

T.C.
SÜLEYMAN DEMİREL ÜNİVERSİTESİ



MÜHENDİSLİK MİMARLIK FAKÜLTESİ
ELEKTRONİK VE HABERLEŞME MÜHENDİSLİĞİ
BÖLÜMÜ

BİTİRME ÖDEVİ
FPGA YONGA MİMARİSİ ve KULLANIMI

DANIŞMAN
YRD. DOÇ. DR. ALİ MANZAK

HAZIRLAYAN
ATILLA AYDIN
0111008003

HAZİRAN – 2005
ISPARTA

ONAY

Bu çalışma, Süleyman Demirel Üniversitesi Mühendislik Mimarlık Fakültesi Elektronik ve Haberleşme Mühendisliği Bölümü 2004–2005 öğretim yılı Bitirme Projesi yönergesine uygun olarak hazırlanmıştır ve anılan bölüme sunulmuştur.

ATILLA AYDIN

0111008003

ISPARTA - 2005

.....

Proje Yöneticisi

YRD. DOÇ. DR. ALİ MANZAK

.....

Bölüm Başkanı

PROF. DR. MUSTAFA MERDAN

.....

Sınav Komisyonu Üyesi

.....

Sınav Komisyonu Üyesi

ÖNSÖZ

Hayatım boyunca beni en iyi şekilde yetiştiren, her şeyin en iyisine layık olan aileme çok teşekkür ederim.

Bitirme ödevimi almamda beni teşvik eden Yrd. Doç. Dr. Ali Manzak hocama gönülden teşekkür ederim. Umarım birlikte ileri ki yıllarda ülkemize faydalı olacak nice çalışmalar yaparız.

Süleyman Demirel Üniversitesi Elektronik ve Haberleşme Mühendisliği Bölümü' nde bana bir kimlik kazandıran, hayata dair zorlukları görmemi sağlayan tüm hocalarıma ve arkadaşlarıma teşekkür ederim.

Atilla AYDIN
ISPARTA 2005

ÖZET

FPGA' lar donanımı istenildiđi gibi bařtan yapılandırılabilir yongalardır. Bu alıřmada üretim teknolojileri ve tekniklerini, i mimarilerini ve genel kullanımlarını inceleyeceđim.

İÇİNDEKİLER

1. ALAN PROGRAMLANIR KAPI DİZİLERİ.....	1
1.1. Kullanım Alanları	1
1.1.1. ASIC ve Custom Silicon	1
1.1.2. Digital Signal Processing.....	2
1.1.3. Embedded Microcontroller.....	2
1.1.4. Fiziksel Katman Haberleşmeleri.....	2
1.1.5. Reconfigurable Computing.....	2
1.2. Üretici Firmalar.....	3
2. TEMEL TEKNOLOJİLER.....	4
2.1. Basit programlanabilir fonksiyon	4
2.2. Fusible Link Teknolojisi.....	4
2.3. Antifuse Teknolojisi.....	5
2.4. Mask-Programmed Aygıtlar.....	6
2.5. PROM (Programmable Read-Only Memory) Teknolojisi.....	7
2.6. EPROM (Erasable Programmable Read-Only Memory) Teknolojisi.....	7
2.7. EEPROM (Electrically Erasable Programmable Read-Only Memory).....	8
2.8. FLASH Teknolojisi.....	8
2.9. SRAM (Static Random Access Memory) Teknolojisi.....	9
3. FPGA TEKNOLOJİSİNİN GELİŞİMİ	11
3.1. SPLD ve CPLD.....	11
3.2. PROM.....	12
3.3. PLA (Programmable Logic Array)	12
3.4. PAL (Programmable Array Logic)	13
3.5. CPLD (Complex Programmable Logic Device)	14
3.6. ASIC (Application Specific Integrated Circuit)	15
3.6.1. Full Custom	15
3.6.2. Standard Cell.....	16
3.6.3. Gate Array.....	16
3.6.4. Structured ASIC	17
3.7. FPGA.....	18

4. FPGA MİMARİSİ.....	20
4.1. Üretim Teknikleri	20
4.1.1. SRAM temelli aygıtlar.....	20
4.1.2. Antifuse temelli aygıtlar.....	22
4.1.3. E ² PROM/FLASH temelli aygıtlar.....	23
4.1.4. Karma FLASH-SRAM aygıtlar.....	23
4.2. Birim Hücre Mimarileri	24
4.3.1. MUX tabanlı.....	25
4.3.2. LUT tabanlı	26
4.4. Programlanır Mantık Öğeleri.....	28
4.4.1.1. Logic Cell.....	28
4.4.1.2. Logic Element ve Adaptive Logic Module	28
4.4.2. Slice	29
4.4.3. CLB ve LAB	29
4.4.3.1. Ara bağlantılar.....	30
4.4.3.2. Kontrol Sinyalleri.....	30
4.4.3.3. Adaptive Logic Module.....	30
4.4.3.4. Shared Arithmetic Chain.....	33
4.4.4. Distributed RAM.....	34
4.4.5. Shift Register.....	34
4.4.6. Fast Carry Chain.....	34
4.5. Embedded RAM.....	34
4.6. Embedded Multiplier, Adder, MAC.....	35
4.7. Embedded processor core	36
4.8. Clock Tree	37
4.9. Clock Manager.....	38
4.10. Genel Amaçlı G/Ç.....	41
4.11. Yapılandırılabilir G/Ç Özdirençleri.....	41
4.12. Gigabit Alıcı Vericiler	42
4.13. IP (Intellectual Property) Core.....	42
4.14. Ara Bağlantı Mimarisi.....	43
4.14.1. Altera Ara Bağlantı Yaklaşımı.....	43
4.14.2. Xilinx Ara Bağlantı Yaklaşımı.....	46
4.15. Çekirdek ve G/Ç besleme gerilimleri.....	48

5. FPGA PROGRAMLAMA	50
5.1. Yapılandırma Dosyası.....	50
5.2. Yapılandırma Kipleri	51
5.2.1 Fast Passive Parallel (FPP) kipi.....	52
5.2.2 Active Serial (AS) kipi	52
5.2.3. Passive Serial kipi	52
5.2.4. Passive Paralel Asynchronous (PPA) kipi	53
5.2.5. JTAG kipi	53
6. UYGULAMA GELİŞTİRME	56
6.1. Tasarım Girişi.....	57
6.2. Sentez.....	57
6.3. İşlevsel Benzetim.....	57
6.4. Yerleştirme.....	57
6.5. Zaman Analizi ve Benzetimi.....	57
6.6. Programlama ve Yapılandırma.....	58
6.7. Örnek Uygulama.....	58
6.7.1. Devre Şeması ve Açıklama.....	58
6.7.2. Devrenin aşamalı gerçekleşmesi.....	59
7. SONUÇ.....	67
8. KAYNAKLAR.....	68

1. FIELD PROGRAMMABLE GATE ARRAY

FPGA (Alan Programlanır Kapı Dizileri)' lar yapılandırılabilir mantık blokları ile birlikte bu bloklar arasındaki deęiřtirilebilir ara baęlantılardan oluřan sayısal tmleřik devrelerdir. Tasarım mhendisleri bu yapıları programlayarak muazzam çeřitlikte grevler gerekleyebilirler.

1.1. Kullanım Alanları

Kullanıma ilk bařlandığı 1980' lerin ortalarında FPGA yongalardan oęunlukla ara yapıřtırıcı mantık ve kısıtlı veri iřleme grevlerinde faydalanıldı. Ara yapıřtırıcı mantıklar daha geniř mantık blokları, fonksiyonlar veya cihazlar arasında ara baęlantı olarak kullanıldı. 1990' ların ilk yıllarında artan kapasiteleri sayesinde geniř veri iřlemleri gerektiren haberleřme ve aę ortamlarında kullanımı arttı. 90' ların sonlarına doęru tketickiye ynelik, otomotiv ve endstriyel uygulamalardaki kullanımları devasa byme sergiledi.

FPGA' lar genellikle ASIC tasarımların ilk rnekleri veya yeni bir algoritmanın fiziksel gereklenebilirlięini doęrulamak adına donanım ortamı saęlamak iin kullanılırlar. Bununla birlikte, dřk geliřtirme maliyetleri ve kısa srede pazara sunulabilme zellikleriyle gittike son rn yelpazesindeki yerlerini almaktadırlar.

2000' lerin ilk yıllarında milyonlarca kapı ieren yksek performanslı FPGA' lar piyasada yerini aldı. Bazıları gml mikroiřlemci ekirdekleri, yksek hızlı Giriř/ıkıř (I/O) arayzleri, gml RAM ve DSP bekleri sunmaktadır. Sonu olarak gnmz FPGA' ları beř ana pazar kolunda yer bulmaktadırlar:

1.1.1. ASIC ve Custom Silicon

ASIC ve Custom Silicon, byk hacimli yongalar gereklemeye imkan veren fakat uzun tasarım sreci gerektiren yntemlerdir. nceleri sadece bu yntemlerle gereklenebilen eřitli tasarımlardaki kullanımı giderek artmaktadır.

1.1.2. Digital Signal Processing

Yüksek hızlı DSP geleneksel olarak özellikle uyarlanmış mikroişlemciler (DSP) ile gerçekleştirilmektedir. Günümüz FPGA' ları DSP işlemlerini kolaylaştıracak çarpıcılara, özel aritmetik yönlendiricilere ve büyük miktarlarda dahili RAM' e sahip teknolojidedirler. Bu özellikler FPGA' lerce sağlanan yoğun paralel işlem yeteneğiyle birleştiğinde sonuç en hızlı DSP yongasından 500 kez ve daha fazla hızlı olabilmektedir.

1.1.3. Embedded Microcontroller

Küçük kontrol fonksiyonları genellikle özel amaçlı mikrodenetleyiciler ile çözümlenir. FPGA fiyatları düşmekte ve bununla birlikte en küçük modelleri bile özel I/O fonksiyonlarını içeren işlem çekirdeklerini sağlayabilecek kapasitededir. Bu yönleriyle çekim noktası olmaktadırlar.

1.1.4. Fiziksel Katman Haberleşmeleri

FPGA' ler uzun zamandır fiziksel katman ile yüksek seviyeli haberleşme protokol katmanları arasında arayüz bağlantısını gerçekleştirmektedirler (Glue Logic). Son teknoloji FPGA' ler haberleşme ve ağ oluşturma fonksiyonlarını tek cihazda birleştiren yüksek hızlı alıcı-verici birimlerini içerebilmektedir.

1.1.5. Reconfigurable Computing

FPGA' larca sağlanan içsel paralellik ve tekrar yapılandırılabilirliği sağlayan "hardware accelerate" yazılım algoritmalarını kullanmaktır. Çeşitli firmalar halen yeni çözümler keşfetmek amacıyla donanım benzetimlerinden şifreleme tahlillerine kadar büyük FPGA temelli tekrar yapılandırılabilir hesaplama tertibatları hazırlamaktadırlar.

1.2. Üretici Firmalar

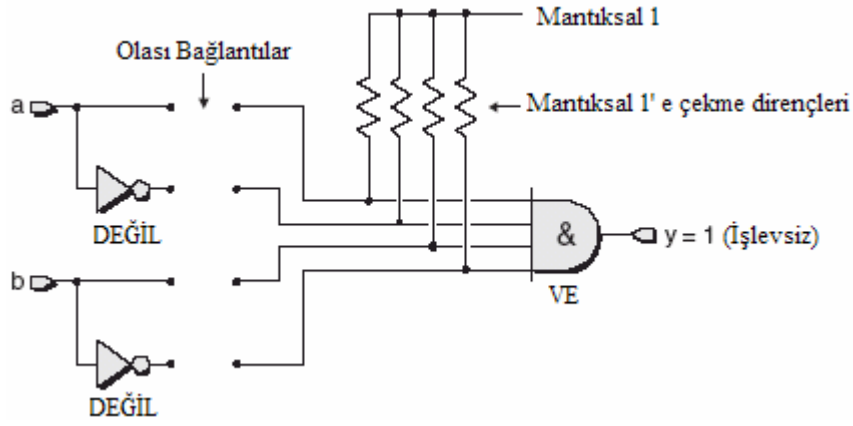
Actel	;	www.actel.com
Altera	;	www.altera.com
Anadigm	;	www.anadigm.com
Atmel	;	www.atmel.com
Lattice Semiconductor	;	www.latticesemi.com
Leopard Logic	;	www.leopardlogic.com
QuickLogic	;	www.quicklogic.com
Xilinx	;	www.xilinx.com

Bu firmalar arasında Xilinx ve Altera en büyük hacimli FPGA sağlayıcılarıdır. Her firmanın kendilerine has teknik üstünlükleri olup araştırma konum bu noktaları içermediği için tek tek şirket bazında inceleme yapmadım. Bu çalışmada daha çok Altera ve Xilinx şirketlerinin en yeni ürünlerini inceledim. Diğer şirketlerin ürünlerini üretim teknik ve teknolojileri açılarından dikkate aldım.

2. TEMEL TEKNOLOJİLER

2.1. Basit programlanabilir fonksiyon

a ve b girişleri y çıkış olacak şekilde basit bir programlanabilir işlev üzerinde düşünecek olursak ileriki safhaları daha kolay canlandırabiliriz:



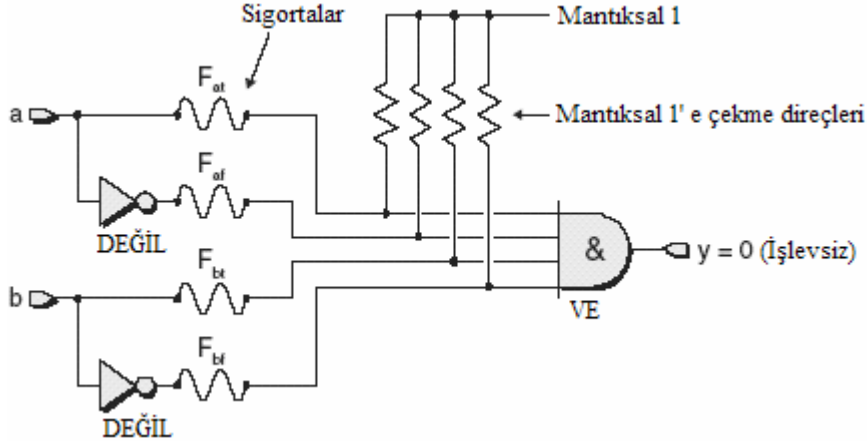
Şekil 2.1. : Basit programlanabilir işlev

a ve b girişleri gerçek ve tümlenmiş değeri ile verilmiştir. Olası bağlantı noktaları gözükmemektedir. Görüleceği üzere herhangi bir bağlantı olmaması durumunda tüm girişler pull-up (mantıksal 1'e çekme) dirençleri yoluyla mantıksal 1'e çekilmiş olur. y çıkışı her zaman mantıksal 1 olur.

2.2. Fusible Link Teknolojisi

Kullanıcıya aygıtını programlama izni veren ilk teknolojilerdendir. Aygıtlar, olası tüm bağlantıları olacak şekilde üretilirler. Bağlantılar sigorta (fuse) olarak isimlendirilir. Günlük hayattaki sigortaların ilgili teknoloji ile üretilmiş şeklidirler.

Mevcut haliyle VE kapısı nedeniyle çıkış daima mantıksal 0'dır. Tasarımcı istenmeyen sigortaları aygıt girişine nispeten yüksek voltaj ve akım darbeleri uygulayarak temizler.

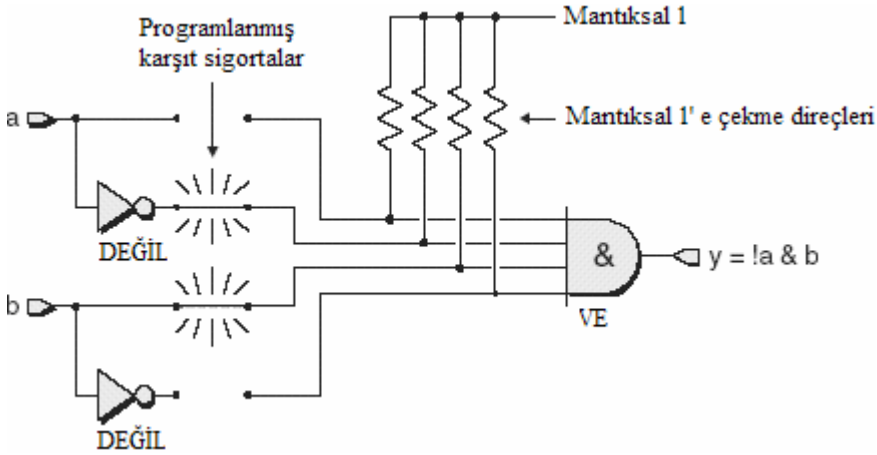


Şekil 2.2. : Programlanmamış eritilebilir bağlantılar

Eritilebilir-bağlantı teknolojisine sahip cihazlar bir kez programlanır. OTP (One Time Programmable) olarak isimlendirilirler. Çünkü yakılan bölgede geri dönüş yoktur.

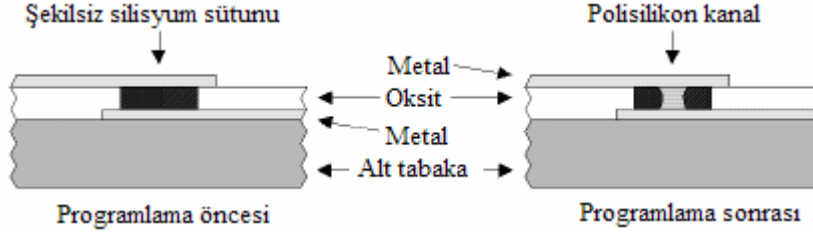
2.3. Antifuse Teknolojisi

Eritilebilir teknolojiye karşıt olarak her programlanabilir bölüm antifuse bağlantıya sahiptir. Yapılandırılmadan önce karşıt sigorta, açık devre olarak düşünebileceğimiz yüksek direnç özelliğindedir.



Şekil 2.3. : Programlanmış karşıt sigorta bağlantıları

Tasarıma göre aygıt girişine nispeten yüksek gerilim ve akım darbeleri uygulayarak seçim gerçekleşir. Antifuse noktaları başlangıçta iki metal yolu bağlayan biçimsiz (amorf) yapıdadır. Bu haliyle direnç değeri 10^9 ohmu aşan yalıtkandır.



Şekil 2.4. : Karşıt sigorta gerçekleşmesi

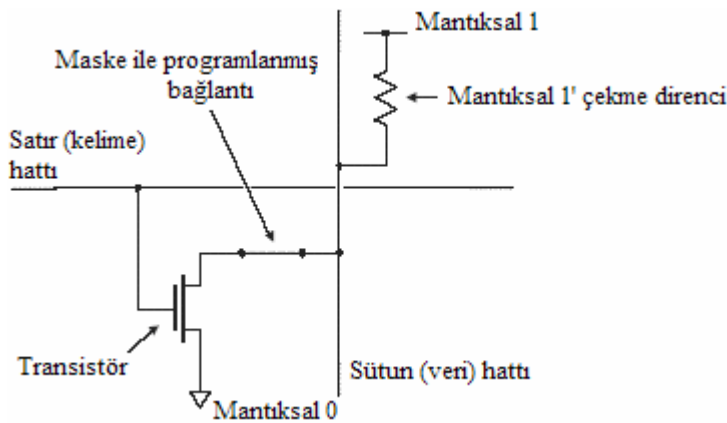
Bu yalıtkan yapı, programlama ile iletken polisilikon' a dönüşür. Eritilebilir bağlantı teknolojisinde olduğu gibi bir kez programlanabilir.

2.4. Mask-Programmed Aygıtlar

Elektronik sistemlerde bellek birimleri iki ana sınıfta yer alırlar: Read Only Memory (ROM) ve Random Access Memory (RAM).

ROM' lar mask-programmed olarak adlandırılırlar. Çünkü üretim safhasında içerdikleri veriler photo-mask isimli şablonlarca kodlanır.

Bütün ROM' lar dizi yapısı oluşturacak şekilde satır (word) ve sütun hatları (data) ihtiva eder. Her sütun pull-up direncine sahiptir. Her satır sütun kesişiminde ilişkilendirilmiş transistör ve maske ile programlanmış olabilecek bağlantı vardır. Satır aktif olduğunda ilişik transistör sütun hattını mantıksal 0' a çekecektir. Şayet transistör maske ile sütuna bağlanmamış ise transistör tepki göstermeyecek ve mantıksal 1' e çekme direnci sütun hattını mantıksal 1' e sürecektir.



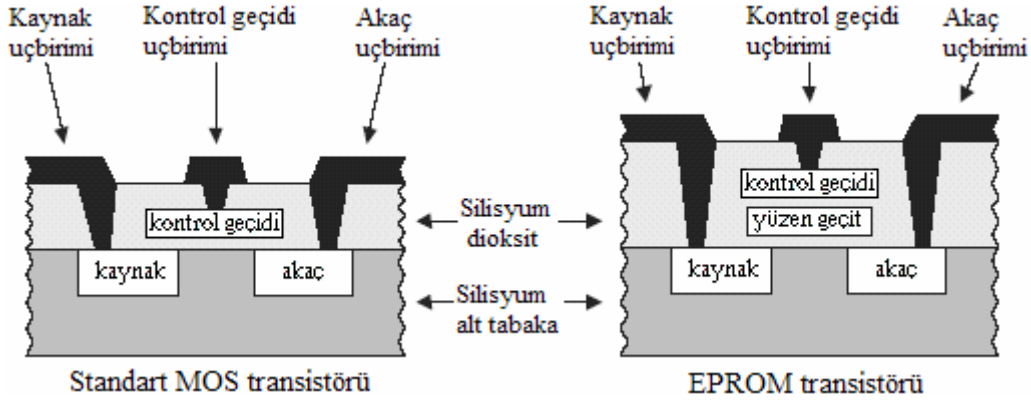
Şekil 2.5. : Transistör tabanlı maske ile programlanmış ROM hücresi

2.5. PROM (Programmable Read-Only Memory) Teknolojisi

Maske ile programlama masraflı olduğu için bu noktalar tümünden sigorta bağlantıları ile donatılır. Başlangıçta satırlar aktif olduğunda transistörler çalışır ve sütunları mantıksal 0'a çeker. Tasarıma göre istenen programlama sigorta teknolojisine uygun şekilde yapılır. Tasarım mühendislerince basit fonksiyonları gerçeklemek için kullanışlı bulunur. Maliyetleri daha ucuzdur.

2.6. EPROM (Erasable Programmable Read-Only Memory) Teknolojisi

Silinip Programlanabilir Salt Okunur Bellek' ler ilk Intel firmasınınca 1971 yılında takdim edildi. EPROM transistör, MOS transistörden farklı olarak oksit katmanı ile yalıtılmış olan ikinci polisilikon' a; yüzen geçide (floating gate) sahiptir.



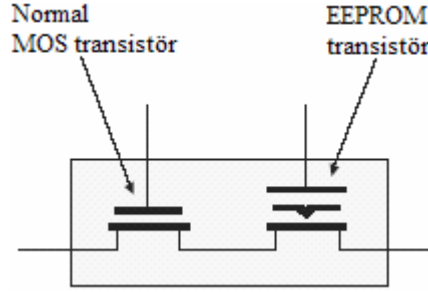
Şekil 2.6. : Standard MOS ve EPROM transistör yapıları

Başlangıçta kayan geçit şarj edilmemiştir ve kontrol geçidinin çalışmasını etkilemez. Kontrol ve kanal uçları arasına programlama amaçlı (12V civarında) gerilim uygulandığında faal elektronlar yüksek enerjili elektron akıtma (high energy electron injection) olarak bilinen işlem ile oksitten kayan geçite yönelirler. Programlama sinyali kesildiğinde eksi yük kayan geçitte kalır. Normal şartlarda 10 yıl kadar dağılmadan korunabilir. Eksi yük kontrol geçidinin normal çalışmasını engeller. Başlangıç durumunda EPROM transistörler yüksüzdür. Satırlar aktif olduğunda transistörler çalışır ve transistörler kendi sütun hatlarını mantıksal 0'a çekerler. Programlama sonrası yüklü transistör hücreleri mantıksal 1 değeri içerir. Yük boşalımı ile bir EPROM hücre silinebilir. Boşalım için gerekli enerji morötesi (Ultraviolet - UV-) ışınım ile sağlanır.

Paketleme masrafları ve 20 dk.' yı bulan silme işlemi ana sorunlarıdır. Transistör boyutları küçüldükçe ve tasarım yoğunluğu arttıkça UV ışınımına karşı daha korunmasız olmaktadır.

2.7. EEPROM (Electrically Erasable Programmable Read-Only Memory)

E²PROM olarak ta bilinen Elektriksel Silinip Programlanı Salt Okunur Bellek' ler eşdeğer bir EPROM hücreden 2.5 kat daha geniş yer kaplar. Çünkü iki transistör ve aralarındaki boşluk vardır.



Şekil 2.7. : E²PROM hücre






E²PROM transistörü de EPROM transistör gibi kayan geçide sahiptir. Ancak bu geçidi çevreleyen oksit daha incedir. İkinci transistör elektriksel silme işlemi içindir.

2.8. FLASH Teknolojisi

Flash temelli bileşenler çeşitli mimariler kullanabilir. Bazıları EPROM hücre ile aynı alana sahip ama E²PROM' deki gibi ince oksit katmanlı olacak şekilde tek yüzen geçitli transistör hücresine sahip olabilir. Bu mimari elektrik ile silinebilir ancak tüm cihaz veya büyük kısımlar halinde temizlenir. Diğer bir mimari E²PROM benzeri iki transistörlü hücreye sahiptir. Bu suretle kelime kelime silinebilir ve tekrar programlanabilir.

Flash hücrelerin ilk sürümleri tek bitlik veri saklayabiliyorlardı. 2002 yılından bu yana bu kapasite arttı. İlgili teknikte Flash transistördeki yüzen geçit iki bit temsil etmek için ayrı depolama seviyeleri içerir.

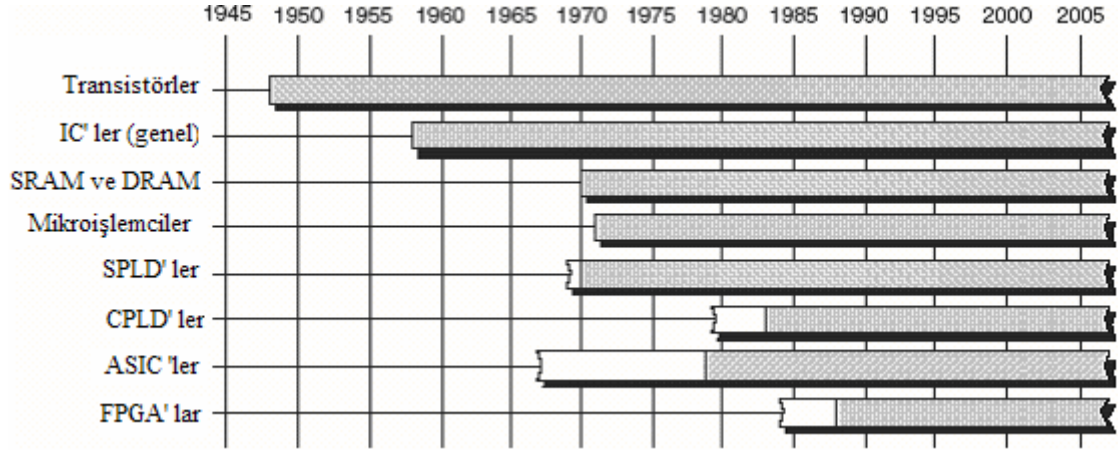
Çizelge 2.1. : Üretim teknolojilerinin karşılaştırılması

Teknoloji	Sembol	Tekrar Programlanır	Geçicilik	Kullanım
Fusible-link		Hayır	Hayır	SPLD
Antifuse		Hayır	Hayır	FPGA
EPROM		Evet Devre dışında	Hayır	SPLD ve CPLD
EEPROM / FLASH		Evet Devre içerisinde	Hayır	SPLD ve CPLD (bazı FPGA)
SRAM		Evet Devre içerisinde	Evet	FPGA (bazı CPLD)

3. FPGA TEKNOLOJİSİNİN GELİŞİMİ

Doğrudan giriş yapmadan önce ilgili teknolojilere kısaca değineceğim. Aşağıdaki çizgede ilgili teknolojilerin yıllara göre dağılımı görülmektedir.

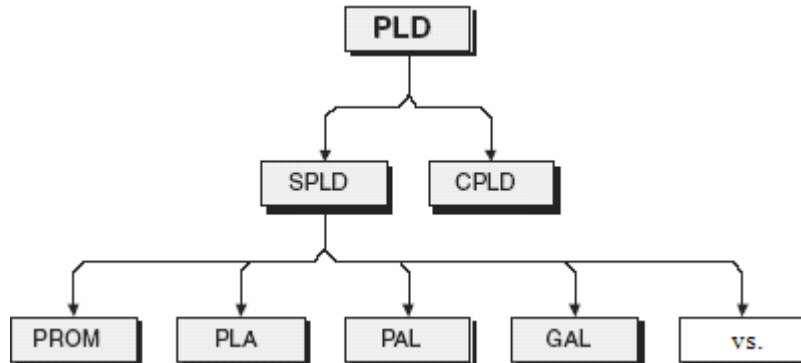
Çizelge 3.1. : İlgili teknolojilerin yıllara göre dağılımı



Zaman çizelgesindeki beyaz çubuklar o tekniğin vücut bulduğunu fakat bu zaman zarfında uygulamasına geçilmediğini göstermektedir. Örneğin, Xilinx firması FPGA'ı 1984 yılında geliştirmesine rağmen bu alana ancak 90'lı yıllarda öncelik vermiştir.

3.1. SPLD ve CPLD

İlk programlanabilir IC'ler genel olarak "programmable logic device (PLD)" olarak takdim edilir. Özgün bileşen, 1970 yılında sunulan ve daha basit olan PROM'lardır ama programlanabilir mantık aygıtları PLD'ler kabul edilir. Genel sınıflandırma aşağıda verilmiştir.

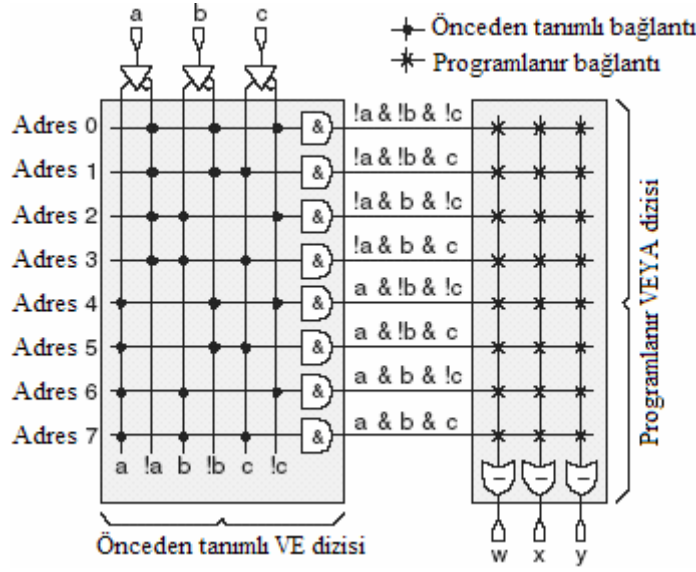


Şekil 3.1. : Programlanabilir mantık aygıtları genel sınıflandırma

Şekilde verilenler haricinde EPLD, E²PLD ve FLASH tasarımları da mevcuttur.

3.2. PROM

En basit PLD çeşididir. Sabit VE kapısı fonksiyonlarının VEYA kapı fonksiyonlarını sürmesi şeklinde oluşturulmuştur.



Şekil 3.2. : 3-giriş, 3-çıkışlı programlanmamış PROM

Her VE fonksiyonu a,b ve c girişlerinden gelen gerçek veya tümleyen değerlerinden üçü ile bağlantılıdır. Yine her VEYA fonksiyonu VE dizisi çıkışlarından gelen sekiz giriş sahiptir. Yukarıdaki yapı ile 3 giriş 3 çıkışlı tüm fonksiyon birleşimleri sağlanabilir. Gerçekte çok daha fazla sayıda giriş ve çıkış barındırırlar. İstenilen fonksiyona göre VEYA kapıları programlanır. VEYA dizisindeki programlanabilir bağlantılar eritilir bağlantı, EPROM ve E²PROM hücresi olabilir.

3.3. PLA (Programmable Logic Array)

Programlanabilir mantık dizisi teknolojisi PROM kısıtlamalarını aşarak takriben 1975' te geliştirildi. Basit PLD' ler içerisinde kullanıcıya en fazla yapılandırma imkanı verendir. Çünkü VE ve VEYA dizilerinin her ikisi de programlanabilir.

VE dizisindeki VE fonksiyon sayısı, giriş sayısından bağımsızdır. Ek VE kapıları dizideki satırlara kolayca dahil edilebilir.

PLA' lara göre daha hızlıdır. Ancak kısıtlı sayıda terimin birlikte VEYA' lanabilmesi olumsuz yanındır.

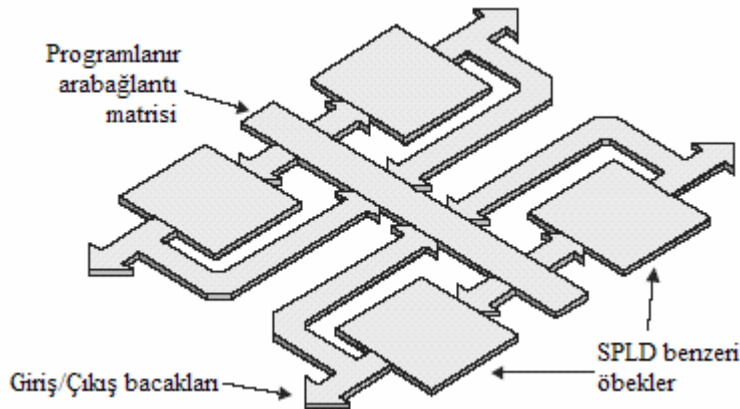
GAL (Generic Array Logic), PAL mantığında olup CMOS elektriksel silinir (E^2) özellik sağlar. Lattice Semiconductor Corp. tarafından üretilmektedir.

Verilen basit örneklere rağmen gerçek cihazlar daha büyüktür ve daha fazla programlama seçeneği gereksinimi duyarlar. Çıkışın terslenmesi, üç durumlu çıkış, kayıtlı veya mandallı çıkış, ilgili bacağı çıkış veya ek giriş olarak kullanılabilmesi gibi çeşitli seçenekler olabilir.

3.5. CPLD (Complex Programmable Logic Device)

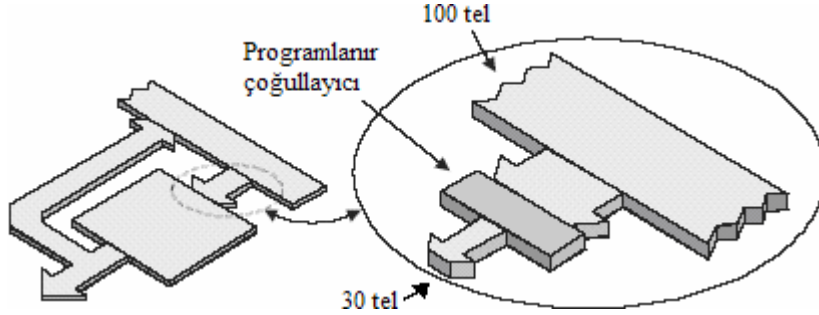
Artan kapasite ihtiyacı neticesinde 1984 yılında Altera firması CMOS ve EPROM teknolojilerine dayanan karmaşık programlanır mantık aygıtlarını (CPLD) geliştirdi. CMOS kullanarak küçük güç tüketimi ile muazzam işlevsel yoğunluk ve karmaşıklık sağlanmış oldu. Programlanırlık EPROM hücrelerine dayandığından, geliştirme ve ön ürün için yeğlendi.

Genel olarak G/Ç öbeği ve aynı programlanır ara bağlantı matrisini paylaşan SPLD (Simple PLD) öbeklerinden oluşur. Ara bağlantılar ile sinyaller aygıtın bir bölümünden diğer tüm bölümlerine ulaşabilir.



Şekil 3.5 : Genel CPLD yapısı

Programların ara bağlantı 100 tel içerebilir iken tek SPLD öbeği sınırlı sayıda tel barındırabilir. Bu nedenle SPLD öbekleri ara bağlantı matrisine, programların çoğullayıcı (MUX) kullanarak ara yüzlenir.

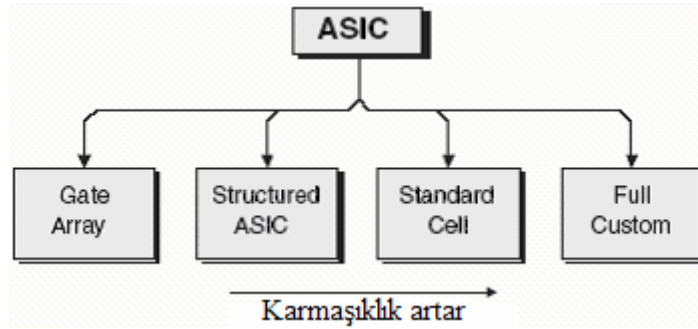


Şekil 3.6. : CPLD' de programların çoğullayıcı kullanımı

Üretici firmaya ve aygıt ailesine göre CPLD programların anahtarları EPROM, E²PROM, FLASH veya SRAM hücre tekniğini kullanabilir.

3.6. ASIC (Application Specific Integrated Circuit)

Uygulamaya özgü tümdevre kendi içerisinde dört alt başlıkta incelenir.



Şekil 3.7. : ASIC tasarım yöntemleri

3.6.1. Full Custom

Tasarlanan devrelerin hızı ve karmaşıklığı arttıkça tasarımcının üretim sürecine müdahalesi de artar. Yüksek hızlı sistemler ile gizliliği önemli olan devreler Full-Custom yöntemi ile tasarlanır. Bu yöntemde üreticinin gönderdiği kütüphaneler kullanılmaz. Üreticiden metal bağlantı yüzeyleri sayısı, iki kanal arası izin verilen en kısa mesafe gibi teknolojik bilgiler alınır. Üreticiye masklar dışında herhangi bir bilgi gönderilmediği ve

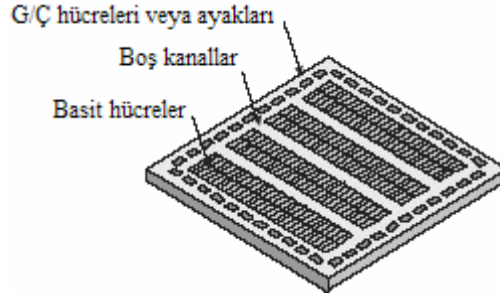
üreticinin kütüphaneleri kullanmadığı için tasarımın çözülme olasılığı yoktur. Tasarımı ve üretimi, önceki iki yöntemle göre daha uzun süren bir yöntemdir.

3.6.2. Standard Cell

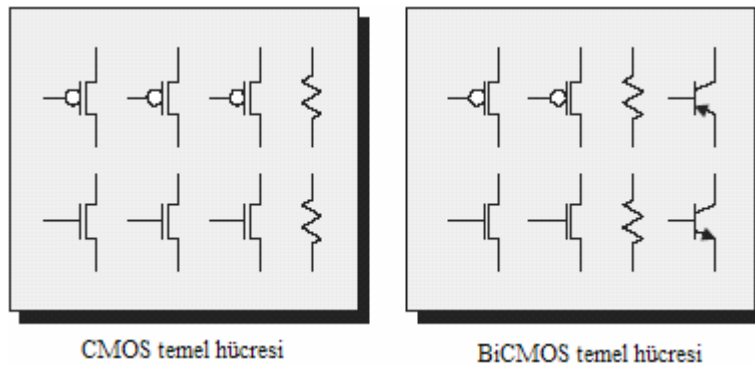
Bu yöntemde de üreticinin kütüphaneleri kullanmakla birlikte yonga üzerindeki hücrelerin konumu tasarımcı tarafından belirlenir. Gate array yöntemi gibi temel hücre yaklaşımı yoktur. Kütüphane içinde yer alan modeller üzerinde her hangi bir değişiklik yapılmamakla birlikte, yonga üzerinde oluşturulacak serimde tasarımcının söz sahibi olması, üretim sonuçlarının benzetim sonuçlarına daha yakın olmasını sağlar. Sistem gizliliği daha fazladır. Ancak yine de üreticinin modellerinin kullanılmış olması, çok zor olmakla birlikte, sistemin çözülme olasılığını gündeme getirebilmektedir.

3.6.3. Gate Array

Bağlantısız transistör ve direnç içeren temel hücreler mantığına dayanır. Her ASIC sağlayıcısı özel birim hücrelerinde en uygun bileşen dağılımını belirler.



Şekil 3.8. : Gate array yaklaşımındaki yonga görünümü



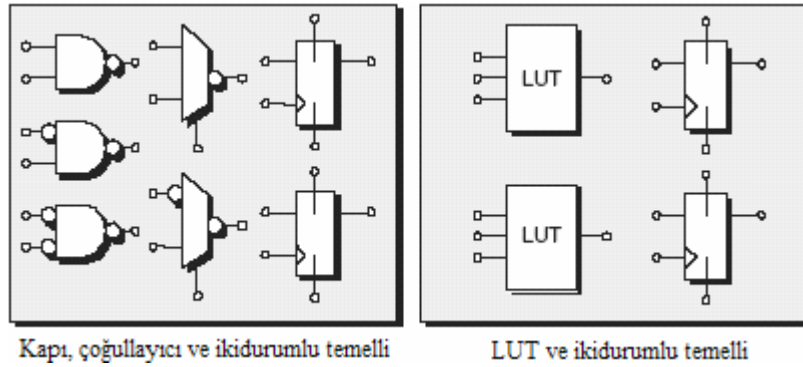
Şekil 3.9. :Basit gate array temel hücreleri

En kısa sürede sonuç alınabilen, tasarımı ve üretimi kolay olan ASIC tasarım ve üretim yöntemidir. Üreticiye masklarla birlikte devre şemaları da verildiği için tasarım gizliliği en az olan yöntemdir. Ancak tasarım ve üretim süreçlerinin kısılalığı, ASIC tasarımına yeni başlayanlar için, sonuçları kısa bir süre içinde görmek açısından, bu yöntemi çekici bir duruma getirmektedir.

Olumsuz tarafları, çoğu tasarım önemli miktarda içsel kaynağı kullanmadan bırakır. Bağlantı düzenlemeleri kısıtlanır ve en iyi durum sağlanamaz. Bu da tasarımın başarımını ve güç tüketimini olumsuz etkiler.

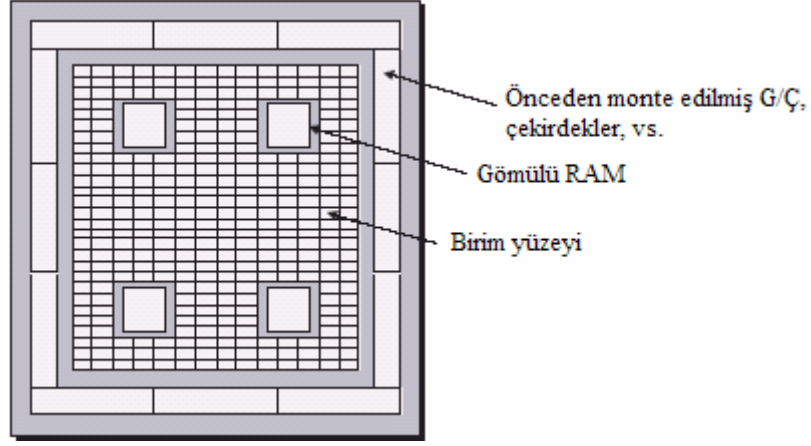
3.6.4. Structured ASIC

On yıllık geçmişine rağmen son iki-üç yıldır ASIC tasarım maliyeti ve geliştirme zaman değerlerini düşürme çabalarının sonucunda kullanılmaktadır. Her aygıtta “module” (birim) veya “tile” (kiremit) olarak isimlendirilen temel öğeler vardır. Aşağıdaki şekilde öğe içerikleri genel olarak verilmiştir.



Şekil 3.10. : Yapısal ASIC temelleri

Çoğu yapısal ASIC mimarisi için özelleşmiş iki veya üç metalleme katmanı yeterlidir. Bu sayede tasarımı oluşturma maliyeti ve zamanı önemli ölçüde azalır. Ancak yapısal ASIC, Standard Cell yöntemine göre aynı fonksiyonu gerçeklemek için silikon alanı ve güç tüketimi açılarından yaklaşık üç kat fazlasına gereksinim duyar.

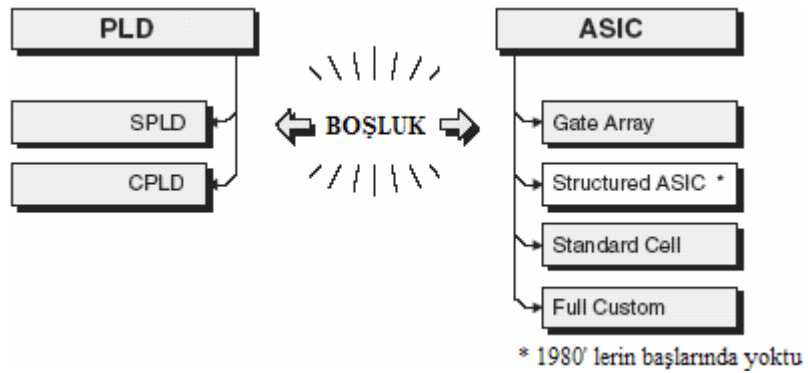


Şekil 3.11. : Yapısal ASIC genel görünümü

3.7. FPGA

Sayısal tümdevre sürecinde 80' li yıllarda belli boşluklar görülmeye başlandı. Bir tarafta SPLD ve CPLD gibi programlanır yongalar vardı. Bunlar yüksek yapılandırılabilirlik, hızlı tasarım ve değişiklik sürelerine sahiptiler ama geniş ve karmaşık tasarımları destekleyemiyorlardı.

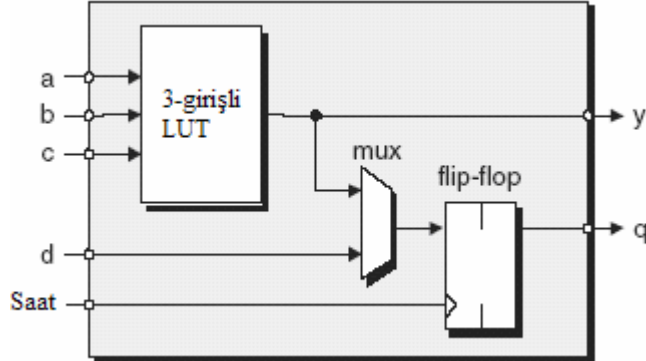
Diğer tarafta ASIC tasarım bulunmaktaydı. Çok geniş ve karmaşık işlevleri desteklemelerine rağmen, oldukça pahalı ve zaman harcayan sürece sahiptiler. Üstelik tasarımın geri dönüşü yoktur.



Şekil 3.14. : PLD ve ASIC yaklaşımları arasındaki boşluk

Bu aralığı doldurmak amacıyla Xilinx firması FPGA adını verdiği yeni bir IC sınıfı geliştirdi ve 1984 yılında pazara sunulacak hale getirdi. İlk FPGA' ler CMOS tabanlı ve yapılandırma için SRAM hücreleri kullanıyordu. İlk örnekleri çok daha basit olmalarına rağmen temelde var olan mimari çoğu açıdan hala kullanılmaktadır.

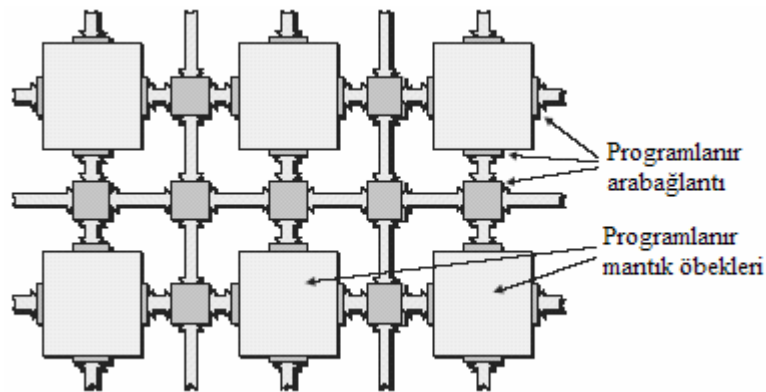
İlk yongalar 3-girişli LUT, yazmaç ve MUX' tan oluşan programlanır mantık öbeklerine dayanıyorlardı.



Şekil 3.15. : Programlanır mantık öbeğini biçimlendiren anahtar öğeler

SRAM hücreleri vasıtası ile yongadaki her mantık öbeği farklı işlevler için yapılandırılabilir. Her yazmaç mantıksal 1 veya mantıksal 0 yapılarak başlangıç durumuna getirilmeli ve ikidurumlu veya mandal gibi davranması için ayarlanmalı. İkidurumlu seçilmişse, yazmaç saat darbesinin düşen veya yükselen kenarı ile tetiklenmesi için yapılandırılmalı. İkidurumluyu besleyen çoğullayıcı (MUX), LUT çıkışı veya ayrı giriş için yapılandırılmalı. LUT da her hangi 3 girişli mantık işlevi için ayarlanmalı. Aşağıda bir LUT örneği verilmiştir.

Aşağıdaki basitleştirilmiş şekle ek olarak: yerel bağlantıları es geçecek yüksek hızlı genel ara bağlantı yolları, başlıca G/Ç bacakları vardır.

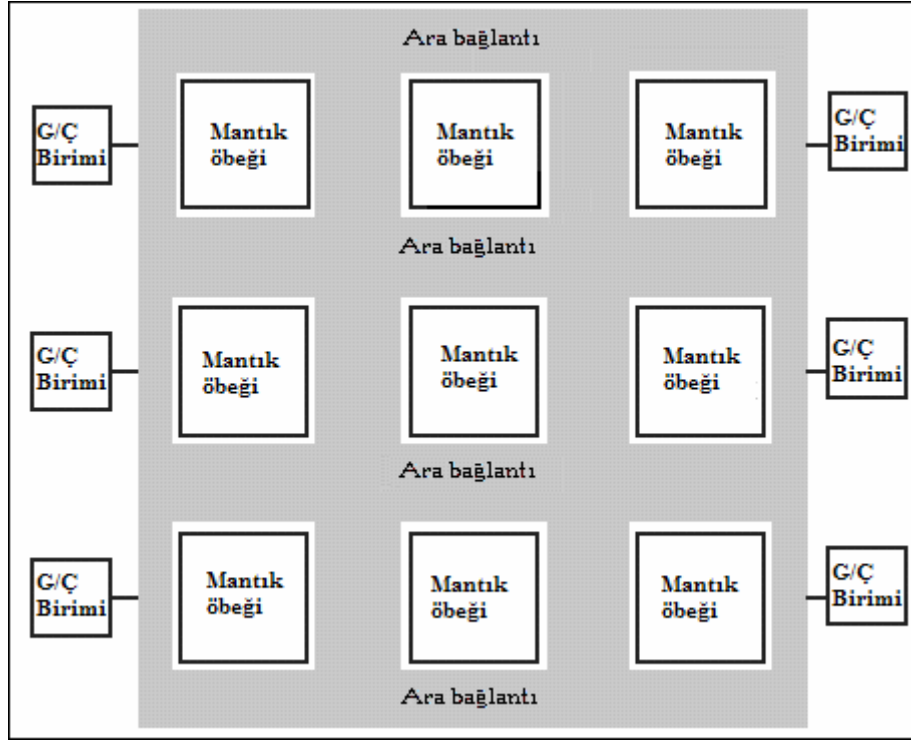


Şekil 3.16 : Genel FPGA mimarisi üstten görünümü

4. FPGA MİMARİSİ

Genel anlamıyla FPGA' lar üç ana birime gereksinim duyarlar:

- Birleşimsel mantık
- Ara bağlantı
- G/Ç bacakları



Şekil 4.1. : FPGA yongasının genel görünümü

4.1. Üretim Teknikleri

FPGA yongaları farklı teknolojileri baz alarak üretilir. Yonga üreticileri farklı tekniklerde uzmanlaşmışlardır.

4.1.1. SRAM temelli aygıtlar

FPGA' larda çoğunlukla SRAM temelli yapılandırma hücreleri yaklaşımı kullanılır. Bu tekniğin ana üstünlüğü yeni tasarım fikirlerin çabucak geliştirilebilir ve sınanabilir olmasıdır. Bu sayede standartlar ve protokoller geliştirirken daha kolay uzlaştırılabilir. Ayrıca düzeneğe ilk güç verildiğinde başlangıçta öz sına (Self Test), kart/sistem sınaması gibi

işlevleri gerçekleştirmesi için ve sonra ana görevi için programlanabilir. Yani sistem üzerinde programlanabilir.

SRAM temelli yaklaşımın bir diğer üstünlüğü, teknolojinin ön saflarında yer almasıdır. FPGA sağlayıcılarının talepleri neticesinde bellekler üzerinde uzmanlaşmış firmalar bu alanda araştırma ve geliştirme için (R&D) büyük kaynak harcamaktalar. Üstelik SRAM hücreler yongadaki diğer birimlerle tamamen aynı CMOS teknolojisi ile üretildiklerinden özel işlem basamağı gerektirmezler.

Olumsuz tarafı ise sistemin her açılışında aygıtın tekrar yapılandırılmak zorunda olmasıdır. Bu durum harici bellek gibi fazladan masraf ve alan teşkil eden birimler gerektirir. Bir diğer olumsuzluğu büyük yönlendirme gecikmesidir.

Tasarım açısından fikri mülkiyeti (Intellectual Property – IP) korumak zordur. Bunun nedeni aygıtı programlamak için gerekli olan yapılandırma dosyasının harici bellekte saklanmasıdır.

Günümüzde yapılandırma dosyasının içeriğini okuyup ilgili şematik veya netlist gösterimini üretecek ticari araçlar mevcut değildir. Ancak tecrübe ve bilgisayar yardımıyla bu sınır aşılabilmektedir.

Bazı günümüz FPGA' ları bit katarı şifreleme (bit stream encryption) kavramını destekler. Bu yöntemde nihai yapılandırma verisi harici belleğe depolanmadan önce şifrelenir. Şifreleme anahtarı FPGA içerisindeki özel SRAM temelli yazmaca JTAG bağlantı noktası üzerinden yüklenir. İlgili mantık birimleri ile birleşmesi durumunda bu anahtar şifrelenmiş bit katarının çözümlenmesine müsaade eder.

Şifre anahtarı için kullanılan yazmacın içeriğini sürdürmek devre kartı üzerinde yedek pil gerektirir. Bu pil yıllarca dayanabilir ama bu gereksinim kart üzerinde boyut, ağırlık ve maliyet gibi olumsuzluklar doğurur.

4.1.2. Antifuse temelli aygıtlar

SRAM temelli yaklaşımdan farklı olarak devre dışında özel programlayıcılar ile programlanır. Bu yaklaşımda veriler uçucu değildir. Sistem güçten kesildiği zamanda yapılandırma verilerini saklaması harici bellek gereksinimini ortadan kaldırır.

Bir diğer önemli üstünlüğü ara bağlantı yapısının doğal olarak ışınım etkisine görece bağışıklı olmasıdır. Bu durum askeri ve uzay uygulamalarında özel ilgi sebebidir. Çünkü SRAM temelli bileşenler ışınım maruz kalacak olursa hatalara yol açabilir. Antifuse bir kez programlandıktan sonra bu yolla değiştirilemez. Ancak aygıttaki herhangi bir flip-flop ışınım hassasiyetini sürdürür. Bu yüzden yoğun ışınımlı ortamlar için flip-floplar “triple redundancy design” yöntemi ile korunmalıdırlar. Bu yöntem her yazmacın üç kopyasının olması ve hata durumunda çoğunluktaki değer hatalıyı düzeltmesi şeklindedir.

Muhtemelen en önemli özelliği yapılandırma verisinin FPGA içerisine gömülmesidir. Her antifuse işlendiğinde, aygıt programlayıcı o ögenin tamamen programlandığının tespitine kadar sınamasını devam ettirir. Sonrasında sıradaki antifuse ögesine geçer. Üstelik aygıt programlayıcıları yapılandırmanın başarılı biçimde gerçekleştiğini özdevimli olarak doğrulayabilirler. Bu özellik milyonlarca programlanır öge içeren yongalar için önemlidir.

Sonradan herhangi programlama verisinin aygıttan okunmasını engellemek için özel güvenlik antifuse ögesi ayarlamak mümkündür. Programlanmış ve programlanmamış antifuse ögeleri özdeş görüldüğünden ve antifuse ögeleri iç metalleme katmanlarına gömülü olduklarından tersine mühendislik neredeyse olanaksızdır.

Antifuse ögesi SRAM temelli ögeye göre kendi başına daha az yer kaplıyor ve enerji harcıyor olmasına rağmen her öge için fazladan programlama devresi gerekir. Bu nedenle bu açılardan önemli fark yaratmaz. Yönlendirme gecikmesi daha küçüktür. Bu antifuse tekniğini SRAM’ e göre daha hızlı kılar.

Antifuse tekniği ana üretim sürecine ek birkaç basamak daha gerektirir. Teknolojisi SRAM temelli yaklaşımın birkaç nesil gerisindedir. Bu güç tüketimi ve hız gibi üstünlüklerini bertaraf eder. Temel kusuru bir kez programlanır (OTP) olmasıdır. Bu nedenle uygulama geliştirme için doğru tercih olmaz.

4.1.3. E²PROM/FLASH temelli aygıtlar

SRAM karřıtlarına benzer olarak yapılandırma hücreleri birbirlerine uzun ötelemeli yazmaç biçimli zincirler şeklinde bağlanmıřlardır. Bu yongalar aygıt programlayıcıları ile devre dıřında programlanabilir. Bazı çeřitleri devre içinde de programlanabilir (In System Programmable). Ancak programlama hızları SRAM temelli bileřenlere göre yaklaşık üç kat fazladır.

Programlandıktan sonra içerięi uçucu olmadığından sisteme ilk güç verildięinde hazır olacaktadırlar. Koruma amaçlı olarak bazıları 50 bit ile birkaç yüz bit genişliğinde olabilen “multibit key” kavramını kullanır. Yongayı programlarken kullanıcı tanımlı bit dizisi veri güvenliği amacıyla yüklenir. Aygıttan veriyi okumak veya yeni yapılandırmayı yüklemek için anahtarın kopyası JTAG bağlantı noktası ile aktarılmalıdır.

SRAM hücrelerine göre daha küçük olduklarından ara bağlantı gecikmeleri daha azdır. Ancak standard CMOS teknolojisine ilaveten yaklaşık beř işlem basamaęı gerektirdięinden SRAM temelli aygıtların birkaç nesil gerisinde kalır. Bununla beraber içerdikleri çok sayıda pull-up dirençlerinden dolayı görece yüksek duraęan güç tüketimi vardır.

4.1.4. Karma FLASH-SRAM aygıtlar

Her yapılandırma öęesi FLASH (veya E²PROM) hücre ve SRAM hücre birleřiminden oluşur. Kullanımı sınırlıdır.

Bu durumda FLASH hücre önceden programlanabilir. Sonrasında sisteme güç verildięinde FLASH içerięi ilişkili SRAM hücrelerine kopyalanır. Bu teknik antifuse aygıtlardaki kalıcılığı verir. Ancak antifuse temelli bileřenlerden farklı olarak sistemde yerleřik iken SRAM hücreleri tekrar yapılandırma için sonradan kullanılabiliriz. Dięer bir şekilde FLASH hücreler kullanılarak sistem dahilinde veya aygıt programlayıcı ile devre dıřında tekrar programlanabilir.

Çizelge 4.1. : Üretim tekniklerinin karşılaştırılması

Özellik	SRAM	Antifuse	EEPROM / FLASH
Teknoloji noktası	En gelişkin (State of the Art)	Birkaç nesil geride	Birkaç nesil geride
Tekrar Programlanırlık	Evet (Sistem içerisinde)	Hayır	Evet (Sistemde ve devre dışı)
Tekrar programlama hızı	Hızlı	- - -	SRAM' den 3 kat yavaş
Uçuculuk (başta programlanmalı)	Evet	Hayır	Hayır (gerekirse yapılabilir)
Harici program verisi gereksinimi	Evet	Hayır	Hayır
İlk örnek geliştirme	Evet (çok iyi)	Hayır	Evet (kabul edilebilir)
Başlangıçta çalışırlık	Hayır	Evet	Evet
IP güvenliği	Kabul edilebilir (özellikle bitstream şifreleme kullanırken)	Çok iyi	Çok iyi
Hücre genişliği	Geniş (6 transistör)	Çok küçük	Orta (2 transistör)
Güç tüketimi	Orta	Düşük	Orta
Işınım dayanırlığı	Hayır	Evet	Hayır

4.2. Birim Hücre Mimarileri

FPGA' ları diğer aygıtlardan ayıran ana özelliği ağırlıklı olarak programlanır ara bağlantılar içerisine gömülü, çok sayıda kısmen küçük programlanır mantık öbekleri içermeleridir.

İnce taneli (fine-grained) mimaride, her mantık öbeği sadece basit işlevler gerçeklemek için kullanılabilir. Örneğin bu ögeler ilkel mantık kapıları gibi (VE, VEYA, VE DEĞİL, vb.) veya depolama ögesi gibi (D tipi flip-flop, D tipi latch, vb.) herhangi 3 girişli işlev için yapılandırılabilir.

Birleştirici mantık (glue logic) ve durum makineleri gibi düzensiz yapılara ilaveten ince taneli mimari, paralel uygulamalardan faydalanan sistolik algoritmalar yürütürken özellikle verimli olurlar.

İri taneli(coarse-grained) mimaride, her mantık öbeği ince taneli yaklaşıma göre daha fazla mantık birimi içerir. Örneğin, bir mantık öbeği dört tane 4 girişli LUT, 4 adet MUX, 4 adet D tipi flip-flop ve bazı hızlı taşıma mantıkları içerebilir.

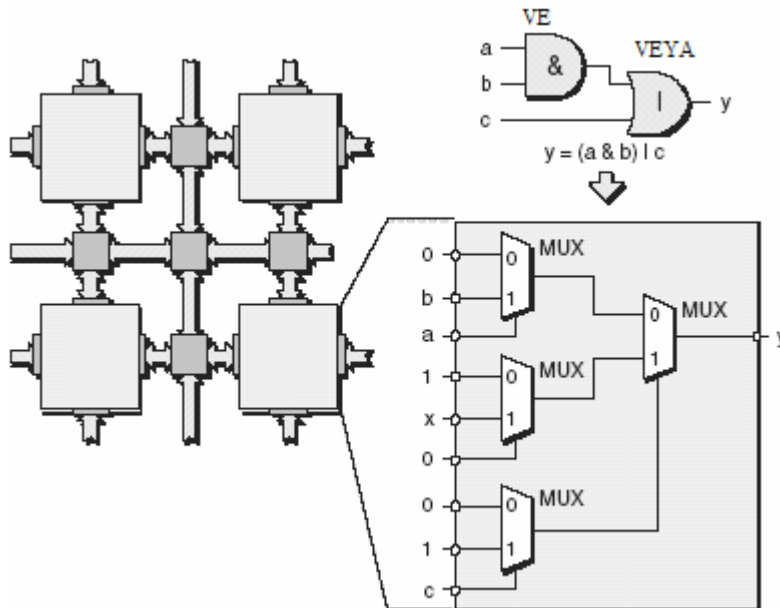
İnce taneli gerçeekte bu öbeklerce desteklenen işlevselliğe nazaran öbek içi ve dışında görece daha fazla sayıda bağlantı gerekir. Öbek boyutu arttıkça, öbek içerisindeki bağlantı sayısı desteklediği işlevselliğe kıyasla azalacaktır. FPGA boyunca yayılım gecikmesinde öbekler arası ara bağlantılar hesaba katıldığı için bu sonuç önemlidir.

Son zamanlarda bazı firmalarca geliştirilen iri taneli aygıtlar düğüm dizileri içermeye başladı. Bunlar hızlı fourier dönüşümü (FFT) gibi algoritmik işlevlerden genel amaçlı işlemci çekirdeklerine kadar yüksek derecede karmaşık işlem öğelerini içerir. Bu aygıtlar FPGA olarak sınıflandırılmasalar da bu alanda yer işgal ederler. Dolayısıyla LUT temelli FPGA mimari çoğu kez orta taneli (medium-grained) olarak sınıflandırılır. Neticesinde, iri taneli olarak adlandırma yeni düğüm temelli aygıtlar için kullanılmaya başlanmıştır.

İki temel programlanır mantık öbeği düzenlemesi vardır: MUX tabanlı ve LUT tabanlı

4.3.1. MUX tabanlı

Çoklayıcı yaklaşımına örnek olarak 3 girişli $y = (a \& b) | c$ işlevini MUX içeren öbek ile gerçekleştirilmesini inceleyelim.

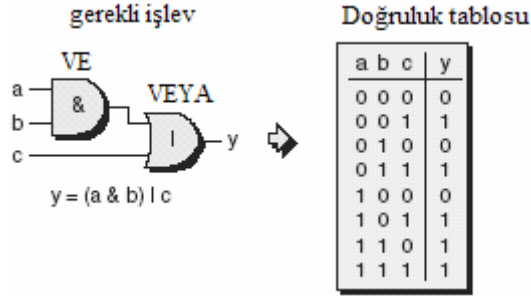


Şekil 4.2. : MUX tabanlı mantık bloğu

Aygıt mantıksal 0, mantıksal 1 veya diğer öbek ve asıl girişten gelebilecek sinyallerin asıl veya tümleyen değeri ile programlanabilir. Bu durum olası işlev için sayısız yol demektir. Yukarıdaki örnekte ortadaki MUX'ın girişindeki x ifadesi, girişin mantıksal 1 veya mantıksal 0 olmasının işlemi değiştirmeyeceğini belirtir. (x : don't care)

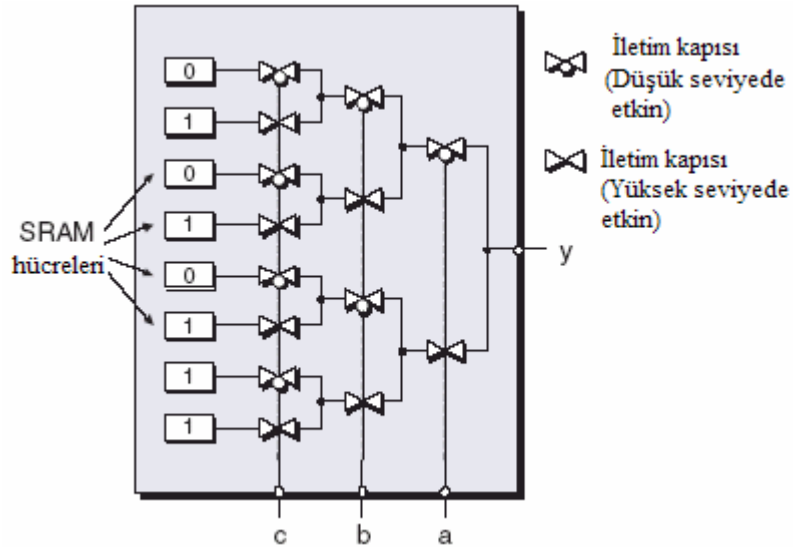
4.3.2. LUT tabanlı

Bir grup giriş sinyali başvuru çizelgesi (LookUp Table) için işaretçi olarak kullanılır. Çizelgenin içeriği, istenen değeri içeren her giriş birleşimini gösteren hücreler ile düzenlenir.



Şekil 4.3. : Gerekli işlev ve doğruluk tablosu

Bu işlev 3-girişli LUT' u uygun değerler ile yükleyerek yapılabilir. Örnek için LUT'ların SRAM hücrelerden oluştuğunu düşünelim. Kullanılan ana teknik istenen SRAM hücreleri seçmek için basamaklı iletim geçitlerini, girişleri kullanarak belirlemektir.



Şekil 4.4. : İletim kapısı yaklaşımı kullanılan sadeleştirilmiş LUT gösterimi

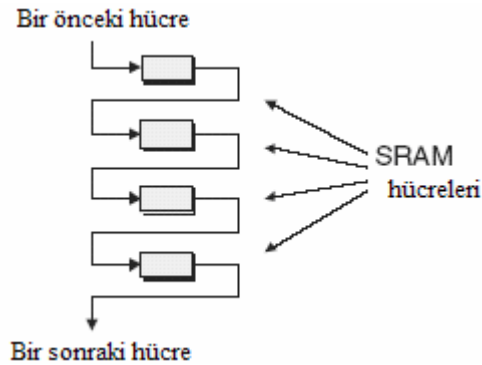
İletim geçidi etkin ise (active), sinyal girişinden çıkışına geçer. Etkin değil ise çıkışı sürdürdüğü tele elektriksel olarak bağlantısızdır.

N-girişli LUT ile n-girişli herhangi birleşimsel işlem gerçekleştirilebilir. Daha fazla kapı eklemek daha karmaşık işlemlere müsaade eder ama her giriş eklentisinde SRAM hücre sayısı ikiye katlanır. Hücre sayısı 2^n olarak ifade edilebilir. İlk FPGA'ler 3-girişli LUT tabanlı iken giriş sayısı 4, 5, 6 olarak arttı. Ancak mevcut kabul olan 4 girişli LUT ile kullanım en iyi durumda dengelenmektedir.

Birkaç kademe derinliğindeki mantık kapıları için LUT yaklaşımı kaynak kullanımı ve giriş-çıkış gecikmeleri açısından daha verimli olacaktır. Örnek işlevde bu kademe ikidir. Ancak tasarım içerisinde gerçekleştirilecek olan 2-girişli VE gibi küçük işlevlerde tüm LUT harcanacaktır. Dolayısıyla israf ve gereksiz gecikme olacaktır. Bu tarz küçük işlevleri çokça içeren tasarımlarda MUX tabanlı yaklaşım daha üstündür. Günümüzde FPGA mimarisi çoğunlukla LUT tabanlıdır.

LUT'ların SRAM tabanlı özünü neticesinde SRAM hücreler bazı enteresan olanaklar sağlar. Öncül davranışları başvuru çizelgesine ilaveten bazı satıcılarca LUT'ları oluşturan hücreler küçük RAM öbekleri olarak kullanılır. Örneğin, 4-girişli LUT'u şekillendiren 16 hücre 16x1 RAM olarak kullanılabilir. Bu "distributed RAM" olarak isimlendirilir. Çünkü LUT'lar yonga yüzeyine yayılmıştır.

Bütün yapılandırma hücreleri uzun zincirler halinde birlikte dizilmişlerdir. Bazı sağlayıcılar SRAM hücrelerin bağımsızca zincirin ana gövdesini oluşturacak şekilde kaydırmalı yazmaç olarak davranmalarını sağlar. Böylece her LUT çok amaçlı olarak kullanılabilir.



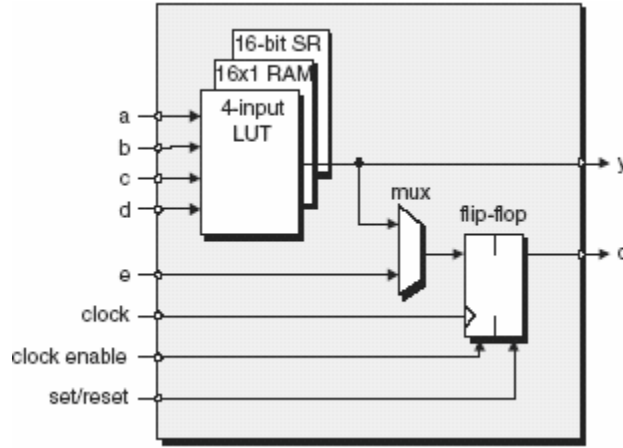
Şekil 4.5. : Zincir halinde bağlanmış yapılandırma hücreleri

4.4. Programların Mantık Öğeleri

Her programların öbek bir veya birkaç LUT' a ilaveten çoklayıcı ve yazmaç gibi öğeler içerir. Her üretici farklı isimler kullanır. Bunlara aşağıda değinilmiştir.

4.4.1.1. Logic Cell

Xilinx, çağcıl FPGA örneklerinde yapı taşını "Logic Cell (LC)" olarak isimlendirir. LC, 4-girişli LUT, çoklayıcı ve yazmaç içerir. LUT aynı zamanda 16x1 RAM ve 16 bitlik kaydırmalı yazmaç olarak davranabilir. Bunlara ek olarak aritmetik işlemler için özel hızlı elde mantık yapıları içerir.



Şekil 4.6. : Sadeleştirilmiş Xilinx LC

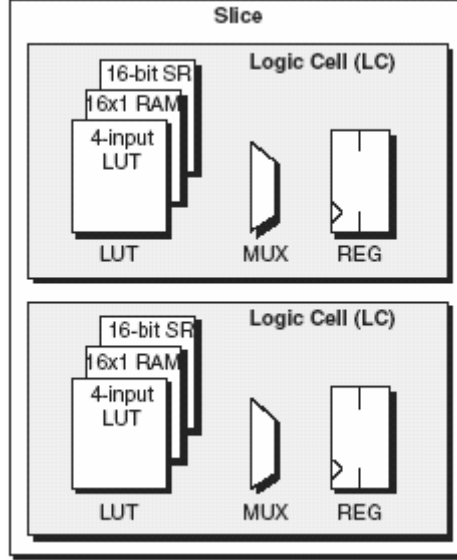
Yazmaç flip-flop veya latch olacak şekilde seçilebilir. Clock, düşen veya yükselen kenar tetikleme için ayarlanabilir. Clock enable ve set/reset sinyalleri vardır.

4.4.1.2. Logic Element ve Adaptive Logic Module

Altera, FPGA' larında yapı taşını "Logic Element (LE)" olarak isimlendirir. Stratix ailesiyle birlikte bu isimlendirme "Adaptive Logic Module (ALM)" olarak değişmiştir. Bazı farklılıklarla birlikte tüm yaklaşım LC ile çok benzerdir.

4.4.2. Slice

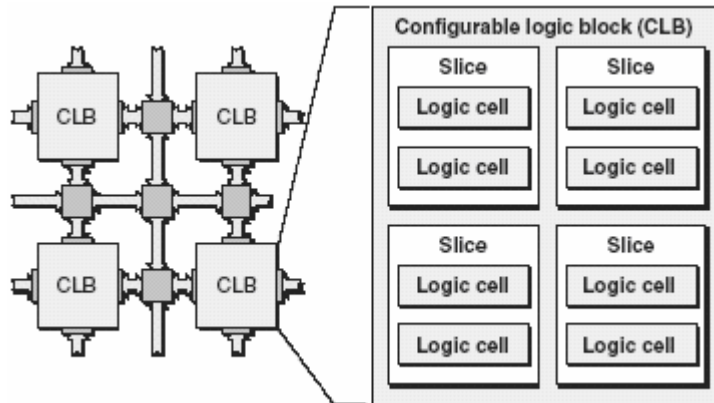
Hiyerarşi de bir sonraki basamak Xilinx firmasınınca “slice” olarak isimlendirilir. Her slice 2 LUT içerir. Her logic cell’ in LUT, MUX ve yazmacı kendi giriş ve çıkışlarına sahiptir. Ancak clock, clock enable ve set/reset sinyalleri her iki LC için ortaktır.



Şekil 4.7. : İki LC içeren slice görünümü

4.4.3. CLB ve LAB

Bir üst basamakta Xilinx’ in “Configurable Logic Block (CLB)” ve Altera’ nın “Logic Array Block (LAB)” olarak isimlendirdikleri yapılar vardır.



Şekil 4.8. : Dört slice içeren CLB görünümü

Her CLB aygıtına göre iki veya dört slice içerir. CLB' ler arasında hızlı programlanır ara bağlantılar vardır. Bunlar komşu slice' ları birleştirmek için kullanılır. En hızlı ara bağlantı slice içerisindeki LC' ler arasındadır. CLB içerisindeki slice' lar arasındaki ara bağlantı daha yavaştır. Bunu CLB' ler arasındaki ara bağlantı izler.

Daha ayrıntılı açıklama için Stratix II ailesini baz alırsak; her LAB sekiz ALM, elde zincirleri, paylaşılmış aritmetik zincirleri, LAB kontrol sinyalleri, yerel ara bağlantıları ve yazmaç zincir bağlantı hatları içerir.

4.4.3.1. Ara bağlantılar

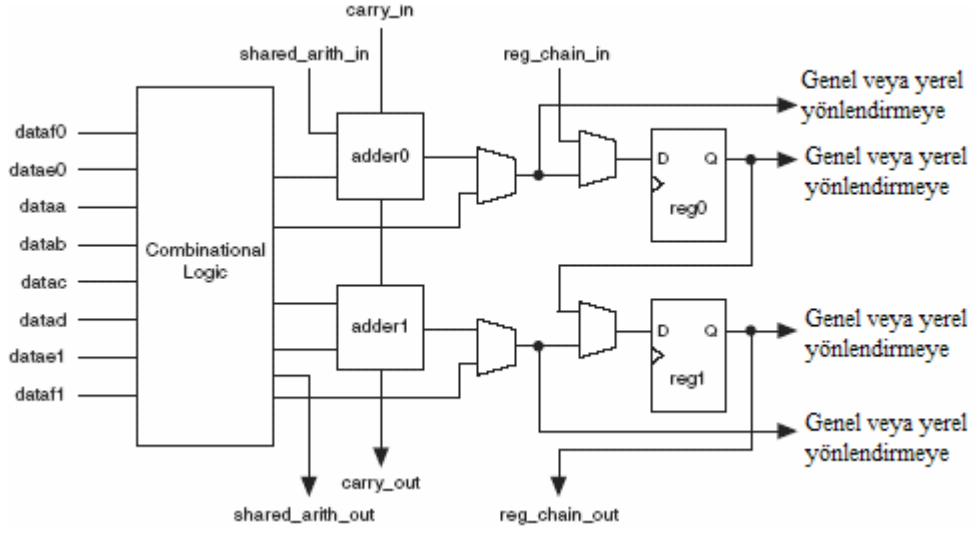
LAB yerel ara bağlantıları aynı LAB içerisindeki ALM' leri sürebilir. Bu yerel bağlantılar satır-sütun ara bağlantılarınca ve aynı LAB içerisindeki ALM çıkışlarınca sürülebilir.

4.4.3.2. Kontrol Sinyalleri

Kontrol sinyalleri saat, saat etkinleştirme, eşzamanlı ve eşzamansız temizleme, eşzamansız önceden ayarlama/yükleme ve eşzamanlı yükleme kontrolü sinyallerini içerir.

4.4.3.3. Adaptive Logic Module

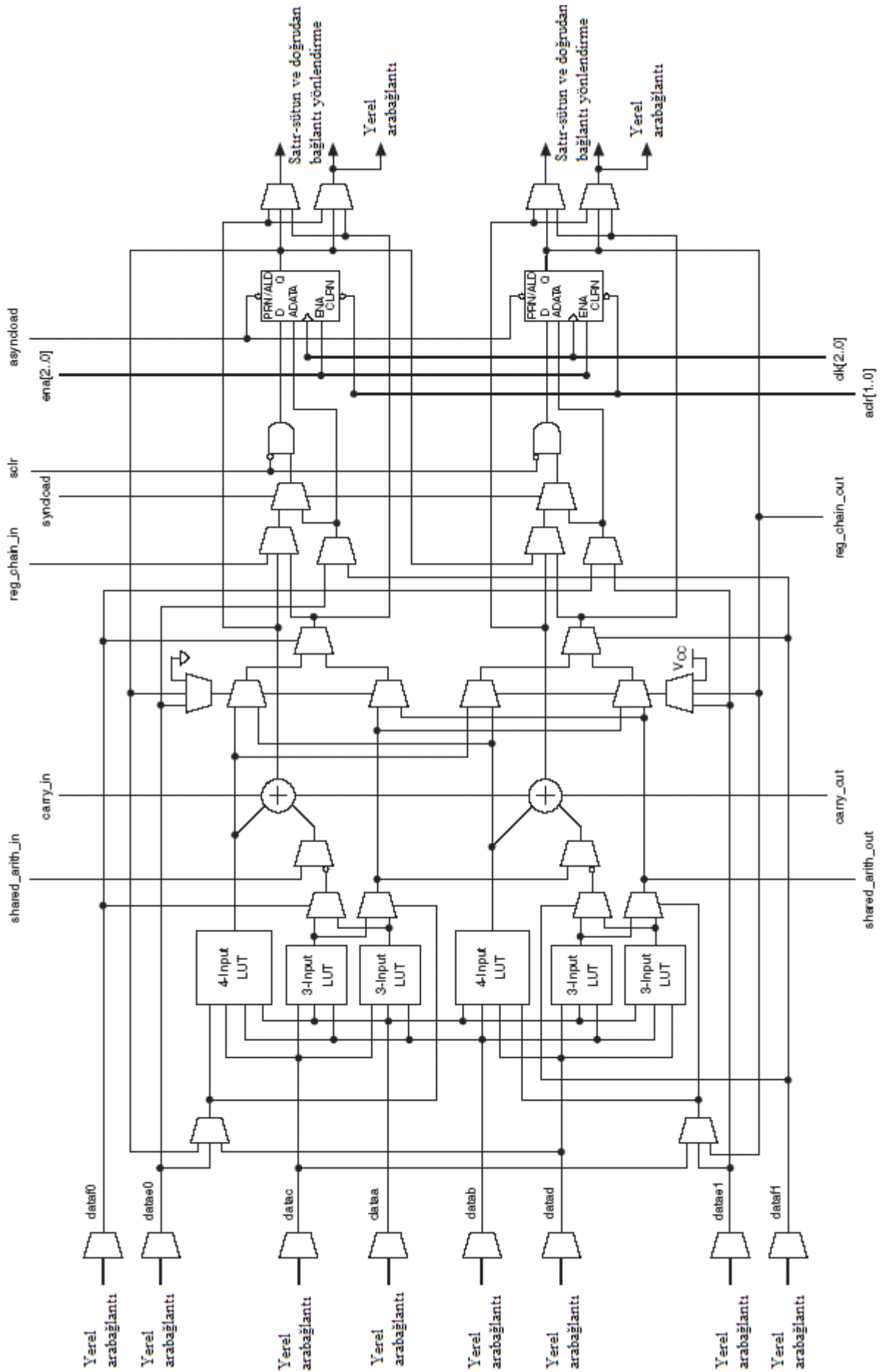
Uyarlanabilir LUT kaynakları, iki programlanır yazmaç, iki adanmış tam toplayıcı, elde zinciri, paylaşılmış aritmetik zinciri ve yazmaç zinciri içerir. Her ALM yerel, satır-sütun, elde zinciri, paylaşılmış aritmetik zinciri, yazmaç zinciri ve doğrudan bağlantı ara bağlantılarını sürebilir.



Şekil 4.9. :Stratix II ALM yüzeysel görünümü

Yazmaç saatini ve temizleme kontrol sinyallerini genel sinyaller, genel amaçlı G/Ç bacakları veya herhangi içsel mantık düzenlemesi sürebilir. Genel amaçlı G/Ç bacakları veya herhangi içsel mantık düzenlemesi saat etkinleştirme, ön ayarlama, eşzamansız yükleme ve eşzamansız veri yükleme bağlantılarını sürebilir. Eşzamansız veri yükleme girişi ALM' nin yazmaç paketleme için de kullanılan datae veya dataf girişlerinden gelir. Birleşimsel işlevler için yazmaçlar atlanır ve LUT çıkışı doğrudan ALM çıkışlarını sürer.

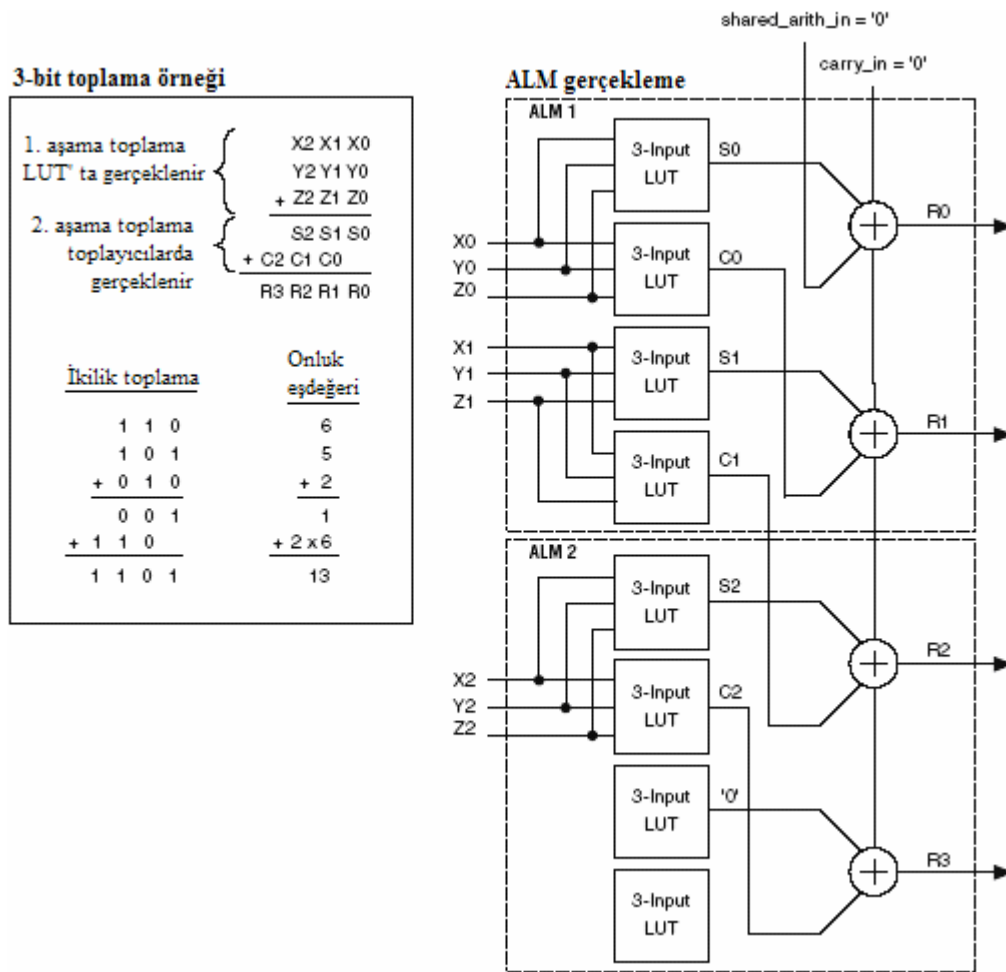
Diğer sayfada Stratix II ailesinin ALM öbeği ayrıntılı olarak verilmiştir.



Şekil 4.10. : Altera Stratix II - ALM ayrıntılı görünümü

Her ALM yerel ve satır-sütun yönlendirme kaynaklarını süren iki takım çıkışa sahiptir. LUT, toplayıcı ve yazmaç çıkışları bu çıkışları birbirlerini etkilemeden sürebilir. Bu sayede LUT ve toplayıcı bir çıkışı sürerken yazmaç diğer çıkışı sürebilir. “Register packing” olarak isimlendirilen bu özellik sayesinde aygıtın kullanımı artar. Çünkü yazmaç ve birleşimsel mantık birimi ilişiksiz işlevler için kullanılabilir.

Stratix II ALM şu kiplerden birinde işleyebilir: Normal kip, Genişletilmiş LUT kipi, Aritmetik kip, Paylaşılmış Aritmetik kip



Şekil 4.11. : Paylaşılmış aritmetik kipi kullanarak 3-bit toplama

4.4.3.4. Shared Arithmetic Chain

Paylaşılmış aritmetik kipte, elde zinciri yönlendirmesine ek olarak üç giriшли toplamayı gerçekleştirmek için paylaşılmış aritmetik zinciri kullanılır.

4.4.4. Distributed RAM

4-girişli LUT göz önüne alırsak, 16 SRAM hücresi gerekecektir. Her CLB şu şekillerde yapılandırılabilir:

Single-port RAM : 16x8 bit RAM, 32x4 bit RAM, 64x2 bit RAM, 128x1 bit RAM

Dual-port RAM : 16x4 bit RAM, 32x2 bit RAM, 64x1 bit RAM

Tek girişli RAM' de okuma ve yazma işlemi için ortak veri yolu kullanılır. Çift girişli RAM' de ise okuma ve yazma için ayrı veri yolları vardır. Dolayısıyla okuma ve yazma eş zamanlı olarak gerçekleştirilebilir.

4.4.5. Shift Register

4 girişli LUT 16 bitlik kaydırmalı yazmaç olarak kullanılabilir. Slice' da LC' ler arasındaki ve slice' lar arasındaki özel bağlantıları sayesinde bir kaydırmalı yazmaçın son biti bir diğerin ilk bitine bağlanabilir. Bu sayede 128 bitlik kaydırmalı yazmaç gerçekleştirilebilir.

4.4.6. Fast Carry Chain

Çağcıl FPGA' larda hızlı elde zincirlerini gerçeklemek için özel mantık işlevleri ve ara bağlantılar vardır. Her LC özel elde mantık işlevi içerir. Her slice' da LC' ler, CLB' de slice' lar ve CLB' ler arasında ara bağlantılar ile ilişki kurulmuştur. Bu birim, sayıcılar ve aritmetik işlevler gibi mantıksal işlevlerinin başarımını artırır.

4.5. Embedded RAM

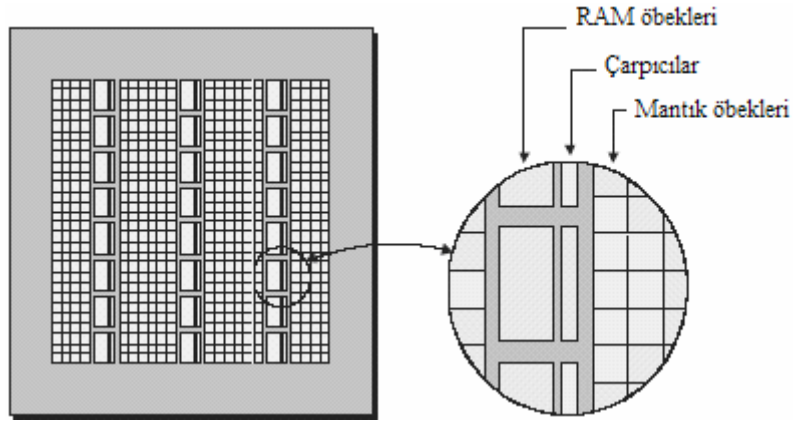
Günümüz FPGA' ları gereksinimler neticesinde büyük yığınlar halinde gömülü RAM içerirler. Bunlar "e-RAM" veya "block RAM" olarak isimlendirilir. Mimariye göre bu öbekler yonganın dış kenarlarına, yüzeyine dağıtılmış şekilde veya sütunlar halinde olabilir.

Her yonga bu öbeklerden yüzlerce içerebilir. Toplam kapasite birkaç milyon biti bulabilir. Her RAM öbeği bağımsız veya birlikte daha büyük öbekler halinde kullanılabilir.

Bu öbekler tek ve çift girişli RAM, FIFO işlevleri ve durum makinelerini gerçeklemek gibi değişik amaçlarla kullanılır.

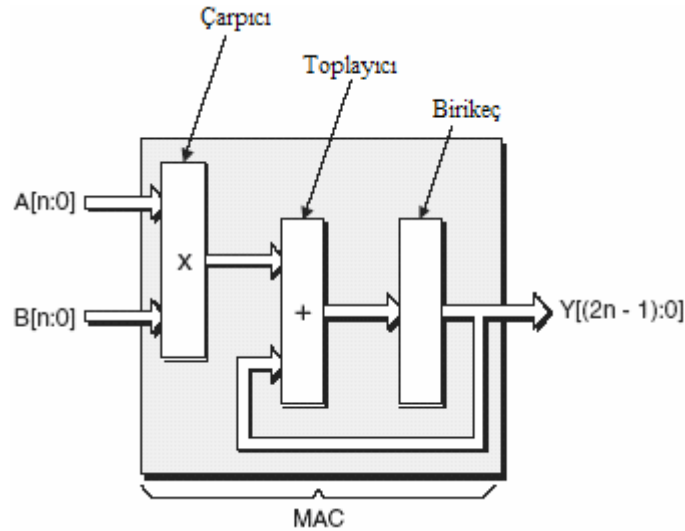
4.6. Embedded Multiplier, Adder, MAC

Çarpma gibi işlevler programlanır öbekler ile yapılacak olursa kendiliğinden yavaş olacaktır. Birçok uygulama bu tarz işlevleri gerektirdiğinden çoğu FPGA özel dahili çarpma öbekleri içerir. Bunlar özellikle gömülü RAM öbeklerine yakın yerleştirilir.



Şekil 4.12. : Yongaya gömülü çarpıcı ve RAM öbek sütunlarının kuş bakışı görünümü

Benzer olarak bazı FPGA' ler özel toplayıcı öbekleri içerirler. DSP tarzı uygulamalarda "multiply and accumulate (MAC)" olarak isimlendirilen işlem çok yaygındır.



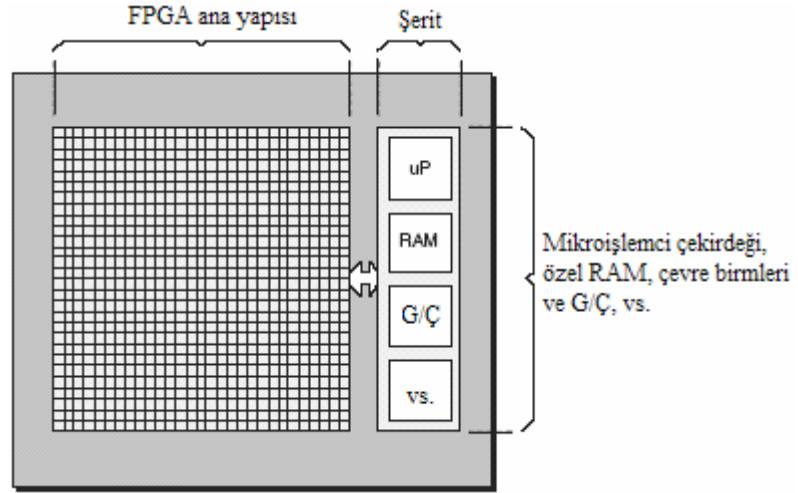
Şekil 4.13. : MAC işlevinin gösterimi

Bazı FPGA' lar çarpan öbeğine ek olarak toplayıcı öbeğini veyahut her ikisini de içeren MAC öbeğini dahili olarak sunmaktadırlar.

4.7. Embedded processor core

Çoğu tasarım değişik amaçlarla mikroişlemci kullanır. Yakın zamana kadar bu işlemleri kart üzerinde ayrı devre ile sağlanıyordu. Son zamanlarda üst uç FPGA' ler “microprocessor core” olarak isimlendirilen gömülü mikroşlemciler içermeye başladı. Bu sayede kart maliyeti, boyutu ve karmaşıklığı azaldı.

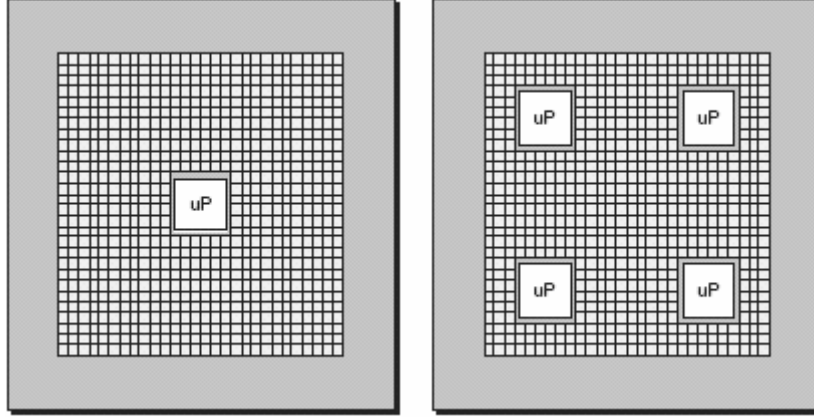
“Hard microprocessor core” önceden tanımlanmış dahili öbektir. Bu tarz çekirdekler iki ana yaklaşım ile gerçekleştirilir. İlki çekirdek birimini kenarda şerit üzerine yerleştirme şeklindedir.



Şekil 4.14. : Ana yapı dışındaki gömülü çekirdek görünümü

Bu gerçeğin bir üstünlüğü üretim sürecinin özdeş olmasıdır. Diğer bir üstünlüğü, bu şerit üzerine mikroşlemci çekirdeğini tamamlayıcı birimler eklenebilir.

Bir diğer mimari yaklaşımı, çekirdeklerin doğrudan FPGA yerleşimi içerisine gömülmesidir. Bir, iki ve dört çekirdekli tasarım günümüzde mümkündür.



Şekil 4.15. : Ana yapı içerisinde bir ve dört gömülü çekirdek

Tasarım araçlarınca bu çekirdeklerin varlığı hesaba katılmalıdır. Kullanılacak olan bellek gömülü RAM öbeklerince ve çevresel işlevler genel amaçlı programların birimlerce oluşturulmalıdır. Bu yaklaşım yan şerit yaklaşımına göre biraz daha hız üstünlüğü sağlar.

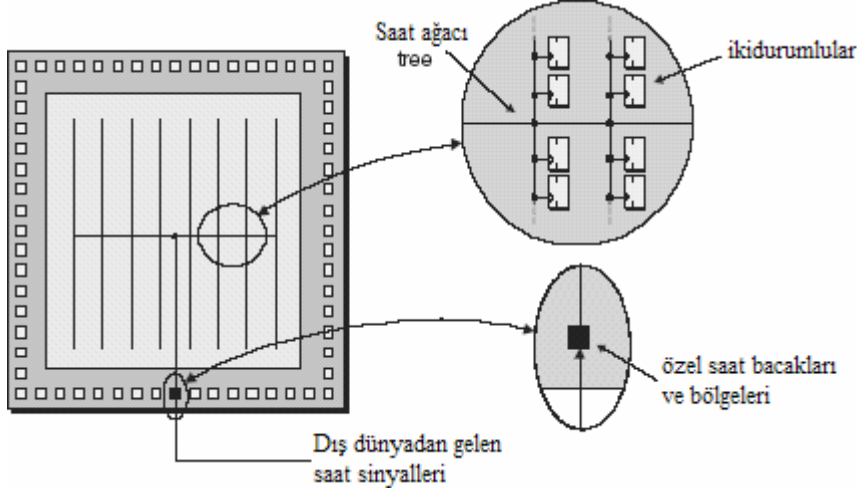
Fiziksel gerçekleştirme yerine, programların mantık öbekleri mikroişlemci çekirdeği gibi davranacak şekilde yapılandırılabilir. “Soft microprocessor core” olarak isimlendirilirler.

Yapılandırılan çekirdekler sabit yaklaşıma göre daha basit ve yavaşlardır. Bu hız düşümü %30–50 arasındadır. Ancak gerektiğinde gerçekleşmesi ve ilgili programların öbekler tükeninceye kadar çoğaltılabilmesi sağladığı üstünlüklerdir.

4.8. Clock Tree

FPGA içerisindeki tüm anuyumlu öğelerin saat sinyalinde sürülmeye gereksinimi vardır. Saat sinyali genellikle dışarıdan sağlanır ve özel saat giriş bacağından FPGA’ e girer. Sonrasında ilgili yazmaçlara yonga boyunca yönlendirilir.

Aşağıda basitleştirilmiş gösterim ile saat hattı yayılımı verilmektedir. Bu ağ “clock tree” olarak isimlendirilir. Şekilde de görüleceği üzere ikidurumlular ağacının yaprakları gibidir.



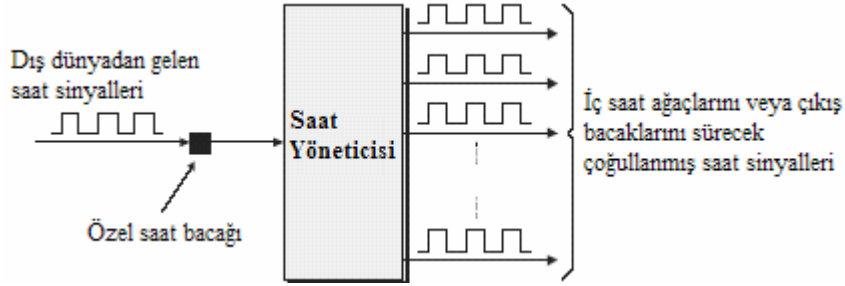
Şekil 4.16. : Basit saat ağacı

Bu yapı bütün ikidurumluların mümkün olduğu kadar uyumlu şekilde aynı saat sinyalini görmelerinden emin olmak için kullanılır. Zincir dizilimi kullanılacak olsaydı saat giriş bacağına en yakın olan, zincirin sonundaki ikidurumludan epey öce saat sinyalini alacaktı. Bu çarpıklık istenmeyen sorunlara neden olacaktı. Ağaç yapısında çarpıklık çok daha azdır.

Saat ağacı özel yollarca gerçekleştirilmiştir ve genel amaçlı programlanır ara bağlantılardan ayrıdır. Yukarıdaki benzetim fazlasıyla sadeleştirilmiştir. Gerçekte birden fazla saat bacağı mümkündür ve yongada birçok saat ağacı vardır.

4.9. Clock Manager

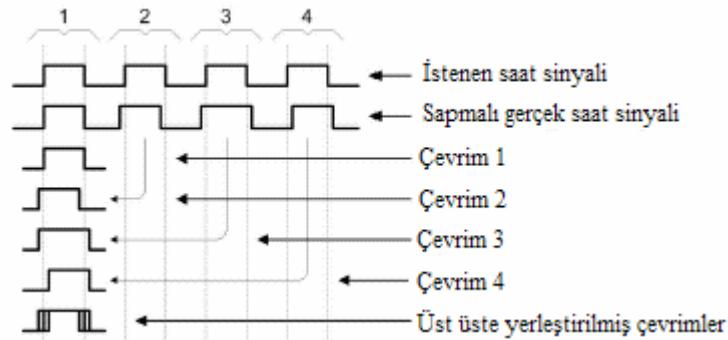
Saat giriş bacağı doğrudan iç saat ağacına bağlamak yerine, bu girişler “clock manager” adı verilen özel fiziksel bağlantılı birimleri sürmek için kullanılabilir. Bu saat yöneticileri, saat sinyallerini çoğaltırlar. Yapılan düzenleme ile çoğullanmış sinyalleri, dahili saat ağaçlarına veya diğer aygıtlara sunmak üzere harici çıkış bacaklarına iletirler.



Şekil 4.17. : Saat sinyallerini çoğullayan saat yöneticisi

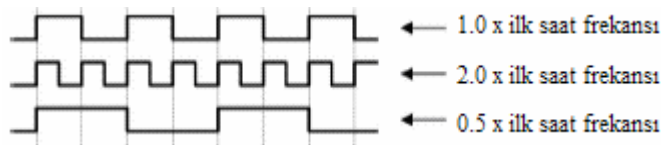
Her FPGA ailesi kendine has saat yönetici türüne sahiptir. Yonga içerisinde sayıları değişebilir ve türüne göre aşağıdaki özellikleri sağlayabilirler:

Zaman sapması (jitter) kaldırma: Saat sinyali köşeleri gerçekte olması gerektiğinden biraz erken veya daha geç gelebilir. Saat yöneticisi, zaman sapmalarını algılamak ve düzeltmek için kullanılabilir.



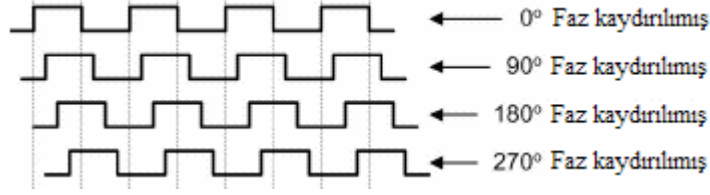
Şekil 4.18. : Bulanık saat ile sonuçlanan sapmalar

Frekans sentezi: Sağlanan saat sinyali frekansı tam olarak tasarım mühendisinin arzu ettiği gibi olmayabilir. Bu durumda, saat yöneticisi asıl sinyalden çarpma ve bölme ile gerekli sıklıktaki sinyali türetebilir.



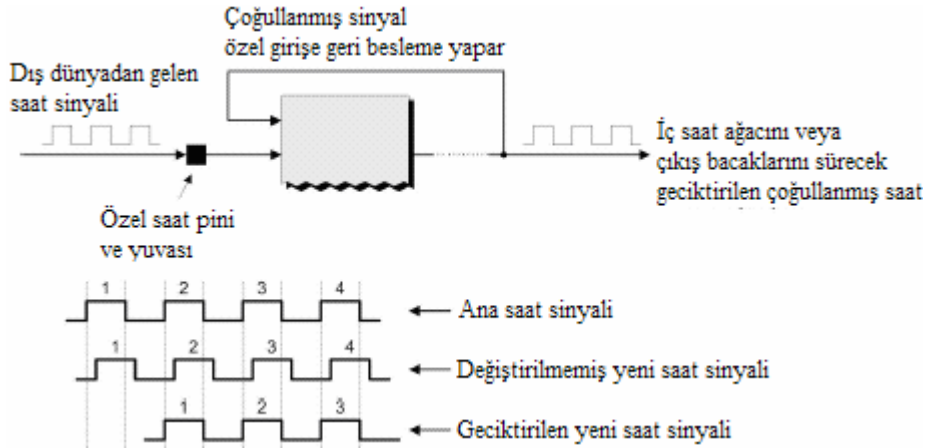
Şekil 4.19. : Frekans sentezi için saat yöneticisinin kullanımı

Faz kaydırma: Bazı tasarımlar diğerlerine göre kaydırılmış (geciktirilmiş) saat sinyalleri kullanımı gerektirir. Kimi saat yöneticileri üç evreye, kimisi dört evreye, kimisi de tercihe göre değiştirilebilir sayıda evreye izin verir.



Şekil 4.20. : Çoğullanmış sinyallerde faz kaydırılması

Özdevimli Çarpıklık düzeltme: Ana saat sinyali ile aynı frekans ve fazda olan çıkış saat sinyalini düşünelim. Saat yöneticisi gerçekleştirdiği işlemler sonucunda sinyale bazı gecikme unsurları ekleyecektir. Bununla birlikte, sürülen kapılar ve saat yayılımı için kullanılan bağlantılarca daha önemli gecikmeler eklenecektir. Ana sinyal ve çıkış sinyalinin nasıl kullanılacağına bağlı olarak FPGA' da veya devre kartında bundan dolayı çeşitli hatalar meydana gelecektir. Bu yüzden, saat yöneticisi çıkış sinyalini beslemek için özel girişe sahiptir. Saat yöneticisi iki sinyali karşılaştırarak, çıkış saatine yeteri kadar gecikme ekler ve ana saat ile hizalar. Bu işlem ilk çıkış saat sinyaline uygulanır. Diğer çıkışlara ilkinde göre faz hizalaması yapılır.



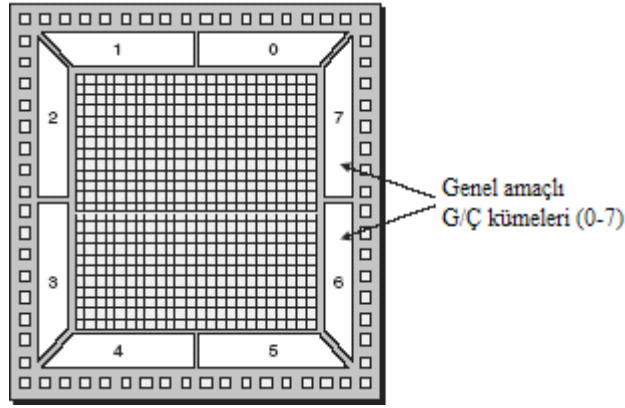
Şekil 4.21. : Geciktirme yapılarak ana saat ile uyumlandırma

Bazı FPGA saat yöneticileri PLL (Phase Locked Loop), diğerleri DLL (Digital Delay Locked Loop) tekniğine dayanır. Faz kilitli döngü örneksel ve sayısal teknikler ile DLL sadece sayısal teknik ile gerçekleştirilebilir. Sayısal gecikme yöntemi duyarlılık, kararlılık, güç yönetimi, gürültü duyarsızlığı ve zaman sapması başarımları gibi üstünlükler sağlar.

4.10. Genel Amaçlı G/Ç

Bugünün FPGA paketleri 1000 veya daha fazla bacağına sahip olabilirler. Bunlar paketin tabanı boyunca sıralı şekilde düzenlenirler.

Gereksinime göre tasarımda farklı elektriksel standartlar kullanılır. Bu zorluğu aşmak için FPGA' ların genel amaçlı G/Ç birimleri gereken standarda uyacak şekilde sinyal alma ve üretme için yapılandırılabilir. Bu genel amaçlı G/Ç sinyalleri belli kümelerle bölünecektir.



Şekil 4.22. : Genel amaçlı G/Ç kümeleri

Her küme, özel G/Ç standardını desteklemek için ayrı ayrı yapılandırılabilir. Bu özellik, çoklu standart ile çalışmaya, FPGA' ları farklı G/Ç standartları arasında ara yüz olarak kullanmaya izin verir.

4.11. Yapılandırılabilir G/Ç Özdirençleri

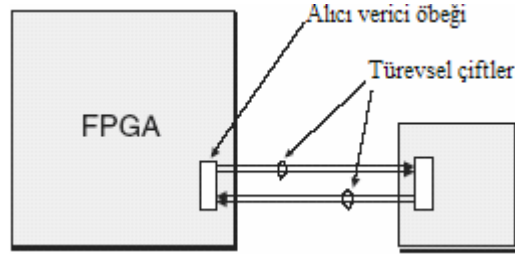
Günümüz hızlı sistemlerinde sinyallerin geri yansımalarını önlemek için FPGA' nın giriş ve çıkış bacaklarına uygun sonlandırıcı dirençler bağlanmalıdır.

Geçmişte bu dirençler ayrı bileşenler olarak devre kartı üzerine ekleniyordu. Bacak sayısı arttıkça ve aralarındaki mesafe daraldıkça sorunlar çıktı. Bu sebepten ötürü günümüz FPGA' ları dahili sonlandırma dirençleri kullanırlar. Bu dirençlerin değerleri, farklı çevresel birimleri ve G/Ç standartlarını uzlaştırmak için yapılandırılabilir.

4.12. Gigabit Alıcı Vericiler

Geleneksel yöntemlerde aygıtlar arası veri aktarımı için “bus” adı verilen yollar kullanılır. Artan talep fazla yol demektir. Bu durum, kullanılan bacak sayısını arttıracaktır. Bu yolları biçimlendirme kartı daha karmaşık hale getirecektir. Üstelik sinyallerin doğruluk sorunlarını yönetmek gittikçe zorlaşacaktır.

Bu sebeplerden ötürü günümüz FPGA’ ları özel gigabit alıcı verici öbekleri içerirler. Bu öbekler iletim ve alma için birer çift türevsel sinyal kullanır. Yani sinyal mantıksal tersi ile aktarılmaktadır.



Şekil 4.23. : Yüksek hızlı alıcı verici kullanımı

Bu alıcı vericiler saniyede milyarlarca bit aktarımı sağlayacak kadar yüksek hızlarda işlerler. Üstelik FGPA birden fazla bu alıcı verici öbeği içerebilir.

4.13. IP (Intellectual Property) Core

Fikri mülkiyet çekirdekleri mantık tasarımında kullanılan, önceden test edilmiş çok karmaşık sistem seviyesindeki işlevlerdir. IP çekirdeği şu yararları sağlar: pazara daha hızlı sunum, geliştirme sürecinde sadelik ve kısalma, tasarım riski azalır, yazılım derleme işlemleri azaltır, doğrulama zamanı küçülür, tahmin edilebilir verim ve işlevsellik sağlar.

Donanımsal IP, önceden gerçekleştirilmiş mikroişlemci çekirdeği, gigabit alıcı verici, çarpıcı, toplayıcı, MAC birimi gibi öbekleri kapsar. Bu öbekler güç tüketimi, alan kullanımı ve başarımlarından daha verimli olmayı sağlar.

Yazılımsal IP, yüksek seviyeli işlevler için hazırlanmış kaynak kütüphanelerinin kullanıcı tarafından tasarıma eklenmesini tanımlar. Bu kütüphaneler PCI yolu arayüzü, DSP filtresi, PCMCIA arayüzü gibi karmaşık sistem seviyeli işlevler içerir.

4.14. Ara Bağlantı Mimarisi

FGPA' larda programlanır mantık öbekleri arasındaki iletişim ve bu programlanır öbeklerin istediğimiz işlev doğrultusunda yapılandırılması ara bağlantılar sayesinde gerçekleşir. Bir FPGA aygıtı içerisinde ara bağlantıların kapladığı alan %80' leri bulmaktadır. Dolayısıyla üreticiler bu büyük kütlenin en verimli şekilde yerleşimi için büyük çaba sarf etmektedirler. En büyük iki üreticiyi baz alacak olursak: Xilinx ve Altera' nın kullandıkları bağlantı yaklaşımları birbirlerinden farklıdır.

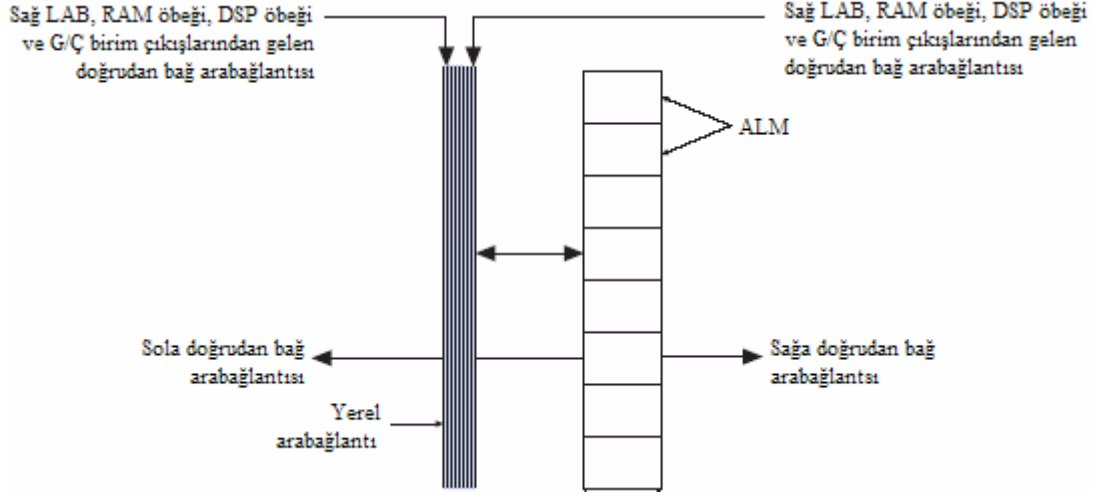
4.14.1 Altera Ara Bağlantı Yaklaşımı

ALM (veya LE), bellek öbekleri, G/Ç bacakları ve varsa DSP öbekleri arasındaki bağlantıyı "MultiTrack" bağlantı yapısı olarak isimlendirdiği, "DirectDrive™" teknolojisi ile yapar. Multitrack bağlantılar, öbekler içi ve arasında sürekli ve başarıma göre yapılandırılan farklı uzunluk ve hızdaki yönlendirme hatlarını içerir.

MultiTrack ara bağlantılar yayılmış satır ve sütun bağlantıları içerir. Stratix II ailesi için satır bağlantıları şu kaynakları içerir:

- LAB' lar ve bitişik öbekler arasındaki "Direct link" ara bağlantıları.
- Sağa ve sola dört öbek kateden R4 ara bağlantıları
- Aygıt boyunca yüksek hızlı R24 bağlantıları

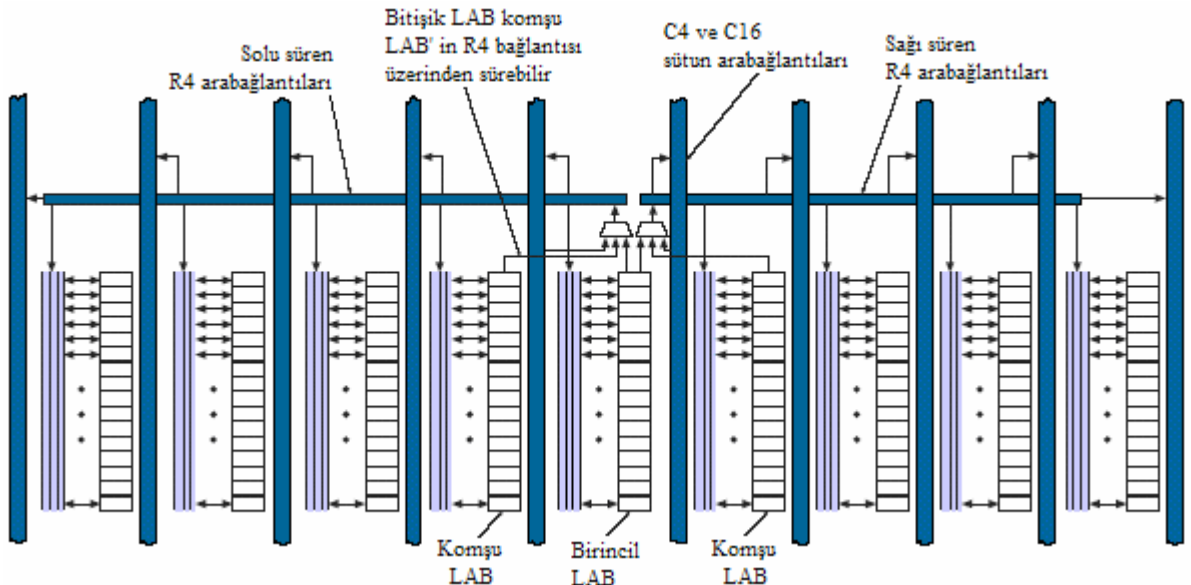
Sütun bağlantılarını kullanmadan yapılan "Direct link" hattı karşılıklı iletişim hızını artırır. Doğrudan bağ arabağlantıları, LAB' ların, RAM veya DSP öbeklerinin sağ ve sol komşularını yerel arabağlantıları ile karşılıklı hızlıca sürmesini sağlar.



Şekil 4.24. : Doğrudan bağ bağlantısı

Her LAB sağını veya solunu sürmek için kendi R4 bağlantı takımına sahiptir. R4 bağlantıları DSP ve RAM öbeklerince, satır G/Ç birimlerin sürülebilir ve R4 bağlantıları da onları sürer. R4 bağlantıları diğer R4 bağlantılarını, R24 bağlantılarını ve diğer satırla bağlantı için de C4 ve C16 bağlantılarını sürer.

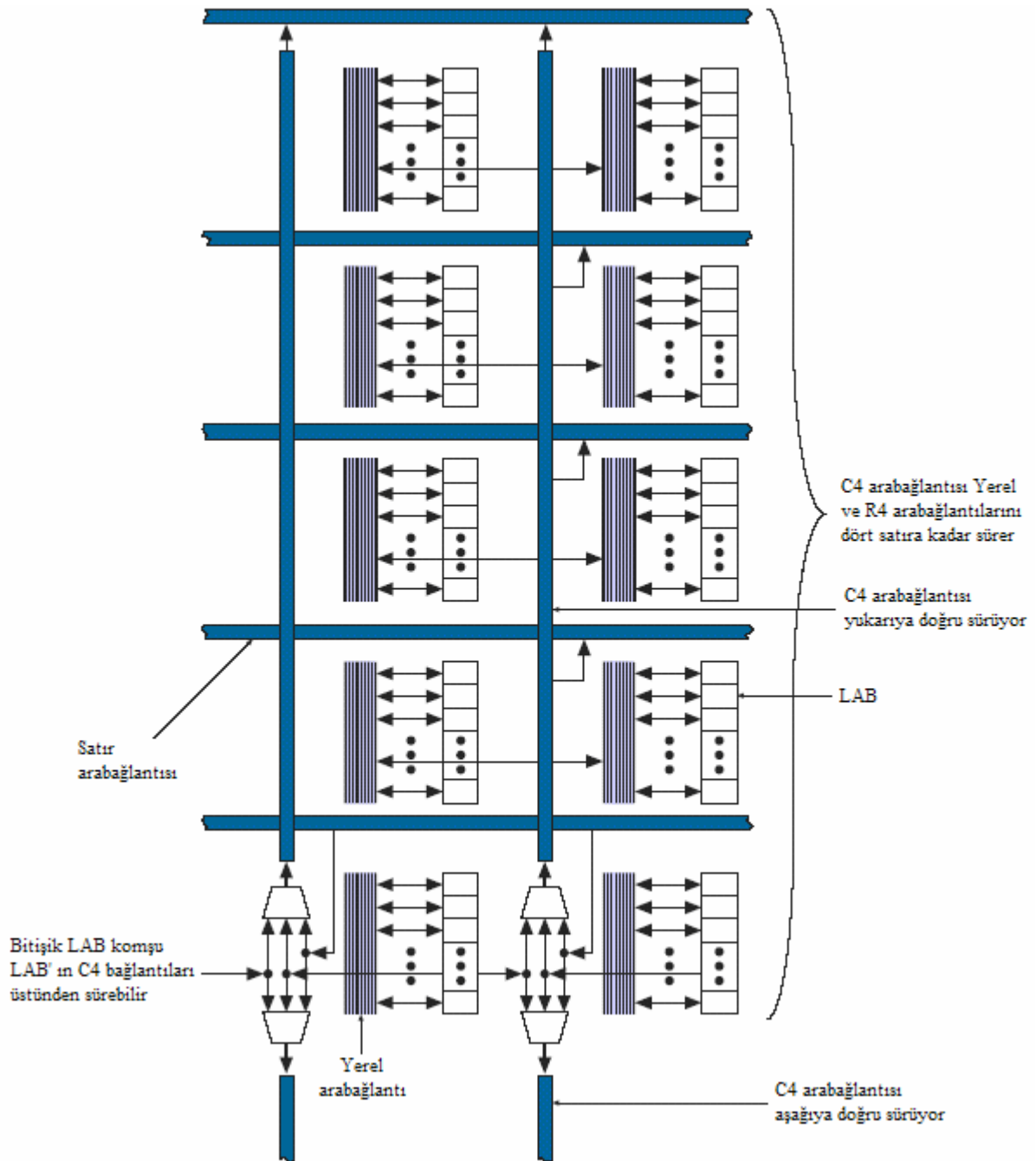
R24 satır bağlantısı 24 LAB' a yayılır ve LAB' lar, bellek öbekleri, DSP öbekleri ve satır G/Ç birimleri arasında en hızlı, uzun satır bağlantısını sağlar. Doğrudan LAB yerel bağlantılarını sürmez. Bunu R4 ve C4 bağlantıları üzerinden gerçekleştirir. R24 bağlantıları diğer R24, R4, C4, C16 bağlantılarını sürer.



Şekil 4.25. : R4 arabağlantısı gösterimi

Sütun bağlantıları, satır bağlantıları ile benzer işleyip sinyal yönlendirmesini dikey olarak gerçekleştirirler. Şu kaynakları içerir:

- LAB içerisindeki paylaşılmış aritmetik zinciri ara bağlantıları
- LAB içi ve arası elde zinciri ara bağlantıları
- LAB içerisindeki yazmaç zincir bağlantıları
- Yukarı ve aşağıya dört öbek mesafeli C4 bağlantıları
- Aygıt boyunca yüksek hızlı dikey yönlendirme için C16 sütun bağlantıları



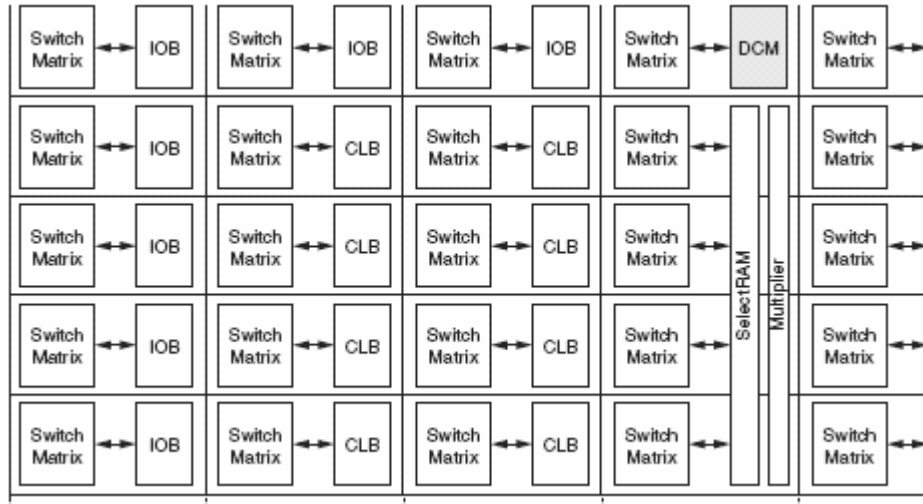
Şekil 4.26. : C4 arabağlantısı gösterimi

4.14.2. Xilinx Ara Bağlantı Yaklaşımı

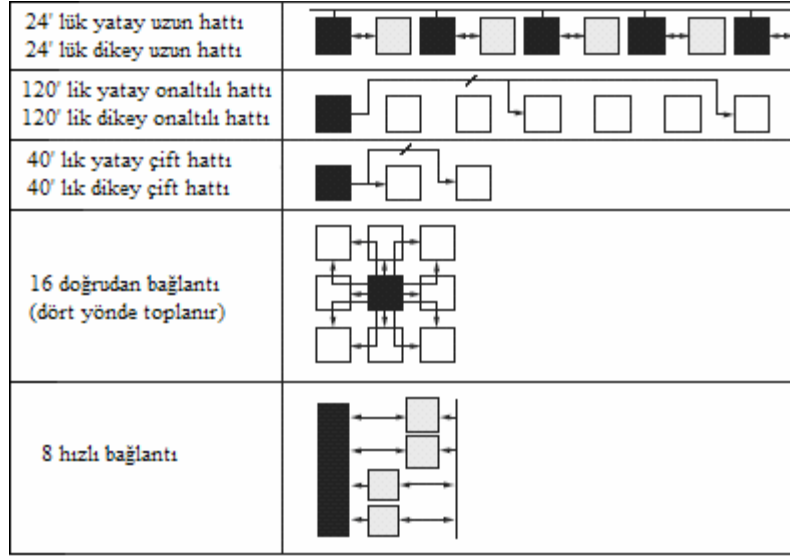
CLB öbekleri arasında dikey ve yatay yönlendirme kanallarından oluşan genel yönlendirmeli matris kaynakları kullanılır. Kullanılan teknoloji “Active Interconnect” olarak isimlendirilir. Yerel ve genel yönlendirme kaynakları hız ve zamanlama gereksinimlerini karşılayacak şekilde ayarlanmıştır. Bu teknolojide tamamıyla tamponlanmış programlanır yönlendirme matrisleri kullanılır. Şu ana bileşenleri içerirler:

- CLB ‘lere komşu General Routing Matrix (GRM) öbeği. GRM öbeği yatay ve dikey yönlendirme kaynaklarının bağlandığı anahtar matrisleridir (switch matrix). Yine bu öbek sayesinde CLB genel amaçlı yönlendirmeye erişebilir.
- Çift yönlü olarak dikey ve yatay aygıtı kat eden long (uzun) hatlar.
- Dört yöne de, sonraki üçüncü veya altıncı öbeklere erişecek şekilde yönlendirme yapabilen hex (onaltılı) hattı.
- Dört yöne de, sonraki ilk veya ikinci öbeklere erişecek şekilde yönlendirme yapabilen double (çift) hattı
- Dikey, yatay ve köşegen öbek komşularına sinyal yönlendirmeyi sağlayan direct connect (doğrudan bağlantı) hattı
- LUT çıkışından LUT girişine CLB içi yerel arabağlantı olan fast connect (hızlı bağlantı) hattı

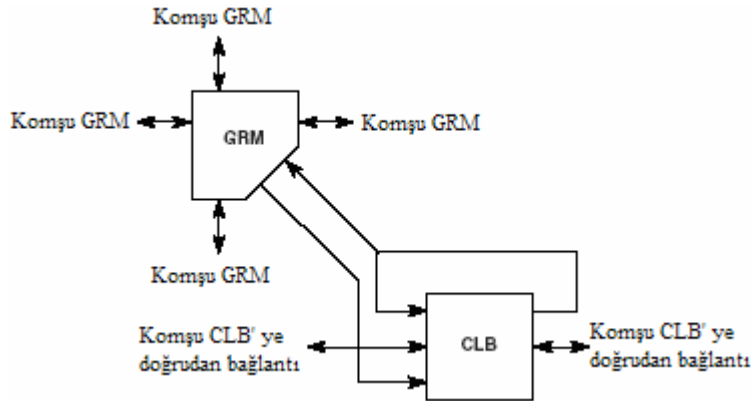
Aşağıda bazı yönlendirme kaynakları gösterilmiştir.



Şekil 4.27. : Yönlendirme kaynakları yüzeysel görünümü



Şekil 4.28. : Sıradüzensel yönlendirme kaynakları

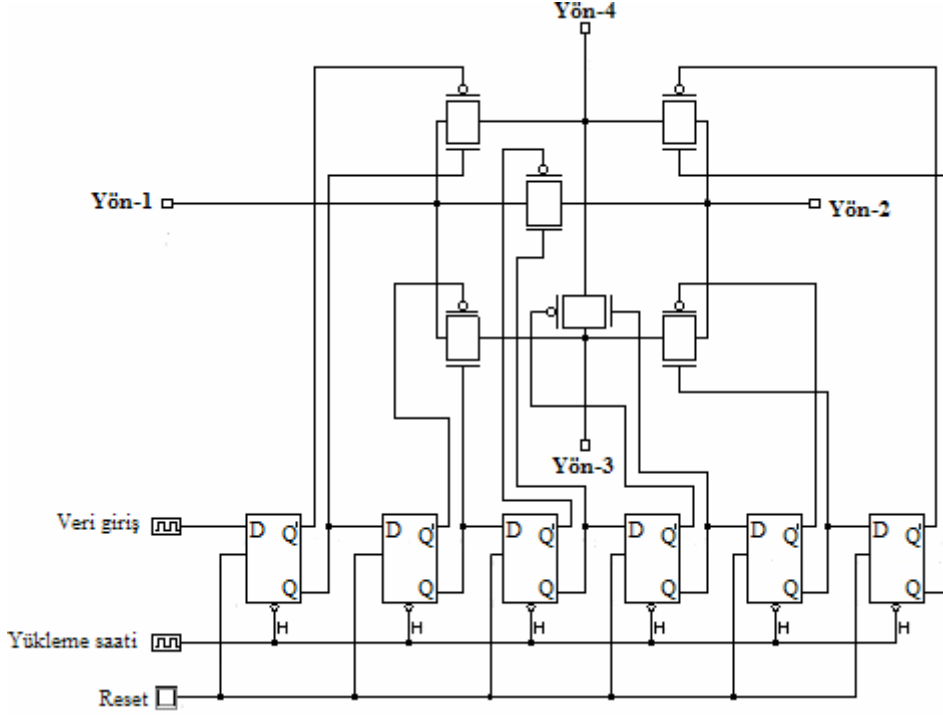


Şekil 4.29. : Virtex ailesi yerel yönlendirme

Genel ve yerel bağlantı türlerine ek olarak adanmış iç bağlantılar da vardır:

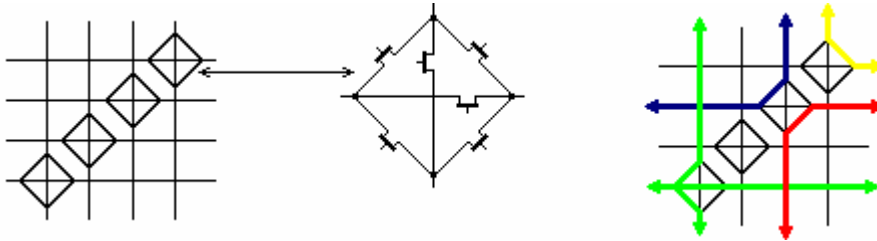
- Dikey komşulara bağlanan elde zinciri
- Yatay komşulara bağlanan sum of products (çarpımlar toplamı) zinciri
- Dikey komşulara bağlanan kaydırmalı yazmaç zinciri
- 3-durumlu yollar için yatay yönlendirme kaynaklarını süren 3-durumlu tamponlar

Yönlendirme işlemi temelde anahtarlama matrisleri ile gerçekleşir. Her anahtarlama matrisi bağlantı noktası 6 geçiş transistörü içerir. Her geçiş kapısı D tipi yazmaç ile kontrol edilir. Bu yazmaçlar birbirine bağlıdır. Dolayısıyla tek giriş ve saat girişi yapılandırmak için yeterlidir.



Şekil 4.30. : Anahtarlama matrisi için iletim kapıları ve yazmaç düzeni

İstenen bağlantı için adresi bilinen yazmaç mantıksal 1' e çekilir. Bu sayede bağlantı hattı üzerindeki bir bitlik tel yönlendirilir. Bağlantı ağının çok bitli olduğu düşünülürse anahtarlama matrislerinin büyük yer kapladığı daha iyi anlaşılacaktır. Daha iyi canlandırmak için aşağıdaki örneğe bakılabilir.



Şekil 4.31. : Anahtar matrisinin yapılandırılması

4.15. Çekirdek ve G/Ç besleme gerilimleri

Yıllar geçtikçe silikon yonga üzerindeki yapıların boyutları küçüldü. Çünkü küçük transistörler daha az maliyet ve güç tüketimi ve daha yüksek hız demektir.

Bu besleme FPGA' nın iç mantığı için kullanılır. Bu nedenle “core voltage (çekirdek gerilimi)” olarak adlandırılır. Ancak farklı G/Ç standartları, çekirdek geriliminden farklı gerilim seviyelerindeki sinyalleri kullanabilir. Dolayısıyla her genel amaçlı G/Ç kümesi kendi besleme bağlantılarına sahip olmalıdır.

Çizelge 4.2. : Çekirdek gerilimi ile teknoloji kullanımı karşılaştırılması

<u>Yıl</u>	<u>Çekirdek gerilimi (V)</u>	<u>Teknoloji kullanımı (nm)</u>
1998	3.3	350
1999	2.5	250
2000	1.8	180
2001	1.5	150
2003	1.2	130
2005	1.2	90

350 nm üretim teknolojisinden bu yana çekirdek gerilimindeki düşüş teknoloji noktasıyla paralellik göstermiştir. Ancak 1V' un altına inilmesi transistör anahtarlama eşiği ve gerilim düşümü gibi fiziksel sebeplerce engellenmektedir.

5. FPGA PROGRAMLAMA

Her FPGA satıcısı kendine has terminoloji, teknik ve iletişim kurallarına sahiptir. Yine ürün aileleri arasında bile ayrıntılar değişebilmektedir. Temelde FPGA iki durumda çalışır:

- Kullanıcı kipi ; Normal çalışma durumudur.
- Yapılandırma kipi ; İlgili bacaklar ile programlama durumudur.

FPGA yapılandırma daha öncede belirtildiği gibi programlama hücreleri ile gerçekleşir. Yapılandırma için gerekli temel basamaklar vardır:

- Yapılandırma dosyasının oluşturulması
- Kullanılacak yapılandırma kipinin seçimi
- Yapılandırma için tercih edilen donanım hazırlanması

Yapılandırma için üç temel yöntem kullanılır:

- Bilgisayara ile programlama kartı arasına bağlanan kablo (seri, paralel, usb) ile arayüz program kullanarak
- Kart üzerindeki FPGA' ya bağlı özel bellek yardımıyla her enerji verildiğinde
- Kart üzerindeki uyarlanmış mikrodenetleyici ile programlama.

5.1. Yapılandırma Dosyası

Tasarımı gerçeklemek için çeşitli araç ve akışlar vardır. Bütün bu işlemlerden sonra “configuration file” veya “bit file” olarak isimlendirilen ve özel işlevi gerçeklemek için hazırlanan bilgiyi FPGA' ya yüklemek gelir. Yapılandırma dosyası üretici firmaların sunduğu ürün geliştirme programları tarafından üretilir.

SRAM tabanlı FPGA' larda, yapılandırma dosyası yapılandırma verisi ve yapılandırma komutlarını içerir. Yapılandırma verisi, doğrudan programlanır mantık öğelerinin durumunu belirlemek için kullanılır. Yapılandırma komutları ise aygıt yapılandırma verileri ile ne yapacağını söyler.

E² ve FLASH tabanlı aygıtlar, SRAM tabanlı yaklaşıma benzer şekilde programlanır. Antifuse tabanlı FPGA' larda ise yapılandırma dosyası ağırlıklı olarak antifuse öğelerinin işleneceği yapılandırma verisinin gösterimini içerir. Bu yongalar özel programlayıcı aygıtlar içerisinde programlanır. Yapılandırma verisi ana bilgisayardan programlayıcıya aktarılır. Bu özel programlayıcılar, yapılandırma verisi kılavuzluğunda, seçilen bacaklardan görece yüksek gerilim ve akım uygulayarak ilgili antifuse öğesini gerçekleştirir.

Bellek kullanımı ve zaman tasarrufu için yapılandırma verileri sıkıştırılarak aktarılır. Sıkıştırılan veriler ilgili yonga içerisinde gerçek zamanlı olarak açılır. Bu sayede veri akışında %50 oranlarında azalma olur.

Tasarım güvenliği için (bit stream) bit katarı şifreleme yöntemi kullanılır. Bu amaçla kullanılan yonga üzerinde şifre çözücü birim vardır. Şifrelenmiş yapılandırma verisi bu birime kayıtlı şifre anahtarı ile örtüşüyorsa veri çözülür ve yapılandırma gerçekleşir. Aksi durumda harici olarak kopyalama ve tersine mühendislik gerçekleştirilemez. Yonga üzerindeki çözücü birim farklı amaçlar için kullanılamaz. Şifre anahtarı Xilinx Virtex-4 için 256-bit, Altera StratixII için 128-bittir ve AES (Advanced Encryption Standard) algoritması kullanılır. 256 bit için 1.1×10^{77} farklı anahtar birleşimi mümkündür.

5.2. Yapılandırma Kipleri

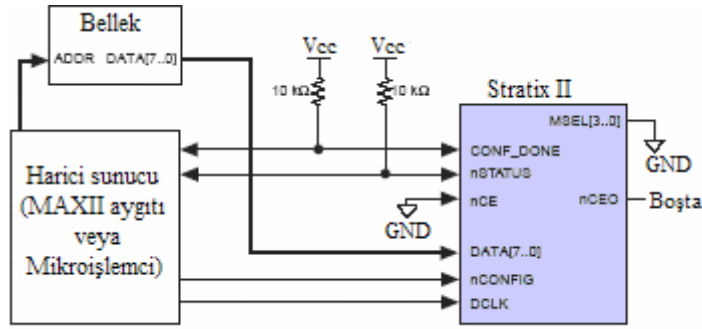
FPGA yongaları yapılandırmak için kullanılan bağlantı noktaları ve farklı yaklaşımlar vardır. Aygıtın istenen kipte çalışması için yapılandırma kipi bacakları mevcuttur. Farklı üreticilerin kullandığı yöntemler isim olarak değişse de genel mantık olarak benzerdirler. Örnek olarak aşağıda Altera Stratix II yongasında kullanılan kip seçenekleri genel anlamda görülmektedir.

Çizelge 5.1. : Stratix yapılandırma kipleri

Yapılandırma kipi	MSEL3	MSEL2	MSEL1	MSEL0
Fast passive parallel (FPP)	0	0	0	0
Açma ve/veya güvenlik özelliği etkin FPP	1	0	1	1
Passive parallel asynchronous (PPA)	0	0	0	1
Passive serial (PS)	0	0	1	0
Fast AS (40 MHz)	1	0	0	0
AS (20 MHz)	1	1	0	1
JTAG tabanlı yapılandırma	GND	GND	GND	GND

5.2.1 Fast Passive Parallel (FPP) kipi

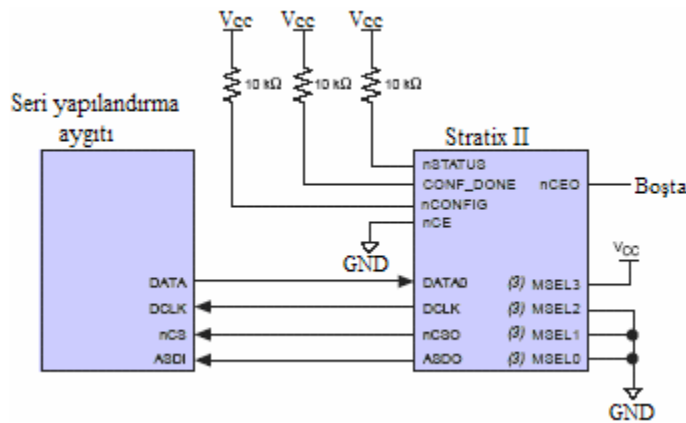
Hızlı yapılandırma talebine cevap verir. Her saat çevriminde 8-bitlik aktarım mevcuttur. Şifreleme ve sıkıştırma özelliklerini destekler.



Şekil 5.1. : Harici sunucu kullanarak FPP ile tek aygıt yapılandırma

5.2.2 Active Serial (AS) kipi

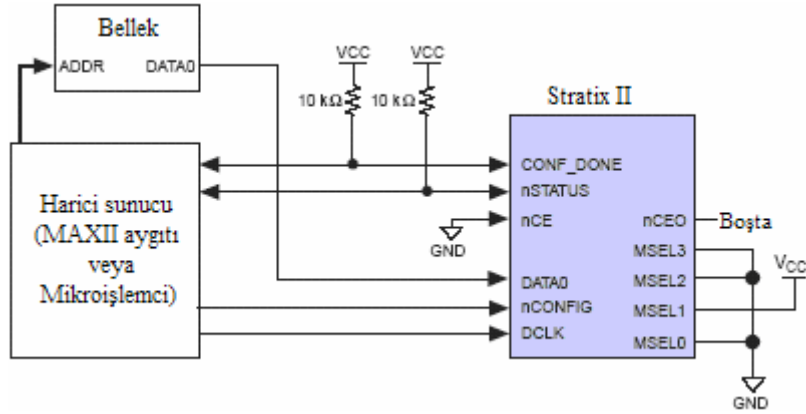
Bu kip ile flash tabanlı bellek ve az sayıda bacak kullanan düşük maliyetli seri yapılandırma aygıtları kullanılabilir. Aktarım her saat çevriminde bir bit olacak şekildedir. Sıkıştırma ve şifreleme özelliklerini destekler.



Şekil 5.3. : AS ile tek aygıt yapılandırma

5.2.3. Passive Serial kipi

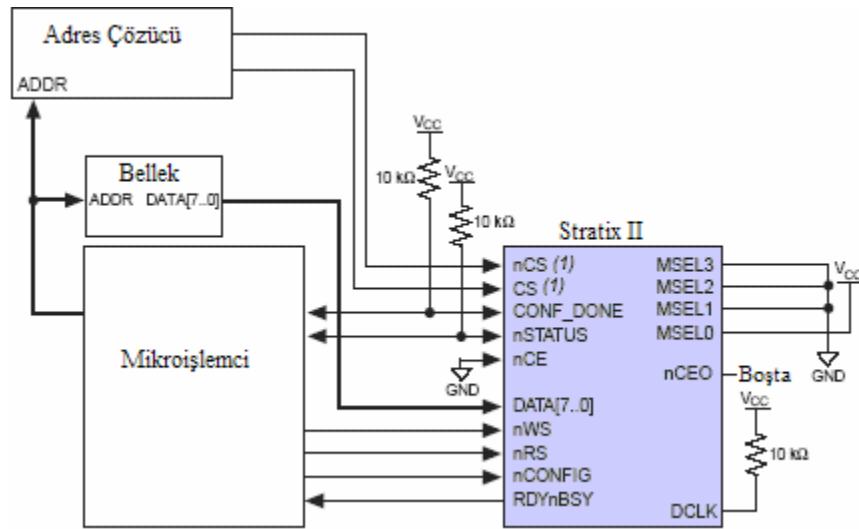
Harici aygıtlar kullanılarak yapılır. Yapılandırma kontrolü bu aygıtlarca gerçekleşir. Aktarım yine her saat çevriminde bir bittir. Sıkıştırma ve şifreleme özellikleri kullanılabilir.



Şekil 5.4. : Harici sunucu kullanarak PS ile tek aygıt yapılandırma

5.2.4. Passive Paralel Asynchronous (PPA) kipi

Yapılandırma verisinin bellekten mikroişlemci gibi bir sunucu aracılığıyla aktarıldığı yöntemdir. Her saat çevriminde 8-bit aktarım sağlar. Şifreleme ve sıkıştırma özelliklerini desteklemez.



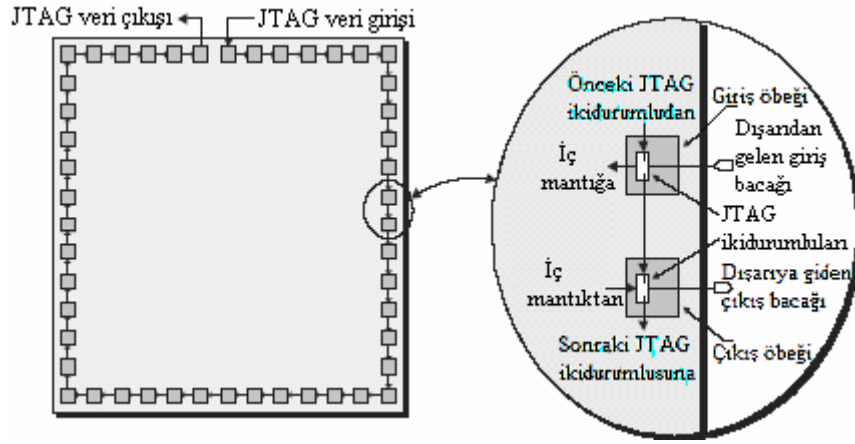
Şekil 5.5. : Mikroişlemci kullanarak PPA ile tek aygıt yapılandırma

5.2.5. JTAG kipi

Bu yöntem, Joint Test Action Group tarafından sınır tarama testi için IEEE Std. 1149.1 olarak geliştirilmiştir. Boundary-Scan Test (BST) olarak isimlendirilen bu test mimarisi kart üzerindeki bileşenlerin verimli şekilde sınanmasını sağlar. BST mimarisi ile bacak bağlantıları

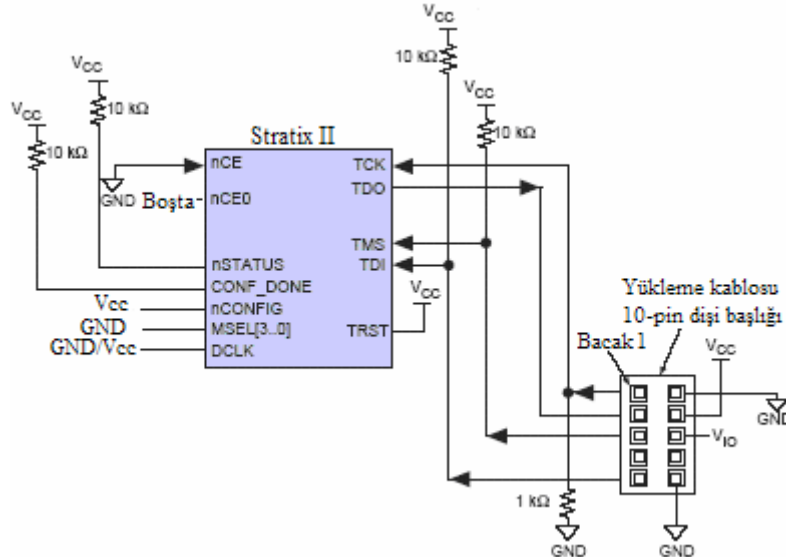
fiziksel prob kullanmadan sınanmaktadır. JTAG devresi aynı zamanda yapılandırma verisini aktarmak için kullanılabilir.

FPGA içerisindeki her G/Ç bacağına ilişkilendirilmiş JTAG yazmacı vardır. Bu yazmaçlar kendi aralarında zincir halinde bağlanmıştır. JTAG veri girişi bacağı ile JTAG özel komut yazmaçlarına komutlar yüklenebilir. Bu komutlar ile FPGA' nın yapılandırma amaçlı iç SRAM kaydırmalı yazmaçlarına JTAG zinciri bağlanır. Bu sayede JTAG bağlantı noktaları programlama için kullanılabilir.



Şekil 5.6. : JTAG sınır tarayan yazmaçları

JTAG bağlantı noktaları, diğer kipler işlerken de mevcuttur. Her hangi bir kip ile başlanıp sonrasında JTAG yapılandırma ile devam edilebilir. JTAG tabanlı yapılandırmanın diğerlerine göre önceliği vardır. MSEL bacakları Şifreleme ve sıkıştırma özelliklerini desteklemez.



Şekil 5.7. : Yükleme kablosu kullanarak JTAG ile tek aygıt yapılandırma

FPGA sağlayıcılarının ve yan kuruluşların sunduğu birçok programlama ve geliştirme kartları mevcuttur. Kullanılacak FPGA yongasının özellikleri belirlendikten sonra uygun donanım seçilmelidir. Aşağıda iki adet geliştirme kartı verilmiştir.



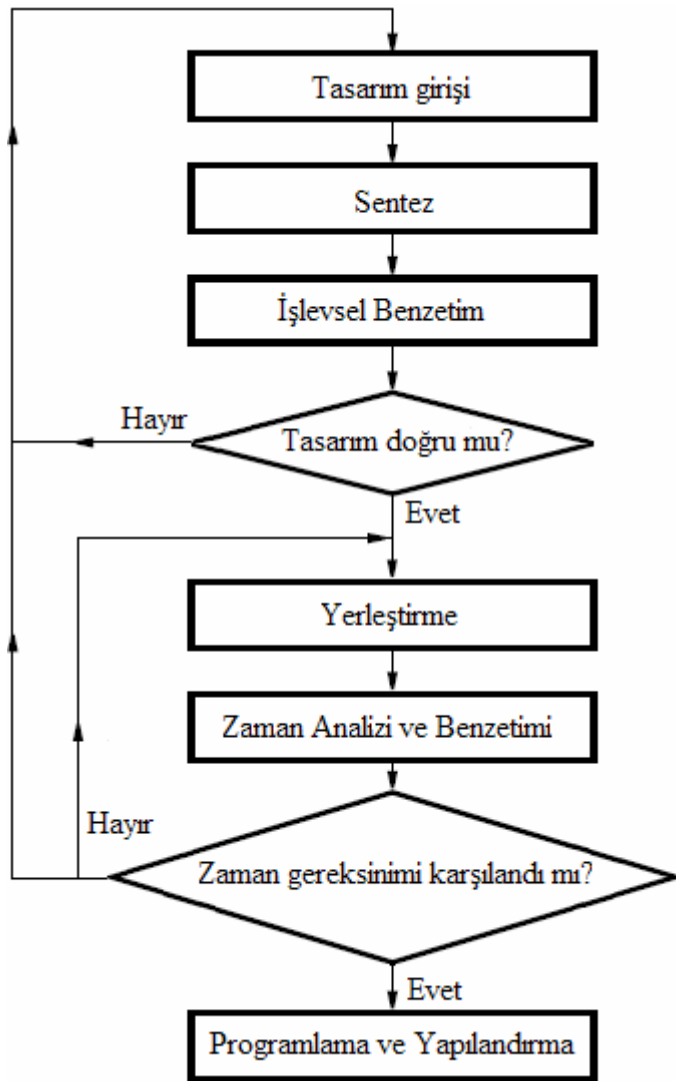
Şekil 5.8. :Stratix GX Geliştirme Kartı



Şekil 5.9. : Cyclone II EP2C35 DSP Geliştirme Kartı

6. UYGULAMA GELİŞTİRME

FPGA yongasında istediğimiz programı gerçeklemek için aşağıda verilen akış şeması takip edilir. CAD (Computer Aided Design) olarak bilinen bilgisayar destekli tasarım yazılımları sayesinde günümüzde tasarımcılar üzerinden büyük yük kalkmıştır. Tasarım aşamasında büyük zaman alabilecek basamaklar ortadan kaldırılıp tasarımcının özel yeteneği ile fark yaratılmaktadır.



Şekil 6.1. : FPGA akış şeması

6.1. Tasarım Girişİ

İstenen devre şematik veya öbek çizimler ile yapılır. Artan kapı kullanımı sonucunda Verilog veya VHDL gibi HDL (Hardware Description Language) donanım tanımlama dillerinden herhangi birini kullanmak kaçınılmaz olmuştur. Bu diller, mantıksal gerçeklemeler için özelleşmişlerdir. Bu sayede elektronik bileşenlerin davranışsal ve yapısal tanımlaması kolayca yapılarak esnek ve hızlı tasarım yapılabilir. Aksi takdirde milyonlarca kapıyı tanımlamak mümkün olmazdı.

6.2. Sentez

CAD sentez araçları ile devremiz için gerekli olan mantıksal ögeler ve bu ögeler arasındaki bağlantılar oluşturulur.

6.3. İşlevsel Benzetim

Sentezlenen devre bu aşamada işlevsel doğruluğu açısından sınanır. Hata durumunda tasarımda değişiklik yaparak sentez ve benzetim basamakları tekrarlanır.

6.4. Yerleştirme

İlgili yerleştirme araçları, bağlantı listesinde tanımlanmış mantıksal ögelerin FPGA yongası içerisindeki gerçek mantıksal ögelere yerleşimi yapar. Mantıksal ögeler arasındaki gerekli bağlantılar için yönlendirme hatları belirlenir.

6.5. Zaman Analizi ve Benzetimi

Yerleştirilmiş devrenin çeşitli yolları arasındaki yayılım gecikmeleri analiz edilir. Bu sayede devre beklenen başarımlar için sınanır ve sonucunda gerekli bağlantılar arasındaki zaman gecikmeleri gösterilir.

Zaman benzetiminde ise yerleştirilmiş devre zamanlama ve doğruluk açılarından sınanır. İstenen zaman gereksinimleri sağlanamaz ise tasarım sürecinde en başa dönülür ve

tasarımda belirlenen hata ile ilgili gerekli geliştirme yapılır. İşlemler benzetim sonuçları yeterli oluncaya kadar tekrarlanır.

6.6. Programlama ve Yapılandırma

Son aşamada belirli donanımlar üzerinden, gerçek FPGA yongası arzu edilen devreyi gerçeklemek için programlanır. Yapılandırma ile mantıksal öğeler istenen şekilde yapılandırılır ve içerikleri belirlenir.

6.7. Örnek Uygulama

FPGA yongası : Altera, Cyclone II EP2C35F672C6

Program geliştirme ortamı : Altera Quartus II v5.0 Web Edition Full

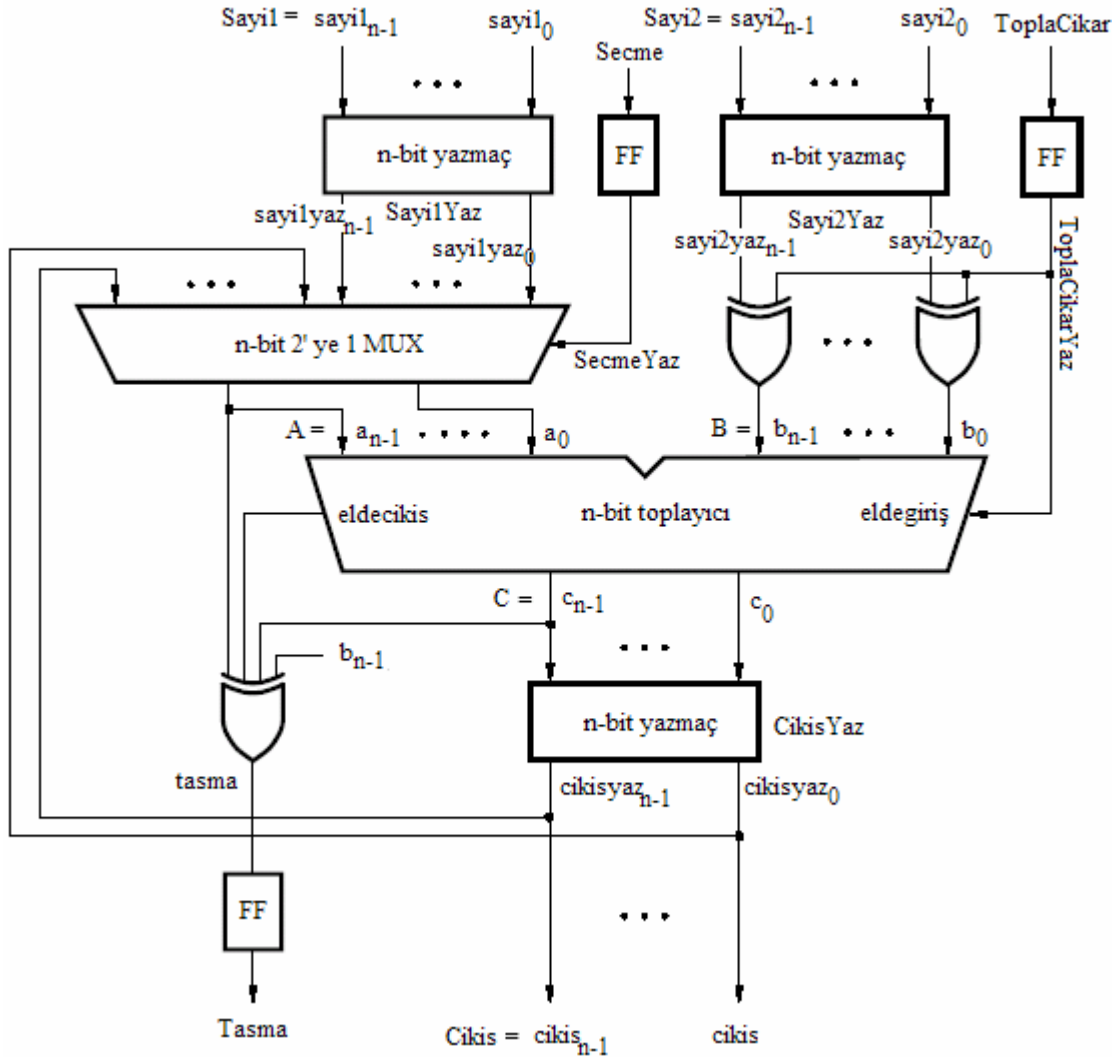
Devre : n-bit Toplama ve Çıkarma Devresi

Kodlama : Verilog HDL

6.7.1. Devre Şeması ve Açıklama

Aşağıda şeması verilen n-bit toplama ve çıkarma devresine Sayı1 ve Sayı2 isimli n-bit iki sayı girmektedir. Devrede iki kontrol biti vardır. İlki ToplaCikar girişidir. ToplaCikar= 0 olursa Çıkış = Sayı1 + Sayı2 ve ToplaÇikar = 1 olursa da Çıkış = Sayı1 - Sayı2 olarak gerçekleşir. Bilindiği gibi XOR kapısı $F = A'B + AB'$ işlemi gerçekleştirir. İstedığımız sayının tersini bu sayede alabiliriz. İkilik sayıların çıkarması yapılırken çıkarılacak sayının tersi (complement) alınıp "1" eklenir. Daha sonra iki sayı toplanır. Devrede ToplaÇikar değeri bu nedenle eldegiriş' i sürer.

Diğer kontrol Seçme girişidir. Seçme = 0 olduğu zaman Çıkış = Sayı1 ± Sayı2 'dir. Seçme = 1 yapıldığı taktirde Çıkış = Çıkış ± Sayı2 olur. Bu durumda değer biriktirme yapılabilir.



Şekil 6.2. : N-bit toplama ve çıkarma devresi

6.7.2. Devrenin aşamalı gerçekleştirilmesi

1. İlk önce www.altera.com internet adresinden ücretsiz Quartus II v5.0 WE programını indirdim. Altera Quartus' u tercih etmemim sebebi ücretsiz ve 150 günlük tam sürüme yakın kullanım izni vermesidir. Boyutu 216 MB'dir. Bunun yanında ara yüzünün kullanıcı dostu olması yeni başlayanlar için faydalıdır. Bunun yanında Xilinx firmasının da ISE WebPACK™ 7.1i programı mevcuttur.

2. Devre şeması ve gerekli basamaklar kabaca oluşturuldu. İstenen özellikler için devre şeması çizildi.

3. Quartus II' yi proje oluşturmak için açtım. "Toplama_Cikarma" isimli proje oluşturdum. Kullanmak istediğim FPGA yongasını seçtim. Dosya eklemeyi çıkardım.

4. Yeni Verilog (toplacikar.v) dosya açtım. Devreyi gerçeklemek için gerekli kodları yazdım.

```
// n-bit toplama-çıkarma devresi verilog kodu(32-bitlik yaptım)
// En üst seviye birim tanımlanır
module toplacikar (Sayi1, Sayi2, Saat, Sifirlama, Secme, ToplaCikar, Cikis,
                  Tasma);
parameter n=32; // 32 bitlik toplama-çıkarma için
input [n-1:0] Sayi1, Sayi2; //girişler tanımlanır
input Saat, Sifirlama, Secme, ToplaCikar;
output [n-1:0] Cikis; //Çıkışlar tanımlanır
output Tasma;
//Tutulmasını istediğimiz değişkenleri yazmaca atarız
reg SecmeYaz, ToplaCikarYaz, Tasma;
reg [n-1:0] Sayi1Yaz, Sayi2Yaz, CikisYaz;
// Arada kullanılacak telleri tanımlarız
wire [n-1:0] A, B, C, Cikis;
wire eldecikis, tasma;

// Birleşimsel mantık devresi tanımlanır
assign B = Sayi2Yaz ^ {n{ToplaCikarYaz}};
// B = Sayi2 XOR n(ToplaCikar) ataması
mux2ye1 cogullayici (Sayi1Yaz, Cikis, SecmeYaz, A);
//Sayı1 veya Cikis seçiliyor
defparam cogullayici.k = n;
//Çoğullayıcı için k = n olarak belirtiliyor
toplakbit nbit_toplayici (ToplaCikarYaz, A, B, C, eldecikis);
//A ve B toplanıyor
defparam nbit_toplayici.k = n;
// toplayıcı için k = n olarak belirleniyor
assign tasma = eldecikis ^ A[n-1:0] ^ B[n-1:0] ^ C[n-1:0];
// tasma teli = XOR(A,B,C,eldecikis)
assign Cikis = CikisYaz;
//CikisYaz, Cikis teline atanıyor

//İkidurumlu ve yazmaç tanımlama
always @ (posedge Sifirlama or posedge Saat)
//saat ve Sifirlama yükselen kenarlarda sürekli geçerli
if(Sifirlama == 1) // Sifirlama = 1
begin //öbek başlangıcı
    Sayi1Yaz <= 0;
    Sayi2Yaz <= 0;
    CikisYaz <= 0;
    SecmeYaz <= 0;
    ToplaCikarYaz <= 0;
    Tasma <= 0;
end //öbek sonu
else // Sifirlama = 0
begin
    Sayi1Yaz <= Sayi1; // değer atama
    Sayi2Yaz <= Sayi2;
```

```

    CikisYaz <= C;
    SecmeYaz <= Secme;
    ToplaCikarYaz <= ToplaCikar;
    Tasma <= tasma;
end
endmodule //En üst birim sonu

//k bit 2ye1 çoğullayıcı
module mux2ye1(X, Y, Secme, F);
//mux2ye1 isimli çoğullayıcı birim tanımlanıyor
parameter k=8;
// k' ya bir değer atanmak zorunda. Çünkü dizide kullanılıyor
input [k-1:0] X, Y;
input Secme;
output[k-1:0] F;
reg [k-1:0] F;
always @ (X or Y or Secme) // X, Y, Secme her zaman işletilir
if (Secme == 0) //Sayı1 etkin
    F = X;
else
    F = Y; //1. sayı Çıkış
endmodule //çoğullayıcı birimi sonu

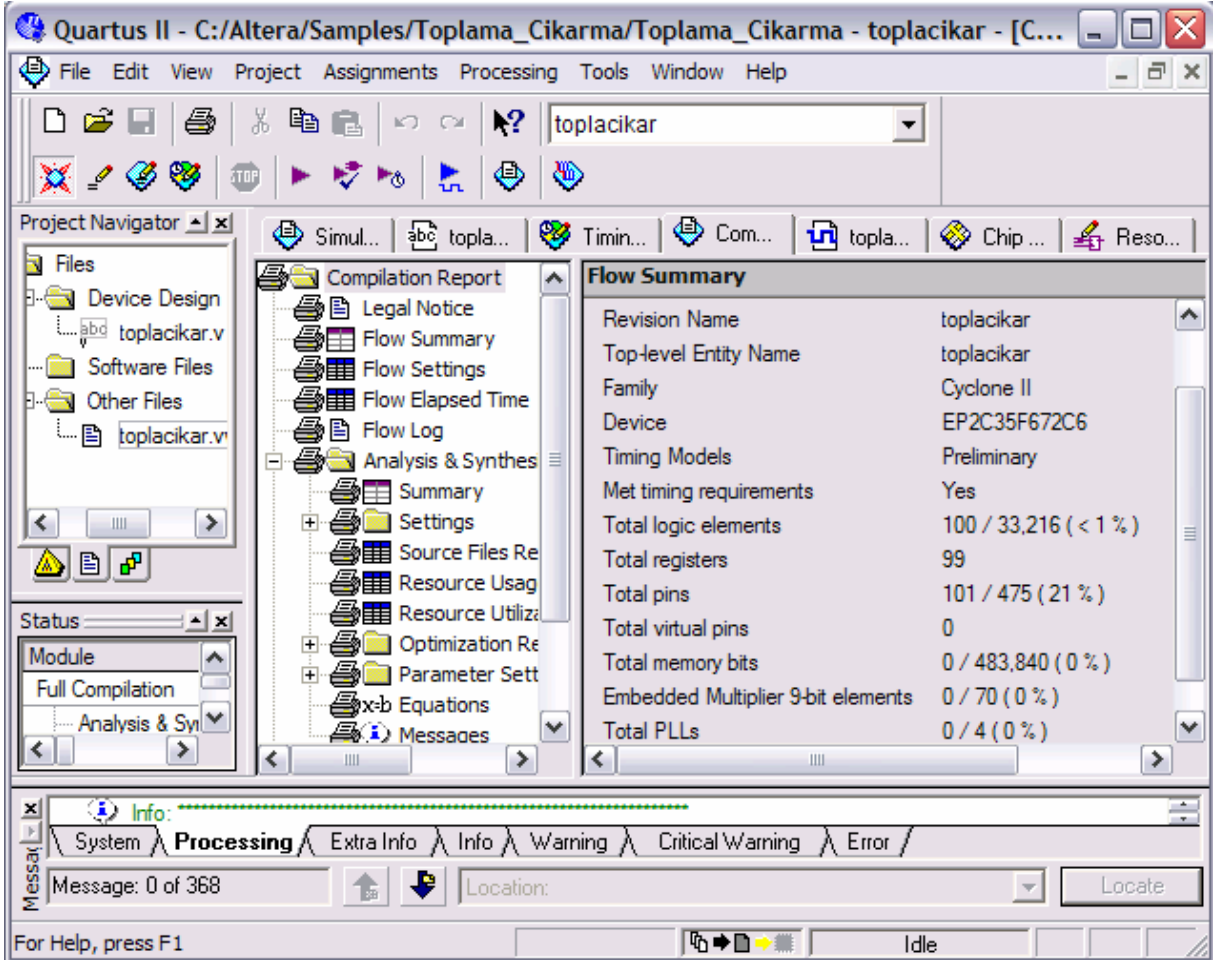
//k bit toplayıcı
module toplakbit (eldegiris, V, W, Toplam, eldecikis);
//toplama birimi tanımlanıyor
parameter k = 8;
input [k-1:0] V, W;
input eldegiris;
output [k-1:0] Toplam;
output eldecikis;
reg [k-1:0] Toplam;
reg eldecikis;
always @ (V or W or eldegiris) // V, W, eldegirisi sürekli işler halde
    {eldecikis, Toplam} = V + W + eldegiris; // tam toplama eşitliği
endmodule

// Program sonu

```

5. Oluşturduğum verilog dosyasını projeye ekledim. İşlemler sıradüzensel olarak gerçekleştirildiği için en üst birim ismine dikkat ederek projenin ilk bölümünü tamamladım.

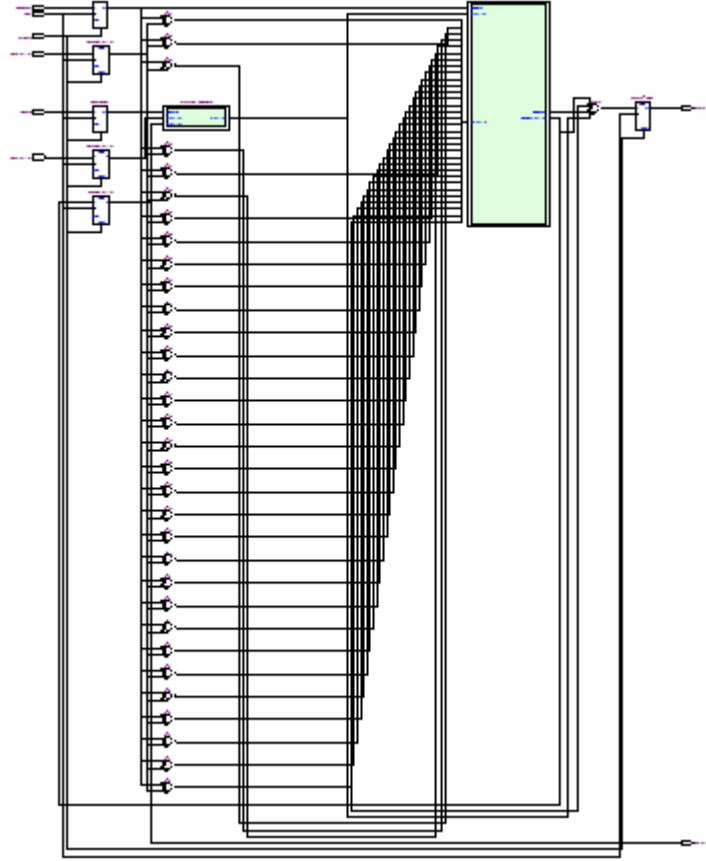
6. Devre kodunun doğruluğunu sınamak için Processing/Start Compilation seçerek projeyi derlettirdim. İstenirse Processing başlığı altından farklı sınamalar tek tek uygulanabilir. Devre analizi ve sentezi, yerleştirme, zaman analizi hep bu şekilde yapılır. Derlemenin başarı ile sonuçlandığına ilişkin rapor çıkacaktır. Bu raporu inceleyerek tasarım ile ilgili birçok bilgiye ulaşılabilir. Assignment/Settings başlığı altından derleme ilgili tercihler belirlenebilir.



Şekil 6.3. :Toplama Çıkarma devresi verilog kodunun derlenmesi

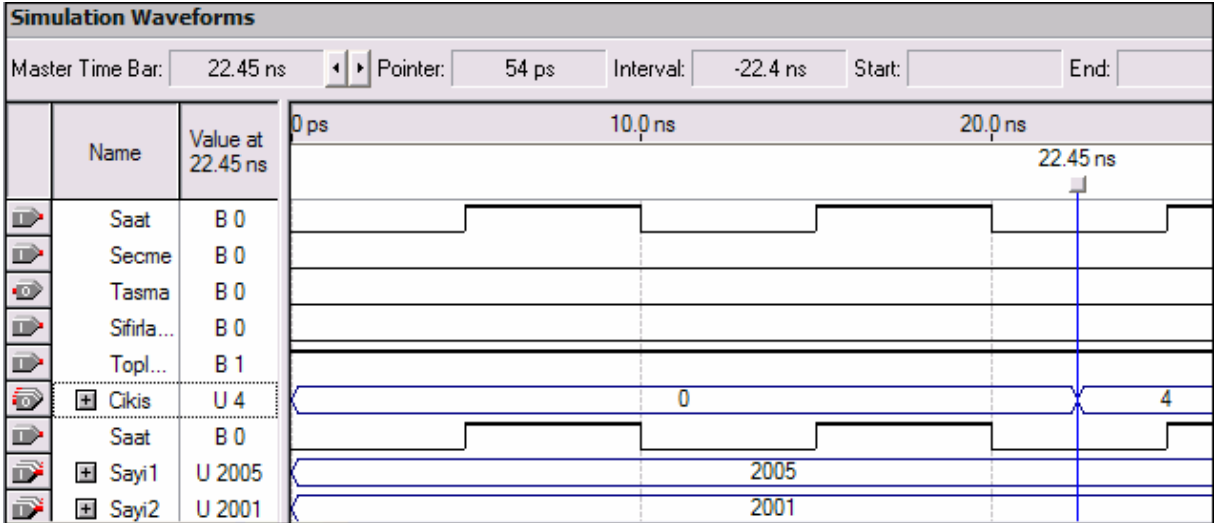
7. Assignment/Timing Closure Floorplan alt programı çalıştırdım. Bu program zaman analizi yapabilmek için Cyclone II üzerinde devrenin kabaca gösterimini sağlar. Burada critical path, fan-in, fan-out gibi zamanlama ile ilgili birçok değerlendirmeye ulaşılabilir. İstenlere View başlığı altından kolayca ulaşılmaktadır. Kısayol olarak Yakınlaştırma için: Ctrl+Space, Uzaklaştırma için Ctrl+Shift+Space tuş birleşimleri kullanılabilir.

8. Kodlama ile oluşturduğumuz devrenin öbekler halinde bağlantısını görmek için Tool/RTL Viewer alt programını çalıştırdım. Buradan kapı seviyesine kadar tüm bağıntıları görebiliriz. Yine bağlantılar, bacaklar ve büyük birimler sıradüzensel olarak görülebilmektedir. RTL Viewer ile verilog kodundan devremizi oluşturduk. Oluşturulan devre üzerinde gezinerek işleyişi daha iyi anlayabiliriz.



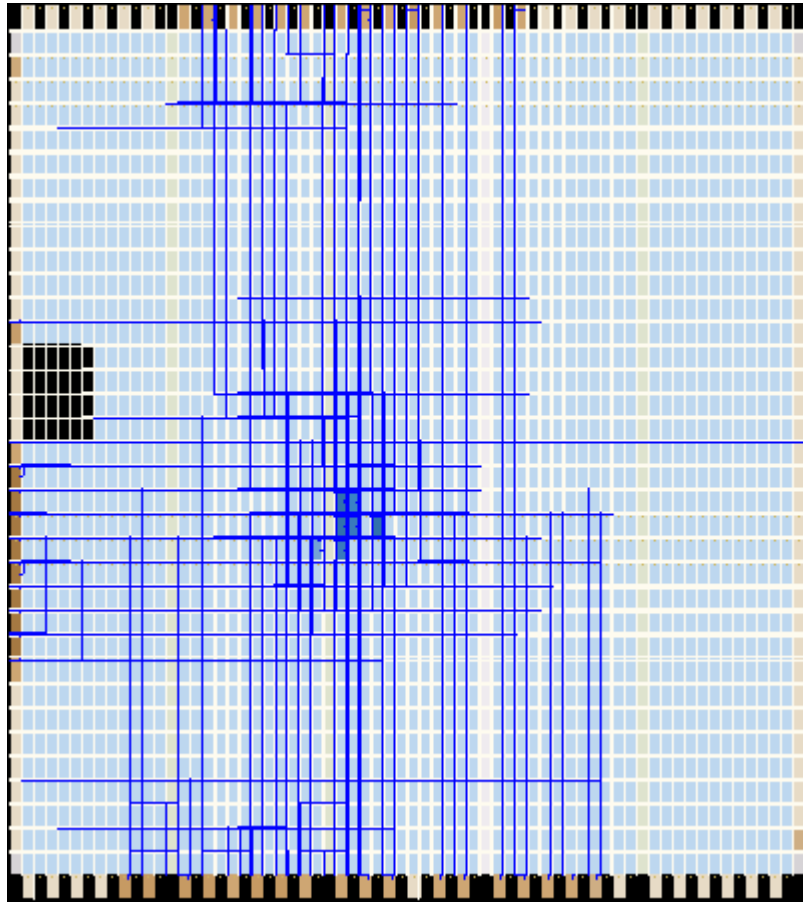
Şekil 6.4. : 32-bit Toplama-Çıkarma verilog kodundan oluşturulan devre şeması

9. Devremizin düzgün çalışıp çalışmadığını benzetim ile gözlemleyebiliriz. Bu amaçla il önce New/Other Files/Vector Waveform File(*.vwf) ile yeni dalga şekli gösterimi dosyası açtım. Açılan dosyada sol tarafa görülmesi istenen sinyaller yerleştirilir. Sinyaller istenildiği gibi değerlendirilebilir. Bunun için sol tarafa “Insert Node or Bus” ile sinyal kaynakları atıldıktan sonra, istenen sinyalin üzerine sağ tıklanarak Value başlığı altından seçim yapılır. Saat, Sayi1, Sayi2, Tasma, ToplaCikar, Secme, Cikis ve Sifirlama sinyallerini ekledim. Kontrol sinyallerini, giriş sayılarını istediğim gibi belirledikten sonra Processing/Start Simulation ile benzetimi yaptım. Gerçektende istediğim neticeyi elde ettim. Aşağıda benzetim sonuçlarını gösteren benzetim dalga şekilleri görülmektedir. 10ns periyotlu saat ile çıkarma sonucu 22.45ns ‘de elde edilmektedir.



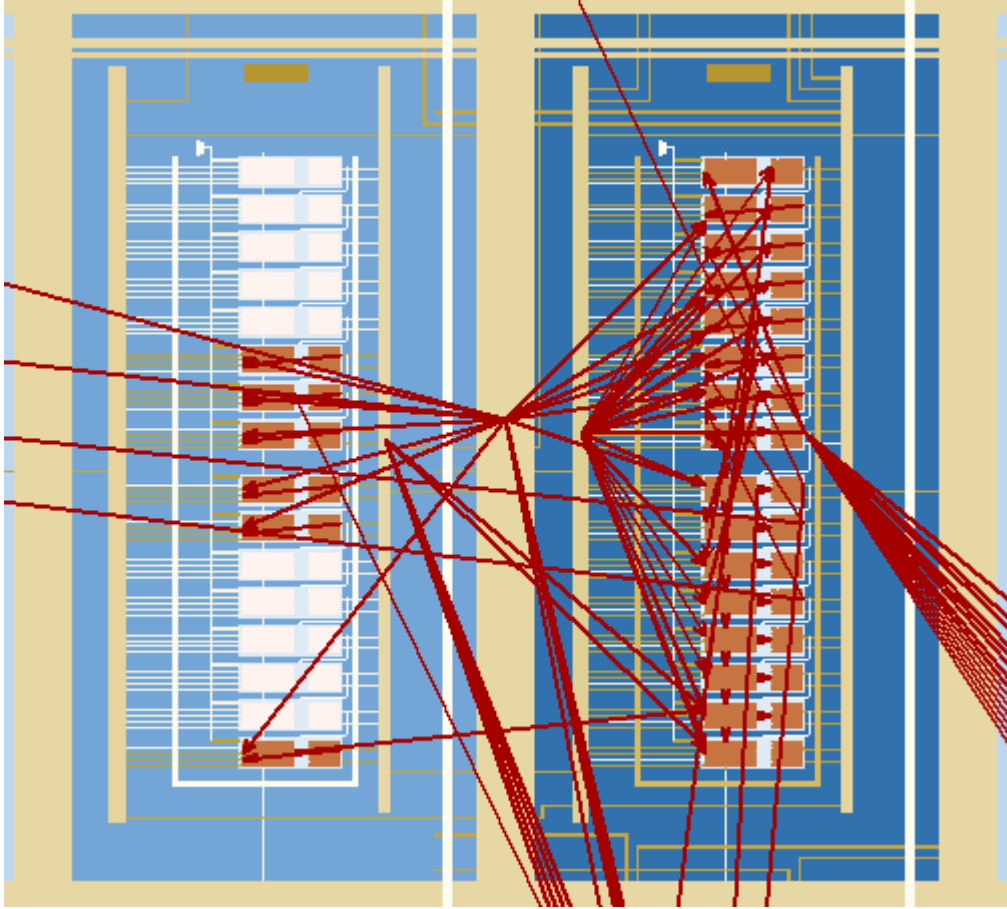
Şekil 6.5. : Dalga şekli benzetim sonuçları

10. Toplama/Çıkarma devre tasarımının Cyclone II FPGA yongası üzerindeki yerleşimini Tool/Chip Editor alt programı ile görebiliriz. Aşağıdaki resimden daha net görülebileceği gibi tasarım yonga üzerinde çok küçük yer kaplamaktadır. (ortada koyu mavi bölge)



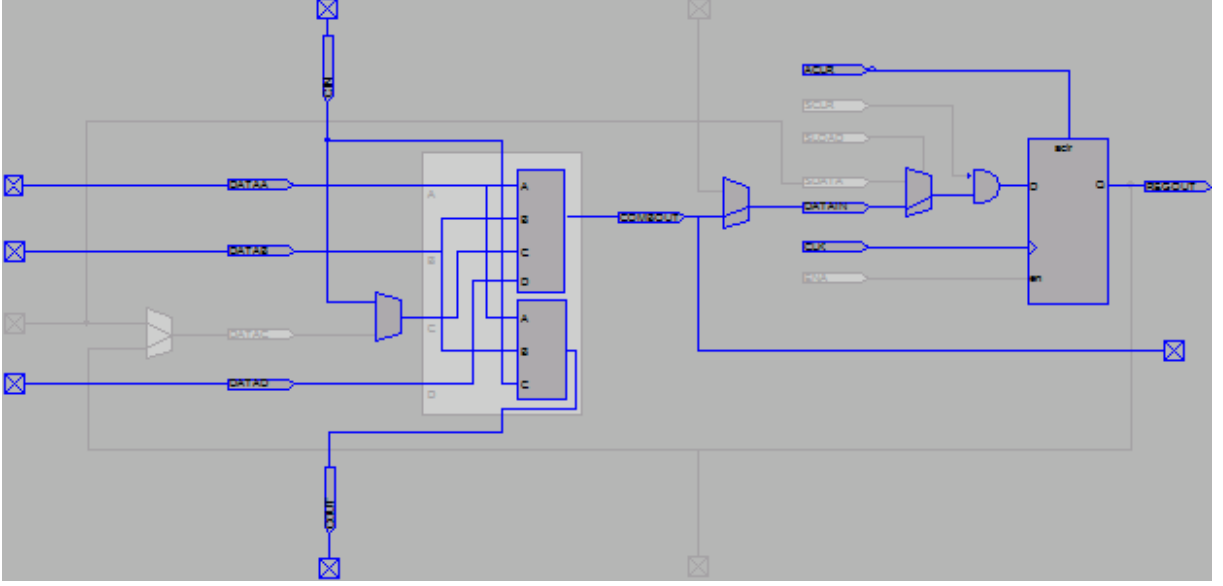
Şekil 6.6. : 32 bit Toplama/Çıkarma devresinin Cyclone II üzerindeki yerleşimi

Giriş/Çıkış ve programlanır hücre yerleşimi derleme işlemine en uygun şekilde yapılır. Yerleşimi istediğimiz şekilde değiştirebiliriz. Belirli ölçüde verimi yükseltmek mümkündür. Yonga üzerinde yaklaşarak şekli ayrıntı görebiliriz. FPGA mimarisi bölümünde anlatılan birimleri yonga üzerinde görebiliriz. Kullanılan arabağlantı türleri, kullanılan LAB' lar ve G/Ç noktaları rahatlıkla belirlenebilir ve bu sayede tasarım süreci daha açık hale gelir. İstenirse aşağıdaki gibi yönlendirmeler görülebilir. Bu aşamalar sadece gösterim için olmayıp tasarımcı tercihi ile yerleşim değiştirilebilir. İnce ayar yapılabilir.



Şekil 6.7. : Genel hatlarıyla LAB ve Arabağlantılar üzerinden çıkış yelpazesi

11. Yonga üzerindeki gösterim yukarıdaki şekil ile sınırlı değildir. İstenirse turuncu renkte gözükken mantık öğelerinin üzerine çift tıklanarak iç yapılandırması genel hatlarıyla görülebilir. Bu alt program “Resource Property Editor” olarak isimlendirilir. Bir sonraki sayfada LAB içerisinde yer alan LE’ nin aritmetik kipte çalışması görülmektedir.



Şekil 6.8. : Cyclone II'nin LE Aritmetik kipte çalışması

Chip Editor' u kullanarak pek çok bilgiye ulaşabiliriz. İstedığımız bağlantının kaynağı, hedefi, kullanıcı ve kullandığı kaynak isimleri ve yerleşimleri kolaylıkla gözlenebilir. İstedığımız öğenin üzerine sağ tıklayarak, Locate başlığı altından çıkan alt programları çalıştırabiliriz. Bu sayede zaman analizine (Timing Closure Floorplan), Mantıksal (RTL Viewer) görünümüne, bacak yerleşimine (Pin Planner) veya doğrudan kaynak tasarım dosyasına geçebilir ve tam konumunu tayin edebiliriz.

Burada gösterilmedi ama istenirse tasarım, hazır öbekler ve tasarımcının oluşturacağı öbekler arasındaki bağlantılar yapılarak ta kolaylıkla gerçekleştirilebilir. "Block Diagram / Schematic File" seçilip (*.bdf), üretici firmanın sunduğu kütüphaneleri veya oluşturulacak yeni öbekleri kullanarak sistemli tasarımlar gerçekleştirilir.

Görüldüğü gibi sunulan araçlar ile tasarımcı istediği esneklikte devre tasarlayabilir. Günümüzde birçok üretici firmalar ve özel tasarım araçları geliştiren diğer kuruluşlar sayesinde tasarım süreci çok hızlanmıştır. Artan teknoloji kullanım gereksinimi, her geçen gün daha hızlı ve daha başarılı bilgisayar destekli araçları pazara sunmayı gerektirmektedir.

7. SONUÇ

Genel hatlarıyla anlatmaya çalıştığım FPGA' lar ve dolayısıyla yonga tasarım dünyası her geçen gün insan hayatının vazgeçilmez unsurları olarak pekişmektedirler. Tümüleşik devreler 50 yıllık geçmişine rağmen günümüze kadar inanılmaz gelişme kaydetmiştir. Uzayan tasarım süreçleri, artan maliyet, yenilenme gereksinimleri ve gelişen teknoloji FPGA 'ları teknolojinin baş tacı yaptı. Bu sayede askeri, uzay, veri işleme, otomotiv gibi can alıcı teknoloji gerektiren alanlarda kullanımı gittikçe yaygınlaşmıştır.

Hazırlamış olduğum bitirme ödevi ile FPGA yongalarını etraflıca tanıma fırsatı buldum. Lisans öğrenimimi bitirirken FPGA; kullanım alanlarını, üretim teknolojileri ve tekniklerini, iç mimarilerini, genel kullanım yöntemlerini kabaca olsa da öğrenmiş olmaktan son derece mutluyum. Umarım bitirme ödevim benden başkalarına da faydalı olur.

8. KAYNAKLAR

- [1] MAXFIELD, C., 2004; "The Design Warrior's Guide to FPGAs"
- [2] WOLF, W. , 2004; "FPGA-Based System Design"
- [3] SICARD, E. ve BENDHIA, S. D., 2003; "Deep-submicron CMOS circuit design"
- [4] ALTERA, 2005; "Stratix II Device Handbook"
- [5] ZEIDMAN, B. , 2001 ; "Introduction to CPLD and FPGA Design"
- [6] BROWN, S. ve ROSE, J. ,2000; "Architecture of FPGAs and CPLDs: A Tutorial"
- [7] PARNELL, K. ve MEHTA, N., 2002; "Programmable Logic Design Quick Start Hand Book"
- [8] XILINX, 2005; " Virtex-II Platform FPGAs Complete Data Sheet"
- [9] ALTERA, 2005; "Cyclone Device Handbook, Volumes 1&2"
- [10] XILINX, 2005; "Virtex™ 2.5 V Field Programmable Gate Arrays Data Sheet"