CPE 626
Advanced VLSI Design
Lecture 2

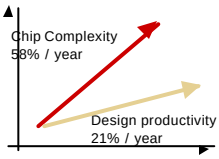
Aleksandar Milenkovic

http://www.ece.uah.edu/~milenka
http://www.ece.uah.edu/~milenka/cpe626-04F/
milenka@ece.uah.edu

Assistant Professor
Electrical and Computer Engineering Dept.
University of Alabama in Huntsville

The Need for IP Cores

- Benefits of HDL-based design
  - Portability
  - Technology independence
  - Design cycle reduction
  - Automatic synthesis and Logic optimization
- … But, the gap between available chip complexity and design productivity continues to increase



⇒ Use IP cores

New Generation of Designers …

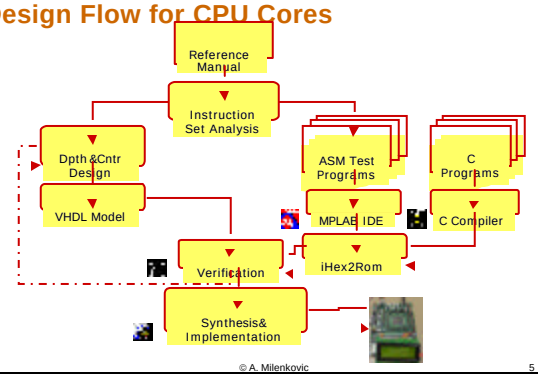
- Emphasis on hierarchical IP core design
- Design systems, not components!
- Understand hardware/software co-design
- Understand and explore design tradeoffs between complexity, performance, and power consumption

⇒ Design a soft processor/micro-controller core

UAH Library of Soft Cores

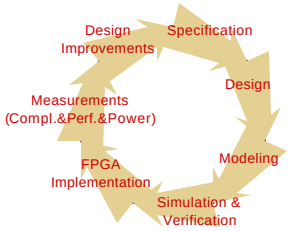
- Microchip's PIC18 micro-controller
- Microchip's PIC16 micro-controller
- Intel's 8051
- ARM Integer CPU core
- FP10 Floating-point Unit (ARM)
- Advanced Encryption Standard (AES)
- Video Processing System on a Chip

Design Flow for CPU Cores



Soft IP Engineering Cycle

- Encompasses all relevant steps



- Put together knowledge in digital design, HDLs, computer architecture, programming languages
- State-of-the-art devices
- Work in teams

PIC18 Greetings

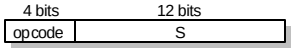
http://www.ece.uah.edu/~milenka/pic18/pic.html

Designing a simple CPU in 60 minutes

- LaCASA step-by-step tutorial
  - http://www.ece.uah.edu/~lacasa/tutorials/mu0/mu0tutorial.html
- Design, verify, implement, and prototype a rudimentary processor MU0
- Modeling using VHDL
- Simulation using ModelSim
- Implement using Xilinx ISE and a SpartanII device

MU0 – A Simple Processor

- Instruction format



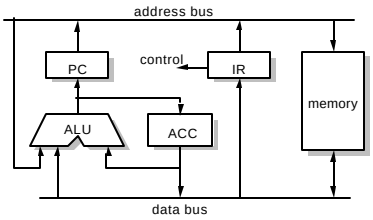
- Instruction set

Instruction	Opcode	Effect
LDA S	0000	ACC := mem_{16}[S]
STO S	0001	mem_{16}[S] := ACC
ADD S	0010	ACC := ACC + mem_{16}[S]
SUB S	0011	ACC := ACC - mem_{16}[S]
JMP S	0100	PC := S
JGE S	0101	if ACC >= 0 PC := S
JNE S	0110	if ACC !=0 PC := S
STP	0111	stop

MU0 Datapath Example

- Program Counter – PC
- Accumulator - ACC
- Arithmetic-Logic Unit – ALU
- Instruction Register
- Instruction Decode and Control Logic

Follow the principle that the memory will be limiting factor in design: each instruction takes exactly the number of clock cycles defined by the number of memory accesses it must take.

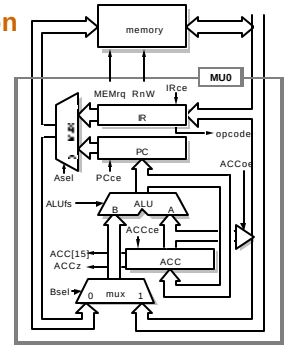


MU0 Datapath Design

- Assume that each instruction starts when it has arrived in the IR
- Step 1: EX (execute)
  - LDA S: ACC <- Mem[S]
  - STO S: Mem[S] <- ACC
  - ADD S: ACC <- ACC + Mem[S]
  - SUB S: ACC <- ACC - Mem[S]
  - JMP S: PC <- S
  - JGE S: if (ACC >= 0) PC <- S
  - JNE S: if (ACC != 0) PC <- S
- Step 2: IF (fetch the next instruction)
  - Either PC or the address in the IR is issued to fetch the next instruction
  - address is incremented in the ALU and value saved into the PC
- Initialization
  - Reset input to start executing instructions from a known address; here it is 000hex
    - provide zero at the ALU output and then load it into the PC register

MU0 RTL Organization

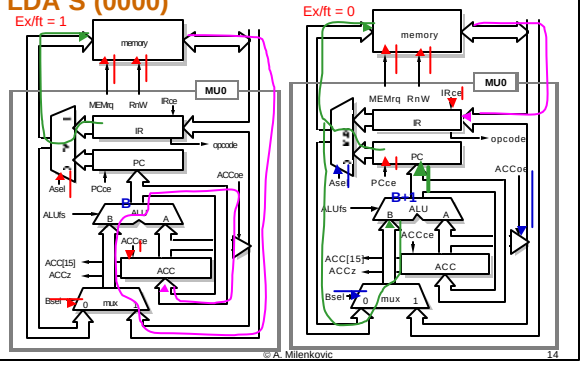
- Control Logic
  - Asel
  - Bsel
  - ACCce (ACC change enable)
  - PCce (PC change enable)
  - IRce (IR change enable)
  - ACCoe (ACC output enable)
  - ALUfs (ALU function select)
  - MEMrq (memory request)
  - RnW (read/write)
  - Ex/ft (execute/fetch)



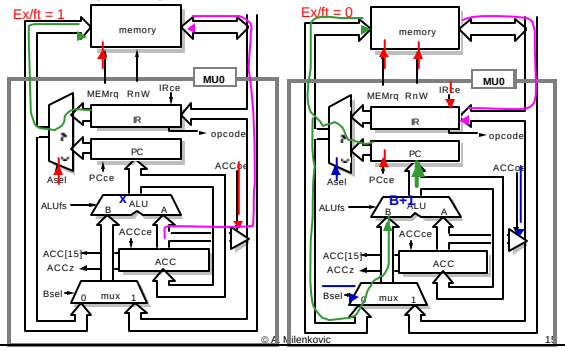
MU0 control logic

	Inputs					Outputs									
	Opcode		Ex/ft		ACC15		Bsel		PCce		ACCoe		MEMrq		Ex/ft
Instruction			Reset		ACCz		Asel	ACCce	IRce		ALUfs		RnW		
Reset	xxxx	1	x	x	x	0	0	1	1	1	0	= 0	1	1	0
LDA S	0000	0	0	x	x	1	1	1	0	0	0	= B	1	1	1
	0000	0	1	x	x	0	0	0	1	1	0	B+1	1	1	0
STO S	0001	0	0	x	x	1	x	0	0	0	1	x	1	0	1
	0001	0	1	x	x	0	0	0	1	1	0	B+1	1	1	0
ADD S	0010	0	0	x	x	1	1	1	0	0	0	A+B	1	1	1
	0010	0	1	x	x	0	0	0	1	1	0	B+1	1	1	0
SUB S	0011	0	0	x	x	1	1	1	0	0	0	A-B	1	1	1
	0011	0	1	x	x	0	0	0	1	1	0	B+1	1	1	0
JMP S	0100	0	x	x	x	1	0	0	1	1	0	B+1	1	1	0
JGE S	0101	0	x	x	0	1	0	0	1	1	0	B+1	1	1	0
	0101	0	x	x	1	0	0	0	1	1	0	B+1	1	1	0
JNE S	0110	0	x	0	x	1	0	0	1	1	0	B+1	1	1	0
	0110	0	x	1	x	0	0	0	1	1	0	B+1	1	1	0
STOP	0111	0	x	x	x	1	x	0	0	0	0	x	0	1	0

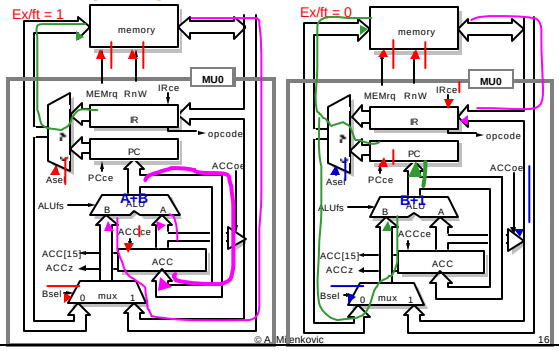
LDA S (0000)



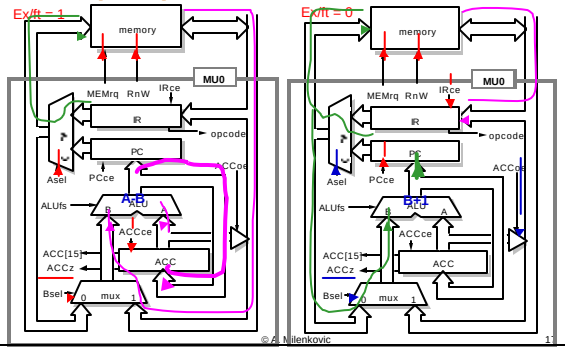
STO S (0001)



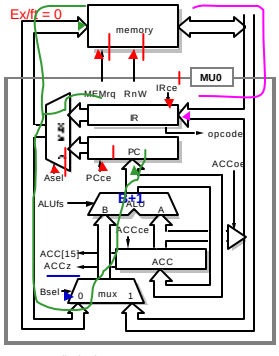
ADD S (0010)



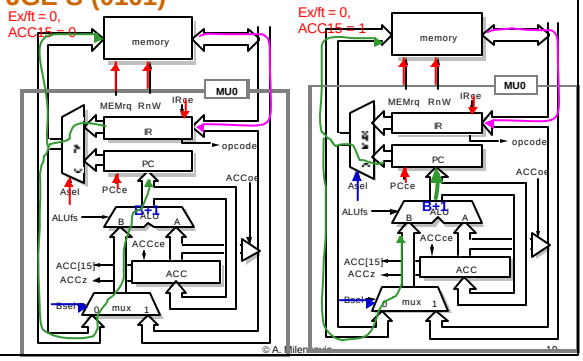
SUB S (0011)



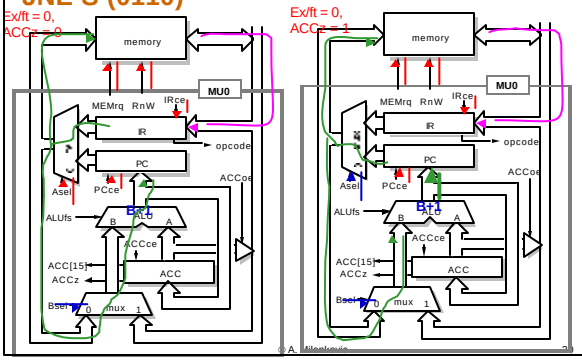
JMP S (0100)



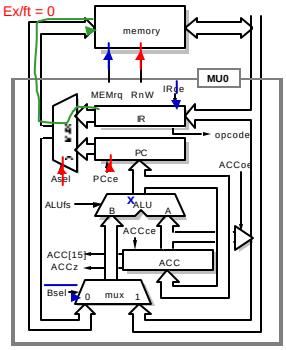
JGE S (0101)



JNE S (0110)



STP (001)



Reset