FPGA Kullanarak 16 Bitlik Mikroişlemci Tasarımı
Designing of a 16 – bit Microprocessor by Using FPGA

Emre ÖZTÜRK, Herman SEDEF

Elektronik ve Haberleşme Mühendisliği
Yıldız Teknik Üniversitesi
emre.ytu@gmail.com

Elektronik ve Haberleşme Mühendisliği
Yıldız Teknik Üniversitesi
sedef@yildiz.edu.tr

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 16 bitlik veri yoluna sahip bir mikroişlemciyi 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 paper introduces a 16 – bit microprocessor that adopts von-Neumann architecture and is implemented on FPGA (Field Programmable Gate Array) . In addition to that, the paper presents a software development environment for designed microprocessor.

Microprocessor 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. Giriş

FPGA günümüzde endüstriyel alanda kullanımı gün geçtikçe artan bir elektriksel elemandır. 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. Bu yöntem ile tasarımlar neredeyse maliyetsiz olarak bir ön test geçirir. Keza FPGA üzerinde yapılan modelleme HDL denilen programlama dili ile yapılır.

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 [1]. 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 [2]. 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 [3]. 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 [4]. 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 [5].

Bu çalışmada ise 16 – bit veri yoluna , 1 adet akümülatör, 8 adet genel amaçlı yazmaç, 3 adet indeks yazmacı ve 1 adet yığıt göstergeci yazmacı olmak üzere toplam 13 adet yazmaca, 8 Kilobyte kapasiteli bir bellek birimine, 34 adet komuttan oluşan bir komut setine ve 50 Mhz maksimum çalışma frekansına sahip bir mikroişlemci 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. Mikroişlemci barındırdığı FPGA mimarisine uygun komutlarıyla program çıktılarını FPGA kartı çıkışlarında gösterebilmektedir.

2. Mikroişlemci Özellikleri

Bu bölümde, tasarlanan mikroişlemcinin özelliklerinden bahsedilcektir. 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. Bu mikroişlemcinin komut seti ise kendine özeldir. Bu yüzdendir ki çalışmada yazılım geliştirme amaçlı ayrı bir ortam tasarlanarak sunulmuştur.

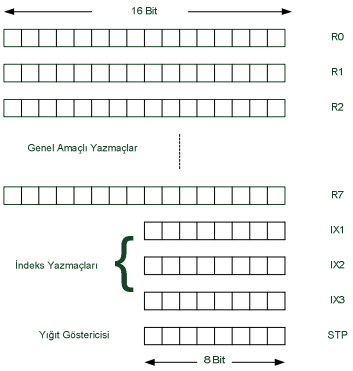
2.1. Komut Seti

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.

- Bütünlük : Komut seti mikroişlemci özelliklerinin tümünün kullanabileceği şekilde en az bir komut içermelidir.
- Ortogonallik : Komut seti operasyonel olarak birbiriyle hemen hemen aynı işi yapan, görev bakımından benzeşen komutlar içermemelidir.
- Geriye dönük uygunluk : Komut Seti bulunduğu mikroişlemci ailesinin mevcut mikroişlemciden daha öncekilerinin komut setlerini içermelidir.
- Genişleyebilirlik : Komut seti gerektiğinde adresleme bakımından genişletilebilir komutlardan oluşmalıdır. Bu mikroişlemcinin tasarımında geriye dönük uygunluk ve genişleyebilirlik bir miktar göz ardı edilmiştir. Çünkü bu mikroişlemci herhangi bir mikroişlemci ailesine dahil olmayan ve adresleme alanları daha sonra sanal bellek ya da ön belleğe alma gibi değişik metodlarla değiştirebilir durumdadır. Tasarlanan mikroişlemci 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. Komut seti tablosu Ek A’ da verilmiştir.

2.2. Kayıtçı Dosyası

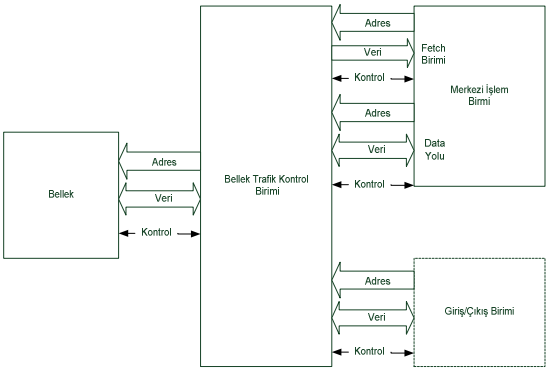
Kayıtçı dosyası genellikle mikroişlemci içinde tasarlanan küçük ve hızlı bir ara katman depolama aracıdır. Kayıtçı dosyasındaki veriler aritmetik işlem birimi ve kontrol birimi için çok önemlidir. Kayıtçı dosyasının büyüklüğü ve fonksiyonları önemli tasarım parametreleridir. Tasarlanan mikroişlemcinin kayıtçı dosyası, 8 adet genel amaçlı kayıtçı, 3 adet indeks kayıtçısı ve 1 adet bellek-yığıt göstericisinden oluşur.



Şekil 1: Kayıtçı dosyası.

2.3. Sistem Mimarisi

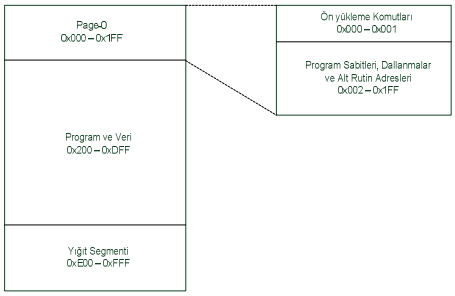
Tasarlanan mikroişlemci von-Neumann mimarisini karakterize eden 3 adet alt sistemden oluşur. Bu alt sistemler birbirleri ile global yollar üzerinden haberleşir. Her ne kadar bellek yolları CPU ve I/O birimi tarafından paylaşılsa da bellek trafik kontrol birimi bu paylaşımda her hangi bir çakışma olmaması adına tasarlanmıştır. Kalan yolların hiç biri paylaşımlı değildir. Daha ziyade iki birimi birbirine bağlamak amaçlı kullanılmıştır.



Şekil 2: Sistem mimarisi.

2.4. Bellek Organizasyonu

Von-Neumann mimarisi gereği bu çalışmada program ve veri belleği aynı blok içerisinde yer alır. Şekil 3' te tasarlanan mikroişlemcinin bellek organizasyonu gösterilmektedir.



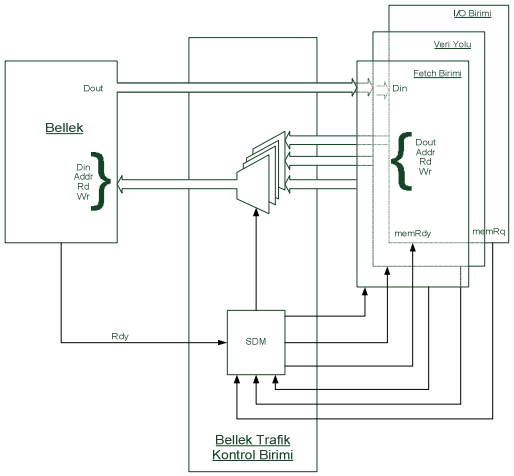
Şekil 3: Bellek organizasyonu.

Şekil 3' te görüldüğü gibi ana bellek birimi, program komutları, veri ve yığıt bölgesini içerir.

- Ön Yükleme Komutları: İlk kelime (word) olan 0x000, 0x001' de de belirtildiği üzere ilk programın başlangıcı olan 0x200' e atlama (jump) komutunu içerir.
- Program Sabitleri, Dallanmalar ve Alt Rutin Adresleri: Bu bölge Page-0 olarak tanımlanan alanın ikinci bölümüdür. Bu bölümde tanımlanan program sabitleri, dallanma ve alt rutin çağrıları için kullanılacak etiket adresleri tutulur.
- Program ve Veri Segmenti: Bu bölüm programları makina dilindeki görünümüyle tutar. Aynı zamanda program koşarken işlenen veriler de burada tutulur.
- Yığıt Segmenti: Bu bölüm LIFO (Last In First Out) mantığıyla çalışır. Erişim sadece PUSH ve POP komutlarıyla yapılır.

2.5. Bellek Trafik Kontrol Birimi

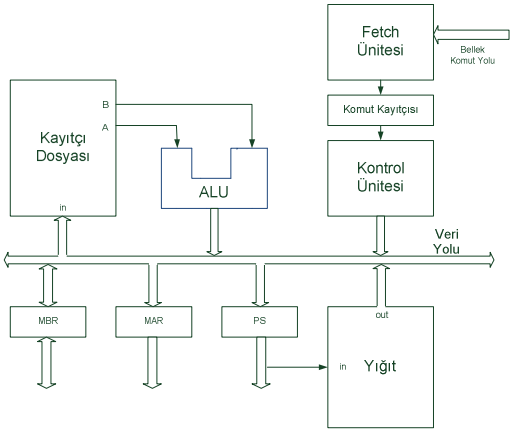
Tasarlanan mikroişlemci belleğe erişim için iki adet yol içerir. Birisi program için, diğeri ise veri içindir. Bellek biriminin iki ayrı portu olmadıkça bu iki yoldan belleğe aynı anda ulaşmak mümkün değildir. Bu erişim kontrollü bir biçimde yapmak diğer bir yöntemdir. Aslında bu yöntemde yine iki birimden aynı anda erişim olmamaktadır. Böyle bir durumda her bir birim belleğe ayrıcalıklı bir erişiminin olduğunu düsüşünür. Bellek trafik kontrol birimi üç adet ana birim ve bellek arasında bu görevi üstlenir. Bu birimler; Fetch birimi, veri yolu 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.



Şekil 4: Bellek trafik kontrol birimi

2.6. CPU Mimarisi

Tasarlanan işlemcinin bloklar halinde gösterimi aşağıdaki gibidir.



Şekil 5: CPU mimarisi.

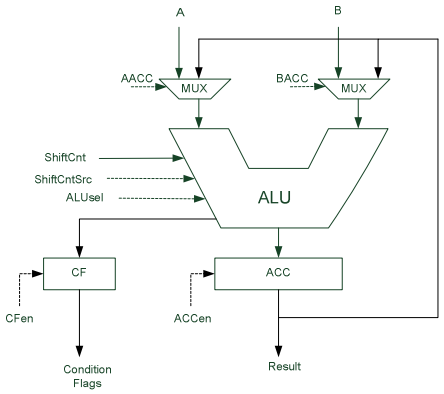
- Kayıtçı Dosyası: Hesaplanan anlık veriyi tutar.
- ALU : Aritmetik ve Lojik İşlem Birimi.
- Fetch Ünitesi : Bellekten komutları alıp getiren ünitedir.
- Komut Kayıtçısı : O anda yürütülen komutu tutan 16 bitlik kayıtçıdır.
- Kontrol Ünitesi : Çözme ve Yürütme ünitelerinden oluşur.
- PS : Program Sayacı : 12 bitlik ileri sayaçtır. Bir sonra yürütülecek komutun adresini belirler.
- Yığıt : Bir alt rutine girildiğinde program sayacının o anki değerini tutar ve alt rutinden çıkıldığında bu bilgiye bakılır.
- 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.

- 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.
- 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 tanımlansan da aynı yol üzerinden adres bilgisi de taşınabilmektedir.

2.7. Aritmetik Lojik İşlem Birimi

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



Şekil 6: Aritmetik lojik işlem birimi

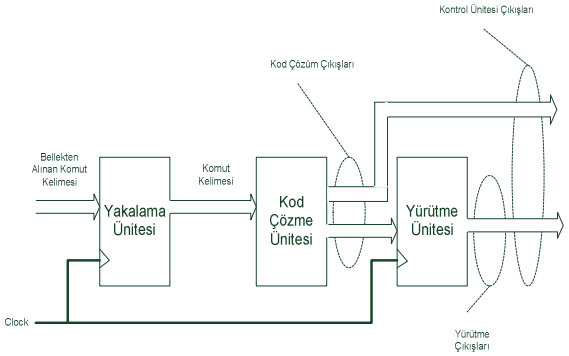
2.7.1. ALU' nun Kontrol Edilmesi

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

- ALUsel (ALU Select) : Hangi operasyonun yürütüleceğini belirleyen clocktan bağımsız sinyaldir. Alabileceği değerler ALU' nun gerçekleştirebileceği operasyonlarla sınırlıdır.
- ACCen (Acumulator Enable) : ALU operasyonu tamamlandıktan sonra akümülatörü aktif etmek için kullanılan clocka bağımlı sinyaldir.
- CFen (Control Flags Enable) : Hesaplama işlemi bittikten sonra durum bayraklarını aktif etmek için kullanılan cloka bağlı sinyaldir.
- AACC (A is the ACC) : Sol taraftaki operandın akümülatör mü yoksa bir kayıtçı mı olduğunu ALU' ya bildiren clocktan bağımsız sinyaldir.
- BACC (B is the ACC) : Sağ taraftaki operandın akümülatör mü yoksa bir kayıtçı mı olduğunu ALU' ya bildiren clocktan bağımsız sinyaldir.
- ShiftCntSrc (Shift Count Source) : Kaydırma operasyonunda baz alınacak operandın B girişi mi yoksa ShiftCount girişi mi olacağını ALU' ya bildiren asenkron sinyaldir.

2.8. Kontrol Ünitesi ve Boru Hattı

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



Şekil 7: Kontrol ünitesi ve alt bileşenleri.

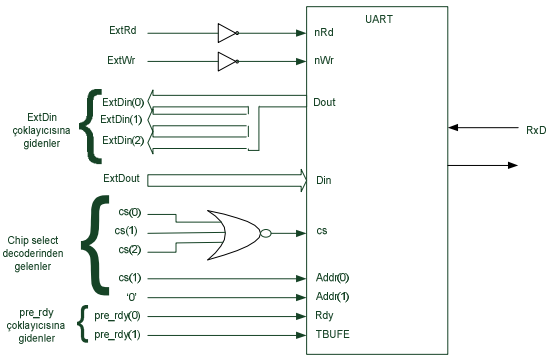
Tasarlanan mikroişlemci boru hattı yapısına sahiptir. Bu yapı iki aşamadan meydana gelmiştir. Bunlar, komutu getirilmesi (fetch) , kod çözme ve yürütme aşamasıdır. Boru hattı yapısı sadece bir slot uzunluğundadır. Bunun anlamı verilen bir zamanda kod çözme ve yürütme işlemi yapılırken fetch ünitesi sıradaki (sadece bir sonraki) komutu getirme işlemini yapabilir. Daha uzun bir boru hattı yapısı bu mikroişlemci 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 rutinin ilk işlemi boru hattının en tepesine taşınması gerekmektedir. Bu tepeye koyma işlemi boru hattının uzunluğuyla doğru orantılı olarak oldukça fazla saat darbesiyle gerçekleştirilebilir.

2.9. Universal Asynchronous Receiver Transmitter

Tasarlanan mikroişlemcinin giriş çıkış fonksiyonlarını gerçeklemek için sisteme bir Universal Asynchronous Receiver / Transmitter (UART) arayüzü eklenmiştir. Seri iletişimi sağlayan ve VHDL tasarımı oldukça karmaşık olan bu komponent OpenCores web sitesinden[6] hazır modül olarak indirilmiş ve yazılımsal olarak mevcut sisteme uygun hale getirilmiştir. Buna ilave olarak mikroişlemcinin I/O tasarımı interrupt (kesme) sinyallerini desteklemediğinden orjinal UART tasarımından bu sinyaller de kaldırılmıştır. UART mikroişlemciye Şekil 8' de görüldüğü gibi 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.



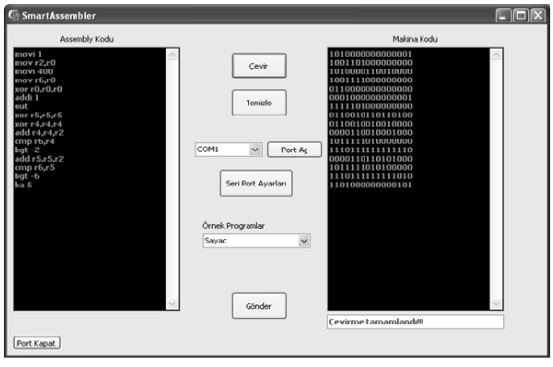
Şekil 8: UART arayüzü.

3. Mikroişlemicinin Programlanması

Tasarlnan mikroişlemci 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 mikroişlemcinin RAM' ine yazılır ve program sayacı sıfırlanarak yüklenen programın çalışması sağlanır.

3.1. Smart Assembler Yazılım Geliştirme Arayüzü

Smart Assembler, Delphi programı ile C++ dili kullanılarak yazılmış bir arayüz progamıdır. Arayüz temel olarak, yazılan mnemonic kodları mikroişelmcinin 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' daki gibidir.



Şekil 9: Smart Assembler arayüzü.

4. Örnek Programlar

Bu bölümde tasarlanan mikroişlemcinin komut seti kullanılarak hazırlanmış iki adet program anlatılacaktır.

4.1. 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 değere 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.

4.2. 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
bl 13  ; PC' a 13 değeri yüklenilir ve alt programa dallanılır.
cmp r3,r4
bgt -2
mov r0,r7
out                ; Sonuç (120) 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
bret               ;Alt progamdan geri dönülür.
```

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

5. Sonuçlar ve Öneriler

Çalışma sonucunda ağırlıklı olarak von – Neumann mimarisi özellikleri barındıran, 16 – bit veri yoluna , 1 adet akümülatör, 8 adet genel amaçlı yazmaç, 3 adet indeks yazmacı ve 1 adet yığıt göstergeci yazmacı olmak üzere toplam 13 adet yazmaca, 8 Kilobyte kapasiteli bir bellek birimine, 34 adet komuttan oluşan bir komut setine ve 50 Mhz maksimum çalışma frekansına sahip bir mikroişlemci tasarlanmıştır. Tasarlanan mikroişlemcinin en çabuk gerçekleştirdiği komut olan ‘ out ‘ komutu 3 saat darbesi yani 60 ns. sürede sonuçlanmaktadır. En yavaş gerçekleştirdiği komut ise lojik öteleme işlemi komutu olan ‘ srl ‘ ve ‘ sll ‘ komutlarıdır. Bu komutlar 66 saat darbesi yani 1320 ns. sürede sonuçlanmaktadır. 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. Bu çalışmanın birkaç seviye ileri gitmesi bazı geliştirmelerin yapılması ile mümkündür. Örneğin, çalışmada öteleyici olarak barrel shifter denen yapılar kullanılmıştır. Barrel shifter yapısı belli büyüklükteki kelimeyi (word) verilen bir değer kadar, tek bir saat darbesinde, öteleyebilen yapılardır. Ancak bu yapılar kelime uzunluğuna göre ardışık şekilde bağlanmış çoklayıcılardan oluşur. Çalışmada 16 bitlik bir öteleme yapmaya olanak sağlamak için tam 64 adet çoklayıcı ardışıl olarak bağlanmıştır. Bu da oldukça yüksek propagasyon gecikmelerine sebep olmaktadır. Bu nedenle çoklayıcı yapılarının paralel çalışma prensibi gözetilerek tasarlanması çalışma için önemli bir iyileştirme olabilir. 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 kendi karar vermesi sağlanabilir. Mevcut tasarıma daha fazla çevre birimi desteklemek üzere geliştirmeler yapılabilir. Örneğin; harici sistem saati üreteci, rastgele numara üreteci, paralel arayüz, VGA arayüzü bunlardan başlıcalarıdır.

6. Kaynaklar

[1] Başak S. (2008), SelCPU. *CPU Turkey 2008 Gömülü Sistemler Yarışması.*

[2] Ergin O. vd. (2008), Kasırga. *CPU Turkey 2008 Gömülü Sistemler Yarışması.*

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

[4] Ertürk S. vd. (2008), Cpu_kulis, *CPU Turkey 2008 Gömülü Sistemler Yarışması.*

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

[6] OpenCores web sitesi: www.opencores.org.

Ek A

Komut Seti

İşlem Kodları	Komutlar	Komut Türleri	İşlemler
00001	add	R	Ra ← Rb + Rc
00010	addi	I	Brkç ← Brkç + Anlık Değer
00011	sub	R	Ra ← Rb - Rc
00100	subi	I	Brkç ← Brkç - Anlık Değer
00101	mul	T	Üst+Alt ← Ra * Rb
00110	muli	I	Üst+Alt ← Brkç * Anlık Değer
00111	mulu	T	Üst+Alt ← Ra * Rb
01000	and	R	Ra ← Rb Λ Rc
01001	andi	I	Brkç ← Brkç Λ Anlık Değer
01010	or	R	Ra ← Rb V Rc
01011	ori	I	Brkç ←Brkç V Anlık Değer
01100	xor	R	Ra ← Rb (XOR) Rc
01101	xori	I	Brkç ← Brkç (XOR) Anlık
01110	not	T	Ra ← !Rb
01111	sll	T	Ra ← Ra << Rb
10000	srl	T	Ra ← Ra >> Rb
10001	sla	T	Ra ← Ra <<< Rb
10010	sra	T	Ra ← Ra >>> Rb
10011	mov	T	Ra ← Rb
10100	movi	I	Brkç ← Anlık Değer
10101	lw	T	Ra ← Bellek [Rb]
10110	sw	T	Bellek [Rb] ← Ra
10111	cmp	T	Ra > Rb ise Büyük Bayrağı = 1 Ra < Rb ise Küçük Bayrağı = 1
11000	beq	I	Sıfır Bayrağı = 1 ise PS = PS + İşaretle gen.Anlık Değer
11001	bne	I	Sıfır Bayrağı = 0 ise PS = PS + İşaretle gen.Anlık Değer
11010	ba	I	PS = Anlık Değer
11011	bl	I	PS_ret = PS PS = Anlık Değer
11100	bret	I	PS = PS_ret
11101	bgt	I	Büyük Bayrağı = 1 ise PS = PS + İşaretle Gen. Anlık Değer
11110	blt	I	Küçük Bayrağı = 1 ise PS = PS + İşaretle Gen. Anlık Değer
1111100	in	S	Brkç = Anlık Değer (Sıfırla Gen.)
1111101	out	S	Brkç [7:0] dışarı verilir.
1111110	hlt	S	PS = PS
1111111	nop	S	PS ← PS + 1