

MultiCore vs FPGA

John Williams
PetaLogix /
University of Queensland

<john.williams@petalogix.com>

Disclaimer

- I'm an FPGA/Linux hacker and business owner
- I'm not a multicore expert
- My professional focus is on Embedded Linux
 - Creating custom FPGA-based SoC architectures and helping customers do Linux with it.

Agenda

- Why CPUs are awesome
- Why CPUs stink
- Why FPGAs are awesome
- Why FPGAs stink
- What to do, what to do?

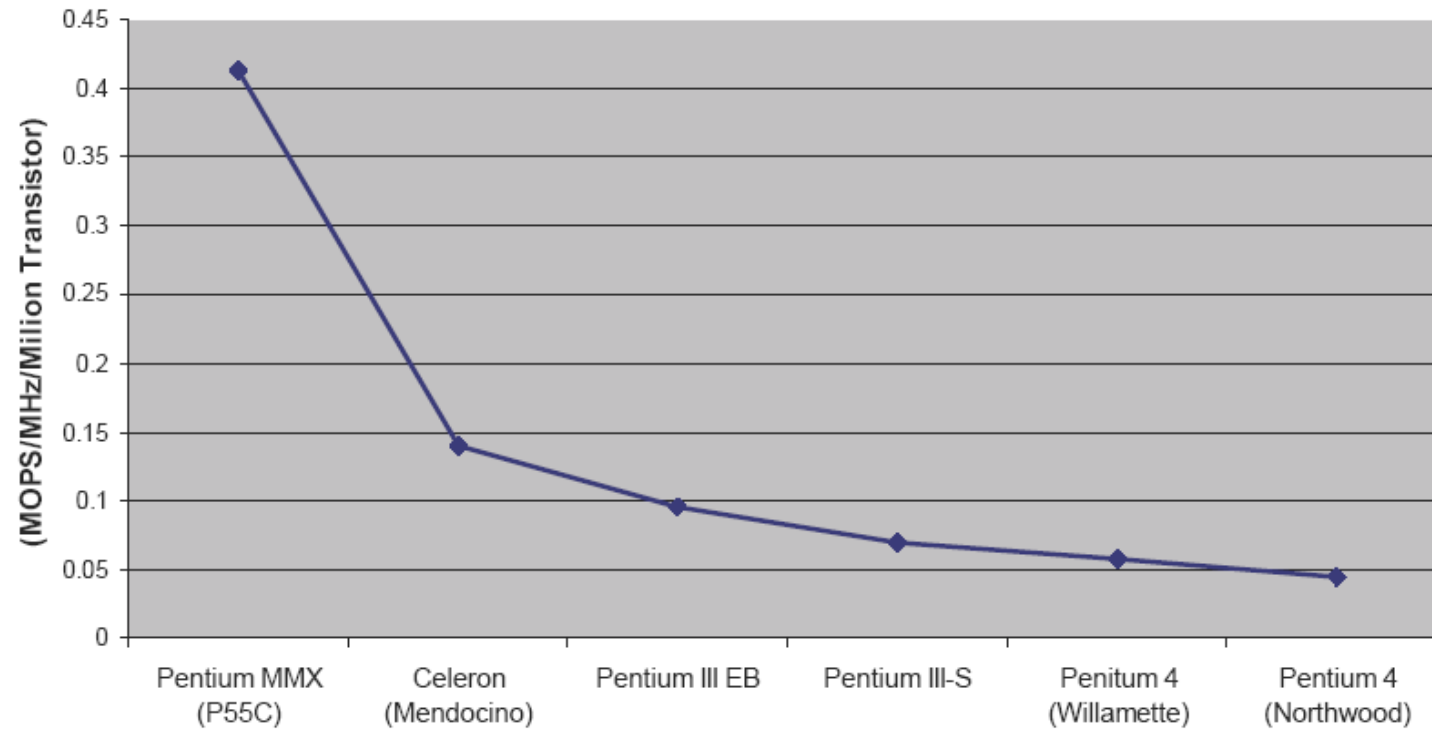
Why CPUs are awesome

```
void main(void) {  
    printf("hello, world\n");  
}
```

For the multicore crowd

```
void hello(void) {  
    printf("hello\n");  
}  
  
void main(int argc, char *argv[]) {  
    pthread_t ti[10];  
    for(int i=0;i<10;i++)  
        pthread_create(&ti[i], NULL, hello, NULL);  
}
```

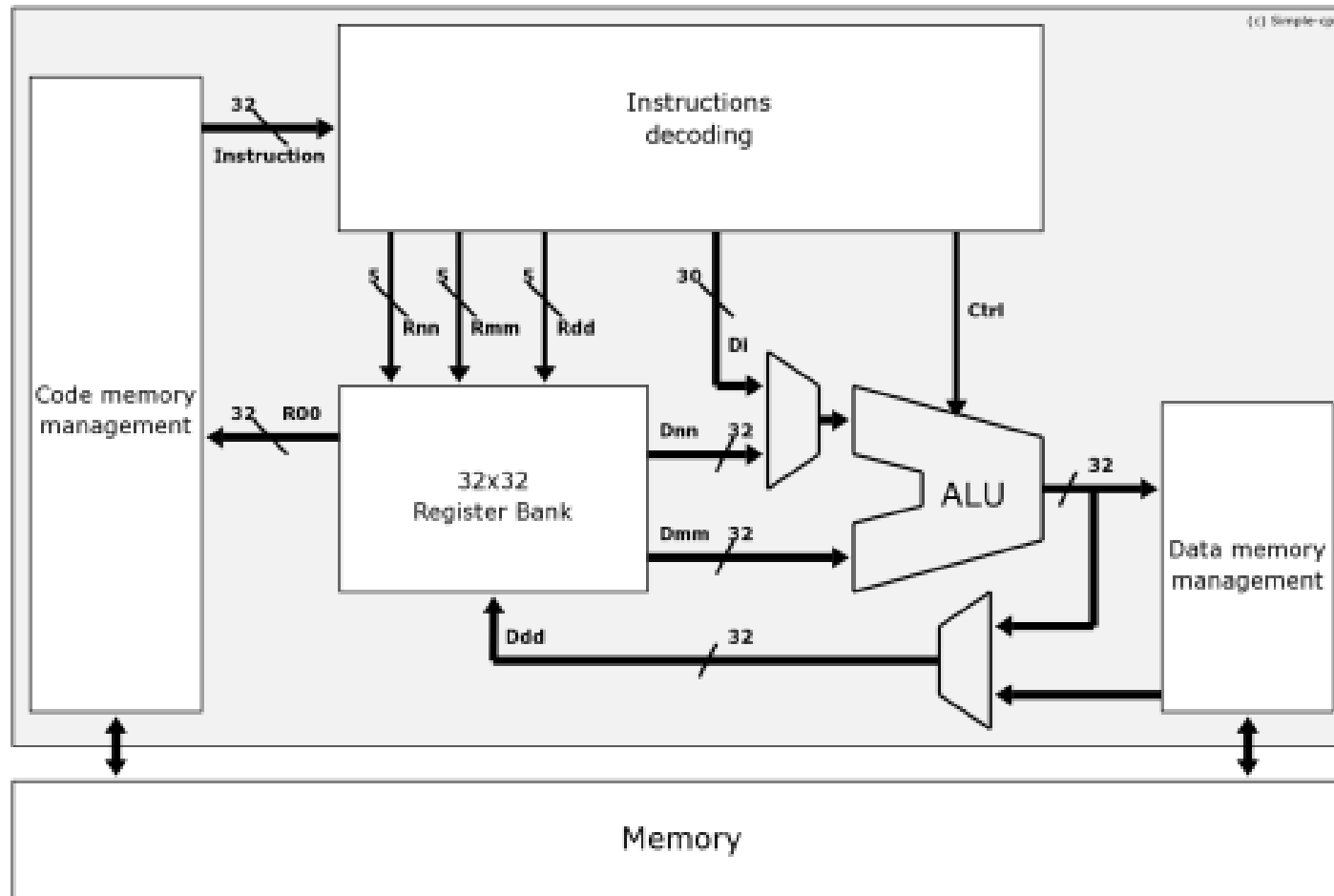
Why CPUs stink – Part I



[Wawrzynek]

Why CPUs stink – Part II

[www.simple-cpu.com]



Q: Where does the computational work happen?

AMD Phenom II X6



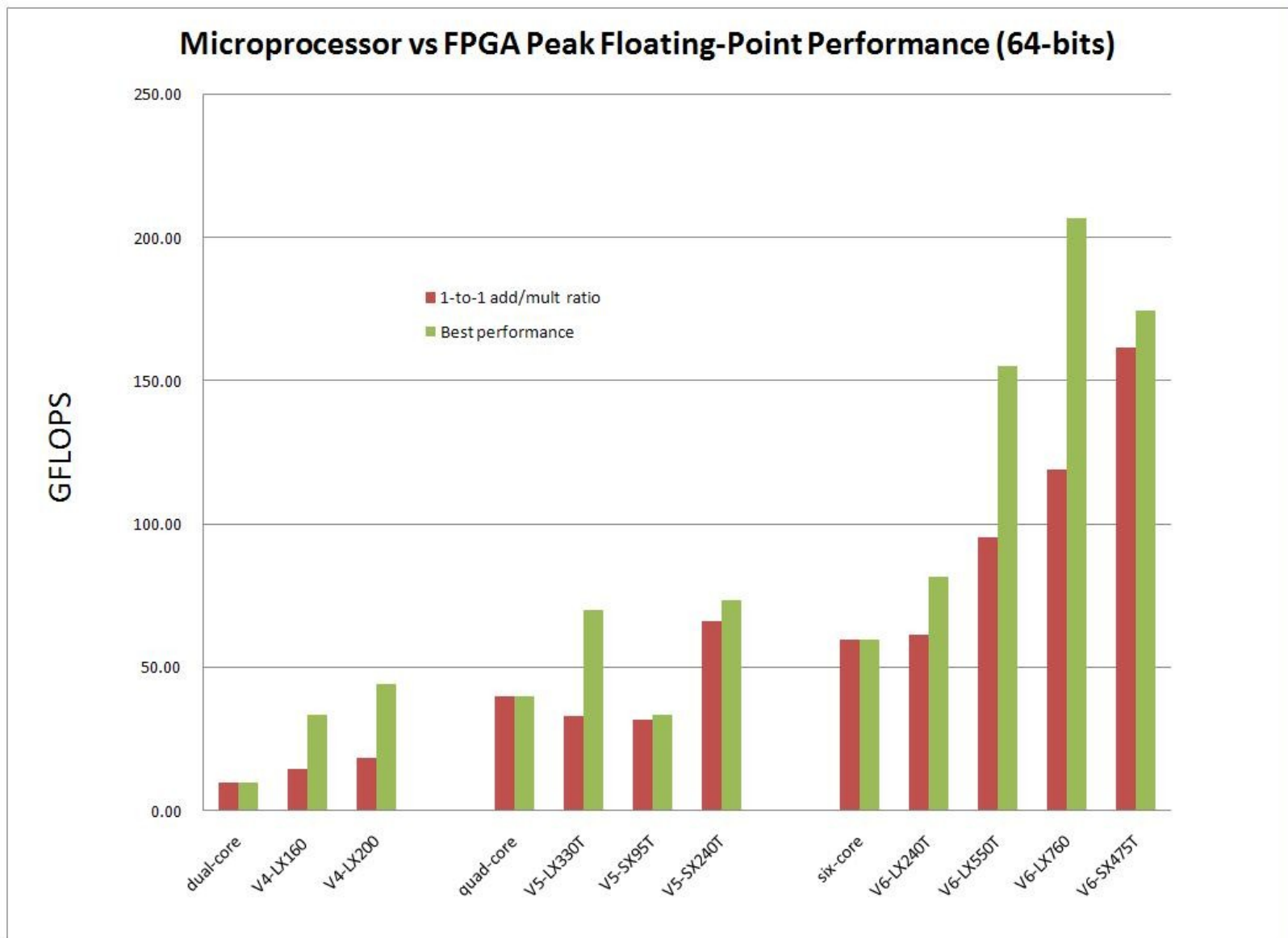
Where's Wally?

Why FPGAs are awesome

- Hardware is intrinsically parallel
 - Logic gates aren't 'sequential' unless you build it that way
- FPGAs are massive, programmable arrays of logic
 - Mountains of compute, mountains of I/O
- Fine-grained – compute at the bitwidth you require, where you require it.

Peak performance comparison

[www.hpcwire.com]



Why FPGAs stink

- There's a reason they call it *hardware*!
- Traditional HW design performed at very low level of abstraction
 - Design entire system at bit-level?

VHDL 'hello world'

```
architecture rtl of hello_world is
    constant CLK_FREQ : integer := 20000000;
    constant BLINK_FREQ : integer := 1;
    constant CNT_MAX : integer := CLK_FREQ/BLINK_FREQ/2-1;
    signal cnt : unsigned(24 downto 0);
    signal blink : std_logic;
begin
    process(clk)
    begin
        if rising_edge(clk) then
            if cnt=CNT_MAX then
                cnt <= (others => '0');
                blink <= not blink;
            else
                cnt <= cnt + 1;
            end if;
        end if;
    end process;

    led <= blink;

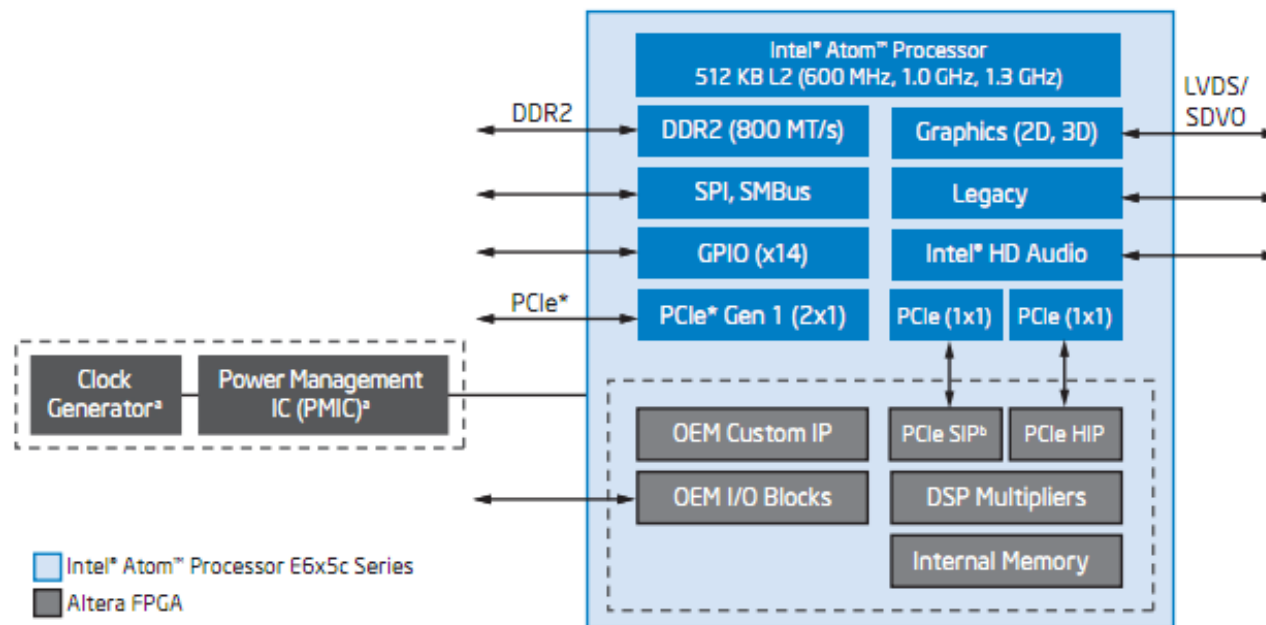
end rtl;
```

Hardware platforms

- At a simplistic level there are really only two architectures
 - FPGA on the I/O bus
 - FPGA on the memory bus

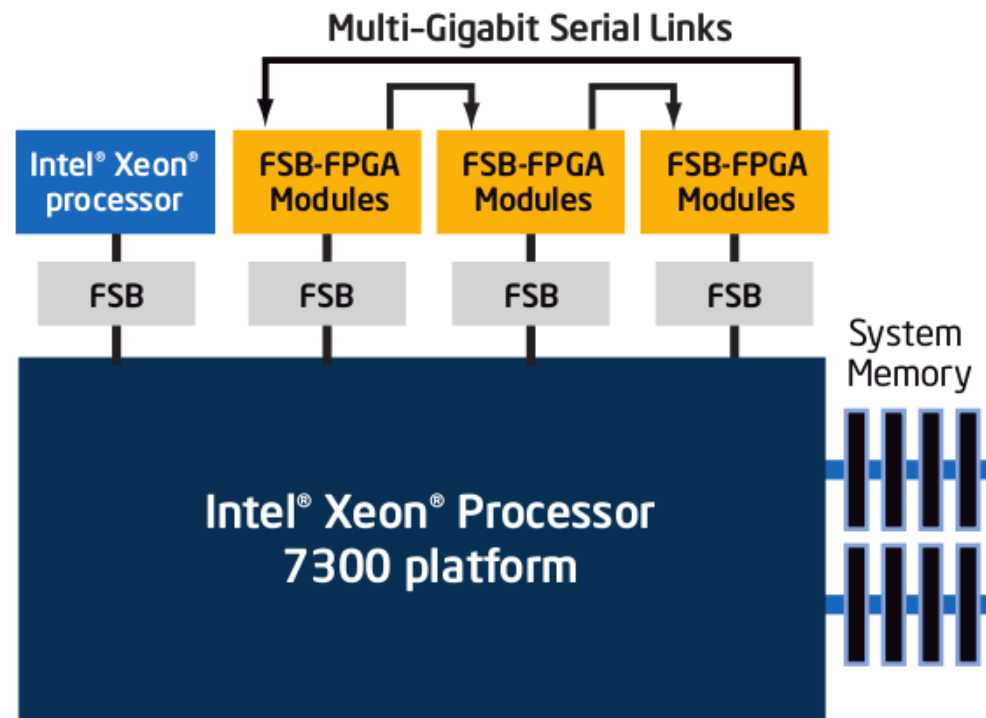
Intel Atom E6x5C

- 600MHz Atom + Altera FPGA
- 2x PCIe 1x links between CPU and FPGA



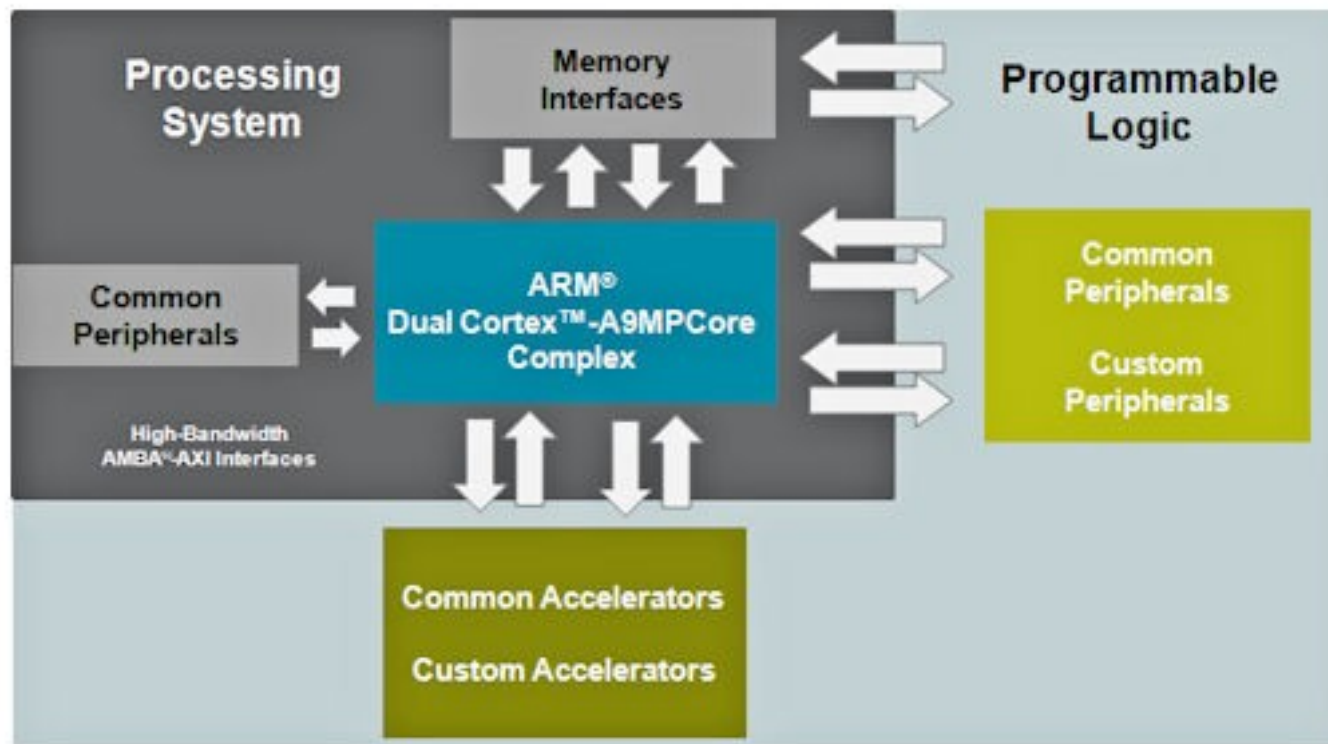
Xilinx ACP

- Large Xilinx FPGA in Xeon socket / FSB
- Cache-coherent system memory interface
- Several commercial variants available



Xilinx EPP

- Dual ARM Cortex-A9 + SoC
- High bandwidth, cache-coherent fabric interconnect



Others

- Huge variety of “FPGA on backplane” devices
- Mostly variants on a theme
 - One or more large FPGAs
 - Multiple memory banks
 - 1x – 16x PCIe links
 - Multi Gbps I/O
 - Optical, etc

What about the tools?

- VHDL?!!
 - no...
- High Level Synthesis is the key
 - C to gates
 - A holy grail for a long time
 - Early tools were
 - Weak
 - tightly constrained
 - Only usable by their creators

What about the tools?

- This is changing
- Many commercial vendors (serious \$\$\$)
 - AutoESL
 - Synfora (bought by Synopsis in 2010)
 - Catapult C (Mentor)
 - CoDeveloper (Impulse)
- Open Source
 - C-to-Verilog
 - GAUT
 - SPARK (dead?)

What about the tools?

- C is a terrible language for expressing parallelism
 - Functional languages map nicely onto circuits
 - e.g. Haskell
- Graphical tools
 - Matlab/Simulink
 - National Instruments LabView
 - ...

Other thoughts

- OpenCL
 - Open Compute Language
 - Vendor-neutral (unlike CUDA)
 - Intended for heterogenous computing environments
 - Still seems fairly immature?
- Does it make sense on FPGAs?
 - Maybe
 - OpenCL to express parallelism
 - HLS to synthesis individual kernels
- Watch this space

etc

- Berkeley Emulation Engine
 - Internet in a box
 - Many-core research
-

Conclusion

- FPGA **vs** multicore is a false dichotomy
- FPGA **and** multicore will be (part of) the answer
- If you think the single → multi threaded software transition was bumpy, you ain't seen nothing yet
- Hardware technology and architectures are less important than tools