SAKLAYICI – BELLEK MİMARİSİNDE, 16 – BİTLİK,
GÖMÜLÜ SİSTEM (MİKROİŞLEMCİ) TASARIMI VE SENTEZLENMESİ

İsmail GÜDENLER

Yüksek Lisans Tezi

Elektrik – Elektronik Mühendisliği Anabilim Dalı

Temmuz - 2010

SAKLAYICI – BELLEK MİMARİSİNDE, 16 – BİTLİK, GÖMÜLÜ SİSTEM (MİKROİŞLEMCİ) TASARIMI VE SENTEZLENMESİ

İsmail GÜDENLER

Dumlupınar Üniversitesi
Fen Bilimleri Enstitüsü
Lisansüstü Yönetmeliği Uyarınca
Elektrik – Elektronik Mühendisliği Anabilim Dalında
YÜKSEK LİSANS TEZİ
Olarak Hazırlanmıştır.

Danışman: Yrd. Doç. Dr. Ahmet ÖZMEN
Yardımcı Danışman: Doç. Dr. Ahmet ALTUNCU

Temmuz - 2010

KABUL ve ONAY SAYFASI

İsmail GÜDENLER' in YÜKSEK LİSANS tezi olarak hazırladığı SAKLAYICI – BELLEK MİMARİSİNDE, 16 – BİTLİK, GÖMÜLÜ SİSTEM (MİKROİŞLEMCİ) TASARIMI VE SENTEZLENMESİ başlıklı bu çalışma, jürimizce lisansüstü yönetmeliğin ilgili maddeleri uyarınca değerlendirilerek kabul edilmiştir.

...../...../.....
(Sınav tarihi)

Üye : .....

Üye : .....

Üye : .....

Fen Bilimleri Enstitüsün Yönetim Kurulu'nun ...../...../..... gün ve ..... sayılı kararıyla onaylanmıştır.

Prof. Dr. Atalay KÜÇÜKBURSA
Fen Bilimleri Enstitüsü Müdürü

SAKLAYICI – BELLEK MİMARİSİNDE, 16 – BİTLİK, GÖMÜLÜ SİSTEM (MİKROİŞLEMCİ) TASARIMI VE SENTEZLENMESİ

İsmail GÜDENLER
Elektrik - Elektronik Mühendisliği, Yüksek Lisans Tezi, 2010
Tez Danışmanı: Yrd. Doç. Dr. Ahmet ÖZMEN
Yardımcı Danışman: Doç. Dr. Ahmet ALTUNCU

ÖZET

Bu çalışmada 16 bit adres ve veri yollarına sahip, saklayıcı-bellek mimarisinde tasarlanmış bir işlemci sunulmaktadır. Tasarlanan işlemciye, 64Kx16' ya kadar dışarıdan bellek ilave edilebilmektedir. İlave edilecek belleğin, program ve veri için ortak olarak kullanılması düşünülmüştür, yani işlemci Von Neumann mimarisinde tasarlanmıştır. İşlemci tasarımında Verilog donanım tanımlama dili kullanılmıştır ve tasarım kapı seviyesinde tanımlanmıştır. İşlemci son haliyle 35 komut ile alışılmış tüm işlemleri yapabilmektedir. Bu komutların dağılımı şöyledir: 17 ALU, 10 kontrol, 5 saklayıcı işlemleri, 1 yükle-yaz ve 2 diğer. Tasarlanan işlemci, 2 adet 16-bitlik giriş/çıkış iskeleleri ve yönlendiricileri ile donatılmıştır. Bu iskelelerdeki her bit birbirinden bağımsız olarak giriş veya çıkış olarak ayarlanabilmektedir. İşlemci 4 ayrı kaynaktan gelecek kesmelere vektörlü olarak cevap verebilmektedir. Kesmeler sağlanan kontrol saklayıcılarıyla programlanabilmektedir. Gerçeklenen 16x16' lık yığın saklayıcıları sayesinde 16 defa iç-içe (nested) alt programa dallanmak mümkündür. Alt programa dallanmalarda sadece geri dönüş adresi saklanmaktadır. Tasarlanan işlemci ve işlemciye adres ve veri yolları üzerinden bağlanan grafik işlem birimi, klavye kontrolcüsü ve zamanlayıcı gibi çevresel donanımlar Xilinx Spartan 3E Starter Kit üzerinde fiziksel olarak gerçeklemiştir. FPGA içerisinde fiziksel olarak gerçeklenen sisteme, hazırlanan derleyici kullanılarak yazılan test programları yüklenmiş ve koşturulmuştur.

Anahtar Kelimeler: Bilgisayar Mimarisi, FPGA, Mikroişlemci, Programlanabilir Lojik.

DESIGN AND SYNTHESIS OF A 16 – BIT EMBEDDED SYSTEM (MICROPROCESSOR) IN REGISTER – MEMORY ARCHITECTURE

İsmail GÜDENLER
Electrical and Electronics Engineering, M. S. Thesis, 2010
Thesis Supervisor: Yrd. Doç. Dr. Ahmet ÖZMEN
Co-Supervisor: Doç. Dr. Ahmet ALTUNCU

SUMMARY

In this thesis, a processor core which is designed in register - memory architecture is presented. Designed processor core has 16 bits address and data buses. 64Kx16 bits of memory can be connected externally which is used for both instruction and data memories. So the processor core is designed in Von Neumann architecture. Verilog HDL is used while designing the processor and the design is described at gate level. The processor core can execute all familiar processor operations by using 35 instructions. The distribution of these instructions is as follows: 17 ALU, 10 control, 5 register operations, 1 load-store and 2 other. Designed processor core has 2 16-bits input - output ports and data direction registers. All of the bits in these ports can be configured as inputs or outputs independently. The processor core can respond interrupts from 4 different resources by using pre-defined interrupt vectors. These interrupts can be programmed via interrupt control registers. Owing to implemented 16x16 stack registers, it is possible to branch 16 nested subroutines. Only return address is stored while branching to subroutines. Designed processor core and the peripheral hardware like graphical processing unit, keyboard controller and timer which are connected to processor core through address and data buses, are physically implemented on Xilinx Spartan 3E Starter Kit. The test software which is programmed using by designed compiler is loaded into the system and executed successfully.

Keywords: Computer Architecture, FPGA, Microprocessor, Programmable Logic.

TEŞEKKÜR

Bu tez çalışması boyunca eşsiz yardım ve katkılarını esirgemeyen, yol gösteren değerli danışman hocalarım Yrd. Doç. Dr. Ahmet ÖZMEN‘ e ve Doç. Dr. Ahmet ALTUNCU’ ya teşekkürü bir borç bilirim.

Ayrıca derleyici programı ve donanımsal tasarım sürecindeki yardımlarından dolayı değerli dostum Ercan Doğan’ a, yüksek lisans öğrenimim boyunca maddi manevi desteğinden dolayı TÜBİTAK BİLİM İNSANI DESTEKLEME DAİRE BAŞKANLIĞI (BİDEB)’ na, sağladıkları donanımsal ve yazılımsal desteklerden dolayı Çizgi Elektronik ailesine teşekkür ederim.

Son olarak varlıklarını hep yanımda hissettiğim, hayatım boyunca desteklerini benden esirgemeyen aileme teşekkürlerimi sunarım.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (Devam)

Sayfa

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

SİMGELER ve KISALTMALAR DİZİNİ

Kısaltmalar	Açıklama
CPU	Central Processing Unit (Merkezi İşlem Birimi)
ALU	Aritmetic Logic Unit (Aritmetik Mantık Birimi)
ISA	Instruction Set Architecture (Komut Kümesi Mimarisi)
HSA	Hardware System Architecture (Donanım Sistem Mimarisi)
CISC	Complex İnstruction Set Computer (Karmaşık Komut Setli Bilgisayar)
RISC	Reduced İnstruction Set Computer (İndirgenmiş Komut Setli Bilgisayar)
EPIC	Explicitly Parallel İnstruction Computer (Açık Paralel Komutlu Bilgisayar)
RAM	Random Access Memory (Rastgele Erişimli Bellek)
ROM	Read Only Memory (Salt Okunur Bellek)
HDL	Hardware Description Language (Donanım Tanımlama Dili)
VHDL	Very High Speed Integrated Circuit Hardware Description Language (Çok Hızlı Tümdevre Donanım Tanımlama Dili)
RTL	Register Transfer Level (Saklayıcı Transfer Seviyesi)
VLSI	Very Large Scale Integration (Çok Büyük Ölçekli Entegrasyon)
PLD	Programmable Logic Device (Programlanabilir Lojik Devre)
PLA	Programmable Logic Array (Programlanabilir Lojik Dizi)
PAL	Programmable Array Logic (Programlanabilir Dizi Lojiği)
CPLD	Complex Programmable Logic Device (Karmaşık Programlanabilir Lojik Devre)
FPGA	Field Programmable Gate Array (Sahada Programlanabilir Kapı Dizisi)
MPGA	Mask Programmable Gate Array (Maske Programlamalı Kapı Dizisi)
ASIC	Application Spesific Integrated Circuit (Uygulamaya Özel Entegre)
PROM	Programmable Read Only Memory (Programlanabilir Salt Okunur Bellek)
EPROM	Erasable Programmable Read Only Memory (Silinebilir Programlanabilir Salt Okunur Bellek)
EEPROM	Electronically Erasable Programmable Read Only Memory (Elektrikle Silinebilir Programlanabilir Salt Okunur Bellek)
SRAM	Static Random Access Memory (Statik Rastgele Erişimli Bellek)

SİMGELER ve KISALTMALAR DİZİNİ (Devam)

Kısaltmalar	Açıklama
CLB	Configurable Logic Blocks (Düzenlenebilir Lojik Bloklar)
LC	Logic Cell (Lojik Hücre)
LUT	Look-Up Table (Doğruluk tablosu)
MUX	Multiplexer (Çoğullayıcı)
FF	Flip – Flop
LAB	Logic Array Block (LAB)
LM	Logic Module (Lojik Modül)
IOB	Input/ Output Blocks (Giriş/Çıkış Blokları)
FIFO	First In First Out (İlk Giren İlk Çıkar)
MAC	Multiply and Accumulate (Çarp ve Biriktir)
DCM	Digital Clock Manager (Dijital Saat Yöneticisi)
ISP	In System Programmable (Sistem İçi Programlanabilir)
PCB	Printed Circuit Board (Baskılı Devre Kartı)
VGA	Video Graphics Array (Video Grafik Dizisi)
ASCII	American Standard Code For İnformation Interchange

1. GİRİŞ

Sürekli gelişen ve karmaşıklığı artan elektronik sistemler içinde mikroişlemcilerin önemli bir yeri vardır. Günümüzde mikroişlemciler ve mikrodenetleyiciler hemen hemen her alanda karşımıza çıkmaktadırlar. Mikroişlemciler genellikle masaüstü bilgisayarlar ve sunucu tabanlı makineler gibi yoğun ve hızlı işlem gücü gerektiren alanlarda kullanılır ve harici hafıza ve giriş çıkış birimleri ile birlikte çalışırlar. Mikrodenetleyiciler ise gömülü sitemlerde, kontrol uygulamalarında karşımıza çıkmaktadırlar ve hafıza ve giriş çıkış birimleri genellikle kullandıkları işlemci çekirdeği ile aynı yonga içerisinde bulunur. Gömülü sitemlerde gerçek zamanlı çalışma, zamanlamanın kritik olduğu uygulamalarda çok önemlidir. Bazı uygulamalarda işlemcinin işlemesi gereken her komutun kaç saat darbesi süreceğinin ve bir kesme sinyali algılandığında kesme vektörüne kaç saat darbesi sonra ulaşılacağının kesin olarak bilinmesi gerekebilir. Zamanlama açısından kritik kod parçaları makine dilinde yazılacağından gerçek zamanlı bir işlemcinin makine dilinde kolayca programlanabilir olması gereklidir.

Mikroişlemci tasarımındaki çalışmalar transistörün bulunduğu 1948 yılından günümüze kadar devam etmektedir. 1971 yılında Intel' in ilk işlemcisi 4004 sadece 740 kHz' de çalışabilirken bugün masaüstü bilgisayarlarda kullandığımız mikroişlemciler 3 GHz' in üzerindeki hızlarda çalışabilmektedirler [3]. Mikroişlemcilerdeki gelişme sadece saat frekanslarında değil, mimari yapılarında gerçekleşmiştir. İlk tasarımlara göre çok karmaşık yapıda olan günümüz işlemcileri, cache (tampon) bellek, pipeline, floating point unit (kayar noktalı işlem birimi), memory management unit (hafıza yönetim birimi), prefetch (önceden komut çekme), branch prediction gibi birimler içermektedirler.

Elektronik sistemlerin yapılarındaki giderek artan karmaşıklık nedeniyle kağıt kalem kullanılan geleneksel devre tasarım yöntemleri, yerini tanımlama ve sentezleme yöntemlerine bırakmıştır [5]. Bu yeni yöntemler, VHDL (Very High Speed Integrated Circuit Hardware Description Language) ve Verilog HDL gibi donanım tanımlama dillerinin geliştirilmesi ile ortaya çıkmıştır. Programlanabilir lojik sistemlerin kapasitesindeki ve maksimum saat frekansındaki artış donanım tanımlama dillerinin gelişmesiyle paralel olmuştur [3]. 1990' lardan sonra FPGA (Field Programable Gate Array)' ler hızlı tasarım süreçleri nedeniyle ASIC (Application Specific Integrated Circuit)' lerin yerini almaya başladılar [4]. Günümüzde FPGA mimarisini kullanan sistemlerin tasarımında farklı yöntemler izlenebilmektedir fakat en çok tercih edilen yöntem, donanım tanımlama dilleridir. FPGA' ler yapılan tasarımın, Verilog yada VHDL gibi bir yüksek seviye donanım tanımlama dili aracılığıyla, yonga üretiminde kullanılan

teknoloji ile ilgilenmeden RTL (Register Transfer Level) seviyesinde geliştirilmesine imkan verir.

FPGA bilgisayar mimarisi eğitimi için de ideal bir ortamdır [7]. FPGA içerisinde gerçeklenen gömülü sitemlerde kullanılan işlemci çekirdeğinin yanına istenilen çevresel donanımlar eklenebilir. Böylece tasarımda esneklik sağlanır. FPGA' in tekrar programlanabilme özelliği, gömülü sistemdeki donanımın istenildiğinde güncellenmesine imkan verir. FPGA içerisindeki gerçeklenen gömülü sisteme ait işlemcide tasarımcının tanımladığı özel komutlar bulunabilir, bazı komutlar co-processor (yardımcı işlemci) yardımıyla işlenebilir. Önde gelen FPGA üreticileri XILINX ve ALTERA' nın FPGA içerisinde gerçeklenen ve ihtiyaca göre bazı özellikler eklenip çıkartılabilen mikroişlemci çekirdekleri vardır. XILINX FPGA' lerde küçük uygulamalar için 8-bit Picoblaze, daha kapsamlı uygulamalar için pipeline, floating point ve memory management unit gibi gelişmiş özellikleri bulunan 32-bit Microblaze kullanılabilir. ALTERA FPGA' lerde ise 32-bit NIOS-II kullanılabilmektedir. FPGA tabanlı birkaç mikroişlemci çalışması literatürde verilmiştir [3, 2, 9].

Bu çalışmada, eğitim amaçlı gerçek zamanlı uygulamalarda kullanılabilecek 16-Bit veri ve adres yolu genişliğine sahip olan, bellek-saklayıcı mimarisini kullanan ve tamsayılarla işlem yapabilen bir gömülü sistem tasarlanıp gerçeklenmiştir. Gömülü sistemde 32 adet 16 bitlik saklayıcıları içeren bir saklayıcı dizisi ve A (akümülatör) saklayıcısı bulunmaktadır. Bu saklayıcı dizisinde bulunan saklayıcıların 14'ü sistem tarafından veya giriş/çıkış birimleri tarafından kontrol saklayıcısı olarak kullanılmakta olup, 18 adet saklayıcı ana bellekten daha hızlı erişilmek üzere kullanıcıya bırakılmıştır. Sistem Von Neumann mimarisinde olup veri ve program belleği olarak 64 K-Word dahili (sentezleme aşamasında) veya harici bellek ilave edilebilmektedir. Sistem her biri bit bazında giriş veya çıkış olarak yönlendirilebilen 2 adet 16-Bitlik giriş çıkış iskelesi ile donatılmıştır; ayrıca, dört farklı harici kaynaktan gelebilecek kesmeler için "vektörlü kesmeyi" desteklemektedir. Sistemde 16 Word derinliğinde bir yığın gerçeklenmiş olup, alt programlara ve kesme servis rutinlerine dallanmalarda geri dönüş adresi bu yığına atılmaktadır. İşlemci çarpma ve bölme komutları dahil 35 komut ile ihtiyaç duyulan tüm işlemleri etkin bir şekilde gerçekleyebilmektedir. Komutlar yapısına göre ivedi, doğrudan, doğal, dolaylı ve sıralı adresleme modlarını kullanabilmektedir. Komutların icra süreleri: 2 (doğal), 3 (ivedi) 4 (doğrudan), 5 (sıralı, dolaylı) saat darbesi sürmekte; çarpma ve bölme işlemleri ilave olarak 16 saat darbesi daha almaktadır.

Gömülü sistem modüler olarak kapı seviyesinde tasarlanmış olup, her bir modül son sisteme dahil edilmeden önce davranışsal eşleniği ile kapsamlı bir teste tabi tutulmuştur. Tasarlanan modüller ve tüm sistem Verilog-HDL ile kodlanmış olup derleyici ve editör olarak

XILINX ISE 10.1 ortamı kullanılmıştır. Modüllerin tasarım ve test aşamasından sonra, tüm işlemci simülasyon yoluyla test edilmiş ve üzerinde örnek program koşturulmuştur. Simülasyonda MODELSIM SE Verilog kullanılmış olup kesme girişlerini test etmek için XILINX ISE programındaki Test Bench Waveform yardımı ile farklı kesme girişlerine farklı anlarda kesme sinyali uygulanmıştır. Tüm test ve simülasyon aşamaları geçildikten sonra, gömülü sistem Xilinx HW-SPAR3E-SK geliştirme kartı üzerinde gerçeklenmiştir.

2. BİLGİSAYAR SİSTEMLERİ

Bilgiyi giriş olarak alan, bunu belli bir kurala göre işleyen ve sonucu çıktı olarak veren sistemlere basit olarak bilgisayar denir. Makine olarak tanımlanan bilgisayar, veriyi belli bir düzen dahilinde işler. Buradaki veri, işlenecek bilgidir. Verinin işleniş düzenini veya kuralları donanımın dışında komutlar (program) koyar. Sayısal değerler belli bir formatta sisteme yerleştirilmek zorundadır. Sistemdeki herhangi bir fiziksel veya mantıksal parametre ikilik sayılarla ifade edilmektedir. Bilgisayar sistemleri iki temel öğeden oluşmaktadır. Bunlar yazılım ve donanımdır. Her ikisi de birbirinin tamamlayıcıdır, birisi olmazsa diğeri de olmaz. Sistem öncelikli olarak tasarlanırken önce sistemi meydana getirecek elemanlar, yani donanım parçaları göz önüne alınır. Daha sonra yazılım bu yapıya bakılarak yazılır. Yazılım, donanımın hangi yönteme göre nasıl çalışacağını gösteren bir sanal uygulamadır. Donanım ise yazılıma göre belirli zamanlarda devreye girerek fonksiyonlarını yerine getirmekle görevlidir.

Bilgisayarı oluşturan bir sistemdeki temel elemanlar; mikroişlemci (CPU), bellek ve giriş/çıkış (G/Ç) birimleridir. Mikroişlemcinin işleyeceği komutlar ve veriler geçici veya kalıcı belleklerde tutulmaktadır. Bilgiyi oluşturan komut ve veriler bellekte karmaşık veya farklı alanlarda tutulabilir. Bilginin işlenmesi sırasında ortaya çıkabilecek ara değerler, en sonunda sonuçlar bellekte bir yerde depolanmak zorundadır.

Bilgisayarın bilgiyi işlemedeki ana karar vericisi sistemin kalbi sayılan mikroişlemcidir. CPU tarafından gerçekleştirilen iki temel işlem vardır. Birincisi, komutların yorumlanarak doğru bir sırada gerçekleşmesini sağlayan kontrol işlevi, diğeri; toplama, çıkarma ve benzeri özel matematik ve mantık işlemlerinin gerçekleştirilmesini sağlayan icra işlevidir.

Ayrıca sistemin veri giriş çıkışı yapabilmesi için giriş/çıkış birimine gerek vardır. G/Ç birimi, makine ile kullanıcı (veya programcı) arasında bilginin makine dilinden insanın anlayacağı dile çevrilmesinde veya tersi işlemde iletişim (aracı) sağlar.

Sistemin öne çıkan diğer elemanları iletişim yollarıdır. Adres yolu, veri yolu ve kontrol yolu olarak üçe ayrılan iletişim yolları, bilgisayar sistemindeki birimler arasında bilginin taşınmasından sorumludur. Adres yoluna bellekten getirilerek çalıştırılmak istenen komut adresi veya komutun işlenmesiyle bellekten getirilecek verinin adresi konulur. Sonuç olarak, her bilgisayar aşağıdaki elemanlara sahip olmalıdır:

1. Programın yorumlanması ve çalıştırılmasını gerçekleştiren bir mikroişlemci.
2. Bir dizi komutlardan oluşan program ve verilerin sürekli veya geçici depolandığı bellek.

3. Bilgisayarın dış dünya ile bağlantısını sağlayan giriş/çıkış birimi.
4. CPU ve bellek aracındaki bilgi aktarımını ve işlemcinin dış dünya ile iletişimini sağlayan iletişim yolları.

Bilgisayar sistemi tarif edilirken iki temel esastan bahsedilebilir. Bilgisayar organizasyonu ve bilgisayar mimarisi. Bilgisayar mimarisi, bir programın mantıksal çalışmasına doğrudan etki eden özelliklerdir. Bilgisayar organizasyonu, operasyonel birimler ve bunların yapısal özelliklerini veren bağlantıları ifade eden yaklaşımdır; Daha çok yazılım ve donanım arasındaki bağdaştırmayla ilgilidir. Mimari özelliklere komut kümesi, değişik şekillerdeki veri tiplerini temsil etmesi için kullanılan bit sayısı, G/Ç mekanizması ve bellek adresleme tekniklerinin dahil olduğu bir bilgisayar tasarımı girmektedir. Bilgisayar mimarisi, komut kümesinin, donanım elemanlarının ve sistem organizasyonun dahil olduğu bir bilgisayarın tasarımıdır. Mimari iki farklı yaklaşımla tanımlanmaktadır Komut kümesi mimarisi (ISA) ve donanım sistem mimarisi (HSA). ISA, bir bilgisayarın hesaplama karakteristiklerini belirleyen komut kümesinin tasarımıdır. HSA, ise CPU, depolama ve G/Ç sistemlerinin dahil olduğu alt sistem ve bunların bağlantı şeklidir.

2.1. Komut Kümesi Mimarisi

İlk bilgisayar sistemleri için programcı kodlarını makinenin doğrudan özel donanımına göre yazmaktaydı. Bu nedenle bir makine için yazılan program aynı firma tarafından üretilse bile diğer bir makinede çalışmamaktaydı. Programcı tarafından yazılan kodlar donanımı açma anahtarı olarak düşünülebilirler. Her zaman yeni bir makine üretildiğinde yazılım geliştiriciler bu makine için yeni baştan başlamak zorunda kalmaktaydılar. Bundan dolayı bilgisayar sistem tasarımcıları iki önemli sorunla karşılaştılar.

1. Bilgisayar sistemleri ile ilgili işlevselliğin sergilenmesi.
2. Bir dizi komutlardan oluşan program ve verilerin sürekli veya geçici depolandığı bellek.
3. Yazılımın sistemler arasındaki geçişini kolaylaştırması.

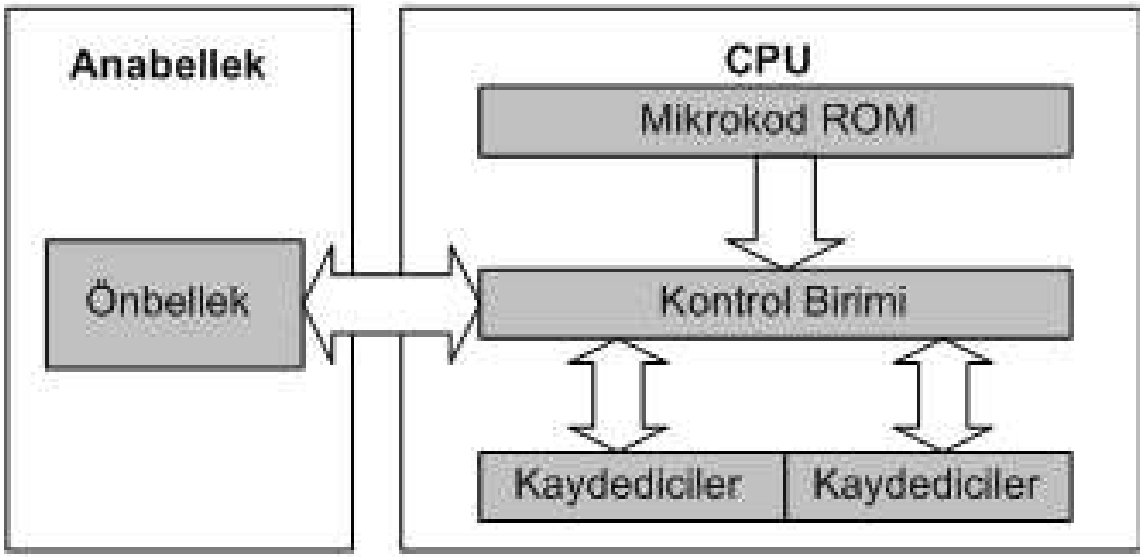
1960 yıllarında IBM firması bu sorunların üstesinden gelmek için, adına komut kümesi mimarisi (ISA) ve mikrokod motoru denilen bir yöntem geliştirmiştir

Mikrokod kullanılarak ISA sisteminin yürütülmesinin başlıca sakıncası, başlangıçta komutların doğrudan çalıştıran sisteme göre yavaş olmasıdır. Daha çok komut demek daha fazla mikrokod, çekirdek büyüklüğü ve güç demektir.

ISA mimarisinin yaşanan aksaklıklarından dolayı daha sonraları, komutların doğrudan donanım elemanları tarafından yorumlanarak sistemin denetlendiği diğer bir mimari yaklaşımı da donanımsal çalışma modelidir. Komutların anlaşılır standartta bir boyuta getirilerek çalıştırıldığı sisteme RISC modeli denilmektedir. Böylece küçük, hızlı ve çok hafifleyen komut kümesiyle, iri hacimli mikrokoda nazaran donanım üzerinde doğrudan hakimiyet kolayca sağlanabilmiştir. RISC tasarımcıları komutların doğrudan icra edildiği eski modele dönerken, ISA kavramı dokunulmadan korunmuştur. Intel, AMD gibi büyük firmalar hala x86 mimarisine dayalı işlemcilerini ISA yaklaşımıyla üretmektedirler. Günümüzde üst düzey entegrasyon ve yarı iletken üretim teknolojilerinin elde edilmesiyle çok daha karmaşık olan donanım temelli sistemler oluşturmak mümkün olmakladır.

2.1.1. CISC mimarisi

CISC mimarisinin karakteristik iki özelliğinden birisi, değişken uzunluktaki komutlar, diğeri ise karmaşık komutlardır. Değişken ve karmaşık uzunluktaki komutlar bellek tasarrufu sağlar. Karmaşık komutlar İki ya da daha fazla komutu tek bir komut haline getirdikleri için hem bellek alanından hem de programda yer alması gereken komut sayısından tasarruf sağlar [8]. Karmaşık komut karmaşık mimariyi de beraberinde getirir. Mimarideki karmaşıklığın artması, işlemci performansında istenmeyen durumların ortaya çıkmasına sebep olur. Ancak programların yüklenmesinde ve çalıştırılmasındaki düşük bellek kullanımı bu sorunu ortadan kaldırabilir.



Şekil 2.1 CISC Mimarisi

Tipik bir CISC komut seti, değişken komut formatı kullanan 120-350 arasında komut içerir. Bir düzineden fazla adresleme modu ile iyi bir bellek yönetimi sağlar. CISC mimarisi çok kademeli işleme modeline dayanmaktadır. İlk kademe, yüksek düzeyli dilin yazıldığı yerdir. Sonraki kademeyi makine dili oluşturur ki, yüksek düzeyli dilin derlenmesi sonucu bir dizi komutlar makine diline çevrilir. Bir sonraki kademede makine diline çevrilen komutların kodları çözülerek, mikroişlemcinin donanım birimlerini kontrol edebilen en basit işlenebilir kodlara dönüştürülür. En alt kademede ise işlenebilir kodları olan donanım aracılığıyla gerekli görevler yerine getirilir.

CISC Mimarisinin Avantajları:

- Mikroprogramlama, yürütülmesi kolaydır ve sistemdeki kontrol biriminden daha ucuzdur.
- Yeni komutlar ve mikrokod ROM'a eklemenin kolaylığı tasarımcılara CISC makinalarını geriye doğru uyumlu yapmalarına izin verir. Yeni bir bilgisayar aynı programları ilk bilgisayarlar gibi çalıştırabilir.
- Verilen görevi yürütmek için daha az komut kullanır.
- Mikroprogram komut kümeleri, derleyici karmaşık olmak zorunda değildir.

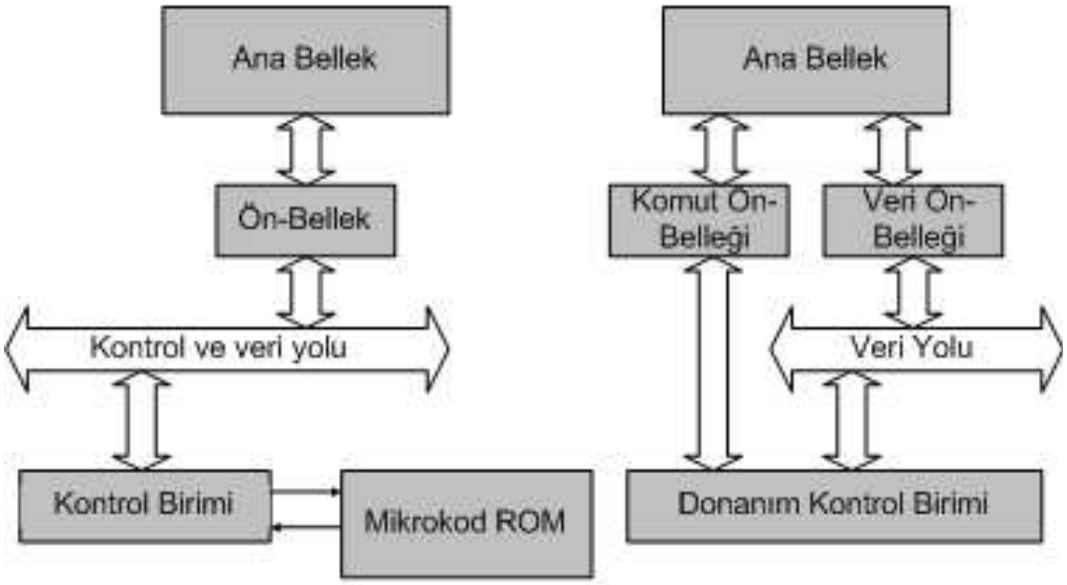
CISC Mimarisinin Dezavantajları:

- İşlemci ailesinin ilk kuşakları genelde her yeni versiyon tarafından kabullenilmiştir, böylece komut kodu ve yonga donanımı bilgisayarların her kuşağıyla birlikte daha karmaşık hale gelmiştir.
- Mümkün olduğu kadar çok komut, mümkün olan en az zaman kaybıyla belleğe depolanabiliyor ve komutlar neredeyse her uzunlukta olabiliyor. Bunun anlamı farklı komutlar farklı miktarda saat çevrimi tutacaktır.
- Çoğu özel güçlü komutlar geçerliliklerini doğrulamak için yeteri kadar sık kullanılmıyor.
- Komutlar genellikle bayrak (durum) kodunu komuta bir yan etki olarak etkinleştirir. Bu ise ek saat darbeleri yani bekleme demektir. Aynı zamanda, sıradaki komutlar işlem yapmadan önce bayrak bitlerinin mevcut durumunu bilmek durumundadır. Bu da yine ek saat darbesi demektir.

2.1.2. RISC mimarisi

RISC mimarisi, CISC mimarili işlemcilerin kötü yanlarına piyasanın tepkisi ve ona bir alternatif olarak, işlemci mimari tasarımlarında söz sahibi olan IBM, Apple ve Motorola gibi

firmalarca sistematik bir şekilde geliştirilmiştir. 1970' lerin ortalarında yarı iletken teknolojisindeki gelişmeler, ana bellek ve işlemci yongaları arasındaki hız farkını azaltmaya başladı. Bellek hızı arttığından ve yüksek seviyeli diller simgesel dilin yerini aldığından, CISC' in başlıca üstünlükleri geçersizleşmeye başladı. Bilgisayar tasarımcıları sadece donanımı hızlandırmaktan çok bilgisayar performansını iyileştirmek için başka yollar denemeye başladılar. Aşağıda günümüz bilgisayarlarında kullanılan, RISC tabanlı fakat mikrokod desteği de sağlayan bir işlemcinin blok şeması görülmektedir.



Şekil 2.2 RISC Mimarisi

IBM, RISC mimarisini tanımlayan ilk şirket olarak kabul edilir. RISC' in felsefesi üç temel prensibe dayanır:

- Bütün komutlar eşit sayıda çevrimde çalıştırılmalıdır.
- ALU işlemlerinin tamamı saklayıcılarda yapılamalıdır.
- Belleğe sadece "load" ve "store" komutlarıyla erişilmelidir: Komut alınıp getirilir ve bellek gözden geçirilir. RISC işlemcisiyle, belleğe yerleşmiş veri bir kaydediciye yüklenir, kaydedici gözden geçirilir ve kaydedicinin içeriği ana belleğe yazılır.
- Bütün icra birimleri mikrokod kullanmadan donanımdan çalıştırılmalıdır: Mikrokod kullanımı, dizi ve benzeri verileri yüklemek için çok sayıda çevrim demektir.

Günümüzün RISC yapısına sahip ticari mikroişlemcilerinde genel olarak iki tarz görülür. Bunlar Berkeley modeli ve Stanford modelidir. RISC mimarisi aynı anda birden çok

komutun birden fazla birimde işlendiği iş-hattı (pipelining) tekniği ve süperskalar yapılarının kullanımıyla yüksek bir performans sağlamıştır.

RISC Mimarisinin Avantajları:

RISC tasarımı olan bir işlemciyi kullanmak, bir CISC tasarımını kullanmaya göre pek çok avantaj sağlar:

- Hız: Azaltılmış komut kümesi pipeline ve superskalar tasarıma izin verdiğinden RISC işlemciler genellikle karşılaştırabilir yarı iletken teknolojisi ve aynı saat oranları kullanılan CISC işlemcilerinin performansının 2 katından daha yüksek performans gösterirler.
- Basit Donanım: RISC işlemcinin komut kümesi çok basit olduğundun çok az yonga uzayı kullanırlar.
- Kısa Tasarım Zamanı: RISC işlemciler CISC işlemcilere göre daha basit olduğundan daha çabuk tasarlanabilirler ve diğer teknolojik gelişmelerin avantajlarını CISC tasarımlarına göre daha çabuk kabul edebilirler.

RISC Mimarisinin Dezavantajları:

CISC tasarım stratejisinden RISC tasarım stratejisine yapılan geçiş kendi problemlerini de beraberinde getirmiştir. Donanım mühendisleri kodları CISC işlemcisinden RISC işlemcisine aktarırken geriye uyumluluğu göz önünde bulundurmak zorundadırlar.

2.1.3. EPIC mimarisi:

RISC mimarisinin altında yatan iki ana tasarım amacı vardır. Birincisi; derleyicinin kullanmadığı veya kullanamadığı komutlar ve adresleme modlarından kurtulmak. İkincisi; Öyle bir çekirdek yaratılmalı ki ileride superölçekli mimariyi oluşturacak iş-hattını kolaylaştırsın. Komutların aynı anda farklı birimlerde farklı şekilde çalıştırıldığı ortamlar Süperölçekli mimari olarak adlandırılır. RISC mimarisinde kullanılan bu süperölçekli yapının tasarlanmasında iki önemli sorun ortaya çıkmaktadır. Birincisi; Komut kümesinde bulunan komutlardan hangilerinin paralel çalıştırılabileceğine karar verilmesi. İkincisi; Paralel çalıştırılabilecek yeterli komutların bulunabilmesi. Bu problemlerin üstesinden gelmek için Intel ve benzer işlemci firmaları yonga alanlarının büyük bir kısmını harcamaktadırlar. EPIC mimarisi işte bu sorunların üstesinden gelmek için tasarlanmıştır.

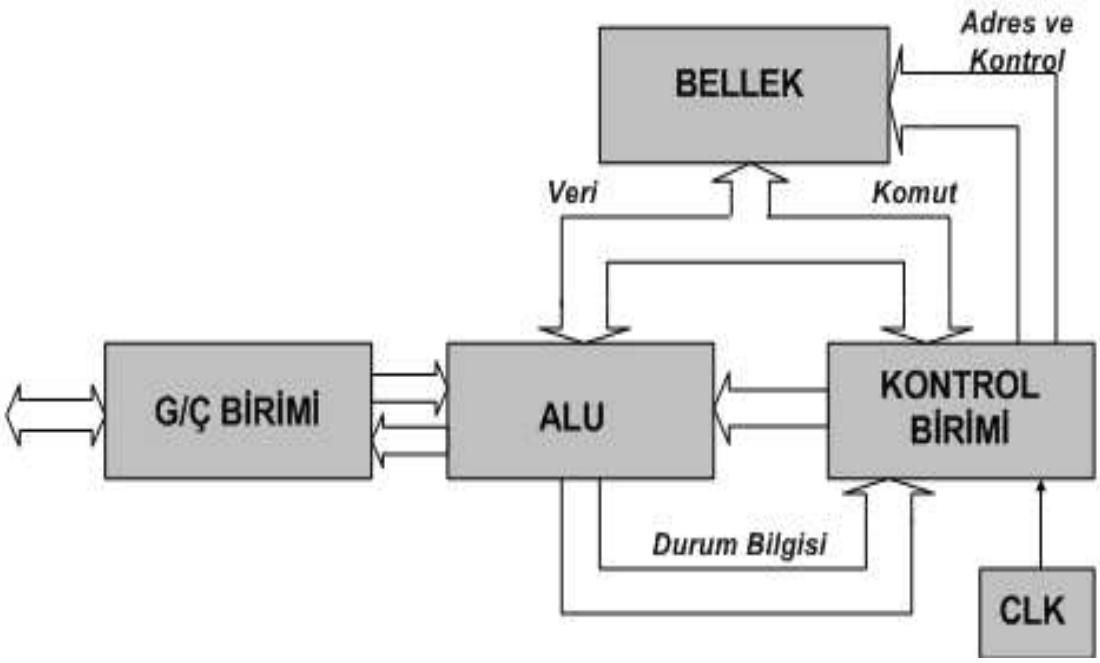
2.2. Donanım Sistem Mimarisi

Bilgisayarın, yüklenen tüm görevleri çok kısa zamanda yerine getirmesinde yatan ana unsur mimarisidir. Mimari, komutların ve verinin bellek ile kaydediciler arasında taşınmasında

kullanılan iletişim yollarının düzeni, kaydedici sayısı ve büyüklükleri gibi tasarım kararlarının dahil olduğu, işlemci yeteneklerinin nihai sonucunu gösteren bir yapıdır. Bilgisayar mimarisinin tasarımı iki yaklaşım üzerinde yoğunlaşmıştır. Bunlar, Von Neumann ve Harvard mimarileridir.

2.2.1. Von Neumann mimarisi

Bilgisayarlarda ilk kullanılan ve adını mimariye yön veren ünlü matematikçi John von Neumann' dan alan bir tasarım yaklaşımıdır. Bazen Princeton mimarisi de denilen bu mimari yapıya sahip ilk bilgisayarlarda, transistör yerine lamba (vakum tüp) kullanılmaktaydı.



Şekil 2.3 Von Neumann Mimarisi

Von Neumann' ın fikrinden yol çıkılarak J. P. Eckert ve J. Mauchly tarafından 1946 yılında ilki geliştirilen bilgisayar beş birimden olunmaktadır. Bunlar; Aritmetik ve mantık birimi, kontrol birimi, bellek, giriş-çıkış birimi ve bu birimler arasında iletişimi sağlayan yollardır.

Von Neumann mimarisinde, veri ve komutlar bellekten tek bir yoldan mikroişlemciye getirilerek işlenmektedir. Program ve veri aynı bellekte bulunduğundan, komut ve veri gerekli olduğunda aynı iletişim yolu kullanılmaktadır. Bu durumda, komut için bir fetch saykılı, sonra veri için diğer bir fetch saykılı gerekmektedir.

Von Neumann mimarisine sahip bir bilgisayar aşağıda sıralı adımları gerçekleştirir:

1. Program sayıcısının gösterdiği adresten (bellekten) komutu algetir.
2. Program sayıcısının içeriğini bir artır.

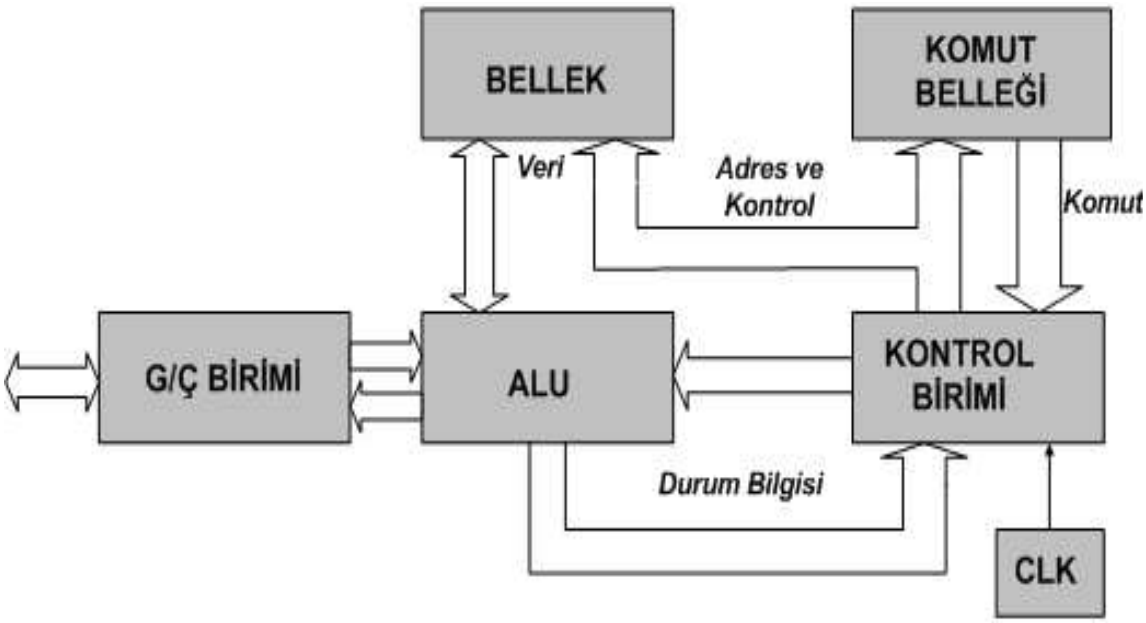
3. Getirilen komutun kodunu kontrol birimini kullanarak çöz. Kontrol birimi, bilgisayarın geri kalan birimlerine sinyal göndererek bazı operasyonlar yapmasını sağlar.
4. 1. adıma geri dönülür.

Von Neumann mimarisinde, veri bellekten alınıp işledikten sonra tekrar belleğe gönderilmesinde çok zaman harcanır. Bu işlemler bilgisayarı yavaşlattığından, bilgisayar tasarımcılarının tabiriyle bir darboğaz oluşturmaktadır. Bundan başka, veri ve komutlar aynı bellek biriminde depolandığından, yanlışlıkla komut diye veri alanından kod getirilmesi sıkıntılara sebep olmaktadır. Bu mimari yaklaşıma sahip olan bilgisayarlar günümüzde, verilerin işlenmesinde, bilginin derlenmesinde ve sayısal problemlerde olduğu kadar endüstriyel denetimlerde başarılı bir şekilde kullanılmaktadır.

Bellek erişiminde, hızlı belleklerden sayılan ön-bellek (cache) sistemlerinin kullanılmasıyla büyük bant genişliği ve düşük gecikme elde edilerek Von Neumann mimarisinin darboğazı aşılabilir.

2.2.2. Harvard mimarisi

Harvard mimarili bilgisayar sistemlerinde, Von Neumann mimarisinden farklı olarak veri ve komutlar ayrı belleklerde tutulmakladır. Buna göre, veri ve komut aktarımında iletişim yolları da bir birinden bağımsız yapıdadır. Komutla birlikte veri aynı saat darbesinde farklı iletişim yolundan ilgili belleklerden alınıp işlemciye getirilebilir. Getirilen komut işlenip ilgili verisi veri belleğinden alınırken sıradaki komut, komut belleğinden alınıp getirilebilir. Bu önden alıp getirme işlemi, dallanma haricinde hızı iki katına çıkarabilmekledir.



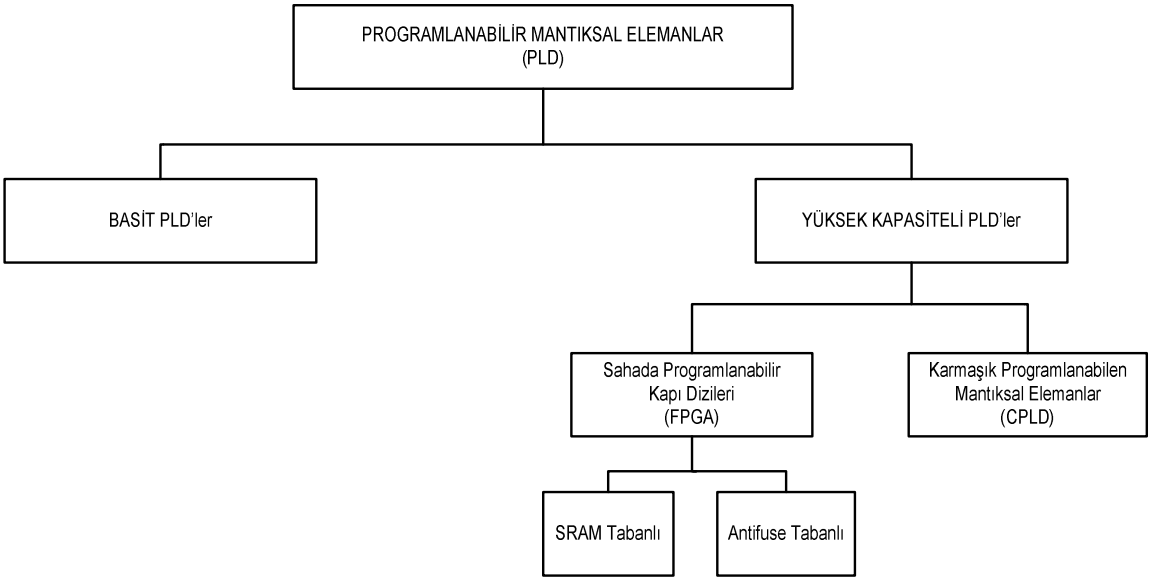
Şekil 2.4 Harvard Mimarisi

Harvard mimarisi günümüzde daha çok sayısal sinyal işlemcilerinde (DSP) kullanılmaktadır. Bu sistemlerde adres uzayı, komutların bulunduğu program belleği ve çeşitli gruplara bölünmüş veri bellekleri olmak üzere üç alana bölünebilir. Mikroişlemci her bir komut çevriminde tüm belleğe erişebilir. Günümüz bilgisayarlarında, bellekle tek yoldan iletişim ve komutla verinin aynı bellekle bulunması problemi ön-bellek sistemleri ile çözülmüştür. Ön-bellekler, komut ve veri olmak üzere ikiye ayrılmış ve işlemcinin içerisine yerleştirilmiştir. İşletim sistemi tarafından ön-belleğin kapasitesine göre ana bellekten veriler ön-belleklere alınır. Önbellek denetleyicisi tarafından komut ve veriler ayrıştırılarak ilgili birimlere yerleştirilir. Mikroişlemci tarafından, komut ön-belleğinden, data veri önbelleğinden alınarak işlenir. Bu işlem, hızlı önbelleklerin ve birbirinden ayrı komut ve veri ön-belleklerin kullanılması bilgisayarın performansını artırmaktadır. Ön-bellek miktarı ne kadar fazla olursa o kadar iyi olmaktadır, fakat maliyeti çok yüksektir.

3. PROGRAMLANABİLİR LOJİK TEKNOLOJİLERİ

Günümüzde çok geniş ölçekli tümdevre (VLSI) tasarımı için farklı seçenekler vardır. Bunların içerisinde yer alan programlanabilir lojik elemanlar (PLDs) en çok kullanılan yapılardır. PLD kısaca kullanıcı tarafından programlanabilen tümdevreler için kullanılan genel bir kavramdır. Üretilmesi planlanan ürün miktarı ve fiyat göz önüne alındığında eğer performans gereksinimleri de karşılanıyorsa programlanabilir elemanların kullanılması tasarım sürecini kısaltmaktadır. Klasik tasarım sürecinde tasarım tamamlandıktan sonra ürünün ortaya çıkması aylar sürmektedir. Tasarımda herhangi bir yanlışın fark edilmesi durumunda ise üretim için harcanan maliyet ve süre tekrar ödenir. Bu maliyet ve sürenin kısaltılması için Maske Programlamalı Kapı Dizileri (MPGA) kullanılmıştır ancak bu teknolojiyle tasarım devreyi gerçekleştirilecek firma tarafından oluşturulur. Gerçekleştirilmesi istenen lojik devreye bağlı olarak metal tabaka üretici firma tarafından son aşamada oluşturulmaktadır. Ancak bu yöntem de tasarım esnekliğini tamamen kullanıcıya bırakmamaktadır. Kullanıcı tarafından programlanabilen yapılar üzerine çalışmalar 1970lerde programlanabilir salt okunur bellekler (PROM) ile başlamıştır. Sahada programlanabilir PROM' un, silinebilir programlanabilir salt okunur bellek (EPROM) ve elektriksel olarak silinebilir ve programlanabilir salt oku bellek (EEPROM) olarak iki tipi geliştirilmiştir. EEPROM defalarca silinebilme ve programlanabilme avantajına sahiptir.

PLD alanındaki gelişmeler sonucunda bu teknoloji kendi içinde birçok gruba ayrılmıştır ve farklı amaçlarla kullanılabilecek yapılar ortaya çıkmıştır. Bu elemanlar temelde lojik devreleri gerçeklemek amacıyla kullanılmaktadır. PLD' ler karmaşıklığına göre basit ve yüksek kapasiteli olmak üzere iki ana gruba ayrılmaktadır. Yüksek kapasiteli olanlar gerçekleştiği fonksiyonların karmaşıklığına göre de FPGA ve CPLD şeklinde alt gruplara ayrılır.

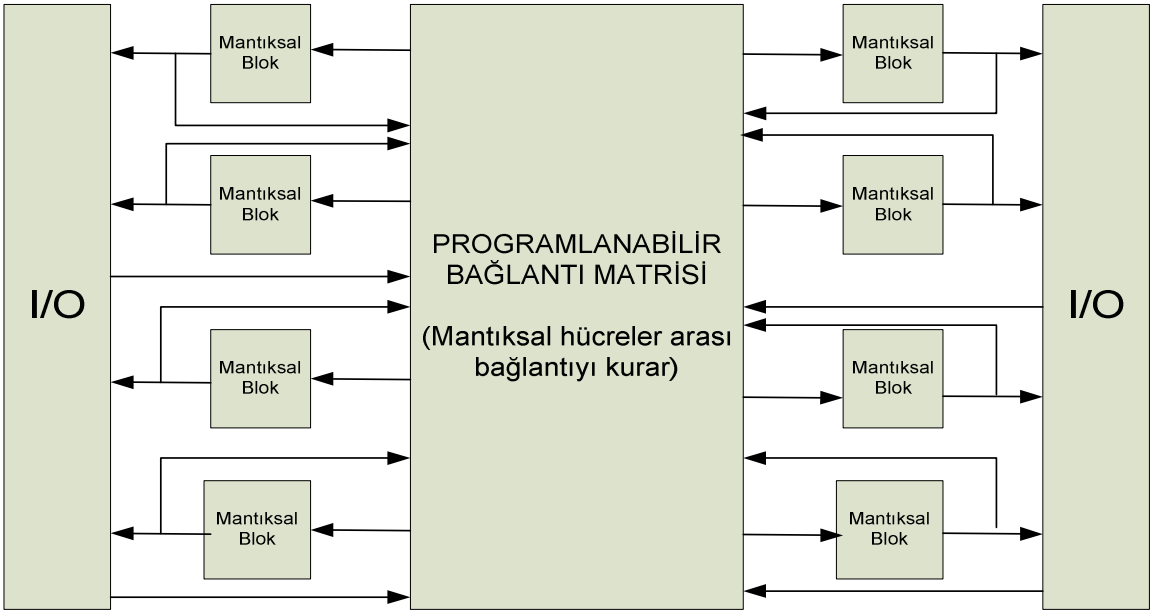


Şekil 3.1 PLD' lerin sınıflandırılması

Basit PLD' ler Programlanabilir Lojik Dizi (PLA) ve Programlanabilir Dizi Lojiği (PAL) olmak üzere ikiye ayrılırlar. PLA' lar PROM' ların hız ve sınırlı sayıda I/O sorununa bir çözüm olarak üretilmişlerdir. Bir PLA; AND ve OR matrisi olarak iki matristen oluşur. AND matrisi fonksiyonun oluşturulmuş olan çarpım terimleri için kullanılır. OR matrisi ile çarpım terimleri OR işlemine tabi tutularak fonksiyonun gerçekleştirilmesi sağlanır. Bu yapıda her iki matris de programlanabilmektedir. Bir diğer PLD yapısı ise PLA' ya benzer yapıda olan fakat OR matrisi sabit olan PAL yapısıdır. Girişler ve çıkışlara MUX, XOR gibi basit lojik devreler eklenmiştir. En önemli farkı flip floplar (FF) gibi saat darbesine ihtiyaç duyan elemanların gerçekleştirilebilmesidir. Böylece FF' ların çokça kullanıldığı durum makineleri gibi sistemlerin gerçekleştirilebilmesine büyük kolaylık sağlamışlardır. PAL' lar aynı zamanda son derece hızlı çalışabilirler. Günümüzde sıkça kullanılan PLD' ler AMD ve ALTERA şirketleri tarafından üretilmektedir. PLD' lerin en büyük dezavantajı bir fonksiyonu çarpımlar toplamı biçiminden gerçekleştirdiklerinden yüksek çarpım terimleri içeren fonksiyonları gerçekleyememeleridir. PLD' lerin performansını iyileştirmek, kullanılan silisyum alanını azaltmak ve güvenilirliği artırmak amacıyla karmaşık PLD' ler (Complex PLD: CPLD) üretilmiştir. Genel bir CPLD' nin içerisinde lojik bloklar bulunur ve bu bloklar küçük PLD yapılarında (Küçük PAL adacıkları) olup birbirleri ile programlanabilir ara bağlantı matrisi kullanarak haberleşirler. Bu şekilde silisyum alanı etkin bir biçimde kullanılmakta ve PLD'lere oranla daha fazla fonksiyon CPLD' lerle gerçeklenebilmektedir. CPLD'ler, basit PLD'ler ile günümüzde kullanılan en karmaşık

yapı olan FPGA' ler arasında geçiş döneminde yer almaktadır yani FPGA mimarisinin temelini CPLD'ler oluşturmaktadır.

CPLD' ler fonksiyon blokları, I/O blokları ve bağlantı matrisleri içerirler. Üreticinin sunduğu EPROM, EEPROM ya da Flash/EPROM gibi teknolojilere bağlı olarak programlanabilirler. Tipik bir fonksiyon bloğu şekilde görülmektedir.



Şekil 3.2 CPLD Mimarisi

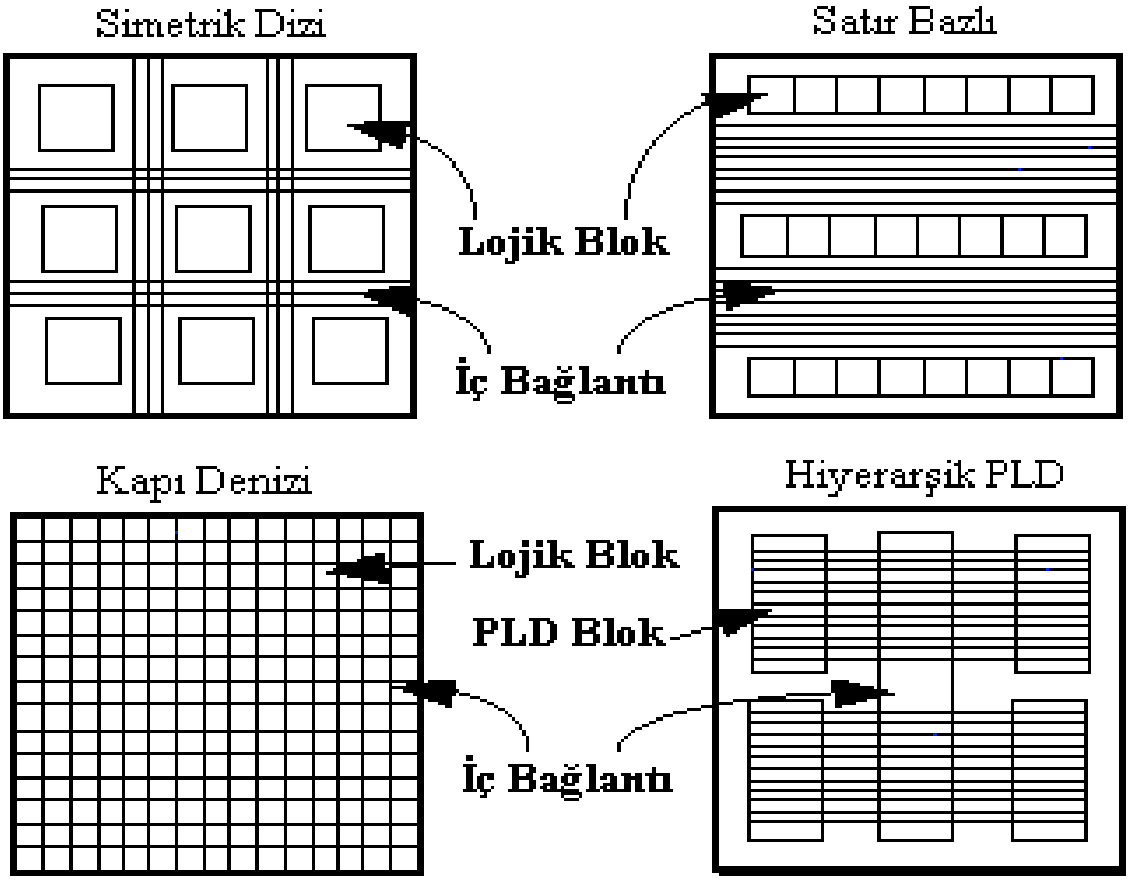
CPLD' lerden sonra 1985'te Xilinx firması sahada programlanabilir kapı dizilerini (FPGA) piyasaya sürmüştür. (Daha sonra ACTEL, ALTERA (ilk CPLD yapısının sahibi), PLESSEY, QUICKLOGIC gibi şirketler kendi FPGA yapılarını sunmuşlardır.) FPGA' ler, programlanabilir lojik bloklar ve ara bağlantılardan oluşur. Kullanıcının tasarladığı lojik devreye göre, tümdevre üreticisi tarafından sağlanan bir yazılım sayesinde lojik bloklar ve aralarındaki bağlantılar programlanır. Tasarım sırasında kullanıcıya sağladığı esneklik, düşük maliyet ve hızlı ilk üretme özelliği ile FPGA' ler sayısal tasarım ortamlarının vazgeçilmez yapıları haline gelmiştir.

3.1. FPGA Mimarisi ve Özellikleri

FPGA' ler ara bağlantıları çeşitli şekillerde gerçekleştirilen lojik bloklardan oluşmaktadır. Ara bağlantılar kullanıcı tarafından programlanabilmektedir. FPGA mimarileri, bağlantı kanallarının yapısına göre dört ana gruba ayrılır:

- Simetrik dizi (Symmetrical Array)

- Sıra tabanlı (Row-Based Array)
- Hiyerarşik PLD (Hierarchical PLD)
- Kapı denizi (Sea of Gate) mimarisi



Şekil 3.3 FPGA sınıfları

Tüm bu FPGA' lerde ara bağlantılar ve bunların nasıl programlanacağı farklıdır. Halen kullanılmakta olan dört programlama teknolojisi bulunmaktadır. Bunlar:

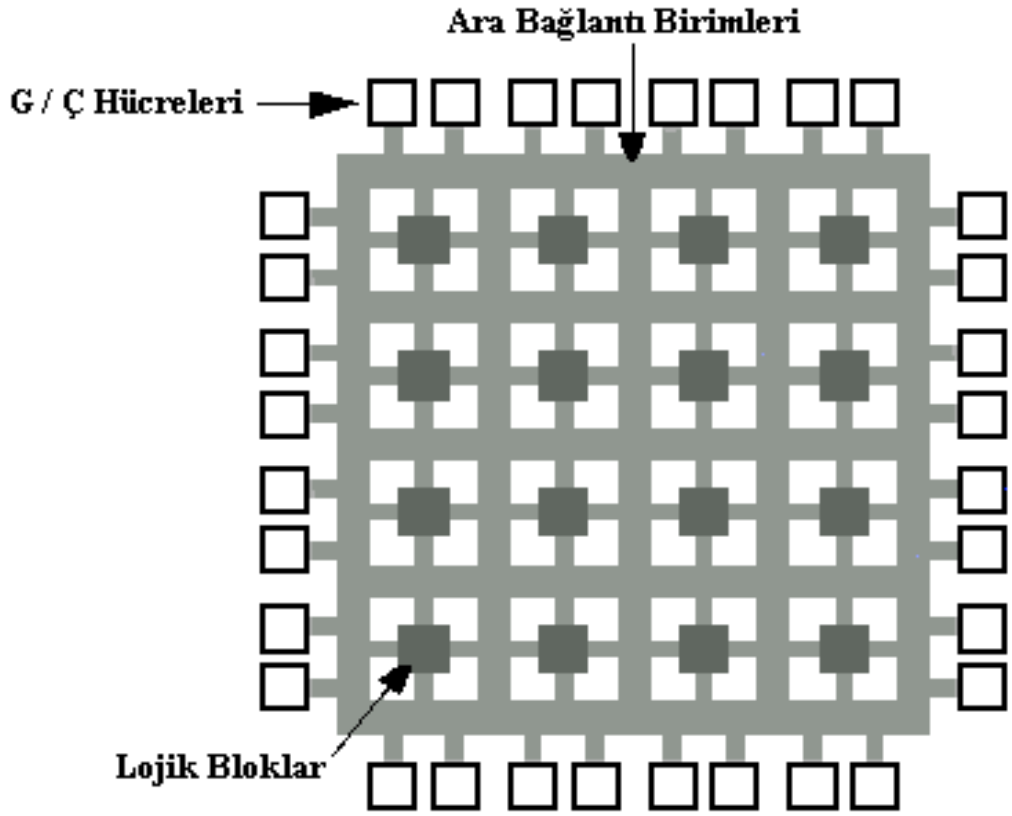
- Statik Rastgele Erişimli Bellek (SRAM) Teknolojisi
- Anti-sigorta (Anti-Fuse) Teknolojisi
- EPROM Teknolojisi
- EEPROM Teknolojisi

Uygulamaya bağlı olarak bir FPGA teknolojisi tercih edilir.

FPGA üç tane önemli düzenlenebilir elemana sahiptir :

- Düzenlenebilir Lojik Bloklar (Configurable Logic Blocks: CLB)
- I/O blokları (Input/ Output Blocks: IOB)

- Ara bağlantılar



Şekil 3.4 Genel FPGA mimarisi

CLB' ler kullanıcı lojiğini oluşturan fonksiyonel elemanlardır. Her FPGA çok sayıda bu programlanabilir lojik bloklardan oluşmaktadır. Entegredeki her lojik blok farklı bir fonksiyonu gerçekleştirmek için uygun SRAM programlama hücreleri vasıtasıyla yapılandırılabilir. IOB' ler FPGA' nın bacakları ile iç işaretler arasında ara yüz oluşturur. Programlanabilir ara bağlantı birimleri ise CLB ve IOB' lerin giriş ve çıkışlarını birleştirmek için uygun hatlar üzerinden yolları belirler. İstenilen düzenleme, lojik fonksiyonların ve ara bağlantıların nasıl gerçekleneceğini belirleyen iç statik bellek hücrelerinin programlanmasıyla sağlanır.

3.1.1. CLB yapısı

CLB' ler, çok karışık veya NAND kapısı kadar basit olabilir. CLB' lerin mimarisi iki ana gruba ayrılır;

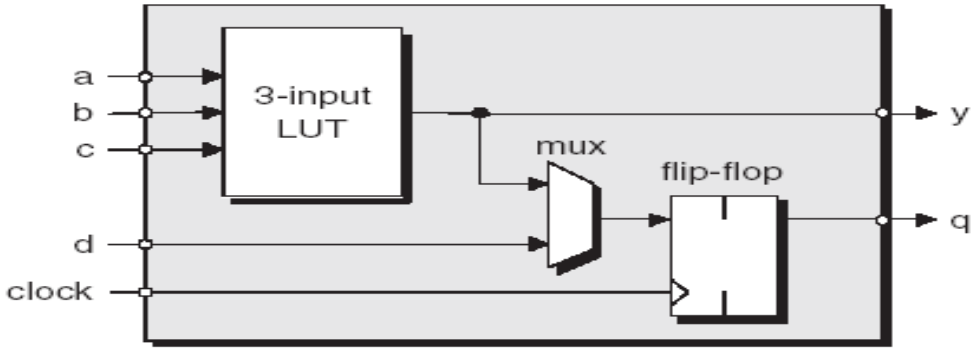
- Doğruluk tablosu (Look-Up Table: LUT) tabanlı
- Çoğullayıcı (Multiplexer: MUX) tabanlı

Bazılarında ise ardışıl devrelerin de gerçeklenebilmesi amacıyla flip flop'lar kullanılmıştır.

3.1.1.1. LUT tabanlı yapı

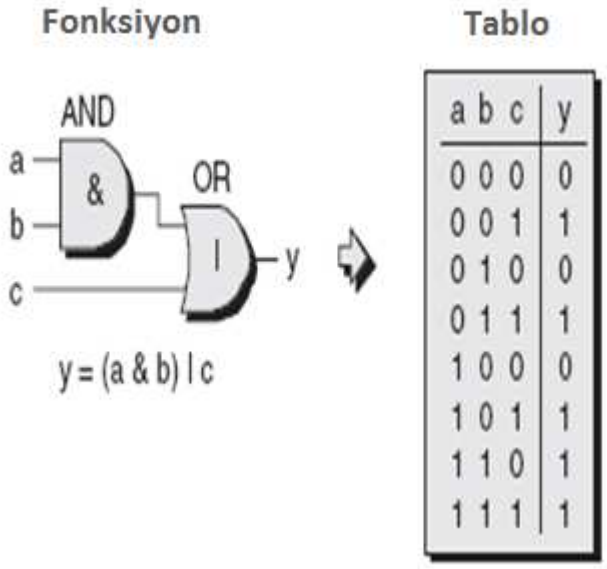
Doğruluk tablosu tabanlı yapının temel bloğu Look-Up Table (LUT) adı verilen ve m değişkenli her Boole fonksiyonunu gerçekleyebilen devredir. m, günümüzde 3 ile 6 arasında olan sabit bir sayıdır. Genelde bu blok, m tane adres ve 1 tane veri yolu olan SRAM ile gerçekleştirilir ve m-LUT olarak adlandırılır. LUT tabanlı yapıda her CLB, bir ya da daha fazla LUT ile FF gibi diğer lojik elemanlardan oluşur. İstenen devrenin gerçekleştirilebilmesi için CLB'ler ara bağlantılar ile birbirlerine bağlanarak daha karmaşık yapılar oluşturulabilir.

Aşağıda LUT tabanlı bir FPGA için lojik birim verilmiştir. FF' u besleyen MUX, LUT' dan gelen çıkışı ya da lojik bloğa ayrı bir girişi kabul etmek için yapılandırılabilir. LUT da herhangi 3 girişli lojik fonksiyonu gerçekleştirmek için yapılandırılabilir.

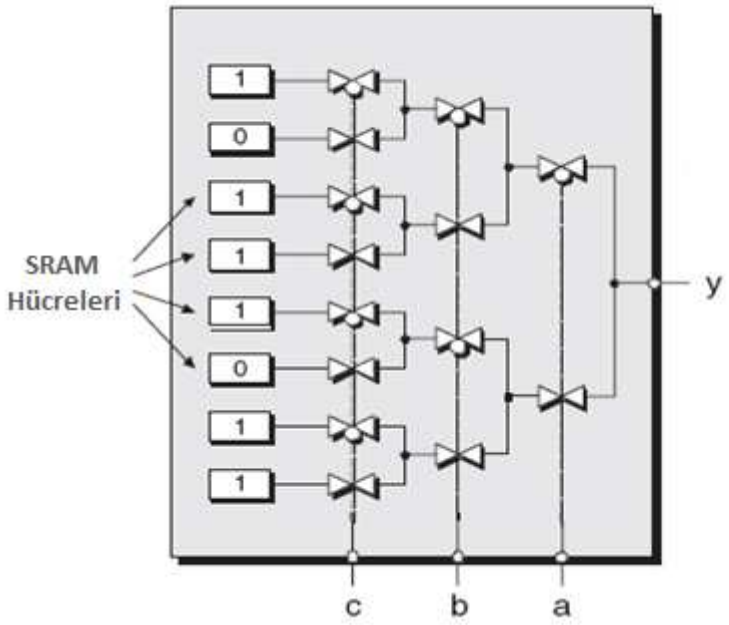


Şekil 3.5 LUT tabanlı bir FPGA' in birim elemanı [12]

Bir grup giriş sinyali LUT' a bir indeks olarak kullanılır. Bu tablonun içeriği, öyle ayarlanır ki her giriş kombinasyonu tarafından gösterilen hücre istenilen değeri içerir. Örneğin y=(a. b) + c' fonksiyonu 3 girişli LUT' a uygun değerleri yükleyerek yapılabilir. Bu örnek için LUT' un SRAM hücrelerinden (Anti-Fuse, EEPROM ya da FLASH hücreleri de olabilir, ileride bahsedilecektir) yapıldığını varsayarsak, genelde kullanılan teknik, şekilde görülen transmisyon kapıları kaskatını kullanarak istenilen SRAM hücresini seçmek için girişleri kullanmaktır. Aslında SRAM hücreleri de konfigürasyon amacı ile birbirine bağlıdır. Ancak, bu bağlantılar şekilden daha basit bir gösterim olması için atılmıştır.



Şekil 3.6-1 LUT' un yapılandırılması [12]



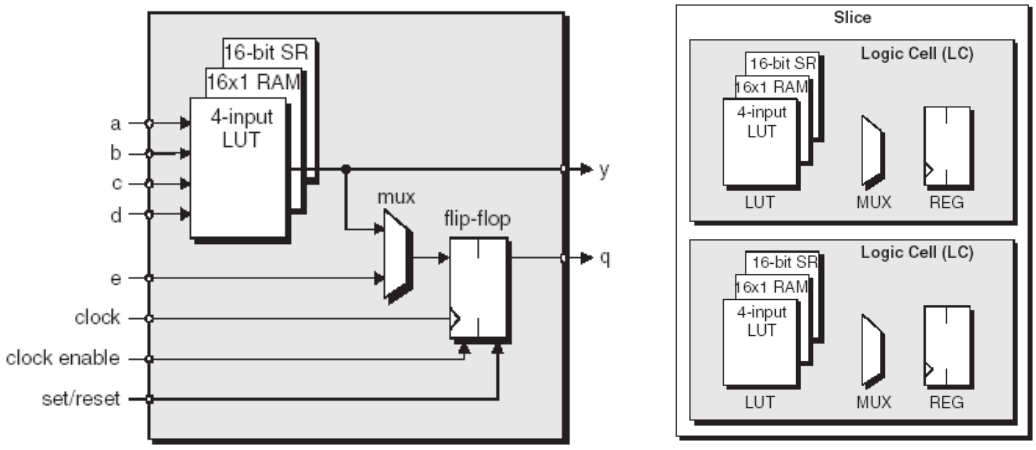
Şekil 3.6-2 LUT' un yapılandırılması [12]

Eğer transmisyon kapısı aktif (enable) yapılırsa, girişteki sinyali çıkışa aktarır. Eğer kapı pasif yapılırsa (disabled), çıkışı sürdüğü hattan elektriksel olarak ayrılır. Transmisyon kapısının sembolü küçük bir daire ile gösterilir. Bu daire, bu kapıların kontrol girişlerinde lojik

0 ile aktif edildiği anlamına gelir. Bu dairelerin olmadığı semboller de bu kapıların lojik 1 ile aktif edildiğini gösterir. Bu temelde, çeşitli SRAM hücrelerinin içeriğinin seçilmesi için nasıl farklı kombinasyonların kullanılabileceğini görmek kolaydır.

"n" girişli LUT ile ilgili en önemli özellik herhangi olası n girişli kombinasyonel lojiği gerçekleyebilmesidir. Daha fazla giriş eklemek daha karmaşık fonksiyonların gösterilmesini sağlar. Ancak, her bir giriş eklediğinizde, SRAM hücrelerinin sayısını iki katına çıkarırsınız. İlk FPGA' ler 3 girişli LUT tabanlıydı. Şu anki ortak görüş 4 girişli LUT'ların uygun değer olduğu yönündedir, fakat yüksek kapasiteli FPGA' lerde 6 girişli LUT'lara rastlamak mümkündür. Geçmişte, bazı entegreler 3 ve 4 girişli LUT'ların karışımı gibi farklı boyutlardaki LUT'ların karışımını kullanarak yapılmışlardı. Çünkü, bu en uygun entegre kullanımını vaat etmişti. Ancak, tasarım mühendislerinin elinde olan ana araçlardan biri lojik sentezidir. Benzerlik ve düzenlilik bir sentez aracının en çok sevdiği şeydir. Bu nedenle, şu anda gerçekten başarılı yapıların tamamı sadece 4 girişli LUT' ların kullanımı temelindedir. Bu, gelecekte tasarım yazılımları karmaşıklaşmaya devam ettikçe karışık boyutlu LUT yapılarının tekrar ortaya çıkmayacağı anlamına gelmemektedir.

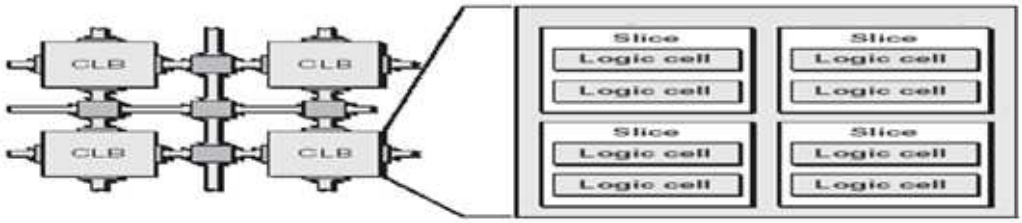
Xilinx'in FPGA' inin temel yapı taşına lojik hücre (lojik Cell: LC) denir. Bir LC, 4 girişli LUT (16x1 RAM ya da 16 bitlik Shift Register gibi davranabilen), bir MUX ve bir Register' dan oluşmaktadır. Şekildeki gösterim kabaca basitleştirilmiş bir gösterimdir. Register şekildeki gibi FF ya da Latch olarak yapılandırılabilir. Saat işaretinin polaritesi (çıkan kenar tetiklemeli ya da düşen kenar tetiklemeli mi) ya da saat enable ve set/ reset sinyalleri (aktif High ya da aktif Low) yapılandırılabilir. LUT, MUX ve Register' a ek olarak LC, aynı zamanda aritmetik operasyonlarda kullanmak için bazı özel hızlı elde lojiği gibi diğer elemanları da içerir.



Şekil 3.7 LC Yapısı ve Dilim (Slice) Yapısı [12]

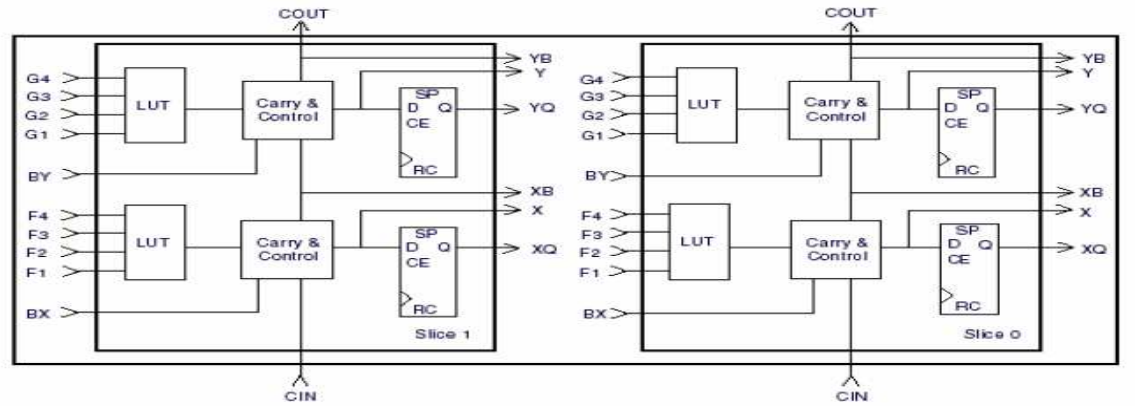
Hiyerarşide bir düzey yukarı çıkarsak, iki LC birbirine bağlanarak bir Slice adını alır. Xilinx' in konfigüre edilebilir lojik blok (Configurable Logic Block: CLB) olarak isimlendirdiği yapı Altera' da lojik dizi bloğudur (Logic Array Block, LAB). Örneğin CLB' leri ele alırsak, bazı Xilinx FPGA'lerinin her CLB' sinde iki Slice bulunurken, diğerlerinde dört Slice bulunur. Günümüzde bir CLB programlanabilir ara bağlantılar denizindeki programlanabilir lojik adası olarak görülmektedir. Aynı zamanda, CLB' nin içerisinde bazı hızlı programlanabilir ara bağlantılar vardır. Bu ara-bağlantı komşu Slice' ları birbirine bağlamak için kullanılmıştır.

LC→Slice (iki LC'li)→CLB (iki veya dört Slice' lı) şeklindeki lojik blok hiyerarşisi ara-bağlantılardaki eşdeğer hiyerarşi tarafından tamamlanmaktadır. Bir Slice' daki LC' ler arasında hızlı ara-bağlantı vardır. CLB 'lerdeki Slice' lar arasında biraz daha yavaş ara-bağlantı ve bunu takip eden CLB' ler arasında ise ondan biraz daha yavaş ara-bağlantı vardır. Bu fikrin amacı, ara-bağlantılardan kaynaklanan aşırı gecikmelerin önlenerek her şeyin birbirine kolayca bağlanabilmesini sağlamaktır.



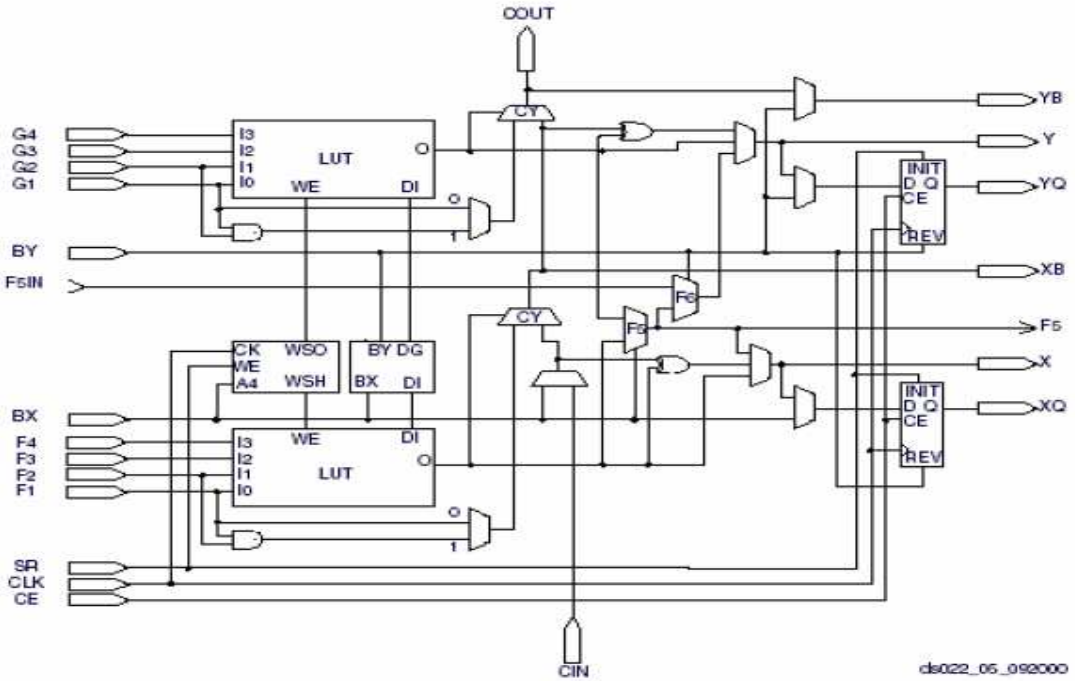
Şekil 3.8 Örnek bir CLB yapısı [12]

Bir Xilinx/Virtex CLB' si, Sekil 8'de gösterildiği gibi, dört adet LC ve gerekli bağlantı devresinden oluşur. İki LC ve gerekli bağlantı devresinden oluşan yapıya ise Slice adı verildiğini daha önceden söylemiştik. FPGA üzerinde gerçeklenen devrelerin kapladığı alan genellikle Slice sayısı cinsinden ifade edilir. Her lojik hücreye ait olan LUT' un, MUX' un ve Register' ın kendi veri giriş ve çıkışları olmasına karşın Slice' ın lojik hücrelerde de yaygın olan bir saat, saat enable ve set/reset i vardır.



Şekil 3.9 Xilinx-VirtexE CLB' si [12]

Bir LUT, 4 değişkenli herhangi bir fonksiyonu gerçekleyebilir. Bu gerçekleme tablo seklinde olduğu için, dört değişkenli ya da daha az değişkene sahip tüm fonksiyonlar FPGA üzerinde aynı karmaşıklığa, dolayısıyla aynı kapı gecikmesine sahiptir. 4 den daha fazla değişkene sahip fonksiyonlar ise, örnekte gösterileceği gibi, dört değişkenli bloklara parçalandıktan sonra LUT' lar üzerinde gerçeklenir.



Şekil 3.10 Xilinx/Virtex Ailesine ait bir Slice' ın iç yapısı [12]

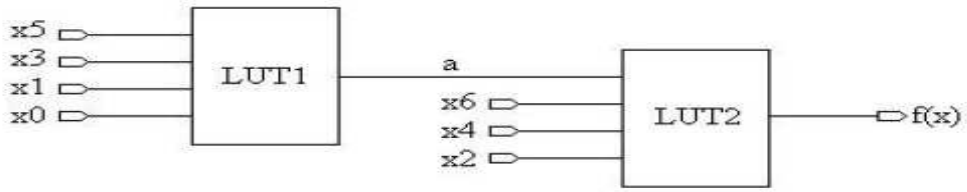
Örnek : $y = x_3 \bullet x_2 \bullet x_1' \bullet x_0$ fonksiyonu tek bir LUT ile gerçeklenirken

$y = (x_5' \cdot x_4 \cdot x_3 \cdot x_2 \cdot x_1' \cdot x_0) + (x_6' \cdot x_5' \cdot x_3 \cdot x_1' \cdot x_0)$ fonksiyonu aşağıdaki şekilde parçalanarak 2 LUT ile gerçeklenebilir.

$y = (x_5' \cdot x_4 \cdot x_3 \cdot x_2 \cdot x_1' \cdot x_0) + (x_6' \cdot x_5' \cdot x_3 \cdot x_1' \cdot x_0) = x_4 \cdot x_2 \cdot (x_5' \cdot x_3 \cdot x_1' \cdot x_0) + x_6' \cdot (x_5' \cdot x_3 \cdot x_1' \cdot x_0)$

$a = (x_5' \cdot x_3 \cdot x_1' \cdot x_0)$ alınırsa; (LUT 1 de gerçeklenir)

$= (x_4 \cdot x_2 \cdot a) + (x_6' \cdot a) \Rightarrow a \cdot (x_6' \cdot x_4 \cdot x_2)$ (LUT 2 de gerçeklenir)

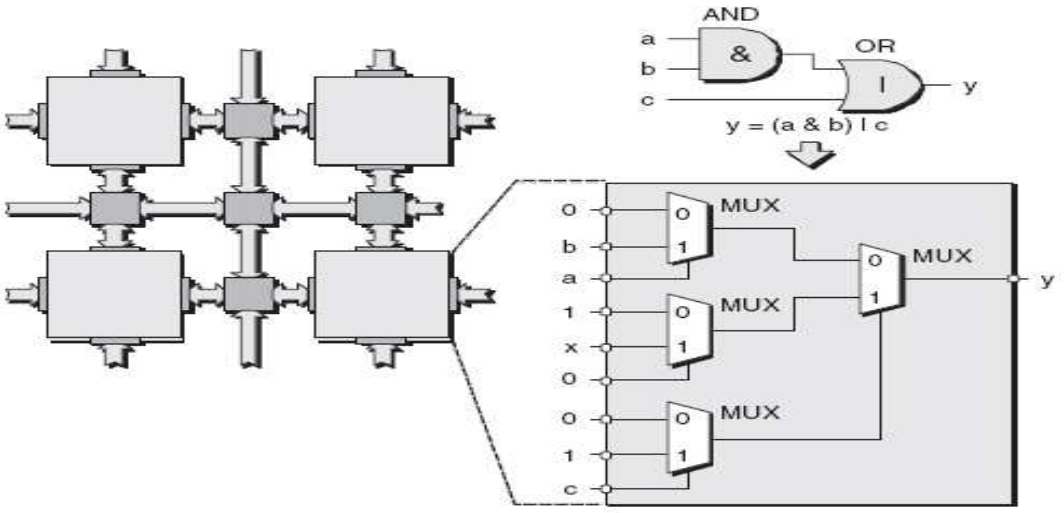


Şekil 3.11 $y = (x_5' \cdot x_4 \cdot x_3 \cdot x_2 \cdot x_1' \cdot x_0) + (x_6' \cdot x_5' \cdot x_3 \cdot x_1' \cdot x_0)$ fonksiyonu

FPGA mimarisinde kullanılan LUT' lar, Boole fonksiyonlarının gerçeklenmesinin yanı sıra, 16x1 bitlik bir RAM olarak da kullanılabilmektedir. Aynı Slice içerisindeki iki LUT birlikte kullanılarak 32x1 bitlik RAM ya da 16x1 bitlik iki portlu RAM gerçeklenebilir. Bunların yanı sıra, bir LUT, 16-bitlik silme (reset) özelliği olmayan Shift Register olarak da kullanılabilir. Slice içerisindeki F5 MUX' u, iki LC' in çıkısını birleştirerek bir Slice içerisinde; 5-değişkenli herhangi bir Boole fonksiyonun, 4 girişli bir MUX' un ya da 9-değişkenliye kadar bazı fonksiyonların gerçeklenmesini sağlar. Benzer şekilde F6 çoklayıcısı, aynı CLB içerisindeki iki Slice' ın F5 çoklayıcısı çıkışlarını birleştirerek bir CLB içerisinde; 6-değişkenli herhangi bir Boole fonksiyonunun, 8 girişli bir çoklayıcının ya da 19-değişkenliye kadar bazı fonksiyonların gerçeklenmesini sağlar.

3.1.1.2. MUX tabanlı yapı

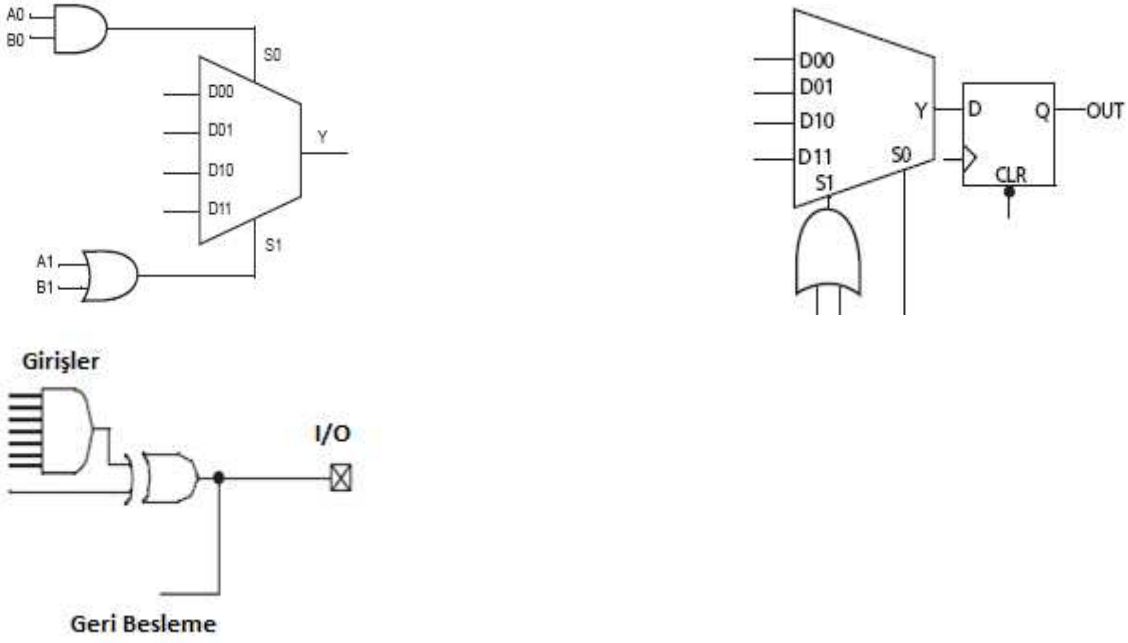
Çoğullayıcı tabanlı yapıda ise CLB' ler MUX' ların çeşitli düzenlemelerinden ve olabildiğince az AND ve OR gibi lojik kapılardan oluşur. Ardışıl devrelerin gerçeklenebilmesi amacıyla MUX tabanlı FPGA' lerin içinde tutucu (Latch) ve flip flop gibi elemanlar kullanılabilmektedir.



Şekil 3.12 MUX tabanlı CLB yapısı

Bir zamanlar, MUX tabanlı yapılarla en iyi sonucun alındığı söylenirdi. Ancak bu sonuçların nasıl daha iyi olduğu açıklanmazdı. Aynı zamanda, MUX tabanlı yapıların kontrol lojiğini gerçeklemede (eğer bu giriş "true" ve bu giriş "false" ise şu çıkışı "true" yap gibi…) avantajı olduğu da söylenir. Buna karşın, bu yapıların bazıları yüksek hızlı elde lojik zincirlerini sağlamazlar.

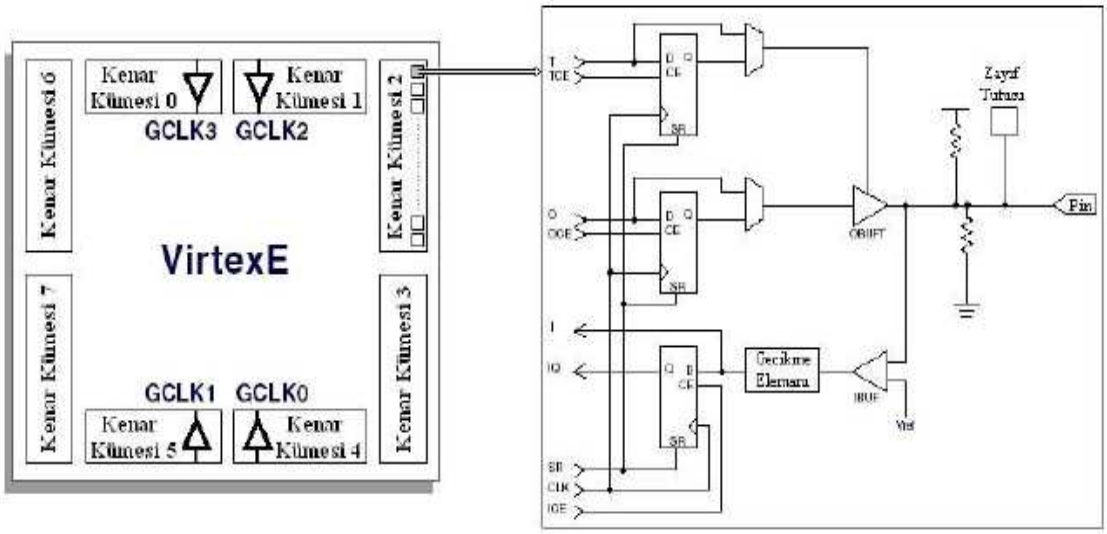
MUX tabanlı yapılara örnek olarak ACTEL firmasının ürettiği FPGA' ler verilebilir. Bu FPGA' ler de LUT tabanlı yapılar gibi yatay olarak dizilmiş lojik modüllerden (Logic Module: LM) ve bu modül dizilerinin arasında bulunan bağlantı yollarından oluşmaktadırlar. ACTEL firmasının 42MX ailesine ait bir FPGA' i incelenirse bu yapı 3 farklı tipte lojik modül içermektedir; bunlar kombinezonsal modül (C-module), ardışıl modül (S- module) ve kod çözücü modül (D module). Kombinezonsal modül ile Boole fonksiyonları, ardışıl modül ile bir ardışıl devre gerçeklenebilir. Kod çözücü modül ile 7 girişli kod çözme işlemi gerçekleştirilebilir. Ayrıca ACTEL FPGA' leri içerisinde SRAM modülleri de dağıtık bir şekilde bulunmaktadır.



Şekil 3.13 C tipi, S tipi ve D tipi modüller [15]

3.1.2. IOB yapısı

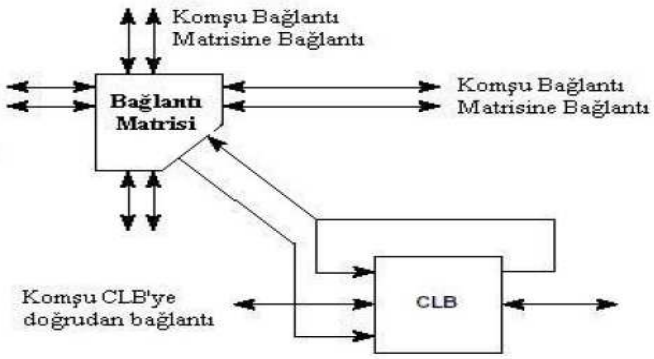
Giriş/Çıkış blokları, kılıf bacaklarıyla tasarım için kullanılan birimler (CLB, Blok RAM) arasında bağlantı kurar. FPGA' lerin giriş çıkış blokları; giriş, çıkış veya giriş-çıkış olarak kurgulanabilir. Sekiz "Kenar Kümesi (Bank)" halinde yerleştirilen giriş çıkış blokları, kümeler halinde, farklı işaretleşme standartlarını destekleyecek şekilde kurgulanabilir. Şekilde Virtex-E yongasının giriş çıkış bloklarının yerleşimi ve bir giriş çıkış bloğunun ayrıntılı seması gösterilmiştir.



Şekil 3.14 Virtex Giriş/Çıkış Bloğu ve Kenar Kümelerinin yerleşimi [12]

3.1.3. CLB' ler arası bağlantılar

Giriş/ Çıkış bloklarıyla tasarım blokları ve tasarım bloklarıyla yine tasarım blokları arasındaki bağlantılar, bağlantı elemanları ile sağlanır. FPGA içerisindeki bağlantı elemanları, CLB' ler arasına satırlar ve sütunlar halinde yerleştirilmiş bağlantı hatları ve bu hatların kesişim noktalarına yerleştirilmiş bağlantı matrislerinden oluşur. Şekilde bir FPGA yongası içerisindeki bağlantı elemanları gösterilmiştir.



Şekil 3.15 FPGA Bağlantı Elemanı [12]

3.2. FPGA' lerin Programlama Teknolojileri

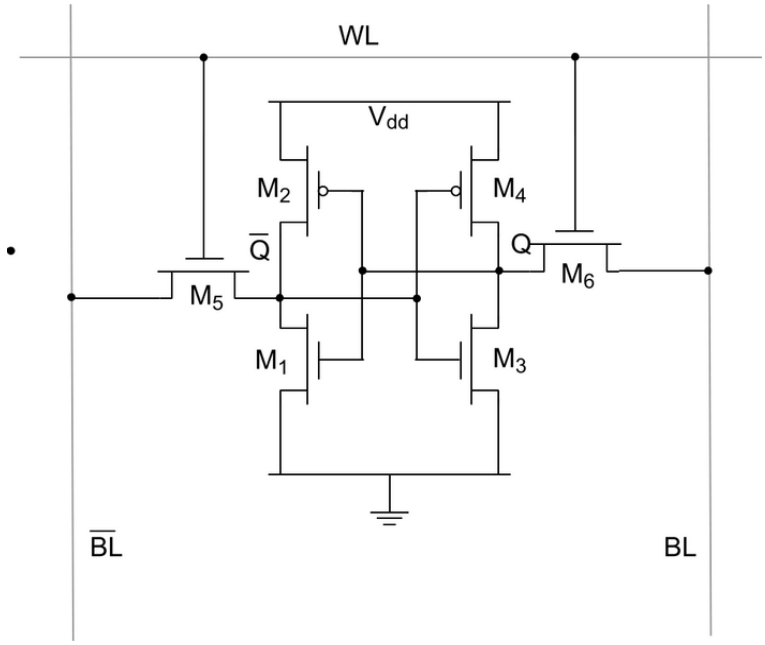
Günümüzde FPGA' lerin üretim safhasında kullanılan başlıca iki çeşit programlama teknolojisi bulunmaktadır: SRAM ve Anti-Fuse. Her iki teknolojinin birbirlerine göre avantaj ve

dezavantajları bulunmaktadır. Bunlar dışında Altera firması tarafından MAX ailesi FPGA' lerde kullanılan EEPROM teknolojisi bulunmaktadır. Bu tip FPGA' lerin da tekrar programlanabilmeleri için devreden sökülerek ultraviyole ışınlarına tutulması gerekmektedir.

Günümüzde en çok kullanılan programlama teknolojileri SRAM ve anti-sigorta programlamadır.

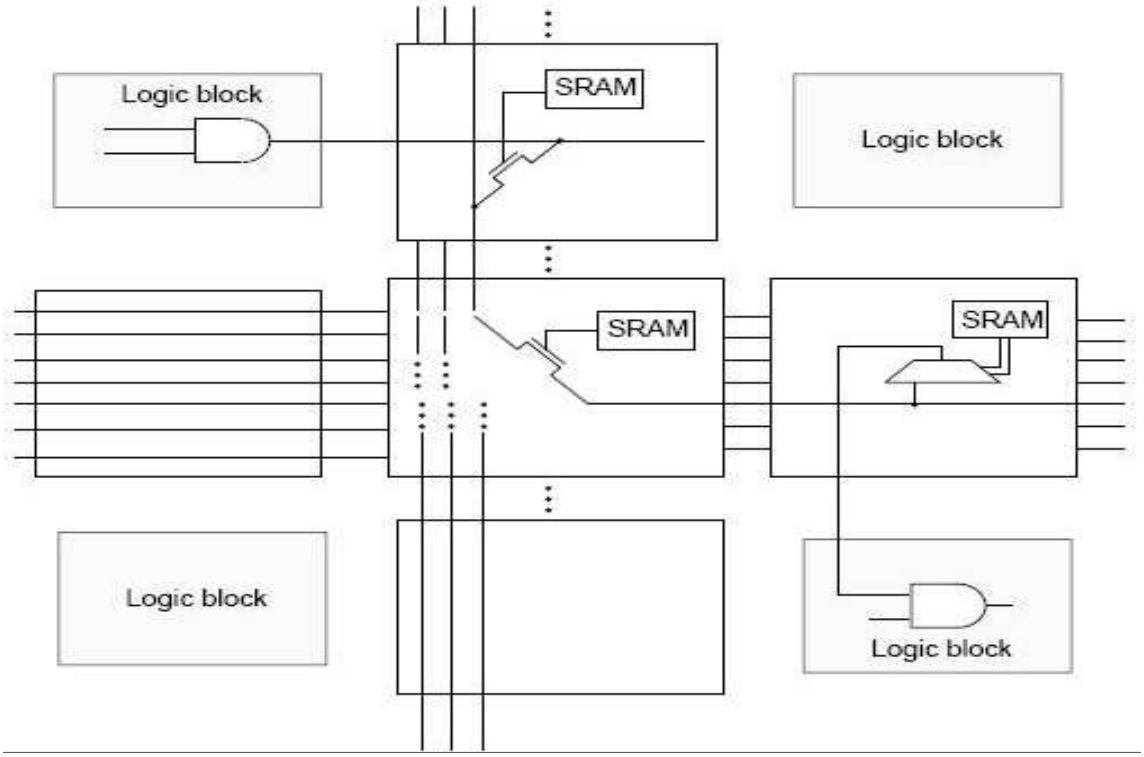
3.2.1. SRAM teknolojisi

Bir SRAM hücresi 6 adet transistörden oluşmaktadır. Sırt sırta pozitif geri beslemeli bağlanmış iki evirici ve bunların çıkışında yer alan iki adet geçiş transistöründen oluşur. Bu eviriciler veri tutucu olarak iş yaparken geçiş transistörleri ise seçme işlemi için kullanılırlar.



Şekil 3.16 SRAM yapısı [12]

SRAM hücreleri tarafından kontrol edilen geçiş transistörleri, iletim kapıları ve MUX' lar yapılabilmektedir. Geçiş transistörleri kullanan uygulamalarda RAM hücresi transistörün iletimde ya da kesimde olmasını sağlar. MUX' lu yapılarda ise RAM hücresi MUX' un girişinin hangi çıkışa bağlanacağını kontrol eder.



Şekil 3.17 SRAM hücrelerinin kullanımı [12]

FPGA' lerin çoğu SRAM konfigürasyon hücresi tabanlıdır. Bu da tekrar tekrar konfigüre edilebilir anlamına gelir.

Bu teknolojinin üstünlüğü, devrenin tekrar düzenlenmesinin hızlı olmasıdır. Programlama işlemi, yalnızca SRAM üzerinde saklanan verilerin değiştirilmesi ile mümkün olduğundan tasarımlar tamamlandıktan hemen sonra FPGA' leri programlamak ve test etmek mümkündür. Böylece, tasarım sırasında oluşacak yeni ihtiyaçları ve standart değişikliklerini karşılamak, hataları tespit etmek ve düzeltmek, fabrikasyon zamanı ve fazla maliyet getirmeden mümkün olmaktadır. Ayrıca, sisteme ilk enerji verildiğinde, FPGA başlangıçta kendi kendine test ya da kart/sistem testini gerçekleştirmek için programlanabilir. Ek olarak, SRAM hücreleri tamamen CMOS teknolojisi kullanılarak üretilirler, böylece, bu malzemeleri üretmek için özel işlemlere gerek kalmaz

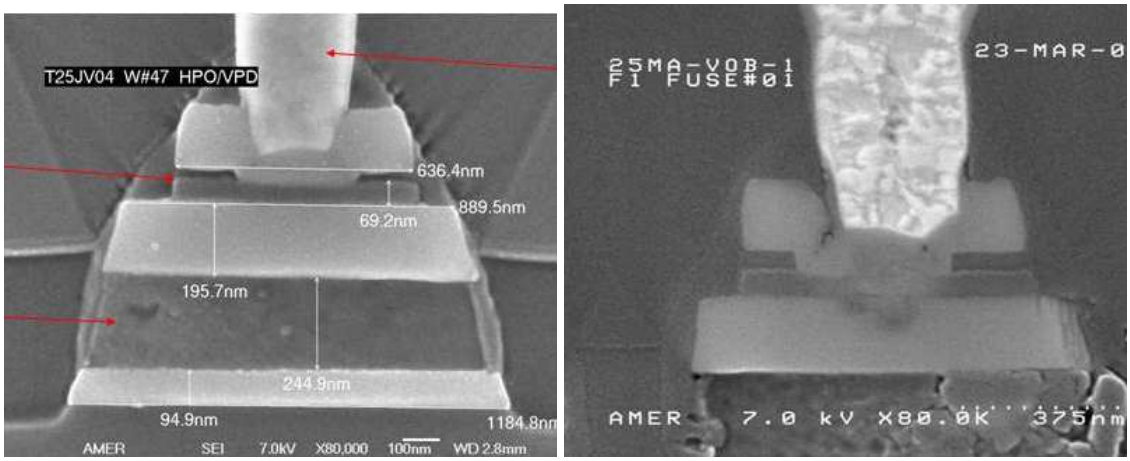
En önemli zayıf yanı ise RAM hücrelerinin gerektirdiği tümdevre alanıdır. Ayrıca uçucudurlar, yani devrenin gücü kesildiği anda programlanmış hücrelerdeki veriler kaybolur. Devreye her güç uygulandığında SRAM hücrelerinin tekrar programlanması gerekmektedir. Bu, ya özel harici bir hafıza entegresi (bu ekstra maliyet demektir) ya da bir mikroişlemci gerektirir.

SRAM-tabanlı entegrelerdeki diğer bir unsur tasarımlardaki IP (Intellectual Property)' nin korunmasının zor olmasıdır. Çünkü, entegreyi programlamak için kullanılan konfigürasyon dosyasının harici bir hafızada saklanır. Şu an, konfigürasyon dosyasının içeriğini okuyacak ve ilgili şematik ya da netlist gösterimini üretecek uygun ticari bir araç bulunmamaktadır. Dünyada, "tasarım IP"'sinin ele geçirilmesi konusunda uzmanlaşan art niyetli mühendislik şirketleri de bulunmaktadır. Aynı zamanda, paranın içerde kalması için hükümetlerinin IP hırsızlığına göz yumduğu ülkeler de var.

Hazır şekilde bulunan teknolojiyi kullanarak bir devre bordu alıp bir tester' a koyup hızlı bir şekilde bütün netlisti açmak oldukça kolaydır. Bu netlist daha sonra kartı çoğaltmak için kullanılabilir. Şu aşamada kalan tek iş boot PROM (ya da EPROM, EEPROM vb.)'undan FPGA konfigürasyon dosyanızı kopyalamaktır ve böylece, bütün tasarımın bir kopyasına sahip olmuş olurlar. İşin iyi bir yanı, bugünün bazı SRAM-tabanlı FPGA' lerin "bitstream encryption" (bit akışı şifrelemesi)' ni desteklemeleridir. Bu durumda, son konfigurasyon datası harici hafızaya yüklenmeden önce şifrelenir. Bazı ilgili lojikle birlikte, bu anahtar gelen şifrelenmiş konfigürasyon bit akışının şifresinin yükleme sırasında çözülmesini sağlar. Şifrelenmiş bit akışını yükleme FPGA' in geri okuma yeteneğini otomatik olarak kaldırır. Bu, geliştirme sırasında şifrelenmemiş konfigürasyon datasının kullanacağınız ve daha sonra, üretim aşamasında geçtiğinizde şifrelenmiş datayı kullanacağınız anlamına geliyor. Bu tasarının asıl dezavantajı, sistemden enerji kesildiğinde FPGA' in şifre anahtarı kaydedicisindeki içeriğin korunması için devre kartı üzerinde bir yedekleme bataryasının gerekli olmasıdır. Bu bataryanın ömrü yıllarca ya da on yıllarca sürecektir, çünkü entegredeki sadece bir kaydediciyi besleyecektir, ancak, boyutu, ağırlığı, karmaşıklığı ve kartın fiyatını artıracaktır.

3.2.2. Anti-Sigorta (Antifuse) teknolojisi

Bir anti-sigorta yüksek empedans konumunda iken (100M-Ohm ile 1G-Ohm (Via teknolojisi ile) arası değerler) programlama gerilimi uygulanarak düşük empedans (sigortalanmış) (100-Ohm lar mertebesi) durumlarına programlanabilir. RAM teknolojisinden daha düşük maliyeti olmasına karşılık bir kere programlanabilen tümdevrelerdir. Dolayısıyla ilk örnek üretim için pahalı bir çözüm olmaktadırlar.



Şekil 3.18 Anti-Fuse teknolojisiyle programlama [15]

Programlanırken sistemde bulunan SRAM-tabanlı entegrelerin aksine, Anti-Fuse tabanlı entegreler özel bir programlayıcı kullanarak offline olarak programlanır. Anti-Fuse tabanlı FPGA'lerin çeşitli avantajları vardır. Bu entegreler nonvolatile' dır (yani enerji kesildiğinde bile konfigürasyon datası kalır). Bu da sisteme enerji verilir verilmez çalışmaya hazır halde oldukları anlamına gelir. Bunu yanında, bu entegreler konfigürasyon datasını saklamak için harici bir hafızaya ihtiyaç duymazlar. Bu da ek malzeme ve kart üzerindeki yerden tasarruf edilmesini sağlar.

Bu entegrelerin ara-bağlantı yapıları "rad hard" (Radiation Hardening)' dır. (Rad hard yöntemleri: fiziksel (kaplama, özel wafer maddesi (SiO, Safir),DRAM), lojik (ECC, 3-redundant element, watchdog)) Bu, bu entegrelerin radyasyonun etkilerine karşı oldukça bağışıklığı olduğu anlamına gelir. Bu, özellikle askeri ve uzay uygulamalarını ilgilendiren bir özelliktir. Çünkü, SRAM-tabanlı malzemelerdeki konfigürasyon hücresinin durumu eğer o hücre radyasyonla karşılaşırsa (ki uzayda çok fazla radyasyon mevcuttur) bozulur. Anti-Fuse bir kez programlandığında, bu şekilde alt üst olmaz.

Bu entegrelerdeki flip-flopların radyasyona karşı hassas olduğuna dikkat etmek gerekir. Bu durumda, radyasyonu yoğun olan ortamlarda yongaların flip-floplarının "triple redundancy design" tarafından korunması gerekir. Bu şu anlama gelir: her kaydedicinin üç kopyası yapılır ve herhangi bir aksaklık durumunda bu üç kaydedicinin içeriklerinin çoğunluğuna bakarak karar verilir. Örneğin iki kaydedici "0" ve bir kaydedici "1" içeriyorsa, içeriğin "0" olduğuna karar verilir ya da tam tersi yapılır.

Ancak, Anti-Fuse tabanlı FPGA'lerin en önemli avantajı belki de konfigürasyon datasının içlerinde derine gömülmesidir. Programlayıcının bu datayı okuması olasıdır, çünkü bu,

aslında programlayıcı bu şekilde çalışır. Her Anti-Fuse işletilirken, programlayıcı o elemanın tamamen programlanıp programlanmadığını belirlemek için test etmeye devam eder. Daha sonra diğer anti-Fuse' e geçer. Dahası, programlayıcı konfigürasyonun başarılı bir şekilde çalışıp çalışmadığını otomatik olarak doğrulamak için kullanılabilir (50 milyon programlanabilir elementten bahsedildiğinde bunu yapmaya değer). Bunu yapmak için, programlayıcının anti-Fuse' lerin gerçek durumlarını okuma ve konfigürasyon dosyasında tanımlanan durumlarla karşılaştırabilmesi gerekir. Buna rağmen, entegre bir kez programlandığında, programlama datasının daha sonra okunmasını önlemek amacıyla özel bir güvenlik anti-Fuse 'u koyulabilir. Üstleri kaldırılsa bile programlanmış ve programlanmamış anti-Fuse' ler aynı görünür. Bütün anti-Fuse' lerin iç metal tabakalara gömülmesinden dolayı art niyetli mühendislerin tasarımı ele geçirmeleri imkansız hale gelir. Anti-Fuse teknolojisi asıl üretim işlemi tamamlandıktan sonra yaklaşık üç ek proses adımının kullanımını gerektirir. Anti-Fuse tabanlı entegrelerin asıl dezavantajı OTP (bir kez programlanabilir) olmalarıdır. Bu da bu entegrelerin geliştirme ve ilk örnek hazırlamada tercih edilmemelerine neden olur.

3.2.3. EEPROM/FLASH teknolojisi

EEPROM ya da FLASH-tabanlı FPGA' ler SRAM-tabanlı FPGA' lere benzerdir. Konfigürasyon hücreleri uzun bir Shift Register zinciri şeklinde bağlanmıştır. Bu entegreler programlayıcı kullanılarak offline olarak konfigüre edilebilir. Alternatif olarak, bazı sürümleri in-system (sistem içi) programlanabilir (ISP)' dir, ancak, programlama süreleri SRAM-tabanlı entegrelere göre üç kat daha uzundur. Bir kez programlandığında içerdikleri data kalıcıdır.

3.2.4. Hybrid FLASH-SRAM teknolojisi

FPGA' lerde, bazı üreticiler programlama teknolojilerinin gizemli birleşimlerini önerirler. Örneğin, her konfigürasyon elemanının FLASH (ya da EEPROM) hücresi ile SRAM hücresinin birleşiminden oluşturulduğu bir entegreyi ele alın. Bu durumda, FLASH elemanları tekrar programlanabilir. Bu durumda, sisteme enerji verildiğinde, FLASH hücrelerinin içeriği ilgili SRAM hücrelerine paralel şekilde kopyalanır. Bu teknik, antifuse entegreler gibi kalıcılık kazandırır. Bu da enerjinin sisteme verilmesiyle entegrenin hemen hazır durumda olması demektir. Ancak antifuse-tabanlı entegreden farklı olarak, daha sonra entegreyi tekrar konfigüre etmek için SRAM hücreleri kullanılabilir. Alternatif olarak, entegre FLASH hücrelerini kullanarak ya sistemdeyken ya da offline olarak programlayıcıyla tekrar konfigüre edilebilir.

3.2.5. SRAM ve Anti-Sigorta teknolojilerinin karşılaştırılması

Anti-sigorta teknolojisinde programlama bağlantılarının dirençleri 100Ω mertebelerinde iken SRAM teknolojisinde programlama bağlantıları 1kΩ mertebelerindedir. Dolayısıyla, anti-

sigorta FPGA' ler daha yüksek performans göstermektedirler. Ayrıca, anti-sigorta FPGA' lerin lojik hücre yapıları SRAM teknolojisine göre daha az silisyum alanı gerektirir. Ancak, anti-sigorta FPGA' ler sadece bir kez programlanabilmektedirler. SRAM FPGA' ler ise bir EPROM veya dış ortama bağlantıyı sağlayabilen bir konektör yardımıyla sistem içerisinde programlanabilmektedirler. Ürünün çalıştığı ortamda güncellenebilmesi SRAM FPGA' lerin seçilmesindeki en önemli etkendir. Tablo 1'de ticari amaçlı kullanılan bazı FPGA' ler ve Tablo 2'de FPGA programlama teknolojilerine ait karşılaştırmalar verilmiştir.

Çizelge 3.1 Bazı ticari FPGA' ler

Üretici	**Mimari**	**Lojik Blok Tipi**	**Programlama Teknolojisi**
Actel	Satır Bazlı	Çoğullayıcı Bazlı	Anti-Sigorta
Altera	Hiyerarşik PLD	Doğruluk Tablosu	Statik RAM
QuickLogic	Simetrik Dizi	Çoğullayıcı Bazlı	Anti-Sigorta
Xilinx	Simetrik Dizi	Doğruluk Tablosu	Statik RAM

Çizelge 3.2 FPGA' lerin programlanma teknolojileri

Programlama Teknolojisi	**Uçuculuk**	**Tekrar Programlanabilme**	**Silisyum Alanı**	**R (Ω)**	**C (ff)**
Statik RAM	Evet	Devre Üzerinde	Büyük	1-2 K	10-20
PLICE Anti-Sigorta	Hayır	Yok	Küçük	300-500	3-5
EPROM	Hayır	Devre Dışında	Küçük	2-4 K	10-20
EEPROM	Hayır	Devre Üzerinde	2*EPROM	2-4 K	10-20

3.3. FPGA İçerisinde Yer Alan Özel Ara yüzler ve Bloklar

3.3.1. Dağıtık RAM' ler ve shift registerlar

Önceden de belirtildiği gibi, her 4 girişli LUT 16x1 RAM olarak kullanılabilir. Daha önceden verilen CLB yapısı için dört Slice konfigürasyonunu düşünerek, bir CLB' deki bütün LUT'lar birlikte Tek-port 16 x 8 bit RAM, 32 x 4 bit RAM, 64 x 2 bit RAM, 128 x 1 bit RAM ya da Çift-port 16 x 4 bit RAM,32 x 2 bit RAM, 64 x 1 bit RAM gerçeklemek için konfigüre edilebilir.

Alternatif olarak, her 4 bitlik LUT 16 bitlik Shift Register olarak kullanılabilir. Bu durumda, slice içersindeki lojik hücreler arasında ve Slice' ların kendi aralarında ayrılmış özel

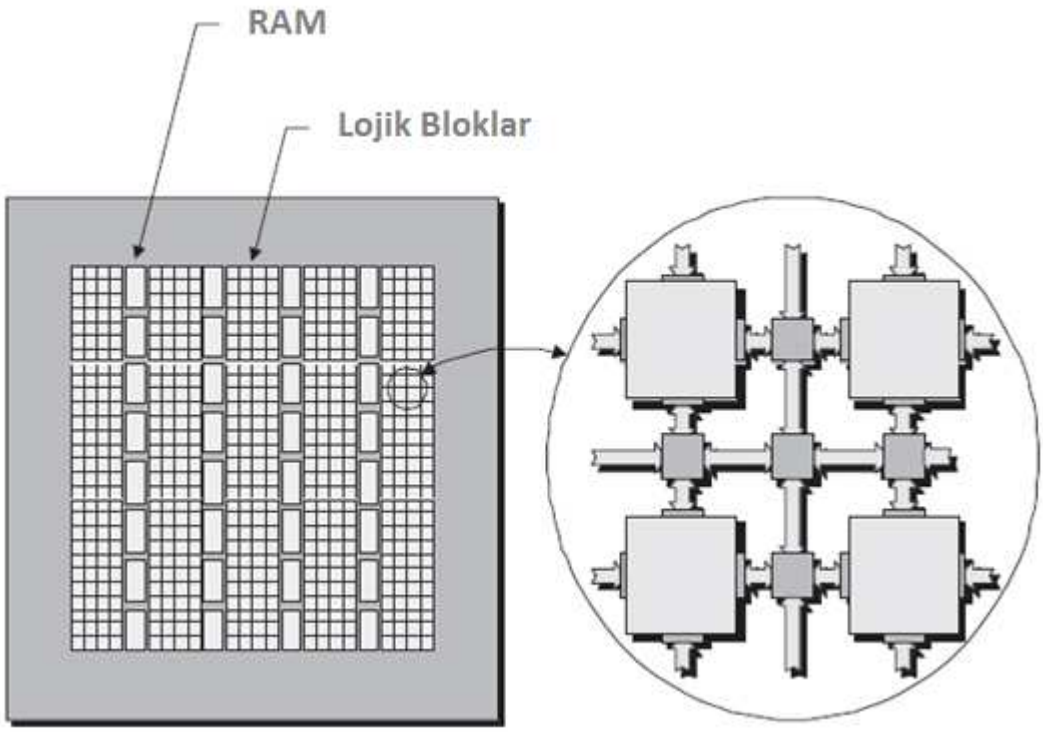
bağlantılar vardır. Bu, sıradan bir LUT' un çıkışını (16 bitlik Register içersinde seçilen bitin içeriğini görüntülemek için kullanılabilir) kullanmadan bir Shift Registerin son bitini diğer Shift Register ın ilk bitine bağlanmasını sağlar. Bu da tek bir CLB içersindeki LUT'ların 128 bite kadar çıkan bir Shift Register ı gerçeklemek için birlikte konfigüre edilmelerini sağlar.

3.3.2. Hızlı elde zincirleri

Modern FPGA' lerin hızlı carry zincirlerini gerçeklemek için gerekli olan özel lojik ve ara-bağlantıları içermeleri kilit bir özelliktir. Önceki kısımda verilen CLB' ler bağlamında, her LC özel carry lojik içerir. Bu, her slice' daki iki LC arasında, her CLB' deki slice'lar arasında ve CLB' lerin kendi aralarında ayrılmış ara-bağlantılar vasıtasıyla tamamlanır. Bu özel carry lojik ve ayrılmış bağlantılar, sayıcılar ve aritmetik fonksiyonlar (toplayıcılar gibi) gibi lojik fonksiyonların performansını artırır. Hızlı carry zincirlerinin olması DSP gibi uygulamalar için kullanılan FPGA' lerde kolaylık sağlar.

3.3.3. Gömülü RAM'ler

Birçok uygulama hafıza kullanımına ihtiyaç duyar. Bu nedenle, FPGA' ler şimdi, e-RAM ya da Blok RAM olarak adlandırılan oldukça büyük gömülü RAM' ler içerir. Malzemenin yapısına bağlı olarak, bu RAM blokları entegrenin çevresine yerleştirilebilir, çipin yüzeyine dağıtılabilir ya da kolonlar halinde yerleştirilebilir. Entegresine bağlı olarak, böyle bir RAM birkaç binden onlarca binlik bite kadar data alabilir. Dahası, bir entegre, onlarcadan yüzlerceye kadar bu RAM bloklarından içerebilir. Böylece, birkaç yüz bin bitten birkaç milyon bite kadar toplam saklama kapasitesi sağlarlar. Her bir RAM bloğu bağımsız olarak kullanılabilir ya da çoklu bloklar daha büyük blokları oluşturmak için birlikte bağlanabilirler. Bu bloklar farklı amaçlar için kullanılabilirler. Örneğin, standart tek ya da çift-port RAM oluşturmak için, FIFO (first-in first-out) fonksiyonları vb…



Şekil 3.19 Gömülü RAM yapısı [12]

3.3.4. Gömülü çarpıcılar, toplayıcılar, MAC' lar

Çarpıcılar gibi bazı fonksiyonlar eğer çok sayıda programlanabilir lojik bloğu birlikte bağlayarak gerçeklenirse niteliği gereği yavaş olurlar. Bu fonksiyonlar birçok uygulamada gerekli oldukları için birçok FPGA' in özel fiziksel olarak bağlı çarpıcı (multiplier) blokları vardır. Bunlar tipik olarak gömülü RAM bloklara yakın mesafede yerleştirilmişlerdir. Çünkü bu fonksiyonlar genellikle birbirleriyle bağlantılı olarak kullanılırlar. Benzer şekilde, bazı FPGA' lerin ayrılmış toplayıcı blokları vardır. DSP tipi uygulamalarda çok yaygın olarak kullanılan bir işlem de multiply-and-accumulate (MAC) (çarp-ve-biriktir) şeklinde adlandırılır. Bu fonksiyon adından da anlaşılacağı gibi iki sayıyı birbiriyle çarpar ve sonucu bir akümülatörde saklanan toplama ekler. Eğer çalıştığınız FPGA sadece gömülü çarpıcıları destekliyorsa, bu fonksiyonu gerçeklemek için çarpıcıyla bir grup programlanabilir lojik bloktan oluşan bir toplayıcıyı birbirine bağlamak zorundasınız. Sonuç ise bazı ilgili flip-floplarda, bir blok RAM' de ya da bir grup dağıtılmış RAM' de saklanır. Eğer FPGA' lerin gömülü toplayıcıları da olsa her şey daha kolay olurdu. Bazı FPGA' ler MAC' ları gömülü fonksiyonlar olarak sağlarlar.

3.3.5. Diğerleri

Günümüzde FPGA firmaları büyük rekabet içerisindedirler. Bu yarışta ön plana çıkmak için firmalar FPGA' lerinin özelliklerini her gün arttırmaktadır. Bunlar; içerdiği lojik kapı ve bellek kapasitesi, maksimum çalışma frekansı, güç tüketimi, ısınma, çevresel bileşenler için sahip olduğu ara yüzler, işaret uyumluluğu, PCB tasarımı sırasındaki kolaylıklar, konfigüre edilebilme kolaylığı, güvenlik gibi özelliklerdir.

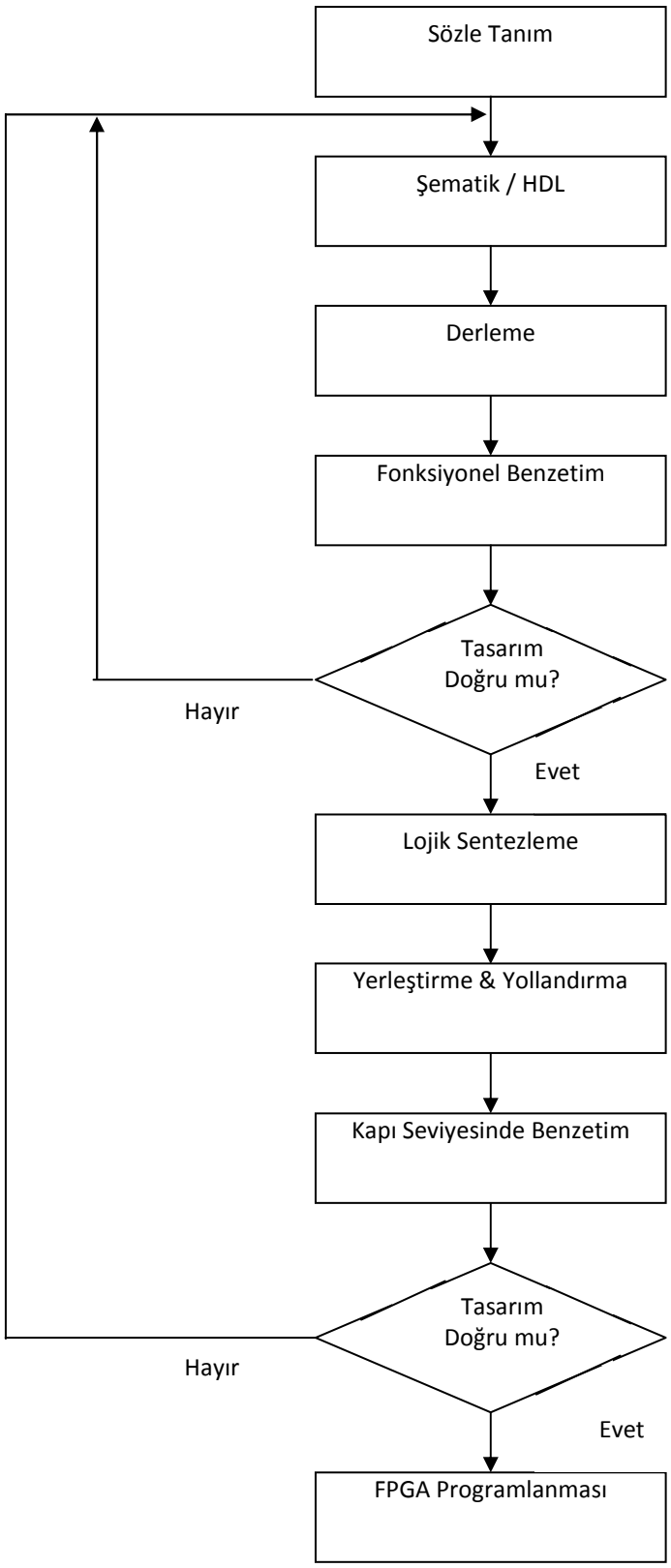
Xilinx FPGA' ler içerisinde DCM (Digital Clock Manager) adı verilen bir yapı saat işaretinin yonga içerisindeki dağıtımını ve saat işaretine özel birçok ihtiyacı karşılamaktadır. Bunlar saat işaretindeki zayıflama, kayma, frekans sentezleme, faz kaydırma, farklı saat işaretleri üretme gibi özelliklerdir.

Xilinx firmasının FPGA dünyasına kattığı önemli özelliklerden birisi de PowerPC tabanlı işlemcileri FPGA içerisine çekirdek şeklinde gömülebilmesidir. Şuanda 550MHz ile en hızlı Xilinx FPGA ailesi olan Virtex5 içerisine PowerPC 440 işlemcisi "hard core" olarak gömülebilmektedir. Aynı şekilde 450 MHz de çalışabilen Virtex 4 Pro ailesi ile de aynı FPGA içerisine 2 adet PowerPC 405 işlemci çekirdeği yerleştirilebilmektedir. Bu işlemcilerin FPGA içerisinde bulunmasıyla birlikte Co-Design adı verilen yapılar oluşturulabilmektedir. Böylece ağır aritmetik işlemler donanım tabanlı gerçeklenirken durum makinesi gerektiren protokol işleri işlemci üzerinde koşan yazılımla gerçeklenebilir.

3.4. FPGA' lerın Programlanması

- ✓ Devrenin sözle tanımı yapılır.
- ✓ Şematik veya HDL kullanılarak tasarım
- ✓ Devreye ait standart bağlantı listesi (Netlist)
- ✓ Fonksiyonel benzetim (Functional Simulation)
- ✓ Lojik sentezleme
- ✓ Kullanılacak FPGA seçimi
- ✓ Sentezleme işleminde varsa, devreye ait lojik kısıtlamalar (I /O, zamanlama, yerleştirme, saat frekansı, kritik yollar,..) kullanıcı kısıtlama dosyası (user constraints file, Xilinx)
- ✓ Lojik sentezleyici ile istenilen fonksiyonların gerekli lojik indirgemeleri (logic optimization)

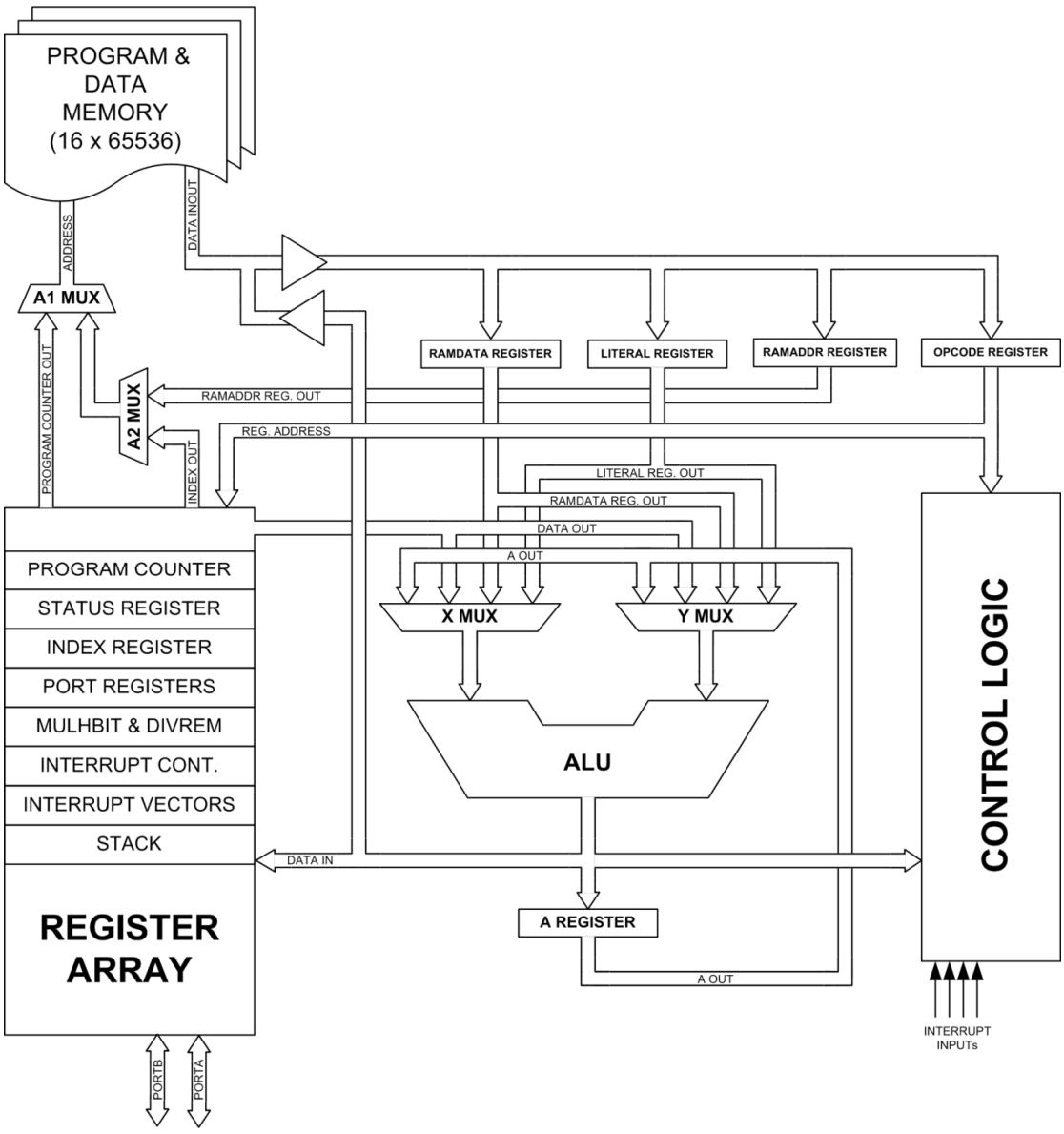
- ✓ Elde edilen lojik fonksiyonlar FPGA içerisindeki lojik bloklarla eşleştirilmesi işlemi (technology mapping) kapı seviyesinde bir bağlantı listesi oluşur.
- ✓ Teknoloji haritalaması sırasında, kullanıcı kısıtlama dosyası da kullanılarak zamanlama gereksinimi karşılanmak amacıyla gerekirse daha fazla lojik eleman kullanımı.
- ✓ Sentezleme sonrasında, yerleştirme ve yollandırma (placement and routing) (Bu adımda, devre fonksiyonları ile eşleştirilmiş lojik bloklar FPGA içerisinde uygun yerlere yerleştirilir ve bu bloklar arasındaki bağlantılar oluşturulur. Lojik yolların daha kısa olması amacıyla birbirleriyle ilişkili CLB' ler yakın yerleştirilir. Yollandırmada ise bağlantılar uygun şekilde seçilir. Örneğin, tasarımın birçok alanında bir işarete ihtiyaç varsa en küçük gecikmeyi sağlamak amacıyla uzun bir yol kullanılır.
- ✓ Bu aşamadan sonra kapı seviyesinde benzetimin gerçekleştirilmesi uygun olacaktır. Çünkü artık bütün CLB' lere (LUT veya çoğullayıcılar ve flip flop' lara) ve yollandırma bağlantılarına ait gecikmeler gerçeğe çok yakın olarak elde edilmiştir. Bu gecikmeler de eklenerek devrenin benzetimi yapıldığında, zamanlama ve hız açısından kritik yollar saptanabilir. Yerleştirme ve yollandırma sonrası benzetimlerde istenilen sonuçlar elde edildikten sonra FPGA' nın programlanması aşamasında kullanılacak bit dizisi, üreticinin sağladığı yazılımla elde edilir. FPGA' nın uygun donanım kullanılarak programlanmasıyla tasarım tamamlanır.



Şekil 3.20 Tasarım Akışı

4. İŞLEMCİ TASARIMI

Bu tez çalışmasında Von Neumann mimarisinde tasarlanmış olan işlemcinin blok diyagramı aşağıdaki şekilde görülmektedir. Tasarlanan işlemci her hangi bir konfigürasyonda 64kx16 'ya kadar bellek elemanını (RAM, ROM, EEPROM…) doğrudan adresleyebilmektedir. İşlemcide, kullanılacak RAM haricinde, işlemci içerisindeki 32 adet 16 bitlik saklayıcıları içeren ve RAM' den daha hızlı işlem yapılabilen bir **saklayıcı dizisi** ***(register array)*** bulunmaktadır. **A (akümülatör)** saklayıcısı ise **saklayıcı dizisinden** ayrı olarak, karalama yapmak için tasarıma eklenmiştir. Aşağıda işlemciye ait blok diyagramı ve modüllere ait açıklamalar görülmektedir.



Şekil 4.1 Tasarlanan işlemcinin blok görünümü.

4.1. Tasarlanan İşlemcinin Donanımsal Bileşenleri

Yukarıdaki blok diyagramda görülen lojik devrelere ait kısa açıklamalar aşağıda verilmiştir.

A1 Çoğullayıcı (A1 MUX):

Adres hattının, program belleğindeki bir sonraki komutu okumak için **program sayacı** *(program counter)* tarafından veya işlenen komutun parametresi olan veri okuma – yazma işlemleri için gerekli adresleri içeren ilgili saklayıcılar tarafından kullanılmasını sağlayan çoğullayıcıdır.

A2 Çoğullayıcı (A2 MUX):

Doğrudan adresleme ve dolaylı *(indirect)* adresleme arasında geçişi sağlayan çoğullayıcıdır.

Bellek Veri Saklayıcısı (RAMDATA REGISTER):

RAM' den okunan verinin yapılacak işlemde kullanılmak üzere kaydedildiği saklayıcıdır.

Sabit İşlenen Saklayıcısı (LITERAL REGISTER):

Programdan okunan sabit işlenen değerinin kaydedildiği saklayıcıdır.

Bellek Adres Saklayıcısı (RAMADDR REGISTER):

Programdan okunan RAM adresinin kaydedildiği saklayıcıdır.

Komut Saklayıcısı (OPCODE REGISTER):

Bellekten okunan komutun kaydedildiği saklayıcıdır.

X ve Y Çoğullayıcıları (X-MUX ve Y-MUX):

ALU girişlerine herhangi bir veri saklayıcısı içindeki bilginin getirilmesini sağlayan çoğullayıcılardır.

Akümülatör (A REGISTER):

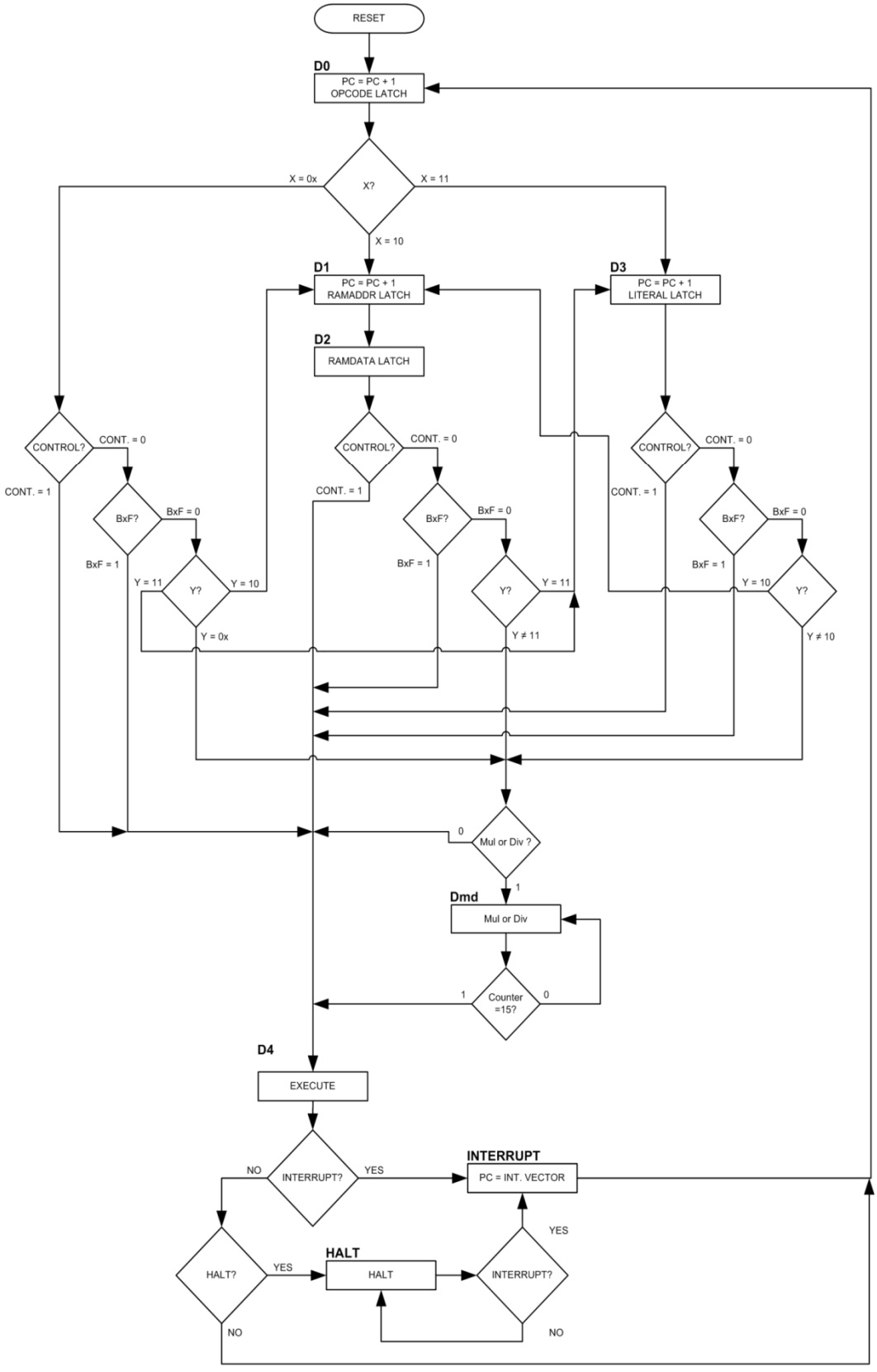
Hızlıca işlem yapmak için, karalama olarak kullanılabilecek saklayıcıdır.

Aritmetik Lojik Birim (ALU):

Aritmetik lojik işlemlerin yapıldığı birimdir. Bir işlem sonucunun oluştuğu her komut ALU' dan geçer. ALU' nun hangi fonksiyonu yerine getireceğine kontrol lojiği karar verir.

4.1.1. Kontrol lojiği tasarımı

İşlemcinin, komutları yorumlayıp, istenilen şekilde gerçekleştirmesini sağlayan lojik devredir. Kontrol lojiğine ait durum akış diyagramı Şekil 2'de görülmektedir. Basit komutların işlenebilmesi için kontrol lojiğinde beş adet durum bulunmaktadır. Bunlara ek olarak çarpma ve bölme işlemlerinin yapıldığı durum **dmd**, **kesme durumu** ve **halt komutu** işlendiğinde, işlemcinin kesme gelinceye kadar bekleyeceği durum tasarıma eklenmiştir. İşlemci, ilk çalışma anından ya da reset' ten sonra ilk saat darbesinin yükselen kenarı ile birlikte **D0** durumuna gelir. Bu durumda saat darbesinin alçalan kenarı ile işlenecek komut kaydedilir ve program sayıcı bir arttırılır. Bu durumdan sonra saat darbesinin yükselen kenarı ile gidilecek durum, işlenmekte olan komutun, işlemde kullanacağı ilk verinin tipine bağlıdır. Bu veri A saklayıcısı, saklayıcı dizisindeki *(register array)* bir saklayıcı, RAM' deki bir saklayıcı veya programdan okunan sabit bir veri olabilir. Bu veri A saklayıcısından veya saklayıcı dizisinden okunacaksa, veriye ulaşmak için saat darbesi gerekmediğinden, başka bir duruma geçilmez. Veri RAM den okunacaksa, önce programda belirtilen RAM adresini okumak için **D1** durumuna, sonra okunan adresteki veriyi okumak için **D2** durumuna geçilir. Programdan sabit değer okumak için ise **D3** durumuna geçilir. Bu aşamadan sonra işlenmekte olan komut bir kontrol komutu *(control)* veya tek saklayıcı verisi ile gerçekleştirilen *(BxF)* bir komut ise doğrudan **D4** durumuna geçilir. Eğer komut iki saklayıcı verisini gerektiren bir komut ise **D0** durumundan sonraki işlem ikinci değişken için aynen tekrarlanır. Komutta kullanılacak bütün veriler işlemci içerisindeki ilgili saklayıcılara kaydedildikten sonra, işlem sonucunu istenilen saklayıcıya kaydetmek için **D4** durumuna geçilir. **D4** durumundan sonra eğer kesme geldiyse **kesme durumuna** geçilir ve ilgili kesme vektörü program sayıcıya kaydedilir. Kesme yoksa ve işlenen komut halt komutuysa, işlemci **halt durumuna** gelir ve kesme gelinceye kadar bekler. Bu işlemler bittikten sonra bir sonraki komutu okumak için **D0** durumuna geri dönülür. FPGA ile gerçeklenecek işlemcinin maksimum saat frekansını arttırmak ve işlemcinin FPGA içerisinde kapladığı alanı azaltmak için, çarpma ve bölme komutları, **dmd** durumuna girilerek, normal komut icra süresinden 16 saat darbesi fazla harcanarak icra edilmektedir. Dallanma komutları programdaki bir etiketin adresine bağlı olarak değil de işlenmekte olan komut ile etiket adresi arasındaki farka göre çalışmaktadır.. Böylece program parçaları bellekte yer değiştirseler bile tekrar derlenmeden çalışabilmeleri sağlanmıştır.



Şekil 4.2 Kontrol lojiği durum akış diyagramı

4.1.2. Saklayıcı dizisi

İşlemci içerisindeki saklayıcı dizisi, işlemcinin çalışması ve konfigürasyonu için gereken kontrol, giriş/çıkış saklayıcıları ve verileri geçici olarak tutmak için kullanıcıya bırakılan saklayıcıları içerir. Aşağıdaki tablo kontrol saklayıcılarını göstermektedir.

Çizelge 4.1: Saklayıcı Dizisi

ADRES	ADI	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00	PC																
01	STATUS											A<B	A>B	A=B	V	C	Z
02	INDEX																
03	PORTA																
04	PORTB																
05	INOUTA																
06	INOUTB																
07	MULHBIT																
08	DIVREM																
09	INTCON	IM	IE	I3E	I2E	I1E	I0E	I3M	I2M	I1M	I0M						
0A	I0VECTOR																
0B	I1VECTOR																
0C	I2VECTOR																
0D	I3VECTOR																
0E - 1F	REG(x)	Kullanıcıya ayrılmış saklayıcılar															

Program Sayacı (Program Counter (PC)):

Program hafızasındaki işlenecek komutun adresini tutan ve program hafızasından veri okunduğunda içindeki değer bir artan saklayıcıdır. Dallanma komutlarında program sayacı işlenmek istenen alt programın başlangıç adresinin değerini almaktadır ve komut işlendiği andaki değeri yığına kaydedilmektedir. Alt programdan geri dönerken program sayacına yığındaki değer geri yüklenmektedir.

Durum Saklayıcısı (Status):

Yapılan işlemlerle ilgili bayrakların bulunduğu saklayıcıdır. Sıfır bitinden başlayarak sırasıyla Z, C, V, A=B, A>B ve A<B bayrakları bu saklayıcının içindedir.

Sıralama Saklayıcısı (Index):

Dolaylı adresleme modunda erişilmek istenen RAM adresinin kaydedildiği saklayıcıdır. 0xFFFF fiziksel adresi ile işlem yapıldığında bu saklayıcının içerisindeki değer işlem yapılacak ram adresi olarak kullanılır.

İskele Saklayıcıları (Porta & Portb):

A ve B iskelelerine erişmek için kullanılan saklayıcılar.

Yönlendirme Saklayıcıları (Inouta & Inoutb):

A ve B iskelelerindeki her bir bitin bir birinden ayrı olarak giriş veya çıkış olarak ayarlanmasını sağlayan saklayıcılardır. İskelelerdeki her bir bit için 0 değeri çıkış, 1 değeri giriş anlamındadır.

Çarpma Ve Bölme Saklayıcıları (Mulhbıt & Dıvrem):

Bir çarpma işlemi yapıldığında yüksek değerli bitlerin ve bir bölme işlemi yapıldığında bölme işlemindeki kalan değerinin kaydedildiği saklayıcılardır.

Kesme Kontrol Saklayıcısı (Intcon):

Kesmelerle ilgili konfigürasyonu yapmayı sağlayan saklayıcıdır. İşlemcide önem sırasına göre ***int0***, ***int1***, ***int2*** ve ***int3*** olmak üzere dört adet kesme girişi vardır. Bu saklayıcının 15. Biti kesme maskesidir ve program bir kesmeye girdiğinde kendiliğinden lojik – 1 olur. Böylece işlemcinin, bir kesmeye ait komutlar icra edilirken başka bir kesmeye girmesi engellenir. Kesme programı bitiğinde yine kendiliğinden sıfırlanır ve kesme programı içerisindeyken bir başka kesme geldiyse işlemci gelen kesmeye ait program parçasına dallanır. Yani bu bit, kesmeleri tamamen engellemez, sadece geçici olarak devre dışı bırakır. Bu bit sıfırlandığı anda, sıfırlanmadan önce bir kesme gelmişse program o kesmenin vektörüne dallanır. Bu bit ayrıca programcı tarafından da kesmeleri bekletmek amacı ile kullanılabilir. *INTCON* saklayıcısın 14. Biti kesmelere izin verme bitidir ve kesmeleri kabul etmek için bu bitin lojik – 1 olması gereklidir. 13 – 10 bitleri her bir kesmeye ait izin verme bitleridir. 9 – 6 bitleri ise her bir kesmenin yükselen kenarda mı yoksa alçalan kenarda mı kabul edileceğini belirleyen bitlerdir. Bu bitler lojik – 0 ise kesme yükselen kenarda, lojik – 1 ise alçalan kenarda kabul edilir.

Kesme Vektörleri:

Her bir kesmeye ait kesme vektörlerinin kaydedildiği saklayıcılardır. Bir kesme sinyali algılandığında, program ilgili kesmeye ait vektöre dallanacaktır.

Yığın (Stack):

İşlemcide, saklayıcı dizisinin içinde fakat diziyi oluşturan saklayıcılardan ayrı olarak tasarlanmış 16 adet yığın saklayıcısı bulunur. Böylece işlemci program akışı bozulmadan 16 adet alt programa dallanabilmektedir. Alt programa dallanmalarda sadece geri dönüş adresi saklanmaktadır.

4.2. Komutlar

Aşağıdaki tabloda XX ve YY, ALU' nun iki girişindeki iki çoğullayıcıların seçme bitlerini temsil ederler. Bu bitler aşağıdaki durumlarda olabilirler:

00 = A saklayıcısı seçilir

01 = Saklayıcı dizisinden REGAD adresi ile gösterilen saklayıcı seçilir.

10 = RAM' den gelen veri seçilir.

11 = ROM'dan okunan sabit veri seçilir.

H biti 0 olduğunda işlem sonucu, XX ile seçilen saklayıcıya, 1 olduğunda YY ile seçilen saklayıcıya kaydedilir. Böylece bütün saklayıcılara erişilebilir. Örnek olarak, ADD ve ADDI komutlarının yaptıkları işlemler, sadece ADD komutu ile yapılabilir. İşlemcide, ayrıca dolaylı adresleme modu da vardır. Dolaylı adreslemede, erişilmek istenen RAM adresi saklayıcı dizisinin 0x02 adresine yazılır. Daha sonra programda RAM' in H'FFFF adresi ile işlem yapılmak istendiğinde saklayıcı dizisinin H'02 adresine yazılmış olan adres ile işlem yapılır. Komutların bellekte kapladığı alan komutun hangi veri tipini kullandığına bağlıdır. RAM adresi ve program belleğinden okunan sabit değerler, komuttan sonraki adreslere, önce XX ile seçilen sonra YY ile seçilen olmak üzere yazılırlar. A saklayıcısı ve saklayıcı dizisi arasındaki işlemler 1 word, sabit değer veya RAM kullanan komutlar 2 word, aynı anda hem sabit değer hem de RAM kullanan komutlar bellekte 3 word yer kaplayacaklardır.

Çizelge 4.2 Komutun RAM' de dizilişi

Komut	ADR	Komut kelimesi
Ram Adresi	ADR + 1	Varsa kullanılan RAM adresi
Sabit Değer	ADR + 2	Varsa RAM'deki sabit değer

Çizelge 4.3 İşlemciye ait komut kümesi

KOMUT		FORMAT	ÖRNEK	İŞLEM
ADD	Addition	00000-XX-YY-H-0-REGAD	ADD X, Y, H	H = X + Y
SUB	Subtraction	10000-XX-YY-H-0-REGAD	SUB X, Y, H	H = X - Y
INC	Increase	00001-XX-YY-H-0-REGAD	INC X, Y, H	H = X + 1
DEC	Decrease	10001-XX-YY-H-0-REGAD	DEC X, Y, H	H = X – 1
MULU	Multiplication (unsigned)	00010-XX-YY-H-0-REGAD	MULU X, Y, H	H = X x Y
MUL	Multiplication	10010-XX-YY-H-0-REGAD	MUL X, Y, H	H = X x Y
DIVU	Division (unsigned)	00011-XX-YY-H-0-REGAD	DIVU X, Y, H	H = X / Y
DIV	Division	10011-XX-YY-H-0-REGAD	DIV X, Y, H	H = X / Y
CLR	Clear All Bits	00100-XX-00-0-0-REGAD	CLR X	X = 16'b0
SET	Set All Bits	10100-XX-00-0-0-REGAD	SET X	X = 16'b1
BCF	Clear Sellected Bit	00101-XX-FBIT-REGAD	BCF X, b	X[b] = 0
BSF	Set Sellected Bit	10101-XX-FBIT-REGAD	BSF X, b	X[b] = 1
AND	Logic – AND	00110-XX-YY-H-0-REGAD	AND X, Y, H	H = X & Y
OR	Logic – OR	10110-XX-YY-H-0-REGAD	OR X, Y, H	H = X | Y
XOR	Logic – EXOR	00111-XX-YY-H-0-REGAD	XOR X, Y, H	H = X ⊕ Y
XNOR	Logic – EXNOR	10111-XX-YY-H-0-REGAD	XNOR X, Y, H	H = X ⊗ Y
NOT	Logic – NOT	01000-XX-YY-H-0-REGAD	NOT X, Y, H	Y = X'
SLL	Shift Logical Left	01001-XX-YY-H-0-REGAD	SLL X, Y, H	Y = X << 0
SLR	Shift Logical Right	01010-XX-YY-H-0-REGAD	SLR X, Y, H	Y = 0 >> X
SAL	Shift Arithmetic Left	01011-XX-YY-H-0-REGAD	SAL X, Y, H	Y = X << Carry
SAR	Shift Arithmetic Right	01100-XX-YY-H-0-REGAD	SAR X, Y, H	Y = Carry >> X
SWP	Swap	01101-XX-00-0-0-REGAD	SWP X, Y, H	H = {X[7:0], X[15:8]

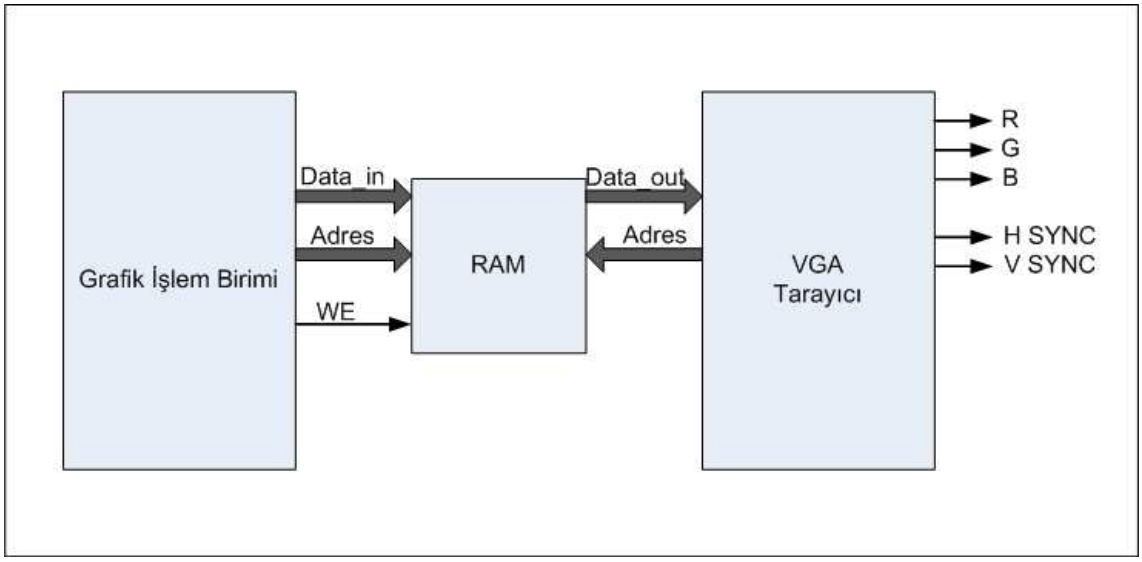
CMP	Compare Data	01110-XX-YY-000-REGAD	CMP X, Y	Flags = X – Y
MOV	Move Data	01111-XX-YY-1-0-REGAD	MOV X, Y	Y <= X
BTJEZ	Bit Test, Jump If Equal to Zero	11000-XX-FBIT-REGAD	BTJEZ X, b	Jump If X[b] = 0
BTJES	Bit Test, Jump If Equal to Set	11001-XX-FBIT-REGAD	BTJES X, b	Jump If X[b] = 1
BEQ	Branch Equal to Zero	11010–11–00–0000000	BEQ LABEL	If Zero = 1 PC <= LABEL
BNE	Branch Not Equal to Zero	11010-11-01-0000000	BNE LABEL	If Zero = 0 PC <= LABEL
BRA	Branch Always	11010-11-10-0000000	BRA LABEL	PC <= LABEL
BAL	Branch and Link	11010-11-11-0000000	BAL LABEL	PC <- LABEL, PC => Stack
RETURN	Return From Subroutine	11011-00-00-0000000	RETURN	PC <= Stack
RETLA	Return and Load REG_A	11011-11-00-0000000	RETLA	PC <= Stack, REG_A <= Literal
RETI	Return From Interrupt	11100-00-00-0000000	RETI	PC <= Stack, INT_MASK <= 0
HALT	System Halt	11101-00-00-0000000	HALT	Halt Operation
NOP	No Operation	11110-00-00-0000000	NOP	No Operation

A saklayıcısı ve saklayıcı dizisi arasındaki bütün işlemler 2 saat darbesi ile yapılır. Yapılacak işlem, sabit değer kullanacaksa 3 saat darbesi, RAM kullanacaksa 4 saat darbesi, ikisini birden kullanacaksa 5 saat darbesi ile yapılır. Çarpma ve bölme komutları normal komut icra süresinden 16 saat darbesi fazla sürede icra edilir.

4.3. Grafik İşlem Birimi

FPGA içerisinde gerçeklenen, işlemciye adres ve veri yolları üzerinden bağlanabilen ve VGA ara yüzü üzerinden herhangi bir monitöre bağlanabilmeyi sağlan bir grafik işlem birimi tasarlanmıştır. Tasarlanan grafik işlem birimi, 640 x 480 piksel çözünürlükte, 50 Hz ekran tarama frekansında ve 8 renk ile işlem yapmaktadır. 8 Renk ile çalışmak için piksel başına 3 bit yeterli olmaktadır. İşlemci, grafik işlem birimine aşağıda belirtilen saklayıcıları kullanarak ekrana yazı yazma, nokta boyama, dörtgen çizme, çizgi çizme ve üçgen çizme gibi işleri yaptırabilmektedir. Üçgen çizme işlemi yapılabildiğinden monitör üzerinde 3 boyutlu şekiller

oluşturmak mümkün olmaktadır. Grafik işlem birimi, işlemciden aldığı komutları kullanarak VGA ara yüzü üzerinden monitörü sürekli tarayan bir durum makinesi ile paylaştığı bir RAM' e ekranda gösterilecek piksellere ait renk bilgilerini yazmaktadır. Grafik arabiriminin blok şeması aşağıda görülmektedir.



Şekil 4.3 Grafik işlem birimine ait blok diyagramı

Gerçeklenen grafik işlem birimine ait saklayıcı tablosu aşağıdaki gibidir:

Çizelge 4.4 Grafik işlem birimine ait saklayıcı tablosu

ADRES	**ADI**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
00	Cmd	W	CMD					Size			B	Bcolor			Fcolor		
01	Char_Code									ASCII karakter kodu							
02	Vaddr/P1V	Koordinat															
03	Haddr/P1H	Koordinat															
04	Vstep/P2V	Koordinat															
05	Hstep/P2H	Koordinat															
06	P3V	Koordinat															
07	P3H	Koordinat															

Cmd saklayıcısı grafik işlem birimine gönderilecek komutun yazıldığı saklayıcıdır. Bu saklayıcıya bir yazma işlemi yapıldığında W bayrağı 1 olur. Grafik işlem birimi yazılan komutu yorumlayıp yapılması istenen işi bitirdiğinde bu bayrağı sıfırlar. Bu saklayıcı içersinde bulunan

Size bitleri, eğer yazı yazma komutu işlenecekse yazının boyunu ayarlamaya yarar. B biti yazının arka plan renginin olup olmayacağını, Bcolor bitleri arka plan rengini, Fcolor bitleri ise grafik işlem birimi tarafından yapılan herhangi bir işte hangi rengin kullanılacağını belirtir. CMD bitlerinin alabileceği değerler ve anlamları aşağıdaki gibidir:

0x00 : Ekrana yazı yazma.
0x01 : Nokta boyama.
0x02 : Dörtgen çizme.
0x03 : Çizgi çizme.
0x04 : Üçgen çizme.

Ekrana Yazı Yazma:

Char_Code saklayıcısında ASCII kodu belirtilen karakter, Vaddr ve Haddr koordinatıyla belirtilen piksel den başlayarak B biti 1 ise Bcolor rengindeki arka palan üzerine fcolor renginde yazılır. Size bitleri ile yazı boyu ayarlanabilir.

Nokta Boyama:

Vaddr ve Haddr koordinatıyla belirtilen piksel fcolor rengine boyanır.

Dörtgen Çizme:

Vaddr ve Haddr koordinatıyla belirtilen piksel' den başlanarak, dikeyde Vstep, yatayda Hstep kadar piksel ileri gidilerek Fcolor renginde bir dörtgen çizilir.

Çizgi Çizme:

Vaddr ve Haddr koordinatıyla belirtilen piksel' den başlanarak, dikeyde Vstep, yatayda Hstep kadar pixel ileri gidilerek Fcolor renginde bir Çizgi çizilir.

Üçgen Çizme:

"P1V, P1H", "P2V, P2H", "P3V, P3H" koordinatlarıyla belirtilen noktalar arasına Fcolor renginde bir üçgen çizilir.

4.4. Derleyici

İşlemcinin kolay programlanabilmesi ve geliştirme sürecini hızlandırmak amacıyla C# dili kullanılarak bir derleyici programı hazırlanmıştır. Derleyici programı, aşağıdaki tabloda derleyici programının kabul ettiği şekilde yazılışları bulunan işlemci komutlarını makine kodlarına çevirir.

Çizelge 4.5 Derleyici notasyonundaki komut listesi

ARİTMETİK LOJİK KOMUTLAR		
No	Komut	Açıklama
1	ADD A, B, h	Toplama: A ile B'yi toplayıp sonucu h=0 ise A'ya, değilse B'ye yazar.
2	SUB A, B, h	Çıkarma: A'yı B'den çıkarıp sonucu h=0 ise A'ya, değilse B'ye yazar.
3	INC A, B, h	Arttır: A'nın içeriğini 1 arttırır sonucu h=0 ise A'ya, değilse B'ye yazar.
4	DEC A, B, h	Azalt: A'nın içeriğini 1 azaltır sonucu h=0 ise A'ya, değilse B'ye yazar.
5	MULU A, B, h	Çarp: A ile B'yi işaretsiz çarpıp sonucun alt bitlerini h=0 ise A'ya, değilse B'ye yazar.
6	MUL A, B, h	Çarp: A ile B'yi işaretli çarpıp sonucun alt bitlerini h=0 ise A'ya, değilse B'ye yazar.
7	DIVU A, B, h	Böl: A'yı B'ye işaretsiz bölüp sonucu h=0 ise A'ya, değilse B'ye yazar
8	DIV A, B, h	Böl: A'yı B'ye işaretli bölüp sonucu h=0 ise A'ya, değilse B'ye yazar.
9	AND A, B, h	And: A ile B'ye lojik "ve" işlemi uygulayıp sonucu h=0 ise A'ya, değilse B'ye yazar.
10	OR A, B, h	Or: A ile B'ye lojik "veya" işlemi uygulayıp sonucu h=0 ise A'ya, değilse B'ye yazar.
11	XOR A, B, h	Xor: A ile B'ye "xor" işlemi uygulayıp sonucu h=0 ise A'ya, değilse B'ye yazar.
12	XNOR A, B, h	Xnor: A ile B'ye "xnor" işlemi uygulayıp sonucu h=0 ise A'ya, değilse B'ye yazar.
13	NOT A, B, h	Not: A'nın tersini h=0 i ise A'ya, değilse B'ye yazar.
14	SLL A, B, h	Sola kaydır: A'yı lojik olarak sola kaydırıp sonucu h=0 ise A'ya, değilse B'ye yazar.
15	SLR A, B, h	Sağa kaydır: A'yı lojik olarak sağa kaydırıp sonucu h=0 ise A'ya, değilse B'ye yazar.
16	SAL A, B, h	Sola kaydır: A'yı aritmetik olarak sola kaydırıp sonucu h=0 ise A'ya, değilse B'ye yazar.
17	SAR A, B, h	Sağa kaydır: A'yı aritmetik olarak sağa kaydırıp h=0 ise A'ya, değilse B'ye yazar.
PROGRAM AKIŞI KONTROL KOMUTLARI		
18	BTJEZ A, Bit	0 ise atla: A'nın işaret edilen biti 0 ise bir sonraki komutu işlemeden atlar.
19	BTJES A, Bit	1 ise atla: A'nın işaret edilen biti 1 ise bir sonraki komutu işlemeden atlar.
20	BEQ Etiket	0 ise dallan: Z bayrağı 1 ise etikete dallanır.
21	BNE Etiket	0 değilse dallan: Z bayrağı 0 ise etikete dallanır.
22	BRA Etiket	Her zaman dallan: Etikete dallanır.

23	BAL Etiket	Dallan ve sayacı kaydet: Dallanır ve bir sonraki komut adresini yığına kaydeder.
24	RETURN	Dön: Alt programdan ana programa döner.
25	RETLA Sabit	Dön ve akümülatörü yükle: Alt programdan döner ve akümülatöre sabit yükler.
26	RETI	Kesmeden dön: Kesme programından ana programa döner.
27	HALT	Bekle: kesmeleri beklemek üzere bekleme konumuna girer.
SAKLAYICI İŞLEMLERİ KOMUTLARI		
28	CLR A	Temizle: A'nın bütün bitlerini sıfırlar.
29	SET A	Doldur: A'nın bütün bitlerini 1 yapar.
30	BCF A, Bit	Bit sıfırla: A'nın işaret edilen bitini sıfırlar.
31	BSF A, Bit	Bit doldur: A'nın işaret edilen bitini 1 yapar.
32	SWP A, B, h	Değiştir: A'nın alt bitleri ile üst bitlerini değiştirip h=0 ise A'ya, değilse B'ye yazar.
YÜKLE-YAZ KOMUTLARI		
33	MOV A, B	Taşı: A'nın içeriğini B'ye kaydeder.
DİĞER KOMUTLARI		
34	CMP A, B	Karşılaştır: A ile B'yi karşılaştırıp sonuca göre bayrakları düzenler.
35	NOP	Boş işlem: Hiçbir işlem yapmaz.

Derleyicide yukarıda verilen komut açıklamalarındaki **h** biti yerine **R (right)** veya **L (left)** yazılarak işlem sonucun, sağdaki veya soldaki saklayıcıya kaydedilmesi sağlanabilir. Bir komutun program belleğinde kaplayabileceği en fazla adres alanı 3 word olduğundan, **BTJEZ** ve **BTJES** komutları yukarı da açıklanan koşullar sağlandığında okunacak komutun adresini 3 ileri götürmektedirler. Bu nedenle, bu komutlardan sonra yazılan komut, program hafızasında 3 word'den daha az yer kaplıyorsa, 3'e tamamlamak için komuttan sonra **NOP** komutları eklenmelidir. **INC, DEC, NOT, SLL, SLR, SAL, SAR, SWP** komutlarının işlenmesi için tek saklayıcı içeriği yeterlidir fakat işlem sonucu kaynak saklayıcıdan farklı bir saklayıcıya kaydedilebilmektedir. Bu nedenle, derleyicide yazım kolaylığı için bu komutlarla birlikte tek saklayıcı kullanılırsa **h** biti 0, iki saklayıcı kullanılırsa **h** biti 1 yapılmaktadır ve derleyici bu komutlar ile birlikte **R** veya **L** yazılmasına izin vermemektedir.

4.5. Fiziksel Olarak Gerçekleştirilen Uygulamalar

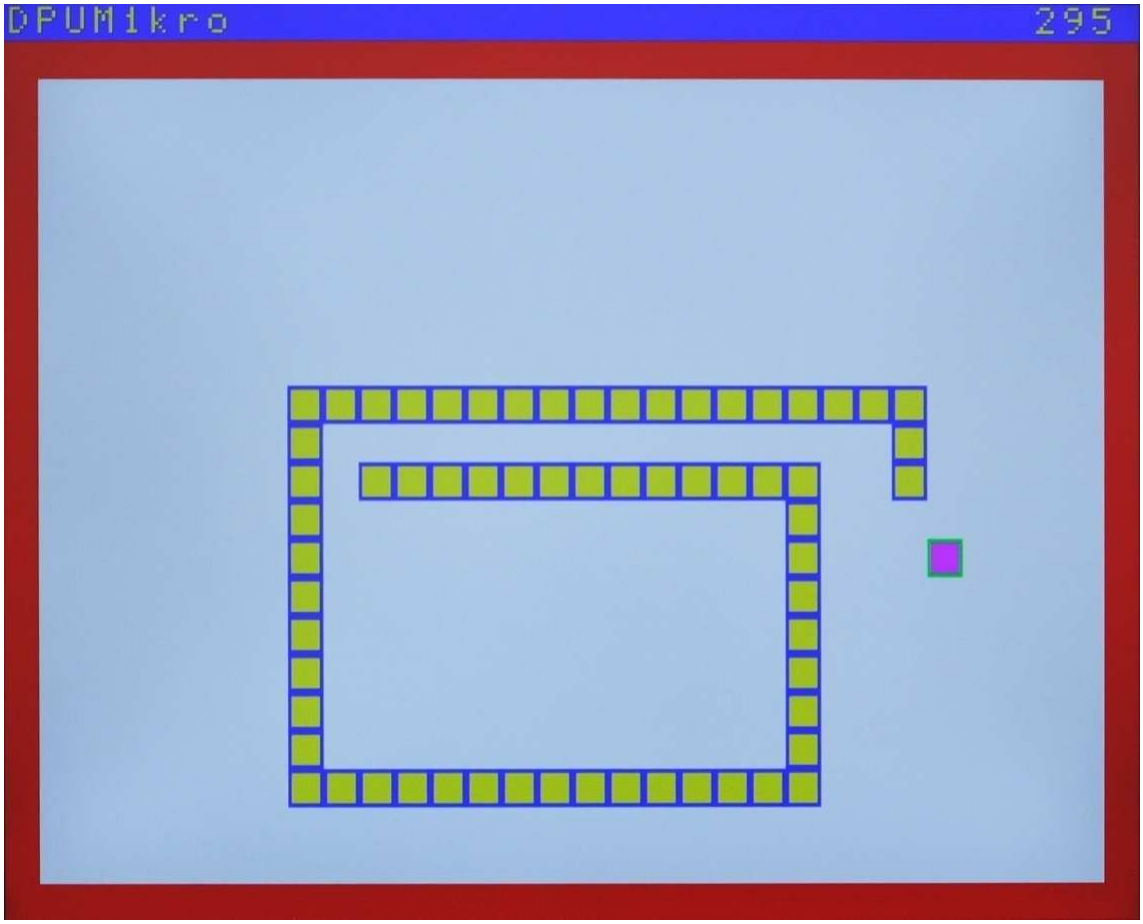
Tasarlanan işlemci ve grafik işlem birimi kullanılarak Spartan 3E Starter Kit üzerinde 2 adet uygulama gerçekleştirilmiştir. FPGA içerisinde işlemciyle birlikte kullanılmak üzere, çevresel donanımlar tasarlanıp oluşturulan bilgisayar sistemine eklenmiştir. Bu sistemde FPGA içerisinde bulunan BlockRam'ler kullanılarak işlemcinin kullanacağı hafıza birimi ihtiyacı karşılanmıştır. Dışarıdan alınan 50 MHz' lik saat darbesi çip içersinde 25 MHz' e indirilerek bütün modüllere saat darbesi dağıtılmıştır. Gerçeklenen uygulamalarda belirli zaman aralıklarında kesme üretmek için programlanabilir bir zamanlayıcı (timer) tasarıma eklenmiştir. Bilgisayar sistemine dışarıdan veri girişini sağlamak için bir klavye kontrol devresi de eklenmiştir. PS/2 ara yüzünü kullanan bütün klavyeler böylece tasarlanan siteme bağlanabilmektedir. Ayrıca 3 boyutlu küp uygulamasında trigonometrik işlemleri gerçekleştirebilmek için sinüs ve kosinüs tabloları işlemci tarafından direkt olarak adreslenebilecekleri şekilde oluşturulan sisteme bağlanmıştır. Tüm bu oluşturulan modüller sentezlenip FPGA içerisine yüklenmeye hazır hale getirilmişlerdir. Bütün tasarım çip içerisinde 25 MHz' de çalışmaktadır. Xilinx' in standart ayarları kullanıldığında tüm tasarım FPGA içerisinde % 75 yer kaplamaktadır. Aşağıda Tasarımın çip içerisindeki yerleşimini gösteren tablo görünmektedir.

Device Utilization Summary				[-]
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	1,633	9,312	17%	
Number of 4 input LUTs	6,056	9,312	65%	
Logic Distribution				
Number of occupied Slices	3,498	4,656	75%	
Number of Slices containing only related logic	3,498	3,498	100%	
Number of Slices containing unrelated logic	0	3,498	0%	
Total Number of 4 input LUTs	6,277	9,312	67%	
Number used as logic	6,056			
Number used as a route-thru	221			
Number of bonded IOBs	43	232	18%	
IOB Flip Flops	64			
Number of RAMB16s	20	20	100%	
Number of BUFGMUXs	6	24	25%	
Number of MULT18X18SIOs	9	20	45%	

Şekil 4.4 Tasarımın FPGA içerisindeki yerleşimi

4.5.1. Yılan oyunu

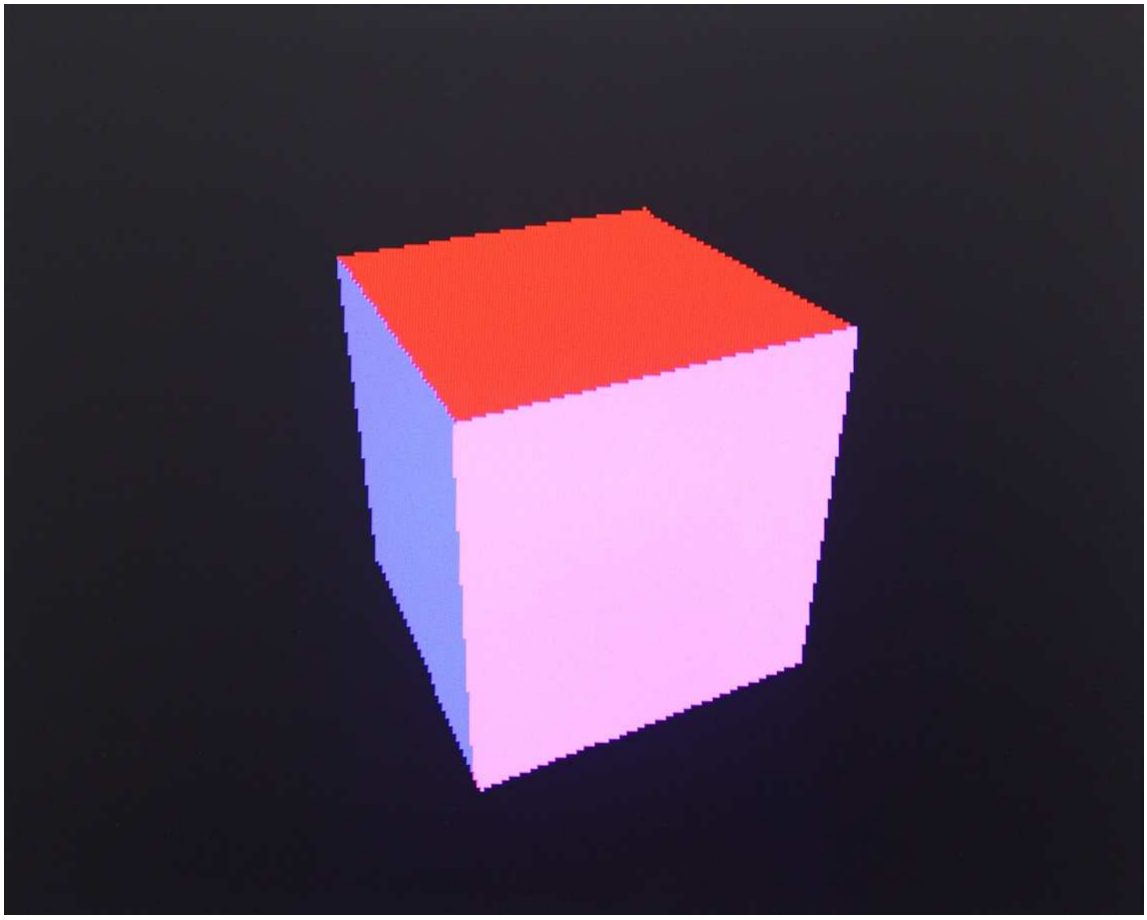
İşlemci üzerinde makine dilinde yazılan bir yılan oyunu denemesi başarıyla çalıştırılmıştır. Oyunun kontrollerini sağlamak için yine Verilog HDL kullanılarak bir klavye kontrolcüsü de FPGA içerisinde gerçeklenmiştir. Oyuna ait ekran görüntüsü aşağıda görülmektedir.



Şekil 4.5 Yılan oyununa ait ekran görüntüsü

4.5.2. 3 boyutlu küp uygulaması

İşlemci ve tasarlanan grafik işlem birimi kullanılarak, yoğun matematiksel işlemler gerektiren 3 boyutlu küp uygulaması makine dilinde hazırlanıp FPGA içerisinde gerçeklenen bilgisayar sistemine yüklenmiştir. Ekranda oluşturulan küp 3 eksen etrafında bir birinden bağımsız olarak döndürülebilmektedir.



Şekil 4.6 3 Boyutlu küp uygulamasına ait ekran görüntüsü

5. SONUÇLAR

Sonuç olarak, bu çalışmada makine dilinde dahi kolayca programlanabilen gerçek zamanlı işlemler için kullanılabilecek güçlü bir mikroişlemci elde edilmiştir. Tasarlanan mikroişlemci sentezlenerek FPGA' ye yüklenme hazır hale getirilmiştir. Bu işlemci, FPGA gibi programlanabilir lojik devre elemanları ile kullanılarak yapılan tasarımlarda, hali hazırda Xilinx ve Altera gibi firmaların sağladıkları ve Soft Processor Core olarak bilinen işlemcilere alternatif olarak kullanılabileceği gibi, tasarıma eklenecek çeşitli çevresel donanımlar ve hafıza birimleriyle birlikte ASIC olarak da gerçeklenebilir. Aşağıdaki tabloda, mevcut işlemcilerin komut icra süreleri değerlendirilerek yapılan bir karşılaştırma verilmiştir.

Çizelge 4.6 İcra süresi karşılaştırması

	Komut icra süresi (Saat Darbesi)			
Adresleme Modu	**Tasarlanan İşlemci**	**8051**	**6800**	**PIC16F**
Saklayıcılar arası	2	12	2 - 3	4
Direkt RAM	4	12	6	4
İvedi - Saklayıcı	3	12	2 - 3	4
İvedi - RAM	5	24	YOK	YOK
Sıralı	4	12	6	4
İvedi - Sıralı	5	24	YOK	YOK
Dallanma	3	24	4	4 - 8
Çarpma / Bölme	+ 16	48	YOK	YOK

Tasarım fiziksel olarak gerçeklendiği Spartan 3E FPGA yerine daha ileri teknolojileri kullanan bir FPGA seçilerek tasarımın daha yüksek saat frekanslarında koşması sağlanabilir. Ayrıca eğer tasarım ASIC olarak gerçeklenirse çalışma frekansı yine yükselecektir.

KAYNAKLAR DİZİNİ

[1] Palnitkar S., 1996, Verilog HDL A Guide to Digital Design and Synthesis, SunSoft Press

[2] Janiszewski, I., Baraniecki, R., Siekierska, K., 1999, A Reusable Microcontroller Core's Design, IEEE, VHDL International Users Forum Fall Workshop (VIUF '99), pp. 14-21.

[3] ALAER, E., TANGEL, A., YAKUT, M. 2008, MIB-16 FPGA Based Design and Implementation of a 16-Bit Microprocessor for Educational Use, WSEAS TRANSACTIONS on ADVANCES in ENGINEERING EDUCATION, pp. 326-330.

[4] Herman, H. S., Srihari, C., Matthew, M., 2000, Pipeline Reconfigurable FPGAs", Journal of VLSI Signal Processing Systems, pp. 24, 129-146.

[5] Bezarra, E. A., Gough, M.P., 1999, A Guide to Migrating from Microprocessor to FPGA Coping the Support Tool Limitations, ELSEVIER Microprocessor and Microsystems 23, pp. 561-572.

[6] Mano, M. M., 1996, Computer System Architecture, Prentice Hall

[7] Gray, J., 2000, Hands-on computer architecture: teaching processor and integrated systems design with FPGAs, Workshop On Computer Architecture Education, Article No.: 17

[8] Shaout, A., Eldos, T., 2003, On the Classification of Computer Architecture, International Journal of Science and Technology

[9] Lee, W. F., Shakaff, A.Y.M., 2007, Implementation of 128/256 Bit Data Bus Microprocessor Core on FPGA, ICGST-PDCS Journal, Volume 7

[10] Mano, M. M., Kime C. R., 2004, Logic and Computer Design Fundamentals, Prentice Hall

[11] Wilson G. R., 2002, Embeded Systems and Computer Architecture, Newnes

[12] http://www.xilinx.com/support/education-home.htm

[13] http://www.altera.com/education/edu-index.html

[14] http://www.opencores.org

[15] http://www.actel.com