Using FPGAs to create a complete computer system for the classroom

Marcela Melara (WS '12)
Advisor: Marc Corliss
Hobart and William Smith Colleges, Geneva, NY
Summer 2010

FPGAs

- **Definition** - ***FPGA (Field-Programmable Gate Array)***: Integrated circuit made to be configured by the designer
  - **used to build reconfigurable digital circuits → undefined function at time of manufacture → must be reconfigured (i.e. programmed) before use**
- **FPGAs consist of grid of programmable logic elements, which are "wired together" when device is configured**
- **Configuration design specified by a hardware description language (e.g. Verilog)**
- **Common applications:**
  - **digital signal processing**
  - **car multimedia systems**
  - **medical imaging**
  - **speech recognition**
  - **many, many more...**
- **Our application:**

  processor core that can be flexibly modified according to gradual improvements

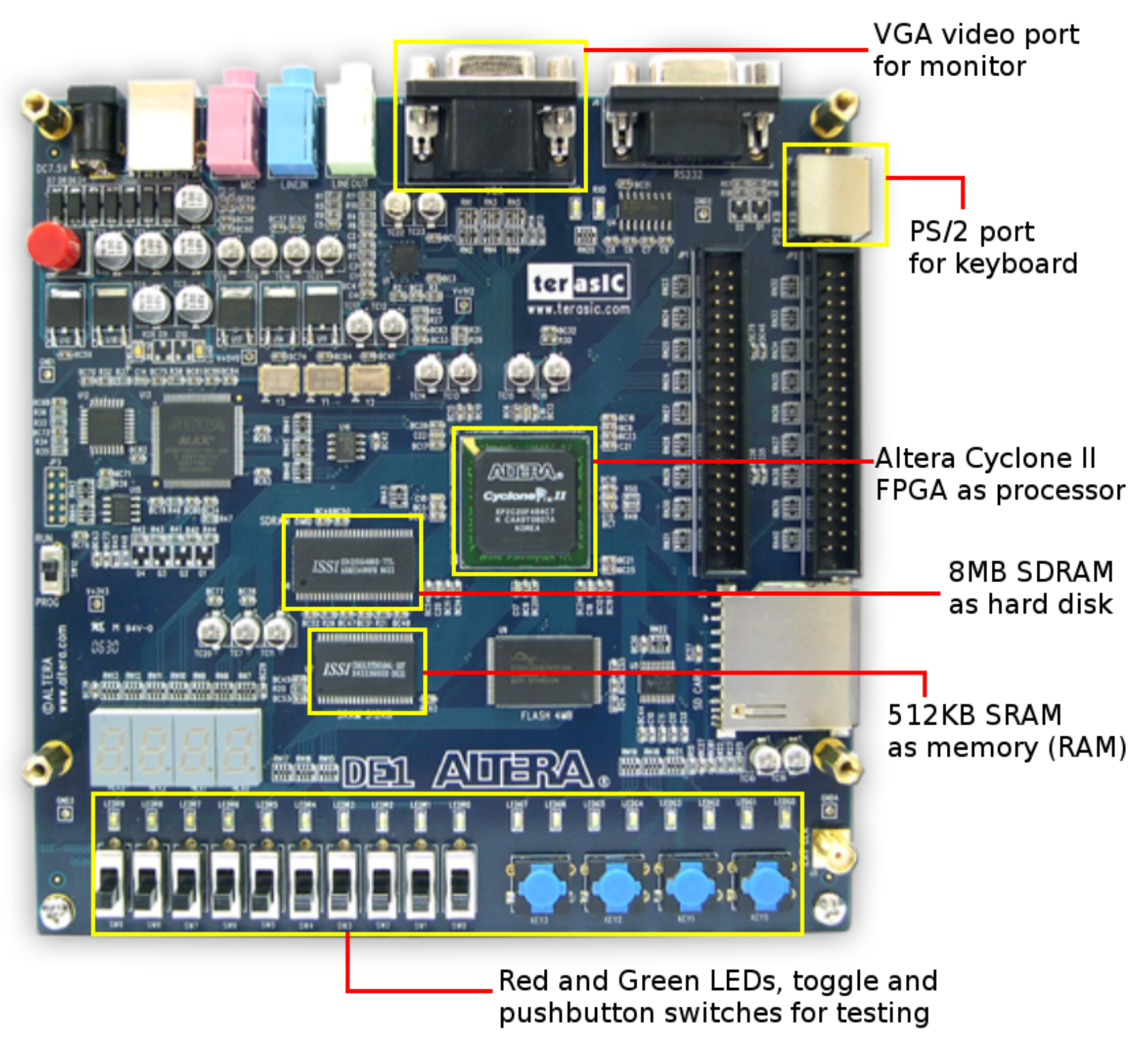

VGA video port for monitor
PS/2 port for keyboard
Altera Cyclone II FPGA as processor
8MB SDRAM as hard disk
512KB SRAM as memory (RAM)
Red and Green LEDs, toggle and pushbutton switches for testing

Fig. 1 Altera DE1 Board Overview; components used in the project are marked
image source: www.woorimtni.co.kr/terasic/

Overview

- **Project goal: create computer system for pedagogical purposes**
  - **will be used in CS courses such as CPSC 220 Computer Architecture or CPSC 431 Operating Systems**
- **Used FPGAs to implement processor for the computer system**
- **Why use FPGAs?**
  - **flexibly programmable soft-processors, faster than software simulation, cheaper**
- **Major tasks of project:**
  1) design and program basic version of Larc processor with RAM
  2) incorporate support for monitor and keyboard for interaction with computer
  3) add support for an operating system (simple OS compatible with Larc)
  4) incorporate hard disk to support file system
  5) add support for multi-tasking via timer
- **Possible future projects:**
  1) continue enhancing processor's performance
  2) create more application-specific processors for use in other scientific fields

A complete system

- **Hardware components of the complete computer:**
  - **processor and memory (RAM)**
  - **keyboard and monitor**
  - **hard disk**
- **Each of these components has a controller to provide an abstract interface for the processor**
- **Definition** – ***Controller***: A digital circuit which interfaces two separate pieces of hardware and controls the flow of data between them
- **All the components are connected by a high-speed bus**
- **Definition** – ***Bus***: A system of parallel wires that transfer data between computer components

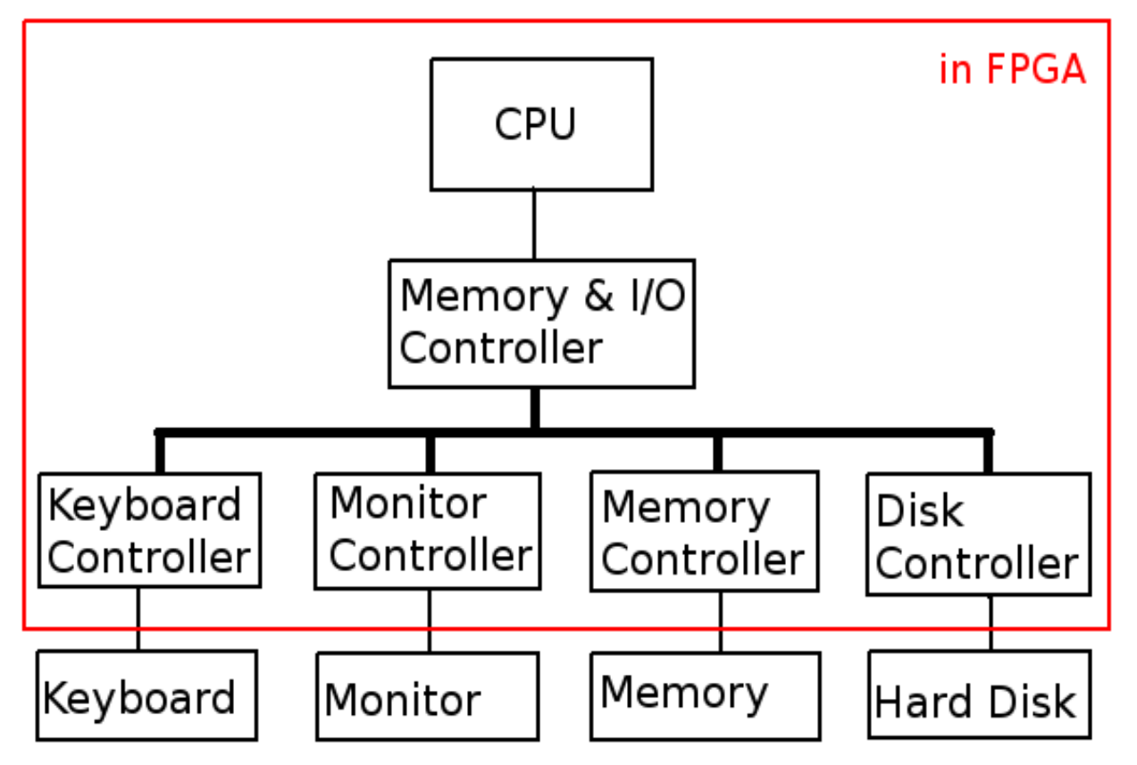


Fig. 2 Broad overview of the Larc processor

Fig. 3 The complete computer system

The Larc Processor

Processor Schematic:

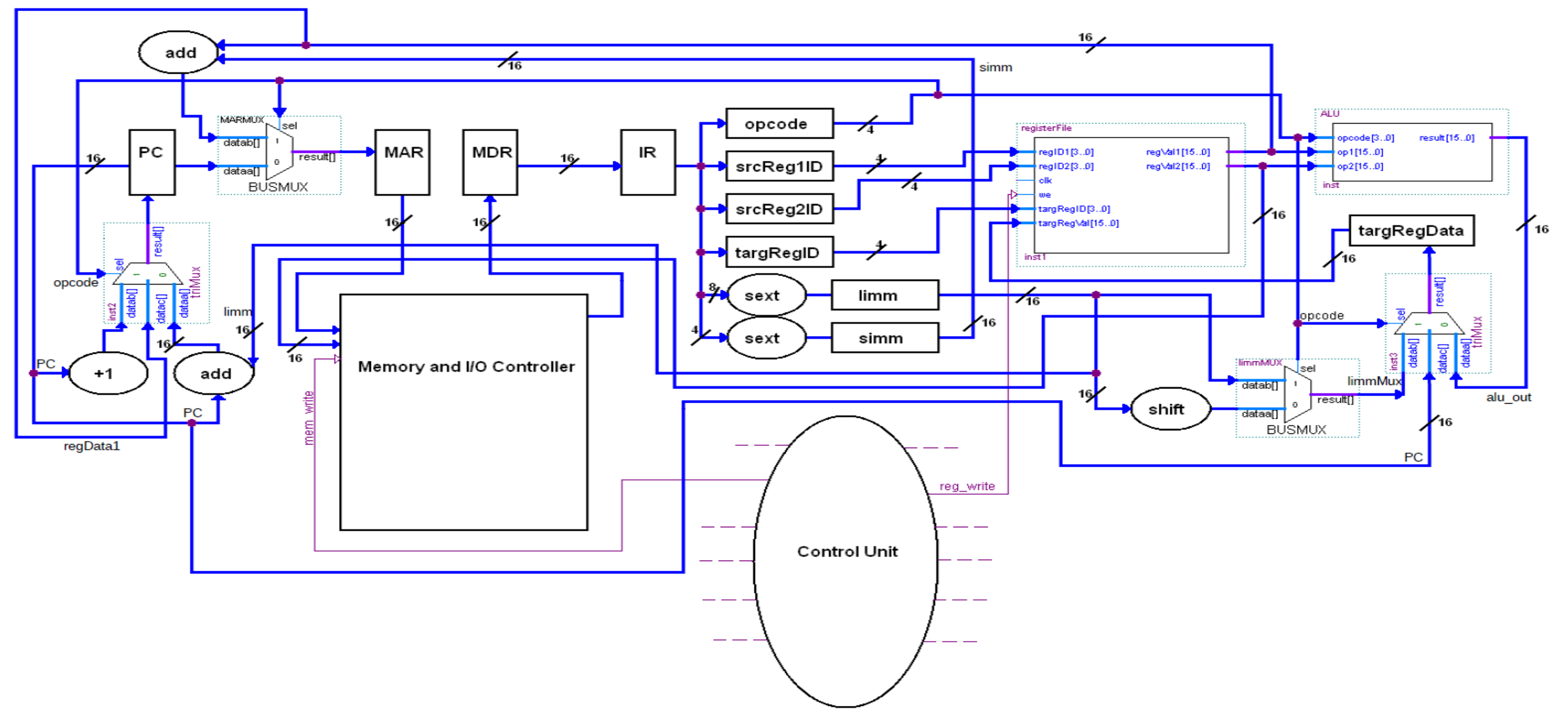


Fig. 4 Larc Processor schematic

- **Major components of the processor:**
  - **Registers (e.g. PC, MAR, opcode etc.)**
  - **ALU**
  - **Register File**
  - **Memory and I/O controller**
  - **Control Unit**
- **Data flow in the processor is managed by the Control Unit**

Legend:

Blue lines denote data transfer

Purple lines denote control signals

Processor Control:

Legend:

Yellow circles denote the states of the control unit

Arrows show direction of control flow

- **The Control Unit in the processor is managed via a Finite State machine**
- **Definition** – ***Finite State Machine***: Behavioral model which specifies a finite set of states, a state transition function, and state events
- **Finite State machines are implemented in hardware through a combinational and sequential circuit**

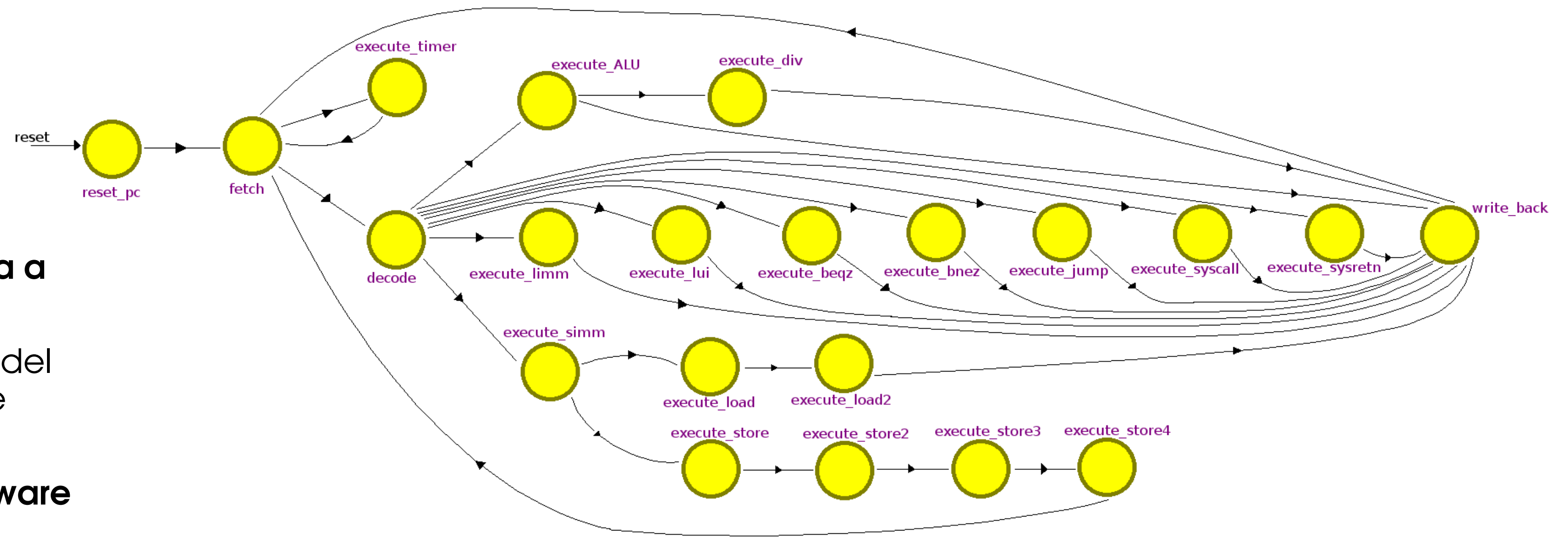
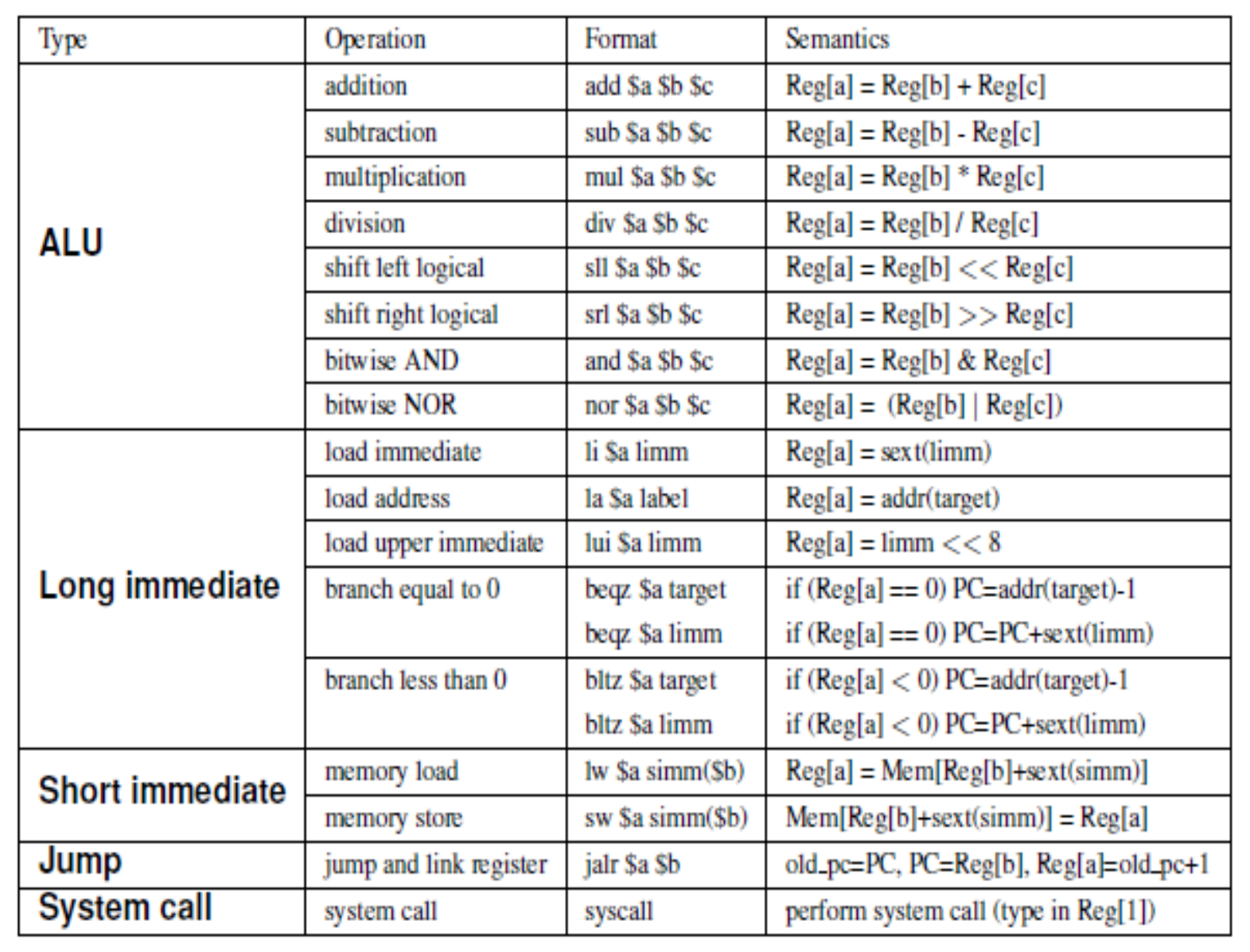


Fig. 5 Control Finite State Machine

The Larc ISA

- **Definition** - ***ISA (Instruction Set Architecture)***: Specification of set of instructions a processor can execute, and memory layout
- **Subset of MIPS**
- **Processor width: 16 bits**
- **16 instructions: 8 ALU, 2 branches, 2 memory, 2 immediate loads, 1 jump, and 1 system call**
- **Memory: 16-bit address, 16-bit data → address space 2^{16} → total memory size = 128KB**
- **16 Registers: 13 general-purpose, 3 specialized → form the register file**
- **Uses memory-mapped I/O for input/output communication**



Type	Operation	Format	Semantics
ALU	addition	add \$a \$b \$c	Reg[a] = Reg[b] + Reg[c]
	subtraction	sub \$a \$b \$c	Reg[a] = Reg[b] - Reg[c]
	multiplication	mul \$a \$b \$c	Reg[a] = Reg[b] * Reg[c]
	division	div \$a \$b \$c	Reg[a] = Reg[b] / Reg[c]
	shift left logical	sll \$a \$b \$c	Reg[a] = Reg[b] << Reg[c]
	shift right logical	srl \$a \$b \$c	Reg[a] = Reg[b] >> Reg[c]
	bitwise AND	and \$a \$b \$c	Reg[a] = Reg[b] & Reg[c]
	bitwise NOR	nor \$a \$b \$c	Reg[a] = (Reg[b] | Reg[c])
Long immediate	load immediate	li \$a limm	Reg[a] = sext(limm)
	load address	la \$a label	Reg[a] = addr(target)
	load upper immediate	lui \$a limm	Reg[a] = limm << 8
	branch equal to 0	beqz \$a target	if (Reg[a] == 0) PC=addr(target)-1
		beqz \$a limm	if (Reg[a] == 0) PC=PC+sext(limm)
	branch less than 0	bltz \$a target	if (Reg[a] < 0) PC=addr(target)-1
		bltz \$a limm	if (Reg[a] < 0) PC=PC+sext(limm)
Short immediate	memory load	lw \$a simm(\$b)	Reg[a] = Mem[Reg[b]+sext(simm)]
	memory store	sw \$a simm(\$b)	Mem[Reg[b]+sext(simm)] = Reg[a]
Jump	jump and link register	jalr \$a \$b	old_pc=PC, PC=Reg[b], Reg[a]=old_pc+1
System call	system call	syscall	perform system call (type in Reg[1])

Fig. 6 Base Larc Assembly instructions

The System Architecture

- **The System Architecture is the part of the ISA that specifies support for an operating system**
- **Definition** - ***Operating System (OS)***: A software program (or a set thereof) which acts as an intermediary between application programs and computer hardware
- **Added support for a simple OS to the processor via the implementation of a system call instruction**
  - **User applications communicate with the OS through this instruction**
- **OS polls I/O devices through memory-mapped registers**
  - **A simple (inefficient) way to communicate with devices (e.g. monitor, keyboard)**
- **Added support for multi-tasking using a timer in the processor**
  - **Multiple processes execute concurrently, timer allows for preemptive switching**

References

Hamblen, James O., Tyson S. Hall, and Michael D. Furman. *Rapid Prototyping of Digital Systems: SOPC Edition*. New York, NY: Springer, 2008. Print
Patterson, David A., and John L. Hennessey. *Computer Organization and Design*. 3rd ed. San Francisco, CA: Elsevier, 2005. Print.
www.altera.com www.alteraforums.com