OpenRISC Tabanlı Bir Kırmıküstü Sistemin FPGA Üzerinde Gerçeklenmesi ve Linux Kernel Kurulumu Implementation of an OpenRISC Based SoC and Linux Kernel Installation on FPGA

Latif Akçay[1], Mehmet Tükel[2], Sıddıka Berna Örs[1]
[1]Elektronik ve Haberleşme Mühendisliği Bölümü, İstanbul Teknik Üniversitesi, İstanbul, Türkiye
{akcayl,orssi}@itu.edu.tr
[2]Anka Mikroelektronik Sistemler, İstanbul, Türkiye
mehmet.tukel@ankasys.com

***Özetçe* —Gömülü sistem tasarımları ve uygulama alanları günümüzde oldukça yaygınlaşmıştır. Bu çalışmada gömülü sistem tasarımlarında kullanılabilecek, açık kaynak kodlu, OpenRISC tabanlı kırmıküstü sistemlerin FPGA (Field Programmable Gate Array) üzerinde gerçeklenmesi ve Linux işletim sistemi çekirdeğinin çalıştırılması anlatılmıştır. İşlemci ve çevresel birimlere ait kaynak kodların açık ve kullanıcı tarafından değiştirilebilir olması ve herhangi bir lisans ücretinin talep edilmemesi OpenRISC tabanlı kırmıküstü sistemleri diğer çözümlerden farklı kılmaktadır. Ayrıca bu çalışmada OpenRISC tabanlı işlemciler üzerinde çalıştırılabilir kod derlemek için uyarlanan GNU GCC derleyici takımının ve yazılım simülatörünün bilgisayara kuruluşu ve kullanılışı gösterilmiştir.**

Anahtar Kelimeler*—*OpenRISC, kırmıküstü sistem, donanım tasarımı, linux kernel, mikroişlemci, gerçeklenebilir işlemci.

***Abstract*—Embedded system designs and applications have been more common today. The aim of this study is to tell how to implement open source OpenRISC based SoC's, which can be used for embedded system designs, on FPGA and how to install Linux Kernel. It makes OpenRISC based SoC's different from other Soc's due to it's modifiable open source codes for processors and all peripherals, and no license fee demand. Furthermore, it has been showed how to install and use GNU GCC toolchain and software simulator, or1ksim, which had been ported to the system for generating executable codes that can be run on OpenRISC based processors.**

Keywords*—*OpenRISC, SoC, hardware design, linux kernel, microprocessor, softcore processor, .

I. GİRİŞ

Açık kaynak kodlu tasarım düşüncesi başlangıcından bu yana ciddi bir ilerleme kaydetmiştir [1]. Günümüzde siber güvenlik konularına kadar uzanan birçok alanda önemli bir rol oynamaktadır [2]. Özellikle yazılım dünyasında açık kaynak kodlu ve lisans ücreti olmayan çok sayıda program bulunmaktadır. Bununla beraber özellikle son yıllarda açık kaynak kodlu donanım tasarımları da ortaya çıkmıştır. Tasarımcılar Verilog [3], VHDL (VHSIC Hardware Description Language) [4] veya SystemC [5] gibi donanım tanımlama dilleri kullanılarak geliştirdikleri tasarımlara ait kodları paylaşmakta ve isteyen herkesin kullanımına izin vermektedir. Tasarımların kaynak kodlarının açık olması lisans ücreti olmadığı anlamı taşımasa da genel olarak kullanıcılardan böyle bir talepte bulunulmamaktadır. Bu çalışmaların en önemli tarafı ise donanım tasarımı konusunda eğitici bir rehber konumunda olmalarıdır. Özellikle karmaşık sistemlerin tasarımı ve gerçeklenmesi konusunda ciddi bir kaynak haline gelen bu çalışmaların çoğu OpenCores topluluğu tarafından ortak bir alandan paylaşılmaktadır [6].

Bu çalışmanın ilk bölümünde gerçeklenebilir işlemci kavramı açıklanmış ve günümüzde en çok bilinen dört tanesi kısaca kıyaslanmıştır. Ayrıca bu sistemleri kullanmanın avantajlı yönlerine dikkat çekilmiştir. Daha sonra açık kaynak kodlu OpenRISC 1000 komut seti ailesine mensup işlemciler üzerinde makina kodu çalıştırmak için gerekli olan GNU GCC (GNU Compiler Collection) derleyici takımının bilgisayara kuruluşu ele alınmıştır. Sonraki bölümlerde ise bu işlemciler ve etrafına bağlanmış olan çevresel birimlerin oluşturduğu kırmıküstü sistemlerin, Digilent firmasına ait Atlys geliştirme kartı üzerinde gerçeklenmesi ve uygulamaları anlatılmıştır [7].

II. GERÇEKLENEBİLİR İŞLEMCİLER VE KIRMIKÜSTÜ SİSTEMLER

A. Tanım ve Kıyaslama

Gerçeklenebilir işlemci kavramı; gerçek bir işlemcinin Verilog veya VHDL dili ile yazılmış bir modeli olarak tanımlanabilir [8].

Tasarıma göre değişmekle birlikte gerçeklenebilir işlemcilerin birçok özelliği kullanıcı tarafından değiştirilebilmektedir. Örneğin işlemcinin veri veya komut ön belleği olup olmayacağı, olacaksa miktarının ne kadar olacağı, iletişim hattı sayısı veya veri arayüzü seçimi gibi özellikler uygulamaya göre ayarlanabilmektedir. En çok bilinen gerçeklenebilir işlemciler Xilinx firmasının geliştirdiği MicroBlaze [9], Aeroflex Gaisler firmasının geliştirdiği Leon [10] ve tamamen gönüllü katılımcıların ortak çalışmalarıyla geliştirilen OR1200 [11] işlemcileridir. Tablo I'de bu işlemcilere ait bazı önemli özelliklerin karşılaştırılması gösterilmektedir. Tablodaki değerler işlemcilerin versiyonu yükseldikçe değişebilmektedir.

978-1-5090-1679-2/16/$31.00

İşlemci	Or1200	Microblaze	Leon3
Kaynak Kodlar	Açık	Kapalı	Açık
Lisans Kısıtlamaları	Yok	Var	Var
Frekans (MHz)	50-300	200	125-400
Komut Kümesi	32-Bit RISC	32-Bit RISC	32-Bit RISC
İletişim Hattı	5 seviye	5 seviye	7 seviye
Uyumlu Arayüz	Wishbone	AXI/PLB	AMBA
Ön Bellek	64 KB	64 KB	256 KB
Adres-Veri Hattı	32-Bit	32-Bit	32-Bit
Gerçekleme	FPGA/ASIC	FPGA	FPGA/ASIC

Tablo I: Gerçeklenebilir İşlemcilerin Karşılaştırılması [8].

B. Kımıküstü Sistemlerin Kullanım Amaçları

Kırmıküstü sistemler bir veya daha çok işlemci ve belirli bir arayüz ile bağlanmış çeşitli çevresel elemanların oluşturduğu yapıya verilen isimdir. Gerçeklenebilir işlemciler kullanılarak geliştirilen kırmıküstü sistemlere arayüz ile uyumlu olarak tasarlanmak koşulu ile kullanıcı tasarımı donanımlar veya geliştiricilerin sunduğu çeşitli hazır donanımlar eklenebilir. Böylece oldukça esnek bir tasarım imkanı sunulmuş olmaktadır [8]. İhtiyaç duyulabilecek birçok donanımın hazır olarak sunulması tasarım sürecinin hızlanması da sağlamaktadır. Tüm bunlara ek olarak kırmıküstü sistemlerin FPGA üzerinde gerçeklenebilir olması, muhtemel hataların önceden tespit edilip çözülmesini mümkün kılmaktadır.

III. OpenRISC Tabanlı Kırmıküstü Sistemler

A. OpenRISC 1000 Komut Kümesi

OpenRISC 1000 açık kaynak kodlu komutlar geliştirmek üzere RISC (Reduced Instruction Set Computing) prensipleriyle tasarlanmış bir komut kümesi mimarisidir. Bu mimari geniş bir yelpazede (gömülü sistemler, otomotiv, taşınabilir bilgisayarlar vs) kullanılabilecek 32 ve 64 bit açık kaynak kodlu komutlar üretmeyi hedeflemektedir [13].

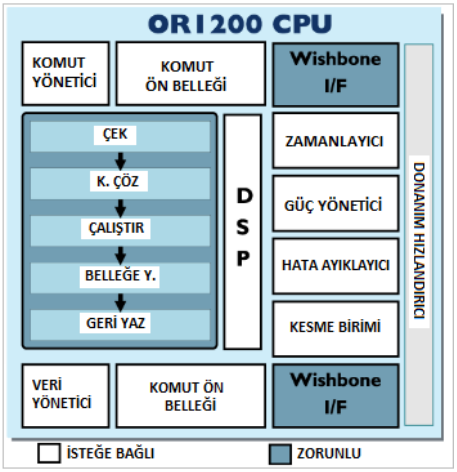
Bu kapsamda hazırlanan tasarımlar tamamen açıktır ve dileyen herkes bedel ödemeden bu tasarımları olduğu gibi veya üzerinde değişiklik yaparak kullanma hakkına sahiptir [12].

B. OR1200 İşlemci Mimarisi

OpenRISC 1000 komut kümesi gerçeklemelerinin ilki Damjen Lampret tarafından yapılmış ve OR1200 CPU (Central Processing Unit) olarak adlandırılmıştır [11]. OR1200 çekirdeğinin genel özelliklerini şöyle sıralamak mümkündür [11];

- Yüksek performanslı işlem yapabilme yeteneği
- Yüksek hızlarda çalışan veri bellek ve bellek yönetimi
- Wishbone Arayüz uyumu
- İşlemciye ilişkin yapısal parametrelerinin kullanıcı tarafından kolayca değiştirilebilir olması

Şekil 1'de OR1200 işlemci çekirdeğinin yapısı gösterilmiştir. Bazı birimler işlemcinin çalışabilmesi için zorunlu olarak gerçeklenmelidir. Diğer birimlerin gerçeklenmesi ise tercihe bağlıdır.



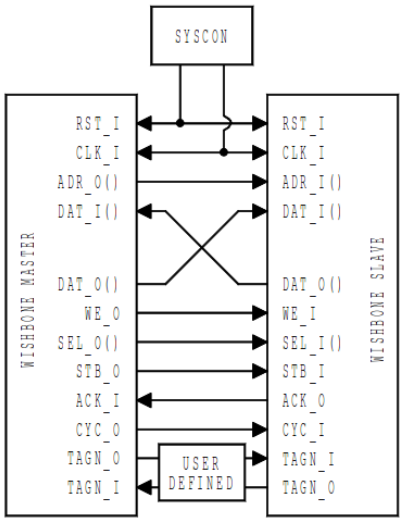
Şekil 1: OR1200 CPU İsteğe Bağlı Ve Zorunlu Birimler [11].

C. Wishbone Kırmıküstü Sistem Arayüzü

Wishbone arayüzü bir sistemde birimler arasındaki haberleşmenin sağlanaması için tasarlanmış olan açık kaynak kodlu bir veri yoludur [14]. Wishbone tasarımı kullanıcıya esnek bir iletişim yapısı sunmaktadır [14].

OpenRISC tabanlı kırmıküstü sistemlerde Wishbone arayüzü kullanılmaktadır. Kullanıcılar eklemek istediği donanımlara Wishbone arayüzü de yazmalıdır.

Genel olarak bir Wishbone arayüzlü sistemde iki birim arasındaki haberleşme sinyalleri (senkron saat, reset, adres, veri, birim seçildi, işlem tamamlandı, hata, yeniden dene ve kullanıcı tanımlı işaretler) Şekil 2'de gösterilmiştir.



Şekil 2: Wishbone Arayüz Sinyalleri [14].

D. or1k- Derleyici Takımı

Derleyici Takımı (Toolchain) yazılım dünyasında metin editörü üzerinde yazılmış bir kodun makina komutları haline getirilebilmesi için gerekli olan tüm programların oluşturduğu bir

grup yazılım anlamına gelmektedir [15]. Gömülü sistem uygulamalarında sık kullanılan C ve C++ dillerinde yazılan kodların OpenRISC işlemcileri üzerinde çalıştırılması için GNU GCC derleyici takımı bu mimariye uyarlanmıştır. Üç farklı C kütüphanesi ile derlenebilen GCC'nin OpenRISC versiyonu bu kütüphanelerin işleveselliklerine göre adlandırılmıştır.

- **or1k-elf- :** İşletim sistemi olmayan sistemlerde çalışacak uygulamalar için (Newlib Kütüphanesi)
- **or1k-linux-uclibc :** Linux işletim sistemi üzerinde çalışacak uygulamalar için (uClibc kütüphanesi)
- **or1k-linux-musl :** Linux işletim sistemi üzerinde çalışacak uygulamalar için (musl kütüphanesi)

Derleyici takımı ve yazılım simülatörünün kurulumuna ilişkin adımlar topluluğun resmi internet sayfasında yayınlanmaktadır [16].

Kurulum yapıldıktan sonra aşağıdaki örneklerde olduğu gibi basit derlemeler yapılabilir. İki örneğin üreteceği deneme.elf dosyasıının farkı birisinin gerçekten OpenRISC bir işlemci üzerinde çalışabilir olması, diğerinin ise or1ksim yazılım simültörü üzerinde çalışmasından kaynaklanmaktadır.

or1k-elf-gcc -mboard=atlys deneme.c -o deneme.elf

or1k-elf-gcc -g -mboard=or1ksim deneme.c -o deneme.elf

Derlenmiş bir kod simülatör üzerinde aşağıdaki gibi denenebilmektedir:
or1k-elf-gdb deneme.elf
target sim
load
run

IV. KIRMIKÜSTÜ SİSTEMİN FPGA ÜZERİNDE GERÇEKLENMESİ

A. Donanım Kodları ve Yardımcı Yazılımlarla Yapılan İşlemler

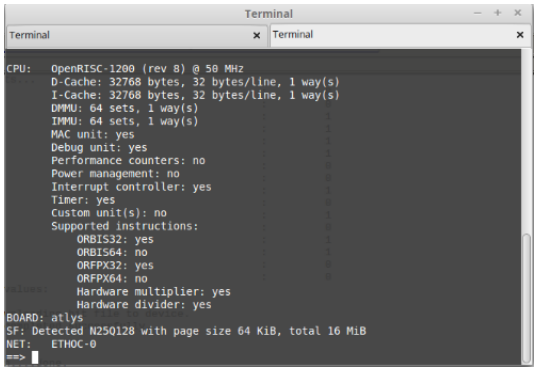
OpenRISC 1000 komut kümesinin OR1200 işlemci gerçeklemesi ve bu işlemciye Wishbone arayüzü ile bağlanmış olan çevresel birimlerin oluşturduğu ORPSoC-v2 ve ORPSoC-v3 (OpenRISC Reference Platform System on Chip) kırmıküstü sistemlerine ait donanım kodları ve bu kodların kullanımını kolaylaştıran bazı yardımcı yazılımlar projeye ait resmi depolama alanı olan GitHub internet sitesinden ücretsiz olarak indirilebilmektedir. Kırmıküstü sistemler OpenRISC tabanlı OR1200 ve mor1kx işlemcilerine ait donanım kodları ve bu işlemcilere bağlı SPI, UART, Ethernet, Gpio, ac97, DDR RAM arayüzü gibi birçok birim içermektedir. Varsayılan ayarlarda kodların sentezlenip yapılandırma dosyalarının üretilmesi için kolaylık sağlamak adına küçük bazı yazılımlar kodlarla birlikte sunulmaktadır. Söz konusu yardımcı yazılımlar, Xilinx veya Altera yazılımlarını arka planda kullanarak sentezler ve yapılandırma dosyası üretir. Aşağıda gösterildiği gibi basit bir komutla ORPSoC-v2 kırmıküstü sistemi için FPGA yapılandırma dosyası üretilebilir. Bu dosya ile FPGA yapılandırıldığında OR1200 işlemci çevresel birimler çalışmaya başlar.

make orpsoc.bit

İşlemci çalışmaya başladığı andan itibaren hangi işlemleri yapacağı kullanıcı tarafından rom.v adlı Verilog dosyası içinde belirlenebilir. SPI (Serial Peripheral Interface) Flash içerisine yazılan programı alıp RAM belleğe kopyalamaya yarayan algoritma indirilen dosyaların içerisinde hazır olarak sunulmuştur. Bu kod derlenip elde edilen komutlar rom.v dosyasına yazılırsa SPI Flash üzerine yazılan her türlü uygulama çalıştırılabilir. Ayrıca FPGA yapılandırma dosyası da SPI Flash formatına dönüştürülebilir. Bu sayede geliştirme kartı enerjisi kesilse bile kırmıküstü sisteme ait yapılandırma dosyası SPI Flash içerisinde sakladığı için kaybedilmeyecektir. Geliştirme kartı her açıldığında FPGA SPI Flash üzerinden yapılandırılır. SPI Flash içerisine sıfırıncı adresten başlayarak yapılandırma dosyası (Atlys üzerindeki FPGA için yaklaşık 1.5 MB) ve OpenRISC işlemcisi üzerinde çalıştırılmak istenen herhangi bir uygulama yazılırsa tüm yazılım ve donanım tasarımı silinmeyecek şekilde saklanmış olur. Bu işlemlerin yapılabilmesi için yazılacak tüm dosyalar .mcs formatına çevrilmelidir. Bu format çevrimi ve yazdırma işlemleri Xilinx firmasının Impact adlı yazılımı ile kolayca yapılabilmektedir.

B. U-Boot Uygulaması

U-Boot (Universial Boot Loader) ARM, MIPS, PowePC gibi birçok platformu destekleyen, açık kaynak kodlu bir dosya transfer yazılımıdır [17]. Bu çalışmada U-Boot kaynak kodları or1k-elf derleyici takımı ile derlenmiştir. U-Boot programı bilgisayar ile Ethernet birimi üzerinden haberleşmektedir. Böylece or1k-elf derleyici takımı araçlarıyla derlenen her türlü uygulama U-Boot programı kullanılarak Ethernet üzerinden doğrudan RAM belleğe transfer edilip çalıştırılabilir. Öncelikle transfer işlemi için TFTP protokolü bilgisayara kurulmuştur. Bu sayede geliştirilen uygulamalar hızlı bir şekilde test edilebilmektedir. Bu kullanımda bir defaya mahsus olmak üzere FPGA yapılandırma dosyası ve derlenmiş U-boot kodu Impact arayüzü ile Spi Flash içerisine yazılır. Bu aşamadan sonra kart her açıldığında FPGA Spi Flash üzerinden yapılandırılır ve OR1200 işlemcisi U-Boot kodunu çalıştırır. U-Boot UART üzerinden haberleşilen terminalde sağlamaktadır. Şekil 3 bu terminal ekranını göstermektedir. Bu sayede denenmek istenen herhangi bir uygulama önce bilgisayardan transfer edilir (tftp <uygulama.bin>) ve ardından (bootm) çalıştırılır. Sonraki bölümde anlatılan Linux Kernel uygulaması da U-boot ile transfer edilerek çalıştırılmıştır.



Şekil 3: Geliştirme Kartında Çalışan U-Boot Programının Sunduğu Arayüz

C. Linux Kernel Kurulumu

Linux Kernel 3.1 sürümünden itibaren OpenRISC 1000 tabanlı işlemcilere destek verilmektedir [18]. Bu çalışmada

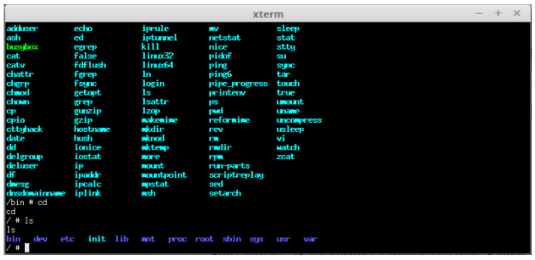
OpenRISC projesinin depolama alanında bulunan en güncel Linux Kernel sürümü kullanılmıştır. Linux Kernel hem or1ksim yazılım simülatörü üzerinde denenmiş hem de Atlys geliştirme kartı üzerinde çalıştırılmıştır. Bunun için öncelikle Linux Kernel kurulumu yapılırken mimari olarak OpenRISC seçilmiştir.

export ARCH=openrisc

export CROSS_COMPILE=or1k-elf-

Bu seçim yapıldığında varsayılan durumda simülatör üzerinde çalışacak biçimde bir derleme ayarı yapılmıştır. Kullanıcı bir değişiklik yapmadığı sürece bu ayarlara göre derleme yapılır. Dolayısıyla make komutu verildikten sonra derleme tamamlandığında simulatör üzerinde çalışabilecek bir vmlinux dosyası üretilecektir. Bu dosya simülatör ile aşağıdaki gibi kullanılarak Linux Kernel or1ksim üzerinde çalıştırılmış olur. or1ksim simülatörü Linux Kernel'in terminal modelini sunamamasından dolayı bu özelliği destekleyen xterm adlı bir terminal emülatar programı kurulmuş ve OpenRISC Linux Kernel ayarlarını saklayan or1ksim.cfg doyasında UART çıkışı xterm programı olarak değiştirilmiştir [19]. Bu sayede xterm üzerinden Linux Kernel terminaline erişilebilmiştir. Aşağıda gösterilen komut, derlenmiş Linux Kernel'in simülatör üzerinde çalıştırılmasını sağlamaktadır. Xterm üzerinde alınan görüntü ise Şekil 4'deki gibidir.

or1k-elf-sim -f arch/openrisc/or1ksim.cfg vmlinux

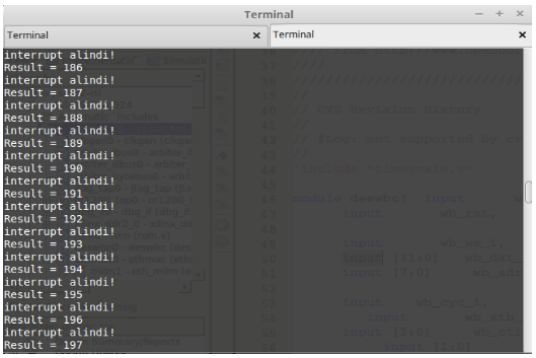


Şekil 4: Linux Kernel'in Simülatör Üzerinde Çalıştırılması.

Linux Kernel simülatör üzerinde çalıştırıldıktan sonra Atlys geliştirme kartında ORPSoC kırmıküstü sistem üzerinde de çalıştırılmıştır. Bunun için öncelikle derleme aşamasında Linux'un sunduğu derleme ayarları menüsüne make menuconfig komutu ile girilir ve Atlys geliştirme kartının özelliklerinin yazılı olduğu atlys.dts adlı dosya seçilir. Ayrıca eğer daha önceden hazırlanan kırmıküstü sistem varsayılan ayarların dışında daha farklı özellikler içeriyorsa bu değişikliklerin Linux Kernel içerisinde bulunan OpenRISC dizininde bulunan ayar dosyalarında da uygulanması gerekir. Bunlara ek olarak Linux Kernel'e ait istenilen birçok ayarlama da yine menüden yapılabilmektedir. İşlemlerin tamamlanıp menünün kaydedilmesinden sonra derleme yapılır (make komutu) ve çalıştırılabilir vmlinux dosyası üretilir. Bu dosya derleyici takımı kullanılarak ikilik formata çevrilir. Daha sonra U-Boot ile transfer edilebilecek formata dönüştürülür.

or1k-elf-objcopy -O binary vmlinux vmlinux.bin
mkimage -n 'Linux for OpenRISC' -A or1k -O linux -T kernel -C none -a 0 -e 0x100 -d vmlinux.bin uImage

Bu işlemlerin ardından uImage dosyası U-Boot ile Geliştirme kartına transfer edilir ve RAM belleğe kopyalanır. Ardından bootm komutuyla Linux Kernel çalıştırılmaya başlanır. Kısa bir süre sonra Linux Kernel'in terminale düştüğü gözlenir. Yani sistem çalışmaya başlar. ORPSoC kırmıküstü sistem üzerinde çalışan Linux Kernel Şekil 5'deki gibi bir ekran görüntüsü oluşturmaktadır.



Şekil 5: Linux Kernel'in Atlys Geliştirme Kartında ORPSoC Kırmıküstü Sistem Üzerinde Çalıştırılması.

KAYNAKÇA

[1] Özaygen, Altay Ş., and O. D. T. Ü. Bilim. "Yenilik olarak özgür-açık kaynak kodlu yazılımların yayılımı: ODTÜ örneği." (2004).

[2] Gözükeleş, İ. "Özgür/Açık kaynak kodlu yazılım ve ulusal yazılım politikaları." Elektrik Mühendisliği 422 (2004).

[3] IEEE Standard for Verilog Hardware Description Language," in IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001) , vol., no., pp.0_1-560, 2006 doi: 10.1109/IEEESTD.2006.99495

[4] IEEE Standard VHDL Language Reference Manual," in IEEE Std 1076-2008 (Revision of IEEE Std 1076-2002) , vol., no., pp.c1-626, Jan. 26 2009 doi: 10.1109/IEEESTD.2009.4772740

[5] SystemC Language, http://www.accellera.org/downloads/standards/systemc, Tarih:23.01.2016

[6] OpenCores, http://opencores.org/, Tarih:23.01.2016

[7] Development Board Documentation, Digilent, Atlys Spartan FPGA

[8] Jason G. Tong, Ian D. L. Anderson and Mohammed A. S. Khalid, "Soft-Core Processors for Embedded Systems", The 18th International Conference on Microelectronics (ICM), 2006

[9] Xilinx MicroBlaze Product , http://www.xilinx.com/products/intellectual-property/microblazecore.html, Tarih:23.01.2016

[10] Gaisler Leon, http://www.gaisler.com/index.php/products/processors/leon3, Tarih:23.01.2016

[11] Lampret D., 2001, OpenRISC 1200 Core Specification, http://www.isy.liu.se/en/edu/kurs/TSEA44/OpenRISC/or1200_spec. pdf, revision 0.7, Tarih:23.01.2016

[12] OpenCores, or1k Toolchain, http://opencores.org/or1k/Main_Page, Tarih:23.01.2016

[13] Lampret D., OpenRISC 1000 Architecture Specification Manual, http://opencores.org/or1k/Architecture_Specification Tarih:23.01.2016

[14] Peterson W.D., 2002, Wishbone System-on-Chip (SoC)Interconnection Architecture For Portable IP Cores, http://cdn.opencores.org/ downloads/wbspec_b3.pdf, revision B.3.,Tarih:23.01.2016

[15] Toolchain, http://www.nongnu.org/avr-libc/user-manual/overview.html, Tarih:23.01.2016

[16] OpenRISC GNU, http://opencores.org/or1k/OpenRISC_GNU_tool_chain, Tarih:23.01.2016

[17] Das, U. "Boot–the Universal Boot Loader." http://www.denx.de/wiki/U-Boot, Tarih: 23.01.2016.

[18] OpenCores, or1k Linux, http://opencores.org/or1k/Linux, Tarih: 24.11.2015.

[19] Quercia, V. ve O'Reilly, T., 1990. Volume Three: X Window System User's Guide, O'Reilly & Associates.