YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FPGA KULLANARAK 16 BİTLİK MİKROİŞLEMCİ TASARIMI

Elektronik ve Haberleşme Müh. Emre ÖZTÜRK

FBE Elektronik ve Haberleşme Mühendisliği Anabilim Dalı Elektronik Programında Hazırlanan

YÜKSEK LİSANS TEZİ

Tez Danışmanı : Prof. Dr. Herman SEDEF

İSTANBUL, 2010

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

Λ	Lojik VE
V	Lojik VEYA
<<	Sola lojik öteleme
>>	Sağa lojik öteleme
<<<	Sola aritmetik öteleme
>>>	Sağa aritmetik öteleme

KISALTMA LİSTESİ

FPGA	Field Programmable Gate Array
CPU	Central Processing Unit
ALU	Arithmetic Logical Unit
RISC	Reduced Instruction Set Computer
ISA	Instruction Set Architecture
LIFO	Last In First Out
MBR	Memory Buffer Register
MAR	Memory Address Register
PS	Program Sayacı
BRKÇ	Birikeç
ACC	Akümülatör
UART	Universal Asynchronous Receiver/Transmitter
RAM	Random Access Memory
SW	Switch
LED	Light Emitting Diode
LCD	Liquid Crystal Display
HDL	Hardware Description Language
VGA	Video Graphics Array

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖNSÖZ

Çalışmada, içerisinde milyonlarca lojik kapı barındıran bir yapı olan FPGA üzerinde, bu kapıları birbirleriyle ilişkilendirmek suretiyle bir mikroişlemci mimarisi tasarlamak ve bu mimarinin çalışırlığını ispatlamak amaçlanmıştır.

Üniversite akademik çalışmalarının amacının not ya da ünvan değil öğrenmek olduğunu bana idrak ettiren çok değerli danışman hocam Prof. Dr. Herman SEDEF başta olmak üzere, tez çalışmam boyunca bilgi ve deneyimlerini benden esirgemeyen arkadaşım Hakkı KURUMAHMUT' a, bu zamana kadar herhangi bir konuda vazgeçme noktasına geldiğim her an manevi destekleriyle bana tekrar direnç kazandıran aileme, akademik kariyerim için iş saatlerinde bana esneklik sağlayan yöneticim Kutlay TETİK' e teşekkürü bir borç bilirim.

Emre ÖZTÜRK

Kasım, 2010

ÖZET

Tarih boyunca insan oğlu kendi yapabildiklerinden daha fazla aritmetik işlem yapabilen makinalar geliştirmişlerdir. Bu geliştirilenler çok ilkel tasarımlarla başlasa da insan oğlunun gittikçe artan bilgi birikimiyle oldukça karmaşık tasarımlar da meydana çıkmaya başlamıştır. Son yıllarda farklı hesaplama teknikleri barındıran birçok bilgisayar mimarisi geliştirilmiştir. Bunlardan en çok kabul görenlerden biri de Von-Neumann mimarisidir. Çok yetenekli bir matematikçi olan John Luis Von-Neumann, 1945 yılında genel amaçlı, programlanabilir ve içerdiği bellek birimi vasıtasıyla veri, aynı zamanda program kaydedebilen (yeniden programlanabilir) bir mimari ortaya koymuştur.

Bu çalışma, Von-Neumann mimarisi baz alınarak, FPGA (Field Programmable Gate Array) üzerinde tasarlanan bir mikroişlemci olan HSEO16' yı konu edinmektedir. Buna ek olarak çalışma, ortaya konan tasarım üzerinde yazılım geliştirmek için ayrıca bir yazılım geliştirme ortamı sunmaktadır.

Bu çalışmadaki mikroişlemci, RISC (Reduced Instruction Set Computer) mimarisinden de bazı kesitler barındırmaktadır. Adından da anlaşılacağı gibi RISC, azaltılmış komut kümesi içeren mimari anlamındadır. Öyle ki, bu çalışmada kullanılan komut kümesi normalden daha az ve adresleme modları sınırlı sayıdadır.

ABSTRACT

History has marked a large number of man endeavours towards building machines that are capable of performing arithmetic operations more efficiently than he can do himself. These started with very primitive instruments but evolved over the course of time due to the accumulative knowledge of man kind. In the recent decades, many computer architectures exhibiting various design methodologies and computation models have been developed. One of the most widely accepted of which is von-Neumann architecture.

The brilliant mathematician, John Louis von-Neumann (1903 - 1957) proposed - in 1945 - a model for a general purpose computer that provides programmability and re-programmability thanks to a memory structure that stores programs and data.

This thesis introduces HSEO16, A microprocessor that adopts von-Neumann architecture and is implemented on FPGA (Field Programmable Gate Array) . In addition to that, the thesis presents a software development environment for HSEO16.

HSEO16 also exhibits the characteristics of a RISC (Reduced Instruction Set Computer). It has a small set of instructions and a limited number of addressing modes.

1. GİRİŞ

FPGA günümüzde endüstriyel alanda kullanımı gün geçtikçe artan bir tümleşik devredir. Başlıca kullanım alanları, kontrol sistemleri, işaret ve görüntü işleme, kablosuz ağlar ve bir alt kullanım alanı olan modellemedir. Burada modellemeden kasıt lojik mantıkla çalışan dijital bir elemanın tasarımdan once FPGA üzerinde modellenmesi ve çalışma şartlarının incelenmesidir.

Modelleme alanının en bilinen örnekleri ise FPGA kullanılarak mikroişlemci tasarımı konusundadır. Örnek vermek gerekirse, ülkemizde düzenlenen CPU Turkey 2008 yarışmasında konuyla ilgili olarak, sanal işlemci tasarımı ve fiziksel işlemci tasarımı kategorilerinde bazı çalışmalar yapılmıştır. Sanal işlemci tasarımı kategorisi birincisi Başak' ın (2008) SelCPU adlı çalışmasında 32 – bit veri yolu, 16 Gigabyte kapasiteli bir bellek ve toplam 17 yazmaç barındıran bir işlemci modellenmiştir. Yine aynı kategoride ikincilik kazanan Ergin vd.' nin (2008) tasarladığı Kasırga çalışmasında 16 – bit veri yolu, 4 Kilobyte kapasiteli bir bellek ve toplam 9 yazmaç barındıran bir işlemci ortaya konmuştur. Fiziksel işlemci kategorisi birincisi Özmen vd.' nin (2008) tasarladığı DPUMikro çalışmasında 16 – bit veri yolu, 64 Kilobyte bellek kapasitesi ve toplam 5 adet yazmacı olan bir deneysel işlemci ortaya konmuştur. Yine aynı kategoride Ertürk vd.' nin (2008) tasarladığı Cpu_kulis adlı çalışmada 16 – bit veri yolu, 2 Kilobyte bellek kapasitesi ve 6 adet yazmacı olan bir deneysel işlemci ortaya konmuştur. Uluslararası literatürde ise Chang vd.' nin (2005) yaptığı çalışmada 25 MIPS (Million instruction per second) mertebesinde işlem gerçekleştirebilen bir işlemci geliştirilmiştir. Başka bir FPGA kartında da bir görüntü işleme donanımı gerçeklenip bu kartların birbirine uygun şekilde bağlanmasıyla görüntü işleme hızı gerçeklenen işlemci vasıtası ile arttırılmıştır.

Bu çalışmada ise 16 – bit veri yoluna, 1 adet akümülatör, 8 adet genel amaçlı yazmaç, 3 adet indeks yazmacı, 2 adet çarpım birimi yazmacı, MAR, MBR, ve IR olmak üzere toplam 17 adet yazmaca, 8 Kilobyte kapasiteli bir bellek birimine, 33 adet komuttan oluşan bir komut setine ve 65.95 MHz maksimum çalışma frekansına sahip bir mikroişlemci (HSEO16) tasarlanmıştır. Mikroişlemciye erişimin kolaylaştırılması ve mikroişlemciye dışarıdan program yüklenebilmesi hedeflenerek bir de yazılım geliştirme arayüzü (Smart Assembler) tasarlanmıştır. Bu arayüz sayesinde kullanıcı tarafından işlemci komut seti kullanılarak hazırlanan programlar seri port üzerinden mikroişlemciye yüklenebilmektedir.

2. FPGA ve ALANDA PROGRAMLANABİLİRLİK

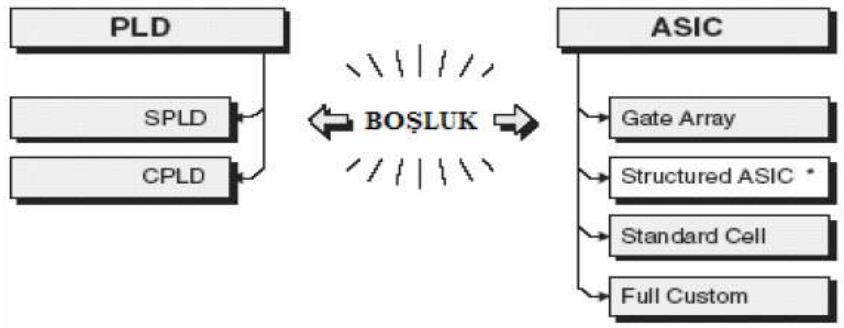
FPGA, Field Programmable Gate Array teriminin kısaltmasıdır. Türkçe karşılığı ise Alanda Programlanabilir Kapı Dizileri' dir. Kapı ve dizi kelimeleri ardı ardına sıralanmış lojik kapılar anlamına gelirken alanda programlanabilirlik biraz daha detaylandırılması gereken bir terimdir.

Alanda programlanabilirlik terimini açıklarken kullandığımız kod yazım dili olan HDL (Hardware Description Language) ile tanımladığımız yapıların bir donanıma denk düştüğünü unutmamamız gerekir. Öyle ki; şematik tasarım vb. tasarım yöntemlerinde çoğunlukla elde edilen donanım üretim bandından çıktıktan sonra test edilebilmektedir. Bu aşamada bulunan bir hata düzeltilmek istendiğinde, donanımın şematik tasarımında ilgili değişiklikler yapıldıktan sonra tekrar üretim aşamasına geçilir. Bu da tasarımcıya önemli bir maliyet getirir. FPGA gibi tümleşik devreler üzerinde tanımladığımız şeyler hali hazırda bir donanıma denk düştüğünden, her hangi bir değişiklik gerektiğinde sadece kodumuzu değiştirerek FPGA elemanını tekrar programlamak yeterli olacaktır. İşte bu bize alanda programlanabilirlik terimini açıklamaktadır.

2.1 FPGA' in Doğuşu

Programlanabilir lojik tüm devreler ilk olarak PLA (Programmable Logic Array) , PAL (Programmable Array Logic) , GAL (Generic Array Logic) , CPLD (Complex Programmable Logic Device) gibi yapılarla ortaya çıkmışlardır. Bunlardan PLA ve PAL içerdikleri kapı dizilerinin sabit, sadece bir kez programlanabilir ve karmaşık yapılar için son derece yetersiz yapılardır. GAL ise PLA gibi yapılara gore biraz daha kapasiteli ve tekrar programlanabilir olmasına rağmen karmaşık ve geniş yapıların tasarımı için uygun değildir. CPLD ise saydıklarımız içinde en kapasiteli olan yapıdır. Tekrar programlanbilirlik özelliği de mevcuttur. Fakat içindeki dizi yapıları sabit olduğundan daha ziyade kombinasyonel lojik tasarımlar için uygundur. Bütün bu eksikliklerle ASIC (Application Specific Integrated Circuit) tasarımında oldukça vakit kaybettiğini düşünen elektronik dünyası FPGA tümleşik devresini tasarlamıştır. Öyle ki; FPGA içinde her kapı bir diğerinden bağımsız, oldukça büyük kapasiteye sahip, bağımsız kapıların ilişkilendirilmesi tamamen kullanıcıya bırakılmış ve LUT (Look Up Table) mantığına göre çalışan bir yapıdır. Yani CPLD gibi sadece kombinasyonel yapılara değil içerisinde olası durum makinaları barındıran ardışıl yapıların tasarımına imkan sağlamaktadır. FPGA sayesinde tasarımcılar ASIC yapılarda çok büyük

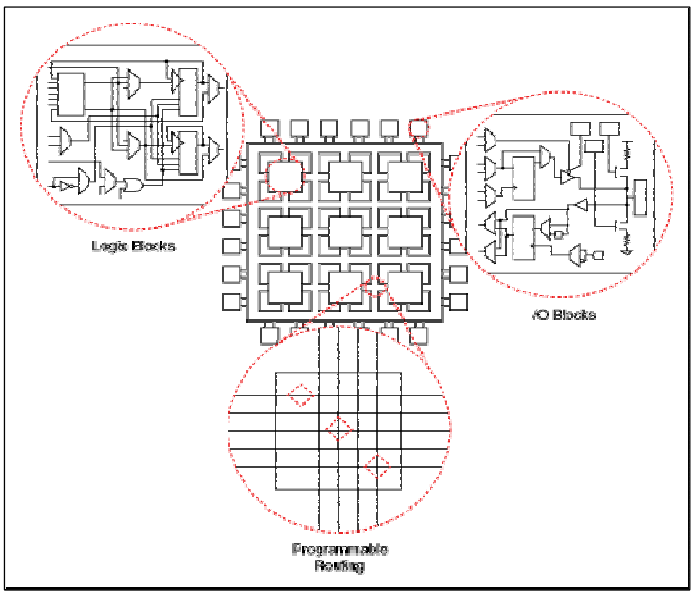
maliyet azaltma şansı yakalamışlardır. Ticari anlamda ilk FPGA (XC2064) Xilinx firması tarafından 1985 yılında piyasaya sürülmüştür.



Şekil 2.1 FPGA' in doğuşu

2.2 FPGA Alt Bileşenleri

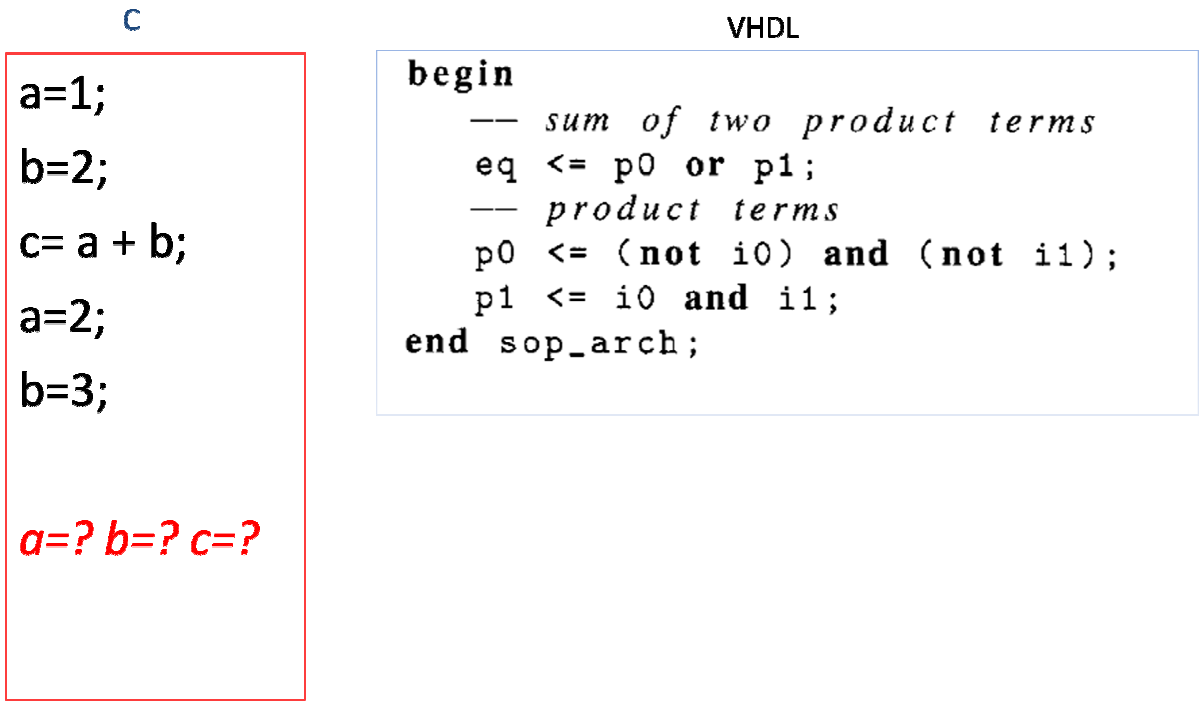
FPGA tümleşik devresi üç temel yapıdan oluşur. Bunlar, birden fazla lojik kapıyı içinde barındıran lojik bloklar, bu lojik blokları birbirine bağlayarak ilişkilendirilmelerini sağlayan programlanabilir ara bağlantılar ve FPGA elemanını dış dünyaya bağlayan giriş – çıkış portlarıdır. Şekil 2.2' de bu alt bileşenler gösterilmiştir.



Şekil 2.2 FPGA alt bileşenleri

2.3 Paralel İşlem Yeteneği

FPGA elemanının en önemli işlemsel artısı paralel işlem yapabilme özelliğidir. Öyle ki; HDL dilinde değişken sinyaller tanımlanabilmektedir. Bu sinyallerin bir girişe atanarak oluşturacağı sonuç, giriş sinyallerindeki olası bir değişiklikte hemen güncellenir. Bu konuyu Şekil 2.3' teki örnek üzerinde detaylandırabiliriz.



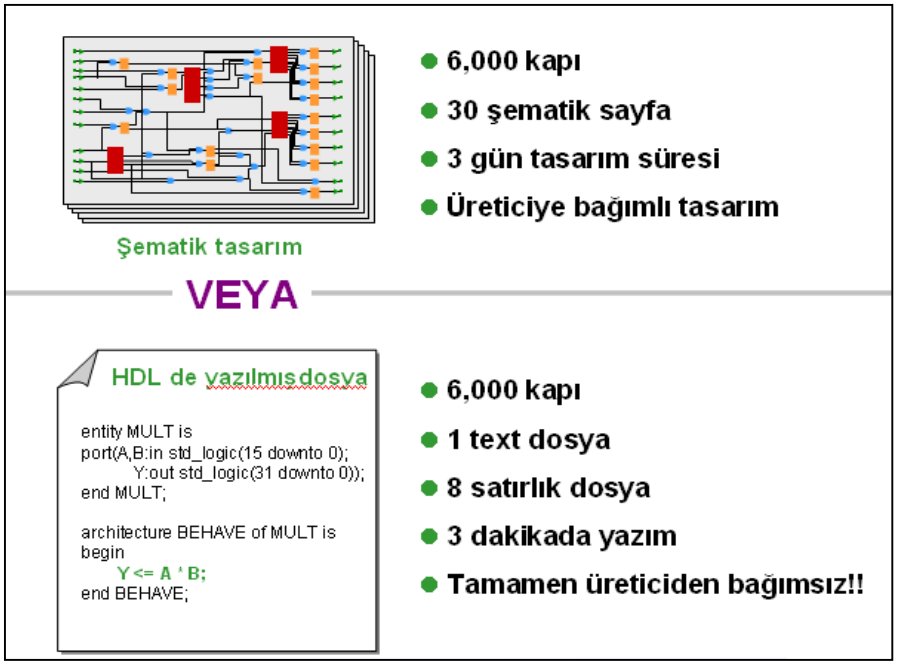
Şekil 2.3 Paralel işlem örneği

Şekil 2.3' te bir adet C programlama dilinde, bir adet ise VHDL progamlama dilinde yazılmış iki adet kod parçası görülmektedir. C dilinde yazılmış olana dikkat edersek, *a* ve *b* değişkenlerine bir değer atandıktan sonra, bu değerlerin toplamı *c* değişkenine atanmıştır. Bu aşamadan sonra ise *a* ve *b* değişken değerleri güncellenmiştir. Bu program sonucunda *a* ve *b* değişkenleri güncellenmesine rağmen *c* değeri *a* ve *b* değişkenlerinin ilk değerleri toplamı olan 3' e eşit olacaktır. Çünkü program sayacı birer birer artarak işlemlerin sırasıyla yürütülmesini sağlamaktadır. VHDL örneğinde ise, *eq* sinyalinin *p0* ve *p1* sinyallerinin lojik veya işlemine tabi tutulması sonucu elde edilen sinyale eşit olduğu programın en başında belirtilmiştir. Bu noktadan sonra *p0* ve *p1* sinyalleri güncellenmektedir. İşte bu noktada, bir önceki örneğin aksine *eq* sinyali, *p0* ve *p1* sinyallerinin güncel hallerinin lojik veya işlemine tabi tutulması sonucu oluşan çıkış sinyaline eşittir.

2.4 HDL Tasarım Dili ve Şematik Tasarıma Göre Avantajları

HDL (Hardware Description Language) elektronikte herhangi bir yazılım programlama dilinin elektronik devreleri, özellikler sayısal devreleri, tasarlama formu olarak düşünülebilir. HDL ile sayısal bir devrenin operasyon şekli, tasarımı ve organizasyonu belirlenebilmektedir. Ayrıca HDL üzerinde yazılan bir kodun denk düştüğü donanımın simülasyonu da mümkündür. Sadece düğümleri birbirine bağlamakla yetinen lojik tasarım dillerinden (netlist dilleri) farklı olarak HDL zamanlama konusunda da tasarımcıya sayısız olanaklar sağlamaktadır. Bu özellik saat darbesi bazlı çalışan donanımların tasarımları için çok önemli bir özelliktir.

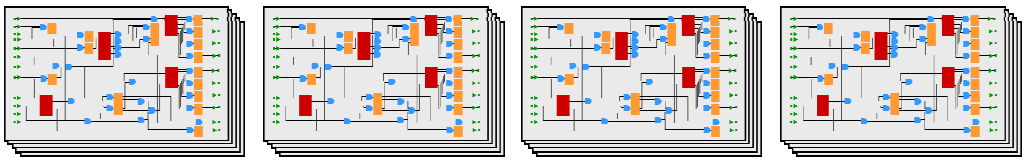
Şematik tasarım ile HDL tasarımı karşılaştırmak için gerçek bir örnekten faydalanılmıştır. Aşağıdaki iki şekilde 16x16' lık bir çarpma devresinin hem şematik tasarım yöntemiyle hem de HDL ile tasarlandıktan sonra mevcut yapının 32x32' lik bir çarpma devresine dönüştürülmesi gösterilmiştir.



Şekil 2.4 16x16 çarpma devresi

Görüldüğü üzere 16x16' lık bir çarpma devresi yapabilmek için şematik tasarımda oldukça fazla emek harcamak gerekir. Bu tasarımcıya zaman ve maliyet olarak geri döner. Aynı yapının HDL ile tasarımında ise yaklaşık 6000 adet lojik kabı tek bir text dosyada birkaç

satırda tanımlanmaktadır. Bu noktada tamamen üreticiden bağımsız olunması da başlı başına bir avantajdır.



30 sayfayı 3 kere kopyala ve 90 sayfada düzeltme: 4 saat

VEYA

15 yerine 31,
31 yerine 63:
4 saniye

```
entity MULT is
port(A,B:in std_logic(15 downto 0);
        Y:out std_logic(31 downto 0));
end MULT;

architecture BEHAVE of MULT is
begin
      Y <= A * B;
end BEHAVE;
```

(31 → 63)

Şekil 2.5 32x32 çarpma devresine dönüşüm

Şekil 2.5' te mevcut 16x16' lık tasarımın 32x32' lik hale getirilmesi gösterişlmiştir. Şematik tasarımda mevcxut sistemin üç defa kopyalanması ve bunlar üzerinde oldukça fazla düzenleme yapılması gerekirken, HDL tasarımda tanımladığımız lojik vektör sinyallerinin genişliklerini girişler için 15' ten 31'e, çıkış için ise 31' den 63' e çıkarıp tekrar derleme yapılması yeterlidir. Bu işlem derleme aşaması dahil en fazla 2 dk. sürmektedir.

3. BİLGİSAYAR MİMARİSİ TEMELLERİ

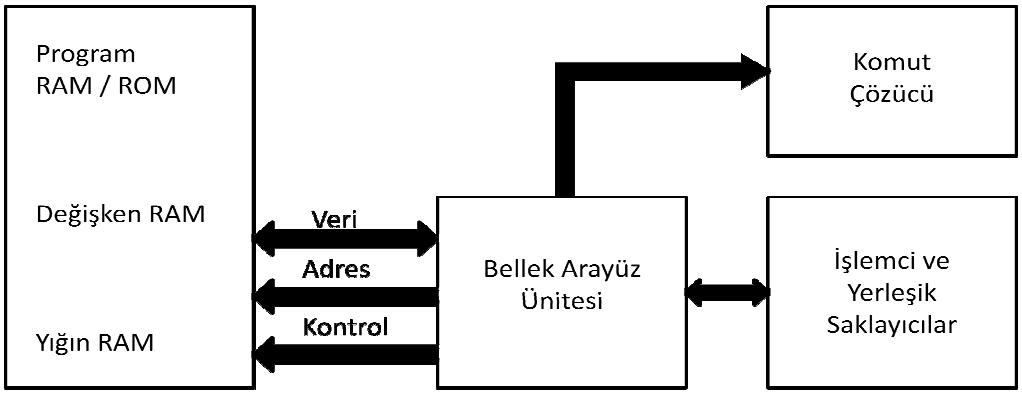
Bu bölümde, literatürde ortaya konulan en temel bilgisayar mimarileri ve bunların sınıflandırılması anlatılacaktır.

3.1 Bilgisayar Mimarilerinin Sınıflandırılması

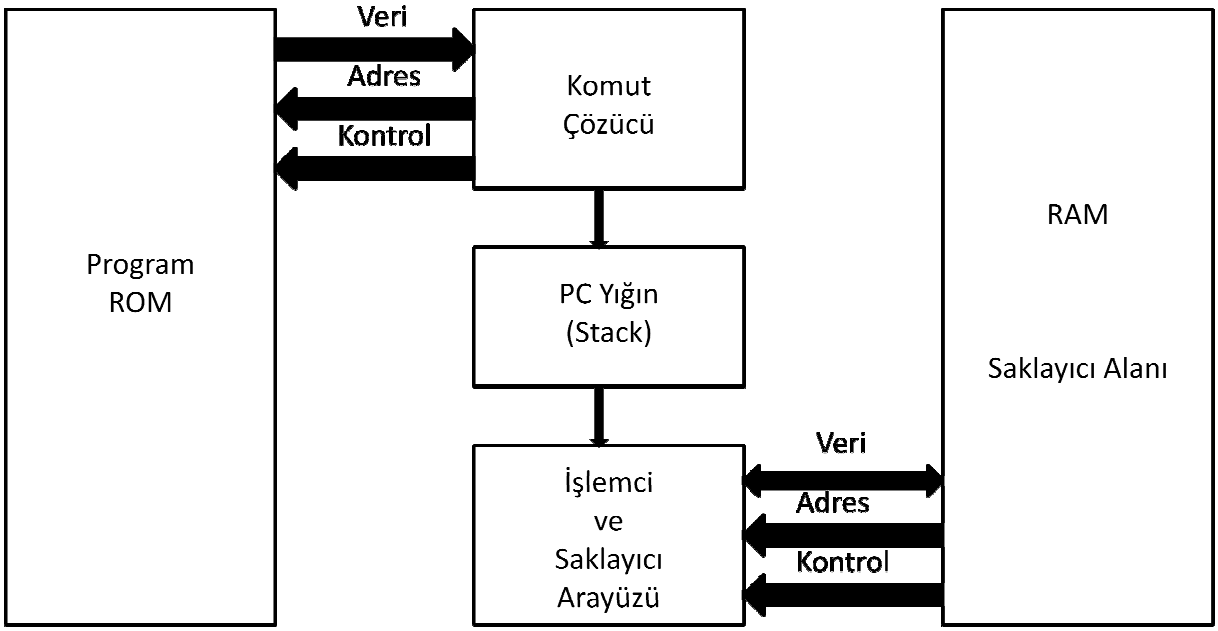
Bilgisayar mimarileri bellek organizasyon açısından ve komut işleme tekniği açısından olmak üzere iki ayrı sınıflandırmaya tabi tutulmuştur. Bu mimariler ile detaylı bilgi alt bölümlerde verilecektir.

3.1.1 Bellek Organizasyonu Açısından Mimari Yapılar

Mikroişlemciler bellek kullanımı açısından Von Neumann ve Harvard olarak adlandırılan iki mimariden biri üzerine tasarlanırlar. Von Neumann mimarisi Princeton Üniversitesi tarafından diğeri de adından da anlaşılacağı üzere Harvard Üniversitesi tarafından tasarlanmıştır. O günkü teknolojiye uygun olan Von Neumann mimarisi tercih edilse de ileriki yıllarda teknolojinin uygun hale gelmesi sonucu Harvard mimarisi 1970' li yılların sonunda özellikle mikrodenetleyici tasarımında standart hale gelmiştir. Bu iki mimariyi ayıran en önemli özellik birinde (Von Nuemann) program ve veri belleğinin aynı yerde olması, diğerinde (Harvard) ise program ve veri belleğinin ayrı yerlerde konumlandırılmış olmasıdır. Şekil 3.1 ve Şekil 3.2' de Von Nuemann ve Harvard mimarileri blok diyagram olarak verilmiştir.



Şekil 3.1 Von Neumann mimarisi



Şekil 3.2 Harvard mimarisi

3.1.2 Komut İşleme Tekniği Açısından Mimari Yapılar

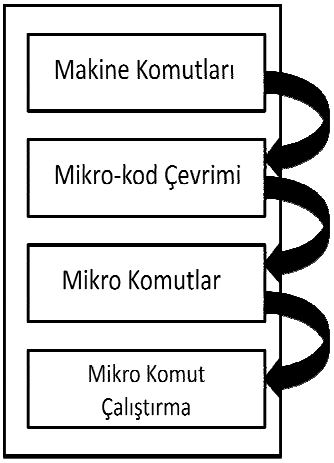
Mikroişlemciler genellikle komut işleme tekniği açısından iki grup mimari altında sınıflandırılır:

CISC: Karmaşık Komut Kümeli Bilgisayar (Complex Instruction Set Computer)
RISC: Azatılmış Komut Kümeli Bilgisayar (Reduced Instruction Set Computer)
Alt bölümlerde ilgili mimarilere ait detaylı bilgi verilecektir.

3.1.2.1 CISC Çekirdekler

Bu mimaride mikroişlemci çok sayıda komut içerir ve her eylem için bir komut tanımlanmıştır. Buradaki yaklaşım " *donanım her zaman yazılımdan hızlıdır*. " gerçeğidir. CISC, karmaşık komut kümeli bilgisayar anlamına gelmektedir. Böylece yüzlerce komutun arasından seçilen komutlarla yazılan bir program daha kısa olabilmektedir. Her işlem için farklı bir komut kullanmak işlemleri hızlandırır; ancak donanımın yükünü arttırır (yani tüm devre boyutu ve güç grekesinimi artar). CISC mimarisinde oldukça çeşitli olan komutları çalıştırmak için mikro-kod kullanılmaktadır. Farklı uzunlukta olan bu komutların çözümünde

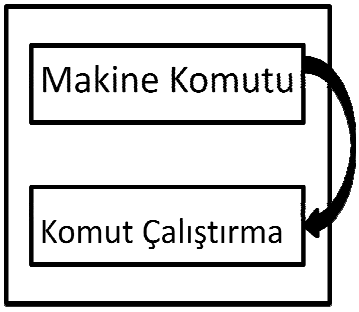
oldukça karmaşık devrelere (kod çözücülere) ihtiyaç vardır. Şekil 3.3 CISC çekirdeklerdeki çalışma akışını göstermektedir.



Şekil 3.3 CISC çekirdek çalışma akışı

3.1.2.2 RISC Çekirdekler

Hemen hemen hiç kimse bir CISC makinedeki karmaşık assembly dili komutlarının tamamını kullanmamaktadır. Günümüzde programcılar tüm karmaşık komut kümelerini neredeyse hiç kullanmayan yüksek düzeyli dil derleyicilerini tercih etmektedirler. Bu yüzden daha az, basit ve hızlı komutlar, uzun, karmaşık ve daha yavaş CISC komutlarından verimlidir. RISC mimari, daha basit komutlar kullanarak tüm devre karmaşıklığını azaltmaktadır. Ancak RISC komutlarının daha kısa olması belirli bir görevin tamamlanabilmesi için daha fazla komuta gereksinim duyulmasına yol açabilir. Ayrıca RISC mimariler için üretilen derleyiciler daha once CISC mimarisinde bulunan donanım birimlerinin görevini üstlenmek üzere ekstra komutlar kullanmaktadır. Şekil 3.4' te RISC mimarisi çalışma akışı gösterilmektedir.



Şekil 3.4 RISC çekirdek çalışma akışı

4. TASARLANAN MİKROİŞLEMCİNİN DONANIMI VE YAZILIMI

Bu bölümde tasarlanan ve HSEO16 olarak isimlendirilen mikroişlemcinin özelliklerinden bahsedilecektir. Basit bir Von-Neumann bilgisayar mimarisine sahip olan HSEO16, mimari karakteristiklerine uygun şekilde üç ana birimden oluşur. HSEO16 bir tek blok ana bellek birimi üzerinde işlem yapar. Bölüm 3.1.2.2' de de anlatılmış olan RISC mimarisi özelliklerinin hemen hemen hepsini bünyesinde barındırır.

İlk olarak ISA (Instruction Set Architecture) yani komut seti yapısından bahsedilecektir. Çoğu mikroişlemci tanıtımlarında öncelikle mikroişlemcinin neler yapabildiği üzerinde durulur ki bu da aslında ISA' nın içeriğiyle belirlenir. Birçok tasarımcı komut setlerini tasarlarken mevcut mikroişlemcilerin komut setlerini direkt olarak kullanır. Bu onlara mevcut yazılım geliştirme ortamlarını aynen kullanabilme imkanı verir. HSEO16' nın komut seti ise kendine özeldir. Bu yüzdendir ki çalışmada yazılım geliştirme amaçlı ayrı bir ortam tasarlanarak sunulmuştur. Bu bölümün daha sonraki aşamalarında tasarlanan mikroişlemcinin bütün alt elemanları ve birimleri ayrıntılı şekilde anlatılacaktır.

4.1 Komut Yapısı

Komut yapısı bir mikroişlemciyi makina dilinde programlama perspektifiyle tanımlar, aşağıdaki yapı ve işlemsellerden oluşur.

1. Komut Seti
2. Kayıtçı Dosyası
3. Adresleme Modları
4. Veri Tipleri ve Veri Tanımlamaları
5. Gerçek zamanlı operasyonlar

4.1.1 Komut Seti

İyi bir komut seti tasarlamak bunu yapmanın sistematik bir yolu olmadığından aslında oldukça zor bir iştir. Tasarımcılar bu konuya iteratif olarak eğilse de bir komut setinin sahip olması gereken bazı temel unsurlar vardır. Bunlar, bütünlük, ortogonallik, geriye dönük uygunluk ve genişleyebilirliktir.

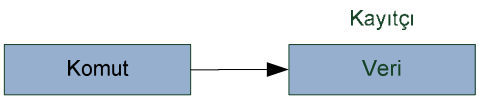
1. Bütünlük: Komut seti mikroişlemci özelliklerinin tümünün kullanabileceği şekilde en az bir komut içermelidir.
2. Ortogonallik: Komut seti operasyonel olarak birbiriyle hemen hemen aynı işi yapan, görev bakımından benzeşen komutlar içermemelidir.
3. Geriye dönük uygunluk: Komut Seti bulunduğu mikroişlemci ailesinin mevcut mikroişlemciden daha öncekilerinin komut setlerini içermelidir.
4. Genişleyebilirlik: Komut seti gerektiğinde adresleme bakımından genişletilebilir komutlardan oluşmalıdır. HSEO16' nın tasarımında geriye dönük uygunluk ve genişleyebilirlik bir miktar göz ardı edilmiştir. Çünkü HSEO16 herhangi bir mikroişlemci ailesine dahil olmayan ve adresleme alanları daha sonra sanal bellek ya da ön belleğe alma gibi değişik metodlarla değiştirebilir durumdadır. HSEO16 tüm özelliklerini kullanıcısına sunar şekilde bir komut setine sahiptir ve komutlar arasında herhangi bir benzeşme yoktur. Bu yönüyle bütünlük ve ortogonal olma özelliklerini bünyesinde barındırır. HSEO16 komut seti ile ilgili daha detaylı bilgi Bölüm 4' te verilecektir.

4.1.2 Adresleme Modları

Adresleme modları bir mikroişlemcide veriye erişme biçimlerini belirler. HSEO16 dört tip adresleme modunu destekler.

4.1.2.1 Direkt Adresleme

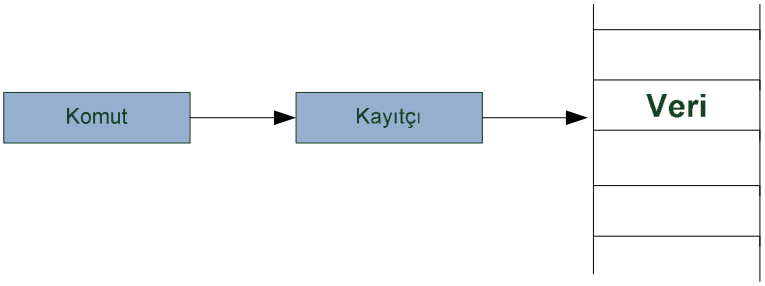
Veri akümülatörde ya da kayıtçı dosyasındaki bir kayıtçıda tutulur.



Şekil 4.1 Direkt adresleme modu

4.1.2.2 Endirekt Adresleme

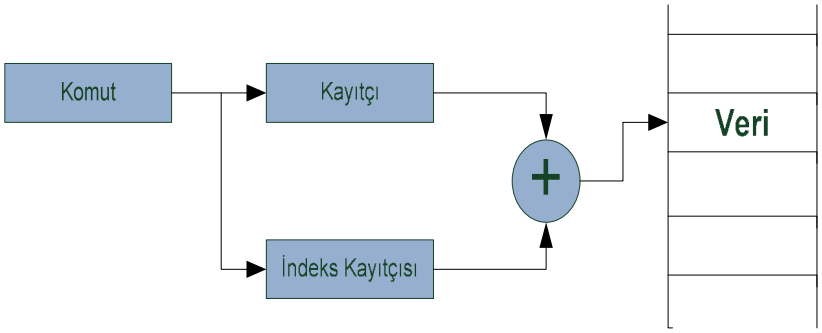
Kullanılacak verinin bulunduğu adres kayıtçı dosyasındaki bir kayıtçıda tutulur.



Şekil 4.2 Endirekt adresleme modu

4.1.2.3 İndeksli Adresleme

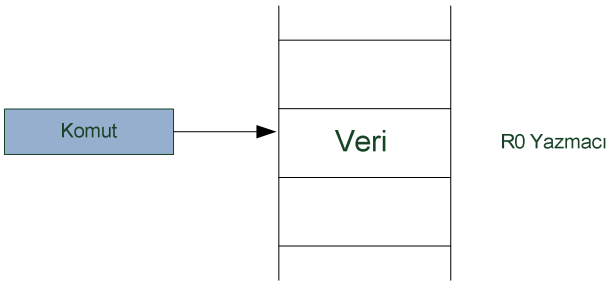
Endirekt adreslemeye çok benzer fakat kullanılacak verinin yeri bir genel amaçlı kayıtçı ile bir indeks kayıtçısının içeriğinin toplamı ile elde edilir.



Şekil 4.3 İndeksli adresleme modu

4.1.2.4 Hemen Adresleme

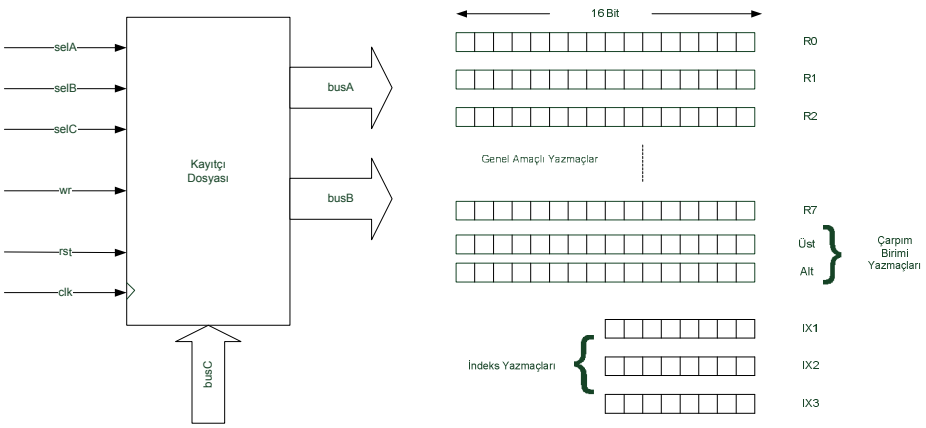
HSEO16 mikroişlemcisinde hemen (Immediate) adresleme alınan verinin her zaman daha önceden belirlenmiş olan bir adresleme bölgesine konması şeklinde gerçeklenmiştir. Bahsi geçen bölge tasarımda R0 yazmaçıdır. Hemen adresleme yönteminde komuttan hemen sonra operand olarak mikroişlemcinin bildiği bir veri tipi girilmelidir.



Şekil 4.4 Hemen adresleme modu

4.1.3 Kayıtçı Dosyası

Kayıtçı dosyası genellikle mikroişlemci içinde tasarlanan küçük ve hızlı bir ara katman depolama aracıdır. Kayıtçı dosyasındaki veriler aritmetik işlem birimi ve kontrol birimi için çok önemlidir. Kayıtçı dosyasının büyüklüğü ve fonksiyonları önemli tasarım parametreleridir. HSEO16 kayıtçı dosyası, 8 adet genel amaçlı kayıtçı, 2 adet çarpım birimi kayıtçısı ve 3 adet indeks kayıtçısından oluşur (Şekil 4.5).



Şekil 4.5 HSEO16 kayıtçı dosyası

4.1.3.1 Genel Amaçlı Yazmaçlar

Genel amaçlı yazmaçlar hesaplama işlemleri için gerekli operandları ve sonuçları kaydetmek için kullanılır. HSEO16 göreceli olarak çok fazla kayıtçı içermemektedir. Bu bize iki avantaj sağlar. Birincisi, görev geçişlerindeki işlem hızı, ikincisi tasarımın yapıldığı silikon alandan yer tasarrufudur.

4.1.3.2 İndeks Yazmaçları

İndeks yazmaçları adından da anlaşılacağı üzere indeksli adresleme yöntemi için tasarlanan bir yazmaçtır. İndeks verisini tutmakla görevlidir.

4.1.3.3 Çarpım Birimi Yazmaçları

HSEO16 mikroişlemcisinde çarpma işlemi (işaretli ve işaretsiz) ALU kullanılmadan çarpım birimi denilen özel bir modül tarafından yapılır. 16x16 çarpma işlemi yapabilen bu modül 32 bitlik sonuç değerinin en anlamlı 16 bitini ÜST, kalan 16 bitini ise ALT adı verilen yazmaçlara kaydeder.

4.1.4 Veri Tipleri

Bu bölümde HSEO16 işlemcisinin desteklediği veri tipleri anlatılacaktır.

4.1.4.1 İşaretli Tam Sayılar

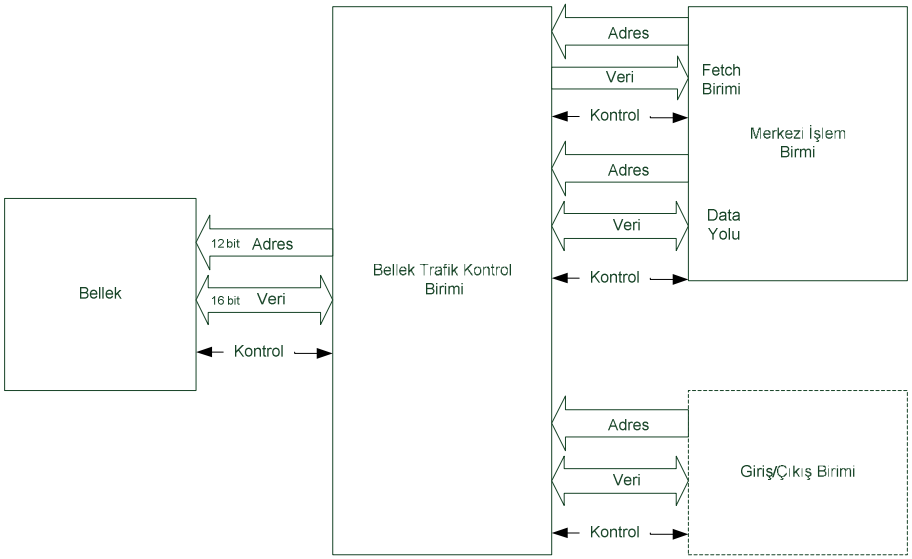
HSEO16 -32768 ile 32767 tam sayı aralığını destekler. Negatif sayılar 2' ye tümleyeni ile temsil edilir.

4.1.4.2 İşaretsiz Tam Sayılar

HSEO16 0 ile 65535 arasındaki işaretsiz tam sayıları destekler.

4.2 Tasarlanan Sistemin Mimarisi

HSEO16 Von-Neumann mimarisini karakterize eden üç adet alt sistemden oluşur. Bu alt sistemler birbirleri ile global yollar üzerinden haberleşir. Her ne kadar bellek yolları CPU ve I/O birimi tarafından paylaşılsa da Bellek Trafik Kontrol Birimi (Bölüm 4.2.3) bu paylaşımda her hangi bir çakışma olmaması adına tasarlanmıştır. Kalan yolların hiç biri paylaşımlı değildir. Daha ziyade iki birimi birbirine bağlamak amaçlı kullanılmıştır.



Şekil 4.6 Tasarlanan sistemin mimarisi

4.2.1 Merkezi İşlem Birimi

Merkezi işlem birimi (CPU) mikroişlemcinin adeta kalbi gibidir. Hesaplama işlemleri ve mimarinin geri kalanının yönetimi için gereken kontrol sinyalleri bu birimde üretilir. Bölüm 4.3 CPU' ya genel bir bakışı içermektedir.

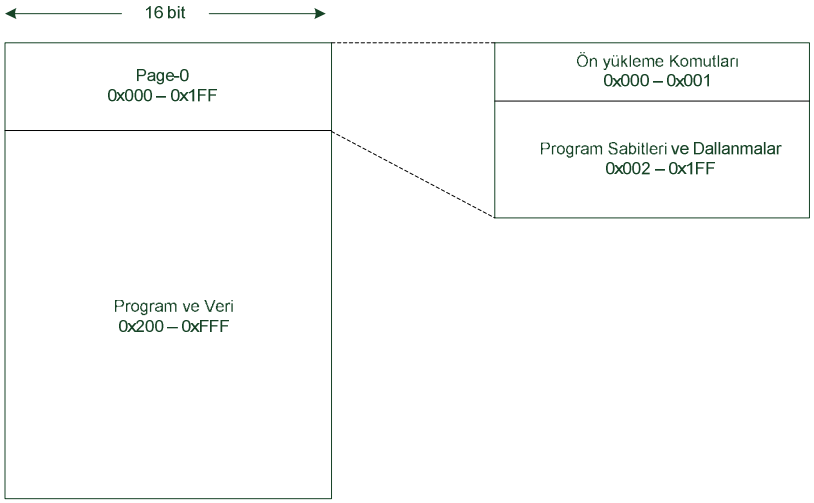
4.2.2 Ana Bellek Birimi

Yazılım tasarımı ile ilgilenen herkes ana bellek biriminin Von-Neumann mimarisinin en büyük avantajlarından biri olduğunu dile getirir. Ana bellek aynı donanım üzerinde farklı programlar koşturabilmeyi, dolayısıyla bilgisayarlarda farklı uygulamaların bir arada bulundurulabilmesi olanağını verir. Bu avantajı daha iyi anlayabilmek için herhangi bir program belleği kullanmayan klasik hesap makinalarını örnek olarak öne sürebiliriz. Hesap

makinası ile işlemler yapar ve bunların sonuçlarını elde ederiz. Fakat bir hesap makinasını devresel olarak değişikliğe uğratmadan, örneğin bir kelime işlemci olarak kullanamayız.

4.2.2.1 Bellek Organizasyonu

Von-Neumann mimarisi gereği bu çalışmada program ve veri belleği aynı blok içerisinde yer alır. Şekil 4.7 HSEO16 bellek organizasyonunu göstermektedir.



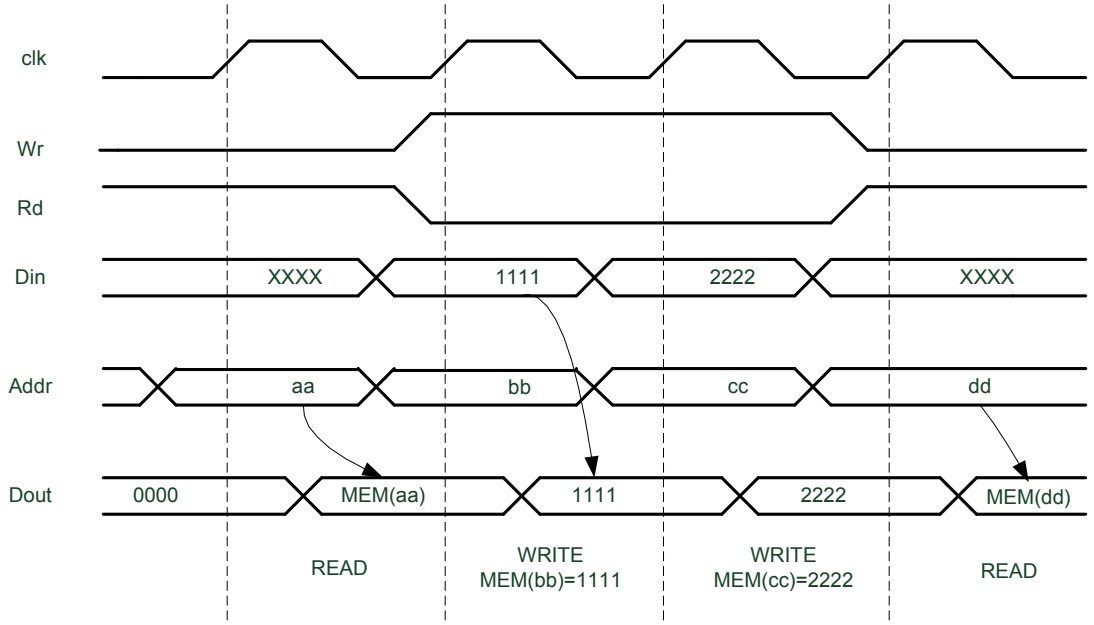
Şekil 4.7 Bellek organizasyonu

Şekil 4.7' de görüldüğü gibi HSEO16 ana bellek birimi, program komutları ve veri bölgesini içerir.

1. Ön Yükleme Komutları: İlk kelime (word) olan 0x000, 0x001' de de belirtildiği üzere ilk programın başlangıcı olan 0x200' e atlama (jump) komutunu içerir.
2. Program Sabitleri ve Dallanmalar: Bu bölge Page-0 olarak tanımlanan alanın ikinci bölümüdür. Bu bölümde tanımlanan program sabitleri ve dallanmalar için kullanılacak etiket adresleri tutulur.
3. Program ve Veri Segmenti: Bu bölüm programları makina dilindeki görünümüyle tutar. Aynı zamanda program koşarken işlenen veriler de burada tutulur.

4.2.2.2 Bellek Çevrimi

HSEO16 işlemcisinde bellek çevrimi aşağıdaki şekilde gerçekleşmektedir. Saat sinyalinin artan kenarlarında okuma (Rd) ya da yazma (Wr) sinyallerinden hangisin aktif olduğuna bakılır. Eğer okuma sinyali aktif ise bellek ilgili adresteki veriyi çıkışına verir. Eğer yazma sinyali aktif ise bellek Din (Data In) portuna gelen veriyi ilgili adresinteki bölgeye yazar. Şekil 4.8 burada anlatılanları özetlemektedir.



Şekil 4.8 Bellek çevrimi

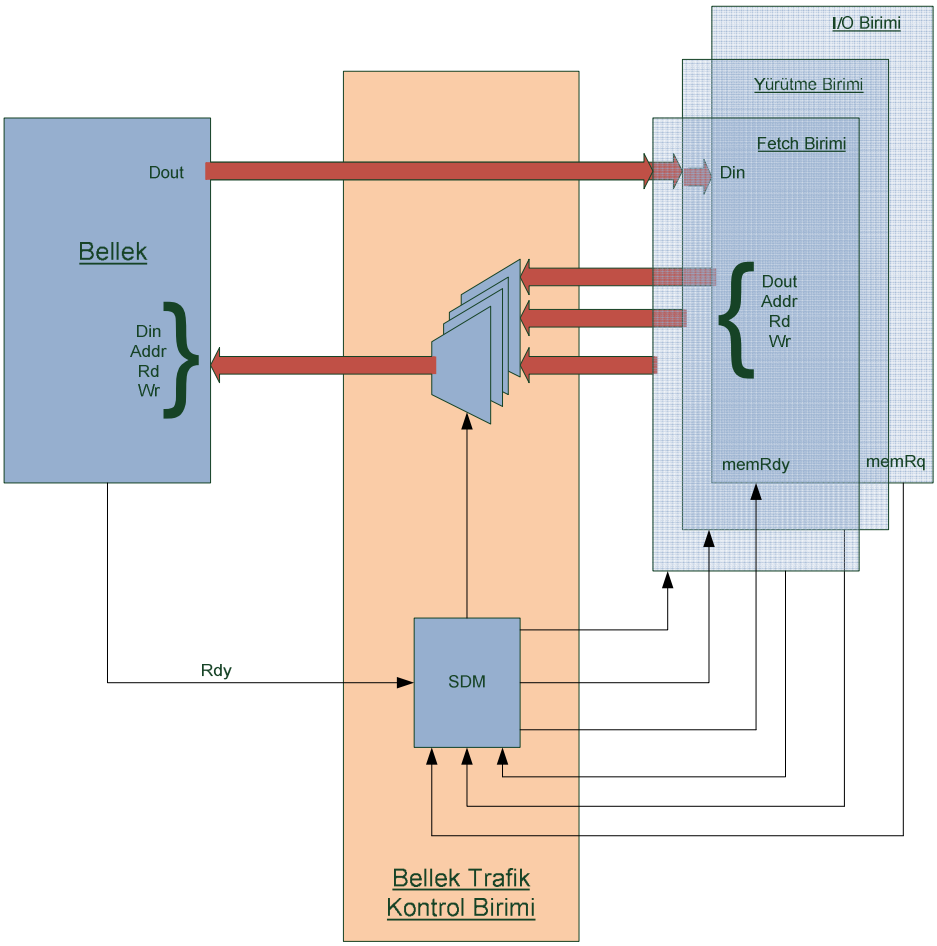
4.2.3 Bellek Trafik Kontrol Birimi

Daha önce de bahsedildiği üzere, HSEO16 Von Neumann mimarisi gereğince belleğe erişim için iki adet yol içerir. Birisi program için, diğeri ise veri içindir. Bellek biriminin iki ayrı portu olmadıkça bu iki yoldan belleğe aynı anda ulaşmak mümkün değildir. Bu erişim kontrollü bir biçimde yapmak diğer bir yöntemdir. Aslında bu yöntemde yine iki birimden aynı anda erişim olmamaktadır. Böyle bir durumda her bir birim belleğe ayrıcalıklı bir erişiminin olduğunu düşünür. HSEO16 Bellek trafik kontrol birimi üç adet ana birim ve bellek arasında bu görevi üstlenir. Bu birimler; Fetch birimi, yürütme birimi ve giriş/çıkış birimidir. Bellek trafik kontrol birimi duruma göre birimlere erişim hakkı verir. Birimlerin öncelikleri o anki aktif birimin ne olduğuna göre değişiklik arz etmektedir (Çizelge 4.1) .

Aktif Birim	Yakalama	Veri Yolu	Giriş/Çıkış Birimi	Hiç biri
1	Yakalama	Veri Yolu	Giriş/Çıkış Birimi	Yakalama
2	Veri Yolu	Yakalama	Yakalama	Veri Yolu
3	Giriş/Çıkış Birimi	Giriş/Çıkış Birimi	Veri Yolu	Giriş/Çıkış Birimi

Çizelge 4.1 Bellek birimi erişim öncelikleri

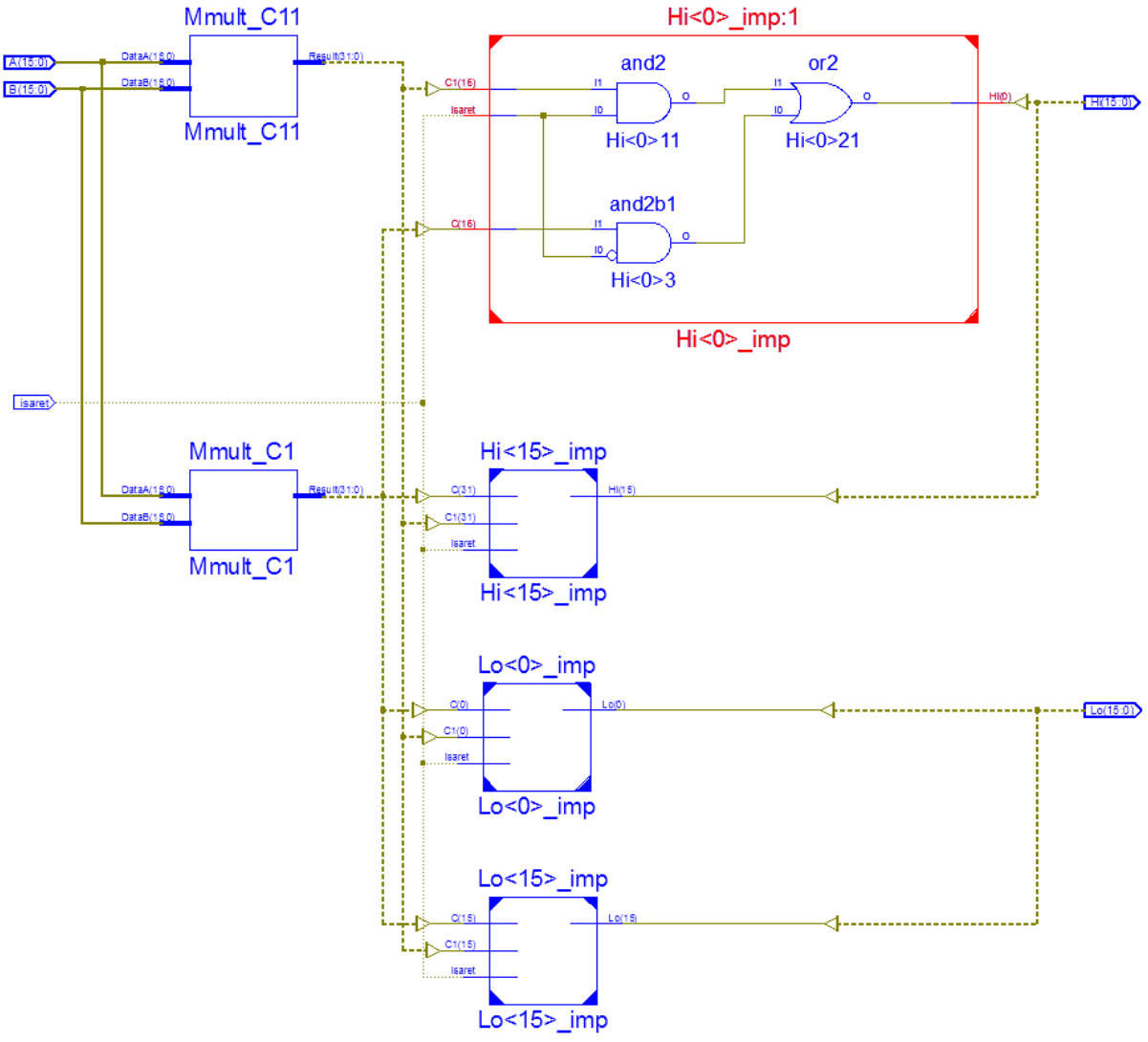
Bellek trafik kontrol birimi, bir sonlu durum makinası (SDM) tarafından kontrol edilen bir dizi çoklayıcı olarak tasarlanmıştır. SDM' nin mevcut durumu her çoklayıcı için olması gereken giriş portunu belirler. Sadece Rdy (Memory Ready) sinyali farklı bir şekilde üretilir. Bu sinyal bellek birimi tarafından üretilir ve aktif birime lojik 1 değeri gönderilir. Diğer birimlere Rdy sinyali lojik 0 olarak iletilir.



Şekil 4.9 Bellek trafik kontrol birimi

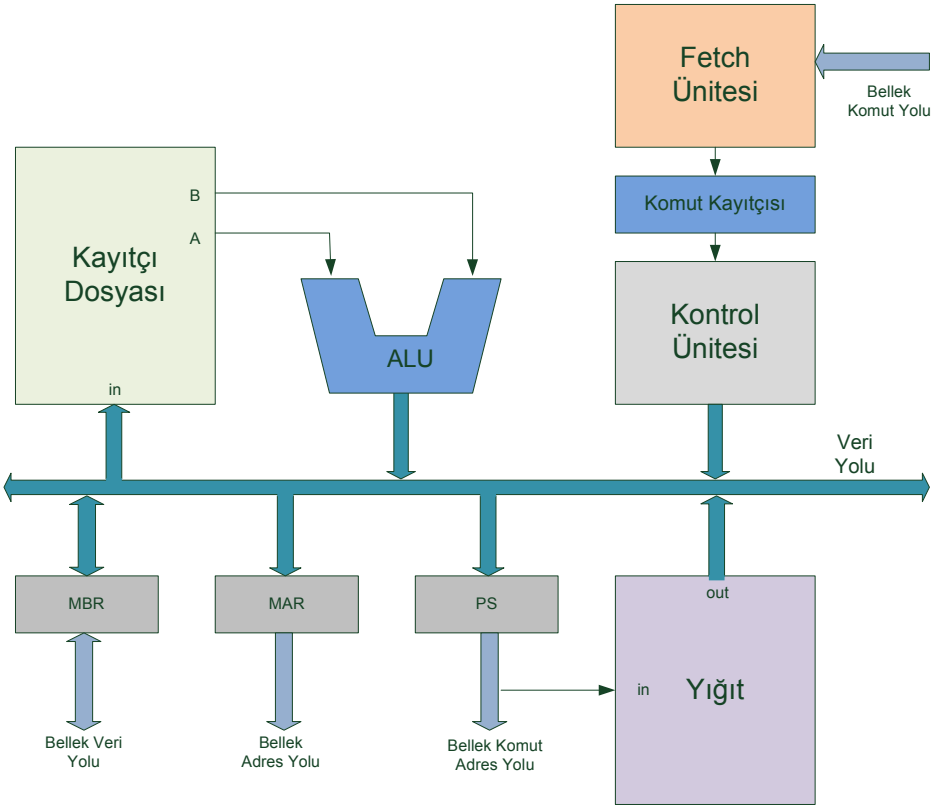
4.2.4 Çarpım Birimi

HSEO16 mikroişlemcisinde çarpma işlemi (işaretli ve işaretsiz) ALU kullanılmadan çarpım birimi denilen özel bir modül tarafından yapılır. 16x16 çarpma işlemi yapabilen bu modül 32 bitlik sonuç değerinin en anlamlı 16 bitini ÜST, kalan 16 bitini ise ALT adı verilen yazmaçlara kaydeder. Şekil 4.10' da HSEO16 çarpım birimi devre şematiği olarak gösterilmiştir.



Şekil 4.10 Çarpım Birimi

4.3 CPU Mimarisi



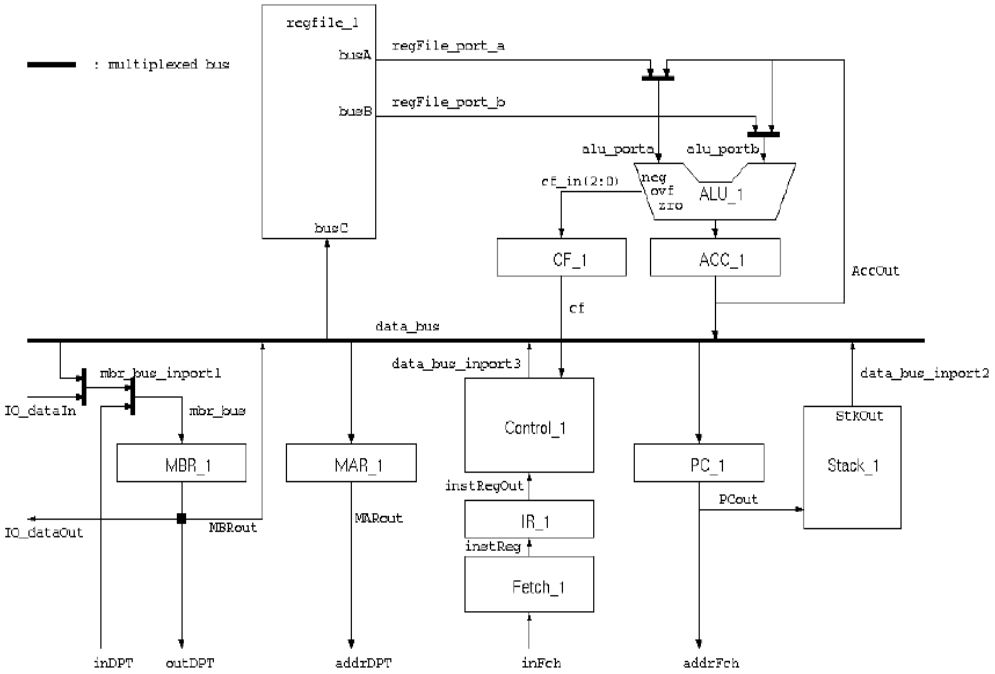
Şekil 4.11 CPU mimarisi

Şekil 4.11' de gösterildiği üzere CPU mimarisi aşağıda açıklanan bloklardan oluşmuştur.

1. Kayıtçı Dosyası: Hesaplanan anlık veriyi tutar. Bölüm 4.1.3‘ te detaylı olarak anlatılmıştır.
2. ALU: Aritmetik ve Lojik İşlem Birimi. Bölüm 6' da detaylı olarak anlatılmıştır.
3. Yakalama Ünitesi: Bellekten komutları alıp getiren ünitedir. Bölüm 7.2' de detaylı olarak anlatılmıştır.
4. Komut Kayıtçısı: O anda yürütülen komutu tutan 16 bitlik kayıtçıdır.
5. Kontrol Ünitesi: Kod Çözme, yürütme ve durum kontrol ünitelerinden oluşur. Bölüm 7' de detaylı olarak anlatılmıştır.
6. PS (Program Sayacı) : 12 bitlik ileri sayaçtır. Bir sonra yürütülecek komutun adresini belirler.

7. Yığıt: Bir alt programa girildiğinde (sadece ‘bsr’ komutuyla) program sayacının o anki değerini tutar ve alt programdan çıkıldığında bu bilgiye bakılır. Bölüm 6.5’ te detaylı olarak anlatılmıştır.
8. MAR: Memory Address Register, veri yolu tarafından ulaşılacak bellek bölgesinin adresini tutar. MAR, komutun türüne göre kayıtçı dosyasındaki bir ya da iki kayıtçı tarafından doldurulur. Bazı durumlarda ise kontrol ünitesi tarafından direkt ulaşılarak doldurulur.
9. MBR: Memory Buffer Register, belleğe yazılacak ya da bellekten okunacak veriyi tutan 16 bitlik kayıtçıdır. Aynı kayıtçı Giriş/Çıkış ünitesi ile yapılan veri transferlerinde de kullanılır.
10. Veri Yolu: 16 bitlik bir yoldur. Birçok FPGA mimarisinde üç durumlu sinyaller kullanılamadığı için bu çalışmada veri yolu çoklayıcı bloklarıyla tasarlanmıştır. Her ne kadar ismi veri yolu olarak tanımlansa da aynı yol üzerinden adres bilgisi de taşınabilmektedir.

Gerçek sinyal ve blok isimleriyle CPU mimarisi Şekil 4.12’ deki gibidir. Kontrol sinyallerine sayılarının çok fazla oluşundan dolayı bu şekilde yer verilmemiştir.



Şekil 4.12 CPU mimarisi (2)

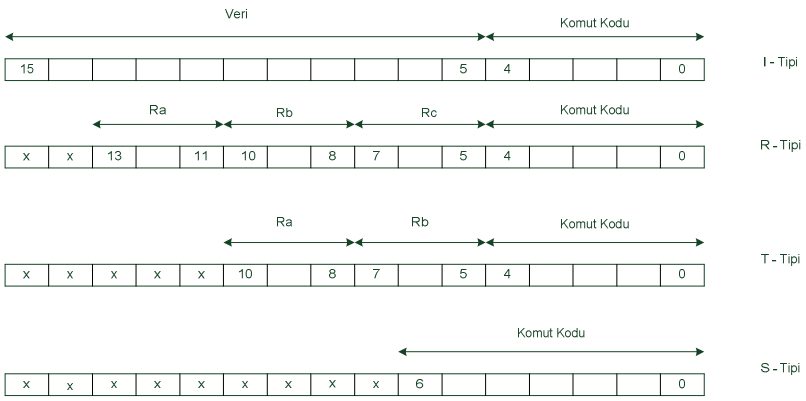
5. KOMUT SETİ

Bu bölümde HSEO16 komut seti ve işlevleri anlatılacaktır.

5.1 HSEO16 Komut Tasarımı

Sanal tasarım sırasında gerçekleştirilen ilk adım komutların belirlenmesi ve komut içindeki bitlerle komutların ve yazmaçların nasıl ifade edileceğinin belirlenmesidir. HSEO16' nın komutları 16 bit genişliğinde olup anlık değeler 11 bit ile gösterilmektedir. İşletilecek olan komut ise 16 bitlik buyruğun ilk 5 veya 7 biti ile gösterilmektedir.

HSEO16 işlemcisinde 4 tür komut bulunmaktadır. Bunlar I-tipi, R-tipi, T-tipi ve S-tipi komutlardır. I-tipi komutlar anlık değerlerle yapılan işlemleri ifade etmektedirler. 5 bit komutu bildirirken kalan 11 bit ise anlık değerdir. Anlık değerlerle işlem yapan komutlar genellikle birikeci kullanmaktadırlar. Bu tür işlemlerde yazmaçlardan alınan değerler değil, komutla gelen anlık değerler kullanılır. R-tipi komutlar yazmaçlarla işlem yapan komutlardır. İşlem iki yazmaçtan gelen değerler arasında yapılır ve bir sonuç yazmacına bulunan sonuç yazılır. Komuttaki ilk 5 bit yapılacak işlemi bildirirken sonraki 3 bit sonuç yazmacını, sonraki 6 bit ise işlemin yapılacağı değerlerin bulunduğu 2 yazmacı göstermektedir. T-tipi komutlar ise yine 5 bitlik işlem kodu ve 2 yazmaçtan oluşurlar. Komuttaki 2.yazmaç adresi üzerinde işlem yapılacak yazmacı gösterirken ilk yazmaç ise yine sonucun yazılacağı yazmaçtır. S-tipi komutlar ise sistem ile ilgili komutlardır. Yani program işleyişi ve giriş çıkış portlarına gelen değerlerin yönlendirilmesiyle ilgili komutlarda kullanılırlar. HSEO16 mikroişlemcisinin tüm komutları iki byte uzunluğundadır. Şekil 5.1 bahsi geçen komut tiplerini göstermektedir.



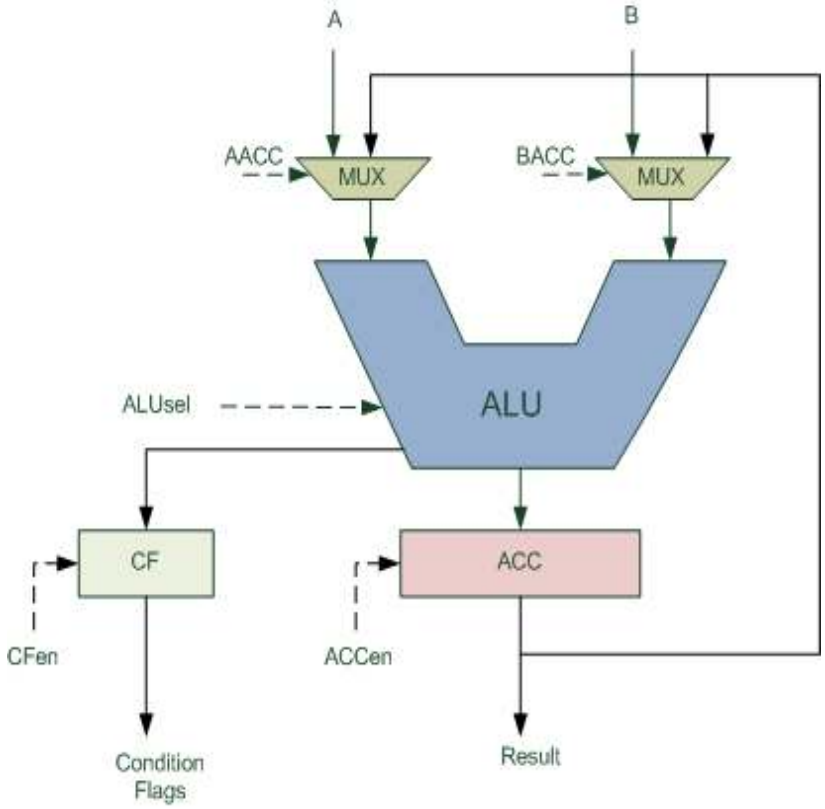
Şekil 5.1 HSEO16 komut tipleri

İşlem Kodları	Komutlar	Komut Türleri	İşlemler	Clock Sayısı
00001	add	R	Ra ←Rb + Rc	1
00010	addi	I	Birikeç ← Birikeç + Anlık Değer	1
00011	sub	R	Ra ←Rb - Rc	1
00100	subi	I	Birikeç ← Birikeç - Anlık Değer	1
00101	mul	T	Üst+Alt← Ra * Rb	2
00110	muli	I	Üst+Alt ← Birikeç * Anlık Değer	2
00111	mulu	T	Üst+Alt← Ra * Rb	2
01000	and	R	Ra ← Rb Λ Rc	1
01001	andi	I	Birikeç ← Birikeç Λ Anlık Değer	1
01010	or	R	Ra ← Rb V Rc	1
01011	ori	I	Birikeç ←Birikeç V Anlık Değer	1
01100	xor	R	Ra ← Rb (XOR) Rc	1
01101	xori	I	Birikeç ← Birikeç (XOR) Anlık Değer	1
01110	not	T	Ra ← !Rb	1
01111	sll	T	Ra ← Ra << Rb	2
10000	srl	T	Ra ← Ra >> Rb	2
10001	sla	T	Ra ← Ra <<< Rb	2
10010	sra	T	Ra ← Ra >>> Rb	2
10011	mov	T	Ra ← Rb	1
10100	movi	I	Birikeç ← Anlık Değer	1
10101	lw	T	Ra ← Bellek [Rb]	1
10110	sw	T	Bellek[Rb] ← Ra	1
10111	cmp	T	Ra>Rb ise Büyük Bayrağı = 1 Ra<Rb ise Küçük Bayrağı = 1	2
11000	beq	I	Sıfır Bayrağı = 1 ise PS=PS + İşaretle genişletilmiş anlık değer	2
11001	bne	I	Sıfır Bayrağı = 0 ise PS=PS + İşaretle genişletilmiş anlık değer	2
11010	ba	I	PS= Anlık Değer	1
11011	bsr	I	PS_ret = PS PS = Anlık Değer	1
11100	ret	I	PS= PS_ret	1
11101	bgt	I	Büyük Bayrağı = 1 ise PS=PS + İşaretle genişletilmiş anlık değer	2
11110	blt	I	Küçük Bayrağı = 1 ise PS=PS+İşaretle genişletilmiş anlık değer	2
1111101	out	S	Birikeç [7:0] dışarı verilir.	1
1111110	hlt	S	PS = PS	1
1111111	nop	S	PS ← PS+1	1

Çizelge 5.1 HSEO16 komut seti

6. Aritmetik Lojik İşlem Birimi

HSEO16 işlemci aritmetik lojik işlem birimi (ALU) kombinasyonel bir lojik çekirdek yapıdan ve işlem sonuçlarını tutabilmek için tasarlanmış bir akümülatörden oluşur.



Şekil 6.1 ALU (Aritmetik Lojik İşlem Birimi)

6.1 ALU Operandları

ALU yürütülecek işlemin türüne göre bir ya da iki operandı giriş olarak kabul eder. Bu operand ya da operandlar aşağıdakilerden biri olabilmektedir.

1. Kayıtçı dosyasındaki kayıtçılardan biri: Komuttaki Rx ifadesine denk düşer. Bu tanımlamada ’x‘ 0 ile 7 arasında bir değerdir.
2. Akümülatör: Komuttaki ACC ifadesine denk düşer.
3. Kaydırma Sayacı: Sadece kaydırma işlemleri için geçerli olan operanddır.

6.2 ALU Hesaplama Sonuçları

Daha önceden de açıklandığı üzere ALU işlem sonuçları bir değişikliğe uğratılmadan ya da yönlendirilmeden önce akümülatör denen yapıda tutulur. ALU yürüttüğü işlemin türüne ve sonucuna göre üç adet durum bayrağını değiştirebilir.

Bayrak	Anlam	Durum
NEG	Negatif	Sonuç negatif ise 1, değilse 0
OVF	Taşma	Sonuç işaretli 16 bit sayı gösterim aralığını geçmiş ise 1, geçmemiş ise 0
ZRO	Sıfır	Sonuç sıfır ise 1, değilse 0
BUY	Büyüktür	Ra > Rb ise 1, değilse 0
KUC	Küçüktür	Ra < Rb ise 1, değilse 0

Çizelge 6.1 Durum bayrakları

6.3 ALU' nun Kontrol Edilmesi

Kontrol ünitesi ALU' ya aşağıda anlatıldığı şekilde bazı kontrol sinyalleri gönderir.

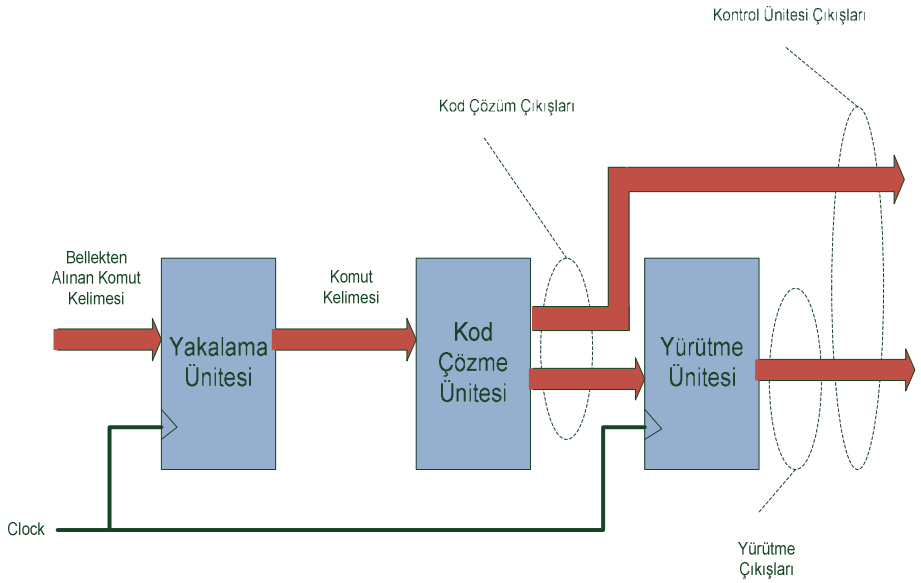
1. ALUsel (ALU Select) : Hangi operasyonun yürütüleceğini belirleyen clocktan bağımsız sinyaldir. Alabileceği değerler ALU' nun gerçekleştirebileceği operasyonlarla sınırlıdır.
2. ACCen (Acumulator Enable) : ALU operasyonu tamamlandıktan sonra akümülatörü aktif etmek için kullanılan clocka bağımlı sinyaldir.
3. CFen (Control Flags Enable) : Hesaplama işlemi bittikten sonra durum bayraklarını aktif etmek için kullanılan cloka bağlı sinyaldir.
4. AACC (A is the ACC) : Sol taraftaki operandın akümülatör mü yoksa bir kayıtçı mı olduğunu ALU' ya bildiren clocktan bağımsız sinyaldir.
5. BACC (B is the ACC) : Sağ taraftaki operandın akümülatör mü yoksa bir kayıtçı mı olduğunu ALU' ya bildiren clocktan bağımsız sinyaldir.

7. KONTROL ÜNİTESİ

Bu bölümde kontol ünitesi mimarisi, boru hattı (pipelining) yapısı ve kontrol ünitesi alt bileşenleri anlatılacaktır.

7.1 Kontrol Ünitesinin Yapısı ve Boru Hattı

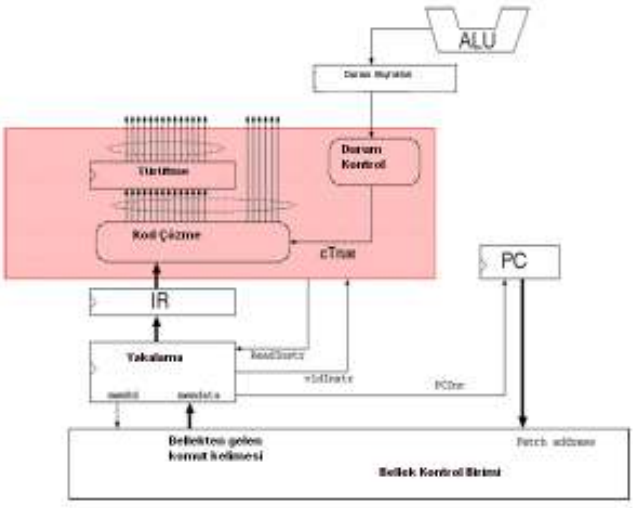
HSEO16 kontrol ünitesi üç alt üniteden oluşur. Bunlar yakalama (fetch) ünitesi, kod çözme ünitesi (decode) ve yürütme (execute) ünitesidir.



Şekil 7.1 Kontrol ünitesi alt bileşenleri

HSEO16 işlem boru hattı yapısına sahiptir. Bu yapı iki aşamadan meydana gelmiştir. Bunlar, komutu getirilmesi (fetch) , kod çözme ve yürütme aşamasıdır.

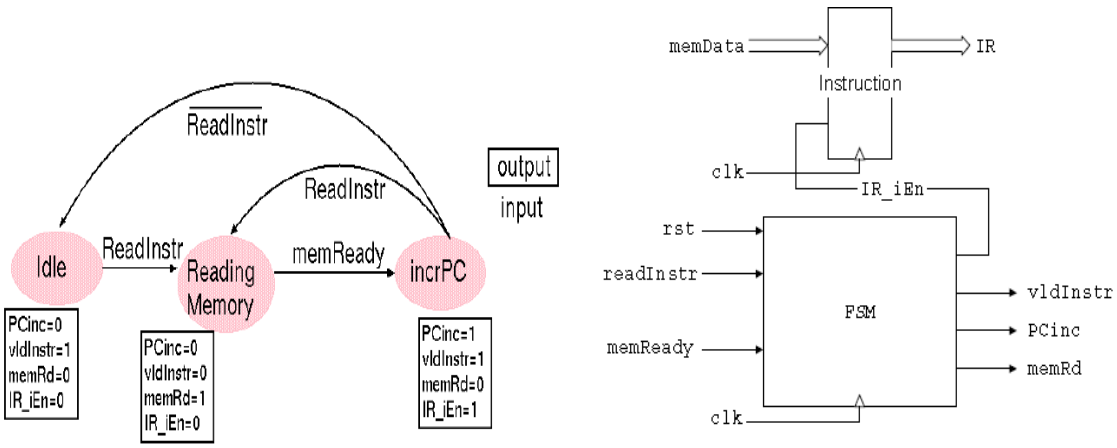
HSEO16 boru hattı yapısı sadece bir slot uzunluğundadır. Bunun anlamı verilen bir zamanda kod çözme ve yürütme işlemi yapılırken fetch ünitesi sıradaki (sadece bir sonraki) komutu getirme işlemini yapabilir. Daha uzun bir boru hattı yapısı HSEO16 mikroişlemcisi için uygun görülmemiştir. Öyle ki; uzun boru hatları önemli dallanma hatalarına sebebiyet vermektedir. İşlemci bir dallanma görevini gerçekleştirecekse öncelikle tüm boru hattının boşaltılması ve dallanılan alt programın ilk işlemi boru hattının en tepesine taşınması gerekmektedir. Bu tepeye koyma işlemi boru hattının uzunluğuyla doğru orantılı olarak oldukça fazla clock sayısında gerçekleştirilebilir.



Şekil 7.2 Kontrol ünitesi alt bileşenleri (2)

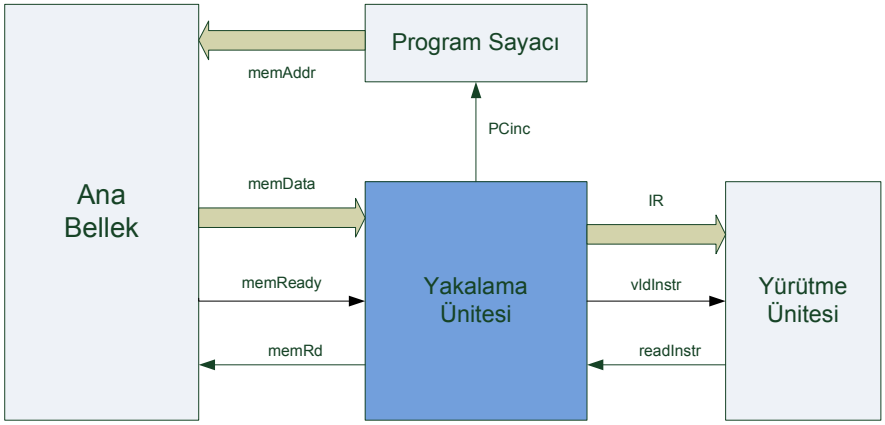
7.2 Yakalama Ünitesi

Yakalama Ünitesi (fetch unit) komutları ilgili adresten alıp getiren ünitedir. Program sayacı bir sonra işletilecek olan komutun adres bilgisini tutar ve fetch ünitesine yardımcı olur. Fetch ünitesi bu çalışmada bir sonlu durum makinası olarak tasarlanmıştır. Fetch ünitesi her hangi bir komutun işlteilmesi gerekmiyorsa durur ki; bu da komut kayıtçısının sıradaki komut bilgisini tuttuğu ancak yürütme ünitesinin hala mevcut komutu yürütmekle meşgul olduğu durumdur. Şekil 7.3 yakalama ünitesinin durum diyagramını göstermektedir. Bu diyagramda geçen sinyal adları onların bulunduğu noktada aktif (lojik 1) olduğu anlamına gelir.



Şekil 7.3 Yakalama ünitesi durum ve blok diyagramı

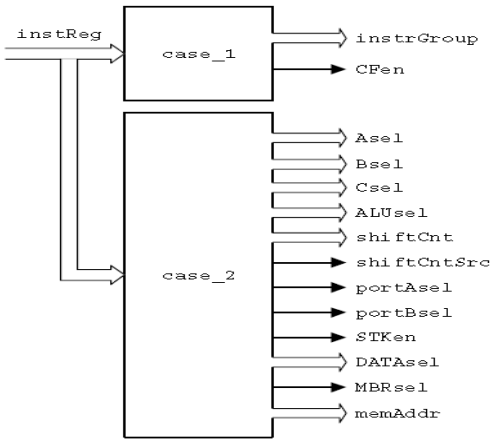
Şekil 7.4 fetch ünitesinin diğer bileşenlere olan arayüzünü ve bunlarla ilişkisini göstermektedir.



Şekil 7.4 Yakalama ünitesi arayüzü

7.3 Kod Çözme Ünitesi

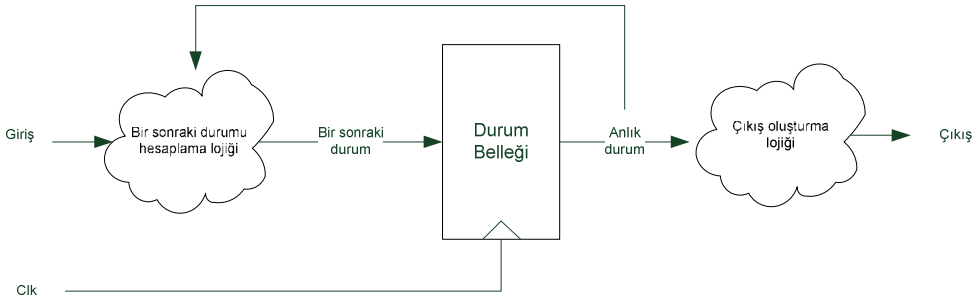
Kod çözme ünitesi kombinasyonel lojik bir devre olarak tasarlanmıştır. Yürütme ünitesine çözdüğü kod parçacığını ve clocksuz kontrol sinyallerini gönderir. Burada clocksuz olarak belirtilen sinyaller yürütme çevrimleri boyunca değişikliğe uğramazlar. Diğer sinyaller ise clocklu bir şekilde yürütme ünitesine gönderilen sinyallerdir. Yürütme ünitesi bu sinyalleri kullanarak işlemcinin başka birimlerine senkron sinyaller gönderir. Şekil 7.5 kod çözme ünitesi bloğu ve ilgili sinyalleri göstermektedir.



Şekil 7.5 Kod çözme ünitesi

7.4 Yürütme Ünitesi

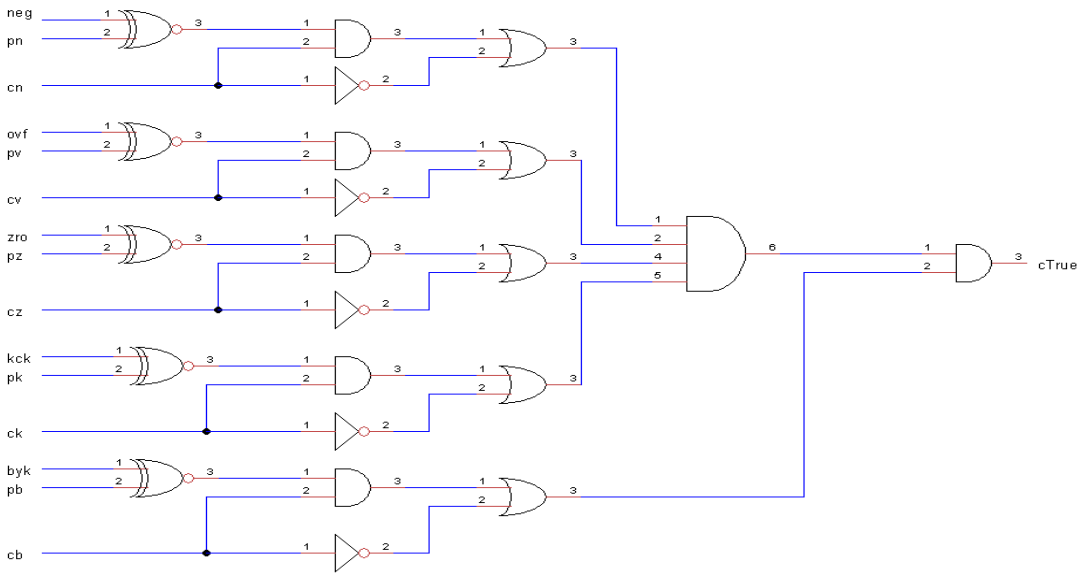
Yürütme ünitesi, kod çözme ünitesi tarafından üretilen sinyaller ile sürülür ve komutun işletiminin kontrolü adına mikroişlemcinin diğer birimlerine belli kontrol sinyalleri üretip gönderir. Üç durumlu Moore makinası mantığıyla tasarlanamıştır.



Şekil 7.6 Yürütme ünitesi

7.5 Durum Kontrol Ünitesi

Durum kontrol ünitesi, durum bayrakları kayıtçısından aldığı girişlere göre kod çözme ünitesine iletilen cTrue sinyalini üretir. Bu ünite tamamen kombinasyonel yapıda lojik bir devreden oluşur. Şekil 7.7 durum kontrol ünitesinin iç yapısını göstermektedir.



Şekil 7.7 Durum kontrol ünitesi

7.6 Donanımsal Yığıt

Donanımsal yığıt program içinde bir alt programa dallanılması gerektiğinde (sadece ‘bsr’ komutu için) program sayacının en son değerini tutar ve alt program dönüşünde (‘ret’ komutuyla) programın kaldığı yerden devam etmesini sağlar. Kullanıcının buraya erişimi yoktur.

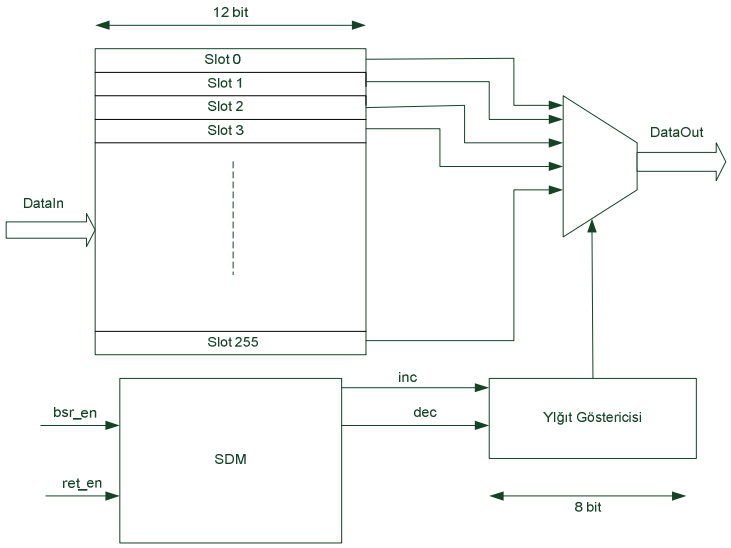
HSEO16 donanımsal yığıtı 256 adet 12-bitlik slot ve 1 adet 8-bit aşağı-yukarı sayıcıdan oluşur. Tasarım basit bir sonlu durum makinasi kullanılarak yapılmıştır. HSEO16 donanımsal yığıtının işlemlerinin aşağıda da detaylandırıldığı üzere klasik yığıt işlemlerinden farkı yoktur.

7.6.1 Donanımsal Yığıta Veri Yerleştirmek

İlk clock darbesinde giriş verisi yığıt göstericisi tarafından gösterilen slot bölgesine itilir. İkinci clock darbesinde ise sayaç değeri bir arttırılır.

7.6.2 Donanımsal Yığıttan Veri Çekmek

İlk clock darbesinde önce sayaç değeri bir azaltılır. İkinci clock darbesinde ise yığıt göstericisi tarafından gösterilen slottan veri çekilir ve kaydedilir.



Şekil 7.8 Donanımsal yığıt

8. UNİVERSAL ASENKRON ALICI ve VERİCİ ARAYÜZÜ

HSEO16' nın giriş çıkış fonksiyonlarını gerçeklemek için sisteme bir Universal Asenkron Alıcı / Verici (UART) arayüzü eklenmiştir. Seri iletişimi sağlayan ve VHDL tasarımı oldukça karmaşık olan bu komponent OpenCores web sitesinden (www.opencores.org) hazır modül olarak indirilmiş ve yazılımsal olarak mevcut sisteme uygun hale getirilmiştir. Buna ilave olarak HSEO16 I/O tasarımı interrupt (kesme) sinyallerini desteklemediğinden orjinal UART tasarımından bu sinyaller de kaldırılmıştır.

8.1 UART Fonksiyonları

UART aşağıdaki çizelgede görülen üç görevi yerine getirebilmektedir. Dolayısıyla bu VHDL tasarımına üç ayrı modül olarak yansımıştır.

	Fonksiyon	Modül
1	Alıcı	Rx
2	Verici	Tx
3	Durum belirtme	Status

Çizelge 8.1 UART fonksiyonları

8.2 UART Arayüzü

UART HSEO16' ya şekil 8.1' de görüldüğü şekilde bağlanır. Şekildeki sinyaller aşağıda açıklanmıştır.

ExtRd ve ExtWr: I/O Ünitesi okuma ve yazma sinyalleri (Aktif "1") .

nRd ve nWr: UART okuma ve yazma sinyalleri (Aktif "0") .

ExtDin(0)..(2) : I/O Ünitesi veri giriş yolu (çoklayıcıya bağlı) .

Dout: UART veri çıkışı.

ExtDout: I/O Ünitesi çıkış yolu.

Din: UART veri giriş yolu.

cs(0)..(2) : I/O Ünitesi chip select sinyali.

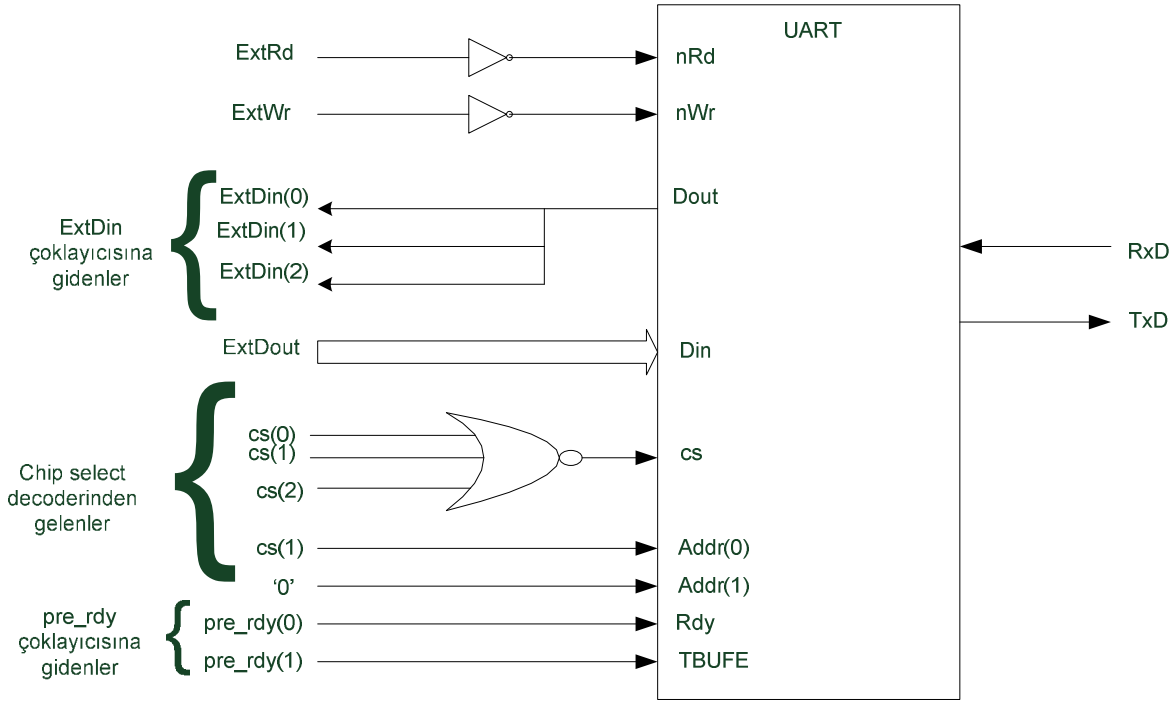
cs: UART chip select sinyali

addr(0) ve addr(1) : UART adres girişi.

pre_rdy(0) ve pre_rdy(1) : I/O Ünitesi hazırlık durumu öncesi sinyali.

Rdy: UART alıcı hazır sinyali.

TBUFE: UART verici tamponun boş oldğunu ve iletim için hazır olduğunu belirtir.



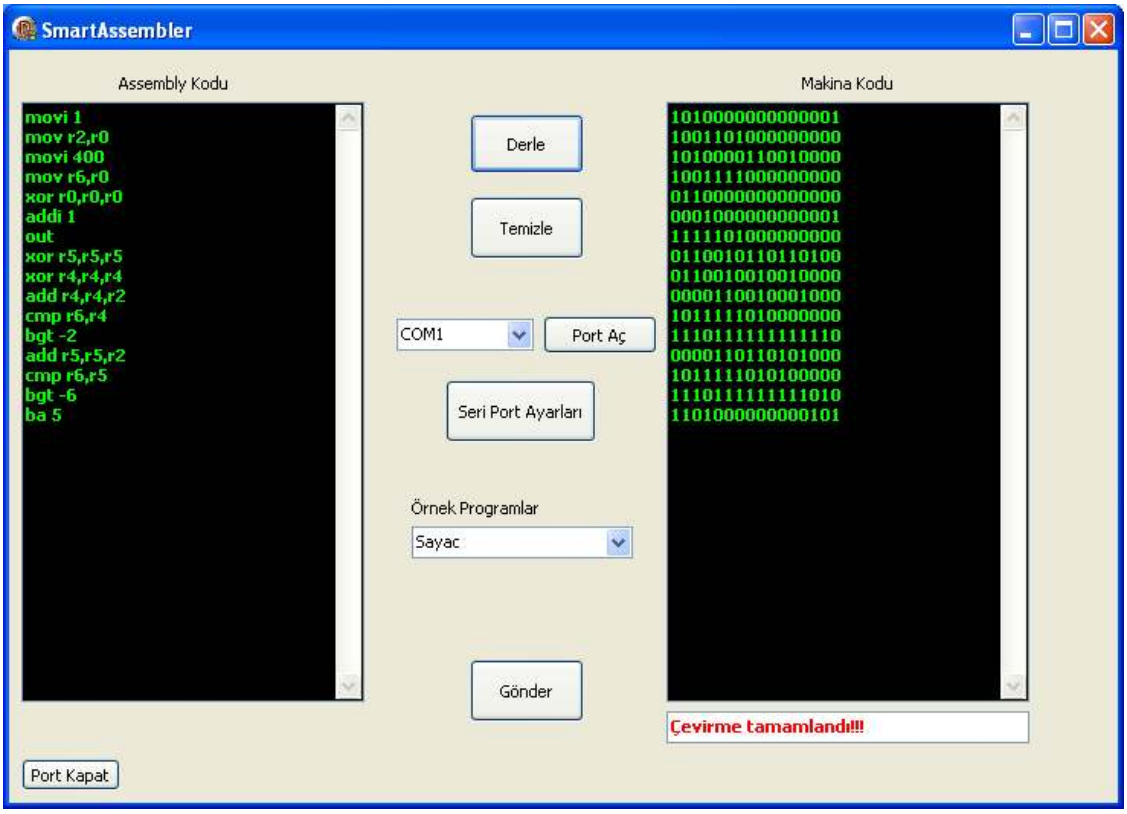
Şekil 8.1 UART arayüzü

9. HSEO16' nın PROGRAMLANMASI

HSEO16 UART üzerinden seri olarak programlanır. Bunun için kullanıcıya yazılım geliştirme imkanı sağlayan Smart Assembler programı geliştirilmiştir. Smart Assembler üzerinde makina diline çevrilen bit dizileri UART üzerinden HSEO16' nın RAM' ine yazılır ve program sayacı sıfırlanarak yüklenen programın çalışması sağlanır.

9.1 Smart Assembler Yazılım Geliştirme Arayüzü

Smart Assembler, Delphi programı ile C++ dili kullanılarak yazılmış bir arayüz progamıdır. Arayüz temel olarak, yazılan mnemonic kodları HSEO16' nın anlayacağı şekilde bit dizilerine dönüştürüp seri port haberleşmesini kullanarak yollar. Program PC üzerinde çalıştığı için burada bahsi geçen seri port haberleşmesi PC ile FPGA geliştirme kiti arasındaki haberleşmedir. Smart Assembler arayüz programının ana hatlarıyla görünüşü Şekil 9.1' deki gibidir.



Şekil 9.1 Smart Assembler arayüzü

9.1.1 Smart Assembler Arayüz Bileşenleri

Assembly Kodu Alanı: Bu alan kullanıcının program parçacığını yazdığı alandır. Büyük - küçük harflere, boşluklara duyarlıdır. Herhangi bir komutun operandları birden fazla ise bu operandlar birbirinden virgül ila ayrılmalıdır.

Makina Kodu Alanı: Bu alan assembly kodu alanına girilen kodun makina diline çevrilmiş halini gösteren alandır.

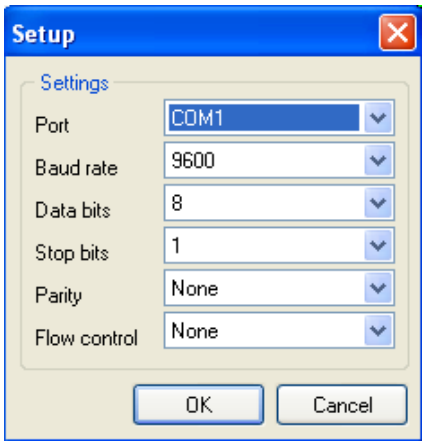
Derle Butonu: Derle butonuna basıldığında assembly kodu alanına girilen komutlar makina dilinde derlenerek makina kodu alanına 16 bitlik diziler halinde basılır.

Temizle Butonu: Temizle butonuna basıldığında tüm kod ekranı temizlenir.

Seri Port Menüsü: Bu menüden bağlı bulunan PC' nin ilişkili portu seçilir.

Port Aç Butonu: Bu butona basıldığında seri port menüsünden seçilen PC portu ile FPGA kitinin seri arayüzü arasında iletişim kanalı kurulur.

Seri Port Ayarları Butonu: Bu butona basıldığında Şekil 9.2' deki gibi seri port ayar menüsü ekrana gelir. Burada ilgili PC portu seçildikten sonraki değerler hatasız bir seri iletişim için yine Şekil 9.2' deki gibi olmalıdır.



Şekil 9.2 Seri port ayar ekranı

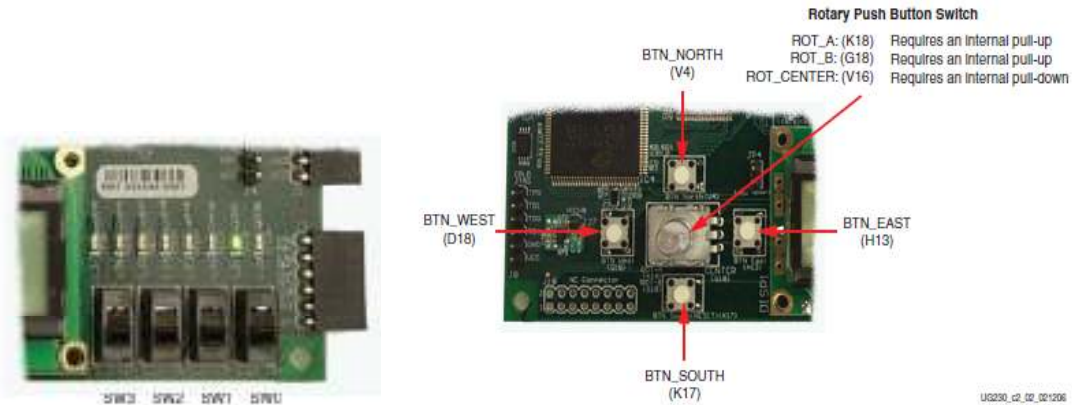
Gönder Butonu: Gönder butonuna basıldığında çevirme işlemi tamamlandıktan sonra elde edilen bit dizileri seri port üzerinden FPGA kitine aktarılır. Gönderme işlemi tamamlandıktan sonra tasarladığımız UART bloğu bu datayı alarak ilgili RAM bölgelerine yazar.

Port Kapat Butonu: Port kapat butonuna basıldığında seçili PC COM portu ile FPGA kiti arasındaki iletişim kanalı kesilir.

Örnek Programlar Menüsü: Bu menüde çalışırlığı test edilmiş dört adet program bulunmaktadır. Kullanıcı bu menüden istediği programı işaretleyip hızlı bir şekilde yükleme ve çalıştırma işlemi gerçekleştirebilir.

9.2 Yüklenen Programın Koşturulması ve Gözlemlenmesi

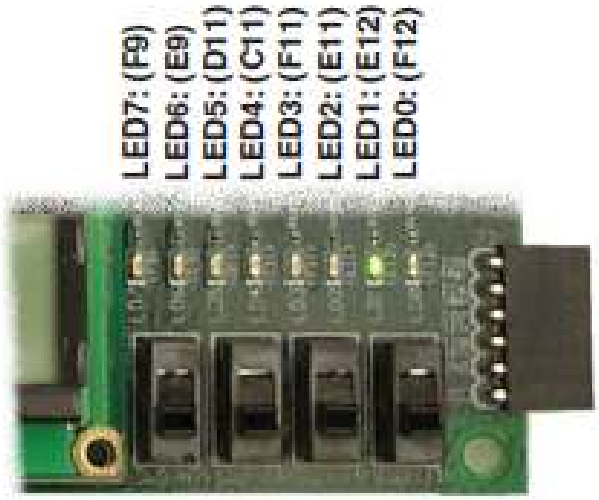
Arayüzden alınan bit dizilerinin ilgili RAM bölgelerine yazılması ve program sayacının sıfıra set edilmesinden sonra HSEO16 programı koşturmak için kullanıcıdan bir tetikleme bekler. FPGA kiti üzerindeki SW3 (switch 3) lojik 1' e çekildiğinde program çalışmaya başlar. Anahtar lojik 1 değerinde iken program sayacı sıfırlanmak isterse bu FPGA kiti üzerindeki H13 butonu ile yapılır. SW3' ü kapatıp tekrar açarak da bu işlem gerçekleştirilebilir.



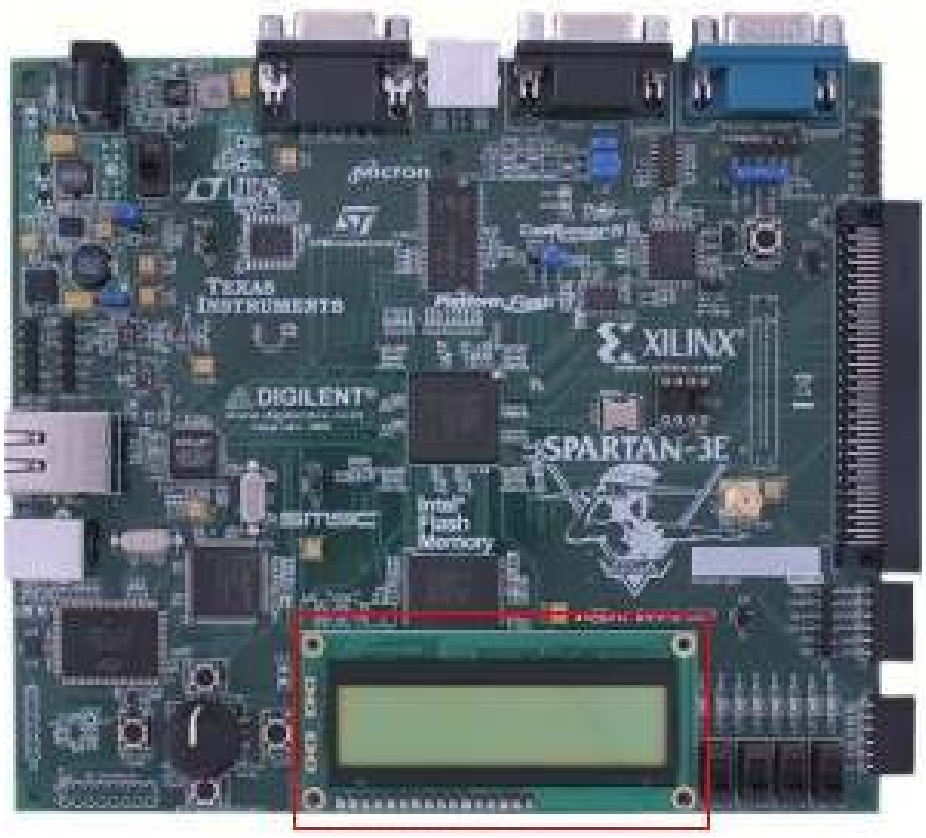
(a) (b)

Şekil 9.3 (a) Spartan 3E anahtarları (b) Spartan 3E push buttonları

Tasarımda koşan bir programın gözlemlenmesi iki şekilde olabilmektedir. Bunlardan biri FPGA kiti üzerindeki LED çıkışlar, bir diğeri ise yine kit üzerindeki LCD ekrandır.



Şekil 9.4 Spartan 3E LED çıkışları



Şekil 9.5 Spartan 3E LCD ekranı

10. ÖRNEK PROGRAMLAR

Bu bölümde mikroişlemcinin tanımlı komut seti buyruklarıyla yazılmış ve çalışırlığı test edilmiş örnek programlar anlatılacaktır.

10.1 Binary Sayaç Programı

Program:

```
movi 1              ; R0 yazmacına 1 değeri yazılır.
mov r2,r0           ; R0 yazmacındaki değer R2 yazmacına taşınır.
movi 400
mov r6,r0
xor r0,r0,r0        ; R0 özel veya R0 işleminin sonucu R0' a yazılır.
addi 1              ; R0 yazmacındaki değeri 1 değeri eklenir.
out                 ; R0 yazmacındaki değer LED çıkışlara verilir.
xor r5,r5,r5
xor r4,r4,r4
add r4,r4,r2
cmp r6,r4           ; R4 ile R6 değeri karşılaştırılır.
bgt -2              ; R6 daha büyük ise 2 komut geriye gidilir.
add r5,r5,r2
cmp r6,r5
bgt -6
ba 5                ; Koşulsuz olarak PC değeri 5 yapılır.
```

Bu program FPGA kartı LED çıkışlarına sıfırdan başlayarak binary sayaç çıkışlarını yollar ve bu değer aynı zamanda LCD ekranda görülür.

10.2 Fibonacci Sayacı Programı

Bilindiği üzere Fibonacci Serisi kendinden önceki iki elemanın toplamı serinin sıradaki elemanını verecek şekildedir.

Program:

```
mov r6,r0          ; R0 yazmacındaki değer R6 yazmacına taşınır.
movi 0             ; R0 yazmacına sıfır değeri yüklenir.
mov r3,r0
movi 1
mov r4,r0
out                ; R0 yazmacındaki değer LED çıkışlara verilir.
add r5,r5,r7       ; R7 yazmacı ile R5 yazmacındaki değer toplanır ve sonuç R5' e yazılır
add r1,r1,r7
cmp r6,r1          ; R6 ile R1 karşılaştırılır.
bgt -2             ; R6 büyük ise 2 komut geriye gidilir.
mov r1,r7          ; R7 yazmacındaki değer R1' e taşınır.
cmp r6,r5          ; R6 ile R5 karşılaştırılır.
bgt -6             ; R6 büyük ise 6 komut geriye gidilir.
mov r5,r7
add r2,r4,r3
mov r3,r4
mov r4,r2
mov r0,r2          ; Bir sonraki Fibonacci sayısı elde edildi.
out
add r5,r5,r7
add r1,r1,r7
cmp r6,r1
bgt -2
mov r1,r7
cmp r6,r5
bgt -6
mov r5,r7
ba 19
```

Bu program 1 sayısından başlayarak Fibonacci Serisi elemanlarını LED çıkışlara verir ve aynı zamanda LCD ekrana basar.

10.3 Faktoriyel Programı

```
Program:
movi 1
mov r1,r0
movi 5              ; R0 yazmacına faktoriyeli alınacak değer yüklendi.
mov r3,r0           ; Bu değer R3' e taşınır.
movi 1
mov r4,r0
mov r7,r0
bsr 13              ;  PC' a 13 değeri yüklenilir ve alt programa dallanılır.
cmp r3,r4
bgt -2
mov r0,r7
out                 ; Sonuç LED çıkışlara gönderilir.
ba 11
mov r5,r7
add r4,r4,r1
mov r6,r4
mov r7,r1
mul r5,r6           ; R5 ile R6 yazmaç değerleri çarpılır.
mov r1,r7
mov r7,r2
ret
```

Bu program, üçüncü satırda R0 yazmacına yüklenen değerin faktoriyelini alır ve bu değeri LED çıkışlara binary olarak, LCD ekrana ise decimal olarak basar.

10.4 Gel - Git Programı

```
Program:
movi 128
mov r1,r0
movi 1
```

```
mov r2,r0
movi 500
mov r6,r0
movi 0
mov r4,r0
mov r5,r0
movi 128          ; R0 yazmacına 128 değeri yüklenir.
out               ; 128 değeri LED çıkışlara gönderilir ve tüm LED ler yanar.
add r5,r5,r2
add r4,r4,r2
cmp r6,r4
bgt -2
xor r4,r4,r4
cmp r6,r5
bgt -6
xor r5,r5,r5
srl r0,r2         ; R0 yazmacındaki değer R2 yazmacındaki değer kadar sağa lojik  ötelenir.
out               ; R0 yazmacındaki değer LED çıkışlara verilir.
add r5,r5,r2
add r4,r4,r2
cmp r6,r4
bgt -2
xor r4,r4,r4
cmp r6,r5
bgt -6
xor r5,r5,r5
sub r3,r0,r2      ; R2 yazmacındaki değerden R0 yazmacındaki değer çıkarılır. Sonuç R3' e
                  ; yazılır.
beq 2
ba 10
out
add r5,r5,r2
add r4,r4,r2
cmp r6,r4
```

```
bgt -2
xor r4,r4,r4
cmp r6,r5
bgt -6
xor r5,r5,r5
sll r0,r2          ; R0 yazmacındaki değer R2 yazmacındaki değer kadar sola lojik ötelenir.
out
add r5,r5,r2
add r4,r4,r2
cmp r6,r4
bgt -2
xor r4,r4,r4
cmp r6,r5
bgt -6
xor r5,r5,r5
sub r3,r0,r1       ; R1 yazmaç değerinden R0 çıkarılır. Sonuç R3 yacmacına yazılır.
beq -42            ; Sonuç sıfır ise 42 komut geriye gidilir.
ba 32
```

Bu program sonsuz bir döngü içinde LED çıkışlarda gel-git hareketi yaptırır.

11. SONUÇLAR ve ÖNERİLER

Bu çalışmada, FPGA elemanı kullanılarak bir mikroişlemci (HSEO16) tasarlanmış ve gerçekleştirilmiştir. Tasarlanan mikroişlemci ağırlıklı Von Neumann mimarisi özelliklerini barındırmaktadır. Mikroişlemci, 16 – bit veri yoluna, 17 adet yazmaca, 8 Kilobyte kapasiteli bir bellek birimine, 33 adet komuttan oluşan bir komut setine, 65.95 MHz maksimum çalışma frekansına ve 32.975 MIPS işlem kapasitesine sahiptir. Çizelge 11.1' de HSEO16' nın temel özellikler yönünden bazı benzer çalışmalarla karşılaştırılması yapılmıştır. Bu karşılaştırma sonucunda gerçekleştirilen mikroişlemcinin tabloda belirtilen diğer çalışmalara göre özellikle çalışma frekansı, kayıtçı sayısı, komut sayısı ve MIPS değerlerine göre üstünlükleri olduğu belirlenmiştir. Mikroişlemciye erişimin kolaylaştırılması ve mikroişlemciye dışarıdan program yüklenebilmesi hedeflenerek bir de yazılım geliştirme arayüzü (Smart Assembler) tasarlanmıştır. Bu arayüz sayesinde kullanıcı tarafından işlemci komut seti kullanılarak hazırlanan programlar seri port üzerinden mikroişlemciye yüklenebilmektedir. Mikroişlemci barındırdığı FPGA mimarisine uygun komutlarıyla program çıktılarını FPGA kartı çıkışlarında gösterebilmektedir.

Çalışma	Veri Yolu (bit)	Maksimum Çalışma Frekansı	Adres Yolu (bit)	Kayıtçı Sayısı	Komut Sayısı	MIPS (Million Instructions Per Second)
HSEO16	**16**	**65,95 MHz**	**12**	**17**	**33**	**32,97**
Cpu_Kulis	16	52,68 MHz	10	6	30	17,56
DPUMikro	16	38,66 MHz	15	5	28	38,66
Kasırga	16	43,58 Mhz	11	9	38	21,79
SelCPU	32	25 MHz	32	13	77	25

Çizelge 11.1 HSEO16 ve diğer çalışmaların karşılaştırılması

Mevcut tasarımda kontrol ünitesi, giriş-çıkış ünitesi bir işlemi bitirinceye kadar pasif vaziyettedir. Bu bazı işlemler için elverişli olmamaktadır. Örneğin giriş-çıkış ünitesi büyük bir veri bloğu çıkarırken kontrol ünitesinin durması gerekmemektedir. Kontrol ünitesi geliştirilerek bir giriş-çıkış işlemi esnasında durup durmayacağına kendisinin karar vermesi sağlanabilir. Mevcut tasarım üzerinde daha fazla çevre birimi desteklemek üzere geliştirmeler yapılabilir. Örneğin; harici sistem saati üreteci, rastgele sayı üreteci, paralel arayüz, VGA arayüzü bunlardan başlıcalarıdır. Şu anda mevcut olmayan yığın (stack) bellek alanı sisteme eklenerek PUSH ve POP yığın komutları kullanıma sunulabilir.

KAYNAKLAR

Baron R.J. ve Higbie L. (1992), Computer Architecture. Addison-Wesley Publishing Company, United Kingdom.

Başak S. (2008) , SelCPU. CPU Turkey 2008 Gömülü Sistemler Yarışması.

Chang C. , Huang C. , Lin Y. , Huang Z. ve Hu T. (2005), " FPGA Platfrom for CPU Design and Applications" , Nanotechnology, 2005. 5th IEEE Conference, 187-190 vol 1.

Ergin O. (2008) , Kasırga. CPU Turkey 2008 Gömülü Sistemler Yarışması.

Ertürk S. (2008) , Cpu_kulis, CPU Turkey 2008 Gömülü Sistemler Yarışması.

Gaonkar R.S. (1999) , Microprocessor Architecture, Programming and Applications with the 8085. Prentice Hall, New Jersey.

Mano M.M. (1982) , Computer System Architecture. Prentice Hall, New Jersey.

Özcerit A. , Çakıroğlu M. , Bayılmış C. (2005) , 8051 Mikrodenetleyici Uygulamaları , İstanbul, 17-20.

Özmen A. , Güdenler İ. ve Doğan E. (2008) , DPUMikro, CPU Turkey 2008 Gömülü Sistemler Yarışması.

Uzun T. (2009) , Mikroişlemci Sistemleri , İstanbul, 89-91.

Wear L.L. , Pinkert C.R. , Wear L.C. , ve Lane W.G. (1991) , An Introduction to Hardware and Software Design. McGraw-Hill Education, United Kingdom.

İNTERNET KAYNAKLARI

www.opencores.org

ÖZGEÇMİŞ

Doğum tarihi	23.12.1985	
Doğum yeri	İstanbul	
İlköğretim	1991 - 1999	Pilot Cengiz Topel İ.Ö.O
Lise	1999 - 2003	Kabataş Erkek Lisesi
Lisans	2003 - 2007	Yıldız Teknik Üniversitesi Elektrik – Elektronik Fak. Elektronik ve Haberleşme Mühendisliği Bölümü

Çalıştığı Kurumlar

2007 – 2010	Nortel Networks Netaş A.Ş.