Internet on Chip

Gigabit Ethernet and Programmable Hardware

Dr. Johannes Wolkerstorfer

www.xFace.at

Realraum Graz, January 26th 2010

Motivation

How does Internet data look like?

- On the wire
- In the chip

How can packets be processed?

- What is the advantage of hardware

Real world example: PRISM project

- Privacy for Internet monitoring

PRISM application

PRIvacy-aware Secure Monitoring

- http://www.fp7-prism.eu/
- European research project
  - Austrian partner: FTW (Forschungs-zentrum Telekommunikation Wien

Goals

- Monitor network traffic at ISP
  - Detection of attacks
- Maintain privacy of user information (law)
  - e.g. IP address

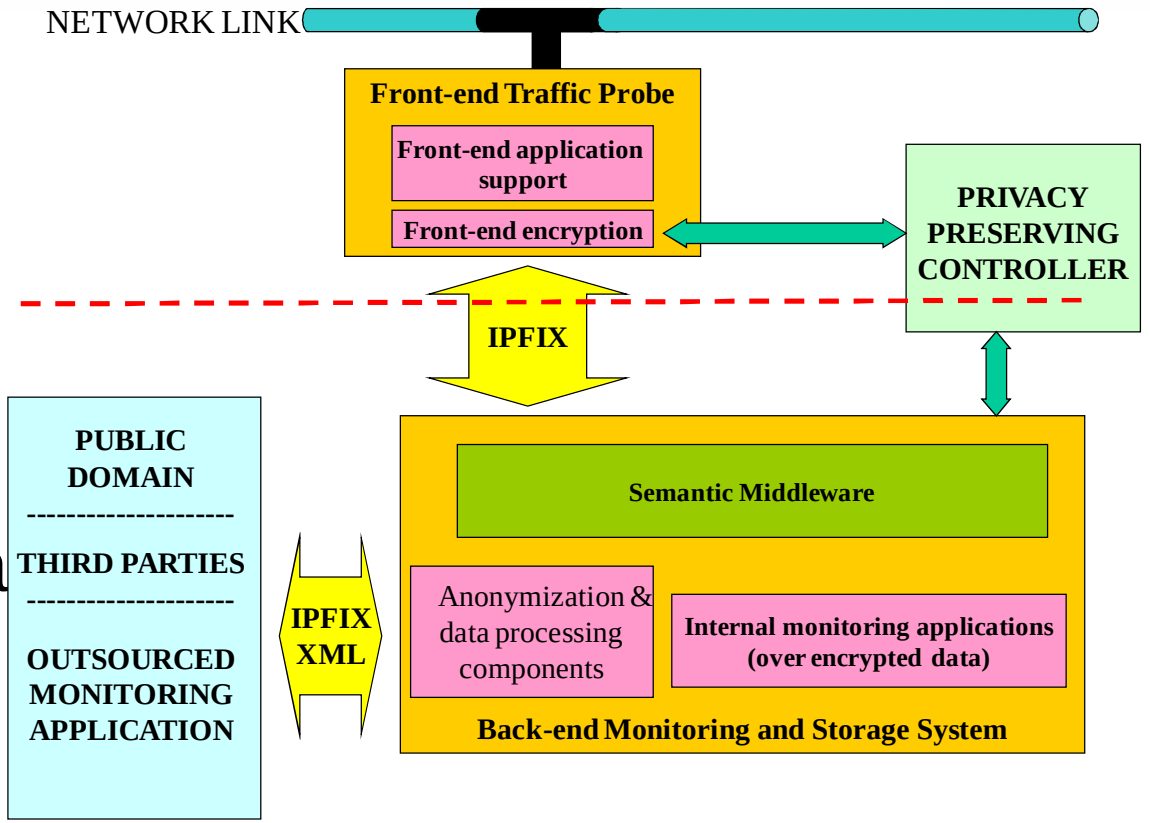
PRISM architecture

Frontend

- Monitoring of Internet traffic

Backend

- Storage of encrypted data
- interface to monitoring applications



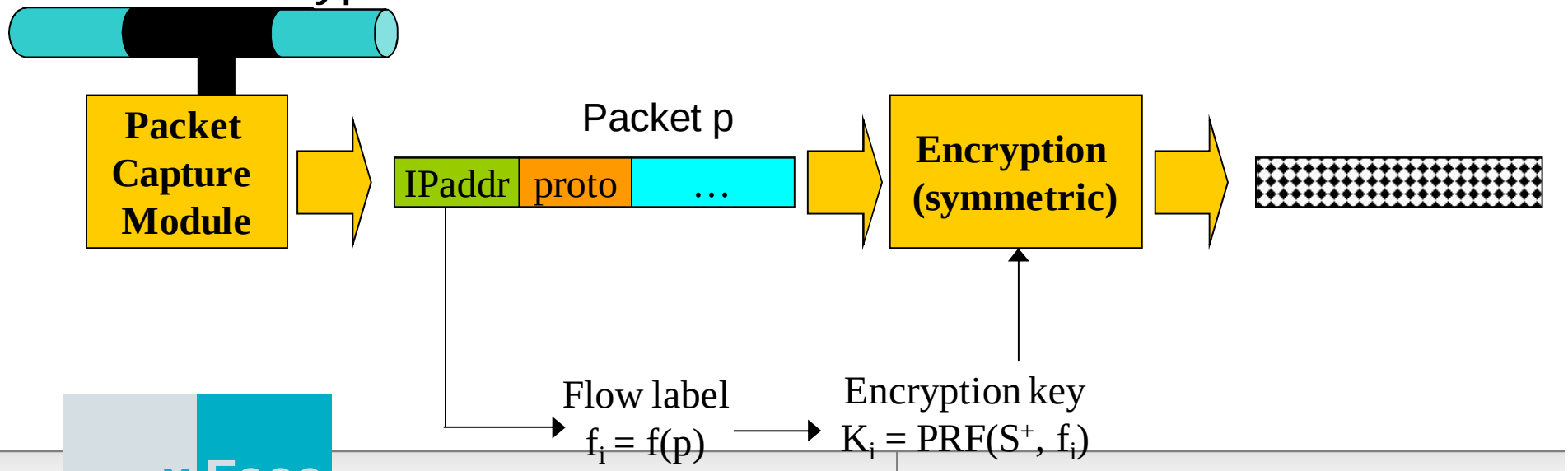
PRISM frontend

Functionality

- Flow extraction
  - classification of packets
    - normal | flow | anomaly
- Encryption

Requirements

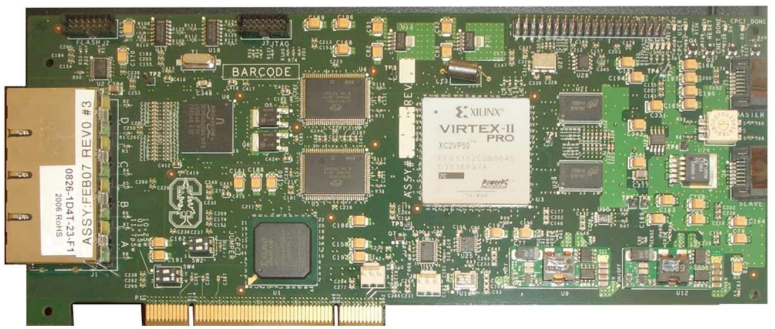
- Throughput: 1 Gbps
- Per-packet basis
  - stateless



NetFPGA project

NetFPGA

- FPGA board with
  4 Gigabit Ethernet ports
- Opensource HW & SW
  - provided by Stanford Univ
- Applications
  - Router, switch, traffic gen.
  - Network security
  - Intrusion detection



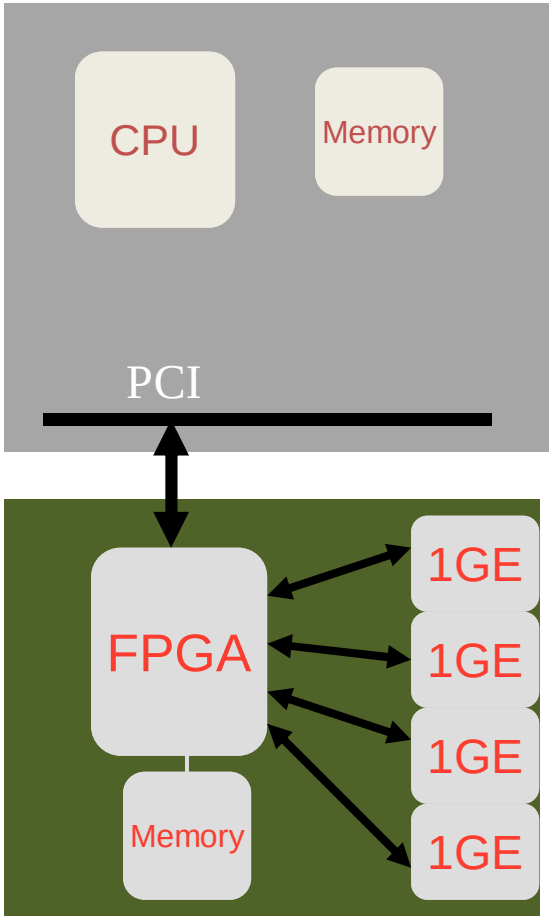
NetFPGA Board

NetFPGA system

Host PC
- Linux PC
- Kernel-mode driver
  - 4 additional network interfaces

PCI interface
- Network traffic
- Management of NetFPGA

PC with NetFPGA



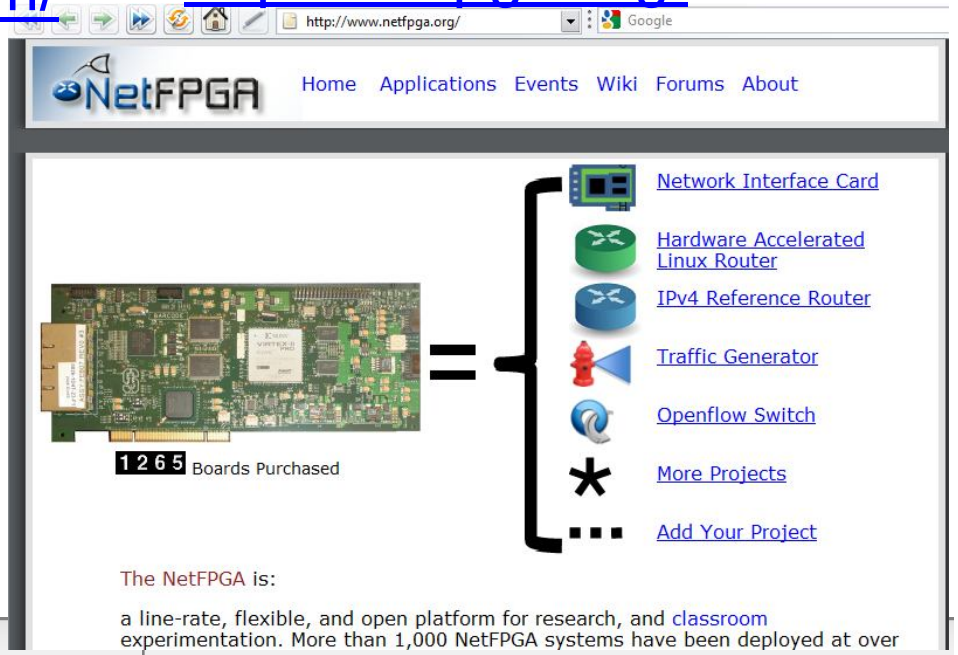
NetFPGA Where to get?

Hardware

- From Digilent
- 600 US$ for research inst.
- http://www.digilentinc.com/

Software / HDL

- From Stanford University
- http://netfpga.org/



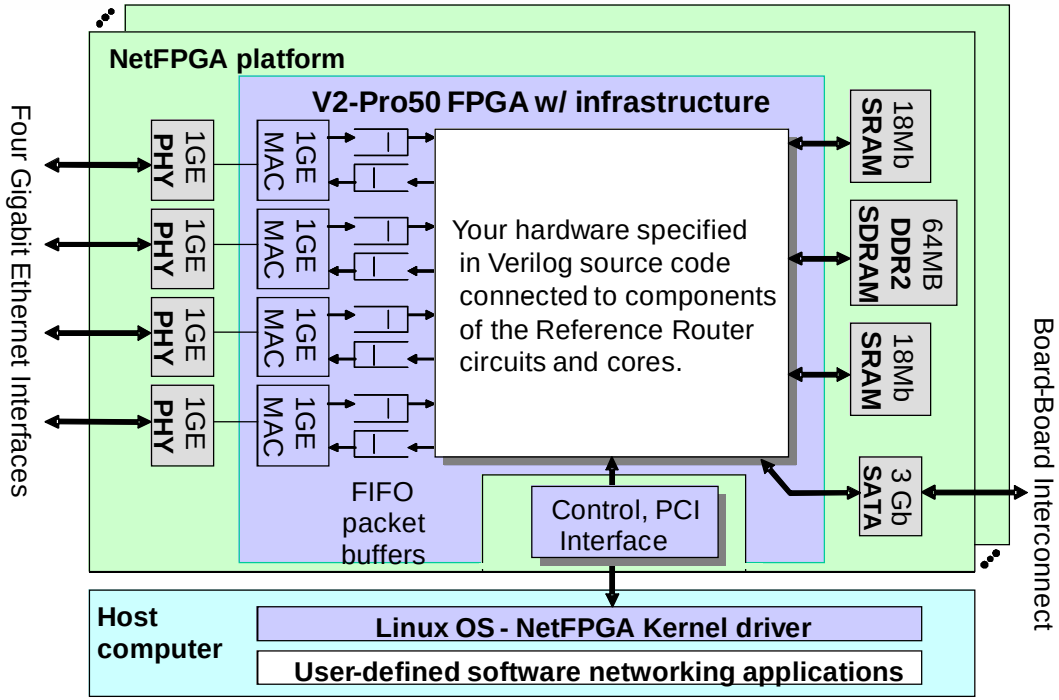
NetFPGA architecture

NetFPGA components

- Xilinx Virtex-2 Pro FPGA for User Logic
- Xilinx Spartan for PCI Host Interface
- Cypress: 2 * 2.25 MB ZBT SRAM
- Micron: 64MB DDR2 DRAM
- Broadcom: 4 Gigabit Ethernet PHYs

FPGA (Xilinx Virtex-II Pro)

- Accommodates own project
- FPGA configured via PCI bus



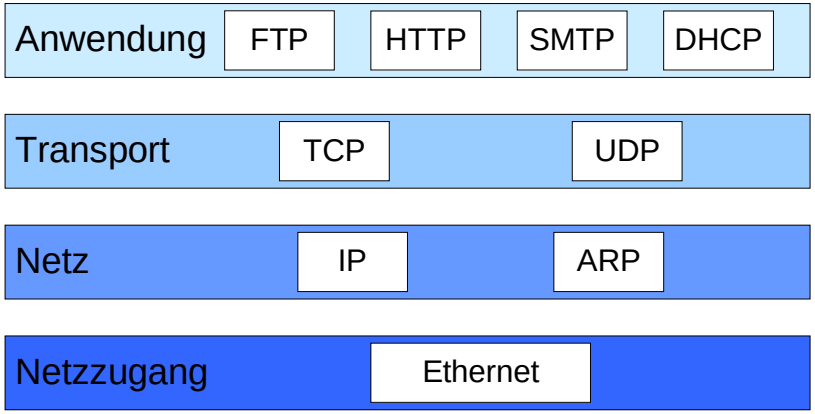
Gigabit Ethernet

Packet switching network

OSI Schichtmodell

- TCP/IP: Transmission Control Protocol
  - Transport-Layer (Layer 4)
  - Gesicherte Datenverbindung
- IP: Internet Protocol
  - Network-Layer (Layer 3)
  - Datenvermittlung im Internet
- Ethernet (IEEE 802.3)
  - Data-Link Layer (OSI Layer 2)
    - Sicherung
  - Physical Layer (OSI Layer 1)
    - Bitübertragung

Interfaces: GMII for PHY



Ethernet frames

Protocol / Packet / Frame

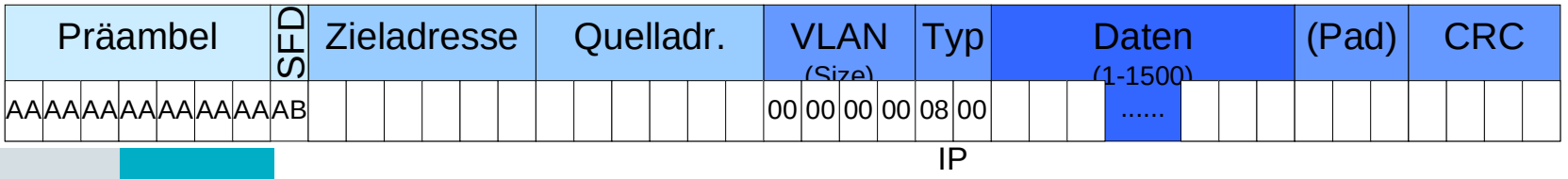
- Protocol: standard for communication
- Packet: Part of data
- Frame: Packet in transmission

Ethernet Frame

- Preamble
  - 56 Bit: alternating 0 und 1
  - For Synchronization
- SFD
  - Start of Frame Delimiter

Ethernet Frame (ctd)

- Source and target address
  - 48-Bit MAC-Address
  - Unique for every NIC
    - Manufaqcturer pefix
  - Do not mix up with IP address
- Paket-Typ
  - z.B. 0x0800: IP
- Prüfsumme
  - CRC



Ethernet MAC

- FIFO: First in – First out
  - Buffering of network data
- MAC: Media access control
  - Controls access to PHY
  - Framing of packets
  - CRC Checksum
- PHY: Physical layer
  - Electric standard
  - Encoding of signals
  - 1000 Base-T GbE: 4 twisted pairs

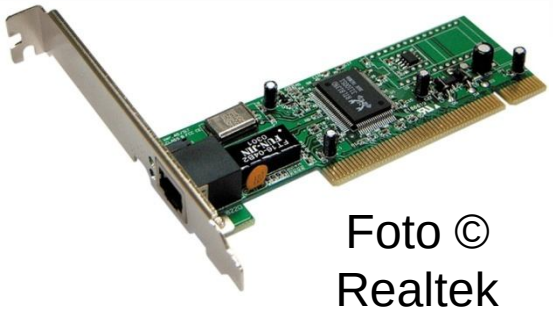
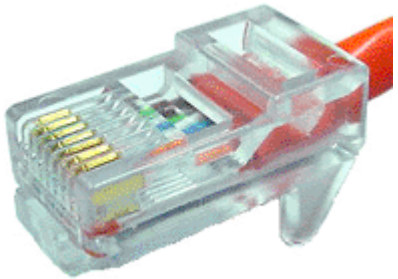
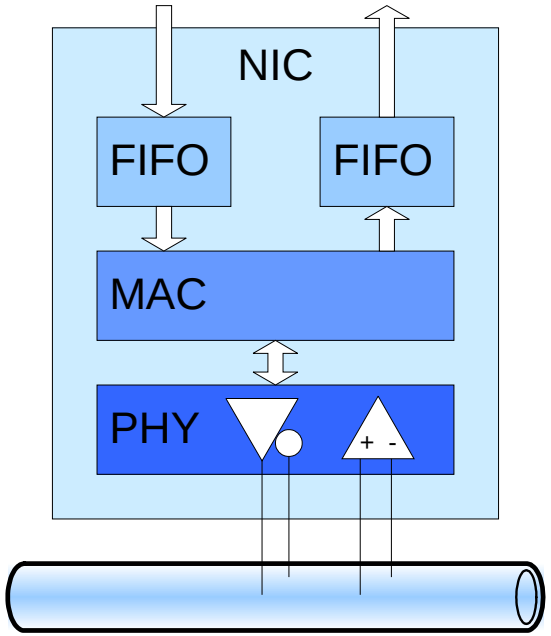
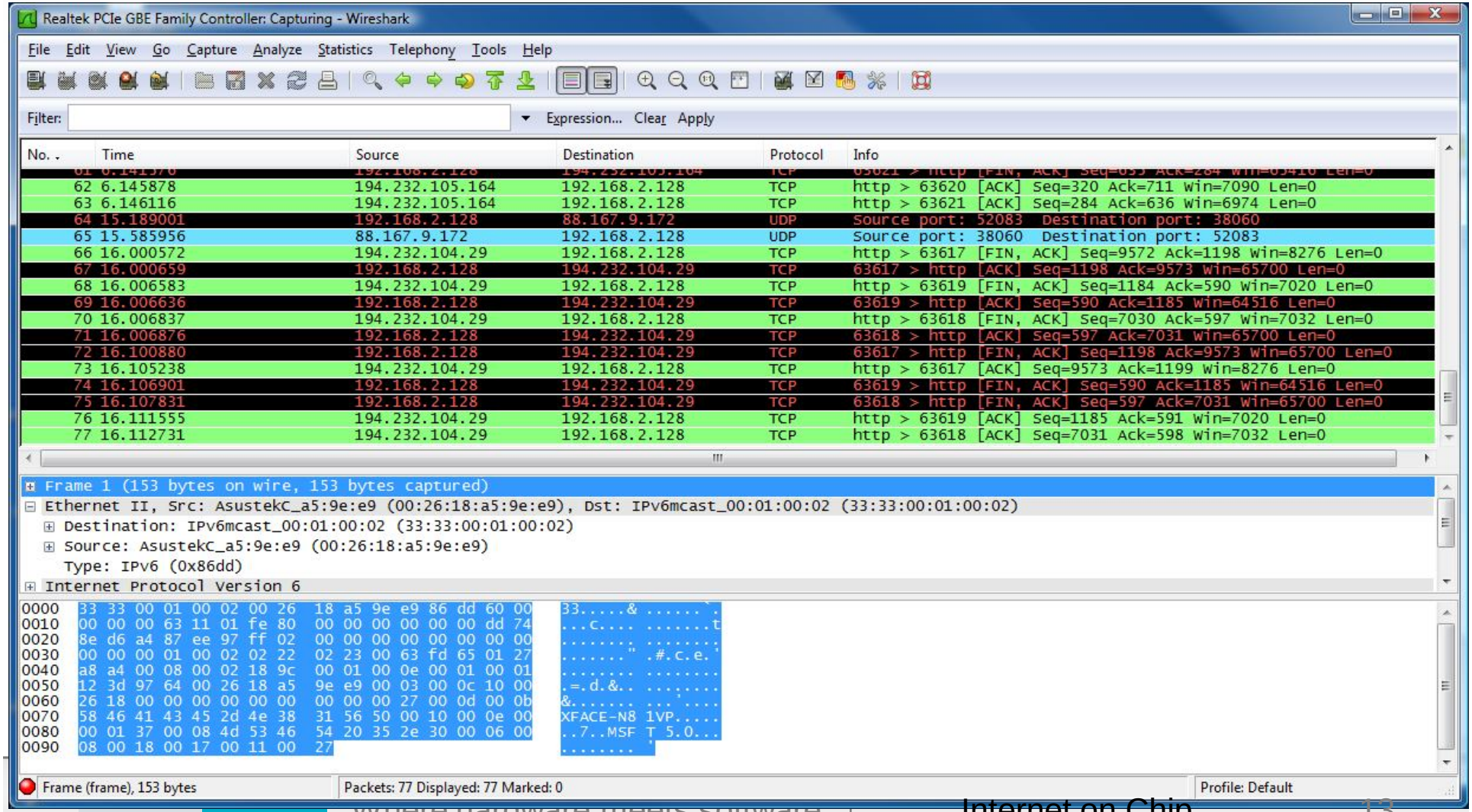


Foto © Realtek





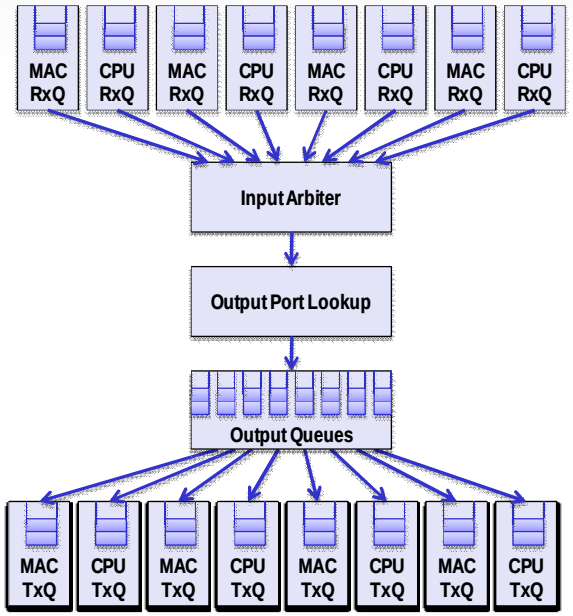
Ethernet packets: Wireshark

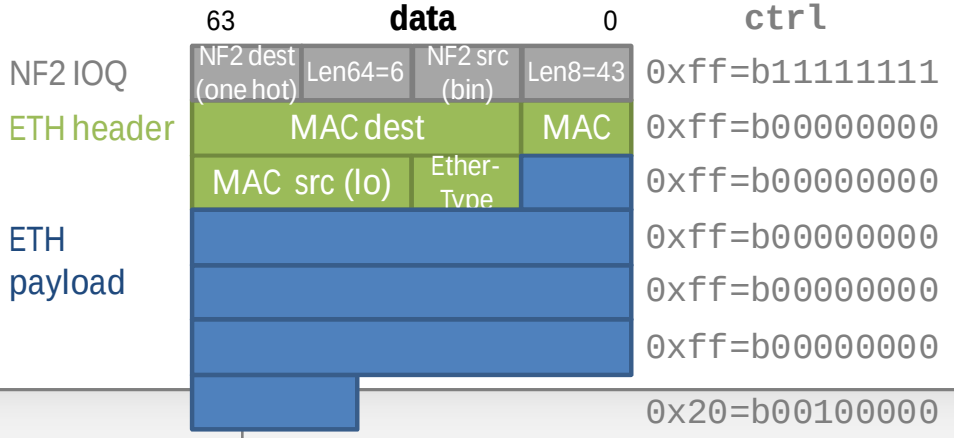


Ethernet packets in NetFPGA

GMII: Gigabit media independent interface
- connects to the PHY
- 8-bit interface; 125 MHz

user_data_path interface
- 64-bit interface, 125 MHz
- FIFO handshake

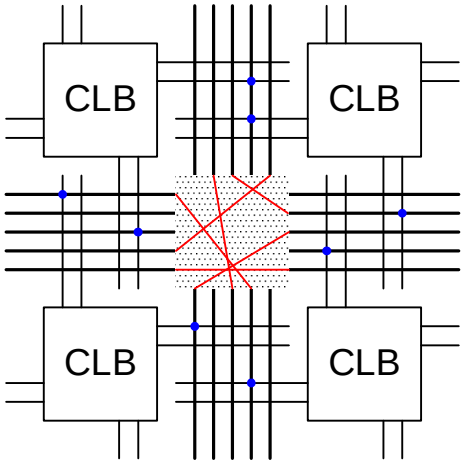




FPGA

Field programmable gate array (FPGA)

- Manufactured digital hardware
- Configurable hardware resources (CLB)
  - Lookup-table based logic
  - Flipflop for data storage
- Memory resources
  - block ram, distributed RAM
- Programmable interconnect



FPGA: Xilinx Virtex-II Pro

NetFPGA

– Xilinx Virtex-II Pro 50

Device[1]	RocketIO Transceiver Blocks	PowerPC Processor Blocks	Logic Cells[2]	CLB (1 = 4 slices = max 128 bits)		18 X 18 Bit Multiplier Blocks	Block SelectRAM+		DCMs	Maximum User I/O Pads
				Slices	Max Distr RAM (Kb)		18 Kb Blocks	Max Block RAM (Kb)		
XC2VP2	4	0	3,168	1,408	44	12	12	216	4	204
XC2VP4	4	1	6,768	3,008	94	28	28	504	4	348
XC2VP7	8	1	11,088	4,928	154	44	44	792	4	396
XC2VP20	8	2	20,880	9,280	290	88	88	1,584	8	564
XC2VPX20	8[4]	1	22,032	9,792	306	88	88	1,584	8	552
XC2VP30	8	2	30,816	13,696	428	136	136	2,448	8	644
XC2VP40	0[3], 8, or 12	2	43,632	19,392	606	192	192	3,456	8	804
XC2VP50	0[3] or 16	2	53,136	23,616	738	232	232	4,176	8	852
XC2VP70	16 or 20	2	74,448	33,088	1,034	328	328	5,904	8	996
XC2VPX70	20[4]	2	74,448	33,088	1,034	308	308	5,544	8	992
XC2VP100	0[3] or 20	2	99,216	44,096	1,378	444	444	7,992	12	1,164

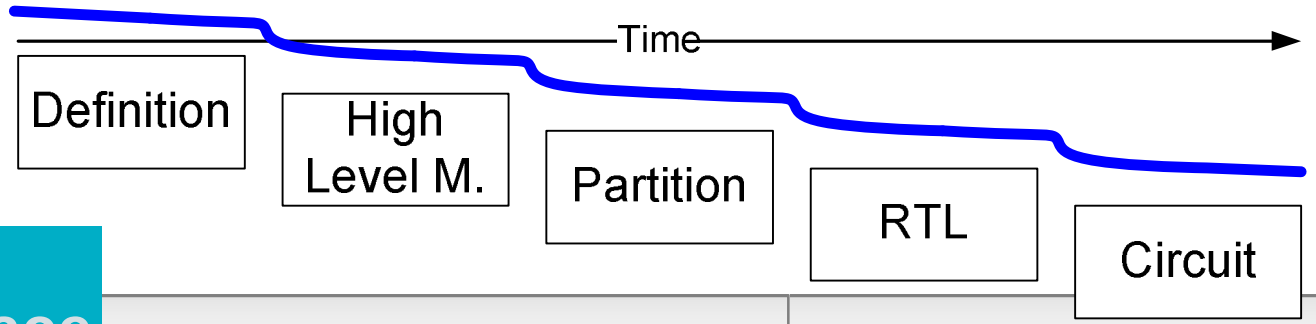
Design methodology

Top down design

- Informal specification
- High-level model: C/C++, Java
- HDL simulation
- Synthesis, place & route

No hardware debugging

- Sufficient testing of HDL model



Design tools: high level

High-level model:

C/C++

- Functional model
- Performance not important
- Evaluation of algorithms

```
108  void enckeyshare::encryptPay
109  //dbg   cout << "Encrypting
         payloadByte_t   dil[ethe
         unsigned int dilLen = di
         // Output IV
         dout.addPayload(_keyStor
         printHex(_keyStore[IV],
         // Padding of input data
         din.getPayloadBytes(dil,
117  //dbg   printHex(dil, "  loa
118      unsigned int blocks = _a
119      printHex(dil, "enc-pad",
120      // AES-128 CBC encryptio
121      _aesCbc.StartEncryption(
122      _aesCbc.Encrypt(dil, dol
123      _aesCbc.getIV(_keyStore[
124      dout.addPayload(dol, blo
125  }
```

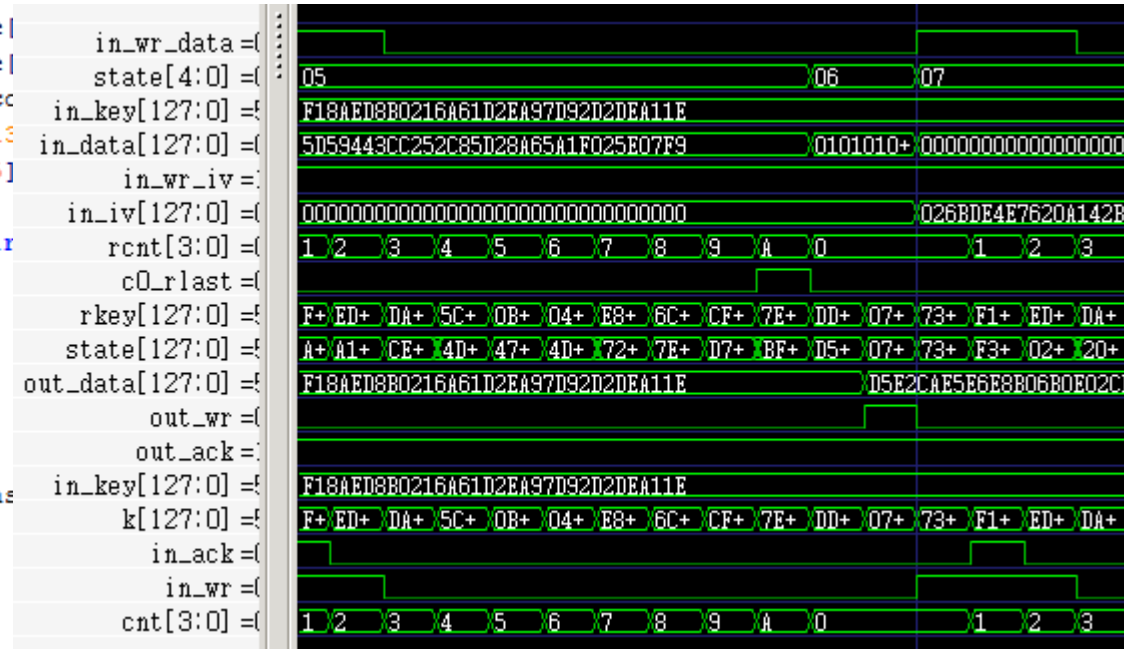
```
------------------------------------------------------
---
---   Passing Ethernet packet through enckeyshare
---

Status of enckeyshare
EncKeyshare Unit
enc-S* = KEY1S= 0x00000000000000000000000000000000
enc-S**= KEY2S= 0x00000000000000000000000000000000
enc-AES IV =    0x00000000000000000000000000000000
enc-ADP RAND =  0x00000000000000000000000000000000
enc-COEFFS  =   0x0
enc-XCOORD  =   0x0

--->    Input packet
eth-MACdest =   01:02:03:04:05:06
eth-MACsrc =    aa:bb:cc:dd:ee:ff
eth-ETHtype =   0800                    (IP)
eth-Payload =   4500002eb3fe000080110540c0a8002c
          c0a8000404000400001a2de800010203
          0405060708090a0b0c0d0e0f1011
eth-CRC32 = ab187387                (Verified)

enckeyshare processes Packet packet:

    Output packet   --->
eth-MACdest =   01:02:03:04:05:06
eth-MACsrc =    aa:bb:cc:dd:ee:ff
eth-ETHtype =   0800                    (IP)
eth-Payload =   4500002eb3fe000080110540c0a8002c
          c0a8000404000400001a2de800010203
          0405060708090a0b0c0d0e0f1011
eth-CRC32 = ab187387                (Verified)

Passed.
```

Design tools: Verilog

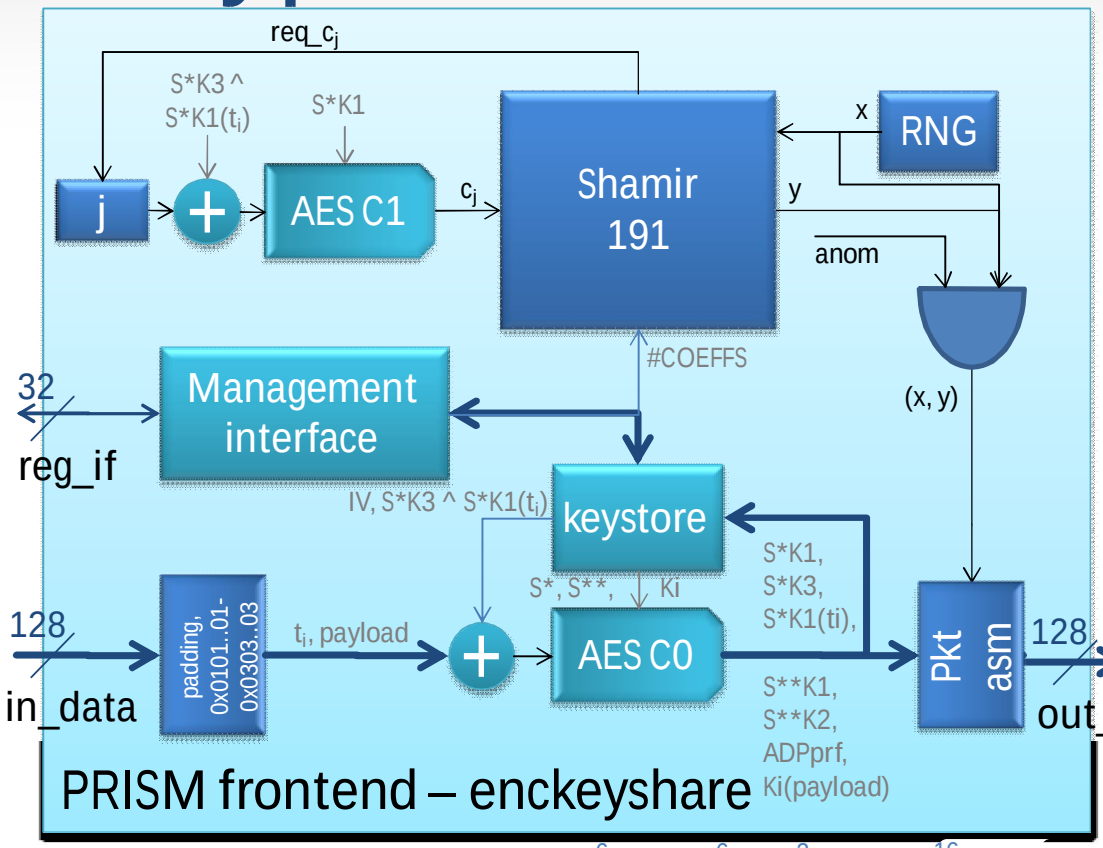
Modelling of digital hardware

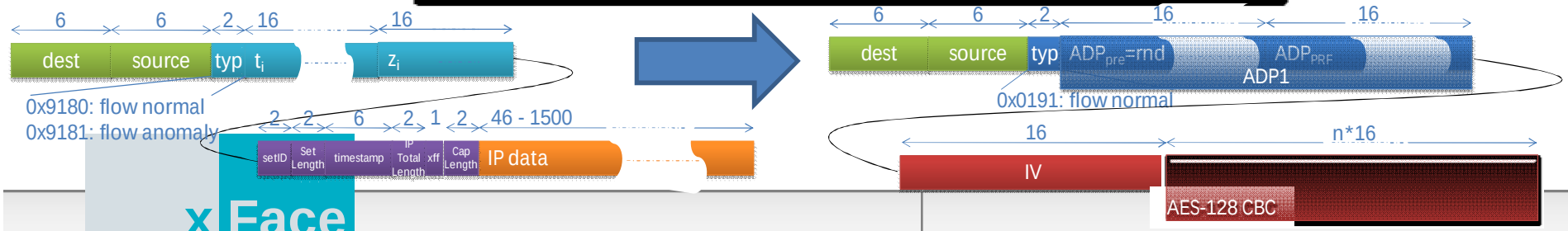
– Hardware description language (HDL): Verilog

```
47
48  assign {sbyte[15],sbyte[14],sbyte[13],sbyte[
49         sbyte[ 7],sbyte[ 6],sbyte[ 5],sbyte[
50         {mixcol[3],mixcol[2],mixcol[1],mixcc
51  assign state = {sbyte[15],sbyte[14],sbyte[13
52                 sbyte[ 7],sbyte[ 6],sbyte[ 5]
53
54  always @(posedge clk or posedge reset) begin
55    if (reset) begin
56      rlastFF <= 0;
57      out_wrFF <= 0;
58    end else begin // posedge clk
59      if (rlastFF) dataout <= state;
60      rlastFF <= rlast;
61      out_wrFF <= out_wrFF && ~out_ack || rlas
62    end // posedge clk
63  end // always
```

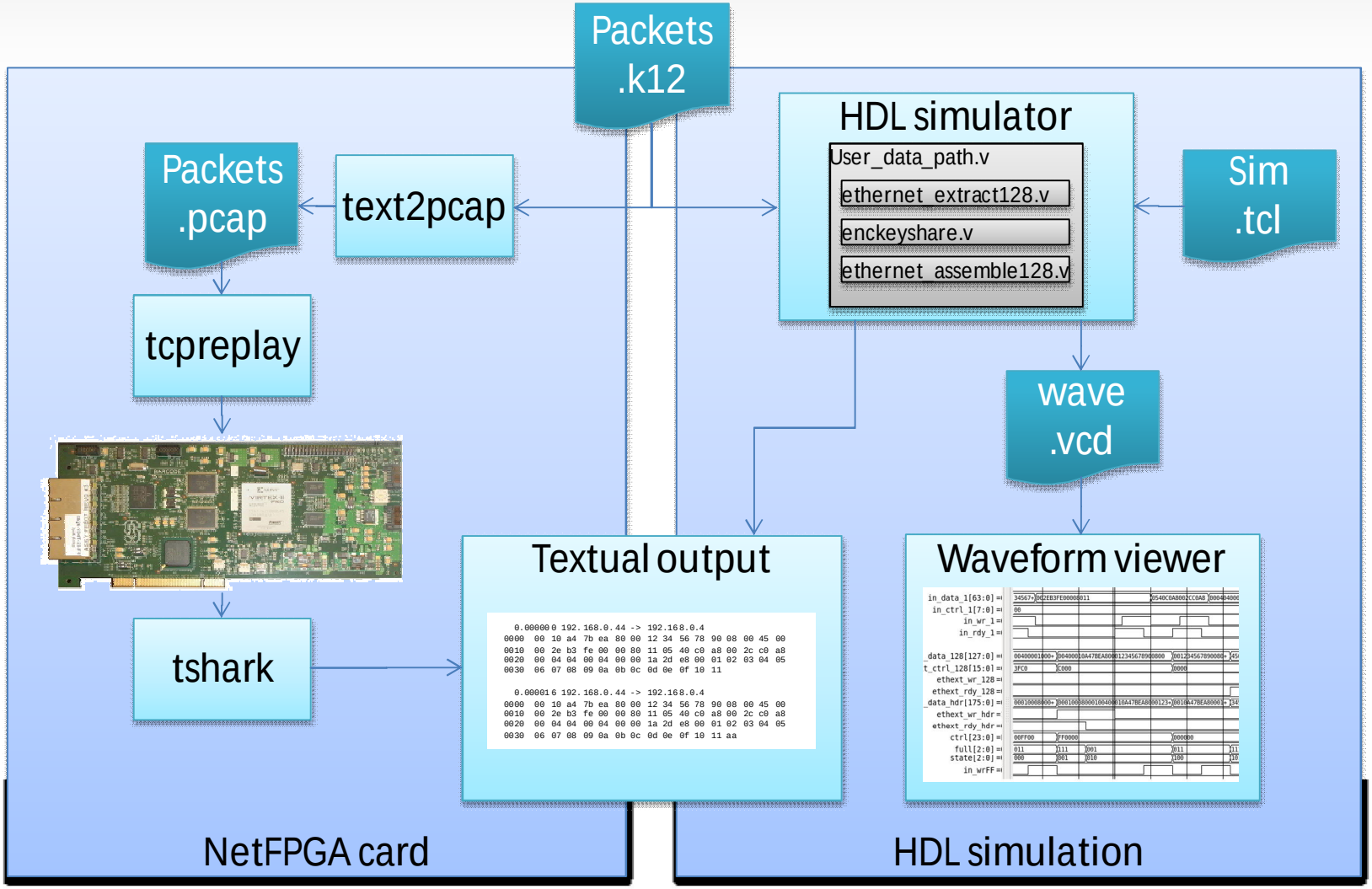


PRISM: Crypto hardware





Design tools: Verification



Conclusions

Gigabit Ethernet and digital hardware

- Can be done with FPGAs
- Hardware and Software nearly affordable

Results

- 1 Gigabit Ethernet encryption: AES-128 CBC
- Computation of Shamir keyshares

Feature	Value	Remark
BRAM	18 (of 232: 7%)	two AES cores (c0, c1) and one round-key generator
slices	3749 (of 23616: 15%)	mostly caused by Shamir unit (191*16 multiplier)
f_max	165.4 @ MHz	XST synthesis result
Throughput	1 GBit/s	maximum throughput @ f_clk=125 MHz
Latency	75 cycles (600 ns)	delay caused by enckeyshare; required interframe gap