

Extending a MIPS Simulator for FPGA Support

Pete Sanderson, Otterbein University

Ken Vollmar, Missouri State University

FPGA overview

- Field Programmable Gate Array (FPGA) is “programmable hardware.”
- Uses “hardware simulation” to execute a circuit design
- Design a circuit using components:
 - gates,
 - “mega-functions” (e.g., ALU),
 - Intellectual property (IP)
 - VHDL or Verilog hardware design language
- Available in a range of capacity, cost, and environment
- Very attractive in an educational environment: flexible, re-usable, low cost.

MARS Overview

- **MIPS Assembler** and **Runtime Simulator**
- An IDE for MIPS assembly language programming
- Designed for use with *Computer Organization and Design* by Patterson & Hennessy
- First release in 2005. Two releases per year.
- Extended to support FPGA
 - Configurable memory model
 - Export memory contents to text file
- JAR download www.cs.missouristate.edu/MARS

G:\WARS\Wars4.1\release\Fibonacci.asm - MARS 4.1

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Fibonacci.asm

```

1 # Compute first twelve Fibonacci numbers and put in array, then print
2 .globl main
3 .data
4 fibs: .word 0 : 12      # "array" of 12 words to contain fib values
5 size: .word 12          # size of "array"
6 .text
7 main:
8     la $t0, fibs         # load address of array
9     la $t5, size         # load address of size variable
10    lw $t5, 0($t5)        # load array size
11    li $t2, 1             # 1 is first and second Fib. number
12    sw $t2, 0($t0)        # F[0] = 1
13    sw $t2, 4($t0)        # F[1] = F[0] = 1
14    addi $t1, $t5, -2     # Counter for loop, will execute (size-2) times
15 loop: lw $t3, 0($t0)     # Get value from array F[n]
16        lw $t4, 4($t0)     # Get value from array F[n+1]
17        add $t2, $t3, $t4   # $t2 = F[n] + F[n+1]
18        sw $t2, 8($t0)     # Store F[n+2] = F[n] + F[n+1] in array
19        addi $t0, $t0, 4    # increment address of Fib. number source
20        addi $t1, $t1, -1   # decrement loop counter
21        bgtz $t1, loop     # repeat if not finished yet.
22        la $a0, fibs       # first argument for print (array)
23        add $a1, $zero, $t5 # second argument for print (size)
24        jal print          # call print routine.
25        li $v0, 10         # system call for exit

```

Line: 1 Column: 1 ☒ Show Line Numbers

Mars Messages Run I/O

Clear

Registers		Coproc 1	Coproc 0
Name	Number	Value	
\$zero	0	0x00000000	
\$at	1	0x00000000	
\$v0	2	0x00000000	
\$v1	3	0x00000000	
\$a0	4	0x00000000	
\$a1	5	0x00000000	
\$a2	6	0x00000000	
\$a3	7	0x00000000	
\$t0	8	0x00000000	
\$t1	9	0x00000000	
\$t2	10	0x00000000	
\$t3	11	0x00000000	
\$t4	12	0x00000000	
\$t5	13	0x00000000	
\$t6	14	0x00000000	
\$t7	15	0x00000000	
\$s0	16	0x00000000	
\$s1	17	0x00000000	
\$s2	18	0x00000000	
\$s3	19	0x00000000	
\$s4	20	0x00000000	
\$s5	21	0x00000000	
\$s6	22	0x00000000	
\$s7	23	0x00000000	
\$t8	24	0x00000000	
\$t9	25	0x00000000	
\$k0	26	0x00000000	
\$k1	27	0x00000000	
\$gp	28	0x00001800	
\$sp	29	0x00003ffc	
\$fp	30	0x00000000	
\$ra	31	0x00000000	
pc		0x00000000	
hi		0x00000000	
lo		0x00000000	



G:\WARS\Wars4.1\release\Fibonacci.asm - MARS 4.1

File Edit Run Settings Tools Help

Run speed at max (no interaction)

Edit Execute

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00000000	0x20082000	addi \$8,\$0,0x2000	8: la \$t0, fibs # load address of ...
	0x00000004	0x200d2030	addi \$13,\$0,0x2030	9: la \$t5, size # load address of ...
	0x00000008	0x8dad0000	lw \$13,0x0000(\$13)	10: lw \$t5, 0(\$t5) # load array size
	0x0000000c	0x240a0001	addiu \$10,\$0,0x0001	11: li \$t2, 1 # 1 is first and s...
	0x00000010	0xad0a0000	sw \$10,0x0000(\$8)	12: sw \$t2, 0(\$t0) # F[0] = 1
	0x00000014	0xad0a0004	sw \$10,0x0004(\$8)	13: sw \$t2, 4(\$t0) # F[1] = F[0] = 1
	0x00000018	0x21a9fffe	addi \$9,\$13,0xfffe	14: addi \$t1, \$t5, -2 # Counter for loop...
	0x0000001c	0x8d0b0000	lw \$11,0x0000(\$8)	15: loop: lw \$t3, 0(\$t0) # Get value from a...
	0x00000020	0x8d0c0004	lw \$12,0x0004(\$8)	16: lw \$t4, 4(\$t0) # Get value from a...
	0x00000024	0x016c5020	add \$10,\$11,\$12	17: add \$t2, \$t3, \$t4 # \$t2 = F[n] + F[n+1]
	0x00000028	0xad0a0008	sw \$10,0x0008(\$8)	18: sw \$t2, 8(\$t0) # Store F[n+2] = F...
	0x0000002c	0x21a9fffe	addi \$8,\$8,0x0004	19: addi \$t0, \$t0, 4 # increment address

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00002000	0x00000001	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002020	0x00000000	0x00000000	0x00000000	0x00000000	0x0000000c	0x68540020	0x69462065	0x616e6f62
0x00002040	0x20696363	0x626d756e	0x20737265	0x3a657261	0x0000000a	0x00000000	0x00000000	0x00000000
0x00002060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x000020e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00002100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

0x00002000 (.data) ☒ Hexadecimal Addresses ☒ Hexadecimal Values ☐ ASCII

Mars Messages Run I/O

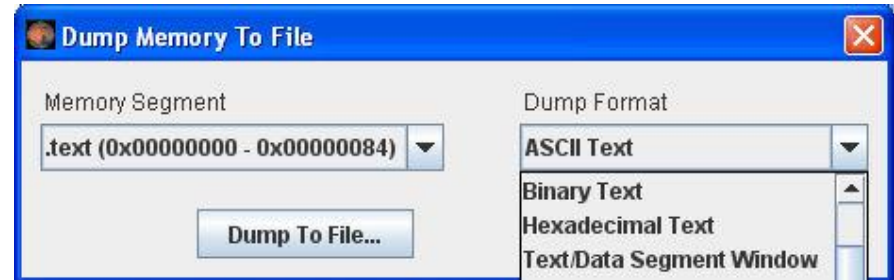
Reset: reset completed.

Clear

Registers

Name	Coproc 1	Coproc 0
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00002000
\$t1	9	0x00000000
\$t2	10	0x00000001
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x0000000c
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x00001800
\$sp	29	0x00003ffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00000014
hi		0x00000000
lo		0x00000000

MARS features for FPGA support

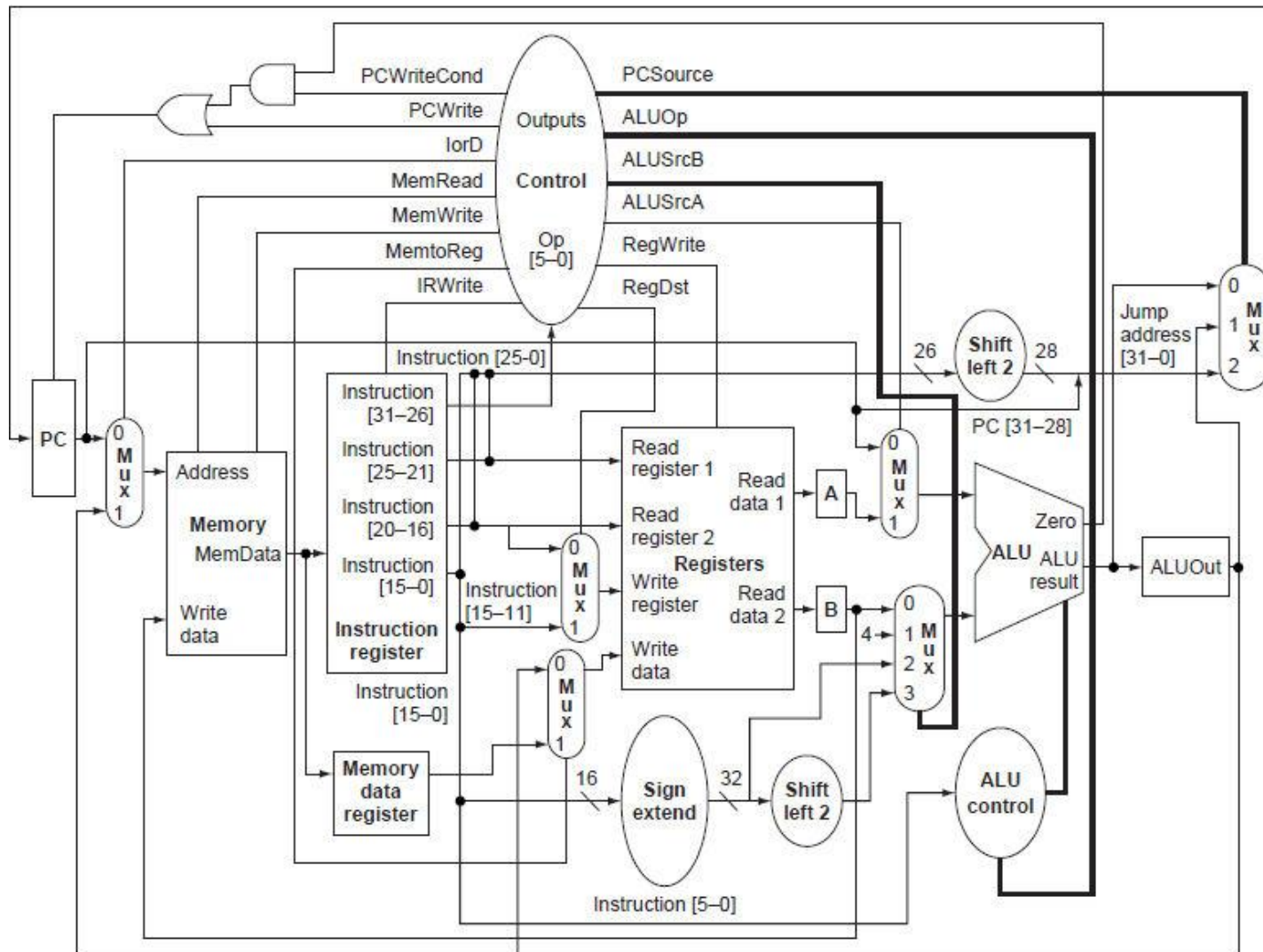


MIPS circuit design alternatives

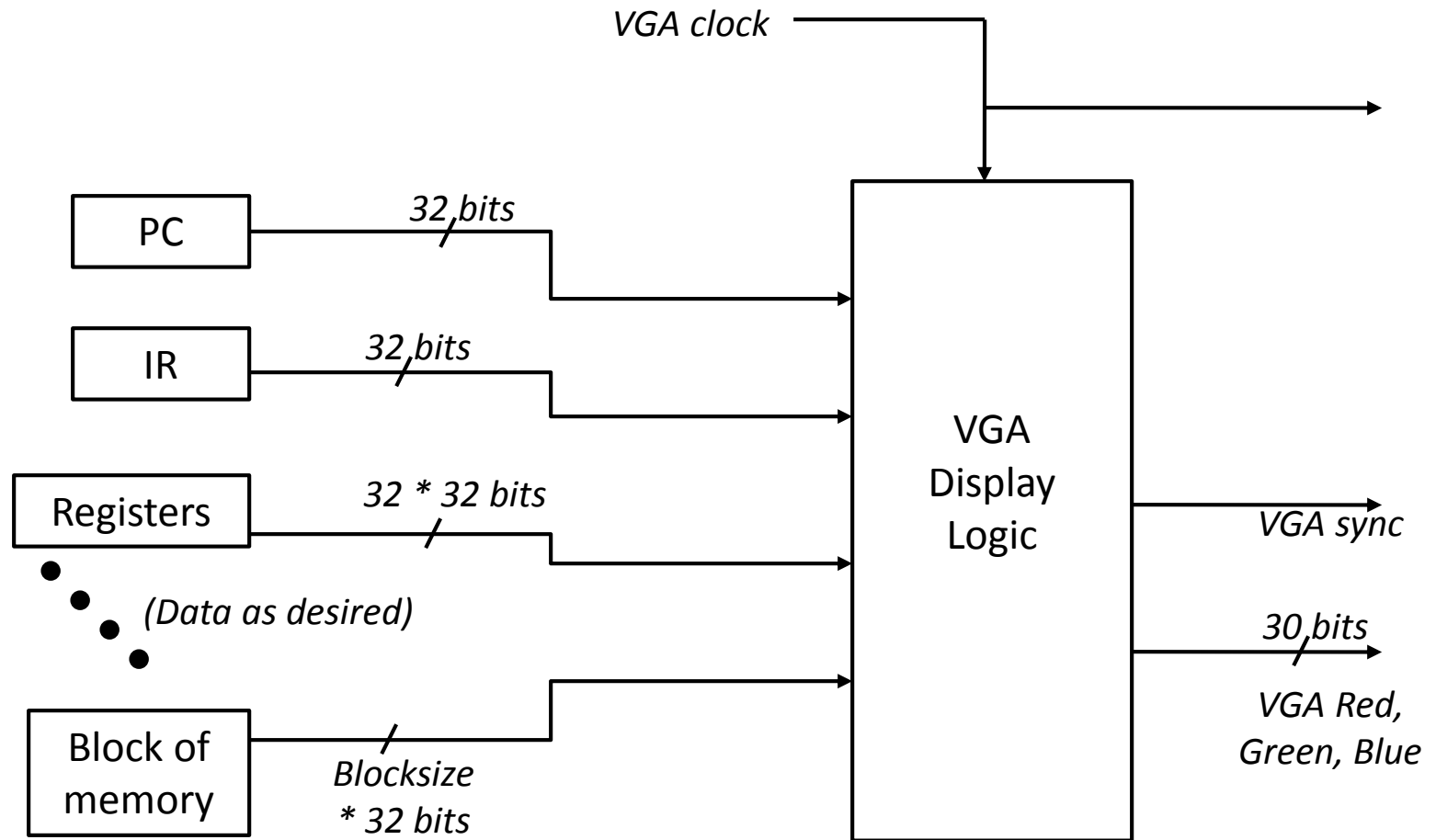
- Single-cycle
- Multi-cycle
- Pipelined

Two different circuits necessary

- MIPS circuit and the VGA display of the MIPS state
- Issue of “MIPS clock speed” vs. “VGA clock speed”
- MIPS memory not implemented in the FPGA itself, inefficient use of FPGA hardware simulation resources
- Instead, MIPS memory is contained in a separate RAM on the FPGA board
- This involves separate “memory load” steps for the MIPS memory and the MIPS circuit



Computer Organization and Design, 3rd Edition, Figure 5-28.



Usage in an architecture class

- Reinforce the concept of MIPS execution on various platforms:
 - A genuine MIPS processor
 - The MARS simulator of MIPS execution (running in Java on some other processor)
 - FPGA simulation of a MIPS processor (running with variable clock speed)
- Use spec sheets and manufacturer's data to estimate run times of different FPGA and MIPS implementations
- Add instructions to the MIPS implementation on FPGA

Acknowledgements

The FPGA project was supported by a SIGCSE
Special Projects Grant

Questions?