FreeBSD support for Stanford NetFPGA

Wojciech A. Koszek
wkoszek@FreeBSD.org

2009.09.17

Work was done as a part of the internship at:

Helsinki Institute of Information Technology
<http://www.HIIT.fi>

Ericsson Nomadic Lab
<http://www.ericsson.com>

Helsinki, Finland

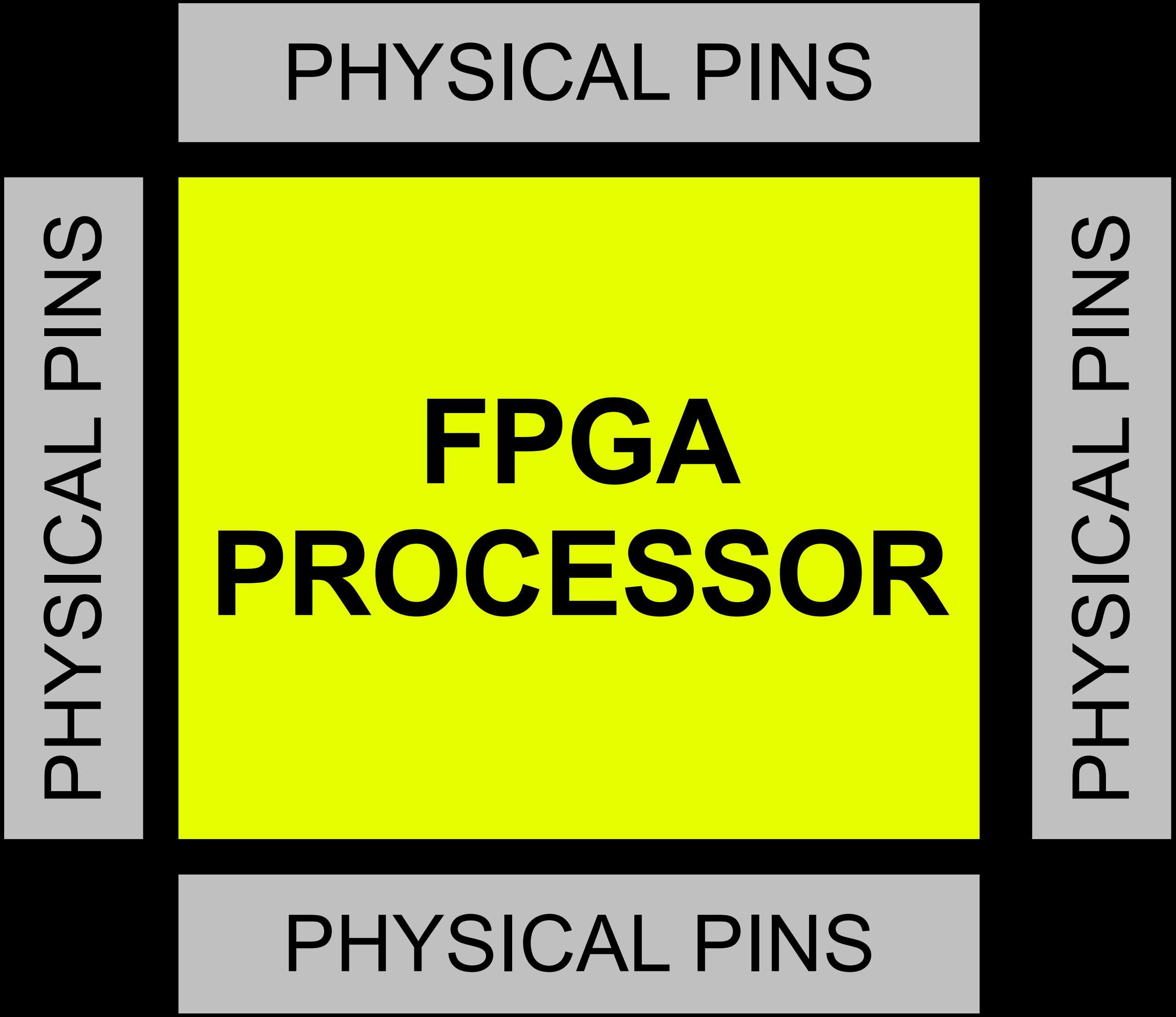
Code I'm going to discuss:

http://people.freebsd.org/~wkoszek/netfpga

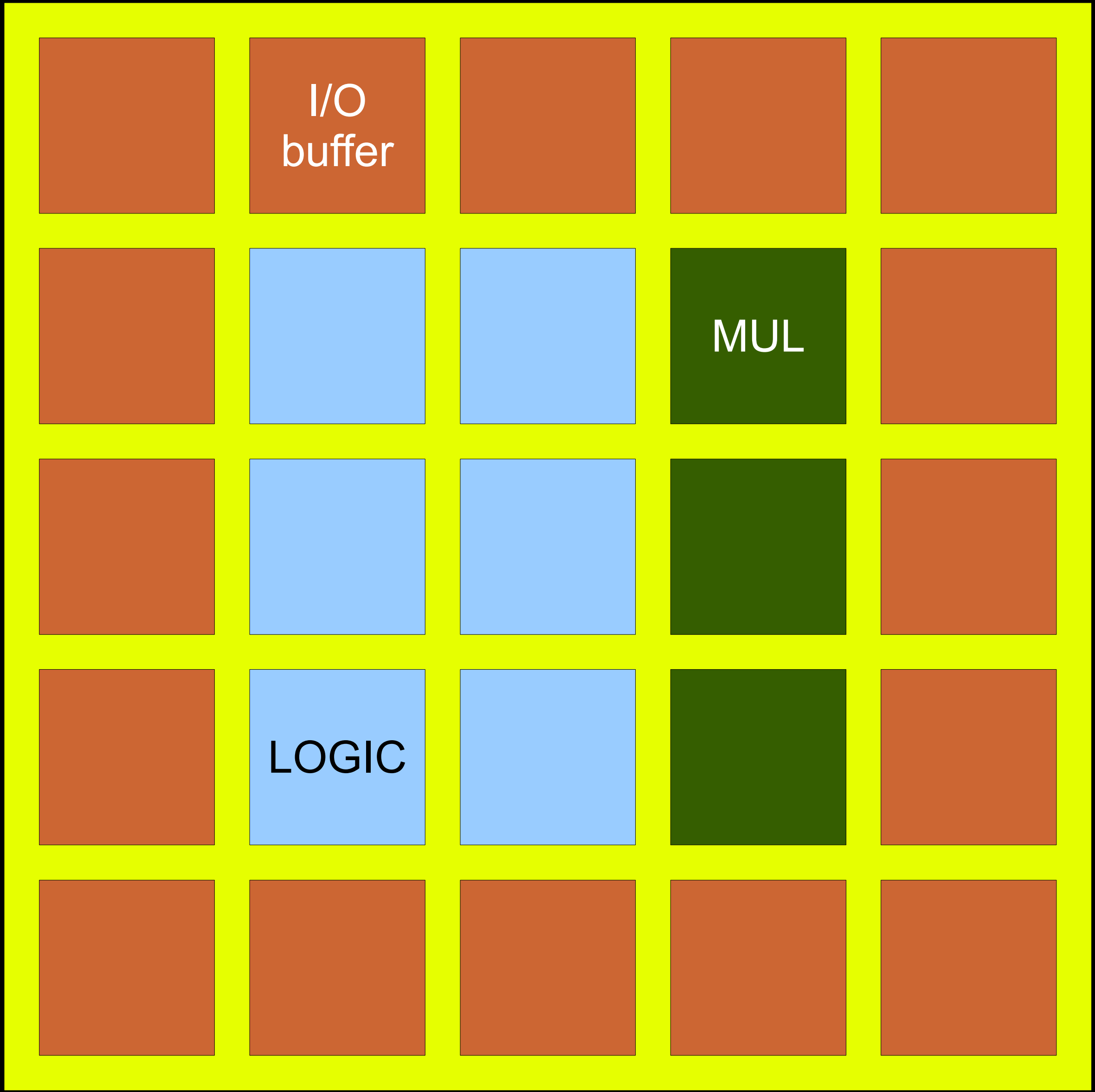
`netfpga-devel@` mailing list has this code as well

Very short introduction to FPGAs

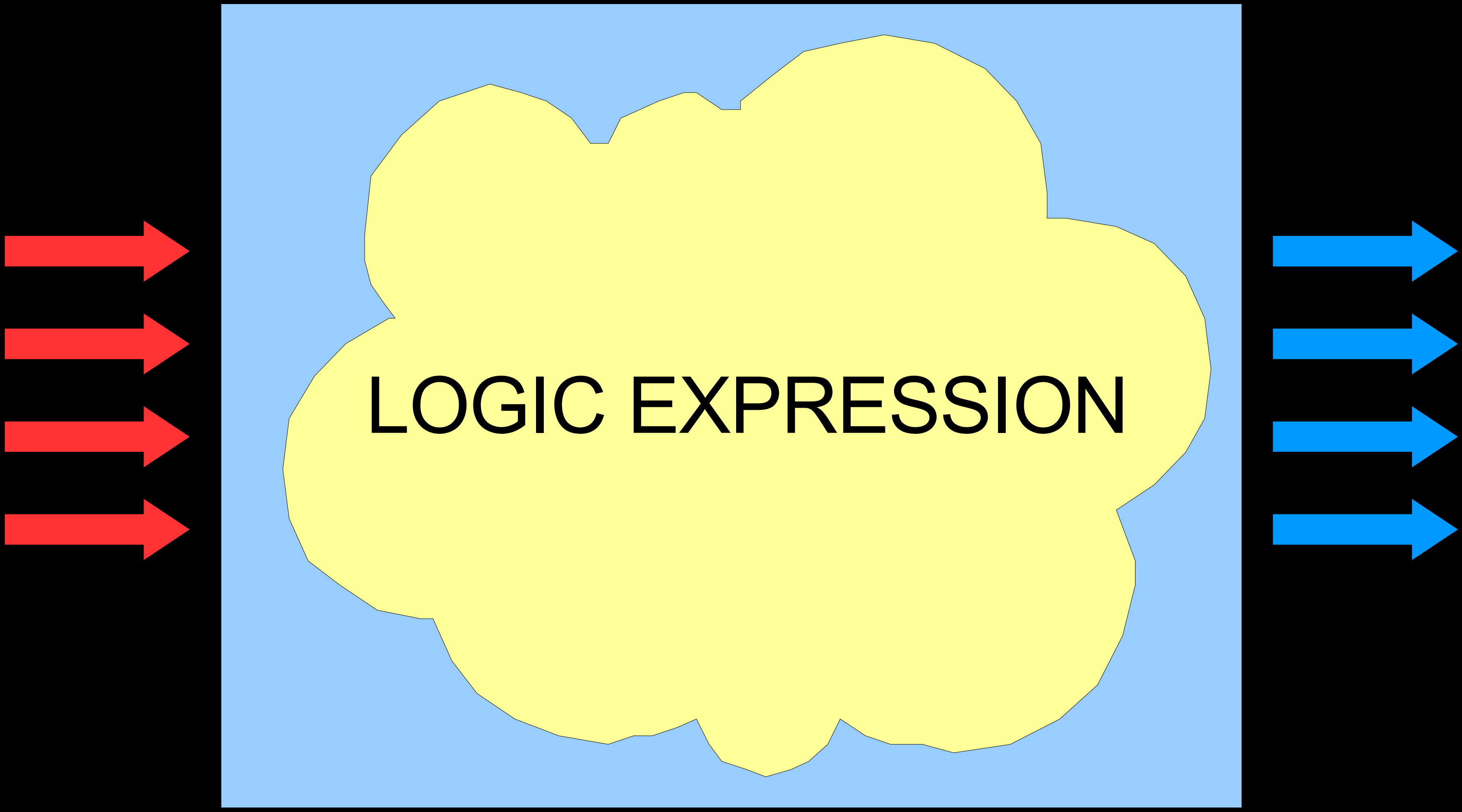
Physically, FPGA processor is just a chip



FPGA processor is just a bunch of digital logic blocks

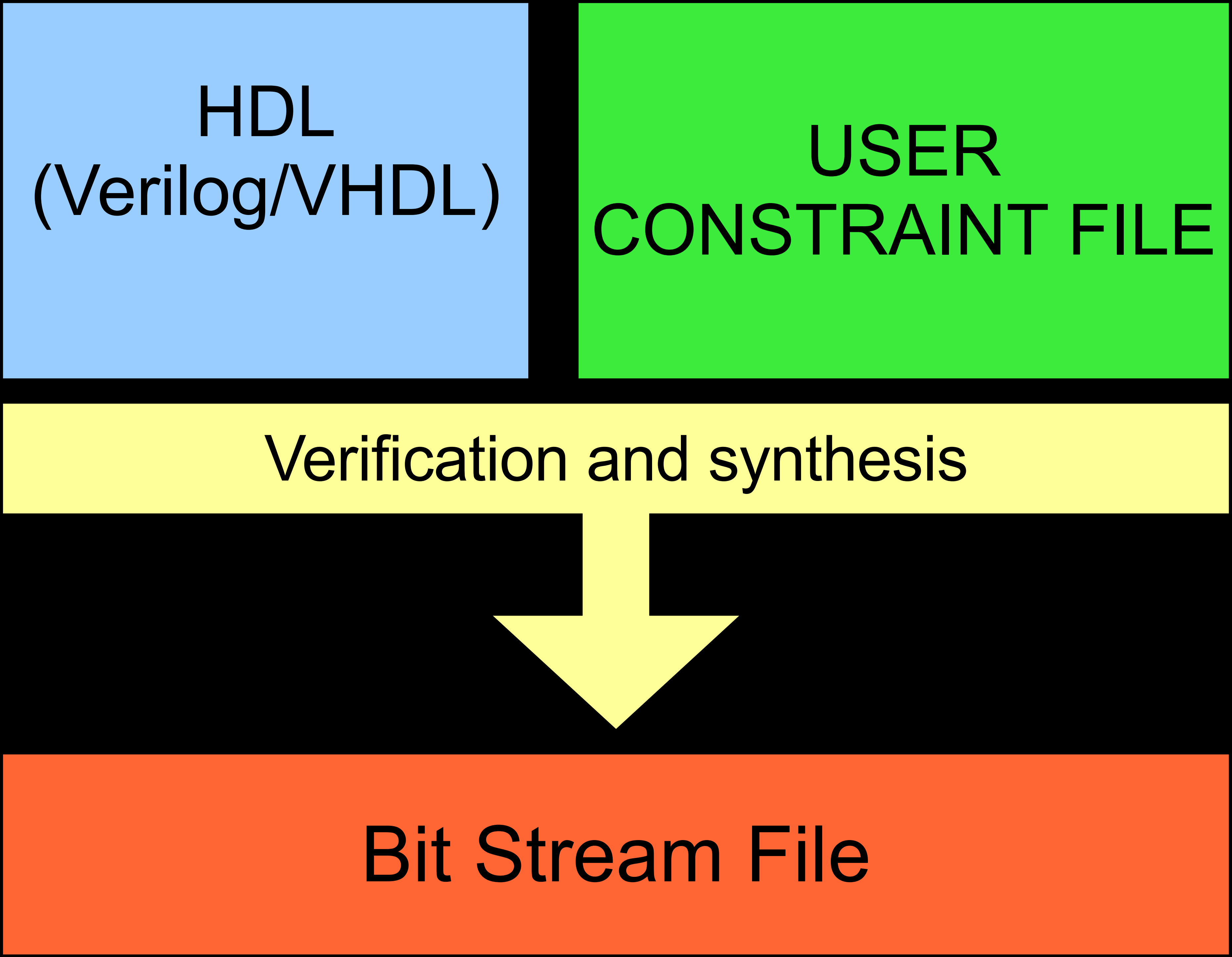


Each digital logic block can be programmed to perform specific action on its inputs



Block functionality and block connections can be changed at will with **Hardware Description Language** like Verilog

Physical connections between HDL's names and physical PINs are specified in separate **User Constraints File** (UCF) file



HDL
(Verilog/VHDL)

USER
CONSTRAINT FILE

Verification and synthesis

Bit Stream File

Examples of cool stuff people do in FPGAs right now

- Accelerated computations
  - Cryptography
  - Compression
- Complete Systems-on-Chip
  - CPU with MMU
- **Accelerated Networking**

FPGA chip can perform specific task much, much faster than conventional CPUs

FPGA work in FreeBSD

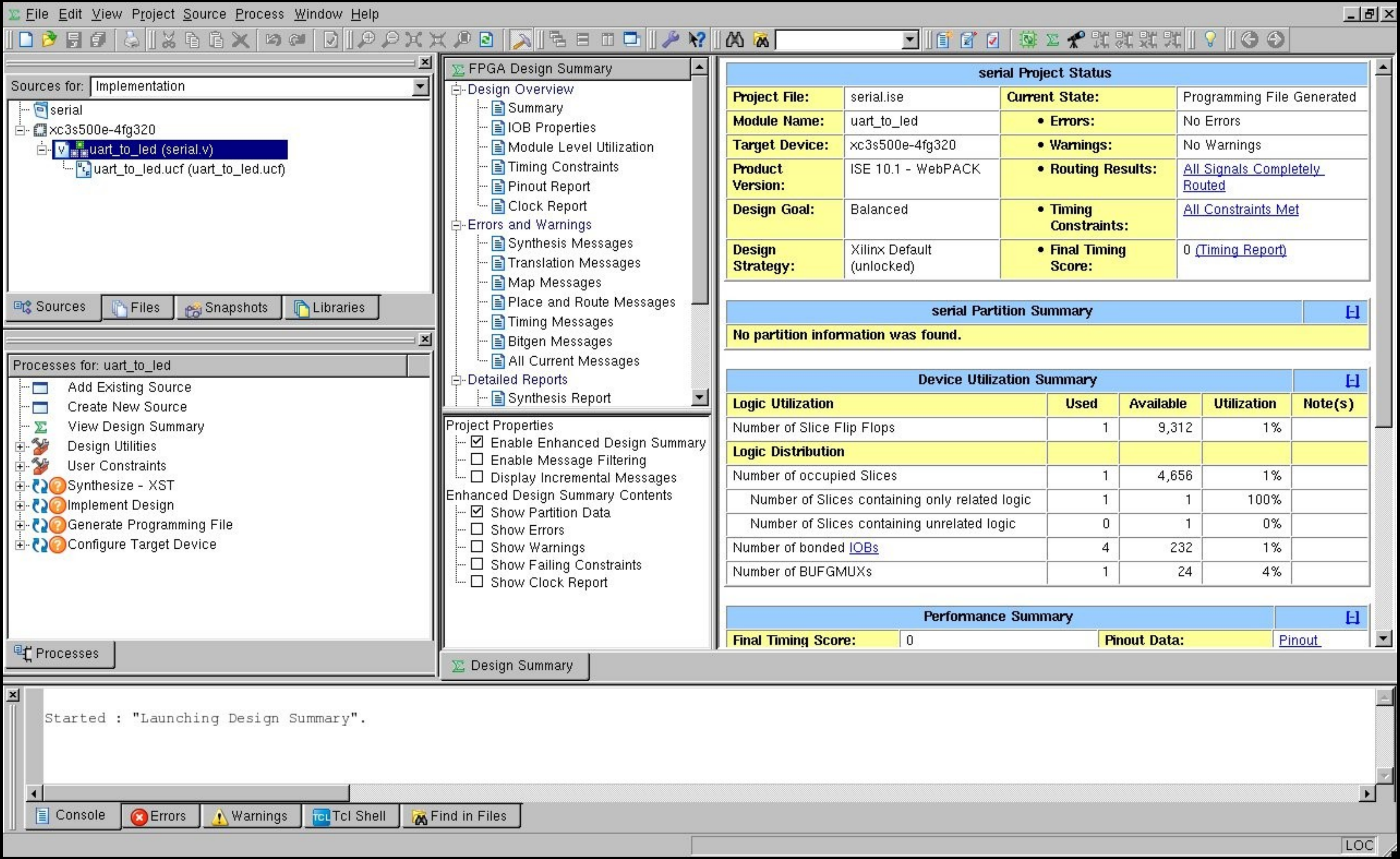
Xilinx ISE WebPack
(IDE for Verilog)
used to work for me year ago
on FreeBSD

(Linux emulation layer)

FPGA processor programming used to work for me as well

With xc3sprog project tools, I was able to program Xilinx Spartan 3 Starter Kit

WebPACK 10.1 with recent FreeBSD-CURRENT



Project Properties
- ☑ Enable Enhanced Design Summary
- ☐ Enable Message Filtering
- ☐ Display Incremental Messages

Enhanced Design Summary Contents
- ☑ Show Partition Data
- ☐ Show Errors
- ☐ Show Warnings
- ☐ Show Failing Constraints
- ☐ Show Clock Report

serial Project Status			
Project File:	serial.ise	**Current State:**	Programming File Generated
Module Name:	uart_to_led	• **Errors:**	No Errors
Target Device:	xc3s500e-4fg320	• **Warnings:**	No Warnings
Product Version:	ISE 10.1 - WebPACK	• **Routing Results:**	All Signals Completely Routed
Design Goal:	Balanced	• **Timing Constraints:**	All Constraints Met
Design Strategy:	Xilinx Default (unlocked)	• **Final Timing Score:**	0 (Timing Report)

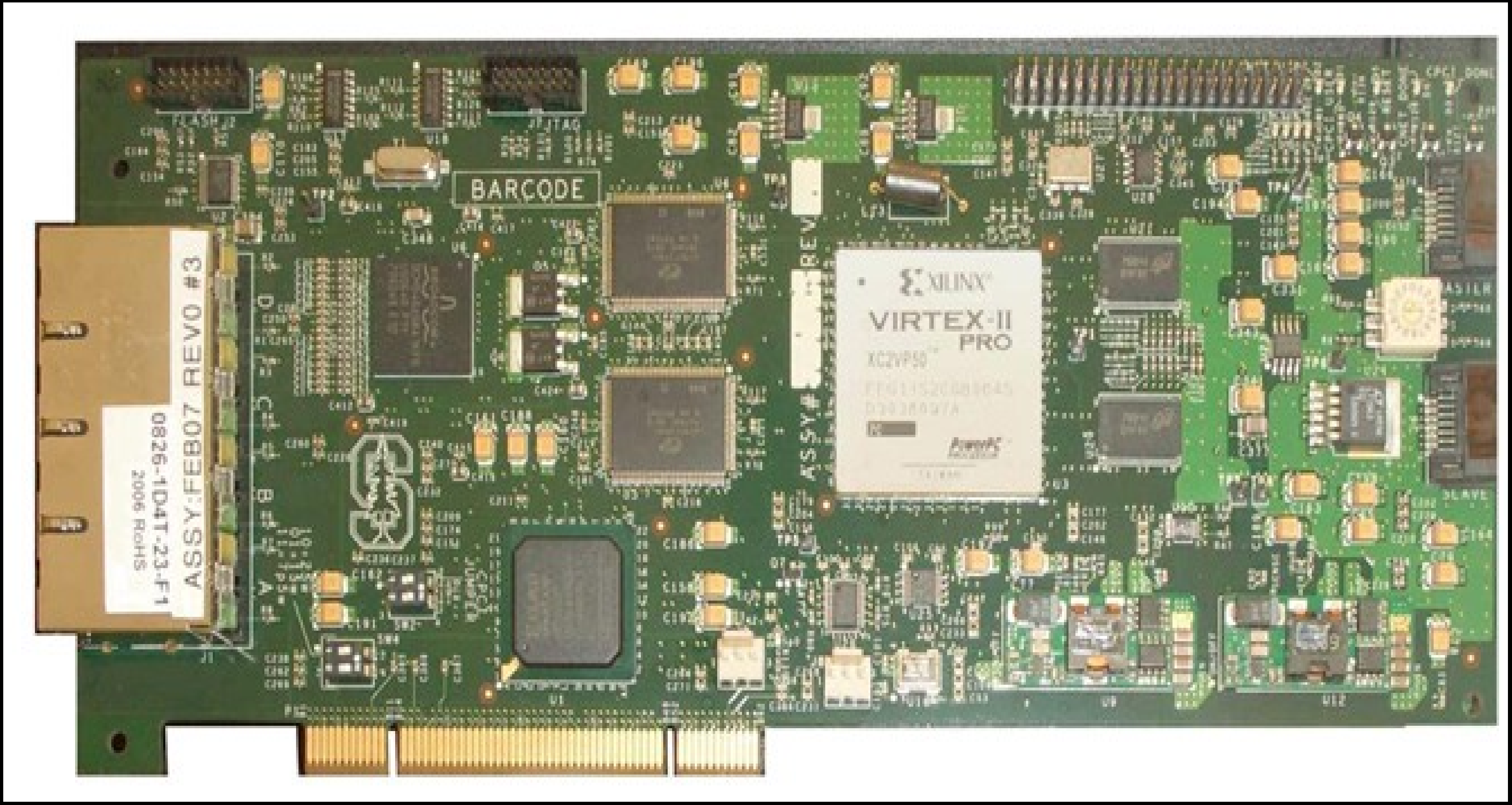
serial Partition Summary
No partition information was found.

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	1	9,312	1%	
Logic Distribution				
Number of occupied Slices	1	4,656	1%	
Number of Slices containing only related logic	1	1	100%	
Number of Slices containing unrelated logic	0	1	0%	
Number of bonded IOBs	4	232	1%	
Number of BUFGMUXs	1	24	4%	

Performance Summary			
Final Timing Score:	0	**Pinout Data:**	Pinout

```
Started : "Launching Design Summary".
```

NetFPGA card



Yet another network adapter?!

..well, sort of:

- 4 ports of Gigabit Ethernet handled by Broadcom 5464SX
- 2 high speed, serial I/O connectors
- 64MB of DDR2 DRAM, 4.5MB of SRAM
- PCI interface

...but FPGA processors are present as well

Other NetFPGA Work

www.netfpga.org

OpenFlow Reference implementation
Reference Ipv4 Router
Reference Ethernet Switch
Reference 4-port NIC
VPN appliances
various buffer-queue stuff

Past projects have all been developed on LINUX :-(

How does it look like in practice?

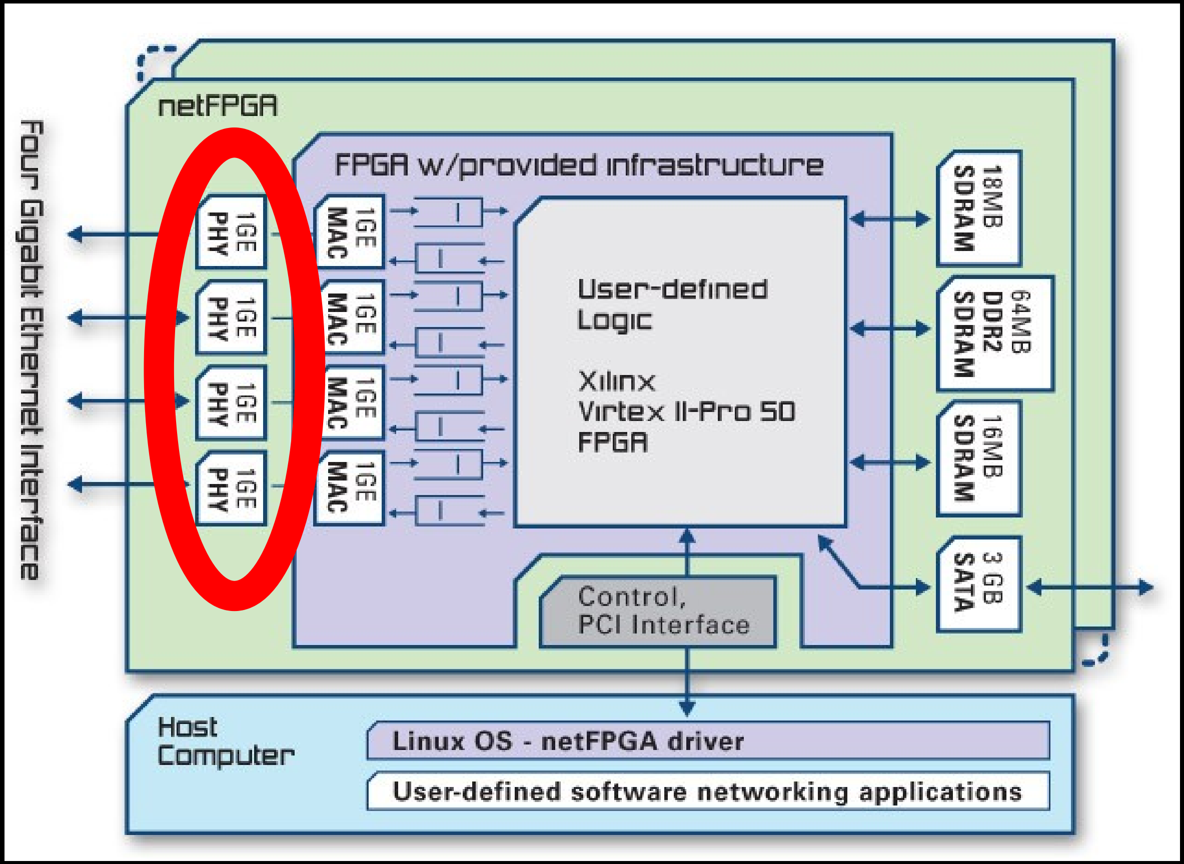
NetFPGA: 3 puzzles

``Firmware'': functionality provider (bitstream)

Userspace tools: for firmware (bitstream) upload

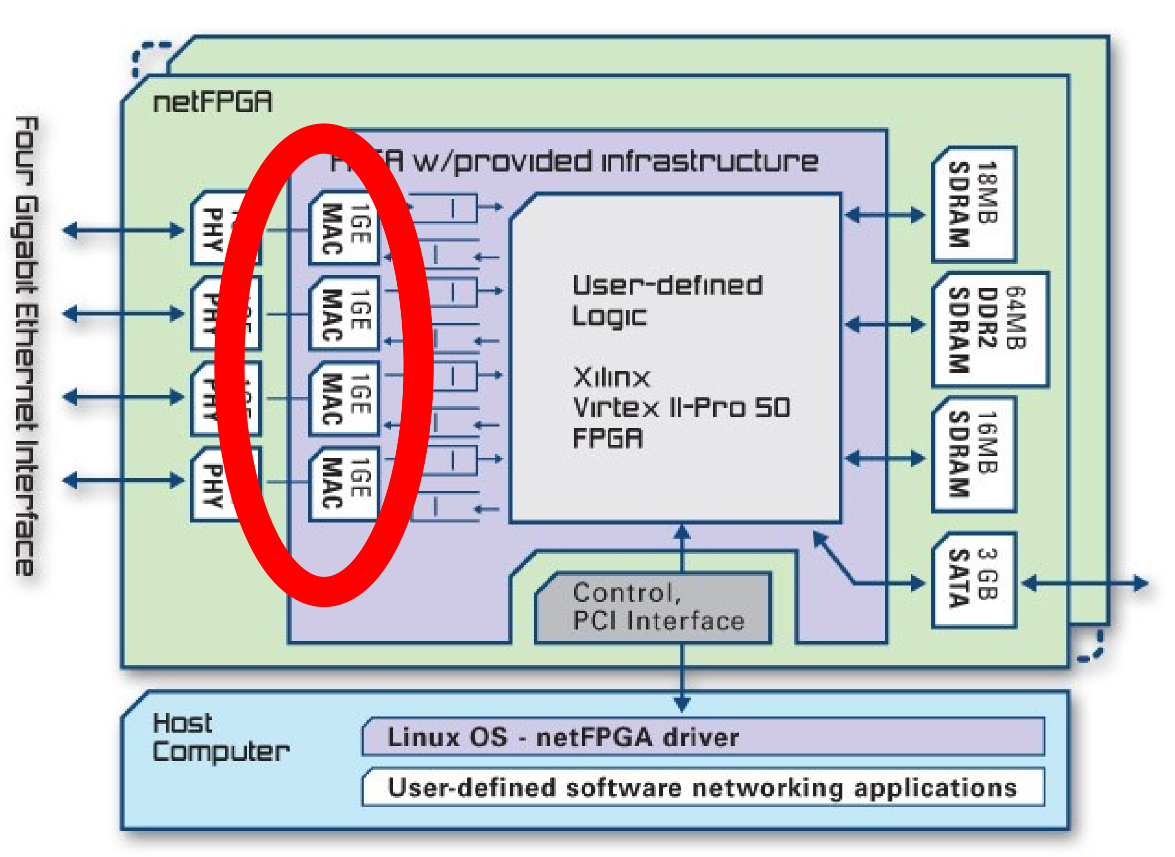
Kernel driver: low-level glue

Broadcom PHY deals with physical aspects of the Ethernet



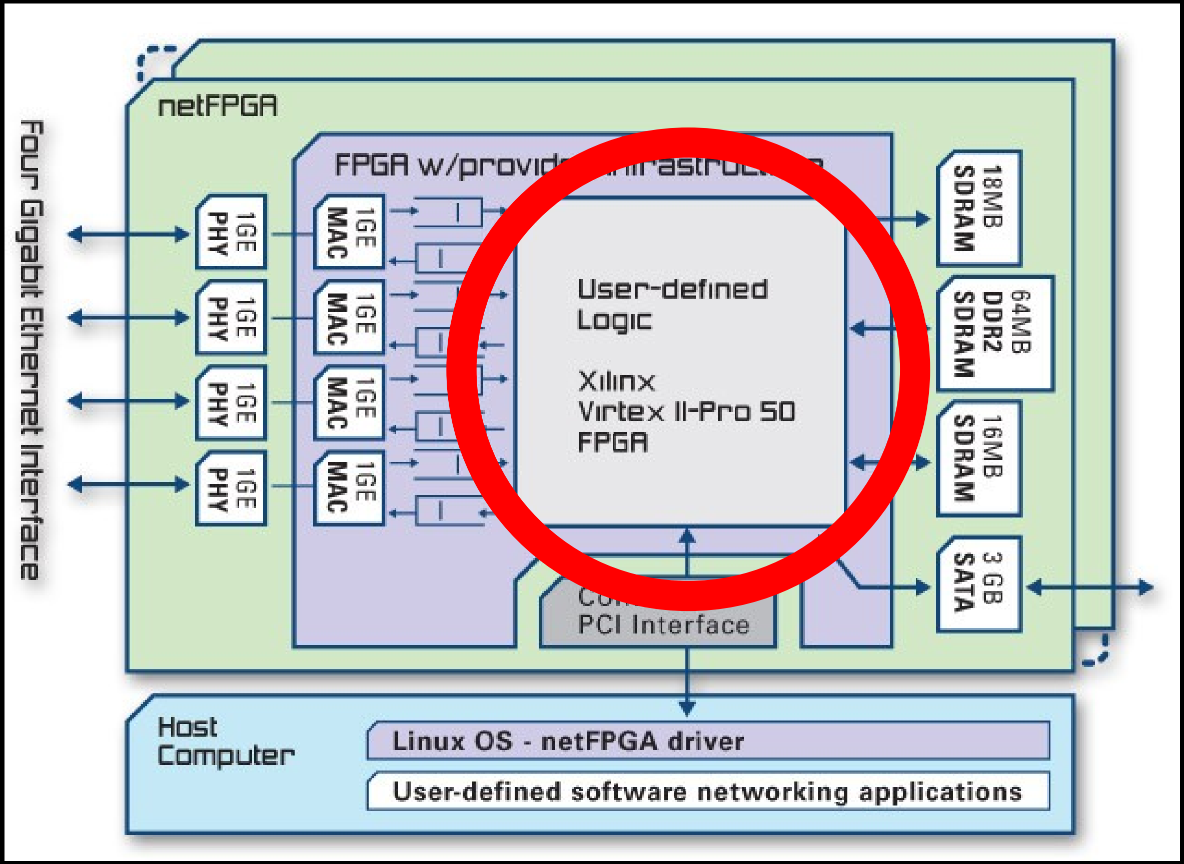
This chip doesn't have documentation available publicly :-(

Broadcom chip is tied to Xilinx Virtex II FPGA processor


Four Gigabit Ethernet Interface
netFPGA
FPGA w/provided infrastructure
PHY
PHY
PHY
PHY
1GE MAC
1GE MAC
1GE MAC
1GE MAC
User-defined Logic
Xilinx Virtex II-Pro 50 FPGA
18MB SDRAM
64MB DDR2 SDRAM
16MB SDRAM
3 GB SATA
Control, PCI Interface
Host Computer
Linux OS - netFPGA driver
User-defined software networking applications

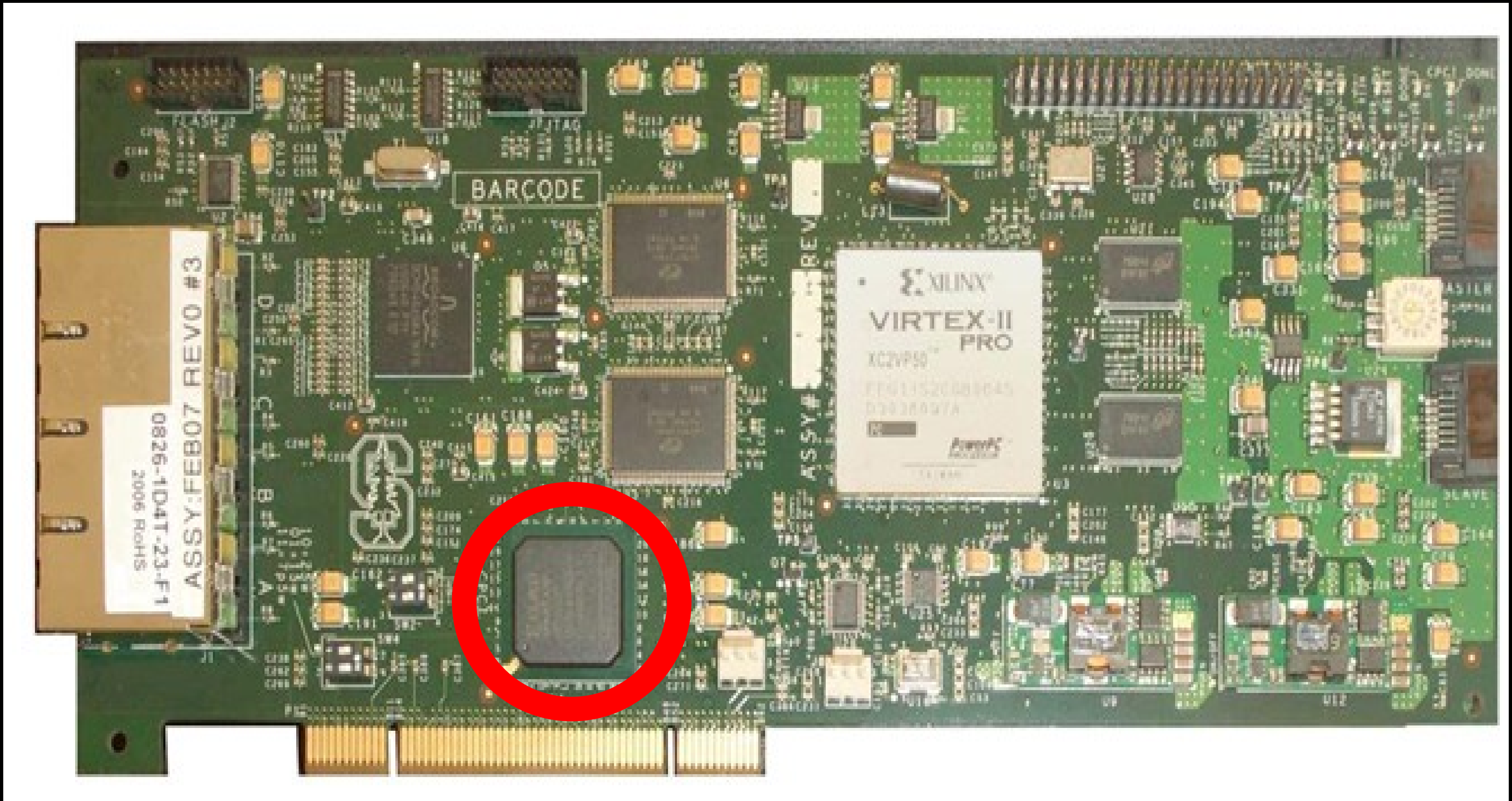
Processor implements Ethernet MAC functionality (MAC this is provided by Xilinx)

The rest of the functionality comes from the designer



My work was based on ready-to-use reference design called ``4 port 1Gbit Ethernet NIC''

PCI communication is handled by separate, smaller FPGA chip



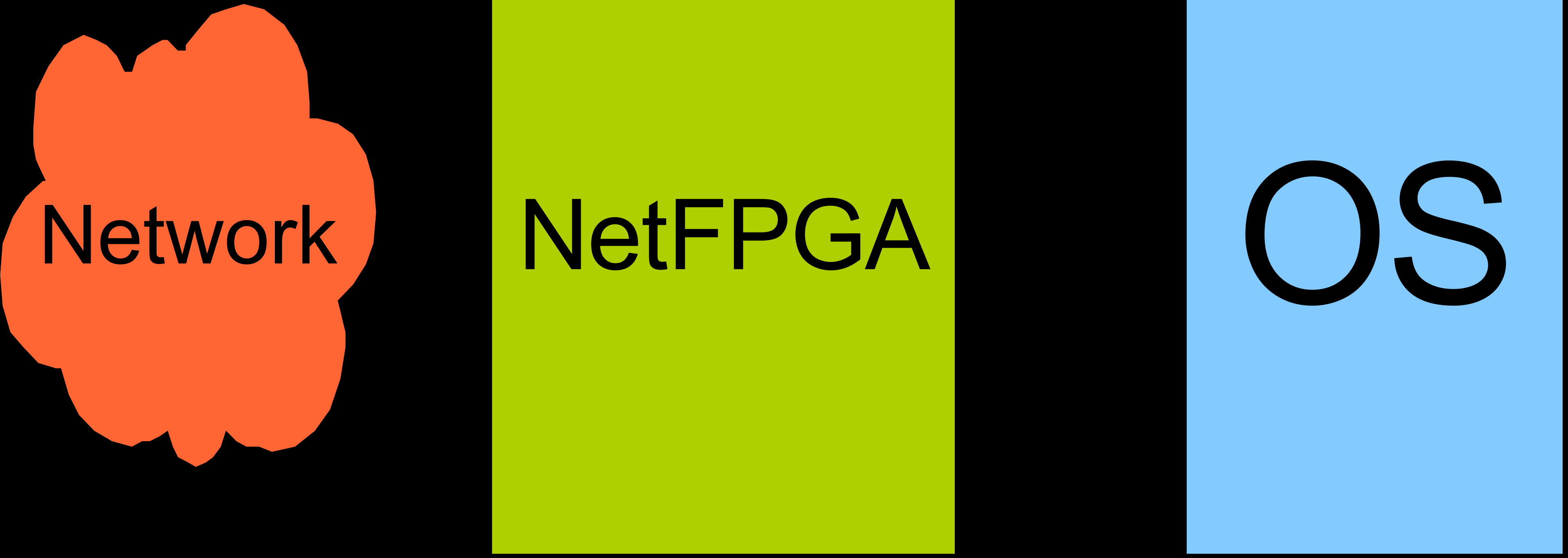
NetFPGA naming

CPCI: small FPGA (Spartan2) responsible for PCI interface

CNET: BIG FPGA for Ethernet control

Packet transmission in the NetFPGA world

Idle state

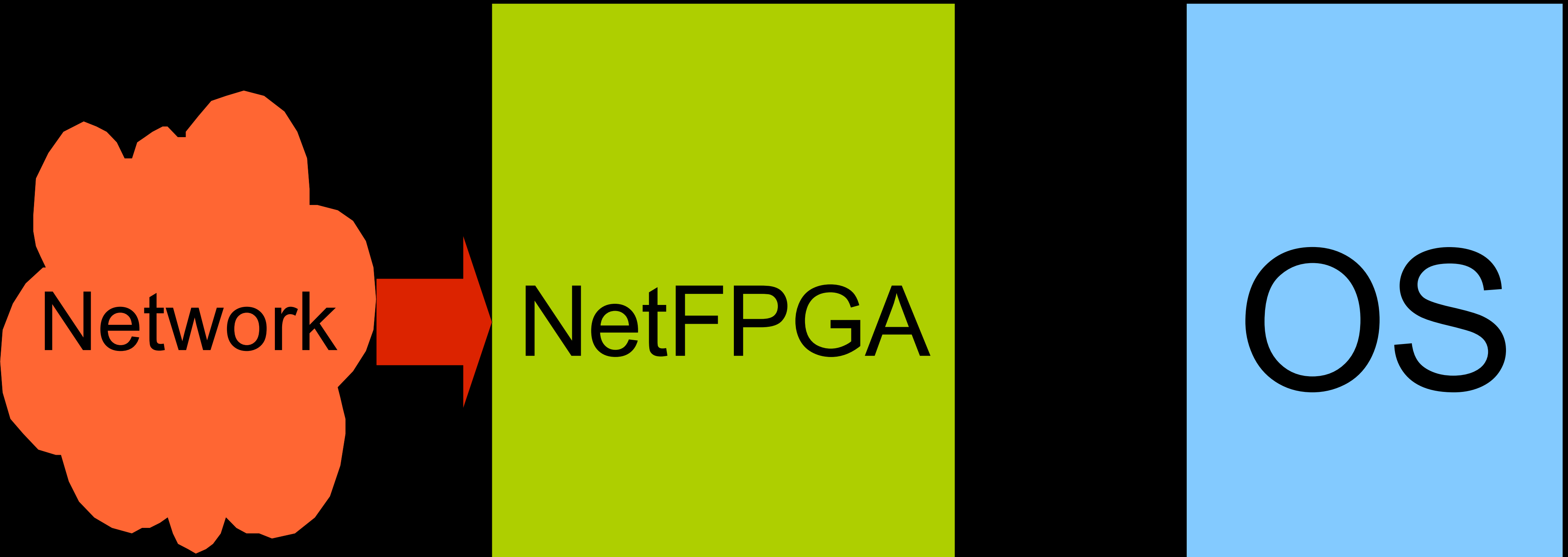


Network

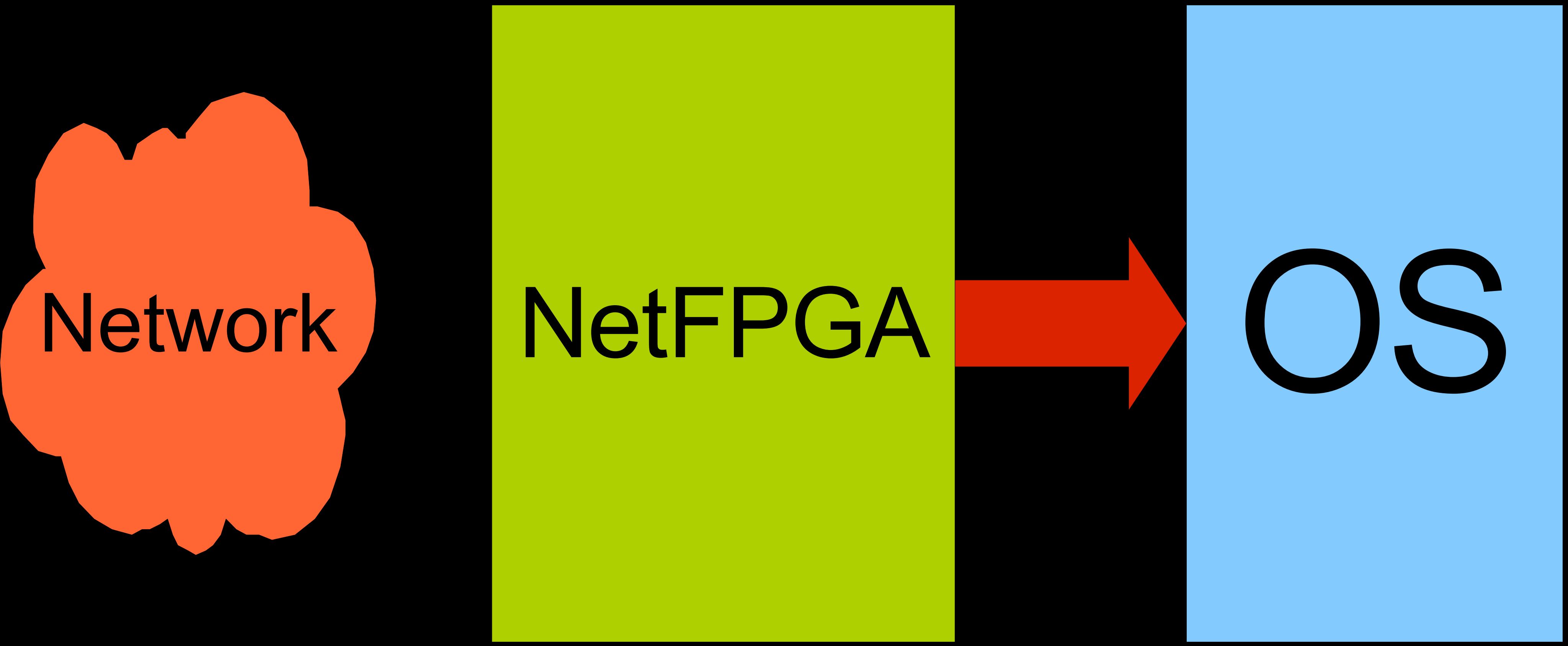
NetFPGA

OS

Data is being sent to the card

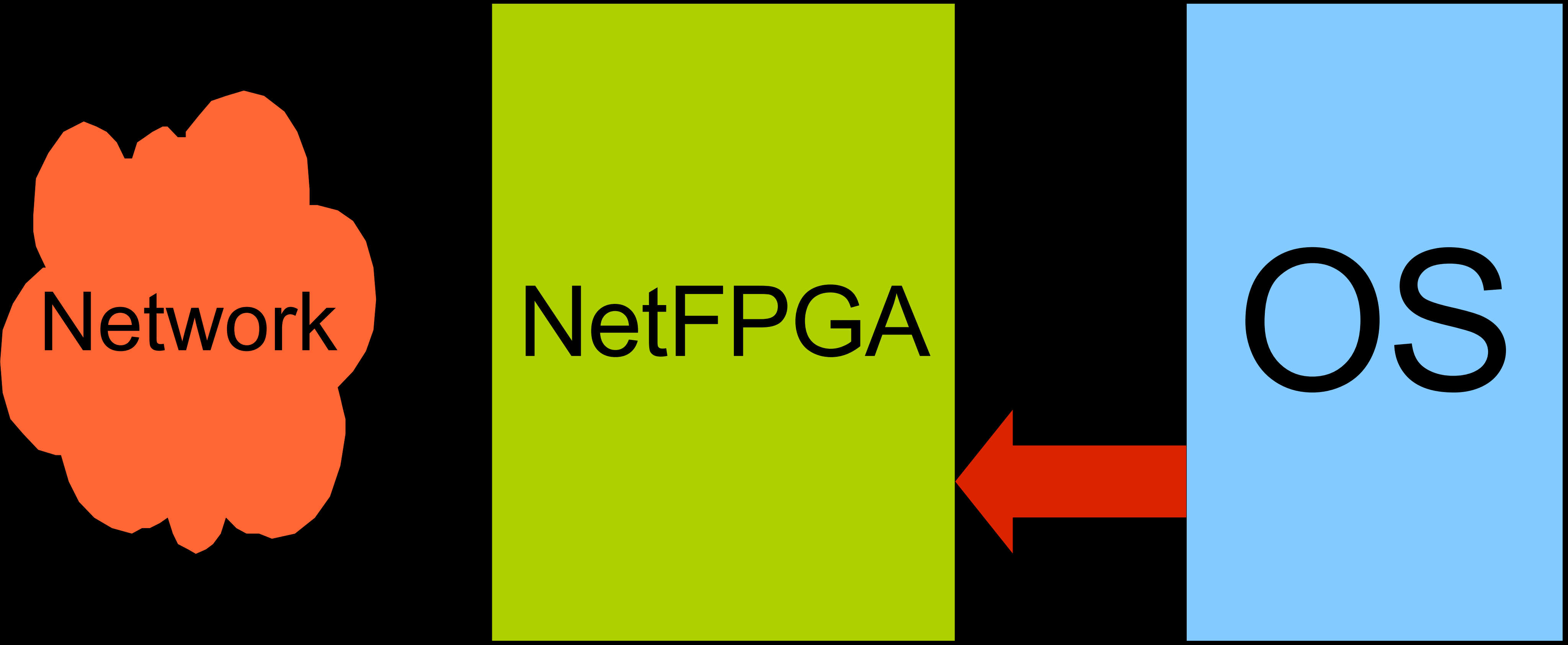


Interrupt is delivered



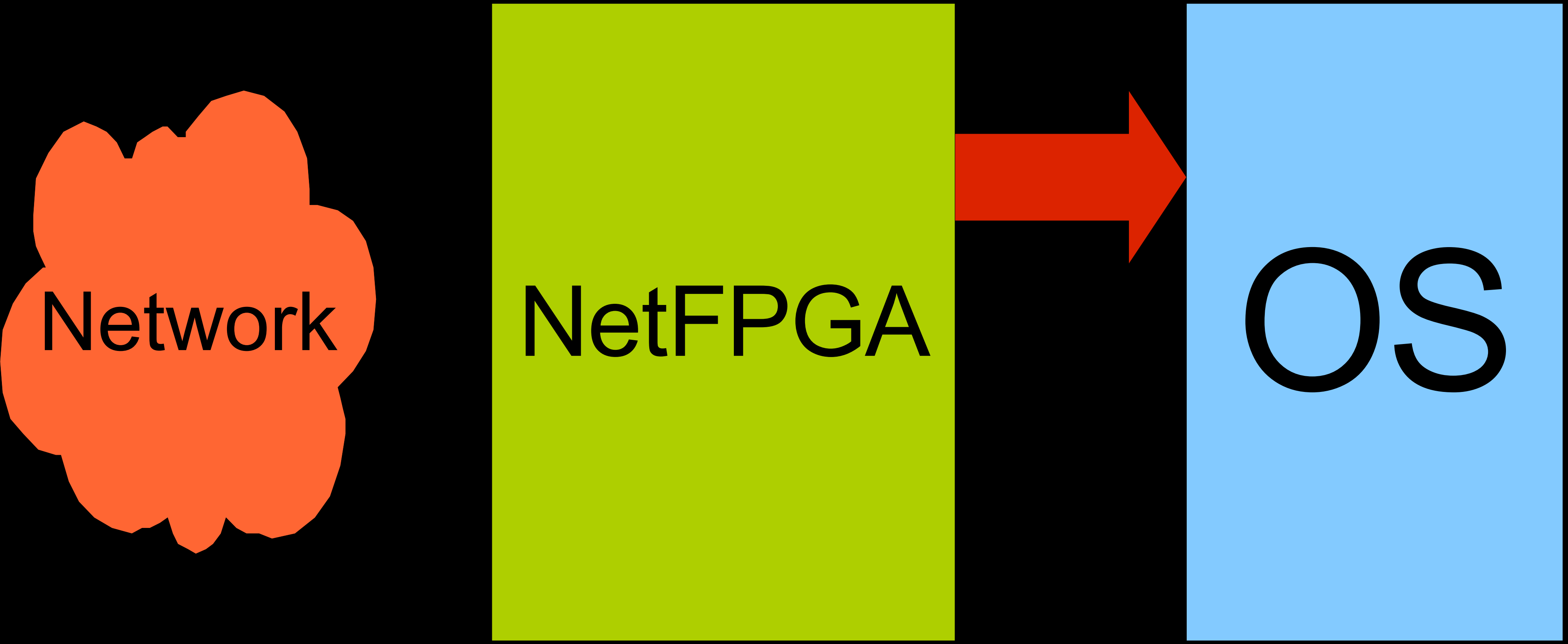
„DATA AVAILABLE”

In order to „see”, which port has a data available, you read a register



You get the transfer length this way too

DMA transfer is started

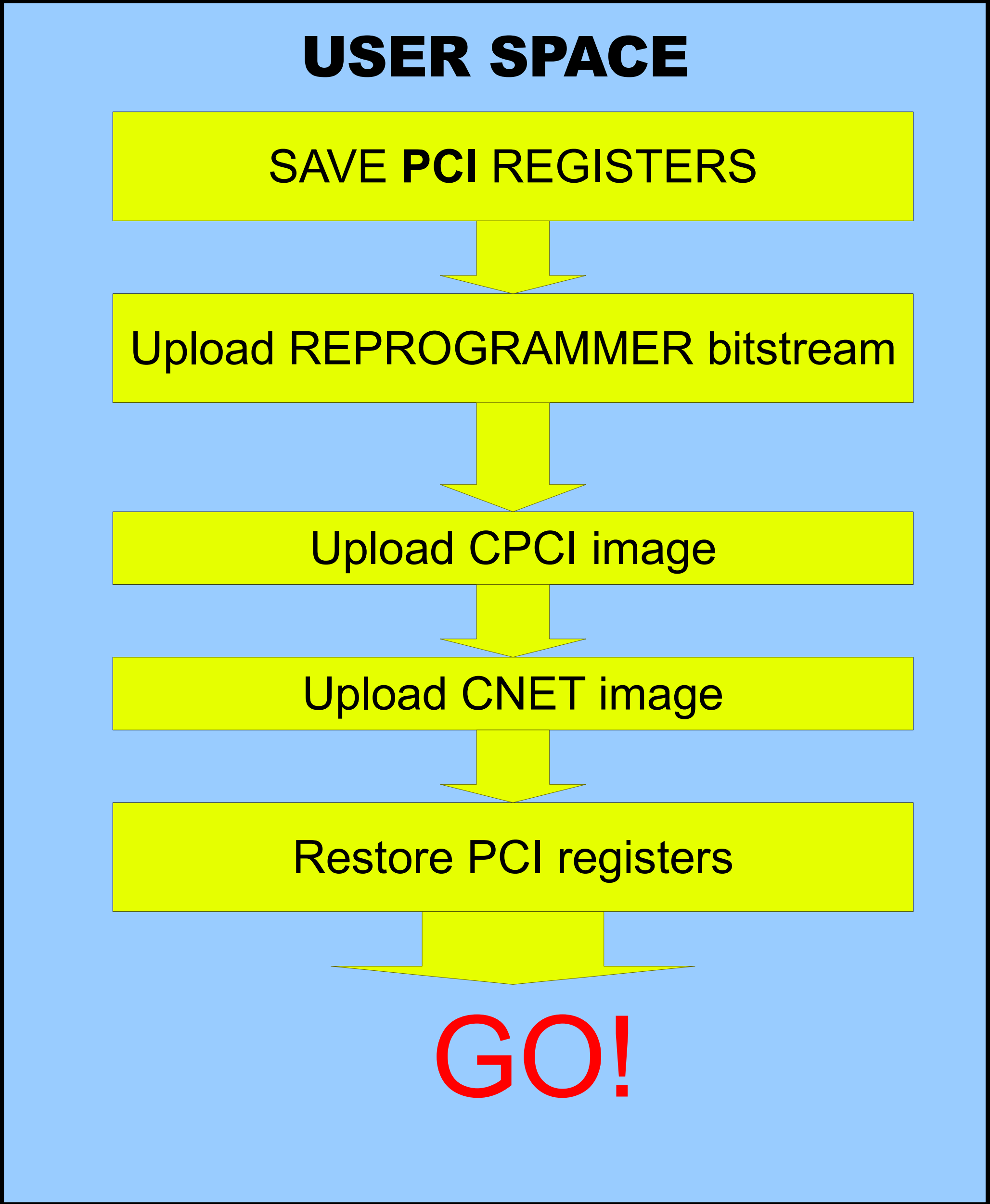


Data is copied into the single linear buffer prepared by the host

Please note we don't have a direct access to hardware rings and descriptors

NetFPGA programming

NetFPGA programming (Linux)



USER SPACE

SAVE **PCI** REGISTERS

Upload REPROGRAMMER bitstream

Upload CPCI image

Upload CNET image

Restore PCI registers

GO!

Current utilities in Linux

- Read card's memory chunk
- Write to the card
- Program the card
- Reprogram CPCI

..utilities:

- `readreg`
- `writereg`
- `nf2_download`
- `cpci_reprogram.pl`

NetFPGA Linux driver

In Linux driver:

PCI registers saved from userspace :-(

Additional dependency on user-space tools

Linux NetFPGA driver is 'monolithic'

CPCI/CNET programming is tied to Ethernet port structures

Use of ifnet-specific ioctl()s

Other problems

See `netfpga-devel@` mailing list

FreeBSD driver design

Driver is BSD-licensed

Driver has two parts

- Programming (/dev/...) interface
- Ethernet (``ifconfig'') layer

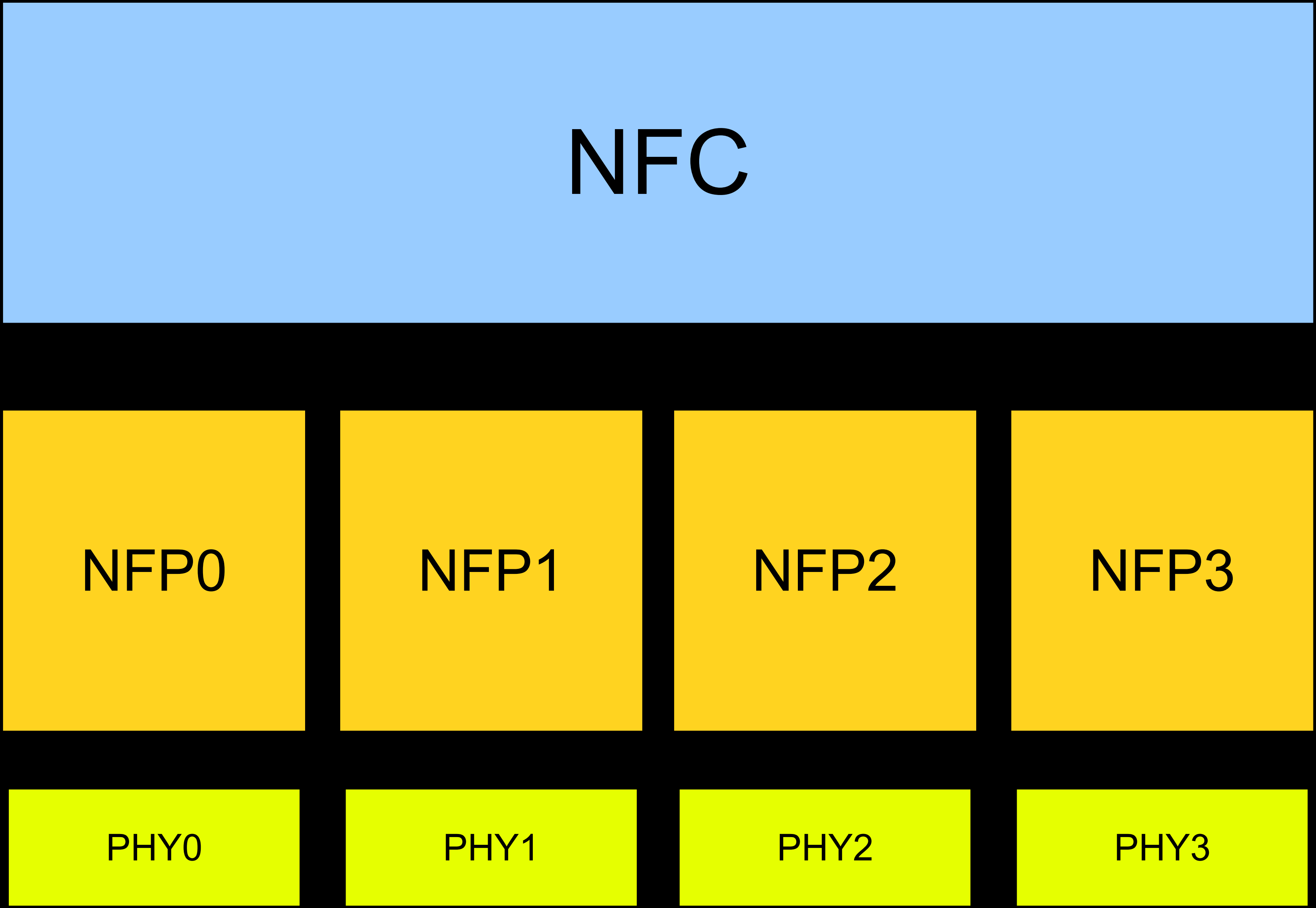
Card itself appears to NEWBUS as

``NetFPGA controller''

Later called (NFC)

It's up to the controller to export CPCI/CNET interface

Each NFC has 4 Ethernet ports, later called ``NetFPGA ports'' (NFP)

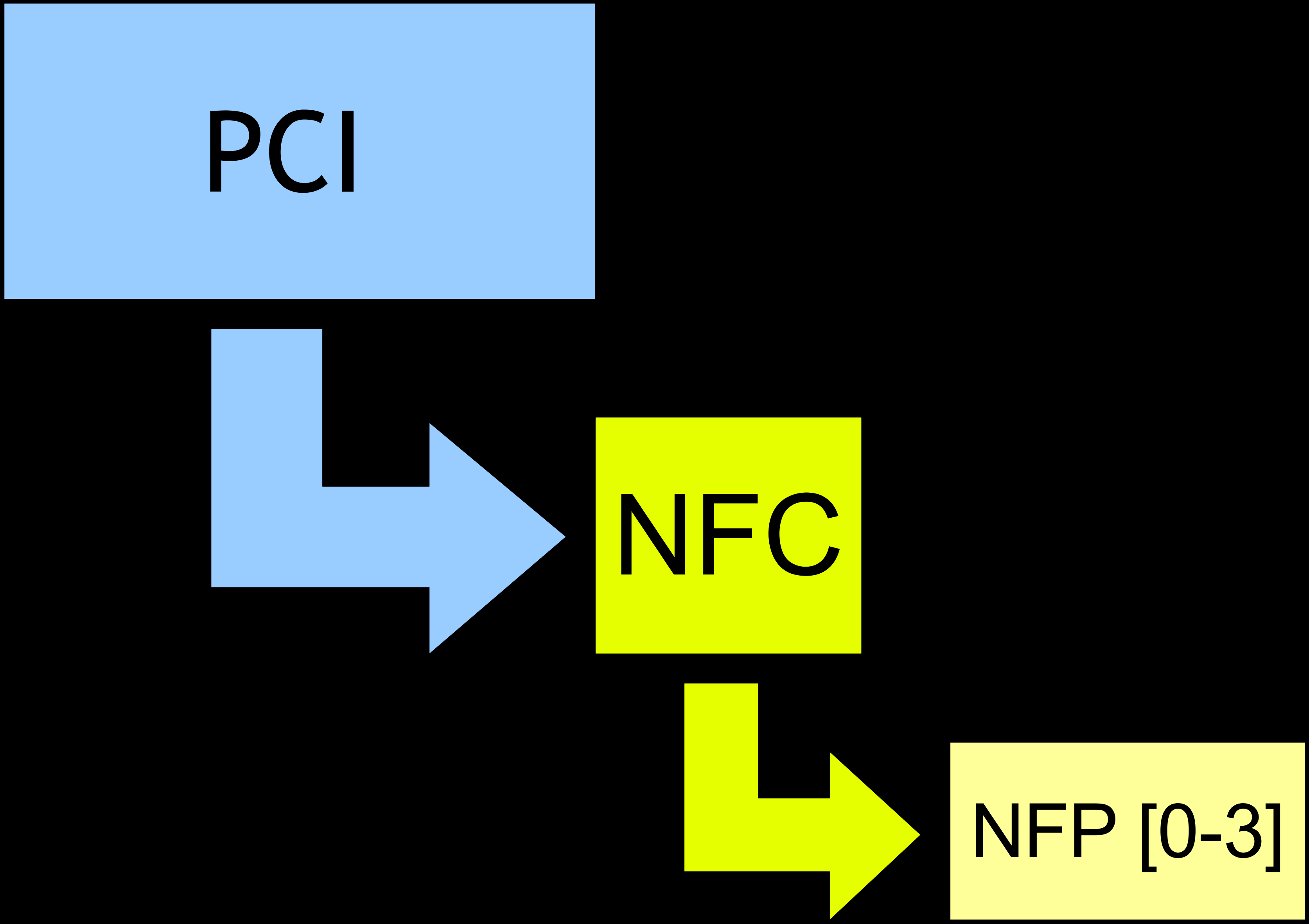


NFC

NFP0 NFP1 NFP2 NFP3

PHY0 PHY1 PHY2 PHY3

Appearance of NetFPGA in the FreeBSD (`devinfo -rv`)



NFC

Programming interface appears as separate device

`/dev/netfpga[0-9]+`

You just send ioctl() commands there

`ioctl()` handler detects the fact of ``Programming”

It saves registers and restores them once device is being closed

In FreeBSD, NFC also exports string with register offsets via sysctl() interface

The plan is to fight with quite dynamic nature of HDL specifications and stay away from ABI breakages

```
nf_read_reg(”REGISTER”)
```

instead of

```
nf_read_reg(REGISTER)
```

It would be nice to be able to enable Ethernet interface layer only when there's Ethernet support in a bitstream

Any ideas for „Ethernet” detection?

NFPs

NFPs are handled by separate module (driver)

NFPs are started from NFC attach routine with:

```
bus_generic_attach()
```

Every NFP is visible to the system as Ethernet interface

```
nf2c0: flags=8843<UP,BROADCAST,RUNNING,
      SIMPLEX,MULTICAST> metric 0 mtu 1500
      options=28<VLAN_MTU,JUMBO_MTU> ether
  ▪   00:6e:66:32:63:30
      inet 10.0.0.1 netmask 0xff000000
      broadcast  10.255.255.255 media:
      Ethernet autoselect (none)
```

NetFPGA programming in FreeBSD

One program – nfutil(8) deals with CPCI reprogramming, CNET programming and register access nfutil(8) is built on top of the *libnetfpga*, library for NetFPGA operations nfutil(8) will probably have to provide some `argv[0]` tricks in order to mimic Linux utilities

Right now nfutil(8) has hierarchical commands:

```
nfutil image write <file>

nfutil cpci write <file>

nfutil reg read <register>
```

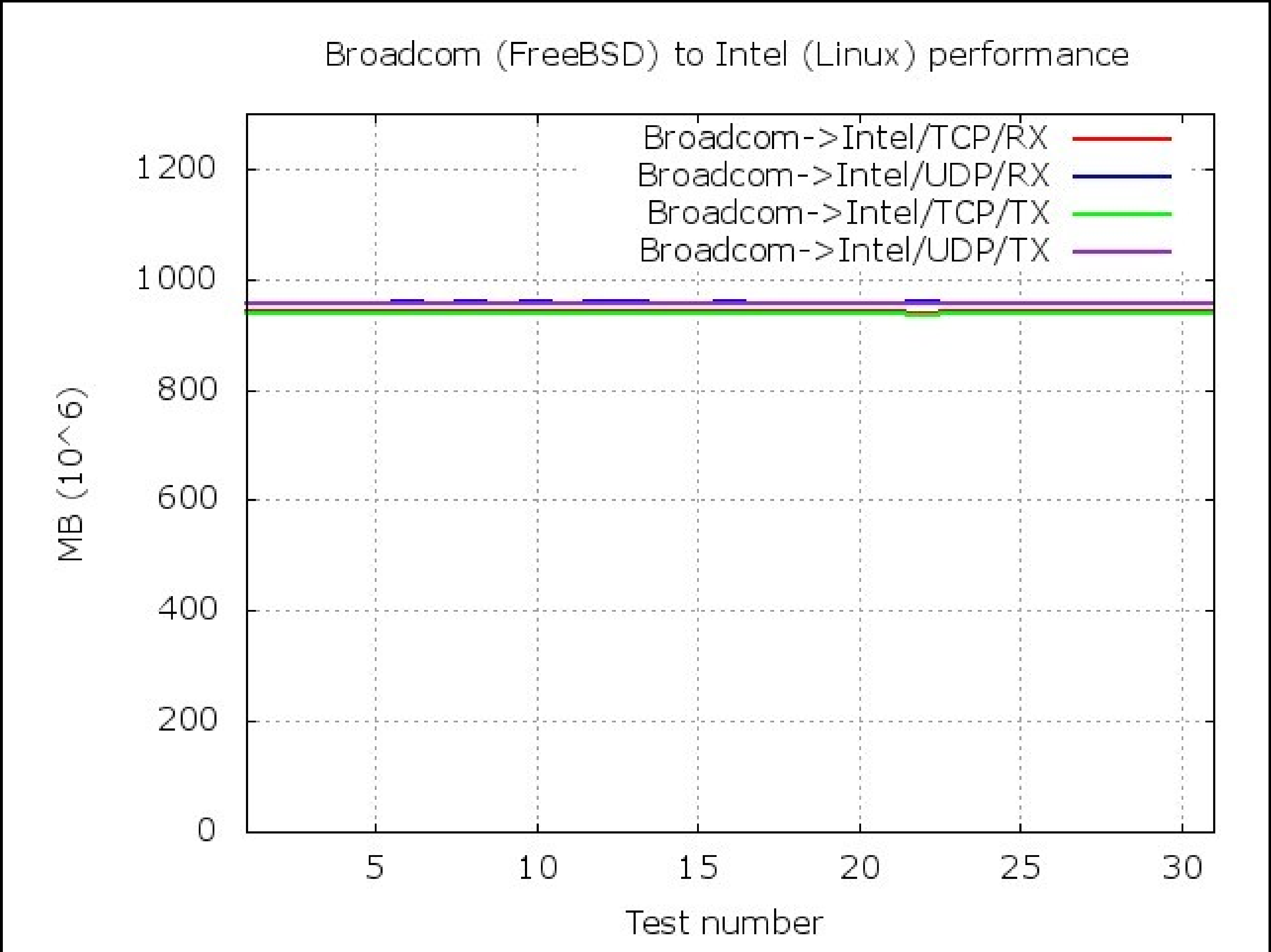
There's is libcla, library for hierarchical command handling

`<file>` arguments point to bitstream files

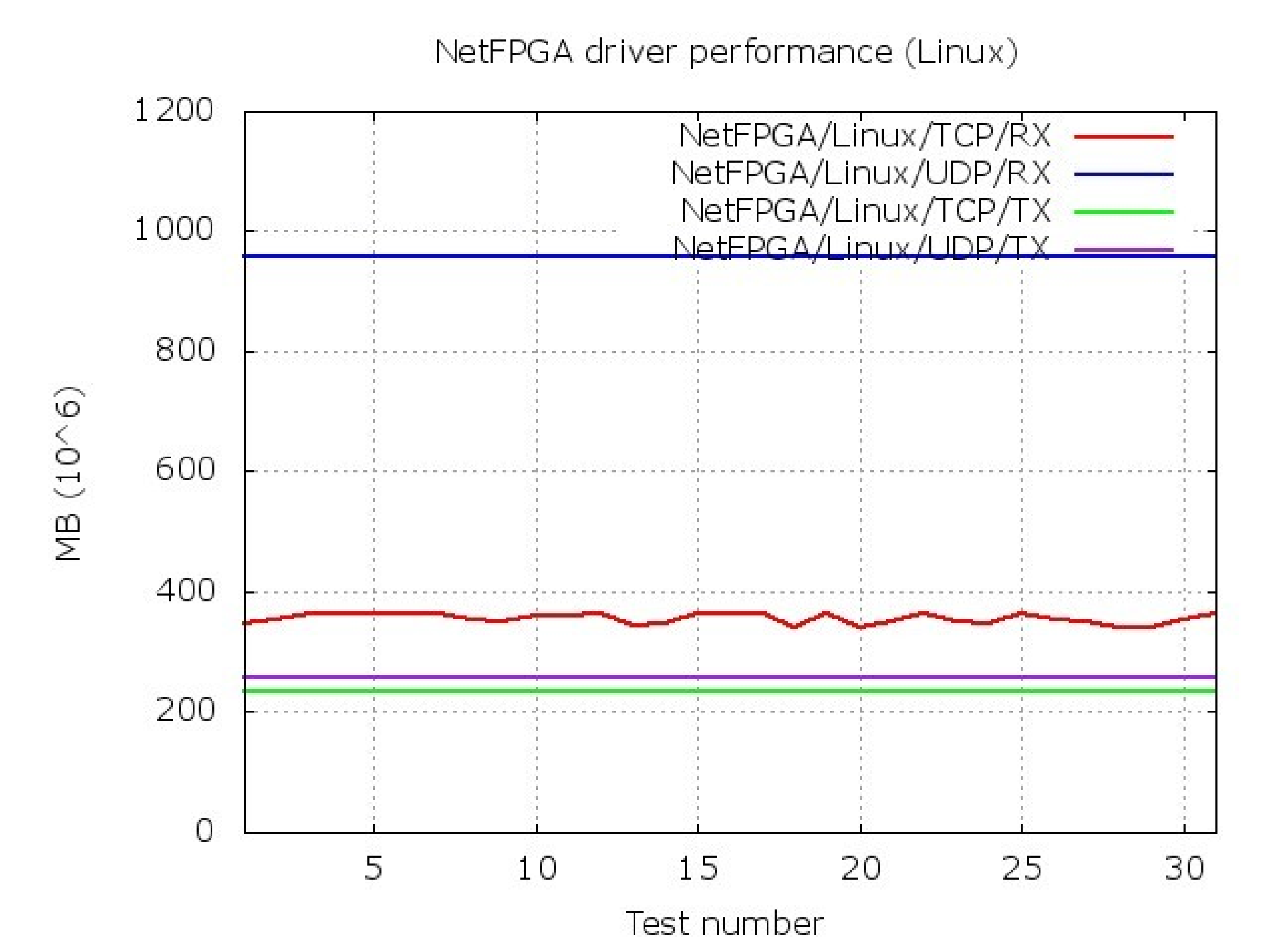
(there's a libxbf for bitstream handling as well)

Some numbers

Non-NetFPGA performance



NetFPGA performance (Linux)



NetFPGA performance (FreeBSD)

ENOTYET :-(

Problems

Minimum DMA transfer is 60 bytes

I could get handling of this limitation wrong

I could get software mitigation of typical ring/descriptors primitives wrong

(There's no access to hardware-assisted RX/TX of packets)

Linux driver has more than one buffer for transmission but only one for receiving...

Reset of the PHY chip seems to take some ``nondeterministic amount of time”

No Broadcom chip specs :-(

I now got some support about MAC/PHY/CNET reset order

No hardware to test :-(

Problems with new register system made it impossible to work with latest NetFPGA release

I used 1.2.5 release

RFC

NetFPGA comes with broken firmware

New firmware has to be uploaded just after computer boot

NetFPGA firmware

- Licensing
  - NetFPGA code and Verilog files: BSD license
  - MAC IP Core from Xilinx:
    - Should be OK to redistribute
- CPCI reprogramming could happen as a part of driver attach routine:
  - CPCI image is relatively small

Summary

FreeBSD (very) experimental support is here...

- Card is detected and can be programmed
- Programming utilities are here
- Basic network functionality works
  - Ping program is able to transmit/receive packets
  - Basic benchmarking works

Future

Plans

- Bring NetFPGA support to the FreeBSD source code base
    - Work on stability
    - Work on PERFORMANCE
        - Being better than Linux would be nice!

FreeBSD/NetFPGA out-of-box?

Card driver (netfpga.ko)

Port driver (if_nf.ko)

How do we handle unregistered PCI Vendor and PCI Device numbers?

Other plans?

Getting an access to the FPGA hardware at home?

2 x Future

Support for NetFPGA-ng

(planned release: somewhere in 2010)

NetFPGA-NG:

4x10Gbit SFP+

Biggest Virtex 5 FPGA processor

Maybe improved interrupt policy?

Other FPGA-based cards

- CESNET/INVEA-TECH cards
  - Virtex 5
  - 2 x 10GBit (4 x 10GBit in preparation)
  - Industry standard
  - not sure about PCI-e interface

Getting support for more FPGA-based accelerators

Special THANKS…

- Pekka Nikander (Ericsson)
- Jussi Kangasharju (HIIT)

Code I talked about:

http://people.freebsd.org/~wkoszek/netfpga

`netfpga-devel@` mailing list has this code as well

This presentation will be available on:

http://FreeBSD.czest.pl/~wkoszek/netfpga/

and

http://people.FreeBSD.org/~wkoszek/netfpga/

Q/A

The End

Wojciech A. Koszek
wkoszek@FreeBSD.org

2009.09.17