

Layered Protocol Wrappers for Internet Packet Processing in Reconfigurable Hardware

Florian Braun, John Lockwood, Marcel Waldvogel

Lockwood@arl.wustl.edu
Assistant Professor

Washington University
Department of Computer Science
Applied Research Lab
1 Brookings Drive
Saint Louis, MO 63130

<http://www.arl.wustl.edu/arl/projects/fpx/>
<http://www.hoti.org/>

Supported by: NSF ANI-0096052 and Xilinx Corp.



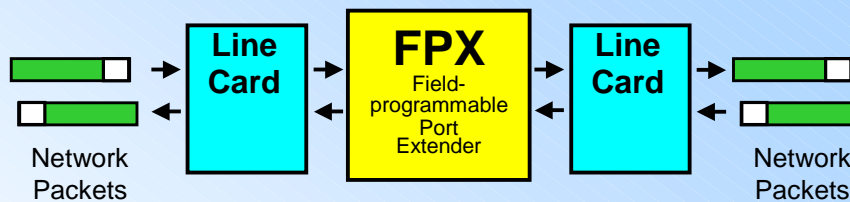
Outline

- **Hardware-Accelerated Packet Processing**
 - Field Programmable Port Extender (FPX)
- **Layered Internet Protocol Wrappers**
 - Library of Synthesizable components
- **Application Example**
 - Simple, Line-speed Encryption module in wrapper
- **Implementation Results**
 - Post-synthesis chip utilization, throughput, latency
- **Conclusions**



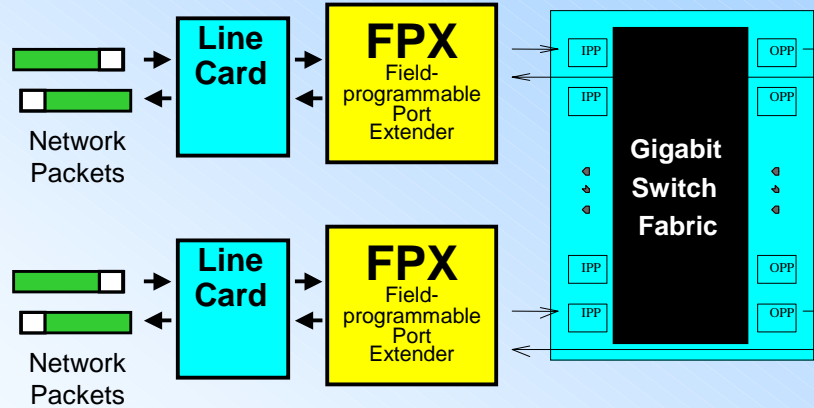
Hardware-Accelerated Packet Processing

Configuration of FPX Packet Processor



- Packet processing hardware performs:
 - Packet classification
 - Packet forwarding
 - Address Translation
 - Data modification
 - Packet buffering
 - Active Networking (Application-level data processing)

Configuration of FPX in Internet Router



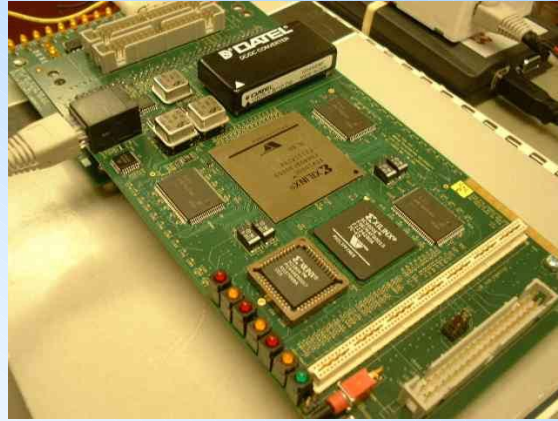
- Additionally, Router interface performs:
 - Internet route lookup
 - Traffic policing and shaping

WUGS and FPX

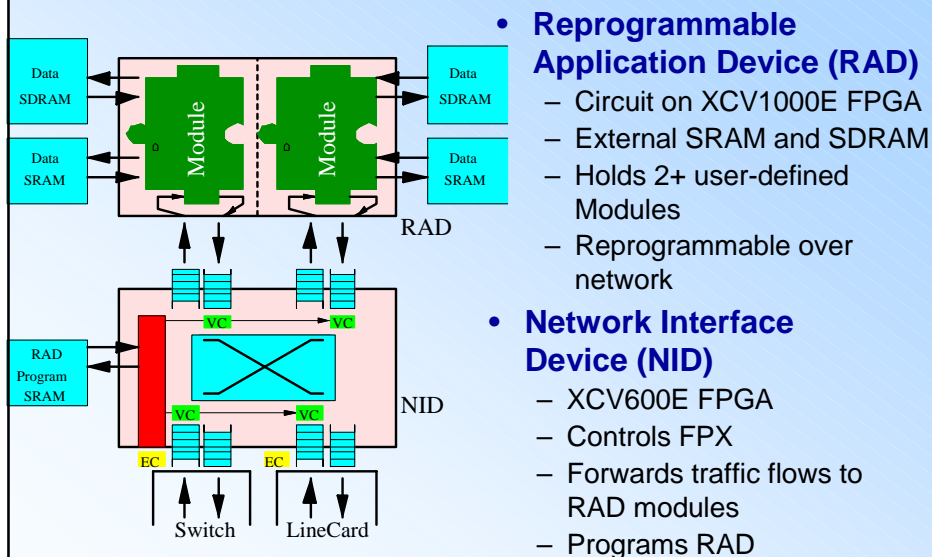


Field programmable Port EXtender (FPX)

- FPX fits between switch and line-card
- FPGA Hardware for user-defined applications
- Run-Time Reconfigurable

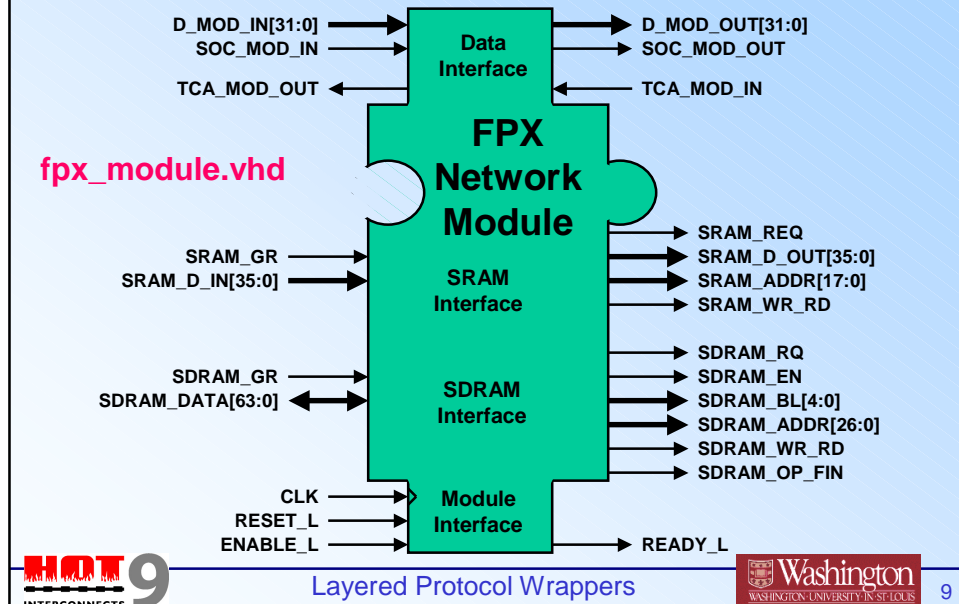


Architecture of the FPX



- **Reprogrammable Application Device (RAD)**
 - Circuit on XCV1000E FPGA
 - External SRAM and SDRAM
 - Holds 2+ user-defined Modules
 - Reprogrammable over network
- **Network Interface Device (NID)**
 - XCV600E FPGA
 - Controls FPX
 - Forwards traffic flows to RAD modules
 - Programs RAD

Network Module Hardware Interface

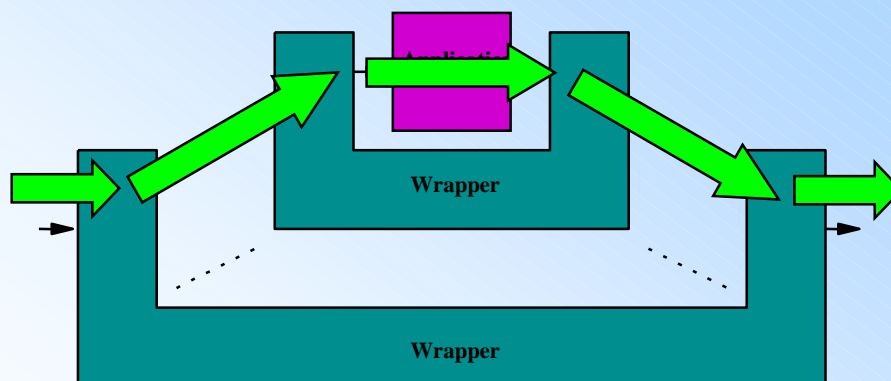


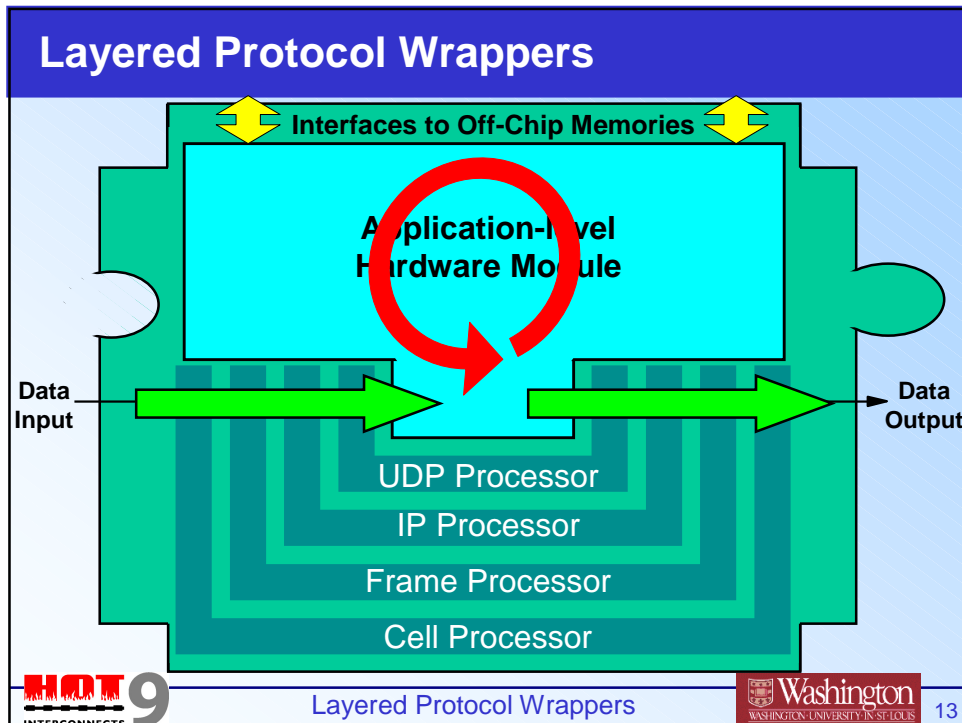
Layered Internet Protocol Wrappers

The Layered Protocol Wrapper Library

- Circuits that streamline functions to process cells, frames, IP packets, and UDP datagrams in reprogrammable hardware
- A layered design that consists of different processing circuit in each layer
- Allows application to be implemented at a level where important details are exposed and irrelevant details are hidden

Basic Concept of the Protocol Wrappers

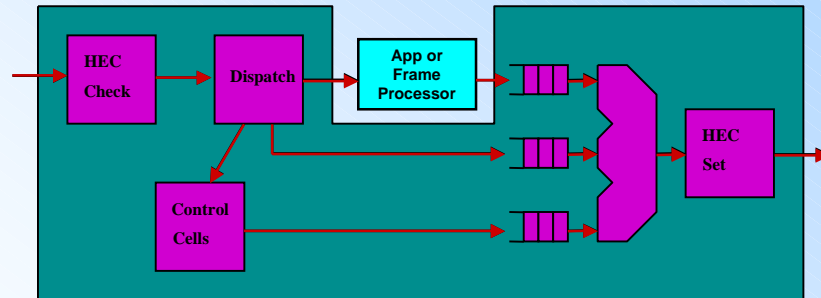




- ## The Cell Processor
- **Cell Functions**
 - Drops cells with bad cell headers
 - Recomputes cell header for outgoing cells
 - **Flow Functions**
 - Bypasses cells for different flows
 - **Implements Control-Plane Interface**
 - Handles control cells
 - Write control / read status
- Layered Protocol Wrappers

The Cell Processor (Detail)

- Data Flow inside the Cell Processor

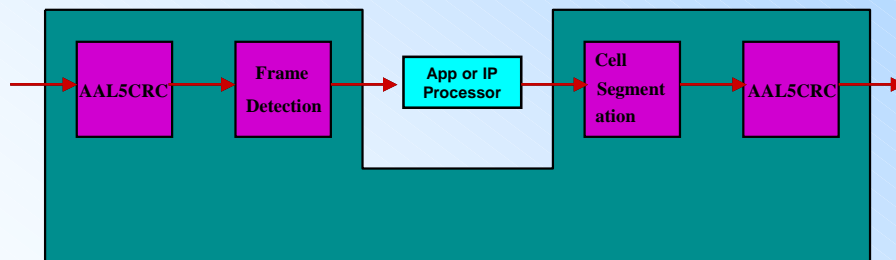


The Frame Processor

- Segmentation and Reassembly
 - Reassembles cells into frames
 - Segments frames into cells
- Frame Functions
 - Detects frame boundaries
 - Checks & Generates AAL5 CRC

The Frame Processor (Detail)

- Data Flow inside the Frame Processor

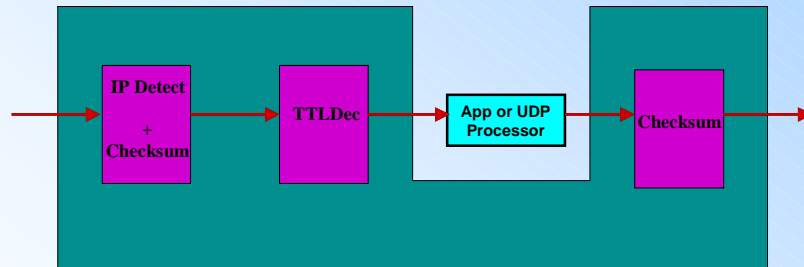


The IP Processor

- IP Version
 - Verifies IP version
- Header Checksum
 - Drops packet if the Header Checksum fails
 - Decrements TTL field (Optional)
- Provides
 - Signal start of payload (SOP)
 - Header Checksum Recomputation

The IP Processor (More)

- Data Flow inside the IP Processor

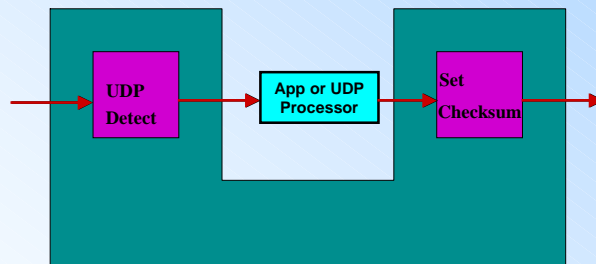


User Datagram Protocol (UDP) Processor

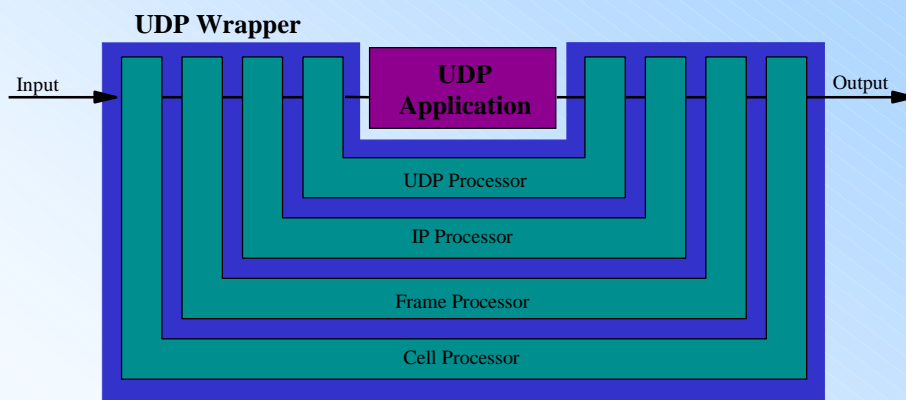
- Checks Protocol
 - Expects protocol ID (17)
- Provides Application-level signals
 - Start Of Datagram (SOD)
- Checksums
 - Checks UDP checksum
 - Generates UDP checksum

The UDP Processor (More)

- Data Flow inside the UDP Processor



Wrappers Example: An UDP Application



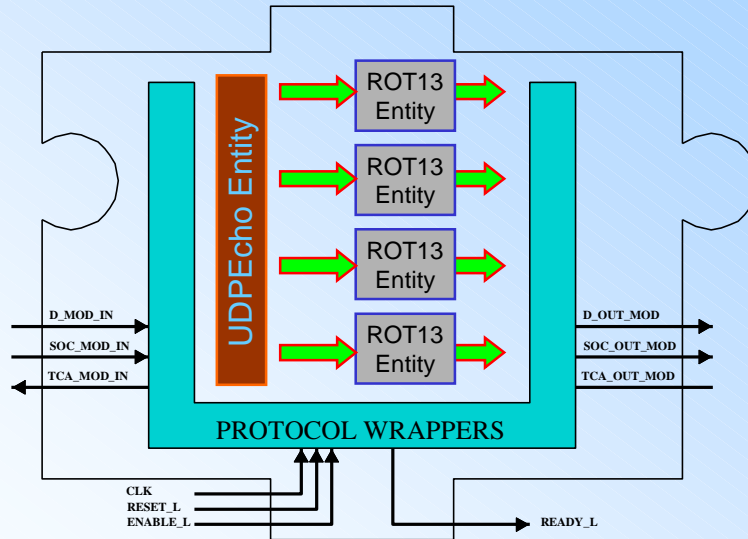
Application Example

Data Encryption

The Layered Protocol Application Example

- Network data encryption / decryption using ROT13 algorithm
 - Rotates characters by 13 places
 - 'A' \leftrightarrow 'N', 'M' \leftrightarrow 'Z', 'a' \leftrightarrow 'n', 'm' \leftrightarrow 'z'
 - Encryption Example:
 - 'Hello World' encrypts to 'Uryyb Jbeyq'
 - Decryption Example:
 - 'Uryyb Jbeyq' decrypts to 'Hello World'

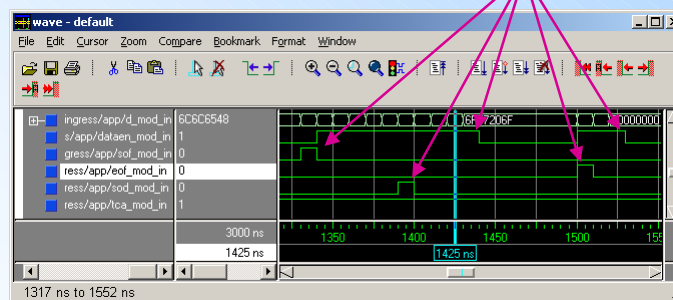
The Rot13 Encryption Module



Simulating the ROT13 Module (More)

- The input data coming into the module

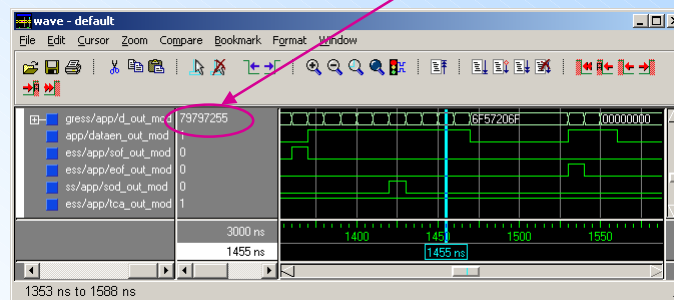
The last two valid words are the ATM Trailer



Simulating the ROT13 Module (More)

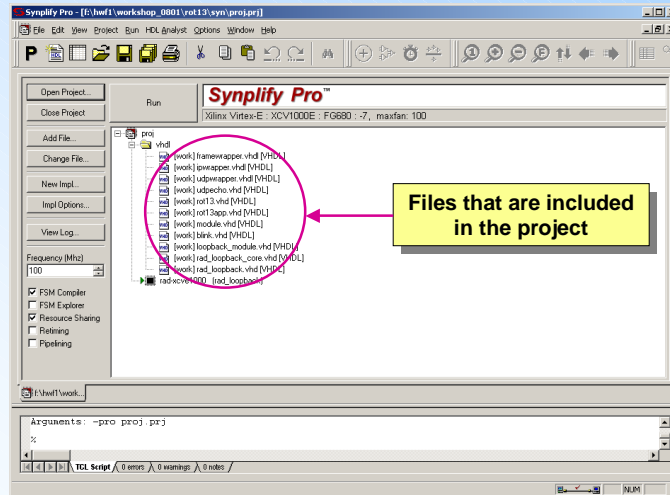
- The output data going out of the module

The UDPPayload has been encrypted / decrypted



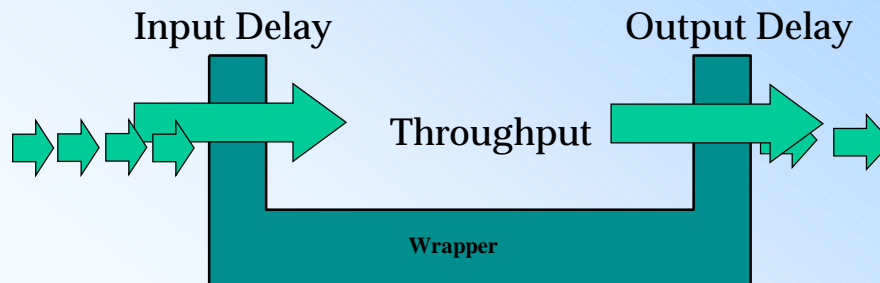
Implementation Results

Synthesizing the ROT13 Module (More)



Performance Results

- Short UDP datagram – one cell
- Long UDP datagram with 512 bytes payload



Synthesis Results (Chip Area & Data Rate)

- Wrappers Synthesized, Routed, and Placed on Xilinx XCV1000E-7 FPGA

	Space/LUTs	Clock / Data Rate
Cell Processor	781 (3%)	125 (3.2 Gbps)
Frame Processor	1251 (5%)	116 (3.0 Gbps)
IP Processor	1009 (4%)	109 (2.9 Gbps)
UDP Processor	550 (2%)	114 (2.9 Gbps)

Results (Clock Cycle Delays)

	Delay for short packages		Delay for long packages	
	Input	Output	Input	Output
Cell Processor	4	6	4	6
Frame Processor	21	22	10	31
IP Processor	36	39*	24	197*
UDP Processor	39	44*	27	202*

* Delay dependant of frame length

Conclusions

- Layered Protocol Wrappers
 - Simplify Development of Internet Apps in Reprogrammable Hardware
 - Circuits in library process: cells, frames, IP packets, & UDP datagrams
 - Application implemented at level where important details are exposed and irrelevant details are hidden
 - Low Chip Utilization
 - 14% of XCV1000E
 - High Performance
 - Throughput: 2.9 Gbps for OC48+ in hardware
 - Delay: 0.5 μ sec to process UDP/IP datagram
 - Performance scales with FPGA Technology
 - Potentially OC-192 (10 Gbps or better) in ASICs or newer FPGAs
 - Potentially OC-768 (40 Gbps or better) with parallel data
 - Synthesizable cores available for download
 - <http://www.arl.wustl.edu/arl/projects/fpx/>

On-Line References

- FPX Homepage
 - <http://www.arl.wustl.edu/arl/projects/fpx/>
- Protocol Wrappers Package
 - Project Page
 - [./fpx/wrappers](#)
 - Downloadable TAR File
 - [./fpx/wrappers/-wrappers.tar.gz](#)
 - Technical Report
 - [./fpx/wrappers/-wucs-01-10.pdf](#)

Acknowledgements

The authors would like to thank
Henry Fu (hwf1@arl.wustl.edu)

- First student to use the wrappers
- Developed KCPSM and Rot13 Application Modules
- Just finished as student at Washington University
- Starting at Stanford University in Fall'01