KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

16-BİTLİK MİKROİŞLEMCİNİN FPGA MİMARİLERİ KULLANILARAK GERÇEKLEŞTİRİLMESİ

YÜKSEK LİSANS

Elektronik ve Haberleşme Müh. Esma ALAER

Anabilim Dalı: Elektronik ve Haberleşme Mühendisliği

Danışman: Yrd.Doç. Dr. Ali Tangel

KOCAELİ, 2006

KOCAELİ ÜNİVERSİTESİ * FEN BİLİMLERİ ENSTİTÜSÜ

16-BİTLİK MİKROİŞLEMCİNİN FPGA MİMARİLERİ KULLANILARAK GERÇEKLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Elektronik ve Haberleşme Müh. Esma ALAER

Tezin Enstitüye Verildiği Tarih: 26/05/2006

Tezin Savunulduğu Tarih: 16/06/2006

Tez Danışmanı	Üye	Üye
Yrd.Doç. Dr. Ali TANGEL	Doç. Dr. Bekir ÇAKIR	Yrd. DOÇ. Dr. Ahmet Turan ÖZCERİT

KOCAELİ, 2006

ÖNSÖZ ve TEŞEKKÜR

Günümüz teknolojisinde mikroişlemciler alanında büyük gelişmeler gözlenmektedir. Bu alanda önemli bir yere sahip olan Intel firması, mikroişlemci tasarımına 4004 CPU ile başlamış, 8008,80x86 ile devam etmiş ve günümüzde de Pentiumlarla bu pazarda olağanüstü bir yere sahip olmuştur.

Ülkemizde ise mikroişlemcilerden çok mikro denetleyiciler üzerine çalışmalar yapılmış, bu çalışmalarda bulunan mikroişlemci birimi ise tamamen tasarlanmayıp hazır modüller kullanılmıştır.

Bu tezde, 16-bitlik mikroişlemci cipinin tasarımı, bir donanım tanımlama dili (VHDL) yardımıyla FPGA üzerinde gerçekleştirilmiştir.

Bu çalışmanın her safhasında gayretini esirgemeyen ve özveri ile yaklaşan danışmanım Yrd. Doç. Dr. Ali TANGEL'e, değerli görüşleri ile bana yol gösteren Yrd. Doç. Dr. Mehmet YAKUT'a, bu noktaya gelişimde her zaman yanımda olan aileme, özellikle babam A. Osman ESEN'e, maddi ve manevi desteğini esirgemeyen eşim Hacı ALAER'e, teşekkürü bir borç bilirim.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

TABLOLAR DİZİNİ

SİMGELER VE KISALTMALAR LİSTESİ

ALU	: Arithmetic Logic Unit
CISC	: Complex Instruction Set Computing
CPU	: Central Processing Unit
EPROM	: Erasable Programmable Only Memory
E^2PROM	: Electronically Erasable Programmable Read Only Memory
FPGA	: Field Programmable Gate Array
FPU	: Floating Point Unit
G/Ç	: Giriş / Çıkış
HDL	: Hardware Description Language
ISE	: Integrated Software Environment
IP	: Instruction Pointer
LUT	: Look Up Table
MAR	: Memory Address Register
MBR	: Memory Buffer Register
MİB	: Merkezi İşlem Birimi
PAR	: Place and Route
PC	: Program Counter
PLA	: Programmable Logic Array
PLD	: Programmable Logic Device
PROM	: Programmable Read Only Memory
PSW	: Program Status Word
RAM	: Random Access Memory
RISC	: Reduced Instruction Set Computing
ROM	: Read Only Memory
RTC	: Real-Time Counter
SR	: Status Register
SRAM	: Static Random Access Memory
SOC	: System on Chip
SP	: Stack Pointer
VHDL	: Very High Speed Integrated Circuit Hardware Description Language
VLSI	: Very Large Scale Integration

16-BİTLİK MİKROİŞLEMCİNİN FPGA MİMARİLERİ KULLANILARAK GERÇEKLEŞTİRİLMESİ

Esma ALAER

Anahtar Kelimeler: FPGA, HDL, SoC, Mikroişlemci

Özet: Dijital devrelerin karmaşıklığının giderek artması tasarım metotlarının da gelişmesini gerektirmektedir. Geleneksel metotların yerini, tasarım süresini ve maliyetini azaltan, esnek yapıları sayesinde tasarımcıya büyük kolaylık getiren "donanım tanımlama dilleri" (HDL) almıştır. Bir donanım tanımlama dili yardımıyla bir çok alt birimden oluşan tasarım tek bir elemana yüklenebilir ve bu sayede tek bir çip içerisinde bir sistem (system-on-chip-SoC) oluşturularak baskı devre çıkarma, lehimleme gibi hem uzun süren hem de maliyeti artıran işlemler devreden çıkartılabilir.

"SoC" teknolojisinde tasarımı meydana getiren birimler bir donanım tanımlama yardımıyla ifade edilirler. Tanımlanan sistemin mantıksal olarak doğruluğunun testi için fonksiyonel simülasyon aşaması gerçekleştirildikten sonra bütün birimler sentezlenerek tasarımı meydana getiren bağlantılar oluşturulur. Sentezleme sonrasında ise yerleştirme ve bağlama basamağına geçilir. Yerleştirme ve bağlama işleminden sonra yapılan simülasyon ile tasarımın zaman düzleminde gerekli şartları sağlayıp sağlamadığı gözlemlenebilir. Bütün birimler, sentezleme ve yerleştirme işlemleri sonunda tekrar programlanabilir bir tümleşik devreye aktarılır. Sistemin uygulama aşamasında gerek büyük kapasiteleri ve gerekse de tasarım çabukluğundan dolayı FPGA tümleşik devreleri tercih edilmektedir.

Bu çalışmada 16-bit mikroişlemcinin FPGA mimarileri kullanılarak tüm alt birimleriyle birlikte tasarımı sunulmaktadır. Tasarlanan mikroişlemci, 16-bit adres yolu ve 16-bit veri yoluna sahiptir. 16 tane genel amaçlı saklayıcı, program sayıcı ve 3-bit durum saklayıcı içermektedir. Her bir komut 16-bit kelime uzunluğuna sahiptir. Mikroişlemci toplama, çıkarma, çarpma, bölme gibi aritmetik komutları; AND, OR, NOT, XOR gibi lojik komutları; yükleme ve saklama gibi yer değiştirme komutlarını yürütebilmektedir. Mikroişlemcinin frekansı, aritmetik ve lojik komutlar için yaklaşık 3 MHz, yer değiştirme komutları için yaklaşık 1,5 MHz ve ivedi yer değiştirme komutları için yaklaşık 2,3 MHz'dir.

THE DESIGN AND IMPLEMENTATION OF A 16-BIT MICROPROCESSOR USING FPGA ARCHITECTURES

Esma ALAER

Keywords: FPGA, HDL, SoC, Microprocessor

Abstract: As a result of the increase in digital circuits' complexity, new design methods need to be developed. Traditional methods have been replaced with "hardware description languages (HDL), which have flexible structures, short design cycle and low-cost. A design composes a number of sub-units, which can be uploaded into a programmable device by using an HDL. Therefore, a system (SoC) in a single chip can be created without doing processes which increase the cost and time-to- market, such as forming a PCB and soldering.

In SoC technology, design units are described by using a hardware description language. To test the logical verification of the described system, a functional simulation process is realized, and then the net-list of the design is formed with synthesizing all units. After timing simulation, routing processes are realized. Later, timing simulation can be done to observe whether the design meets necessary timing requirements or not. Then all units are uploaded into a programmable device. In the implementation step, FPGA devices are preferred because of their huge capacity and reducing the design cycle.

This study presents a design and implementation of 16-bit microprocessor with VHDL by using FPGA. The microprocessor can directly access to the memory which consists of 16-bit words, addressed by a 16-bit word-address. Instructions are all multiples of 16-bit words, and are stored in this memory. There are 16 general purpose registers (R0–R15), a program counter (PC) and a condition code register (CC). The microprocessor can execute 16 instructions such as add, subtract, multiply, divide, load and store. The frequency of the microprocessor is about 3 MHz for operand such as add, subtract, multiply and divide and about 1.5 MHz for operand such as load and store and about 2.3 MHz for operand such as quick load and quick store.

BÖLÜM 1. GİRİŞ

Teknolojideki hızlı gelişmelere paralel olarak yeniden yapılandırılabilir mimarilerin kaynak kapasiteleri (yoğunlukları) ve sağlayabildikleri maksimum sinyal frekansı da artmaktadır [1]. Alan programlanabilir Kapı Dizileri (Field Programmable Gate Array-FPGA) binlerce kapıdan oluşur ve karmaşık sistemlerin, yeniden programlanabilir mimariler ile gerçeklenmesine imkan tanır.

Günümüzde, FPGA mimarileri kullanılarak tasarlanacak bir sistem için çeşitli tasarım metotları bulunmaktadır. Donanım tanımlama dilleri, tasarım sürecini ve maliyeti azaltması gibi avantajları sebebiyle kullanılan mevcut metotlar içinde en çok tercih edilenidir [2].

Bilgisayarlarda yoğun bir biçimde kullanılan mikroişlemciler, bir donanım tanımlama dili yardımıyla tasarlanarak FPGA mimarileri üzerinde gerçeklenebilir. Böylece çevre birimlerinin özellikleri değiştirilebilen modüler bir tasarım sağlanmış olur. Sonuç olarak düşük maliyet / kabul edilebilir performans kriterini sağlayabilen çok amaçlı bir mikroişlemci tasarlanmış olur.

Mikroişlemci üzerine çalışmalar 1948'lerde transistörün bulunmasıyla başlamış ve günümüzde de hızlı bir şekilde devam etmektedir. Intel firmasının 1971'lerde ürettiği 4004 mikroişlemcisi 740 KHz frekansa sahip iken günümüz mikroişlemcilerinde bu hız 3GHz'i geçmiş durumdadır [3]. Ülkemizde mikro denetleyicilerle ilgili birçok çalışma yapılmasına rağmen bu projelerdeki mikroişlemci birimi tasarlanmayıp hazır devreler kullanılmıştır.

Bu tezde 16-bit mikroişlemcinin tüm alt birimleriyle birlikte tasarımı sunulmaktadır. Tasarlanan mikroişlemci, 16-bit adres yolu ve 16-bit veri yoluna sahiptir. 16 tane genel amaçlı saklayıcı, program sayıcı ve 3-bit durum saklayıcı içermektedir. Her bir komut 16-bit kelime uzunluğuna sahiptir.

Tasarlanan 16-bitlik mikroişlemci ile ilgili bilgi verilmeden önce tezin 2. bölümünde mikroişlemcinin tanımı yapılarak mikroişlemcilerin tarihi gelişiminden bahsedilmiş, mikroişlemcilerin kullanım alanları verilmiştir.

Tezin 3. bölümünde 16-bitlik mikroişlemcinin tasarım aşamaları açıklanarak mikroişlemciyi oluşturan birimlerin tasarım özellikleri, VHDL dilindeki tanımları, şematik gösterimleri ve fonksiyonel olarak doğrulanması hakkında bilgi verilmiştir.

4. bölümde 16-bitlik mikroişlemcinin gerçekleme ve doğrulama işlemleri açıklanmıştır. Tasarlanan mikroişlemcinin gerçeklendikten sonra doğru bir şekilde çalışıp çalışmadığının gözlemlenebilmesi için doğrulanması gerekmektedir. Bölüm 4'te tasarlanan mikroişlemcinin işleyebildiği tüm komutlar hakkında bilgi verilmiştir. Kullanılan geliştirme ekipmanları, yazılımları ile ilgili detaylı bilgi ise ekler kısmında açıklanmıştır.

BÖLÜM 2. MİKROİŞLEMCİLERE GİRİŞ

2.1. Mikroişlemci Nedir?

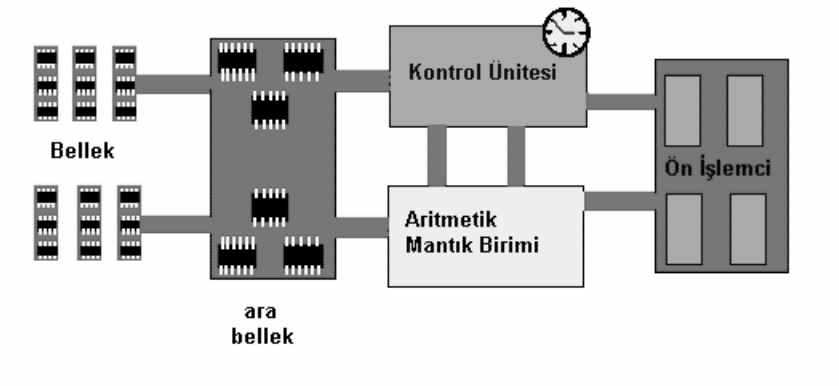
Mikroişlemci; gerek yaptığı işlemlerin mikro saniyeler mertebesinde olması gerekse içindeki elektronik devrelerin ve bölümlerin mikron boyutlarında olması nedeniyle bu adı almıştır. Mikroişlemci, bilgisayarın değişik birimleri arasında veri akışı ve veri işleme görevlerini yerini getiren büyük ölçekli veya çok büyük ölçekli entegre devredir. Mikroişlemci entegre devresi, yazılan programları meydana getiren makine kodlarını yorumlamak ve yerine getirmek için gerekli olan tüm mantıksal devreleri içerir. Adından da anlaşılacağı gibi mikroişlemci (veya işlemci) aritmetik ve mantıksal işlemleri yapabilen bir elektronik çiptir. Boyutları çok küçük olmasına rağmen içinde binlerce, yüz binlerce veya milyonlarca elektronik devre elemanı bulunduran mikroişlemci, aslında matematiksel işlemleri, mantıksal "var" yada "yok" temelinden yararlanarak hesaplar.

2.2. Mikroişlemcinin Temel Birimleri

Bir mikroişlemci beş temel birimden oluşur:

- Kontrol birimi,
- Aritmetik ve lojik birim (ALU),
- Genel amaçlı saklayıcılar,
- Özel amaçlı saklayıcılar,
- Teknoloji bağımlı birimler

Mikroişlemciye ait temel birimler Şekil 2.1'de görülmektedir.



Şekil 2.1.Mikroişlemciye ait temel birimler

2.2.1. Kontrol birimi

Kod hafızasından yazılıma ait kodu okur, değerlendirir ve sonuca göre gerekli işlemleri yapar. Mikroişlemcinin içerisinde olan bütün veri aktarım işlemleri, ALU ile olan haberleşme işlemleri ve diğer çevre birimler ile ilgili olan eşzamanlı çalışma işlemleri bu birim tarafından kontrol edilir.

2.2.2. Aritmetik ve mantık birimi (ALU)

MİB (Merkezi İşlem Birimi) içinde yapılması gereken aritmetik ve lojik işlemler ALU içinde gerçeklenir. Karşılaştırma ve karar verme işlemleri de bu birim içinde gerçeklenir. ALU'nun üzerinde işlem yapacağı verilerden birincisi (birinci işlenen) Akümülatörde bulunur, ALU'nun yetenekleri, MİB'i doğrudan etkilemektedir. Bu nedenle, ALU'nun yapabildiği işlemler, MİB'in yeteneklerini belirleme açısından önem taşır. ALU, ACC ve Durum Saklayıcı ile doğrudan ilgilidir.

ALU'nun gerçekleştirdiği işlemler şunlardır:

- Aritmetik işlemler: ALU toplama ve çıkarma işlemlerini yerine getirebilmelidir. Çarpma ve bölme işlemini her ALU yerine getiremez. Bu açıdan çarpma ve bölme işlemini yapabilen ALU' ler üstün sayılırlar.

- Lojik İşlemler: ALU içinde temel VE, VEYA ve SEÇKİN VEYA işlemleri yerine getirilir.
- Karşılaştırma ve Karar Verme İşlemleri: İki büyüklüğün birbirine göre büyüklük, küçüklük veya eşitlik karşılaştırması yapılabilir. Ortaya çıkan duruma göre karar verilir ve sonuç Durum Saklayıcısına işlenir.

2.2.3. Genel amaçlı saklayıcılar

Mikroişlemcinin içinde, geçici olarak verilerin saklandığı saklayıcılardır. 8-bit mikroişlemcilerin çoğunda, bu saklayıcılar üzerinde, veri saklama haricinde bir işlem yapılmaz. Bu mikroişlemcilerde, genel amaçlı bir saklayıcıda bulunan bir veri üzerinde, bir aritmetik veya lojik işlem yapılması durumunda, bu saklayıcıdaki veri, özel bir saklayıcı olan genellikle A veya ACC (Accumulator) kısaltmasıyla belirtilen bir saklayıcıya alınması gerekir. Aritmetik ve lojik işlemlerin sadece ACC' de yapıldığı, bu tür 8-bit mikroişlemcilere ACC tabanlı olarak anılırlar.

Teknolojinin gelişmesiyle, genel amaçlı saklayıcıların işlevleri, sayıları ve bit uzunlukları artmıştır. 16-bit ve 32-bit mikroişlemcilerdeki genel amaçlı saklayıcılar üzerinde, aritmetik ve lojik işlemleri yapmak mümkündür.

2.2.4 Özel amaçlı saklayıcılar

Kontrol birimi tarafından yürütülecek komuta işaret etmek amacıyla, bütün mikroişlemcilerde özel bir saklayıcı bulunur. Bir çok 8-bit mikroişlemcide bu saklayıcının ismi program sayıcıdır (Program Counter-PC). Bu saklayıcı, bir sayma işlemi yapmayıp mikroişlemci tarafından hafızada yürütülecek komutun adresine işaret eder. Bu yüzden program sayıcıya, komut işaretçisi (Instruction Pointer-IP) demek daha uygundur. Yürütülen komutun uzunluğuna göre, PC (veya IP) bir sonraki komutun adresini işaret edecek şekilde kontrol birimi tarafından otomatik arttırılır veya azaltılır.

PC (veya IP) saklayıcısında buluna adres, hafıza adres saklayıcısında (Memory Address Register-MAR) tutulur. Bu sayede mikroişlemcinin harici adres yoluna,

belli bir süre sabit duracak bir adres çıkartılır. Bu adres, dış hafıza ve giriş /çıkış birimlerini adreslemede kullanılır.

Mikroişlemciye veri giriş/çıkış işlemi, hafıza buffer saklayıcısı (Memory Buffer Register-MBR) yoluyla olur. Bu hafıza veya giriş biriminden mikroişlemciye okunan veri, mikroişlemcini içindeki bir saklayıcıya MBR yoluyla gelir. Mikroişlemcinin içindeki bir saklayıcıdan dışarı gönderilen veri, hafızaya veya bir çıkış birimine yazılmadan bir süre MBR'da tutulur.

Komut saklayıcısı, IR, hafızadan gelen işlem kodlarının durak yeridir. Bu saklayıcıya alınan işlem kodları, kontrol birimi tarafından çözülerek mikroişlemci çalışır.

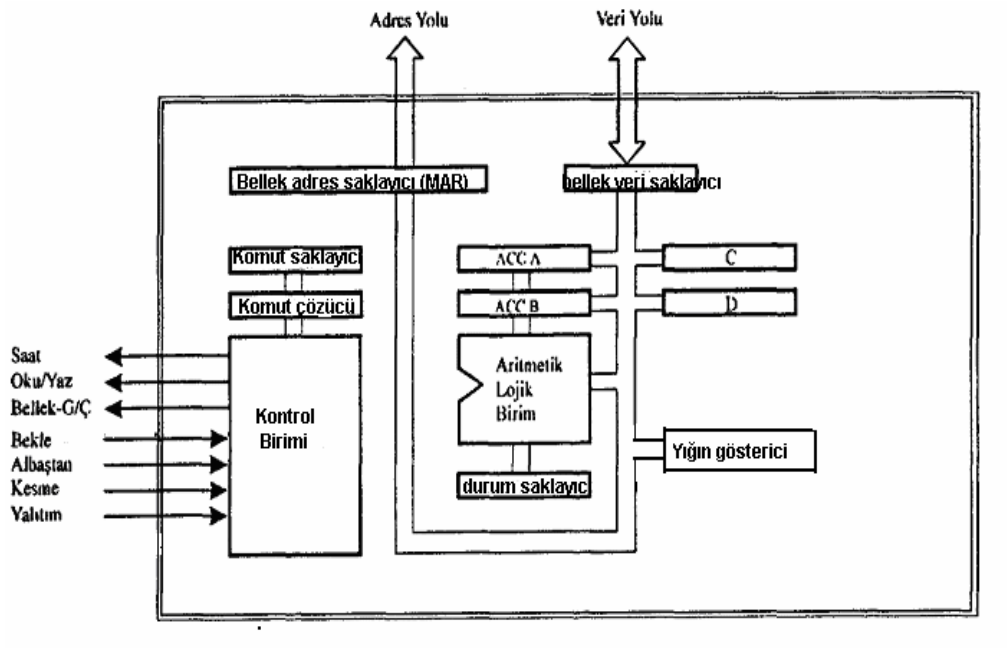
Mikroişlemcinin en son yapmış olduğu bir aritmetik ve lojik işlemin durumunu belirten saklayıcı durum saklayıcıdır (Status Register-SR) ve günümüz işlemcilerinde genellikle bayraklar (Flags-F) veya program durum kelimesi (Program Status Word-PSW) saklayıcı olarak adlandırılır. Mikroişlemcinin hafıza alanında yığın (stack) diye adlandırılan özel bir bölüm bulunur. Bu özel hafıza bölümü ile mikroişlemci arasında, bazı özel program komutlarıyla (push ve pop) veya mikroişlemci çalışmasındaki bazı işlemlerde (alt program çağrımları veya kesmeler gibi), otomatik olarak, veri transferi gerçekleşir. Mikroişlemcinin içinde, hafızadaki bu özel alana işaret eden bir yığın işaretçisi (Stack Pointer-SP) bulunur.

2.2.5. Teknoloji bağımlı birimler

Teknolojik gelişmeler, mikroişlemci tüm devrelerinde yüksek yoğunluğu mümkün kılmış ve milyonlarca transistor bir tüm devreye yerleştirilebilmiştir. Daha önceleri dışarıda olup mikroişlemcinin içine giren ilk birimlerin başında kayan nokta birimi (Floating Point Unit-FPU) gelir.

Gelişmiş mikroişlemcilerdeki, işlemci üzeri artan veri saklama kapasitesi, çok sayıda saklayıcı ve ön hafıza (cahce) gibi birimlerden oluşmuştur. Ayrıca, mikroişlemciye daha fazla hafıza adresleme yapısı sağlayan görüntü hafıza (virtual memory) yönetim birimi de mikroişlemcinin içinde yer almıştır.

Bir mikroişlemciye ait iç yapı Şekil 2.2'de görülmektedir.

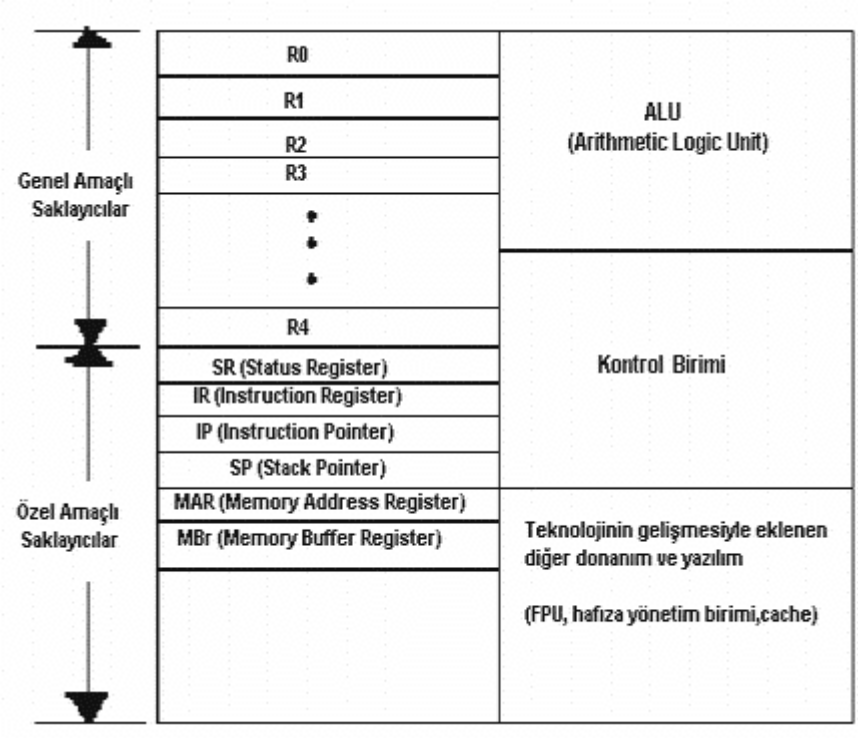


Şekil 2.2. Mikroişlemcinin iç yapısı [4]

2.3. Mikroişlemci Mimarisi

Mikroişlemciler mimari yapılarına göre farklılık gösterirler. Mikroişlemcinin mimarisi denildiğinde; mikroişlemci içindeki kaydedicilerin büyüklüğü ve yapısı ile kendi aralarında mümkün olan veri ve komut transferleri akla gelmelidir. Komut kümesi bir mikroişlemcinin tanıdığı komutlardır ve iç saklayıcı kümesi de, mikroişlemcinin çalışması sırasında geçici verilerin saklandığı bellek hücreleridir. Ortak bir mimariye sahip işlemciler, aynı komutları tanıdıklarından aynı programları çalıştırabilirler. Komut ve saklayıcı kümeleri farklı olan mikroişlemciler genellikle aynı programları çalıştıramazlar.

CISC (Complex Instruction Set Computers - Karışık Komut Kümeli Bilgisayarlar) mimarisi, Intel 80486, Pentium ve Motorola 68030 gibi işlemcileri oluştururken; RISC (Reduced Instruction Set Computers - Azaltılmış Komut Kümeli Bilgisayarlar) mimarisi, Motorola, PowerPC ve MIPS işlemcilerinde kullanılmaktadır. Şekil 2.3'de tipik bir mikroişlemcinin iç mimari modeli görülmektedir.



Şekil 2.3. Tipik bir mikroişlemcinin iç mimari modeli [5]

2.3.1. CISC işlemciler

Bu mimarinin geliştirildiği 1960 ve 1970'li yıllarda RAM'lerin sınırlı ve pahalı olması, az bellek kullanımını gerektirdi. Az bellek kullanımı, komutların ve mimarinin kompleks olmasına sebep oldu.

CISC mimarisine sahip mikroişlemcilerin transistör sayısının fazla olması nedeniyle, bu işlemcilerin ebadı büyüktür. Ayrıca, fazla ısı üreteceğinden gelişmiş soğutma sistemleri kullanılmalıdır. Bunlardan dolayı, CISC tabanlı işlemciler diğerine göre daha pahalıdır.

2.3.2 RISC işlemciler

RISC işlemcili sistemlerde amaç, komut işlenmesinin olabildiğince hızlı olmasıdır. Komutların basit ve az olması, işlemcinin uzun ve karışık olandan daha hızlı çalışabilmesini sağlar. Bu mimariyi kullanan işlemciler, aynı anda birden fazla komutun işlendiği kanal tekniğine (pipeline) sahiptir ve yüksek performansta çalışır. Kanal tekniği ile herhangi bir komutun işlenmesindeki adımlar şöyledir:

Kanal tekniği ile çalışan işlemcilerde birinci adımda komut kodu çözülür, ikinci adımda birinci komutun üzerinde çalışacağı veri (işlenen) kaydediciden alınırken, sıradaki ikinci işlenecek olan komutun kodu çözülür. Üçüncü adımda ilk komutun görevi ALU'da yerine getirilirken, ikinci komutun işleyeceği veri (işlenen) alınıp getirilir. Bu anda sıradaki üçüncü komutun kodu çözülür ve işlem böylece devam eder (Topaloğlu,1999). Kanal tekniği, komutları kademeli olarak işler; bu teknikte komutlar, her bir basamağında aynı işlemin uygulandığı birimlerden geçerler ve aynı anda paralel olarak birden fazla iş yapabilmektedirler.

Genellikle kanal tekniğini kullanan RISC çipleri, eşit uzunlukta segmentlere bölünmüş komutları çalıştırırlar. RISC mimarisinde tüm komutlar 1 birim uzunlukta olduklarından komut kodunu çözme işlemi kolaylaşır. Komut kodlarının hızlı çözülmesi ise çevrim zamanının düşmesini sağlar. Sistemde kullanılan saklayıcıların simetrik bir yapıda olması da derleme işlemini kolaylaştırır.

RISC mimarisinin önemli üstünlüklerine karşın bazı dezavantajları da mevcuttur. RISC mimarisi, CISC'in güçlü komutlarından yoksundur; bu nedenle de aynı işlemi yapmak için daha fazla komut işlemesi gerekir. Bundan dolayı da bant genişliği artar. Ayrıca; bu tasarım tekniği yüksek bellek kullanımını gerektirmektedir.

2.4. Mikroişlemcilerin Özellikleri

Mikroişlemcilerin sınıflandırılabilmesi için ölçü kabul edilen en temel özellikleri şunlardır:

a. Kelime uzunluğu (bit uzunluğu): Mikroişlemcilerin bir defada işleyebileceği kelime uzunluğu, paralel olarak işlenen veri bitlerinin sayısıdır. İşlemciler, her bir saat çevriminde, o anda sırada olan komutları ve bunlara göre de bellekteki verileri mikroişlemcinin tipine göre gruplar halinde işlerler. Komutların veya verilerin küçük gruplar halinde işlenmesi hızda azalmaya neden olur. Mikroişlemciler için 4-8-16-32 ve 64 bitlik veri uzunlukları, standart haline gelmiştir. İşlemcilerde yapılan aritmetiksel işlemlerin doğruluk oranı, bit uzunluğu büyüklüğü ile doğru orantılı olarak artmaktadır (8-bit için %0.4 iken 16-bit için %0.001'dir). Kelime uzunluğunun büyük olması; aynı anda daha çok işin yapılmasını sağlar ve bu uygulama programları için büyük kolaylıktır.

b. Mikroişlemcinin tek bir komutu işleme hızı: Saat frekansı her zaman gerçek çalışma frekansını yansıtmasa da; bir mikroişlemcinin hızıyla doğrudan ilgilidir. Bir mikroişlemcinin hızını artıran temel unsurlar şöyle sıralanabilir:

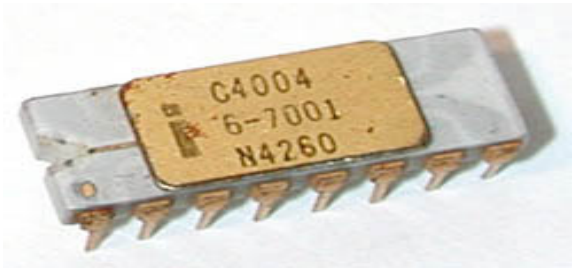
- Merkezi işlem biriminin devre teknolojisi ve planı
- Kelime uzunluğu
- İşlemci komut kümesi çeşidi
- Zamanlama ve kontrol düzeni
- Kesme altyordamlarının çeşitleri
- Bilgisayar belleğine ve giriş/çıkış aygıtlarına erişim hızı

c. Mikroişlemcinin doğrudan adresleyebildiği bellek büyüklüğü: Mikroişlemci, adres yolu aracılığıyla ana belleği adresleyebilir. Adres yolu, işlemcinin yapısına göre değişir ve adres yolu hattı çok olan bir sistemin adresleme kapasitesi de o kadar büyüktür.

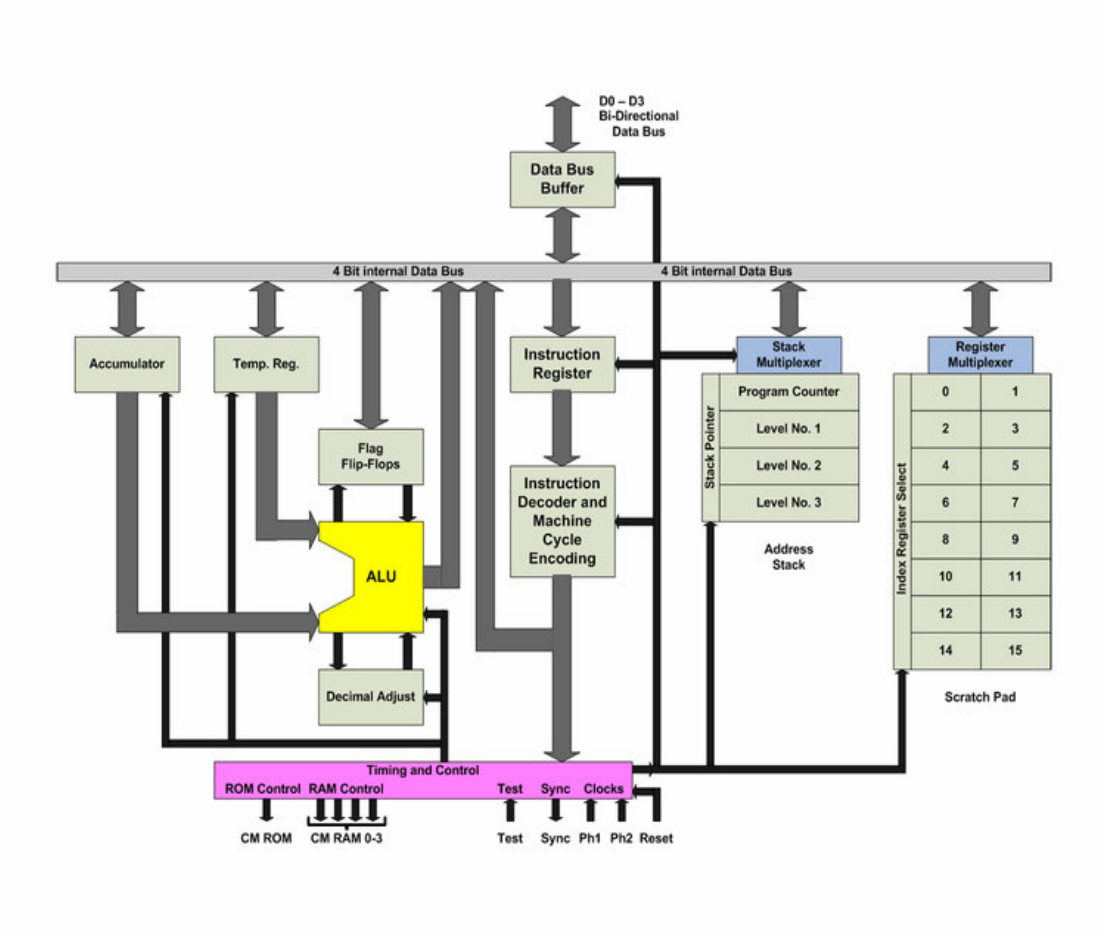
Bu üç ana özelliği dışında mikroişlemcileri dolaylı olarak etkileyen çeşitli özellikler vardır: Mikroişlemci üzerinde kullanılabilecek kaydedici sayısı ve tipleri; programcının elde edebileceği çeşitli komutlar ve bellek adreslerken ihtiyaç duyduğu farklı adres modları; kullanılan işletim sisteminin uyumluluğu gibi.

2.5 Mikroişlemcilerin Tarihi Gelişimi

İlk mikroişlemci Intel 4004 adı ile 1971 yılında üretilmiştir. 4004 işlemci sadece toplama ve çıkarma işlemlerini yapabilen 4 bitlik bir işlemci idi. Fakat bütün birimler tek bir çipte toplandığı için bu çok önemli bir gelişme idi. 4004'ten önce bilgisayarlar birden fazla çip kullanılarak veya farklı bileşenlerin birleştirilmesi ile üretiliyordu. 4004 ile taşınabilir elektronik hesap makineleri de büyük bir gelişme kaydetmişti. İşlemcinin maksimum saat frekans hızı 740 KHz idi. Şekil 2.4'te Intel 4004 işlemci entegresi ve Şekil 2.5'te de bu işlemciye ait iç yapı görülmektedir.



Şekil 2.4. Intel 4004 işlemci entegresi [3]



Şekil 2.5. Intel 4004 işlemcisinin iç yapısı [3]

Günümüzde mikroişlemciler çoğunlukla bilgisayarlarda kullanılmaktadır. Teknolojinin gelişimi doğrultusunda, daha önce mikroişlemci tüm devresi dışında olan pek çok Giriş / Çıkış ve bellek alt birimleri, MİB üzerine taşınmıştır.

Kişisel bilgisayarlar (PC) için geliştirilen ilk mikroişlemci Intel 8080'dır ve 8-bit teknolojiye sahip olup 1974 yılında tanıtılmıştır. Bununla birlikte bilgisayar dünyasındaki gerçek sıçrama 1979 yılında üretilen Intel 8088'dir. Intel 8088 işlemcisi IBM PC'lerde kullanılmıştır (1982). Daha sonraları 80286, 80386, 80486, Pentium I, Pentium II, Pentium III, Pentium 4 işlemcileri geliştirilmiştir. Tüm bu işlemciler Intel tarafından geliştirilmiş olup hepsi 8088 işlemcisinin temel tasarımı üzerinde yapılan değişiklikler ile geliştirilmiştir. Pentium 4 işlemcisi orijinal 8088 işlemcisi üzerinde çalışan her türlü kodu 5000 kat daha hızlı çalıştırabilmektedir.

2.5.1. Intel ailesinin gelişimi

Intel firması 1968 yılında bellek tümleşik devreleri yapmak üzere kuruldu. Üretecekleri bir hesap makinesi için CPU tümleşik devresi isteyen, hesap makinesi üreten bir firmanın talebi; ve yine üretecekleri bir terminal için özel bir tümleşik devre isteyen, diğer bir firmanın istediklerini karşılamak için, Intel firması 4004 (1971) ve 8008 (1972) CPU'larını üretti.

Mikroişlemciler ve mikrobilgisayarların sınıflandırılmasında en temel ölçü, mikroişlemcinin tümleşik devre üzerinde işlem yaptığı en uzun verinin bit sayısı, yani kelime uzunluğudur. 4-bit işlemci olan 4004 ve 8-bit işlemci olan 8008'den başlayarak, mikroişlemciler ve mikrobilgisayarlar için, 4-bit, 8-bit, 16-bit, 32-bit ve 64-bit gibi veri uzunluk standartları doğmuştur.

Intel, bu ilk müşterilerinden başkasının, 4004 ve 8008 tümleşik devrelerine ilgi göstereceklerini tahmin etmediği için, üretim hattını düşük kapasiteli tutmuştur. Fakat tahminlerin aksine, bu tümleşik devrelere çok büyük bir ilgi oldu. Bunun sonucu ve aynı zamanda 8008'in 16 KB'lık bellek limitini aşmak amacıyla, Intel firması 1974 yılında genel amaçlı 8080 CPU'sunu üretti. Birden bu tümleşik devreye büyük bir talep oldu ve kısa bir süre içinde 8080, 8-bit mikroişlemci endüstri standardı oldu. Intel, iki yıl sonra 1976'da gelişmiş bir 8080 işlemcisi olan 8085 piyasaya sürdü.

Intel 1978 yılında ilk 16-bit mikroişlemcisi olan 8086'yı üretti. 8086 daha önceki 8080/8085 ürününe bazı yönlerde benzemesine karşın, iki işlemci ailesi birbiri ile uyumlu değildi. Bir yıl sonra 1979'da üretilen, 8086'nın 8-bit veri yoluna sahip sürümü olan 8088, 1981 yılında üretilen IBM PC mikrobilgisayarlarının ilk işlemcisi olmuştur. Kısa sürede endüstrinin 16-bit mikroişlemci standardı olan 8086/8088, günümüze kadar uzanan pek çok ürünü ile x86 ailesi diye adlandırılan mikroişlemci ailesinin çekirdeği oldu.

Daha sonraki yıllarda x86 ailesinin diğer ürünleri, 80186/80188 ve 80286 üretildi. 80186 işlemcisi 8086'nın tümleşik devre üzeri çeşitli çevre birimlerine sahip olan sürümüdür. 80188 işlemcisi ise, 8-bit veri yoluna sahip bir 80186 işlemcisidir. Tasarımlarında fazla çevre birimi istemeyen 80186/80188 işlemcilerinin genelde değişmez bir programla, kontrol uygulamaları içinde yer alarak mikro denetleyici gibi kullanılmaları amaçlanmıştır. Buna rağmen bu iki işlemci yaygın olarak kullanılmamıştır.

8086/8088 işlemcilerinin 1 MB bellek ile sınırlı adres alanı, 1980'lerin ortalarına doğru bir çok uygulama için ciddi bir problem olmaya başlamıştı. Bu yüzden Intel, x86 çekirdeğinin bir üst uyumlu sürümü olan 80286 işlemcisini üretti. Bu işlemci, 16 MB'lık adres alanı ile beraber temel 8086/8088 komut kümesine sahipti. 80286, IBM PC/AT ve orta model PS/2 bilgisayarlarında kullanıldı ve daha önceki 8088 gibi büyük bir başarı kazandı.

Intel için bir sonraki adım, 1985 yılında üretilen, bir tümleşik devre üzerinde gerçek 32-bit CPU olan 80386DX oldu. 80286 gibi bu mikroişlemci de çok yaygın olarak kullanıldı. 1988 yılında, harici 16-bit veri yoluna sahip 80386SX işlemcisi üretildi. 80486, 80386'nın bir üst uyumlu modeliydi. Bütün 80386 programları, 80486 makinelerinde bir değişiklik yapılmadan çalışabilecekti. Bu iki işlemci arasındaki temel fark, 80486 ve 80386'nın özelliklerine ek olarak, yardımcı işlemcisi olan bir kayan nokta birimine(FPU), 8 KB önbelleğe (cache) ve bir bellek yönetim birimine tümleşik devre üzerinde sahip olmasıdır. Ayrıca bir 80486, 80386'dan çok daha hızlıdır.

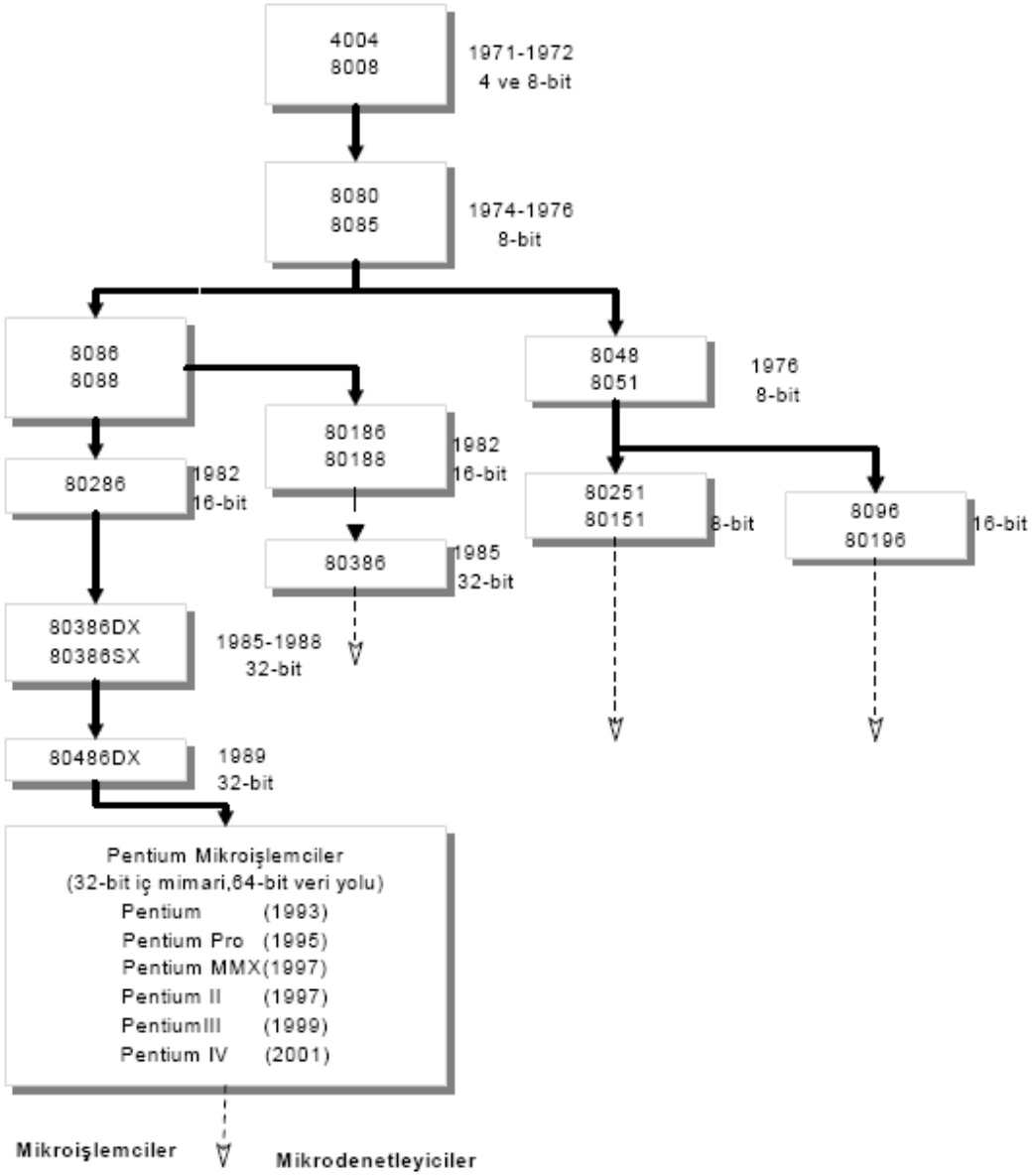
1993 yılında piyasaya sürülen Pentium, temel mimari olarak çok farklı bir mikroişlemci olmayıp, Intel'in yaklaşık her 2-3 yılda ürettiği yeni bir x86 işlemcisidir. Bu yapı IA-32 (Intel Architecture-32) olarak belirtilen 80386/80486 ile başlayan 32-bit mimarinin bir uzantısıdır. 1993 yılından sonraki Pentium işlemcilerinin hızları artmış, üzerlerindeki önbellek yapıları değişmiş ve kapasiteleri artmıştır. Bazılarında tümleşik devre üzeri komut, bazılarında multi medya desteği

sağlanmıştır. Ayrıca, yeni Pentium mimari yapılarında ileri düzeyde, detay farklılıkları da vardır. Bütün bu farklılıklara rağmen, 1978 yılından başlayan 8086/8088 işlemcilerindeki x86 çekirdeği, bu işlemcilerde de bulunmaktadır. Yeni işlemcilerde çalışacak eski programların uyumluluğu için bu mimari gelişim, Intel'in bütün x86 işlemcilerinde sağlanmıştır. Pentium işlemcisiyle x86 ailesinin veri yolu uzunluğu 64-bit olmuştur. Intel, Pentium ile RISC mimari tasarım kavramlarından olan superskalar mimariyi kullanmaya başladı. Diğer bir RISC mimari birimi olan dallanma tahmini donanım yapısı, JUMP ve CALL komutlarıyla yapılan dallanmalarda önemli zaman kazançları sağlayarak çalışma performansını arttırdı. Bu işlemcide ayrıca yürütme performansını önemli olarak etkileyen tümleşik devre üzerinde birinci seviye (L1), ayrı 8 KB kod ve 8 KB veri önbellekleri bulunur.

Pentium Pro; 8086/8088, 80286, 80386, 80486 ve Pentium işlemcilerinden sonra gelen 6. nesil olduğu için, ilk çıktığı sıralarda P6 kod adıyla anılmış ve önemli mimari ekler sunmuştur. P6 mimarisi, dinamik yürütme teknolojisi olarak belirtilen çoklu dallanma tahmini, veri akış analizi ve tahmini yürütme mimari yapılarını içermektedir. Pentium Pro'ya dört yeni adres hattı daha eklenerek adres yolu 36-bit yapıldı. Bu sayede doğrudan adreslenebilir adres alanı 4 GB'tan 64 GB'a artırılmış oldu. Intel firması ilk kez 256 K, 512 K veya 1MB olabilen L2 önbelleğini Pentium Pro işlemcinin üzerine yerleştirdi. Pentium gibi Pentium Pro da 8 KB'ı kod, 8 KB'ı veri için toplam 16 KB'lık L1 önbelleğine sahiptir. Intel firması bir PC'ye DSP özelliği kazandırmak için MMX olarak adlandırılan bir teknolojiyi, Pentium işlemcilerine 1997'den itibaren koymaya başladı. MMX teknolojisi multi medya işlemleri için 57 tane yeni komut sunmaktadır.

Intel Pentium II işlemcisi , Pentium Pro ve MMX teknolojilerinin birleşimi ile üretildi. Bu işlemcide bulunan 32 KB (16 KB/16KB) L1 önbellek, yoğun olarak kullanılan veriye hızlı erişimi sağlar. Ayrıca tümleşik devre üzerinde 512 KB'tan başlayan L2 ön belleği bulunur. Intel daha ucuz PC'ler ve sunucu makineleri için piyasaya ucuz (celeron) ve pahalı (xeon) Pentium II tabanlı iki farklı mikroişlemci sundu. Bu piyasa yaklaşımı daha sonraki Pentium III ve diğer ürünlerde de devam etmiştir.

Pentium III mikroişlemcisi 1999 yılının başında Intel tarafından piyasaya sunulmuştur. Pentium III ile gelen önemli bir yenilik, "Internet Streaming SIMD Extensions" olarak adlandırılan bir yapıdır. Bu mimari yapı ile işlemciye, ileri görüntü işleme, 3D, ses ve video ses tanıma gibi uygulamalarda kullanılabilecek 70 tane yeni komut eklenmiştir. Pentium III ayrıca P6 mikro mimarisinin dinamik yürütme, çoklu dallanma tahmini, veri akışı analizi ve tahmini yürütme çok işlemli sistem yolu ve Intel MMX teknolojisini içerir. Pentium III, PC ve Internet hizmetleri ve ağ erişim güvenliği için planlanan yapı bloklarından ilki olan işlemci seri numarasını sunar. Pentium IV, Intel'in 1995'ten beri tamamen yenilenmiş x86 mikroişlemcisidir. Şekil 2.6'da Intel mikroişlemcilerin gelişimi ve Tablo 2.1'de Intel'in mikroişlemci ailesi temel özellikleri görülmektedir.



Şekil 2.6. Intel mikroişlemcilerin gelişimi [5]

Tablo 2.1. Intel'in mikroişlemci ailesi temel özellikleri [5]

İşlemci	Yıl	Saklayıcı/Veri YoluGenişliği	Adres Alanı	Önemli Özellikler
4004	1971	4\4	640 byte	İlk mikroişlemci,2300 transistör,10 mikron
8008	1972	8\8	16K	İlk 8-bit işlemci,108Khz
8080	1974	8\8	64K	İlk genel amaçlı CPU, 6 mikron,6000 transistör
8085	1976	8\8	64K	Gelişmiş 8080,6200 transistör
8086	1978	16\16	1M	İlk 16-bit CPU, 5-10MHz,29000 transistör,3 mikron
8088	1979	16\8	1M	1981'de üretilen IBM PC'deki ilk işlemci,8-bit veri yolu 8086
80186	1982	16\16	1M	8086+I/0
80188	1982	16\8	1M	8088+ I/0
80286	1982	16\16	16M/1G	134000 transistör, 1.5 mikron,IBM PC/AT'nin i,lk işlemcisi
80386DX	1985	32\32	4G/64T	Intel'in ilk 32-bit işlmecisi,275000 transistör,1 mikron
80386SX	1988	32\16	4G/64T	80286 yoluna sahip 80386
80486DX	1989	32\32	4G/64T	80386+FPU+cache,1.2 milyon transistör,0.8 mikron
Pentium	1993	32\64	4G/64T	3.1 milyon transistör,0ç8 mikron,60-200 MHz,superscalar mimari
Pentium Pro	1995	32\64	64G/64T	5.5 milyon transistör,0.32 mikron, tümdevre üzeri L2 cache, P6 mimarisi: çoklu dallanma tahmin, veri akışı analizi ve tahmini yürütme
Pentium MMX	1997	32\64	4G/64T	4.5 milyon transistör, multi-medya ekleri
Pentium II	1997	32\64	64G/64T	7.5 milyon transistör,0.25 mikron, 300-450 MHz,MMX+ Pro Teknolojisi
Pentium III	1999	32\64	64G/64T	9.5 milyon transistör,0.18 mikron,450-500 MHz, 3D grafikler ve daha fazla multi-medya desteği

2.5.2 Motorola ailesinin gelişimi

Intel, 8080 işlemcisini duyurduktan kısa bir süre sonra, rakip firma olan Motorola, 6800 mikroişlemcisini üretti. 6800 işlem kapasitesi olarak 8080 ile karşılaştırılabilir güce sahipti ve 1970 yıllarında endüstride yaygın olarak kullanıldı. Motorola daha sonra, 6800 ile uyumlu ve 16-bit aritmetik işlem desteğine sahip 6809 işlemcisini üretti.

Motorola, 1979 yılında, daha önceden yapılmamış, bugün bile çok az firmanın yapabileceği bir şey yaptı: 6800 ve 6809 ile uyumlu olmayan tamamıyla yeni bir CPU olan 68000 mikroişlemcisini sundu. Bu tasarım ile amaçlanan, 8086'ın önüne geçmek ve daha değişik bir kitleye hitap etmekti. Veri yolu 16-bit olmasına karşın, bütün iç mimari 32-bit saklayıcılara sahipti.

68000'in gelişmiş özelliklerinden dolayı, sistem programcıları, UNIX gibi karmaşık işletim sistemlerini, 68000 işlemcili makinelerde gerçekleştirmek istediler. Bu tür işletim sistemleri için, görüntü (virtual) hafıza, çok-kullanışlı / çok görevli (multi-user /multi-tasking) program desteğinin işlemci donanımı tarafından sağlanıyor olması gerekiyordu. 68000 bazı destekleri kısmen sağlamasına rağmen tam değildi. Motorola bu problemi, gerekli özelliklere sahip 68010 mikroişlemcisi ile çözdü. Kısa bir süre sonra, daha fazla (2 gigabyte) adres alanına sahip 68012 mikroişlemcisini üretti.

Motorola gerçek 32-bit 68020 işlemcisini sunduğunda, bu iki tüm devre ölmüş oldu. Bu işlemci 32-bit veri yoluna, 32-bit çarpma ve bölme komutlarına sahipti. 68020'in halefi olan 68030, 68020'yi içermekte ve ayrıca aynı tüm devre üzerinde hafıza yönetim birimine de sahipti.

68040, 80486 gibi, tüm devre üzerinde FPU, hafıza yönetim birimi ve ön hafıza içermektedir. Yaklaşık olarak 68040 ve 80486 eşit karmaşıklığa sahip olduklarından, tüm devre üzerinde yaklaşık olarak aynı sayıda (68040 için 1.16 milyon ve 80486 için 1.2 milyon) transistor içermektedir. Motorola ailesinin bir özeti Tablo 2.2'de verilmiştir.

Tablo 2.2. Motorola mikroişlemci ailesi ve temel özellikleri [5]

İşlemci	Yıl	Saklayıcı/Veri Yolu Genişliği	Adres Alanı	Önemli Özellikler
6800	1974	8 / 8	64K	Endüstride çok kullanıldı
68000	1979	32 / 16	16M	Macintosh'ün ilk işlemcisi
68008	1982	32 / 8	4M	8-bit veri yollu ürün
68010	1983	32 / 16	16K	Görüntü hafıza desteği
68012	1983	32 / 16	2G	68010'un geniş adres alanlı sürümü
68020	1984	32 / 32	4G	Gerçek 32-bit CPU, iş istasyonlarında çok kullanıldı
68030	1987	32 / 32	4G	tüm devre-üzeri hafıza yönetim birimi
68040	1989	32 / 32	4G	68030'un daha hızlı sürümü
PowerPC	1993	32 / 32	4G	Motorola+IBM+Apple, RISC mimarisi

2.6 Mikroişlemcilerin Uygulama Alanları

Mikroişlemcilerin yeteneklerinin zamanla artması, kullanım alanlarında çeşitlik ve yaygınlığa neden olmuştur. Mikroişlemcilerin kullanım alanlarını iki genel başlıkta toplayabiliriz.

2.6.1. Atanmış bilgisayar uygulamaları

Belli bir amaca ulaşmak için gerçeklenmiş ve bilgisayar içeren dizgelere, atanmış bilgisayarlı dizgeler adı verilmektedir Atanmış bilgisayar uygulamalarına bazı örnekler aşağıda sıralanmıştır:

- Bilgisayar destekli üretim tezgahları
- Mikroişlemci kullanan otomatik çamaşır makineleri
- Mikroişlemci içeren mikrodalga fırınlar
- İklimlendirme dizgeleri
- Bilgisayarlı otomobil yakıt dizgeleri

Verilen örneklerden de anlaşılacağı gibi, atanmış bilgisayar, ilişkili olduğu dizge içinde gömülü olarak yer almaktadır. Bu nedenle, çoğu kez kullanıcı tarafından fark edilmez.

2.6.2 Genel amaçlı bilgisayar uygulamaları

Genel amaçlı bilgisayar, standart bir donanım ile kullanıcıya sunulan bilgisayardır. Bu tür bilgisayarlara örnek olarak:

- Ana bilgisayarlar
- İş istasyonları
- Bireysel bilgisayarlar (PC) verilebilir.

Mikroişlemciler, bilgisayarın her iki tür uygulaması için de önemli ivme kaynağı olmuştur, Mikroişlemciler üretilmeye başlamadan önce, atanmış bilgisayar uygulamaları yok denecek kadar azdı. Mikroişlemci öncesi bilgisayarların büyük boyutta olmaları ve pahalı olmaları, atanmış bilgisayar uygulamalarına olanak vermemiştir. Örneğin, mikroişlemci öncesinde, bilgisayarla yönetilen bir çamaşır makinesi düşünülemezdi.

Geçen 25 yıl içinde mikroişlemci tabanlı dizge tasarımı uygulamalarının sayısı çok hızlı artmıştır. Hemen hemen her konuda mikroişlemcili dizge uygulamasına rastlanmaktadır. Mikroişlemcili dizge tasarımında, tasarıma uygun mikroişlemci seçimi yapılmaktadır Örneğin, bir çamaşır makinesi, mikrodalga fırın veya benzer ölçekte uygulamalar için 8 bitlik mikroişlemciler yeterli olmaktadır Buna karşın bir üretim tezgahının denetimi yada bir robot denetimi için 16 hatta 32 bitlik mikroişlemciler gerekmektedir. Uygulamaların çeşitliliği nedeniyle, değişik kelime uzunluğu (8, 16, 32 bit) olan mikroişlemciler üretilmektedir.

Sadece 8 bitlik mikroişlemcilerin üretildiği yıllarda, mikroişlemci tabanlı, genel amaçlı bilgisayar örneklerine rastlanmıştır. Bu dönemde gerçeklenen ve adına "ev bilgisayarı" denilen ilginç örnekler hatırlardadır. 16 bitlik mikroişlemcilerin

üretilmesinin ardından bireysel bilgisayarlarda önemli bir atılım gözlenmiştir. Bu dönemde, IBM uyumlu PC'ler ve Apple uyumlu bilgisayarlar yaygın biçimde kullanılmıştır. IBM uyumlu PC'lerde Intel ailesinin mikroişlemcileri (I8088, I80186, I80286, I80386, I80486, Pentium) kullanılırken, Apple uyumlu bilgisayarlarda Motorola ailesinin mikroişlemcilerinin (MC68000, MC68020, MC68030, MC68040) kullanılması yeğlenmiştir. Mikroişlemciler, iş istasyonlarında da kullanılmıştır. İş istasyonlarının üretiminde belli süre Motorola MC 680X0 ailesinin işlemcileri kullanılmıştır.

BÖLÜM 3. 16-BİTLİK MİKROİŞLEMCİNİN TASARIMI

3.1 Çok Hızlı Tümleşik Devre Donanım Tanımlama Dili (VHDL)

Elektronik sistemlerin karmaşıklığının artması tasarım yöntemlerinin de gelişmesini gerektirmiştir. Bu sebeple, geleneksel "kağıt ve kalem kullanarak tasarımı yap" ve "devreyi kurarak denemeleri yap" yöntemlerinin yerini "tanımla ve sentezle" yöntemleri almıştır [6]. Tanımlama ve sentezleme yöntemleri, donanım tanımlama dillerinin (Hardware Description Languages-HDLs) ortaya çıkmasına neden olmuştur. VHDL (Very High Speed Integrated Circuit Hardware Description Language) günümüzde kullanılan en popüler donamım tanımlama dillerinden birisidir [7].

VHDL, büyük ölçekli dijital tasarımların dokümantasyonu, doğrulanması ve sentezlenmesi işlemlerini gerçekleştirmek için kullanılır. Aynı VHDL kodları ile bu üç farklı işlemin gerçekleştirilebilmesi, tasarım süresi ve kolaylığı açısından büyük avantajlar sağlamaktadır. VHDL'in avantajları aşağıda verilmiştir.

- Güç ve Esneklik: VHDL tasarımları yazılım tabanlı olduğundan dolayı birçok fonksiyon tek bir donamım ile gerçekleştirilebilir.

- Çipten Bağımsız Tasarım: Özel işlevleri yerine getiren birçok entegrenin VHDL ile gerçeklenmesi mümkün olduğu için tasarımlarımız çipten bağımsızdır.

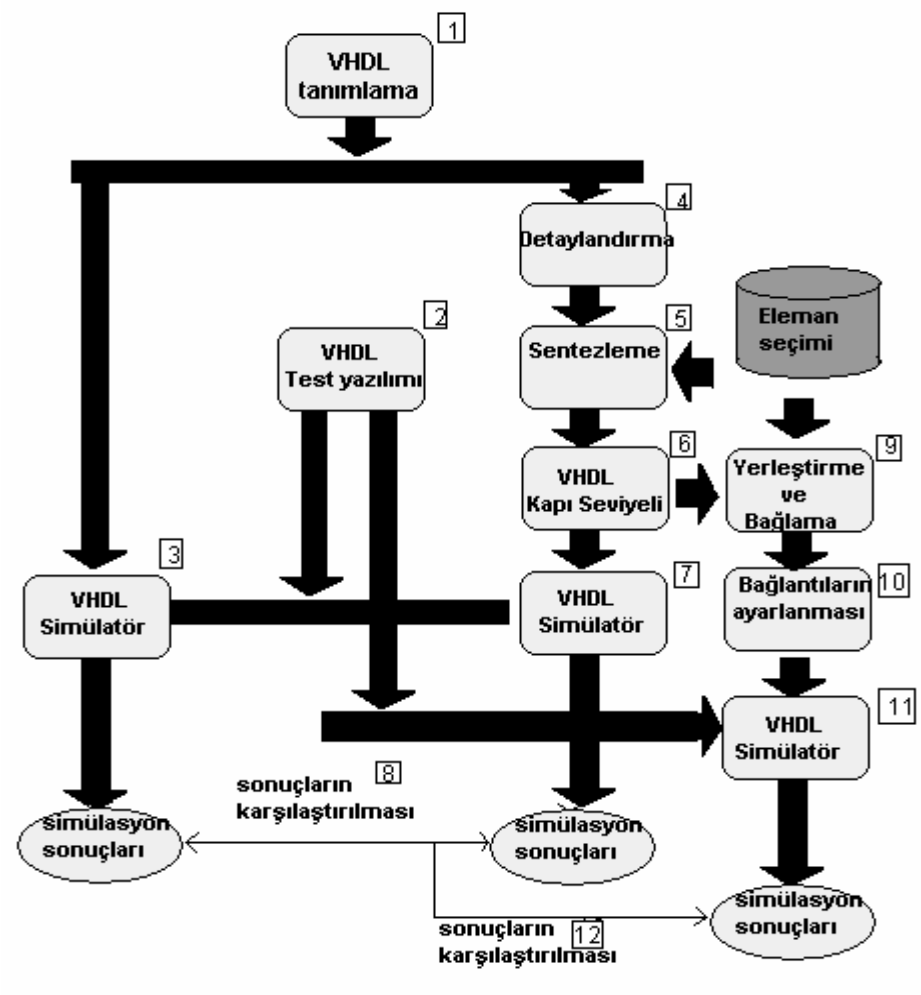
- Taşınabilirlik: Yapılan tasarım bir dosya olduğundan mevcut başka bir tasarım dosyasının içine veya bir çipin içine eklenebilir. Böylelikle yapılan her tasarım farklı bir modül olarak düşünülebilir.

- Test Edilebilirlik: Tasarımı yapılan VHDL devreleri emülatörler yardımıyla gerçek zamanlı olarak, simülatörler yardımıyla da sanal olarak test edilebilir.

- Piyasaya Hızlı Çıkış ve Düşük Maliyet: Mevcut bir sistem üzerinde yapılması istenen bir değişiklik donanım olarak değil de yazılım olarak yapıldığından tasarım süresi kısa sürmektedir ayrıca birçok entegre VHDL çipleri ile tasarlanabildiğinden tasarım maliyeti düşüktür.

3.2. Tasarım Aşamaları

16-bitlik mikroişlemcinin tasarımı VHDL yardımıyla gerçekleştirilmiştir. Şekil 3.1 'de tasarımın gerçekleştirilme aşamaları görülmektedir.



Şekil 3.1. 16-bitlik mikroişlemcinin tasarım aşamaları [8]

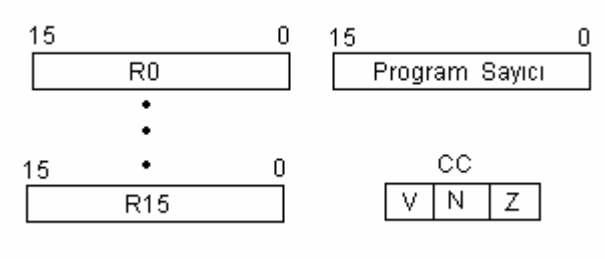
1. Aşamada; VHDL dilinde tasarımın tanımlanması yapılır. (Xilinx ISE 6.3i)
2. Aşamada; VHDL test yazlımı oluşturulur ve bu yazılım sayesinde tanımlanan fonksiyonların test işlemi gerçekleştirilebilir.
3. Aşamada; test yazılımının fonksiyonel simülasyonu gerçekleştirilir. (ModelSim)
4. Aşamada; tanımlanan tasarımın yapısal optimizasyon işlemi sentezleyici program tarafından gerçekleştirilir.
5. Aşamada; seçilmiş olan donanımın zamanlama ve kapasite sınırlamalarını karşılayacak şekilde kapı-seviyeli optimize edilmiş bir tanımlama oluşturulur.
6. Aşamada; kapı-seviyeli VHDL tanımlaması oluşturulur.
7. Aşamada; 2. aşamadaki test yazılımı ile tekrar simülasyon işlemi gerçekleştirilir.
8. Aşamada; gerçekleme aşamasının doğruluğunun testi için 7. aşamadaki kapı-seviyeli simülasyon sonuçları ile 3. aşamadaki VHDL tanımlama simülasyon sonuçları karşılaştırılır.
9. Aşamada; netlistler (bağlantılar) oluşturularak tasarımın yerleştirme ve bağlama işlemleri gerçekleştirilir.
10. Aşamada; kullanılmayan veya gereksiz kısımlar elenir.
11. Aşamada; tasarımın hedef elemana yerleştirilmeye hazır hale geldikten sonraki simülasyon işlemi gerçekleştirilir (Place & Route işleminden sonraki simülasyon).
12. Aşamada ise 11. aşamada gerçekleştirilmiş olan simülasyon ile 3. aşamada gerçekleştirilmiş olan simülasyon sonuçları karşılaştırılarak fiziksel gerçekleme işleminin doğrulanması işlemi gerçekleştirilir.

Bu çalışmadaki 16-bitlik mikroişlemcinin tasarım aşamaları VHDL yardımıyla gerçekleştirilmiştir. Tasarımın tanımlama işlemi Xilinx ISE 6.3i geliştirme yazılımı içerisindeki VHDL editörü ile yapılmıştır. Tanımlama işleminin fonksiyonelliğinin doğrulanması işlemi için yapılan simülasyon, Xilinx ISE 6.3i yazılımı ile entegre olan ModelSim 5.8v simülatörü yardımıyla gerçekleştirilmiştir. Hedef elemana yerleştirilmeye hazır hale geldikten sonraki simülasyon (Post-layout simulation) işlemi için de ModelSim simülatörü kullanılmaktadır.

3.3. 16-Bitlik Mikroişlemcinin Tasarım Özellikleri

Bu bölümde 16-bitlik mikroişlemcinin tüm alt birimleriyle birlikte tasarımı sunulmaktadır. Tasarlanan mikroişlemci, 16-bitlik adres yolu ve 16-bitlik veri yoluna sahiptir. 16 tane genel amaçlı saklayıcı, program sayıcı ve 3-bitlik durum saklayıcı içermektedir

3-bitlik durum saklayıcı her aritmetik ve lojik işlemden sonra güncellenmektedir. Z (zero) biti aritmetik veya lojik işlemin sonucunun sıfır olduğunu, N (negative) biti aritmetik işlemin sonucunun negatif bir sayı olduğunu ve V (overflow) biti aritmetik işlem sonucunda bir taşma olduğunu göstermektedir. Mikroişlemciye ait saklayıcılar Şekil 3.2'de gösterilmiştir.



Şekil 3.2. Mikroişlemciye ait saklayıcılar

Mikroişlemci, komutların saklandığı, 16-bit kelime uzunluğuna sahip, 16-bitle adreslenen bir harici hafıza içermektedir. Bütün komutlar 16-bit uzunluğundadır. Program sayıcı bir sonra işlenecek olan komutun adresini saklamaktadır ve her bir komutun işletilmesinden sonra içeriği bir artmaktadır.

16-bitlik mikroişlemcinin tasarımında, mikroişlemciyi oluşturan birimler birbirinden bağımsız olarak ele alınmış ve her bir birim bireysel olarak tasarlanarak simüle edilmiştir. Tasarımı oluşturan birimler / fonksiyonel bloklar aşağıda verilmiştir.

➢ 16-bitlik mikroişlemcinin en üst birimi (Mib16): Pin bağlantılarının olduğu birim.

o Aritmetik ve lojik birim (Alu_16): Bu birimde aritmetik ve mantık işlemleri geçekleştirilmektedir.

o Tamponlama birimi (Buffer_16): Bu birim üç-durumlu tampon görevi yaparak yol geçişlerini kontrol etmektedir.

o Kontrol birimi (Controller): Kontrol sinyallerinin üretildiği birim.

o Tutma birimi (Latch): Verilerin geçici olarak tutulduğu birim.

o Tutma-Tamponlama birimi (Latch_Buffer_16): Tamponlama ve tutma işleminin aynı anda yapıldığı birim.

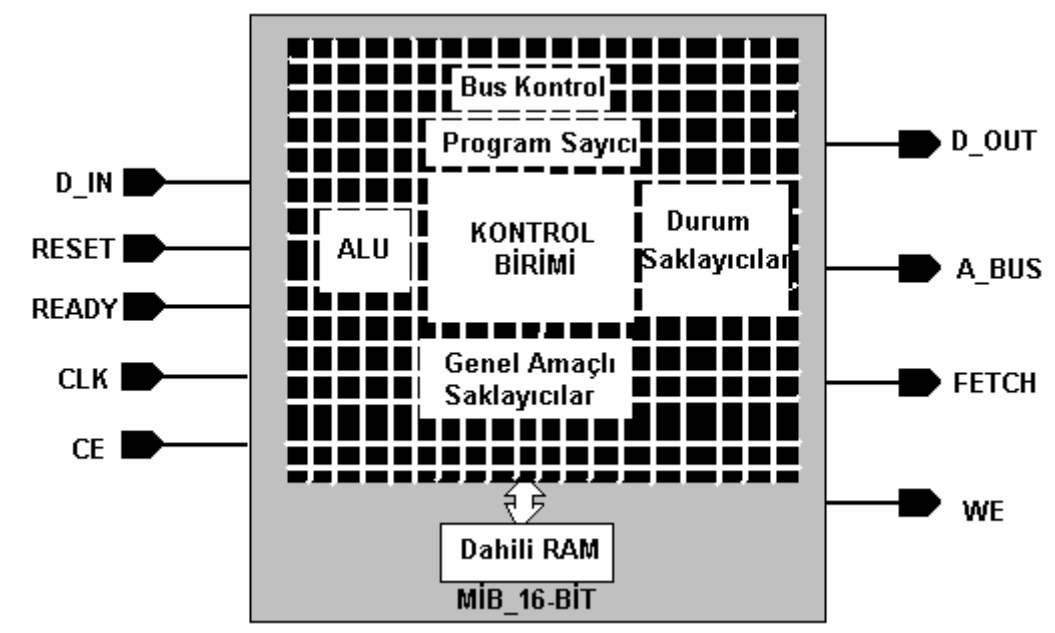
o Multiplexer birimi (Mux2): Seçici devre.

o Program sayıcı devre (PC_reg): Bir sonraki komutun adresini içeren birim.

o 16-bitlik yeniden yazılabilir okunabilir kaydedici (reg_file_16_rrw):

o 4-bitlik sinyalden 16-bitlik sinyal üreten birim (signext_4_16)

➢ Dahili 64Kx16 RAM



Şekil 3.3. 16-bitlik mikroişlemcinin blok diyagramı

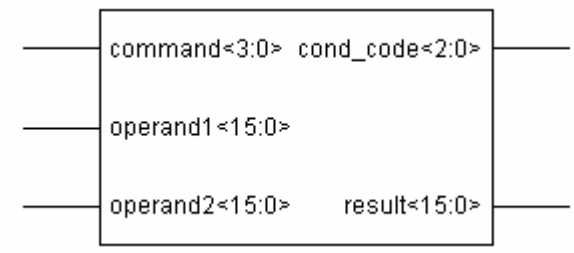
Şekil 3.3'de 16-bitlik mikroişlemcinin blok diyagramı görülmektedir. Her bir birim modüler olarak tasarlanmış olduğundan ihtiyaca veya uygulamanın çeşidine göre gereksiz birimler tasarımdan kolaylıkla çıkarılabilir veya RAM'in büyüklüğü, komut seti sayısı isteğe göre belirlenebilir..

16-bitlik mikroişlemcinin tasarımı alt birimlerden üst birimlere doğru belirli bir hiyerarşik düzene göre gerçekleştirilmiştir. Tüm birimler tasarlandıktan sonra birleştirme işlemi gerçekleştirilerek sistem oluşturulmuştur.

Bundan sonraki başlıklarda 16-bitlik mikroişlemciyi oluşturan birimlerin tasarım özellikleri ve simülasyon sonuçları verilecektir.

3.4. Aritmetik ve Lojik Biriminin (ALU) Tasarımı

Şekil 3.4'de ALU biriminim şematik gösterimi görülmektedir. Giriş uçları için gerekli olan sinyaller genel amaçlı kaydedicilerden, kontrol biriminden ve durum saklayıcısından gelmektedir. Çıkış uçları ise kaydediciler ve durum saklayıcısına bağlıdır.



Şekil 3.4. ALU biriminin şematik gösterimi

command : ALU biriminin yürüteceği komutu gösteren giriş sinyalidir. Kontrol biriminden gelmektedir.
operand1 : ALU biriminin işleme koyacağı verilerden ilkini gösteren giriş sinyalidir. Tasarlanan mikroişlemci 16-bitlik veri yoluna sahip olduğu için ALU birimi de 16-bitlik verileri işleyecek şekilde tasarlanmıştır.

operand2: ALU biriminin işleme koyacağı verilerden ikincisini gösteren giriş sinyalidir.
result : ALU biriminin gerçekleştirdiği işleminin sonucunu gösteren çıkış sinyalidir.
con_code : Durum kodlarını içeren çıkış sinyalidir. Durum saklayıcısına bağlıdır. Sonuca bakarak sonucun sıfır ve/veya negatif olup olmadığına ve işlem sonucunda taşma olup olmadığına karar verir.

Aşağıda ALU biriminin VHDL dilindeki "entity" tanımlaması görülmektedir.

```
entity ALU_16 is
port (  operand1 : in   std_logic_vector (15 downto 0);
         operand2 : in   std_logic_vector (15 downto 0);
              result : out std_logic_vector (15 downto 0) bus;
        cond_code : out std_logic_vector (2 downto 0);
        command : in   ALU_command);
end ALU_16;
```

3.4.1. ALU biriminin yürütebildiği komutlar

ALU birimi 12 adet komutu gerçekleştirebilmektedir. Bunlardan 18 tanesi aritmetik komut, 4 tanesi ise lojik komuttur.

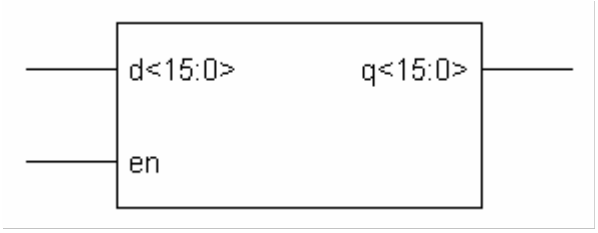
Tablo 3.1'de ALU biriminin gerçekleştirebildiği aritmetik ve lojik komutlar görülmektedir.

Tablo 3.1. ALU biriminin gerçekleştirebildiği aritmetik ve lojik komutlar

Komut	**Adı**	**Fonksiyonu**	**Opkod**
Add	Toplama	r3 ← r1+r2	0000
Sub	Çıkarma	r3 ← r1-r2	0001
Mul	Çarpma	r3 ← r1*r2	0010
Div	Bölme	r3 ← r1/r2	0011
Addq	İvedi Toplama	r3 ← r1+i8	0100
Subq	İvedi Çıkarma	r3 ← r1-i8	0101
Mulq	İvedi Çarpma	r3 ← r1*i8	0110
Divq	İvedi Bölme	r3 ← r1/i8	0111
Land	Lojik AND	r3 ← r1&r2	1000
Lor	Lojik OR	r3 ← r1!r2	1001
Lxor	Lojik XOR	r3 ← r1+r2	1010
Lmask	Lojik Maskeleme	r3 ← r1&-r2	1011

3.5. Tutucu (Latch) Biriminin Tasarımı

Şekil 3.5'de Latch biriminin şematik gösterimi görülmektedir. Bu birim verilerin geçici olarak saklandığı yerdir.



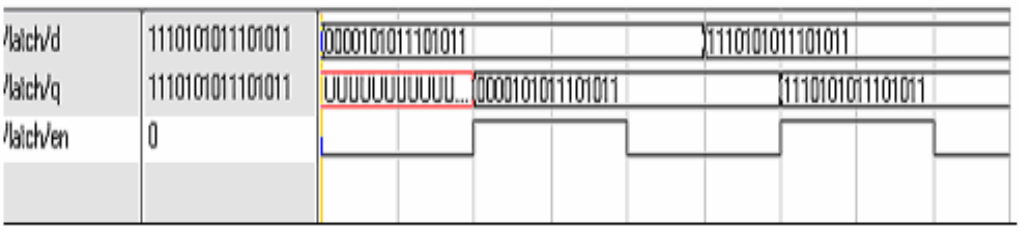
Şekil 3.5. Latch biriminin şematik gösterimi

"d" sinyali giriş verisini, "q" sinyali çıkış verisini ve "en" sinyali ise çip yetkilendirme sinyalini göstermektedir. "en" sinyali "1" olduğunda "d" giriş verisi "q" çıkışına verilir. Giriş verisi değişmedikçe çıkış verisi aynı kalır.

Aşağıda Latch biriminin VHDL dilindeki "entity" tanımlaması görülmektedir. "generic" olarak tanımlanan veri yolu değiştirilerek Latch biriminin saklayabileceği veri uzunluğu belirlenebilir.

```
entity latch is
generic (width : positive );  --Latch biriminin veri saklama uzunluğu
port (  d : in std_logic_vector(width-1 downto 0);
          q : out std_logic_vector(width-1 downto 0);
         en : in std_logic);
end latch;
```

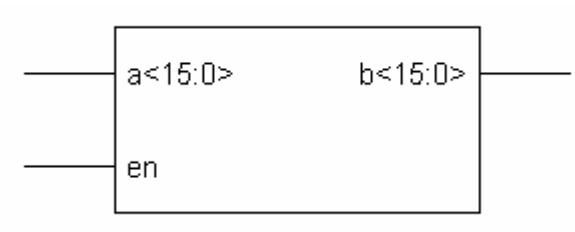
Latch birimine ait simülasyon sonuçları Şekil 3.6'da görülmektedir.



Şekil 3.6. Latch biriminin simülasyon sonuçları

3.6 Tampon (Buffer) Biriminin Tasarımı

Şekil 3.7'de Buffer biriminin şematik gösterimi görülmektedir. Bu birim yol kontrolünü sağlamaktadır.



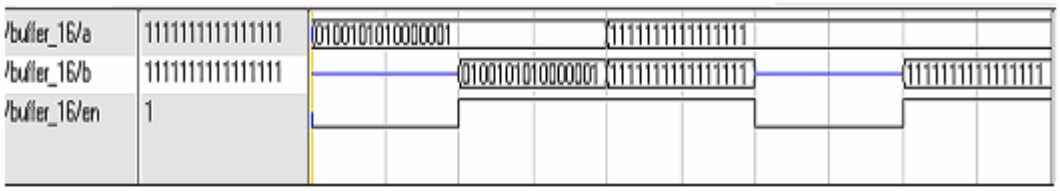
Şekil 3.7. Buffer biriminin şematik gösterimi

"a" sinyali giriş verisini, "b" sinyali çıkış verisini ve "en" sinyali ise çip yetkilendirme sinyalini göstermektedir. "en" sinyali "1" olduğunda "a" giriş verisi "b" çıkışına verilmekte, "0" sıfır olduğunda "b" çıkış sinyali yüksek empedans konumuna getirilerek yoldan yalıtılmaktadır.

Aşağıda Buffer biriminin VHDL dilindeki “entity” tanımlaması görülmektedir.

```
entity buffer_16 is
port (  a : in   std_logic_vector (15 downto 0);
          b : out std_logic_vector (15 downto 0) bus;
        en : in   std_logic);
end buffer_16;
```

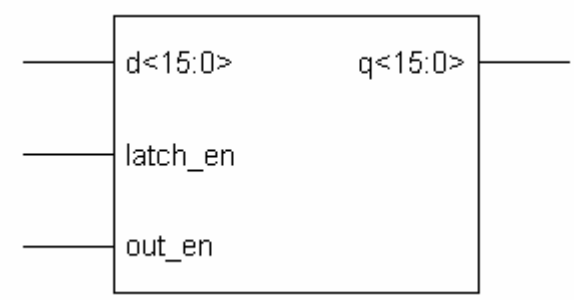
Buffer birimine ait simülasyon sonuçları Şekil 3.8’de görülmektedir.



Şekil 3.8. Buffer biriminin simülasyon sonuçları

3.7 Tutma-Tamponlama Biriminin Tasarımı

Şekil 3.9’da Tutma-Tamponlama biriminin şematik gösterimi görülmektedir. Bu birim hem saklama hem de tamponlama işlemini gerçekleştirmektedir.



Şekil 3.9. Tutma-Tamponlama biriminin şematik gösterimi

d: 16-bitlik giriş sinyali

q: 16-bitlik giriş sinyali

latch_en: Giriş verisini “d” sinyaline aktarmaya yarayan giriş yetki sinyali

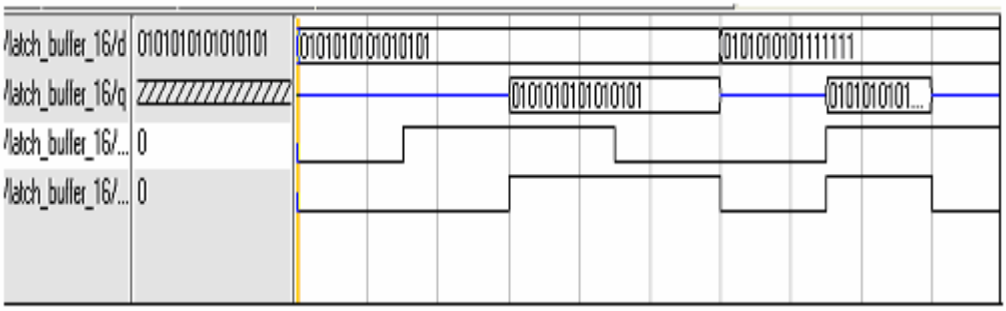
out_en: çıkış verisini “q” sinyaline aktarmaya yarayan çıkış yetki sinyali

“latch_en” sinyali “1” olduğu sürece girişe gelen veri “d” sinyaline alınır; “out_en” sinyali “1” olduğunda “d” sinyali “q” çıkış sinyaline aktarılır. “out_en” sinyali “0” olduğu sürece “q” sinyali yüksek empedans konumuna getirilerek yoldan yalıtılmaktadır.

Aşağıda Tutma-Tamponlama biriminin VHDL dilindeki “entity” tanımlaması görülmektedir.

```
entity latch_buffer_16 is
port (        d :  in  std_logic_vector (15 downto 0);
                q : out std_logic_vector (15 downto 0) bus;
     latch_en : in   std_logic;
       out_en : in   std_logic);
end latch_buffer_16;
```

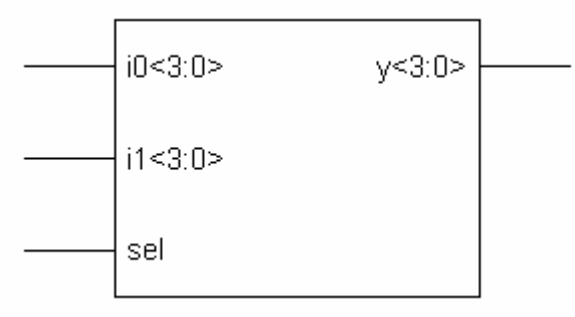
Tutma-Tamponlama birimine ait simülasyon sonuçları Şekil 3.10’da görülmektedir.



Şekil 3.10. Tutma-Tamponlama biriminin simülasyon sonuçları

3.8. Seçici (Mux2) Biriminin Tasarımı

Şekil 3.11'de Seçici biriminin şematik gösterimi görülmektedir. Bu birim seçme işlemini gerçekleştirmektedir.



Şekil 3.11. Seçici biriminin şematik gösterimi

"i0" ve "i1" giriş sinyallerini, "sel" seçme hattını ve "y" çıkış sinyalini göstermektedir. "sel" sinyali, "0" ise "y" sinyaline "i0" sinyali, "1" ise "i1" sinyali aktarılır.

Aşağıda Seçici biriminin VHDL dilindeki "entity" tanımlaması görülmektedir. "generic" olarak tanımlanan veri yolu değiştirilerek Seçici biriminin veri uzunluğu belirlenebilir.

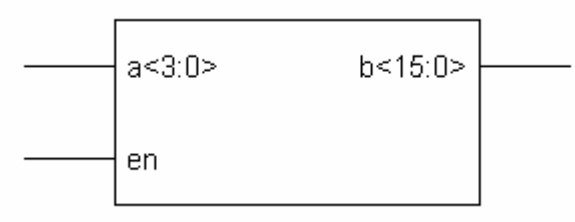
```
entity mux2 is
generic (width : positive);
port (   i0 : in   std_logic_vector(width-1 downto 0);
         i1 : in   std_logic_vector(width-1 downto 0);
          y : out std_logic_vector(width-1 downto 0);
        sel : in   std_logic);
end mux2;
```

Seçici birimine ait simülasyon sonuçları Şekil 3.12'de görülmektedir.

Şekil 3.12. Seçici biriminin simülasyon sonuçları

3.9. Sinyal Genişletme (Signext_4_16) Biriminin Tasarımı

Şekil 3.13'da Sinyal genişletme biriminin şematik gösterimi görülmektedir. Bu birim 4-bitlik veriyi 2'ye tümleme mantığını kullanarak 16-bitlik veriye genişletmektedir.



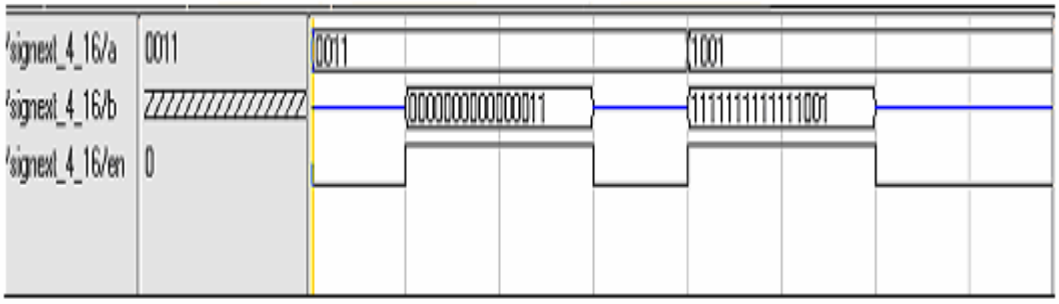
Şekil 3.13. Sinyal Genişletme biriminin şematik gösterimi

Sinyal Genişletme birimi "en" yetki sinyali "1" olduğunda 4-bitlik "a" giriş sinyalinden 16-bitlik "b" çıkış sinyali oluşturur. "en" sinyali "0" olduğu sürece "b" sinyali yüksek empedans konumuna getirilerek yoldan yalıtılmaktadır.

Aşağıda Sinyal Genişletme biriminin VHDL dilindeki "entity" tanımlaması görülmektedir.

```
entity signext_4_16 is
port (  a : in  std_logic_vector (3 downto 0);
          b : out std_logic_vector (15 downto 0) bus;
        en : in std_logic);
end signext_4_16;
```

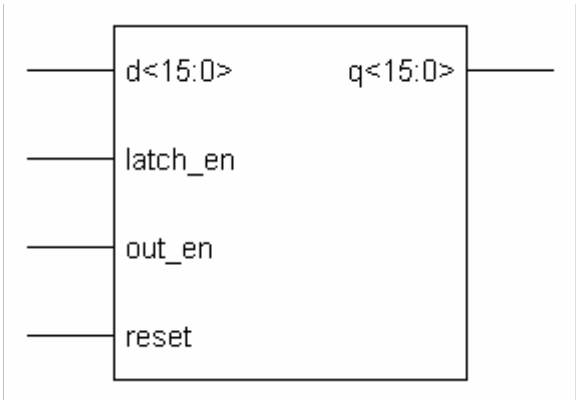
Sinyal Genişletme birimine ait simülasyon sonuçları Şekil 3.14'de görülmektedir.



Şekil 3.14. Sinyal Genişletme biriminin simülasyon sonuçları

3.10. Program Sayıcı (PC_reg) Biriminin Tasarımı

Şekil 3.15'de Program Sayıcı biriminin şematik gösterimi görülmektedir. Bu birim bir sonraki komutun adresini göstermektedir.



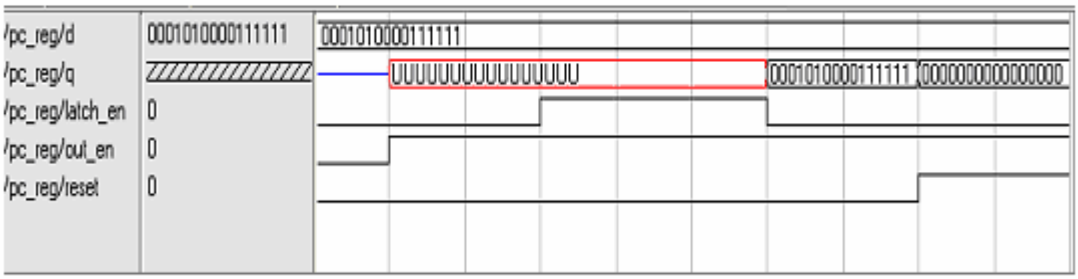
Şekil 3.15. Program Sayıcı biriminin şematik gösterimi

Bu birim Tutma-Tamponlama birimi gibi işlemektedir. Tek fark "reset" sinyali sayesinde birimin içeriğinin sıfırlanabilmesidir.

Aşağıda Program Sayıcı biriminin VHDL dilindeki “entity” tanımlaması görülmektedir.

```
entity PC_reg is
port (          d : in   std_logic_vector (15 downto 0);
                q : out std_logic_vector (15 downto 0) bus;
       latch_en : in   std_logic;
         out_en : in   std_logic;
          reset : in   std_logic);
end PC_reg;
```

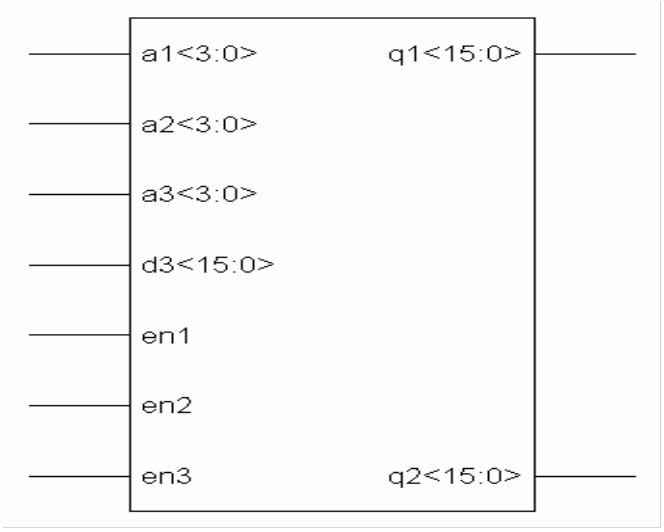
Program Sayıcı birimine ait simülasyon sonuçları Şekil 3.16’da görülmektedir.



Şekil 3.16. Program Sayıcı biriminin simülasyon sonuçları

3.11. Kaydedici (Reg_file_rrw) Biriminin Tasarımı

Şekil 3.17’de Kaydedici biriminin şematik gösterimi görülmektedir. Bu birim işlenecek verilerin ve işlem sonuçlarının saklandığı birimdir.



Şekil 3.17. Kaydedici biriminin şematik gösterimi

"a1"ve "a2" sinyalleri kaydediciden okunacak verinin adresini; "a3" sinyali kaydediciye yazılacak verinin adresini; "d3" sinyali kaydediciye yazılacak 16-bitlik veriyi; "en1", "en2" ve "en3" sinyalleri yetkilendirme sinyallerini; "q1"ve "q2" sinyalleri kaydediciden okunacak 16-bitlik çıkış verilerini göstermektedir.

Aşağıda Kaydedici biriminin VHDL dilindeki "entity" tanımlaması görülmektedir. "generic" olarak tanımlanan veri yolu değiştirilerek Kaydedici biriminin adres uzunluğu belirlenebilir.

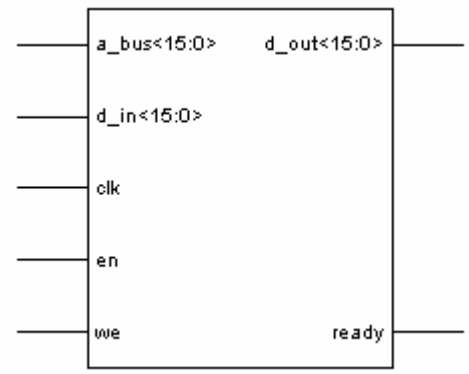
```
entity reg_file_16_rrw is
generic (depth : positive);     -- adres bitlerinin sayısı
port (   a1 : in std_logic_vector (depth-1 downto 0);
            q : out std_logic_vector (15 downto 0) bus;
         en1 : in std_logic;
          a2 : in std_logic_vector (depth-1 downto 0);
          q2 : out std_logic_vector (15 downto 0) bus;
         en2 : in std_logic;
          a3 : in std_logic_vector (depth-1 downto 0);
          d3 : in  std_logic_vector (15 downto 0) ;
         en3 : in std_logic);
end reg_file_16_rrw;
```

Kaydedici birimine ait simülasyon sonuçları Şekil 3.18'de görülmektedir.

Şekil 3.18. Kaydedici biriminin simülasyon sonuçları

3.12 Hafıza (Memory) Biriminin Tasarımı

Şekil 3.19'da Hafıza biriminin şematik gösterimi görülmektedir. Bu birim 16-bitlik komutların saklandığı birimdir. Belirli bir kısmı komutlar için kullanılırken, diğer kısım verilerin saklanması için kullanılabilir.



Şekil 3.19. Hafıza biriminin şematik gösterimi

d_in: 16-bitlik giriş verisidir. Hafıza yazılacak veriyi taşır.

d_out: 16-bitlik çıkış verisidir. Hafızadan okunacak veriyi taşır.

d_bus: Hafızaya yazılacak veya hafızadan okunacak verinin adresini içeren sinyaldir.

clk: Saat frekans sinyalidir. Hafıza birimi mikroişlemciyle senkron çalıştığı için saat frekansı bu birimle aynıdır.

en: yetki sinyalidir. Sinyal "1" olduğunda hafıza biriminden okuma ve bu birime yazma işlemleri gerçekleştirilebilir.

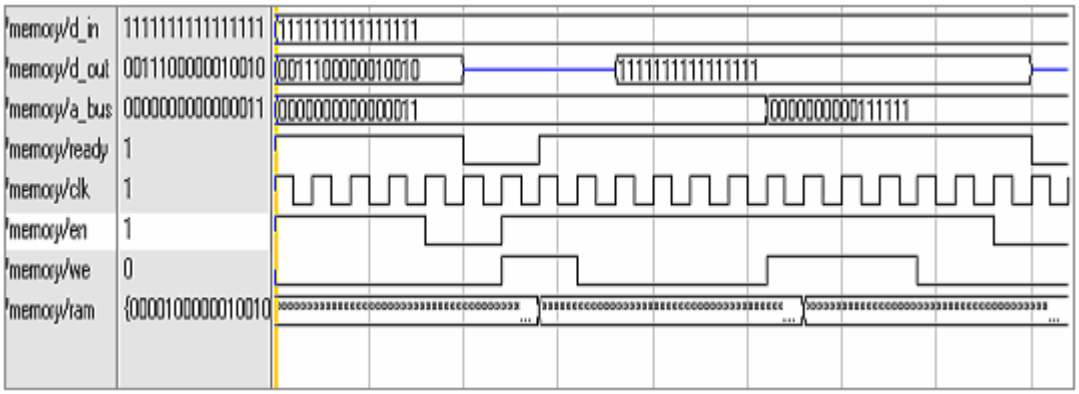
we: Okuma ve yazma yetki sinyali. "we" "1" olduğunda yazma, "0" olduğunda ise okuma işlemi yapılır.

ready: Okuma ve yazma işlemleri tamamlandığında lojik "1" sinyali üretir.

Aşağıda Hafıza biriminin VHDL dilindeki "entity" tanımlaması görülmektedir.

```
entity memory is
port(   d_in : in    std_logic_vector (15 downto 0);
        d_out : out std_logic_vector (15 downto 0) bus;
        a_bus : in  std_logic_vector (15 downto 0);
        ready :out std_logic;
          clk  :in    std_logic;
           en  :in    std_logic;
           we  :in    std_logic);
end memory;
```

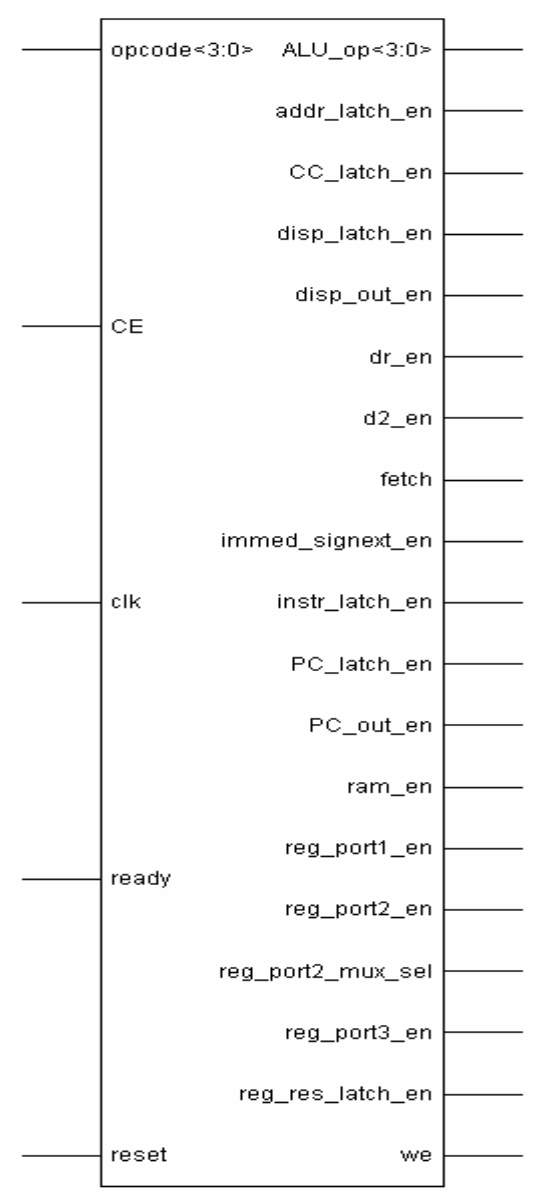
Hafıza birimine ait simülasyon sonuçları Şekil 3.20'de görülmektedir.



Şekil 3.20. Hafıza biriminin simülasyon sonuçları

3.13. Kontrol (Control) Biriminin Tasarımı

Şekil 3.21'de Kontrol biriminin şematik gösterimi görülmektedir. Bu birim mikroişlemcinin doğru bir biçimde çalışabilmesi için kontrol sinyallerini üretmektedir. Bu nedenle bu birim, mikroişlemcinin beyni vazifesini görmektedir.



Şekil 3.21. Kontrol biriminin şematik gösterimi

opcode: Mikroişlemcinin işleyeceği komutu gösteren giriş sinyalidir. Toplam 16 adet komut olduğu için bu sinyal 4-bitliktir.

CE: Mikroişlemciyi yetkilendirme sinyalidir. “CE” sinyali “1” olduğunda mikroişlemci aktiftir.

clk: Mikroişlemciye ait saat frekansıdır. Yükleme yapılacak Spartan-3 bordunun saat frekansı olan 50 MHz’e eşittir.

ready: Hafıza biriminden gelen giriş sinyalidir. Okuma ve yazma işlemlerinin tamamlanıp tamamlanmadığını gösterir.
reset: Mikroişlemcinin içeriğini sıfırlamak için kullanılır.

Alu_op: ALU biriminin hangi komutu işleyeceğini işaret eden çıkış sinyalidir.

fetch: Hafıza biriminden komut mu yoksa verimi okunacağını gösteren çıkış sinyalidir. Bu sinyal “1” olduğunda hafızadan komut okunuyor, “0” olduğunda ise veri okunuyor demektir..

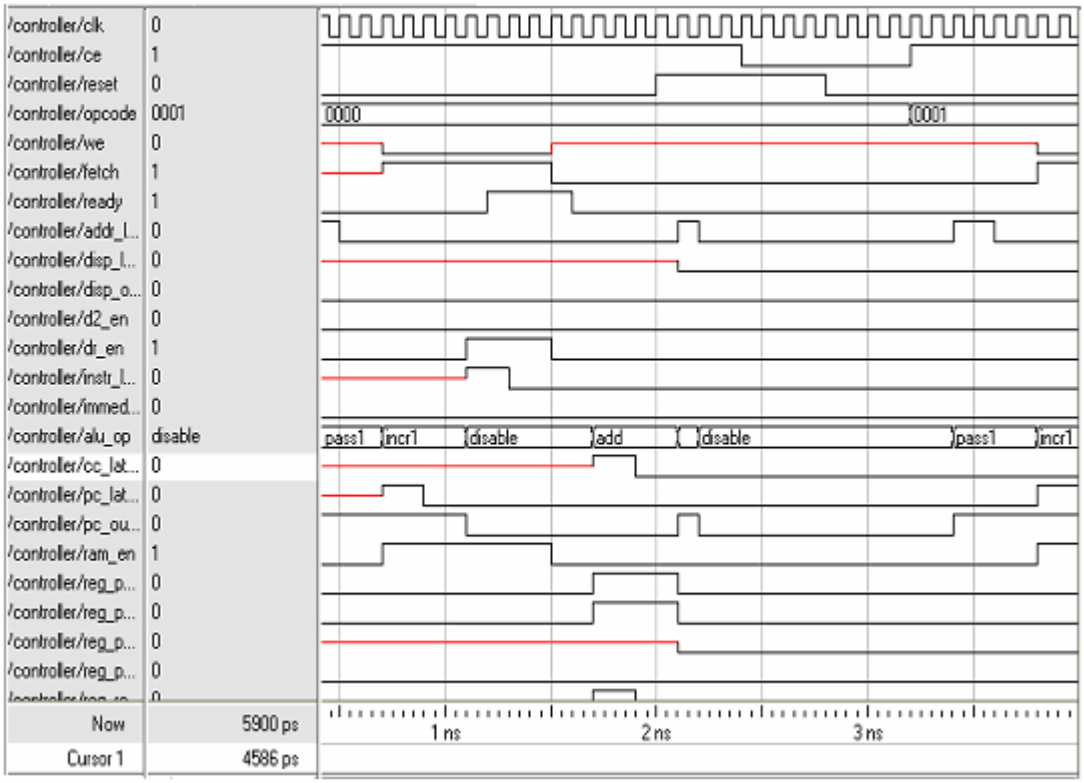
we: Hafızadan okuma işleminin mi yoksa hafızaya yazma işleminin mi yapılacağını gösterir. “we” sinyali “0” ise okuma, “1” ise yazma işlemi yürütülür.

addr_latch_en, disp_latch_en, disp_out_en, d2_en, dr_en, instr_latch_en, CC_latch_en, immed_signext_en, PC_latch_en, PC_out_en, reg_port1_en, ram_en, reg_port2_en,reg_port3_en, reg_port2_mux_sel, reg_res_latch_en: Diğer birimlere ait yetkilendirme sinyalleridir. Bu sinyaller “1” olduğunda birimler aktiftir; diğer durumda pasiftir.

Aşağıda Kontrol biriminin VHDL dilindeki “entity” tanımlaması görülmektedir.

```
entity controller is
port (  clk : in std_logic;
        CE : in std_logic;
        reset : in std_logic;
        opcode : in std_logic_vector (3 downto 0);
        ready : in std_logic;
        we, fetch : out std_logic;
        addr_latch_en : out std_logic;
        disp_latch_en : out std_logic;
        disp_out_en : out std_logic;
        d2_en : out std_logic;
        dr_en : out std_logic;
        instr_latch_en : out std_logic;
        immed_signext_en : out std_logic;
        ALU_op : out ALU_command;
        CC_latch_en : out std_logic;
        PC_latch_en : out std_logic;
        PC_out_en : out std_logic;
        ram_en : out std_logic;
        reg_port1_en : out std_logic;
        reg_port2_en : out std_logic;
        reg_port3_en : out std_logic;
        reg_port2_mux_sel : out std_logic;
        reg_res_latch_en : out std_logic);
end controller;
```

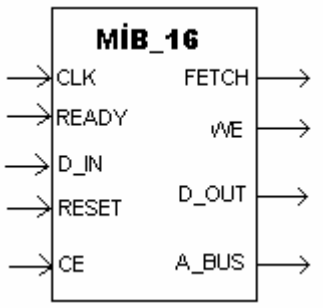
Kontrol birimine ait simülasyon sonuçları Şekil 3.22'de görülmektedir.



Şekil 3.22. Kontrol biriminin simülasyon sonuçları

3.14. Mikroişlemci (Mib_16) Biriminin Tasarımı

Şekil 3.23'de Mikroişlemciye ait giriş / çıkış uçları görülmektedir. Bu birim en üst birimdir. Diğer tüm birimleri kapsar ve bütün birimler arasındaki bağlantılar bu birim içerisinde gerçekleştirilir.



Şekil 3.23. Mikroişlemciye ait giriş / çıkış uçları

Bu birimin VHDL dilindeki “entity” tanımlaması ise aşağıdaki gibidir.

```
entity mib16 is
port (d_in : in    std_logic_vector (15 downto 0);
      d_out : out std_logic_vector (15 downto 0) bus;
      a_bus : out std_logic_vector (15 downto 0);
         CE : in std_logic;
         we : out std_logic;
      fetch : out std_logic;
      ready : in std_logic;
         clk : in std_logic;
       reset : in std_logic);
end mib16;
```

Bu birime ait simülasyon sonuçları bir sonraki bölümde verilmiştir.

BÖLÜM 4. 16-BİTLİK MİKROİŞLEMCİNİN GERÇEKLENMESİ VE DOĞRULANMASI

4.1 16-Bitlik Mikroişlemcinin Gerçeklenme Aşaması

Bölüm 3'de anlatıldığı üzere VHDL dilinde tanımlaması yapılan ve fonksiyonel olarak çalışması doğrulanan 16-bitlik mikroişlemcinin gerçeklenme aşaması, Xilinx 6.3i yazılımı yardımıyla gerçekleştirilmiştir. Tanımlanan ve simüle edilen tüm birimler sentezlenerek gerekli olan bağlantılar (Net-lists) oluşturulmuştur. Bu bağlantılar standart hücrelerden, Giriş / Çıkış ve RAM hücrelerinden meydana gelmiştir. Bağlantıların oluşturulmasıyla tasarım, gerçeklenme aşamasına hazır hale gelmiştir. Gerçekleme aşamasında aşağıdaki işlemler sırasıyla takip edilmektedir:

Gerçekleme aşaması (Implementation):

- Kullanıcı sınırlama dosyası (User Constraint File)
- Çevirme (Translate)
- Haritalama (Map)
- Yerleştirme ve Bağlama (Place and Route)
- Birim düzenleyici (Floorplanner)
- FPGA düzenleyici (FPGA Editor)
- Tahmini güç tüketimi (XPower)

Programlama dosyası oluşturma aşaması (Program File Genaration):

- BitGEN (Binary uzantılı dosya oluşturucu)
- IMPACT (Hedef elemana yükleme programı)

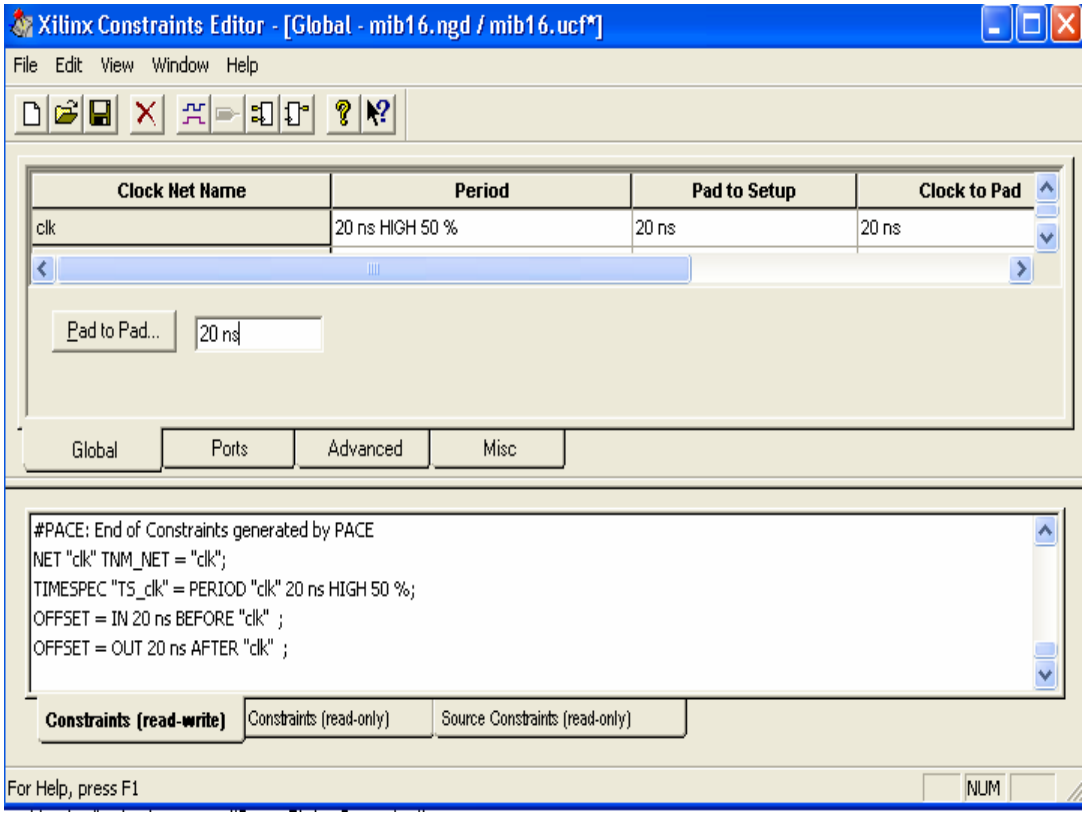
4.1.1. Kullanıcı sınırlama dosyası (User constraint file)

Tasarımın ilk aşamasında yapılması gereken kullanıcı sınırlama (UCF) işlemi ile, tasarlanan mantıksal blokların zamanlama, sentezleme ve yerleştirme ile ilgili sınırlamaları tanımlanabilir. Xilinx programında 3 farklı sınırlama türü mevcuttur;

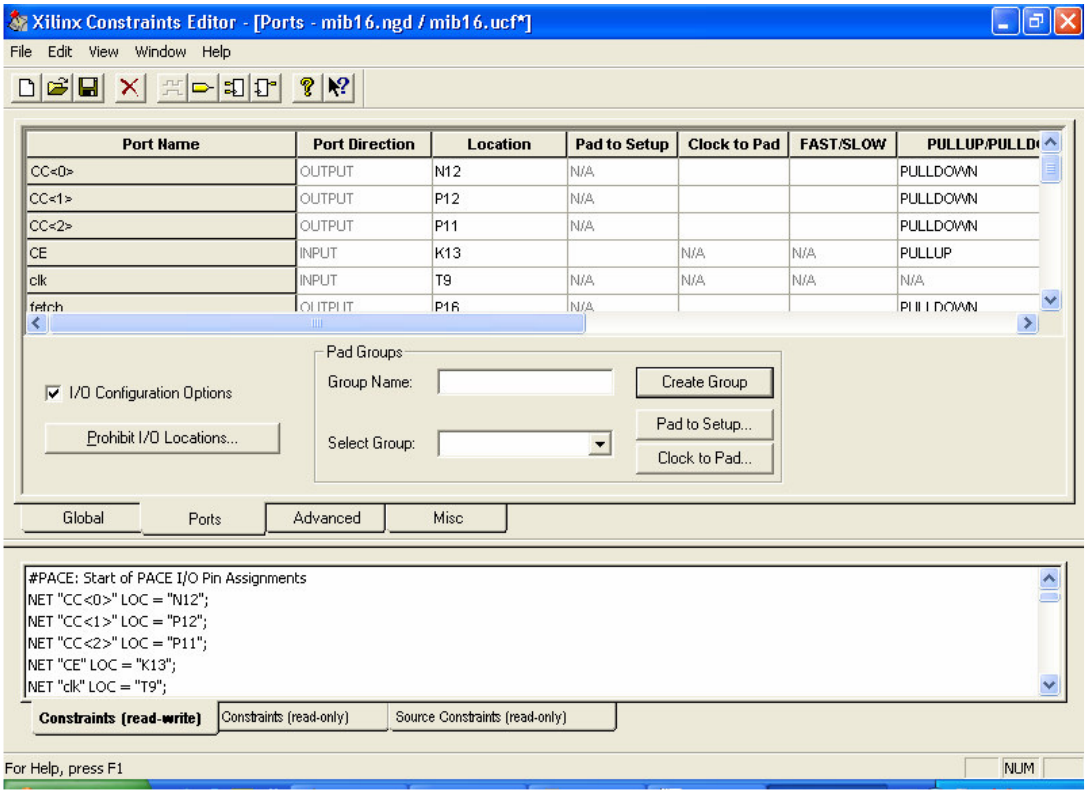
- Zamanlama sınırlamaları (Timing Constraint).
- Pin sınırlamaları (Package Pins Constraints).
- Alan sınırlamaları (Area Constraint).

4.1.1.1. Zamanlama sınırlamaları (Timing Constraint)

Zamanlama sınırlamaları ile herhangi bir bağlantıya, noktaya yada giriş / çıkış pinlerine zamanlama sınırlaması koyabiliriz. Tasarımın minimum saat sinyal değerini belirleyebiliriz. Şekil 4.1'de 16-bitlik mikroişlemcinin saat sinyali için zamanlama sınırlaması, Şekil 4.2'de ise Giriş / Çıkış pinleri için sınırlamaları görülmektedir.



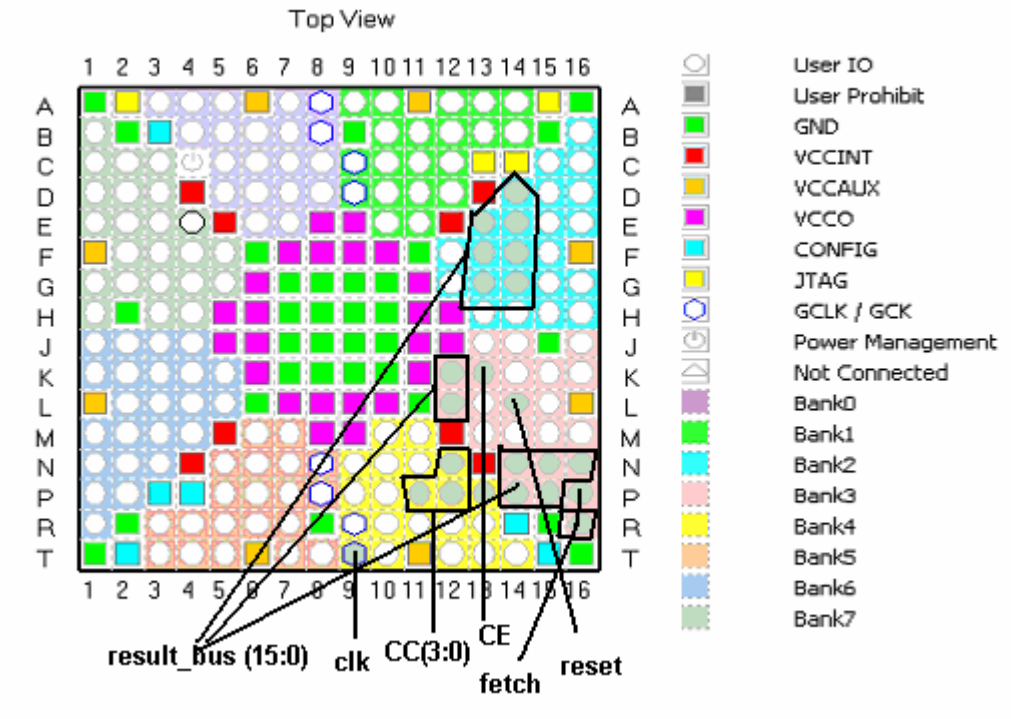
Şekil 4.1. 16-bitlik mikroişlemcinin saat sinyali sınırlaması.



Şekil 4.2. Giriş / Çıkış pinleri sınırlamaları

4.1.1.2. Pin sınırlamaları (Package Pins Constraints)

Pin sınırlamaları ile tasarımımızda bulunan mevcut pinlerin atamalarını yapabiliriz. Giriş /Çıkış pinlerinin yerlerini ve türlerini belirleyebilir, kullanılmasını istemediğimiz pinleri kullanıma kapatabiliriz. Tasarım hata kontrolü (Design Rule Check-DRC) ile atamalarda meydana gelebilecek muhtemel hataları görerek düzeltebiliriz. Şekil 4.3 16-bitlik mikroişlemcinin giriş / çıkış pinlerinin yerleşimi görülmektedir.



Şekil 4.3 16-bitlik mikroişlemcinin giriş / çıkış pinlerinin yerleşimi

4.1.1.3. Alan sınırlamaları (Area constraint)

Alan sınırlama işlemi ile, seçilen hedef elemanın kaynakları görsel olarak gözlemlenebilir, mantıksal bloklar için alan sınırlaması getirilebilir ve giriş / çıkış pinlerinin bağlantıları görülebilir.

4.1.2. Çevirme (Translate) işlemi

Gerçekleme işleminin ilk basamağıdır. "Translate" işleminde, sentezleme aşaması sonunda oluşturulan giriş bağlantıları (Input netlists) ve tasarım sınırlama bilgileri (User constraint file) birleştirilerek NGD uzantılı bir çıkış dosyası oluşturulur.

4.1.3. Haritalama (Mapping) işlemi

Çevirme işlemi gerçekleştirildikten sonra oluşturulan NGD uzantılı dosya haritalama işleminde kullanılır. Bu aşamada ilk olarak mantıksal tasarım hata kontrolü (Logical Design Rule Check-DRC) yapılır. Hata kontrolünden sonra ise tasarım, kullanılacak hedef eleman içindeki kaynaklar (Lojik hücreler, Giriş/ Çıkış hücreleri ve diğer elemanlar) uygun olarak haritalandırılır. Bu işlem sonunda NCD uzantılı çıkış dosyası oluşturulur.

4.1.4. Yerleştirme ve Bağlama (Place and Route) işlemi

Haritalama işleminden sonra gerçekleştirilen işlemdir. Oluşturulan tasarım hedef eleman içerisine yerleştirilir ve birimler arasındaki bağlantılar gerçekleştirilir. Yerleştirme ve bağlama işlemi iki farklı kritere göre yapılır.

- Maliyet: Bağlantı uzunlukları ve yönlendirme yapılacak mevcut kaynakların durumu göz önüne alınarak yapılan yerleştirme ve bağlama işlemi
- Zamanlama: Zamanlama sınırlamaları göz önüne alınarak yapılan yerleştirme ve bağlama işlemi.

Yerleştirme ve bağlama işlemi sonunda da NCD (Haritalama işlemi sonunda oluşturulan NCD dosyasının yeniden düzenlenmiş hali) uzantılı bir çıkış dosyası oluşturulur.

4.1.5. Birim düzenleyici (Floorplan) programı

Floorplan programı, tasarımda bulunan birimlerin veya mantıksal blokların, hedef eleman içerisindeki yerlerinin gözlenmesi, tespiti ve düzenlemesi için kullanılır. Floorplaner ile kullanılacak Giriş / Çıkış hücrelerinin, hafıza bloklarının veya standart hücrelerin hedef eleman içindeki yerleri değiştirilerek yer optimizasyonu sağlanır.

Floorplaner, tasarım işlemleri boyunca farklı aşamalarda kullanılabilir.

- Haritalama işlemi gerçekleştirilmeden önce.
- Yerleştirme ve bağlama işlemi gerçekleştirilmeden önce.
- Yerleştirme ve bağlama işlemi gerçekleştirildikten sonra.

Şekil 4.4'de 16-bitlik mikroişlemcinin yerleştirme ve bağlama işlemi gerçekleştirildikten sonraki Floorplan görüntüsü görülmektedir.

Şekil 4.4. 16-bitlik mikroişlemcinin yerleştirme ve bağlama işlemi gerçekleştirildikten sonraki Floorplan görüntüsü

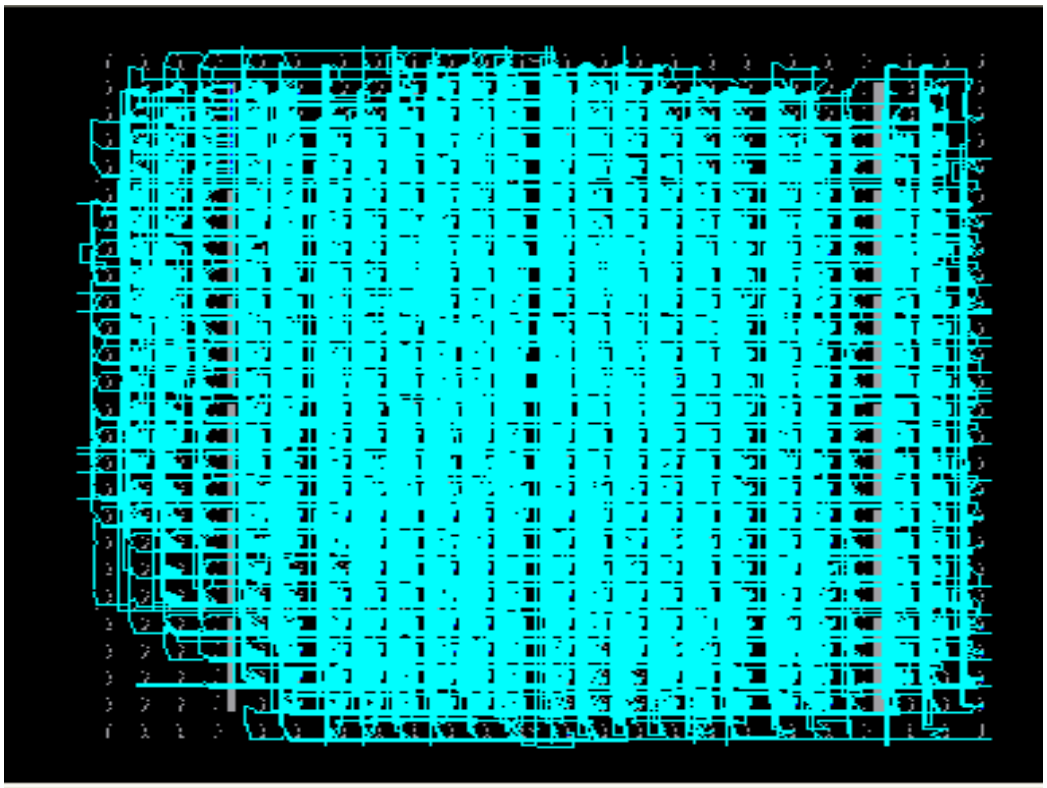
4.1.6. FPGA düzenleyici (FPGA Editor)

FPGA düzenleyici, hedef elemanın (FPGA-Field Programmable Gate Array) gözlenmesi ve konfigüre edilmesini sağlayan grafiksel bir uygulama programıdır.

FPGA düzenleyici ile aşağıdaki işlemler gerçekleştirilebilir:

- Otomatik yerleştirme ve bağlama işlemi gerçekleştirilmeden önce kritik elemanlar, yerleştirme ve bağlama işlemine tabi tutulabilir.
- Hedef eleman üzerindeki bağlantılar ve sinyallere prob eklenerek inceleme yapılabilir.

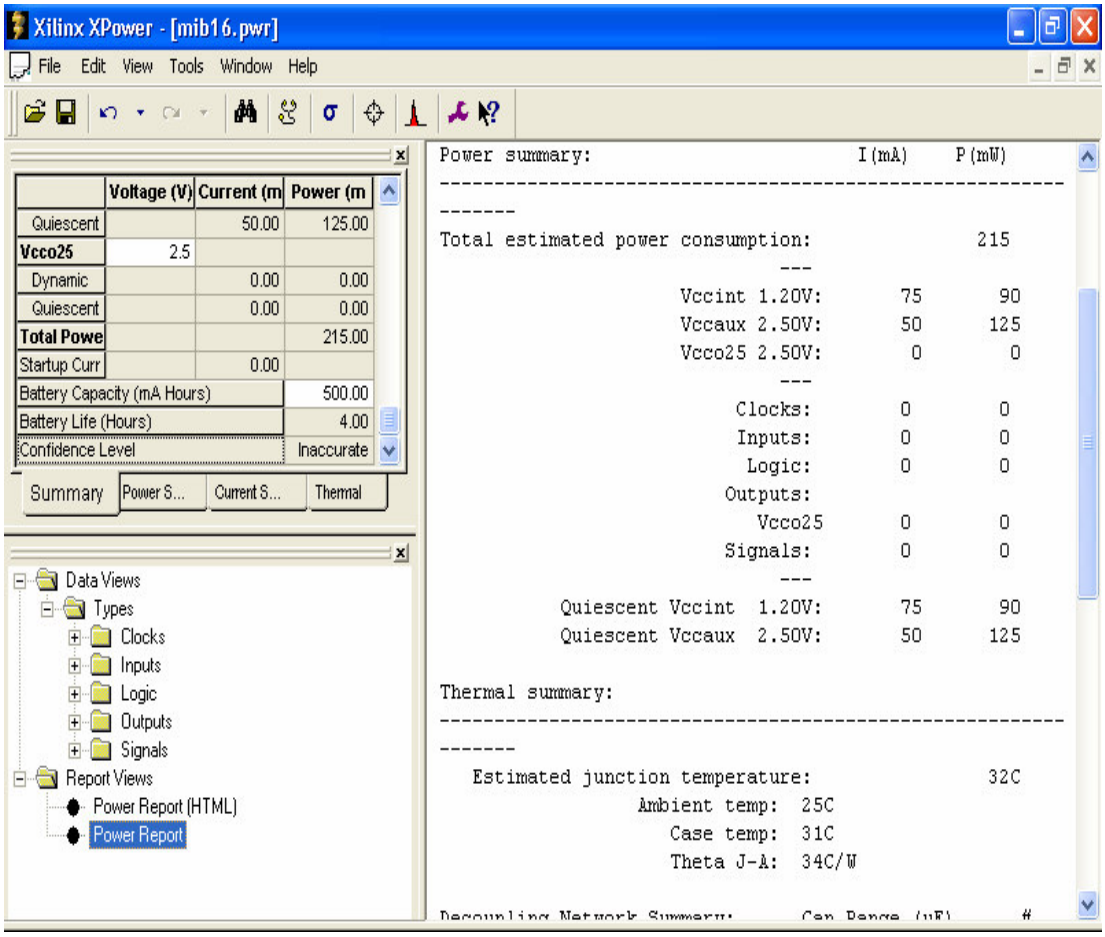
Şekil 4.5'de 16-bitlik mikroişlemci tasarımının yerleştirme ve bağlama işlemi sonrasında hedef eleman içerisindeki bağlantıları görülmektedir.



Şekil 4.5. 16-bitlik mikroişlemci tasarımının yerleştirme ve bağlama işlemi sonrasında hedef eleman içerisindeki bağlantıları

4.1.7. Güç tüketim tahmini (XPower)

XPower, kullanılacak hedef elemanın toplam güç tüketimi ve çalışma sıcaklıklarını hesaplayan programdır. Şekil 4.6'da 16-bitlik mikroişlemcinin toplam güç tüketimi, çalışma sıcaklığı ve 500mA / saatlik bir pil ile çalışma süresi görülmektedir. 16-bitlik mikroişlemcinin normal çalışma modunda toplam güç tüketimi 215 mW, tahmini bağlantı sıcaklığı 32 C ve 500 mA'lik bir pil ile çalışma süresi 4 saattir.

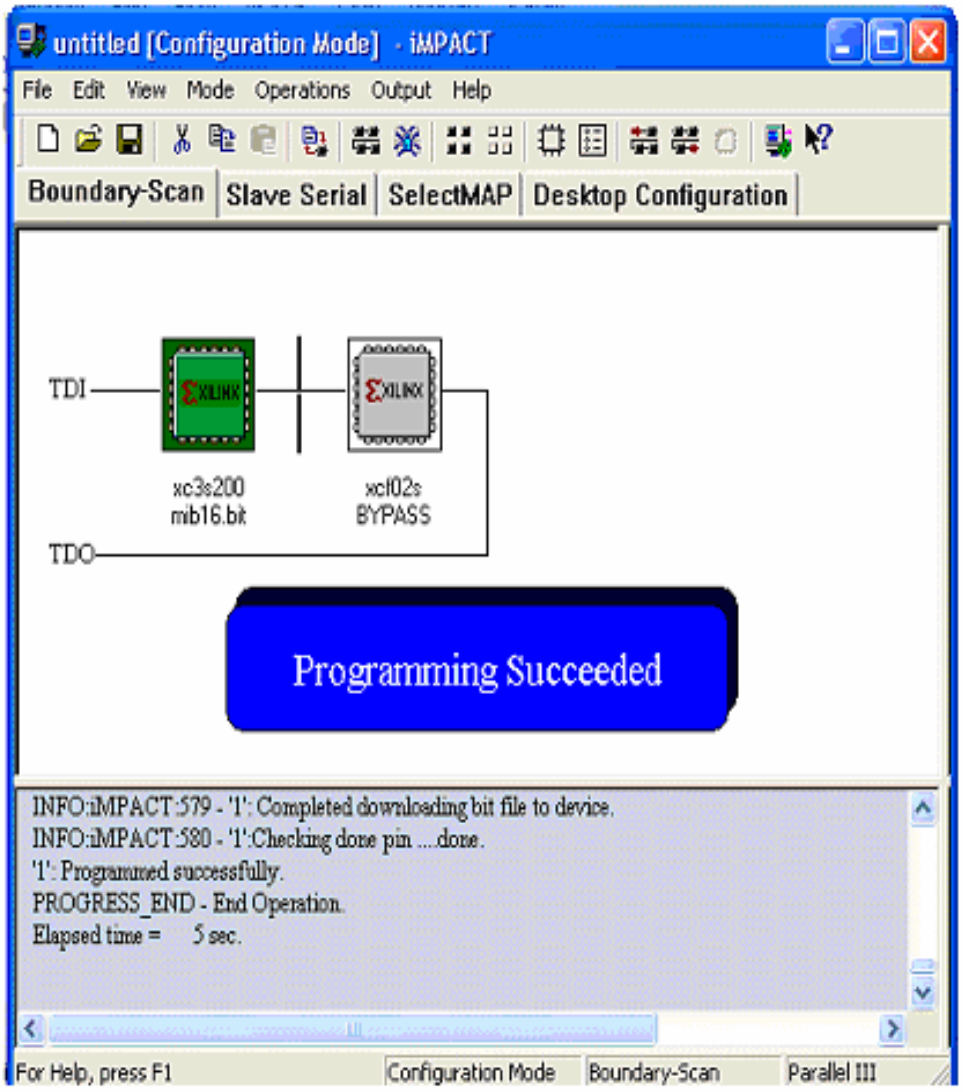


Şekil 4.6. 16-bitlik mikroişlemcinin tahmini güç tüketim raporu

4.1.8 Programlama dosyası oluşturma aşaması (Programing file generation)

16-bitlik mikroişlemci tasarımının, yerleştirme ve bağlama işlemine tabi tutulduktan sonra hedef elemana yüklenebilmesi için programlama dosyası oluşturulması (Programing file generation) gerekmektedir. Bu aşamada ilk olarak Xilinx FPGA elemanlarının konfigürasyon dosyası olan "BIT" uzantılı programlama dosyası

oluşturulur. CPLD'lerde aynı fonksiyonu "JEDEC" uzantılı dosya gerçekleştirmektedir. Program dosyası oluşturulduktan sonra Xilinx ISE yazılımında mevcut olan IMPACT uygulama programı ile tasarım, hedef elemana yüklenir. Yükleme işlemini gerçekleştirmek için Xilinx firması tarafından üretilen Spartan-3 XC3S200-FT256 deneme kartı kullanılmıştır. (FPGA programlanabilir elemanları ve tasarımda kullanılan Spartan-3 XC3S200-FT256 hedef elemanı hakkında detaylı bilgi Ek-B' de, proto board ile ilgili bilgi ise Ek-C' de verilmiştir.) Bu kartın içersindeki en büyük kapasiteli tek port blok RAM, 10-bitlik adres yoluna sahip olduğu için yükleme yapılmadan önce 16-bitlik adres yolu 10-bit adres yoluna dönüştürülmüştür. Şekil 4.7'de 16-bitlik mikroişlemcinin IMPACT programı vasıtasıyla hedef elemana yüklenmesi görülmektedir.



Şekil 4.7. 16-bitlik mikroişlemcinin hedef elemana yüklenmesi

4.2. 16-Bitlik Mikroişlemcinin Doğrulanması

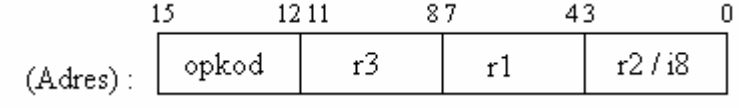
16-bitlik mikroişlemcinin gerçeklenme aşamasından sonra doğrulanma işlemi için ModelSim 5.8v'i simülasyon programı kullanılmıştır.

16-bitlik mikroişlemcinin yürütebildiği komutlar ve her bir komutun formatı Tablo 4.1 ve Tablo 4.2'de verilmiştir.

Tablo 4.1. Aritmetik ve lojik komutlar

Komut	**Adı**	**Fonksiyonu**	**Opkod**
Add	Toplama	r3 ← r1+r2	0000
Sub	Çıkarma	r3 ← r1-r2	0001
Mul	Çarpma	r3 ← r1*r2	0010
Div	Bölme	r3 ← r1/r2	0011
Addq	İvedi Toplama	r3 ← r1+i8	0100
Subq	İvedi Çıkarma	r3 ← r1-i8	0101
Mulq	İvedi Çarpma	r3 ← r1*i8	0110
Divq	İvedi Bölme	r3 ← r1/i8	0111
Land	Lojik AND	r3 ← r1&r2	1000
Lor	Lojik OR	r3 ← r1!r2	1001
Lxor	Lojik XOR	r3 ← r1+r2	1010
Lmask	Lojik Maskeleme	r3 ← r1&-r2	1011

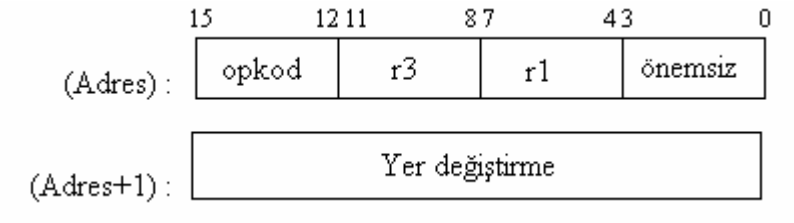
Komut seti 4-bitlik dört bölüme ayrılmıştır ilk alan opkodu göstermektedir. R3 sonucun saklanacağı saklayıcının adresini, r1 ve r2 kaynak saklayıcı adresini göstermektedir. İ8 ise 2'ye tümleme işlemiyle 4-bitlik sayıdan 16-bitlik sayı üretmede kullanılmaktadır.



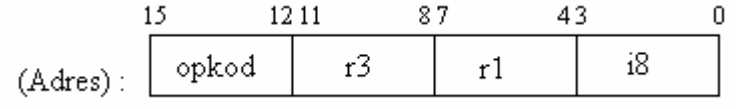
Tablo 4.2. Saklama ve yükleme komutları

Komut	**Adı**	**Fonksiyonu**	**Opkod**
Ld	Yükleme	r3 ← M[r1+disp16]	1100
St	Saklama	M[r1+disp16] ← r3	1101
Ldq	İvedi Yükleme	r3 ← M[r1+i8]	1110
Stq	İvedi Saklama	M[r1+i8] ←r3	1111

Hafızadan yükleme ve hafızaya saklama komutları, yer değiştirme adresinin uzun veya kısa oluşuna göre iki formata sahiptir. Uzun yer değiştirmeler için format:

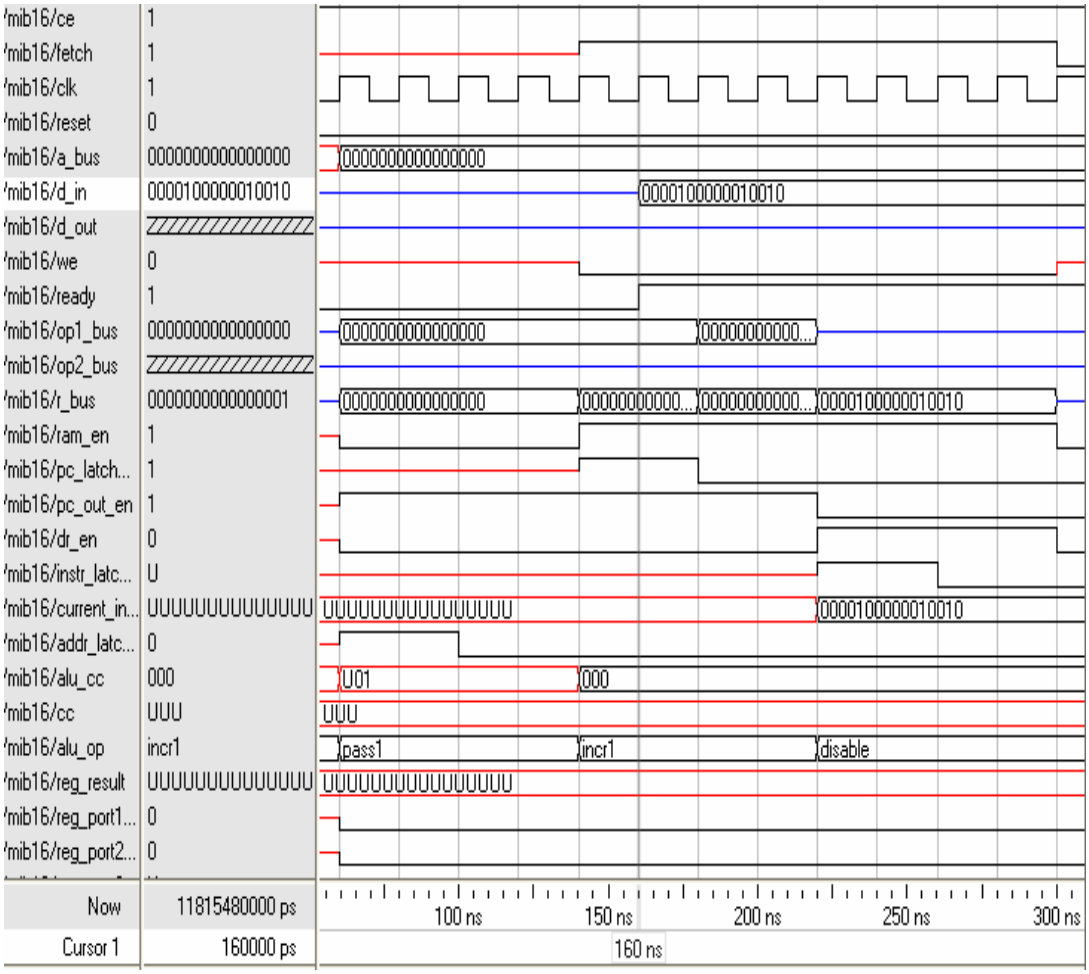


Kısa yer değiştirmeler için format ise şu şekildedir:



4.2.1. Okuma işleminin doğrulanması

Şekil 4.8'de Okuma işlemine ait sinyal değişimleri görülmektedir. "0000000000000000" adresindeki "0000100000010010" verisi okunarak "d_in" sinyaline alınmıştır.

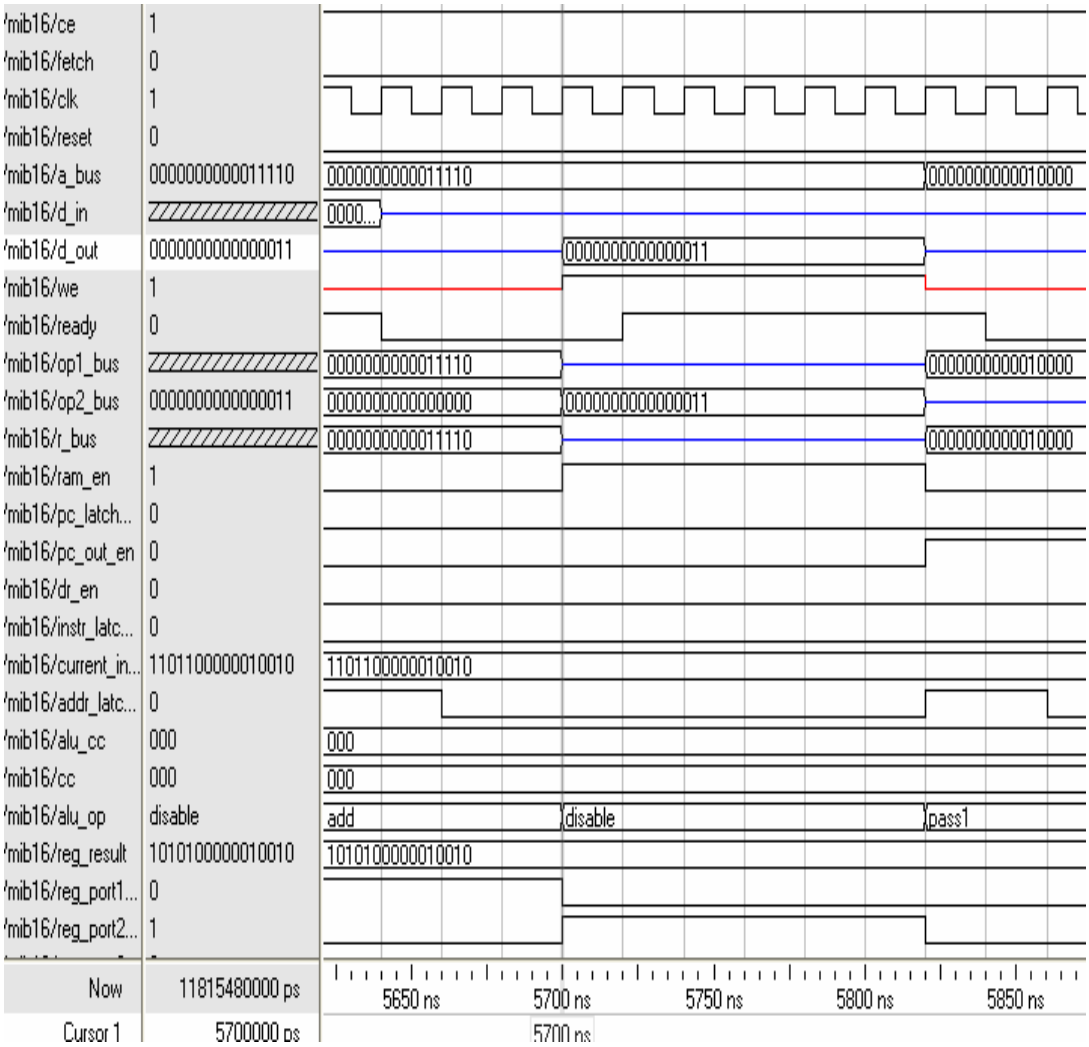


Şekil 4.8. Okuma işlemine ait sinyal değişimleri

Okuma işleminde öncelikle hafızadan okunacak verinin adresi Program Sayıcıdan alınır ve adres yoluna (a_bus) verilir. Hafıza birimine ait yetkilendirme sinyali (ram_en) aktif hale getirilir. Okuma işlemi gerçekleştirileceğinden "we" sinyali "0" konumuna alınır ve okuma işlemi saat sinyali "1" olduğunda başlar. Hafızadan okunan veri, mikroişlemcinin veri giriş yolu olan "d_in" e alınır. Okunan veri komut ise "fetch" sinyali "1" konumunda; eğer veri ise "0" konumundadır. Okuma işlemi tamamlandığında "ready" sinyali "1" olur.

4.2.2. Yazma işleminin doğrulanması

Şekil 4.9'da Yazma işlemine ait sinyal değişimleri görülmektedir. "d_out" sinyalindeki "0000000000000011" verisi hafızanın "0000000000011110" adresine yazılmıştır.

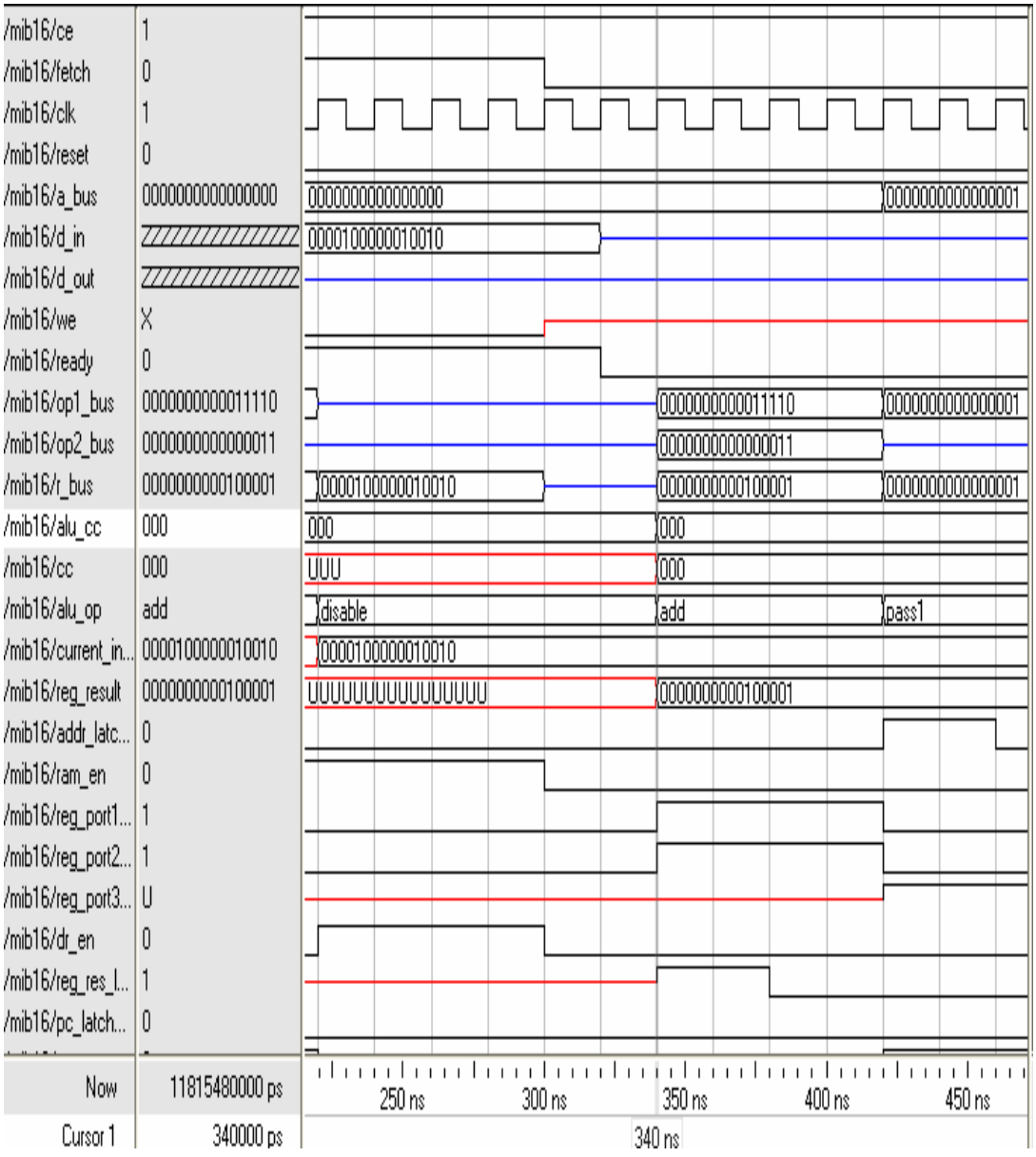


Şekil 4.9. Yazma işlemine ait sinyal değişimleri görülmektedir.

Hafızaya yazılacak verinin adresi a_bus'ta bulunmaktadır. Yazma işleminin yapılabilmesi için hafıza birimine ait yetkilendirme sinyali (ram_en) aktif hale getirilir. Yazma işlemi gerçekleştirileceğinden "we" sinyali "1" konumuna alınır ve yazma işlemi başlar. Mikroişlemcinin veri çıkış yolu olan d_out sinyalindeki veri, a_bus'ta bulunan adrese saat sinyali "1" olduğunda yazılır. Yazma işlemi tamamlandığında "ready" sinyali "1" olur.

4.2.3. Toplama komutunun doğrulanması

Şekil 4.10'da Toplama komutuna ait sinyal değişimleri görülmektedir.



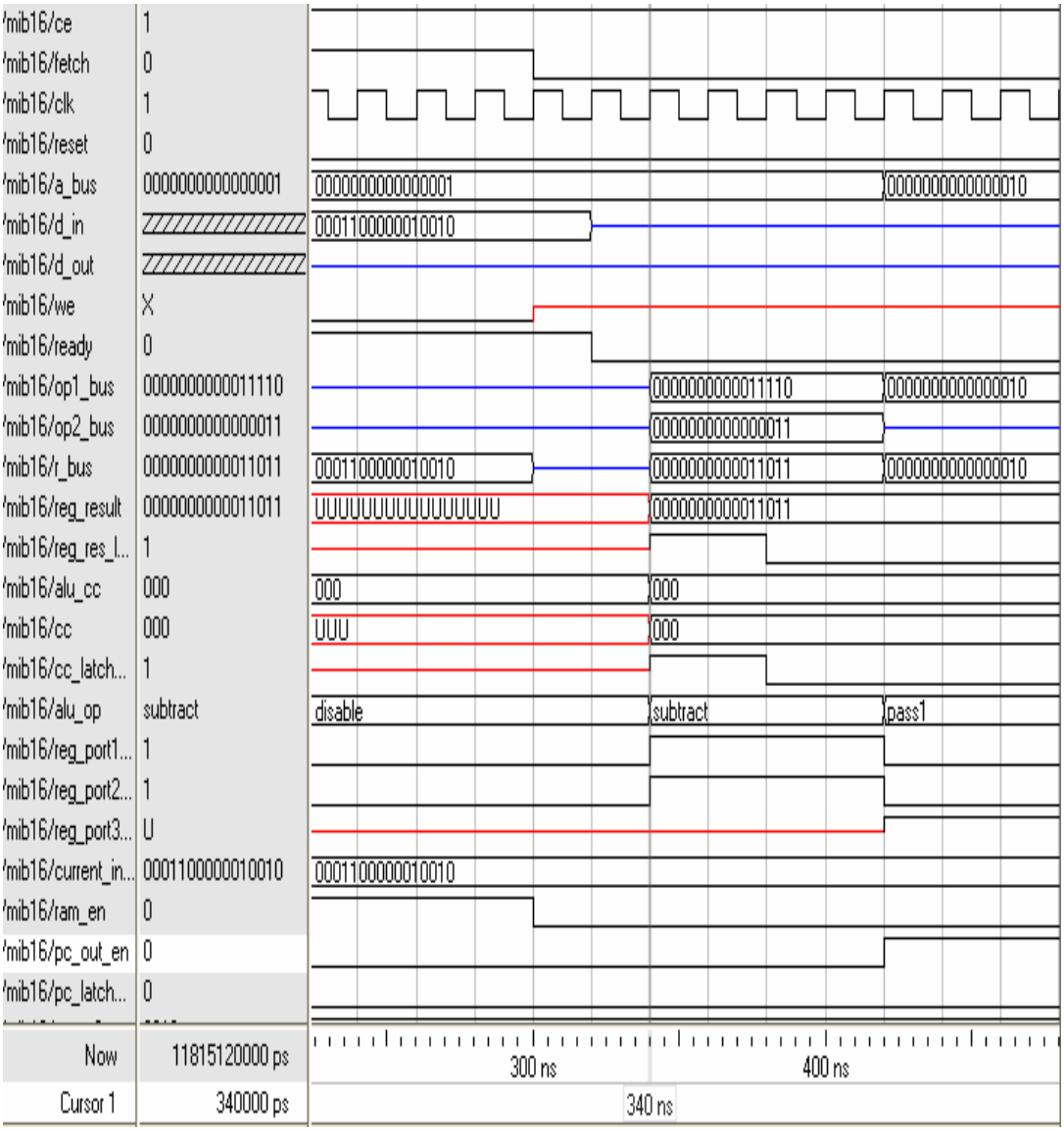
Şekil 4.10. Toplama komutuna ait sinyal değişimleri

Toplama işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve komutta belirtilen kaydedicilerdeki verilerden ilki op1_bus'a, ikincisi ise op2_bus'a alınır ve toplama işlemine tabi tutulur. Sonuç, result_bus' a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC)

içeriği belirlenir. Şekil 4.10'da "0001" adresli kaydedicideki "0000000000011110" verisi "0010" adresli kaydedicideki "0000000000000011" verisi ile toplanarak "0000000000100001" sonucu elde edilmiştir. Sonuç komutta belirtilen "1000" adresli kaydediciye yazılmıştır. İşlemi onluk tabanda ifade edecek olursak 1. kaydedicideki "30" sayısı ile 2. kaydedicideki "3" sayısı ile toplanmış ve "33" sayısı elde edilmiştir.

4.2.4. Çıkarma komutunun doğrulanması

Şekil 4.11'de Çıkarma komutuna ait sinyal değişimleri görülmektedir.

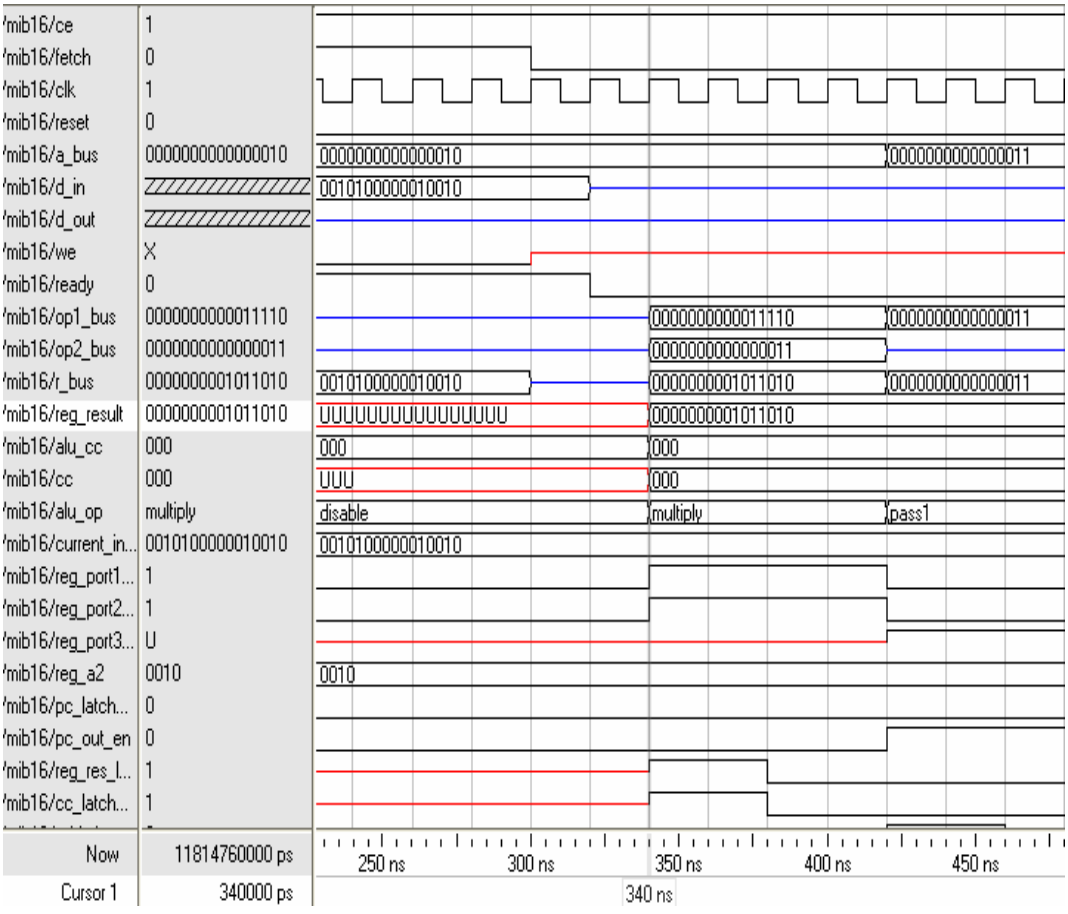


Şekil 4.11. Çıkarma komutuna ait sinyal değişimleri

Çıkarma işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve komutta belirtilen kaydedicilerdeki verilerden ilki op1_bul'a, ikincisi ise op2_bul'a alınır ve çıkarma işlemine tabi tutulur. Sonuç, result_bus'a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC) içeriği belirlenir. Şekil 4.11'de "0001" adresli kaydedicideki "0000000000011110" verisinden "0010" adresli kaydedicideki "0000000000000011" verisi çıkartılarak "0000000000011101" sonucu elde edilmiştir. Sonuç komutta belirtilen "1000" adresli kaydediciye yazılmıştır. İşlemi onluk tabanda ifade edecek olursak 1. kaydedicideki "30" sayısından 2. kaydedicideki "3" sayısı çıkartılmış ve "27" sayısı elde edilmiştir.

4.2.5. Çarpma komutunun doğrulanması

Şekil 4.12'de Çarpma komutuna ait sinyal değişimleri görülmektedir.

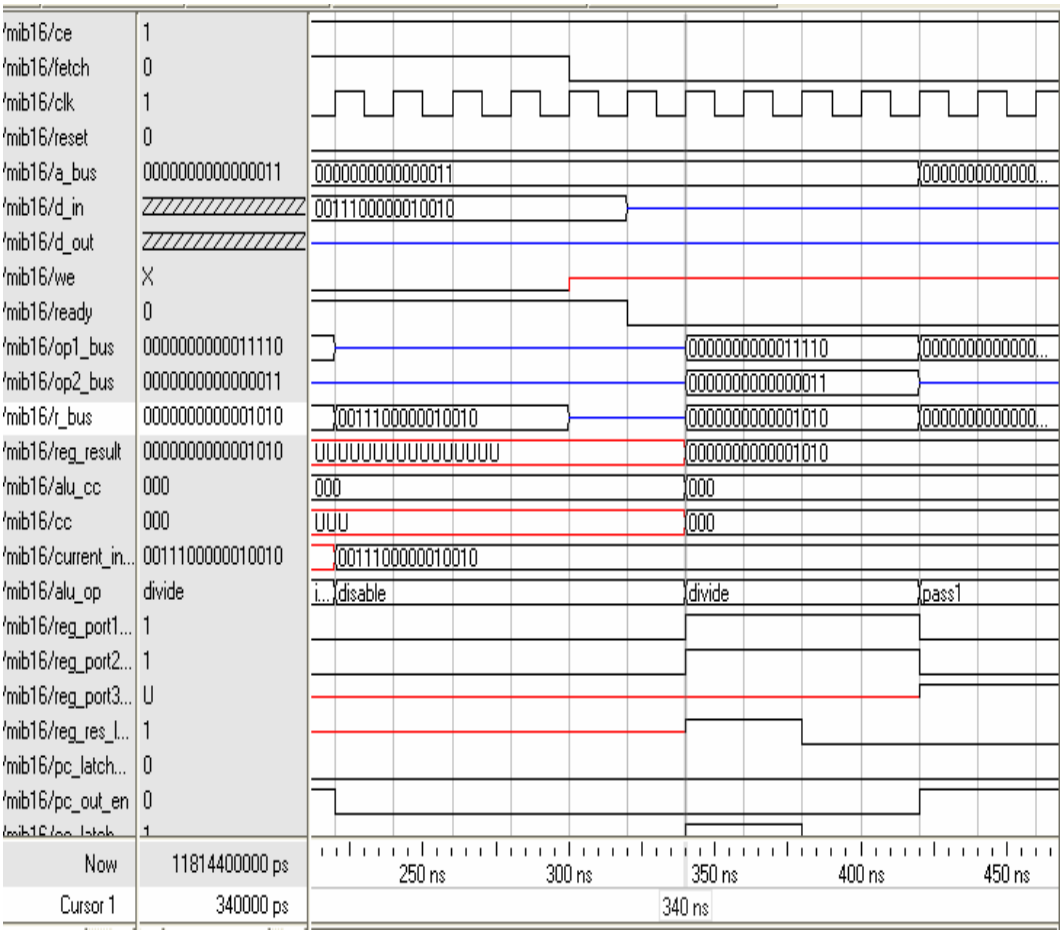


Şekil 4.12. Çarpma komutuna ait sinyal değişimleri

Çarpma işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve komutta belirtilen kaydedicilerdeki verilerden ilki op1_bus'a, ikincisi ise op2_bus'a alınır ve çarpma işlemine tabi tutulur. Sonuç, result_bus'a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC) içeriği belirlenir. Şekil 4.12'de "0001" adresli kaydedicideki "0000000000011110" verisi ile "0010" adresli kaydedicideki "0000000000000011" verisi çarpılarak "0000000001011010" sonucu elde edilmiştir. Sonuç komutta belirtilen "1000" adresli kaydediciye yazılmıştır. İşlemi onluk tabanda ifade edecek olursak 1. kaydedicideki "30" sayısı ile 2. kaydedicideki "3" sayısı çarpılmış ve "90" sayısı elde edilmiştir.

4.2.6. Bölme komutunun doğrulanması

Şekil 4.13'de Bölme komutuna ait sinyal değişimleri görülmektedir.

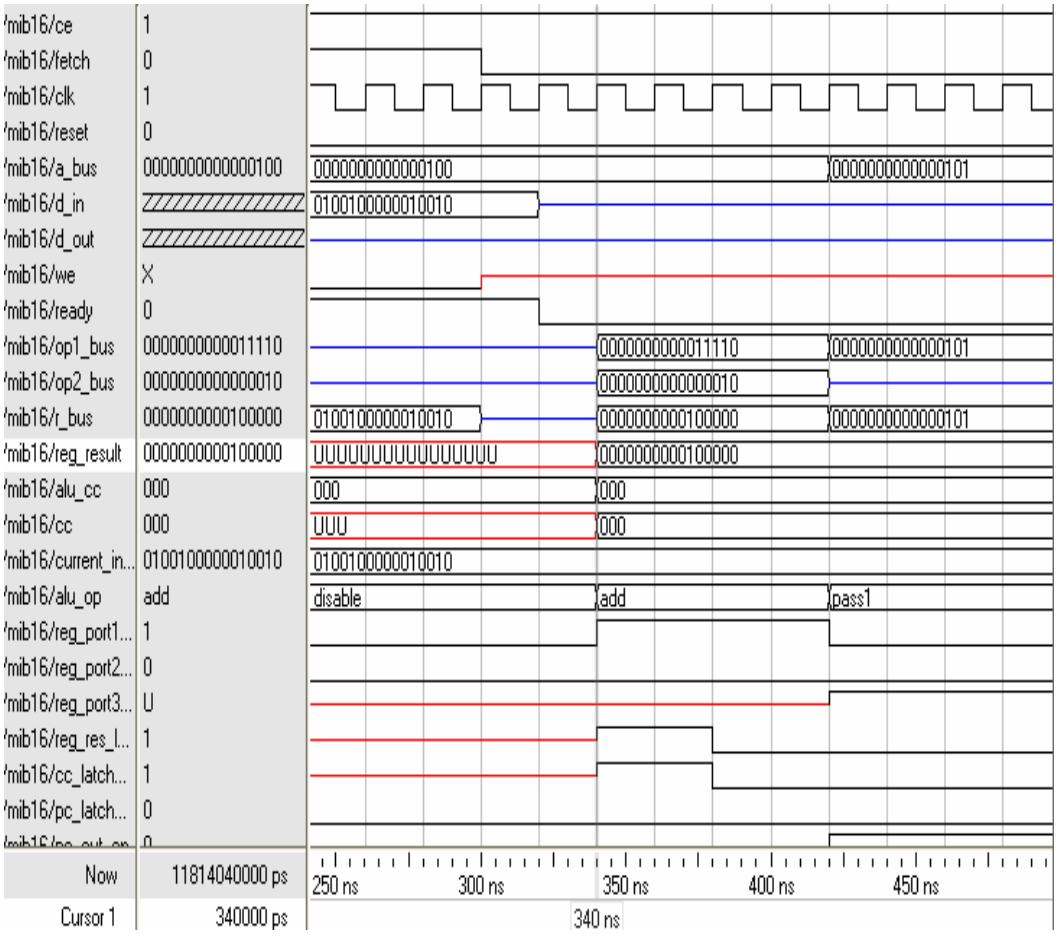


Şekil 4.13. Bölme komutuna ait sinyal değişimleri

Bölme işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve komutta belirtilen kaydedicilerdeki verilerden ilki op1_bus'a, ikincisi ise op2_bus'a alınır ve bölme işlemine tabi tutulur. Sonuç, result_bus'a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC) içeriği belirlenir. Şekil 4.13'de "0001" adresli kaydedicideki "0000000000011110" verisi "0010" adresli kaydedicideki "0000000000000011" verisine bölünerek "0000000000001010" sonucu elde edilmiştir. Sonuç komutta belirtilen "1000" adresli kaydediciye yazılmıştır. İşlemi onluk tabanda ifade edecek olursak 1. kaydedicideki "30" sayısı 2. kaydedicideki "3" sayısına bölünerek "10" sayısı elde edilmiştir.

4.2.7. İvedi Toplama komutunun doğrulanması

Şekil 4.14'de ivedi toplama komutuna ait sinyal değişimleri görülmektedir.

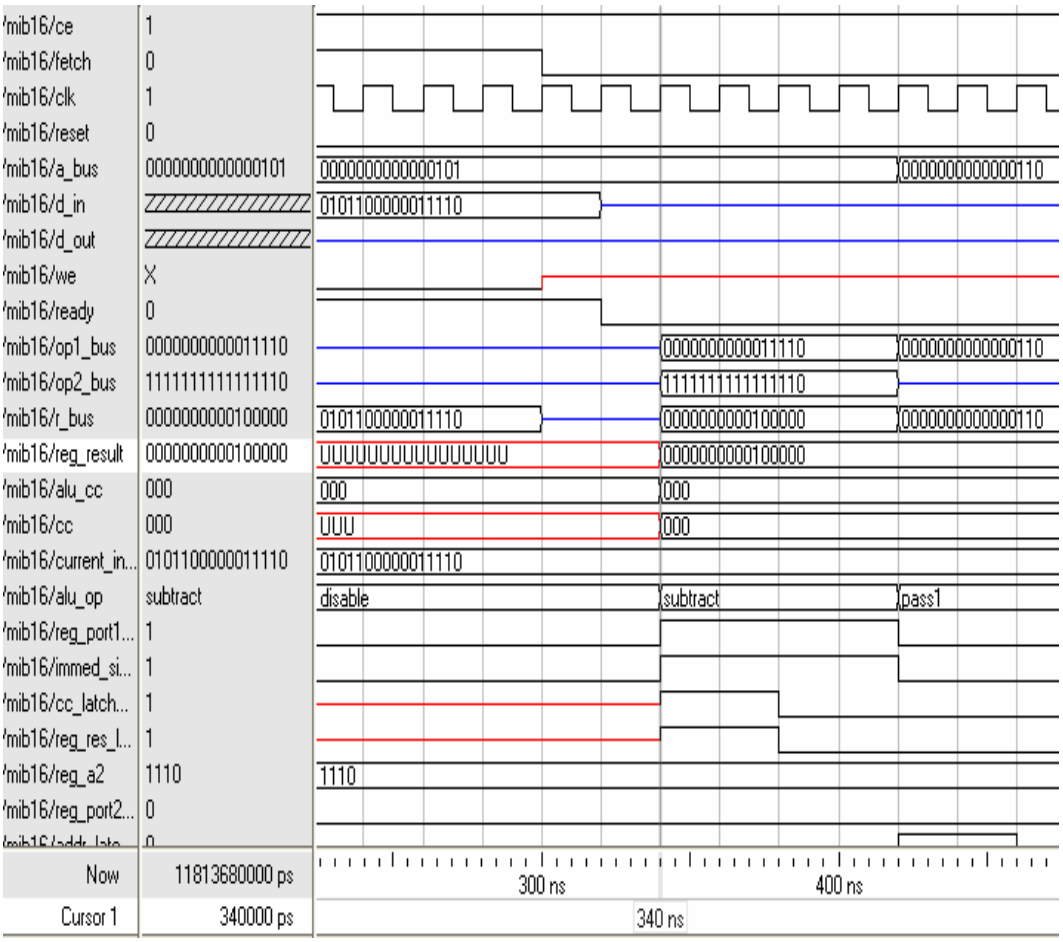


Şekil 4.14. İvedi Toplama komutuna ait sinyal değişimleri

İvedi toplama işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve işleme tabi tutulacak verilerden ilki, komutta adresi belirtilen kaydediciden elde edilir ve op1_bus'a alınır. İkinci veri ise 16-bitlik komutun son dört bitindeki 4-bitlik verinin 16-bitlik veriye genişletilmesiyle elde edilir ve op2_bus'a alınarak toplama işlemine tabi tutulur. Sonuç, result_bus'a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC) içeriği belirlenir. Şekil 4.14'de "0001" adresli kaydedicideki "0000000000011110" verisi, "0010" 4-bitlik verisinin 2'ye tümleme işlemiyle 16-bitlik veriye genişletilmiş şekli olan "0000000000000010" verisi ile toplanarak "0000000000100000" sonucu elde edilmiştir. Sonuç, komutta belirtilen "1000" adresli kaydediciye yazılmıştır. İşlemi onluk tabanda ifade edecek olursak 1. kaydedicideki "30" sayısı, "2" sayısı ile toplanarak "32" sayısı elde edilmiştir. Komuta ivedi toplama denmesinin nedeni, ikinci verinin kaydediciden alınması yerine 4-bitlik veriden 16-bitlik veri elde edilerek işlem yapılmasıdır.

4.2.8. İvedi Çıkarma komutunun doğrulanması

Şekil 4.15'de İvedi Çıkarma komutuna ait sinyal değişimleri görülmektedir.



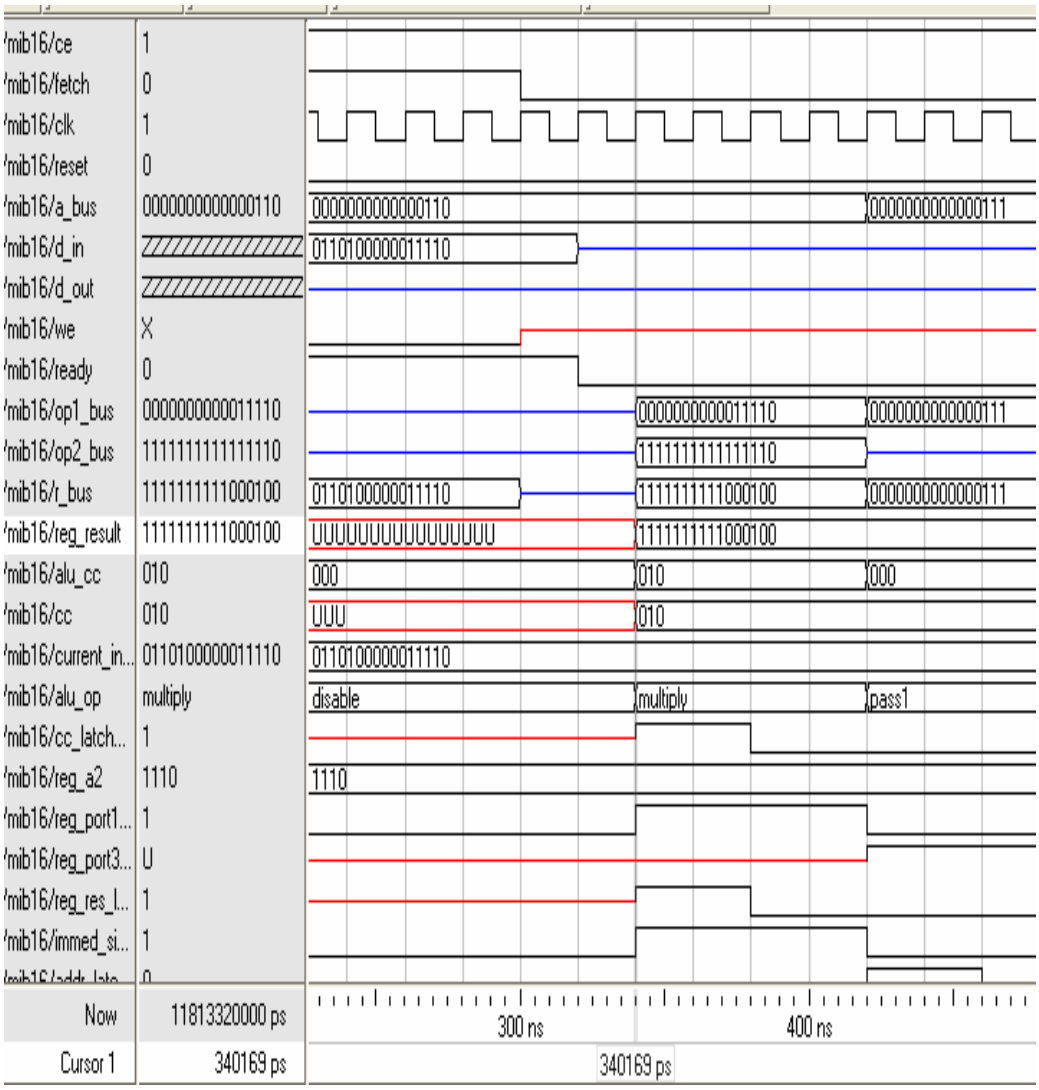
Şekil 4.15. İvedi Çıkarma komutuna ait sinyal değişimleri

İvedi çıkarma işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve işleme tabi tutulacak verilerden ilki, komutta adresi belirtilen kaydediciden elde edilir ve op1_bus'a alınır. İkinci veri ise 16-bitlik opkodun son dört bitindeki 4-bitlik verinin 16-bitlik veriye genişletilmesiyle elde edilir ve op2_bus'a alınarak çıkarma işlemine tabi tutulur. Sonuç, result_bus'a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC) içeriği belirlenir. Şekil 4.15'de "0001" adresli kaydedicideki "0000000000011110" verisinden, "1110" 4-bitlik verisinin 2'ye tümleme işlemiyle 16-bitlik veriye genişletilmiş şekli olan "1111111111111110" verisi çıkartılarak "0000000000100000" sonucu elde edilmiştir. Sonuç, komutta belirtilen "1000"

adresli kaydediciye yazılmıştır. İşlemi onluk tabanda ifade edecek olursak 1. kaydedicideki “30” sayısından, “-2” sayısı çıkartılarak “32” sayısı elde edilmiştir. Komuta ivedi çıkarma denmesinin nedeni, ikinci verinin kaydediciden alınması yerine 4-bitlik veriden 16-bitlik veri elde edilerek işlem yapılmasıdır.

4.2.9. İvedi Çarpma komutunun doğrulanması

Şekil 4.16’da İvedi Çarpma komutuna ait sinyal değişimleri görülmektedir.

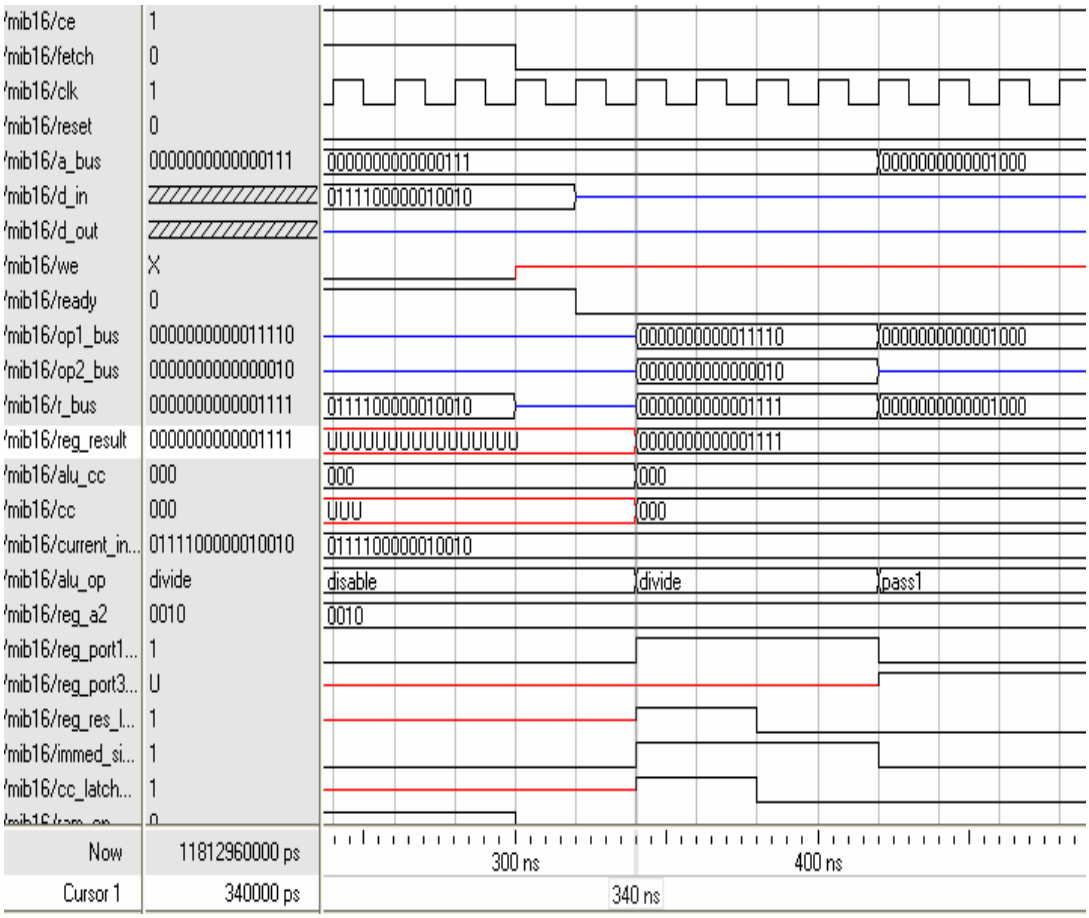


Şekil 4.16. İvedi Çarpma komutuna ait sinyal değişimleri

İvedi çarpma işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve işleme tabi tutulacak verilerden ilki, komutta adresi belirtilen kaydediciden elde edilir ve op1_bus'a alınır. İkinci veri ise 16-bitlik opkodun son dört bitindeki 4-bitlik verinin 16-bitlik veriye genişletilmesiyle elde edilir ve op2_bus'a alınarak çıkarma işlemine tabi tutulur. Sonuç, result_bus'a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC) içeriği belirlenir. Şekil 4.16'de "0001" adresli kaydedicideki "0000000000011110" verisi, "1110" 4-bitlik verisinin 2'ye tümleme işlemiyle 16-bitlik veriye genişletilmiş şekli olan "1111111111111110" verisi ile çarpılarak "1111111111000100" sonucu elde edilmiştir. Sonuç, komutta belirtilen "1000" adresli kaydediciye yazılmıştır. İşlemi onluk tabanda ifade edecek olursak 1. kaydedicideki "30" sayısı, "-2" sayısı ile çarpılarak "-60" sayısı elde edilmiştir. Sonuç negatif olduğu için bayrakları gösteren "CC" sinyalindeki negatif biti "1" olmuştur. Komuta ivedi çarpma denmesinin nedeni, ikinci verinin kaydediciden alınması yerine 4-bitlik veriden 16-bitlik veri elde edilerek işlem yapılmasıdır.

4.2.10. İvedi Bölme komutunun doğrulanması

Şekil 4.17'de İvedi Bölme komutuna ait sinyal değişimleri görülmektedir.



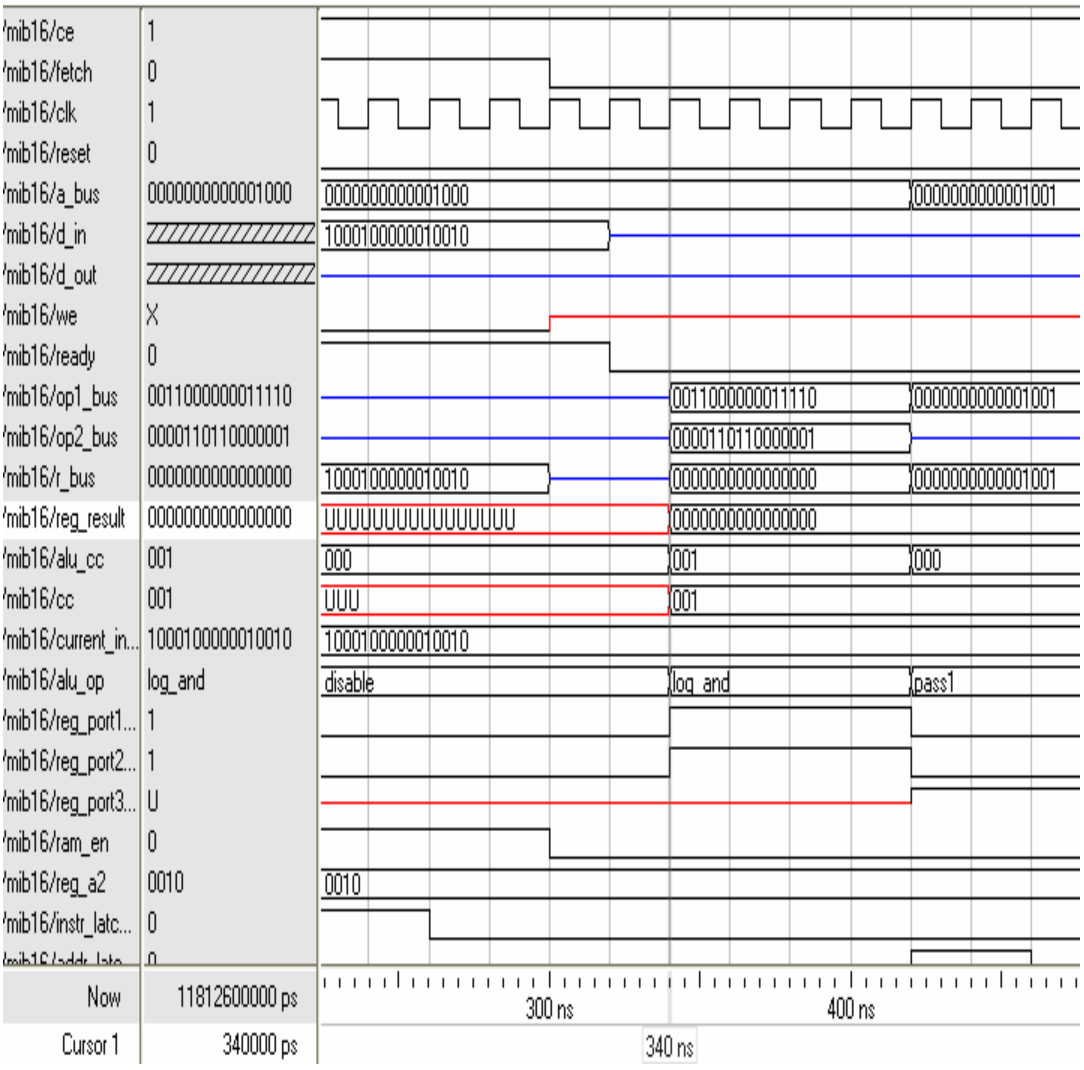
Şekil 4.17. İvedi Bölme komutuna ait sinyal değişimleri

İvedi bölme işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve işleme tabi tutulacak verilerden ilki, komutta adresi belirtilen kaydediciden elde edilir ve op1_bus'a alınır. İkinci veri ise 16-bitlik opkodun son dört bitindeki 4-bitlik verinin 16-bitlik veriye genişletilmesiyle elde edilir ve op2_bus'a alınarak toplama işlemine tabi tutulur. Sonuç, result_bus'a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC) içeriği belirlenir. Şekil 4.17'de "0001" adresli kaydedicideki "0000000000011110" verisi, "0010" 4-bitlik verisinin 2'ye tümleme işlemiyle 16-bitlik veriye genişletilmiş şekli olan "0000000000000010" verisine bölünerek "0000000000001111" sonucu elde edilmiştir. Sonuç, komutta belirtilen "1000" adresli kaydediciye yazılmıştır.

İşlemi onluk tabanda ifade edecek olursak 1. kaydedicideki “30” sayısı, “2” sayısına bölünerek “15” sayısı elde edilmiştir. Komuta ivedi bölme denmesinin nedeni, ikinci verinin kaydediciden alınması yerine 4-bitlik veriden 16-bitlik veri elde edilerek işlem yapılmasıdır.

4.2.11. Lojik AND komutunun doğrulanması

Şekil 4.18’de Lojik AND komutuna ait sinyal değişimleri görülmektedir.



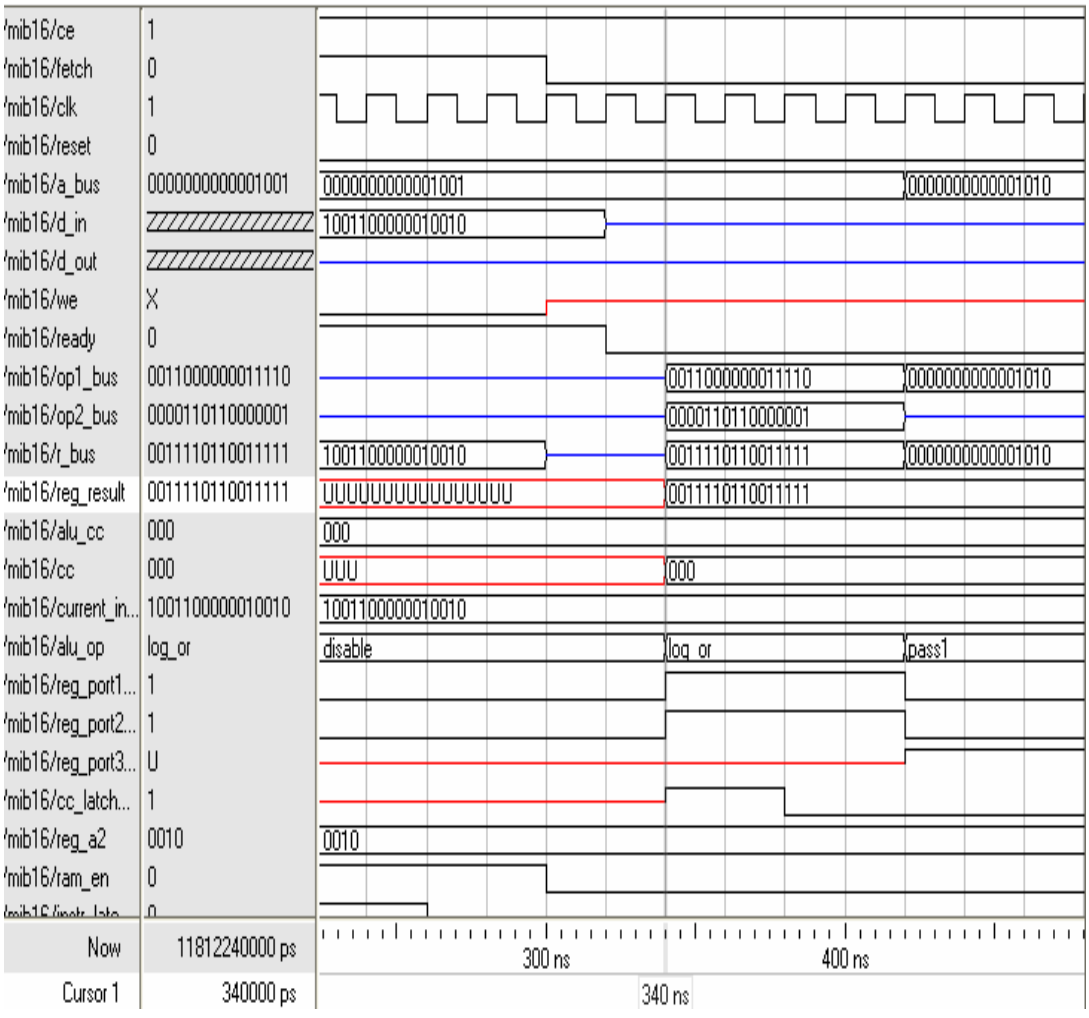
Şekil 4.18. Lojik AND komutuna ait sinyal değişimleri

Lojik AND işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve komutta belirtilen kaydedicilerdeki verilerden ilki op1_bus’a, ikincisi ise

op2_bus'a alınır ve Lojik AND işlemine tabi tutulur. Sonuç, result_bus'a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC) içeriği belirlenir. Şekil 4.18'de "0001" adresli kaydedicideki "0000000000011110" verisi ile "0010" adresli kaydedicideki "0000110110000001" verisi Lojik AND işlemine tabi tutularak "0000000000000000" sonucu elde edilmiştir. Sonuç komutta belirtilen "1000" adresli kaydediciye yazılmıştır. Sonuç sıfır çıktığından bayrakları gösteren "CC" sinyalindeki sıfır biti "1" olmuştur.

4.2.12. Lojik OR komutunun doğrulanması

Şekil 4.19'da Lojik OR komutuna ait sinyal değişimleri görülmektedir.

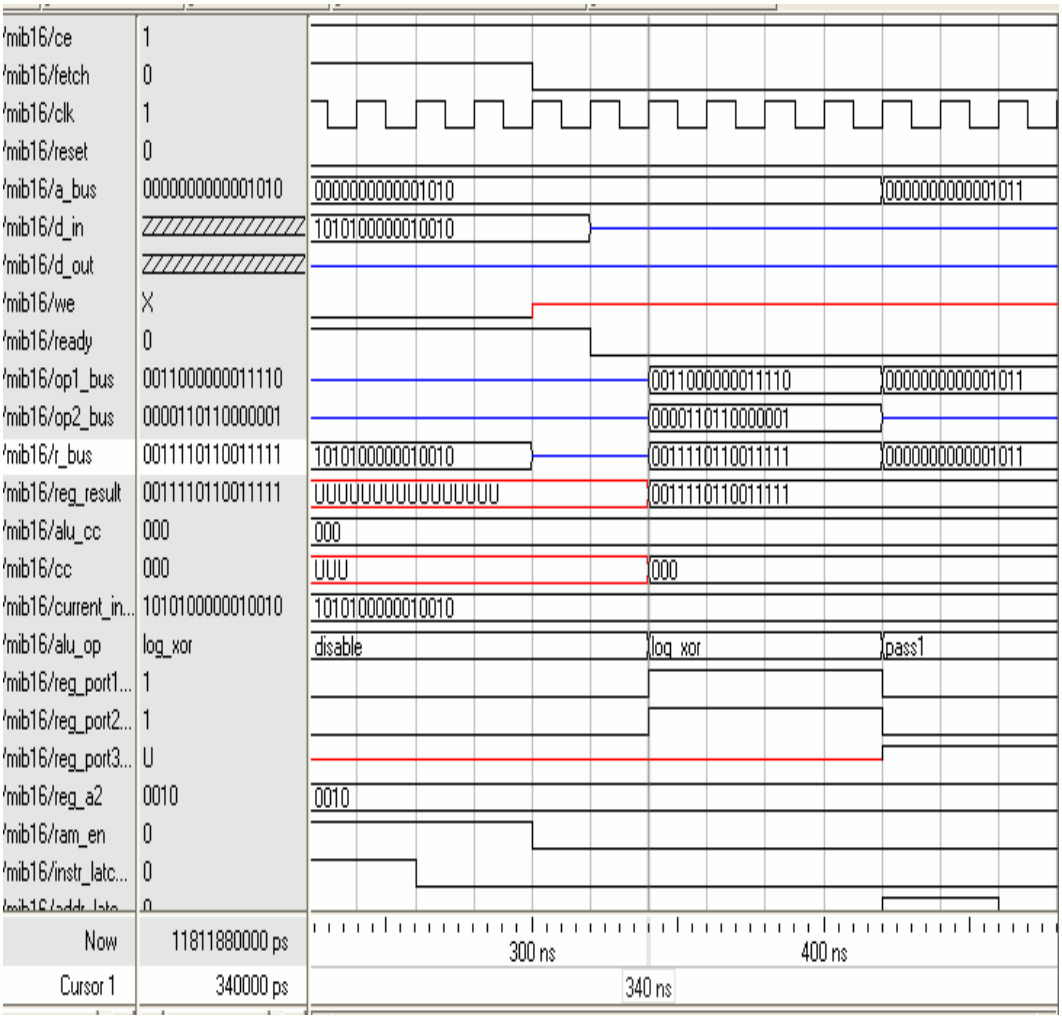


Şekil 4.19. Lojik OR komutuna ait sinyal değişimleri

Lojik OR işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve komutta belirtilen kaydedicilerdeki verilerden ilki op1_bus'a, ikincisi ise op2_bus'a alınır ve Lojik OR işlemine tabi tutulur. Sonuç, result_bus'a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC) içeriği belirlenir. Şekil 4.19'da "0001" adresli kaydedicideki "0000000000011110" verisi ile "0010" adresli kaydedicideki "0000110110000001" verisi Lojik OR işlemine tabi tutularak "0011110110011111" sonucu elde edilmiştir. Sonuç komutta belirtilen "1000" adresli kaydediciye yazılmıştır.

4.2.13. Lojik XOR komutunun doğrulanması

Şekil 4.20'de Lojik XOR komutuna ait sinyal değişimleri görülmektedir.

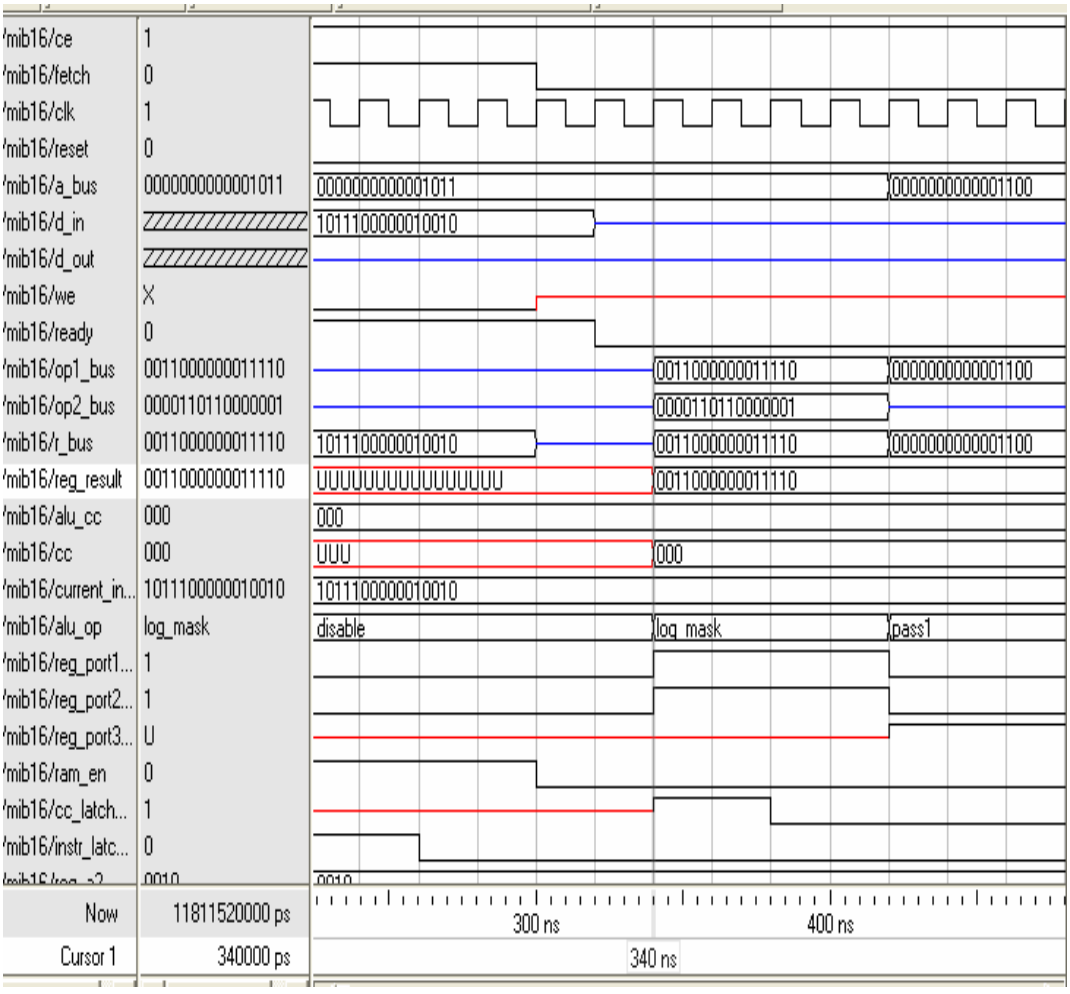


Şekil 4.20. Lojik XOR komutuna ait sinyal değişimleri

Lojik XOR işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve komutta belirtilen kaydedicilerdeki verilerden ilki op1_bus'a, ikincisi ise op2_bus'a alınır ve Lojik XOR işlemine tabi tutulur. Sonuç, result_bul'a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC) içeriği belirlenir. Şekil 4.20'de "0001" adresli kaydedicideki "0000000000011110" verisi ile "0010" adresli kaydedicideki "0000110110000001" verisi Lojik XOR işlemine tabi tutularak "0011110110011111" sonucu elde edilmiştir. Sonuç komutta belirtilen "1000" adresli kaydediciye yazılmıştır.

4.2.14. Lojik Maskeleme komutunun doğrulanması

Şekil 4.21'de Lojik maskeleme komutuna ait sinyal değişimleri görülmektedir.

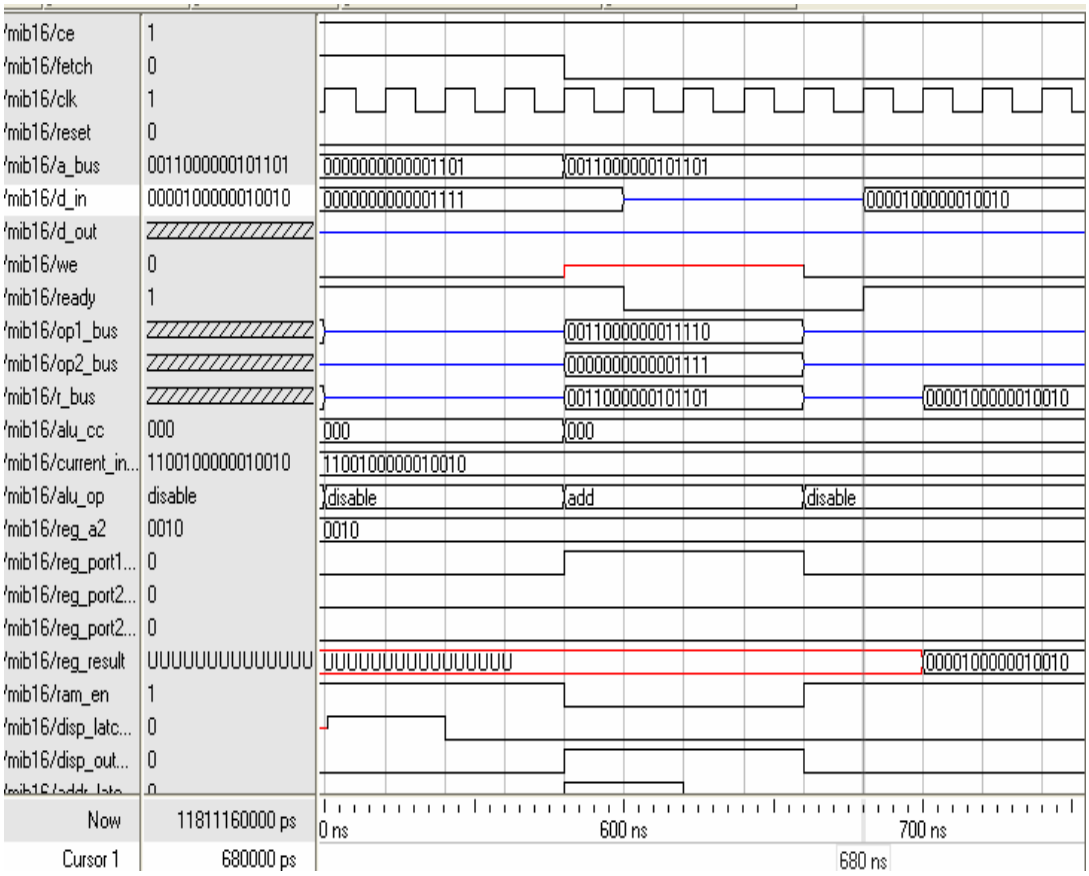


Şekil 4.21. Lojik maskeleme komutuna ait sinyal değişimleri

Lojik maskeleme işlemi, ilgili komutun hafızadan okunmasıyla başlar. Okunan komut çözülür ve komutta belirtilen kaydedicilerdeki verilerden ilki op1_bus'a, ikincisi ise op2_bus'a alınır ve Lojik maskeleme işlemine tabi tutulur. Lojik maskeleme işlemi, ikinci verinin tersi alınarak birinci veri ile lojik AND işlemine tabi tutulmasıdır. Sonuç, result_bus'a alınır ve yine komutta belirtilen kaydediciye yazılır. Sonuca bakılarak durum saklayıcının (CC) içeriği belirlenir. Şekil 4.21'de "0001" adresli kaydedicideki "0000000000011110" verisi ile "0010" adresli kaydedicideki "0000110110000001" verisi Lojik maskeleme işlemine tabi tutularak "0011000000011110" sonucu elde edilmiştir. Sonuç komutta belirtilen "1000" adresli kaydediciye yazılmıştır.

4.2.15. Yükleme komutunun doğrulanması

Şekil 4.22'de Yükleme komutuna ait sinyal değişimleri görülmektedir.

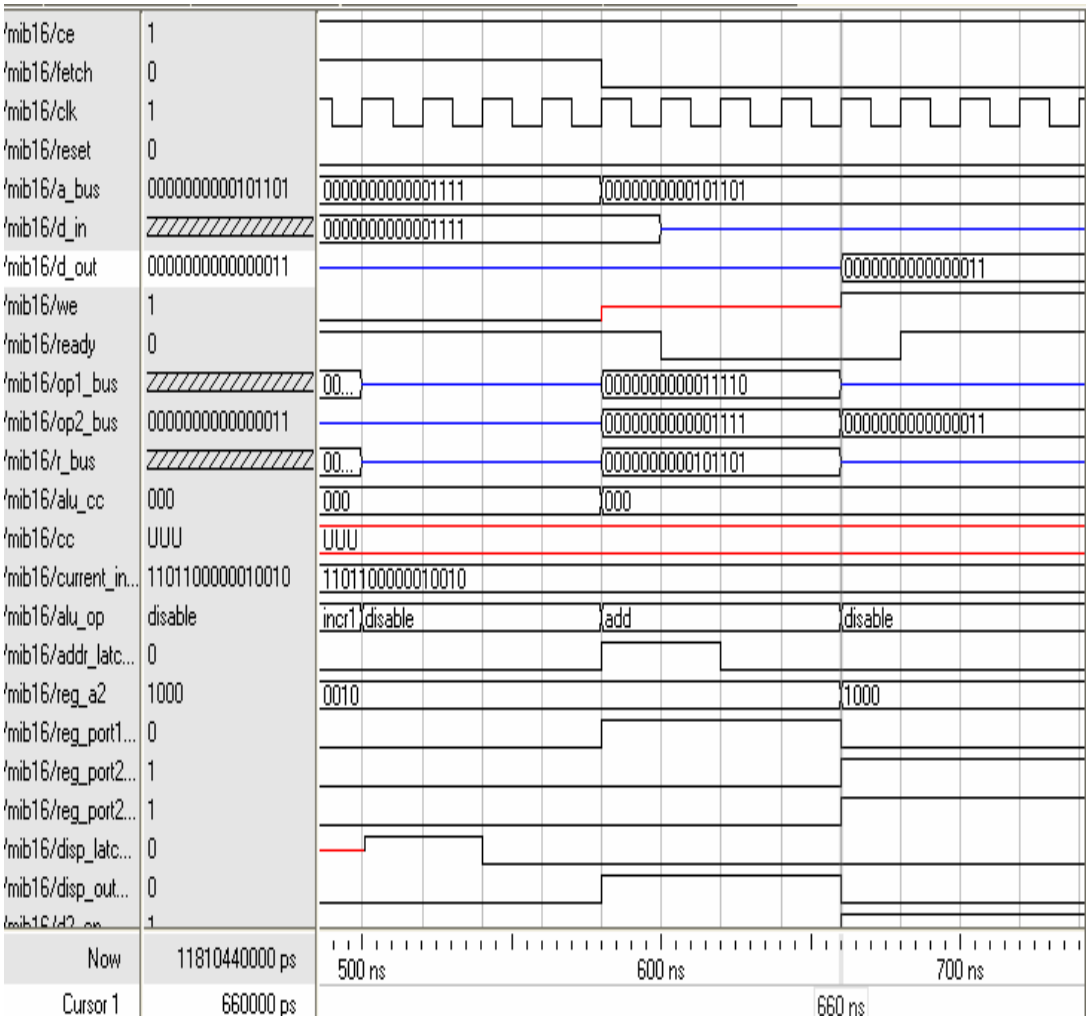


Şekil 4.22. Yükleme komutuna ait sinyal değişimleri

Yükleme işlemi, ilgili komutun hafızadan okunmasıyla başlar. Adres yolunda (a_bus) bulunan adres değeri bir artırılarak hafızadaki bir sonraki veri op2_bus' a alınır. Bu veri, hafızadan yükleme yapılacak verinin adresini bulmada indis adres görevini üstlenir. op1_bus'taki adres ile op2_bus'ta bulunan adres değeri toplanarak hafızadan yükleme yapılacak verinin adresi elde edilir. Hafızadan okuma yapılacağı için "we" sinyali "0" olur ve belirtilen adresteki veri, mikroişlemcinin giriş veri yolu olan "d_in" sinyaline alınır. Şekil 4.22'de hafızada "0011000000101101" adresinde bulunan "0000100000010010" verisi, "d_in" girişine yüklenmiştir.

4.2.16. Saklama komutunun doğrulanması

Şekil 4.23'de Saklama komutuna ait sinyal değişimleri görülmektedir.

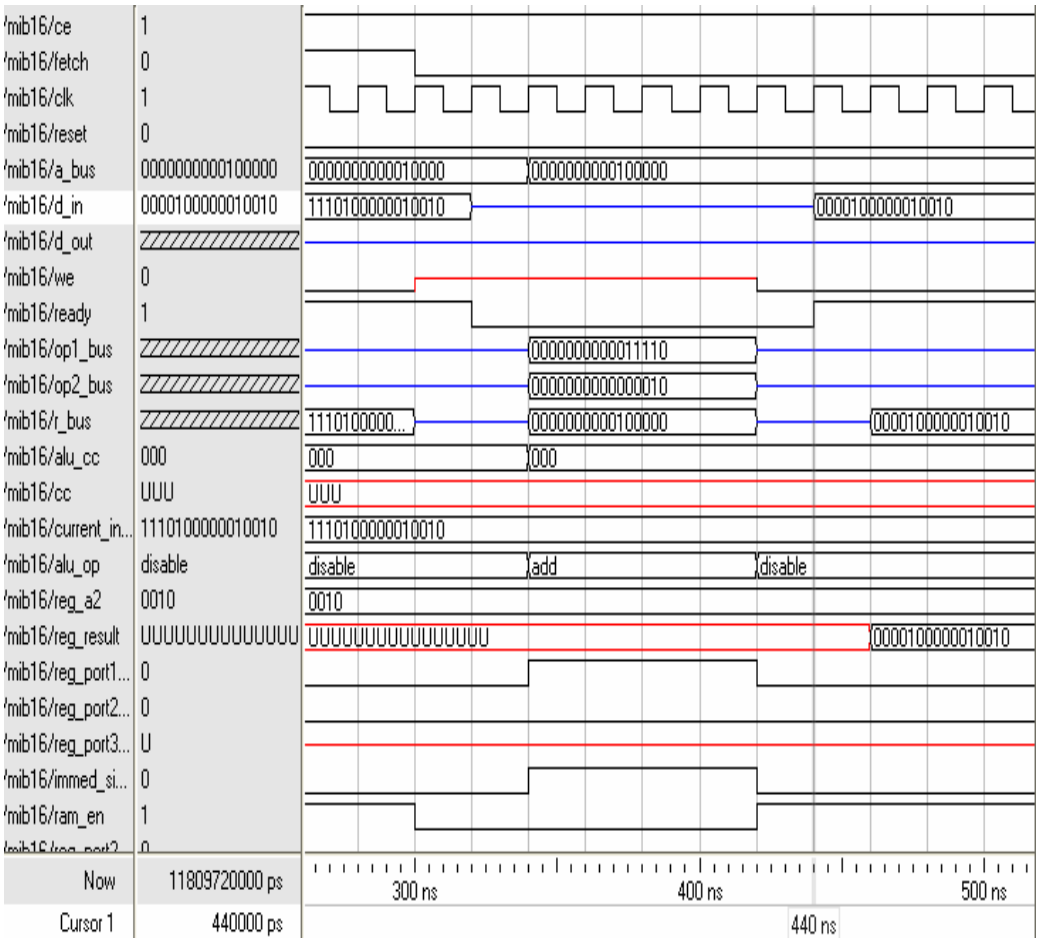


Şekil 4.23. Saklama komutuna ait sinyal değişimleri

Saklama işlemi, ilgili komutun hafızadan okunmasıyla başlar. Adres yolunda (a_bus) bulunan adres değeri bir artırılarak hafızadaki bir sonraki veri op2_bus' a alınır. Bu veri, hafızaya saklanacak verinin adresini bulmada indis adres görevini üstlenir. op1_bus'taki adres ile op2_bus'ta bulunan adres değeri toplanarak hafızaya saklanacak verinin adresi elde edilir. Hafızaya yazma işlemi yapılacağı için "we" sinyali "1" olur. "1000" adresli kaydedicideki veri alınarak mikroişlemcinin çıkış veri yolu olan "d_out" sinyaline alınır ve belirtilen adrese yazılır. Şekil 4.23'de "1000" adresli kaydedicide bulunan "0000000000000011" verisi, hafızanın "0000000000101101" adresine saklanmıştır.

4.2.17. İvedi Yükleme komutunun doğrulanması

Şekil 4.24'de İvedi Yükleme komutuna ait sinyal değişimleri görülmektedir.

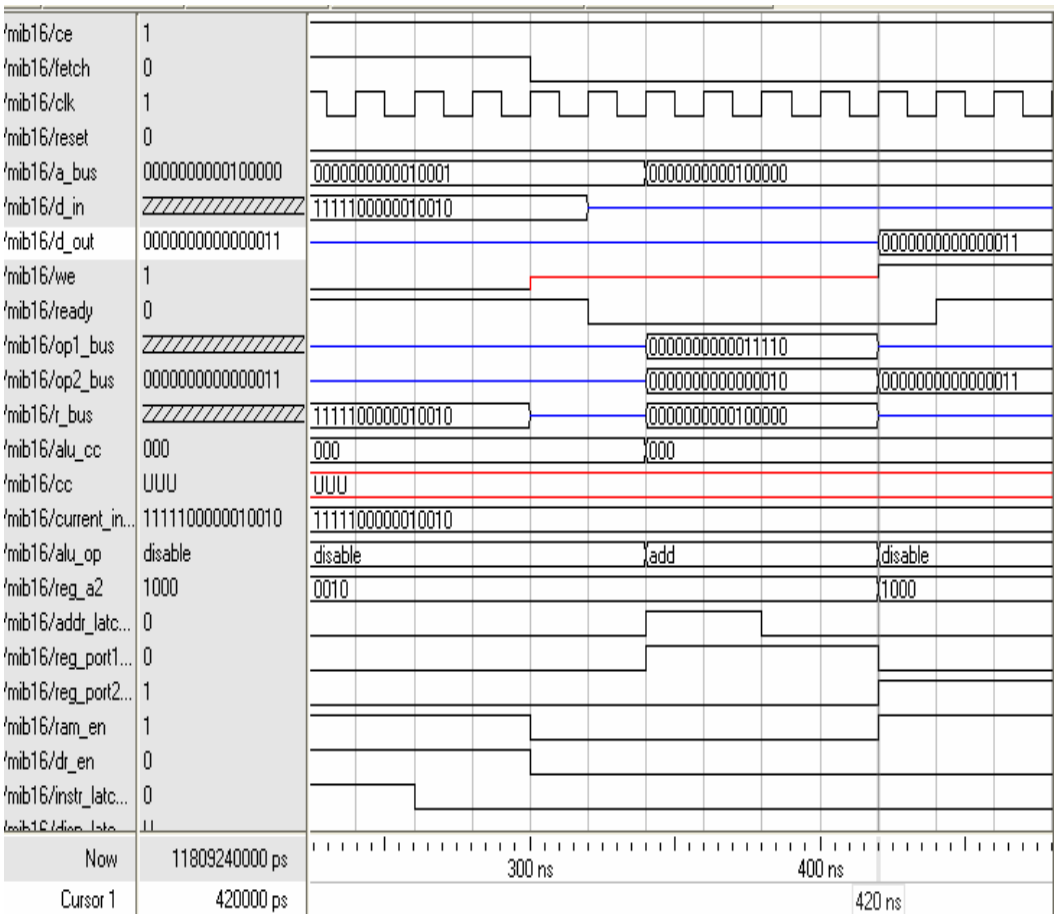


Şekil 4.24. İvedi Yükleme komutuna ait sinyal değişimleri

İvedi Yükleme işlemi, ilgili komutun hafızadan okunmasıyla başlar. 16-bitlik komutun son dört bitindeki “0010” 4-bitlik verisinin 2’ye tümleme işlemiyle 16-bitlik veriye genişletilmiş şekli olan “0000000000000010” verisi op2_bus’ a alınır. Bu veri, hafızadan yükleme yapılacak verinin adresini bulmada indis adres görevini üstlenir. op1_bus’taki adres ile op2_bus’ta bulunan adres değeri toplanarak hafızadan yükleme yapılacak verinin adresi elde edilir. Hafızadan okuma yapılacağı için “we” sinyali “0” olur ve belirtilen adresteki veri, mikroişlemcinin giriş veri yolu olan “d_in” sinyaline alınır. Şekil 4.24’de hafızada “0000000000100000” adresinde bulunan “0000100000010010” verisi, “d_in” girişine yüklenmiştir.

4.2.18. İvedi Saklama komutunun doğrulanması

Şekil 4.25’de İvedi Saklama komutuna ait sinyal değişimleri görülmektedir.



Şekil 4.25. İvedi Saklama komutuna ait sinyal değişimleri

İvedi Saklama işlemi, ilgili komutun hafızadan okunmasıyla başlar. 16-bitlik komutun son dört bitindeki "0010" 4-bitlik verisinin 2'ye tümleme işlemiyle 16-bitlik veriye genişletilmiş şekli olan "0000000000000010" verisi op2_bus'a alınır. Bu veri, hafızaya saklanacak verinin adresini bulmada indis adres görevini üstlenir. op1_bus'taki adres ile op2_bus'ta bulunan adres değeri toplanarak hafızaya saklanacak verinin adresi elde edilir. Hafızaya yazma işlemi yapılacağı için "we" sinyali "1" olur. "1000" adresli kaydedicideki veri alınarak mikroişlemcinin çıkış veri yolu olan "d_out" sinyaline alınır ve belirtilen adrese yazılır. Şekil 4.25'de "1000" adresli kaydedicide bulunan "0000000000000011" verisi, hafızanın "0000000000100000" adresine yazılarak bu komut sayesinde saklanmıştır.

5. SONUÇLAR VE ÖNERİLER

Bu tezde 16-bitlik mikroişlemcinin tüm alt birimleriyle tasarımı ve simülasyonu yapılmıştır. 16-bit mikroişlemci, Xilinx ISE 6.3i programı kullanılarak VHDL (Very High Speed Integrated Circuit Hardware Description Language) dilinde gerçekleştirilmiş ve Xilinx Spartan-3 deneme kartına yüklenerek kartın izin verdiği ölçüde sınırlamalar altında denenmiştir. Kartın üzerindeki giriş anahtarlarından veri girişi yapılmış, sonuçlar ise 8 adet LED ve 4 haneli 7 segmentli display üzerinde gözlenmiştir.

Çarpma işlemi, sonucu 16 biti aşmayan sayılar için; Bölme işlemi ise kalansız bölünen sayılar için gerçeklenmiştir. Tasarlanan mikroişlemci, Toplama, Çıkarma, Çarpma, Bölme ve Lojik işlemlerini 340 nsn (nano saniye)'de; Yükleme ve Saklama işlemlerini 660 nsn'de; İvedi Yükleme ve İvedi Saklama işlemlerini 440 nsn'de gerçekleştirmektedir. Sonuç olarak mikroişlemcinin performansı, Aritmetik ve Lojik işlemler için yaklaşık 3 MHz, Yükleme ve Saklama işlemleri için yaklaşık 1,5 MHz ve İvedi Yükleme ve İvedi Saklama işlemleri için yaklaşık 2,3 MHz'dir.

Tasarlanan 16-bitlik mikroişlemcinin saat frekansı, yükleme yapılan Spartan-3 XC3S200-FT256 bordunun saat frekansı olan 50 MHz'e ayarlanmıştır. Bu da mikroişlemcinin saat frekansını sınırlamaktadır. Tasarlanan 16-bitlik mikroişlemci 16-bit adres yoluna sahip olmasına rağmen yükleme yapılan bordun en büyük kapasiteli RAM'i, 10-bit adres yolu kullandığı için, 16-bitlik adres yolu, 10-bitlik adres yoluna dönüştürülmüştür; bu özellik de hafıza kapasitesini sınırlamaktadır.

Gerçeklenen mikroişlemcinin performansını iyileştirmek için daha üst seviyeli bir bord (Xilinx Virtex II-Pro, Virtex-4 gibi) kullanılabilir. Mikroişlemciye, Kayan Nokta Birimi (FPU) eklenerek kalanlı bölme işlemi de yaptırılabilir. Çarpma işleminin performansı da hızlı çarpma algoritmaları gerçeklenerek arttırılabilir. Lisansı ve gerekli donanımı temin edildiği taktirde Xilinx firmasının sunduğu Chip

Scope Pro yazılımı ile Giriş / Çıkış ve Kontrol sinyallerinin osiloskop görüntüleri kolaylıkla bilgisayara aktarılabilir.

Bu çalışma Xilinx ISE 6.3i programı kullanılarak gerçeklenmiştir. Aynı çalışma Altera, Quick Logic, Lattice gibi farklı firmaların programları kullanılarak da tasarlanabilir.

Tasarlanan mikroişlemci, VHDL dilinde yazıldığından geliştirilmeye açıktır. Kullanım alanlarına göre komut seti değiştirilebilir, veri ve adres yolu uzunlukları artırılabilir. Tasarıma çevre birimleri eklenerek mikro denetleyici içerisinde kullanılabilir. Ayrıca Assembler derleyici geliştirilerek tasarlanan mikroişlemci, programlamaya uygun hale getirilebilir.

KAYNAKLAR

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

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

[3] http://en.wikipedia.org/wiki/Intel_4004, **(Ziyaret Tarihi: 13 Aralık 2006).**

[4] David A. Patterson, John L.Hennessy., "Computer Organization and Design: The Hardware/Software Interface", 1996.

[5] Dr.Haluk Gümüşkaya., "Mikroişlemciler ve Bilgisayarlar", Alfa Basım Yayım, 12-65, (2000).

[6] Borgatti, M., Lertora, F., Foret, B., Cali L., "A Reconfigurable System Featuring Dynamically Extensible Embedded Microprocessor, FPGA and Customizable I/O", ***IEEE Custom Integrated Circuits Conference,*** 13-16, (2002).

[7] Ireneusz Janiszewski, Robert Baraniecki, Krystyna Siekierska, "A reusable microcontroller core's design", ***IEEE, VHDL International Users Forum Fall Workshop (VIUF '99),*** 14-21, (1999).

[8] Murat, Çakıroğlu., "Gerçek Zaman Sayma Birimi İçeren SAU80C51 Mikro denetleyicisinin FPGA Mimarileri Kullanılarak Geliştirilmesi", Yüksek Lisans Tezi, ***Sakarya Üniversitesi Fen Bilimleri Enstitüsü***, Sakarya, 29-32, (2003).

[9] Guthikonda V.Rao,Ph.D., "Microprocessors and Microcomputer Systems", 1982.

[10] Thomas Richard McCalla., "Digital Computer and Logic Design", 1992.

[11] Esma Esen., Şule Çetin., "4-bitlik Mikroişlemcinin Tasarımı", Bitirme Tezi, ***Kocaeli Üniversitesi Elektronik ve Haberleşme Mühendisliği***", Kocaeli, 2003.

[12] Ashenden, Peter J., "The VHDL Cookbook", 1990

[13] Xilinx ISE 6 Software Manuals and Help

[14] Spartan-3 FPGA Family: Complete Data Sheet

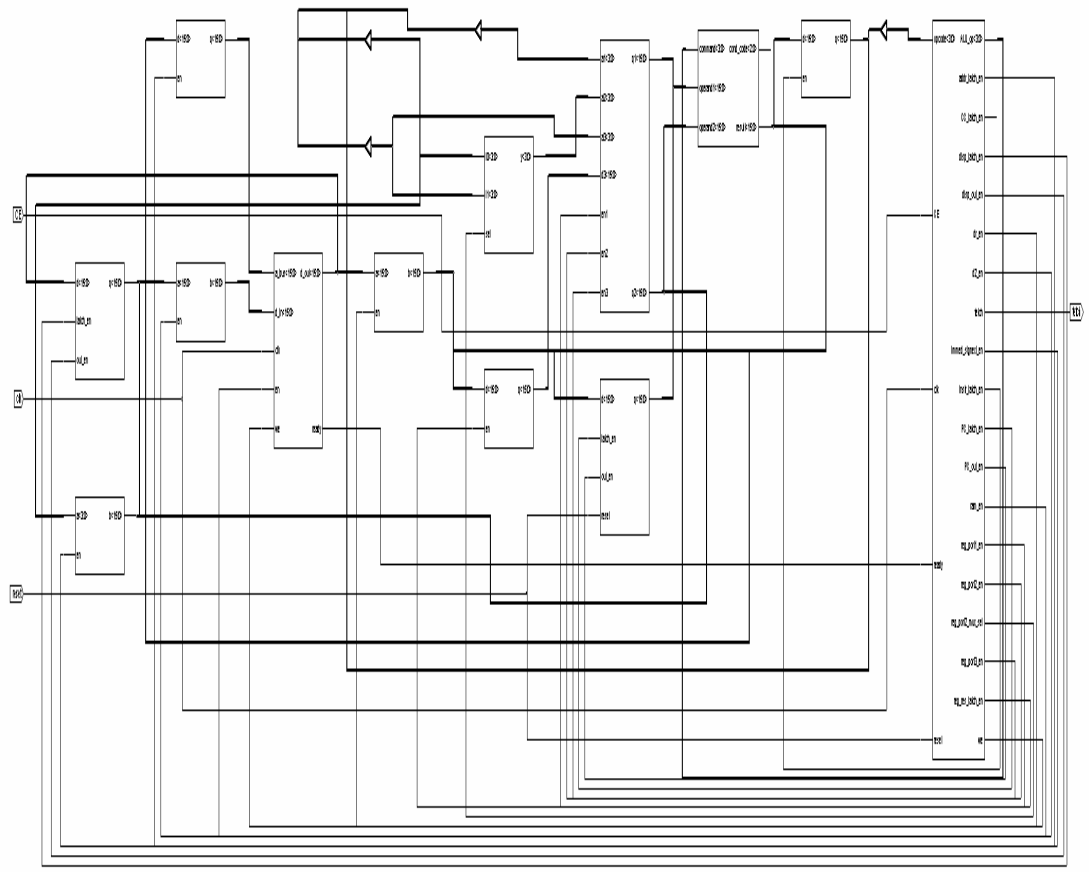
[15] Spartan-3 FPGA Family: Pinout Descriptions

[16] http://www.digilentinc.com/Products/Detail.cfm?Nav1=Products&Nav2=Programmable&Prod=S3BOARD, **(Ziyaret Tarihi: 21 Mart 2006).**

[17] http://www.intel.com/pressroom/kits/quickrefyr.htm, **(Ziyaret Tarihi: 13 Aralık 2006).**

[18] http://www.xilinx.com, **(Ziyaret Tarihi: 10 Şubat 2006).**

EK A. 16-BİTLİK MİKROİŞLEMCİNİN ŞEMATİK GÖSTERİMİ



Şekil A.1. 16-bitlik mikroişlemcinin şematik gösterimi

EK B. FBGA PROGRAMLANABİLİR MİMARİLERİ VE SPARTAN AİLESİ HAKKİNDA GENEL BİLGİ

B.1. Programlanabilir Devre Elemanları ve Gelişimleri

Programlanabilir yapılar, uygulamaya yönelik olarak programlanabilen genel amaçlı tümleşik devrelerdir. Programlanabilir yapılar sırasıyla şu şekilde bir gelişim süreci geçirmişlerdir;

1. Programlanabilir Salt Okunur Bellekler (Programmable Read Only Memory,PROM)

2. Silinebilir ve Programlanabilir Salt Okunur Bellekler (Erasable Programmable Read Only Memory, EPROM)

3. Elektriksel Silinebilir ve Programlanabilir Salt Okunur Bellekler (Electronically Erasable Programmable Read Only Memory, EEPROM)

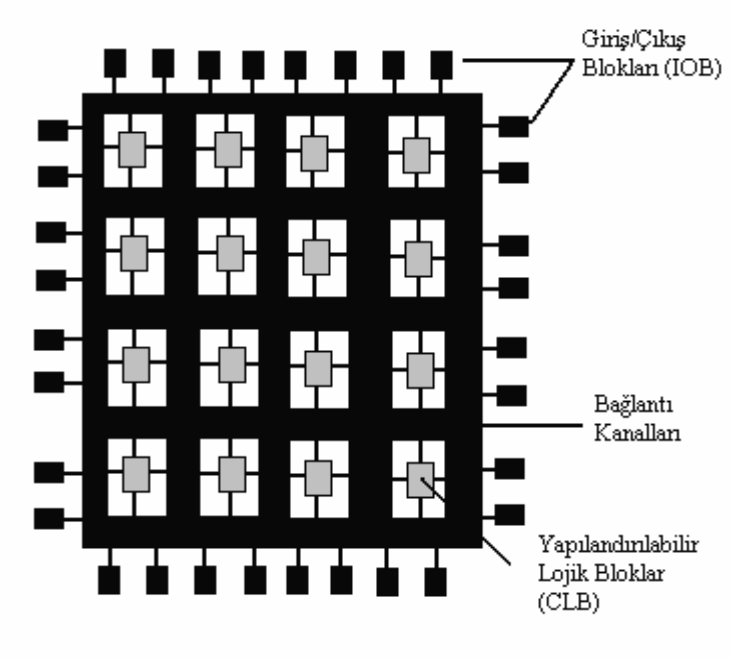
4. Programlanabilir Lojik Devreler (PLD). PLD'ler; Programlanabilir Dizi Lojiği (Programmable Array Logic, PAL) ve Programlanabilir Lojik Dizisi (Programmable Logic Array,PLA) olmak üzere iki ayrı kısımda incelenir.

5. Maske Programlanabilir Kapı Dizileri (Mask Programmable Gate Array,MPGA)

MPGA ve PLD'leri üstünlüklerini kullanarak ve zayıflıklarını ortadan kaldırarak 1985 yılında Xilinx firması tarafından Alan Programlanabilir Kapı Dizileri (FPGA) geliştirilmiş ve piyasaya sürülmüştür. FPGA'ler programlanabilir lojik bloklar ve ara bağlantılardan oluşmuştur. Kullanıcının tasarladığı lojik devreye göre, üretici tarafından sağlanan bir yazılım sayesinde lojik bloklar ve aralarındaki bağlantılar programlanır.

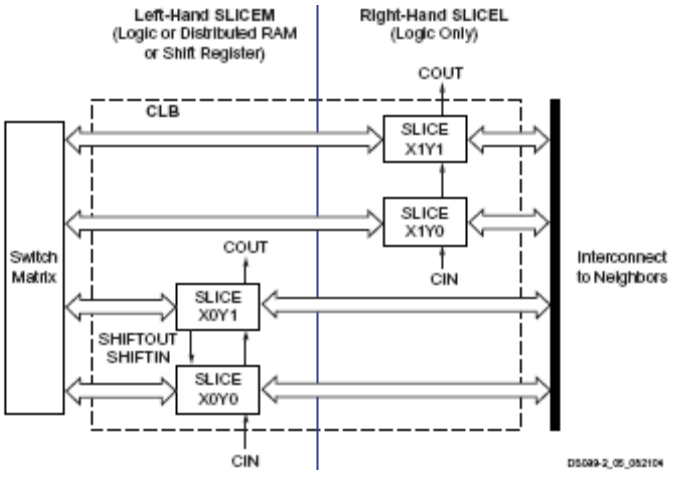
B.2. Alan Programlanabilir Kapı Dizileri(FPGA)

Merkezinde Lojik Bloklar, Bağlantı Kanalları ve Giriş/Çıkış Bloklarından oluşan FPGA'lerin basitleştirilmiş iç yapıları şekil B.1'de verilmiştir.



Şekil B.1. Bir FPGA'nin basitleştirilmiş iç yapısı

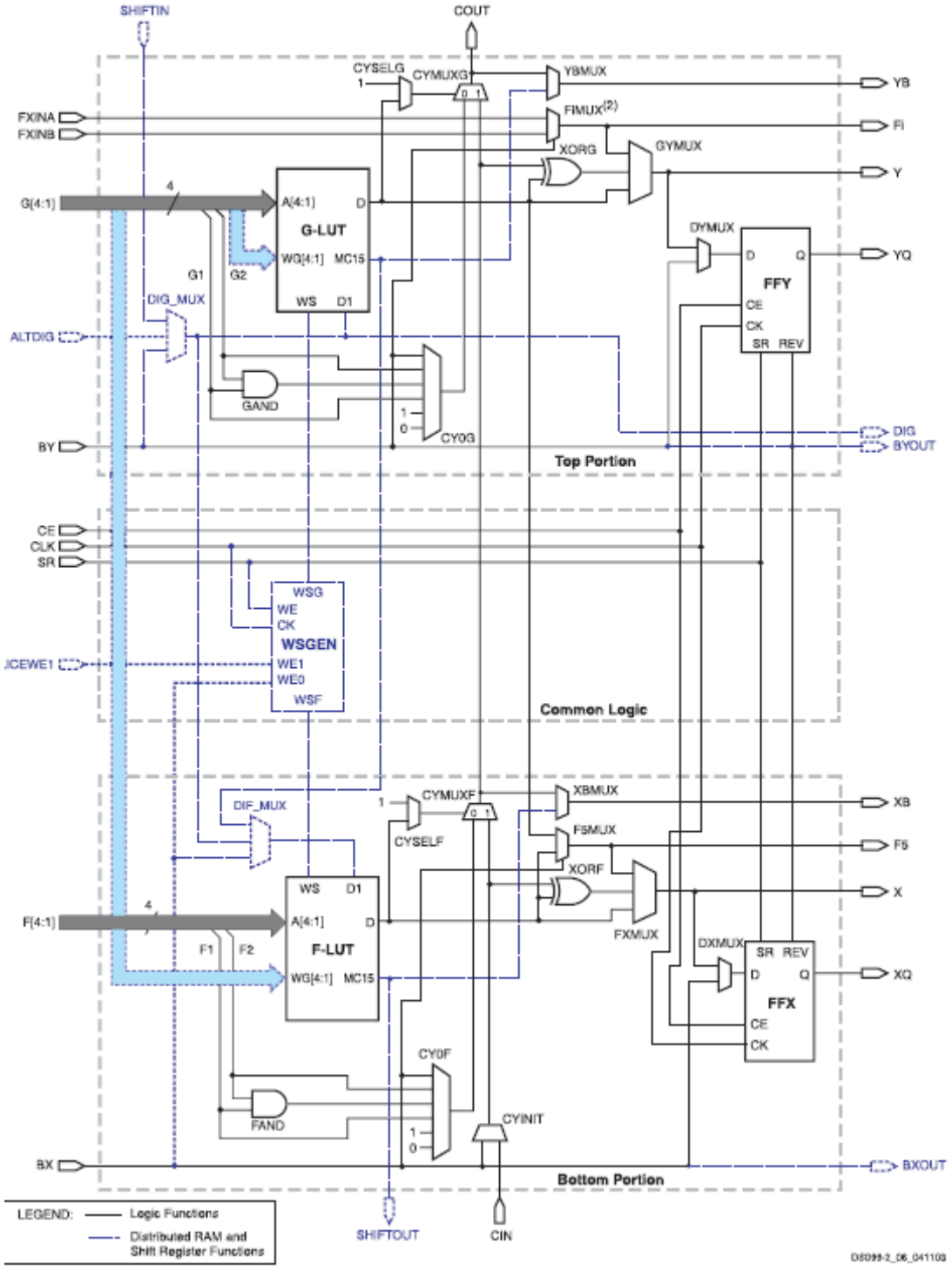
Yapılandırılabilir lojik bloklar, mantıksal fonksiyonların gerçeklendiği yapılardır. Spartan-3 ailesinde CLB'ler (Complex Logic Block) dizi şeklinde sıralanmış yapıdadır. Her bir CLB elemanı Genel Bağlantı Matrisine erişebilmek için Switch Matrise bağlanmıştır. CLB hücreleri içlerinde hızlı geri beslemesi olan 4 ayrı lojik birimden (slice) oluşur. Bu parçaların, Şekil B.2'de görüldüğü gibi, bağımsız birer elde lojik bağı ve bir tane de ortak bağları bulunmaktadır.



Şekil B.2. Spartan-3 CLB elemanının basitleştirilmiş şeması [14]

Her bir CLB'nin içerisinde 2 tane 4-girişli fonksiyon jeneratörü, elde ve aritmetik lojik kapıları, fonksiyon çoğullayıcıları, 2 tane depolama elemanı vardır. Her bir 4-girişli fonksiyon jeneratörü; 4-girişli LUT (Look-Up-Table), 16 bitlik seçilebilir RAM hafızası veya 16 bitlik değişken kademeli saklayıcı olarak programlanabilir.

Fonksiyon jeneratörünün çıkışları, lojik birimlerinin çıkışlarını ve depolama elemanın girişlerini oluşturur. Bu olay Şekil B.3'de daha ayrıntılı olarak görülmektedir.



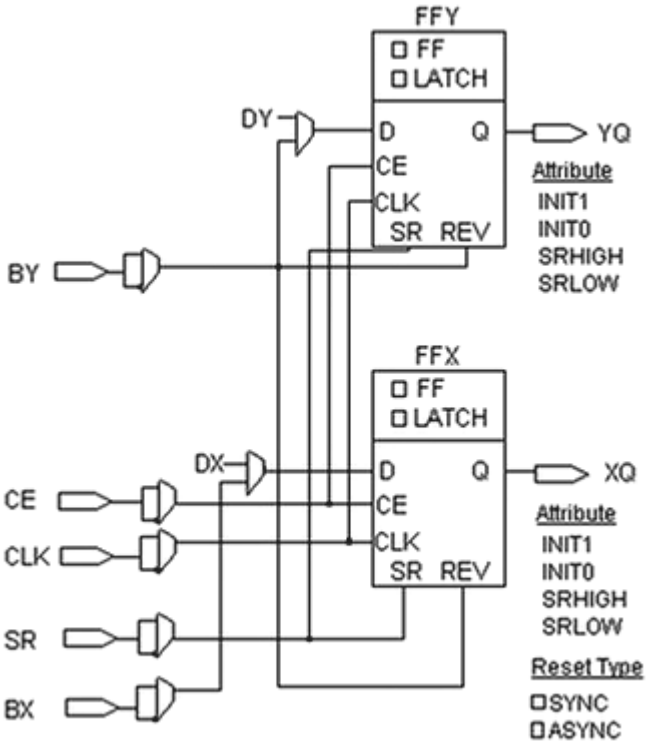
Şekil B.3. Lojik birim (slice) içerisindeki elemanlar [14]

B.2.1. Look-Up Table

Spartan-3 fonksiyon jeneratörleri, 4-girişli bir Look-Up-Table(LUT) olarak çalışırlar. Bu bağımsız 4 giriş, mantıksal işlemleri gerçekleştirebilecek fonksiyon jeneratörlerini oluşturur.

B.2.2 Register/Latch

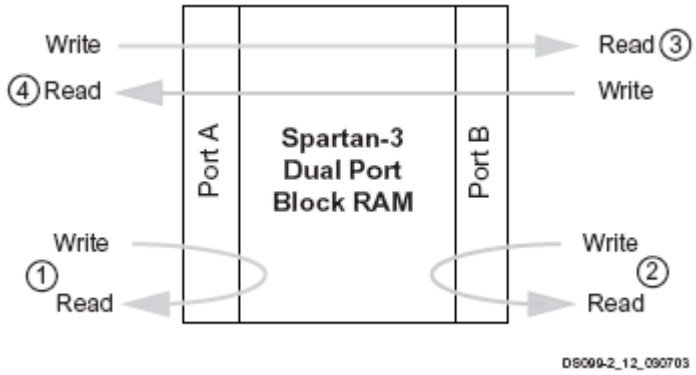
Spartan 3 hafıza elemanı kenar tetiklemeli D tipi flip-flop kullanılarak tasarlanmıştır. D girişleri direk olarak X veya Y çıkışlarında DX veya DY girişleri kullanılarak sürülebileceği gibi, fonksiyon jeneratörlerini geçen birim girişleri yolu ile BX veya BY girişleri kullanılarak da sürülebilir.



Şekil B.4. Mantıksal Birimdeki Register / Latch Konfigürasyonu [14]

B.2.3. Dağıtılmış seçilebilir RAM bellek

Her bir fonksiyon jeneratörü, dağıtılmış seçilebilir RAM bellek denilen eş zamanlı RAM kaynaklarından oluşur. Seçilebilir RAM elemanları aşağıda gösterildiği gibi CLB'lerle ayarlanabilir.



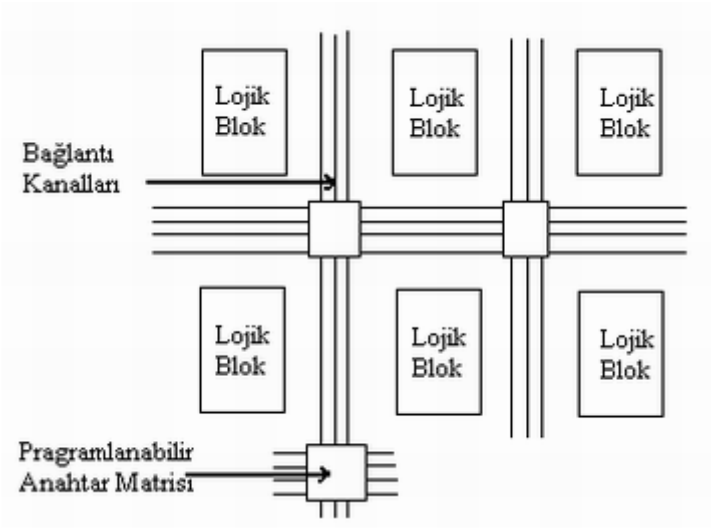
Şekil B.5. Blok RAM'e ait veri yolları [14]

- Tek Portlu 16x1 bit RAM
- Tek Portlu 32x1 bit RAM
- Tek Portlu 64x1 bit RAM
- Tek Portlu 128x1 bit RAM
- Çift Portlu 16x1 bit RAM
- Çift Portlu 32x1bit RAM
- Çift Portlu 64x1 bit RAM

Dağıtılmış seçilebilir RAM bellekler eş zamanlı kaynaklardır. Kombinasyonel okuma erişim zamanları gerçekten hızlıdır. Böylece eş zamanlı yüksek hızda yazma işlemine olanak tanımaktadır.

B.3. FPGA'lerin Programlanma Teknolojileri

FPGA mimarileri bağlantı kanallarının yapısına göre Simetrik Dizi Mimarisi, Sıra Tabanlı Mimari ve Kapı Denizi Mimarisi olmak üzere 3 ana gruba ayrılır. Bu mimarilerden en çok kullanılan Simetrik Dizi Mimarisi aşağıda Şekil B.6'da verilmiştir. Spartan 3 ailesi FPGA'lerde de bu tür mimariler kullanılmıştır.



Şekil B.6. Simetrik Dizi Mimarisi

FPGA'lerin programlanabilirlik özellikleri, içerdikleri programlanabilir anahtarlardan ve lojik bloklardan gelmektedir. Bu anahtarların programlanma tekniğine göre, FPGA'lerin programlanma teknolojileri belirlenir. Bu programlama için dört teknoloji kullanılmaktadır. Bunlar Statik RAM, Antifuse, EPROM ve EEPROM programlama teknolojileridir. Uygulama cinsine göre belli bir teknolojinin bazı özellikleri olabilir. Buna göre uygulama için en uygun olan teknoloji seçilir.

Programlanabilir anahtarlardan istenen özellikler aşağıdaki gibidir:

- Çok sayıda programlanabilir anahtar, entegre içinde gerçeklenebilmeli.
- İşlev gördükleri bağlama kanallarında minimum parazittik kapasite oluşturmaları.
- İletim durumunda küçük, kesim durumunda ise çok yüksek direnç değeri göstermeleri.
- Yarı iletken üzerinde mümkün olduğunca küçük alan kaplamaları.

B.3.1. Statik RAM programlama teknolojisi

Statik RAM (SRAM) programlama teknolojisinde, programlanabilir bağlantılar SRAM hücresi tarafından kontrol edilen geçiş transistoru, iletim kapısı veya çoklayıcı yapısında gerçeklenmektedir.

Geçiş transistorunun bağlantı telleri arasında, yüksek direnç gösterdiği konumda bağlantı kanalları açık, düşük direnç gösterdiği durumda ise bağlantı kanalları kısa devre kabul edilir. Çoklayıcı kullanılan yapılarda is SRAM hücresi, girişin hangi çıkışa bağlanacağını kontrol eder. Fakat SRAM hücreleri, gerilim uygulanmadığı taktirde silindiği için, devrenin gerilimi kesildiğinde, her defasında bu hücrelerin tekrardan programlanmaları gerekmektedir.

SRAM hücrelerinin yarı iletken üzerinde çok büyük yer kaplamasına rağmen, FPGA'lerin sistem üzerindeyken tekrar programlanabilmeleri için bu teknoloji gerekmektedir.

B.3.2. Antifuse programlama teknolojisi

Antifuse teknolojisinde, FPGA programlanmadan önce, bağlama kanalları arasındaki bağlantılar kurulmamış durumdadır. Programa esnasında uygulanan gerilim ile gerekli bağlantılar oluşturulur. Bu tür programlamada, programlanma süresi daha kısadır.

Bu teknolojide, programlamadan önce metal kontaklar arasında dielektrik madde vardır. Yeterli gerilim verildiği taktirde bu dielektrik madde eriyerek iletkenlerin birbirine temas etmesini sağlar. Böylece kontaklar arsında akım geçebilecek bir bağlantı oluşur.

Antifuse programlama tekniği ile programlanan FPGA'ler tekrar programlanamazlar. İşte bu yüzden dolayı, ilk örnek üretimi çok pahalı bir örnek olmaktadır. Diğer teknolojilerle karşılaştırıldığında, antifuse'ler daha küçük bir alana ihtiyaç duymaktadırlar.

B.3.3. EPROM ve EEPROM programlama teknolojisi

Bu programlama teknolojisinde kullanılan yapı, EPROM belleklerde kullanılan yapıya benzemektedir. MOS transistorlardan farklı olarak, bir EPROM transistorda seçme geçidi (select gate) ve yüzen geçit (floating gate) bulunur. Yüzen geçidin hiçbir bağlantısı yoktur ve programlanmadığı taktirde üzerinde hiçbir yük bulundurmaz transistor programlanmak istendiğinde, kaynak ve savak arasında bir akım akıtılır. Bu akıtılan akım, yüzen geçitle dielektrik arasında yük tutulmasını ve transistorun her zaman iletimde kalmasını sağlar. Yüzen geçitteki yükün boşaltılmasıyla transistorun yeniden programlanabilir hale gelmesi sağlanır. Bu yükün boşaltılması elektrikle veya morötesi ışıkla sağlanabilir.

Tablo B.1'de FPGA'lerde kullanılan programlama teknikleri ve özellikleri verilmiştir.

Tablo B.1. FPGA'lerin programlama teknikleri ve özellikleri

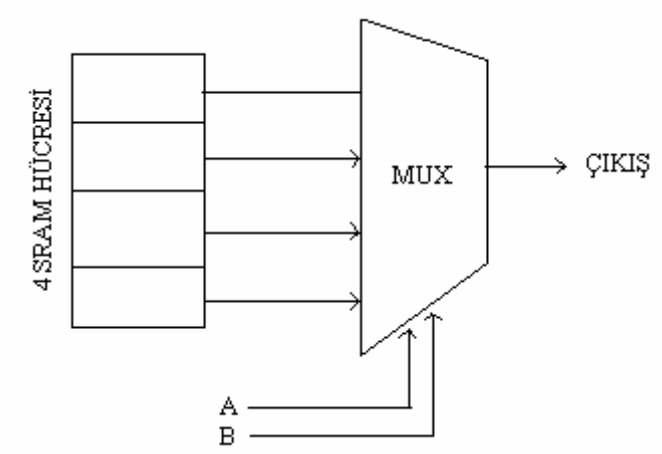
Programlama Teknolojisi	Uçuculuk	Tekrar Programlanabilme	Silisyum Alanı
SRAM	Evet	Evet (Devre Üzerinde)	Büyük
ANTİFUSE	Hayır	Hayır	Küçük
EPROM	Hayır	Evet (Devre Dışında)	Küçük
EEPROM	Hayır	Evet (Devre Üzerinde)	2 x EPROM

B.4. FPGA'lerin Lojik Hücre Yapısı

FPGA'ler lojik hücre yapısına göre; Doğruluk Tablosu Tabanlı ve Çoklayıcı Tabanlı olmak üzere iki ana gruba ayrılırlar.

B.4.1. Doğruluk tablosu tabanlı yapı

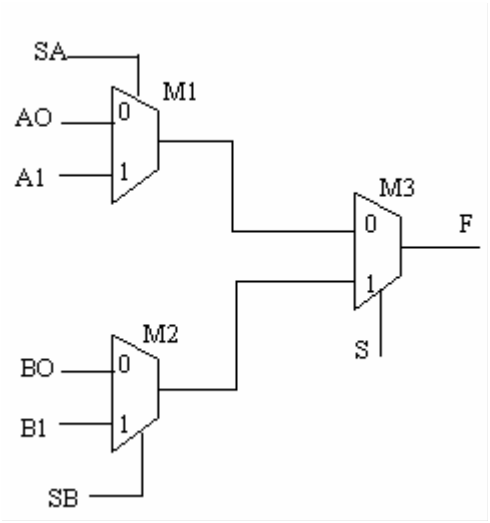
Bu yapının temel bloğu, LUT (Look-Up-Table) adı verilen ve en az 2-değişkenli her mantıksal işlemleri gerçekleyebilen bir devredir. Genelde "m" tane adres, bir tane de veri yolu olan SRAM hücresi ile gerçekleştirilir ve "mLUT" olarak adlandırılır. Her temel blok bir veya daha fazla LUT ile flip-flop gibi diğer elemanlardan oluşur. Şekil B.7'de 2 girişli bir LUT yapısı görülmektedir.



Şekil B.7. 2 girişli LUT yapısı

B.4.2. Çoklayıcı tabanlı yapı

Bu tür yapılarda çeşitli konfigürasyonlar ve olabildiğince az sayıda VE ile VEYA gibi lojik kapılar kullanılır. Bu tür yapıdaki FPGA'lerde latch ve flip-flop gibi elemanlar bulunmadığı için, lojik işlemler çoklayıcı tabanlı yapılarda gerçekleştirilir. Şekil B.8de çoklayıcı tabanlı lojik hücre yapısı verilmiştir.



Şekil B.8.Çoklayıcı tabanlı lojik hücre yapısı

B.5. Spartan-3 Field Gate Array (FPGA) Ailesi Üyeleri

Buraya kadar olan kısımda FPGA yapıları hakkında genel bilgi verilmiştir. Bu kısımda ise Spartan-3 ailesi hakkında bilgi verilecektir. Yapılan çalışmada kullanılmış olan FPGA elemanı XCS200 dür. Tablo B.2'de Spartan-3 ailesi elemanları ve özellikleri verilmiştir.

Tablo B.2. Spartan-3 ailesi elemanları ve özellikleri [14]

Eleman	Sistem Kapıları	Lojik Hücreler	Maksimum Kullanılabilir I/O	Blok RAM Bitleri	Dağıtılmış RAM Bitleri
XC3S50	50K	1,728	124	72K	12K
XC3S200	200K	4,320	173	216K	30K
XC3S400	400K	8,064	264	288K	56K
XC3S1000	1000K	17,280	391	432K	120K
XC3S1500	1500K	29,952	487	576K	208K
XC3S2000	2000K	46,080	565	720K	320K
XC3S4000	4000K	62,208	712	1728K	432K
XC3S5000	5000K	74,880	784	1872K	520K

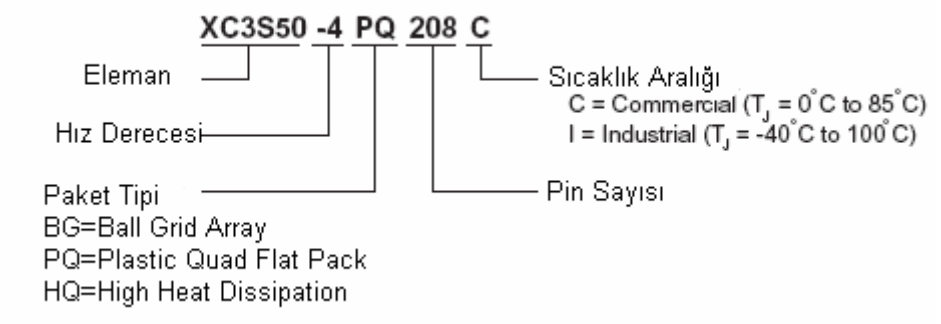
B.6. Spartan-3 Ailesi Elemanlarının Farklı Paketlerinin Giriş / Çıkış Hatları

Tablo B.3. Spartan-3 ailesi elemanlarının kullandıkları maksimum Giriş / Çıkış hatları [14]

Paket	XC3S50	XC3S200	XC3S400	XC3S1000	XC3S1500	XC3S2000	XC3S4000	XC3S5000
VQ100	63	63	---	---	---	---	---	---
CP132	89	---	---	---	---	---	---	---
TQ144	97	97	97	---	---	---	---	---
PQ208	124	141	141	---	---	---	---	---
FT256	----	173	173	173	---	---	---	---
FG320	---	----	221	221	221	---	----	---
FG456	---	---	264	333	333	333	---	---
FG676	---	---	---	391	487	489	489	---
FG900	---	---	---	---	---	565	633	633
FG1156	---	---	---	---	---	---	712	784

B.7 Spartan-3 Ailesi Elemanlarının Kod Bilgileri

Spartan-3 ailesi elemanları da bilinen diğer elektronik devre elemanları gibi üzerlerinde bulunan kodlarla ifade edilirler. Bu kodlar kullanıcıya bir nevi kılavuzluk eder yani elemanın tipi, paket tipi, pin sayısı gibi önemli bilgileri kullanıcıya sunar. Örnek olarak Spartan-3 ailesinden XC3S50 elemanının kodları Şekil B.9'da verilmiştir.

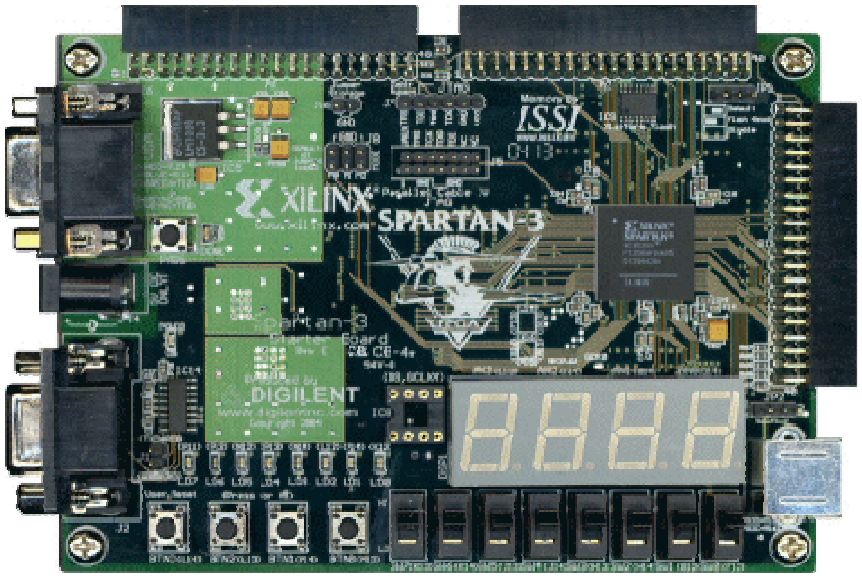


Şekil B.9. XC3S50 elemanının kod bilgileri

EK C. GELİŞTİRME DONANIMININ YAPISI

C.1. Deneme Kartı

Geleneksel yaklaşımla yapılan deneylerde kullanılan, üzerinde entegre ve elektrik düzeneği bulunduran deney setleri hem pratik değiller hem de sonuçlar yeterince etkili değildir. Artık bu deney setleri yerine, zamandan ve paradan kazanç sağlayan PC kartlar kullanılmaktadır. Şekil C.1'de görülen Xilinx firması tarafından üretilen ve yapılan çalışmanın uygulama alanında kullanılan Spartan-3 XC3S200-FT256 deneme kartı da bu kartlardan biridir.



Şekil C.1. Spartan-3 XC3S200-FT256 kartı [16]

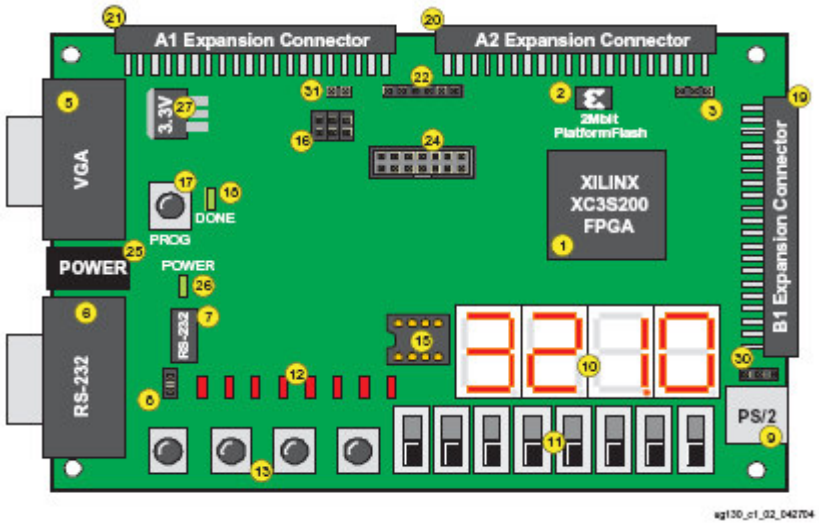
Kartın özellikleri şu şekildedir:

- 12 tane 18 bit çoklayıcı, 216Kbit blok RAM ve 500MHz'e kadar çıkabilen saat frekans hızı
- Bord üzerinde 2Mbit Platform Flash Rom (XCF02S)
- 9 led,4 rakamlı 7 segmentli gösterge,4 buton ve 8 tane kayan anahtar
- seri port,VGA port ve PS/2 fare/klavye port

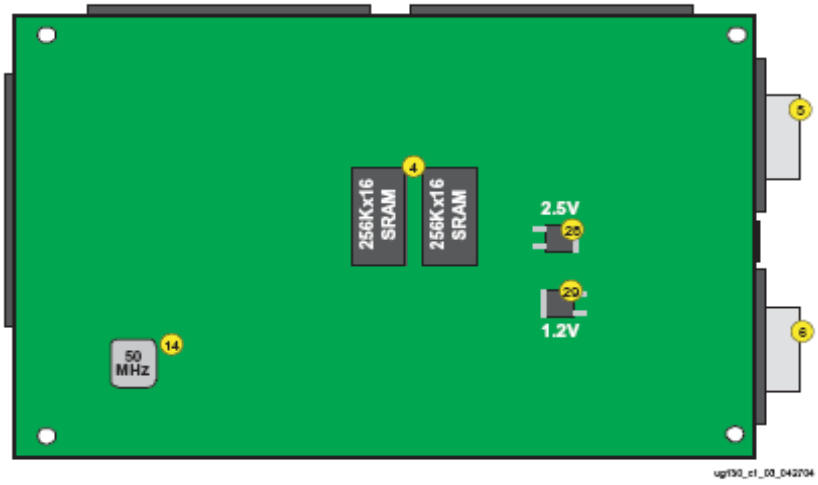
- 3 tane 40 bacaklı genişleme konnektörü
- 3 tane yüksek-akım gerilim düzenleyici (3.3V, 2.5V ve1.2V)
- JTAG3 programlama kablosu ve P4& MultiPRO kablolarla çalışma
- 1Mbyte board üzerinde 10ns SRAM (256 x 32)

C.2. Kart Hakkında detaylı Bilgi

Şekil C.2 ve Şekil C.3 Spartan-3 kartındaki elemanların yerleşimini göstermektedir.



Şekil C.2. Spartan-3 kartının üst kısmı [14]



Şekil C.3. Spartan-3 kartının alt kısmı [14]

- 200.000 kapıya sahip Xilinx Spartan-3 XC3S200 FPGA (1)
- 12 adet 18K-bit blok RAM (216K bits)
- 12 adet 18x18 çoklayıcı
- 4 adet dijital saat yönetici (DCMs)
- 173 adet kullanıcı tanımlı Giriş/Çıkış sinyali

- 2Mbit XilinxXCF02S Platform Flaş,PROM (2)
- 1 M-byte Hızlı Asekron SRAM (4)
- 3-bit, 8 renkli VGA gösterge portu (5)
- 9 bacaklı RS-232 seri port (6)
- RS-232 çevirici (7)
- PS/2 fare/klavye portu (9)
- 4 karakter, 7 segmentli LED gösterge (10)
- 8 adet kayan anahtar (11)
- 8 adet LED çıkışı (12)
- 4 adet buton (13)
- 50 MHz kristal osilatör saat kaynağı (14)
- yardımcı kristal osilatör saat kaynakları için soket (15)
- FPGA'nin yeniden konfigürasyonunu sağlayan buton (17)
- FPGA'nin başarılı bir şekilde yüklendiğini gösteren LED (18)
- Spartan-3 bordunu genişletmek amacıyla kullanılabilen 3 adet 40 bacaklı bağlantı portu (19) (20) (21)
- Düşük maliyetli yükleme kablolar (23) için JTAG portu (22)
- + 5 V AC güç adaptörü girişi (25)
- Devreye bağlı gücün açık olduğunu gösteren LED (26)
- Bord üzerinde 3,3 V (27), 2,5 V (28) ve 1,2 V (29) gerilim düzenleyiciler

ÖZGEÇMİŞ

Esma ALAER, 1981 yılında Afyon'da doğmuştur. İlk öğrenimini Bilecik Edebali İlkokulunda, orta öğrenimini Bilecik Anadolu Lisesi'nde tamamlamıştır. Lise eğitimini Afyon Süleyman Demirel Fen Lisesi'nde tamamladıktan sonra Kocaeli Üniversitesi Elektronik ve Haberleşme Mühendisliği Bölümü'ne girmiştir. 2003 yılında bu bölümden mezun olmuştur. 2003-2006 tarihleri arasında aynı üniversitede yüksek lisans eğitimini tamamlamıştır.

Yayınlar

Esma Esen., Şule Çetin., "4-bitlik Mikroişlemcinin Tasarımı", Bitirme Tezi, ***Kocaeli Üniversitesi Elektronik ve Haberleşme Mühendisliği***", Kocaeli, 2003.