SelCPU

Version 1.02

Temmuz 2008

Proje Ekibi

Selçuk BAŞAK

Bilgisayar Bilimleri Mühendisi
(YTÜ 1998)

selcuk@selsistem.com

SelSistem Bilgi ve İletişim Teknolojileri

www.selsistem.com

Önsöz

SelCPU, Cpu-Turkey yarışmasına katılmak amacıyla tasarlamış bir işlemcidir. Projenin amacı yarışmanın özgün ve sade bir mimariye sahip bir işlemci tasarlamak, geliştirmek ve uygulamaya almaktır. Cpu-Turkey yarışması, yıllardır beklediğim işlemci tasarımı konusunda bir çalışma yapmama vesile olmuştur. Bu proje öncesinde Türkiye'deki piyasa koşullarında işlemci tasarımı yapmak ancak akademik çalışma için olabileceğini düşünüyordu. Ancak projede kazanmış olduğum deneyim, işlemci ve sistem tasarımının ticari olarak da Türkiye koşullarına uygun ve hatta bu şekilde sunabileceğiniz çözümlerin hayalleriniz ile sınırlı olduğunu gördüm. Bu anlamda böyle bir yarışma ile Türkiye'de teknoloji konusunda neler yapılabileceğini kamuoyuna göstermemize fırsat veren ve projemizi değerli görüp geliştirme kiti ve yazılım ile destek vermelerinden ötürü Cpu-Turkey ekibine teşekkür ederim.

Selçuk BAŞAK

Temmuz 2008

Özet

SelCPU, Cpu-Turkey yarışmasına katılmak amacıyla tasarlamış 32-bit bir işlemcidir. SelCPU, Yarışmanın temel sınırlama ve teknik özellikler temel alınarak tasarlanmıştır. SelCPU tasarlanırken, özgün ve sade bir mimariye sahip olması hedeflenmiştir. İşlemcimiz FGPA platformu üzerinde çalışacak şekilde geliştirilmiştir. İşlemciye ek olarak ram, rom, klavye, ekran, lcd, led, interrupt kontroller gibi çevre birimleri de proje kapsamında geliştirilerek, gerçek anlamda uygulamaya alınmıştır.

İçerik

Giriş

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 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.

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.

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.

Proje kapsamında SelCPU'u kullanan bir bilgisayar sistemi (SelSistem) geliştirilmiştir. SelSistem SelCPU ,Ram,ROM ve çevre birimi olarak LED,LCD,VGA,PS/2 Klavyesi olan temel bir bilgisayar sistemidir. İlerleyen süreçte SelSistem, SelCPU tabanlı bir kişisel bilgisayar sistemi olacaktır. Düşük hızlı ram kullanılması durumunda tasarıma cache eklenebilir.

Sistem Analizi

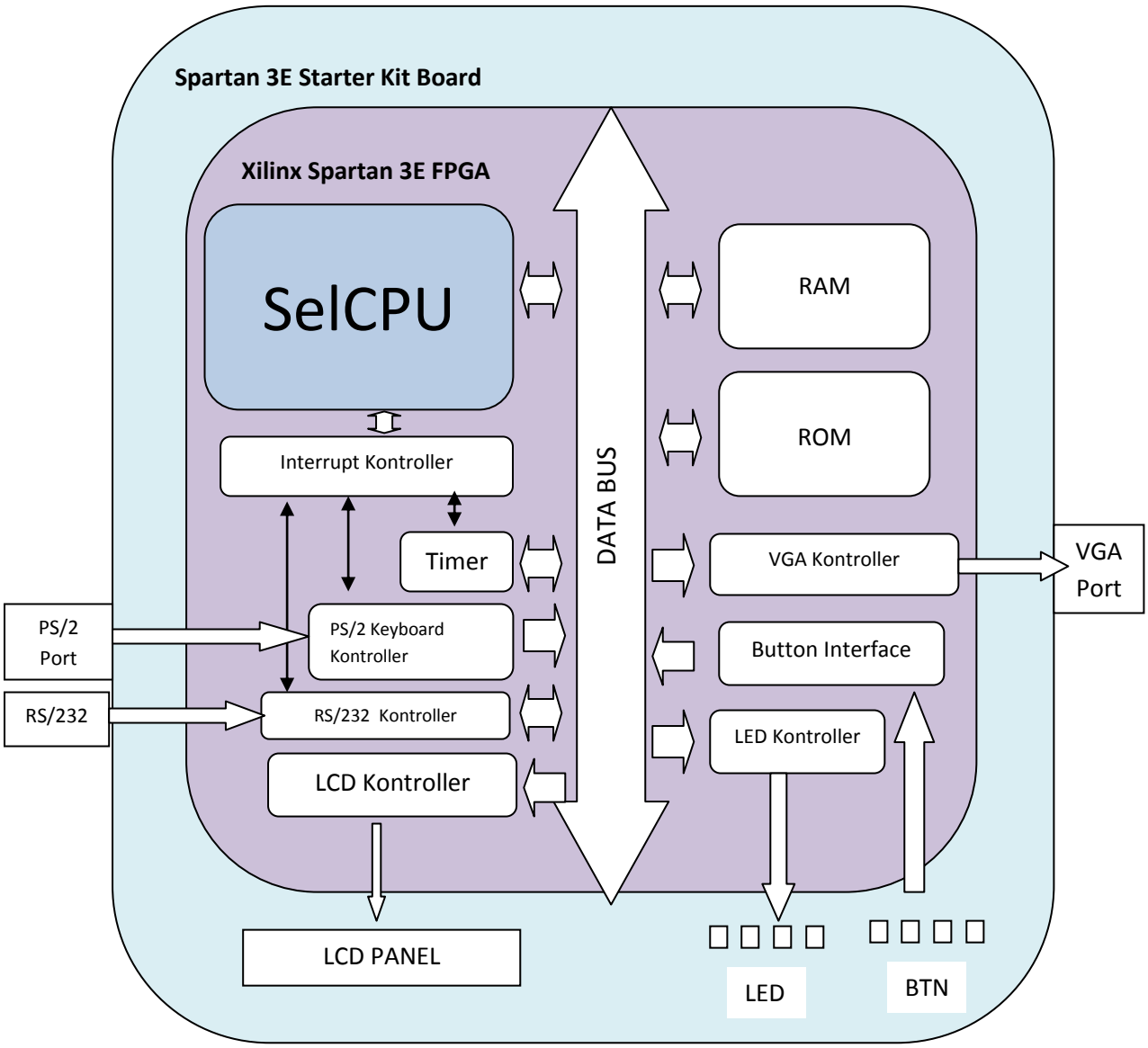
SelCPU tasarımı öncesinde yapılan araştırmada yarışmada belirtilen 16-bit işlemci tasarımı yeterli görülmeyip ileriye yönelik kullanılabilirlik açısından 32-bit işlemci olarak tasarlanmıştır. İşlemcinin özgün, sade ve tutarlı olması hedeflenmiştir.

Sistem Xilinx ISE Foundation 9.2i kullanılarak Verilog HDL ile Xilinx Spartan3E Starter Kit Platformu hedef alınmıştır. Simülasyon için ISE Simulator ve ModelSim araçları seçilmiştir.

Sistem taslak çalışmaları kâğıt üzerinde yapılarak değerlendirilmiş ve uygun taslak mimari belirlenmiştir.

SelSistem Mimarisi ve FPGA Platform

SelCPU Xilinx Spartan3E Starterkit FGPA platformu üzerinde geliştirilmiştir. Bu platformda Xilinx Spartan 3E FPGA ve Ethernet,ADC,PS/2,RS232,VGA, LED ,LCD,tuşlar birçok çevre birimi , DDR RAM, Flash Memory gibi hafıza birimi bulumaktadır. SelSistem olarak belirtiğimiz bilgisayar sistemi bu platformdaki çevre birimleri için kontroller ve RAM, ROM ve SelCPU'u içermektedir. RAM ve ROM olarak FPGA içerisinde bulunan Block RAM kaynakları kullanılmıştır. İleride RAM olarak DDR-RAM kullanılacaktır.



Şekil -1 : SelSistem Mimarisi

SelSistem/SelCPU Hafıza Organizasyonu

Hafıza 32 bit olarak adreslenmiş ve veri erişimi 32-bit word aligned olarak erişilecek şekilde tasarlanmıştır.
Maksimum adreslenebilir hafıza Flat 2^32 = 4G Word = 4*4 = 16 GB'tır.

Address Range	Purpose
0xF200 1000-0xF200 1FFF	Ekran Renk Buffer (6-Bit)
0xF200 0000-0xF200 0FFF	Ekran Karakter Buffer (8-Bit)
0xF100 0000-0xF100 001F	LCD Ekran Karakter Buffer (8-Bit)
0xF000 0020-0xF000 0027	Buttons (32 Bit)
0xF000 0010	RS232 Buffer & Status(32 Bit)
0xF000 0002	Keyboard Key & Status Word (32 Bit)
0xF000 0001	Timer Control Byte (32-Bit)
0xF000 0000	LED Control Byte (8-Bit)
0xF000 0000- 0xFFFF FFFF	I/O Device Mapped Memory Area
0x0001 0000- 0xEFFF FFFF	General Use (Defined by OS) (RAM)
0x0000 8400	Test ve uygulama kodları başlangıç adresi
0x0000 8000	SelCPU Sistem Başlangıç Adresi
0x0000 8000- 0x0000 FFFF	System Boot Strap ROM(BIOS or OS)
0x0000 0411	Saat Bilgisi
0x0000 0403	Maksimum Cursor Pozisyonu
0x0000 0402	Ekran Hafızası Başlangıç Addresi Pointerı
0x0000 0401	Cursor Posisyonu
0x0000 0400	Stack Başlangıcı
0x0000 0100- 0x0000 7FFF	System Information Area (Used by BIOS & OS) (RAM)
0x0000 0000- 0x0000 00FF	256 Interrupt Vektor Adresleri (RAM)

SelCPU Interrupt Vektörleri

Interrupt vektörleri 0-15 External Interrupt'lar için ,16-255 arası interruplar OS/BIOS/System Services için ayrılmıştır.

SelCPU Mimari Tasarımı

Sistem analizinde belirlenen taslak mimari üzerinde detaylı çalışılarak işlemcinin komut seti ve komut formatı ve kontrol logic belirlenmiştir. İşlemci, hardwired kontrol logic kullanılarak geliştirilmiştir. Mimari komut seti açısından RISC mimarisidir.

Kelime Uzunluğu: 32-Bit

Adresleme: 32 Bit

Adresleme Modları: 7 adet.
Implied ,Register, Register Indirect, Immediate(Direct),Indexed Register Adress Mode, Base Register Address, Relative Address (only for brunch)

I/O: Memory Mapped I/O Interface

Programmed I/O : Supported via Memory Mapped I/O Interface

DMA: Supported : Supported via Memory Mapped I/O Interface

CPU Control: Hardwired Control Logic, Multi-Cycle Execution with FSM
Overlapped Execute/Pre-Fetch Cycles with Branch Prediction

Registerlar:

R0	Zero Register (Constant)
R1	Accumulator
R2-R7	General Purpose Register
SP	Stack Pointer Register
BP	Current Stack Frame Base Pointer
BR	Base Register
PC	Program Counter
PSW	Program Status Word Z(zero),S(sign),V(overflow),C(Carry), B(Borrow), IE(Interrupts Enabled)
IR	Instruction Register
RNEG1	-1 Register (Constant)
IM32	Immediate Operand Register

Komut Formatı:

Komut [RD],[RS,[RX|Immediate]]

3 register, 2 register + 16bit Immediate ,2 register + 32bit Immediate, 2 register, Immediate,1 register, no address

Register-Register-Register (32Bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE 6 bit						0	RD				RS				RX				NOT USED (RESERVED)													R

Register-Register-Immediate 16

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE 6 bit						1	RD				RS				0	IMMEDIATE 16															

Register-Register-Immediate 32 - (opcode length:64 bit)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE 6 bit						1	RD				RS				1	NOT USED (RESERVED)															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IMMEDIATE32																															

No Operand Instructions

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE 6 bit						NOT USED (RESERVED)																									

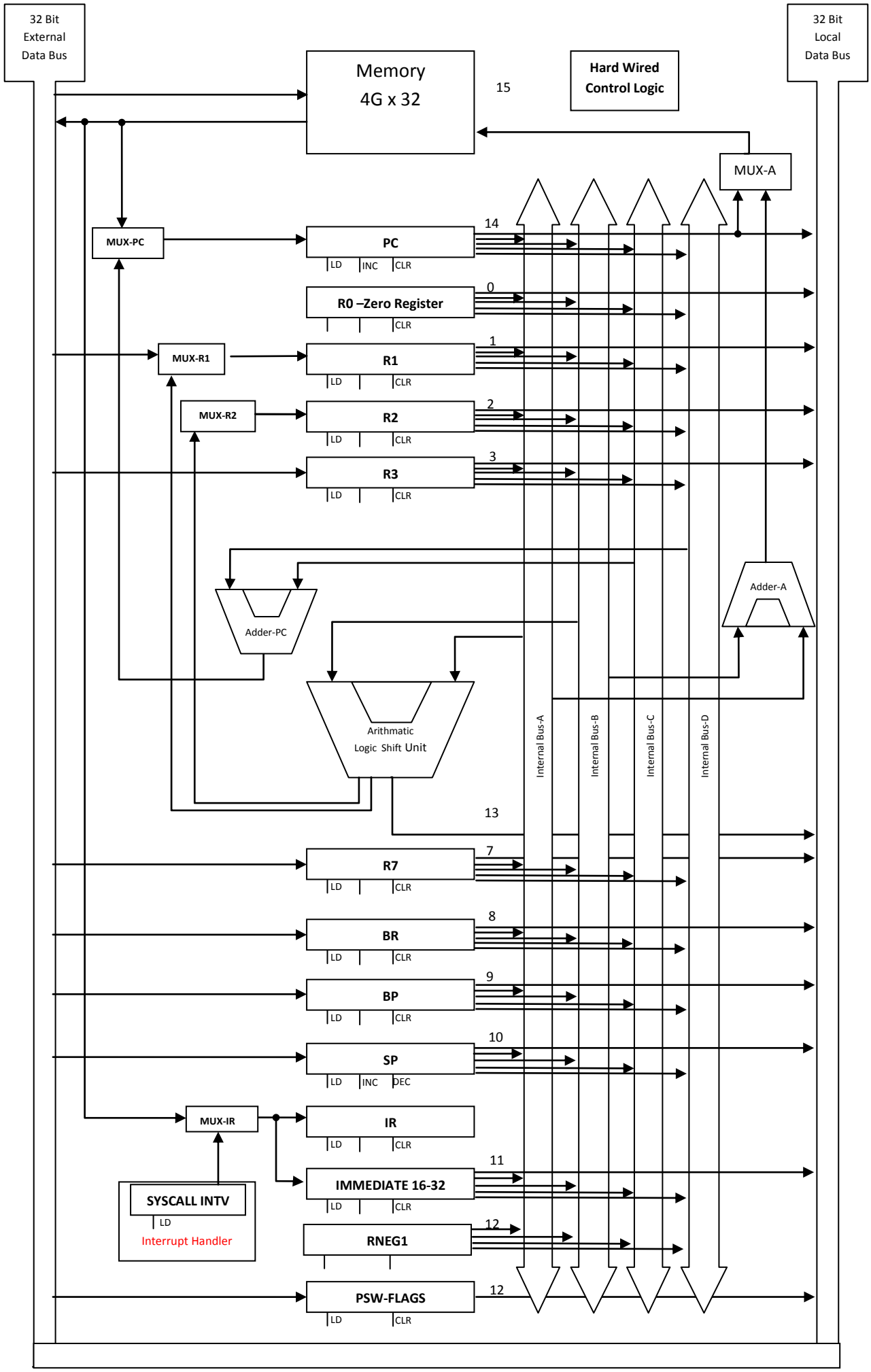
IR[25] : Immediate Exists
IR[25] &IR[16] : Immediate 32 instruction
IR[25] &IR[16]' : Immediate 16 instruction
IR[31] & IR[30] : Memory Access Instructions
IR[31:26] Opcode
RD = IR[24:21] Destination Register
RS = IR[20:17] Source Register
RX = IR[16:13] Second Source Register (if IR[25] = 0)

IM32_F: Immediate fetch flag
IR[0]:Register Relative RS+/-RX Mode
Adder-A.Mode = IR[25]' & IR[0] ;Mode=0 => Add; Mode=1 => Subtract
Adder-PC.Mode = IR[25]' & IR[0] ;Mode=0 => Add; Mode=1 => Subtract

CPU Instruction Cycles:

S0:
INSTRUCTION FETCH CYCLE (T=0 & IM32_F =0 & (IPF=0 | IE=0))
MUX-IR<-0, MUX-A <- 0, M.Read <- 1, DATABUS <-15, PC.INC <- 1, IR.LD <- 1, IM32.LD <- 1, IM32_F<- IR[25] & IR[16],T.INC <- (IR[25] & IR[16])' , MUX-IM16<- IR[25] & IR[16]' (IMMEDIATE[31:16]<- IMMEDIATE[15]), T.CLR<-0
S1:
IMMEDIATE32 FETCH CYCLE (T=0 & IM32_F =1)
MUX-IR<-0,MUX-A <- 0;M.Read <- 1,PC.INC <- 1,IM32.LD <-1, IM32_F <-0, T.INC <-1, S.CLR<-0
S2:
EXECUTE CYCLE (T=1)
< INSTRUCTION SPECIFIC CONTROL LOGIC >
S.INC <- (OPCODE =**syscall** | OPCODE =**iret**)
S.CLR<- (OPCODE =**syscall** | OPCODE =**iret**)'
S3:
EXECUTE CYCLE (T=2) syscall &iret- memory access
< INSTRUCTION SPECIFIC CONTROL LOGIC >
S.INC <- (OPCODE =**syscall**), T.CLR<- (OPCODE =**syscall**)'
S4:
EXECUTE CYCLE (T=3 : syscall- memory access
< INSTRUCTION SPECIFIC CONTROL LOGIC >
S.INC <-0,T.CLR<-1
S5:
INTERRUPT CYCLE (T=0 & IM32_F =0 & IPF=1 & IE=1)
IPF.CLR<-1,MUX-IR<-1,IR.LD<- 1,IM32.LD<- 1 (INTVR),T.INC <-1, T.CLR<-0
S6:
CPU HALT State
Waits until interrupt if enabled
S7:
RESET State

SelCPU Mimari Blok Şeması



Komut Seti

MOV

mov rd,rs

Registerlar arası veri transferi yapar

rd <= rs

MOVI

movi rd,immediate

Register'a sabit değer atar

rd<=immediate

ADD

add rd,rs,rx

Registerlar ile toplama işlemi yapar

rd<=rs+rx (Etkilenen Flagler: c,z,v,s)

ADDI

addi rd,rs,immediate

Register ile sabit değeri toplama işlemi yapar.

rd<=rs+immediate (Etkilenen Flagler: c,z,v,s)

SUB

sub rd,rs,rx

Registerlar ile çıkartma işlemi yapar.

rd<=rs-rx (Etkilenen Flagler: b,z,v,s)

SUBI

subi rd,rs,immediate

Register'dan sabit değeri çıkartma işlemi yapar,

rd<=rs-immediate (Etkilenen Flagler: b,z,v,s)

ADDC

addc rd,rs,rx

addc rd,rs,immediate

Add with carry işlemi yapar

rd<=rs+rx+c

rd<=rs+immediate+c (Etkilenen Flagler: c,z,v,s)

SUBB

subb rd,rs,rx

subb rd,rs,immediate

subtract with borrow işlemi yapar.

rd<=rs-rx-b

rd<=rs- immediate-b (Etkilenen Flagler: b,z,v,s)

MUL

mul rs,rx

Registerlar ile çarpma işlemi yapar. Sonucu [r2: r1] olarak r1,r2 registerlarına kaydeder.

[r2,r1]<=rs*rx

MULI

muli rs,immediate

Register ile sabit değer çarpma işlemi yapar.Sonucu [r2: r1] olarak r1,r2 registerlarına kaydeder.

r2,r1<=rs*immediate

MUL

mulu rs,rx

mulu rs,immediate

Unsigned çarpma işlemi yapar. Sonucu [r2: r1] olarak r1,r2 registerlarına kaydeder.

[r2,r1]<=rs*rx

[r2,r1]<=rs*immediate

AND

and rd,rs,rx

Registerlar ile bitwise and işlemi yapar.

rd<=rs &rx (Etkilenen Flagler: z)

ANDI

andi rd,rs,immediate

Register ile sabit değeri bitwise and işlemi yapar,
rd<=rs & immediate (Etkilenen Flagler: z)

OR

or rd,rs,rx

Registerlar ile bitwise or işlemi yapar.
rd<=rs | rx (Etkilenen Flagler: z)

ORI

ori rd,rs,immediate

Register ile sabit değeri bitwise or işlemi yapar,
rd<=rs | immediate (Etkilenen Flagler: z)

XOR

xor rd,rs,rx

Registerlar ile bitwise or işlemi yapar.
rd<=rs ^ rx (Etkilenen Flagler: z)

XORI

xori rd,rs,immediate

Register ile sabit değeri bitwise or işlemi yapar,
rd<=rs ^ immediate (Etkilenen Flagler: z)

NOT

not rd,rs

Registerlar'ın bitwise complementini alır
rd<=~rs (Etkilenen Flagler: z)

CMP

cmp rs,rx

Registerlar ile çıkartma işlemi yapar. Sonucu kaydetmez
rs-rx (Etkilenen Flagler: b,z,v,s)

CMPI

cmpi rs,immediate

Register'dan sabit değeri çıkartma işlemi yapar, Sonucu kaydetmez
rs-immediate (Etkilenen Flagler: b,z,v,s)

SLL

sll rd,rs,rx
sll rd,rs,immediate

Shift logical left işlemi yapar.
rd<= rs << rx
rd<= rs << immediate (Etkilenen Flagler: z)

SLA

sll rd,rs,rx
sll rd,rs,immediate

Shift arithmatic left işlemi yapar.
rd<= rs << rx
rd<= rs << immediate (Etkilenen Flagler: z)

SRL

sll rd,rs,rx
sll rd,rs,immediate

Shift logical right işlemi yapar.
rd<= rs >> rx
rd<= rs >> immediate (Etkilenen Flagler: z)

SRA

sll rd,rs,rx
sll rd,rs,immediate

Shift aritmatic right işlemi yapar.
rd<= rs << rx
rd<= rs << immediate (Etkilenen Flagler: z)

RLC

sll rd,rs

Rotate logical left with carry işlemi yapar.

rd<= {rs[30:1],c} , c<=rs[31] (Etkilenen Flagler: z)

RRC

sll rd,rs

Rotate Logical Right with carry işlemi yapar.

rd<= {c,rs[31:0]} , c<=rs[0] (Etkilenen Flagler: z)

CLV

clv

Clear overflow flag

v<=0

STV

stv

Set overflow flag

v<=1

CLB

clb

Clear barrow flag

b<=0

STB

stb

Set barrow flag

v<=1

CLC

clc

Clear carry flag

c<=0

STC

stc

Set carry flag

c<=1

BA

ba rs,rx
ba rs,im
ba rs
ba im

Branch always

BRA

bra label

Branch relative always

BEQ

beq rs,rx
beq rs,im
beq rs
beq im

Branch if equal (z=1)

BREQ

breq label

Branch relative if equal (z=1)

BNEQ

bneq rs,rx
bneq rs,im
bneq rs
bneq im

Branch if not equal (z=0)

BRNEQ

brneq label

Branch relative if not equal (z=0)

BC

bc rs,rx
bc rs,im
bc rs
bc im

Branch if carry set (c=1)

BRC

brc label

Branch relative if carry set (c=1)

BNC

bnc rs,rx
bnc rs,im
bnc rs
bnc im

Branch if carry clear (c=0)

BRNC

brnc label

Branch relative if carry clear (c=0)

BPOS

bpos rs,rx
bpos rs,im
bpos rs
bpos im

Branch if s=0 & z=0

BRPOS

brpos label

Branch relative if s=0 & z=0

BNEG

bneg rs,rx
bneg rs,im
bneg rs
bneg im

Branch if s=1

BRNEG

brneg label

Branch relative if s=1

BV

bv rs,rx
bv rs,im
bv rs
bv im

Branch if overflow (v=1)

BRV

brv label

Branch relative if overflow (v=1)

BGT

bgt rs,rx
bgt rs,im
bgt rs
bgt im

Branch if greater

BRGT

brgt label

Branch relative if greater

BGTE

bgte rs,rx
bgte rs,im
bgte rs
bgte im

Branch if greater or equal

BRGTE

brgte label

Branch relative if greater or equal

BLT

blt rs,rx
blt rs,im
blt rs
blt im

Branch if less

BRLT

brlt label

Branch relative if less

BLTE

blte rs,rx
blte rs,im
blte rs
blte im

Branch if less or equal

BRLTE

brlte label

Branch relative if less or equal

BHI

bhi rs,rx
bhi rs,im
bhi rs
bhi im

Branch if higher, unsigned comparison

BRHI

brhi label

Branch relative if higher, unsigned comparison

BHIE

bhie rs,rx
bhie rs,im
bhie rs
bhie im

Branch if higher or equal, unsigned comparison

BRHIE

brhie label

Branch relative if higher or equal, unsigned comparison

BLO

blo rs,rx
blo rs,im
blo rs
blo im

Branch if lower, unsigned comparison

BRLO

brlo label

Branch relative if lower, unsigned comparison

BLOE

bloe rs,rx
bloe rs,im
bloe rs
bloe im

Branch if lower or equal, unsigned comparison

BRLOE

brloe label

Branch relative if lower or equal, unsigned comparison

NOP

nop

no operation

STI

sti

enable interrupts

CLI

cli

disable interrupts

HLT

hlt

İşlemciyi durdur.

NOP

nop

no operation

BL

bl rs,rx
bl rs,im
bl rs
bl im

Branch & link , subroutine call

BRL

brl label

Branch relative & link , subroutine call

LW

lw rd

load register from memory
rd<=memory

SW

sw rd

store register to memory
memory <= rd

PUSH

push rd

push register to memory stack
stack memory<=rd

POP

pop rd

pop register from memory stack
rd<= stack memory

RET

ret

return from subroutine

SYSCALL

Syscall im

call system service no: im

IRET

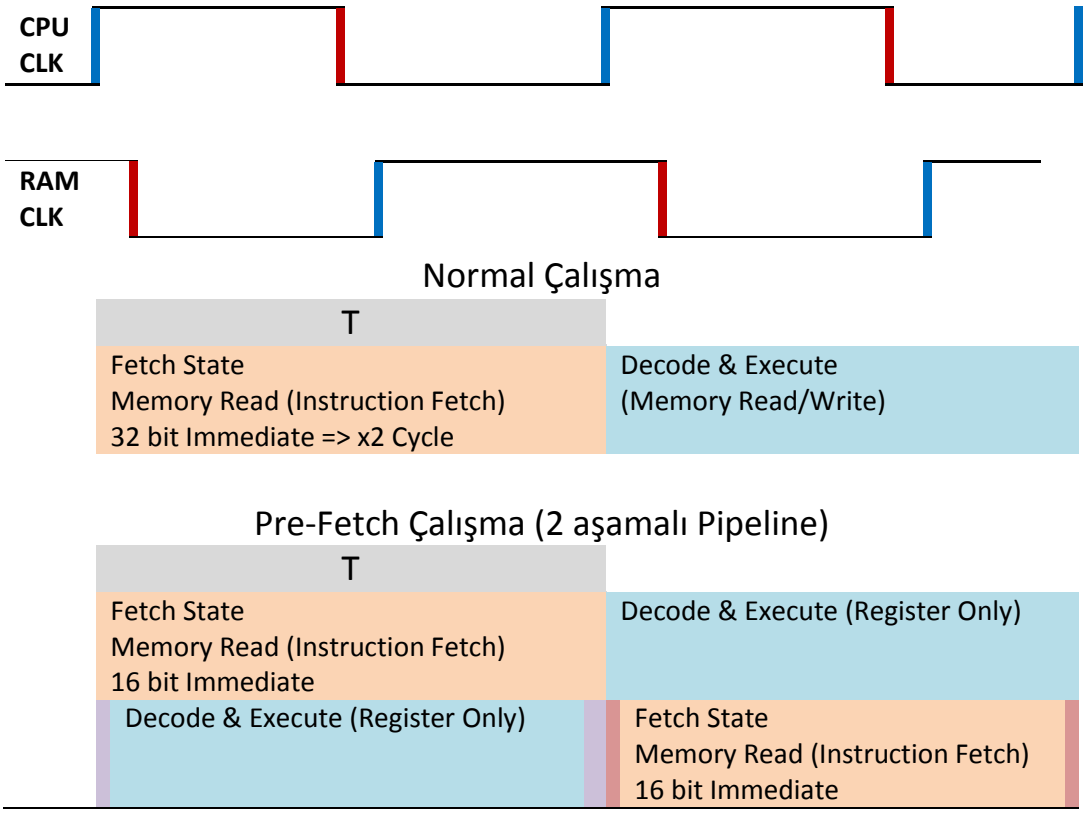
iret

return from interrupt service routine

Veri Transfer Zamanlaması

RAM/ROM erişimi negatif geçişte(Kırmızı), diğer veri transferleri ise pozitif (mavi) geçişte yapılmaktadır.

Zamanlama diyagramı:



Çalışmakta olan komut hafıza erişimi olan bir komut değil ise, Decode & Execute sırasında bir sonraki PC hesaplanıp (eğer branch var ise hedef konumu belirlenip) pre-Fetch yapılır. Böylece bir komut bir clock'ta çalıştırılabilir. Eğer komut 32 bit immediate 32 değer içeriyor ise doğal olarak Fetch iki cycle da gerçekleşir. Hafıza erişimi olan kumutlar ise pre-fetch yapamaz ve execute sonrasındaki cycle da fetch işlemi yapılır. Bu şekilde işlemcinin tüm bölümleri aynı anda efektif olarak kullanılmaya çalışılır. İşlemcinin çalışmasında hazard control gereği oluşmaz ve bu nedenle çok aşamalı pipeline processing'e göre avantajlı ve sadedir.

Verilog Modülleri

SelSistem.v
SelSistem bilgisayar sisteminin en üst modülüdür.

SelCPU.v
İşlemcinin register,bus ve mux'larını ve address adderlarını içerir.

Alu.v
Aritmetik,logic ve shift işlemlerini gerçekleştirir.

Control_logic.v
CPU çalışmasını FSM ile kontrol eder.

Led_controller.v
LED kontrolünü sağlar.

Lcd_controller.v
LCD'yi konfigurasyonunu yapar ve lcd çıktısı sağlar.

Vga_controller.v
Analog VGA (HYSNC,VSYNC,R,G,B) 640x800 60Hz sinyali üretir ve donanım olara karakter generator ile text mode console sağlar. 8 renk desteği vardır. Text mode ekran 80x30 olarak kullanılır. Font Tipi olarak lucida console kullanılmıştır. Türkçe desteği vardır. VGA sinyalleri, font ve başlangıç ekranı için ilk değerler geliştirdiğimiz bir program ile otomatik olarak hazırlanmıştır.SYNC sinyalleri için zamanlama hazırlanmış olan bu ilk değerleri indeksleyerek yapılmıştır.Lucida Console fontu bu program ile taranarak karakter generatore ilk değer olarak atanmıştır.

Ps2_keyboard_controller.v
PS/2 Türkçe Q klavye için native destek sağlar. Scan kodlarının karaktere dönüşümü, capslock,numlock,scroll lock gibi durum bilgisi, diğer shift karakterlerinin bilgisini 32-bit olarak paketler. Böylece sistemde çalışacak programın ayrıca dönüşüm işlemi yapmasına gerek kalmamıştır. Interrupt kontrollerına 1 nolu cihaz olarak bağlanmış ve tuşa basıldığında interrupt üretmektedir. Interrupt servisinde bu tuş okunmakta ve ekrana yazılmaktadır.

Status Byte:3 Shift State Byte:2 Scan Code:Byte1 Karakter:Byte1

Timer.v:
Timer 500ms periyodunda interrupt üretir.Interrupt kontrollerına 0 nolu cihaz olarak bağlıdır.
Interrupt servisi çalışma süresini sayar , ekrana ve LCD'ye yazar

Rs232_controller.v
Rs232 ile veri alış verişi yapmayı sağlar. Veri alışta interrupt desteği vardır. Göndermeden önce transmit buffer'ı kontrol edilebiliyor böylece üst üste yazma engelenebiliyor. Örnek sistemde rs232 üzerinden hafızaya program yüklenebiliyor.

Interrupt_controller.v
Daisy chain interrupt controller olarak tasarlandı. En yüksek öncelikli 0 en düşük öncelikli 15 olarak16 cihaz desteği vardır.

Selsistem.ucf
FPGA bağlantı ve clock için user contraintlerini içerir.

Testler

Sistemin behavioral simülasyonu Önce Xilinx ISE Foundation 9.2i Xilinx ISE Simulator kullanılarak yapılmıştır. Sonra ise hem behavioral hem de FPGA için Post-PAR simulasyonları ModelSim ile yapılmıştır.

Simulasyon için verilog test bench (test_v.v) oluşturularak clock sinyali oluşturularak yapılmıştır.

SelCPU Assembler ile test için binary instruction datası oluşturulup. “Selsistem.v” içinde ROM’a initial değer olarak atanmıştır.

Test için veriler “selsistem.v” içinde ROM’a external dosyadan aktarılarak ile yapılmıştır.

SelCPU klasöründe test.data ve bootstrap_code.data olarak iki adet veri dosyası vardır. “bootstrap_code.data” sistemin resetlendiğinde çalışacağı 0x00008000 adresine yüklenir. Bu dosya öncelikle interrupt vektörlerini örnek interrupt handlerlara atamasını yapar. Memory stack ayarlarını yapar (bu örnekte SP<= 0x000000400). Sonra kontrolü test kodlarının başlayacağı 0x00008400 adresine verir.
“test.data” dosyası 0x00008400 adresine yüklenir. Kullanıcı kodları buradan çalışmaya başlar. Örnek dosyada her bir komut grubuna ait birer işlem hazırlanmış ve testleri yapılarak işlemcinin çalışması denetlenmiştir.

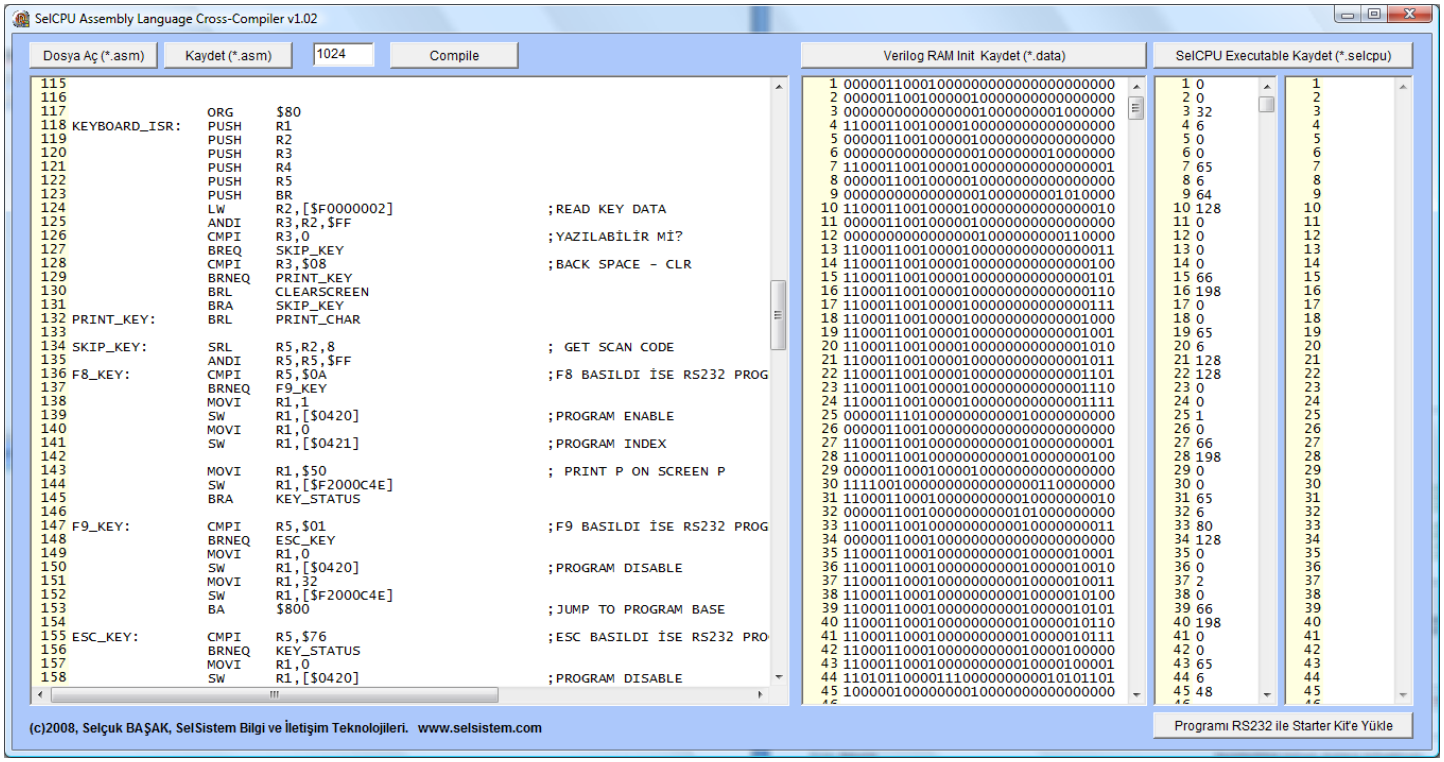
Farklı testler için SelCPU Assembler ile program yazıp simülasyon dosyası “test.data” olarak SelCPU klasörüne kopyalayarak yapılabilir.

Test ve BootStrap kodları SelCPU Assembler Aracı kullarak geliştirilmiştir. SelCPU Assembler Aracı gerçek anlamda program geliştirmek için çok kullanışlı değildir ancak test uygulaması geliştirmek için dikkatli olarak kullanılmalıdır. SelCPU Assembler Komutların operand kontrolünü yapmamaktadır.İleride daha gelişmiş bir assembler ve c/c++ compiler geliştirilecektir.

Örnek test uygulaması timer,klavye,lcd,led ve ekran arayüzünü kullanan, basit bir terminal uygulamasıdır. Ekrandan klavye ile giriş yapılabilmekte ve aynı zamanda program asal sayı araması yapar ve bulduğu asal sayıları ekrana basar. Çalışma süresi ve klavye durum bilgisi de ekrana yazılır.

Testlerde performas arttırmak amacıyla bu tasarımında mux yerine tri-state bus denenmiştir. Ancak kullanılan FPGA de tri-state , logic pull up ile gerçekleştirilmektedir. Sentezleme sonucunda daha kötü performans alınmıştır. Bu nedenle mux kullanımıştır. Farklı bir platformda data buslar tri-state ile denenebilir.

SelCPU Assembly Cross-Compiler




```
                ORG     $80
KEYBOARD_ISR:   PUSH    R1
                PUSH    R2
                PUSH    R3
                PUSH    R4
                PUSH    R5
                PUSH    BR
                LW      R2,[$F0000002]          ;READ KEY DATA
                ANDI    R3,R2,$FF
                CMPI    R3,0                    ;YAZILABİLİR Mİ?
                BREQ    SKIP_KEY
                CMPI    R3,$08                  ;BACK SPACE - CLR
                BRNEQ   PRINT_KEY
                BRL     CLEARSCREEN
                BRA     SKIP_KEY
PRINT_KEY:      BRL     PRINT_CHAR

SKIP_KEY:       SRL     R5,R2,8                 ; GET SCAN CODE
                ANDI    R5,R5,$FF
F8_KEY:         CMPI    R5,$0A                  ;F8 BASILDI İSE RS232 PROG
                BRNEQ   F9_KEY
                MOVI    R1,1
                SW      R1,[$0420]              ;PROGRAM ENABLE
                MOVI    R1,0
                SW      R1,[$0421]              ;PROGRAM INDEX

                MOVI    R1,$50                  ; PRINT P ON SCREEN P
                SW      R1,[$F2000C4E]
                BRA     KEY_STATUS

F9_KEY:         CMPI    R5,$01                  ;F9 BASILDI İSE RS232 PROG
                BRNEQ   ESC_KEY
                MOVI    R1,0
                SW      R1,[$0420]              ;PROGRAM DISABLE
                MOVI    R1,32
                SW      R1,[$F2000C4E]
                BA      $800                    ;JUMP TO PROGRAM BASE

ESC_KEY:        CMPI    R5,$76                  ;ESC BASILDI İSE RS232 PRO
                BRNEQ   KEY_STATUS
                MOVI    R1,0
                SW      R1,[$0420]              ;PROGRAM DISABLE
```

```
00000110001000000000000000000000
00000110010000010000000000000000
00000000000000001000000001000000
11000110010000100000000000000000
00000110010000010000000000000000
00000000000000010000000010000000
00000110010000010000000000000000
11000110010000100000000000000001
00000110010000010000000000000000
11000110010000100000000000000010
00000110010000010000000000000000
00000000000000010000000110000000
11000110010000100000000000000011
11000110010000100000000000000100
11000110010000100000000000000101
11000110010000100000000000000110
11000110010000100000000000000111
11000110010000010000000000001000
11000110010000100000000000001001
11000110010000100000000000001010
11000110010000100000000000001011
11000110010000100000000000001101
11000110010000100000000000001110
11000110010000100000000000001111
00000111010000000000010000000000
00000110010000000000000000000000
11000110010000000000010000000001
11000110010000000000010000000100
00000110001000010000000000000000
11111001000000000000000110000000
11000110001000000000010000000010
00000110010000000000010100000000
11000110010000000000010000000011
00000110010000000000000000000000
11000110001000000000010000010001
11000110001000000000010000010010
11000110001000000000010000010011
11000110001000000000010000010100
11000110001000000000010000010101
11000110001000000000010000010110
11000110001000000000010000010111
11000110001000000000010000100000
11000110010000000000010000100001
11010110000111000000000010101101
10000010000000010000000000000000
```

Program geliştirmek için geliştirmiş olduğumuz assembly cross compiler aracı ile hazırlanan programlar derlenebiliyor.Çıktı olarak Verilog RAM initial değer dosyası hazırlayabiliyor. RS232 üzerinden direk yüklenebilecek kod üretimi yapılıyor ve sisteme rs232 portu ile program yüklemesini de yapılabiliyor.

SelCPU Assembler DD (Define 32 bit Data) ve ORG (Set Offset) ifadelerini de desteklemektedir.

Proje Geliştirme Süreci

Kullanılan FPGA Modeli : Xilinx Spartan 3E xc3s500e-4fg320

Proje için harcanan toplam zaman aşağıda belirtilmiştir. (ön araştırmalar hariç)

1-2 gün genel mimari tasarım

3-4 gün detaylı mimari tasarım

4-5 gün verilog ile selcpu tasarımın gerçeklenmesi , testi ve assembler aracının geliştirilmesi. örnek test programı geliştirme ve testleri

10-15 gün verilog ile selsistem çevre birimlerinin tasarımın gerçeklenmesi ,optimizasyonu ve testleri .

Günde yaklaşık olarak 19-20 saat çalışılmıştır.

Proje Aşamaları:

1- En Üst Düzey Mimari Tasarımı ve Blok Şeması
2- Register Organizasyonu
3- Hafıza Organizasyonu
4- Adresleme Modları
5- Instruction Set Tasarımı
6- CPU Instruction Cycles
7- Interrupt Handling
8- Register Transfer Functions
9- Control Logic Unit Tasarımı
10- ALU Tasarımı
11- VGA Kontroller
12- LCD Kontroller
13- LED Kontroller
14- Timer
15- Interrupt Kontroller
16- PS/2 Keyboard Interface Kontroller
17- Verilog HDL ile Tasarım, Sentezleme ve Simulasyon
18- Program Geliştirme Aracı - Assembler
19- Pipeline execution(pre-fetch)
20- RS232 Kontroller
21- Button Kontroller

Eksik Özellikleri:

Associative Cache Memory
Virtual Memory Support
Memory Protection
Privileged Instruction Protection

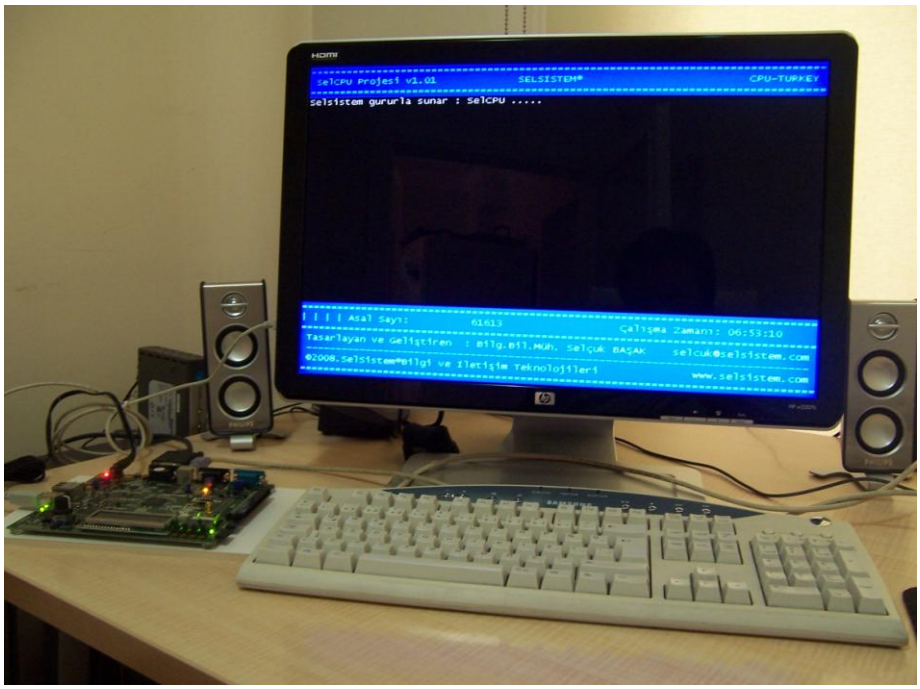
Sonuç

SelCPU tasarlandı ve gerçeklendi ve yukarıda belirtilen SelSistem bilgisayar sistemi üzerinde başarılı bir şekilde uygulamaya alındı.

SelCPU FPGA üzerinde sentezleme ve place & route sonucunda kullanılan platformda maksimum 25 Mhz ile çalışabilmektedir.
Kullanılan platformdaki 50 MHz'lik bir clock frekansı ikiye bölünerek işlemci 25Mhz ile çalıştırılmıştır. Ayrıca FPGA üzerinde bulunan DCM kullanılarak ay frekansta +27ns faz farklı ikinci bir clock'ta RAM ve çevre birimleri için üretilmiştir. VGA 640x480 sinyal üretimi için de 25 MHz frekans gerekmektedir.

İşlemcide Cpu-Turkey projesi için belirtilen komut seti ve ek komutlar birlikte toplam 77 adet komut geliştirildi. Asıl olarak 77 komut olsa da assembler ile INC/DEC gibi genel psudo komutlarda sisteme dahil edilebilir. (Ör: INC R1 ⇔ ADD R1,R1,1)

İleride SelCPU ve SelCPU ile çalışan SelSistem bilgisayar sistemine geliştirilerek ve bir kişisel bilgisayara dönüştürülecektir. Sistem için Geliştirilecek veya adopte ediecek bir C/C++ compiler'ı da yapılacaktır.



SelSistem Ekran Görüntüleri

1- M.MORRIS MANO : Computer System Architecture , Prentice Hall, 1993

2- EVITA Verilog Turtorial,ALDEC,1996

3- Spartan-3E Starter Kit Board User Guide

4- Xilinx ISE Software Manuals

5- Cpu-Turkey, http://www.cpu-turkey.com/downloads/