FGPA ile Gömülü Sistem Tasarımına Giriş
Introduction to Embeded System Design Using FPGA

Selçuk BAŞAK

Bilgisayar Mühendisliği Bölümü
Yıldız Teknik Üniversitesi, İstanbul
selcuk@selsistem.com

Özetçe

Bu çalışmada, programlanabilir lojik hakkında temel bilgiler, CPLD ve FPGA cihazları, FPGA ile gömülü sistem tasarımı ve örnek bir uygulama sunulmuştur.

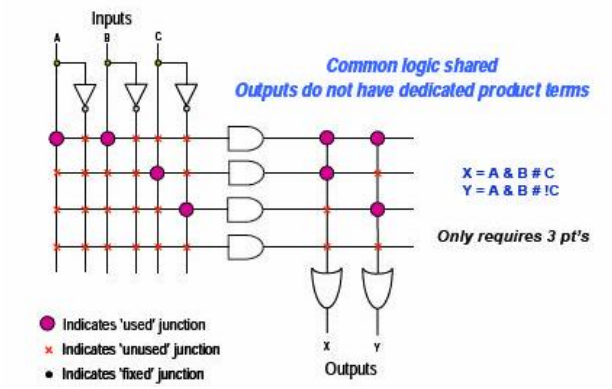
Abstract

In this study, fundamental information about programmable logic, CPLD and FPGA devices and embedded system design and a sample application are presented.

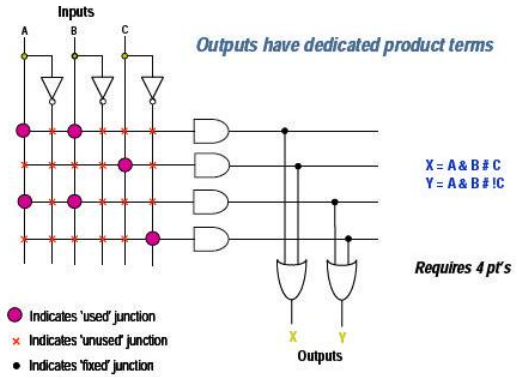
1. Giriş

1.1. Programlanabilir Lojik

1970'lerin sonlarında standart lojik birimlerim kullanımı oldukça yaygınlaşmıştı. Çok sayıda standart lojik birimlerden elektronik kartlar ile sistemler geliştirilmekteydi. Esnek lojik tasarım ihtiyaçlarına yönelik olarak Signetics firmasından Ron Cline, PLA(programmable logic array) fikrini sunmuştur. Signetics firması, önce Philips, sonrasında ise Xilinx tarafından satın alınmıştır. PLA dikey ve yatay hatlardan ve AND - OR kapılarına bağlanan bir mimariye sahiptir. AND-OR kapılarının çıkışında flip-floplar bulunmakta ve bunların çıkışı da cihazın çıkışlarına bağlıdır. Dikey ve yatay hatların kesişim noktalarında sigortalar bulunuyor. Yazılım araçları ile istenmeyen sigortalar "patlatılarak" istenilen fonksiyon için programlama yapılabilir. PLA mimarisi oldukça esnekti ancak zamanın üretim teknolojisi 10µm olması nedeniyle, giriş-çıkış gecikme süresi (yayılma gecikmesi) uzun ve bu nedenle yavaştı.

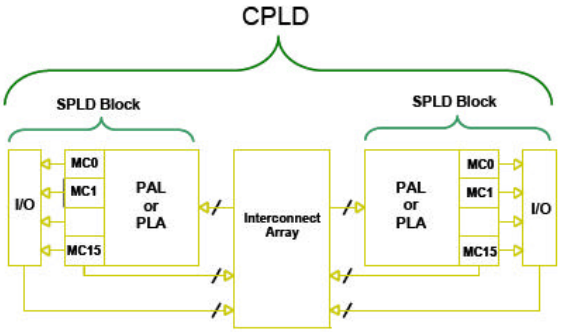


Şekil 1: SPLD Mimarileri: PLA



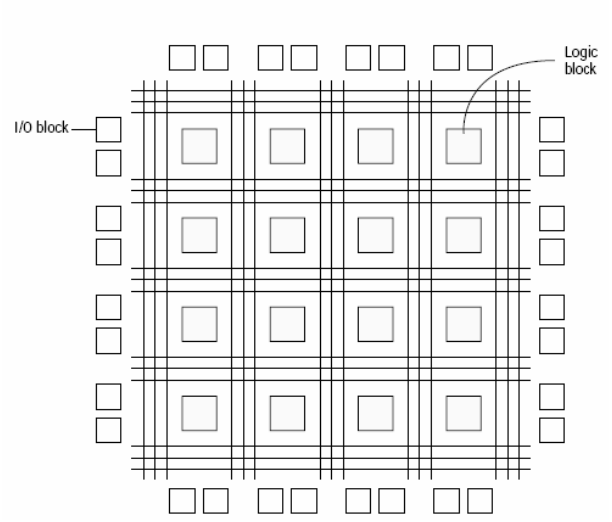
Şekil 2: SPLD Mimarileri:PAL

MMI firması (AMD tarafından satın alınmıştır) başka bir yaklaşımda bulunarak ve PLA mimarisinden farklı olarak OR dizesini sabitlemiştir ve PAL (programmable array logic) mimarisini geliştirmiştir. PAL mimarisi daha sade ve daha hızlı (yayılma gecikmesi daha kısa) ancak PLA'e göre daha az esnektir. Bu mimariler genel olarak SPLD (Simple Programmable Logic Device) olarak adlandırılır. Programlama, özel bir programlama cihazı ile istenmeyen bağlantıların açık devre yapılması ile yapılmaktaydı. SPLD'ler, standart lojik entegrelerinin (örneğin:7400 serisi) 50 katından daha fazla sayıda kapı içermekte ve daha tutarlı bir yapı sunmaktadır.



Şekil 3: CPLD Mimarisi

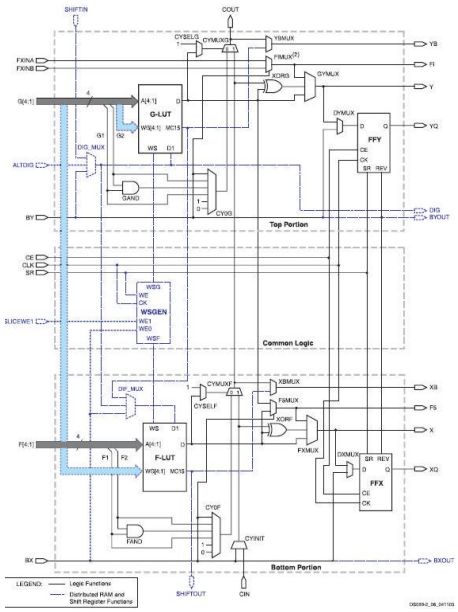
CPLD, mimari yapısı, bir çok küçük SPLD'ler (macrocell) ve bunların arasında genel amaçlı ara bağlantılardan oluşur. Basit lojik fonksiyonlar macrocell içinde gerçekleştirilirken, daha karmaşık fonksiyonlar birden çok macrocell'in genel amaçlı ara bağlantılar ile birleştirilerek elde edilebilmektedir. CPLD 200MHz hıza ulaşabilmektedir. CPLD, tasarım kolaylığı, geliştirme maliyetlerinin düşük olması, gibi bir çok avantaj sağlamaktadır. CPLD cihazlarının bir özelliği de konfigürasyonunu kendi üzerinde saklayabilmesidir.



Şekil 4: FPGA Mimarisi

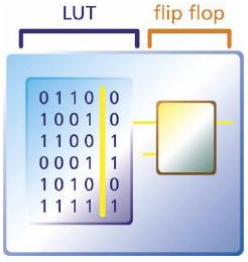
FPGA, 1985 yılında Xilin'in kurucusu Ross Freeman tarafından icat edilmiştir. FPGA mimari yapısı, tamamı kullanıcı tarafından belirlenebilen lojik bloklar ve ara bağlantılarından oluşmuştur. FPGA'ler Günümüzde 10 milyon eşdeğer kapı sayısına (iki girişli AND olarak) ulaşmıştır.

FPGA'ler Yapılandırılabilir Lojik Bloklar (Configurable Logic Blocks - CLBs) ve Giriş/Çıkış Blokları (I/O -Blocks-IOBs) içermektedir. CLB hem senkron hem de kombinasyonel devreler için temel lojik kaynakları içerir.

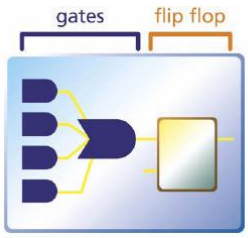


Şekil 5: Xilinx XC3S200 CLB (Basitleştirilmiş)

FGPA'lerin, OT(-One Time Programmable) ve SRAM-tabanlı programlanabilen olarak iki temel tipi vardır.Bu iki FPGA tipinde lojik blokların fiziksel gerçeklenmesi ve bağlantıları oluşturmak için kullanılan mekanizma farklıdır. Daha yaygın olarak kullanımı olan SRAM-tabanlı FPGA'ler istenildiği kadar tekrar programlanabilir. SRAM FPGA'ler zaten özel bir çeşit hafıza gibidir ve sisteme her güç verildiğinde yeniden programlanır. Bu nedenle SRAM FPGA'leri programlamak için PROM veya benzeri bir hafıza birimine ihtiyaç duyulur. SRAM lojik blokları, loijk kapılar yerine LUT(Lookup Table-Arama Tablosu) içerir. LUT girişlere göre çkış sonucunu belirler.



Şekil 6: SRAM Lojik Blok



Şekil 7: OTP Lojik Blok

OTP FPGA'ler çip içindeki bağlantıları kurmak için anti-fuse kullanır. (anti-fuse: sigortadan farklı olarak normalde açık sadece istenen bağlantılar kurulması ile yapılıyor.) OTP FPGA'leri programlamak için PROM veya diğer hafıza birimine gereksinim duymazlar. Ancak tek bir kez programlanabildiklerinden herhangi bir tasarım değişikliğinde eski çipi atmak gerekir. OTP lojik blokları, çıkışında kendisine ait filip-floplar bulunduran bir PLD yapısına çok benzer.

FPGA'ler CLB ve IOB bileşenlerinin yanında DSP modülleri, Multiplier, dahili RAM veya ROM Hafıza blokları, fiziksel CPU gibi kaynaklara da sahip olabilirler. Sistem tasarımında bu kaynaklardan yararlanılarak verimli ve hızlı çözümler geliştirilebilir.
FPGA'ler PLL veya DLL gibi zamanlama senkronizasyonu mekanizmalarına da sahip olabilmektir. Bu şekilde yapılan tasarımın çip üzerinde tasarımda belirlenen gecikmeler gerçeklenebilir.

FPGA'ler genellikle benzerleri ASIC'lere (application specific integrated circuit) göre daha yavaştırlar. Ancak ASIC'ler karmaşık tasarımların üstesinden gelemezler, aynı zamanda da daha fazla güç tüketirler. Bunların yanında FPGA'lerin pazara giriş zamanı düşük (daha kısa geliştirme zamanı), çıkan hataları saha da düzeltebilme ve daha az tekrarlamayan mühendislik maliyeti gibi avantajları vardır.

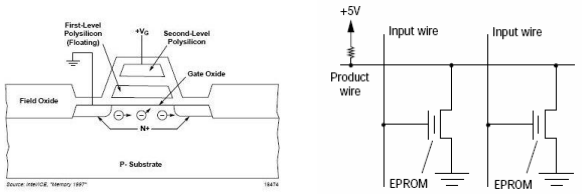
CPLD'ler ve FPGA'ler oldukça fazla sayıda programlanabilir lojik element içermektedirler. CPLD'in lojik kapı sayısı birkaç binlerden on binler düzeyine kadar olabilmektedir. FPGA'ler de ise bu sayı on binlerden milyonlarca ya kadar çıkabilmektedir. CPLD'ler ile FPGA'ler arasındaki başlıca fark mimari ile ilgilidir. Bir CPLD oldukça sınırlı bir yapıya sahiptir. Bir ya da daha fazla sum of products lojik dizisi nispeten daha az sayıda zaman yazmacını beslemektedir. Bunun sonucu daha az esnekliktir (öngörülebilir zamanlama gecikmesi ve daha yüksek oranda bağlantı oranı avantajı ile birlikte). Diğer taraftan FPGA mimarisi, bağlantı konusunda baskındır. Bu durum FPGA'leri daha esnek ve daha karmaşık dizaynlara hitap eden bir hale getirmektedir. CPLD'ler ile FPGA'ler arasındaki dikkate değer bir başka fark, bir çok FPGA üst düzey gömülü fonksiyonlara (toplama,çarpma) ve gömülü hafızalara sahiptir.

1.2. Programlanabilen Anahtarlama Teknolojileri

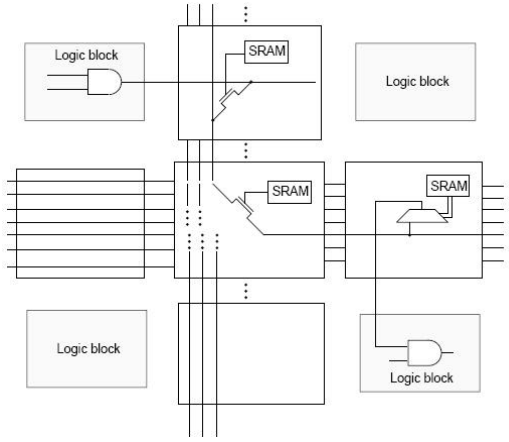
İlk geliştirilen programlanabilen anahtar PLA'lerde kullanılan (fuse) sigortaydı. CPLD'lerde temel anahtarlama teknlojisi olarak EPROM ve EEPROM'larda da kullanılana benzer "floating gate transistor" kullanılır. FPGA'lerde ise SRAM (Static RAM) ve antifuse yapısı kullanılır. Antifuse değiştirilmiş CMOS teknolojisi ile üretitilir. Antifuse normalde açık devredir. Programlama ile düşük dirençli bir bağlantıya dönüştürülür.

Tablo 1: Programlanabilen Anahtarlama Teknolojileri

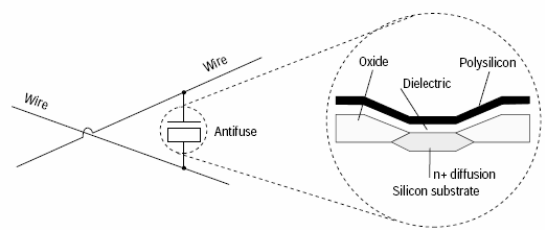
Switch type	Reprogrammable?	Volatile?	Technology
Fuse	No	No	Bipolar
EPROM	Yes (out of circuit)	No	UVCMOS
EEPROM	Yes (in circuit)	No	EECMOS
SRAM	Yes (in circuit)	Yes	CMOS
Antifuse	No	No	CMOS+



Şekil 8 : EPROM Programlanabilen Anahtarlar



Şekil 9: SRAM Kontrollü Programlanabilen Anahtarlar



Şekil 10: Antifuse Yapısı

2. Donanım Tanımlama Dilleri (Hardware Description Languages), Verilog/VDHL

2.1. Donanım Tanımlama Dilleri - HDL

İlk donanım tanımlama dilleri 1977 yılında, Carnegie Mellon Universitesinde 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 kullanılmıştır ancak sentezinde kullanılmamıştır. KARL'ın VLSI çip yerleşim planı ve yapısal donanım tasarım desteği vardı. KARL'ı temel alan ABL, 1980'lerin başında, the ABLED grafik VLSI tasarım editöründe kullanılmak üzere, telekomünikasyon araştırma merkezi CSELT, Torino, Italy tarafından geliştirildi. 1983'te Data-I/O ABEL dilini geliştirdi. ABEL programlanabilir lojik cihazları için ve temel olarak sonlu durum makinelerini tasarlamak için kullanılmıştı.

İlk modern HDL olan Verilog, 1985 yılında Gateway Design Automation tarafından geliştirilmiştir. Cadence Design Systems tarafından geliştirilen Verilog-XL HDL simulatörü verilog simulatörlerinin de-facto standardı oldu. 1987'de Amerika Birleşik Devletleri Savunma Bakanlığı, VHDL (Very High Speed Integrated Circuit Hardware Description Language.) geliştirilmesini istedi. Başlangıçta, Verilog ve VHDL şematik devre tasarımlarının dökümantasyonu ve simulasyonu için kullanımıştır. HDL simülasyonu mühendislere şematik seviyeden daha fazla soyutlama imkanı sağlamdı ve tasarım kapasitesini yüzlerce transistörden binlercesine taşıdı.

HDL'den lojik sentezlemenin yapılması ile, HDL dijital tasarımın arkaplanından önplanına geçti. Sentezleme araçları HDL dosyalarını (RTL - register transfer level formatında), üretilebilir kapı/transistör seviyesinde netlist oluştutur. Sentezlebilir HDL - RTL dosyaları yazmak için tasarımcının, tecrübeli olması ve de özen göstermesini gerekmektedir. Sentezlenen RTL sonucu elde edilen sonuç, şematik tasarıma göre nerdeyse her zaman daha fazla kaynak kullanan 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 digital tasarımın merkezine getirmiş, hem de dijital devre tasarımda teknolojik bir devrim olmuştur. Birkaç yıl içinde hem VHDL ve Verilog, elektronik endistrisinde 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) 'lerden yararlanılır. VHDL ve Verilog HDL dosyaları, algoritmaların daha kolay ifade edilebileceği C/C++ gibi standart yazılım programlama dillerinden veya SystemC gibi donanıma tasarımına özelleşmiş programlama dillerinden derlenerek de elde edilebilir. Bu sayede daha karmaşık sistemler daha da üst seviyeli bir soyutlama ile gerçeklenebilir. Örneğin DSP uygulamaları veya karmaşık algoritmalar içeren yapıların gerçeklenmesi oldukça kolaylaştırabilir.

2.2. Verilog

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 geliştirildi. Cadence Design System, Gateway Design Automation'ı satın aldı ve 1990 yılında Verilog'u genel kullanıma açtı ve bunun sonucu olarak Verilog 1995 yılında IEEE tarafından standartlaştırıldı. Verilog 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 ile (Çıkışları giriş sinyallerinin dönüşümü ile) ve Davranışsal olarak (Devreden beklenen davranışın ifasesi şeklinde) üç değişik yapıda ifade edilebilir. Bu şekilde daha üst seviyeli modüllerde soyutlamaya gidilebilmesini ve alt seviyeli modüllerinde de verimli olarak ifade etme imkanı sağlar. Şekil 11'de yönü seçilebilen 4-bitlik bir sayaç'ın sentezlenebilir olarak verilog ile ifadesi vardır.

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

reg [3:0] count_int = 0;

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

assign COUNT_OUT = count_int;

endmodule
```

Şekil 11: Verilog ile 4-Bit Sayaç

2.3. VHDL

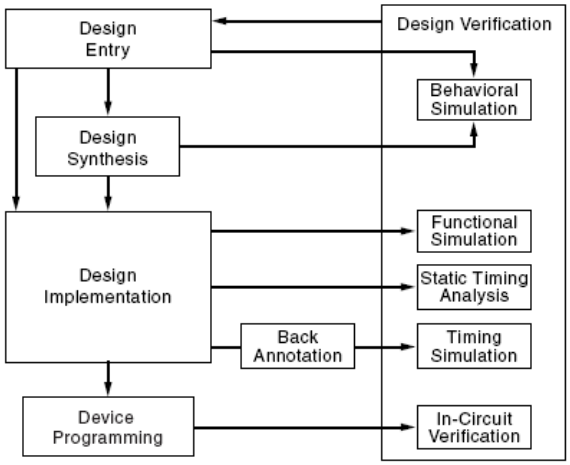
VHDL (Very high speed integrated circuit Hardware Description Language) Amerika Birleşik Devletleri Savunma Bakanlığı sponsorluğunda elektronik cihazların modellenmesi ve simülasyonu için desteklenen bir HDL olarak geliştirilmiştir. VHDL'in geliştirilmesindeki amaç modelleme, simülasyondu. Ancak sonradan araştırtmacılar tarafından tasarım aşamalarının otomasyonunu sağlayabilmek için sentezlemede de kullanılmaya başlandı. Başlangıçta askeri amaçlı projeler için kullanılsa da sonrasında özel sektörde askeri olmayan projelerde de kullanılmaya başlandı. VHDL IEEE tarafından ilk olarak 1987 yılında standartlaştırıldı. VHDL, Verilog'a göre daha esnek sistem tasarımlarına imkan vermekle beraber üst seviyeli tasarıma daha uygundur. Ancak öğrenilmesi ve kullanımı biraz daha karmaşık ve uzundur. Aşağıda yönü seçilebilen 4-bitlik bir sayaç'ın 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 12: VHDL ile 4-Bit Sayaç

3. FPGA Platformunda Tasarım Süreçleri

3.1. Genel Tasarım Adımları



Şekil 13: FPGA Tasarım Adımları

3.1.1. Design Entry

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

3.1.2. Behavioral Simulation

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

3.1.3. Design Synthesis

Tasarım girişinde HDL kullanılması durumunda gerçekleme yapılabilmesi için HDL'den sentezlemesi yapılır. Şematik tasarımda ise, hedeflenen cihazda bulunan temel birimler kullanıldığından için bu adıma gerek yoktur. Sentezleme sonucunda tasarlanan sistemi oluşturuan lojik blokları ve bunların bağlantılarını içeren bir netlist oluşturulur.

3.1.4. Post-Synthesis Behavioral Simulation

HDL sentezlendikten sonra elde edilen sistemin davranışsal simülasyonu yapılır ve tasarımın doğrulaması yapılır. Eğer düzeltme gerekirse tasarım girişinde değişikliler yapılır ve tekrar sentezlenir.

3.1.5. Design Implementation

FPGA platformunda gerçekleme işlemi sırayla, "Translate", "Map" ve "Place&Route(PAR)" olarak üç alt işlem olarak yapılır. "Translate" alt işlemi gerçeklemeye verilen netlist ve tasarım kısıtlamalarını (örneğin giriş/çıkış pinlerini) birleşitirir. "Map" işlemi tasarımı hedeflenen cihaz üzerinde bulunan kaynaklara eşleştirir."PAR" işlemi de tasarım zamanlama kısıtlarına göre tasarımı hedef cihaz üzerindeki yerleşimini ve yolları belirler.

3.1.6. Functional Simulation

Fonksiyonel simülasyon, "Translate" işleminden sonra, gerçeklemeden önce tasarımda bulunan lojik hataların tespit edilmesi için yapılır. Zamanlama bilgisi henüz belli olmadığından simülasyon testleri birim zaman gecikmesi ile yapılır.

3.1.7. Static Timing Analysis

Sabit Zamanlama analizi, "Map" işleminden sonra tasarımdaki lojik gecikmeleri inceleme imkanı sağlar. Her ne kadar bağlantı yol gecikmelerini içermese de, tasarımın lojik gecikmelerinden kaynaklanabilecek potansiyel hatalarının belirlenmesinde yardımcı olur.

3.1.8. Timing Simulation

Zamanlama Simülasyonu, PAR işlemi sonucunda belirlenen hedef cihaz konfigürasyonunun, standart lojik ve yol gecikmeleri ile beraber ifade edilen (Back Annotation) HDL üzerinden yapılır. Böylece tasarımın hedef cihaz üzerinde yol ve lojik gecikmeleri ile beraber simülasyonu ve doğrulaması yapılır.

3.1.9. Device Programming

Gerçeklenen tasarım programlama formatına dönüştürülerek hedef cihaz'ın konfigürasyonun saklandığı PROM'a yüklenir.

3.1.10. In-Circuit Verification

Hedef cihaza konfigürasyon yüklendikten sonra, yazılım araçları kullanarak çalışma zamanında tasarım doğrulaması yapılabilir.

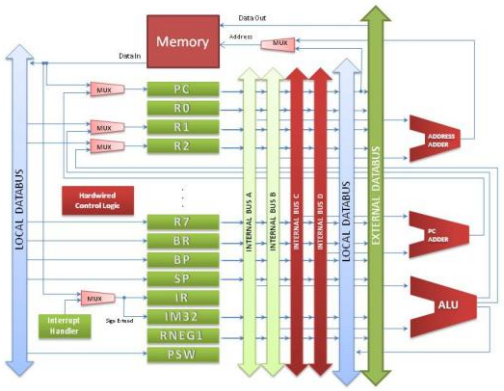
4. SelCPU işlemcisi ile Xilinx Spartan 3E FPGA üzerinde SOPC (System on A Programmable Chip) Uygulaması

4.1. SelCPU Soft Core Processor

SelCPU 32 bit RISC, von neuman mimarisinde, 7 değişik adresleme modu, 16-bit,32-bit immediate operand, multi-cycle, Memory Stack , 16 external interrupt, DMA, Memory Mapped I/O destekleyen, 9 genel amaçlı olmak üzere toplam 13 Registera sahip, tek komutta 3,2,1 veya 0 adres içerebilen bir işlemcidir. SelCPU işlemcisi Soft Core olarak (HDL ile ifade edilerek) Xilinx Spartan 3E FPGA üzerinde gerçeklenmiştir. Bazı üst düzey FPGA'ler içlerinde Hard Core (Silikona gömülü olarak) işlemci kaynakları da bulundurur. Örneğin, Xilinx Virtex5 FXT serisi FPGA'ler içlerinde 2 adet PowerPC 440 işlemci bloğu içerirler.

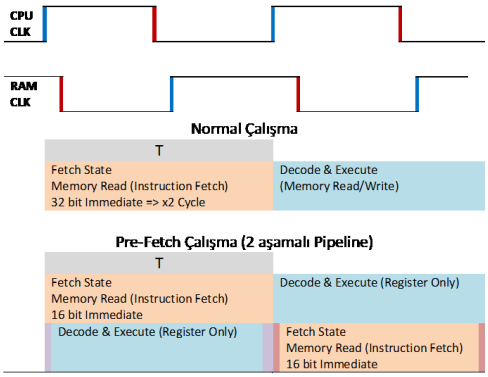
SelCPU her hangi bir RISC işlemci temel alınmadan özgün bir mimariyle tasarlanmıştır. İşlemci hafıza erişim komutları dışındaki komutlarda execute ile fetch işlemini aynı anda yapılabilmektedir. Ancak klasik anlamda pipeline processing ve cache memory kullanılmamıştır.

İşlemci çalışması ,özel durumlar(interrupt,halt) hariç, fetch&decode ve execute olarak iki temel evreden oluşmaktadır. Komutun içeriğine göre eğer 32 bit immediate varsa fetch evresi 2 cycle olabilmektedir. syscall,iret hariç diğer tüm Tüm komutlar ise 1-cylce'da çalışacak şekilde tasarlanmıştır.



Şekil 14: SelCPU İşlemci Mimarisi

bl (branch and link) dahil olmak üzere tüm program kontrol komutlarıda tek execution cycle ile çalıştırılabilmektedir. Mimari tasarımdan dolayı fetch ile execute aşamaları aynı cycle'da çalışmaktadır. Bunu yapabilmek için memory ile cpu clockları arasında faz farkı kullanılmaktadır.

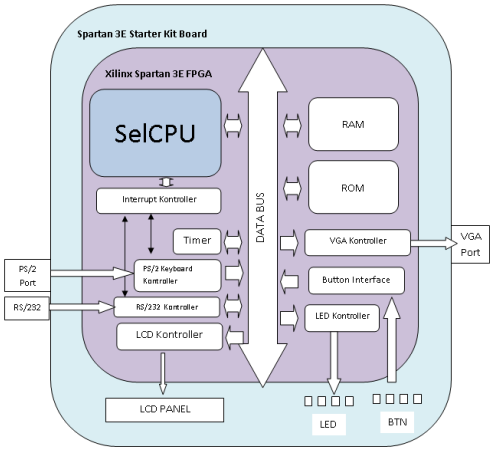


Şekil 15: SelCPU İşlemcisi Çalışma Zamanlaması

İşlemcide syscall, iret komutlar dışındaki komutların execute aşaması 1 cycle'dır. Hafıza erişimi yapan komutlar dışındaki komutlar execute aşamasında iken bir sonraki komutun hafızadan pre-fetch işlemi yapılır. Branch komutlarında da execute aşamasındaki sonuca göre (daha register'lara kaydedilmeden) branch prediction yapılır.

4.2. FPGA üzerinde SelCPU ile SOPC uygulaması

Sistemin tasarımı Verilog ile Xilinx ISE Foundation kullanılarak, uygulaması Xilinx Spartan3E FPGA Starter Kit üzerinde yapılmıştır. System-On-Chip uygulaması için Xilinx Spartan 3E FPGA Starter Kit'inde bulunan çevre birimleri (LCD Panel ,Ledler, P/S2 Keyboard Port ,RS-232 ,VGA ve tuşlar) direk olarak FPGA pinleri ile sürülecek şekildedir. Bu çevre birimleri için FPGA üzerinde denetleyiciler ve arayüzler tasarlanmıştır. Ayrıca daisy-chain interrupt denetleyicisi ve timer modülleri de FPGA içerisinde gerçeklenmiştir. Sistem de RAM ve ROM olarak FPGA de bulunan Block RAM/ROM kaynakları kullanılmıştır. Açılış ROM'unda olarak sistemin çevre birimlerine temel hizmetleri verecek bir BIOS yazılımı yüklüdür. Sistem üzerinde çalışacak programlar "SelCPU Assembly Cross Compiler" derlenip, FPGA konfigürasyonu ile ROM'a yüklenebileceği gibi, çalışma zamanında BIOS desteği ile RS/232 arayüzü ile aktarılabilmektedir.



Şekil 16: SelCPU İşlemcisi Çalışma Zamanlaması

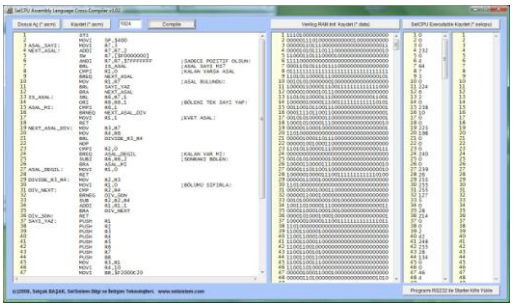
VGA çıkışı 640x480 çözünürlüktedir. Video buffer olarak FPGA kaynakları kısıtlı olduğundan ekram çıktısı olarak karakter modu oluşturulmuştur. Kullanılan karakter fontu ile 30 satır ve 80 satırdan oluşan bir ekran karakter ve renk bilgisi için buffer kullanılmıştır. Karakter jeneratörü ile ekran görüntülenmesi sağlanmıştır.

Şekil 17: Spartan 3E Starter Kit

LCD panel ile FPGA 4 bit arayüz ile bağlıdır. LCD panel kullanımı FSM (Sonlu Durum Makinesi) ile buffer üzerinden basılacak şekilde tasarlanmıştır. Böylece çok sade bir donanım arayüzü ile buffer hafıza adresine yazılan bilgi, LCD arayüzü ile LCD panele aktarılır.

Şekil 18: SelCPU SOPC Ekran Görüntüsü

PS/2 Klavye arayüzü, basılan tuşun scan kodunun yanında, FPGA üzerinde karakter dönüşümünü Türkçe olarak büyük/küçük harf ve shift tuşlarına göre yaparak karakter bufferına yazar. Böylece BIOS yazılımında Türkçe karakter dönüşümü yapmaya gerek kalmaz.



Şekil 19: SelCPU Assembly Cross Compiler

Sistemde donanım- yazılım arayüzü mümkün olduğunca sade şekilde tasarlanmıştır.

5. Sonuçlar

Bu sunumda FPGA üzerinde bir işlemci tasarımı ve uygulması anlatıldı. Gömülü sistem çözümlerinde mikroişlemci tabanlı çözümlere göre FPGA çok daha esnek ve güçlü ve paralel çalışabilecek çözümler sunabilir. FPGA'ler özellikle tasarımın sürekli olarak güncellenmesi gereken alanlarda ASIC'lere göre tasarım ve üretim maliyeti açısından büyük avantaj ve kolaylık sağlarlar. Daha karmaşık algoritmaları yazılım programlama dillerinde ifade edilerek bunların otomatik olarak HDL'le dönüştürülmesi ile karmaşık algoritmalarında FPGA üzerinde başarıyla uygulanması sağlanmıştır. Hızları gittikçe yükselen FPGA'ler ile DSP işlemcileri ile sağlanan performansın onlarca kat fazlası elde edilebilmektedir.

6. Kaynakça

[1] Karen Parnell, Nick Mehta. Introduction to Programmable Logic, Xilinx, 2004.

[2] Aldec, Enhanced Verilog Turtorial with Applications Rev.1.0, 1998

[3] Aldec, Enhanced VHDL Turtorial with Applications Rev.2.1 1998

[4] Selçuk Başak, FPGA Üzerinde SelCPU İşlemcisi İle System-On-Chip Uygulaması, GömSis Gömülü Sistemler ve Uygulamaları Sempozyumu, İTÜ Bilişim Enstitüsü, 2008