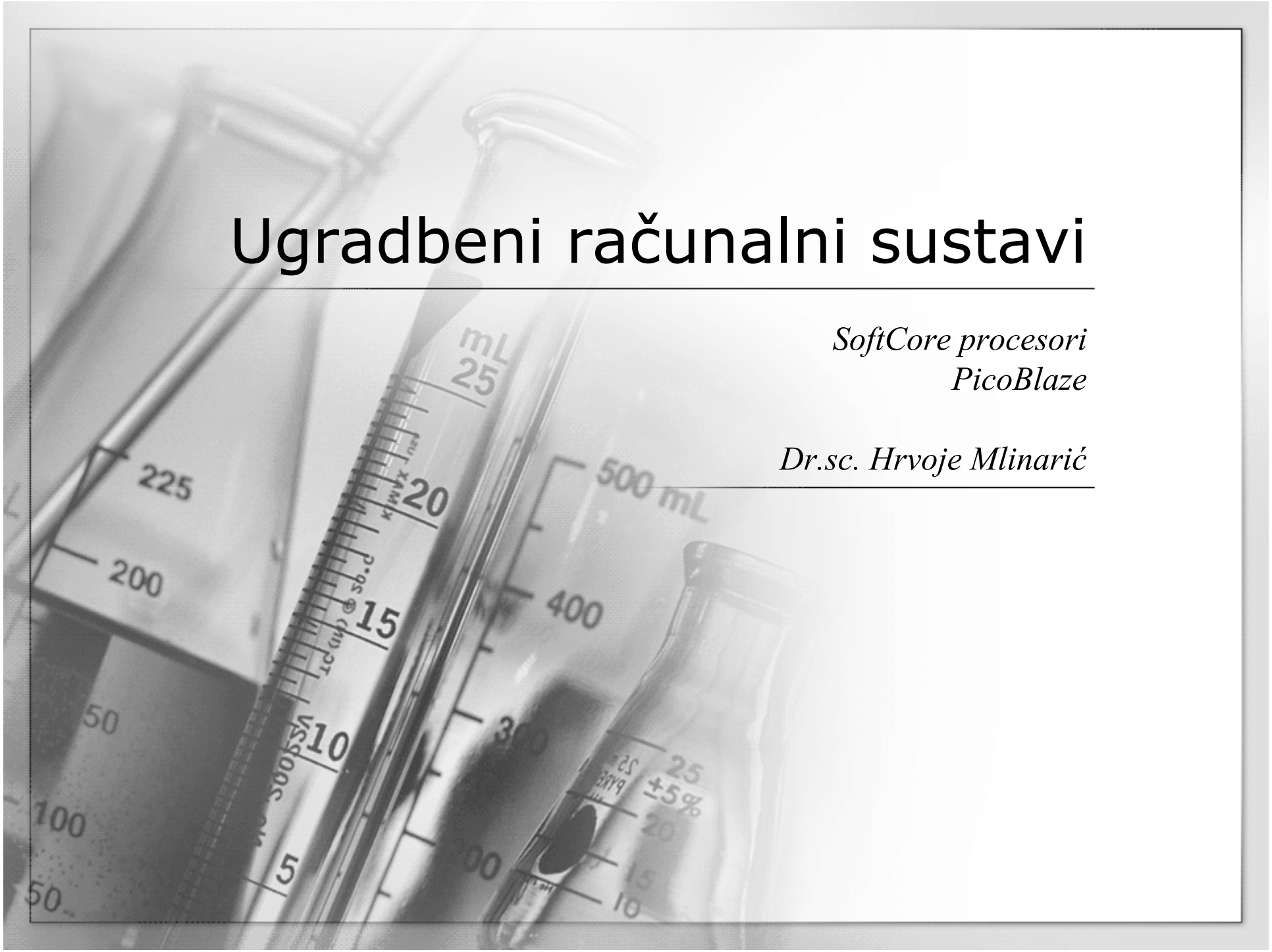


Ugradbeni računalni sustavi

SoftCore procesori
PicoBlaze

Dr.sc. Hrvoje Mlinarić

Dizajn ugradbenih sklopova

- Svaki ugradbeni računalni sustav ima nekakav procesor koji vrši upravljanje i kontrolu
- Postoje tri moguća rješenja izvedbe takvog procesora
  - Discrete processor
  - Hard processor Core
  - Soft Procesor Core
- Ovisnosti o potrebama na nama je da se odlučimo za jedno od navedena tri rješenja

Diskretni procesori

- Diskretni ili “off the shelf”
- Tradicionalan pristup
- Veliki izbor različitih proizvođača
- Veliki izbor različite funkcionalnosti
- Izvedeni kako ASIC sklopovi koji osim samoga procesora uglavnom sadrže i neke periferije
- Ispravan odabir odgovarajućeg procesora može biti vrlo zahtijevan zadatak

Hard Processor Core

- Posebni namjenski sklopovi unutar komponente (uglavnom FPGA)
- S obzirom na izvedbu mogu raditi na velikim brzinama
- Prednost im je što postoje u okruženju gdje se njegova okolina može jednostavno prilagoditi potrebama dizajna.
- Ne omogućavaju prilagodbu procesora dizajnu
- Samo pojedini članovi porodica FPGA sklopova imaju takvu mogućnost
- Vrlo ograničen broj proizvođača i FPGA sklopova

Soft Processor Core

- SoftCore procesori
- U potpunosti izveden u logici
- Zbog toga ne može raditi na maksimalnim brzinama samog sklopa
- Uglavnom višestruko sporiji od diskretnih i “hard” procesora
- Uglavnom pogodni za sustave gdje brzina izvođenja algoritma nije presudna
- Kontrola GPIO
- Iako nije ograničena na samo takvu primjenu.

Soft Processor Core

- Zašto koristiti SoftCore procesore (SCP)?
  - Omogućavaju vrlo veliku fleksibilnost
  - Tijekom izrade dizajna vrlo česta situacija je da se pojavljuju nove potrebe i novi zahtjevi – korištenjem SCP možemo jednostavno riješiti takve probleme
  - Vrlo jednostavno dodavanje novih periferija u sustav
  - Jednostavna izrada dodatnih potrebama prilagođenih periferija
  - U krajnjem slučaju moguće je mijenjati i samu arhitekturu SCP i prilagođavati je našim potrebama
  - U konačnici proces izrade se višestruko ubrzava (vrijeme je novac ☺)
  - Više procesora u jednom sustavu

XILINX SCP

- Za XILINX FPGA sklopove postoji velik broj procesorskih arhitektura koje su izvedene u FPGA logici
  - 8051 arhitekture
  - Z80 arhitekture
  - PIC arhitekture
  - ATMEL arhitekture
  - FRISC ☺
  - I mnoge druge

XILINX SCP

- Za potrebe svojih korisnika XILINX je razvio dva SCP koja su optimizirani za XILINX FPGA sklopove
  - MicroBlaze
    - 32-bitni RISC mikroprocesor
    - Podržan preko EDK alata koji nisu besplatni
  - PicoBlaze
    - 8-bitni RISC mikroprocesor
    - Vrlo jednostavan mikroprocesor
    - U potpunosti besplatan za korištenje

PicoBlaze – KCPSM3

- 8-bitni mikro-kontroler
- Prilagođen za:
  - Spartan-3
  - Virtex-II
  - Virtex-II Pro
- Jednostavna RISC arhitektura
- Potpuno besplatan
- (K)Constant (C)Code (P)Programmable (S)State (M)Machine

PicoBlaze – KCPSM3

- Dizajn zauzima svega 96 Slices na Spartan-3 sklopu
  - 5% - XC3S200
  - 2% - XC3S500 (lab. vježbe)
  - 0,3% - XCS5000

- XST izvješće
  - LUT1 : 2
  - LUT2 : 6
  - LUT3 : 68
  - LUT4 : 33

  -109 LUT = 55 slice

  - MUX : 48
  - XORCY : 35

  -Ne koriste se LUT tablice

  - FD : 76

  - Ne koriste se LUT tablice

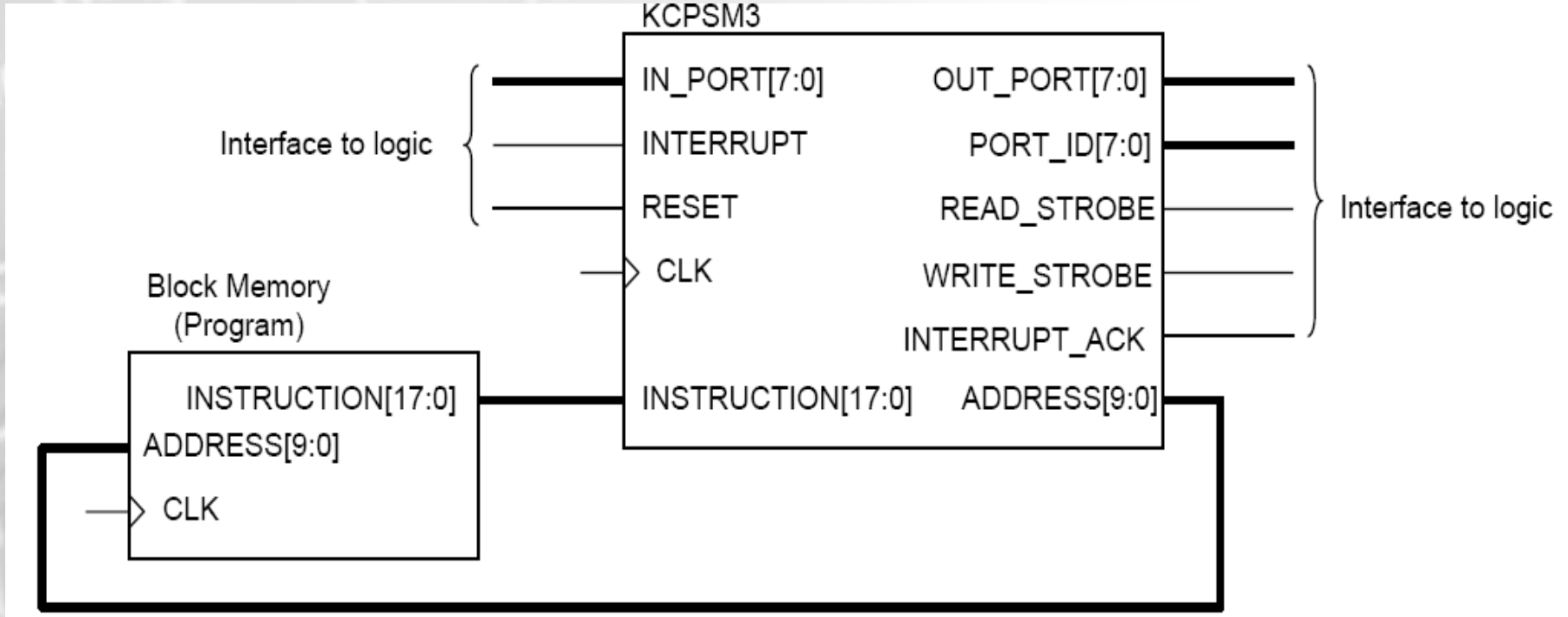
  - RAM16x1D : 8
  - RAM32X1S : 10
  - RAM64X1S : 16

  -34 slice

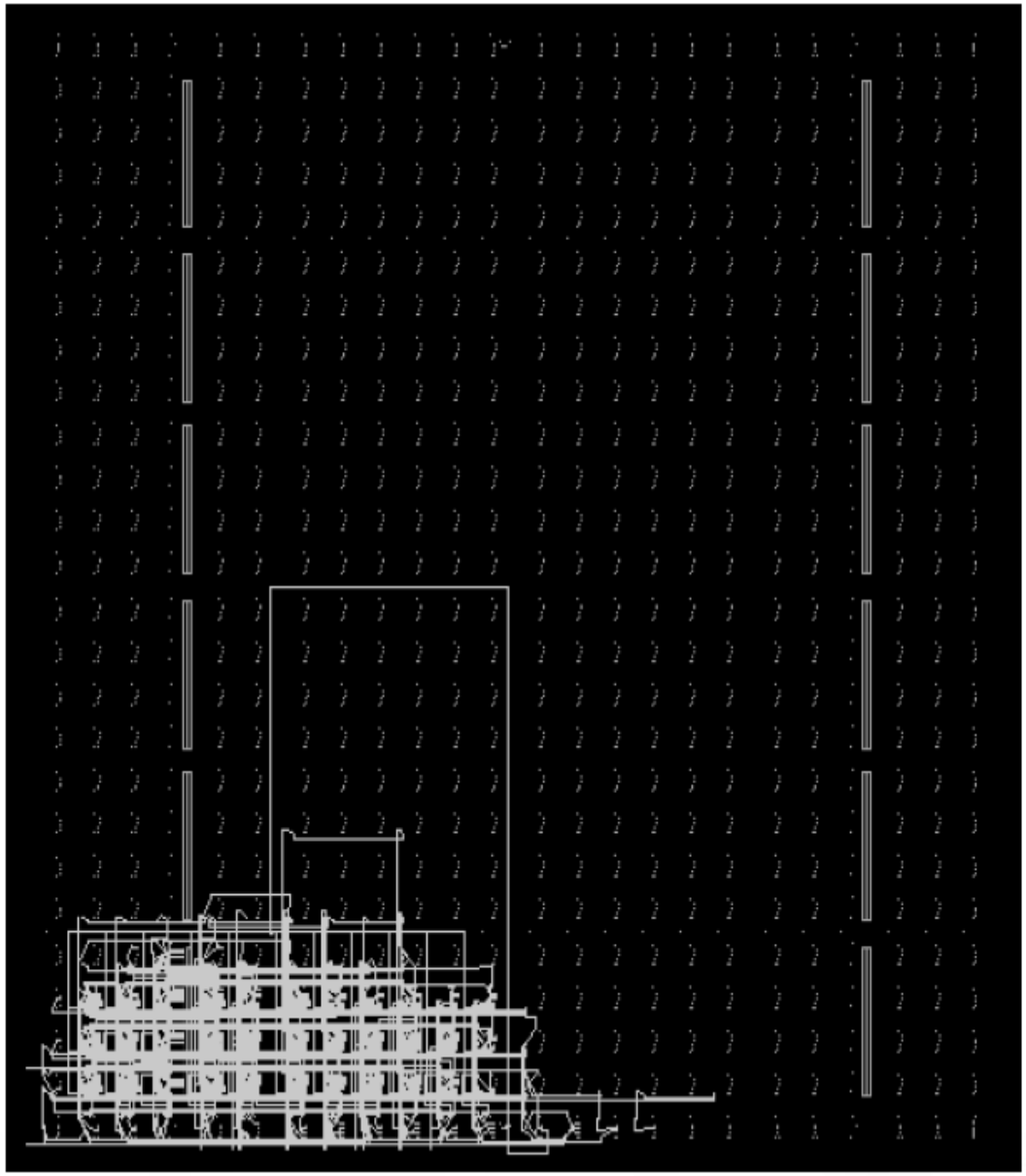
  = Ukupno 96 slice

PicoBlaze – KCPSM3

- Brzina 43 – 66 MIPS
- Koristi jedan blok RAM za memoriju (1024 instrukcija)
- U potpunosti implementiran nije potrebno dodavati vanjske komponente



PicoBlaze – KCPSM3



This plot from the FPGA Editor viewer shows the macro in isolation within the XC3S200 Spartan-3 device.

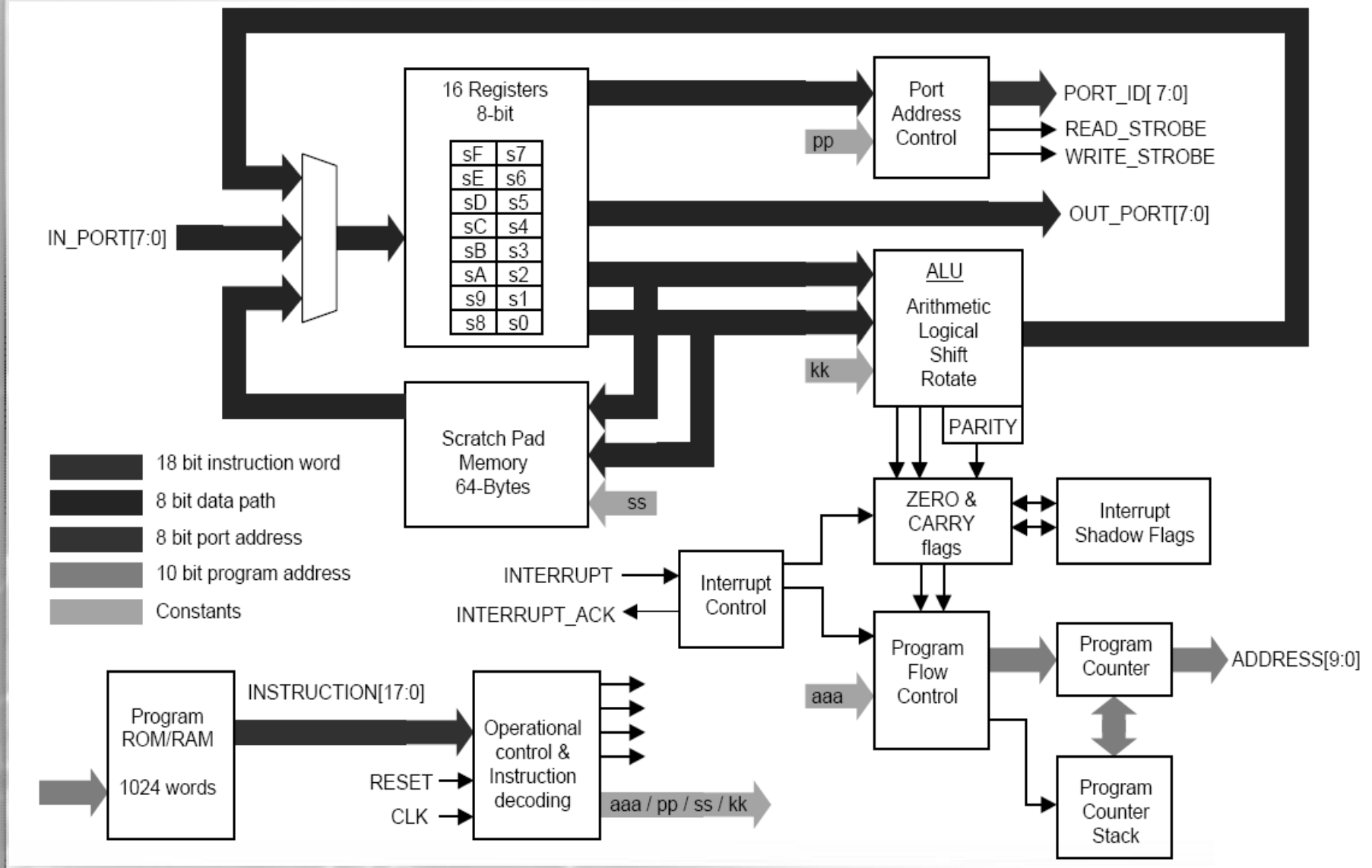
96 Slices

5% of XC3S200 Spartan-3 device

~87MHz in -4 Speed Grade

~43.5 MIPS

Arhitektura



Arhitektura

- Organizacija memorije – Harvardska

- Programska memorija 1024 instrukcija
  - U koliko trebamo više memorije preporuka je da se program podjeli u dva dijela i da se koriste dva ili više procesora.
  - Za memorijski zahtjevne programe koristiti neki drugi SoftCore procesor.

Arhitektura

- Radna memorija
  - 16 registara opće namjene
    - 8 bitni registri
    - Oznake: s0,s1,s2, … ,sA,sB,sC,sD,sE,sF
    - Svi registri su ravnopravni
    - Nema akumulatorske arhitekture
  - Scratch Pad memorija
    - 64 bajta memorija opće namijene
    - Sadržaj bilo kojeg od 16 registara opće namijene moguće je pohraniti i učitati iz Scratch Pad korištenjem naredbe STORE i FETCH.
    - 6-bitna adresa podatka u Scratch Pad-u može biti zadana direktno u naredbi ili indirektno preko jednog od registara opće namijene

Arhitektura

- ALU (Arithmetich Logic Unit)
  - 8-bita
  - ALU naredbe imaju jedan ili dva operanda ovisno o tipu naredbe
  - Rezultat operacije se uvijek sprema u prvi operand
  - Drugi operand može biti jedan od registara opće namijene ili 8 bitna konstanta
- Zastavice
  - Postoje dvije zastavice ZERO i CARRY
  - Stanje zastavica mijenjaju ALU naredbe
  - CARRY zastavica
    - se koristi kod aritmetičkih operacija
    - Prilikom rotacije i pomaka
    - Kod naredbe TEST koristi se za detekciju ODD PARITY

Arhitektura

- Reset
  - Postavlja procesor u početno inicijalno stanje
  - Počinje izvođenje programa od adrese ‘000’
  - Onemogućuje se prihvat prekida
  - Zastavice se postavljaju u inicijalno stanje
  - CALL/RETURN stog se postavlja u inicijalno stanje
- ULAZ/IZLAZ
  - 256 ulaza i 256 izlaza
  - 8-bita za adresiranje preko registra opće namijene ili konstantom
  - INPUT i OUTPUT naredbe

Arhitektura

- Prekidi
  - Postoji jedna prekidna linija
  - Dodavanjem dodatne logike može se spojiti više interapt linija
  - Aktiviranjem prekida izvršava se naredba CALL 3FF, gdje se nalazi prekidni potprogram.
  - Prihvatom prekida generira se puls INTERRUPT_ACK
  - Zastavice se pohranjuju
  - Za povrat i prekida koristi se RETURNI

Korištenje PicoBlaze-a VHDL

- PicoBlaze se koristi kako makro koji postoji u VHDL-u i kao predefinirani makro.

```
component kcpsm3
    Port (      address : out std_logic_vector(9 downto 0);
            instruction : in std_logic_vector(17 downto 0);
                port_id : out std_logic_vector(7 downto 0);
           write_strobe : out std_logic;
               out_port : out std_logic_vector(7 downto 0);
            read_strobe : out std_logic;
                in_port : in std_logic_vector(7 downto 0);
              interrupt : in std_logic;
          interrupt_ack : out std_logic;
                  reset : in std_logic;
                    clk : in std_logic);
    end component;
```

```
processor: kcpsm3
    port map(      address => address,
               instruction => instruction,
                   port_id => port_id,
              write_strobe => write_strobe,
                  out_port => out_port,
               read_strobe => read_strobe,
                   in_port => in_port,
                 interrupt => interrupt,
             interrupt_ack => interrupt_ack,
                     reset => reset,
                       clk => clk);
```

Spajanje programskog ROM-a

- Asembler program generira VHDL datoteku koja sadrži opis ROM memorije
- Tako generirana datoteka može služiti za implementaciju i simulaciju dizajna
- Ime komponente *prog_rom* definirana je imenom vašeg programa

```
component prog_rom
   Port (      address : in std_logic_vector(9 downto 0);
           instruction : out std_logic_vector(17 downto 0);
                   clk : in std_logic);
   end component;
```

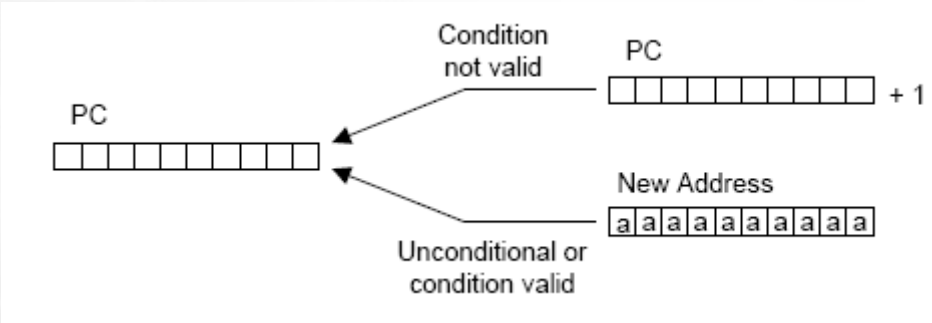
```
program: prog_rom
    port map(      address => address,
                instruction => instruction,
                        clk => clk);
```

PicoBlaze - Naredbe

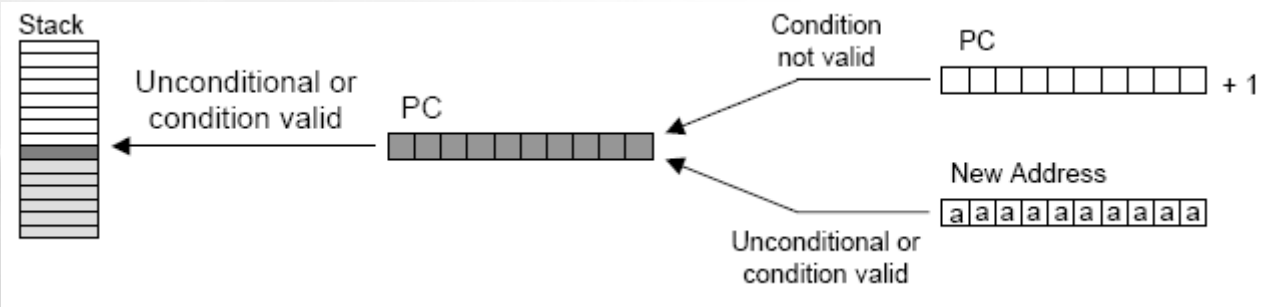
- Naredbe za kontrolu toka programa
- Aritmetičke naredbe
- Logičke naredbe
- Naredbe pomaka i rotacije
- Naredbe za rad s podatcima
- Ulazno-Izlazne naredbe
- Naredbe za rad s prekidima

Naredbe za kontrolu toka programa

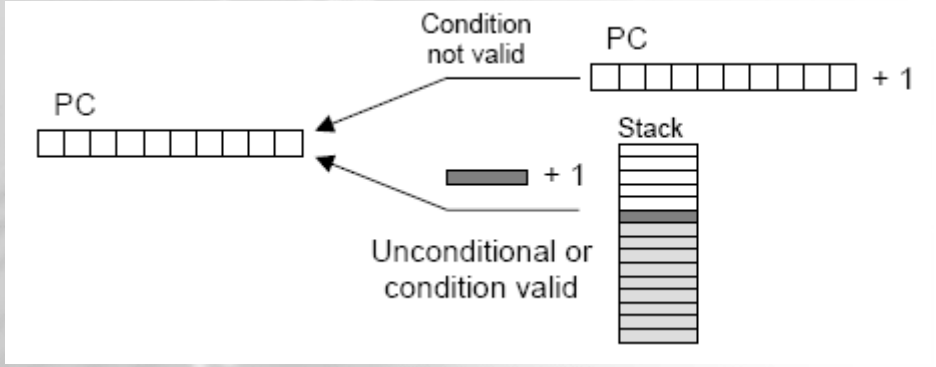
```
JUMP aaa
JUMP Z,aaa
JUMP NZ,aaa
JUMP C,aaa
JUMP NC,aaa
```




```
CALL aaa
CALL Z,aaa
CALL NZ,aaa
CALL NC,aaa
```



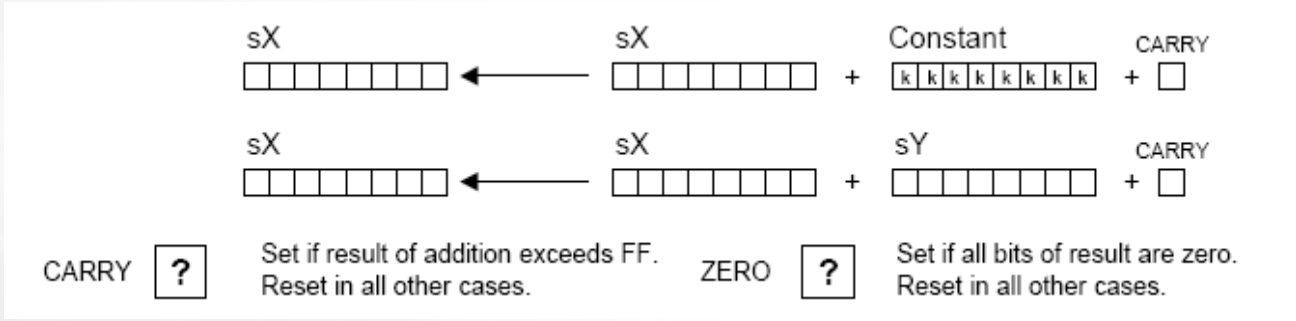

```
RETURN
RETURN Z
RETURN NZ
RETURN C
RETURN NC
```



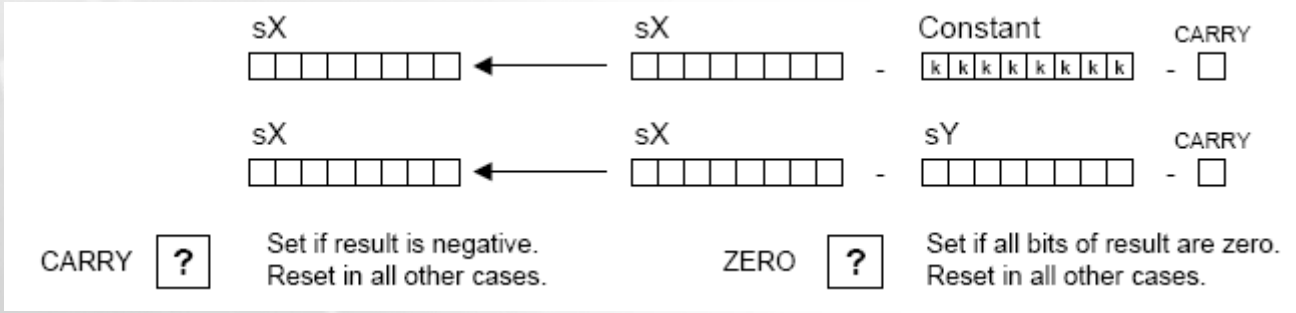
aaa – predstavlja adresu u rasponu od 000 do 3FF

Aritmetičke naredbe

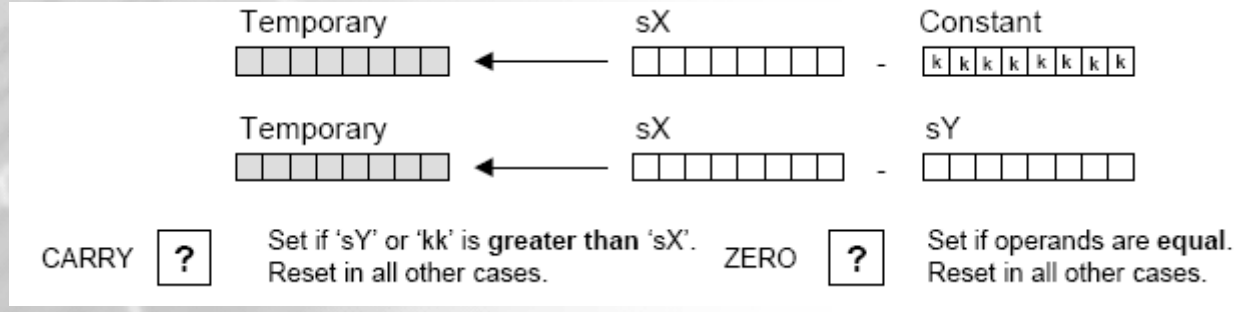
```
ADD sX,kk
ADDCY sX,kk
ADD sX,sY
ADDCY sX,sY
```




```
SUB sX,kk
SUBCY sX,kk
SUB sX,sY
SUBCY sX,sY
```

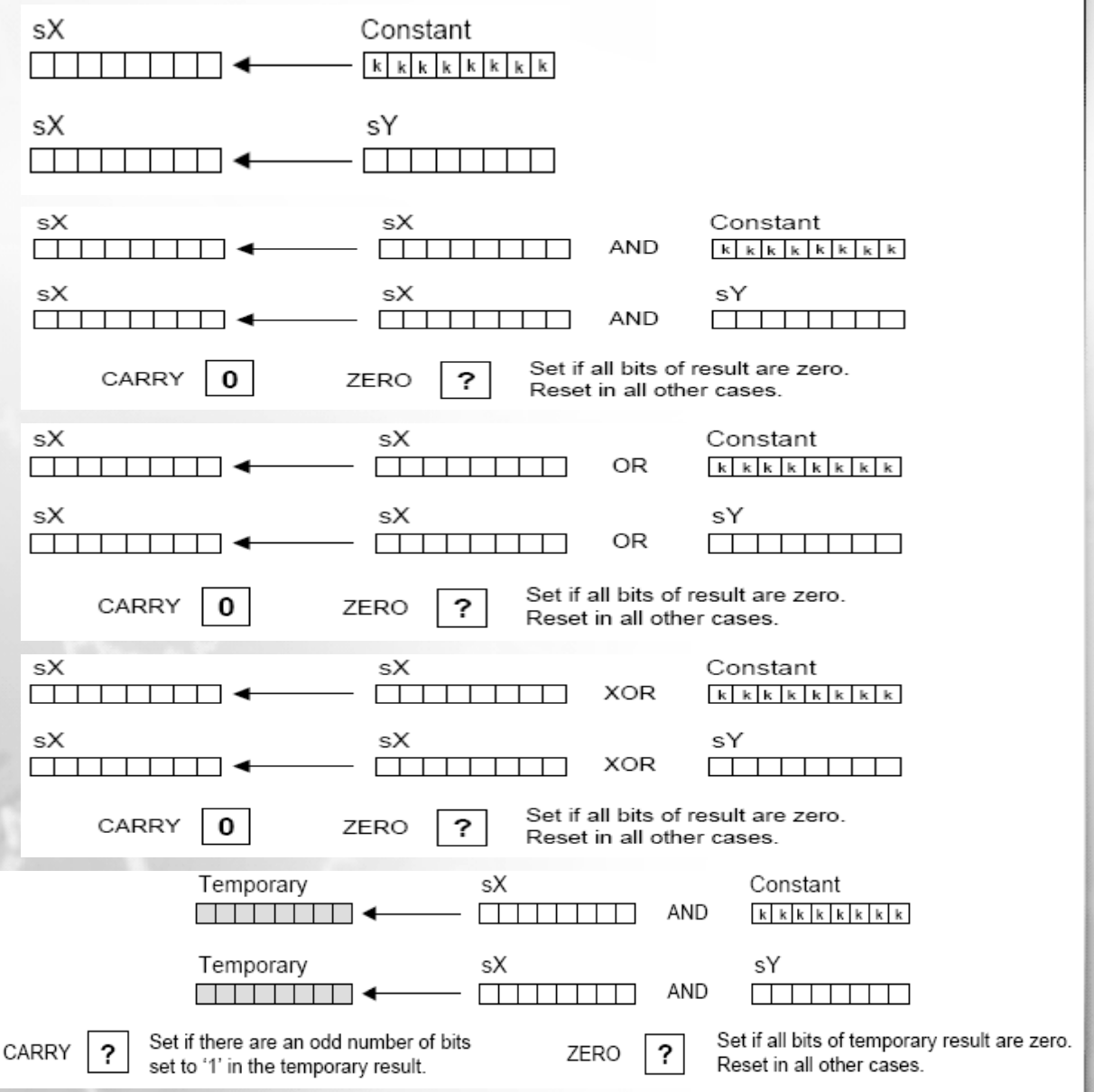



```
COMPARE sX,kk
COMPARE sX,sY
```



Logičke naredbe

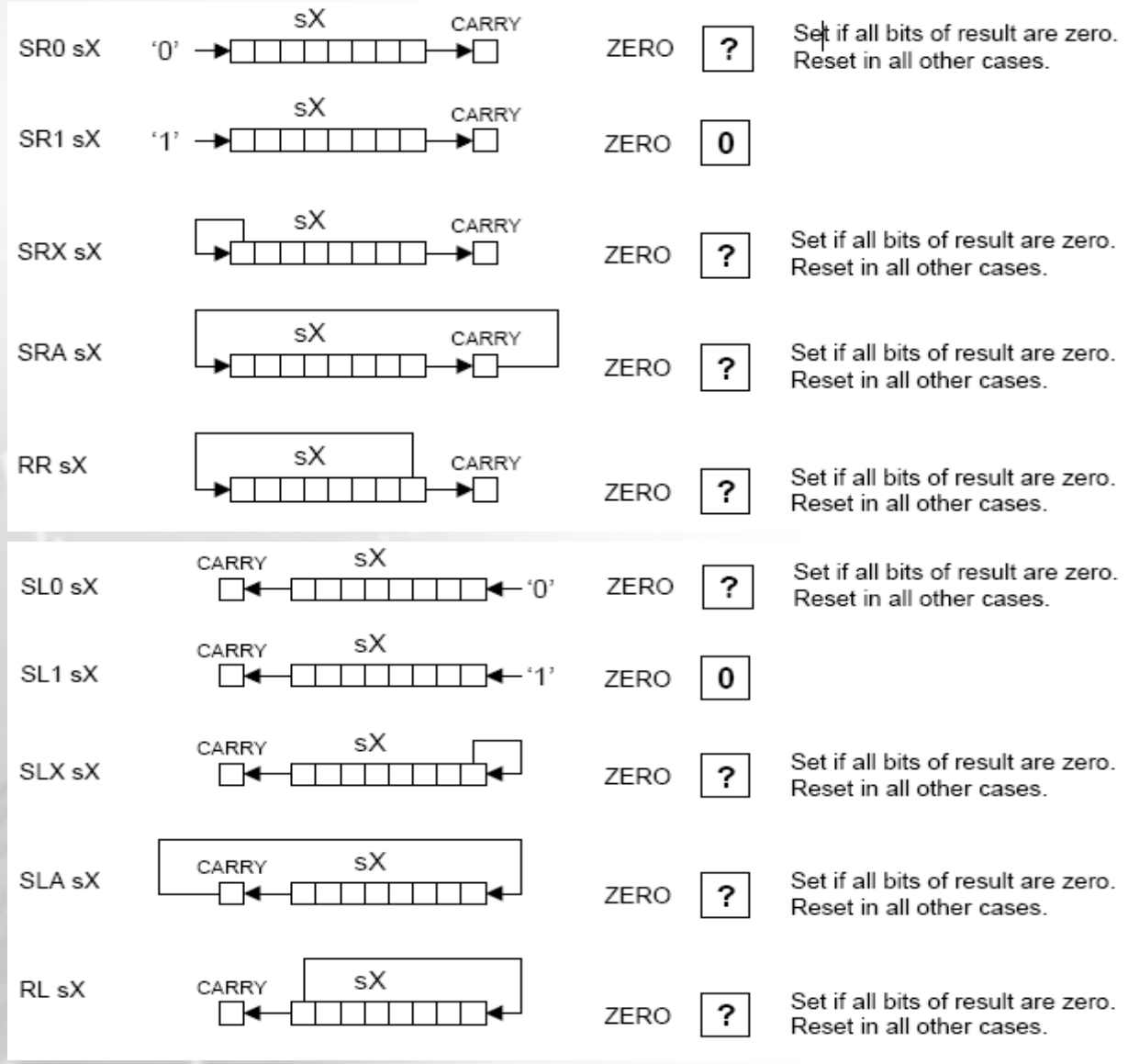
```
LOAD sX,kk
LOAD sX,sY

AND sX,kk
AND sX,sY

OR sX,kk
OR sX,sY

XOR sX,kk
XOR sX,sY

TEST sX,kk
TEST sX,sY
```



Naredbe pomaka i rotacije

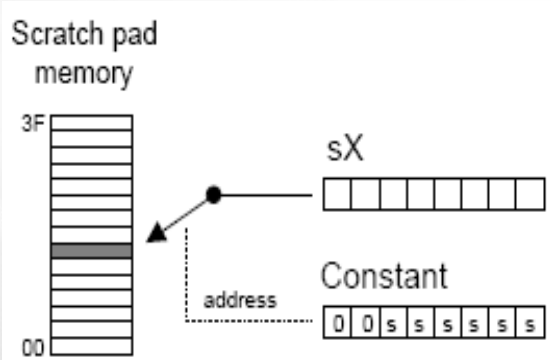
SR0 sX
SR1 sX
SRX sX
SRA sX
RR sX

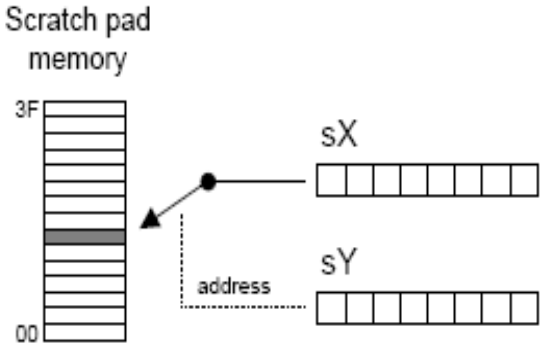
SL0 sX
SL1 sX
SLX sX
SLA sX
RL sX



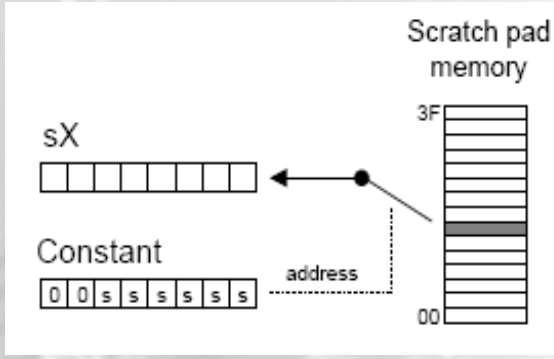
Naredbe za rad s podatcima

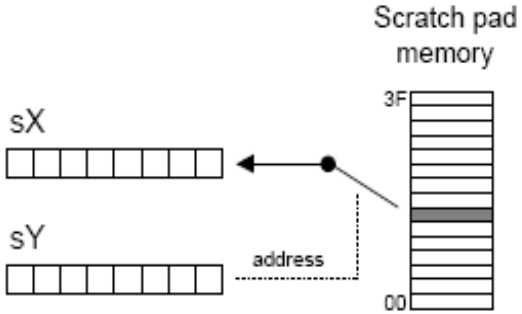
```
STORE sX,SS
STORE sX,(sY)
```





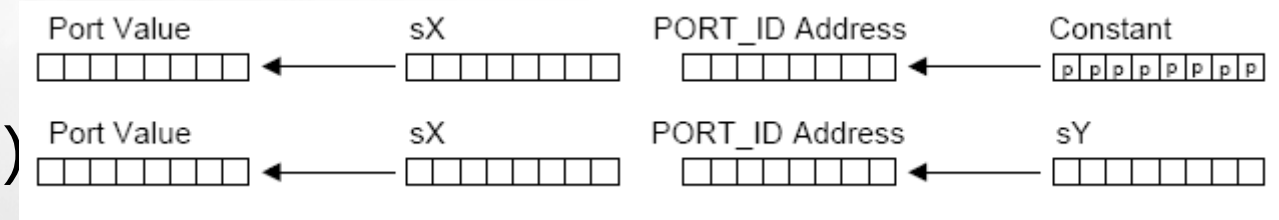

```
FETCH sX,SS
FETCH sX,(sY)
```



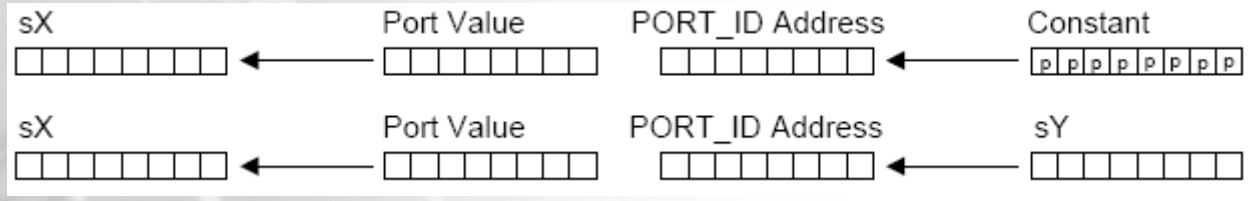


Ulazno-Izlazne naredbe

```
OUTPUT sX,PP
OUTPUT sX,(sY)
```



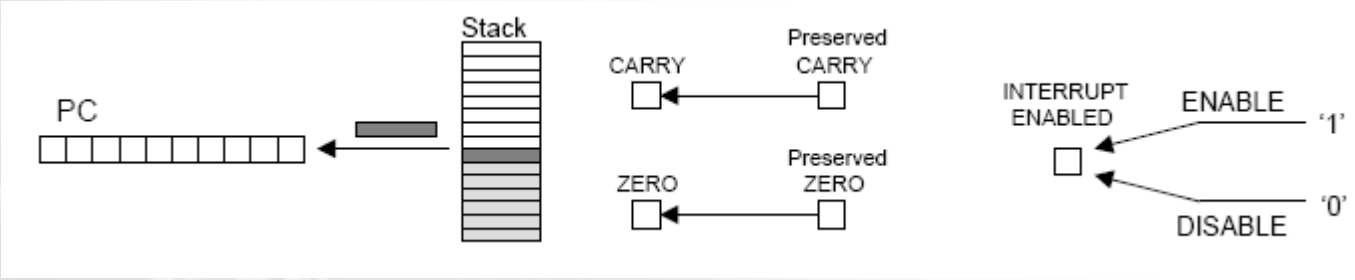

```
INPUT sX,PP
INPUT sX,(sY)
```



Naredbe za rad s prekidima

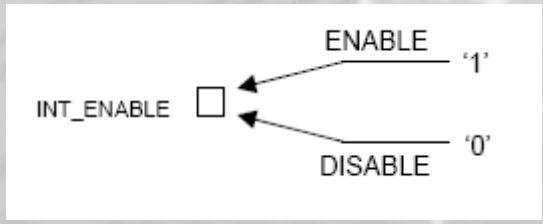
RETURNI ENABLE

RETURNI DISABLE



ENABLE INTERAPT

DISABLE INTERAPT



KCPSM3 Instruction Set

'X' and 'Y' refer to the definition of the storage registers 's' in the range 0 to F.
'kk' represents a constant value in the range 00 to FF.
'aaa' represents an address in the range 000 to 3FF.
'pp' represents a port address in the range 00 to FF.
'ss' represents an internal storage address in the range 00 to 3F.

Program Control Group

```
JUMP aaa
JUMP Z,aaa
JUMP NZ,aaa
JUMP C,aaa
JUMP NC,aaa

CALL aaa
CALL Z,aaa
CALL NZ,aaa
CALL C,aaa
CALL NC,aaa

RETURN
RETURN Z
RETURN NZ
RETURN C
RETURN NC
```

Note that call and return supports up to a stack depth of 31.

Arithmetic Group

```
ADD sX,kk
ADDCY sX,kk
SUB sX,kk
SUBCY sX,kk
COMPARE sX,kk

ADD sX,sY
ADDCY sX,sY
SUB sX,sY
SUBCY sX,sY
COMPARE sX,sY
```

Interrupt Group

```
RETURNI ENABLE
RETURNI DISABLE

ENABLE INTERRUPT
DISABLE INTERRUPT
```

Logical Group

```
LOAD sX,kk
AND sX,kk
OR sX,kk
XOR sX,kk
TEST sX,kk

LOAD sX,sY
AND sX,sY
OR sX,sY
XOR sX,sY
TEST sX,sY
```

Storage Group

```
STORE sX,ss
STORE sX,(sY)
FETCH sX,ss
FETCH sX,(sY)
```

Shift and Rotate Group

```
SR0 sX
SR1 sX
SRX sX
SRA sX
RR sX

SL0 sX
SL1 sX
SLX sX
SLA sX
RL sX
```

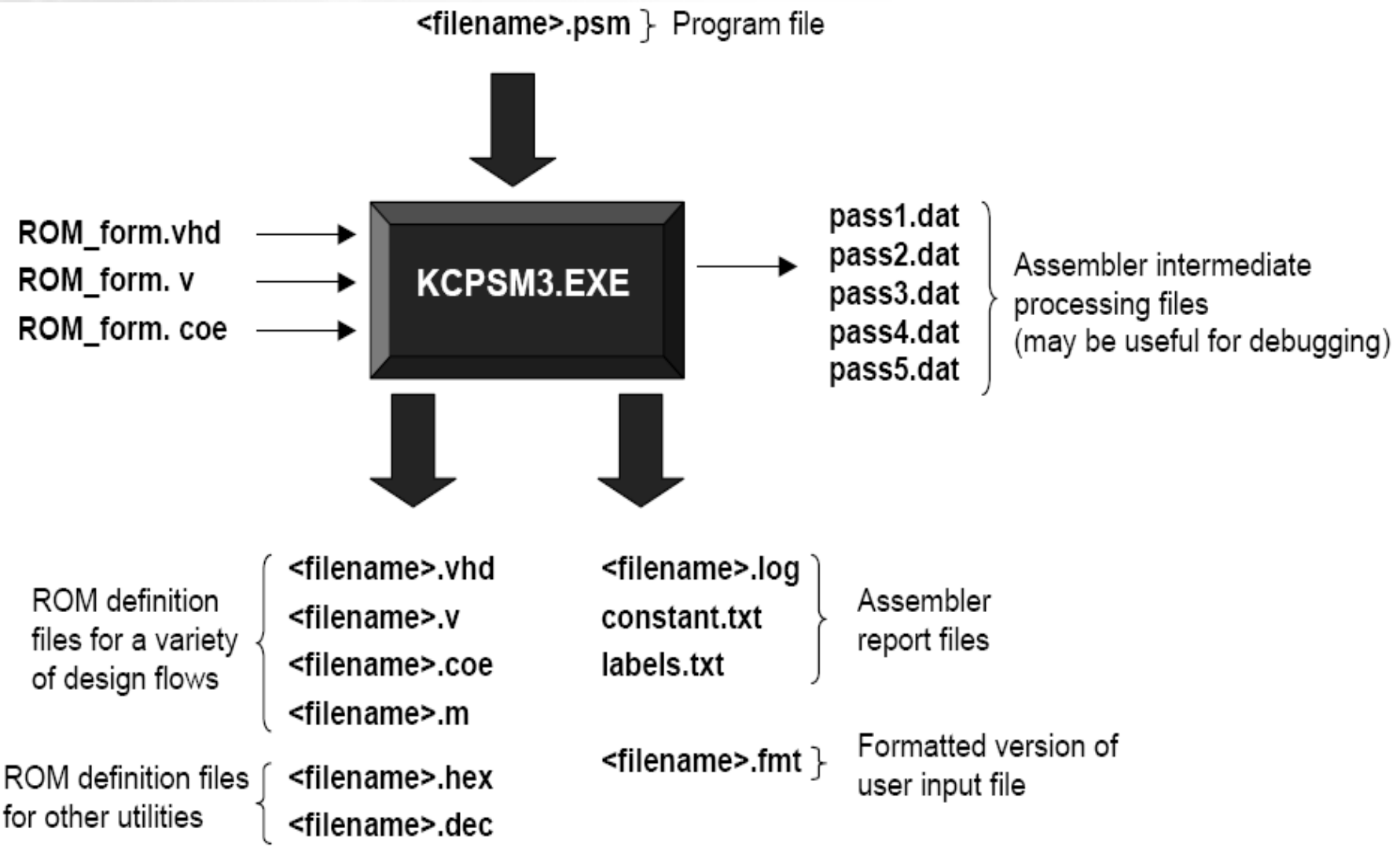
Input/Output Group

```
INPUT sX,pp
INPUT sX,(sY)
OUTPUT sX,pp
OUTPUT sX,(sY)
```

PikoBlaze asembler

- Asembler je dostupan u obliku DOS EXE datoteke `kcpsm3.exe`
- Za korištenje asemblera potrebno je iskopirati četri datoteke u radni direktorij:
  - KCPSM3.EXE
  - ROM_form.vhd
  - ROM_form.v
  - ROM_form.coe
- Program pišete u bilo kojem text editoru (Notepad, Wordpad, ili neki slični)
- Program mora imati exstenziju .psm, a ime može imati samo 8 slova

Asembler


<filename>.psm } Program file
ROM_form.vhd
ROM_form. v
ROM_form. coe
KCPSM3.EXE
pass1.dat
pass2.dat
pass3.dat
pass4.dat
pass5.dat
Assembler intermediate processing files (may be useful for debugging)
ROM definition files for a variety of design flows
<filename>.vhd
<filename>.v
<filename>.coe
<filename>.m
<filename>.log
constant.txt
labels.txt
Assembler report files
ROM definition files for other utilities
<filename>.hex
<filename>.dec
<filename>.fmt
Formatted version of user input file

Asembler pravila pisanja

- Prazne linije se ignoriraju
- Za komentare se koristi znak ; . Sve iza znaka komentara zanemaruje se.
- Registri se adresiraju s slovom ‘s’ iza kojeg slijedi heksa znamenka registra 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- Labele – mogu sadržavati slova i brojke. Ne smiju sadržavati razmak niti znak minus. Mogu imati znak ‘_’. Labele su “case sensitive”.
- Line Labela – isto kao i obična labela samo završava s znakom ‘:’. Koristi se kod JUMP i CALL naredbi

Asembler pravila pisanja

- Naredbe se pišu prema pravilima za pisanje naredbi kako je opisano prije.
- Naredbe nisu “case sensitive”
- Moguće je koristiti razmake i tabulatore u naredbama

Asembler - Konstante

- Asembler naredba CONSTANT
- Pridružuje dvoznamenkastom heksa broju labelu
- Jednostavnije i preglednije je koristiti konstante prilikom pisanja programa

```
CONSTANT light_red, 03
CONSTANT light_sensor, 1F
CONSTANT temp_senzor, 05
```

Asembler – NAMEREG

- Asembler naredba NAMEREG
- Omogućuje pridruživanje imena jednom od 16 registara opće namijene
- Nakon definicije novog imena nije moguće koristiti stare nazive za registar

```
NAMEREG sF, light_controle_msb
NAMEREG sE, light_controle_lsb
NAMEREG sD, new_temp
```

Asembler ADDRESS

- Asembler naredba ADDRESS
- Omogućava definiranje adrese od koje će daljnji tijek programa biti pohranjen
- Adresa mora biti zadana kao troznamenkasta heksa vrijednost

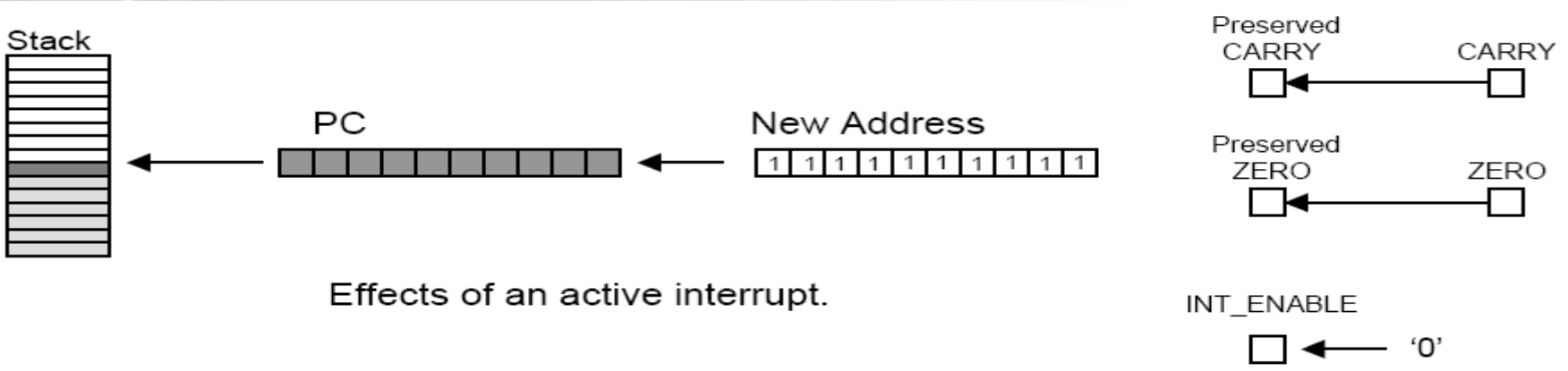
```
ADDRESS 01F
…
…
ADDRESS 3FF
```

PicoBlaze kompatibilnost

	KCPSM	KCPSM2	KCPSM3
Target Devices	Spartan-II, Spartan-IIE, Virtex, Virtex-E	Virtex-II, Virtex-IIPRO	Spartan-3, Virtex-II, Virtex-IIPRO
Program Size	256 instructions (256×16 Block RAM)	1024 instructions (1024×18 Block RAM)	1024 instructions (1024×18 Block RAM)
Registers	16	32	16
Scratch-Pad Memory	-	-	64 Bytes
Size	76 Slices	84 Slices	96 Slices
CALL/RETURN stack	15 levels	31 levels	31 levels
Features and Comments	Smallest and oldest! Very well used and proven. Relatively small program space.	Register rich. Virtex-II devices only. Can **not** migrate design directly to Spartan-3.	COMPARE and TEST instructions, PARITY test, Scratch-pad memory, INTERRUPT_ACK signal

Prekidni sustav

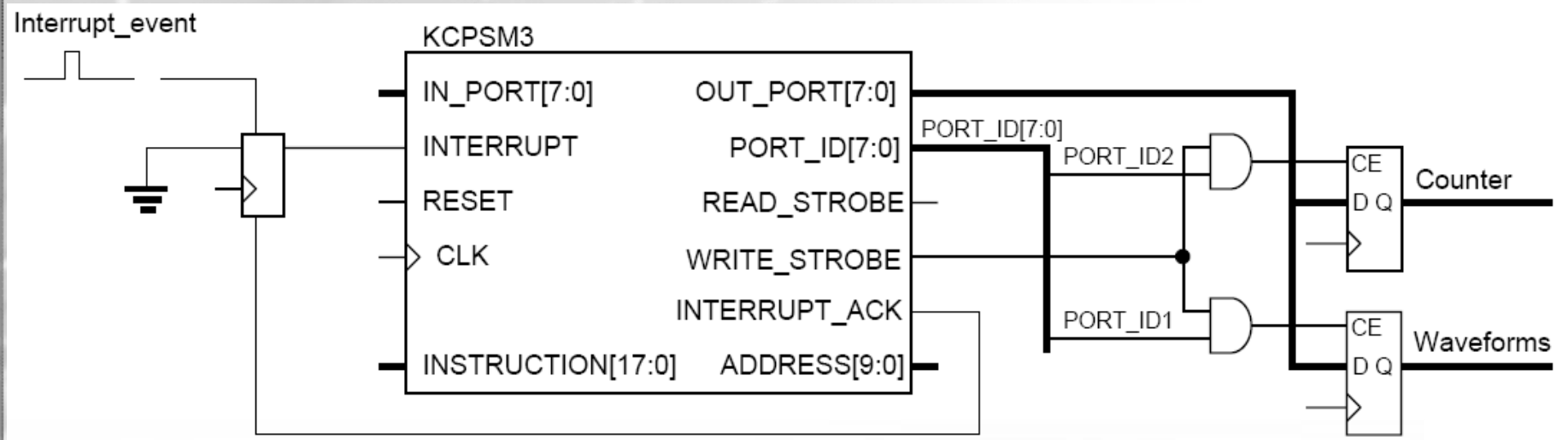
- U početnom stanju prekidi su onemogućeni
  - ENABLE INTERRUPT – omogući prekid
  - DISABLE INTERRUPT – onemogući prekid
- Prihvat prekida:
  - U koliko je prihvat prekida omogućen izvršava se trenutna naredba do kraja i prihvaća se prekid
  - Programsko brojilo sprema se na stog. CARRY i ZERO zastavice spremaju se u zasebne bistabile.
  - Onemogućuje se daljnji prihvat prekida
  - U programsko brojilo upisuje se 3FF
- Za povrat iz prekida koristi se RETURNI



Effects of an active interrupt.

Prekidni sustav

```
interrupt_control: proces(clk)
begin
   if clk'event and clk = '1' then
        if interrupt_ack = '1' then
               interrupt <= '0';
        elsif interrupt_event_1 = '1' then
               interrupt <= '1';
        else
               interrupt <=interrupt;
        end if;
End process interrupt_control;
```



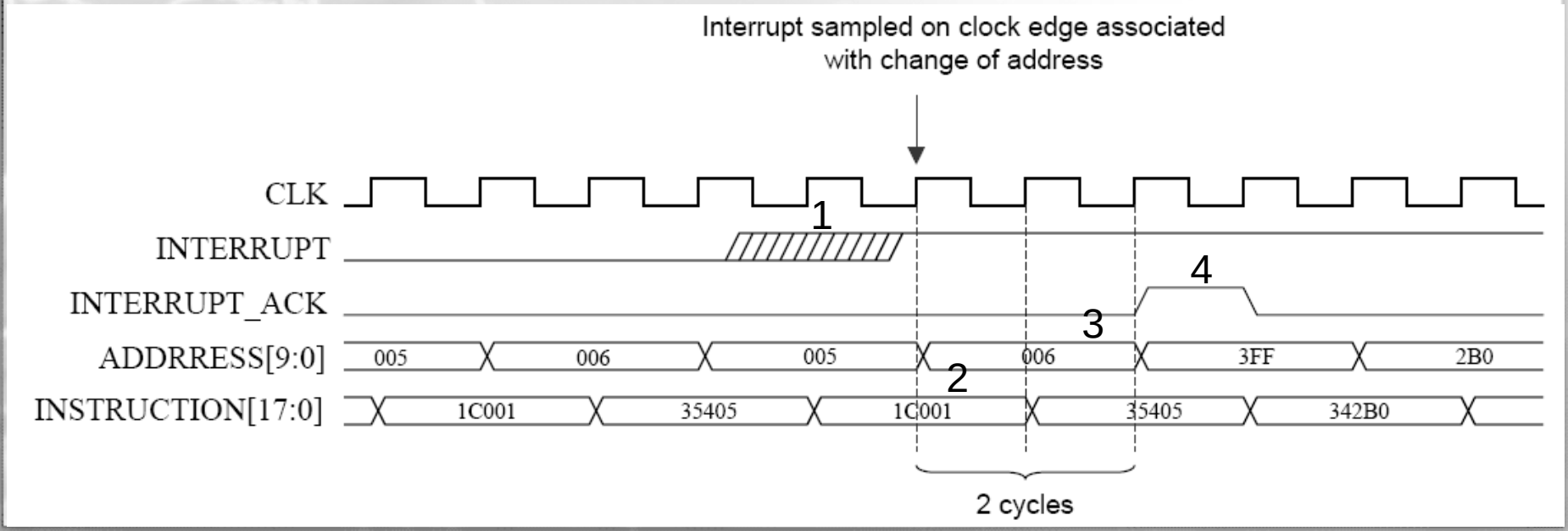
Prekidni sustav

- Primjer prihvata prekida

```
prekid:     ...
            ...
            ...
            RETURNI

            ADDRESS 3FF
            JUMP prekid
```

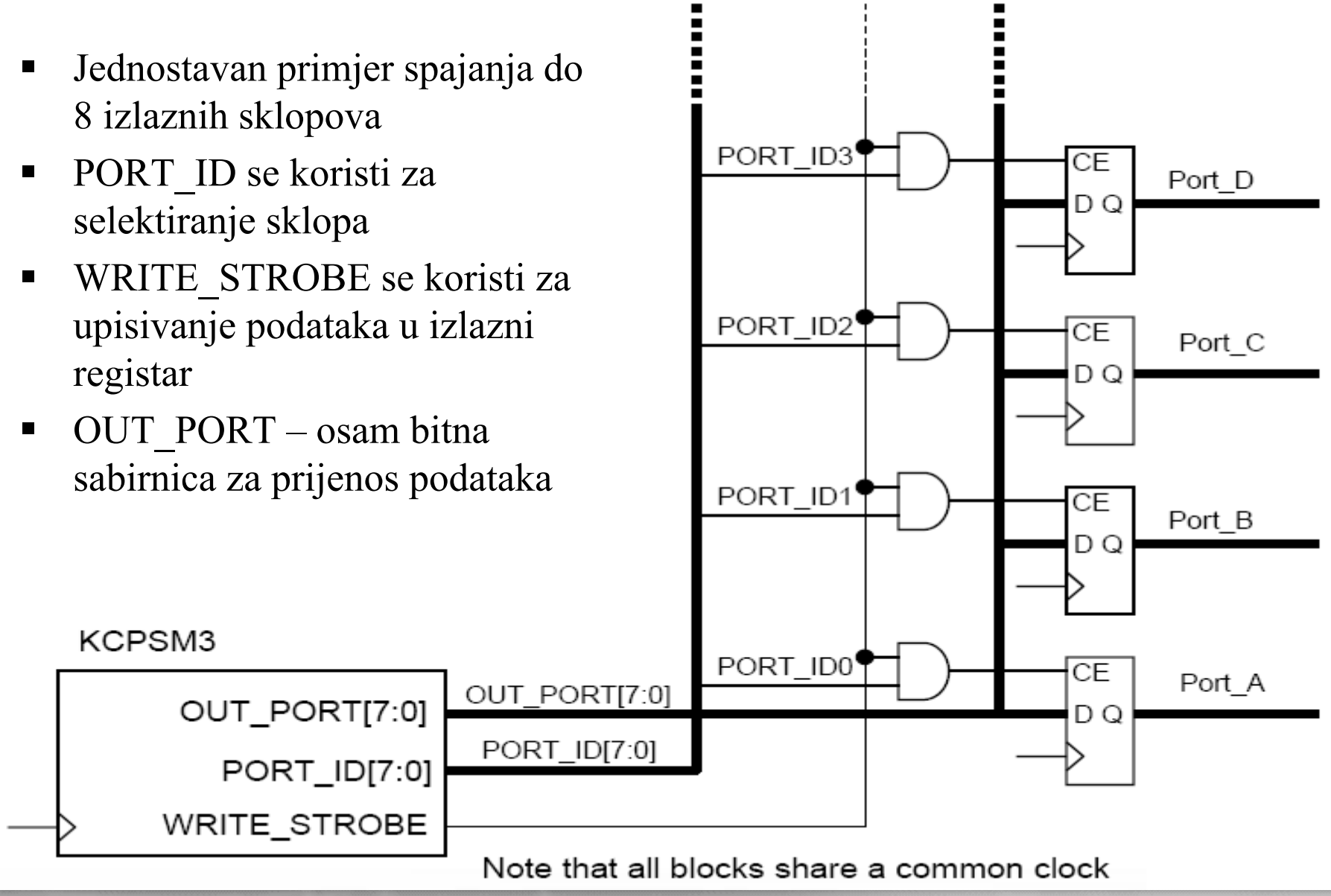
Prekidni sustav

- Vremenski dijagram prihvata prekida
  - 1) Sklop generira prekid
  - 2) Izvršava se naredba do kraja NAREDBA(1C001)
  - 3) Programsko brojilo se sprema na stog
  - 4) Prihvaća se prekid PC<=3FF i generiraj INTERRUPT_ACK



Rad sa izlazima

- Jednostavan primjer spajanja do 8 izlaznih sklopova
- PORT_ID se koristi za selektiranje sklopa
- WRITE_STROBE se koristi za upisivanje podataka u izlazni registar
- OUT_PORT – osam bitna sabirnica za prijenos podataka



Rad sa izlazima

- Primjer VHDL procesa koji obrađuje izlaz podataka prema LED diodama

```
output_ports: process(clk)
  begin
    if clk'event and clk='1' then
      if write_strobe='1' then
        -- Write to LEDs at address 80 hex.
        if port_id(7)='1' then
          led <= out_port;
        end if;
        end if;
    end if;
 end process output_ports;
```

- Primjer asembler koda koji piše na izlaz

```
CONSTANT LED_PORT, 40
...
LOAD   s0,08
OUTPUT s0, LED_PORT
...
```

Rad s ulazima

- Primjer izvedbe VHDL procesa koji obrađuje ulazne sklopove

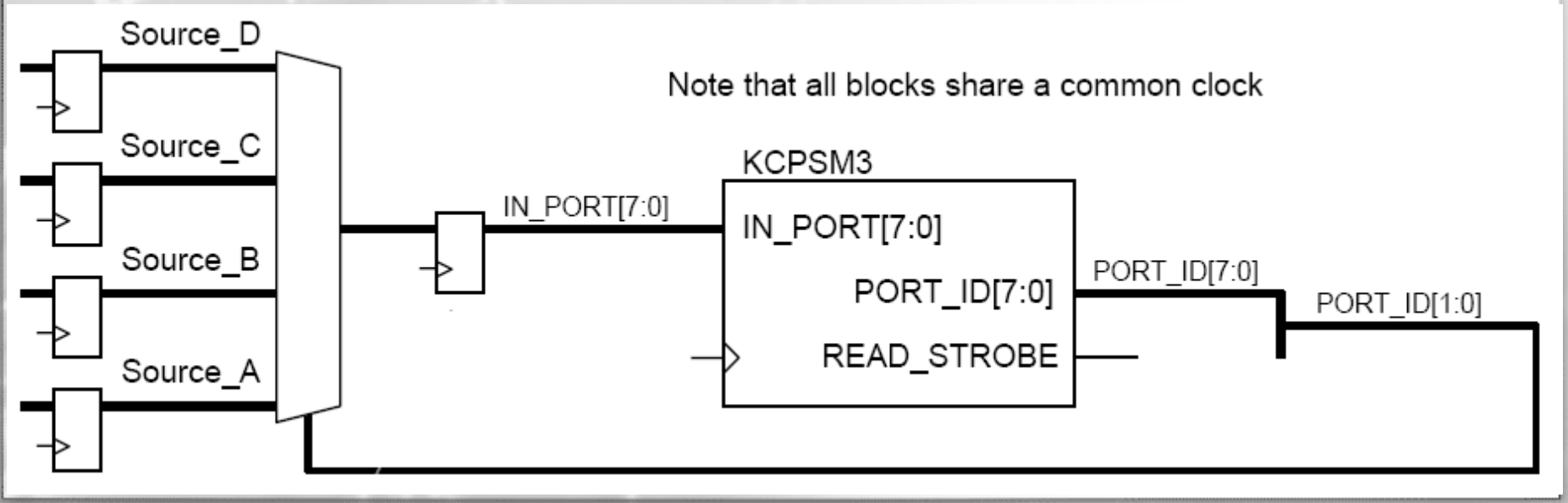
```
input_ports: process(clk)
begin
    if clk'event and clk='1' then
      case port_id(7 downto 6) is
       -- read data at address 80 hex
        when "10" =>
               in_port <= "0000000" & busy_lcd;
       -- Don't care used for all other addresses when
        others =>
               in_port <= "XXXXXXXX";
      end case;
    end if;
end process input_ports;
```

- Primjer asembler programa za čitanje s ulaznog sklopa

```
CONSTANT LCD_DATA, 80
...
INPUT  s1, LCD_DATA
...
```

Rad sa ulazima

- Najjednostavnije je koristiti multipleksor na ulazu
- U koliko sklop mora detektirati čitanje podatka od strane procesora može se koristiti READ_STROBE signal
- PORT_ID predstavlja adresu ulaznog sklopa
- IN_PORT predstavlja ulazni podatak



PicoBlaze – KCPSM3

- Više informacija na:
  - http://www.xilinx.com/products/ipcenter/picoblaze-S3-V2-Pro.htm
  - http://www.xilinx.com/bvdocs/ipcenter/data_sheet/picoblaze_productbrief.pdf
  - http://www.xilinx.com/bvdocs/userguides/ug129.pdf
- Asembler:
  - http://www.fer.hr/_download/repository/KCPSM3.zip
  - http://www.xilinx.com/xlnx/xil_entry2.jsp?sMode=login&group=picoblaze