CS 152
Computer Architecture and Engineering

Lecture 18 – Real Processor Walkthru I

2004-11-02

Dave Patterson

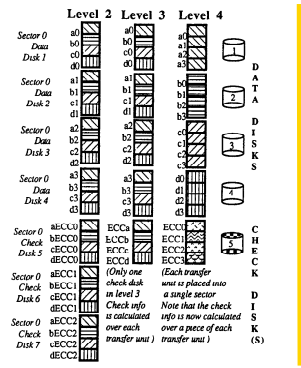
(www.cs.berkeley.edu/~patterson)

John Lazzaro

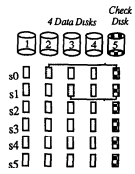
(www.cs.berkeley.edu/~lazzaro)

www-inst.eecs.berkeley.edu/~cs152/

Last Time: A Case for RAID

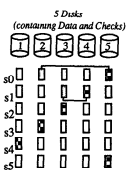


RAID 4



(a) Check information for Level 4 RAID for G=4 and C=1 The sectors are shown below the disks (The checked areas indicate the check information) Writes to s0 of disk 2 and s1 of disk 3 imply writes to s0 and s1 of disk 5 The check disk (5) becomes the write bottleneck.

RAID 5



(b) Check information for Level 5 RAID for G=4 and C=1 The sectors are shown below the disks, with the check information and data spread evenly through all the disks Writes to s0 of disk 2 and s1 of disk 3 still imply 2 writes, but they can be split across 2 disks to s0 of disk 5 and to s1 of disk 4

Original paper now on class website ...

This Week: A Real Processor Walkthru

- Leon: An open-source SPARC CPU
- Configurable HDL designs
- Leon's HDL methodology

Originally, a fault-tolerant CPU for the ESA (Europe's NASA). Now open-sourced, may be freely used in commercial products.

Leon

A VHDL SPARC platform

SPARC Compliant SCD V8

Alternative to Verilog, popular in Europe

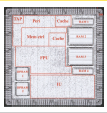
Sun RISC ISA. An IEEE standard: an "open" spec.

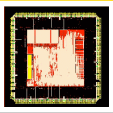
More than a CPU: bus, peripherals, SDRAM, etc.

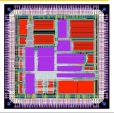
Project website: Gaisler Research, www.gaisler.com

Leon VHDL maps to ASICs, FPGAs

Fabricated ASICs:

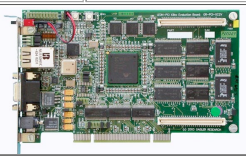






Technology	Area	Timing
Atmel 0.18 CMOS std-cell	35K gates + RAM	165 MHz (pre-layout)
Atmel 0.25 CMOS std-cell	33K gates + RAM	140 MHz (pre-layout, log file)
UMC 0.25 CMOS std-cell	35K gates + RAM	130 MHz (pre-layout)
Atmel 0.35 CMOS std-cell	2 mm2 + RAM	65 MHz (pre-layout, log file)
Xilinx XC2V3000-6	5,000 LUT + block RAM	80 MHz (post-layout, log1, log2)
Altera 20K200C-7	5,700 LCELLs + EAB RAM	49 MHz (post-layout) log file
Actel AX1000-3	7,600 cells + RAM	48 MHz (post-layout) log file

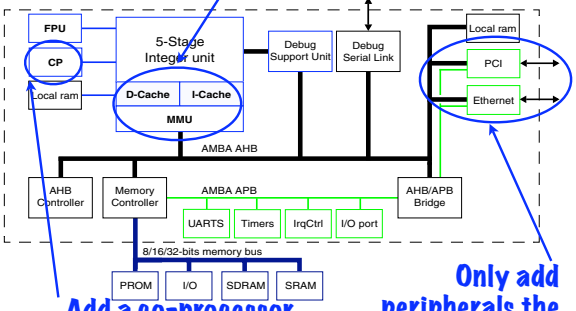
Maps to Xilinx, Altera, Actel FPGAs:



Why use a VHDL core? Configurability!

For "system on a chip." Example: MP3 player.

Customize caches and multiply/divide for the app



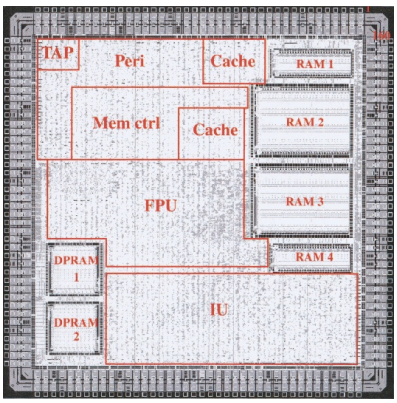
Add a co-processor specialized for app.

Only add peripherals the app needs.

Area impact of configurability

0.35µ, 65 MHz
40 mm²

Removal of FPU would reduce area. (power ? cycle time ?).



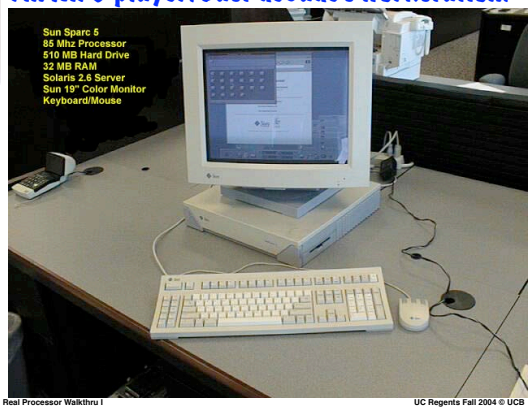
Reality check: What Sun sold in 1994

micro SPARC II

85-170 MHz

0.5 micron process

An MP3 player: Last decade's workstation.



Administrivia: It's Election Day!

- Lab 4 demo on Friday in section. Final report due Monday.
- HW 4: RAID questions online, a Leon question may be added soon. Due Weds 11/10, 5PM, 283 Soda
- Mid-Term II: Tuesday, 11/23, 5:30 to 8:30 PM. Location preference?
- Xilinx field trip date: 11/30. Details on bus transport from Soda Hall soon.

Real stuff: Leon used in GPS receivers

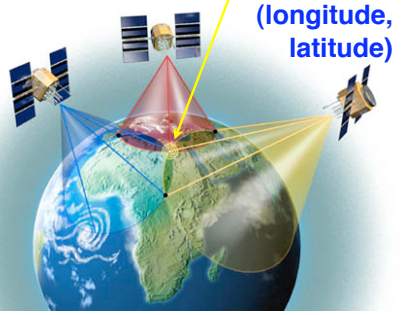
GPS: Global Positioning Satellite

DoD satellite array orbits the earth.

Handheld GPS receiver triangulates satellite radio signals.

CPU intensive, low power.

You are here. Where is here? (longitude, latitude)

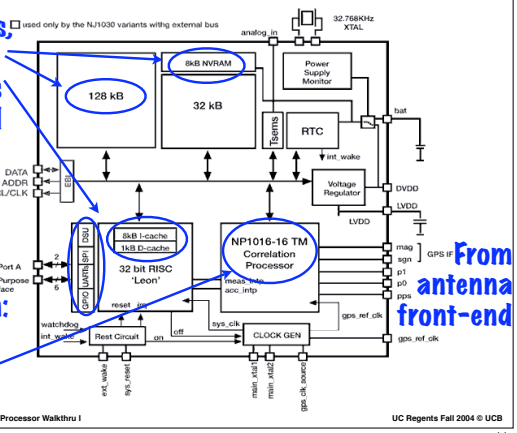


NEMERIX GPS receiver chip uses Leon

CPU options, caches, peripherals customized for app

Extra processor for GPS correlation: Nemerix value-add

From antenna front-end



Leon Customization

(demo)

Leon HDL Methodology

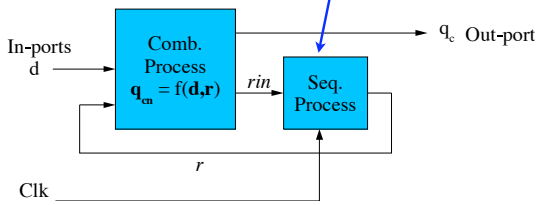
Leon VHDL Methodology: Requirements

- Should be easy to add new features without breaking the CPU.
- Open-source: should be easy to integrate features into source tree.
- Work with many CAD tools, synthesize to many targets (FPGA and ASIC).
- Slides that follow adapted from talk by Jiri Gaisler (Leon chief designer)

#1: "Entity" contains two "processes"

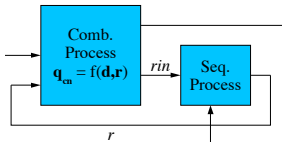
Combinational logic process. "Sensitive" to all inputs.

Flip-flop process. Only sensitive to clock edge.



VHDL "entity" like a Verilog "module"
VHDL "process" like a Verilog "always" statement.
VHDL "sensitive" like Verilog "@" for always.

#2: Naming and typing conventions



The variables r and r_{in} appear in every Leon entity (VHDL for "module"), and hold all register values.

q_{cn}, r, and r_{in} are records (like C structures).

VHDL:

```
type reg_type is record
   irq    : std_logic;
   pend   : std_logic_vector(0 to 7);
   mask   : std_logic_vector(0 to 7);
end record;

signal r, rin : reg_type;
```

Entity input and output also use records.

Example: A "two process" program

```
use work.interface.all;

entity irqctrl is port (
  clk   : in std_logic;
  rst   : in std_logic;
  sysif : in sysif_type;
  irqo  : out irqctrl_type);
end;

architecture rtl of irqctrl is

  type reg_type is record
    irq   : std_logic;
    pend  : std_logic_vector(0 to 7);
    mask  : std_logic_vector(0 to 7);
  end record;

  signal r, rin : reg_type;

begin
```

State record variables

Input, state sensitivity

```
comb : process (sysif, r)
  variable v : reg_type;
begin
  v := r; v.irq := '0';
  for i in r.pend'range loop
    v.pend := r.pend(i) or
      (sysif.irq(i) and r.mask(i));
    v.irq := v.irq or r.pend(i);
  end loop;
  rin <= v;
  irqo.irq <= r.irq;
end process;
```

Combinational process

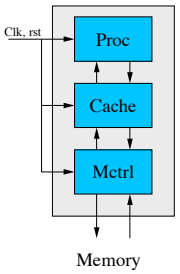
```
reg : process (clk)
begin
  if rising_edge(clk) then
    r <= rin;
  end if;
end process;

end architecture;
```

State process

Rationale: Add new variable to record, and "bookkeeping" is free!

Records simplify hierarchal designs

VHDL Instantiation




```
use work.interface.all;

entity cpu is port (
  clk     : in std_logic;
  rst     : in std_logic;
  mem_in  : in mem_in_type;
  mem_out : out mem_out_type);
end;

architecture rtl of cpu is
  signal cache_out  : cache_type;
  signal proc_out   : proc_type;
  signal mctrl_out  : mctrl_type;
begin

  u0 : proc port map
    (clk, rst, cache_out, proc_out);

  u1 : cache port map
    (clk, rst, proc_out, mem_out, cache_out);

  u2 : mctrl port map
    (clk, rst, cache_out, mem_in, mctrl_out,
     mem_out);

end architecture;
```

Leon methodolody: Use loops for logic

- Used for iterative calculations
- Index variable implicitly declared
- Typical use: iterative algorithms, priority encoding, sub-bus extraction, bus turning

```
variable v1 : std_logic_vector(0 to 7);
variable first_bit : natural;

-- find first bit set
for i in v1'range loop
  if v1(i) = '1' then
    first_bit := i; exit;
  end if;
end loop;

-- reverse bus
for i in 0 to 7 loop
    v1(i) := v2(7-i);
end loop;
```

Danger: The VHDL looks nothing like hardware. Using HDL as "algorithm description language" not as a "hardware description language".

Conclusions

- **Open-source CPUs: an alternative to licensing or designing a core.**
- **NEMERIX uses configurability of Leon to enhance its own IP.**
- **Leon's HDL methodology: records, combinational and state separation.**
- **Differs from Berkeley style: Rely on synthesis to map loops to logic.**