

vaja 05

PicoBlaze KCPSM6

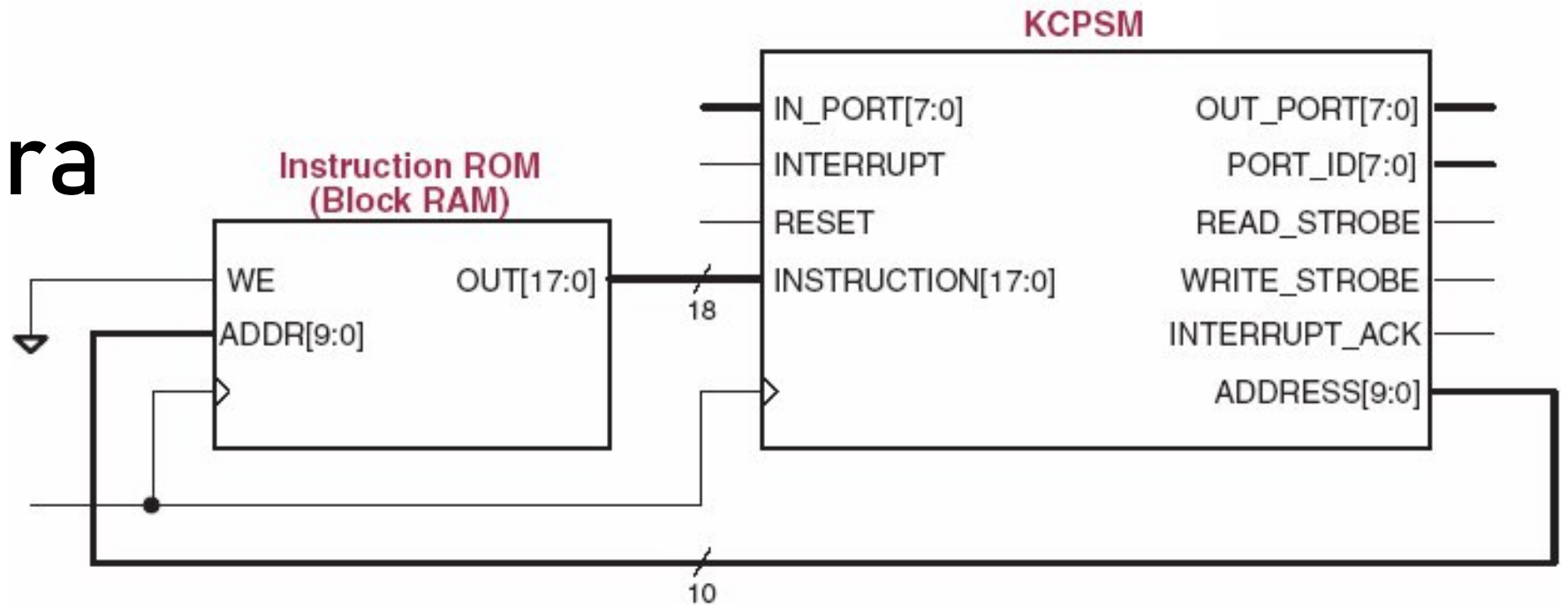
Digitalno načrtovanje – laboratorijske vaje
asistent: Nejc Ilc

Uvod

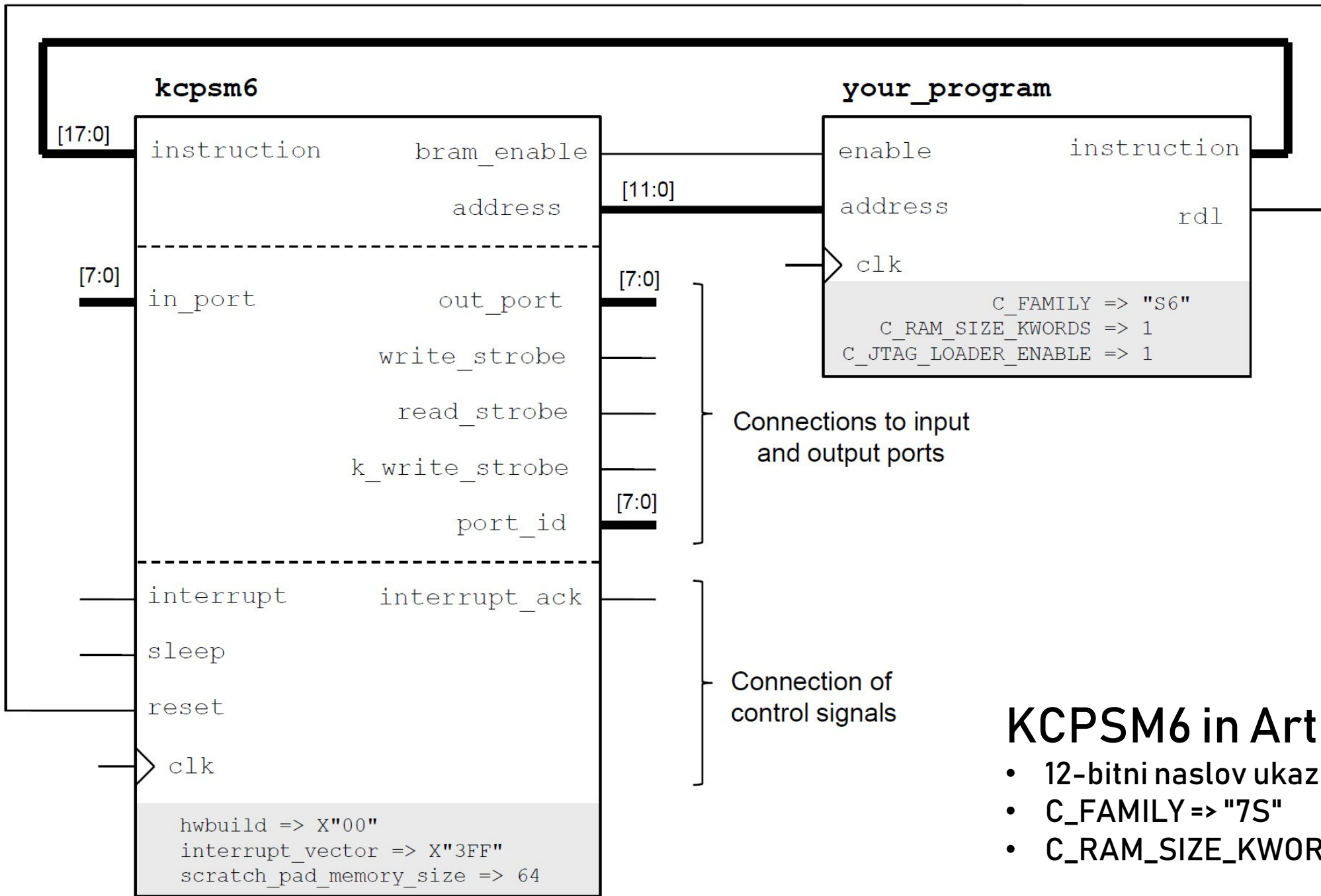


- KCPSM
 - Ken Chapman's Programmable State Machine
 - Konstant Coded Programmable State Machine
- Enostaven mikrokrmilnik (t.i. *soft processor*), prilagojen za celostno vgradnjo v čipe FPGA podjetja Xilinx
 - arhitektura: 8-bitni RISC
 - zmogljivost: 52-119 MIPS
 - zelo varčen z viri - zasede 0,3-5 % vezja FPGA (26 rezin - slices)
 - [odprtokoden](#), prost
- Namen uporabe: kompleksni, časovno nekritični avtomati, ki jih programiramo v zbirniku

Arhitektura



- Implementacijo sestavljata mikrokrmilnik in ukazni pomnilnik (ROM), ki hrani program
 - ukazi so 18-bitni,
 - ukazni pomnilnik hrani do 4 K ukazov (na čipih Artix-7 do 2 K),
 - uporablja komponento "bločni RAM" (BRAM). Vhod WE je 0, kar pomeni, da imamo ROM namesto RAM.
 - Zgornja shema prikazuje KCPSM, ki naslavlja pomnilnik z 10 biti (1 K ukazov).



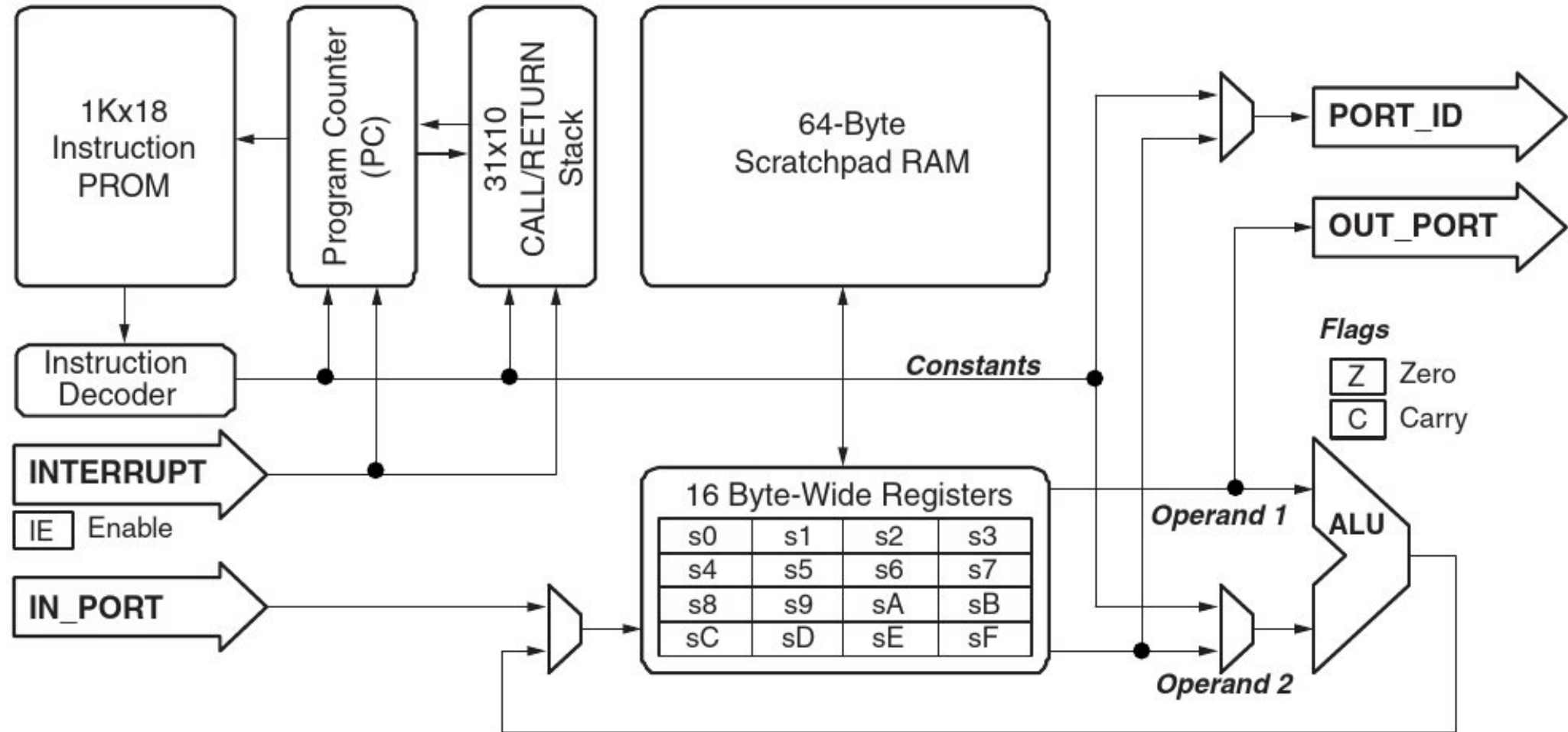
KCPSM6 in Artix-7

- 12-bitni naslov ukaznega pomnilnika
- C_FAMILY => "7S"
- C_RAM_SIZE_KEYWORDS => 2

Arhitektura

- 18-bitni ukazi
 - vsi ukazi se izvajajo 2 urini periodi (RISC)
 - ukazi lahko spreminjajo zastavici Zero (Z) in Carry (C), ki ju lahko uporabimo za procesiranje kaskade 8-bitnih podatkov (npr. 16, 32, 64, ...-bitna števila)
- 2 množici 16 8-bitnih splošno-namenskih registrov
 - poimenovani s0-sF
 - množica A in množica B za hiter preklop med neodvisnimi opravili, denimo, ko obdelujemo prekinitve
- (64/128/256)x8 bitni pomnilnik za hranjenje podatkov (Scratchpad RAM)
- Vhod/izhod
 - 8-bitni naslov V/I vrat PORT_ID (256 možnih vhodov in prav toliko izhodov)
 - 8-bitni podatek, ki se bere iz vhoda IN_PORT ali piše na izhod OUT_PORT
- Prekinitve
 - en prekinitveni vhod (INTERRUPT), potrjevanje (INTERRUPT_ACK)
 - reakcija v 3-4 urinih periodah (čas od aktiviranja prekinitve do vstopa v prekinitveno-servisni program)

Osnovna bločna shema



Nabor ukazov

- Vhod/izhod: INPUT, OUTPUT
- Registri: LOAD, STAR
- Aritmetični/logični: ADD, SUB, AND, OR, XOR, RL, RR, SL, SR,
- Primerjanje: COMPARE, TEST
- Delo s podatkovnim pomnilnikom: FETCH, STORE
- Skoki, klici: JUMP, CALL, RETURN
- Celoten seznam ukazov: glej naslednjo stran ali <https://docs.xilinx.com/v/u/en-US/ug129>, stran 15, Tabela 3-1

KCPSM6 Instruction Set

aaa : 12-bit address 000 to FFF
 kk : 8-bit constant 00 to FF
 pp : 8-bit port ID 00 to FF
 p : 4-bit port ID 0 to F
 ss : 8-bit scratch pad location 00 to FF
 x : Register within bank s0 to sF
 y : Register within bank s0 to sF

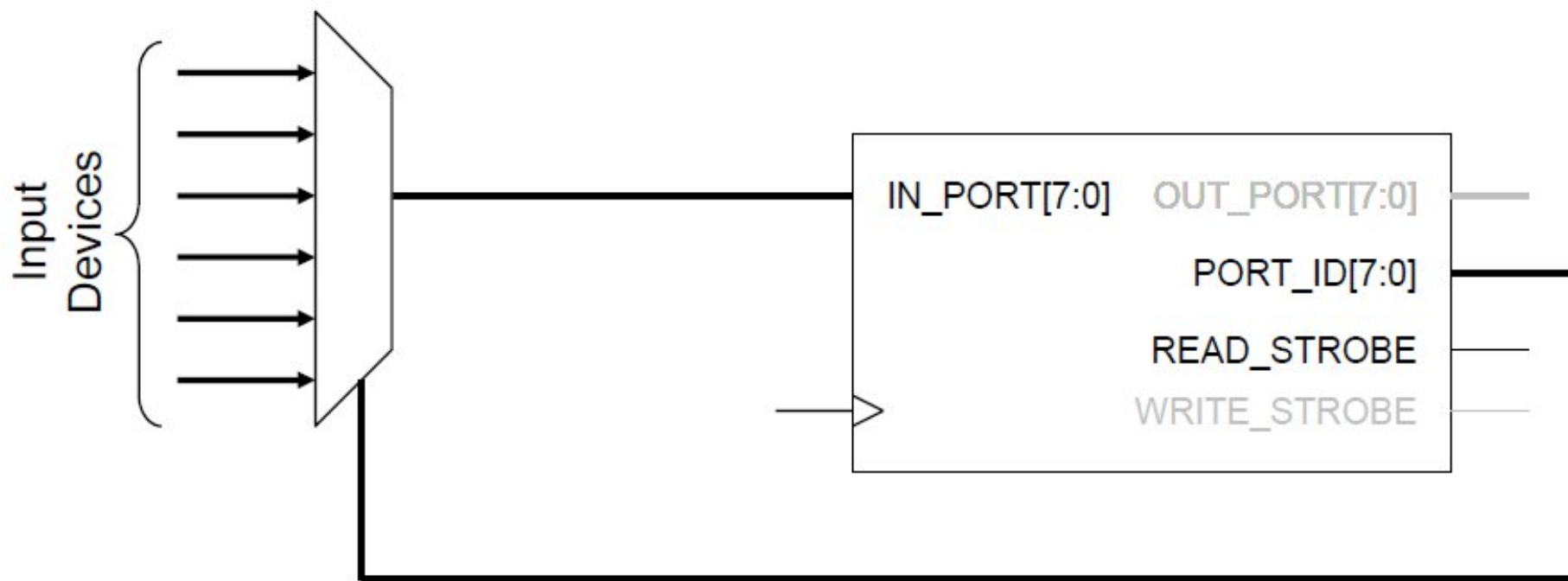
Page	Opcode	Instruction	Page	Opcode	Instruction	Page	Opcode	Instruction
Register loading			Shift and Rotate			Interrupt Handling		
55	00xy0	LOAD sX, sY	67	14x06	SL0 sX	83	28000	DISABLE INTERRUPT
55	01xkk	LOAD sX, kk	67	14x07	SL1 sX	83	28001	ENABLE INTERRUPT
71	16xy0	STAR sX, sY	67	14x04	SLX sX	84	29000	RETURNI DISABLE
Logical			67	14x00	SLA sX	84	29001	RETURNI ENABLE
56	02xy0	AND sX, sY	67	14x02	RL sX	Jump		
56	03xkk	AND sX, kk	68	14x0E	SR0 sX	87	22aaa	JUMP aaa
57	04xy0	OR sX, sY	68	14x0F	SR1 sX	88	32aaa	JUMP Z, aaa
57	05xkk	OR sX, kk	68	14x0A	SRX sX	88	36aaa	JUMP NZ, aaa
58	06xy0	XOR sX, sY	68	14x08	SRA sX	88	3Aaaa	JUMP C, aaa
58	07xkk	XOR sX, kk	68	14x0C	RR sX	88	3Eaaa	JUMP NC, aaa
Arithmetic			Register Bank Selection			89	26xy0	JUMP@ (sX, sY)
59	10xy0	ADD sX, sY	70	37000	REGBANK A	Subroutines		
59	11xkk	ADD sX, kk	70	37001	REGBANK B	92	20aaa	CALL aaa
60	12xy0	ADDCY sX, sY	Input and Output			93	30aaa	CALL Z, aaa
60	13xkk	ADDCY sX, kk	73	08xy0	INPUT sX, (sY)	93	34aaa	CALL NZ, aaa
61	18xy0	SUB sX, sY	73	09xpp	INPUT sX, pp	93	38aaa	CALL C, aaa
61	19xkk	SUB sX, kk	74	2Cxy0	OUTPUT sX, (sY)	93	3Caaa	CALL NC, aaa
62	1Axy0	SUBCY sX, sY	74	2Dxpp	OUTPUT sX, pp	94	24xy0	CALL@ (sX, sY)
62	1Bxkk	SUBCY sX, kk	78	2Bkpp	OUTPUTK kk, p	96	25000	RETURN
Test and Compare			Scratch Pad Memory			97	31000	RETURN Z
63	0Cxy0	TEST sX, sY	(64, 128 or 256 bytes)			97	35000	RETURN NZ
63	0Dxkk	TEST sX, kk	81	2Exy0	STORE sX, (sY)	97	39000	RETURN C
64	0Exy0	TESTCY sX, sY	81	2Fxss	STORE sX, ss	97	3D000	RETURN NC
64	0Fxkk	TESTCY sX, kk	82	0Axy0	FETCH sX, (sY)	98	21xkk	LOAD&RETURN sX, kk
65	1Cxy0	COMPARE sX, sY	82	0Bxss	FETCH sX, ss	Version Control		
65	1Dxkk	COMPARE sX, kk				101	14x80	HWBUILD sX
66	1Exy0	COMPARECY sX, sY						
66	1Fxkk	COMPARECY sX, kk						

Branje vhodov (1)

- Vrednost na vhodu preberemo z ukazom INPUT
- Sintaksa:
 - INPUT sX, pp
 - v register sX shrani stanje vhoda na naslovu pp
 - INPUT sX, (sY)
 - v register sX shrani stanje vhoda na naslovu, ki se nahaja v registru sY

Branje vhodov (2)

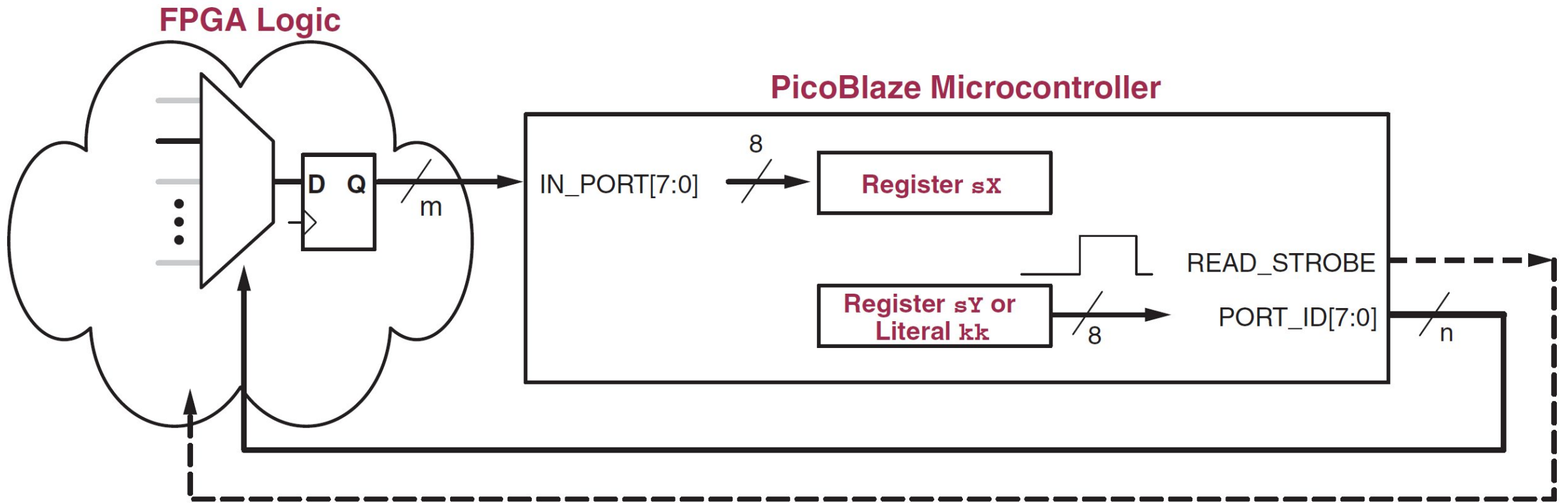
- Na vhodu IN_PORT je vrednost, ki jo preberemo v register
- PORT_ID določa naslov vhoda, ki ga beremo (enega od 256)



Branje vhodov (3)

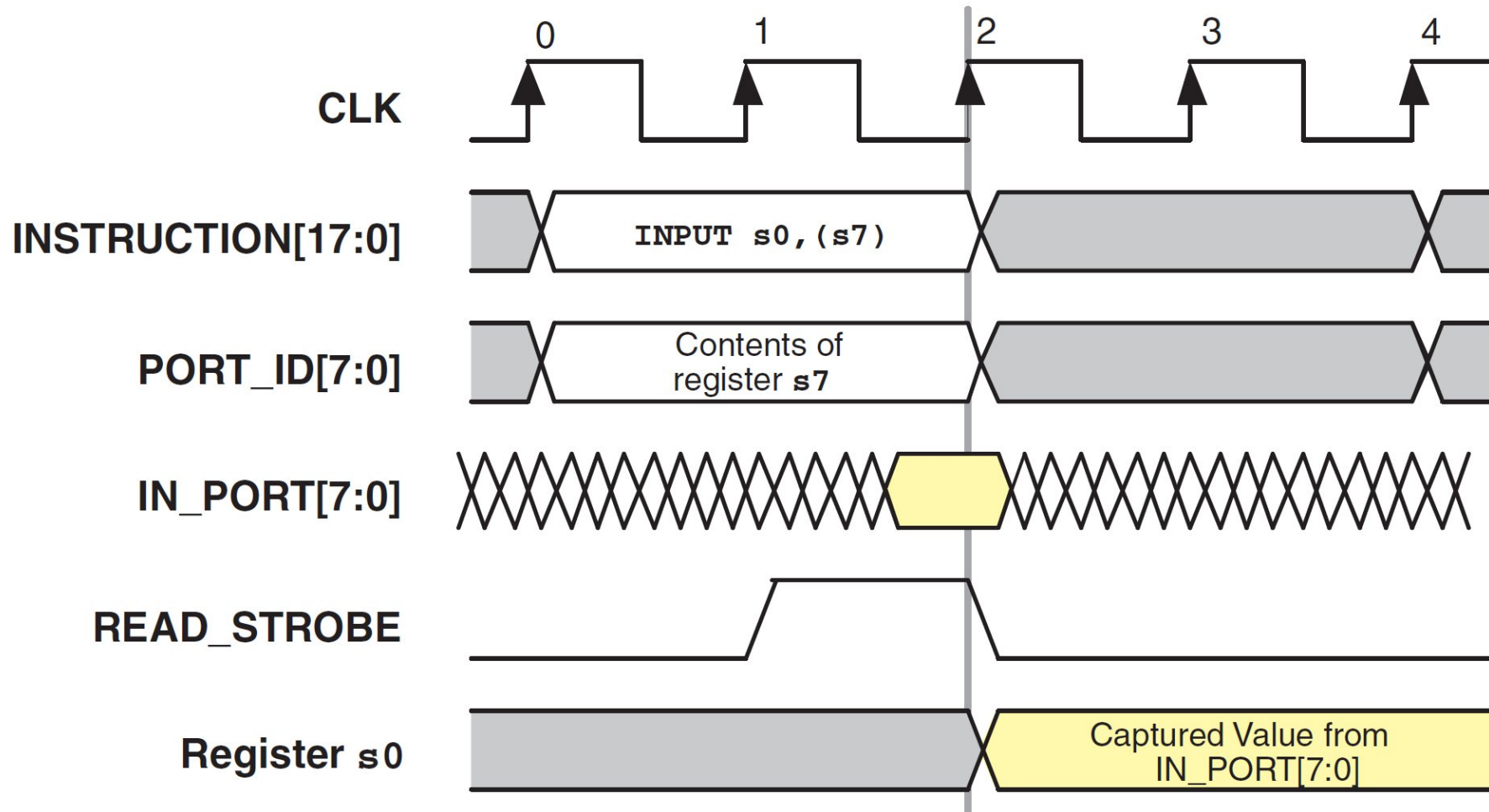
- Med izvajanjem ukaza INPUT se na signalu READ_STROBE ustvari impulz, ki označuje, da beremo vrednost na vhodu;
- nekaterim vhodom je "vseeno" za ta signal (npr. gumb, stikalo)
- spet drugim vhodom pa ni "vseeno", kdaj smo vrednost prebrali oziroma, ali sploh smo jo prebrali (npr. medpomnilnik FIFO). Pri teh se signal READ_STROBE uporabi, da lahko odstranijo prebran podatek (pomaknejo vsebino "naprej").

Branje vhodov (4)

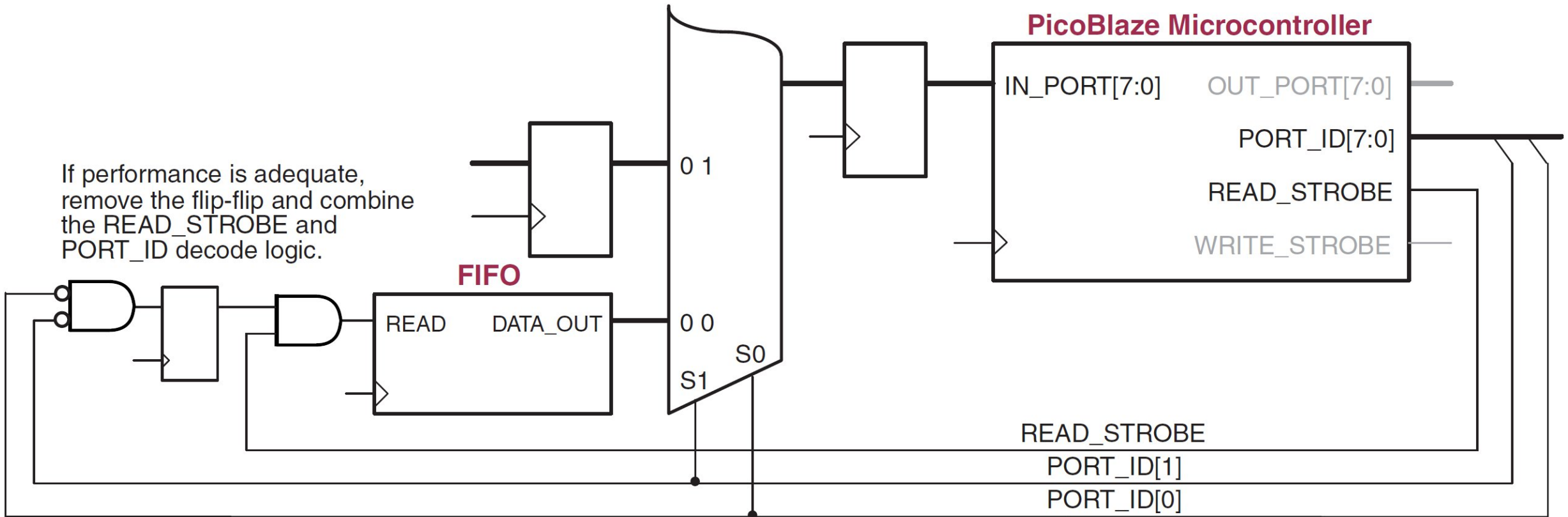


Branje vhodov (5)

The PicoBlaze microcontroller captures the value on IN_PORT[7:0] into register s0 on this clock edge.



Branje vhodov (6)

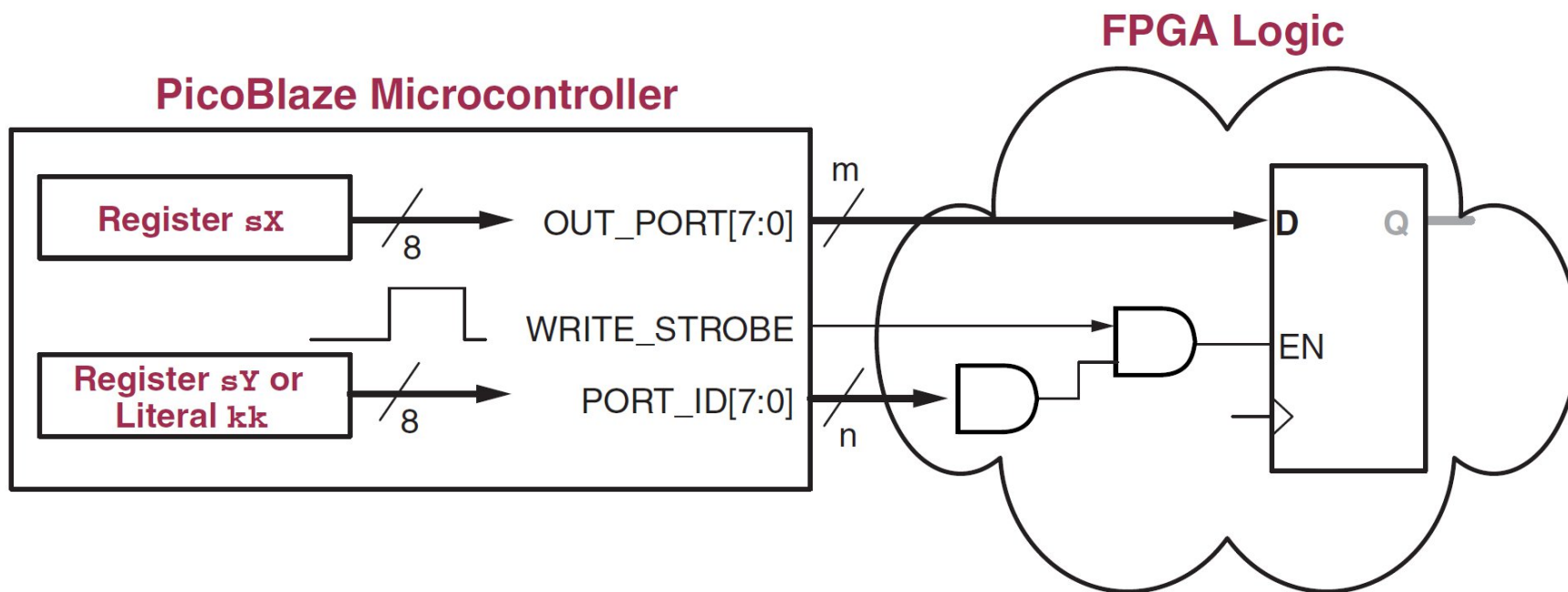


Pisanje na izhod (1)

- Vrednost na izhod postavimo z ukazom OUTPUT
- Sintaksa:
 - OUTPUT sX, pp
 - na izhod z naslovom pp se postavi vsebina registra sX
 - OUTPUT sX, (sY)
 - na izhod z naslovom, ki je shranjen v registru sY, se postavi vsebina registra sX

Pisanje na izhod (2)

- Naslov izhoda se pojavi na signalu PORT_ID
- Na signalu OUT_PORT se pojavi vrednost izhoda
- Ob izvajanju ukaza OUTPUT se na signalu WRITE_STROBE ustvari impulz, ki označuje pisanje izhoda.



Vključitev v projekt (1)

- Izvorna koda mikrokrmilnika in zbirnik sta na voljo na učilnici in na [uradni spletni strani](#)
- Datoteka *kcpsm6_design_template.vhd* vsebuje predlogo kode. V naš projekt kopiramo:
 - deklaracije za:
 - komponenti kcpsm6 in ukazni pomnilnik <your_program>, ki ga preimenujemo, da ustreza datoteki PSM
 - notranje signale

Vključitev v projekt (2)

- za begin
 - instanco processor: kcpsm6
 - popravimo ime signala za uro (clk), da ustreza vhodu v naš glavni modul
 - kcpsm6_sleep <= '0'; -- funkcijo za spanje izklopimo
 - interrupt <= interrupt_ack; -- prekinitve si pogledamo naslednjič
 - instanco program_rom, ki ima ime enako datoteki PSM (brez presledkov!)
 - popravimo ime signala za uro (clk), da ustreza vhodu v naš glavni modul
 - C_FAMILY => "7S"
 - C_RAM_SIZE_KWORDS => 2
 - C_JTAG_LOADER_ENABLE => 1
 - Kodo za preslikavo vhodov/izhodov (če potrebujemo)
 - za učinkovito sintezo upoštevamo navodila glede dekodiranja naslovov port_id
 - vhodi: naslovimo toliko bitov, kolikor je res potrebno: `case port_id(1) downto 0) is`
 - izhodi: en-bit-en-naslov ("one hot"): `if port_id(1) = '1' then -- port 0x2`

Vključitev v projekt (3)

- V projektno mapo `<ime_projekta>.srcs/sources_1/new` kopiramo:
 - `kcpsm6.exe` (če uporabljamo Windows)
 - `kcpsm6.vhd`
 - `ROM_form.vhd`
- V tej mapi tudi naredimo datoteko PSM, ki vsebuje program v zbirniku
 - program lahko urejamo tudi v Vivadu, priporočam spremembo tipa datoteke:
Sources → izberemo datoteko PSM → desni klik → Source File Properties → spremenimo Type v ASM
- Zaženemo prevajalnik
 - `kcpsm6.exe <ime_programa>.psm`
 - napredi lepo formatirano datoteko `.fmt`
 - odprtokodna implementacija Opbasm: <https://kevinpt.github.io/opbasm>
`opbasm <ime_programa>.psm -6 -m 2048 -s 64`
- V Vivadu: Add sources → `kcpsm6.vhd` in `<ime_programa>.vhd`

Izziv

Z uporabo mikrokrmilnika PicoBlaze realizirajte naslednje:

- vseh 16 diod LED želimo krmiliti s stikali
- Gumb BTNC naj določi način:
 - $BTNC = 0 \rightarrow$ stanje LED diod ustreza stanju stikal: $LED[i] = SW[i]$
 - $BTNC = 1 \rightarrow$ invertirano stanje: $LED[i] = \text{not } SW[i]$