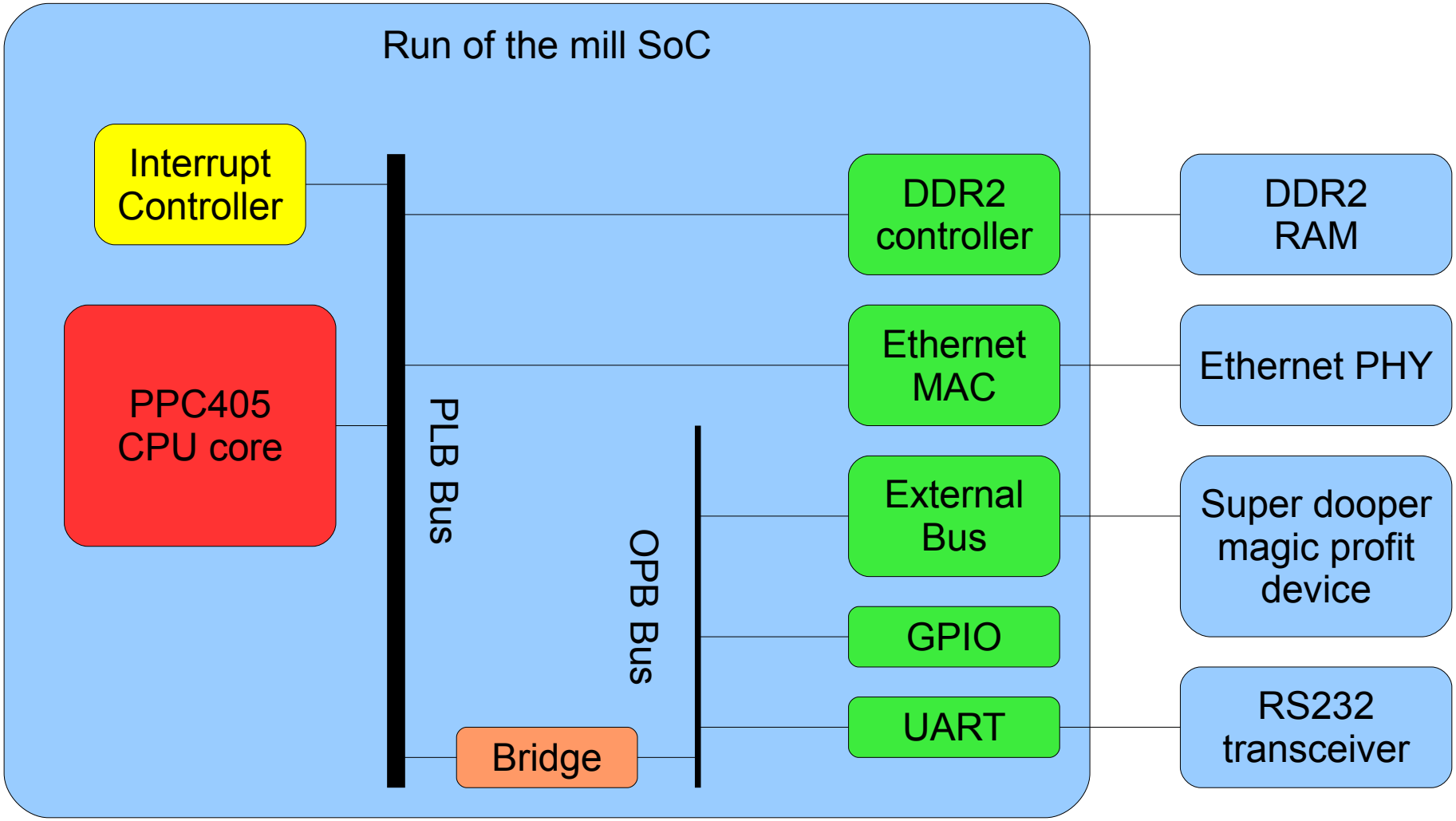
Lessons Learned from Linux on an FPGA

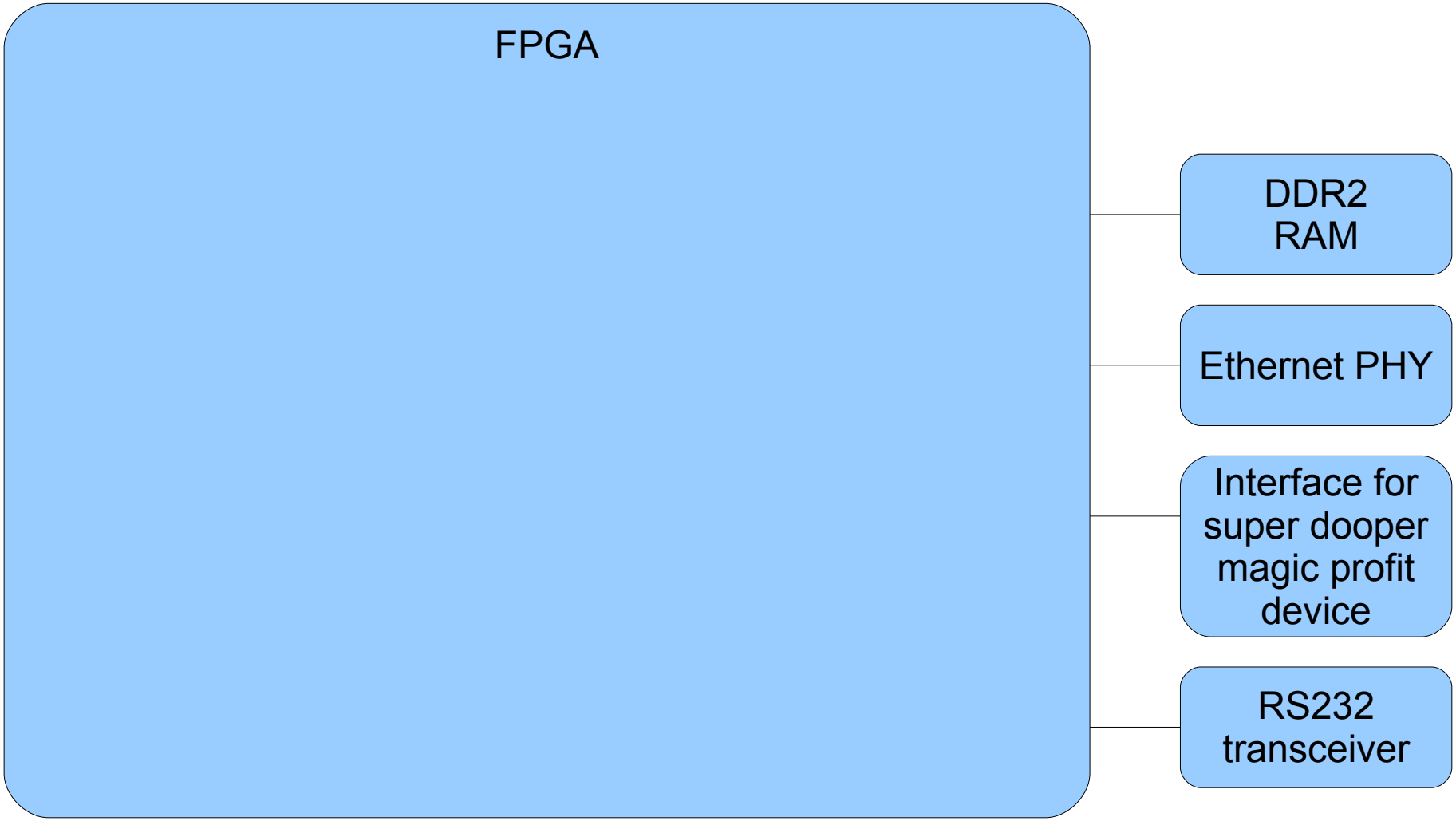
Grant Likely
Secret Lab Technologies Ltd.

Embedded Linux Conference
April 16, 2008

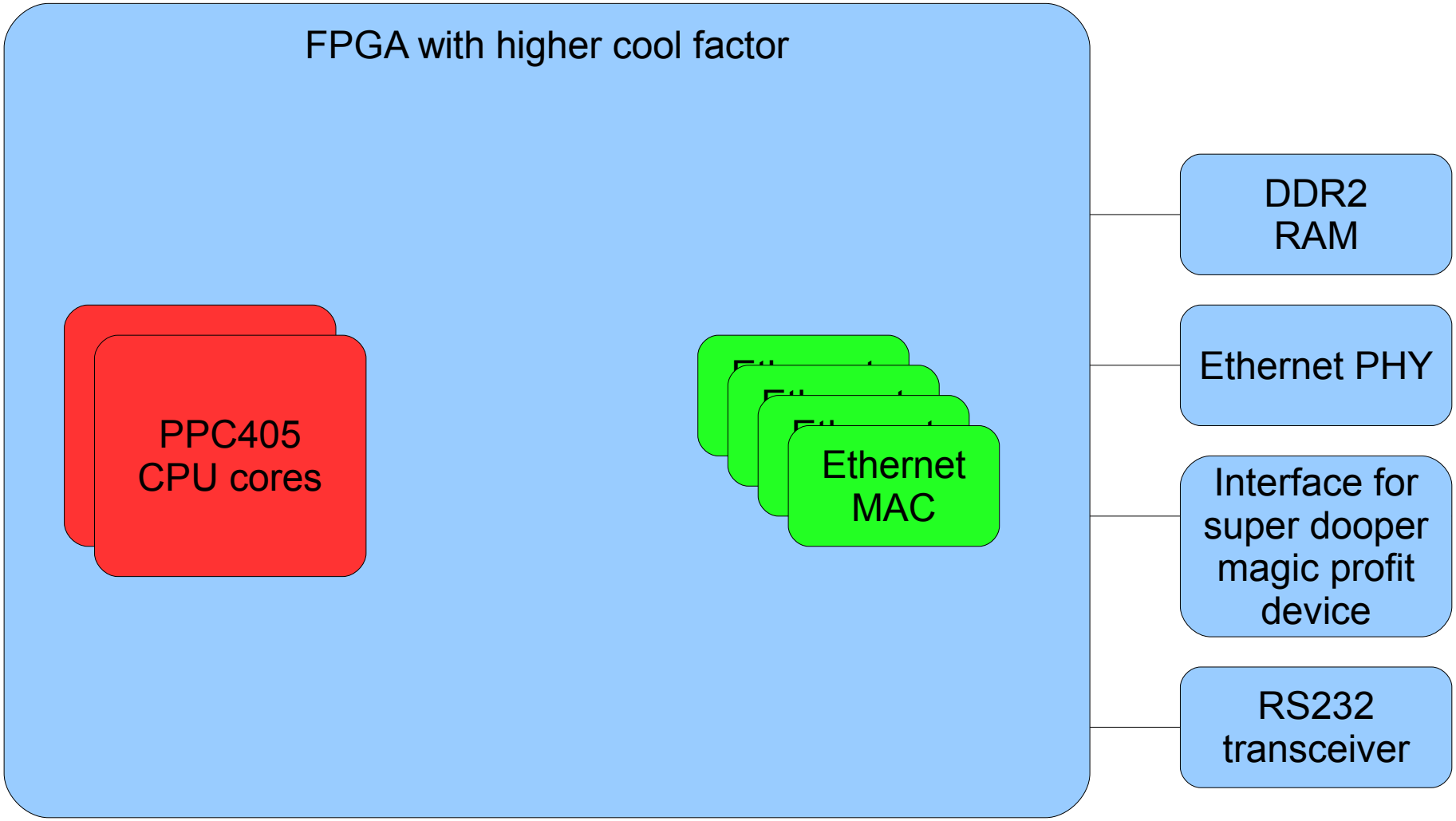
Typical SoC


Run of the mill SoC
Interrupt Controller
PPC405 CPU core
PLB Bus
OPB Bus
Bridge
DDR2 controller
Ethernet MAC
External Bus
GPIO
UART
DDR2 RAM
Ethernet PHY
Super dooper magic profit device
RS232 transceiver

FPGA System


FPGA
DDR2 RAM
Ethernet PHY
Interface for super dooper magic profit device
RS232 transceiver

Virtex 4FX FPGA System



Virtex FPGA Linux Support

- Basic support for PowerPC in mainline
  - Serial ports
  - ML300/403 Framebuffer
  - SystemACE device
- Extra drivers in Xilinx public git tree
  - Ethernet devices, DMA, I2C, GPIO
  - Microblaze support
  - Currently merged with v2.6.24-rc8
  - Rewrite needed before mainline

Lesson Learned: Don't make developers lives hard

- Hardware Engineers don't like to compile kernels
- Software Engineers don't like to synthesis bitstreams
- Nobody likes to compile user space.
- Device Tree is your friend.

Lesson Learned: Get your drivers into mainline

- Andrew talked about this this morning

Lesson Learned: Get your drivers into mainline

- Andrew talked about this this morning
- You're not doing anything that novel anyway

Lesson Learned: Get your drivers into mainline

- Andrew talked about this this morning
- You're not doing anything that novel anyway
- No; you're really not

Lesson Learned: Get your drivers into mainline

- Andrew talked about this this morning
- You're not doing anything that novel anyway
- No; you're really not
- I mean it -- others have fought with what you're fighting with before; someone else will have some advice

Lesson Learned: Hardware is the new Software

- Should follow software best practices
- Revision Control
- Automated builds
- Peer review
  - Let the SW folks look at the HW design!!! (and visa versa)

Lesson Learned: It really might be a hardware bug

- Talk to your hardware engineer **immediately** when you have problems
- They can probe **any** signal inside the FPGA design

Lesson Learned: It's easy to spend all your budget on “boring” stuff

- PCI, USB, ETH, Serial
- Matt Mackall, “If your vendor isn't pushing stuff to mainline, go beat them up”
- You've got better things to do
  - Like the custom application logic you're putting in the FPGA
  - Otherwise you'd probably use an SoC
  - Start the design chosing platforms that already work well.
- Cypress c67x00 driver example

Lesson Learend: Make things Work first...

- ...before you make them fast/small/clever
- Get a stable baseline so that you know when things break
- Things break frequently, so be setup to know **when** things break

Lesson Learned: Prepare for dynamic hardware in the kernel

- **Expect** things to change; they will anyway
- This happens for SoCs to; just at a much lower turnaround rate

Lesson Learned: All of your assumptions are wrong

- Customizable peripherals, don't make assumptions about configuration.
- Design your code to be prepared for changes to functionality
- ie. Xilinx DMA
- But be realistic too. You can't design for something that doesn't exist; but you can design your code for things that are likely to change.
- Avoid hardcoding

Lesson Learned: User Space Sucks

- It's easy to cross compile kernels
- It's hard to cross compile userspace
- Get userspace solved early