T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI SENTEZLENEBİLİR İŞLEMCİ TASARIMI

SELÇUK BAŞAK

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMAN
YRD. DOÇ. DR. SONGÜL ALBAYRAK

İSTANBUL, 2011

T.C.

YILDIZ TEKNİK ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

FPGA TABANLI SENTEZLENEBİLİR İŞLEMCİ TASARIMI

Selçuk BAŞAK tarafından hazırlanan tez çalışması 10.05.2011 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Yrd. Doç. Dr. Songül ALBAYRAK

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Yrd. Doç. Dr. Songül ALBAYRAK

Yıldız Teknik Üniversitesi _____

Prof. Dr. Nizamettin AYDIN

Yıldız Teknik Üniversitesi _____

Prof. Dr. Herman SEDEF

Yıldız Teknik Üniversitesi _____

ÖNSÖZ

Gömülü sistemler otomotiv, yayıncılık, tüketici elektroniği, endüstriyel uygulamalar, sağlık, savunma sanayi, iletişim gibi bir çok alanda yaygın olarak kullanılmaktadır. Gömülü sistemler, hazır standart entegreler kullanarak geliştirilen kartlar ile, uygulamaya özel olarak tasarlanacak entegre devre ile gerçeklenebilir. Uygulamaya özel entegre kullanımı, tasarlanmak istenen sistemin ihtiyaçlarına esnek bir altyapı sunmaktadır. Maske programlı olarak uygulamaya özel entegre devreyi üretmenin ekonomik olarak mümkün olabilmesi için satış rakamlarının milyonları bulması gerekmektedir. Tüketici elektroniği dışındaki sektörlere yönelik uygulamalarda maske programlama uygulamaya özel entegre üretmek ekonomik olmamaktadır. Az miktarda üretilecek ama yüksek performansa ihtiyaç duyan uygulamaya özel entegre devre programlanabilir lojik cihazlar ile gerçeklenmesi mümkün ve ekonomik olabilmektedir. Programlanabilir lojik cihazlar kaynakları ile sınırlı olarak hayalinizden geçen herhangi bir dijital sistemi programlanabilen devreler ile ifade etmenize imkan sağlar. Programlanabilen lojik cihazlar ile komple bir bilgisayar sistemini tek bir çip üzerinde tasarlamak mümkündür. Programlanabilen lojik cihazların asıl avantajı paralel olarak tüm sistemin aynı anda çalışması ile kazanılan performansıtır. Ancak yine de bu işlemleri organize etmek ve üst seviye programla dilleri ile ifade edilen programlar farklı fonksiyonlar kazanabilme gibi özellikler için bir merkezi işlemci birimine ihtiyaç duyulur. Programlanabilen lojik cihalar üzerinde gerçeklenecek işlemcinin performansı standart işlemciler ile kıyaslanamaz. Programlanabilen lojik cihalar üzerinde gerçeklenecek işlemcinin amacı fonksiyonel olmak ve minimum kaynak kullanımı ile maksimum performans sağlamak olmalıdır. Bu tez çalışmasında, 2008 yılında geliştirmiş olduğum ödüllü SelCPU işlemcisinden daha yüksek performanslı ve günümüzde programlanabilir cihazlar üzerinde kullanılan işlemcilerle yarışabilecek bir işlemciyi tasarlamak ve çalışmanın işlemci tasarımı konusunda bir referans olmasını istedim.

Nisan, 2011

Selçuk BAŞAK

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

$f$max	Sistemin kararlı olarak çalışabileceği maksimum frekans

KISALTMA LİSTESİ

ASIC	Application Specific Integrated Circuit
ASSP	Application Spesific Standart Product
CISC	Complex Instruction Set Computer
CLB	Configurable Logic Block
CPLD	Complex Programmable Logic Device
DLL	Delay Lock Loop
DMIPS	Dhrystone Million Instruction Per Second
DSP	Digital Signal Processing
EEPROM	Electrically Erasable Programmable Read Only Memory
EPROM	Erasable Programmable Read Only Memory
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
IEEE	Institute of Electrical and Electronics Engineers
LE	Logic Element
LUT	Lookup Table
MIPS	Million Instruction Per Second
MMU	Memory Management Unit
MPU	Memory Protection Unit
PAL	Programmable Array Logic
PAR	Place And Route
PLA	Programmable Logic Array
PLL	Phase Lock Loop
PROM	Programmable Read Only Memory
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
ROM	Read Only Memory
RTL	Register Transfer Level
SDRAM	Syncronous Dynamic Random Access Memory
SOC	System On A Chip
SOPC	System On A Programmable Chip
SPLD	Simple Programmable Logic Device
SRAM	Static Random Access Memory
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VLSI	Very large Scale Integration

ŞEKİL LİSTESİ

ÇİZELGE LİSTESİ

ÖZET

FPGA TABANLI SENTEZLENEBİLİR İŞLEMCİ TASARIMI

Selçuk BAŞAK

Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Tez Danışmanı: Yrd. Doç. Dr. Songül ALBAYRAK

Gömülü sistemler çok geniş kullanım alanına sahiptirler. Gömülü sistemlerin gerçeklenmesinde SOC (System On a Chip) yapıları yaygın olarak tercih edilmektedir. FPGA (Field Programmable Gate Array) gibi programlanabilir lojik cihazlar SOC yapılarını esnek olarak kullanmaya imkân sağlamaktadır. Birçok FPGA üreticisi FPGA içerisine gömülü olarak SOC yapıları yerleştirmek üzerine çalışmalar yapmaktadır. SOC sistemleri FPGA üzerinde soft-core olarak gerçeklemek mümkündür. Bu sistemlerde kullanılabilecek birçok ticari veya açık kaynaklı soft-core işlemci geliştirilmiştir.

Bu tez çalışmasında, FPGA üzerinde SelCPU2 soft-core işlemcisinin tasarımı yapılmış ve gerçeklenmiştir. SelCPU2 işlemcisinin tasarımında, Nios II işlemcisinin komut seti referans alınarak, az alanda yüksek komut çalıştırma performansı ve yüksek çalışma frekansı hedeflenmiştir. SelCPU2 işlemcisi komut set açısından RISC mimarisindedir. Hafıza erişimi açısından Harvard mimarisindedir, veri ve komut veri yolu ayrıdır. Hafıza erişimi load/store komutları ile yapılır. Yazmaç dosyası 32 adet 32-bit yazmaç içerir. Tek bir komutta iki kaynak ve bir de hedef yazmaç adresleyebilir. ALU komutları yazmaçlar ve/veya komut içinde belirtilen sabit değer üzerinde işlem yaparak sonucu hedef yazmaca aktarır. SelCPU2 iki saat döngüsünde üç aşamalı ardışık düzen işleme(pipeline) yapasındadır ve çoğu komut tek bir saat döngüsünde tamamlayabilir. SelCPU2 işlemcisi donanımsal çarpma ve bölme komutlarını destekler. Opsiyonel olarak veri önbelleği ve yerel bağlı hafıza birimi içerebilir. SelCPU2 işlemcisi SOC sistem veri yolu olarak Avalon-MM ara yüzünü destekler. Altera SOPC Builder aracı ile SOC tasarımda bileşen olarak kullanılabilir. SelCPU2 statik dallanma tahmini birimi içerir ve hatalı tahmin sadece bir saat döngüsü kayba neden olur. SelCPU2 işlemcisi için Assembler ve yeniden uyarlanabilir LCC C derleyicisine arka yüz geliştirilmiştir. Bu sayede ANSI C ile işlemci üzerinde çalışacak program geliştirme imkânı sağlanmıştır.

Yapılan tasarımın performans değerlendirmesi eşdeğer bir SOC sistemi üzerinde Altera Nios II işlemcisi ile karşılaştırmalı olarak yapılmıştır. GCC C derleyicisinin üçüncü optimizasyon seviyesi ile elde edilen ortak assembly kodu ile yapılan Dhrystone performans testinde SelCPU2 işlemcisi 47,43 DMIPS, Nios II/f işlemcisi ise 33,48 DMIPS sonuç almıştır. Sonuç olarak SelCPU2 işlemcisi, Nios II işlemcisine göre %41 daha yüksek performans göstermiştir. Altera Cyclone III FPGA üzerinde SelCPU2 işlemcisinin maksimum çalışma frekansı 85 MHz iken Nios II/f işlemcisinin maksimum çalışma frekansı 150 MHz'dir. Alan kullanımı açısından, SelCPU2 işlemcisi 1.771 LE, Nios II işlemcisinin 2.397 LE kullanmaktadır ve SelCPU2 %26 daha az alan kaplamaktadır. Tahmini güç tüketimi analizi sonucunda SelCPU2 208 mW ile Nios II işlemcisinden %4,3 daha az güç tüketmektedir. Tez çalışması sonucunda tasarlanarak gerçeklenen SelCPU2 işlemcisinin tasarım hedeflerini karşıladığı ve ticari bir işlemci olan Altera Nios II işlemcisine göre daha yüksek performans gösterdiği görülmüştür.

Anahtar Kelimeler: soft-core işlemci, programlanabilir lojik devre tasarımı, işlemci tasarımı

ABSTRACT

FPGA BASED SOFT-CORE PROCESSOR DESIGN

Selçuk BAŞAK

Department of Computer Engineering
MSc. Thesis

Advisor: Assist. Prof.Dr. Songül ALBAYRAK

Embedded systems have wide range of usage area. In implementation of embedded systems, SOC designs take common place. Programmable logic devices, such as FPGA's allow flexible SOC design implementation. In order to address current design trends, many FPGA manufacturers work on placing hard-core SOC systems into FPGAs. Currently, it is possible to implement a soft-core SOC design on a FPGA. There are many commercial or open source soft-core processors available to use on FPGA to implement SOC design.

In this study, SelCPU2 soft-core processor is designed and implemented on a FPGA. In design of SelCPU2 processor, based on the instruction set of Nios II processor, minimal area usage, maximum instruction execution performance and maximum operating frequency are aimed. SelCPU2 is a general purpose Reduced Instruction Set Computer (RISC) processor core and features Harvard memory architecture. It has separate data and instruction buses. Memory access is made using load/store instructions. Register file contains thirtytwo 32-bit general purpose register. A single instruction can address two source registers and a destination register. ALU instructions perform on registers and/or immediate value that instruction may contain, and store result to a target register. SelCPU2 features three stages pipeline processing which spans two clock cycles and executes the most of the instructions in one clock cycle. SelCPU2 features single instruction hardware multiplication and division operations, optional data cache and optional directly attached data memory. SelCPU2 processor supports Avalon-MM interface and it can be used as a custom processor component in a SOC designed with Altera SOPC Builder tool. SelCPU2 features static branch prediction with just one clock cycle wrong prediction penalty. An assembler and a back-end for LCC C compiler is

developed for SelCPU2 processor, so that it is possible to develop a program using ANSI C to run on SOC design.

The performance evaluation of the processor design is made in comparison with an equivalent SOC design using Nios II processor. In the Dhrystone benchmark which is made using the same assembly source generated by GCC C Compiler with third level optimization, SelCPU2 scores 47.43 DMIPS and Nios II/f processor scores 33.48 DMIPS. As a result SelCPU2 processor shows 41% more performance than NIOS II processor. On Altera Cyclone III FPGA, SelCPU2 has maximum operating frequency of 85.22 MHz, while Nios II has 150.20 MHz Area usage of SelCPU2 is 1,771 LEs while Nios II uses 2,397 LEs. As a result SelCPU2 uses 26% less area than Nios II. SelCPU2 has an estimated thermal power dissipation of 208 mW which is 4.3% less Nios II's. SelCPU2 processor that is designed in this study achieves the aims of the design and performed higher than the Altera Nios II processor which is a commercial product.

Key words: soft-core processor, programmable logic design, processor design

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Gömülü sistemler, belirli bir amaca yönelik uygulamayı, donanım ve yazılım bileşenleri ile birlikte gerçekleştirirler. Gömülü sistemlerde işlemci olarak, ASSP (Application Spesific Standart Product - Uygulamaya Özel Standart Ürün), standart işlemciler veya programlanabilir lojik cihazları içinde bulunabilen hard-core (silikona gömülü) işlemciler kullanılabilmektedir. Ancak sabit donanıma sahip fiziksel işlemciler kullanmak tasarımda esnek davranmanızı sınırlayabilmektedir. Ayrıca bu tür işlemciler, ticari bir ürün olduklarından sınırlı bir üretim ve destek ömürleri vardır. Buna alternatif olarak programlanabilen lojik cihazlar içinde programlama ile gerçeklenebilen ve soft-core olarak ifade edilen işlemciler tasarımda esneklik sağlamakta ve üretimden kalkma sorunu yaşamamaktadır. Programlanabilir lojik devreler üzerinde gerçeklenebilen soft-core işlemciler, hard-core olarak tanımlanan sabit fiziksel yapılı işlemcilere göre çok daha düşük performans göstermektedir. Ancak bu eksikliklerini tasarımda sağladıkları esneklik, uygulamaya göre özelleştirilebilmeleri, çoklu çekirdek/işlemci (multi-core/multi-processor) olarak kolaylıkla kullanılabilmeleri gibi avantajları ile kapatabilmektedir.

Soft-core işlemciler ile ilgili ticari, akademik ve açık kaynaklı olarak birçok çalışma bulunmaktadır. En yaygın soft-core işlemciler Nios II (Altera), MicroBlaze (Xilinx), LEON (European Space Agency, Aeroflex Gaisler), Cortex-M1 (ARM), OpenSPARC T1 (Oracle), OpenRISC 1200 (Opencores.org) sayılabilir.

Günümüzde gelişmiş yarı iletken üretebilecek bir tesisi kurmanın ortalama maliyeti 6 - 10 Milyar Dolar'dır [1]. Yeni bir yarı iletken entegre üretimi yapmanın maliyeti ise 40 Milyon Dolar'dır [2]. Üretilen ürünle bu maliyeti karşılamak ve sürdürülebilir bir iş modeli için, elde edilen gelirin %20'si Ar-Ge için harcanmalı ve 100 Milyon Dolarlık bir brüt kâr elde edilmelidir. Toptan satış kar payının %50 olduğu göz önüne alındığında firma bu ürünle yaklaşık 200 Milyon Dolarlık bir Pazar sahibi olmalıdır. Tüketici elektroniği, mobil telefonlar ve bilgisayarlardan başka böyle bir pazar kapasitesi olan çok az sayıda uygulama alanı olması, tek amaçlı veya sabit fonksiyonlu yarı iletken ürüne yatırım yapmayı çok zorlu kılmaktadır. Bu tür ürünlerin, yarı iletken üretim teknolojileri ilerledikçe artan maliyetleriyle ekonomik olarak yapılabilirliği kalmamaktadır. Programlanabilir lojik cihazlar ise burada bir alternatif sunmakta ve geniş bir pazara sahip olmayan yatırımların gerçekleşmesini ekonomik olarak mümkün kılmaktadır. 21. Yüzyılın başında FPGA gibi programlanabilir lojik cihazlar, ASIC'lere göre daha eski bir silikon teknolojisi ile üretilirken, nispeten pahalı bir çözüm alternatifi olmuştur. Günümüzde ise, FPGA'ler genel olarak ASIC'lerden daha ileri üretim teknolojileri ile üretilmektedirler. Bunun sonucu olarak fiyat/performans ve enerji tüketiminde daha avantajlı duruma geçmiştir. Önde gelen FPGA üreticileri SOC (System On A Chip - Çip Üzerinde Sistem) olarak adlandırılan bu tek çip sistemleri desteklemek için FPGA'ler içine hard-core SOC yapıları yerleştirmektedirler [3].

Literatürde ticari ve açık kaynaklı soft-core işlemcilerin karşılaştırmaları ve mevcut işlemcilerle ikili kod uyumlu gerçeklemeleri üzerine çalışmalar yapılmıştır.

Yasuura vd. [4], soft-core işlemci ile gömülü sistem tasarımı ve yazılım-donanım ortak tasarım için Valen-C olarak adlandırdıkları değişken uzunluklu veriler üzerinde çalışan C derleyicisi geliştirmiştir. Valen-C donanım ara yüzü ile işlemci mimarisinden bağımsız program geliştirme imkânı sağlamışlardır. Öne sürdükleri tasarım süreci ile tasarlanan gömülü sistem ve işlemci üzerinde parametrik değişiklikler ile elde edilen sistemin tasarım gereksinimlerine uygun hale getirilmesini sağlamışlardır.

Langen vd. [5], Motorola'nın M-Core M200 mimarisi ile ikili kod uyumlu S-Core soft-core işlemcisini tasarlamıştır. Tasarım önce FPGA üzerinde gerçeklenmiş ve

doğrulaması yapılmış, sonrasında ASIC ile gerçeklenmiştir. S-Core işlemcisinin FPGA ve ASIC gerçeklemeleri ve donanımsal çarpma ve çarpma-toplama birimlerinin, maksimum frekans, alan, güç ve uygulama performansına etkileri değerlendirmiştir.

Plavec [6], yüksek lisans tez çalışması olarak UT NIOS soft-core işlemcisini Altera NIOS işlemcisi ile ikili kod uyumlu olarak geliştirmiştir. Tasarlanan sistem mimari olarak NIOS işlemcisinden farklı olduğu ancak karşılaştırmalı performans değerlendirmesinde benzer sonuçlar elde edildiği belirtilmiştir. Ayrıca işlemci mimarisinde yaptıkları değişiklilerin performansa etkileri değerlendirilmiştir. Performansa en çok yazmaç dosyası boyutunun ve pencere kullanımı olduğunu tespit etmişler ve pencere kullanmayan aynı sayıda yazmaç içeren sistemin daha yüksek performans verdiğini belirtmişlerdir.

Yiannacouras [7], yüksek lisans tez çalışması olarak, FPGA üzerinde farklı mimari özelliklerde soft-core işlemci tasarımı oluşturacak bir araç geliştirmişler ve elde ettikleri işlemciler ile tasarım yaklaşımlarını değerlendirmiştir. Çalışma sonucunda çarpma birimleri ile gerçeklenen kaydırma ve döndürme birimlerinin daha yüksek performans verdiği ve daha az alan kullandığını tespit etmiştir. Ardışık düzen işlemenin (pipeline) daha fazla yer harcadığı ancak her zaman daha fazla performans getirmediğini gözlemlemiştir. Ayrıca aynı sayıda ardışık düzen işleme (pipeline) aşamalı işlemciler arasında da aşamaların yerinin de performansı etkilediğini tespit etmiştir.

Joost [8], endüstride FPGA tabanlı çoklu işlemcili sistemlerin sağladığı tasarım esnekliği, ölçeklenebilirlik ve bakım kolaylığını örnek bir uygulama ile göstermiştir. Örnek uygulamada, Nios II soft-core işlemcisi ile çoklu soft-core tabanlı bir sistemin nasıl tasarlanabileceği gösterilmiştir.

Tong vd. [9], mevcut soft-core işlemcilerinden Nios II/f, MicroBlaze, Xtensa XL, OpenRISC 1200 LEON3'ün performans ve özelliklerini karşılaştırmalı olarak değerlendirmiştir. Bu çalışmada, pratikte kullanılmakta olan soft-core işlemciler ile gerçeklenmiş gömülü sistemlerin iletişim, reklamcılık ve güvenlik alanlarındaki bazı uygulamalarından örnekler verilmiştir. Sonuç olarak, soft-core işlemcilerin gömülü

sistem tasarımlarında yaygınlaşacağı öngörülmüş ve tasarlanmak istenen gömülü sistemin gereksinimlerine göre soft-core işlemci seçimi yapılabileceği belirtilmiştir.

Sheldon vd. [10], soft-core işlemcileri, tasarlanmak istenen uygulamanın en iyi performansta çalışması için gerekli parametrik seçenekleri belirlemek için istatistiksel bir yöntemi olan deneysel tasarımı uygulamıştır. Bu yöntemle giriş parametrelerinin performansa etkileri tespit edilerek, istenen uygulama için en uygun parametreler belirlenmiş ve tek faktörlü yöntemlere göre 3 ila 6 kat daha fazla performans artışı elde etmiştir.

Yu [11], FPGA tabanlı soft-core işlemcilerin performanslarını arttırmak için kullanılabilecek vektör işlemci tasarımını göstermiştir. Geliştirdiği farklı yapılardaki soft-core vektör işlemcilerinin performanslarını, Nios II işlemcisi ile karşılaştırmalı olarak değerlendirmiştir. Sonuç olarak Nios II işlemcisine göre 10,9 kat daha fazla alan kaplayan soft-core vektör işlemci tasarımı ile 16,6 kat daha hızlı sonuç elde etmiştir.

2008'de "CPU Turkey" yarışması kapsamında tasarlamış olduğum SelCPU soft-core işlemcisi, "Sanal İşlemci Tasarımı" dalında birincilik ödülü kazanmıştır [12]. SelCPU işlemcisi RISC mimarisini andıran ancak özgün bir mimariye sahiptir. SelCPU Verilog ile ifade edilmiştir ve ilk olarak Xilinx Spartan 3E FPGA üzerinde gerçeklenmiş ve daha sonra Altera FPGA platformuna aktarılmıştır.

1.2 Tezin Amacı

Bu tez çalışmasının amacı FPGA üzerinde, RISC tabanlı genel amaçlı gömülü sistemler için kullanılabilecek bir soft-core işlemciyi, düşük kaynak ve alan kullanımı, yüksek çalışma frekansı ve yüksek komut çalıştırma performansı hedefleyerek tasarlamak ve gerçeklemektir.

1.3 Bulgular

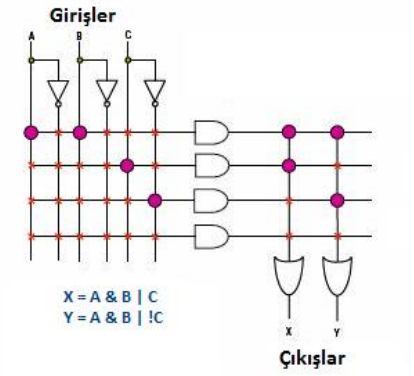
Tez çalışmasında tasarım hedeflerine ulaşabilmek için çeşitli mimari alternatifleri değerlendirilmiştir. Değerlendirmeler sonucunda belirlenen mimari yapı ile SelCPU2

işlemcisinin tasarımı ve gerçeklemesi yapılmıştır. SelCPU2 işlemcisi için C derleyicisi (C Cross-Compiler) geliştirilmiş ve komut seti benzer yapıda olan Nios II işlemcisi ile karşılaştırmalı olarak değerlendirilmiştir. Elde edilen sonuçlarda SelCPU2 işlemcisi, aynı çalışma frekansında Nios II işlemcisine göre %41 oranında daha yüksek performans göstermiştir.

BÖLÜM 2

PROGRAMLANABİLİR LOJİK

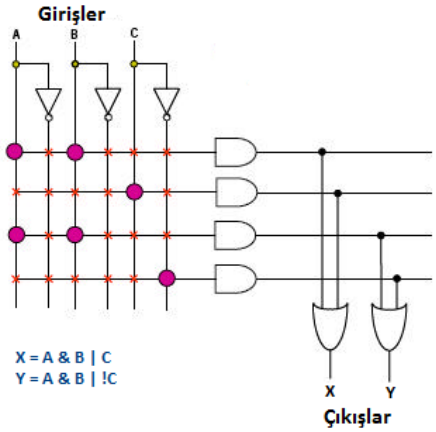
1970'lerin sonlarında standart lojik birimlerinin kullanımı oldukça yaygınlaşmıştı. Dijital sistemler çok sayıda standart lojik birimden oluşan elektronik kartlar ile geliştirilmekteydi. Lojik tasarıma esneklik sağlamak için Signetics firması tarafından, PLA(Programmable Logic Array) yapısı geliştirilmiştir. PLA dikey ve yatay hatlardan ve AND - OR kapılarına bağlanan bir mimariye sahiptir (Şekil 2.1). Dikey ve yatay hatların kesişim noktalarında bulunan sigortalar, yazılım araçları ile "patlatılarak", istenilen fonksiyon için programlanabilmektedir. PLA mimarisi oldukça esnek bir yapıdadır. Ancak zamanın üretim teknolojisi nedeniyle, giriş-çıkış gecikme süresi (yayılma gecikmesi) büyük olduğundan yeterince hızlı değildi [13].



Şekil 2.1 PLA mimari yapısı

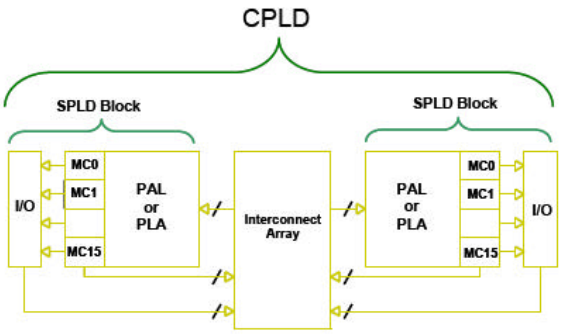
MMI firması başka bir yaklaşımda bulunarak ve PLA mimarisinden farklı olarak OR dizesini sabitlemiştir ve PAL (Programmable Array Logic – Programlanabilir Dize Lojik) mimarisini geliştirmiştir (Şekil 2.2). PAL mimarisi daha sade ve daha hızlıdır, ancak

PLA'e kadar esnek değildir. PAL ve PLA tipindeki programlanabilir cihazlar genel olarak SPLD (Simple Programmable Logic Device – Basit Programlanabilir Lojik Cihaz) olarak adlandırılırlar. SPLD'ler ile lojik tasarımın gerçeklemesi, özel bir programlama cihazı ile istenmeyen bağlantıların açık devre yapılması ile sağlanır. SPLD'ler, standart lojik entegrelerinin 50 katından daha fazla sayıda kapı içermektedir ve daha tutarlı bir yapı sunmaktadır[13].



Şekil 2.2 PAL mimari yapısı

CPLD (Complex Programmable Logic Device-Kompleks Programlanabilir Lojik Cihaz), mimari yapısı, bir çok macrocell olarak adlandırılan SPLD'ler ve bunların arasında genel amaçlı ara bağlantılardan oluşur (Şekil 2.3). Basit lojik fonksiyonlar tek bir macrocell içinde gerçekleştirilirken, daha karmaşık fonksiyonlar birden çok macrocell'in genel amaçlı ara bağlantılar ile birleştirilerek elde edilebilmektedir. CPLD'ler yüksek hızda çalışabilmektedir. CPLD, tasarım kolaylığı, geliştirme maliyetlerinin düşük olması gibi bir çok avantaj sağlamaktadır. CPLD cihazlarının bir özelliği de yapılandırmasını kendi üzerinde saklayabilmesidir.

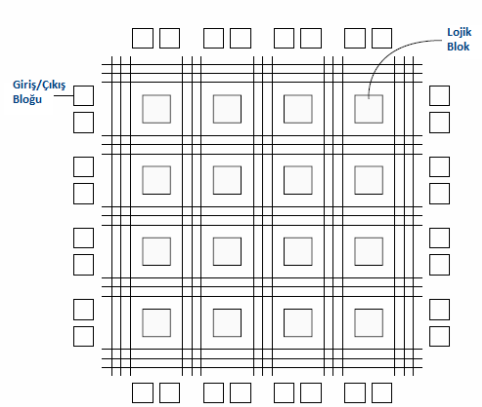


Şekil 2.3 CPLD mimari yapısı

FPGA (Field Programmable Gate Array – Sahada Programlanabilen Kapı Dizesi), 1985 yılında Xilin'in kurucusu Ross Freeman tarafından icat edilmiştir. FPGA mimari yapısı, tamamı kullanıcı tarafından belirlenebilen lojik bloklar ve ara bağlantılarından oluşmuştur.

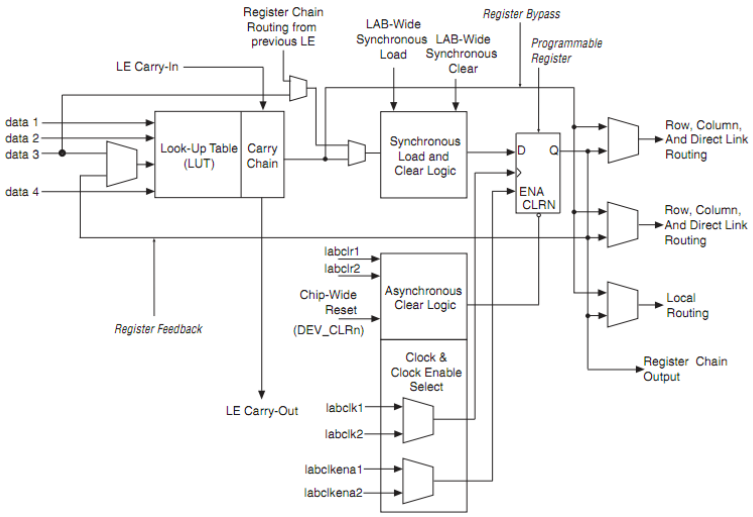
FPGA'ler, temel olarak CLB (Configurable Logic Block – Yapılandırılabilir Lojik Blok) veya LE (Logic Element – Lojik Eleman) olarak adlandırılan lojik bloklar ve Giriş/Çıkış blokları içermektedir (Şekil 2.4). Lojik bloklar hem senkron hem de kombinasyonel lojik devreleri gerçeklemek için temel lojik kaynakları içerir. Giriş/Çıkış blokları ise çipin bacaklarına bağlı birimlerdir ve çeşitli lojik voltaj veya sinyal standartlarına göre programlanabilirler. Lojik bloklar içerisinde kombinasyonel lojik fonksiyonlar, lojik kapılar yerine LUT (Lookup Table- Arama Tablosu) ile gerçekleştirilir. Lojik bloklar ayrıca, çoklayıcı, flip-flop gibi yapılarda içerirler. Altera Cyclone III FPGA'de kullanılan LE yapısı Şekil 2.5'te, ara bağlantı yapısı Şekil 2.6'da gösterilmiştir.

FPGA'ler, lojik ve giriş/çıkış blok bileşenlerinin yanında DSP modülleri (Çarpma, Çarpma ve Toplama), RAM veya ROM Hafıza blokları, fiziksel CPU gibi kaynaklara da sahip olabilirler. Sistem tasarımında bu kaynaklardan yararlanılarak verimli ve hızlı çözümler geliştirilebilir.



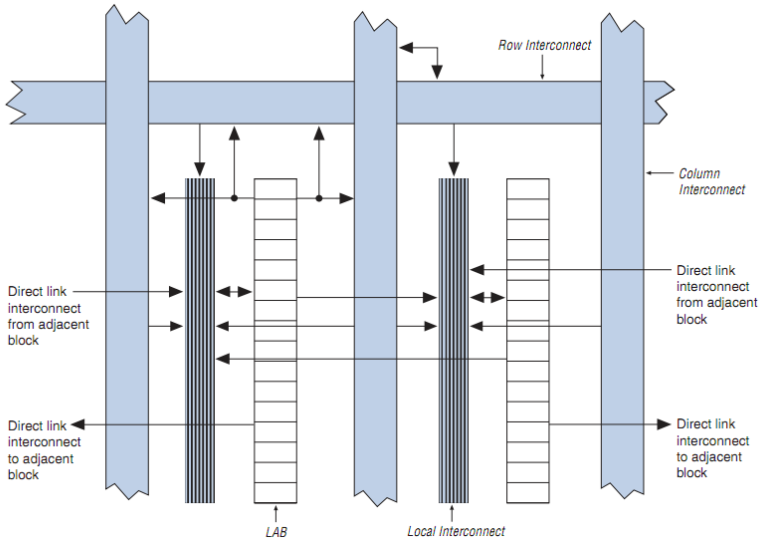
Şekil 2.4 FPGA mimari yapısı

FPGA'ler PLL (Phase Lock Loop) veya DLL (Delay Lock Loop) gibi birimler içerebilirler. Bu birimler ile saat sinyalleri sentezlenebilmektedir. FPGA'ler saat sinyallerinin çipin tümüne bozulmaya uğramadan zamanlama kısıtlarına göre ulaşması için özel saat sinyal yolları gibi mekanizmalara da sahip olabilmektedirler.



Şekil 2.5 Altera Cyclone III FPGA LE yapısı [14]

FPGA'ler genellikle benzerleri ASIC'lere (Application Specific Integrated Circuit) göre daha yavaştırlar. Bunların yanında FPGA'lerin pazara giriş zamanı düşük (daha kısa geliştirme zamanı), çıkan hataları saha da düzeltebilme ve daha düşük mühendislik maliyeti gibi avantajları vardır.

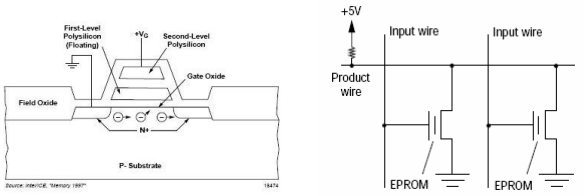


Şekil 2. 6 Altera Cyclone III FPGA ara bağlantı yapısı [14]

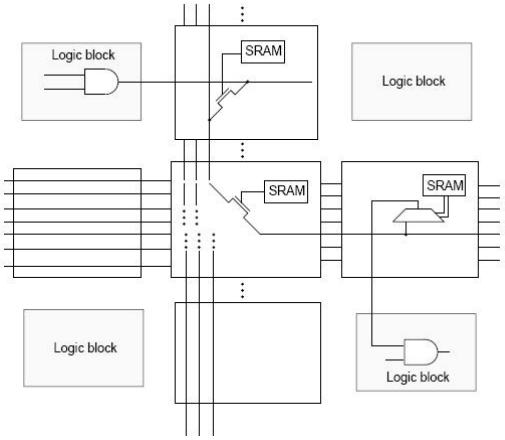
CPLD'ler ve FPGA'ler oldukça fazla sayıda programlanabilir lojik eleman içermektedirler. CPLD'in lojik kapı sayısı birkaç binlerden on binler düzeyine kadar olabilmektedir. FPGA'ler de ise bu sayı on binlerden milyonlarca ya kadar çıkabilmektedir. CPLD'ler ile FPGA'ler arasındaki başlıca fark mimari ile ilgilidir. CPLD'ler daha sınırlı bağlantı ve kaynak içeren bir yapıya sahip olduklarından küçük çaplı

tasarımlara imkan vermektedir. Bunun yanında CPLD'ler öngörülebilir zamanlama gecikmesi ve daha yüksek hızlı bağlantı avantajına sahiptir. FPGA mimarisi ise, bağlantı konusunda baskındır. Bu durum FPGA'leri daha esnek ve daha karmaşık tasarımlara hitap eden bir hale getirmektedir. CPLD'ler ile FPGA'ler arasındaki dikkate değer bir başka fark, birçok FPGA üst düzey gömülü fonksiyonlara (toplama, çarpma) ve gömülü hafızalara sahiptir.

CPLD'lerde temel anahtarlama teknolojisi olarak EPROM ve EEPROM'larda da kullanılana benzer "floating gate transistor" kullanılmaktadır (Şekil 2.7). CPLD'ler yapılandırmalarını kendi üzerlerine saklayabilmektedir. FPGA'lerde yapılandırma yaygın olarak SRAM (Static RAM) ile sağlanır ve istenildiği kadar tekrar programlanabilmektedir (Şekil 2.8). SRAM tabanlı FPGA'ler zaten özel bir hafıza çipi gibidir ve sisteme her güç verildiğinde yeniden programlanmalıdır. Bu nedenle SRAM FPGA'leri programlamak için PROM veya benzeri bir harici hafıza birimine ihtiyaç duyulmaktadır.



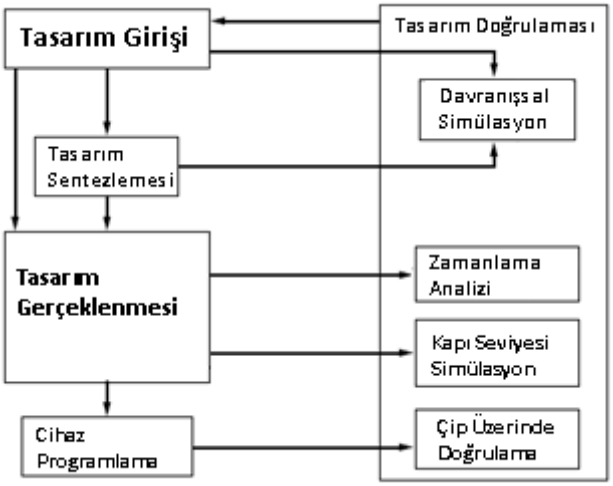
Şekil 2.7 EPROM programlanabilen anahtarlar



Şekil 2.8 SRAM tabanlı programlanabilen anahtar

FPGA üzerinde gerçeklenecek dijital sistemler, FPGA'in yapısı ve sağladığı kaynaklar gözetilerek seçilen tasarım kalıpları kullanılarak ifade edilmelidir. Örneğin, FPGA yapısı gereği bağlantı gecikmeleri nedeniyle "barrel shifter" tasarımı geniş bir çoklayıcı yerine

varsa donanımsal çarpma birimleri ile gerçeklemek daha etkin bir çözüm sunmaktadır. Benzer şekilde geniş bir yazmaç dosyasını flip-floplar üzerinde gerçeklemek yerine hafıza blokları ile gerçeklemek hem alan hem de hız konusunda avantaj sağlayabilir. FPGA üzerinde dijital sistem tasarımı süreci çeşitli aşamalardan oluşmaktadır (Şekil 2.9).



Şekil 2.9 FPGA tasarım süreci

Tasarım Girişi: FPGA Platformunda tasarım süreci bir lojik tasarımın girişi ile başlar. Bu işlem HDL ve/veya Şematik olarak yapılabilir.

Davranışsal Simülasyon: Tasarım girişi yapıldıktan sonra sistemin davranışsal simülasyonu yapılır ve tasarımın doğrulaması yapılır. Eğer düzeltme gerekirse tasarım girişinde değişikliler yapılır.

Tasarımın Sentezlenmesi: Tasarım girişinde HDL (Hardware Description Language-Donanım Tanımlama Dili) kullanılması durumunda gerçekleme yapılabilmesi için HDL'den donanım sentezlemesi yapılır. Tasarım girişi şematik olarak yapılmış ise, şematik tasarım ile ifade edilen sistem hedeflenen cihazın kaynaklarına göre donanım sentezlemesi yapılır. Sentezleme sonucunda tasarlanan sistemi oluşturan lojik blokları ve bunların bağlantılarını içeren bir netlist oluşturulur.

Tasarımın Gerçeklenmesi: FPGA platformunda gerçeklemesi, sentezleme sonucunda elde edilen netlist, giriş/çıkış bacak konfigürasyonu, zamanlama kısıtları ve diğer tasarım kısıtları kullanılarak sırayla, Translate (Çeviri), Map (Eşleştirme) ve PAR (Place and Route – Yerleştirme ve Yönlendirme) olarak üç aşamada yapılır. Translate

aşamasında gerçeklemeye verilen netlist ve tasarım kısıtlamaları birleşitirilir. "Map" aşamasında tasarım, hedeflenen cihaz üzerinde bulunan kaynaklarla eşleştirilir. "PAR" aşamasında tasarım zamanlama kısıtlarına göre tasarımın, hedef cihaz üzerindeki yerleşimi ve bağlantı yolları belirlenir.

Zamanlama Analizi: Tasarımdaki lojik ve yol gecikmeleri inceleme imkânı sağlar. Tasarımın gecikmelerinden kaynaklanabilecek potansiyel hatalarının belirlenmesinde yardımcı olur. Sistemin maksimum çalışma frekansı konusunda bilgi verir.

Kapı Seviyesi Simülasyonu: Tasarım gerçeklemesi işlemi sonucunda belirlenen hedef cihaz yapılandırmasının, standart lojik ve yol gecikmeleri ile beraber ifade edilmesinden oluşan HDL üzerinden yapılır. Böylece tasarımın hedef cihaz üzerindeki yol ve lojik gecikmeleri ile beraber simülasyonu ve doğrulaması yapılır.

Cihaz Programlama: Gerçeklenen tasarım programlama formatına dönüştürülerek hedef cihazın yapılandırmanın saklandığı PROM'a veya FPGA’e yüklenir.

Çip Üzerinde Doğrulama: Hedef cihaza yapılandırma yüklendikten sonra, yazılım araçları kullanarak çalışma zamanında yazılım veya donanım tabanlı lojik analizörler ile tasarım doğrulaması yapılabilir.

BÖLÜM 3

DONANIM TANIMLAMA DİLLERİ

Donanım tanımlama dili, donanımın yapı ve davranışlarının metin olarak ifade edilebilmesini sağlamaktadır. İlk donanım tanımlama dilleri 1977 yılında, Carnegie Mellon Üniversitesinde geliştirilen ISP (Instruction Set Processor) ve Kaiserslautern Üniversitesinde geliştirilen KARL'dır. ISP daha çok programlama diline benziyordu ve girişler ile çıkışlar arasındaki bağlantıları tanımlıyordu. Bu nedenle tasarımların simülasyonunda kullanılmış ancak sentezinde kullanılmamıştır. KARL VLSI çip yerleşim planı ve yapısal donanım tasarım desteği sağlamıştır. KARL'ı temel alan ABL, 1980'lerin başında, ABLED grafik VLSI tasarım editöründe kullanılmak üzere, telekomünikasyon araştırma merkezi CSELT (Torino, Italya) tarafından geliştirilmiştir. 1983'te Data-I/O ABEL dilini geliştirmiştir. ABEL, programlanabilir lojik cihazları için sonlu durum makinelerini tasarlamak için kullanılmıştır. İlk modern HDL olan Verilog, 1985 yılında Gateway Design Automation tarafından geliştirilmiştir. Cadence Design Systems tarafından lisanslanan Verilog-XL HDL simülatörü de-facto standardı olmuştur. 1987'de Amerika Birleşik Devletleri Savunma Bakanlığı talebi ile VHDL (Very High Speed Integrated Circuit HDL) geliştirilmiştir. Başlangıçta, Verilog ve VHDL şematik devre tasarımlarının dokümantasyonu ve simülasyonu için kullanılmıştır. HDL simülasyonu mühendislere şematik seviyeden daha fazla soyutlama imkânı sağladı ve tasarım kapasitesini yüzlerce transistörden binlercesine taşımıştır [15].

HDL'den lojik sentezlemenin yapılması ile HDL dijital tasarımın arka planından ön planına geçti. Sentezleme araçları HDL dosyalarını RTL (register transfer level) formatında, üretilebilir kapı/transistor seviyesinde netlist oluşturur. Sentezlenebilir HDL - RTL dosyaları hazırlamak için tecrübe ve özen gerekmektedir. Sentezlenen RTL

sonucu elde edilen sonuç, şematik tasarıma göre nerdeyse her zaman daha fazla kaynak kullanmakta ve yavaş sonuç vermektedir. Ancak yüksek hızlı, düşük güç tüketimli veya asenkron devreler için HDL sentezleme ön plana çıkmıştır. Özetle HDL lojik sentezlemesi, hem HDL'i dijital tasarımın merkezine getirmiş, hem de dijital devre tasarımda teknolojik bir devrim olmuştur. Birkaç yıl içinde VHDL ve Verilog, elektronik endüstrisinde yaygın HDL olarak kullanılmıştır. Hem Verilog hem de VHDL döngüsel program kontrol komutları içerir ancak bu komutlar sentezlemede kullanılamaz ve bu tip algoritmaları sentezlemek için FSM (Finite State Machine-Sonlu Durum Makinesi) yapılarından yararlanılır.

3.1 Verilog Donanım Tanımlama Dili

Verilog HDL, 1984-1985 yıllarında Gateway Design Automation da Philip Morby tarafından dijital devreleri, modelleme, simülasyon ve analiz amacıyla kolay, basit ve etkili bir şekilde ifade etmeyi hedefleyen bir donanım tanımlama dili olarak geliştirilmiştir. Cadence Design System, Gateway Design Automation'ı satın almıştır ve 1990 yılında Verilog'u genel kullanıma açmıştır. Verilog 1995 yılında IEEE tarafından standartlaştırılmıştır. Verilog yazım şekli ile C programlama dilini andıran ve öğrenilmesi nispeten daha kolay bir dildir. Verilog ile sistem hiyerarşik modüllerden ve bu modüllerin bir birleri ile ilişkileri ile tanımlanır. Verilog ile sistem, yapısal (Düşük seviyeli), veri akışı (Çıkışları giriş sinyallerinin dönüşümü ile) ve davranışsal (Devreden beklenen davranışın ifadesi şeklinde) olarak üç değişik şekilde ifade edilebilir. Bu şekilde daha üst seviyeli modüllerde soyutlamaya gidilebilmesini ve alt seviyeli modüllerinde de verimli olarak ifade etme imkânı sağlar. Şekil 3.1'de yönü seçilebilen 4-bitlik bir sayacın sentezlenebilir olarak Verilog ile ifadesi vardır.

```
module counter(input CLOCK,
input DIRECTION,
output [3:0] COUNT_OUT);

reg [3:0] count_int = 0;

always @(posedge CLOCK)
begin
  if (DIRECTION)
    count_int <= count_int + 1;
  else
    count_int <= count_int - 1;
end

assign COUNT_OUT = count_int;

endmodule
```

Şekil 3.1 Verilog ile 4-Bit sayaç

3.2 VHDL Donanım Tanımlama Dili

VHDL (Very High Speed Integrated Circuit Hardware Description Language) Amerika Birleşik Devletleri Savunma Bakanlığı sponsorluğunda elektronik cihazların modellenmesi ve simülasyonu için geliştirilmiştir. VHDL'in geliştirilmesindeki temel amaç modelleme, simülasyon olmuştur. Ancak sonradan araştırmacılar tarafından tasarım aşamalarının otomasyonunu sağlayabilmek için sentezlemede de kullanılmaya başlanmıştır. Başlangıçta askeri amaçlı projeler için kullanılsa da sonrasında özel sektörde askeri olmayan projelerde de kullanılmaya başlanmıştır. VHDL IEEE tarafından ilk olarak 1987 yılında standartlaştırılmıştır. VHDL, Verilog'a göre daha esnek sistem tasarımlarına imkân vermekle beraber üst seviyeli tasarıma daha uygundur. Ancak yazımı biraz daha karmaşık ve öğrenilmesi nispeten uzundur. Şekil 3.2'de yönü seçilebilen 4-bitlik bir sayacın sentezlenebilir olarak VHDL ile ifadesi vardır.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity counter is
Port ( CLOCK : in STD_LOGIC;
       DIRECTION : in STD_LOGIC;
       COUNT_OUT : out STD_LOGIC_VECTOR (3 downto 0));
end counter;

architecture Behavioral of counter is
signal count_int : std_logic_vector(3 downto 0) := "0000";
begin
  process (CLOCK)
  begin
    if CLOCK='1' and CLOCK'event then
      if DIRECTION='1' then
        count_int <= count_int + 1;
      else
        count_int <= count_int - 1;
      end if;
    end if;
  end process;
  COUNT_OUT <= count_int;
end Behavioral;
```

Şekil 3.2 VHDL ile 4-Bit sayaç

BÖLÜM 4

SOFT-CORE İŞLEMCİLER

Soft-Core işlemciler, donanım tanımlama dilleri ile ifade edilmiş programlanabilen lojik cihazlar üzerinde çalışacak şekilde sentezlenebilen işlemcilerdir. Programlanabilir lojik cihazlar ile gerçeklendiklerinden silikon olarak üretilmiş işlemcilere göre daha düşük performansa sahiptirler ancak kullanıldıkları sistemin ihtiyaçlarına göre işlemciler üzerinde parametrik değişiklikler yapmaya imkân sağlar. Ticari olarak Nios II ve MicroBlaze, açık kaynaklı olarak LEON3 ve OpenRISC 1200 önde gelen Soft-Core işlemcilerdir.

Nios II, Altera firması tarafından NIOS işlemcisinin daha az yer ve daha yüksek performans hedefli olarak geliştirilmiş sürümüdür. Nios II Soft-Core işlemcisi genel amaçlı RISC (Reduced Instruction Set Computer) işlemci çekirdeğine ve Harvard hafıza mimarisine sahiptir. Nios II, 32-bit komut seti mimarisinde, 32 adet genel amaçlı 32-bit yazmaça sahiptir. 32 bit çarpma ve bölme komutları içerebilir. Nios II, Altera Stratix ailesi FPGA'ler üzerinde 150 Dhrystone MIPS performansa sahiptir. Nios II, economy, standard ve fast olarak üç çeşidi vardır. Bu çeşitler arasında pipeline aşaması sayısı, komut ve veri ön belleği, donanımsal çarpma/bölme işlemleri desteği farklılık göstermektedir. Nios II, çok sayıda parametrik özellik içermekte ve kullanıcı tarafından bu özellikler ihtiyaca göre seçilebilmektedir. Varyasyonlara ve seçimlere göre kapladığı alan ve performansı değişmektedir. Nios II, çevre birimlerine hafıza adresli olarak Avalon-MM Interface Bus ile erişir ve bu yapı Altera SOPC Builder aracı kullanılarak oluşturulabilmektedir. Nios II opsiyonel olarak MMU (Memory Management Unit-Hafıza Yönetim Birimi), MPU (Memory Protection Unit – Hafıza Koruma Birimi), FPU (Floating Point Unit) içerebilir. Ayrıca kullanıcı tanımlı komutlar eklemeye imkân

verir.[9]

MicroBlaze, Xilinx firması tarafından geliştirilmiştir. MicroBlaze, gömülü uygulamalar için optimize edilmiş soft-core 32-bit işlemcidir. Xilinx Virtex-4 FPGA üzerinde 200 Mhz'de çalışabilmektedir. Harvard RISC mimarisinde, 32 bit komut, 3 aşamalı ardışık düzen işleme (pipleline) ve 32 adet 32-bit yazmaca, iki seviyeli kesmeye sahiptir. Hafıza çip üstünde veya dışında olabilir. Çip üstündeki hafızaya LMB (Local Memory Bus) ile tek saat vuruşunda erişebilir. Çevre birimleri ve çip dışındaki hafıza birimlerine OPB (On-Chip Peripheral Bus) ile erişir. MicroBlaze parametrik olarak bir çok seçeneğe sahiptir. FPU, donanımsal bölme/çarpma desteği, veri ve komut önbelleği gibi özellikleri opsiyonel olarak içermektedir [9].

OpenRISC 1200, OpenCores.org tarafından geliştirilmiştir. OpenRISC 1200 Soft-Core işlemcisi 32-bit ve 64-bit RISC mimarisi ile ağ, telekom, ev eğlence sistemleri, tüketici elektroniği, otomobil uygulamaları gibi çok geniş kullanım alanı vardır. Bu işlemci yüksek performans, düşük güç tüketimi ve geniş uygulama alanı için optimize edilmiştir. Harvard hafıza mimarisine sahip işlemci, 8 KB veri ve komut önbelleği içermektedir. 32-bit ISA yapısı OpenRISC Temel Komut Setini (ORBIS32) içerir. İşlemci 5 aşamalı ardışık düzen işleme (pipeline) yapısında ve 250MHz'de 250 DMIPS performansa sahiptir. İşlemci 8 adet ek işlem birimi, örneğin FPU, içerebilmekte ve bu birimlere özel komut veya yazmaçlar aracılıyla erişebilmektedir [9].

LEON işlemcisi Avrupa Uzay Ajansı tarafından geliştirilmiştir. LEON2 ve sonra gelen işlemciler Gaisler Research tarafından geliştirilmiştir. LEON2 ve LEON3 soft-core işlemcileri açık kaynaklı olarak VHDL ile modellenmiştir. 32-bit IEEE-1754 SPARC V8 mimarisine uyumludur. İşlemci SPARC V8 uyumlu donanımsal tamsayı çarpma, bölme ve çarpma-toplama birimleri içermektedir. LEON2 5 aşamalı, LEON3 7 aşamalı ardışık düzen işleme (pipeline) yapısına sahiptir. Her ikisi de Harvard hafıza mimarisindedir ve parametrik küme ilişkili (set-associative) önbellek birimi içerir. Yazmaç sayısı 2-32 arasında parametrik olarak belirlenebilmektedir. LEON2 ve LEON3 FPU desteği içermektedir [9]. Çizelge 4.1'de yukarıda belirtilen soft-core işlemcilerin karşılaştırmalı özellikleri verilmiştir.

Çizelge 4.1 Soft-Core işlemcilerin karşılaştırması [9]

Category	Nios II (Fast Core)	MicroBlaze	OpenRISC 1200	LEON3
Maximum MHz	200 (FPGA)	200 (FPGA)	300 (ASIC)	400/125 (ASIC/FPGA)
ASIC/FPGA Technology	– /Stratix and Stratix II	– /Virtex-4	0.18 μm/ –	0.13 μm/Not given
Reported DMIPS	150 DMIPs	166 DMIPs	250 DMIPS	85 DMIPs
ISA	32-bit RISC	32-Bit RISC	32-bit RISC	32 or 64-bit RISC
Cache Memory (I/D)	Up to 64 KB	Up to 64 KB	Up to 64 KB	Up to 256 KB
Floating Point Unit (optional)	IEEE-754	IEEE-754	As peripheral	IEEE-754
Pipeline	6 Stages	3 Stages	5 Stages	7 Stages
Custom Instructions	Up to 256 Instructions	None	Unspecified limit	None
Register File Size	32	32	32	2 to 32
Implementation	FPGA	FPGA	FPGA, ASIC	FPGA, ASIC
Area	700-1800 LEs	1269 LUTs	N/A	N/A

SelCPU Soft-Core işlemcisi, CPU-Turkey 2008 yarışmasında sanal işlemci tasarımı dalında birincilik almıştır [12]. SelCPU işlemcisi 32 bit RISC mimarisindedir. Veri ve adres yolları 32-bit genişliğindedir. Von Neuman hafıza mimarisine sahiptir. Giriş/Çıkış birimlerine erişim hafıza adresli olarak yapılır. RISC mimarisinde bulunmayan, push/pop komutları ve değişken uzunluklu komutlara (32-64bit) sahiptir ve RISC mimarisindeki işlemcilere göre az sayıda (7 Adet) genel amaçlı yazmaca sahiptir. Donanımsal çarpma desteği vardır. Donanımsal bölme işlemini desteklemez. Aynı anda 3 yazmaca (ikisi kaynak, biri hedef olarak) adresleme yapılabilir. Hafıza ile İşlemci saatleri arasında 180 derece faz fark vardır, bu özellikten yararlanarak bir komutu çalıştırırken bir sonraki komutu ön-okumasını yapabilmektedir ve koşullu dallanmalarda koşul kontrolü bir sonraki komutun hafızada adreslenmesinden önce yapılabilmekte ve koşullu dallanmalarda bekleme oluşmamaktadır. SelCPU işlemcisi Xilinx Spartan 3E FPGA üzerinde maksimum çalışma frekansı 25Mhz olarak gerçeklenebilmektedir. Bu tez çalışmasında geliştirilen tamamen yeni bir tasarıma sahip olan SelCPU2 işlemcisi adını SelCPU işlemcisinden almıştır.

BÖLÜM 5

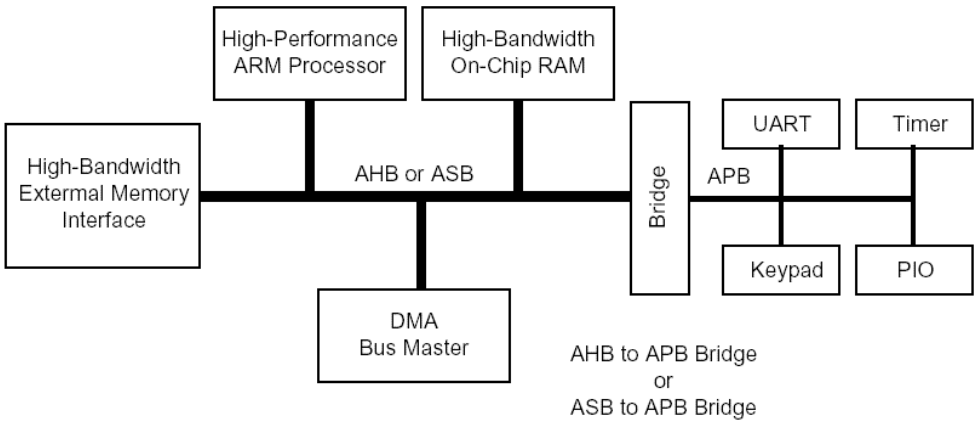
SOC & SOC SİSTEM VERİ YOLLARI

SOC (System On A Chip), bir bilgisayar sisteminin tek bir entegre üzerinde oluşturulmasıdır. SOC yapıları genellikle gömülü sistemlerde genel sistem maliyetini düşürmek ve daha az yer kaplayan bir çözüm için tercih edilmektedir. Tanım olarak benzer olsa da mikro denetleyicilere göre daha gelişmiş ve çok daha yüksek kapasitelidirler. SOC sistemler, pratik uygulamalarında hafıza, flaş bellek gibi çip dışında çevre birimleri de içerebilirler. SOC sistemlerde, çip üstünde ve dışındaki birimler arasındaki veri iletişimini, çip üstündeki veri yolu yapıları ile gerçekleştirir. Yaygın olarak kullanılan SOC sistem veri yolları aşağıda belirtilmiştir.

AMBA (The Advanced Microcontroller Bus Architecture), ARM firması tarafından mikro denetleyiciler için çip üzerinde yüksek performanslı iletişim için tanımladığı veri yolu protokollerini içerir. AMBA AHB (Advanced High-performance Bus) veri yolu protokolü, yüksek performanslı, yüksek frekanslı sistemlere destek verir (Çizelge 5.2). Veri yolu genişliği 32-64-128-256 bit olabilmektedir. Adres yolu genişliği 32 bittir. Veri yolu transferi protokolü, çoklu ana (multi master)/ çoklu köle (multi slave), tekil okuma/yazma, 4-8-16 uzunlukta toplu transfer, ardışık düzen (pipeline) ve 1 bayt, yarım ve tam uzunlukta veri transferlerini destekler. Veri yolları çoklayıcılar ile gerçeklenir. Tri-state desteği yoktur, okuma ve yazma için ayrı veriyolları vardır [16], [17].

AMBA APB (Advanced Peripheral Bus) ise düşük güç tüketimi, basit ara yüzü ile çevre birimi fonksiyonlarını destekler (Çizelge 5.2). Veri yolu genişliği 8-16-32 bit olabilmektedir. Adres yolu genişliği 32 bittir. Veri yolu transferi protokolü, tek ana (multi master)/çoklu köle (multi slave), iki saat vuruşunda tekil okuma/yazma veri

transferlerini destekler [16], [17]. Şekil 5.1'de AMBA AHB ve APB veri yolu ile SOC örneği verilmiştir.



Şekil 5.1 AMBA AHB ve APB veri yolu ile SOC örneği [16]

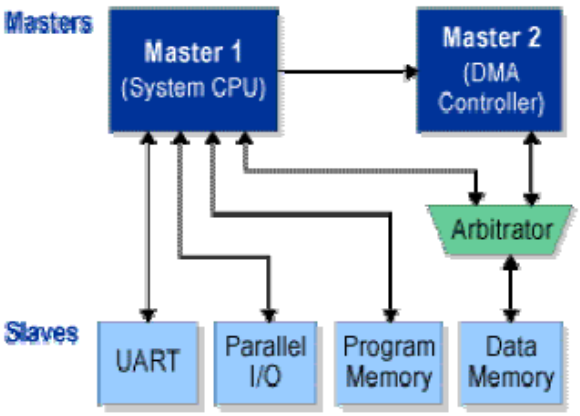
Çizelge 5.1 AMBA AHB veri yolu özellikleri [16]

busname	AMBA AHB (new generation bus)
data bus width	32- 64- 128- 256-bit
address bus width	32 bit
architecture	(Multi) MASTER / (Multi) SLAVE
	arbitration logic interface well defined Single cycle bus master handover possible
data bus protocol	Single READ/WRITE transfer
	burst transfer (4 – 8 – 16 beats)
	Pipelined
	split transactions supported
	Byte/half-word/word transfer support
data ordering	No dynamic endianess
timing	Synchronous, well defined timing specs
interconnection	multiplexed implementation
supported interconnections	Non-tristate
	Separate data read & write bus required
technology	Technology independent

Çizelge 5.2 AMBA APB veri yolu özellikleri [16]

busname	AMBA APB
data bus width	8-16-32-bit
address bus width	32 bit
tagging	No tagging
architecture	(Single) MASTER (bridge) / (Multi) SLAVE
	No arbitration logic needed
data bus protocol	2 cycle single READ/WRITE transfer
	No burst transfer
	Non-Pipelined
timing	Synchronous, well defined timing specs
interconnection	Not defined
supported interconnections	Non-tristate-bus recommended
	Separate data read & write bus recommended
technology	Technology independent
power	Zero power when not in use

Avalon-MM veri yolu, Altera firması tarafından geliştirilmiştir. Altera SOPC Builder aracı ile Avalon-MM veri yolu otomatik olarak oluşturulabilir. Avalon-MM, veri yolları için çip-seçim sinyalleri, çoklayıcıları, bekleme durumu yöneticisi, kesme öncelik atamaları, dinamik veri yolu boyutlandırması, çoklu ana çözümleme mantığını ve gelişmiş veri transfer anahtarlamasını içerir (Çizelge 5.3). Gelişmiş veri transferleri, akan veri transferi (streaming transfer), gecikmeli okuma ve yazmayı içerir. Veri yolu genişliği 8, 16, 32, 64, 128, 256, 512, 1024 bit uzunlukta olabilir. Adres yolu genişliği 32 bittir. Veri yolu transferi protokolü çoklu ana ve çoklu köle yapısını, toplu, gecikmeli ve ardışık düzen (pipeline) ve tek bayt, yarım ve tam veri transferlerini destekler [16], [18]. Avalon-MM veri yolu ile SOC örneği Şekil 5.2'de gösterilmiştir.



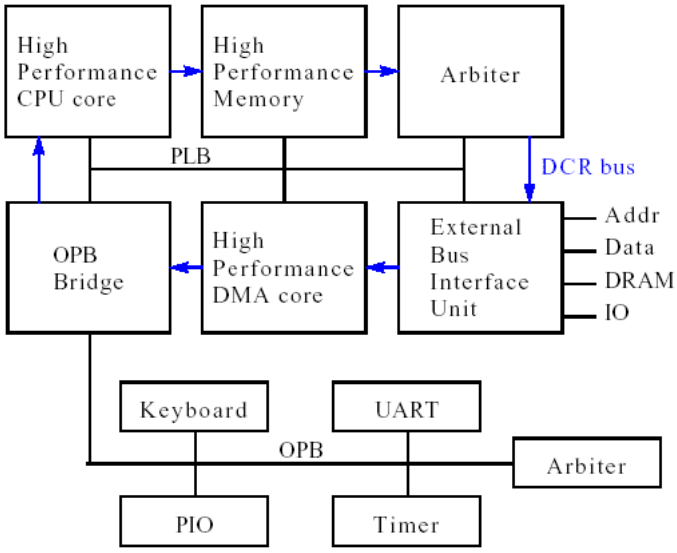
Şekil 5.2 Avalon-MM veri yolu ile SOC örneği [16]

Çizelge 5.3 Avalon-MM veri yolu özellikleri [16]

Busname	AVALON
data bus width	8, 16 or 32 bits
address bus width	32-bit
architecture	multi-master / multi slave
	multi-master arbitration logic
specific features	interrupt-priority assignment
	wait-state generation
	read & write transfers with latency
data bus protocol	one or more bus cycles
	streaming transfers (burst)
	single byte, half word or word transfers
	fixed- or peripheral-controlled wait states
	with or without setup time
timing	all signals synchronous with Avalon clock
	simple timing behavior
size	minimal FPGA resources
supported interconnections	separate address, data and control lines
	tri-state signals (external) only with bridge
technology	Altera Avalon can only be implemented on Altera devices using SOPC Builder

CoreConnect, IBM firması tarafından geliştirilmiş SOC sistem veri yolu mimarisidir. CoreConnect PLB veri yolu protokolü, SOC sistemleri için yüksek performanslı, düşük gecikmeli sistem birimlerini destekler (Çizelge 5.4). Veri yolu genişliği 32- 64-128-256 bit olabilir. Adres yolu genişliği 32 bittir. Veri yolu transferi protokolü, çoklu ana (en fazla 8)/ çoklu köle (multi slave), tekil okuma/yazma, tek saat vuruşunda hem okuma hem yazma, 16-64 uzunlukta toplu transfer, ardışık düzen (pipeline) ve 1 bayt, yarım ve tam uzunlukta veri transferlerini destekler. Veri yolları çoklayıcılar ile gerçeklenir. Tri-state desteği yoktur, okuma ve yazma için ayrı veri yolları vardır [16], [19].

CoreConnect OBP veri yolu protokolü, düşük hızlı çevre birimleri ve düşük güç tüketimi için optimize edilmiştir (Çizelge 5.5). Veri yolu genişliği 8-16-32 bit olabilir. Adres yolu genişliği 32 bittir. Veri yolu transferi protokolü, çoklu ana/ çoklu köle, tekil okuma/yazma, toplu transfer, dinamik veri yolu boyutlandırması, 1 bayt, yarım ve tam uzunlukta veri transferlerini destekler. Veri yolları çoklayıcılar ile gerçeklenir. Tri-state desteği yoktur, okuma ve yazma için ayrı veri yolları vardır [16], [19]. Şekil 5.3'te CoreConnect PLB ve OBP veri yolları ile SOC örneği gösterilmiştir.



Şekil 5.3 CoreConnect PLB ve OBP veri yolları ile SOC örneği [16]

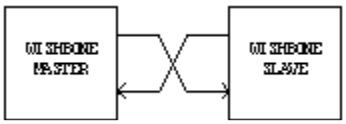
Çizelge 5.4 CoreConnect PLB veri yolu özellikleri [16]

busname	CORECONNECT PLB
data bus width	32-, 64-, 128-,256-bit
address bus width	32-bit (with address pipelining, reducing latency)
architecture	(Multi) MASTER [MAX 8]/ (Multi) SLAVE
	Arbiters with different priority schemes available as soft-core
data bus protocol	Single READ/WRITE transfer
	Overlapped READ & WRITE (2transfers/cycle)
	Burst transfer (16-64 byte bursts)
	pipelining
	Split transfer support
	Special DMA modes (flyby,…)
timing	Fully synchronous
interconnection	multiplexed implementation (=crossbar switch)
supported interconnections	Non tri-state
	Separate data read & write bus
technology	Technology independent

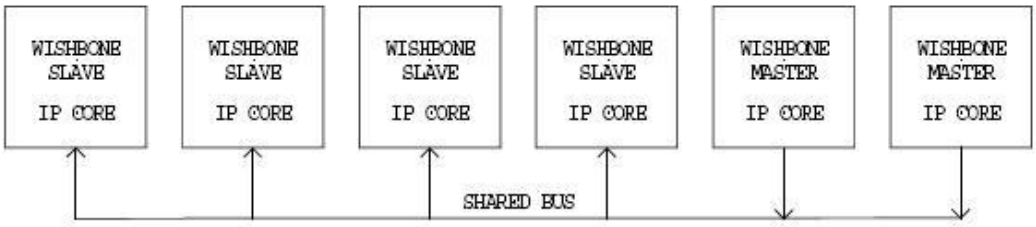
Çizelge 5.5 CoreConnect OPB veri yolu özellikleri [16]

busname	CORECONNECT OPB
data bus width	8-, 16-,32-bit
address bus width	32-bit
architecture	(Multi) MASTER / (Multi) SLAVE
	Arbiters with different priority schemes available as soft-core
	Dynamic bus sizing possible
data bus protocol	Single READ/WRITE transfer
	Burst support
	Retry support
	Single byte, half word or word transfers
	DMA support
timing	Fully synchronous
interconnection	multiplexed implementation
supported interconnections	Non tri-state
	Separate data read & write bus
technology	Technology independent
power	Bus parking support

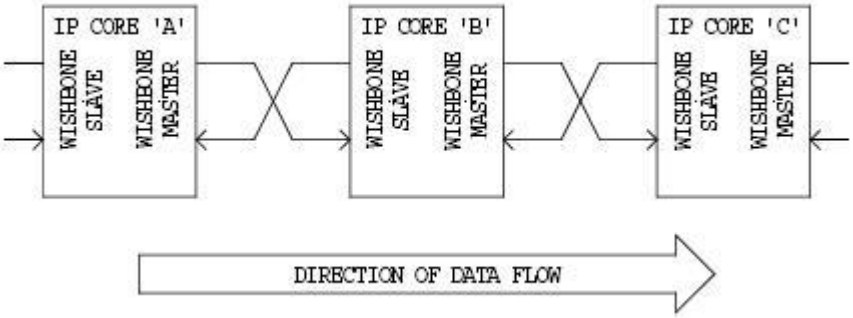
Wishbone, açık kaynaklı SOC veri yolu mimarisidir (Çizelge 5.6). Wishbone veri yolu OpenCore.org tarafından birçok projede kullanılmaktadır. Diğer veri yolu mimarilerine göre daha esnek bir tanımlamaya sahip olması nedeniyle basit ve çok geniş ara yüz konfigürasyonları sağlamaktadır. Veri yolu ve Adres yolu genişliği 8-16-32 64 bit olabilir. Veri yolu transfer protokolü, çoklu ana/çoklu köle, tekil okuma/yazma, blok transfer, tek adımda oku/değiştir/ yaz döngüsü destekler. Ara bağlantıları, noktadan noktaya (Şekil 5.4), veri akışı (Şekil 5.5), paylaşımlı veri yolu (Şekil 5.6), matris anahtarlama (Şekil 5.7) destekler [16], [20].



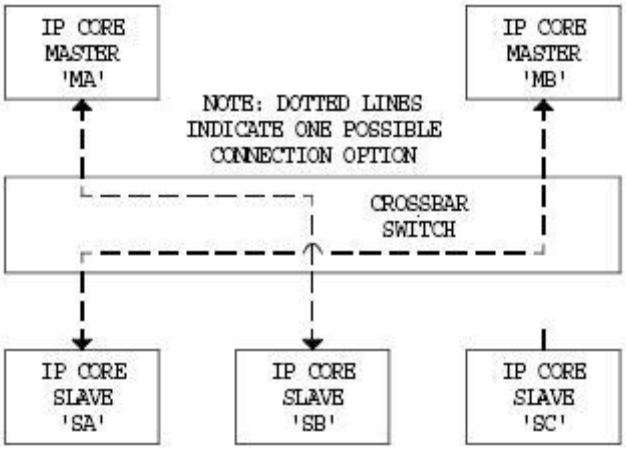
Şekil 5.4 Wishbone veri yolu noktadan noktaya bağlantı [16]



Şekil 5.5 Wishbone veri yolu paylaşımlı bağlantı [16]



Şekil 5.6 Wishbone veri yolu veri akış bağlantısı [16]



Şekil 5.7 Wishbone veri yolu matris anahtarlama bağlantısı [16]

Çizelge 5.6 Wishbone veri yolu özellikleri [16]

busname	Wishbone
data bus width	8 to 64 bits
address bus width	8 to 64 bits
tagging	address, data-in and out, cycle tags are user defined
architecture	(Multi) MASTER / (Multi) SLAVE
	arbitration logic is user defined (priority, round-robin,… arbiter)
data bus protocol	single READ / WRITE cycle
	BLOCK transfer cycle
	RMW (read-modify-write) cycle
	EVENT cycle
	Up to one data transfer per clock cycle.
data ordering	LITTLE and BIG ENDIAN support
timing	synchronous (simple design, ease of test)
	simple timing specs
size	very few logic gates (dependant on architecture)
Interconnection	point to point (a)
	Data flow (b)
	shared bus (c)
	Crossbar switch (d)
supported interconnections	unidirectional bus
	bi-directional bus
	Multiplexer based interconnections
	Tristate based interconnections
	off-chip I/O
	…
technology	Technology independent

BÖLÜM 6

SELCPU2 SOFT-CORE MİKROİŞLEMCİSİ MİMARİ YAPISI

6.1 Temel Özellikleri

SelCPU2, sentezlenebilir donanım tanımlama dili ile ifade edilen ve FPGA üzerinde gerçeklenebilen bir soft-core işlemcidir. SelCPU2 işlemcisinin geliştirilmesinde donanım tanımlama dili olarak Verilog HDL kullanılmıştır. SelCPU2, komut seti açısından RISC mimarisindedir. Hafıza erişimi LOAD/STORE komutları üzerinden yapılır. ALU üzerindeki işlemler sadece yazmaçlar ve komut içinde geçen sabit değerler üzerinde yapılabilmektedir. SelCPU2 işlemcisi 32-bit sabit uzunluklu komutlara sahiptir. SelCPU2 veri yolu ve adres yolu genişliği 32-bit olarak tasarlanmıştır. Komut hafızası ve veri hafızası veri yolları ayrıdır ve hafıza erişimi açısından Harvard mimarisine sahiptir. Çevre birimleri hafıza adresli olarak veri yoluna bağlıdır. Genel amaçlı 32 adet 32-bit yazmaca sahiptir. Ayrıca PC (Program Counter- Program Sayacı), CONTROL (Kontrol) yazmaçları vardır. Tek bir komutta üç yazmaca kadar adresleme yapabilir. İşlemci kontrolü için donanımsal kontrol lojik tercih edilmiştir. Donanımsal çarpma ve bölme işlevlerine sahiptir. SelCPU2 üç aşamalı ardışık düzen işleme (pipeline) yapısı kullanmakta ve komutların çoğu tek saat vuruşunda tamamlayabilmektedir. Ayrıca uzun gecikmeli komutlar (ör: çarpma, bölme, hafıza erişimi vb.) için birden çok saat vuruşlu çalışmada vardır. SelCPU2 işlemcisi opsiyonel olarak veri önbelleğine sahiptir. SelCPU2 SOC sistem veri yolu olarak Avalon-MM desteklemektedir ve Altera SOPC Builder içinde işlemci olarak tanımlanarak kullanılabilmektedir.

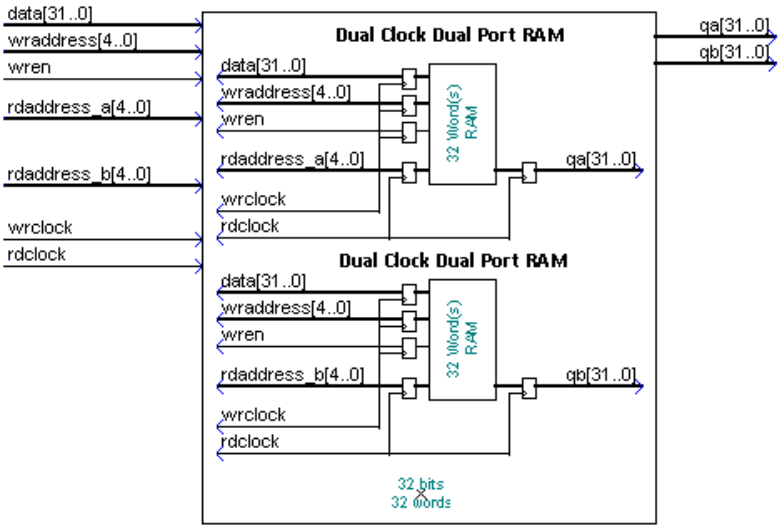
SelCPU2 işlemcisi tasarlanmasında hedef olarak belirlenen komut seti mimarisi için çalışma frekansının en yüksek olması ve kapladığı alanın (kullandığı kaynakların) en az

olması hedeflenmiştir. İşlemci tasarımında Altera Nios II işlemcisi ile performans ve özellik olarak rakip olarak alınmıştır. Tercih edilen komut seti Altera Nios II ve dolayısıyla MIPS komut seti ile benzerlik gösterir. Tasarımda, yapılan denemeler ile elde edilen alt birim gecikme ve komut çalışmasını optimize edecek şekilde ardışık düzen işleme aşama sayısı ve yerleşimi kullanan özgün bir mimari oluşturulmuştur.

6.2 Yazmaç Dosyası

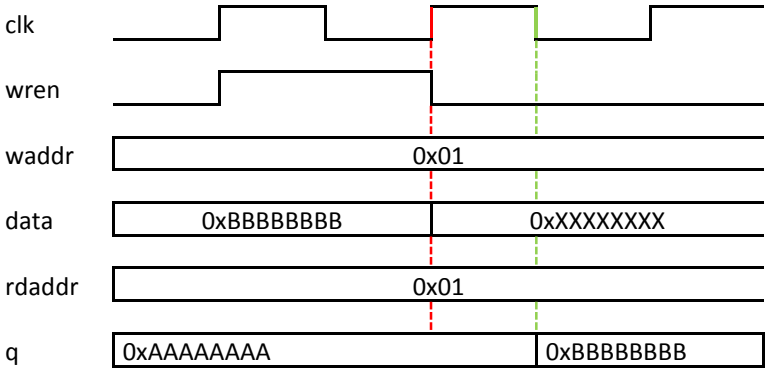
SelCPU2 32 adet 32-bit genel amaçlı yazmaca sahiptir. Yazmaç dosyası düz bir yapıdadır ve r0'dan başlayarak r31'e kadar numaralandırılır. r0 yazmacı sıfır sabit değerindedir ve değişmez. r31 yazmacı call ve callr komutlarında dönüş adresini saklar. r30 yazmacı kesme isteğinden sonra dönüş adresini saklamak için işlemci tarafından kullanılmaktadır.

Yazmaç dosyası, boyutunun büyük olması nedeniyle, yüksek hızda erişilebilmesi için FPGA üzerinde bulunan hafıza blokları ile gerçeklenmiştir. SelCPU2 işlemcisi tek komutta iki kaynak ve 1 hedef yazmaç kullanılabildiğinden, buna uygun olarak 2 okuma 1 yazma portu olacak Tri-Port RAM yapısı kullanılmıştır. Bu yapının gerçeklenmesinde 2 adet True Dual Port RAM birimi kullanılmaktadır (Şekil 6.1). Yazma her iki birime aynı adreslere yapılmakta okuma ise ayrı adreslerden yapılmaktadır.



Şekil 6.1 Yazmaç dosyası, Tri-Port RAM

FPGA üzerinde bulunan Dual Port RAM yapıları aynı adreste hem okuma hem de yazma yapıldığında önceki değeri vermektedir. Bu nedenle tasarımda okuma ve yazma saat işaretinin farklı kenarlarında yapılacak şekilde tasarlanmıştır. Böylece iki önceki komuttan oluşacak bir veri bağımlılığı önlenmiştir. Aynı adrese saat sinyalinin yükselen kenarında yazma işlemi yapılırken, düşen kenarında okuma yapıldığında ek bir veri ilerleme devresi kullanmadan yeni yazılan sonuç okunabilmektedir. Şekil 6.2'de yazmaç dosyası okuma/yazma zamanlamasına örnek verilmiştir.



Şekil 6.2 Yazmaç dosyası okuma/yazma zamanlaması

SelCPU2 ileride ek komutlar ile desteklenerek, gölge yazmaç dosyası içerebilir. Gölge yazmaç dosyası özellikle kesme servislerinde ayrı bir yazmaç dosyası seti kullanılarak daha hızlı bir cevap zamanı sağlayabilir.

6.3 Aritmetik Lojik Birimi

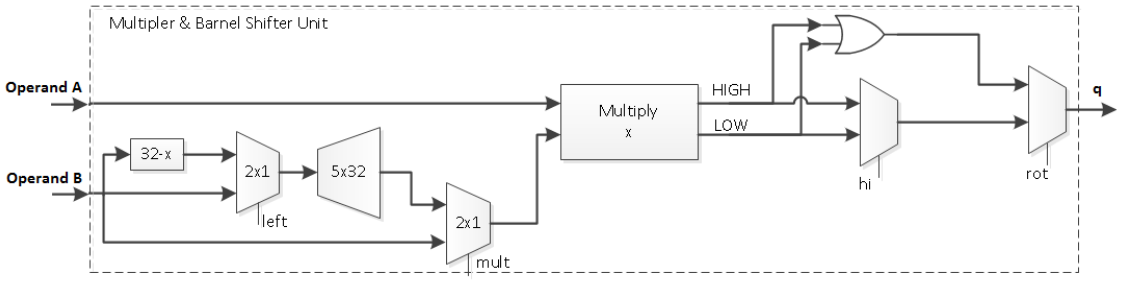
SelCPU2 ALU (Aritmetik Lojik Birimi) işlemlerini genel amaçlı yazmaçlar üzerinde gerçekleştirir. ALU operand olarak genel amaçlı yazmaçlardan bir veya ikisini kullanır ve işlem sonuçlarını yine genel amaçlı bir yazmaca saklayabilir. İkinci operand olarak komut içinde belirtilmiş sabit değerde kullanılabilmektedir. SelCPU2 Aritmetik-Lojik Birimi tarafından desteklenen işlemler Çizelge 6.1'de gösterilmiştir.

Çizelge 6.1 SelCPU2 Aritmetik lojik birimi tarafından desteklenen işlemler

Kategori	**İşlemler**
Aritmetik	Doğal sayı veya tamsayı operandlar üzerinde toplama, çıkarma, çarpma, bölme

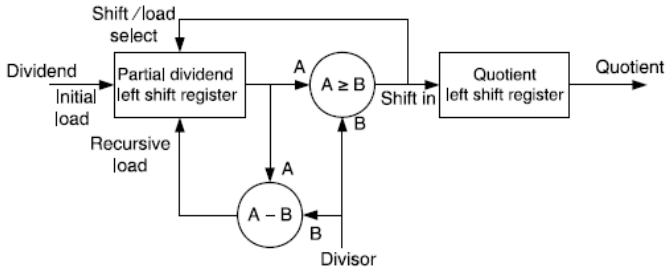
Karşılaştırma	Doğal sayı veya tamsayı operandlar arasında eşit, eşit değil, büyük eşit, küçük
Lojik	AND, OR, NOR, XOR
Kaydırma/Döndürme	Aritmetik ve Lojik sağa, sola kaydırma ve Sağa-Sola döndürme

Çarpma ve kaydırma/döndürme komutları işlemlerini FPGA üzerinde bulunan donanımsal çarpma birimleri ile gerçekleştirir. Tek bir komutta 1 ila 31 bit kaydırma/döndürme yapılabilir. Çarpma ve kaydırma/döndürme Birimi işlemlerini 2 saat döngüsünde tamamlamaktadır. Şekild 6.3'de Çarpma ve kaydırma/döndürme biriminin yapısı gösterilmiştir.



Şekil 6.3 Çarpma ve kaydırma/döndürme birimi

ALU bölme ve kalan (mod) işlemlerini çıkartma ve kaydırma ile 33 saat döngüsünde gerçekler (Şekil6.4). İlk saat vuruşunda bölünen ve bölen değerleri yüklenir. Sonraki her saat döngüsünde çıkartma ve kaydırma uygulanır. Son saat döngüsünde sonuç çıkışa verilir.



Şekil 6.4 Bölme biriminin çıkartma ve kaydırma ile gerçeklenmesi [21]

SelCPU2 işlemcisi kayan noktalı sayılar üzerinde aritmetik işlemleri donanımsal olarak desteklememektedir. Kayan noktalı işlemler, yazılım desteği ile gerçeklenebilmektedir.

6.4 Hafıza ve Giriş/Çıkış Birimi

SelCPU2 ihtiyaca göre özelleştirilebilir bir hafıza organizasyonuna sahiptir. Temel olarak Harvard mimarisinde olup ayrı komut yolu ve veri yoluna sahiptir. Komut ve veri yolu Avalon-MM ara yüzünü desteklemektedir. Opsiyonel olarak veri önbelleği ve yerel hafıza birimi içerebilmektedir.

SelCPU2 komut yolu Avalon-MM ara yüz standardına göre Avalon-MM Master Port olarak gerçeklenmiştir. Komut yolu ardışık düzen işleme (pipeline) olarak çalışabilecek şekilde tasarlanmıştır. Çalışma performansını arttırmak için komut yoluna doğrudan FPGA üzerindeki hafıza blokları bağlanarak kullanılabilir. Komut yolunun sadece tek fonksiyonu vardır, işlenecek komutu komut hafızasından okumak. Komut yolu ile yazma işlemi yapılmaz. Komut yolunda adresleme her zaman 32-bit hizalanmış bayt adresler kullanılarak yapılır ve 32-bit veri okunur. Avalon-MM ara yüzünün dinamik veri yolu boyutlandırma özelliği ile farklı genişlikteki hafıza birimleri ile çalışabilmektedir.

SelCPU2 veri yolu Avalon-MM ara yüz standardına göre Avalon-MM Master Port olarak gerçeklenmiştir. Veri yoluna hem hafıza birimleri hem de çevre birimleri bağlanabilmektedir. Çevre birimlerine hafıza adresli olarak erişilebilmektedir. Veri yolunun iki fonksiyonu vardır. İşlemci tarafından işlenen load komutları ile hafızadan okuma ve store komutları ile hafızaya yazmadır. SelCPU2 işlemcisinde opsiyonel olarak bulunabilen yerel hafıza birimi ile hem okuma hem de yazmayı tek saat vuruşunda tamamlayabilmektedir. Yerel hafıza birimi için 0x90000000 - 0x9FFFFFFF hafıza aralığı ayrılmıştır. Yerel hafıza birimine erişiminde ön bellek kullanılmaz.

SelCPU2 hafıza adreslemesi "little-endian" yapıdadır. Word (4-Byte) ve half-word (2-byte) veriler hafızada daha anlamlı baytları daha büyük adreste olacak şekilde saklanır.

Avalon-MM ara yüzü ile bir hafıza birimi hem komut yolu hem de veri yolu ile işlemciye bağlanabilir. Ancak böyle bir durumda komut erişimi ile veri erişimi arasında bekleme oluşmaması için dual-port hafıza bloğu tercih edilebilir. Eğer tek portlu hafıza birimi kullanılacaksa performans problemi yaşamamak için veri yoluna daha yüksek erişim önceliği verilmelidir.

6.5 Önbellek

Önbelleğin etkin kullanımı işlemcinin kullanılacağı uygulamanın ihtiyaç ve özelliklerine bağlıdır. Örneğin sistem çip üzerinde yerel hafızaya sahip veya hızlı çip dışında SRAM hafıza birimleri kullanılıyorsa önbellekten herhangi bir performans artışı getirmesi beklenemez. Veri hafızasında kullanılacak en büyük boyutlu kritik veriden büyük bir veri önbelleği seçilirse performans artışı izlenebilir. Yetersiz boyutlarda küçük önbellek tercih edilirse ek bir fayda getirmediği gibi bazı durumlarda daha kötü performansa neden olabilir.

SelCPU2 veri hafızası için opsiyonel olarak veri önbelleğine sahiptir. Veri önbelleği, Avalon-MM ile çip üzerinde veya dışındaki hafıza birimlerine ortalama erişim süresini düşürerek performansı artışı sağlayabilir. Önbelleğin olması programların çalışmasında fonksiyonel olarak herhangi bir değişiklik yapmaz ancak hafıza erişim performansını arttırır. Eğer sadece yerel hafıza birimi kullanılacaksa önbellek kullanımı gerekmemektedir.

Çok işlemcili bir sistemde ortak bir hafıza birimine veya hafıza adresli olarak tanımlı çevre birimlerine erişimde önbellek kullanımını atlamak gerekebilir. Önbellek kullanımı 31. adres biti ile seçilebilir. Eğer hafıza adresinin 31. Adres bitinin değeri 1 ise önbellek atlanır yoksa önbellek kullanılır. Bu özellikten dolayı adreslenebilir hafıza 2 GB ile sınırlıdır. Donanımsal sıfırlama sonrasında önbelleğide sıfırlamak için *initd* komutu kullanılabilir. Bu komut ile adreslenen önbellek satırı geçersiz kılınmaktadır. İşlemcinin ilk açılışında tüm önbellek satırları geçersiz olarak işaretlidir. Çok işlemcili bir sistemde ortak hafıza birimlerine erişimden kaynaklanabilecek önbellek-veri tutarsızlığı konusunda SelCPU2'de donanımsal bir destek yoktur, yazılımla veya ek çevre birimleri ile bir çözüm sağlanabilir.

Veri önbelleği, 4 KB, direct mapped cache (doğrudan ilişkili önbellek) olarak tasarlanmıştır. Önbellek erişiminde hafıza adres alanları Şekil 6.5'te gösterilmiştir. Yazma denetimi, write-through (hem hafızaya hem de önbelleğe beraber yazma) olarak tasarlanmıştır. Her bir önbellek satırı 19 bit etiket (tag) bilgisi, 32-bit veri ve 4 bit durum bilgisi olarak toplam 55 bit uzunluğundadır (Şekil 6.6). Durum bilgisinde ilgili baytın geçerli bir değer taşıyıp taşımadığı saklanır. Etiket (tag) bilgisinde hafıza

adresinin 30 ile 12 bit arası saklanır. Önbellek satırları için hafıza adresinin 11 ile 2 bitleri indeks olarak kullanılır. Önbellekte bulunma durumu erişilmek istenen adres, ilgili indeksteki etiket ile eşleşiyor ve veri erişimi boyutu ve erişilmek istenen adresin son iki bitti ile belirtilen bayt verilerinin tümü için durum bilgisi geçerli ise gerçekleşir.

31	30-12	11-2	1-0
	Etiket	Indeks	Ofset

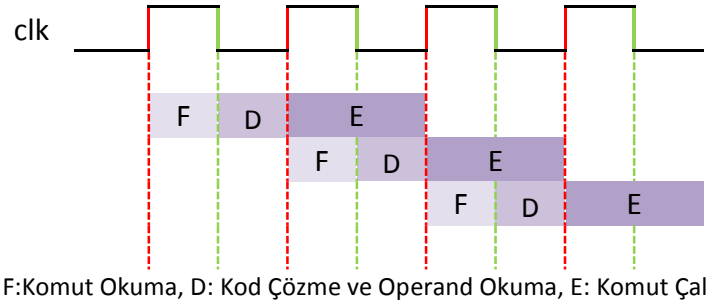
Şekil 6.5 Önbellek adres alanları

Veri	Etiket	Durum
32 bit	19 bit	4 bit

Şekil 6.6 Önbellek satır alanları

6.6 Ardışık Düzen Komut Çalıştırma Aşamaları

SelCPU2 komutları ardışık düzenli (pipeline) olarak "Komut Okuma" (Fetch), "Kod Çözme ve Operand Okuma" (Decode&Operand Fetch) ve "Komut Çalışma" (Execute) aşamaları ile işlenir. Her aşama için işlemci içinde ayrı bir birim gerçekleştirilmiştir. SelCPU2 işlemcisi tasarlanırken, seçilen komut seti mimarisi (ISA- Instruction Set Architecture) için en az bağımlılık oluşturacak ve düşük alan/kaynak kullanımı ve yüksek maksimum çalışma frekansı elde etmek amacıyla çeşitli ardışık düzenli çalışma için farklı alternatifler değerlendirilmiştir. Kullanılan FPGA platformunun özelliklerine göre iki saat döngüsünde gerçekleşen üç aşamalı ardışık düzenli işleme seçilmiştir (Şekil 6.7).



F:Komut Okuma, D: Kod Çözme ve Operand Okuma, E: Komut Çalıştırma

Şekil 6.7 Komut çalıştırma aşamaları

6.6.1 Komut Okuma Aşaması

"Komut Okuma" aşamasında çalıştırılacak komutun komut hafızasından okunması ve Avalon-MM ara yüzü ile iletişim sağlanır. Komut hafızasından okunan komut eğer diğer aşamalar bekleme durumunda değilse direk olarak "Kod Çözme ve Operand Okuma"

aşamasına aktarılır. Eğer işlemci bekleme durumunda ise okunan komut saklanır ve bekleme kalktığında “Kod Çözme ve Operand Okuma” aşamasına aktarılır. Avalon-MM ara yüzünden kaynaklanan bekleme olması durumunda komut adresi saklanır ve işlemci bekletilir.

6.6.2 Kod Çözme ve Operand Okuma Aşaması

“Kod Çözme ve Operand Okuma” aşaması bir alt birim ile gerçeklenmektedir. Çalıştırılacak komut “Komut Okuma” aşamasında hazır ise yazmaç dosyası adreslenerek yazmaç operandlar okunur ve “Komut Çalıştırma” aşamasında kullanılacak kontrol sinyalleri kombinasyonel lojik ile oluşturulur. “Kod Çözme ve Operand Okuma” aşamasında program kontrol komutları ve kesme talepleri Dallanma Kontrol birimini tarafından işlenir. Veri ilerletme birimi ile “Komut Çalışma” aşamasındaki komutun “Kod Çözme ve Operand Okuma” aşamasında kaynak olarak seçilen bir yazmacı hedef olarak göstermesi durumunda bir sonraki komutta “Komut Çalışma” aşamasına operand olarak aktarılacak değerin güncel olmasını sağlar.

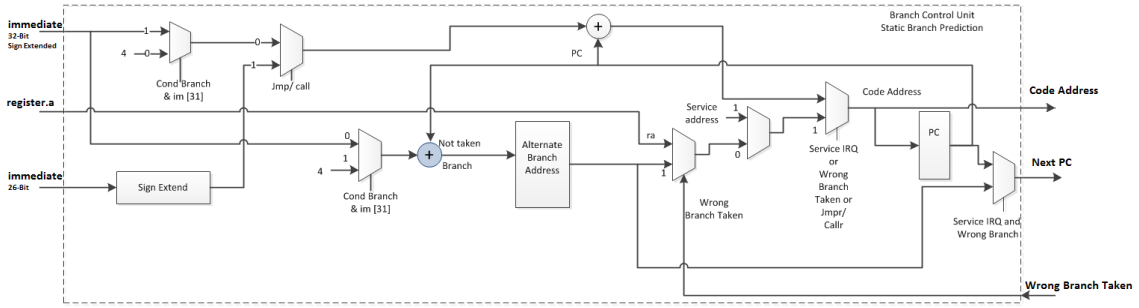
“Komut Çalışma” aşaması bekleme durumunda değilse, “Komut Çalışma” aşamasında kullanılacak kontrol sinyalleri ve operandlar yazmaçlar üzerinden “Komut Çalışma” aşamasına aktarılır. Operand okuma işlemi bu aşamada yapılarak yazmaç dosyasından okuma gecikmesinin “Komut Çalışma” aşamasını uzatması önlenerek maksimum çalışma frekansı arttırılmıştır.

6.6.3 Komut Çalışma Aşaması

“Komut Çalışma” aşaması ALU, hafıza erişimi ve koşullu dallanma hata kontrol birimleri ile gerçekleştirilir. Komutlar özelliklerine göre tek saat vuruşunda veya daha uzun olarak “Komut Çalışma” aşamasında çalıştırılır. Hafızadan okuma ve yazma işlemleri bu aşamada yapılır. “Kod Çözme ve Operand Okuma” aşamasında verilen koşullu atlama kararının sonucu bu aşamada doğrulanır. Hafızadan okuma ve ALU sonucunda elde edilen sonuçlar seçiciler ile seçilerek hedef olarak seçilen yazmaca aktarılır.

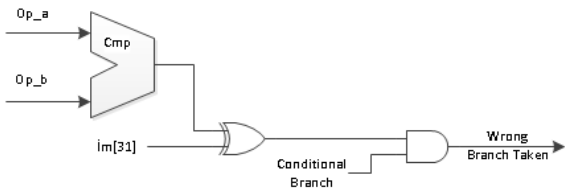
6.7 Dallanma Kontrol Birimi

Dallanma kontrol birimi ile “Kod Çözme ve Operand Okuma” aşamasında atlama, altprogram çağırma, koşullu dallanma komutları ve kesme isteklerine göre bir sonraki çalıştırılacak komut adresi hesaplanır. Koşullu dallanma komutlarında statik dallanma tahmini yapılır. Derleyiciler tarafından gerçeklenecek “for/while” döngülerinde tekrarlanacak adreslerin genelde daha küçük adreslerde, “if-else” komutlarında ise koşulun doğru olması durumunda daha büyük adrese devam edilir. Döngü komutlarında döngünün daha çok devam edeceğini, koşul komutlarında ise koşulun gerçekleşme oranı daha fazla olduğu düşünülerek, Statik dallanma tahmini eğer negatif bir adres etiketi verilmişse dallanma adresi seçilecek, pozitif bir adres etiketi verildiğinde ise bir sonraki komut adresinden devam edecek şekilde gerçeklenmiştir. Dallanma tahmini sonucunda tercih edilmeyen adres alternatif dallanma adresi yazmacı ile saklanır. Şekil 6.8’de Dallanma kontrol birimi yapısı gösterilmiştir.



Şekil 6.8 Dallanma kontrol birimi

Dallanma tahminin doğruluğu bir sonraki saat döngüsünde “Komut Çalışma” aşamasında koşulun karşılaştırma birimi testi sonucunda belirlenir (Şekil 6.9). Eğer koşullu dallanma tahmini hatalı ise “Kod Çözme ve Operand Okuma” aşamasındaki komut ihmal edilerek *nop* (işlem yok) komutu ile devam edilir ve alternatif dallanma adresi yazmacındaki adresten komut çalıştırmaya devam edilir. Hatalı dallanma tahmini sadece bir saat döngüsü gecikmeye neden olmaktadır.



Şekil 6.9 Dallanma tahmini sonuç kontrolü

6.8 Kontrol ve Kesme İşleme Birimi

Kontrol birimi işlemcinin genel çalışma durumunu ve ardışık düzen çalışma aşamalarının uyumlu olarak çalışmasını sağlar.

Kesme kontrol birimi, kesme isteği ile beraber o kesme isteğine ait kesme servis adresini de işlemciye göndermektedir. Eğer gelen bir kesme isteği var ve kesmelere izin verilmiş ise, istek "Kod Çözme ve Operand Okuma" aşamasında dallanma kontrol birimine iletilir. "Komut Çalışma" aşamasındaki komut tamamlandıktan hemen sonra kontrol kesme adresine yönlendirilir ve o anki program sayacı r30 yazmacına saklanır. Kesme servisi bittikten sonra *iret* komutu ile program kaldığı yerden devam eder. Eğer koşullu dallanma komutu sonrasında kesme gelmiş ve dallanma tahmini hatalı ise alternatif dallanma adresi yazmacındaki adres r30 yazmacına yazılır.

6.9 Bekleme Durumları

SelCPU2 işlemcisi aşağıda listelenen durumlarda doğru bir sonuç elde etmek için komut çalışma aşamalarını bekletmektedir.

- Çalışma aşaması birden çok saat döngüsü gerektiren komutlarda (Ör: çarpma, bölme vb.)
- Veri hafızası erişiminde Avalon-MM beklemelerinde
- Komut hafızası erişiminde Avalon-MM beklemelerinde
- jumpr ve callr komutlarında eğer bir önceki komutla veri bağımlılığında

SelCPU2 tasarlanırken veri ilerletme kullanılarak veri bağımlılığından kaynaklanacak birçok bekleme önlenmiştir. Ancak *jumpr* ve *callr* komutlarının gerçeklenmesinde bu

yöntem kullanılırsa sistemin maksimum çalışma frekansı kötü yönde etkileyeceğinden tercih edilmemiştir.

6.10 Komut Yapısı

SelCPU2 işlemcisinin Nios II ve MIPS işlemcilerine benzer olarak üç tip komut yapısı vardır.

- **I-tipi Komut Yapısı:** I-tipi komutlar, 6-bit *opcode* alanı, 5-bit **ra** kaynak yazmacı, 5-bit *rb* hedef yazmacı ve 16-bit *im* sabit değer alanı içerir (Şekil 6.10). Bu tip komutlar *ra* yazmacı ve *im* sabit değerler ile işlem yaparak sonucu *rb* yazmacına saklar.

opcode	ra	rb	im
6	5	5	16

Şekil 6.10 I-tipi komut yapısı

- **R-tipi Komut Yapısı:** R-tipi komutlar kaynak olarak *ra* ve *rb* yazmaçları üzerinde *opex* ile belirlenen komutu gerçekleştirip sonucunu *rc* yazmacında saklayabilir (Şekil 6.11). R-tipi komutlarda opcode 63 değerindedir. Sabit değerli kaydırma işlemlerinde *im5* alanı kullanılmaktadır.

opcode	ra	rb	rc	opex	im5
6	5	5	5	6	5

Şekil 6.11 R-tipi komut yapısı

- **J-tipi Komut Yapısı:**

  J-tipi komutlar *jmp* ve *call* komutlarında bağıl atlama için kullanılır. 26 bitlik bit sabit değerle 256MB aralığında bir atlama yapabilir (Şekil 6.12).

opcode	İm
6	26

Şekil 6.12 J-tipi komut yapısı

6.11 Komut Kategorileri

6.11.1 Veri Transfer Komutları

SelCPU2 üzerinde işlem yapılacak operand tipi açısından load-store mimarisindedir. ALU sadece yazmaçlar ve komut içerisinde geçen sabit değerler üzerinde işlem

yapabilir. Hafıza üzerindeki veri üzerinde işlem yapmak için, veriler load komutları ile yazmaçlara aktarılır ve yazmaçlar üzerinde hesap yapıldıktan sonra yazmaçlarda saklanan sonuç hafızaya store komutları ile yazılır. Load ve store komutları ile veri hafızası ve çevre birimleri ile işlemcinin yazmaçları arasında veri aktarımı yapılır. *ldb* ve *ldh* komutları ile yazmaçlara işaretli sayı olarak yükleme yapılır. İşaretsiz sayı olarak yükleme, normal yükleme sonrasında *and* komutu ile sağlanabilir.

Yazmaçlar arasında veya yazmaçlara sabit değer atamak için *mov* veri transfer komutları kullanılır. Çizelge 6.2'de veri transfer komutları listelenmiştir.

Çizelge 6.2 Veri transfer komutları

Komut	Kullanım	Açıklama
ldw	ldw rb,[ra+im]	ra+im hafıza adresindeki 32-bit değeri rb yazmacına yükle.
stw	stw rb,[ra+im]	rb yazmacındeki 32-bit değeri ra+im hafıza adresine sakla.
ldh	ldh rb,[ra+im]	ra+im hafıza adresindeki 16-bit değeri rb yazmacına yükle.
Sth	sth rb,[ra+im]	rb yazmacındeki 16-bit değeri ra+im hafıza adresine sakla.
ldb	ldb rb,[ra+im]	ra+im hafıza adresindeki 8-bit değeri rb yazmacına yükle.
stb	stb rb,[ra+im]	rb yazmacındeki 8-bit değeri ra+im hafıza adresine sakla.
mov	mov rb,ra	rb = ra
movi	movi rb,im	rb = im , im:tamsayı
movui	movui rb,im	rb = im , im:doğalsayı
movhi	movhi rb,im	rb[31:16] = im
movli	movli rb,im	rb[15:0] = im

6.11.2 Aritmetik ve Lojik İşlem Komutları

SelCPU2 işlemcisi tamsayı ve doğal sayıları üzerinde temel aritmetik ve lojik işlem komutlarını içerir (Çizelge 6.3).

Çizelge 6.3 Aritmetik ve lojik işlem komutları

Komut	Kullanım	Açıklama
add	add rc,ra,rb	rc = ra + rb
addi	addi rb,ra,im	rb = ra + im, im: tamsayı
sub	sub rc,ra,rb	rc = ra – rb
subi	subi rb,ra,im	rb = ra – im, im: tamsayı
mul	mul rc,ra,rb	rc = ra * rb , Sonucun alçak 32 biti
muli	muli rb,ra,im	rb = ra * im, , Sonucun alçak 32 biti
mulxss	mulxss rc,ra,rb	rc = ra * rb , tamsayı*tamsayı sonucun yüksek 32 biti
mulxsu	mulxsu rc,ra,rb	rc = ra * rb , tamsayı*doğalsayı sonucun yüksek 32 biti
mulxuu	mulxuu rc,ra,rb	rc = ra * rb , doğalsayı*doğalsayı sonucun yüksek 32 biti
div	div rc,ra,rb	rc = ra / rb, tamsayı bölme
divu	divu rc,ra,rb	rc = ra / rb, doğal sayı bölme
mod	mod rc,ra,rb	rc = ra mod rb, tamsayı bölümden kalan
modu	modu rc,ra,rb	rc = ra mod rb,doğal sayı bölümden kalan
and	and rc,ra,rb	rc = ra and rb

andi	andi rb,ra,im	rb = ra and im, im: doğalsayı
or	or rc,ra,rb	rc = ra or rb
ori	ori rb,ra,im	rb = ra or im, im: doğalsayı
xor	xor rc,ra,rb	rc = ra xor rb
xori	xori rb,ra,im	rb = ra xor im, im: doğalsayı
nor	nor rc,ra,rb	rc = ra nor rb
nori	nori rb,ra,im	rb = ra nor im, im: doğalsayı

6.11.3 Karşılaştırma Komutları

Karşılaştırma komutları iki yazmaç arasında veya bir yazmaç ve bir sabit değer arasında karşılaştırma yapar sonucu 1 veya 0 olarak bir yazmaca kaydeder (Çizelge 6.4).

Çizelge 6.4 Karşılaştırma komutları

Komut	Kullanım	Açıklama
cmpeq	cmpeq rc,ra,rb	rc = ra == rb
cmpeqi	cmpeqi rb,ra,im	rb = ra == im
cmpne	cmpne rc,ra,rb	rc = ra != rb
cmpnei	cmpnei rb,ra,im	rb = ra != im
cmpge	cmpge rc,ra,rb	rc = ra>=rb, tamsayı
cmpgei	cmpgei rb,ra,im	rb = ra>=im, tamsayı
cmpgeu	cmpgeu rc,ra,rb	rc = ra>=rb, doğal sayı
cmpgeui	cmpgeui rb,ra,im	rb = ra>=im, doğal sayı
cmpgt	cmpgt rc,ra,rb	rc = ra>=rb, tamsayı
cmpgti	cmpgti rb,ra,im	rb = ra>im, tamsayı
cmpgtu	cmpgtu rc,ra,rb	rc = ra>rb, doğal sayı
cmpgtui	cmpgtui rb,ra,im	rb = ra>im, doğal sayı
cmple	cmple rc,ra,rb	rc = ra<= rb, tamsayı
cmplei	cmplei rb,ra,im	rb = ra<= im, tamsayı
cmpleu	cmpleu rc,ra,rb	rc = ra<= rb, doğal sayı
cmpleui	cmpleui rb,ra,im	rb = ra<= im, doğal sayı
cmplt	cmplt rc,ra,rb	rc = ra<= rb, tamsayı
cmplti	cmplti rb,ra,im	rb = ra < im, tamsayı
cmpltu	cmpltu rc,ra,rb	rc = ra< rb, doğal sayı
cmpltui	cmpltui rb,ra,im	rb = ra < im, doğal sayı

6.11.4 Kaydırma ve Döndürme Komutları

Kaydırma ve döndürme komutlarında, kayma/döndürme sayısı yazmaç veya sabit değerle ifade edilebilir (Çizelge 6.5).

Çizelge 6.5 Kaydırma ve döndürme komutları

Komut	Kullanım	Açıklama
sll	sll rc,ra,rb	rc = ra << rb
slli	slli rb,ra,im	rb = ra << im
srl	srl rc,ra,rb	rc = ra >> rb, doğal sayı
srli	srli rb,ra,im	rb = ra >> im, doğal sayı
sra	sra rc,ra,rb	rc = ra >> rb, tamsayı

srai	srai rb,ra,im	rb = ra >> im, tamsayı
rol	rol rc,ra,rb	rc = ra rotate right rb
roli	roli rb,ra,im	rb = ra rotate right im
ror	ror rc,ra,rb	rc = ra rotate right rb
rori	rori rb,ra,im	rb = ra rotate right im

6.11.5 Program Kontrol Komutları

SelCPU2 aşağıdaki koşulsuz dallanma ve alt program çağırma komutlarını destekler (Çizelge 6.6). Alt program komutları dönüş adresini r31 yazmacına kaydeder.

Çizelge 6.6 Koşulsuz dallanma ve alt program çağırma komutları

Komut	Kullanım	Açıklama
call	call label, call im	r31 = pc, pc = pc + im, dönüş adresi r31' e aktarılır
jmp	jmp label, jmp im	pc = pc + im
callr	callr ra	r31 = pc, pc = ra , dönüş adresi r31' e aktarılır
jmpr	jmpr ra	pc = ra
ret	ret	pc = r31, dönüş adresi r31' e aktarılır
iret	iret	pc = r30, r30:kesmeden dönüş adresi

Koşullu dallanma komutlarında ilgili koşul doğru ise dallanma yapılır yoksa bir sonraki komuttan devam edilir (Çizelge 6.7).

Çizelge 6.7 Koşullu dallanma komutları

Komut	Kullanım	Açıklama
beq	beq ra,rb,label	ra == rb => pc = label
bne	bne ra,rb,label	rc = ra != rb => pc = label
bge	bge ra,rb,label	rc = ra>=rb, tamsayı => pc = label
bgeu	bgeu ra,rb,label	rc = ra>=rb, doğal sayı => pc = label
bgt	bgt ra,rb,label	rc = ra>=rb, tamsayı => pc = label
bgtu	bgtu ra,rb,label	rc = ra>rb, doğal sayı => pc = label
ble	ble ra,rb,label	rc = ra<= rb, tamsayı => pc = label
bleu	bleu ra,rb,label	rc = ra<= rb, doğal sayı => pc = label
blt	blt ra,rb,label	rc = ra<= rb, tamsayı => pc = label
bltu	bltu ra,rb,label	rc = ra< rb, doğal sayı => pc = label

6.11.6 Assembly Pseudo Komutları

SelCPU2 assembly dili pseudo komutları, eşdeğer işlemleri gerçek komutlar ile ifade eder (Çizelge 6.8). Pseudo komutlar gerçek komutlarla aynı şekilde kullanılır. Pseudo komutların mevcut komutlar ile çalıştırılmasından dolayı, işlemcide gerçeklenmelerine gerek kalmaz ve kaplanılan alandan kazanç sağlar.

Çizelge 6.8 Assembly pseudo komutları

Pseudo Komut	**Kullanım**	**Eşdeğer Komut**
bgt	bgt ra, rb, label	blt rb, ra, label
bgtu	bgtu ra, rb, label	bltu rb, ra, label
ble	ble ra, rb, label	bge rb, ra, label
bleu	bleu ra, rb, label	bgeu rb, ra, label
cmpgt	cmpgt rc, ra, rb	cmplt rc, rb, ra
cmpgti	cmpgti rb, ra, im	cmpgei rb, ra, (im+1)
cmpgtu	cmpgtu rc, ra, rb	cmpltu rc, rb, ra
cmpgtui	cmpgtui rb, ra, im	cmpgeui rb, ra, (im+1)
cmple	cmple rc, ra, rb	cmpge rc, rb, ra
cmplei	cmplei rb, ra, im	cmplti rb, ra, (im+1)
cmpleu	cmpleu rc, ra, rb	cmpgeu rc, rb, ra
cmpleui	cmpleui rb, ra, im	cmpltui rb, ra, (im+1)
mov	mov rc, ra	add rc, ra, r0
movi	movi rb, im	addi, rb, r0, im
movui	movui rb, im	ori rb, r0, im
nop	nop	cmpeqi r0, r0, r0

BÖLÜM 7

SELCPU2 UYGULAMA-DONANIM ARAYÜZÜ

Bu bölümde SelCPU2 işlemcisi uygulama-donanım ara yüzü (ABI-Application Binary Interface) tanımlanacaktır. Uygulama-donanım ara yüzü, C derleyicisi, veri yapıları, verilerin hafızada yerleşimini, yığın yapısını, yazmaç kullanımları, fonksiyon çağırma, kesme kullanımını ifade eder.

7.1 SelCPU2 Assembler Derleyicisi

SelCPU2 Assembly dili ile programlama desteği için SelCPU2 Assembler geliştirilmiştir. SelCPU2 Assembler ile program hafızası olarak kullanılabilecek FPGA hafıza bloklarının ilk değer ataması için gereken dosyaları oluşturmaktadır.

7.2 LCC C Derleyicisi

LCC (Little C Compiler), yeniden uyarlanabilen açık kaynaklı ANSI C çapraz-derleyicisidir. LCC derleyicisi Alpha, SPARC, MIPS, x86 gibi birçok platform için kod üretebilmektedir. LCC çapraz-derleyici olarak yeniden uyarlaması, hedef platformun makine tanımlama dosyası ile ifade edilmesi ile yapılmaktadır. LCC temel olarak önyüz ve arka yüz olarak iki bölümden oluşmaktadır. Önyüz bölümü standart C ile ifade edilmiş olan programını ayrıştırır ve her bir fonksiyon için DAG (Directed Acyclic Graph)'lar oluşturur. Derleyicinin arka yüz bölümü makine tanımlama dosyası ile ifade edilen platformun kaynak ve kısıtlarına göre, önyüzde elde edilen DAG'lara karşılık gelen assembly kodunu üretir [22].

Bu tez çalışmasında, SelCPU2 işlemcisi için MS Windows platformunda çalışan LCC çapraz derleyicisi oluşturulmuştur. LCC ile derleme sonucunda elde edilen SelCPU2

assembly kodu, SelCPU2 Assembler ile çalıştırılabilir ikili koda dönüştürülmekte ve FPGA üzerindeki kod hafıza bloğu ilk atama bilgisi olarak tanımlanmaktadır. Hafıza blokları için tanımlanan ilk atama değerleri, FPGA yapılandırma dosyasında güncellenerek, FPGA'e yüklenebilmektedir.

7.3 Veri Tipleri

SelCPU2 mimarisi için LCC ANSI C çapraz-derleyicisinde kullanılan veri tipleri, boyutları ve ifade şekilleri (Çizelge 7.1)'de belirtilmiştir.

Çizelge 7.1 Veri tipleri

Veri Tipi	Uzunluğu (Bayt)	İfade Şekli
char, signed char	1	ikinin tümleyeni (ASCII)
unsigned char	1	işaretsiz (ASCII)
short, signed short	2	ikinin tümleyeni
unsigned short	2	işaretsiz
int, signed int	4	ikinin tümleyeni
unsigned int	4	işaretsiz
long, signed long	4	ikinin tümleyeni
unsigned long	4	işaretsiz
pointer	4	işaretsiz
long long	4	ikinin tümleyeni
unsigned long long	4	işaretsiz

7.4 Hafıza Adres Hizalaması

SelCPU2 işlemcisi, program kodunun ve verinin hafıza yerleşiminde bir takım adres hizalamalarını zorunlu tutar. Fonksiyonlar ve komut adresleri 4 bayta hizalanmış olmalıdır. Hafızada veri tipleri, en az kendi uzunluklarında hizalanmış olmalıdırlar. 4 bayttan uzun veri tipleri için adres 4 bayta hizalanmış olması yeterlidir. C struct, union ve string veri tipleri için adres 4 bayta hizalanmış olmalıdır. C struct veri tipi içinde tanımlı bit alanları için adres 4 bayta hizalanmış olmalıdır.

7.5 Yazmaç Kullanımı

LCC derleyicisi, yazmaçları farklı amaçlara göre gruplandırarak kullanır. C dilinde register olarak tanımlı veya derleyicinin optimizasyon amacıyla değişkenler çağrılan tarafından saklanan genel amaçlı yazmaçlara eşlenir. Bu yazmaçlar fonksiyon içinde kullanılacaksa fonksiyon girişinde yığında saklanır ve çıkışında eski değeri yüklenir. Sabit değerler, hesaplanan ara değerler, adres hesaplaması gibi işlemler sonucuna elde

edilen geçici değerler çağıran tarafından saklanan genel amaçlı yazmaçlarda saklanır. Bu yazmaçlar kullanılmadan önce eski değerleri kullanan fonksiyon tarafından saklanmaz. Bunun yerine eğer bu değerler bir fonksiyon çağırma sonrasında kullanılacaksa, bu fonksiyonu çağırmadan önce, çağıran tarafından yığında saklanabilir. SelCPU2 LCC debug desteği için *fp* (*r28*), çerçeve işaretçisi olarak kullanılabilir. Yazmaç kullanımı Çizelge 7.2 de gösterilmiştir.

Çizelge 7.2 Yazmaç kullanımı

Yazmaç	Assembly Adı	Derleyici Tarafından Kullanılıyor	Çağrılan Tarafından Saklanan	Kullanım Şekli
r0	zero	√		0 – Sabit
r1	at			Assembler geçici değişkeni
r2		√		Fonksiyon dönüş değeri (32-bit)
r3		√		Çağıran tarafıdan saklanan genel amaçlı yazmaç
r4		√		Yazmaçla aktarılan parametre (ilk 32-bit)
r5		√		Yazmaçla aktarılan parametre (ikinci 32-bit)
r6		√		Yazmaçla aktarılan parametre (üçüncü 32-bit)
r7		√		Yazmaçla aktarılan parametre (dörtüncü 32-bit)
r8		√		Çağıran tarafıdan saklanan genel amaçlı yazmaçlar
r9		√		
r10		√		
r11		√		
r12		√		
r13		√		
r14		√		
r15		√		
r16		√	√	Çağrılan tarafından saklanan genel amaçlı yazmaçlar
r17		√	√	
r18		√	√	
r19		√	√	
r20		√	√	
r21		√	√	
r22		√	√	
r23		√	√	
r24		√	√	
r25		√	√	
r26		√	√	
r27		√	√	
r28	fp	√	√	
r29	sp	√		Yığın İşaretçisi
r30	ia			Kesme Dönüş Adresi
r31	ra	√		Fonksiyon Dönüş Adresi

7.6 Yığın Yapısı

Yığın yapısını hafızada yukarıdan aşağıya (düşük adrese) doğru ilerleyecek şekilde organize edilmiştir. SelCPU2 işlemcisi yığın yapısı için özel komutlar içermez. Yığın

yapısının gerçeklenmesinde, *sp* (*r28*) yazmacı yığın işaretçisi olarak kullanılır ve en son kullanılan bölümü gösterir. Yığın işaretçisi her zaman 4 bayta hizalanmış olmalıdır.

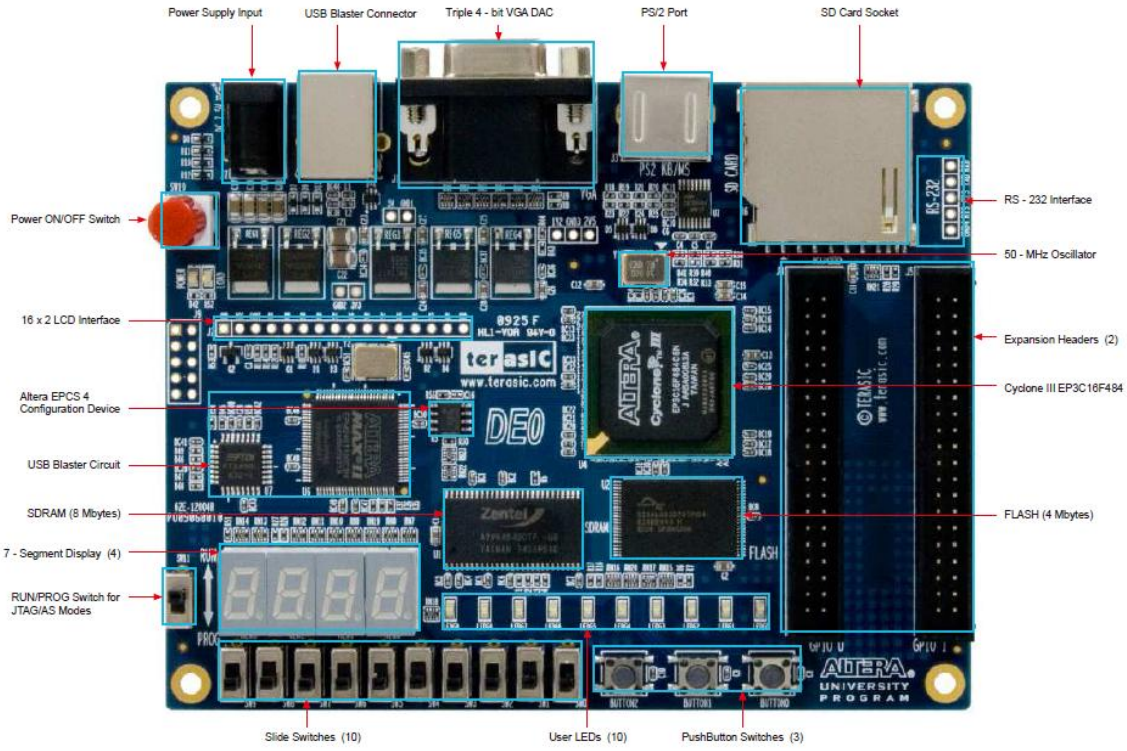
7.7 Fonksiyon Parametreleri ve Dönüş Değeri

Fonksiyon çağırılırken parametreler çağıran yığında parametreler için yer ayırır. Çağrılan fonksiyon girişte, *sp* yazmacını güncelleyerek yığında yer açar. Başka fonksiyonlar çağırıyorsa *ra* (*r31*) yazmacını yığında saklar. Fonksiyon içinde kullanılan çağrılan tarafından saklanması gereken yazmaçları yığında saklar. Fonksiyona aktarılan parametrelerin ilk 16 baytı *r4*-*r7* yazmaçları ile aktarılır, sonra gelen parametreler yığın üzerinden aktarılır. Fonksiyon dönüş değerleri 32-bit'e kadar *r2* yazmacı ile döndürülür, daha uzun veri tipleri için ilk 32-bit *r2* yazmacı ile sonrası yığın üzerinden döndürülür.

BÖLÜM 8

PERFORMANS DEĞERLENDİRMELERİ

SelCPU2 işlemcisi, Altera Nios II işlemcisi ile karşılaştırmalı olarak, işlem kapasitesi ve harcanan güç açısından performansları değerlendirilmiştir. Değerlendirmede Altera Üniversite Programı kapsamında olan Altera DE0 FPGA Eğitim ve Geliştirme Kartı kullanılmıştır (Şekil 8.1).



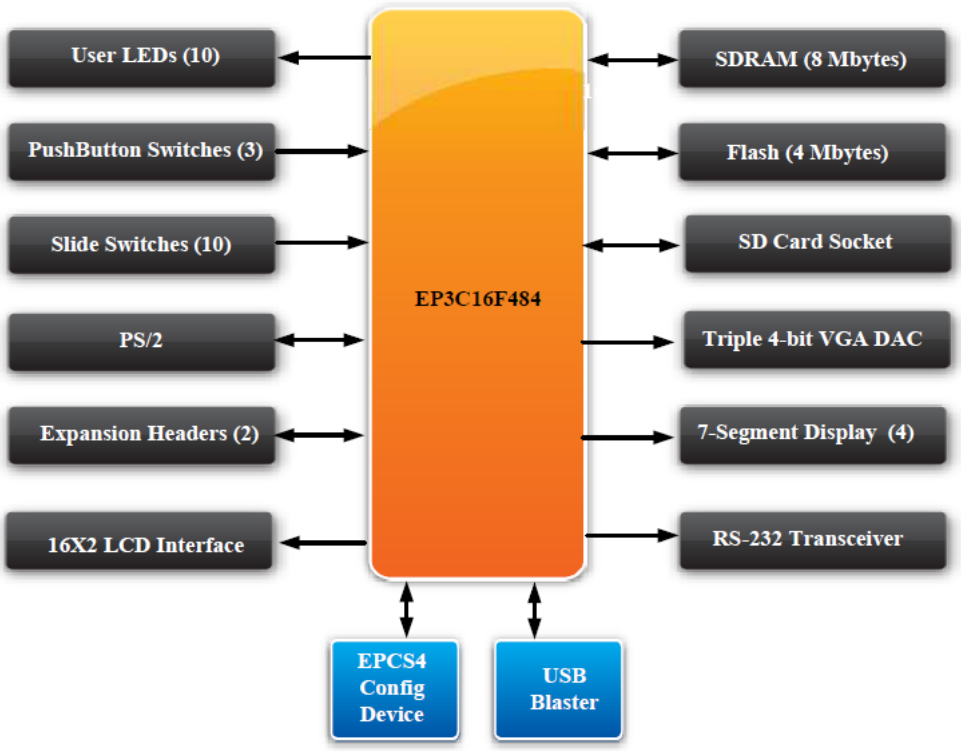
Şekil 8.1 DE0 FPGA eğitim ve geliştirme kartı[25]

DE0 kartı,

- Altera Cyclone® III 3C16F484 FPGA
- Altera Seri Konfigurasyon Cihazı – EPCS4

- USB Blaster (JTAG ve Aktif Seri Programlama için)
- 8-MB SDRAM
- 4-MB Flash Hafıza
- SD Kart Okuyucu
- 3 tuş
- 10 kayan anahtar
- 10 yeşil led
- 50-MHz osilatör, Saat Sinyali için
- VGA DAC (4-bit) , VGA Çıkış Portu
- RS-232
- PS/2 klavye/fare portu
- İki adet 40-pin genişleme portu

donanım bileşenlerini içerir (Şekil 8.2).



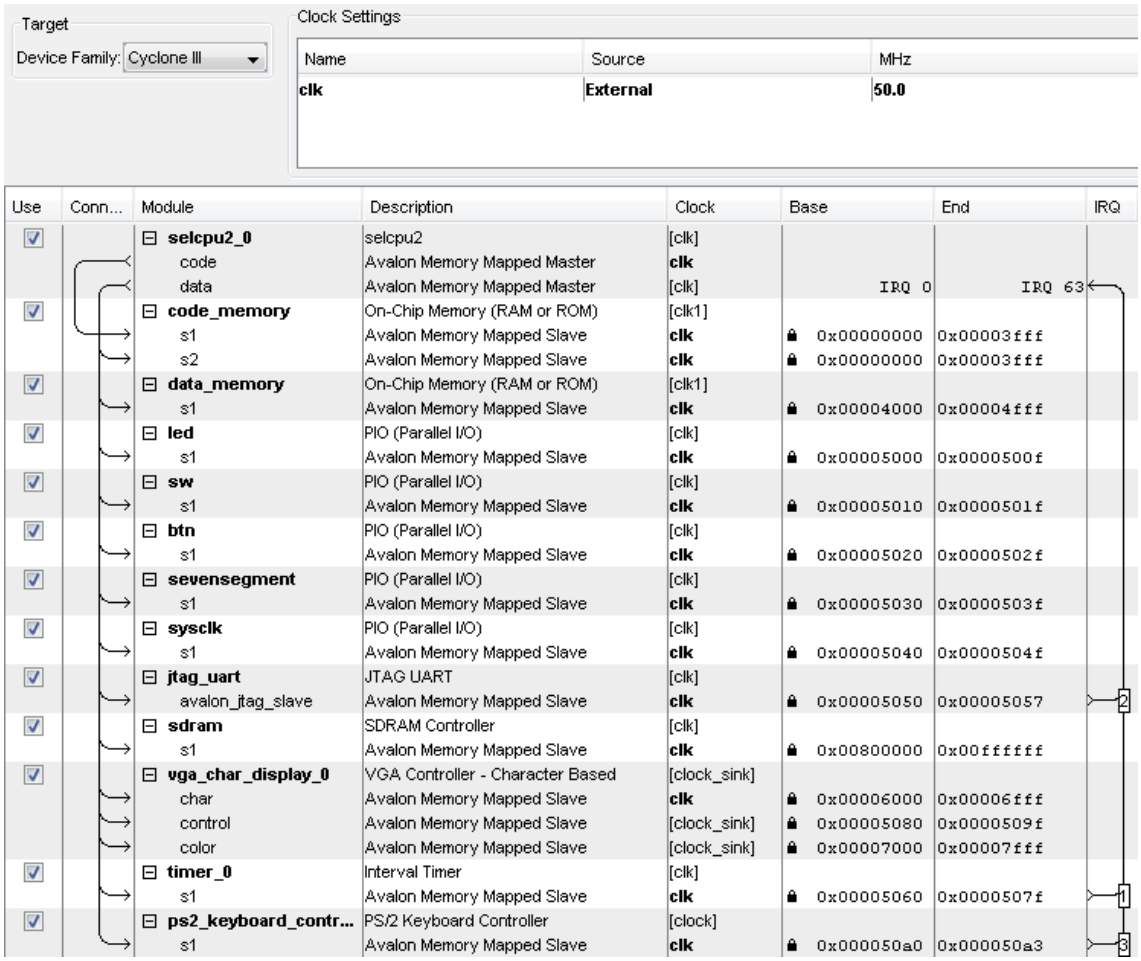
Şekil 8.2 DE0 kartında FPGA ve çevre birimi bağlantıları[25]

Cyclone IIII 3C16F484 FPGA, 15408 adet LE (Logic Element), 56 adet M9K Hafıza Bloğu (Toplam 504K bit RAM), 56 gömülü çarpma modülü, 4 PLL (Phase Lock Loop), 346 adet kullanıcı tanımlı giriş/çıkış bacağı (FineLine BGA 484-pin) içerir.

8.1 Performans Değerlendirmesi Yapılan Sistemler

Performans değerlendirmesi SelCPU2 işlemcisi ile Altera Nios II işlemcisi arasında yapılmıştır. Değerlendirilmenin yapıldığı SOC sistemleri, Altera SOPC Builder aracı ile oluşturulmuştur. Değerlendirmelerde her iki sistemde de komut ve veri hafızası olarak FPGA üzerindeki hafıza blokları kullanılmıştır. Sistem saati olarak 50 MHz osilator kullanılmıştır.

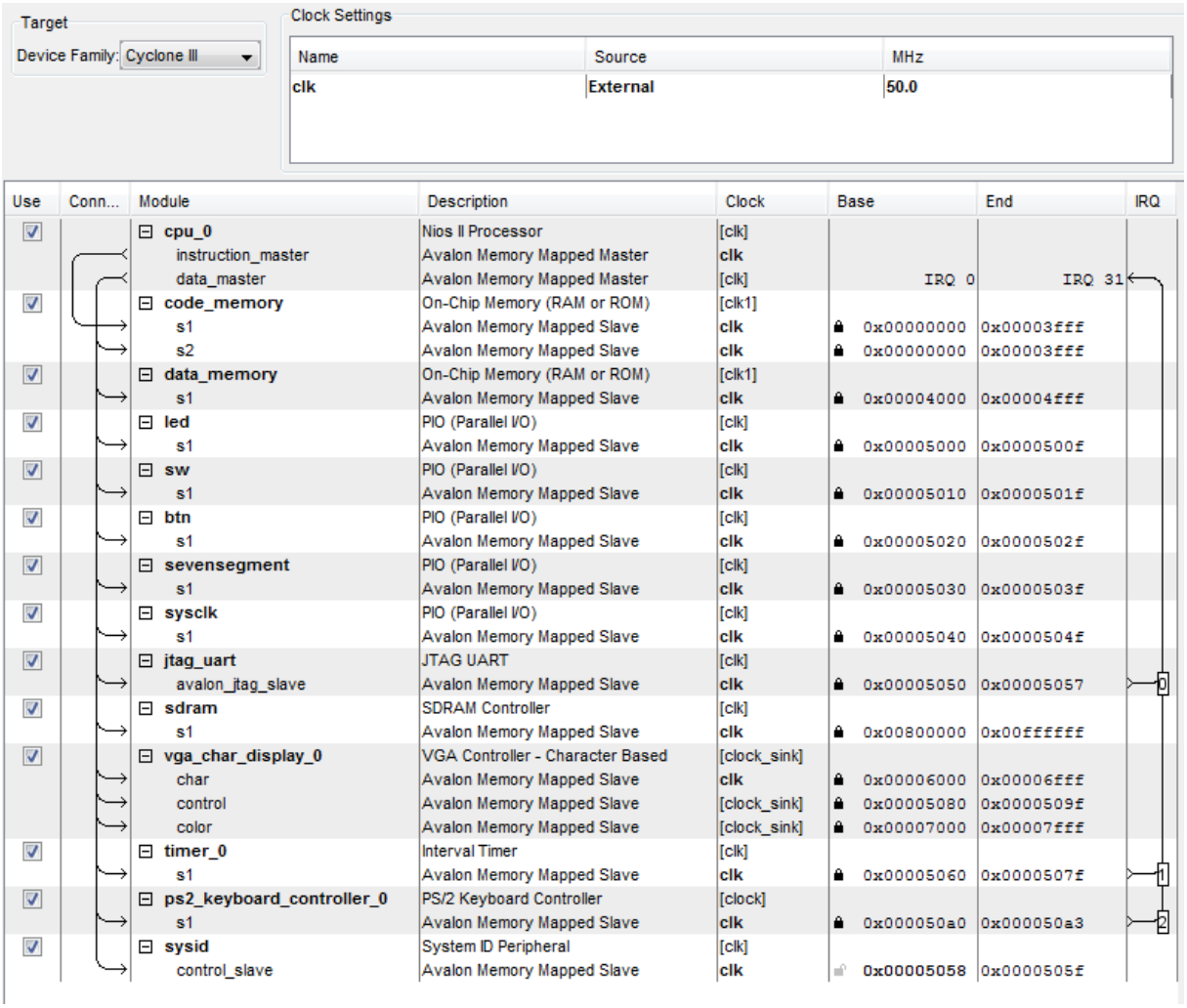
SelCPU2 sisteminde, SelCPU2 işlemcisi veri önbelleği (4 KB), donanımsal çarpma ve bölme desteği ile kullanılmıştır. Değerlendirmeye alınan SelCPU2 işlemcili SOC sistemi Şekil 8.3'te gösterilmiştir.



Şekil 8.3 Değerlendirmeye alınan SelCPU2 işlemcili SOC sistemi

Nios II Sisteminde, Nios II/f işlemci çekirdeği, 4 KB komut ve 4 KB veri önbelleği, donanımsal çarpma ve bölme desteği kullanılmıştır. Sistem performansını net olarak

değerlendirebilmek için JTAG Debug modülü kullanılmamıştır. Değerlendirmeye alınan Nios II/f işlemcili SOC sistemi Şekil 8.4'te gösterilmiştir.



Şekil 8.4 Değerlendirmeye alınan Nios II/f işlemcili SOC sistemi

8.2 Komut Seti Değerlendirmesi

SelCPU2 işlemcisi ANSI C derleyicisinin ihtiyaç duyacağı tüm temel komutları içermektedir. SelCPU2 işlemcisinin komut seti, Nios II işlemcisinin komut setinin alt kümesidir. Nios II işlemcisi opsiyonel olarak ileri önbellek yönetimi, gölge yazmaç seti gibi SelCPU2'de bulunmayan özellikler ve alt birimler için komutlar (flush, rdprs, wrprs, vb.) içerir.

8.3 Komut Çalıştırma Performansı

SelCPU2 işlemcisi 3 aşamalı ardışık düzen işleme (pipeline) ile birçok komutu tek saat vuruşunda çalıştırabilmektedir. Komut çalıştırma performansı ve Nios II işlemcisi ile karşılaştırmalı olarak Çizelge 8.1'de gösterilmiştir. SelCPU2 işlemcisi, jumpr ve callr komutlarında eğer bir önceki komutta yazmaç bağımlılığı varsa bir saat vuruşu bekleme yapar, diğer komutlarda bağımlılıktan beklemeye neden olmaz. Nios II işlemcisinde benzer şekilde bağımlılık oluşturan komutlar bir saat vuruşu beklemeye neden olur. SelCPU2 işlemcisi Nios II işlemcisine göre kaydırma/döndürme komutları dışında daha az saat vuruşunda tamamlayabilmektedir. Bölme işlemlerinde Nios II işlemcisi 4 ile 66 saat vuruşunda sonuç verirken SelCPU2 işlemcisi 33 saat vuruşunda tamamlamaktadır.

Çizelge 8.1 Komut çalıştırma performansları

	Nios II/f [23] 6 Aşamalı Ardışık Düzen		**SelCPU2 3 Aşamalı Ardışık Düzen**	
Komut Grubu	**Saat Vuruşu**	**Ceza**	**Saat Vuruşu**	**Ceza**
Normal ALU komutları (add, cmplt, vb)	1		1	
Koşullu Dallanma (Doğru tahmin, Dallanma Var)	2		1	
Koşullu Dallanma (Doğru tahmin, Dallanma Yok)	1		1	
Koşullu Dallanma (Yanlış Tahmin)	4	Pipeline Boşaltılır	2	
call, jmpi, rdprs	2		1	
ret	3		1	
jmpr, callr	3		1	Bağımlılık
rdctl	1	Bağımlılık	1	
load (Direk Bağlı Hafıza)	1	Bağımlılık	1	
load (Avalon-MM Hafıza)	>1	Bağımlılık	>1	
store (Direk Bağlı Hafıza)	1		1	
store (Avalon-MM Hafıza)	>1		>=1	
initd (Önbellek Sıfırlama)	2		1	
Çarpma Komutları(Gömülü Çarpma Birimleri ile)	5	Bağımlılık	2	
Bölme Komutları(Donanım Destekli)	4 ile 66 arası	Bağımlılık	33	
Kaydırma ve Döndürme Komutları (Gömülü Çarpma Birimleri ile)	1	Bağımlılık	2	
Diğer Tüm Komutlar	1		1	

8.4 Dhrystone Performans Testi

Dhrystone performans testi, 1984 yılında Reinhold P. Weicker tarafından sistem (tamsayı işlem) performansını değerlendirmek amacıyla geliştirilmiştir [24]. Dhrystone performans testi gelişerek genel işlemci performansını temsil etmek için kullanılmıştır.

Daha sonraları SPECint gibi performans değerlendirme araçları çıkmıştır. Dhrystone ücretsiz ve açık kaynaklı olması ve diğer araçların pahalı olması, nedeniyle popüler kalmıştır. Dhrystone 2.1 sürümü, 1988 yılında yayınlanmıştır ve Haziran 2010 itibariyle birkaç küçük değişiklik dışında aynı kalmıştır.

Dhrystone, farklı işlemci mimarileri arasında, MIPS (Million Instructions Per Second) olarak ifade edilen saniyede işlenen işlemci komut sayısından daha anlamlı bir karşılaştırma sunmaya çalışmıştır. Yüksek seviyeli işlemi, CISC mimarisinde bir işlemcisi ile tek bir komutta yapabilirken, aynı işlem ile RISC mimarisinde birçok komutta ancak daha hızlı işlenebilmektedir. Dhrystone skoru, farklı mimariler arasında bir kıyaslama yapabilmek için, test programının saniyede kaç defa tamamlandığı sayılarak belirlenir. Dhrystone performans testinin diğer bir ifade şekli olan DMIPS (Dhrystone MIPS), Dhrystone skorunun, referans olarak alınan VAX 11/780 sisteminden elde edilen 1757 Dhrystone skoruna bölünmesi ile hesaplanır. VAX 11/780 nominal 1 DMIPS sistem olarak değerlendirilir [24].

Dhrystone performans testi, parametrik olarak C derleyicisinden değişken tanımlarında değişkenlerin yazmaçlar ile tutulmasını isteyebilir. Nios II GCC C/C++ derleyicisi ve SelCPU2 için geliştirilen LCC C derleyicisi, değişkenleri otomatik olarak yazmaçlarda tutabilmektedir ve bu seçeneğin performansa olumlu veya olumsuz bir etkisi görülmemiştir. Nios II GCC C/C++ derleyicisi farklı optimizasyon seviyeleri için değerlendirilmiştir. LCC C derleyicisinde tam olarak optimize kod üretmemekte ve optimizasyon işlemi derleyici tarafından kod üretimi sırasında yapılmaktadır. Karşılaştırmanın derleyici etkisinden arınmış olarak yapılabilmesi için Nios II sisteminde üçüncü seviye optimizasyon ile derlenerek oluşturulan assembly kodu, SelCPU2 assembly diline çevrilerek de performans değerlendirmesi yapılmıştır.

Değerlendirilen sistemler için Dhrystone performans testi sonuçları Çizelge 8.2, Çizelge 8.3 ve Çizelge 8.4'de belirtilmiştir.

Çizelge 8.2 Nios II Dhrystone performans testi sonuçları

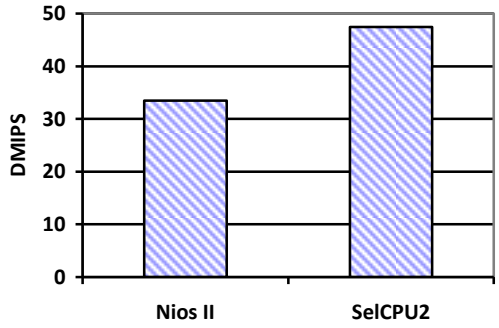
GCC C Derleyicisi Optimizasyon Seviyesi	Dhrystone Skoru	DMIPS	DMIPS/MHz
Yok	12.345	7,03	0,141
Seviye 1 (-O1)	33.333	18,97	0,379
Seviye 2 (-O2)	43.478	24,75	0,495
Seviye 3 (-O3)	58.823	33,48	0,670

Çizelge 8.3 SelCPU2 Dhrystone performans testi sonuçları - LCC

LCC C Derleyicisi Optimizasyon Seviyesi	Dhrystone Skoru	DMIPS	DMIPS/MHz
Yok	50.000	28,46	0,692

Çizelge 8.4 SelCPU2 Dhrystone performans testi sonuçları - GCC

GCC C Derleyicisi Optimizasyon Seviyesi	Dhrystone Skoru	DMIPS	DMIPS/MHz
Seviye 3 (-O3)	83.333	47,43	0,949



Şekil 8.5 DMIPS performans testi karşılaştırması

GCC C derleyicisin üçüncü optimizasyon seviyesi ile elde edilen ortak assembly kodu ile yapılan performans testinde SelCPU2 işlemcisinde 47,43 DMIPS ve Nios II işlemcisinde 33,48 DMIPS sonuç alınmıştır. Sonuç olarak SelCPU2 işlemcisi, Nios II işlemcisine göre %41 daha yüksek performans göstermiştir (Şekil 8.5).

8.5 Temel Algoritma Performansları

Temel algoritmalardan QuickSort, Bubble Sort, Fibonacci Serisi için SelCPU2 ve Nios II işlemcileri ile performansları değerlendirilmiştir. Algoritma değerlendirmelerinde diziler hafızada SDRAM üzerinde saklanmıştır. QuickSort algoritması, rastgele değer atanmış 1.000.000 elemanlık dizi üzerinde yapılmıştır. Bubble Sort algoritması, rastgele

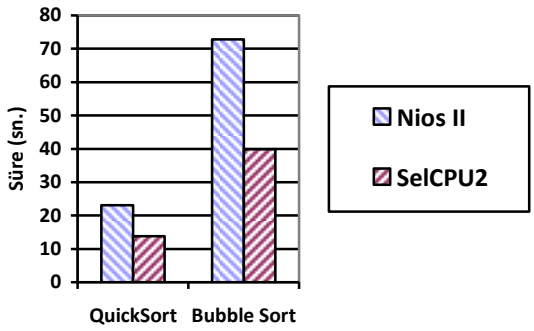
değer atanmış 10.000 elemanlık dizi üzerinde yapılmıştır. Fibonacci Serisi algoritmasında Fibonacci serisinin ilk 1.000.000 elemanı hesaplanarak, hafızadaki diziye atanmıştır. Aşağıda SelCPU2 işlemcisinin algoritmaları çalıştırma süresini Nios II/f işlemcisi ile karşılaştırmalı olarak verilmiştir (Çizelge 8.5, Çizelge 8.6, Şekil 8.6).

Çizelge 8.5 Nios II/f Algoritma Performansları

GCC C Derleyicisi Optimizasyon Seviyesi	QuickSort	Bubble Sort	Fibonacci Serisi
Yok	23.120 ms.	72.800 ms.	906 ms.
Seviye 1 (-O1)	7.926 ms.	16.493 ms.	257 ms.
Seviye 2 (-O2)	7.056 ms.	14.477 ms.	238 ms.
Seviye 3 (-O3)	7.065 ms.	14.477 ms.	217 ms.

Çizelge 8.6 SelCPU2 Algoritma Performansları

LCC C Derleyicisi Optimizasyon Seviyesi	QuickSort	Bubble Sort	Fibonacci Serisi
Yok	13.765 ms.	39.838 ms.	220 ms.



Şekil 8.6 Algoritma performansları karşılaştırması

8.6 Maksimum Çalışma Frekansı ($f$max)

TimeQuest zamanlama analizi aracı ile tasarlanan sistemin kararlı olarak çalışacağı maksimum frekans ($f$max) hesaplanabilmektedir. Değerlendirmeye alınan SelCPU2 ve Nios II sistemlerinin maksimum çalışma frekansı aşağıda gösterilmiştir (Çizelge 8.7).

Çizelge 8.7 Altera Cyclone III FPGA üzerinde maksimum çalışma frekansları

Nios II/f	SelCPU2
150 MHz	85 MHz

8.7 Alan ve Kaynak Kullanımı

Tasarım derleme raporunda sentezlenen sistemin hiyerarşik olarak hangi kaynakları kullandığı raporlanmaktadır. Aşağıda SelCPU2 işlemcisinin kullandığı FPGA kaynakları Nios II/f ile karşılaştırmalı olarak verilmiştir (Çizelge 8.8). SelCPU2 işlemcisi, Nios II işlemcisine göre %26 daha az Lojik Eleman ve %62 daha az Flip-Flop kullanmaktadır. Nios II işlemcisinde komut önbelleğinin olması nedeniyle, hafıza bloğu kullanımı SelCPU2 işlemcisinden daha fazladır. SelCPU2 işlemcisi 8 adet gömülü çarpma birimi kullanırken, Nios II 4 adet kullanmaktadır. Nios II ise çarpma, kaydırma ve döndürme işlemlerini ardışık düzen işleme (pipeline) ile yapmaktadır. Bu sayede Nios II, daha az çarpma birimine ihtiyaç duyar ancak daha fazla Flip-Flop kullanır ve daha geç (5 saat vuruşunda) sonuç üretmektedir. SelCPU2 işlemcisinde çarpma, döndürme kaydırma birimi iki saat vuruşunda ancak ardışık düzen işleme (pipeline) yapı kullanmadan çalışmaktadır.

Çizelge 8.8 Alan ve kaynak kullanımları

Cyclone III Kaynakları	Nios II/f	SelCPU2
LE (Lojik Eleman) Sayısı	2.397	1.771
Flip-Flop Sayısı	1.368	512
Hafıza Blok (M9K) / Hafıza (Bit)	14 / 71.424	9 / 58.368
Gömülü Çarpma Birimi (9x9 Çarpma Birimi)	4	8

8.8 Tahmini Güç Tüketimi Analizi

PowerPlay Güç analiz aracı ile sistemlerin tahmini güç analizi yapılmıştır. Toplam güç tüketiminde SelCPU2 işlemcisi, Nios II işlemcisine göre %4,3 daha az güç harcamaktadır. Aşağıda SelCPU2 işlemcisinin güç tüketimi Nios II işlemcisi ile karşılaştırmalı olarak verilmiştir (Çizelge 8.9).

Çizelge 8.9 Tahmini güç tüketim analizi sonuçları

Güç Tüketimi	Nios II/f	SelCPU2
Çekirdek Dinamik Termal Güç Tüketimi	71,0 mW	61,7 mW
Çekirdek Statik Termal Güç Tüketimi	58,7 mW	58,6 mW
Giriş/Çıkış Termal Güç Tüketimi	95,4 mW	88,6 mW
Toplam Termal Güç Tüketimi	218,4 mW	208,9 mW

BÖLÜM 9

SONUÇ VE ÖNERİLER

Bu tez çalışmasında, SelCPU2 işlemcisi işlevsel, sade, az yer kaplayacak, az kaynak kullanacak, yüksek maksimum çalışma frekansı ve yüksek performans hedefleri ile tasarlanarak gerçeklenmiştir. SelCPU2 işlemcisi ile oluşturulan bir SOC sistemi, Altera Nios II işlemcisi ile oluşturulan eşdeğer bir SOC sistemi ile çalışma performansı, kaynak kullanımı, maksimum çalışma frekansı ve güç tüketimi açısından karşılaştırılmıştır. SelCPU2 işlemcisinin, Nios II işlemcisi ile karşılaştırılmalı olarak ortak assembly kodu ile yapılan Dhrystone performans testinde, Nios II işlemcisine göre %41 daha yüksek performans gösterdiği görüldü. Nios II işlemcisi 6 aşamalı ardışık düzen (pipeline) çalışması sayesinde 150 MHz maksimum frekansına ulaşabilmekteyken, SelCPU2 işlemcisinin 3 aşamalı ardışık düzen çalışması ile maksimum çalışma frekansı 85 MHz olmuştur. Maksimum frekans dışında diğer tüm değerlendirmelerde SelCPU2 işlemcisinin daha üstün sonuç verdiği gözlenmiştir. Nios II işlemcisi, SelCPU2 işlemcisine göre çok daha fazla opsiyonel özellik içermektedir. Ancak SelCPU2 işlemcisinin FPGA üzerinde ihtiyaç duyulabilecek gömülü uygulamaların birçoğunu karşılayacak özellik ve performansa sahip olduğu görülmüştür.

Değerlendirmeler sonucunda SelCPU2 işlemcisinin tasarım hedeflerine ulaşıldığı görülmüştür. İşlemcinin performansını ve özellikleri ile ilgili ileride bir takım çalışmalar yapmak mümkündür. Komut hafızasının çip dışındaki hafıza birimlerinden çalışması durumunda performansı arttırmak için komut ön belleği eklenebilir. Çarpma, döndürme ve kaydırma birimi ardışık düzenli olarak çalıştırılabilir ancak daha fazla kaynak kullanılacağından tercih edilmemiştir. Kullanıcı tanımlı komut çalıştırma özelliği eklenebilir. Bölme işlemi hız için optimize edilerek iki katı hızlı fakat daha fazla alan

kaplayarak tasarlanabilir. Çoklu işlemci desteği eklenebilir. Hafıza koruma veya hafıza yönetim birimi eklenebilir. Kesmelerin yoğun olarak kullanılacağı sistemlerde performansı arttırmak için, gölge yazmaç seti özelliği eklenebilir. Reel sayılar üzerinde işlem yapabilmek için kayan noktalı işlem birimi eklenebilir. Optimize kod üretmek ve GCC C/C++ kütüphanelerinden yararlanmak için GCC C/C++ derleyicisine arka-plan desteği geliştirilebilir.

KAYNAKLAR

[1] Under New Management, The Economist, http://www.economist.com/node/13405279, , 15 Mart 2011.

[2] Kneading Chips, The Economist, http://www.economist.com/node/17528052, 15 Mart 2011.

[3] Altera, "Strategic Considerations for Emerging SOC FPGAs", http://www.altera.com/literature/wp/wp-01157-embedded-soc.pdf, 15 Mart 2011.

[4] Yasuura H., Tomiyama H., Inoue A and Fajar E, (1998). "Embedded System Design Using Soft-Core Processor and Valen-C", Journal of Information Science and Engineering 14: 587-603

[5] Langen D., Niemann J., Porrmann M., Kalte H. and Rückert U., (2002). "Implementation of a RISC Processor Core for SOC Designs -FPGA Prototype vs. ASIC Implementation", Proc. of the IEEE-Workshop: Heterogeneous reconfigurable Systems on Chip

[6] Plavec F.,(2004). Soft-Core Processor Design, Thesis for Master of Applied Science, University of Toronto, Graduate Department of Electrical and Computer Engineering, Toronto.

[7] Yiannacouras P,(2005). The Microarchitecture of FPGA-Based Soft Processors. Thesis for Master of Applied Science, University of Toronto, Graduate Department of Electrical and Computer Engineering, Toronto.

[8] Joost R., Salomon R,(2006). "Advantages of FPGA-Based Multiprocessor Systems in Industrial Applications", Industrial Electronics Society, 2005. IECON 2005. 31st Annual Conference of IEEE, 6-10 Kasım 2005.

[9] Tong J., Anderson I. and Khalid M., (2006). " Soft-Core Processors for Embedded Systems", The 18th International Confernece on Microelectronics (ICM) 16-19 Aralık. 2006, Dhahran, 170 – 173.

[10] Sheldon D., Vahid F. and Onardi S.,(2007). "Soft-core Processor Customization using the Design of Experiments Paradigm", Design, Automation & Test in Europe Conference & Exhibition, 16-20 Nisan 2007., 821- 826.

[11] Yu J. and Lemieux G., (2007). “A Case for Soft Vector Processors in FPGAs”, International Conference on Field-Programmable Technology, ICFPT 2007, 12-14 Aralık 2007, 341-344.

[12] Başak S., SelCPU, CPU Turkey, “http://www.cpu-turkey.com/members/profile.aspx?id=48”, 15 Mart 2011.

[13] Parnell K. and Mehta N.,(2004). Introduction to Programmable Logic, Xilinx Inc.

[14] Logic Elements and Logic Array Blocks in the Cyclone III Device Family http://www.altera.com/literature/hb/cyc3/cyc3_ciii51002.pdf, 15 Mart 2011.

[15] Hardware Description Language, http://en.wikipedia.org/wiki/Hardware_description_language, 15 Mart 2011.

[16] Overview of Embedded Busses, http://emsys.denayer.wenk.be/empro/Overview of Embedded Busses.pdf, 2003.

[17] AMBA Specification Rev. 2.0, http://www.arm.com, 15 Mart 2011.

[18] Avalon Interface Specifications v1.2, http://www.altera.com, 15 Mart 2011.

[19] CoreConnect Bus Architecture , https://www-01.ibm.com/chips/techlib/techlib.nsf/literature/CoreConnect_Bus_Architecture, 15 Mart 2011.

[20] WISHBONE SoC Architecture Specification, Revision B.3, http://opencores.org, 15 Mart 2011.

[21] Kilts S., (2007). Advanced FPGA Design Architecture, Implementation and Optimization, Wiley.

[22] Fraser C. and Hanson D., (1991). A Retargetable Compiler for ANSI C, C. SIGPLAN Notices, 26:29-43.

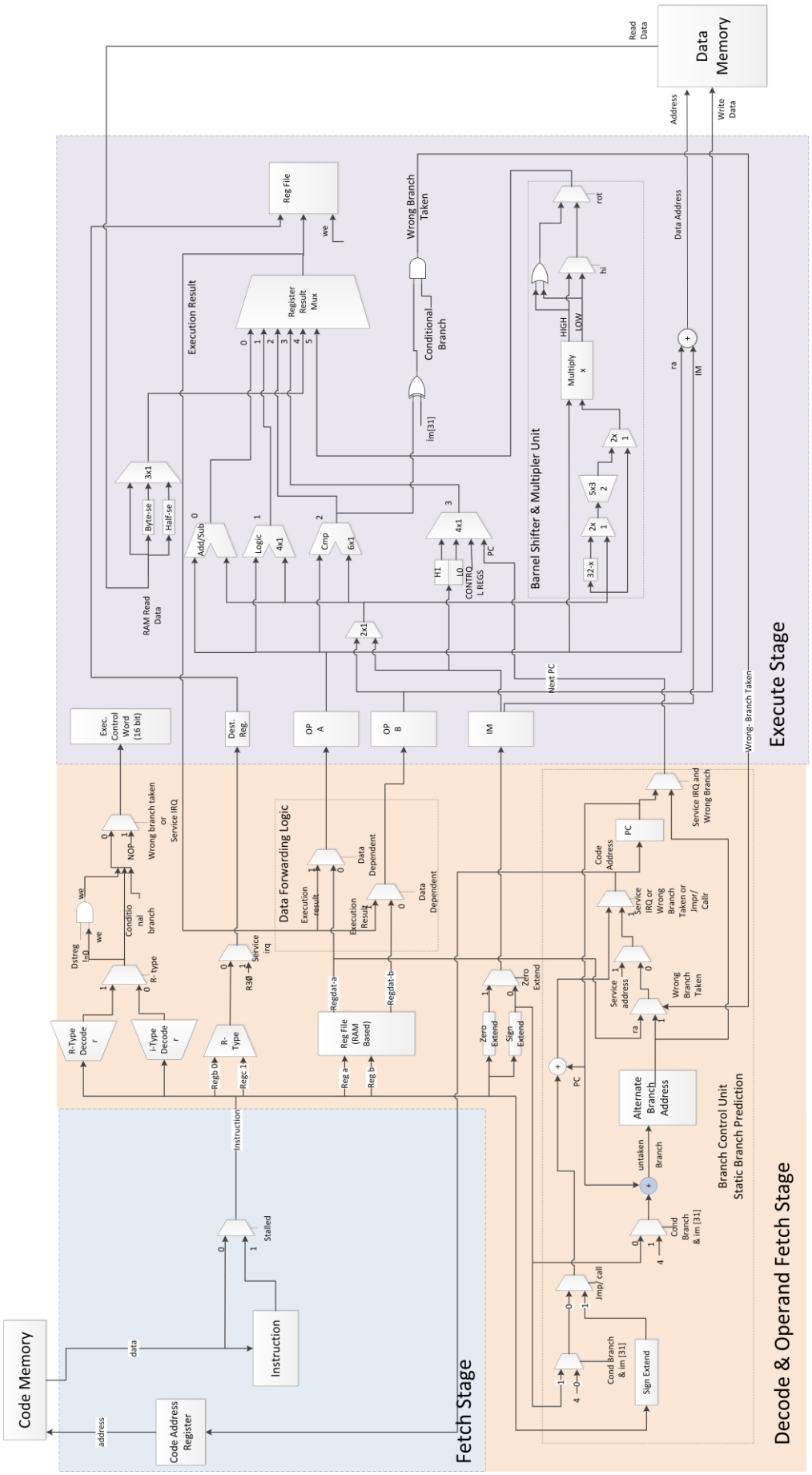
[23] Nios II Processor Reference Handbook, NII5V1-10.0, http://www.altera.com, 15 Mart 2011.

[24] Weicker, R.,(1984). Dhrystone: A Synthetic Systems Programming Benchmark , Communications of the ACM (CACM), 27(10):1013-1030.

[25] Altera DE0 User Manual, http://www.terasic.com, 15 Mart 2011.

EK-A

SELCPU2 MİMARİ YAPISI


Code Memory
Code Address Register
address
data
Instruction
Stalled
Instruction
Fetch Stage
Decode & Operand Fetch Stage
Branch Control Unit
Static Branch Prediction
Alternate Branch Address
PC
untaken Branch
Cond Branch & im [31]
Sign Extend
Jmp/ call
Cond Branch & im [31]
Regb 0
Regc 1
R-Type Decode r
i-Type Decode r
R-Type
R-type
i-Type
Dstreg !=0
we
Conditio nal branch
NOP
Wrong branch taken or Service IRQ
Service irq
R30
Reg File (RAM Based)
Regdat-a
Regdat-b
Reg a
Reg b
Data Forwarding Logic
Execution result
Data Dependent
Zero Extend
Sign Extend
Service address
Service IRQ or Wrong Branch Taken or Jmpr/ Calr
Code Address
Service IRQ and Wrong Branch
Wrong Branch Taken
Exec. Control Word (16 bit)
Dest. Reg.
OP A
OP B
IM
Next PC
Wrong. Branch Taken
Execute Stage
Barrel Shifter & Multipler Unit
Add/Sub
Logic
Cmp
Byte-se
Half-se
RAM Read Data
Execution Result
Register Result Mux
Reg File
we
Wrong Branch Taken
Conditional Branch
Im[31]
HI
LO CONTRO L REGS
PC
4x1
6x1
3x1
2x1
32-x
5x3
Multiply x
HIGH
LOW
rot
hi
ra
IM
Data Address
Address
Write Data
Read Data
Data Memory

EK-B

SELCPU2 KOMUT SETİ

Opcode	Komut	Kullanım	İşlev	Komut Tipi
0	cmpeqi	cmpeqi rb,ra,im	rb = ra==im ? 1 : 0	I-Tipi
1	cmpnei	cmpnei rb,ra,im	rb = ra!=im ? 1 : 0	I-Tipi
2	cmpgei	cmpgei rb,ra,im	rb = ra>=im ? 1 : 0	I-Tipi
3	cmplti	cmplti rb,ra,im	rb = ra<im ? 1 : 0	I-Tipi
4	cmpgeui	cmpgeui rb,ra,im	rb = ra>=im ? 1 : 0	I-Tipi
5	cmpltui	cmpltui rb,ra,im	rb = ra<im ? 1 : 0	I-Tipi
6	andi	andi rb,ra,im	rb = ra & im	I-Tipi
7	ori	ori rb,ra,im	rb = ra | im	I-Tipi
8	beq	beq ra,rb,label	pc = ra==rb ? pc+label : pc + 4	I-Tipi
9	bne	bne ra,rb,label	pc = ra!=rb ? pc+label : pc + 4	I-Tipi
10	bge	bge ra,rb,label	pc = ra>=rb ? pc+label : pc + 4	I-Tipi
11	blt	blt ra,rb,label	pc = ra<rb ? pc+label : pc + 4	I-Tipi
12	bgeu	bge ra,rb,label	pc = ra>=rb ? pc+label : pc + 4	I-Tipi
13	bltu	bltu ra,rb,label	pc = ra<rb ? pc+label : pc + 4	I-Tipi
14	xori	xori rb,ra,im	rb = ra ^ im	I-Tipi
15	nori	nori rb,ra,im	rb = ~(ra | im)	I-Tipi
16	addi	addi rb,ra,im	rb = ra+im	I-Tipi
17	subi	subi rb,ra,im	rb= ra-im	I-Tipi
18	muli	muli rb,ra,im	rb = ra * im	I-Tipi
23	rdctrl	rdctrl rb,im	rb=pc(im:0)/perf MIPS(im:3)	I-Tipi
28	movih	movih rb,im	rb = im:ra[15:0] , (ra,rb same)	I-Tipi
29	movil	movil rb,im	rb = ra[31:16]:im , (ra,rb same)	I-Tipi
30	movsx	movsx rb,ra,im	rb = işaret uzat im=1:ra[7:0] im=2:ra[15:0]	I-Tipi
39	initd	initd [ra+im]	Önbellek satırını geçersiz kıl	I-Tipi
40	ldw	ldw rb,[ra+im]	rb = RAM[ra+im]	I-Tipi
42	ldb	ldb rb,[ra+im]	rb = RAM[ra+im],byte,signed	I-Tipi
43	ldh	ldh rb,[ra+im]	rb = RAM[ra+im],half,signed	I-Tipi
44	stw	stw rb,[ra+im]	RAM[ra+im] = rb	I-Tipi
46	stb	stb rb,[ra+im]	RAM[ra+im] = rb	I-Tipi
47	sth	sth rb,[ra+im]	RAM[ra+im] = rb	I-Tipi

Opcode	Komut	Kullanım	İşlev	Komut Tipi
48	jmpr	jmpr ra	pc=ra	I-Tipi
49	callr	callr ra	pc=ra, retreg=pc+4	I-Tipi
50	ret	ret	pc=ra ,ra is ret, reg r31	I-Tipi
51	iret	iret	pc=ra ,ra is iret, reg r30	I-Tipi
52	jmp	jmp label	pc=pc+im26*4	J-Tipi
53	call	call label	pc=pc+im26*4, retreg=pc+4	J-Tipi

Opex	Komut	Kullanım	İşlev	Komut Tipi
6	and	and rc,ra,rb	rc = ra & rb	R-Tipi
7	or	or rc,ra,rb	rc = ra | rb	R-Tipi
8	cmpeq	cmpeq rc,ra,rb	rc = ra==rb ? 1 : 0	R-Tipi
9	cmpne	cmpne rc,ra,rb	rc = ra!=rb ? 1 : 0	R-Tipi
10	cmpge	cmpge rc,ra,rb	rc = ra>=rb ? 1 : 0	R-Tipi
11	cmplt	cmplt rc,ra,rb	rc = ra<rb ? 1 : 0	R-Tipi
12	cmpgeu	cmpgeu rc,ra,rb	rc = ra>=rb ? 1 : 0	R-Tipi
13	cmpltu	cmpltu rc,ra,rb	rc = ra<rb ? 1 : 0	R-Tipi
14	xor	xor rc,ra,rb	rc = ra ^ rb	R-Tipi
15	nor	nor rc,ra,rb	rc = ~(ra | rb)	R-Tipi
16	add	add rc,ra,rb	rc = ra+ra	R-Tipi
17	sub	sub rc,ra,rb	rc= ra-rb	R-Tipi
18	mul	mul rc,ra,rb	rc = ra * rb,low word	R-Tipi
19	mulxss	mulxss rc,ra,rb	rc = ra * rb,high word, işaretli*işaretli	R-Tipi
20	mulxsu	mulxsu rc,ra,rb	rc = ra * rb,high word, işaretli*işaretsiz	R-Tipi
21	mulxuu	mulxuu rc,ra,rb	rc = ra * rb,high word, işaretsiz*işaretsiz	R-Tipi
22	rol	rol rc,ra,rb	rc = ra rotate-left rb	R-Tipi
23	ror	ror rc,ra,rb	rc = ra rotate-right rb	R-Tipi
24	sll	sll rc,ra,rb	rc = ra << rb	R-Tipi
25	sra	sra rc,ra,rb	rc = ra >> rb	R-Tipi
26	srl	srl rc,ra,rb	rc = ra >> rb	R-Tipi
28	divu	divu rc,ra,rb	rc = ra / rb	R-Tipi
29	div	div rc,ra,rb	rc = ra / rb	R-Tipi
30	modu	modu rc,ra,rb	rc = ra mod rb	R-Tipi
31	mod	mod rc,ra,rb	rc = ra mod rb	R-Tipi
48	cli	cli	ie=0, kesmeleri engelle	R-Tipi
49	sti	sti	ie=1,, kesmelere izin ver	R-Tipi
54	roli	roli	rc = ra rotate-left im5	R-Tipi
56	slli	slli	rc = ra << im5	R-Tipi
57	srai	srai	rc = ra >> im5	R-Tipi
58	srli	srli	rc = ra >> im5	R-Tipi

EK-C

ÖRNEK ASSEMBLY PROGRAMI

```
;SelCPU2 Assembly ile QuickSort Algoritması
;function qs
qs:
addi sp,sp,-48
stw r25,[sp+16]
stw r26,[sp+20]
stw r27,[sp+24]
stw r28,[sp+28]
stw r31,[sp+32]
stw r4,[sp+48]
stw r5,[sp+52]
stw r6,[sp+56]
ldw r15,[sp+52]
mov r28,r15
ldw r14,[sp+56]
mov r27,r14
movi r13,2
add r15,r15,r14
div r15,r15,r13
sll r15,r15,r13
ldw r14,[sp+48]
add r15,r15,r14
ldw r26,[r15]
jmp L.48
L.50:
addi r28,r28,1
L.51:
slli r15,r28,2
ldw r14,[sp+48]
add r15,r15,r14
ldw r15,[r15]
bge r15,r26,L.53
ldw r15,[sp+56]
blt r28,r15,L.50
L.53:
jmp L.55
L.54:
subi r27,r27,1
L.55:
slli r15,r27,2
ldw r14,[sp+48]
add r15,r15,r14
ldw r15,[r15]
```

```
bge r26,r15,L.57
ldw r15,[sp+52]
bgt r27,r15,L.54
L.57:
bgt r28,r27,L.58
movi r15,2
ldw r14,[sp+48]
sll r13,r28,r15
add r13,r13,r14
ldw r25,[r13]sll r15,r27,r15
add r15,r15,r14
ldw r15,[r15]
stw r15,[r13]
slli r15,r27,2
ldw r14,[sp+48]
add r15,r15,r14
stw r25,[r15]
addi r28,r28,1
subi r27,r27,1
L.58:
L.48:
ble r28,r27,L.51
ldw r15,[sp+52]
bge r15,r27,L.60
ldw r4,[sp+48]
ldw r5,[sp+52]
mov r6,r27
call qs
L.60:
ldw r15,[sp+56]
bge r28,r15,L.62
ldw r4,[sp+48]
mov r5,r28
ldw r6,[sp+56]
call qs
L.62:
L.46:
ldw r25,[sp+16]
ldw r26,[sp+20]
ldw r27,[sp+24]
ldw r28,[sp+28]
ldw r31,[sp+32]
addi sp,sp,48
ret
;end qs
;function quicksort
quicksort:
addi sp,sp,-32
stw r31,[sp+16]
stw r4,[sp+32]
stw r5,[sp+36]
ldw r4,[sp+32]
mov r5,r0
ldw r15,[sp+36]
subi r6,r15,1
call qs
L.64:
ldw r31,[sp+16]
addi sp,sp,32
ret
;end quicksort
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	: Selçuk BAŞAK
Doğum Tarihi ve Yeri	: 1977, İstanbul
Yabancı Dili	: İngilizce
E-posta	: selcuk@selsistem.com

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Lisans	Bilgisayar Mühendisliği	Yıldız Teknik Üniversitesi	1998
Lise	Fen	İst. Atatürk Fen Lisesi	1993

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2002 – ….	SelSistem Bilgi ve İletişim Teknolojileri	Genel Müdür / Ar-Ge Müh.
2000-2001	K.K.K Personel Yönetim Bilgi Sistemleri Şb.	OBİ Subayı / Yazılım Müh.
1998-2000	Osmanlı Bankası A.Ş. – Garanti Teknoloji A.Ş.	Sistem Yazılım Müh.

YAYINLARI

Bildiri

1. FPGA Üzerinde SelCPU İşlemcisi İle System-On-Chip Uygulaması, Gömsis Gömülü Sistemler Sempozyumu, İTÜ, 3-4-5 Kasım 2008.