EE 459/500 – HDL Based Digital Design with Programmable Logic

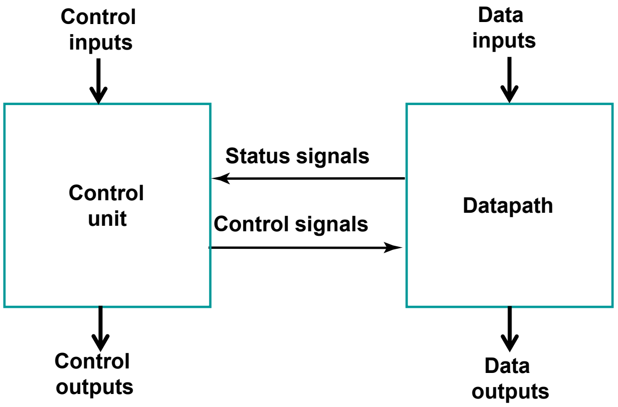
Lecture 17

From special-purpose FSMD to general-purpose microcontroller: Xilinx's PicoBlaze

Overview

- From FSMD to Microcontroller
- PicoBlaze architecture
- PicoBlaze programming
- Summary
- Credits and references

FSMD



Control inputs

Data inputs

Control unit

Status signals

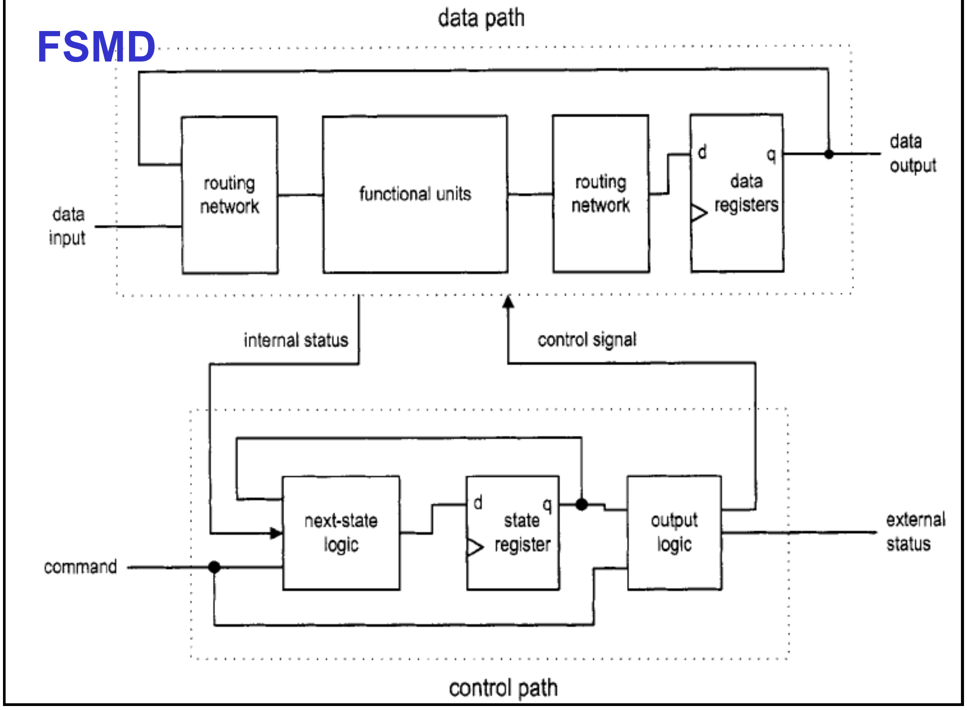
Control signals

Datapath

Control outputs

Data outputs

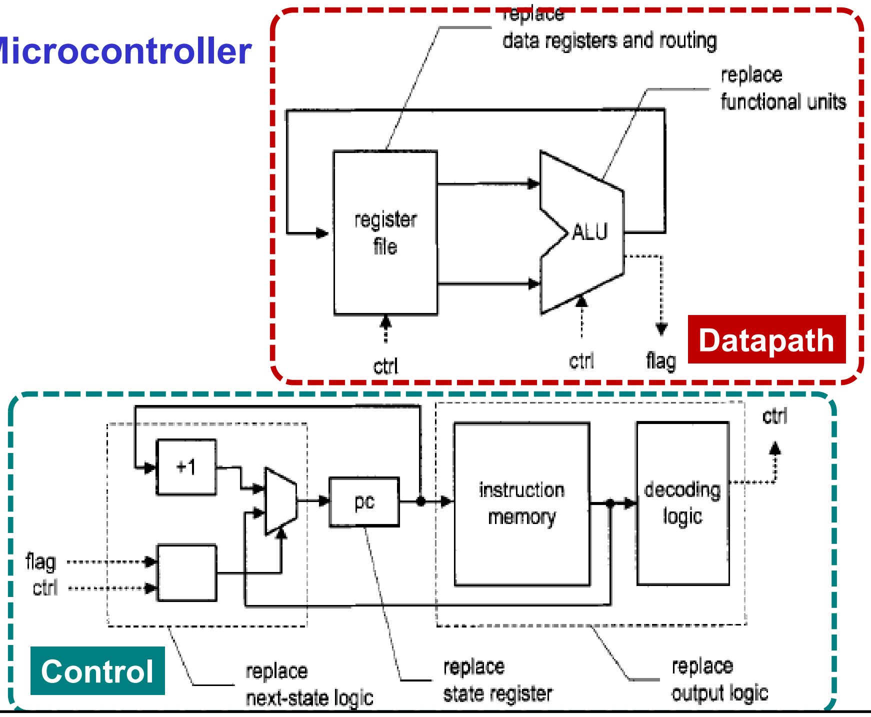
FSMD



From special-purpose FSMD to general-purpose Microcontroller

- FSMD:
  - Components, registers, routing of registers' input and output, number and type of functional units, control of FSM are tailored to specific application
- Alternative:
  - ***Keep the same hardware but use customized software for different applications***
- Transformation of FSMD to a Microcontroller:
  - Use a fixed datapath
  - Use a programmable state machine

Microcontroller



Microcontroller: a general-purpose hardware platform

- Datapath can perform only RT operations:
  - `rd ← r1 op r2`
  - r1, r2: addresses of source registers
  - rd: address of destination register
  - op: one of the available ALU functions
- Control FSM
  - Program counter (***PC***): represents the current state of the control path
  - Each state activates certain control signals: control operation of datapath. These patterns are encoded into ***instructions*** stored in instruction memory.
  - Typically next state is the value of current state plus 1. But there is no limitation where to go next from a given state. Done via several special instructions, such as ***jump***.
- Customizing the system = develop a new sequence of instructions (develop a ***software program***).

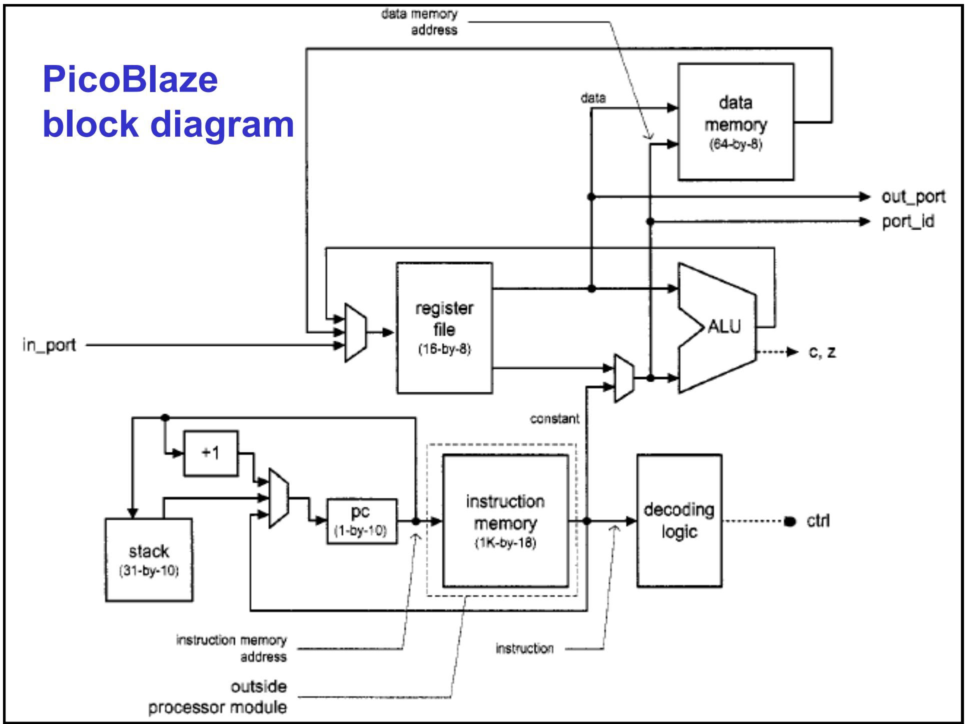
Overview

- From FSMD to Microcontroller
- PicoBlaze architecture
- PicoBlaze programming
- Summary
- Credits and references

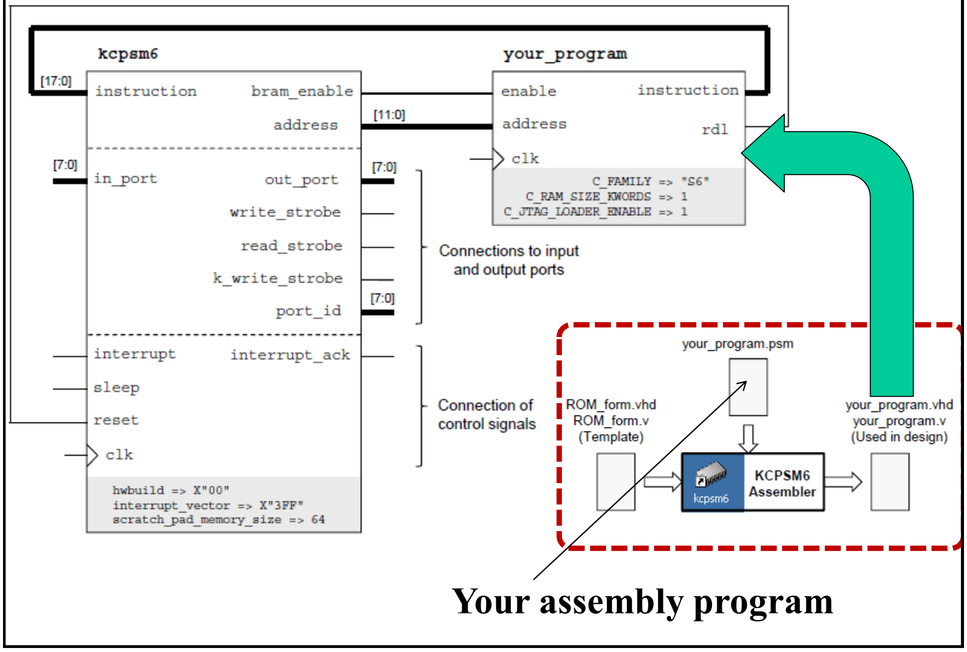
PicoBlaze microcontroller

- RISC architecture of 8-bits
- Latest version is **KCPSM6**
- 8-bit address and data port for access to a wide range of peripherals
- Main characteristics include:
  - Only 26 Slices plus program memory (BRAM).
  - Performance 52 MIPS to 120 MIPs depending on device family and clock rate.
  - Supports programs up to 4K instructions.
  - 32 General Purpose Registers arranged in 2 banks.
  - 256 General Purpose Input Ports.
  - 256 General Purpose Output Ports.
  - 16 Constant-Optimized Output Ports.
  - 64-bytes of scratch pad memory expandable to 128 and 256-bytes (additional 2 and 6 Slices).
  - Fully automatic CALL/RETURN stack supporting nested subroutines to 30 levels.
  - Interrupt with user definable interrupt vector and maximum response time of 4 clock cycles.
  - Power saving features including 'sleep' mode.

PicoBlaze block diagram



PicoBlaze KCPSM6: top-level components and connections



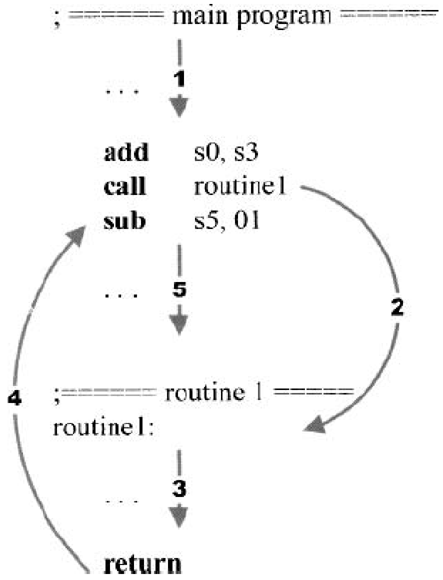
Overview

- From FSMD to Microcontroller
- PicoBlaze architecture
- PicoBlaze programming
- Summary
- Credits and references

Structure of main program

```
    call  initialization_routine
forever:
    call  task_1_routine
    call  task_2_routine
    ...
    call  task_n_routine
    jump  forever
```

Flow of subroutine call



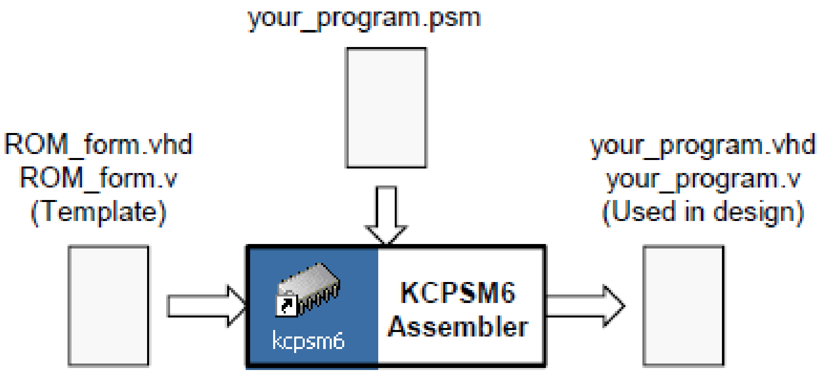
```
; ====== main program ======
   . . .
   add    s0, s3
   call   routine1
   sub    s5, 01
   . . .

;====== routine 1 =====
routine1:
   . . .
   return
```

Steps for Program Development

1. Derive the pseudocode of the ***main program***.
2. Identify tasks in the main program and define them as subroutines.

   If needed, continue refining the complex subroutines and divide them into smaller subroutines.
1. Determine (manually) the register and data RAM use.
2. Write the assembly code for the subroutines.

Compilation with KCPSM6.EXE



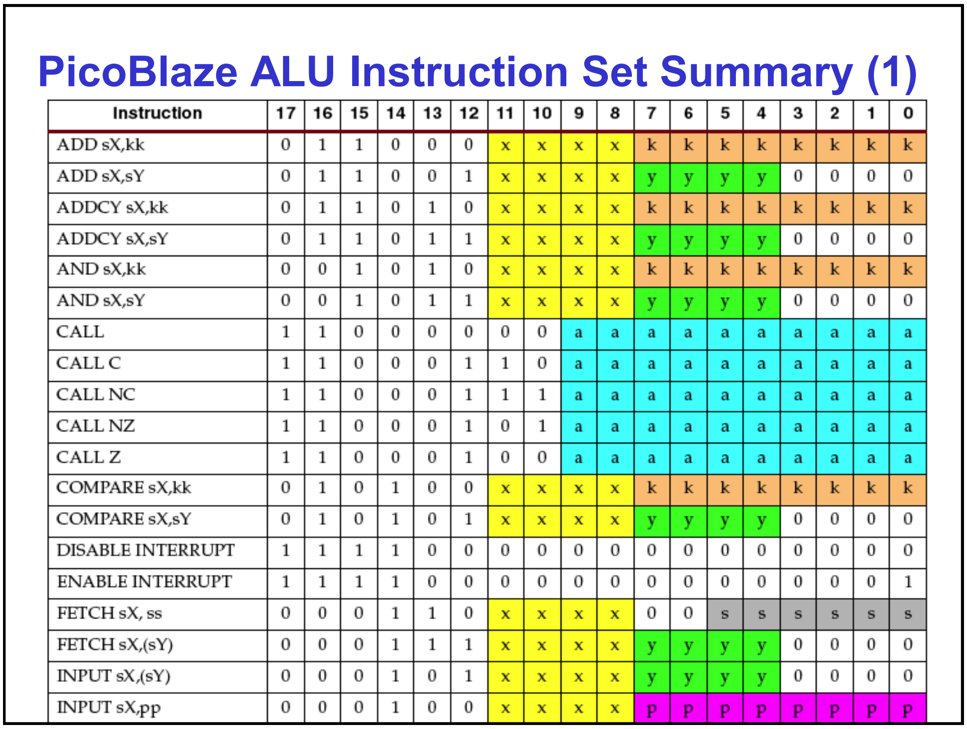
your_program.psm

ROM_form.vhd
ROM_form.v
(Template)

kcpsm6

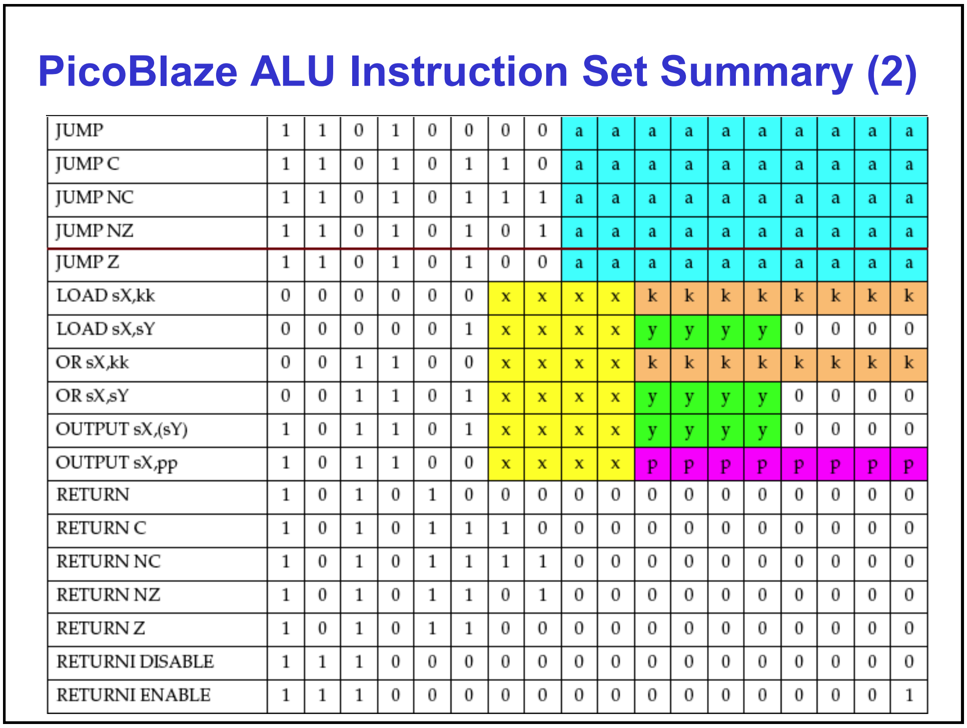
KCPSM6 Assembler

your_program.vhd
your_program.v
(Used in design)



PicoBlaze ALU Instruction Set Summary (1)

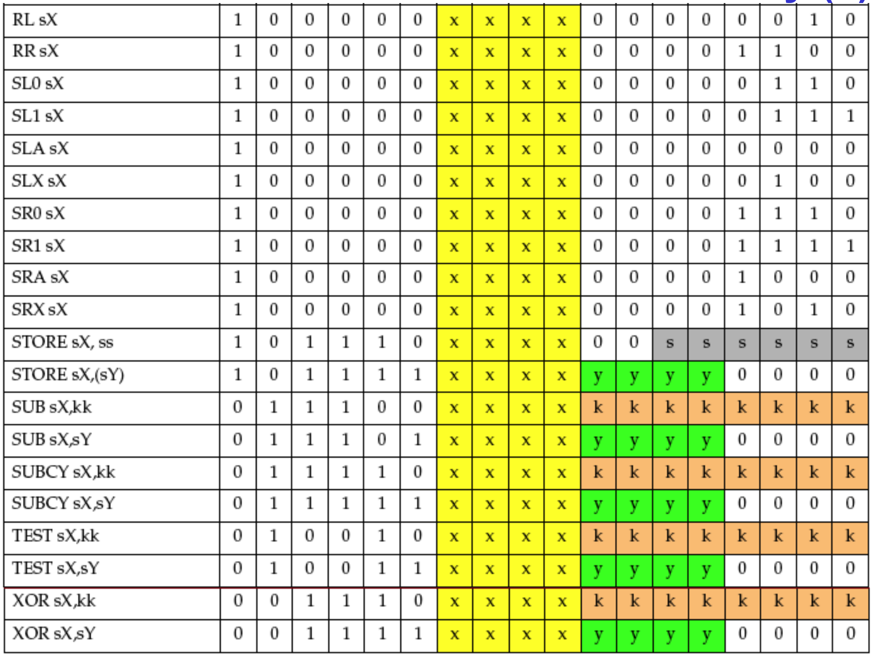
Instruction	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADD sX,kk	0	1	1	0	0	0	x	x	x	x	k	k	k	k	k	k	k	k
ADD sX,sY	0	1	1	0	0	1	x	x	x	x	y	y	y	y	0	0	0	0
ADDCY sX,kk	0	1	1	0	1	0	x	x	x	x	k	k	k	k	k	k	k	k
ADDCY sX,sY	0	1	1	0	1	1	x	x	x	x	y	y	y	y	0	0	0	0
AND sX,kk	0	0	1	0	1	0	x	x	x	x	k	k	k	k	k	k	k	k
AND sX,sY	0	0	1	0	1	1	x	x	x	x	y	y	y	y	0	0	0	0
CALL	1	1	0	0	0	0	0	0	a	a	a	a	a	a	a	a	a	a
CALL C	1	1	0	0	0	1	1	0	a	a	a	a	a	a	a	a	a	a
CALL NC	1	1	0	0	0	1	1	1	a	a	a	a	a	a	a	a	a	a
CALL NZ	1	1	0	0	0	1	0	1	a	a	a	a	a	a	a	a	a	a
CALL Z	1	1	0	0	0	1	0	0	a	a	a	a	a	a	a	a	a	a
COMPARE sX,kk	0	1	0	1	0	0	x	x	x	x	k	k	k	k	k	k	k	k
COMPARE sX,sY	0	1	0	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
DISABLE INTERRUPT	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ENABLE INTERRUPT	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
FETCH sX, ss	0	0	0	1	1	0	x	x	x	x	0	0	s	s	s	s	s	s
FETCH sX,(sY)	0	0	0	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0
INPUT sX,(sY)	0	0	0	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
INPUT sX,pp	0	0	0	1	0	0	x	x	x	x	p	p	p	p	p	p	p	p



PicoBlaze ALU Instruction Set Summary (2)

JUMP	1	1	0	1	0	0	0	0	a	a	a	a	a	a	a	a	a	a
JUMP C	1	1	0	1	0	1	1	0	a	a	a	a	a	a	a	a	a	a
JUMP NC	1	1	0	1	0	1	1	1	a	a	a	a	a	a	a	a	a	a
JUMP NZ	1	1	0	1	0	1	0	1	a	a	a	a	a	a	a	a	a	a
JUMP Z	1	1	0	1	0	1	0	0	a	a	a	a	a	a	a	a	a	a
LOAD sX,kk	0	0	0	0	0	0	x	x	x	x	k	k	k	k	k	k	k	k
LOAD sX,sY	0	0	0	0	0	1	x	x	x	x	y	y	y	y	0	0	0	0
OR sX,kk	0	0	1	1	0	0	x	x	x	x	k	k	k	k	k	k	k	k
OR sX,sY	0	0	1	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
OUTPUT sX,(sY)	1	0	1	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
OUTPUT sX,pp	1	0	1	1	0	0	x	x	x	x	p	p	p	p	p	p	p	p
RETURN	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
RETURN C	1	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0
RETURN NC	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0
RETURN NZ	1	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0
RETURN Z	1	0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
RETURNI DISABLE	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
RETURNI ENABLE	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

PicoBlaze ALU Instruction Set Summary (3)



RL sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	0	1	0
RR sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	1	0	0
SL0 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	1	1	0
SL1 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	1	1	1
SLA sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	0	0	0
SLX sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	0	1	0	0
SR0 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	1	1	0
SR1 sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	1	1	1
SRA sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	0	0	0
SRX sX	1	0	0	0	0	0	x	x	x	x	0	0	0	0	1	0	1	0
STORE sX, ss	1	0	1	1	1	0	x	x	x	x	0	0	s	s	s	s	s	s
STORE sX,(sY)	1	0	1	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0
SUB sX,kk	0	1	1	1	0	0	x	x	x	x	k	k	k	k	k	k	k	k
SUB sX,sY	0	1	1	1	0	1	x	x	x	x	y	y	y	y	0	0	0	0
SUBCY sX,kk	0	1	1	1	1	0	x	x	x	x	k	k	k	k	k	k	k	k
SUBCY sX,sY	0	1	1	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0
TEST sX,kk	0	1	0	0	1	0	x	x	x	x	k	k	k	k	k	k	k	k
TEST sX,sY	0	1	0	0	1	1	x	x	x	x	y	y	y	y	0	0	0	0
XOR sX,kk	0	0	1	1	1	0	x	x	x	x	k	k	k	k	k	k	k	k
XOR sX,sY	0	0	1	1	1	1	x	x	x	x	y	y	y	y	0	0	0	0

Example: 8-bit Binary to BCD Converter

- Done in class

Summary

- We arrived naturally to Microcontrollers from FSMDs
- Microcontrollers:
  - Same hardware but use customized software for different applications
  - Flexibility

References and Credits

- Chapters 14,15 of:
  - Pong P. Chu, FPGA Prototyping by VHDL Examples: Xilinx Spartan-3 Version, Wiley 2008.
- PicoBlaze 8-bit Embedded Microcontroller User Guide
  - http://www.xilinx.com/products/intellectual-property/picoblaze.htm
- http://en.wikipedia.org/wiki/PicoBlaze