İSTANBUL TEKNİK ÜNİVERSİTESİ

ELEKTRİK-ELEKTRONİK FAKÜLTESİ

HÜCRESEL YAPAY SİNİR AĞININ FPGA ÜZERİNDE YAZILIM VE DONANIM BİRLİKTE TASARIMI

BİTİRME ÖDEVİ

SELMAN ERGÜNAY

040050327

Bölümü: Elektronik ve Haberleşme Mühendisliği

Programı: Elektronik Mühendisliği

Danışmanı: Doç. Dr. Müştak Erhan Yalçın

Mayıs 2010

ÖNSÖZ

Öncelikle bu çalışmamın her aşamasında bana yol gösteren değerli hocam Doç Dr. Müştak Erhan Yalçın'a, projelerim esnasında her zaman yardımcı olan Araş. Gör. Ramazan Yeniçeri'ye, Gömülü Sistem Tasarımı Laboratuvarı kurucu, yönetici ve çalışanlarına, tüm Elektronik ve Haberleşme Mühendisliği Bölümü öğretim üyelerine emeklerinden dolayı şükranlarımı sunarım.

Eğitim hayatım boyunca maddi ve manevi desteklerini benden esirgemeyen sevgili annem, babam, ağabeyim ve ablama da sonsuz sevgi, saygı ve teşekkürü borç bilirim.

Mayıs 2010 Selman Ergünay

İÇİNDEKİLER

KISALTMALAR

BRAM : Block Random Access Memory
BSP : Board Support Package
CNN : Cellular Neural Network
EDK : Embedded Development Kit
FPGA : Field Programmable Gate Array
FS-Boot : First Stage Boot
G/Ç : Giriş / Çıkış
HAL : Hardware Abstraction Layer
HYSA : Hücresel Yapay Sinir Ağı
IP : Intellectual Property
LUT : Look Up Table
LMB : Local Memory Bus
MİB : Merkezi İşlem Birimi (CPU)
MMU : Memory Management Unit
NPE : Nonlinear Processing Element
OPB : On-chip Peripheral Bus
PLB : Processor Local Bus
RAM : Random Access Memory
SoC : System on Chip
XPS : Xilinx Platform Studio

ŞEKİL LİSTESİ

ÖZET

HÜCRESEL YAPAY SİNİR AĞININ FPGA ÜZERİNDE YAZILIM VE DONANIM BİRLİKTE TASARIMI

Yerel bağlantıya sahip ağ mimarilerinin tümdevre üzerinde gerçeklenebilirlikleri ile sinir hücresi temel alınarak kurulan doğrusal olmayan devrelerin yani yapay sinir ağlarının birleştirilmesi fikri ilk olarak 1988'de Chua ve Yang tarafından Hücresel Yapay Sinir Ağları (Cellular Neural Network) adı ile ortaya konmuştur. Çalışmalar doğanın da bu keşfi desteklediğini göstermektedir. Canlılarda görsel bilginin ilk işlendiği yerin retina üzerindeki birden fazla katmandan oluşan hücresel mimarideki sinir ağı olduğu bilinmektedir. Bu tür mimarideki sinir hücresi benzeri doğrusal olmayan devre ya da sistemlerin uzay-zaman dalgalarını üretebildiğinin gözlenmesi ile bu tür ağların doğadakinden farklı uygulamaları üzerinde de araştırmalar sürdürülmüştür. Bu araştırmalar ayrıca hücresel yapay sinir ağlarının analog ve sayısal devre gerçeklemeleri ile desteklenmiştir.

İki boyutlu ve üzerinde engeller bulunduran bir düzlemde bir robotun en kısa yol üzerinden hedefe yönlendirilmesi ile ilgili çalışmalar uzay-zaman dalgaları üretip yayabilen bir ağ modeli kullanarak gerçekleştirilmiştir. Bu hücresel doğrusal olmayan ağ modeline ait öykünücüler (emülatörler) Alanda Programlanabilir Kapı Dizileri (FPGA) ile tasarlanmışlardır.

Daha önce FPGA üzerinde salt donanım olarak tasarlanmış olan öykünücüler bu çalışmada yazılım ve donanımın birlikte tasarımı projesi olarak gerçeklenmişlerdir. Böylece donanımın paralel işlem yeteneğinden kaynaklanan hız kazancı ile gerçek zamanda çalışma yeteneği kaybedilmezken, yazılımın getirdiği algoritma geliştirme kolaylığı ve esneklikten faydalanılmıştır. Mikroişlemci ve veri yoluna bağlı olarak doğrusal olmayan işlem birimleri (NPE) ile seri haberleşme kontrol donanımı aynı FPGA içinde yer aldığı için bu yapılar aynı zamanda tek tümdevre üzerinde sistemlerdir (SoC).

Bu projede kullanılan yazılım ve donananımlar son olarak gömülü Linux tabanlı bir sistemde birleştirilmiştir. Elde edilen nihâi sistemde FPGA içerisinde yer alan mikroişlemci DDR bellekteki gömülü Linux işletim sistemini çalıştırmakta ve NPE donanımlarını kontrol eden yazılımlar bu işletim sistemi üzerinden işleyebilmektedir.

SUMMARY

HARDWARE & SOFTWARE CO-DESIGN OF CELLULAR NEURAL NETWORK ON FPGA

The Cellular Neural Networks (CNNs) which was proposed by Chua and Yang in 1988 are based on realization of locally coupled network architecture on very large scale integrated circuits and nonlinear circuits which emulate neuron structure. New finding on biological systems show that there is strong relationship between this idea and nature. For instance, it is known that the first place of processing visual information is a cellular neural network which has got more than one layer on retina. Observation of spatiotemporal wave generation capability of nonlinear circuits and systems have caused research on different applications of these networks from nature. These works are also supported by analog and digital circuit implementation of artificial cellular neural networks.

Projects about solution of robot navigation and shortest path finding problems on a plane that is two dimensional and have obstacles were realized by using a network structure which generates spatiotemporal waves. These emulators which were designed with cellular nonlinear network model were implemented on FPGA

In this work, the emulators which were implemented as hardware on FPGA is realized as hardware-software co-design project. With this approach, both speed advantage of parallel processing capability of hardware and flexibility advantage of software. Because microprocessor, nonlinear processing elements and serial communication control unit are in the same FPGA, this system is also a System on Chip design project.

Hardware and software projects which are used in these systems are added to a embedded Linux based environment. Microprocessor which is placed in FPGA executes embedded Linux operating system from DDR memory and software programs that controls hardware runs on this operating system.

1. GİRİŞ

Gömülü sistemler belirli sayıda işlemi yapmaya adanmış bilgisayar sistemleri olarak tanımlanabilir. Genel amaçlı bilgisayarların aksine kendisi için tanımlanmış bir ya da birkaç görevi yerine getirirler. Günümüzde tıp elektroniğinden savunma sanayine, kullanıcı elektroniğinden uçak ve otomotiv endüstrisine kadar pek çok alanda gömülü sistemler karşımıza çıkmaktadır. MP3 çalar, cep telefonu, sayısal fotoğraf makinesi gibi kullanıcı elektroniği elemanları, mikrodalga fırın, çamaşır makinesi, televizyon gibi ev makinelerinin elektronik kontrol sistemleri, uçuş kontrol ve füze güdüm sistemleri gibi savunma sanayi ürünleri gömülü sistemlere örnek olarak verilebilir.

ARM, MIPS, PowerPC gibi gömülü sistemlerde kullanılan farklı işlemci mimarileri bulunmaktadır. Sistemin kullanım amacına göre işlemci belirlenir. Çeşitli özelleşmiş uygulamalar için tasarlanmış mikroişlemciler de bulunmaktadır. Sayısal işaret işleme konusunda optimize edilmiş sayısal işaret işlemciler (DSP) uygulamaya özel işlemcilere bir örnektir. FPGA'da kullanmak üzere programlanabilirliği yüksek mikroişlemciler de bulunmaktadır.

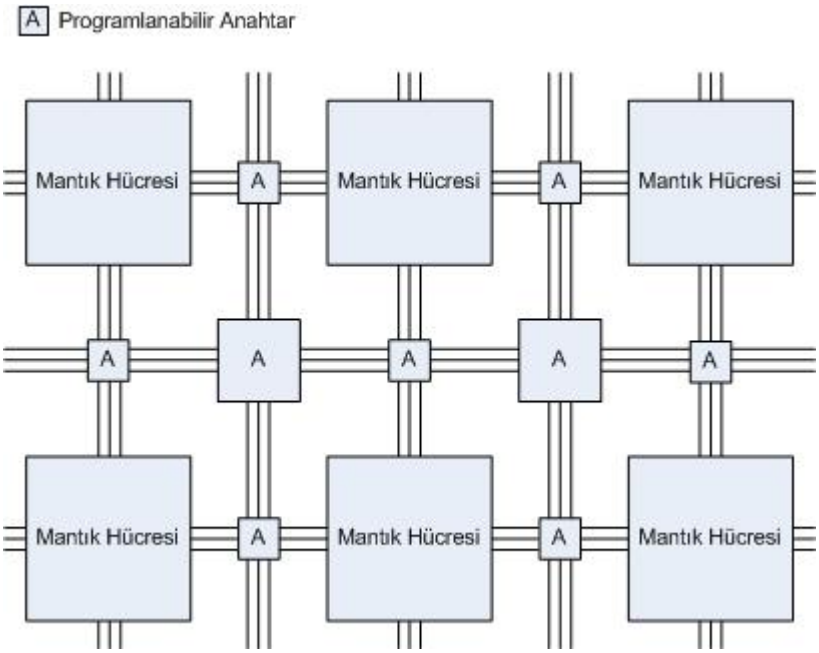
Gömülü sistemler genelde çok sayıda üretildikleri ve sonradan aynı sistem üzerine ekleme yapılmadıkları için en az kaynak kullanmaları üzerine yoğunlaşılır. Maliyeti mümkün olduğunca düşürmek için ihtiyaçları karşılayabilecek en ucuz işlemci, en az bellek ve en düşük depolama alanı gibi en alt seviye sistem kaynağı kullanmaya dikkat edilir. Bu sebeple yazılım ve donanım optimizasyonu önemli bir problem olarak karşımıza çıkmaktadır.

Günümüzdeki karmaşık gömülü sistemler yalnızca yazılım ya da yalnızca donanım ihtiva eden sistemler değil, her ikisinin birlikte tasarlandığı sistemlerdir. Mikroişlemci tabanlı sistemlerin paralel işlemler yapan donanımlar kullanılarak hızlandırılması ileri uygulamalar için kaçınılmaz bir yöntemdir. Bunun yanı sıra işletim sisteminin getirdiği donanım soyutlama katmanı, bellek idaresi, zamanlama yönetimi gibi özellikler ile daha esnek ve gelişmiş sayısal sistemler gittikçe yaygınlık kazanmaktadır.

Alanda Programlanabilir Kapı Dizileri (FPGA) gömülü sistem geliştirmede kullanılan teknolojilerden biridir. Kapladığı alan ve performans açısından diğer yöntemlerden başarısız olsa da geliştirme süresinin kısa olması ve yeniden programlanabilmesi

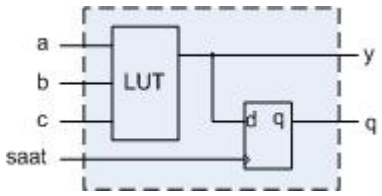
önemli avantajlarıdır. Donanım tanımla dilleri (HDL) ile yazılmış sayısal sistemlerin sentezlenip gerçeklenmesi için genel amaçlı bir bilgisayar yeterli olmaktadır.

FPGA'lar Şekil 1.1 'de görüldüğü gibi programlanabilir anahtarların ve ayarlanabilir mantık hücrelerinin iki boyutlu olarak dizilmesiyle oluşan mantık devreleridir. Bir mantık hücresi basit bir fonksiyonu gerçeklemek üzere yapılandırılabildiği gibi programlanabilir anahtarlar ile mantık hücreleri arasında bağlantılar kurulabilir. Bu şekilde sayısal donanımlar mantık hücreleri ve anahtarların programlanmasıyla gerçeklenir. Devrenin donanım tanımlama dilleri ile tasarımı tamamlandıktan sonra sentezlenmesinin ardından istenen lojik hücre ve anahtar yapılandırmasının yer aldığı veri dizisi FPGA'ya gönderilerek devre gerçeklenmiş olur. [1]



Şekil 1.1: FPGA Matris Yapısı [1]

Şekil 1.1'de görülen mantık hücreleri Şekil 1.2'dekine benzer bir yapıya sahiptir. D tipi flip-flop ile birlikte programlanabilir küçük bir kombinezonsal devre içerir.



Şekil 1.2: Mantık Hücresi Yapısı [1]

Yapılandırılabilir kombinezonsal devre elde etmek için en yaygın kullanılan yöntem LUT (*Look-Up-Table*) yapısıdır. n girişli bir LUT 2^n-1 boyutlu bir bellek elemanı olarak düşünülebilir. Bu belleğin içeriğinin değiştirilmesi yöntemiyle n girişli kombinezonsal fonksiyonlar gerçeklenebilir.

Mikroişlemciler de sayısal devreler oldukları için FPGA üzerinde kullanılabilirler. Böylece yazılım ile kontrol edilebilen sistemlerin FPGA'da uygulanması mümkün hale gelir. Tasarımın ihtiyaçları doğrultusunda FPGA birden fazla mikroişlemciyi de barındırabilir ve bu şekilde çok işlemcili sistemler gerçeklenebilir. Bunun yanı sıra işlemci ile beraber sayısal devreler de aynı FPGA içinde yer alabilir. Tüm sistemin aynı yerde yer almasıyla bağlantılar arası gecikmeler azaldığı için daha hızlı bir sistem elde edilmesinin yanı sıra gürültüye bağışıklığın arttığı da görülür.

Yazılım ve donanımın birlikte tasarlandığı sistemlerde yapı karmaşıklaştıkça yapıyı idare etmek zorlaşır. Mesela aynı anda çalışan programlar arasında kaynak kullanımı ve zamanlamayı düzenlemek önemli bir problemdir. Bu sistemlerde işletim sistemi kullanma ihtiyacı duyulur. Gömülü sistemlerde kullanılan en yaygın işletim sistemlerinden biri de Linux işletim sistemidir.

Bitirme çalışmasında gerçeklenen sistemler için Xilinx tarafından geliştirilmiş olan Spartan 3E1600E FPGA geliştirme kartı kullanılmıştır. Sistemin nihai halinde Microblaze mikroişlemci, hücresel sinir ağı donanımları, Blok RAM ve haberleşme birimleri aynı FPGA içinde yer almaktadır. Mikroişlemci, FPGA geliştirme kartı üzerinde bulunan DDR belleklere yüklenmiş gömülü Linux işletim sistemini çalıştırmakta, işletim sistemi üzerinden çalıştırılan yazılım hücresel sinir ağı donanımları kontrol etmektedir. İşlemci ile kullanıcı arasındaki haberleşme seri port üzerinden sağlanmaktadır.

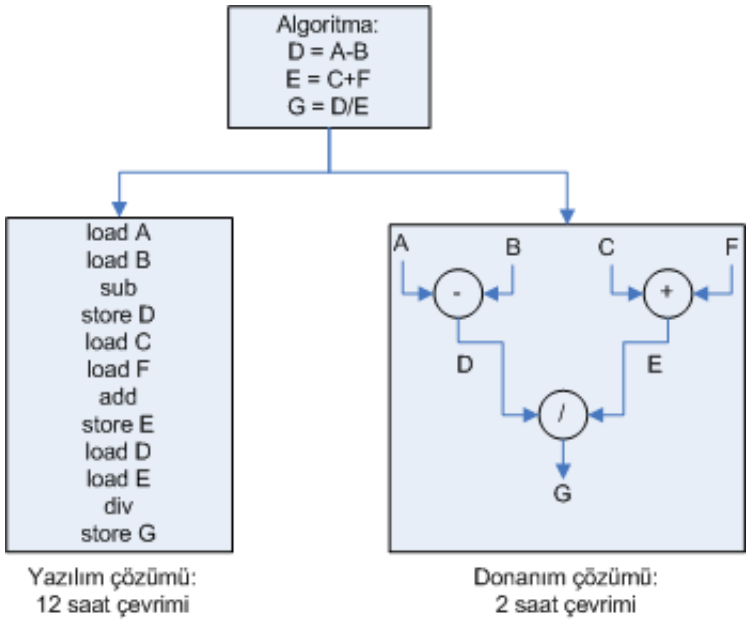
Bu kapsamda yapmış olduğum çalışmaları anlatan bu bitirme ödevi metninin ikinci bölümünde yazılım ve donanımın birlikte tasarımı başlığı altında hücresel sinir ağı donanımlarının gerçeklenmesi anlatılmıştır. Bu başlık altında yapılan iki çalışmadan ilki bir HYSA donanımının yazılım ile kontrolü ile gevşemeli osilatör davranışının elde edilmesidir. İkinci çalışmada ise üç adet NPE (Nonlinear Processing Element, Doğrusal olmayan işlem elemanı) kullanılarak 9*9 boyutlu bir ağ öykünücüsü (emülatörü) elde edilmiştir. Bitirme tezinin üçüncü bölümünde ise işletim sistemi kavramları ile beraber gömülü Linux kullanarak ulaşılan nihai sisteme yer verilmiştir. Bu son sistemde önceki bölümde kullanılan yazılımlar gömülü Linux ortamında çalışarak NPE donanımlarını kontrol etmektedir.

2. YAZILIM VE DONANIMIN BİRLİKTE TASARIMI

Yazılım ve donanımın birlikte tasarımı genelde algoritmik yapısı kurulmuş bir sistemde yazılımı donanım ile hızlandırma anlayışına dayanır. Bilindiği gibi tek çekirdekli merkezi işlem birimine sahip mikroişlemcilerde bir adet komut işaretçisi (instruction pointer) bulunur ve program belleğindeki komutlar saat darbeleri ile sırayla işlenir. C programlama dili gibi orta seviye ve güçlü dillerle bu anlayışa göre algoritmik yapıları gerçeklemek kolaydır. Fakat tek boyutlu işaret işleme, görüntü işleme ve yapay sinir ağları gibi vektör veya matris işlemleri gerektiren konularda yazılımın genel amaçlı işlemciler üzerinde kullanılması gerçek zamanlı sistemlerde ihtiyaçları karşılayamamaktadır.

Bu ihtiyaçlar için sistemin paralel işlemler gerektiren kısımları ayrılarak donanım olarak gerçeklenir. Örnek olarak Şekil 2.1'deki algoritmayı ele alalım. Gerçek zaman gerektiren bir sistemde dışarıdan 4 giriş aldığımızı ve bu girişler için aşağıdaki basit algoritmayı gerçekledikten sonra karmaşık bir algoritmik yapısı olan alt sisteme G girişinin verildiğini varsayalım. Sistemin bu bölümünün yazılımla gerçeklemesi görüldüğü gibi 12 saat çevrimi alırken donanım kullanımı 2 saat darbesi ile işlemini tamamlamış ve bu alt sistemi 6 kat hızlandırmıştır [2]. G girişini alan alt sistemin donanımla gerçeklenmesi ise önemli kazançlar sağlamıyorsa geliştirme süresini, dolayısıyla maliyeti artıracağı için tercih edilmez. Neticede sistemin yazılım ve donanım ayrıştırması önemli bir problem olarak karşımıza çıkmaktadır [3].

Günümüzde kullanılan pek çok sistem bu ayrıklaştırma anlayışına dayanmaktadır. Yazılım getirdiği algoritmik yapıları geliştirme kolaylığı sistemlerin geliştirme süresini dolayısıyla maliyetini azaltmakta, bunun yanı sıra donanımların paralel işlem yapabilme özelliği ile sistemin hızı artırılmaktadır.



Şekil 2.1: Yazılım ve Donanım Çözümlerinin Karşılaştırması [2]

Bu başlık altında gerçeklenmiş olan projelerde de algoritmik olarak sistem tasarımı yapılmış, ardından daha önceki çalışmalarda tasarlanmış olan NPE donanımları yazılım hızlandırma birimleri olarak eklenmiştir. Böylece salt yazılımla gerçekleştirmesi halinde çok daha fazla zaman alacak bir sistem, donanımın paralel işlem yapma özelliği kullanılarak hızlandırılmış, bunun yanı sıra yazılımın esnekliğinden faydalanılmıştır.

2.1. SoC

SoC (System On Chip; Tek Tümdevre Üzerinde Sistem) tasarımı elektronik bir sistemin tüm bileşenlerinin tek bir tümdevre içerisinde birleştirme yaklaşımına dayanır. SoC tasarımlar, düşük güç tüketimi, daha az maliyet, daha yüksek hız gibi özellikleri beraberinde getirir. Bağlantılar tümdevre içindeki yollar ile sağlandığı için bağlantılar arası gecikme süresi azalırken farklı tümdevreler ile tasarlanan sistemlerdeki port gecikmeleri kalkmış olur. Tümdevre sayısının azalması yerden kazancı da beraberinde getirir [4].

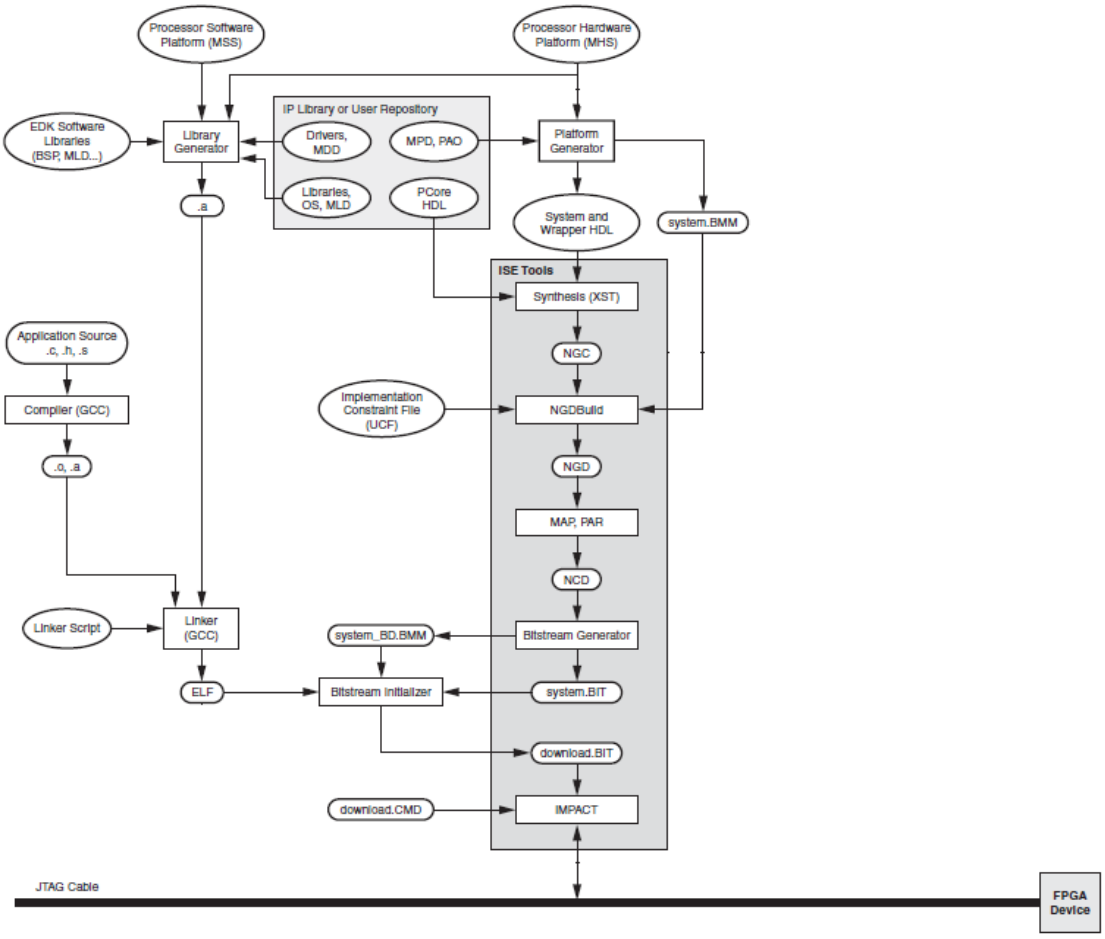
Tipik bir SoC tümdevresi içindeki işlemlerin yönetilmesi için genelde bir ya da birden fazla mikroişlemci ihtiva eder. Yapısındaki bellek birimlerinde gömülü yazılımı saklar ve mikroişlemci bu yazılımı çalıştırır. Tümdevre içindeki sistem bileşenlerini bağlamak için dâhili veri aktarımında kullanmak üzere dâhili veri yolları barındırır [4].

Bu bölümde tasarlanan sistemler birer SoC tasarımlarıdır. Mikroişlemci, NPE birimleri, bellek ve seri haberleşme birimi aynı FPGA üzerinde bulunmaktadır. Dâhili

veri yolu olarak OPB (On Chip Peripheral Bus) veri yolu kullanılmakta ve çalıştırılacak yazılım bellek içine gömülmektedir.

2.2. EDK Ortamı

Xilinx ürettiği FPGA'lar üzerinde mikroişlemci tabanlı sayısal sistemler geliştirmek üzere EDK (Embedded Development Kit) ortamını sunmaktadır. EDK sistemin adreslenmesi, çevre birimlerinin ve FPGA donanımlarının mikroişlemci yollarına bağlanması, iletişim protokollerinin yazılması gibi işlerle uğraşmak yerine yalnızca donanım ve yazılımın tasarımına odaklanmayı sağlar [5]. Şekil 2.2 'de mikroişlemcili bir sistemin tasarım aşamaları görülmektedir. Bu tasarımları ayrı ortamlarda gerçeklemek karmaşık ve zahmetli bir iş olduğu için tüm araçların bir arada bulunduğu ve uyumla çalıştığı böyle geliştirme ortamları ile çalışmak proje süresini önemli ölçüde azaltır.



Şekil 2.2: EDK Ortamında Sistem Geliştirme Aşamaları [5]

Bunun için seçilen FPGA kartına uygun olarak kart üzerindeki çevre birimlerinden ve Xilinx tarafından üretilmiş hazır IP'lerden istenenlerin yer aldığı bir taban sistemi BSB (Base System Builder) ile oluşturur. Bu noktadan itibaren geliştiricinin yapması gereken donanımını yazıp bir yola bağlamak ve yazılımı bir bellek alanına yüklemek üzere oluşturmaktır.

EDK geliştirme ortamında "soft-core" (Microblaze ve Picoblaze gibi) veya "hard-core" (PowerPC) mikroişlemci merkezli donanım projesine kart üzerindeki çevre ve G/Ç birimleri eklenebildiği gibi Xilinx tarafından geliştirilmiş veya kullanıcı tarafından yazılmış donanımlar da eklenebilmektedir. Böylece hem mikroişlemci birimi hem de yazılan donanımlar aynı FPGA üzerinde yer alabilmekte ve aralarındaki iletişim yollar sayesinde sağlanabilmektedir [5].

EDK donanım projesinin yapılandırılmasında XPS (Xilinx Platform Studio) programını kullanmaktadır. Bu program ile mikroişlemcinin yollarına çevre birimleri bağlanmakta ve bu birimler mikroişlemci için adreslenmektedir. Sistemin adres haritasının üretilmesinin ardından EDK, ISE araçlarını kullanarak tüm donanımları sentezler. Bu işlemle birlikte yazılım tasarımına geçmeden önce "Generate Libraries and BSPs" programı çalıştırılarak donanımlara erişmek için kullanılacak kütüphaneler üretilir [5].

Diğer yandan SDK programı yazılım geliştirmesi için kullanılır. Üretilmiş olan kütüphanelerden faydalanarak yazılan C yazılımı hedef mikroişlemci için derlenip bağlandıktan sonra (compile & link) hangi bellek alanına yükleneceği seçilip daha önce ISE araçlarıyla elde edilmiş olan bit dizisi yazılım ile güncellenerek sistem tamamlanmış olur. Güncellenen bit dizisi ile FPGA programlandığında mikroişlemcili sistem çalışmaya başlar. Mikroişlemcinin reset vektörünün atandığı bellekteki yazılım çalışarak hedeflenen işlemi gerçekleştirir [5].

Yazılım ve donanımın birlikte tasarımı başlığı altında yapılan çalışmalarda EDK ortamından faydalanılmıştır. İlk çalışma olan "Bir Relaksasyon Osilatörünün Yazılım ve Donanımın Birlikte Tasarımı ile Gerçeklenmesi" başlıklı çalışmada MicroBlaze işlemci tabanlı bir sisteme bir NPE koyularak osilatör öykünmesi gözlenmiş, ikinci çalışma olan "Doğrusal Olmayan Aktif Dalga Üretecinin Yazılım ve Donanım Birlikte Tasarımı" adlı çalışmada ise üç adet NPE kullanılarak 9*9 boyutunda bir ağın benzetimi yapılmıştır.

2.3. MicroBlaze Mikroişlemcisi

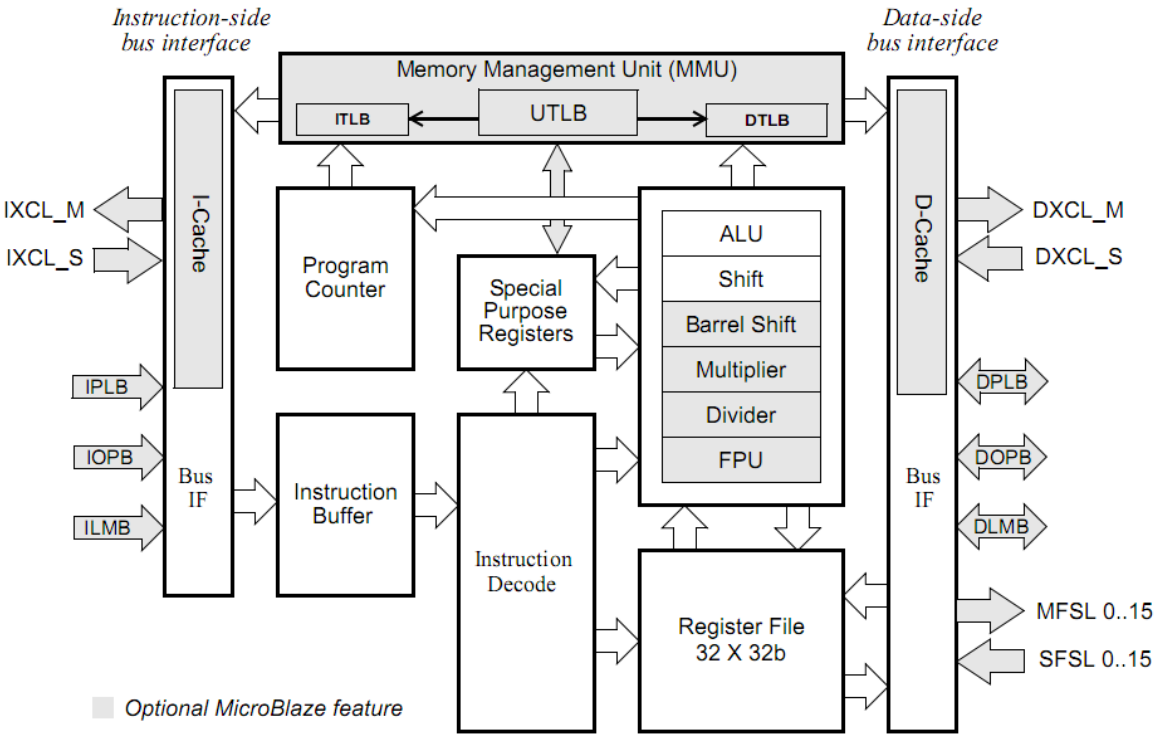
Tasarımda, mikroişlemci olarak Xilinx tarafından geliştirilen 32-bitlik ve RISC mimarisine sahip MicroBlaze işlemcisi kullanılmıştır. Bu işlemci hem FPGA içerisindeki blok RAM'leri hem de harici belleği desteklemektedir. Kontrol edilecek blok RAM boyutu sistem oluşturulurken belirlenir ve kullanılan FPGA'da en fazla 64 KB olabilmektedir. Bu durum yalnızca blok RAM'den çalıştırılacak yazılım için bir kısıtlama olarak görülmektedir. Büyük yazılımlar için çözüm harici DDR bellekleri kullanmaktır. Yazılımın program ve veri parçaları DDR belleğe yerleştirilirken blok RAM'e önyükleyici gömülür. Sistem başlatıldığında MicroBlaze önyükleyiciyi çalıştırır, akabinde ise DDR bellekteki yazılım çalışmaya başlar. Bu proje kapsamında oluşturulan bu sistemlerde ise yazılım boyutları küçük olduğundan BRAM kullanılmıştır. Dolayısıyla önyükleyiciye ihtiyaç duyulmamıştır.

MicroBlaze Harvard bellek mimarisine sahiptir; program ve veri erişimi ayrı bellek alanlarından sağlanır. Her bir adres alanı 32 bit ile adreslenir, bu da program ve veri için 4'er GB'lık bellek alanlarına erişim yeteneği anlamına gelir [6].

MicroBlaze FPGA için optimize edilmiş bir mikroişlemcidir ve birçok özelliği sistem oluşturulurken EDK ortamında belirlenebilmektedir. 32 bitlik 32 adet genel amaçlı kaydedicileri ve 32 bit adres yolu gibi özellikleri sabit iken, iş hattı (pipeline) derinliği, veri yolu sayısı ve türleri, kayan noktalı sayı birimi (FPU), bellek idare birimi (MMU) gibi özellikleri tasarım ihtiyaçlarına göre seçilebilmektedir [6].

Tasarlanan sistemlerde MicroBlaze için çalışma frekansı ile işlemci-veri yolu frekansı 50 MHz seçilmiştir. Veri ve program belleği olarak kullanılacak olan BRAM boyutu 64 KB, RS232 biriminin hızı 115200 olarak belirlenmiştir. Standart giriş ve standart çıkış olarak RS232 seçilmiş, böylece yazılımda kodlanacak olan yazdırma komutlarının verileri seri port üzerine aktarmaları sağlanmıştır. Önyükleme belleği olarak BRAM seçimi ise sistem başlatıldığında burada yer alan yazılımın çalışmasına imkân tanımıştır.

Şekil 2.3'te MicroBlaze işlemcisinin mimari yapısı yer almaktadır. Bu mimaride bellek idare birimi ve çarpma, bölme, kayan noktalı sayı birimleri gibi bileşenlerin sistem yapılandırması esnasında isteğe göre seçilebildiği görülmektedir.



Şekil 2.3: MicroBlaze Mimarisi [6]

2.4. Veri Yolları

Bu sistemde seri porta ve tasarlanan osilatör öykünücüye erişmek için Xilinx'in 128 bit genişlikli PLB (Processor Local Bus) veri yolu kullanılmıştır. Bu yola veri aktarımı ve bu yoldan veri alımı MicroBlaze işlemcisinin PLB denetim birimi tarafından kontrol edilmektedir [6]. PLB'ye bağlanan seri port, MicroBlaze için hem standart giriş, hem de standart çıkış olacak şekilde yapılandırılmıştır. Böylece seri haberleşme programı kullanılarak elde edilen verilerin ana bilgisayar üzerinde görüntülenmesi sağlanmıştır.

Microblaze işlemcisi, blok RAM'e erişimde program ve veri bölümleri için ayrı ayrı LMB (Local Memory Bus) veri yolları kullanır. Adı geçen yollar Xilinx tarafından tasarlanmış DLMB (Data side LMB) ve ILMB (Instruction side LMB) denetleyicileri tarafından kontrol edilen 32 bitlik yollardır [7]. Tasarımlarda MicroBlaze'in reset vektörü blok RAM'in başlangıcını göstermektedir. Böylece blok RAM'e koyulacak yazılım, sistem başladığında MicroBlaze tarafından ILMB ve DLMB yolları üzerinden başlatılacaktır.

Tasarlanan sistemin çalışması, yazılımın parametreleri ve girişleri öykünücü donanımda yer alan kaydedicilere PLB üzerinden yazıp başlama işareti göndererek, üretilen çıkışı DLMB yoluyla belleğe, PLB yoluyla seri porta göndermesine dayalıdır.

2.5. Bir Relaksasyon Osilatörünün Yazılım ve Donanımın Birlikte Tasarımı

2.5.1. Sisteme Genel Bakış

Yazılım ve donanımın birlikte tasarımı başlığı altında yapılan bu ilk çalışmada daha önceki çalışmalarda [8] geliştirilmiş olan doğrusal olmayan işlem birimi (NPE), Microblaze'in PLB veri yolu üzerinden haberleşebileceği biçimde yeniden düzenlenmiş ve gevşemeli osilatör öykünücüye dönüştürülmüştür.

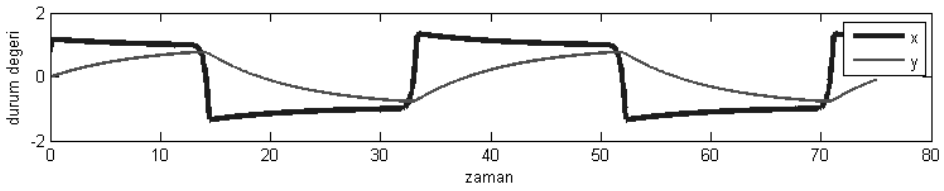
Yalçın'ın 2008'de önerdiği [9] uzay-zaman dalgaları yayan hücresel doğrusal olmayan ağ modeli Denklem (2.1)'de verilmektedir.

$$\begin{aligned} \dot{x}_{i,j} &= \alpha x_{i,j} + \beta y_{i,j} + g(x_{i,j}) + I_{i,j} + u_{i,j}, \\ \dot{y}_{i,j} &= \epsilon x_{i,j} + \sigma y_{i,j}. \end{aligned} \qquad \textbf{(2.1)}$$

Hücre modelinde belirtilen g(.) fonksiyonu doğrusal olmayan terimdir ve Denklem (2.2)'de belirtildiği gibi parça parça doğrusal bir ifadeye karşılık gelmektedir.

$$g\left(x_{i,j}\right) = \begin{cases} \mu \cdot (x_{i,j} - \lambda) & \text{if } x_{i,j} > \lambda; \\ 0 & \text{if } |x_{i,j}| \leq \lambda; \\ \mu \cdot (x_{i,j} + \lambda) & \text{if } x_{i,j} < -\lambda; \end{cases} \qquad \textbf{(2.2)}$$

Ağ üzerindeki esasen gevşemeli osilatör davranışı sergileyen, ikinci mertebeden doğrusal olmayan her bir dinamik hücrenin zaman içerisinde durum değişkenlerinin değişimi Şekil 2.4'de gösterildiği gibidir.

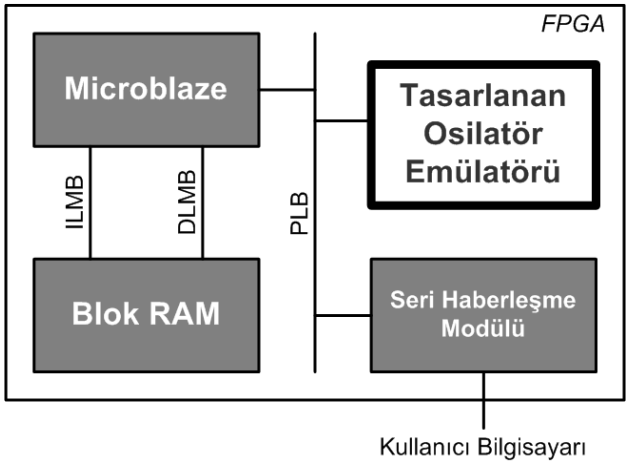


Şekil 2.4: Bir hücrenin salınımı

Öykünücü içerisinde, Denklem (2.1)'i ileri Euler metodu ile ayrıklaştırılmış olarak çözen NPE'nin tüm parametre ve veri girişlerini süren 13 adet 32 bitlik tutucu mevcuttur ve MicroBlaze bu tutuculara değer atayabilmektedir. Öykünücünün durum denklemlerinin ikisini de hesaplaması yaklaşık 180 saat çevrimi zaman almaktadır. Öykünücünün tutucularına parametreler başlangıçta yazılım tarafından yüklenmektedir. Öykünüm boyunca, ileri Euler ayrıklaştırması sebebiyle her yineleme (iterasyon) sonucu yeniden öykünücüye giriş olarak verilmekte ve adım adım Denklem (2.1)'deki diferansiyel denklemler çözülmektedir. Bütün bu işlemler tamamen yazılım kontrolünde yürütülmektedir.

2.5.2. Donanım Projesi

Blok yapısı Şekil 2.5'te görülen sistemin donanım projesinde 1 adet NPE kullanılmıştır. Microblaze işlemcisine PLB yolu üzerinden seri haberleşme modülünün ve NPE'nin, LMB yolu üzerinden de Blok RAM'in bağlı olduğu bu sistem, seri port ile kullanıcı bilgisayarına bağlıdır.



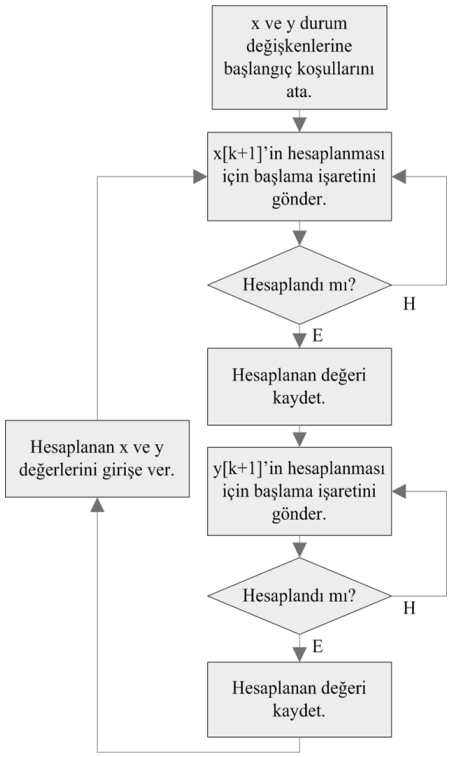
Şekil 2.5: Birinci sistemin blok yapısı

NPE donanımı EDK yardımıyla sistemde kullanılabilir hale getirilmiştir. EDK araçları kullanılarak PLB ile haberleşme protokollerini sağlayan ve 13 adet 32 bitlik yazılım tarafından erişilebilen kaydedici (register) içeren bir ara devre oluşturulmuş ve bu ara devre ile NPE arasında gerekli bağlantılar kurulmuştur. Bu bağlantılar ara devredeki kaydedicilerin NPE giriş ve çıkışları ile olan bağlantılardır. Böylece yazılım tarafından kaydedicilere yapılan atamalar ile NPE girişleri ve parametreleri belirlenmekte, NPE'nin çıkışlarının kaydedicileri biçimlendirmesi ile de veri PLB yolu üzerinden işlemciye alınabilmektedir.

EDK ortamında donanım projesi hazırlandıktan sonra EDK araçları ile donanım elemanları adreslenir ve donanımı idare etmede kullanılacak olan kütüphaneler ile kart destek paketi (BSP) üretilir. Böylelikle elde edilen başlık dosyaları ve kütüphaneler donanımlara erişmede kullanılır. NPE'lerde bulunan kaydedicilerin adresleri ve bu kaydedicilere yazmak ve bunlardan veri okumak için gerekli olan fonksiyonlar da yazılımda kullanmak üzere elde edilmiş olur. Ardından ISE araçları ile donanım projesi sentezlenerek yazılım aşamasına geçilir.

2.5.3. Yazılım Projesi

Sistemin adreslenip yazılım kütüphaneleri ve kart destek paketinin (BSP) oluşturulmasıyla yazılım donanımı kontrol edebilir hale gelir. EDK ortamının bir bileşeni olan SDK (Software Development Kit) üzerinde yazılım tasarımı yapılmıştır. Bu yazılım Şekil 2.6 'daki algoritmanın gerçeklemesidir.



Şekil 2.6: Yazılımın algoritmik akış diyagramı

İlk aşamada NPE'lerin karakterini belirleyen parametreler ile başlangıç koşulları yüklenir. Birinci hesaplamanın başlaması için ilgili kaydedicinin ilgili biti üzerinden yükselen kenar gönderilerek hesaplamanın tamamlanması beklenir. Hesaplandı işareti görüldüğünde NPE'nin çıkışının bağlı olduğu kaydedicideki veri yazılımdaki x değişkeninin değeri olarak atanır. Hemen akabinde y hesaplaması başlatılır. Bu hesaplama da tamamlandığında güncellenen y değeri ile birlikte x değişkeni bir sonraki hesaplamada kullanılmak üzere NPE girişlerine yüklenir.

Bu şekilde hesaplamalar döngü içinde tekrarlanırken x ve y değişkenlerindeki değişimler kullanıcı bilgisayarı tarafından gözlemlenmek üzere seri porta gönderilir. Seri port ile haberleşme sistemi fazlaca yavaşlattığı için birkaç hesaplama değerinden birinin gönderilmesi tercih edilmiştir.

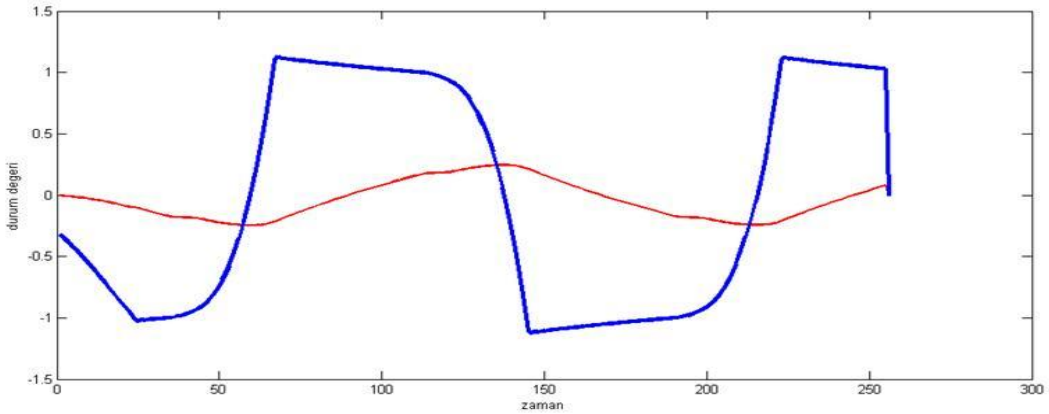
2.5.4. Sonuç

Seri porttan gelen verilerin MATLAB ile alınması ve işlenip çizdirilmesiyle Şekil 2.7'deki grafik elde edilmiştir. Şekilde de görüldüğü üzere istenen osilasyon gerçekleşmiştir. Sonuç olarak Microblaze tarafından kontrol edilen ve tek NPE'den oluşan sistem gevşemeli osilatör olarak çalışmıştır.

Hızlı değişim gösteren eğri x durum değişkenine, yavaş değişen eğri y durum değişkenine aittir. Bu osilatörlerin bir araya getirilmesi ile oluşan ağ üzerinde, komşu hücrelerin birbirine olan etkisi ile osilasyonlar arasında faz farkı meydana gelmektedir. Bu faz farkı neticesinde ağın oluşturduğu iki boyutlu uzayda yayılan dalgalar gözlenebilmektedir.

Yazılım ve donanımın birlikte tasarımı ile uzay-zaman dalgalarını üretecek bu osilatör ağı, bellek miktarının izin verdiği ölçüde, istenen boyuta büyütülebilir. Ayrıca yazılım ile ağ dinamiği, öykünücü donanım dinamiği sabit kalmak üzere istendiği gibi değiştirilebilir. Bu özellik zaman içerisinde değişen sistem yaratmak, dolayısı ile uyarlanabilir algoritmaları da gerçeklemek için kullanılabilir.

Benzer çalışmalar, kullanılan kart üzerindeki DDR bellek tümdevresi ve daha yüksek merteben dinamik sistemlerin öykünümünü sağlayacak donanımlar ile de tasarlanabilir. Bu tür tasarımlarla, bilgisayar ortamında benzetimi uzun zaman alan dinamik sistemler incelenebilir.

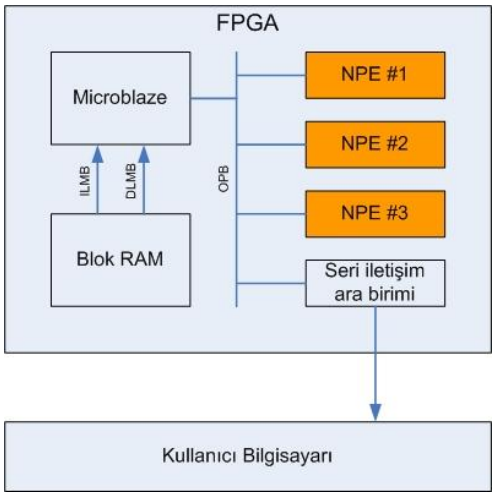


Şekil 2.7: Sistemin salınımı

2.6. Doğrusal Olmayan Aktif Dalga Üretecinin Yazılım ve Donanım Birlikte Tasarımı

2.6.1. Sisteme Genel Bakış

Yazılım ve donanımın birlikte tasarımı olarak gerçeklenen ikinci projede 3 adet NPE kullanılarak 9*9 boyutlu bir ağın emülasyonu amaçlanmıştır. Hazırlanan NPE donanımları OPB veri yoluna bağlanarak Şekil 2.8'deki sistem kurulmuştur. Yazılım ile NPE donanımlarına hesaplamalar yaptırılmakta ve hesaplama sonuçları belleğe kaydedilmektedir.



Şekil 2.8: İkinci sistemin genel yapısı

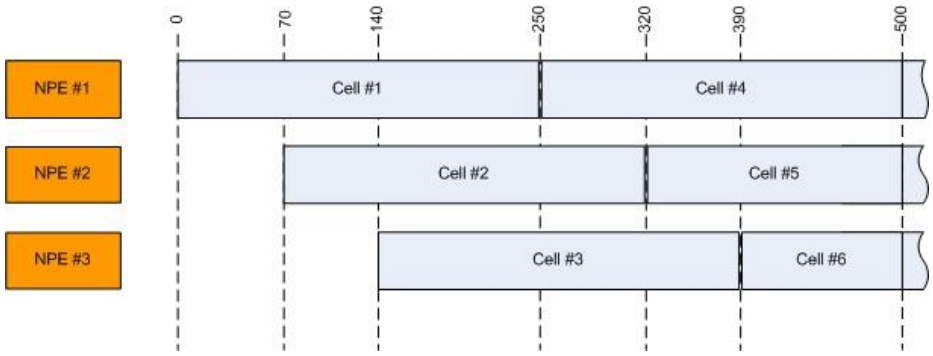
Tasarlanan sistem ayrık modele oturan her işlemi yürütebilir. Farklı işlemler için yapılması gereken NPE parametrelerinin belirlenmesi ve istenen işe uygun yazılımın yazılmasıdır.

2.6.2. Donanım Tasarımı

İlk yazılım ve donanımın birlikte tasarımı projesinde yaptığımızdan farklı olarak 3 adet NPE, EDK ortamında PLB yoluna bağlanmıştır. NPE sayısı yazılımda kullanılacak iş hattı (pipeline) yönteminin zamanlama ihtiyacı doğrultusunda belirlenmiştir. NPE donanımı tam bir hesaplama çevrimini ortalama 180 saat darbesiyle tamamlamaktadır. Microblaze tarafından kontrol edilen bu sistemlerde ise kaydedicilere yapılan atamalar ve bunlardan değer okumalar ile yazılım içindeki karşılaştırmalar sisteme ek bir hesaplama yükü getirmekte ve sistemi yavaşlatmaktadır.

Önceki çalışmadaki sistem kullanılarak yazılım kullanımının kaç saat darbesine mal olduğu ölçülmüştür. Bunun için sisteme fazla sayıda hesaplama yaptırılmış, hesaplamaların başlama ve bitiminde seri porta bir işaret gönderilmesi kodlanmış ve

gelen işaretler arasında geçen süre MATLAB ortamında zaman fonksiyonları yardımıyla tespit edilmiştir. Bu hesaplamalar sonucunda yazılımın sistemi birim hesaplama için yaklaşık 70 saat darbesi yavaşlattığı görülmüştür. Elde edilen verilere göre iş hattında çalışacak bu sistem için 3 adet NPE kullanılması uygundur. Zamanlama diyagramı Şekil 2.9 'da görülmektedir.



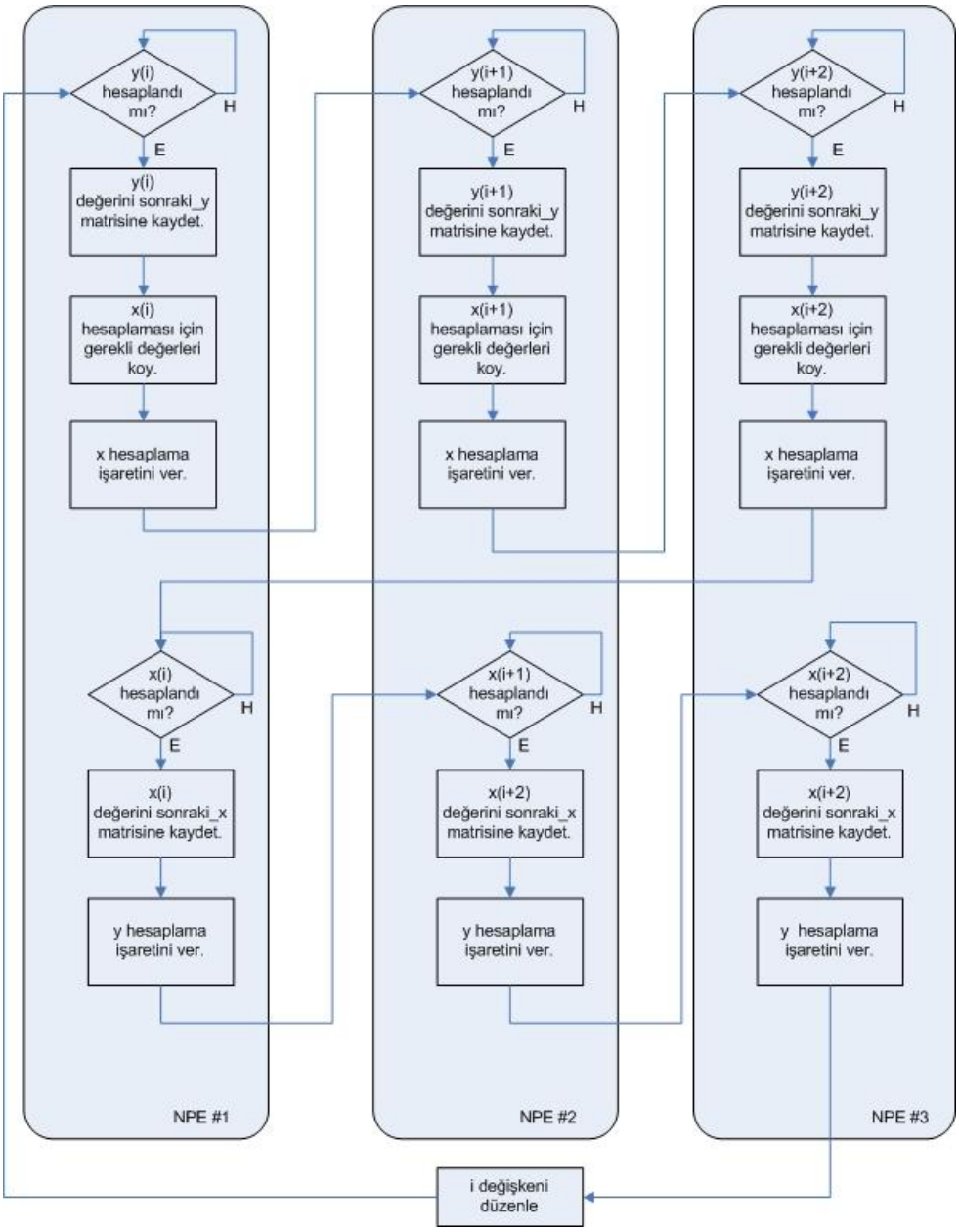
Şekil 2.9: Zamanlama diyagramı

2.6.3. Yazılım Tasarımı

Bu sistem için tasarlanan yazılımda 9*9 boyutundaki ağın x ve y değişkenlerinin şimdiki ve sonraki değerlerini tutmak için toplam dört adet 9*9 matris oluşturulmuştur. Böylelikle elde edilmiş olan ağ öykünücüsündeki hücrelerin x ve y değişkenlerinin şimdiki değerlerini tutan matristen alınan değerlere göre sonraki değer matrisi güncellenmektedir.

Ağda bulunan her bir elemanın (matriste her bir hücreye karşılık gelir) şimdiki değeri ile komşularının şimdiki değerleri NPE hesaplama birimlerinden birine sokularak hesaplanan değer ilgili hücrenin sonraki değer matrisindeki yerine yazılır. Ağın tüm elemanlarının hesaplanmasının ardından şimdiki değer ve sonraki değer matrislerinin işaretçileri değiştirilerek bir sonraki hesaplama için şimdiki değerler belirlenmiş olur.

Bu sistem için oluşturulan yazılımda iş hattı (pipeline) yöntemi ile sistemin yazılımdan kaynaklanan yavaşlamasının azaltılması planlanmıştır. Şekil 2.10'da yazılımın ayrıntılı algoritmik akış diyagramı görülmektedir.



Şekil 2.10: İkinci yazılımın ayrıntılı algoritmik akış diyagramı

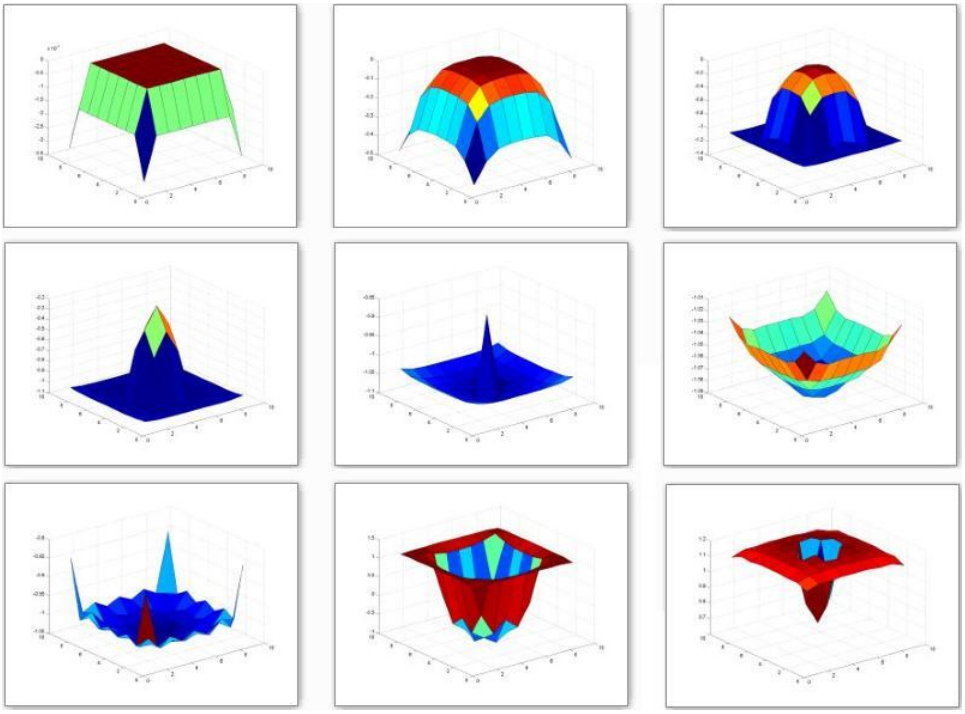
Microblaze tarafından kontrol edilebilen 3 NPE iş hattının aşamalarını oluşturmaktadır. Başlangıçta 1. NPE'ye başlama işareti yazılım tarafından erişilebilen kaydediciler üzerinde değişiklik yapılarak verilmekte, ardından 1. NPE'nin cevabını beklemeden 2. NPE'ye ve peşi sıra 3. NPE'ye başlama işaretleri verilmektedir. Bu işlemler esnasında hesaplamalarını yapmış ve çıkışı üretmiş olan 1. NPE'nin çıkış değeri okunup kaydedilir ve 1. NPE'nin sonraki hesaplamaya başlama işareti verilir.

Aynı işlem diğer iki NPE için de yapıldıktan sonra bir döngü tamamlanmış olur ve bundan sonraki döngü basamakları da aynı şekilde tüm ağ için değerler

hesaplanıncaya kadar gerçekleştirilir. Ağın tamamının hesaplaması yapılıp sonraki değer matrisleri hazır hale geldiğinde sonraki değer ve şimdiki değer matrislerinin işaretçileri yer değiştirir ve böylece önceki adımın sonraki değer matrisi artık şimdiki değer matrisi olmuş olur.

2.6.4. Sonuç

Ayrıntılı akış diyagramı Şekil 2.10'da görülen yazılımın çalışması esnasında tüm ağın bir kez hesaplamasının ardından elde edilen x ve y değerleri standart çıkış olan seri porta gönderilebilir. Bu çalışmada her 3 hesaplamadan biri gönderilerek osilasyonun daha hızlı gözlemlenmesi sağlanmıştır.



Şekil 2.11: İkinci sistemin salınımı

Seri porttan alınan değerler MATLAB ile işlenerek çizdirilmiş ve ağın Şekil 2.11 'deki gibi salınım yaptığı görülmüştür. Sonuç olarak FPGA içinde bulunan Microblaze mikroişlemci, PLB ve LMB kontrol birimleri, seri haberleşme arabirimi ile NPE donanımlarından oluşan sistem, yazılım vasıtasıyla kontrol edilerek 9*9 ağın emülasyonunu yapma ve hesaplama sonuçlarını seri port üzerinden kullanıcı bilgisayarına gönderme işlemlerini başarıyla tamamlamıştır.

Sistemin donanım yapısını değiştirmeden yalnızca yazılımını değiştirerek farklı boyutlardaki ağların emülasyonunu yapmak mümkün hale gelmiştir. NPE parametrelerinin değişmesiyle de ağın davranışı üzerinde değişiklik yapmak mümkündür.

Ağ boyutunun artması Blok RAM boyutu ile sınırlıdır. Daha önce belirttiğimiz gibi Microblaze program ve veri için 4'er GB adresleme yeteneğine sahiptir. Blok RAM'e bir önyükleyici yerleştirilmesi ile DDR bellekler kullanılarak çok daha büyük boyutlu ağlar emüle edilebilir.

3. HYSA DONANIMLARININ GÖMÜLÜ LINUX ÜZERİNDEN KONTROLÜ

Projenin şimdiye kadar olan kısmında arada işletim sistemi olmadan, yazılımın doğrudan donanımı kontrol ettiği sistemler kurulmuştur. Bu bölümde Xilinx Spartan 3E1600E geliştirme ortamı üzerinde Microblaze işlemcinin Linux işletim sistemini çalıştırdığı ve işletim sistemi ortamında iken önceki projelerde hazırlanmış yazılımlar ile HYSA donanımlarının kontrol edildiği nihâi sistem anlatılacaktır.

İlk etapta gömülü sistemlerde işletim sistemi kullanımına dair açıklamanın ardından Linux işletim sistemi ile ilgili genel bilgi verilecek, sonra projede kullanılan ortamın kurulması anlatılacak ve son olarak da gerçeklenen sistem tanıtılarak çalışma biçimine yer verilecektir.

İşletim sistemi donanımları kontrol etmek için kullanılan bir programdır. Temel olarak iki iş yaptığından söz edilebilir. Birincisi bilgisayar sisteminin G/Ç birimleri, işlemci, bellek ve disk alanı gibi donanım ve yazılım kaynaklarını kullanıcı ve uygulama isteklerine göre idare eder. İkinci temel amacı ise yazılım uygulamalarının donanımın ayrıntılı özelliklerini bilmeden kullanmasını sağlamak, yani donanım ve yazılım arasında bir ara yüz vazifesi görmektir [10].

Gömülü sistemlerde işletim sistemi kullanımı sonradan yaygınlık kazanmıştır. Daha öncesinde çoklu görev yönetme kabiliyeti ve kullanıcı ile etkileşimi olmayan veya en az düzeyde olan, donanımı ise doğrudan süren yazılımlar kullanılmaktaydı. Fakat zamanla karmaşıklaşan gömülü sistemlerden beklenen özelliklerin artmasıyla işletimi sistemleri yaygınlık kazanmaya başlamıştır. Bu beklentiler çoklu görev yönetimi, süreç ve bellek idaresi, süreçler arası iletişim gibi özellikler içermesi istenen sistemlerde gömülü işletim sistemi kullanmayı zorunlu kılmıştır.

3.1. Linux İşletim Sistemi

Linux 1991 yılında Linus Torvalds tarafından geliştirilmiş bir işletim sistemidir. Linux çekirdeği hem kişisel hem de ticari kullanım için ücretisiz bir yazılımdır. Açık kaynak kodlu olduğu ve dünyada pek çok kişi tarafından geliştirildiği için güvenilirliği yüksek, sağlam ve güçlü bir işletim sistemidir. 1996 yılında ilk defa gömülü sistemlerde kullanılmaya başlanan Linux, 1999'dan itibaren hızla yaygınlık kazanmaya başlamıştır [11]. Günümüzde işletim sistemi kullanılan gömülü sistemlerin önemli bir kısmında Linux tercih edilmektedir.

Gömülü sistemlerde Linux işletim sisteminin tercih edilme sebeplerinden bazıları aşağıdaki gibi sıralanabilir:

- Linux sisteminde çalışan uygulamalar platformdan bağımsızdır. Linux çekirdeği hemen hemen her işlemci için derlenebildiği gibi farklı işlemci mimarilerinde aynı yazılımlar Linux üzerinde çalışabilir.
- Pek çok donanım üreticisi Linux desteği vermekte, dolayısıyla donanımların çoğu sorunsuz olarak çalıştırılabilmektedir.
- Açık kaynak kodlu olması nedeniyle dünyanın dört bir yanındaki geliştiriciler Linux'a ve Linux tabanlı uygulamalara katkıda bulundukları için Linux hızla daha işlevli ve daha kararlı hale gelen gelişmeye açık bir işletim sistemidir. Pek çok kişi bu işlerle uğraştığı için karşılaşılan problemlerin internet siteleri ve e-posta grupları vasıtasıyla çözülme ihtimali yüksektir. Ayrıca açık kaynak kodlu olması güvenliği de beraberinde getirmekte, güvenlik gerektiren stratejik uygulamalarda tercih edilmesine sebep olmaktadır.
- Linux'un tamamen ücretsiz olması ve Linux üzerinde çalışan açık kaynak kodlu ücretisiz pek çok araç bulunması sistem geliştirme maliyetlerini önemli ölçüde düşürmektedir [11].

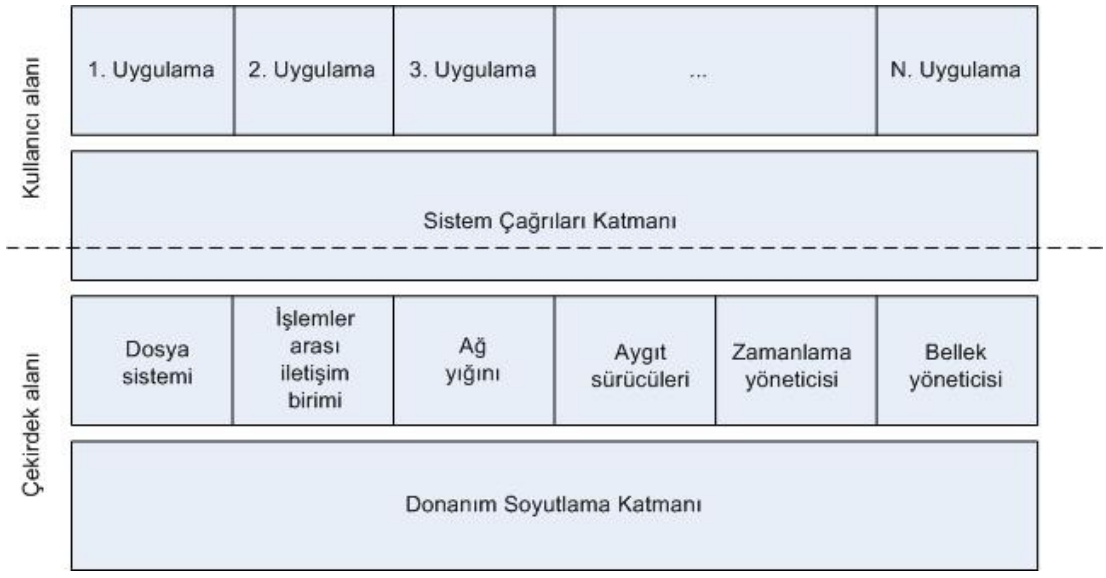
Gömülü Linux işletim sistemleri masa üstü sistemlerden farklı olarak yalnızca sistem için ihtiyaç duyulan özellik ve sürücüleri ihtiva eden küçük sistemlerdir. 16 bit işlemcilerde ve birkaç yüz KB belleğe sahip sistemlerde dahi çalışabilecek çekirdekler mevcuttur. Bu sistemlerde Linux'ta kullanılan büyük ve yetenekli kütüphaneler yerlerini daha az yer kaplayan ve daha basit kütüphanelere bırakmaktadır. Örneğin X pencere sistemi masaüstü sistemlerde çok yaygın iken gömülü sistemlerde çok daha küçük bir türevi nanoX isteğe bağlı olarak kullanılmaktadır [11].

3.1.1. Linux Çekirdeği

Çekirdek, işletim sisteminin ana öğesidir. Genel olarak donanımla birebir ilişkili çalışarak, işletim sistemindeki ve işletim sistemi üzerindeki süreçlere çeşitli hizmetler sağlar. Bu hizmetlerin en önemlileri, süreç yönetimi ve hafıza yönetimi olarak söylenebilir. Linux işletim sistemi kullanımının getirilerini anlamak ve sisteme uygulanıp uygulanmamasına karar vermek için çekirdek yapısını genel olarak anlamak önemlidir. Bu alt başlık bünyesinde çekirdek hakkında temel kavramlar açıklanacaktır.

Sistem başlatıldığında çekirdek RAM üzerine yüklenir sistemin çalışması için ihtiyaç duyulan çoğu önemli görevi yapmakla yükümlüdür. Bu yükümlülük temel olarak iki konuyu kapsar. Birincisi donanım bileşenleri ile iletişim kurmak iken ikinci temel görevi sistemdeki yazılımların çalışması için ortam sağlamaktır. Bu bağlamda bir program bir donanım kaynağını kullanmak istediğinde çekirdeğe bir istek gönderir. Eğer çekirdek uygun bulursa uygun zamanda donanım ile iletişime geçerek bu isteği yerine getirir [10].

Linux yekpâre çekirdek yapısına sahiptir. Bu yapıda kullanıcı ve çekirdek alanları birbirinden ayrıdır. Bir yazılım kullanıcı bölgesinde çalıştığında sistem donanımını ve özel komutları kullanamaz. Ayrıca bir uygulama sanal adres üzerinde çalıştığı için bir başka uygulamanın veya çekirdeğin bellek alanına girip uygulama ve sistem çökmelerine sebep olamaz [11]. Bu mimari Şekil 3.1'de görülmektedir.



Şekil 3.1: Yekpâre çekirdek mimarisi [11]

Şekil 3.1'de görülen donanım soyutlama katmanı (HAL), donanımı platformdan soyutlaması sebebiyle donanımlar için farklı sürücülerin kullanılabilmesine imkân tanır. Kart destek paketi (BSP) olarak da adlandırılan bu birim sistem açılışında donanım aygıtlarının başlatılması ve çalışma esnasında bu aygıtlara erişmeyi sağlar. BSP iki bileşenden oluşur. Bu bileşenlerin ilki mikroişlemci desteği, kincisi ise donanım projesine özgü desteklerdir. Önyükleyici, bellek haritası, sistem zamanlayıcısı, kesme kontrolü, gerçek zaman saati, seri port, veri yolu, DMA ve güç yönetimi destekleri bu kapsama girer [11]. Kısacası BSP donanım projesine göre oluşturulur ve çekirdek yapısına eklenmesiyle işletim sisteminin donanım yapısından haberdâr olmasını ve dolayısıyla donanımları kontrol etmesini sağlar.

Bellek yöneticisi bellek kaynaklarına erişimden sorumludur. Aygıt sürücüleri ve dosya sistemleri gibi çekirdek alt sistemlerine dinamik bellek alanı sağlar. Kullanıcı uygulamalarına sanal bellek alanı sağlayacak yazılımı da içinde barındırır. Linux işletim sisteminde "Demand Paging" denen hafıza yönetimi sistemi kullanılmaktadır. Bu hafıza yönetim biçiminde, hafıza, 4K'lık parçalara bölünmekte, bir sürecin o anda ihtiyaç duyulan kısımları 4K'lık parçalar halinde hafızaya yüklenmektedir. Böylece, çekirdeğin kendisinin bile kullanılmayan kısımları, sanal hafızada durabilmektedir [10]. İşletim sisteminin bir diğer görevi de, ayrı süreçlerin birbirinin hafızasına karışmamasını sağlamak, sistem ile kullanıcı hafıza parçalarını ayırmaktır. Böylece herhangi kullanıcı uygulaması, sistemin çalıştığı alana girerek, sistemin kilitlenmesine ya da yanlış çalışmasına yol açamaz.

Zamanlama yöneticisi çoklu işlem (multitasking) yeteneğini ağlayan birimdir [11]. Bu yetenek sayesinde aynı anda birden fazla program zaman paylaşımı yöntemi ile çalışabilmektedir. Bunun yanı sıra Linux kullanımı yaygınlık kazanmaya başladıkça gerçek zamanlı çalışan uygulamaların desteklenmesi ihtiyacı artmıştır. Bu sebeple zamanlama yöneticisi önem kazanmış ve bu ihtiyaçlar doğrultusunda ileri bir noktaya varmıştır.

G/Ç alt sistemi, sistem üzerindeki aygıtlar ile basit ve tekdüze bir ara yüz oluşturan birimdir. G/Ç alt sistemi ile 3 çeşit aygıt desteklenir. Bu aygıtlar ardışıl bir şekilde veri gönderen klavye, fare, dokunmatik ekran gibi karakter aygıtları, RAM, flash, disk gibi blok erişimli ve dosya sistemi içerebilen blok aygıtları ile ağ aygıtlarıdır. [11]

Aynı anda çalışan işlemler arasındaki veri paylaşımı sanal bellek kullanımının getirdiği kısıtlamalar nedeniyle geçici dosya oluşturma ile sağlanabilir. Bu ise disk dosya sistemine erişimi gerektirdiği için sistemi yavaşlatır. Bunun önüne geçmek için IPC (işlemler arası iletişim) protokolleri geliştirilmiştir [10]**.** Sinyaller, iş hatları,

paylaşımlı bellek birimleri, mesaj kuyrukları gibi yapılar kullanılarak işlemler arası iletişim sağlanır [12].

Linux'ta çeşitli dosya sistemleri sanal dosya sistemi adlı bir katman tarafından yönetilir. Sanal dosya sistemi birimi ile sistemdeki çeşitli aygıtlarda depolanmış verilere erişmek mümkün hale gelir. Örneğin bir dosya sistemine sahip olan bir depolama aygıtından başka bir dosya sistemine sahip bir belleğe *cp* komutu ile veri aktarmak istendiğinde *cp*'nin bu dosya sistemi özelliklerini bilmesi beklenmez. *cp* sanal dosya sistemine sistem çağrısı göndererek görevini yapar. Yani kullanıcı komutu ile depolama aygıtları arasında bir ara yüz vazifesi görür [10].

3.1.2. uClinux

Linux çekirdeği alt başlığında anlatılan bellek yönetim birimi (MMU) genel amaçlı işlemcilerin çoğunda bulunur ve sanal adresten fiziksel adrese geçişi sağlar. MMU bulunmayan bir işlemcide Linux işletim sistemi çalıştırma işi ilk olarak 1998 yılında M68k işlemcisi üzerinde yapılmıştır. Daha sonra giderek yaygınlık kazanan bu Linux sürümü uClinux adıyla gömülü sistemlerde sıkça kullanılmaktadır [13]. uClinux Linux çekirdeğine MMU bulunmayan işlemcilerde çalışma yeteneği ekler, böylelikle MMU içeren ve içermeyen işlemcilerde çalışabilmektedir.

Daha önce de bahsedildiği gibi MMU kullanıcı işlemlerini özel bellek alanlarında çalıştırmakta ve böylece bir işlemin başka bir işlemin bellek alanına girerek onun işlevini bozması mümkün değildir. Burada ise tüm işlemler ortak bellek alanında çalıştığı için program ve sistem çökmeleri meydana gelebilmektedir [11]. Fakat sisteme ek bir işlem yükü getirmektedir. MMU'nun devre dışı bırakılması sistemi hızlandırır. Dolayısıyla aynı işlemci üzerinde uClinux çalışan sistemi, Linux çalışandan daha hızlı olacaktır [13]. Bu özellik gömülü sistemlerde genelde gerçek zaman kısıtlamaları bulunduğundan önemlidir.

Şekil 3.2: Gerçek zamanlı çekirdek mimarisi [11]

Bununla beraber uClinux sistemi alt seviyede daha iyi kontrol imkânı vermektedir. Kullanıcı uygulamaları aygıtlardaki kaydediciler (registers) de dâhil olmak üzere tüm sisteme erişebilir [13]. Üstelik çok daha küçüktür. Linux 2.6 çekirdeği 300 KB bellek alanının altında yer kaplarken uClibc kütüphanesi ile program boyutları da önemli ölçüde azaltılmıştır.

Şekil 3.2'te uClinux tarafından kullanılan gerçek zamanlı çekirdek mimarisi görülmektedir. Bu mimari bellek idare ve koruması için yapılan işlemler ile sistem çağrılarının getirdiği yük bertaraf edildiği için gerçek zamanlı olarak anılır [11].

uClinux ile çok daha fazla işlemci desteklenebilmektedir. Bu işlemcilere Blackfin DSP işlemcileri ile FPGA üzerinde çalışan MicroBlaze işlemciler de dâhildir. Bu proje kapsamında MicroBlaze işlemciler için optimize edilmiş uClinux işletim sistemi kullanılmıştır.

3.1.3. Linux Sisteminin Başlama Aşamaları

Linux işletim sisteminin başlama aşamalarının anlaşılması Linux tabanlı sistemin temel bileşenlerinden olan önyükleyici ve kök dosya sisteminin anlaşılması açısından önem arz etmektedir. Gömülü sistemlerde genelde bu aşamanın mümkün olduğunca hızlı olmasına gayret gösterilir.

Linux sistemini başlatma işlemi 3 temel aşamada incelenebilir [11]. İlk aşama ön yükleyici basamağıdır. Bu aşamanın ilk etabında donanımlar yapılandırılarak başlatılır. MİB hızı, bellek bölümünün başlatılması, silinmesi, ana bilgisayarla haberleşecek olan seri portun yapılandırılması, donanım testleri bu etapta gerçekleşir. Eğer bütün bu işlemle başarıyla tamamlanırsa Linux çekirdeğinin yüklenmesi aşamasına geçilir. Önyükleyici tarafından Linux çekirdeğinin yeri bilinmelidir. Linux genelde sıkıştırılmış bir vaziyette sistemin flash belleğinde bulunur. Bu sistemi açıp belleğe yükleme işlemi bu aşamada gerçekleşir. Bu gibi görevleri yerine getirdikten sonra önyükleyici çekirdek giriş noktasına atlar ve görevini sonlandırır. Artık işletim sistemi çekirdeği çalışmaktadır.

İkinci aşama çekirdeğin başlatılması aşamasıdır. Bu aşamada işlemci ve donanım yapısı çekirdeğe tanıtılır. BSP vasıtasıyla çekirdeğin sistemi kontrol edebilecek duruma gelmesi sağlanır. Çekirdeğe ait alt sistemler başlatılır, sürücüler yüklenir ve kök dosya sistemi yürürlüğe girer.

Üçüncü ve son aşama kullanıcı alanının başlatılmasıdır. Bu aşama işletim sisteminin dağıtımına göre farklılık gösterir. Bu aşama ile birlikte sistemin başlatılması tamamlanmıştır. Kullanıcı uygulamaları artık çalıştırılabilir.

3.2. Proje Ortamının Kurulması

Linux tabanlı bir gömülü sistem için geliştirme ortamı çapraz derleyici, önyükleme programı, GNU yazılımları ile birlikte Linux çekirdeği, C kütüphaneleri ve hata ayıklama (debug) araçlarını içerir. Mikroişlemci tabanlı bir donanım sistemi üzerinde çalıştırılacak yazılımın bu araçlar kullanılarak yapılandırılması, bu araçların ise donanım sistemine göre düzenlenmesi gerekir.

Bu projede PetaLogix firması tarafından geliştirilmiş bir gömülü Linux geliştirme ortamı olan PetaLinux kullanılmıştır. PetaLinux, MicroBlaze için çapraz derleyicinin de dâhil olduğu tüm gerekli programları sağladığı gibi Xilinx ISE ve EDK araçlarını da uyumlu bir şekilde kullanır [14].

Önyükleyicinin ve Linux çekirdeğinin donanıma göre yapılandırılması için donanım haritasının uygun formatta önyükleyici ve çekirdek yapılandırma programlarına eklenmesi gerekir. Bu işi PetaLogix AutoConfig programı yaparak yazılım ve donanım yapılandırmaları arasındaki senkronizasyonu basitleştirir [14].

PetaLinux, ISE ve EDK'nın araçlarını kullandığı ve yalnızca Linux ortamında çalıştığı için tüm ortamlar Linux işletim sistemi üzerine kurulmuştur. Bu masa üstü işletim sistemi için Linux Fedora Core 10 işletim sistemi tercih edilmiştir. Kurulumda dikkat edilmesi gereken en önemli husus, sistem dilinin İngilizce olarak seçilmesidir. Türkçe olarak seçilen tüm Linux dağıtımlarında Xilinx araçları çalışmamaktadır. Daha sonra Xilinx yazılımları gibi çeşitli programları kurmak için gerekli olan, standart C++ kütüphaneleri ile uyumu sağlayan libstdc++.so.5 kütüphanesi sisteme eklenmelidir.

USB kablo ile geliştirme kitini programlamak için Xilinx tarafından sağlanan yazılım *windrvr* modülünü kullandığı ve bu modül 2.6.18'den yüksek çekirdek sürümlerinde çalışmadığı için aşağıdaki adımlar izlenmiştir:

1. Sürücüler için gerekli olan libusb-devel ve libftdi-devel paketleri kuruldu.
2. usb-driver-HEAD.tar.gz[1] dizini, /opt/ dizini içine açıldı.
3. Açılan usb-driver dizini içinde make komutu çalıştırıldı. Bu komutun işlenmesi ile libusb-driver.so adlı bir sürücü oluştu.

İşletim sisteminin USB üzerinden Xilinx aygıtına erişebilmesi için aygıt kimliğinin sistem tarafından bilinmesi gereklidir. lsusb komutuyla listelenen USB aygıtlarından Xilinx, Inc etiketli olan satırdaki kimlik numarası ID 03fd:0008 değerinden farklı ise

[1] http://git.zerfleddert.de/cgi-bin/gitweb.cgi/usb-driver?a=snapshot;h=HEAD; sf=tgz bağlantısından elde edildi.

fxload paketinin yüklenmesi gerekir. Bu paket yüklendikten sonra doğru kimlik bilgisi listelenecektir.

Bundan sonra ISE ve XPS programları kütüphanenin önyükleme işlemi ile birlikte çalıştırıldığında FPGA aygıt programlaması yapılabilecektir:

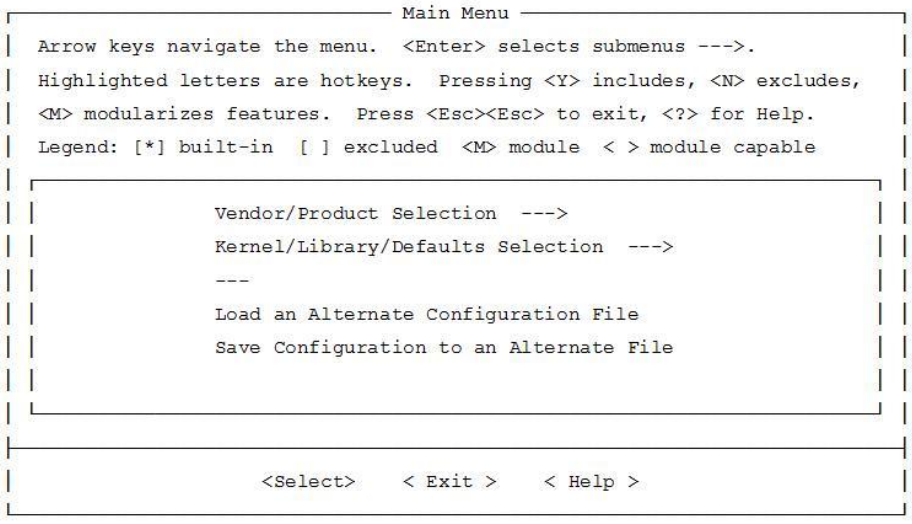
```
LD_PRELOAD=/opt/usb-driver/libusb-driver.so ise
```

```
LD_PRELOAD=/opt/usb-driver/libusb-driver.so xps
```

Projede kullanılan diğer bir yazılım *Kermit*'tir. Kermit, Kolombiya Üniversitesi'nde gerçekleştirilen çok amaçlı bir iletişim yazılımı projesidir. Seri iletişimi gerçekleştirebildiği gibi ağ bağlantısı da yapabilir. Kurulan bağlantı üzerinden hedefe metin dosyaları ve ikili (binary) dosyalar göndermeye imkân sağlar [15]. Projede Kermit programı hem işletim sistemi çekirdeğinin gönderilmesinde hem de işletim sistemi ile ana bilgisayar üzerinden bağlantı kurulmasında kullanılmaktadır.

3.3. PetaLinux ile çekirdek yapılandırması

Donanım projesinin tamamlanıp AutoConfig programı ile sistem haritasının PetaLinux dizin yapısına uygun biçimde kopyalanmasının ardından çekirdek ve önyükleyici yapılandırma aşamasına geçilir. Bu aşamada çekirdek içinde bulunması istenen bileşen, program, araç ve kütüphanelere karar verilecek ve sonunda çalıştırılabilir bir çekirdek dosyası ile donanıma göre hazırlanmış U-boot dosyası elde edilecektir.

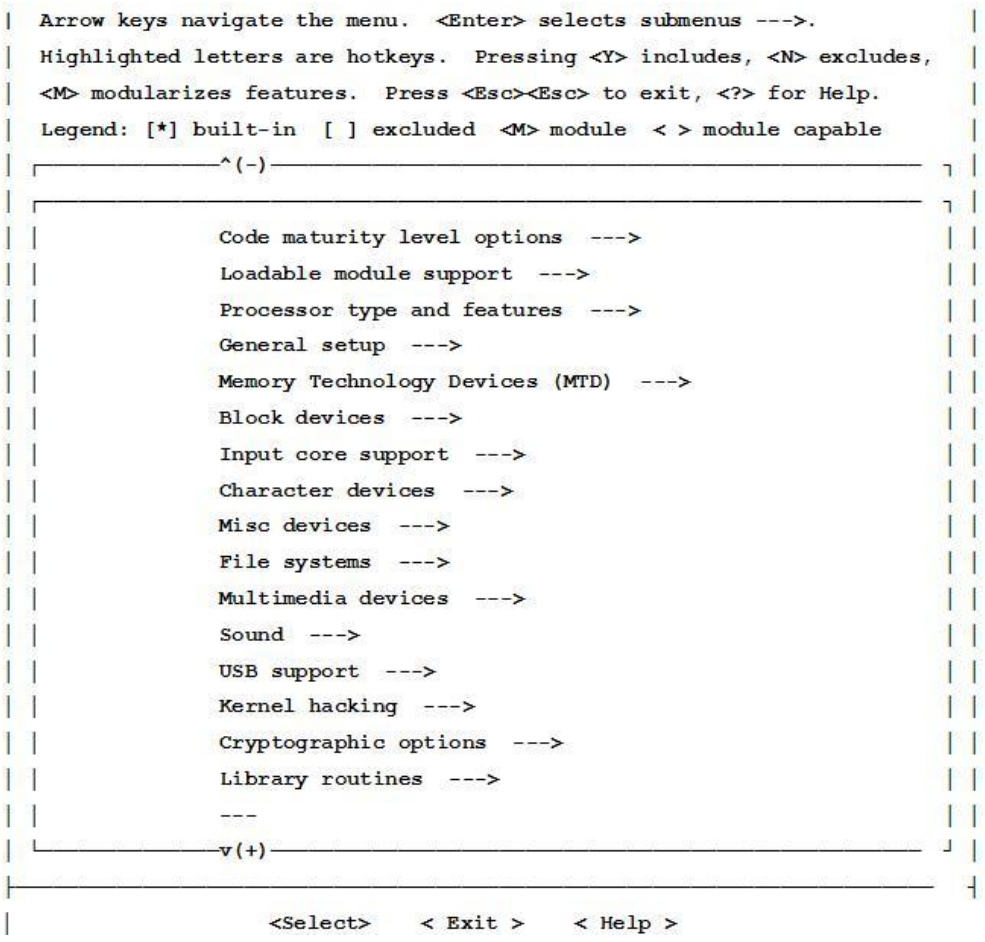


Şekil 3.3: PetaLinux ana menüsü

Şekil 3.3'te görüldüğü gibi ilk aşamada üretici ve ürünün seçilmesi gerekir. Daha sonra çekirdek sürümü ile kullanılacak kütüphanenin seçilmesiyle sonraki aşama olan yapılandırmaya geçilebilir.

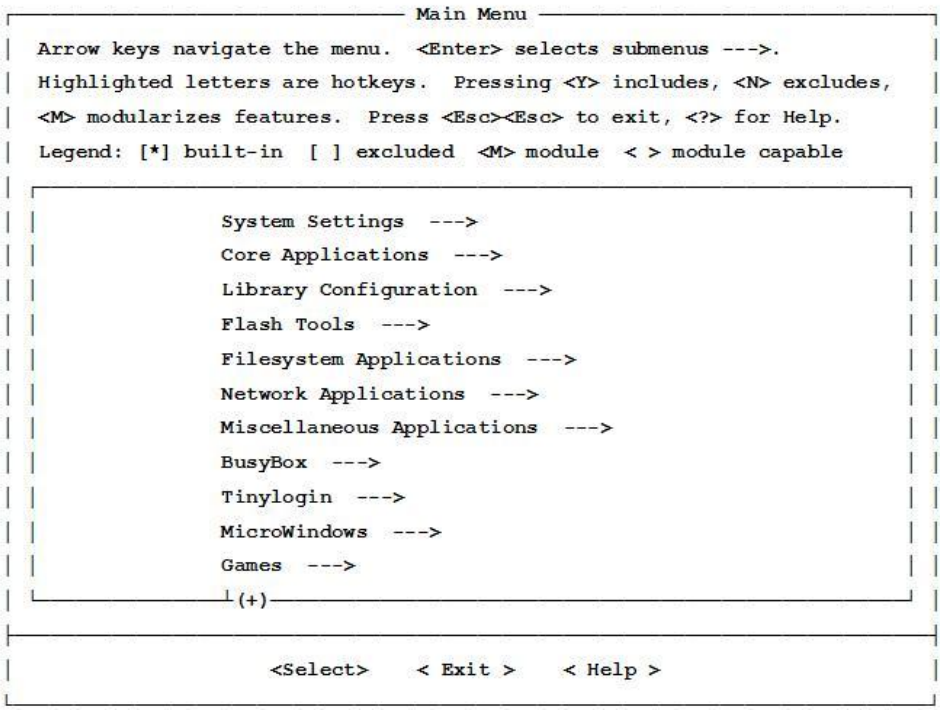
Yapılandırma bölümü iki aşamada incelenebilir. Önce çekirdek ve kütüphane yapılandırması tamamlanır, ardından sisteme eklenecek uygulamalar seçilir. Tasarımcı tarafından yazılan yazılımların derlenip sistem ile bütünleştirilmesi ise çekirdek oluşturulduktan sonra yapılacaktır.

Çekirdek ve kütüphane yapılandırması bölümünde aygıtlar ile ilgili ayarların da yapıldığı Şekil 3.4'te görülmektedir. Linux'ta G/Ç alt sisteminin aygıtları temel olarak üç sınıfta toplayıp buna göre iletişim kurduğu belirtilmişti. Burada da karakter ve blok aygıtlarına ilişkin düzenlemelerin yanı sıra işlemci özellikleri ve desteklenecek dosya sistemleri çekirdek özellikleri belirlenebilmektedir.



Şekil 3.4: PetaLinux'ta çekirdek yapılandırması

Şekil 3.5'te görülen üretici ve kullanıcı ayarları bölümünde işletim sistemi yapısına eklenecek uygulamalar seçilir. Önceki bölümlerde de bahsedildiği gibi gömülü işletim sistemini en az kaynak kullanacak şekilde yapılandırmak temel amaçtır. Bu sebeple ihtiyaçları karşılayacak özellikler ve uygulamalar tek tek belirlenir. Video, resim, ses, şifreleme, ağ gibi çeşitli alanlara ait kütüphane ve programlar ihtiyaçlar doğrultusunda seçilerek sisteme eklenebilir. Burada Linux kullanmanın avantajı görülmektedir. Linux için geliştirilmiş çok sayıda araç sistemde kullanılabilir. Mesela şifreleme (kripto) kullanmak istenen bir sistemde ilgili yazılımları yazmak yerine buradaki yapılandırmada sistem bünyesine hazır olarak koymanın önemli miktarda zaman kazandıracağı aşikârdır.



Şekil 3.5: PetaLinux üretici ve kullanıcı ayarları bölümü

Tüm ayarlamalar yapıldığında derleme safhasına geçilir. Çekirdek ve uygulamalar çapraz derleyici kullanılarak derlenir ve önyükleyici programı sistem haritasına göre oluşturulur. PetaLinux dizini altında oluşturulan dosya sistemi görülebilmektedir.

Bu işlemin ardından kullanıcı tarafından yazılmış olan yazılımların derlenerek dosya sistemine eklenmesi gerçekleştirilir. Neticede sisteme uygun önyükleme programı ile kullanıcı uygulamaları ve hazır programları da kapsayan bir işletim sistemi elde edilmiş olur. Bir sonraki adım elde edilen bu yapıların sisteme yüklenmesidir.

3.4. Önyükleme Sistemi

Sistemde iki aşamalı ön yükleme gerçekleşmektedir. MicroBlaze tabanlı sistem ilk açıldığında bellek haritasının başlangıç adresindeki (0x00000000) komuttan itibaren görevini icra etmeye başlar. Bu adres sistemdeki Blok RAM'in başlangıç adresidir. Bu bölgeye EDK ortamında iken FS-Boot adlı küçük bir önyükleme programı koyulur. FS-Boot programının görevi ana önyükleyici programını çalıştırmaktır. Blok RAM kullanımını en aza indiren FS-Boot flash bellekten U-Boot programını otomatik olarak çalıştırabildiği gibi kullanıcı bilgisayarındaki seri port üzerinden gönderilen U-Boot programını da yükleyip çalışmasını başlatabilmektedir. Şekil 3.6'te FS-Boot'un Kermit'e gönderdiği ileti görülmektedir.

```
================================================

FS-BOOT First Stage Bootloader (c) 2006 PetaLogix

================================================

FS-BOOT: System initialisation completed.

FS-BOOT: Booting from FLASH. Press 's' for image
download.
```

Şekil 3.6: FS-Boot iletisi

Bu sistemde ana önyükleyici programı açık kaynak kodlu U-Boot önyükleyicisidir. İlk etapta PowerPC mimarisi için Wolfgang Denx tarafından geliştirilmiş olan U-Boot daha sonra ARM, M68K, MIPS ve MicroBlaze mikroişlemcileri için de uyarlanmıştır [11]. Gömülü sistemleri için pek çok aracı destekler. Bu önyükleyici ile sistemde bir yerden bir yere veri kopyalamak, depolama alanlarındaki verileri görüntülemek, silmek ve değiştirmek, flash yazma izinlerini düzenlemek, kullanıcı bilgisayarından gelen veriyi istenen hedefe yerleştirmek, işletim sistemini başlatmak gibi pek çok işlemi gerçekleştirilebilmektedir.

Çekirdek yapılandırılması esnasında donanıma göre yapılandırılmış olan U-Boot programı (u-boot.srec) seri port üzerinden FPGA geliştirme kartına gönderilir. Bu noktadan itibaren U-Boot çalışmaya başlar ve seri haberleşme U-boot ile gerçekleştirilir.

3.5. Sistemin Başlatılması

Referans sisteme hücresel sinir ağı donanımlarını ekleyip donanım projesi ile FS-Boot önyükleme programını içeren yazılım projesini sentezledikten sonra FPGA'nın yapılandırılmasıyla Microblaze Blok RAM'de yer alan FS-Boot programını çalıştırır. Kermit ile Şekil 3.6'daki FS-Boot mesajının görülmesinin ardından U-Boot programını içeren *u-boot.srec* dosyası sisteme gönderilir. U-Boot ön yükleyicisinin çalışmaya başlamasıyla işletim sistemi seri port ya da eğer ayarlamaları yapıldıysa ethernet üzerinden gönderilebilir.

Gönderilen işletim sistemi DDR belleklerin üzerine yerleştirilir. Buradan doğrudan çalıştırılmaya başlanacağı gibi flash belleğe kopyalanmasının ardından da çalıştırılabilir. Flash belleğin kullanımıyla kalıcı bellekten faydalanılmış olur ve sonraki sistem açılışlarında kullanıcı bilgisayarından işletim sistemi gönderme zorunluluğu ortadan kalkar. Bunun için ise önce U-Boot ile flash belleğin yazma izinleri düzenlenmelidir. İşletim sisteminin yerleştirileceği adresler arasının yazma koruması kaldırılır ve Linux bu bölgeye yerleştirilir. Otomatik açılış için U-Boot yapılandırması da buna göre yapılmalıdır. İşletim sisteminin başlangıç adresi U-Boot'a tanıtılırsa FPGA geliştirme kiti açıldığında FS-Boot'un otomatik olarak U-Boot önyükleyicisini çalıştırmasını bu programın Linux işletim sistemini başlatması izler.

Bölüm 2.7.3'te bahsedilen sistem başlama aşamaları sırayla gerçekleştirildikten sonra işletim sistemi ile iletişim kurulur. NPE donanımlarını kontrol eden yazılımlar seri port vasıtasıyla iletişim kurulan konsol ile çalıştırılır.

3.6. Sonuç

Gömülü Linux üzerinden çalışan bu sistem proje kapsamındaki son halini almıştır. Bu durumda aynı FPGA üzerinde mikroişlemci, NPE donanımları ve veri yolları haberleşme kontrol birimleri yer almaktadır. Mikroişlemci kart üzerindeki belleklerde bulunan Linux işletim sistemini çalıştırmakta ve seri port üzerinden sistemle iletişim kurulmaktadır. İşletim sistemi dosya sistemi içine yerleştirilmiş olan ve NPE'leri kontrol eden yazılımlar bu ortamda çalıştırılabilmektedir.

FPGA geliştirme kartı üzerine kurulmuş olan ve gömülü işletim sistemi çalıştıran bu sistem genel bir gömülü sistem geliştirme platformu olarak da görülebilir. NPE donanımları ve kontrol yazılımlarının değiştirilmesi ile farklı sistemler kolayca elde edilebilir. Ayrıca çekirdeğin yeniden düzenlenip derlenmesi ile faklı özellikleri ön plana çıkan işletim sistemleri elde etmek mümkündür.

Sistemde kullanılan yazılımlar işletim sisteminin tez metninde bahsedilen özelliklerini kullanmayı gerektiren yazılımlar olmadığı için bu sistemde Linux önemli bir görev ifa etmez. Fakat ileriki çalışmalarda yazılım ve donanımların daha gelişip karmaşıklaşmasıyla kurulan, birden fazla programın aynı anda çalıştığı ve aralarında iletişimin olduğu, dosya sistemleri ile işlemler yapan, aynı anda birden fazla donanımı kontrol eden sistemler için işletim sistemi kaçınılmaz olacaktır ve bu proje kapsamında kurulan bu düzen bahsedilen ileri projeler için bir taban sistem niteliği taşımaktadır.

KAYNAKÇA

1. **Chu, Pong P.** *FPGA Prototyping By VHDL Examples.* New Jersay : Wiley-Interscience, 2008.

2. **Rosinger, Hans-Peter.** *Connecting Customized IP to the MicroBlaze Soft Processor Using the Fast Simplex Link (FSL) Channel.* s.l. : Xilinx, 2004.

3. **Nedjah, Nadia ve Mourelle, Luiza De Macedo.** *Co-Design for System Acceleration, A Quantitive Approach.* s.l. : Springer, 2007.

4. **Xiu, Liming.** *VLSI Circuit Design Methodology Demystified.* s.l. : Wiley-Interscience, 2008.

5. **Xilinx.** *Embedded System Tools Reference Manual.* s.l. : Xilinx, 2007.

6. —. *MicroBlaze Processor Reference Guide.* s.l. : Xilinx, 2007.

7. —. *LMB BRAM Interface Controller.* s.l. : Xilinx, 2009.

8. *An Implementation of 2D Locally Coupled Relaxation Oscillators on an FPGA for Real-time Autowave Generation.* **Yeniçeri, Ramazan ve Yalçın, Müştak E.** 2008.

9. *A simple programmable autowave generator network for wave computing applications.* **Yalçın, Müştak E.** 11, s.l. : IEEE Transactions on Circuits and Systems II-Express Briefs, Nov 2008, Cilt 55.

10. **Bovet, Daniel P. ve Cesati, Marco.** *Understanding the Linux Kernel .* s.l. : O'Reilly, 2000.

11. **Raghavan, P., Lad, Amol ve Neelakandan, Sriram.** *Embedded Linux System Design and Development.* s.l. : Auerbach Publications, 2006.

12. **Rusling, David A.** *The Linux Kernel.* 1999.

13. **Michael Opdenacker.** Introduction to uClinux. *Free-Electrons.* [Çevrimiçi] 20 Kasım 2007. [Alıntı Tarihi: 23 Aralık 2009.] free-electrons.com.

14. **PetaLogix.** User Guide. *PetaLogix Developer Portal.* [Çevrimiçi] 22 06 2009. [Alıntı Tarihi: 28 08 2009.] http://developer.petalogix.com.

15. C−Kermit 8.0 manual page and tutorial. *Kermit Project.* [Çevrimiçi] 22 11 2001. [Alıntı Tarihi: 10 11 2009.] http://www.columbia.edu/kermit/.

ÖZGEÇMİŞ

Selman Ergünay 1987 yılında Kütahya'da doğdu. 2005 yılında Kütahya Fen Lisesi'nden mezun oldu. 2005 yılından beri İTÜ Elektrik-Elektronik Fakültesi Elektronik Mühendisliği programında öğrenim görmektedir.