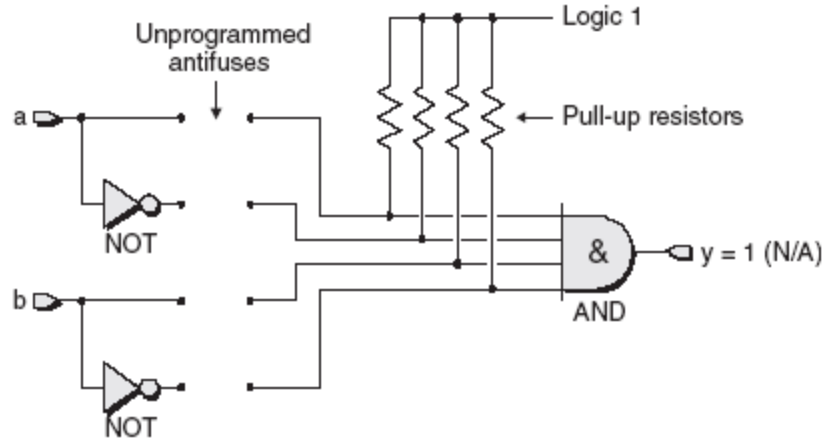
Şekil 2.1

Bu devrede sigortalardan dilediğimizi yakarak (açıkdevre yaparak) $y = \bar{a} \wedge b$ işlevini gerçekleştirebiliriz



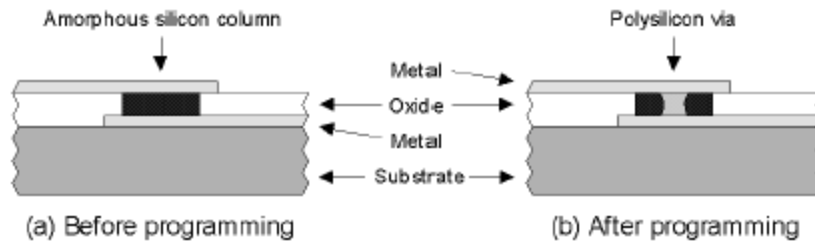
Şekil 2.2

Bu türden programlanabilir cihazların sigortalı teknoloji (fusible technology) kullandığı söylenir. Bir alternatif yaklaşım ise Sigorta Karşıtı (Anti-Fuse) teknolojidir. Bu tür bir cihazda programlanmamış cihaz aşağıdaki gibidir :



Şekil 2.3

Programlanmamış bir bağlantı (sigorta karşıtı veya yol diyelim) milyon ohm lara varan bir yalıtımda amorf silisyum yapıdır :

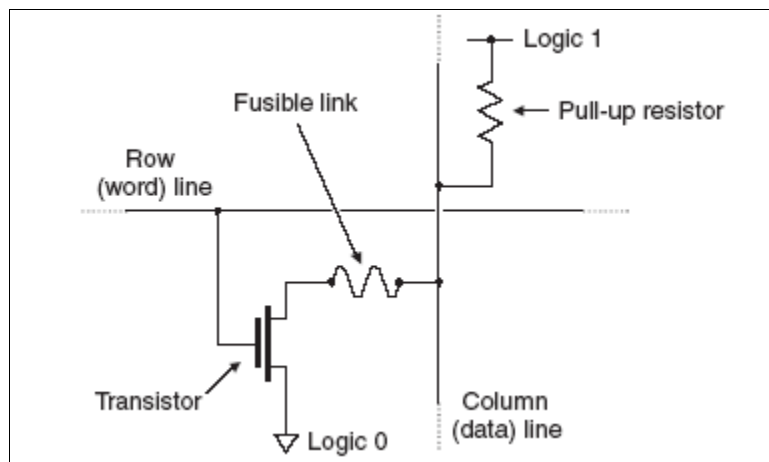


Şekil 2.4

Programlandığında yol, yalıtıcı amorf silisyumdan, ileten polisilisyuma dönüşür.

PROM

Programlanabilir cihazların bir türü de belleklerdir. PROM (Programmable Read Only Memory) adres hatları ile kesişen veri hatları içerir :



Şekil 2.5

Bellekler, seçilen adresteki veriyi çıkışa verirler. Adres seçimi şekildeki gibi, seçilen adrese ait, veriyoluna bağlı transistörü ilettime geçirmektir. Eğer sigorta varsa (kısadevre) çıkış "0"

olacaktır. Adresin çıkışı “1” olması istendiyse sigorta yakılmıştır. Sigortalı teknolojiye sahip cihazlar bu yüzden programlanmadığında “0” çıkış verirler. Günümüzde çoğu bellek türü sigorta karşıtı teknoloji ile üretildiğinden, programlanmamış cihazların tüm adresleri “1” ile doludur (8-bit belleklerde ilk değerler 11111111=255=FF h). PROM lar beslemesine oranla daha yüksek gerilim ile kalıcı olarak programlanırlar. İlk PROM lar 1970 te Harris Semiconductor tarafından üretildi.

EPROM

1971 de Intel, PROM'ların silinebilir türlerini üretti : EPROM (Erasable PROM). Bellek hücrelerinde gerilimle oluşturulan (yani programlama) yükler, yüzen bir kapı oluşturur (transistöre bağlı yol). Bu yol bellek gözünün “0” çıkış vermesini sağlar ve kendi başına 10 yıl civarı bir zaman silinmez. Bu yükü boşaltmak, belleği silmek, yani EPROM u ilk haline getirmek demektir. Silmek için gerekli enerji EPROM larda dışarıdan verilen UV ışık ile sağlanır (5-20 dakika).

E²PROM

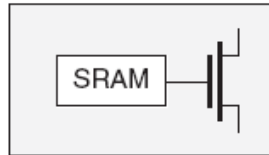
Daha sonra ortaya EEPROM (Electrically Erasable PROM, E²PROM) çıktı. Artık silme işlevi UV ile değil, bellek gözündeki transistöre bağlı bir başka transistöre verilmişti, bu da elektriksel olarak silmek demektir. EEPROM da tek tek istenen bellek gözü bu sayede silinebilir. E²PROM belleklerin ortaya çıkmasıyla, tekrar programlanabilir PLD ler de üretildi (EEPROM veya E²PLD).

FLASH

EEPROM bellekleri FLASH bellekleri takip etti. FLASH ismi, bu cihazların nispeten çok daha hızlı silinebilmesine dayanarak verilmişti. E²PROM dan hızlı silinebilmesi yanında, aynı anda ya tüm cihaz ya da büyük bloklar silinebilir, tek tek silme mümkün değildir.

SRAM

Static bellekler, sınırsızca tekrar yazılabilirliği avantajı yanında, uçucu belleklerdir.



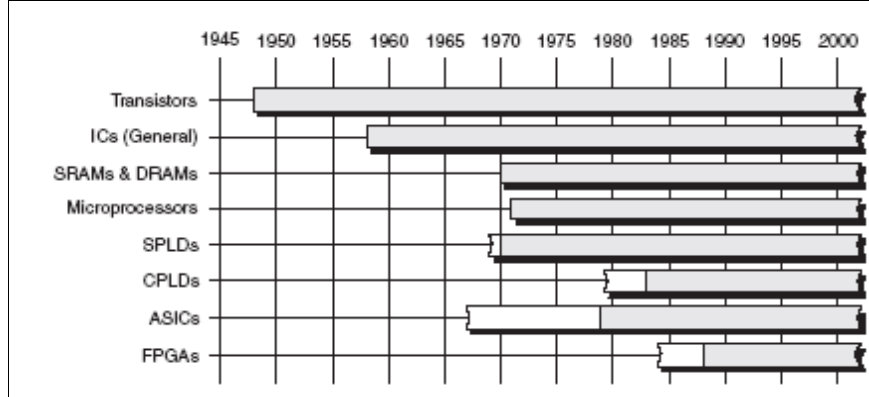
Şekil 2.6 : SRAM Tabanlı bir programlanabilir hücre

Technology	Symbol	Predominantly associated with ...
Fusible-link		SPLDs
Antifuse		FPGAs
EPROM		SPLDs and CPLDs
E ² PROM/ FLASH		SPLDs and CPLDs (some FPGAs)
SRAM		FPGAs (some CPLDs)

Tablo 2.1 : Bahsedilen teknolojiler ve ağırlıklı kullanım alanları

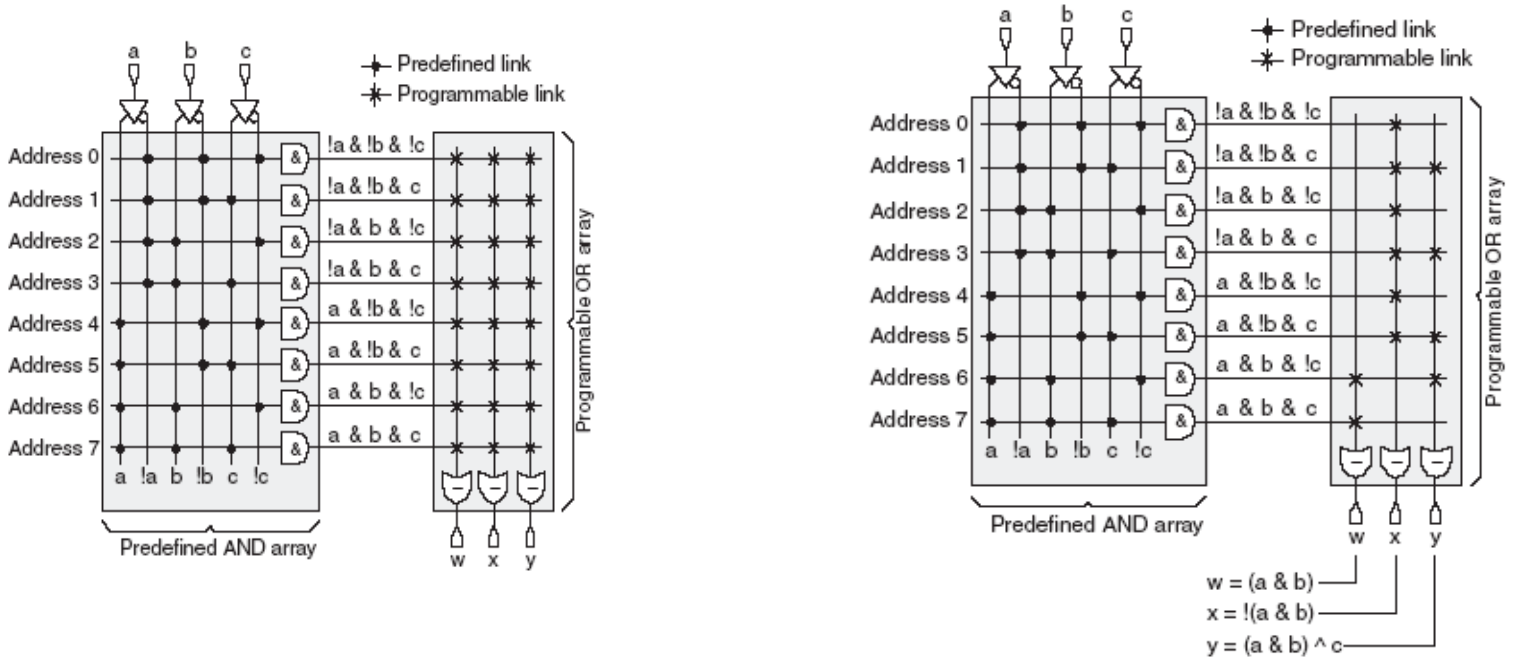
3. Kısa FPGA Tarihi

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 başlarına kadar FPGA lara rağbet etmediler.



Şekil 3.1 : Farklı teknolojilerin ortaya çıkışları

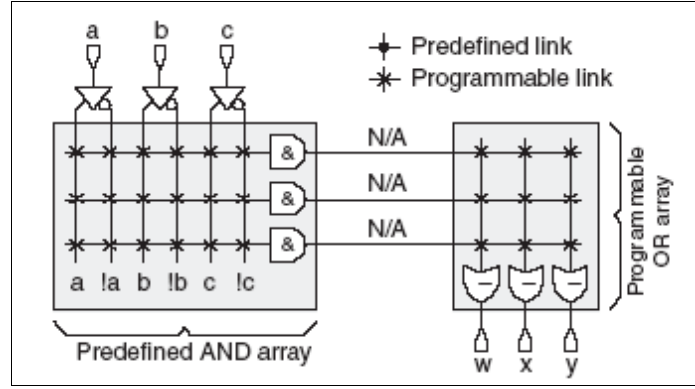
İlk PLD ler basit PROM lardı.



Şekil 3.2

Bu yapıda VE dizileri üretimden sabit olup, VEYA dizilerindeki bağlantılar kullanıcı tanımlıdır.

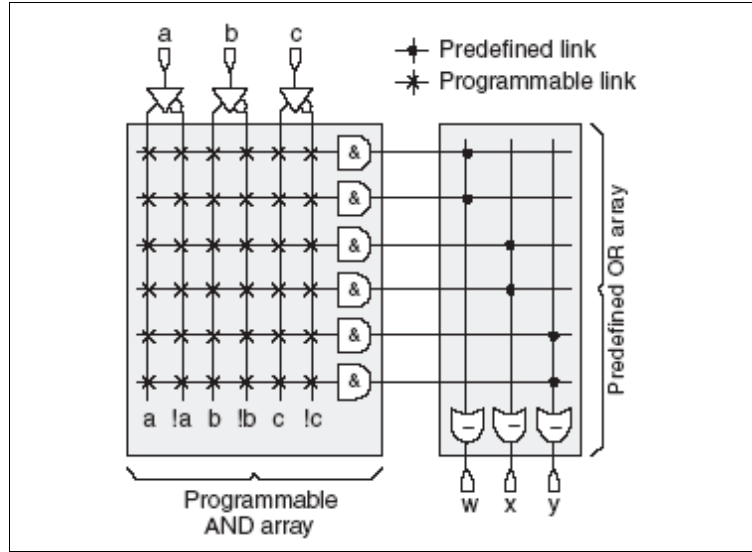
Bir sonraki büyük adım PLA'nın ortaya çıkmasıydı :



Şekil 3.3 : PLA yapısı

Görüldüğü gibi PROMdan farklı olarak, VE dizisindeki bağlantılar da PLA larda programlanabilir bağlantılardır.

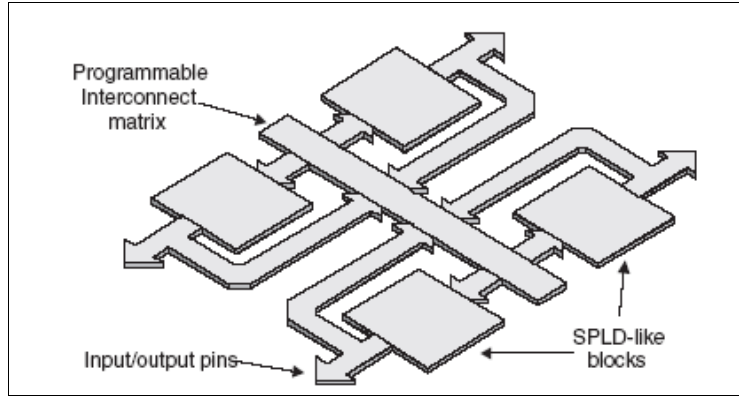
İki alanın da programlanabilir oluşu hız problemi ortaya çıkardı. Bunun üzerine ortaya çıkan PAL yapılar, PROM ların tersi yapıdaydılar ve VE dizisi tarafı programlanabilir olduğundan PROM lardan daha esnektiler.



Şekil 3.4 : 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. Arayollar 100 civarı kablo taşıyabilir.

1980 lerde sayısal tümdevre 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 tümdevre üzerinde değişiklik yapılamamakta idi. 1984 yılında XILINX bu boşluğu dolduracak cihazı piyasaya sürdü.



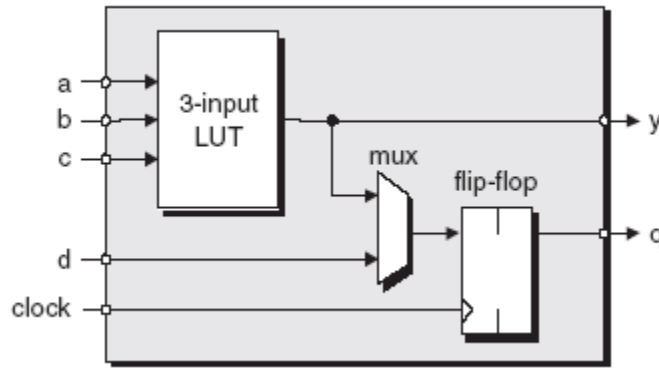
Şekil 3.5 : Yaygın CPLD yapısı

4. Kendi İçinde Farklı FPGA Mimarileri

FPGA lar yapılarına göre başlıca iki gruba ayrılabilir :

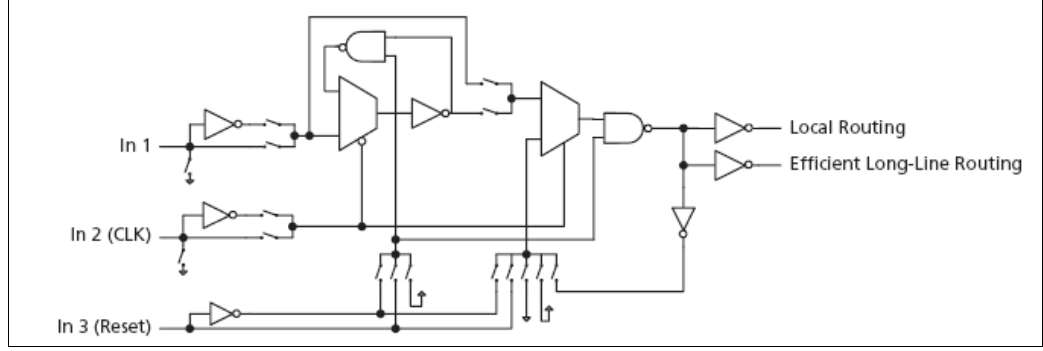
- Doğruluk Tablosu (LUT = Look-Up Table) Tabanlı Yapı
- Çoklayıcı (MUX) Tabanlı Yapı

LUT Tabanlı Yapının temel birimi LUT adı verilen, ve m ($m \geq 2$) değişkenli her Boole işlevini gerçekleyebilen devredir. Verilen bir LUT yapısı için m genelde 3 ile 6 arasında değişen sabit bir sayıdır. Genelde bu yapı, m tane adres 1 tane veriyolu olan statik RAM ile gerçekleştirilir ve kısaca m -LUT olarak adlandırılır. LUT-tabanlı yapıda her temel birim bir ya da daha fazla LUT ile flip-flop gibi diğer lojik elemanlardan oluşur.



Şekil 4.1 : Basit bir LUT-Tabanlı FPGA lojik birimi

MUX-tabanlı yapının temel birimi çoklayıcıların çeşitli kombinasyonlarından ve olabildiğince az VE ve VEYA gibi lojik kapılardan oluşur. Bu yapıdaki FPGA ların içinde veri tutucu ve flip-flop gibi bellek elemanları bulunmadığından çoklayıcılar ile bu elemanların gerçekleştirilmesi gerekir.



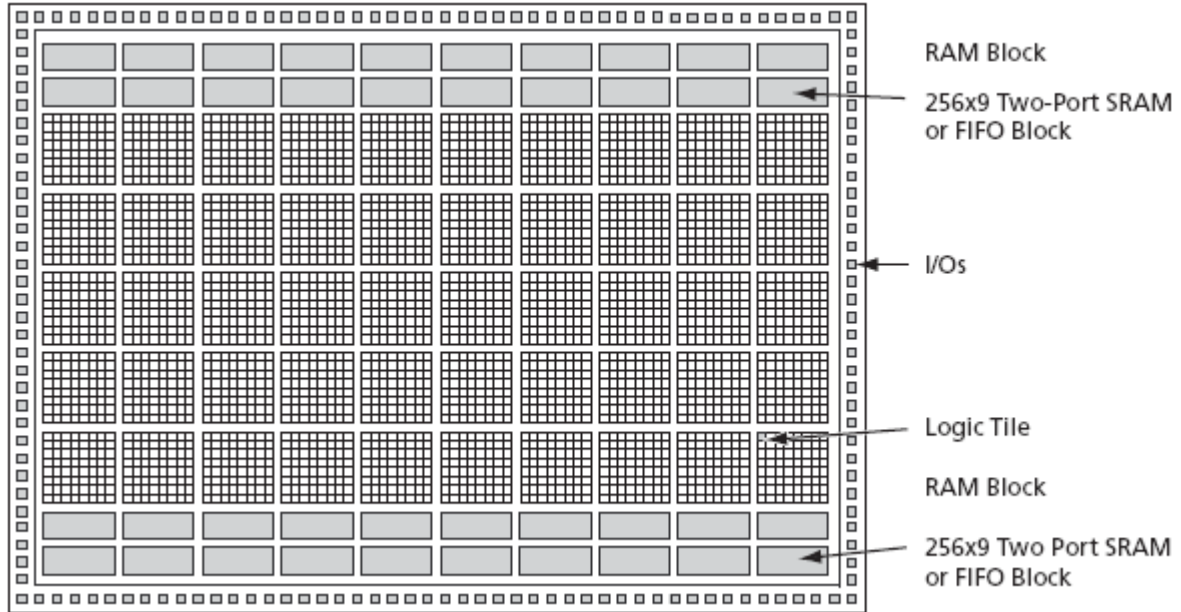
Şekil 4.2 : MUX-tabanlı ACTEL ProASIC PLUS Ailesi FPGA Lojik Birimi

Şekil 4.2 de görülen Lojik Birim (Logic Tile), tümü de evrilebilen üç giriş ve bir çıkışa sahiptir. Çıkış, yerel hızlı yola veya verimli uzun hat yol kaynaklarına bağlanabilir. 3 girişli XOR işlevi dışında tüm 3 girişli işlevler tek birimde gerçekleştirilebilir. Birim, SET veya RESET girişlerine sahip bir mandal (latch) veya, SET veya RESET girişlerine sahip bir FLIP-FLOP olarak ayarlanabilir. Böylece lojik birim, esnek biçimde tasarımların lojik ve ardışıl kapılarını oluşturabilir.

5. ACTEL ProASIC^{PLUS} FPGA lar

Actel'in bu ikinci nesil FLASH FPGA'ları ASIC devrelerin avantajlarını, uçucu olmayan FLASH belleği ile programlanabilir cihazlara taşır. Böylece tasarımcılar mevcut ASIC veya FPGA tasarımlarını kullanarak yüksek yoğunlukta sistemler oluşturabilirler.

ProASIC^{PLUS} ailesi dahili iki adet PLL devresi ile benzersiz bir saat üreticisine sahiptir. Ailede 1 milyona varan kapı eşdeğeri yoğunluk, 198 kbit'e varan çift portlu SRAM, 712 adete varan genel amaçlı G/Ç ile 50MHz PCI başarımına ulaşır. Dahili güvenlik mekanizması, yüklü programa erişimi engeller.



Şekil 5.1 ProASIC^{PLUS} Cihaz Mimarisi

FLASH sigortaları ile konfigürasyon bilgilerini içinde taşıyan cihaz, harici konfigürasyon belleği ve devresine ihtiyaç duymaz. Böylece ilk besleme anında canlıdır ve işlevseldir. Temel birimi, ince granüllü MUX yapıdadır. Böylece %100'e varan kaynak kullanımı sağlar. IEEE 1149.1 standartlarında JTAG portu ile cihaz sistem üzerindeyken tekrar programlama ve test imkanı sağlar.

Projede, ProASIC^{PLUS} ailesinden APA300 cihazını içeren APA-EVAL Kiti kullanılmaktadır.

6. Donanım Tanımlama Dilleri

80 lerin sonlarına doğru devrelerin boyutları ve karmaşıklığı oldukça arttı. Bunun üzerine sözcüğü 5000 kapılı bir devre için bir kaç top kağıtlık şema çizimi gerekiyordu. Elle çizim olmasa da, bilgisayarda bunu göstermek, test etmek, yakalamak ve benzetime (simulasyon) sokmak oldukça güç olmaya başladı. EDA (Elektronik Design Automation = Elektronik Tasarım Otomasyonu) üreticileri de bu durumda Donanım Tanımlama Dilleri'ni ortaya çıkardılar.

Tanımlama seviyeleri :

Genel olarak problem belirlendikten sonra sistem, birbirine bağlantıları tamamen belirli alt parçalara ayrılmalıdır. Bu yapılırken parçaların herbiri bir birim olarak düşünölmeli ve birimlerin arasındaki bağlantılar tamamen belirlenmelidir. Bu parçalama işlemine basit lojik kapılara ulaşınca dek devam edilir. Tanımlama genellikle şu seviyelerde yapılır : Sistem seviyesi, saklayıcı iletişim seviyesi (RTL=Register Transfer Level), lojik seviye ve devre seviyesi...

<i>Tanımlama Seviyesi</i>	<i>Tanımlama Yöntemleri</i>		
	Davranışsal Tanımlama	Yapısal Tanımlama	Fiziksel Tanımlama
<i>Sistem seviyesi</i>	Algoritmalar, akış diyagramları	İşlemciler, bellekler	Baskı devre kartları, tümdevreler
<i>RT Seviyesi</i>	Saklayıcı iletişimleri	Saklayıcılar, fonksiyon birimleri, veri toplayıcılar	Tümdevreler, modüller
<i>Lojik Seviye</i>	Boole fonksiyonları	Kapılar, flip flop lar	Tümdevreler, hücreler
<i>Devre Seviyesi</i>	Transfer fonksiyonları	Transistörler, bağlantılar	Hücreler, yol parçaları

Tablo 6.1 Farklı tanımlama seviyelerine ait elemanlar

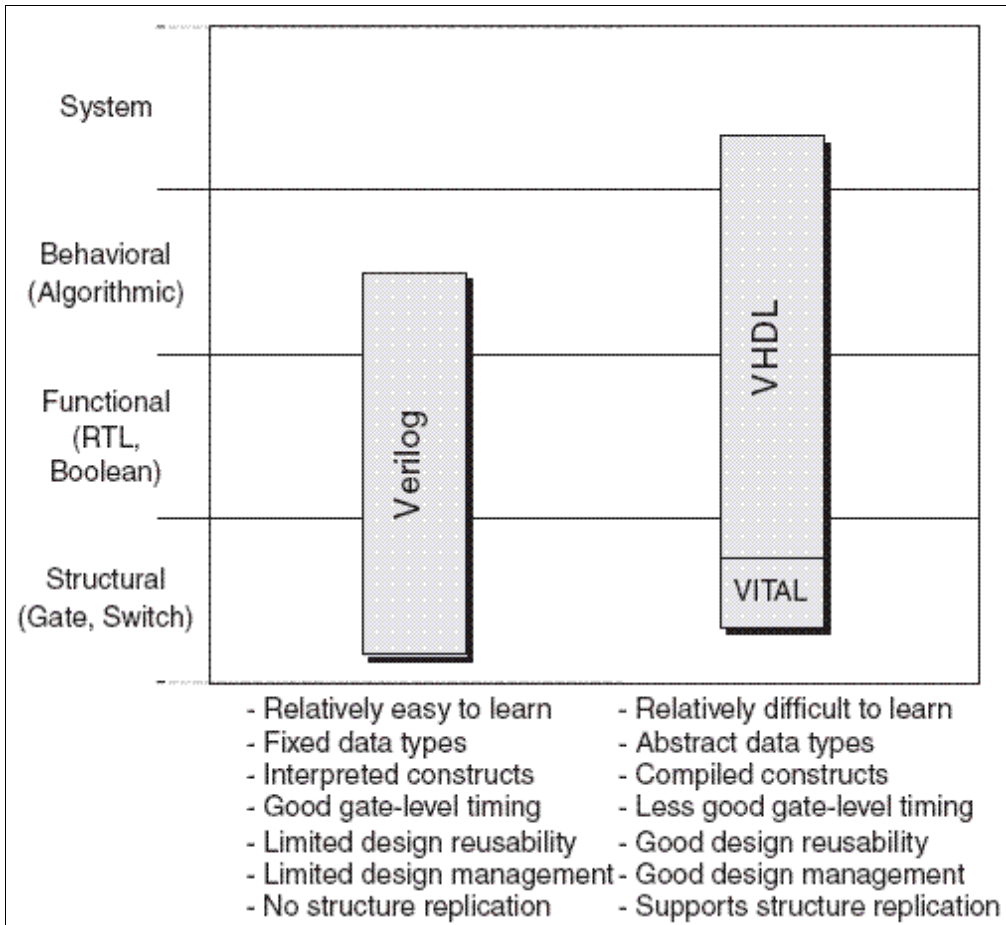
Tanımlama Yöntemleri :

Davranışsal tanımlamada, sistemin nasıl gerçekleşeceği gözönüne alınmaksızın, sistemin işlevi tanımlanır. Bunu için yüksek seviye programlama dillerine benzer ifadeler kullanılabilir. Bu tür tanımlamada kullanıcının da kapı seviyesinde bilgiye sahip olmasına gerek yoktur. Gerekli indirgemeyi derleyici halledecektir. Yapısal tanımlamada sistem, daha önceden tanımlanmış elemanların birleşimi olarak tanımlanır (biz de VHDL dilinde bu yönteme ağırlık vereceğiz). Fiziksel tanımlama tümleşik devrenin silikon üzerinde oluşturulması ile ilgilidir ve konumuz dışındadır.

7. VHDL Programlama Dili

VHDL adı Very High Speed Integrated Circuit (VHSIC) tümcesi ile Hardware Description Language (HDL) ifadesinin birleşmesinden oluşur (VHSIC=Çok Hızlı Tümleşik Devre, HDL=Donanım Tanımlama Dili). Verilog dili ile birlikte en yaygın iki HDL türünden birisidir. Projemizde gerektiğinde HDL olarak VHDL kullanılacaktır.

VHDL ve en büyük rakibi Verilog arasındaki işlevsel fark şekil 6.1 de görülmektedir. Görüldüğü gibi yapıların kopyalanarak çoğaltılmasında uzmanlar VHDL dilinin daha uygun kaçtığını belirtmektedirler. Bu da, tekrarlı yapıya sahip Hücresel Sinir Ağları projesi için VHDL dilinin tercih edilmesi için önemli bir sebeptir. Verilog dilinin öne çıkan yanı da daha kolay öğrenilmesi ve zamanlama konusunda daha hassas oluşu olarak belirtilmektedir.



Şekil 7.1 VHDL ve Verilog İşlevsel Karşılaştırma

VHDL ye giriş :

Şekil 7.2 de VHDL ile bir bitlik karşılaştırıcının farklı tanımlama şekillerine göre yazılmış kodları görülmektedir. Devre “**entity**” (sözlük anlamı “varlık”) de tanımlanan A ve B girişleri ile C çıkışına sahiptir. **entity** de yeni bir modül ismi, giriş/çıkışları tanımlanır. **entity** tanımı ile modül aynı bir kara kutu gibidir. Giriş çıkışlar bellidir ancak içerisi, yani işlevi tanımlanmamıştır. İşlev ise **architecture** (mimari) kısmında tanımlanır. Mimari ve işlevi tanımlamak için üç farklı yaklaşım vardır : Davranışsal (behavioral), veri akışı (dataflow) ve yapısal (structural) yaklaşımlar

- Davranışsal mimaride bir sistemin, bilgisayar programına benzer biçimde ne yaptığı tanımlanır. Nasıl gerçekleşeceği konusunda bilgi verilmez.
- Veri akışı türü mimaride veri işaretlerinin devre içindeki akışı tanımlanır. Bu işlem yapılırken toplayıcılar, karşılaştırıcılar, kod çözücüler gibi daha önceden tanımlanmış kombinezonsal devre fonksiyonları ve basit lojik kapılar kullanılır.
- Yapısal mimari ile, tasarımın alt sistemlerini oluşturan daha önce tanımlanmış modüllerin birbiri ile olan bağlantıları tanımlanır.

```
entity karsilastir is
    port (A,B:in STD_LOGIC ; C : out STD_LOGIC);
end karsilastir;
```

(a)

```
architecture davranis of karsilastir is
begin
    process (A,B)
    begin
        if (A=B) then
            C <= '1';
        else
            C <= '0';
        end if;
    end process;
end davranis;
```

(b)

```
architecture veriakis of karsilastir is
begin
    C <= not (A xor B) after 10 ns;
end veriakis;
```

(c)

```
architecture yapi of karsilastir is

component EXOR_kapi
    port (g0,g1:in STD_LOGIC; C:out STD_LOGIC );
end component;

component DEGIL_kapi
    port (g:in STD_LOGIC; C:out STD_LOGIC );
end component;

signal
    yol:STD_LOGIC;
begin
    K0: EXOR_kapi port map (A,B,yol);
    K1: DEGIL_kapi port map (yol,C);
end yapi;
```

(d)

Şekil 7.2 Karşılaştırıcı için (a) entity (b) Davranışsal mimari
(c) Veri Akış Mimarisi (d) Yapısal Mimari

8. Örnek Devre Gerçekleme

Örnek bir devre gerçeklemek adına tam toplayıcı sentezleyelim. Önce doğruluk tablosunu oluşturalım :

<i>A</i>	<i>B</i>	<i>Eg</i>	<i>T</i>	<i>Ec</i>
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Tablo 8.1 Tam toplayıcı doğruluk tablosu:
A,B : giriş bitleri, Eg : giriş eldesi; T : toplam, Ec : çıkış eldesi

Bu tablodan Karnaugh diyagramı yardımı ile indirgenmiş fonksiyonlarımız olan toplam (T) ve çıkış eldesini (Ec) bulalım :

<i>T=TOPLAM</i>	<i>B,Eg</i>			
<i>A</i>	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$$T = \bar{A} B E g + \bar{A} B \bar{E} g + A B \bar{E} g + A B E g$$

(a)

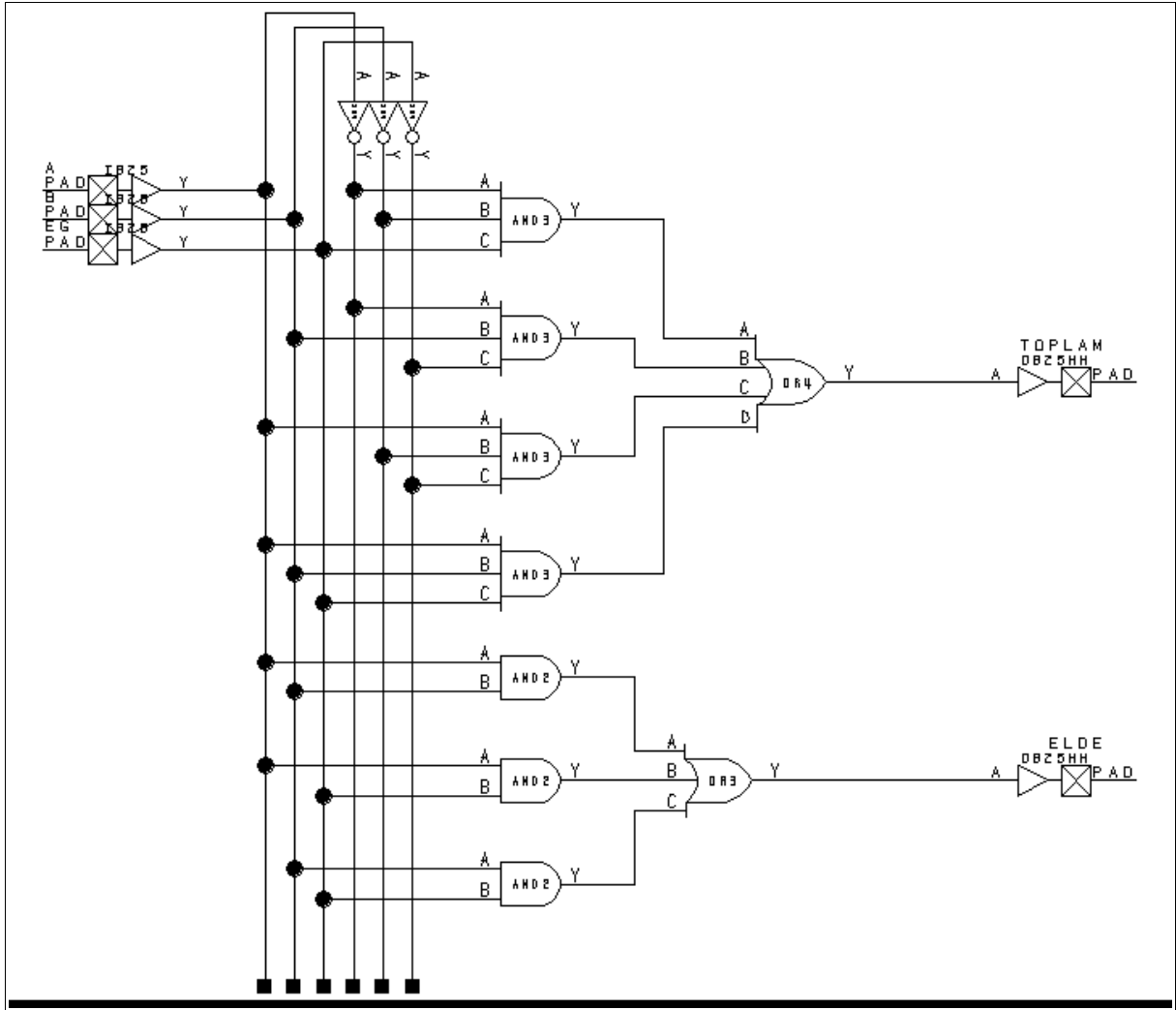
<i>Ec=ELDE</i>	<i>B,Eg</i>			
<i>A</i>	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$$E c = A E g + A B + B E g$$

(b)

Tablo 8.2

İndirgenmiş çarpımlar toplamı biçimindeki iki fonksiyonumuzu VE-VEYA-DEĞİL kapılarıyla gerçeklersek Şekil 8.1 deki devreyi elde ederiz.



Şekil 8.1

Bu şema ile devre sentezleyebilen, VHDL kodunu üretebilen programlar mevcuttur. Biz gene amacımıza uygun şekilde VHDL kodunu kendimiz üretelim :

```
-- tam_topla.vhd
-- deneme3.vhd
-- Erhan Kucukguzel
-- 2005
-- Tam toplayici
```

Kullanılacak kütüphane library ifadesi ile belirtilir. Her entity bildiriminden önce kullanılmalıdır.

```
library ieee;
use ieee.std_logic_1164.all;
library APA;

entity tam_topla is
    port(A,B,Eg:in std_logic; T,Ec:out std_logic);
end entity;

library ieee;
use ieee.std_logic_1164.all;
library APA;

entity VE2 is
    port (A,B:in std_logic; Y:out std_logic);
end entity;
```

```

architecture islev of VE2 is
begin
    Y <= A and B;
end islev;

library ieee;
use ieee.std_logic_1164.all;
library APA;

entity VE3 is
    port (A,B,C:in std_logic; Y:out std_logic);
end entity;

architecture islev of VE3 is
begin
    Y <= A and B and C;
end islev;

library ieee;
use ieee.std_logic_1164.all;
library APA;

entity VEYA4 is
    port (A,B,C,D:in std_logic; Y:out std_logic);
end entity;

architecture islev of VEYA4 is
begin
    Y <= A or B or C or D;
end islev;

library ieee;
use ieee.std_logic_1164.all;
library APA;

entity DEGIL is
    port (A:in std_logic; Y:out std_logic);
end entity;

architecture islev of DEGIL is
begin
    Y <= not A;
end islev;

architecture devre of tam_topla is

signal
    VE3_1_Y,VE3_2_Y,VE3_3_Y,VE3_4_Y,
    VE2_1_Y,VE2_2_Y,VE2_3_Y,
    DA,DB,DEg:std_logic;

component tam_topla is
    port(A,B,Eg:in std_logic; T,Ec:out std_logic);
end component;

component VE2 is
    port (A,B:in std_logic; Y:out std_logic);
end component;

```

Signal bildiriminde tanımlanan değişkenler, devremizde olup, entity bölümünde tanımlı olmayan işaretleri tanımlamak amacıyla kullanılır. Böylece devrenin bileşenleri arasındaki bağlantılar tamamen tanımlanabilir.

Component bölümünde, architecture bölümünde kullanılacak bileşenler, aynı entity bölümündeki şekilde tanımlanır.

```

component VE3 is
  port (A,B,C:in std_logic; Y:out std_logic);
end component;

component VEYA4 is
  port (A,B,C,D:in std_logic; Y:out std_logic);
end component;

component DEGIL is
  port (A:in std_logic; Y:out std_logic);
end component;

```

```
-- tanımlar bitti, devreye gecelim
```

```
begin
```

```

g1 : DEGIL port map (A, DA) ;
g2 : DEGIL port map (B, DB) ;
g3 : DEGIL port map (Eg, DEg) ;
g4 : VE3    port map (DA, DB, Eg, VE3_1_Y) ;
g5 : VE3    port map (DA, B, DEg, VE3_2_Y) ;
g6 : VE3    port map (A, DB, DEg, VE3_3_Y) ;
g7 : VE3    port map (A, B, Eg, VE3_4_Y) ;
g8 : VE2    port map (A, Eg, VE2_1_Y) ;
g9 : VE2    port map (A, B, VE2_2_Y) ;
g10: VE2    port map (B, Eg, VE2_3_Y) ;
g11: VEYA4  port map (VE3_1_Y, VE3_2_Y, VE3_3_Y, VE3_4_Y, T) ;
g12: VEYA4  port map (VE2_1_Y, VE2_2_Y, VE2_3_Y, VE2_3_Y, Ec) ;

```

```
end devre;
```

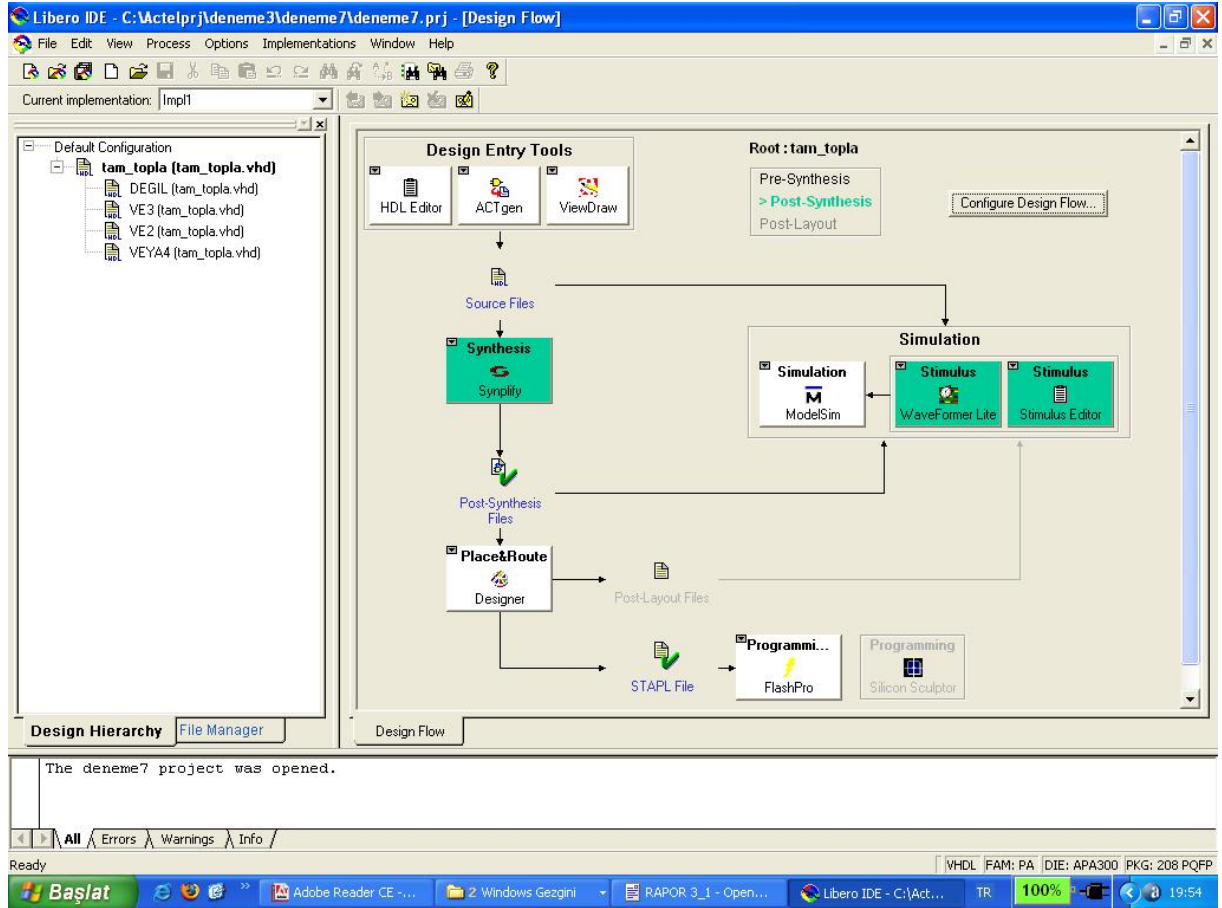
g1...g12 ile 12 adet kapının hem ne çeşit kapı oldukları hem de giriş çıkış işaretlerinin neler olduğu tanımlanır. ÖR: g8, iki girişli VE kapısı olup, girişleri A, Eg çıkışı VE2_1_Y işaretidir. A ve Eg devrenin entity de tanımlı girişi, VE2_1_Y ise architecture'da tanımlı işaretlerden biridir.

9. ACTEL Libero Ortamı

ACTEL firmasının kullanıcılarına sunduğu tümleşik geliştirme ortamı, tasarımcı için birçok ortamı bir arada sunar ve projelerin tüm detaylarını takip etmeyi sağlar. Tasarım akışı (design flow) penceresinde (Şekil 9.1), kullanıcıya sunulan ortamlar görülmektedir:

- Tasarım girişi dahili VHDL editörü veya şema yoluyla (**ViewDraw** programı),
- Sentezleme (netlist üretimi) **Synplicity** yazılımı ile,
- Benzetim (simülasyon) **ModelSim** yazılımı ile,
- İşaret üretimi **WaveFormer** yazılımı ile,
- FPGA içine yerleştirme ve pinlerin seçimi **Designer** yazılımı ile yürütülmektedir.

Tasarımın nasıl akması gerektiği ve adı geçen yazılımların detayları ilerleyen zamanlarda daha derinden işlenecektir.



Şekil 9.1 ACTEL Libero Tümleşik Geliştirme Ortamı

10. Sonuç ve Yorumlar

Şu denilebilir ki, akademik amaçlar ile, yahut küçük bütçeli projelerde FPGA, özel amaçlı tümdevre geliştirmek isteyen tasarımcı ve şirketler için son derece uygun bir yoldur. Erişmesi güç bütçelere sahip şirketler ile bilgi sahibi düşük bütçeli kullanıcı arasında adeta fırsat eşitliği yaratmaktadır. Tüm bu sebeplerden dolayı, özellikle de ülkemiz üniversitelerindeki akademik çalışmalara uygun bir zemindir.

FPGA yapıları da her alanda olduğu gibi gelişmektedir. FLASH bellek ve bellek kilidi teknolojileri, JTAG arayüzün dahil edilmesi bunlara örnektir. Tüm bu yatırımların gelişmesi, gelecekte de FPGA ların yaşamaya devam edeceklerine göstergedir. Kullanımının, mikroişlemciye nazaran güç oluşu sorunu da gün geçtikçe giderilmeye devam etmektedir. VHDL veya sistem bilgisi gerektirmeyen geliştirme ortamları yavaş yavaş ortaya çıkmaktadır.

Görünen o ki, ucuzluğu ve kolaylığı ile mikrodenetleyicili sistemler daima var olacaklardır. FPGA ise, üstün özellikleriyle, üst seviye pazara hitap edecektir. Bu pazarda, maliyetten çok başarımın önemli olduğu askeri uygulamalar, tıp elektroniği, haberleşme sistemleri her zaman yer alacaktır. Yüksek başarım sözkonusu olduğunda FPGA gerektiğinde ucuz çözüm de olabilmektedir, çünkü aynı işi yapmak için daha karmaşık işlemci sistemlerine gerek duyulacaktır.

Bu raporda FPGA dünyasına sadece bir giriş yapılmış olup, daha alınacak çok yol, öğrenilecek pek çok detay olduğu anlaşılmaktadır.

Kaynakça :

- [1] Maxfield, C., “The Design Warrior's Guide to FPGAs”, Elsevier.
- [2] Dervişoğlu, A., “İleri Lojik Devreler Ders Notları”.
- [3] ACTEL ProASIC^{PLUS} Datasheet
- [4] ACTEL Libero User Guide