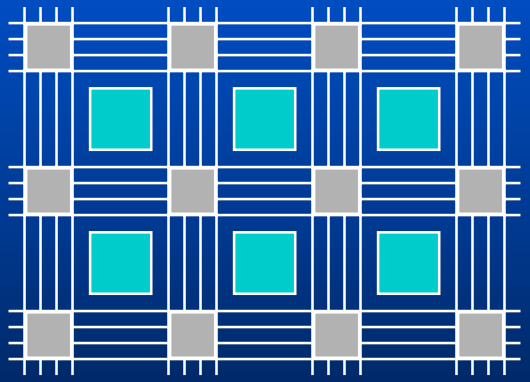
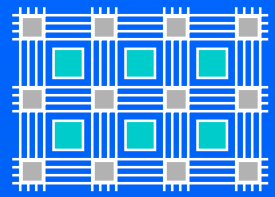
Homebrewing RISCs in FPGAs

Jan Gray (jsgray@acm.org)



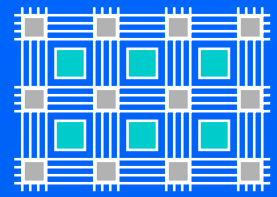
```
lbu r1,4(r31)
```

Introduction



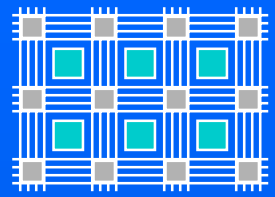
- Recent dense programmable logic enables affordable “desktop processor development”
- Example: J32: 32-bit RISC at 16 MIPS (1995)
  - classic 32-bit RISC
    - 32 registers, 3-operand architecture
  - classic μarch: 4-stage pipeline (IF/RF/EX/WB)
  - on-chip peripherals
    - 32-bit bus, boot ROM, UART, DRAM control, ...

Talk Outline



- FPGAs
- Xilinx XC4000 architecture
- Design Process, CNets HDL
- J32: architecture, μarch, floorplan, peripherals
- Demo
- Ongoing work
- References, Resources

Field Programmable Gate Arrays

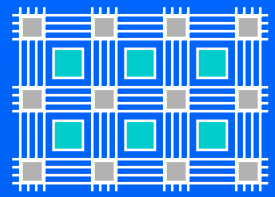


- Evolved from PALs, PLDs, Complex PLDs
- A niche hardware implementation technology

Technology	Gates	Speed	NRE Cost	Part Cost	Spin time
Custom VLSI	<10M	<500 MHz	$20K-$100M	$1-up	weeks
Gate array	<2M	<200 MHz	$10K-$1M	$1-up	days/weeks
FPGA	*<100K*	*<100 MHz*	*$100-$100K*	*$10-$1K*	*minutes/hours*

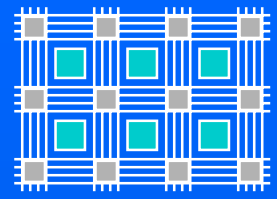
  - Strengths: quick prototyping and time-to-market, reprogrammability, relatively easy to use
  - Weaknesses: cost, density, speed
- Vendors: Xilinx, Altera, Actel, Atmel, Lucent, Cypress, QuickLogic, IBM, Motorola

FPGA Technologies



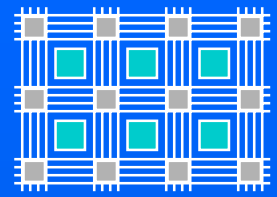
- Programmable Elements
  - Logic blocks
    - combinatorial: lookup tables (LUTs) or gates + muxes
    - sequential: flip-flops
  - Interconnect: metal wire segments connected by...
    - pass transistors driven by SRAM or EEPROM bit cells
    - fuses or antifuses (one-time programmable)
- FPGA architecture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

FPGAs: Easy to Use



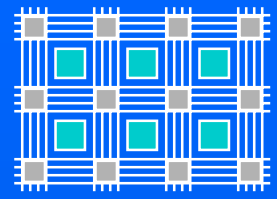
- Idealized digital design model
  - No analog EE: ignore gate and wire capacitances, transistor W/L ratios, etc.
  - Low skew global nets ⇒ No clock skew worries
  - Buffered line drivers ⇒ No gate fan-out worries
- Synchronous design “just works”
  - Avoid async. flip-flop state changes, gated clocks
  - 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



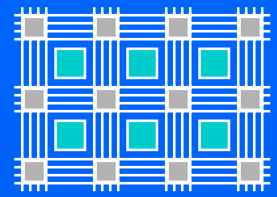
- Conventional: cheap, fast turnaround, field upgradable ASICs (J32)
- Some reconfigurable applications
  - Quickturn: fast design simulation/verification
  - Signal, image processing: filters, warping, music
  - Graphics: span and octtree rendering acceleration
  - Military: target dependent correlation/recognition
  - Cryptography: DES search
  - “Hardware” genetic algorithms

Xilinx XC4000 FPGA



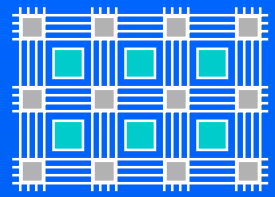
- Introduced in 1992, Xilinx's third generation
- Device family spans 3,000 to 125,000 "gates", 2 ns to 6 ns gate delays, $20 to $1000 Q1
- J32 uses XC4010PC84-5
  - 20x20 Configurable Logic Blocks, 60 I/Os, 5 ns gate delay, $150

XC4000 Configurable Logic Block



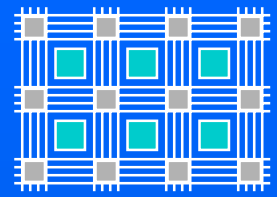
- Two 4-input LUTs and one 3-input LUT
  - Two f(a,b,c,d), or f(a,b,c,d,e, ...), 5 or 7 ns delay
- Two flip-flops w/ clock enable, reset/set
  - 0 ns setup if input is its LUT
- Special features:
  - Writeable LUTs provide 32x1 or 16x2 SRAM
    - 32 CLBs make a 32x32 register file
  - Dedicated fast ripple carry chain hardware
    - 16 CLBs implement 32-bit adder, ~35 ns delay
- *Fig. 1*

XC4000 Interconnect



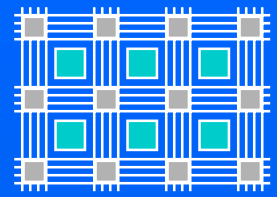
- Hierarchical resources for hierarchical circuits
  - More/better connections to neighbouring CLBs
- Wires per CLB:
  - 8 horizontal + 8 vertical local interconnect
  - 4H+4V double-length: connect every other CLB
  - 6H+6V long lines (2H w/ TBUFs): connect CLBs on a row or column: *buses and control lines*
  - 4V global: *low-skew clock drivers*
  - 2V carry-chain: *fast wide adders/comparators*
- *Fig. 29, 26, 27*

XC4000 I/O Block



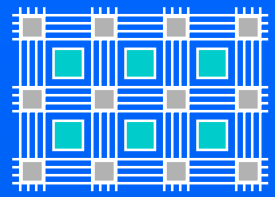
- Input: direct, registered
- Output: direct, registered, 3-stated, pull-ups/pull-downs, slew rate options
- Some IOB pads not bonded to package pins
- *Fig. 16*

FPGA Development Process



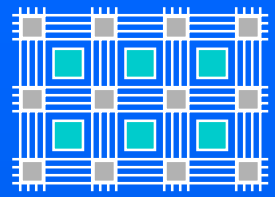
- Design capture: schematics, custom tools, hardware design languages; *must floorplan!*
- Simulate/Verify: *oh well*
- Compile: “technology mapping”, place, route
- Make config bitstream: XC4010: 178,000 bits
- Download:
  - “master load” from byte-wide or bit-serial ROM
  - “slave load” by microprocessor or XChecker pod
- Test/Debug: pod can readback internal state!

CNets HDL



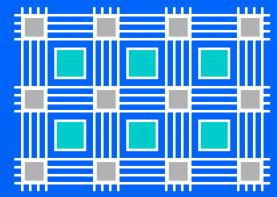
- C++ class library for Xilinx FPGAs
  - Nets, buses, gates, primitives, modules
  - Intuitive notation courtesy overloaded operators
  - C++-based design is compiled, run, emits XNF
- Motivation
  - Inadequacies and costs of schematic capture, Verilog/VHDL synthesis and simulation
  - Power of general purpose programming language
  - Explicit control
    - Placement constraints to achieve floor plan
    - Map particular gate expressions into CLBs

J32 Instruction Set



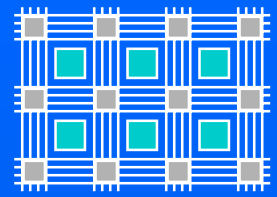
- Instructions
  - add/sub, and/or/xor/xnor, shift/rotate left/right 1,2,4
  - load/store byte/half/word signed/unsigned
  - jump conditional, jump-and-link (two delay slots)
- Condition codes: Zero, Neg, Carry, oVerflow
  - ZNCV on add/sub, ZN.. on op/ld, no change on st/j
- Formats
  - op reg: { 0|op:4|0 rd:5 ra:5 rb:5 }: add r3,r1,r2
  - op imm: { 0|op:4|1 rd:5 ra:5 im:16 }: add r3,r1,42
  - ld/st/j: { 1|op:5 rd:5 ra:5 im:16 }: lbu r1,42(r31)

J32 Design Process



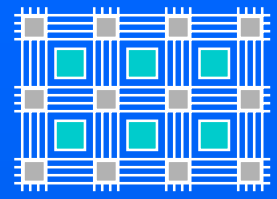
- Evolve “unrealized” 16-bit microcode design
- DLX-like ISA to run C code; DLX-like µarch
- Architecture/microarchitecture codesign
  - Try mapping structures into FPGA elements
  - Iterative floorplanning
  - Example: ld/st byte/halfword requires byte-lane alignment hardware; feasible using TBUFs!

Missing/Compromised Features



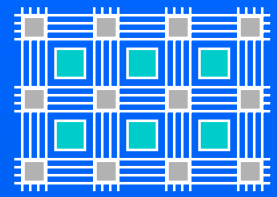
- No barrel shifter: 1-, 2-bit shifts only
- Jumps
  - Reuse effective address adder
    - Saves adder, multiplexor, much interconnect
  - No PC-relative “branch” instructions
  - 2 cycle branch delay
  - Two branch delay slots
    - Better to annul some, give 0, 1, 2 delay slot versions?
- 1+ cycle load/store pipe *stall*
  - Share one memory port

J32 Design Process (2)



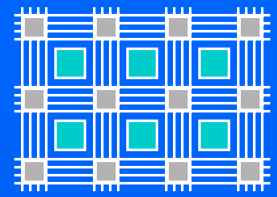
- Challenge: minimize interconnect needs
  - Multiplex any result onto shared 32-bit bus
    - Adder, logic, shifts, load, sign-extensions, store, PC
- Critical paths
  - Register file: result writeback and operand read
  - Execute: operand register, adder, result mux, bypass mux, operand register
  - Memory (lbs): address out, address decode, memory access, result byte alignment, sign ext., result mux, bypass mux, operand register

J32 Pipeline Stages



- IF: Instruction Fetch
  - MAR out; incr. PC; access mem; latch instruction
- RF: Decode and Register File access
  - Decode instruction; read register operands; sign ext. immediate field; select operands
- EX: Execute
  - Add/effective address; logic; shift; select result
  - j: load MAR, PC with ea
  - ld/st: load MAR with ea, stop pipe for mem access
- WB: Write Back results to register file

J32 Pipeline Hazards



- Problem: result dependencies

```
add r1,r1,1   IF RF EX WB
ld  r2,0(r1)     IF RF EX WB
```

- Fix: Software: reorder instructions/insert nops
- Fix: Stall pipeline (not used)

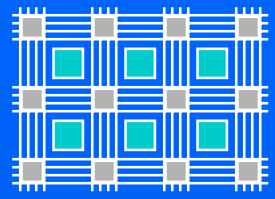
```
add r1,r1,1   IF RF EX WB
ld  r2,0(r1)     IF -- RF EX WB
```

- Fix: register forwarding (register file bypass)

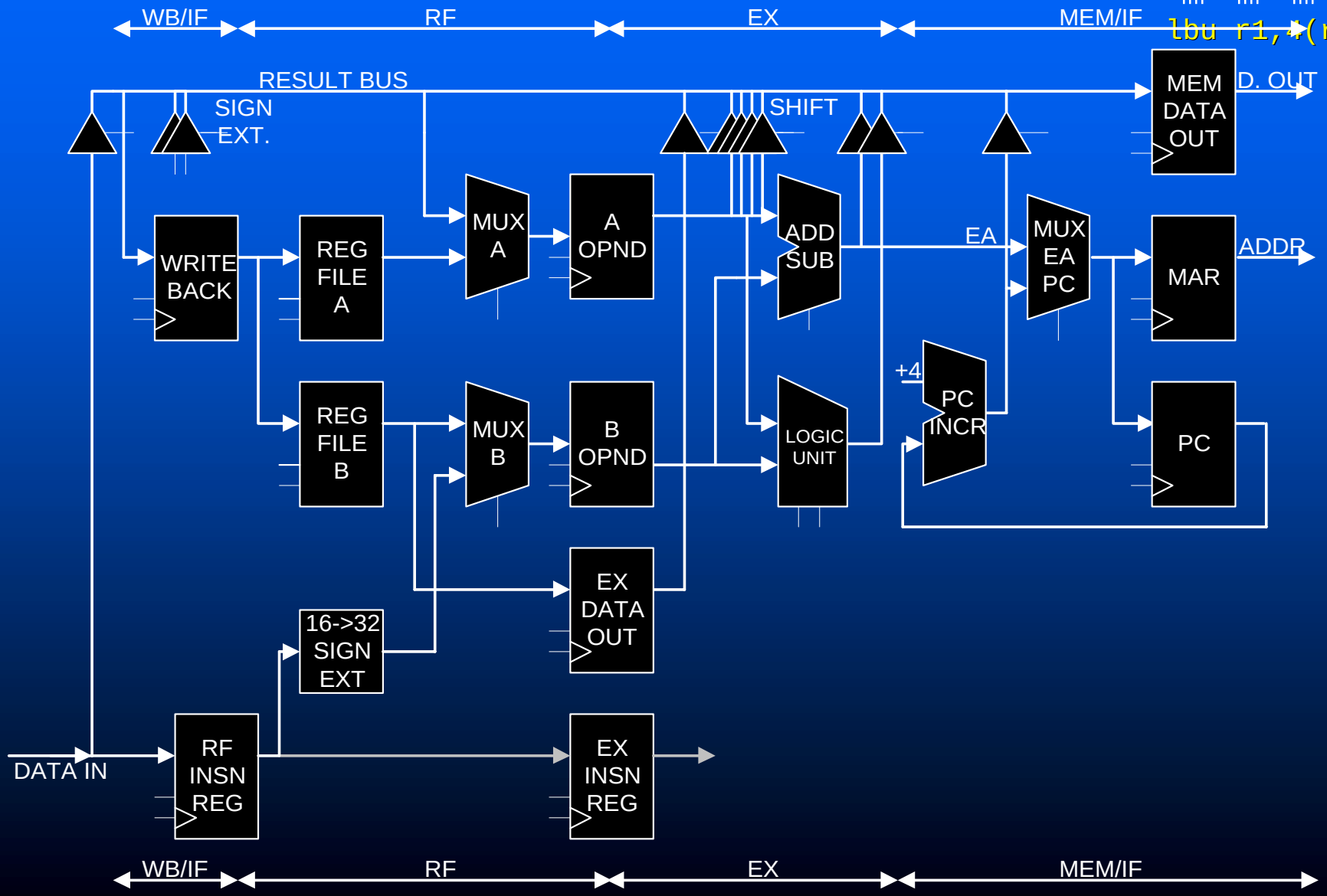
```
add r1,r1,1   IF RF EX WB
ld  r2,0(r1)     IF RF EX WB
```

  - Compromise: A operand only

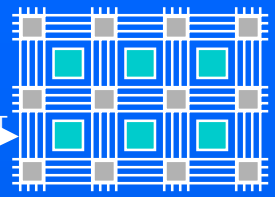
J32 Microarchitecture



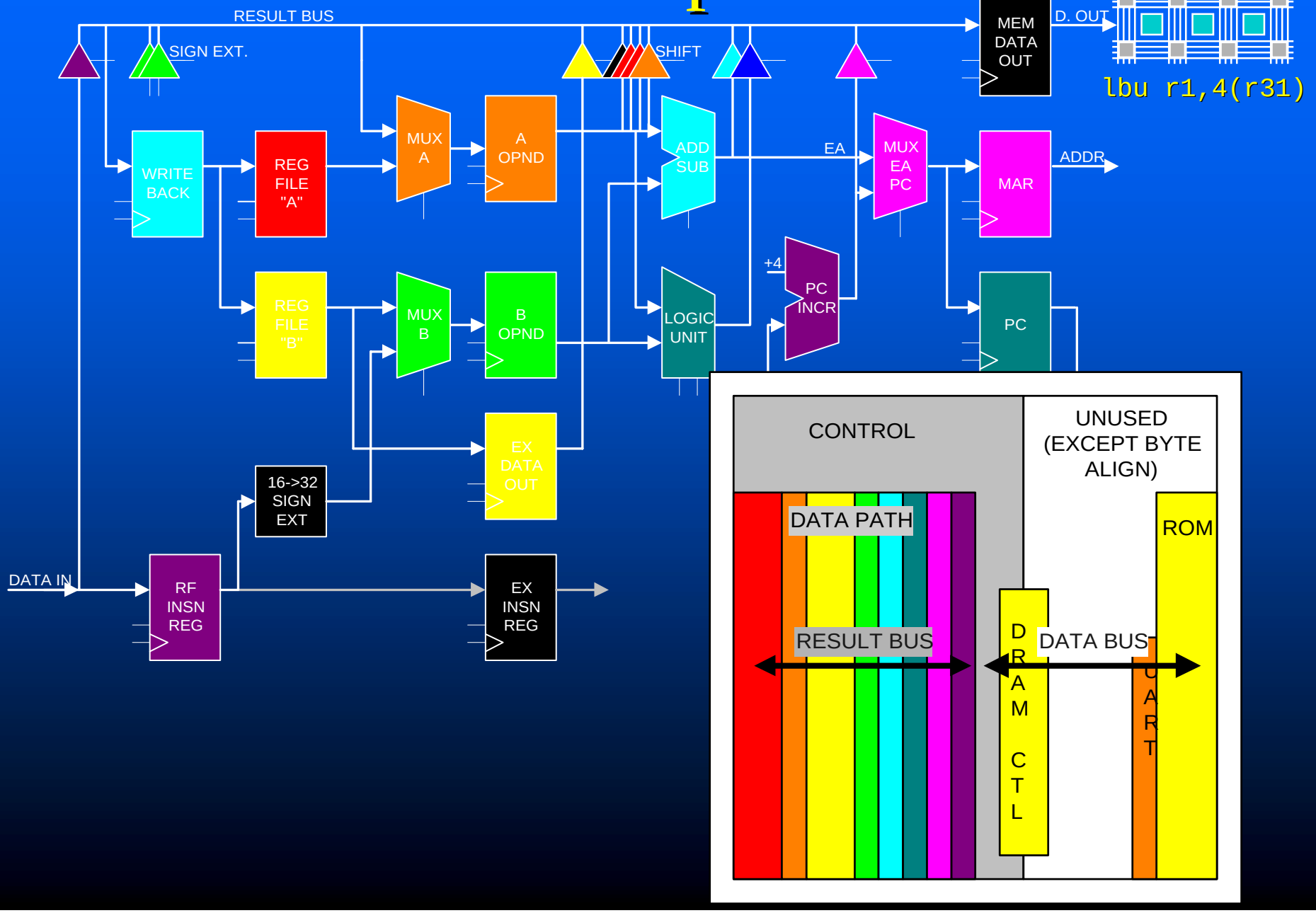
```
lbu r1,4(r31)
```



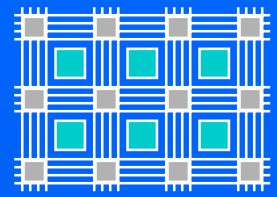
J32 Floorplan



lbu r1,4(r31)

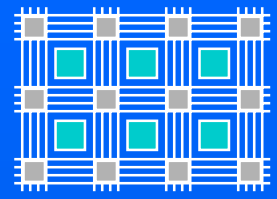


J32 Control Unit



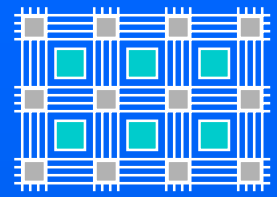
- Hardwired control
- Sequence clock phases

J32 Integrated Peripherals



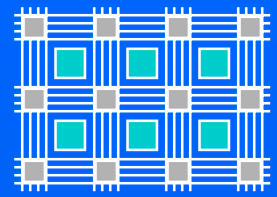
- 32-bit on-chip memory/peripherals bus
  - Address decoding, ack, wait-state insertion
  - Subwords: byte enables, byte-lane shifters
- 32-word boot “ROM”
- UA(R)T
  - Baud rate generator, shift register, busy arbiter
- (DRAM controller)
  - State machine, address mux, RAS/ CAS/, refresh
  - Fast page mode: page address register, comparator

Future Plans



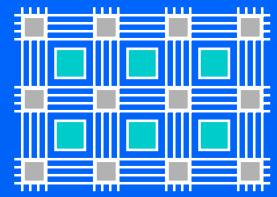
- Redesign CNets as Java HDL, add simulator
- Retarget XC4000E
- Interpreter, operating system
- “J64”
- 3D rendering accelerator

References



- FPGAs
  - Xilinx, The Programmable Logic Data Book, 1994
  - Trimberger, S., Field-Programmable Gate Array Technology, Kluwer.
- RISC architecture/implementation
  - Hennessey and Patterson, Computer Achitecture: A Quantitative Approach, also ... Morgan Kaufman
  - Fielding, Computer Architecture, A Designer's Text Based on a Generic RISC, ...

Resources



- Usenet: comp.arch.fpga, comp.arch
- Web
  - www.xilinx.com, www.altera.com, ...
  - www.io.com/~guccione/HW_list.html
  - www.mrc.uidaho.edu/fpga/fpga.html, www.super.org:8000/FPGA
- Conferences
  - SIGDA Int'l Symp. on FPGAs, Monterey, Feb.
  - IEEE Symp. on FPGAs for Custom Computing Machines, Napa, April