High-Level-Synthesis for FPGA Implementation of Network Protocols

Simon Lever, Univ. Ulm
- Dr. Endric Schubert, Univ. Ulm / Missing Link Electronics

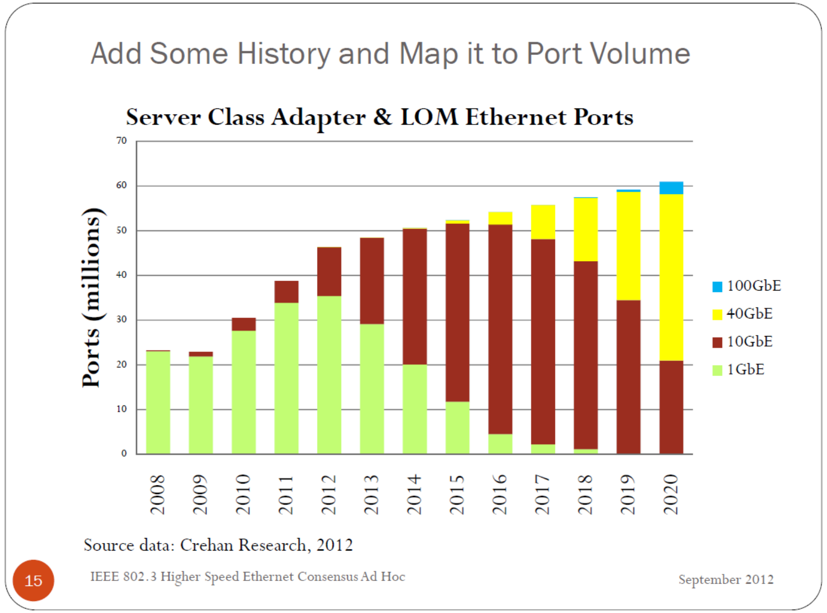
We are
a Silicon Valley based technology company with offices in Germany. We are partner of leading electronic device and solution providers and have been enabling key innovators in the automotive, industrial, test & measurement markets to build better Embedded Systems, faster.

Our Mission is
To develop and market technology solutions for Embedded Systems Realization via pre-validated IP and expert application support, and to combine off-the-shelf FPGA devices with Open-Source Software for dependable, configurable Embedded System platforms

Our Expertise is
I/O connectivity and acceleration of data communication protocols, additionally opening up FPGA technology for analog applications, and the integration and optimization of Open Source Linux and Android software stacks on modern extensible processing architectures.

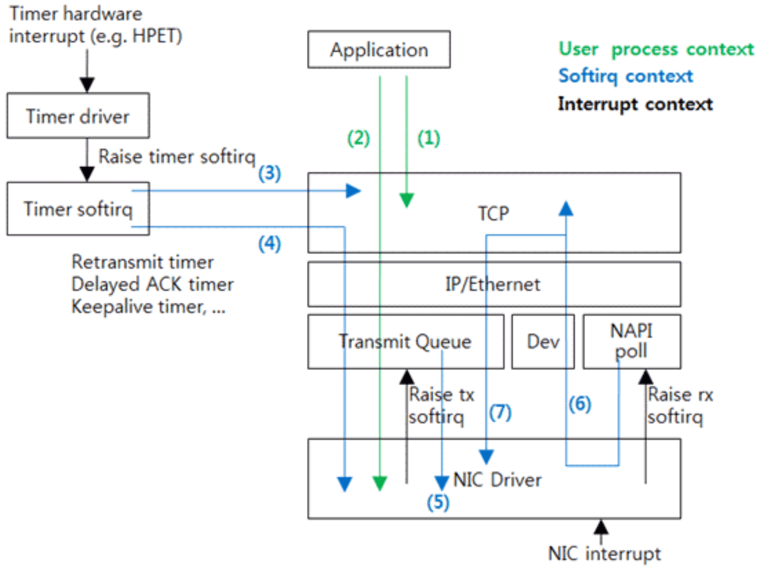
Motivation: Network Processing for Embedded Systems

- 10 GigE will soon push from data center into embedded markets



Transporting 1 bit per second needs 1 Hz

- 1 GigE ➔ 1 CPU at 1 GHz
- 10 GigE ➔ 4 CPUs at 2.5 GHz



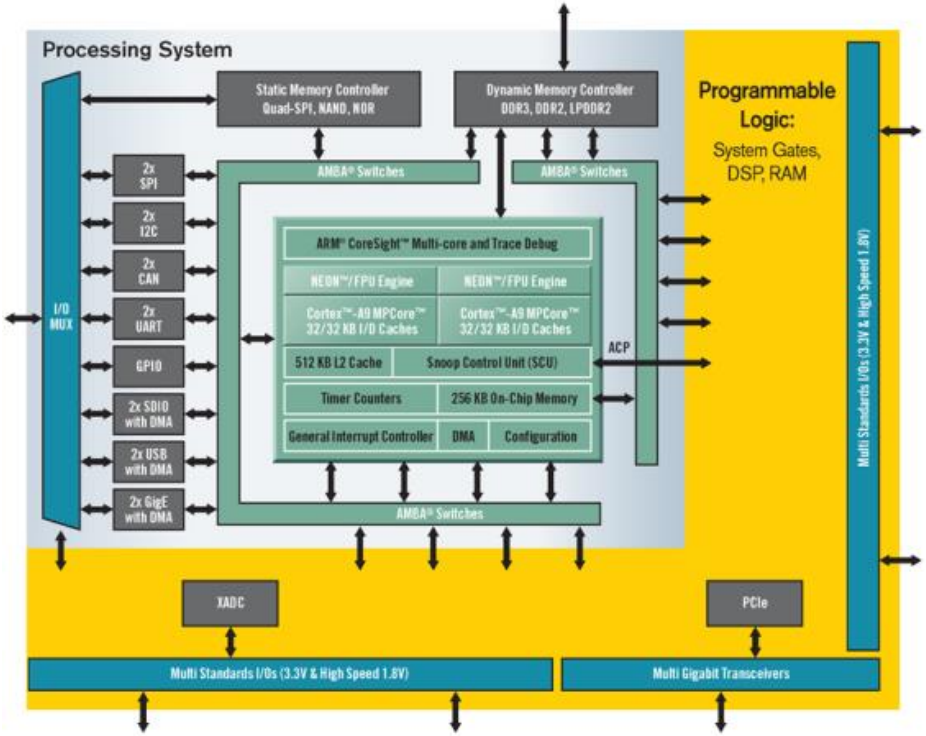
Design Choices for Network Processing in SoC FPGAs

SoC FPGA as (yet) another computer

	Intel i7-4770	Xilinx Zynq 7045
Compute	~100 GFLOPS	5 GFLOPS (PS) 778 GFLOPS (PL)
TDP	84 W	<20 W (typ)

SOC FPGA has 4x more compute

With ¼ the power dissipation!

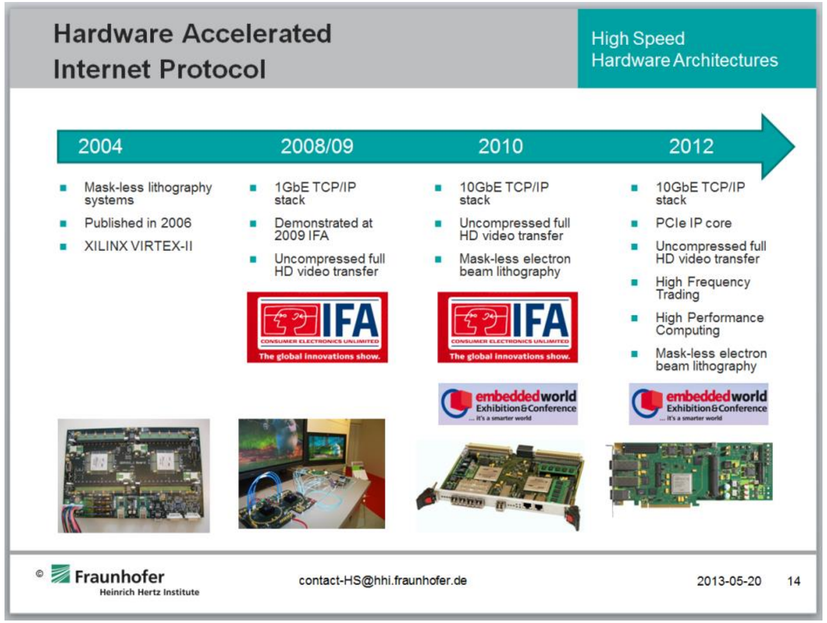


Processing System
Static Memory Controller Quad-SPI, NAND, NOR
Dynamic Memory Controller DDR3, DDR2, LPDDR2
Programmable Logic:
System Gates, DSP, RAM
I/O MUX
2x SPI
2x I2C
2x CAN
2x UART
GPIO
2x SDIO with DMA
2x USB with DMA
2x GigE with DMA
AMBA® Switches
AMBA® Switches
ARM® CoreSight™ Multi-core and Trace Debug
NEON™/FPU Engine
NEON™/FPU Engine
Cortex™-A9 MPCore™ 32/32 KB I/D Caches
Cortex™-A9 MPCore™ 32/32 KB I/D Caches
512 KB L2 Cache
Snoop Control Unit (SCU)
ACP
Timer Counters
256 KB On-Chip Memory
General Interrupt Controller
DMA
Configuration
AMBA® Switches
XADC
PCIe
Multi Standards I/Os (3.3V & High Speed 1.8V)
Multi Gigabit Transceivers
Multi Standards I/Os (3.3V & High Speed 1.8V)

[http://www.xilinx.com/products/technology/dsp.html]

Network Stack in RTL from Fraunhofer Heinrich-Hertz-Institute

- Brings full TCP/UDP/IP connectivity to FPGAs even when there is no CPU available. Accelerate CPUs by offloading TCP/UDP/IP processing into programmable logic.

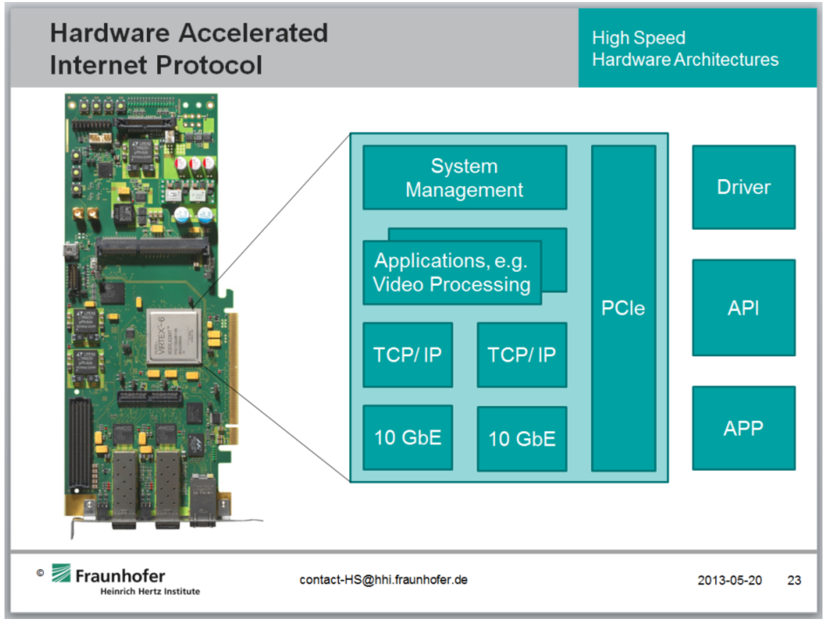


Hardware Accelerated Internet Protocol

High Speed Hardware Architectures

2004	2008/09	2010	2012
▪ Mask-less lithography systems ▪ Published in 2006 ▪ XILINX VIRTEX-II	▪ 1GbE TCP/IP stack ▪ Demonstrated at 2009 IFA ▪ Uncompressed full HD video transfer	▪ 10GbE TCP/IP stack ▪ Uncompressed full HD video transfer ▪ Mask-less electron beam lithography	▪ 10GbE TCP/IP stack ▪ PCIe IP core ▪ Uncompressed full HD video transfer ▪ High Frequency Trading ▪ High Performance Computing ▪ Mask-less electron beam lithography

© Fraunhofer Heinrich Hertz Institute contact-HS@hhi.fraunhofer.de 2013-05-20 14

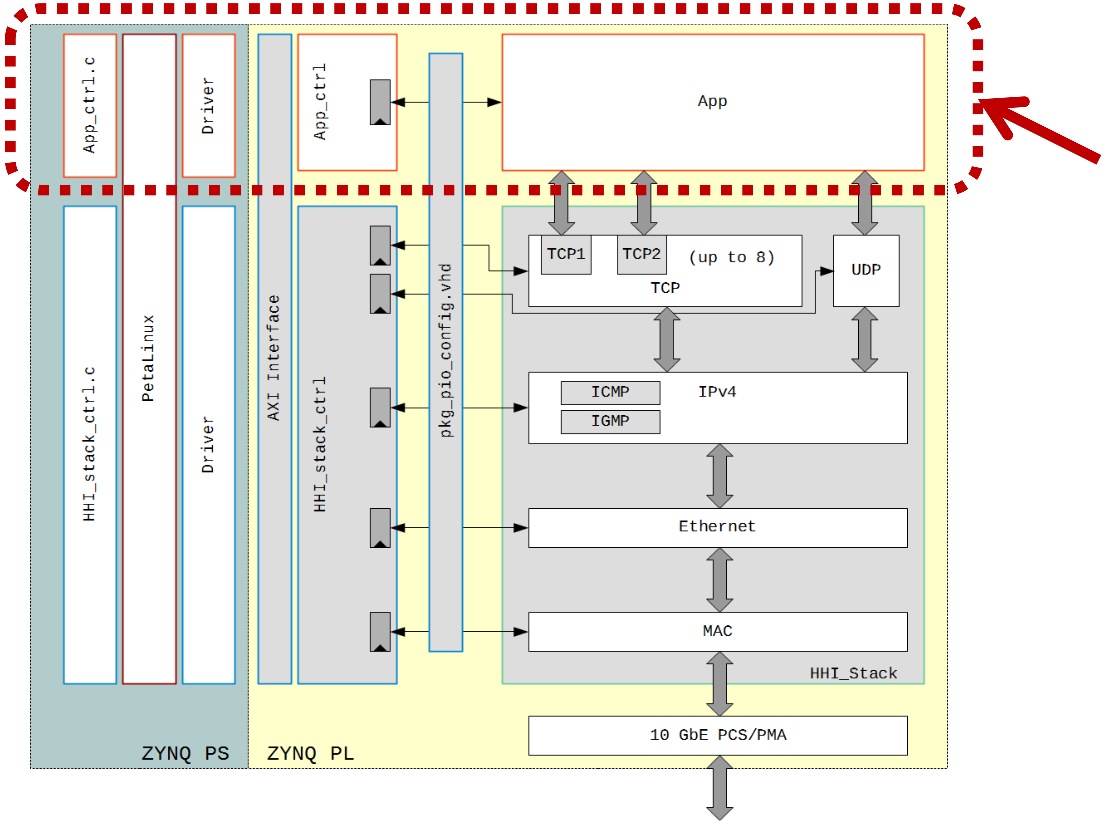


Hardware Accelerated Internet Protocol

High Speed Hardware Architectures

© Fraunhofer Heinrich Hertz Institute contact-HS@hhi.fraunhofer.de 2013-05-20 23

Network Protocol Acceleration Platform Architecture

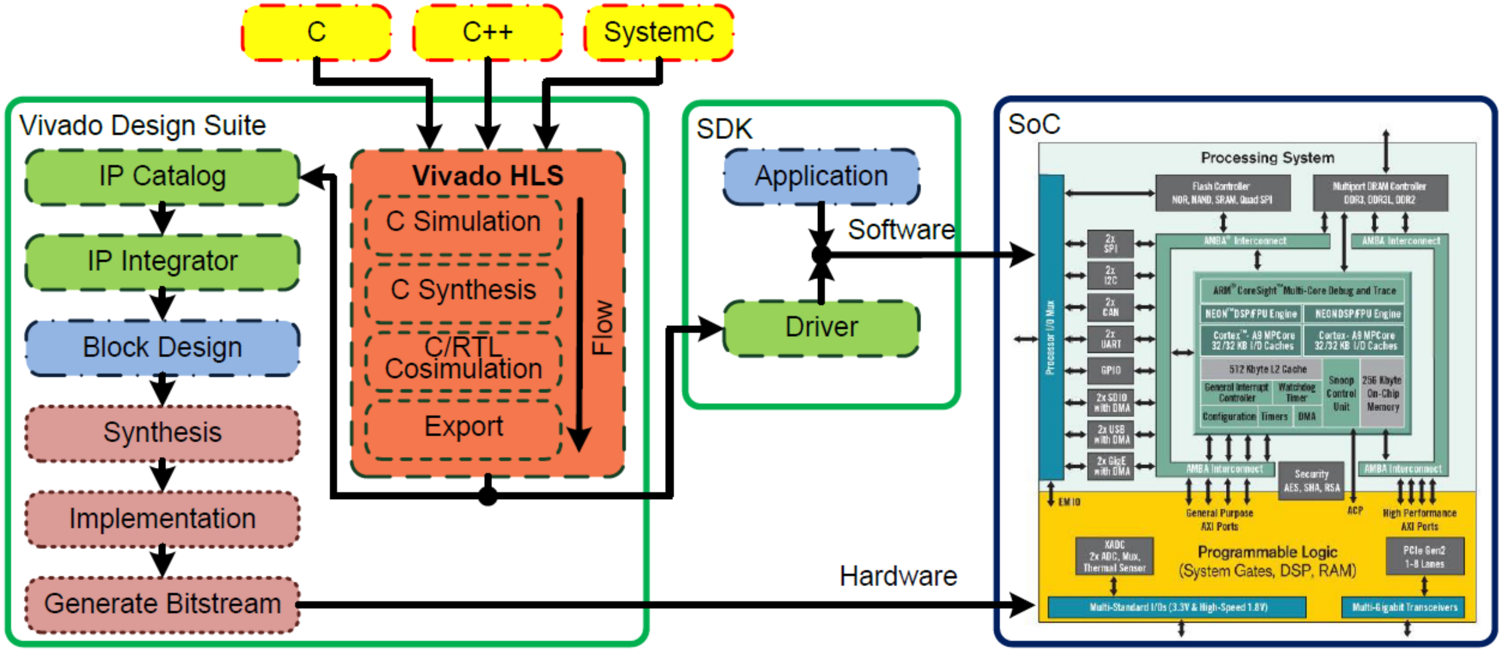


Network protocol processing at application layer (ISO Layer 7) can more efficiently be implemented via a programming approach (in C or C++) than by digital circuit design (in VHDL or Verilog).

mle missing link electronics

High-Level Synthesis Design Flow for SoC FPGA

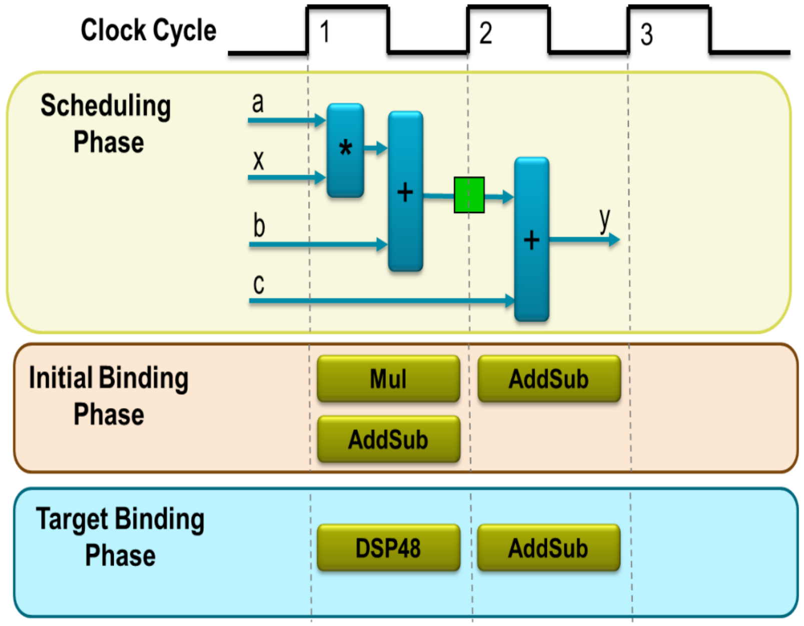
- Input C/C++/SystemC into High-Level Synthesis to generate VHDL/Verilog code

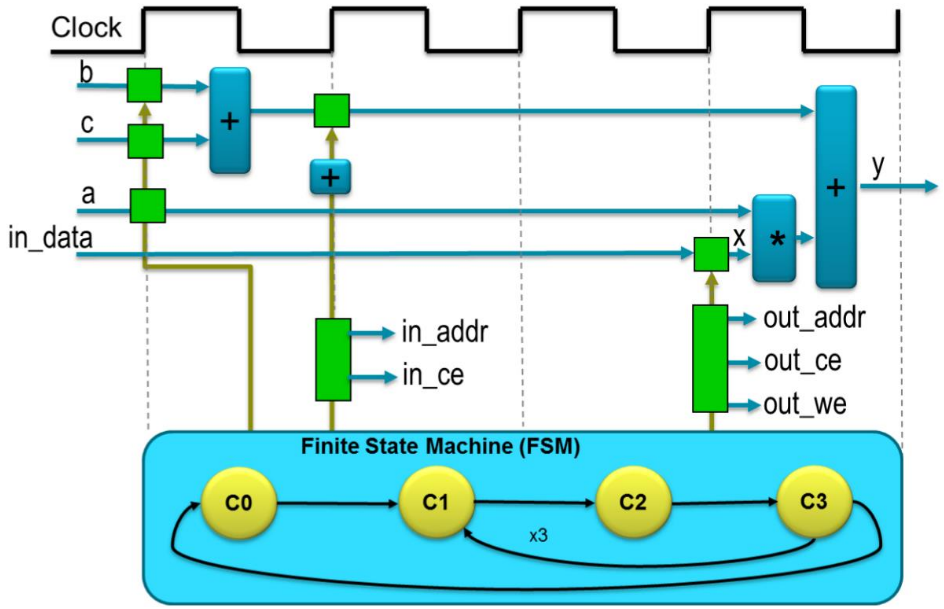


mle
missing link electronics

Working Principles of High-Level Synthesis

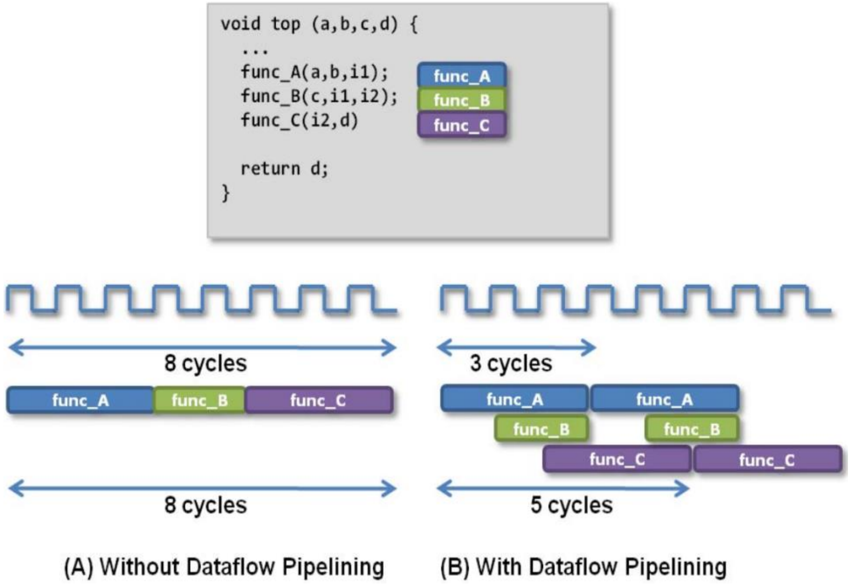
- Design automation runs scheduling and resource allocation to generate RTL code comprising data path plus state machines for control.





Benefits of High-Level Synthesis

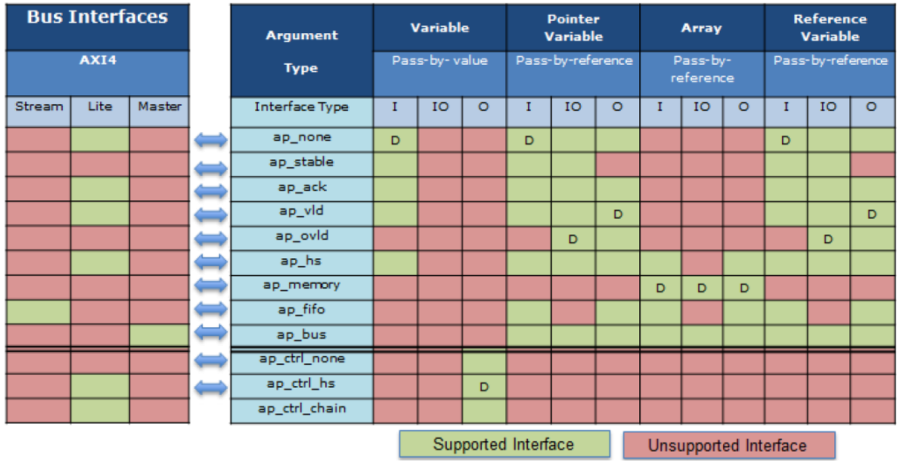
- Automatic performance optimization via parallelization at dataflow level



```
void top (a,b,c,d) {
  ...
  func_A(a,b,i1);
  func_B(c,i1,i2);
  func_C(i2,d)

  return d;
}
```

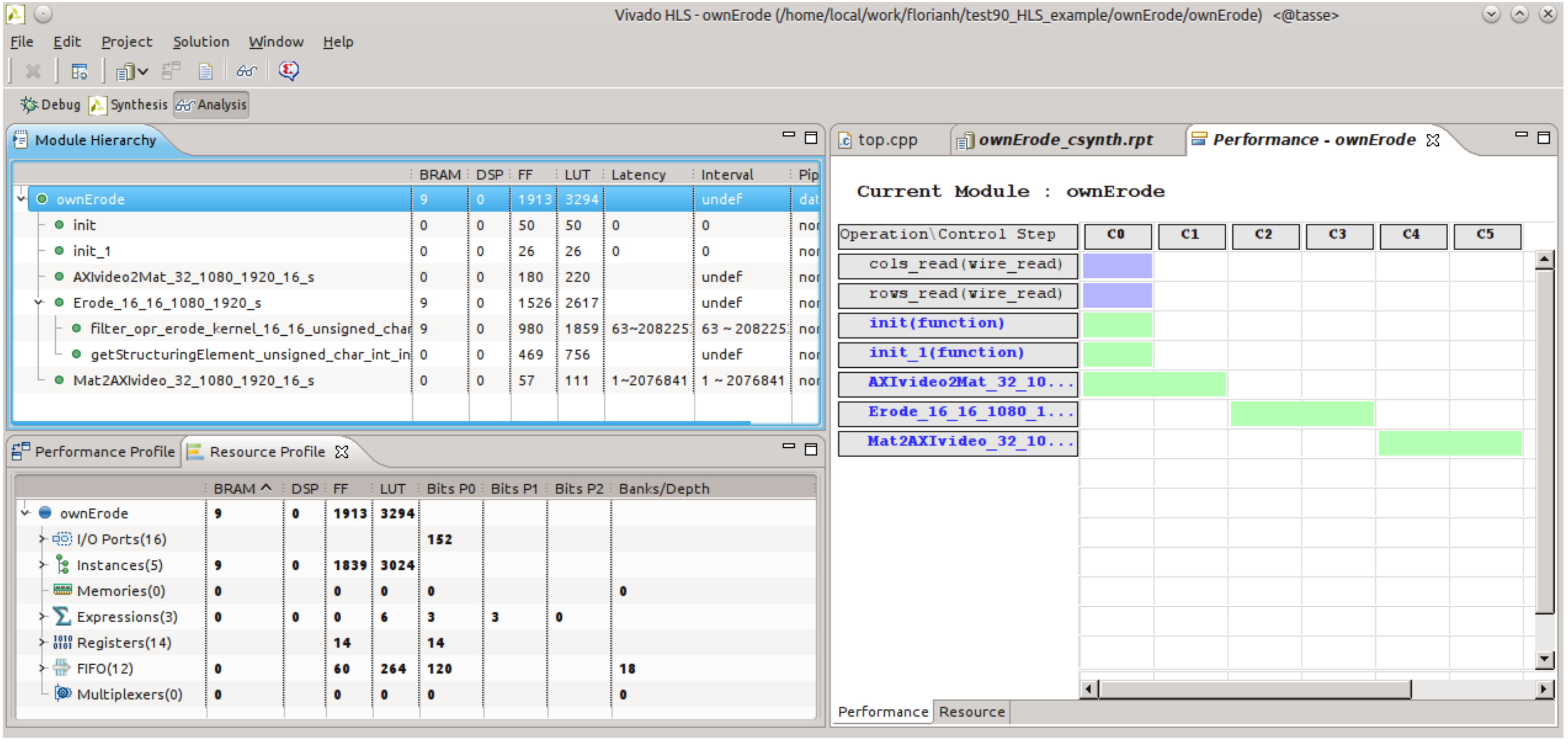
- Automatic interface synthesis and code generation for variety of real-life HW/SW connectivity



Bus Interfaces			Argument Type	Variable			Pointer Variable			Array			Reference Variable		
AXI4				Pass-by- value			Pass-by-reference			Pass-by- reference			Pass-by-reference		
Stream	Lite	Master	Interface Type	I	IO	O	I	IO	O	I	IO	O	I	IO	O
🟥	🟩	🟥	ap_none	🟩 D	🟥	🟥	🟩 D	🟩	🟩	🟥	🟥	🟥	🟩 D	🟩	🟩
🟥	🟥	🟥	ap_stable	🟩	🟥	🟥	🟩	🟩	🟥	🟥	🟥	🟥	🟩	🟩	🟥
🟥	🟩	🟥	ap_ack	🟩	🟥	🟥	🟩	🟩	🟩	🟥	🟥	🟥	🟩	🟩	🟩
🟥	🟩	🟥	ap_vld	🟩	🟥	🟥	🟩	🟩	🟩 D	🟥	🟥	🟥	🟩	🟩	🟩 D
🟥	🟥	🟥	ap_ovld	🟥	🟥	🟥	🟥	🟩 D	🟩	🟥	🟥	🟥	🟥	🟩 D	🟩
🟥	🟩	🟥	ap_hs	🟩	🟥	🟥	🟩	🟩	🟩	🟩	🟥	🟩	🟩	🟩	🟩
🟥	🟥	🟥	ap_memory	🟥	🟥	🟥	🟥	🟥	🟥	🟩 D	🟩 D	🟩 D	🟥	🟥	🟥
🟩	🟥	🟥	ap_fifo	🟥	🟥	🟥	🟩	🟥	🟩	🟩	🟥	🟩	🟩	🟥	🟩
🟥	🟥	🟩	ap_bus	🟥	🟥	🟥	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
🟥	🟥	🟥	ap_ctrl_none	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
🟥	🟩	🟥	ap_ctrl_hs	🟥	🟥	🟩 D	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
🟥	🟩	🟥	ap_ctrl_chain	🟥	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

🟩 Supported Interface 🟥 Unsupported Interface

Visualization and User Interaction in High-Level Synthesis Tool



	BRAM	DSP	FF	LUT	Latency	Interval	Pip
ownErode	9	0	1913	3294		undef	dat
init	0	0	50	50	0	0	nor
init_1	0	0	26	26	0	0	nor
AXIvideo2Mat_32_1080_1920_16_s	0	0	180	220		undef	nor
Erode_16_16_1080_1920_s	9	0	1526	2617		undef	nor
filter_opr_erode_kernel_16_16_unsigned_char	9	0	980	1859	63~208225	63 ~ 208225	nor
getStructuringElement_unsigned_char_int_in	0	0	469	756		undef	nor
Mat2AXIvideo_32_1080_1920_16_s	0	0	57	111	1~2076841	1 ~ 2076841	nor

	BRAM ^	DSP	FF	LUT	Bits P0	Bits P1	Bits P2	Banks/Depth
ownErode	9	0	1913	3294				
I/O Ports(16)					152			
Instances(5)	9	0	1839	3024				
Memories(0)	0		0	0	0			0
Expressions(3)	0	0	0	6	3	3	0	
Registers(14)			14		14			
FIFO(12)	0		60	264	120			18
Multiplexers(0)	0		0	0	0			0

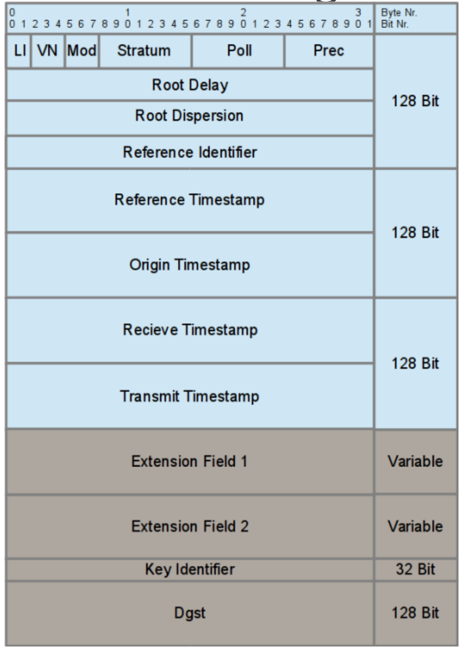
Current Module : ownErode

Operation\Control Step	C0	C1	C2	C3	C4	C5
cols_read(wire_read)	🟦					
rows_read(wire_read)	🟦					
init(function)	🟩					
init_1(function)	🟩					
AXIvideo2Mat_32_10...	🟩	🟩				
Erode_16_16_1080_1...			🟩	🟩		
Mat2AXIvideo_32_10...					🟩	🟩

mle
missing link electronics

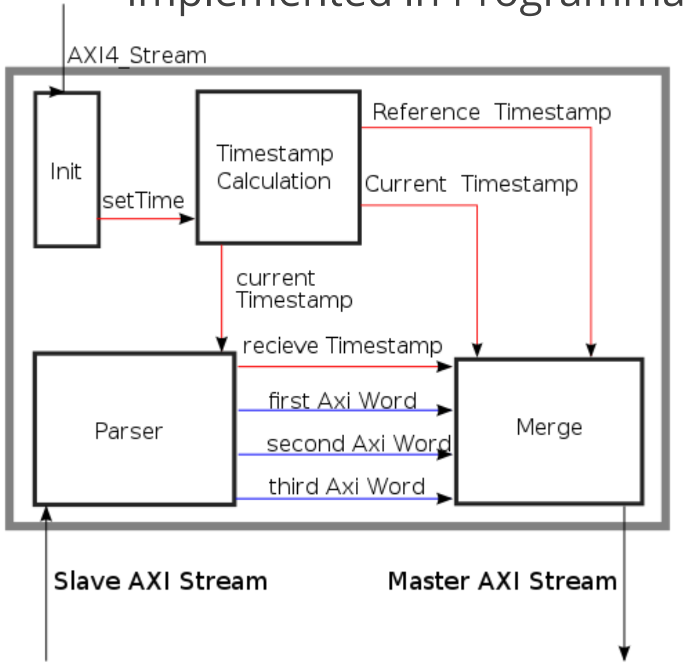
Design Example for Extending NPAP with High-Level Synthesis

- Network Time Protocol (NTP) Packet according to RFC5905



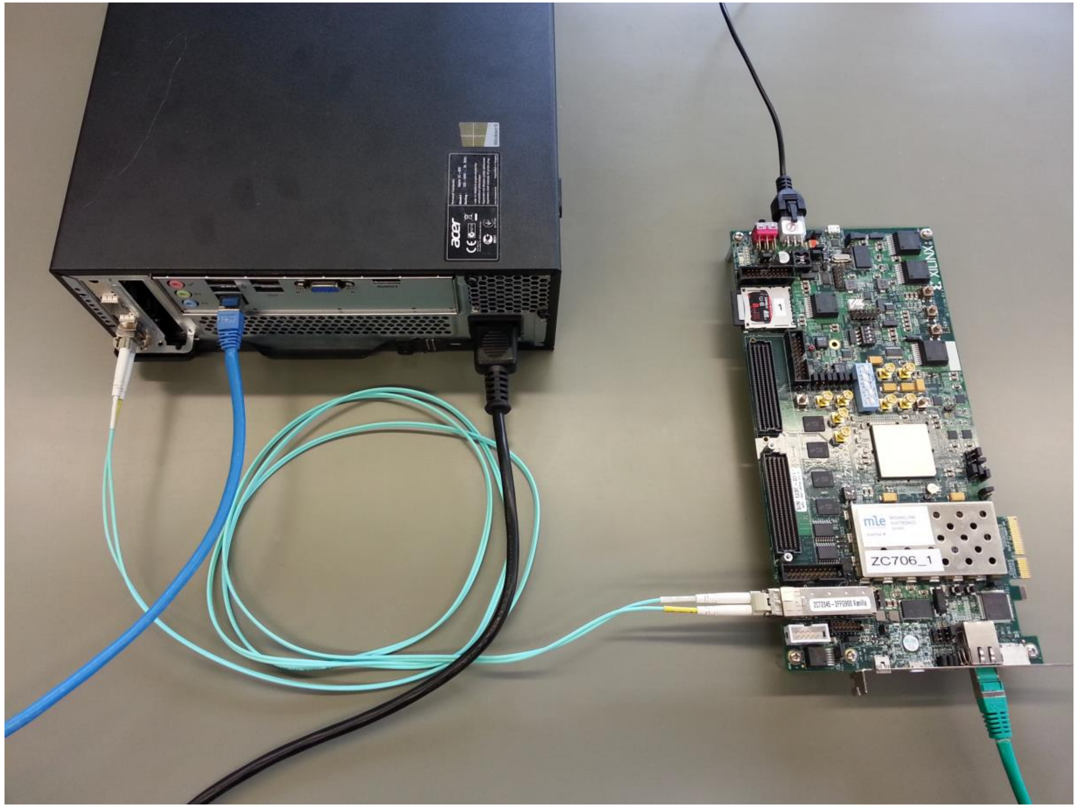
0 1 2 3 4 5 6 7 8 9	1 0 1 2 3 4 5 6 7 8 9	2 0 1 2 3 4 5 6 7 8 9	3 0 1			Byte Nr. Bit Nr.
LI	VN	Mod	Stratum	Poll	Prec	128 Bit
Root Delay						
Root Dispersion						
Reference Identifier						
Reference Timestamp						128 Bit
Origin Timestamp						
Recieve Timestamp						128 Bit
Transmit Timestamp						
Extension Field 1						Variable
Extension Field 2						Variable
Key Identifier						32 Bit
Dgst						128 Bit

- Block diagram of the NTP Server implemented in Programmable Logic



Implementation Example

- Implement the NTP server as an “IP core” using Vivado HLS
- The network processing stack, including the NTP server IP core, runs on Xilinx Zynq-7000 SoC
- Xilinx ZC706 evaluation kit connected via SFP+ with Intel 10GbE NIC inside PC
- PC runs NTP client application to query Zynq-implemented NTP server



Conclusion and References

- Significant productivity increase for protocol oriented or dataflow based design blocks.
- Easy to adopt: Known languages C/C++ combined with known tool chain.
- → Add this to your bag of tricks!

- UG998 - Introduction to FPGA Design Using High-Level Synthesis
- UG871 - Vivado Design Suite Tutorial: High-Level Synthesis
- XAPP1209 - Designing Protocol Processing Systems with Vivado High-Level Synthesis
- UG949 - UltraFast Design Methodology Guide for the Vivado Design Suite

Contact Information

Simon Lever

simon.lever@uni-ulm.de

simon.lever@MLEcorp.com

Dr. Endric Schubert

endric.schubert@uni-ulm.de
endric@MLEcorp.com
Phone US: +1 (408) 320-6139
Phone DE: +49 (731) 141149-66