

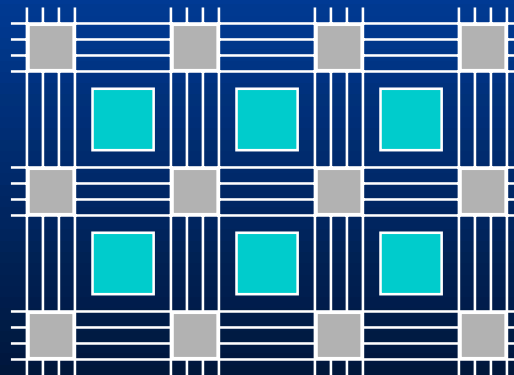
Hands-on Computer Architecture – Teaching Processor and Integrated Systems Design with FPGAs

Jan Gray

Gray Research LLC

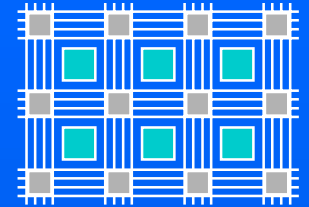
jsgray@acm.org

www.fpgacpu.org



`lw r12, 4(r3)`

Introduction



◆ *Confession - not an educator; unproven ideas*

◆ Recent large, cheap FPGAs and tools enable “desktop processor development”

◆ RISC4005 (Freidin 1990)

◆ j32 (Gray 1995)

◆ XSOC/xr16 Project Kit (Gray 1998)

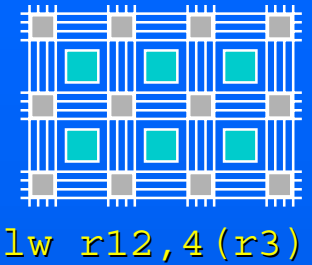
◆ xr16: simple pipelined 16-bit RISC, 33 MHz, 1.4 CPI

◆ XSOC: system-on-a-chip: bus and peripherals

◆ Compiler, simulator, specs, documentation, demos

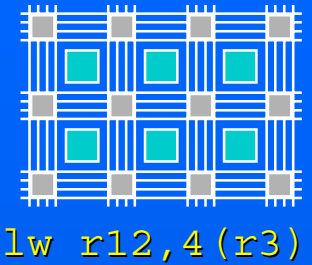
◆ *Can we (should we) use FPGA CPU and SoC kits to teach computer architecture?*

Outline



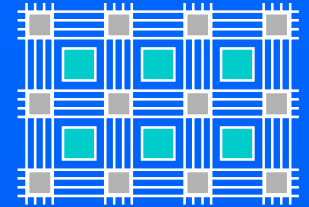
- ◆ The XSOC/xr16 Project Kit
- ◆ Why use a hardware project to teach architecture?
- ◆ Suggested FPGA-based projects
- ◆ FPGAs for advanced architecture study
- ◆ Related work and ongoing work

Enabling Developments



- ◆ Xilinx Student Edition v1.5 (\$100)
 - ◆ Xilinx Foundation Express 1.5
 - ◆ Verilog and VHDL synthesis, schematic capture, simulation
 - ◆ Xilinx FPGA implementation tools
 - ◆ Vanden Bout's *Practical Xilinx Designer Lab Book*
- ◆ XESS XS40 (\$109-\$199)
 - ◆ XC4005 or XC4010, 32-128 KB RAM, 8031, programmable oscillator, parallel, VGA, mouse/keyboard, LED, breadboard interface, *tools, documentation, support mailing list*
- ◆ LCC, a retargetable C compiler (*free*)
 - ◆ Documented in textbook, small, fairly easy to port or enhance
- ◆ Veriwell free Verilog simulator

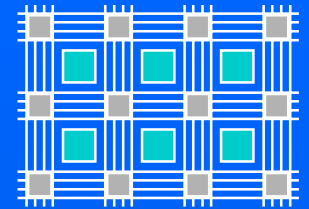
XSOC/xr16 Kit Goals



lw r12,4(r3)

- ◆ *Make integrated system dev more accessible*
- ◆ Subject of *Circuit Cellar* article series
 - ◆ Explain FPGA processor and system design
 - ◆ Build a fast, simple, thrifty, real pipelined CPU
 - ◆ Build CPU+SoC in a tiny XC4005XL
 - ◆ Help launch a free cores development community
 - ◆ Provide on-chip bus architecture for easy cores reuse
 - ◆ Show FPGA CPUs can be practical (=cheap)
 - ◆ Show C compiler support
 - ◆ Show efficient FPGA design “best practices”

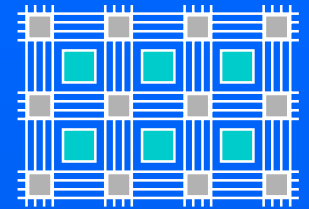
Software Development Tools



`lw r12, 4(r3)`

- ◆ Need C compiler for nascent 16-bit RISC:
port LCC [Fraser and Hanson]
 - ◆ Choose calling conventions
 - ◆ `mips.md` → `xr16.md` (500 LOCs changed)
 - ◆ 32 registers → 16 registers
 - ◆ `sizeof(int) == sizeof(void*) == 4` → 2
 - ◆ Misc: branches; long ints; software mul/div
- ◆ Assembler (1300 LOCs)
 - ◆ Lex, parse, fix far branches, apply fixups, emit
- ◆ Instruction set simulator (400 LOCs, 3 MIPS)

LCC Generated Code



lw r12,4(r3)

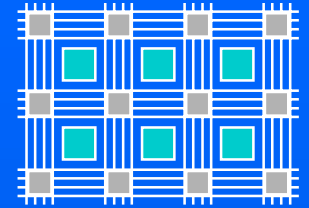
```
◆ typedef struct TN {  
    int k;  
    struct TN *left,*right;  
} *T;
```

```
T search(int k, T p) {  
    while (p && p->k != k)  
        if (p->k < k)  
            p = p->right;  
        else  
            p = p->left;  
    return p;  
}
```

```
◆ _search:  
    br    L3  
L2: lw    r9,(r4)  
    cmp  r9,r3    ; p->k < k?  
    bge  L5  
    lw   r4,4(r4) ; p=p->right  
    br   L6  
L5: lw    r4,2(r4) ; p=p->left  
L6:  
L3: mov   r9,r4  
    cmp  r9,r0    ; p==0?  
    beq  L7  
    lw   r9,(r4)  
    cmp  r9,r3    ; p->k != k?  
    bne  L2  
L7: mov   r2,r4    ; retval=p  
L1: ret
```

; r3=k r4=p

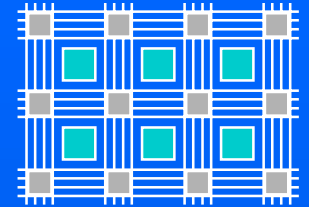
XR16 Instruction Set Design



`lw r12, 4(r3)`

- ◆ Goals and constraints
 - ◆ Compact 16-bit instructions
 - ◆ 16 16-bit registers
 - ◆ Cover integer C operations
 - ◆ KISS
 - ◆ Fast → pipelined → regular
 - ◆ Byte addressable → load/store bytes/words
 - ◆ `disp(reg)` addressing
 - ◆ Long ints → add with carry, shift extended
 - ◆ Scale to 32-bit registers
- ◆ Common instructions
 - ◆ `lw sw mov lea call br cmp`
 - ◆ `mov lea cmp` → add/sub
 - ◆ `disp(reg)` addressing (69%)
- ◆ Resolved:
 - ◆ 3 operand: add sub addi
 - ◆ 4-bit immediates, 12-bit immediate prefix
 - ◆ no condition codes; interlocked add/sub+*bcond*
 - ◆ fast call/return → call/jal
 - ◆ mul/div/shifts in software

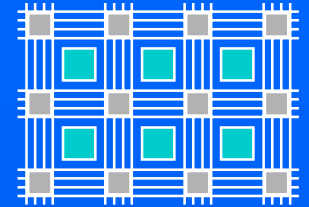
XR16 Instruction Set



`lw r12,4(r3)`

Hex	Fmt	Assembler	Semantics	N
0dab	rrr	add rd,ra,rb	rd = ra + rb;	1
1dab	rrr	sub rd,ra,rb	rd = ra - rb;	1
2dai	rri	addi rd,ra,imm	rd = ra + imm;	1
3d*b	rr	{and or xor andn adc sbc} rd,rb	rd = rd <i>op</i> rb;	1
4d*i	ri	{andi ori xori andni adci sbci slli slxi srai srli srxi} rd,imm	rd = rd <i>op</i> imm;	1
5dai	rri	lw rd,imm(ra)	rd = *(int*)(ra+imm);	2
6dai	rri	lb rd,imm(ra)	rd = *(byte*)(ra+imm);	2
8dai	rri	sw rd,imm(ra)	*(int*)(ra+imm) = rd;	2
9dai	rri	sb rd,imm(ra)	*(byte*)(ra+imm) = rd;	2
Adai	rri	jal rd,imm(ra)	rd = pc, pc = ra + imm;	3
B*dd	br	{br brn beq bne bc bnc bv bnv blt bge ble bgt bltu bgeu bleu bgtu} label	if (<i>cond</i>) pc += 2*disp8;	*
Ciii	i12	call func	r15 = pc, pc = imm12<<4;	3
Diii	i12	imm imm12	imm'next _{15:4} = imm12;	1
7xxx Exxx Fxxx	-	<i>reserved</i>	<i>reserved</i>	

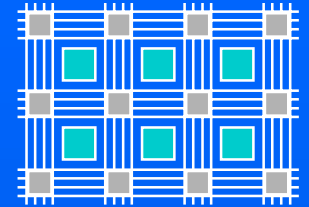
Synthesized Instructions



lw r12,4(r3)

Instruction	Maps to
nop	and r0,r0
mov rd,ra	add rd,ra,r0
cmp ra,rb	sub r0,ra,rb
subi rd,ra,im	addi rd,ra,-im
cmpi ra,im	addi r0,ra,-im
com rd	xori rd,-1
lea rd,imm(ra)	addi rd,ra,imm
lbs rd,imm(ra) <i>(load-byte, sign-extending)</i>	lb rd,imm(ra) xori rd,0x80 subi rd,0x80
j addr	jal r0,addr
ret	jal r0,0(r15)

Compromises



`lw r12,4(r3)`

- ◆ No multiply/divide
- ◆ No barrel shifter: 1-bit shifts only
- ◆ Jumps and taken branches take 3 cycles
 - ◆ Annul the two branch shadow instructions
- ◆ Loads/stores take at least 2 cycles
 - ◆ One shared memory port
- ◆ Simple interrupt/return
- ◆ No floating point

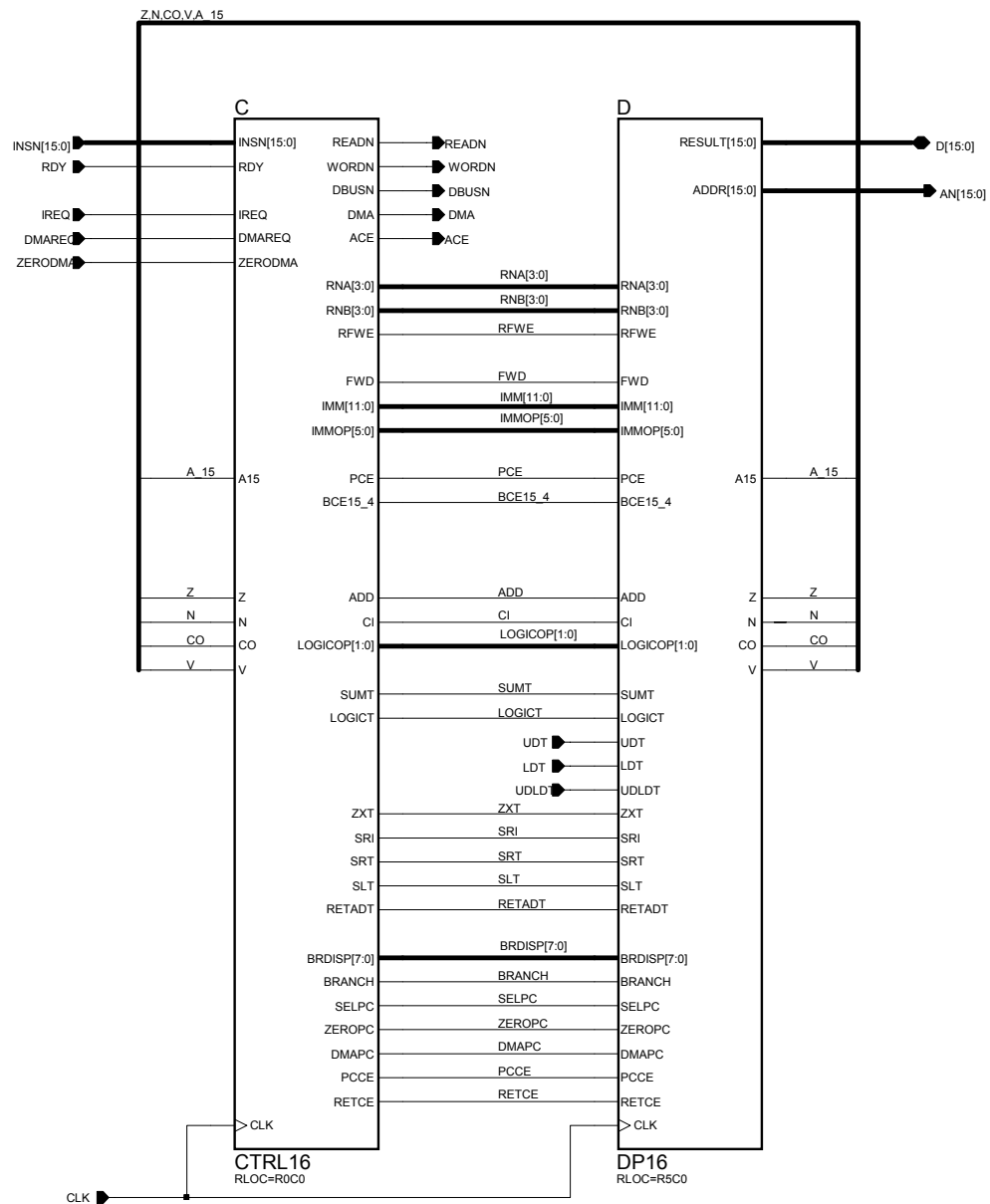
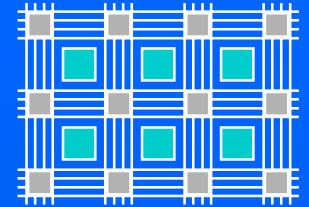


Figure S2: XR16 CPU Schematic

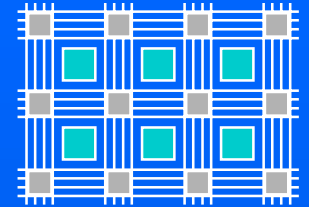
Copyright (C) 2000, Gray Research LLC.	Project: [None]
This work and its use subject to XSOC License Agreement. See LICENSE file.	Macro: XR16
	Date: 02/23/100

Control Unit: Pipeline Stages



`lw r12, 4(r3)`

- ◆ IF: Instruction Fetch
 - ◆ Read instruction at `addr`; prepare $\text{addr} \leftarrow \text{PC} += 2$; capture instruction in Instruction Register (IR)
- ◆ DC: Decode and register file access
 - ◆ Decode instruction; read register operands; prepare immediate field; select operands
- ◆ EX: Execute
 - ◆ Add, logic, shift; select result; write to register file
 - ◆ call/jal: $\text{PC} := \text{sum}$
 - ◆ load/store: $\text{addr} \leftarrow \text{sum}$; stop pipe for access



lw r12,4(r3)

Control Unit: Pipeline Hazards

- ◆ Memory wait states
 - ◆ Stop pipeline
- ◆ Result use data dependence
 - ◆ Result forwarding on one operand only
- ◆ Load/store memory access
 - ◆ Stop pipeline
- ◆ Jumps and taken branches
 - ◆ Annul two following instructions
- ◆ Interrupts
 - ◆ Force *jump-to-int-handler instruction* when safe

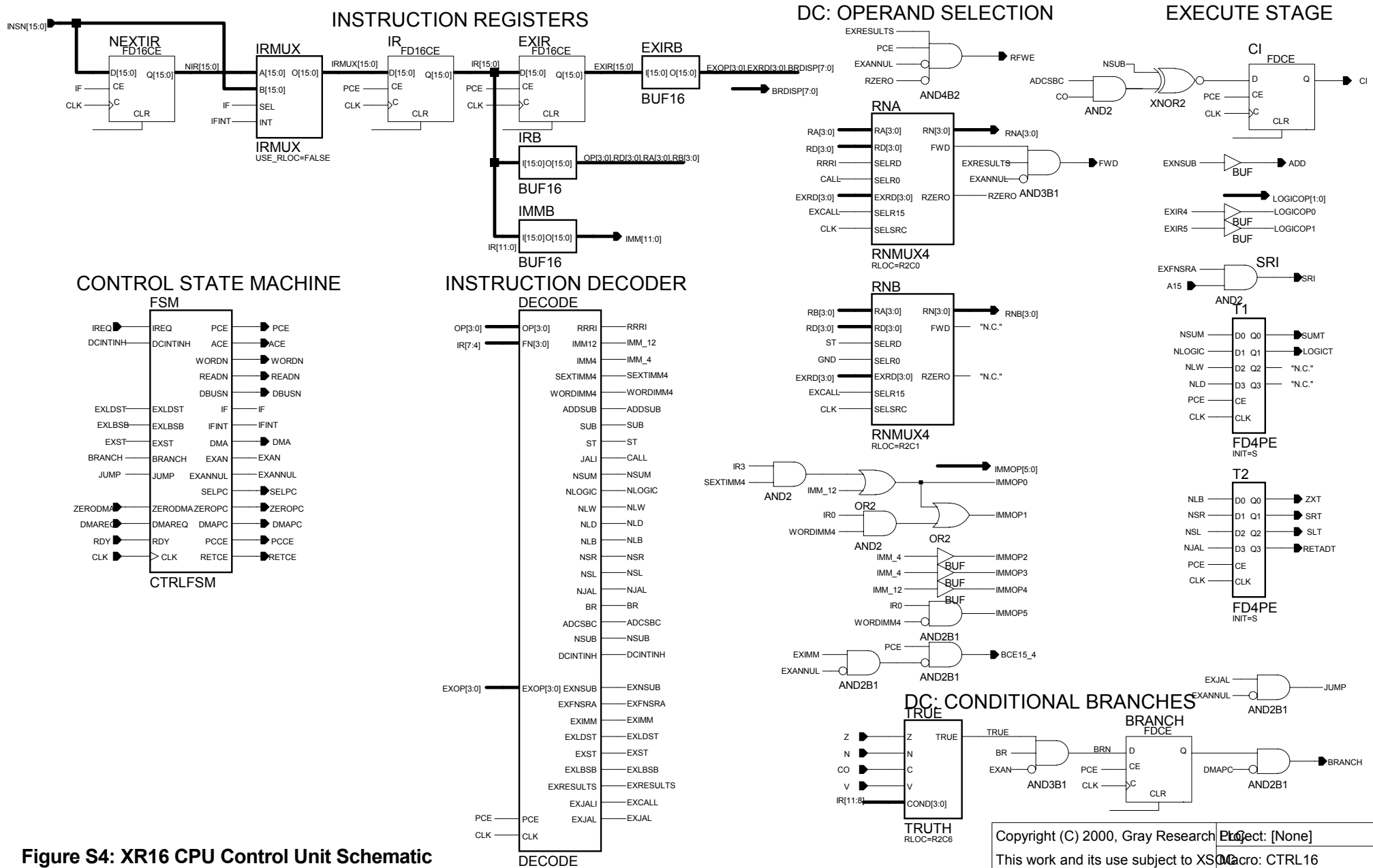


Figure S4: XR16 CPU Control Unit Schematic

Copyright (C) 2000, Gray Research	Project: [None]
This work and its use subject to Xilinx	Macro: CTRL16
License Agreement. See LICENSE	Date: 02/23/100

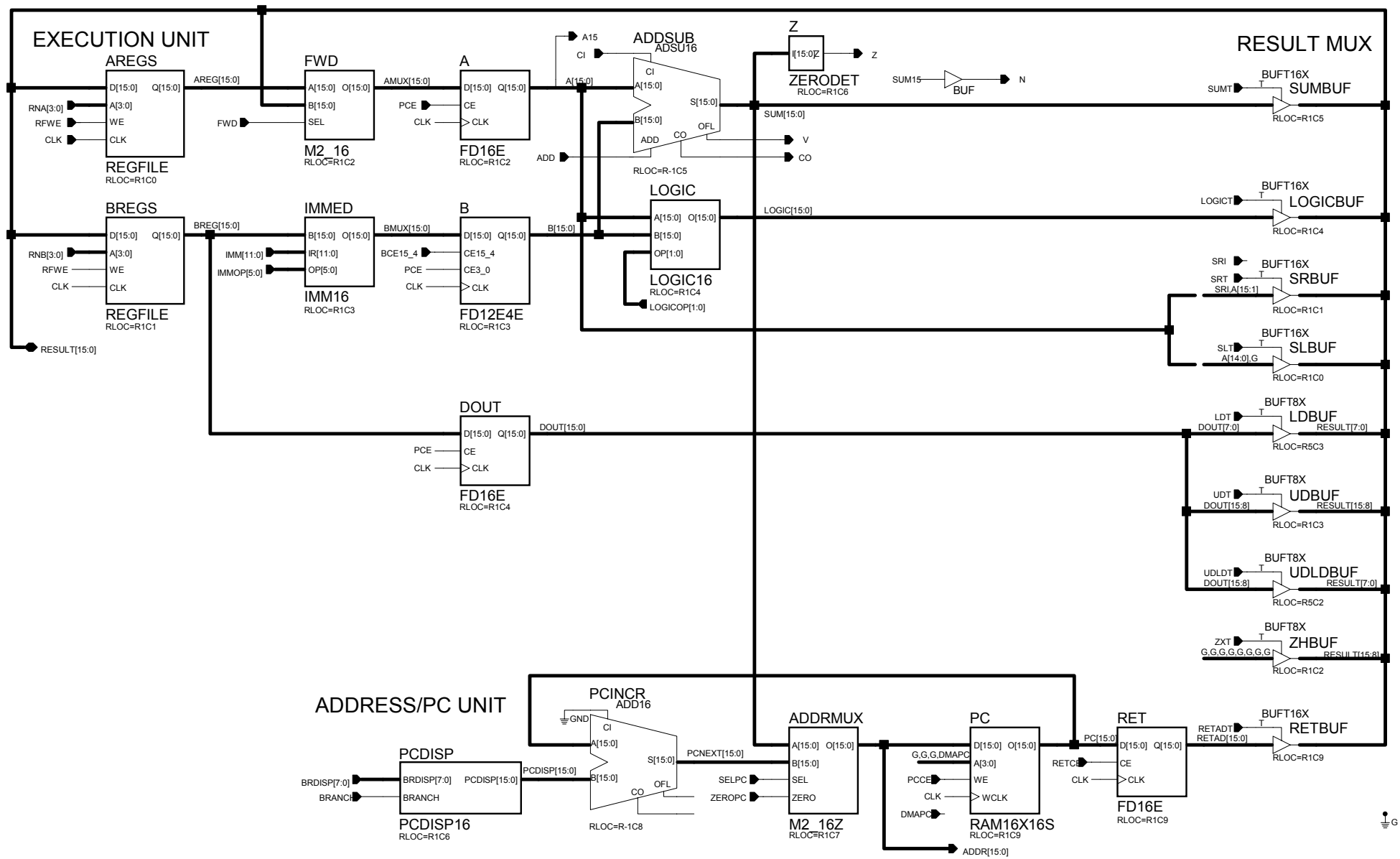
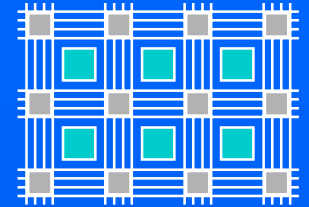


Figure S3: XR16 CPU Datapath Schematic

Copyright (C) 2000, Gray Research	Project: [None]
This work and its use subject to Xilinx	Macro: DP16
License Agreement. See LICENSE	Date: 02/23/100

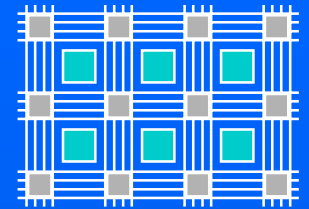
XSOC System-on-a-Chip



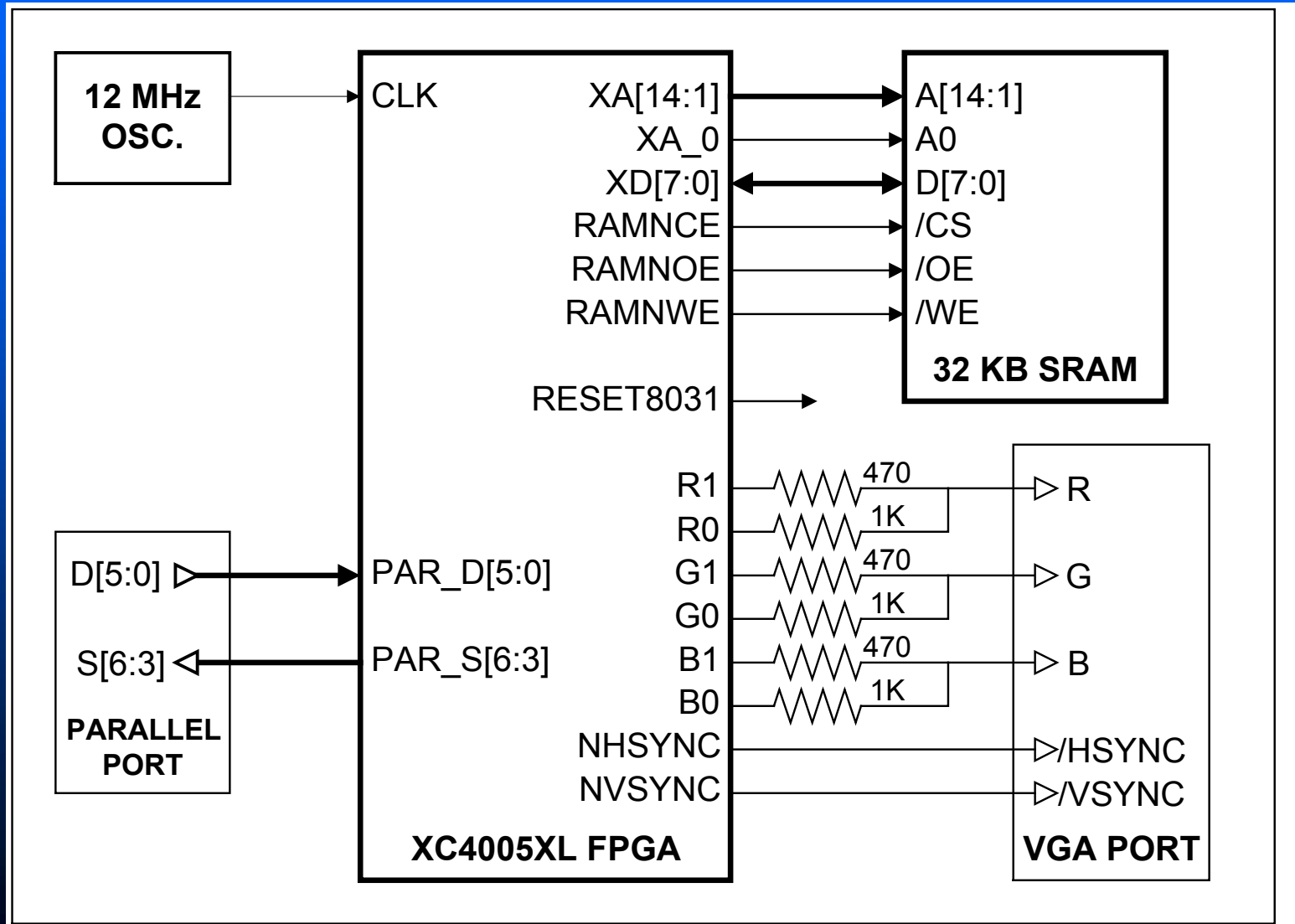
lw r12,4(r3)

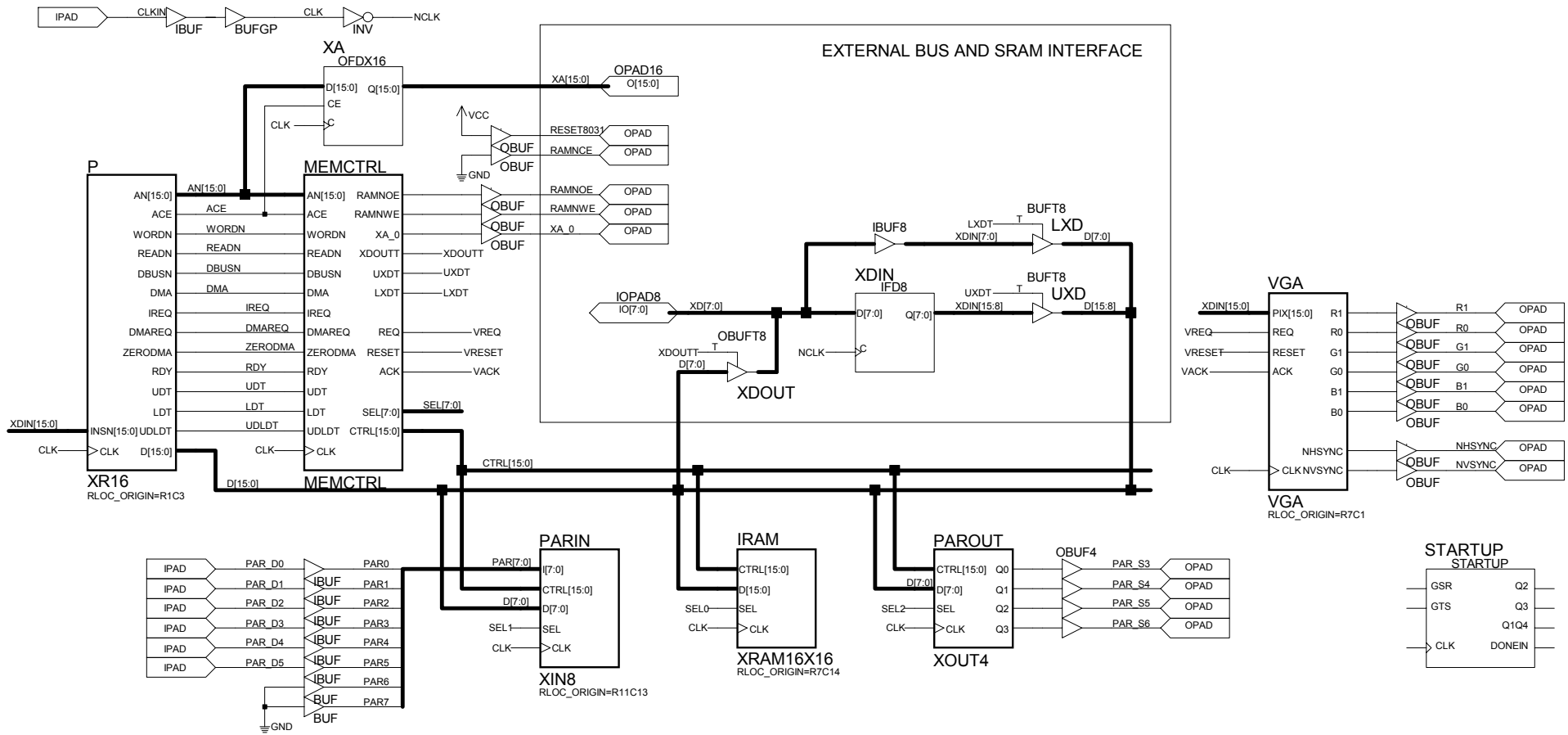
- ◆ *Make integrated system dev more accessible*
- ◆ Integrated system:
 - ◆ Processor, memory, memory controller, on-chip bus, parallel port, ram, video controller
 - ◆ 32 KB SRAM → 576x455 monochrome display

“System-on-a-Chip” in Context

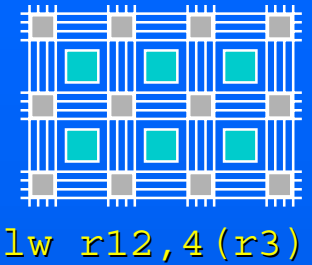


r12,4 (r3)



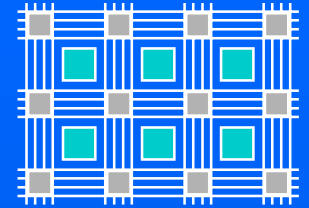


On-Chip Bus



- ◆ 16-bit pipelined on-chip data bus
- ◆ Bus/memory controller
 - ◆ Address decoding, bus controls (output enables, clock enables), external SRAM control
- ◆ Make core reuse easy: *no glue logic required*
 - ◆ Abstract control signal bus
 - ◆ Locally decoded within each core
 - ◆ Just add core, attach data, control, and select lines
 - ◆ Can evolve bus with new features without invalidating existing designs and cores

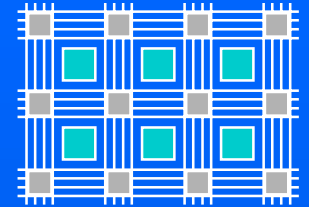
Video Controller



lw r12,4(r3)

- ◆ 32 KB \rightarrow 576x455 monochrome bitmap
- ◆ VGA compatible sync timings
- ◆ Video signal generation
 - ◆ A series of frames, each a series of lines, each a series of pixels
 - ◆ Fetch 16-pixel words, shifts them out serially
 - ◆ DMA read a new word every 8 clocks
 - ◆ Drive horizontal and vertical sync to “frame” pixels
 - ◆ 2 10-bit counters and 4 10-bit comparators

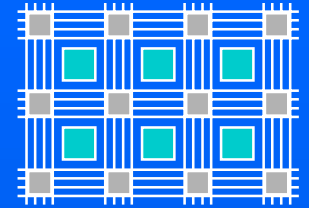
System Bring Up



`lw r12,4(r3)`

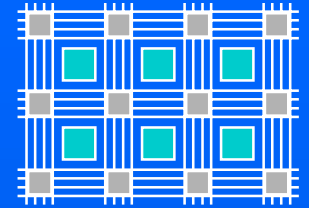
- ◆ 11/98: Proposal, compiler, instruction set, datapath
- ◆ 12/98: Control unit, simulate CPU, target XS40 board, 1 Hz, 20 MHz
- ◆ Debugging with LEDs and stop: goto stop;
- ◆ Later added on-chip bus, external SRAM interface, DMA, video, and interrupts
- ◆ Testing challenge

Hands-on Computer Architecture



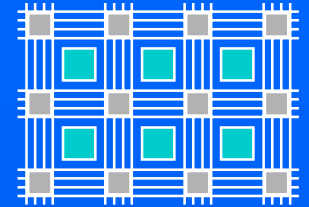
- ◆ Build a working computer in an FPGA 1w r12,4 (r3)
- ◆ Value and appeal of building real hardware
- ◆ Teaching performance tradeoffs
 - ◆ CPI vs. area vs. cycle time vs. power
 - ◆ FPGA EDA tools close the loop, provide “realistic” feedback
 - ◆ Cost of everything
 - ◆ Critical paths and retiming
 - ◆ Interconnect and floorplanning issues
 - ◆ Makes possible *quantitative analysis*

Other Benefits



- ◆ Cores approach helps teach design reuse and embedded system design
- ◆ No hand-waving allowed
- ◆ Learning the testing imperative
- ◆ Living the “system bring up” experience
- ◆ Can simplify instruction set for teaching
 - ◆ Simple enough to understand end-to-end and all the way down
- ◆ Living small
- ◆ FPGA design vocational training

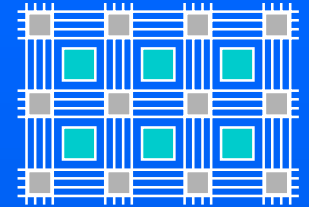
Senior Project Ideas



`lw r12, 4(r3)`

- ◆ Implement a processor core given its spec
- ◆ Improve performance through pipelining
- ◆ Add a cache, MMU, or exceptions
- ◆ Accelerate some inner-loop C code: add a coprocessor or custom instructions
- ◆ Add a new SoC peripheral core, plus interrupt handler/device driver, and testing
- ◆ Develop a test suite to verify the CPU/system
- ◆ Competitive CPU design contest - perf/power

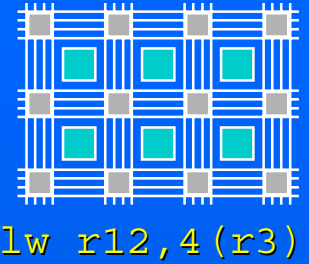
Get Real!



lw r12,4(r3)

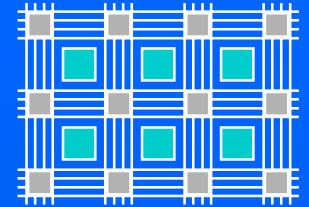
- ◆ FPGA CPUs/SoCs a realistic senior project?
 - ◆ Certainly for computer engineering students with a prerequisite HDL/FPGA digital design course
 - ◆ Maybe (maybe not) for CS students
- ◆ Vary degree of difficulty using kit approach
- ◆ Amortize material and cost across set of courses (intro digital design, compilers, computer architecture, operating systems)
- ◆ Danger: teach arch, not EDA tools quirks

FPGA CPUs for Advanced Architecture Studies



- ◆ Exploit improvements in FPGAs (2^6 in 7 years)
 - ◆ New embedded RAM blocks (Virtex and APEX)
- ◆ xr16: 270 logic cells; xr32: 430-700 logic cells
- ◆ V3200E: 73,000 logic cells!
- ◆ V600E: 15,000 cells + 72 256x16 SRAMs
 - ◆ 8-way chip-multiprocessors *easy*
 - ◆ 16-way MPs with 8 threads/P, 256x32 I\$/P, and shared 1Kx32 L2\$ *feasible*
- ◆ Fast I/O: 200 MHz ZBT SSRAM, 266 MHz DDR SDRAM, up to 300 Mbps/pin chip-chip!

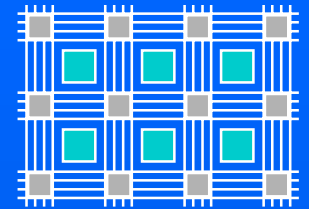
Applications of Block RAMs



```
lw r12, 4(r3)
```

- ◆ Processor
 - ◆ Register files: vector, windowed, multi-context
 - ◆ On-chip RAM, ROM, microcode, stacks
 - ◆ Caches: data, tags, write buffers
 - ◆ Branch history tables, branch target caches
 - ◆ MMUs: segment registers, TLBs (not assoc)
 - ◆ Debug and tracing support
- ◆ System
 - ◆ Interconnects: packet/cell buffers, queues
 - ◆ Graphics: video line buffers, texture caches, display lists, span buffers, color LUTs
 - ◆ GC: write barrier support: page attributes, card marking bits
 - ◆ Multimedia: pixel blocks, coeff and compression tables

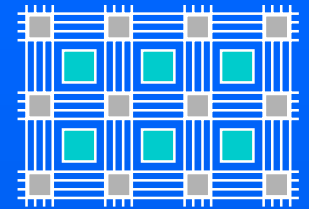
Advanced Project Ideas



lw r12,4(r3)

- ◆ Starting with a scalar kit, design and build a
 - ◆ 2-3 operation LIW
 - ◆ 2-issue superscalar
 - ◆ Chip multiprocessor (+ memory system)
 - ◆ Multithreaded processor
 - ◆ Fault tolerant processor
- ◆ Or add architectural support for
 - ◆ network routing, packet inspection, message passing
 - ◆ non-pausing multithreaded GC
 - ◆ hybrid processor + DSP datapath

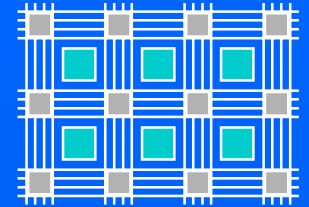
Related Work



lw r12,4(r3)

- ◆ Virginia Tech, *Rapid Prototyping*, 16-bit HOKIE RISC processor in XC4010
- ◆ Cornell EE475 *Microprocessor Architectures*, VHDL design and FPGA verification (simple and pipelined)
- ◆ Hiroshima City University, *City-1*, HDL design of RISC or CISC CPU into XC4010
- ◆ Hamblen, Georgia Tech, CPU design (including MIPS subset), HDL, into Altera 10K20s and 10K70s

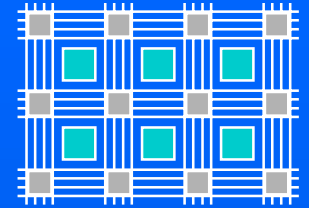
Ongoing Work



lw r12,4(r3)

- ◆ Near term
 - ◆ Package XSOC/xr16 distribution for beta test ✓
 - ◆ Retarget to Verilog ✓
 - ◆ Port to Virtex and Altera architectures
- ◆ Longer term
 - ◆ Help build a community
 - ◆ xr32 $\frac{1}{2}$ ✓
 - ◆ Port GCC/binutils; build eCOS, Linux?
 - ◆ Chip multiprocessors

Conclusions



lw r12,4 (r3)

- ◆ FPGA CPUs are not only feasible, they're a good idea for embedded systems development
- ◆ Building FPGA CPUs and SoCs make computer architecture real, tradeoffs quantitative, and the lessons tangible
- ◆ Project labs based upon FPGA CPU kits can accommodate a range of studies and expertise
- ◆ Students in graduate level comp arch courses can now *build* multiprocessors, etc.
- ◆ *What do you think?*

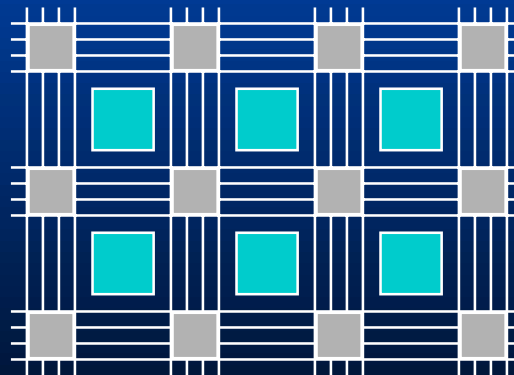
Hands-on Computer Architecture – Teaching Processor and Integrated Systems Design with FPGAs

Jan Gray

Gray Research LLC

jsgray@acm.org

www.fpgacpu.org



`lw r12, 4(r3)`