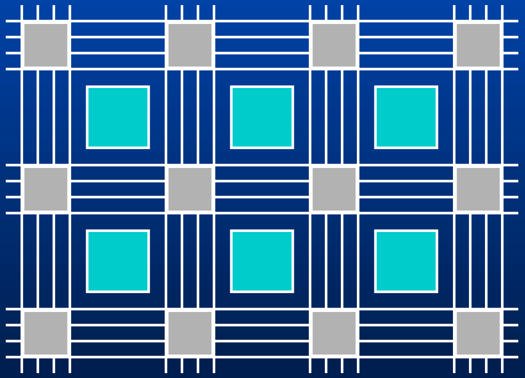
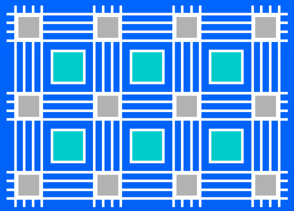
Processor and Integrated System Design in FPGAs

Jan Gray
President, Gray Research LLC
(jan@fpgacpu.org)



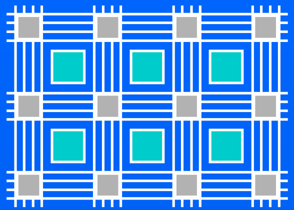
`lw r12,4(r3)`



Introduction

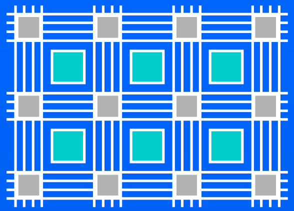
- Recent dense programmable logic enables practical "desktop processor development"
- Examples
  - RISC4005 (Freidin 1990)
  - j32 (1995)
  - XSOC/xr16 (1998)
    - xr16: simple pipelined 16-bit RISC, 33 MHz, 1.4 CPI
    - XSOC: on-chip bus and peripherals
- *Towards a reusable cores community and cost-effective FPGA-based systems-on-a-chip*

Outline



- Introduction to FPGAs
  - Xilinx XC4000XL architecture
  - Development process
- XSOC/xr16 project
  - xr16 tools, architecture, datapath, control unit
  - XSOC design and implementation
- Ongoing work
- References, Resources

Field Programmable Gate Arrays

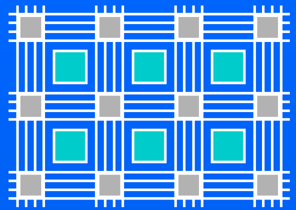


- Evolved from PALs, PLDs, Complex PLDs
- A niche hardware implementation technology

Technology	Gates	Speed	Cost	Part Cost	Spin time
Custom VLSI	<100M	<1 GHz	$50K-up	$1-up	weeks
Gate array	<10M	<400 MHz	$10K-$1M	$1-up	days/weeks
FPGA	*<1M*	*<200 MHz*	*$100-$100K*	*$3-$1K*	*minutes/hours*

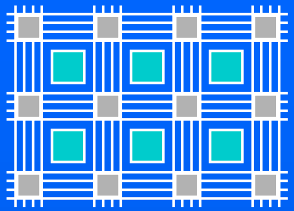
  - Strengths: quick prototyping and time-to-market, reprogrammability, relatively easy to use
  - Weaknesses: cost, density, speed
- Vendors: Xilinx, Altera, Actel, Atmel, Lucent, Cypress, Lattice, QuickLogic

FPGA Technologies



- Programmable Elements
  - Logic
    - combinatorial: lookup tables (LUTs) or gates + muxes
    - sequential: flip-flops
  - Interconnect: metal wire segments connected by...
    - pass transistors driven by SRAM or EEPROM bit cells
    - multiplexers
    - fuses or antifuses (one-time programmable)
- FPGA architecture variations
  - CLB arrays, rows, macrocells; fine/coarse grain
  - RAM, fast wide adders, TBUFs (3-state buffers)

FPGAs: Easy to Use



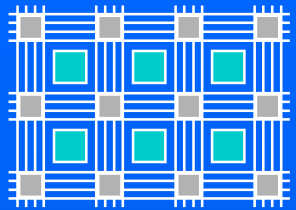
- Idealized digital design model
  - No analog EE: ignore gate and wire capacitance, transistor W/L ratios, etc.
  - Low skew global nets → No clock skew worries
  - Buffered line drivers → No gate fan-out worries
- Synchronous design “just works”
  - Avoid async. flip-flop state changes, gated clocks
  - Try to keep everything on one device
  - Design, simulate, it *should* just work
    - ...even if you’re just a software type
- But, effort to master effective use of resources

FPGA Applications

lw r12,4(r3)

- Conventional: cheap, fast turnaround, field upgradable ASICs
  - Telecom, networking, XSOC/xr16
- (Re-) configurable computing
  - Quickturn: fast design simulation/verification
  - Signal, image processing: radar, radio
  - Graphics: compositing, octtree rendering
  - Military: target dependent correlation/recognition
  - Cryptography: DES search
  - “Hardware” genetic algorithms

Xilinx XC4000 FPGAs



- Introduced in 1992, Xilinx's third generation
- SRAM based
- Device family spans 3,000 to 250,000 "gates", 1 to 3 ns LUT delays, $10 to $1000 Q1
- XSOC uses 3.3V XC4005XLPC84-3
  - 14x14 Configurable Logic Blocks, 1.6 ns LUT delay
  - 60 bonded-out I/O Blocks
  - $20 ($3 Q100K)
- Eclipsed by newer Virtex, Virtex-E families
  - 256x16 SRAM blocks, up to 2M "gates"

XC4000 Configurable Logic Block

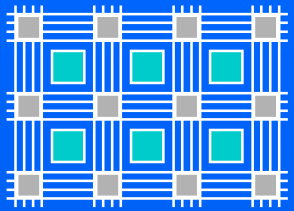
- Two 4-input LUTs
- Two D flip-flops w/ clock enable
- Special features:
  - Writeable LUTs provide 32x1 or 16x2 SRAM
    - 8 CLBs make a 16x16 register file
  - Dedicated fast ripple carry chain hardware
    - 8 CLBs implement 16-bit adder, ~10 ns delay
- *Fig. 1*

XC4000 I/O Block

lw r12,4(r3)

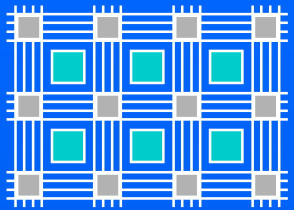
- Input: direct, registered
- Output: direct, registered, 3-stated, pull-ups/pull-downs, slew rate options
- *Fig. 11*

XC4000 Interconnect



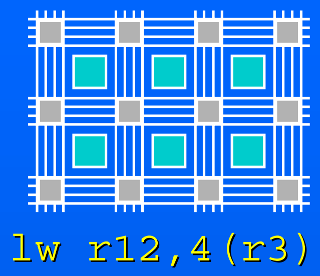
- Programmable Interconnect Points
- Switch matrices
- Input multiplexers
- Hierarchical resources for hierarchical circuits
- Special features
  - Long lines - buses and control lines
    - 3-state buffer per output
  - Low skew global clock lines
- *Fig. 12-16*

FPGA Development Process



`lw r12,4(r3)`

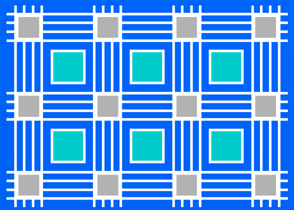
- Design capture: schematics, custom tools, hardware design languages; *floorplan!*
- Simulate/verify
- Compile: "technology mapping", place, route
- Make config bitstream: XC4005XL: 152K bits
- Download:
  - "master load" from byte-wide or bit-serial ROM
  - "slave load" by microprocessor or XChecker pod
- Test/Debug; pod can readback internal state



Talk Outline

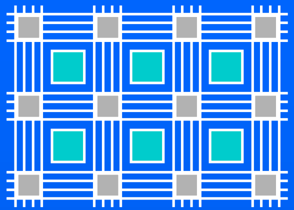
- Introduction to FPGAs
  - Xilinx XC4000XL architecture
  - Development process
- XSOC/xr16
  - xr16 tools, architecture, datapath, control unit
  - XSOC design and implementation
- Demo
- Ongoing work
- References, Resources

XSOC/xr16 Project Goals



- *Make integrated system dev more accessible*
- Subject of *Circuit Cellar* article series
  - Explain FPGA processor and system design
    - Build a fast, simple, thrifty, real CPU
  - Help launch a free cores development community
    - Provide on-chip bus architecture for easy cores reuse
  - Show FPGA CPUs can be practical (=cheap)
    - Not just an ASIC prototyping tool
  - Show C compiler support
  - Show efficient FPGA design “best practices”

Development Tools



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

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:               ; r3=k r4=p
       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
  ```

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+b*cond*
  - fast call/return → call/jal
  - mul/div/shifts in software

XR16 Instruction Set

`lw r12,4(r3)`

Hex	Fmt	Assembler	Semantics	N
0*dab*	rrr	`add rd,ra,rb`	rd = ra + rb;	1
1*dab*	rrr	`sub rd,ra,rb`	rd = ra – rb;	1
2*dai*	rri	`addi rd,ra,imm`	rd = ra + imm;	1
3*d*b*	rr	`{and or xor andn adc sbc} rd,rb`	rd = rd *op* rb;	1
4*d*i*	ri	`{andi ori xori andni adci sbci slli slxi srai srli srxi} rd,imm`	rd = rd *op* imm;	1
5*dai*	rri	`lw rd,imm(ra)`	rd = *(int*)(ra+imm);	2
6*dai*	rri	`lb rd,imm(ra)`	rd = *(byte*)(ra+imm);	2
8*dai*	rri	`sw rd,imm(ra)`	*(int*)(ra+imm) = rd;	2
9*dai*	rri	`sb rd,imm(ra)`	*(byte*)(ra+imm) = rd;	2
A*dai*	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 (*cond*) pc += 2*disp8;	*
C*iii*	i12	`call func`	r15 = pc, pc = imm12<<4;	3
D*iii*	i12	`imm imm12`	imm'next$_{15:4}$ = imm12;	1
7*xxx* **E***xxx* **F***xxx*	–	*reserved*	*reserved*	

Synthesized Instructions

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)` *(load-byte, sign-extending)*	`lb rd,imm(ra)` `xori rd,0x80` `subi rd,0x80`
`j addr`	`jal r0,addr`
`ret`	`jal r0,0(r15)`

Compromised Instructions

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

lw r12,4(r3)

Design Hierarchy: xr16

- XSOC
  - xr16
    - datapath
    - control unit
  - memctrl
  - vga
  - parin, parout, iram

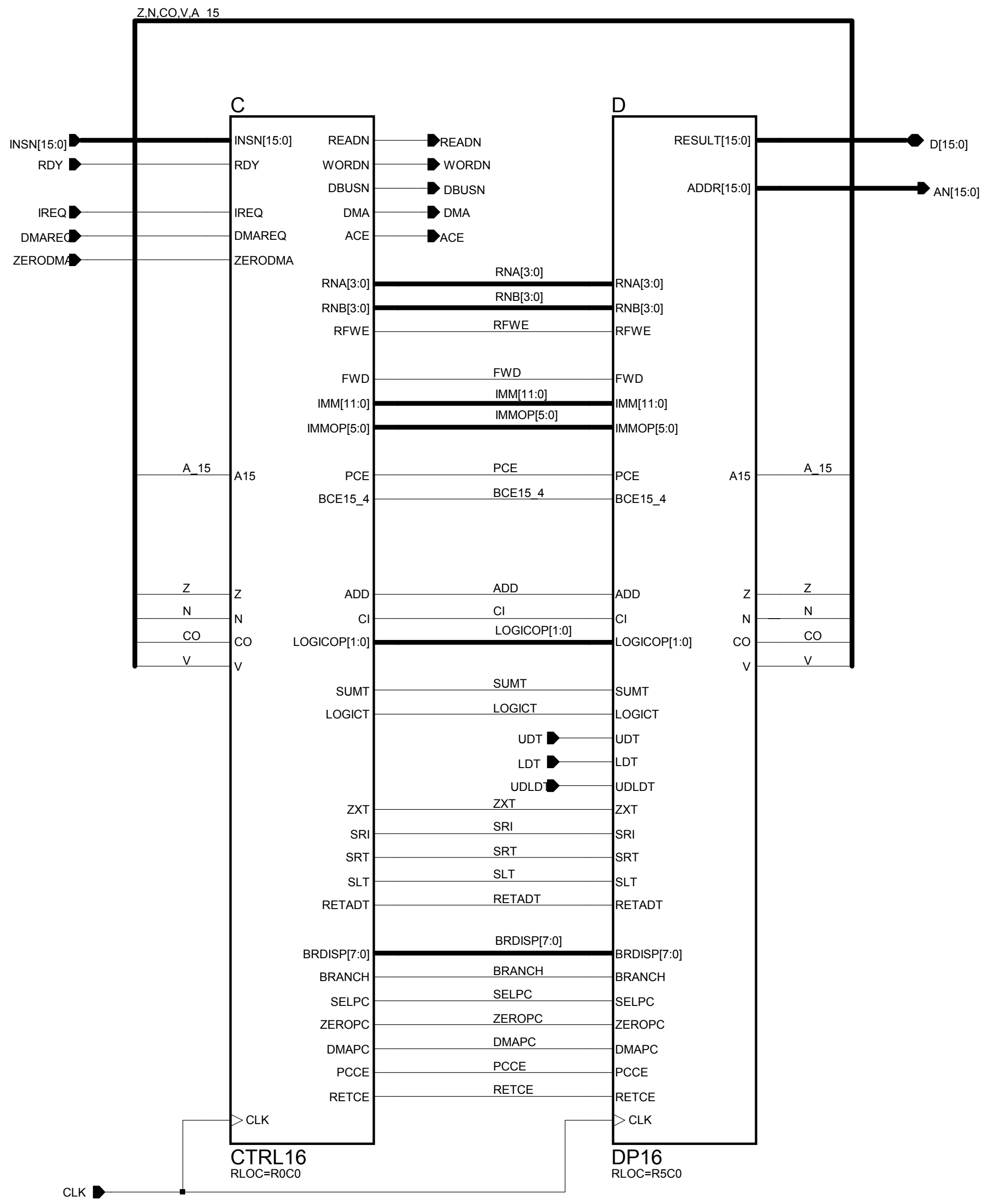


Figure S2: XR16 CPU Schematic

Project: [None]	
Macro: XR16	
Date: 02/23/100	

`lw r12,4(r3)`

Datapath Resources

- 2 read, 1 write per cycle register file
- Immediate operand multiplexer
- A and B operand registers
- Adder, logic unit, shifter
- Result multiplexer
- Zero, negative, carry, overflow detection
- PC, branch displacement multiplexer, PC adder
- Address multiplexer

Design Hierarchy

- XSOC
  - xr16
    - datapath
    - control unit
  - memctrl
  - vga
  - parin, parout, iram

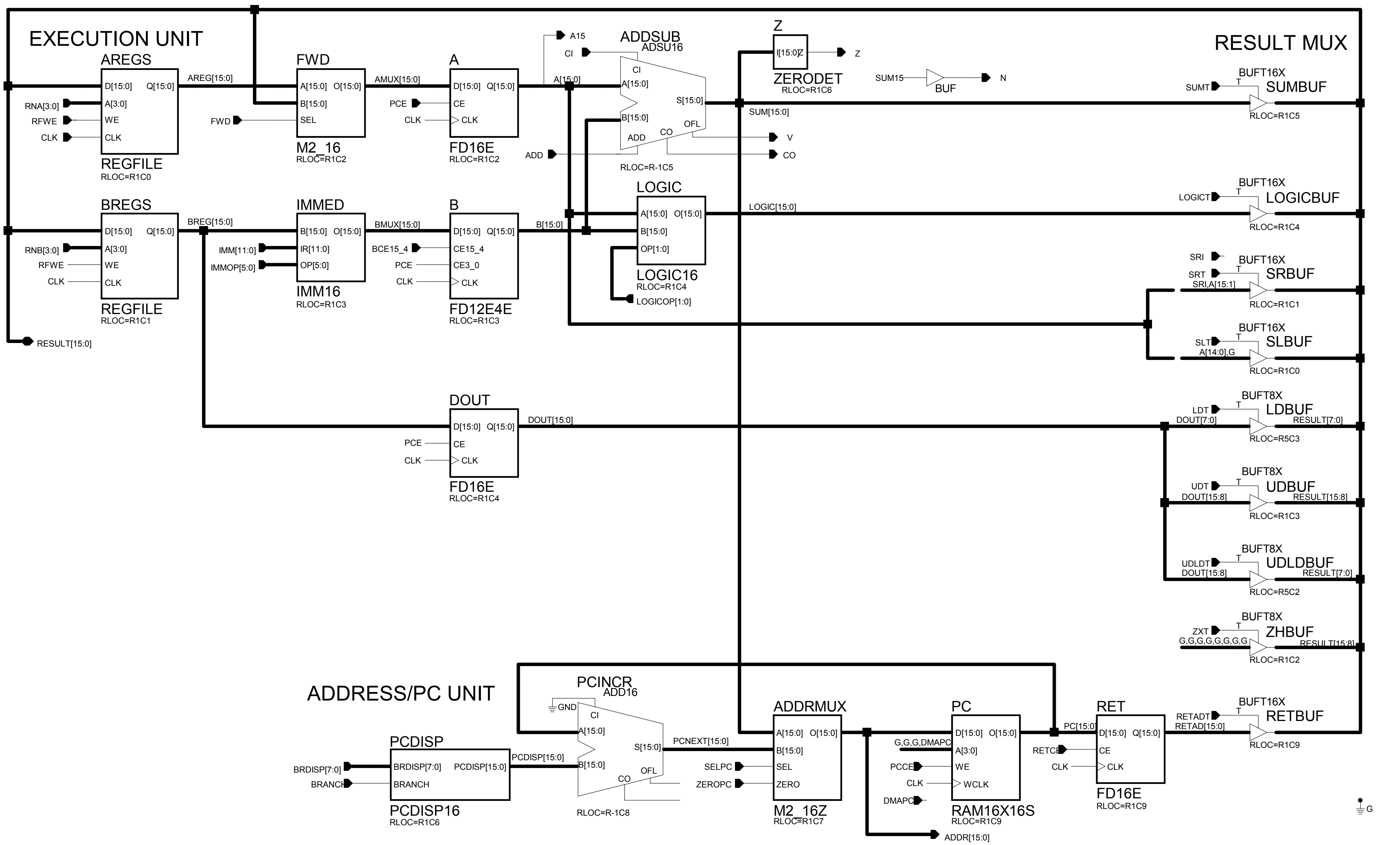
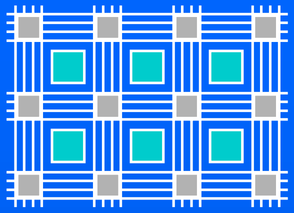
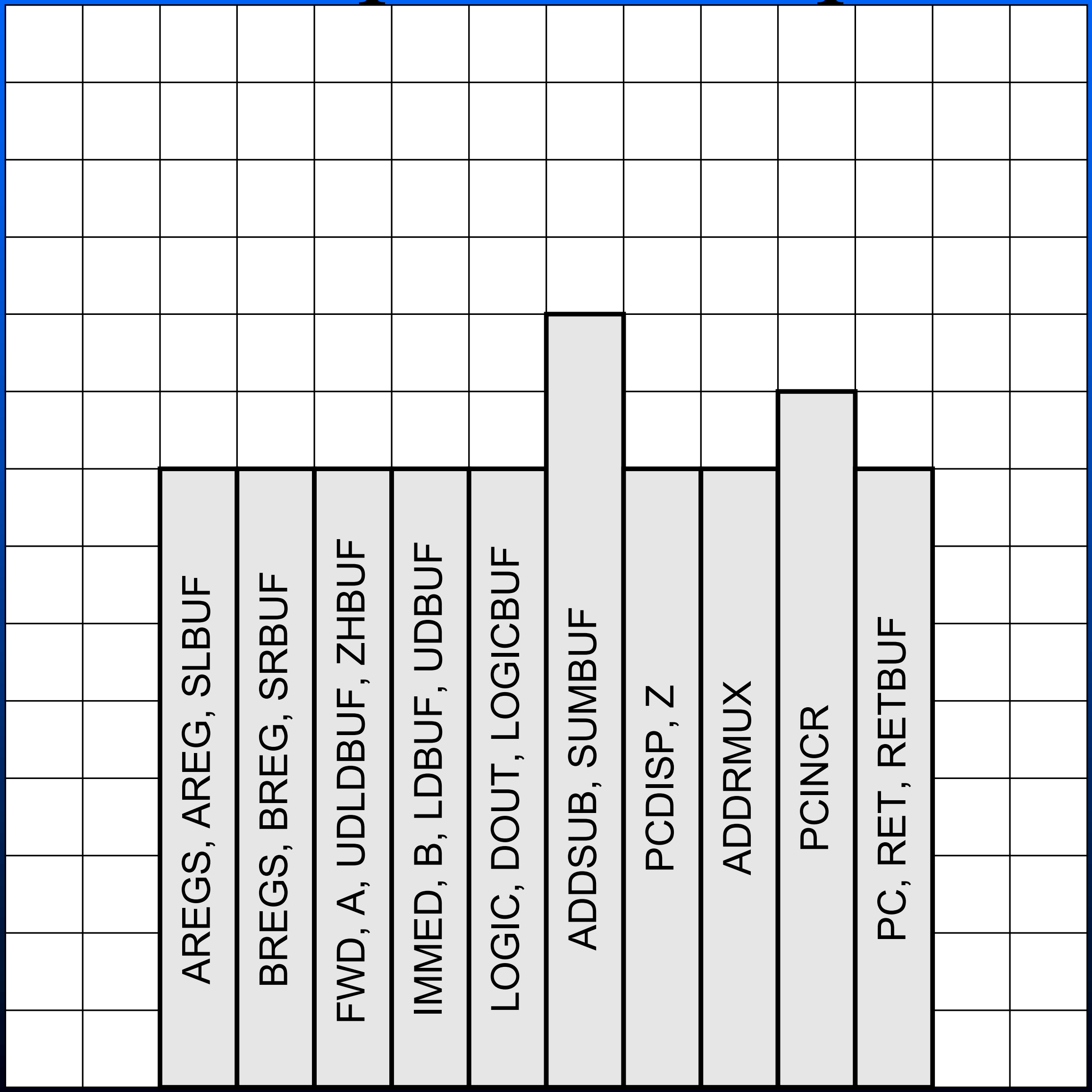


Figure S3: XR16 CPU Datapath Schematic

Project: [None]	
Macro: DP16	
Date: 02/23/100	

Datapath Floorplan


lw r12,4(r3)



Control Unit: Pipeline Stages

lw r12,4(r3)

- IF: Instruction Fetch
  - Read instruction at addr; prepare addr ← 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: PC := sum
  - load/store: addr ← sum; stop pipe for access

Control Unit: Pipeline Hazards

`lw r12,4(r3)`

- Memory wait states
- Result use data dependence
- Load/store memory access
- Jumps and taken branches
- Interrupts

Control Unit Elements

`lw r12,4(r3)`

- Control finite state machine
  - Memory interface controls
- Instruction registers (next, DC stage, EX stage)
- DC: Instruction decoder
- DC: Register file read controls
- DC: Immediate operand and forwarding control
- EX: ALU and result multiplexer control
- EX: Register file write control
- DC+EX: Conditional branch control

Design Hierarchy

- XSOC
  - xr16
    - datapath
    - control unit
  - memctrl
  - vga
  - parin, parout, iram

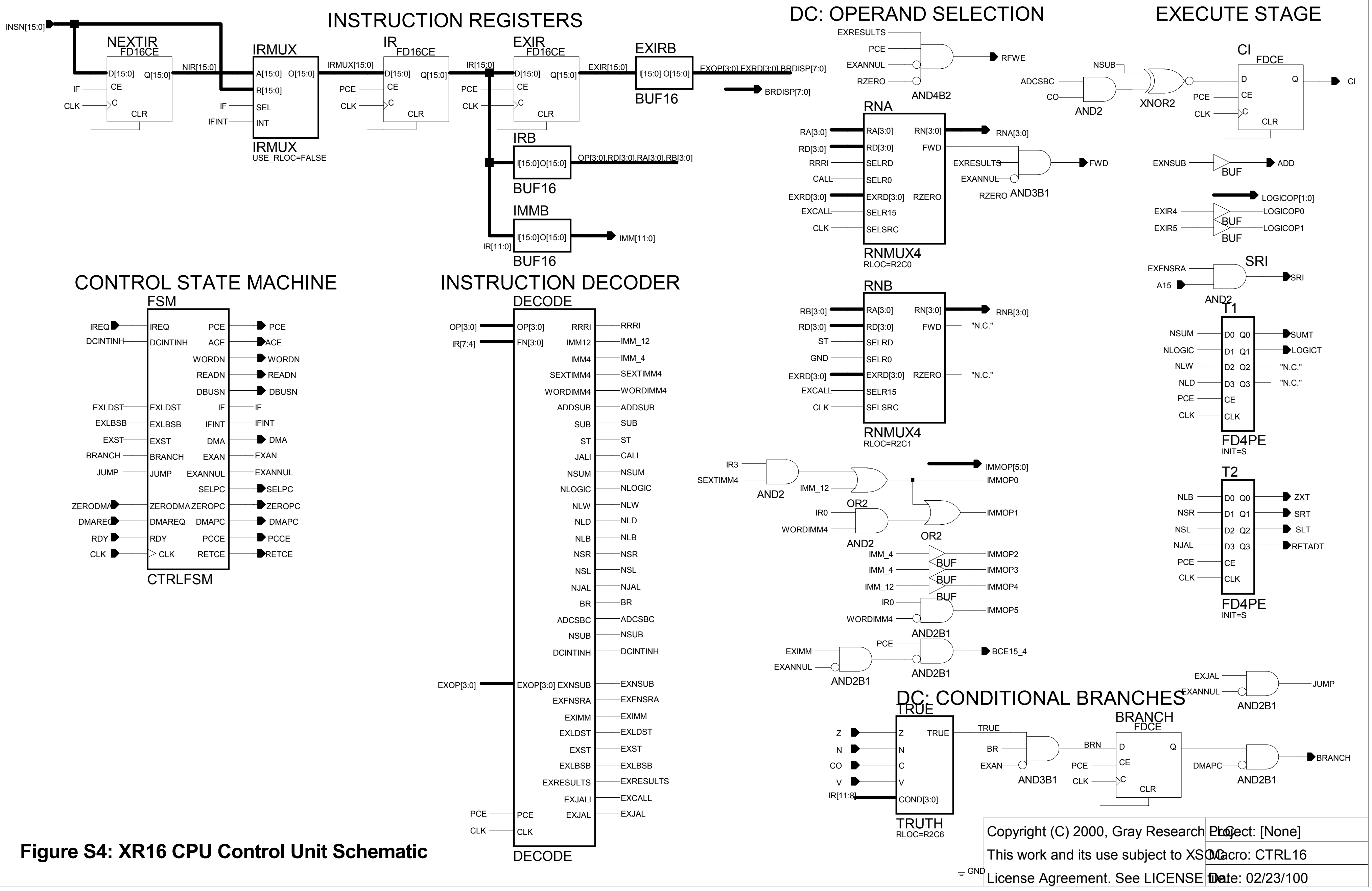


Figure S4: XR16 CPU Control Unit Schematic

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

`lw r12,4(r3)`

Outline

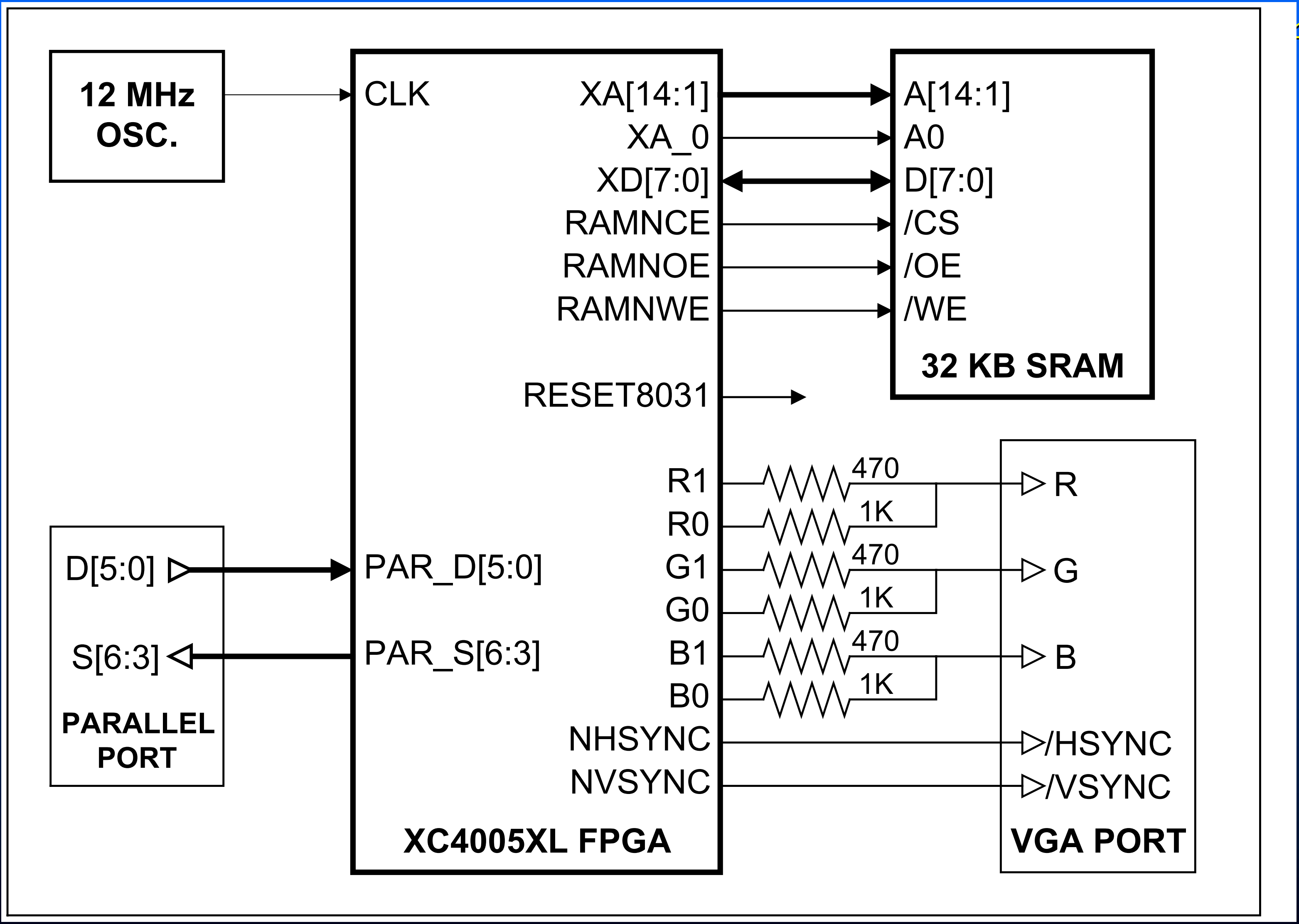
- Introduction to FPGAs
  - Xilinx XC4000XL architecture
  - Development process
- XSOC/xr16 project
  - xr16 tools, architecture, datapath, control unit
  - XSOC design and implementation
- Ongoing work
- References, Resources

XSOC Project

lw r12,4(r3)

- *Make integrated system dev more accessible*
- Build with Xilinx Student Edition ($100)
- Target XESS XS40-005XL student lab board
  - Textbook, exercises, and tools ($100)
- Integrated system:
  - Processor, memory, memory controller, on-chip bus, parallel port, ram, video controller
  - Double-cycle bytewide external SRAM
  - 32 KB SRAM → 576x455 monochrome display

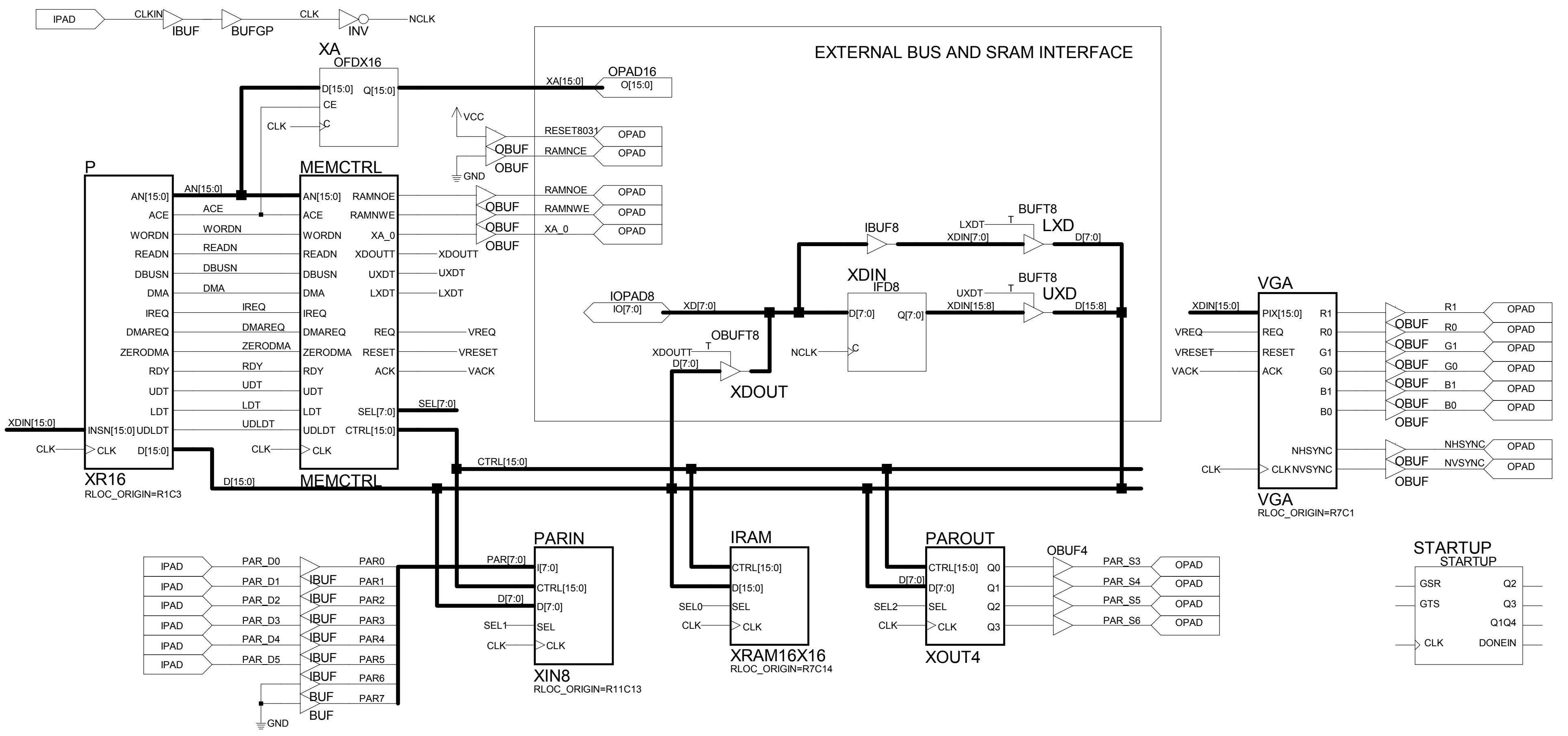
“System-on-a-Chip” in Context



lw r12,4(r3)

Design Hierarchy

- XSOC
  - xr16
    - datapath
    - control unit
  - memctrl
  - vga
  - parin, parout, iram



Project: XSOC
Sheet: XSOC
Date: 06/07/99

On-Chip Bus

`lw r12,4(r3)`

- 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*
  - <u>Abstract control signal bus</u>
  - 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

- 32 KB → 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

lw r12,4(r3)

Recap

- Final floorplan
- FPGA editor
- Demo
  - Source
  - Listing
  - Compile and go

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 ½ ✓
  - Port GCC/binutils; build eCOS, Linux?
  - Chip multiprocessors

`lw r12,4(r3)`

References

- FPGAs
  - Trimberger, S., *Field-Programmable Gate Array Technology*, Kluwer.
- LCC
  - C. Fraser and D. Hanson, *A Retargetable C Compiler: Design and Implementation*, Benjamin/Cummings.
- RISC architecture/implementation
  - D. Patterson and J. Hennessy, *Computer Organization and Design: The Hardware/Software Interface,* also *Computer Achitecture: A Quantitative Approach*, Morgan Kaufmann

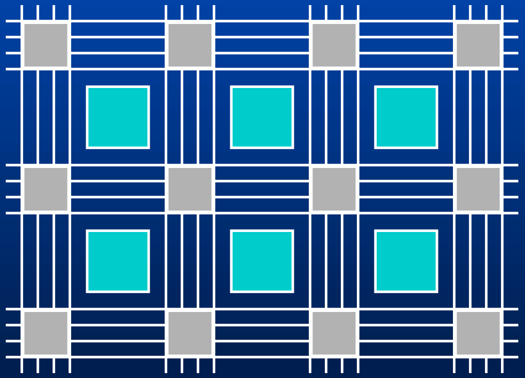
lw r12,4(r3)

Resources

- Usenet: comp.arch.fpga, comp.arch
- Web
  - www.xilinx.com/products/xc4000XLA.htm
  - www.io.com/~guccione/HW_list.html
  - www.fpgacpu.org
- Conferences
  - SIGDA Int'l Symp. on FPGAs, Monterey, Feb.
  - IEEE Symp. on FPGAs for Custom Computing Machines, Napa, April

Processor and Integrated System Design in FPGAs

Jan Gray
President, Gray Research LLC
(jan@fpgacpu.org)



`lw r12,4(r3)`