Programming Modern FPGAs

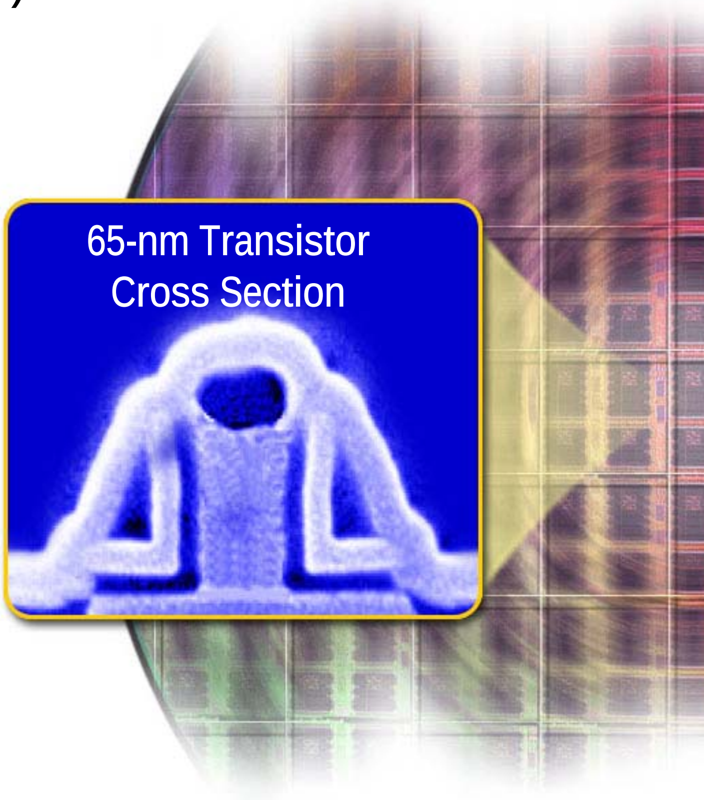
Ivo Bolsens
Xilinx
MPSOC August, 2006

Outline

- Modern FPGA
- FPGA programmable platform
- Programming the FPGA
- Conclusions

Modern FPGA

- 65nm technology, 40-nm gate length (Poly)
- 1.6nm oxide thickness (16 Angstrom)
  - ~5 atomic layers
- Triple-Oxide Technology
  - 3 oxide thicknesses for optimum power and performance
- 1.0 Vcc core
  - Lower dynamic power
- 12 layer copper
- Strained silicon transistor
  - Maximum performance at lowest AC power


65-nm Transistor Cross Section

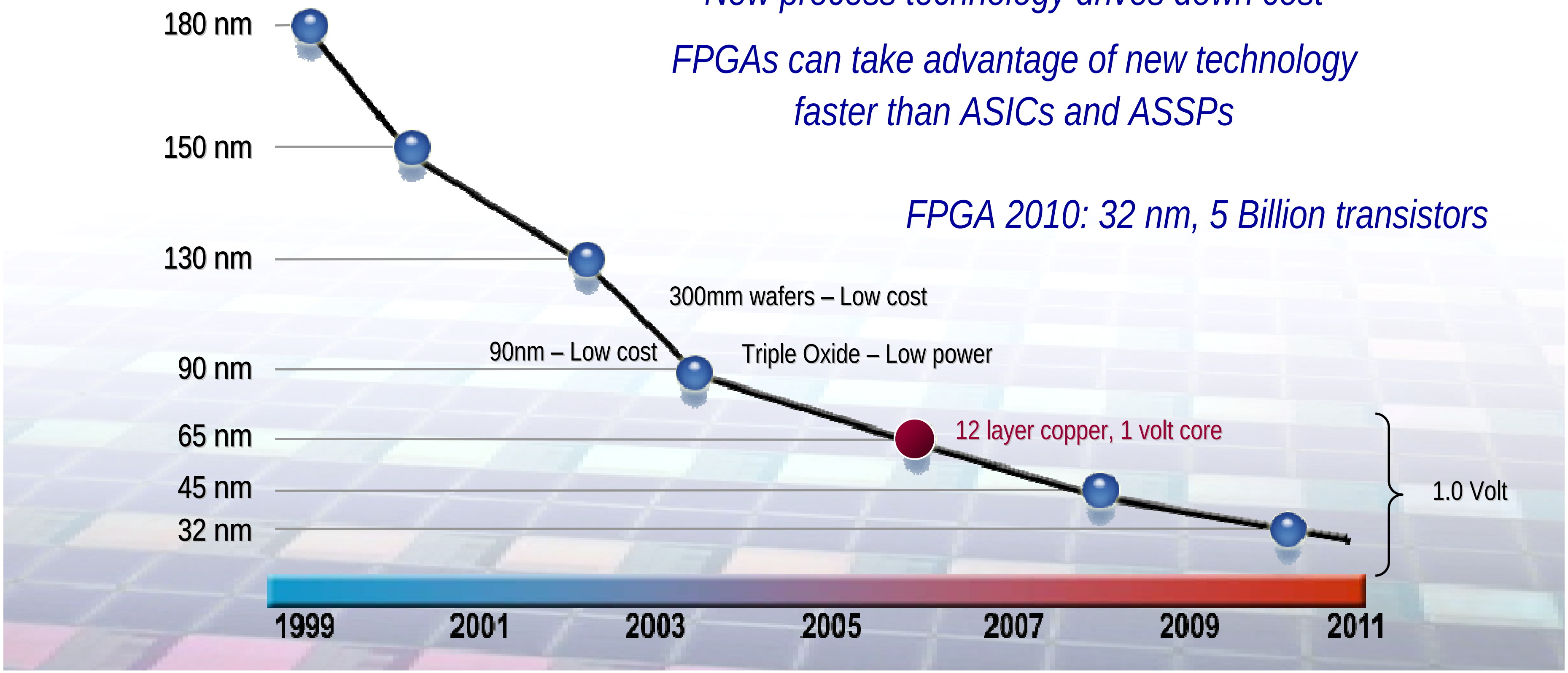
Over 1 Billion Transistors

FPGA Roadmap

New process technology drives down cost

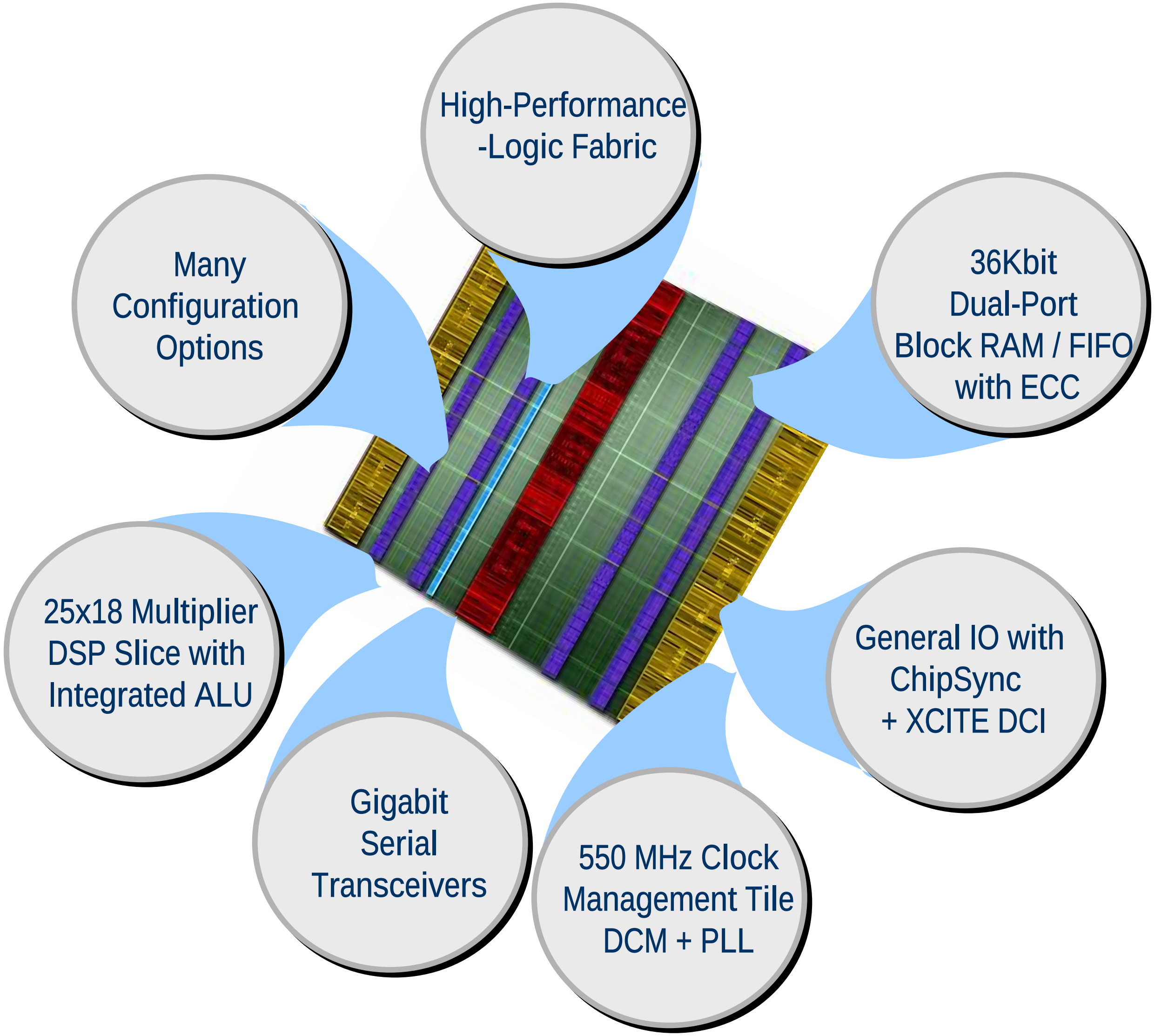
FPGAs can take advantage of new technology faster than ASICs and ASSPs

FPGA 2010: 32 nm, 5 Billion transistors



The cost of IC development increases. Therefore customers want to buy reconfigurable and programmable platforms, instead of developing their own.

FPGA Fabric



High-Performance -Logic Fabric

Many Configuration Options

36Kbit Dual-Port Block RAM / FIFO with ECC

25x18 Multiplier DSP Slice with Integrated ALU

General IO with ChipSync + XCITE DCI

Gigabit Serial Transceivers

550 MHz Clock Management Tile DCM + PLL

Logic Architecture

True 6-input Lookup Table (LUT)

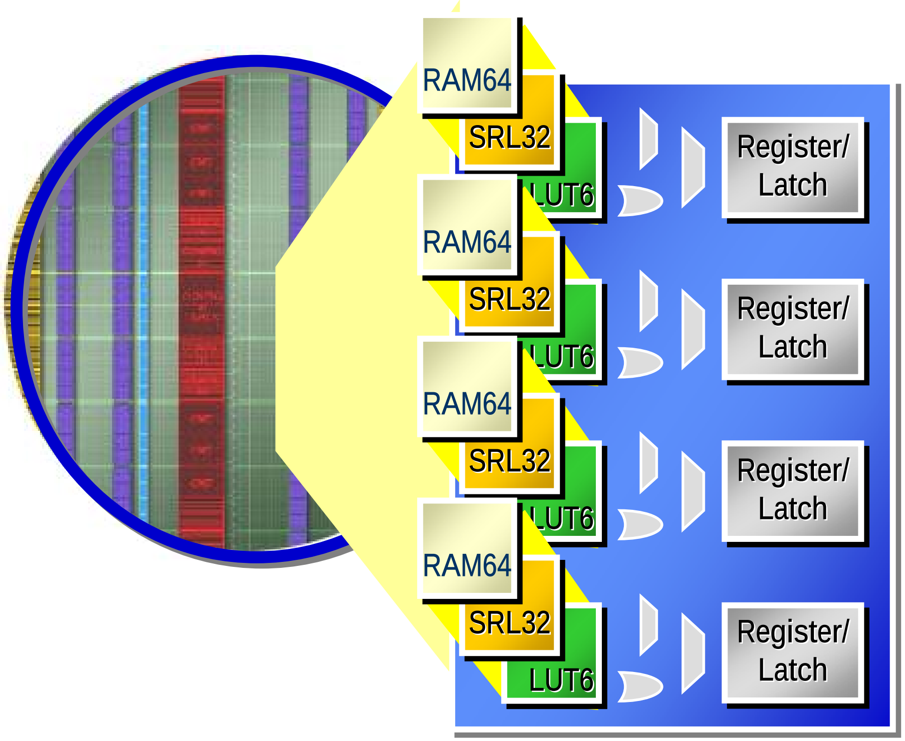
with dual 5-input LUT option

64-bit RAM per M-LUT

about half of all LUTs

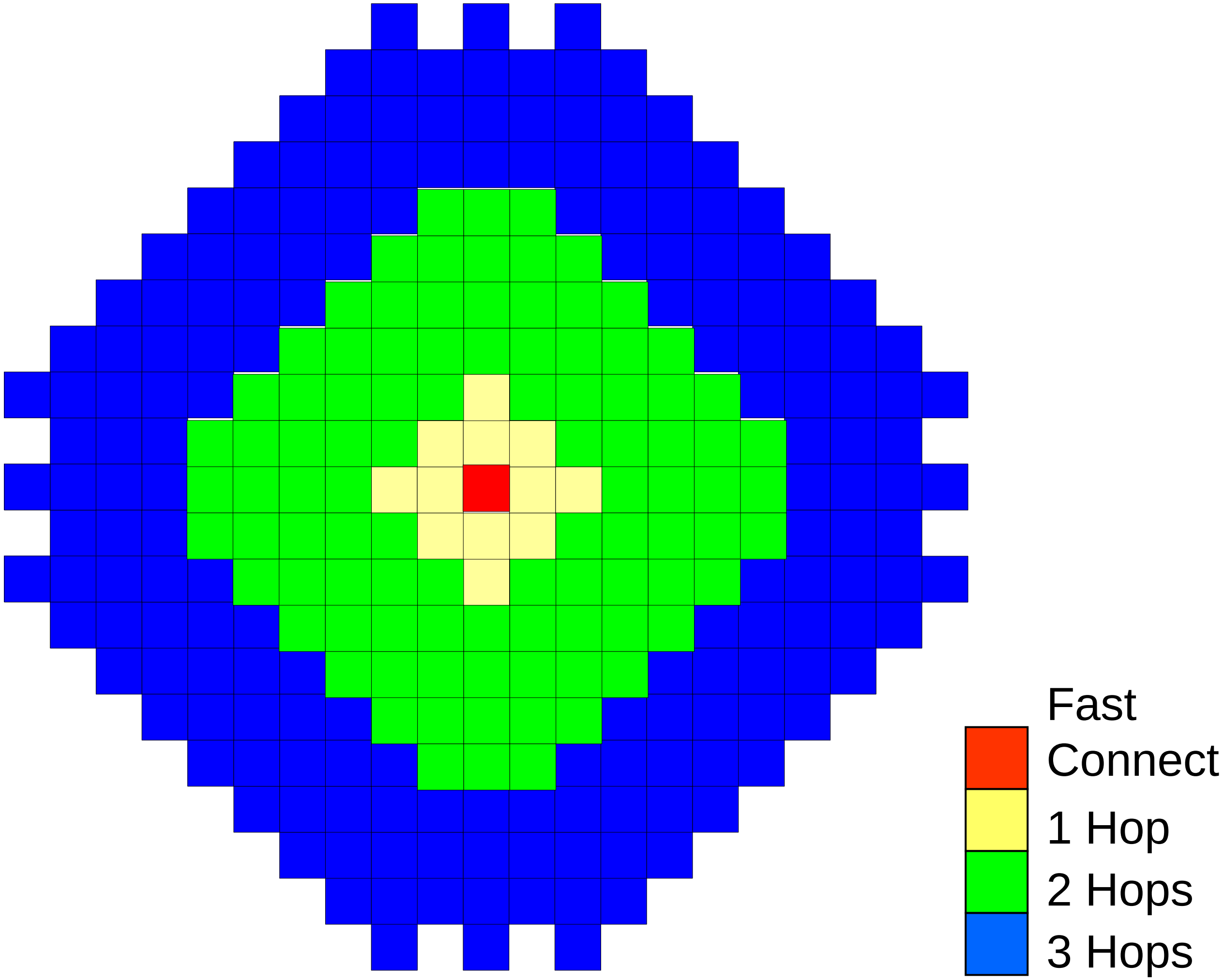
32-bit or 16-bit x 2

shift register per M-LUT



Virtex-5 Routing

Symmetric pattern, connecting CLBs



Same pattern for all outputs

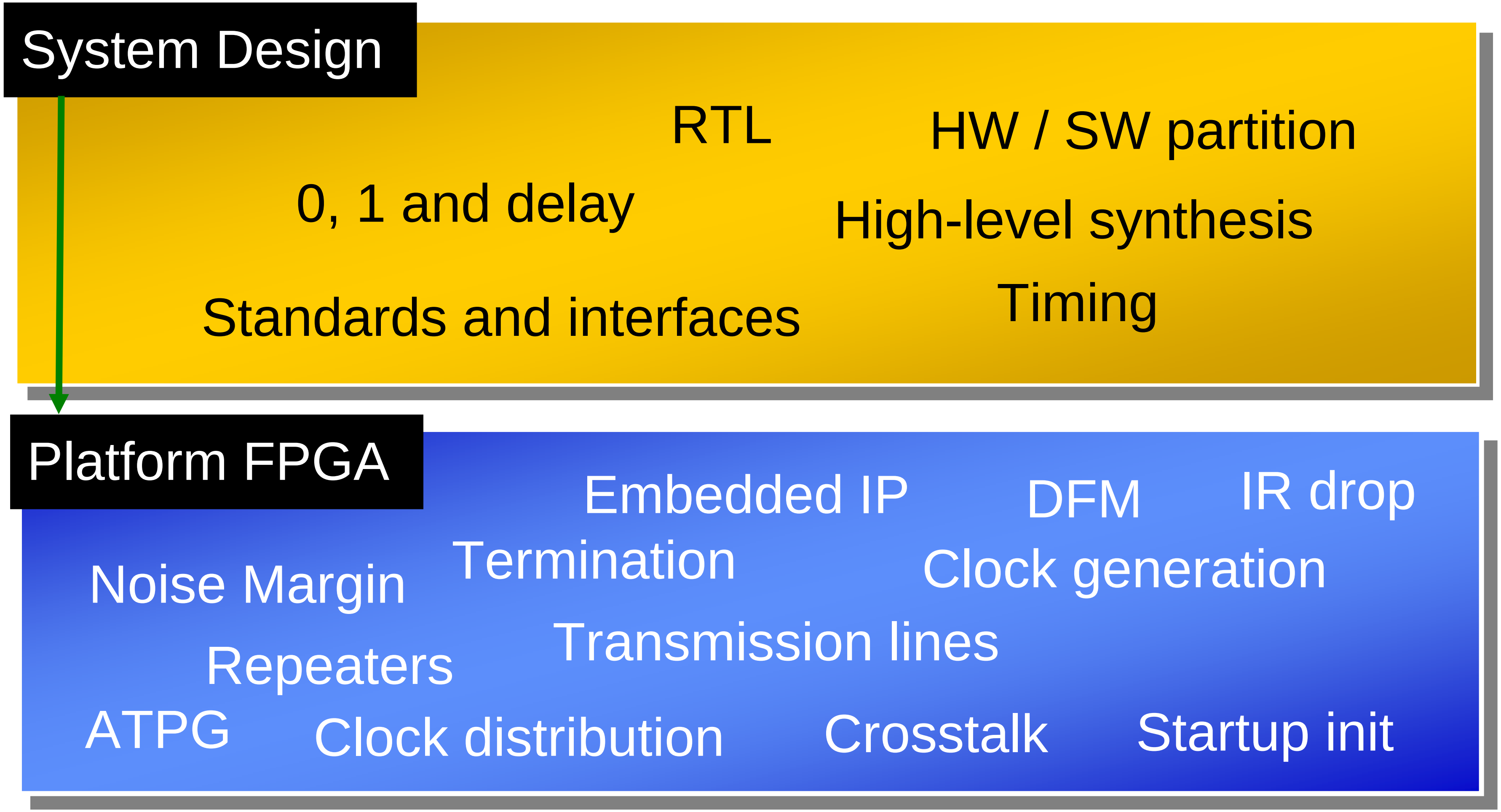
General Purpose I/O (Select I/O)

- All I/O pins are “created equal”
- Compatible with >40 different standards
  - Vcc, output drive, input threshold, single/differential, etc
- Each I/O pin has **dedicated circuitry** for:
  - On-chip transmission-line termination (serial or parallel)
  - Serial-to-parallel converter on the input (CHIPSYNC)
  - Parallel -to-serial converter on the output (CHIPSYNC)
  - Clock divider, and high-speed “regional” clock distribution

Ideal for source-synchronous I/O up to 1 Gbps

Platform FPGAs

Digital System Design Simplified



System Design

RTL

HW / SW partition

0, 1 and delay

High-level synthesis

Standards and interfaces

Timing

Platform FPGA

Embedded IP

DFM

IR drop

Noise Margin

Termination

Clock generation

Transmission lines

Repeaters

ATPG

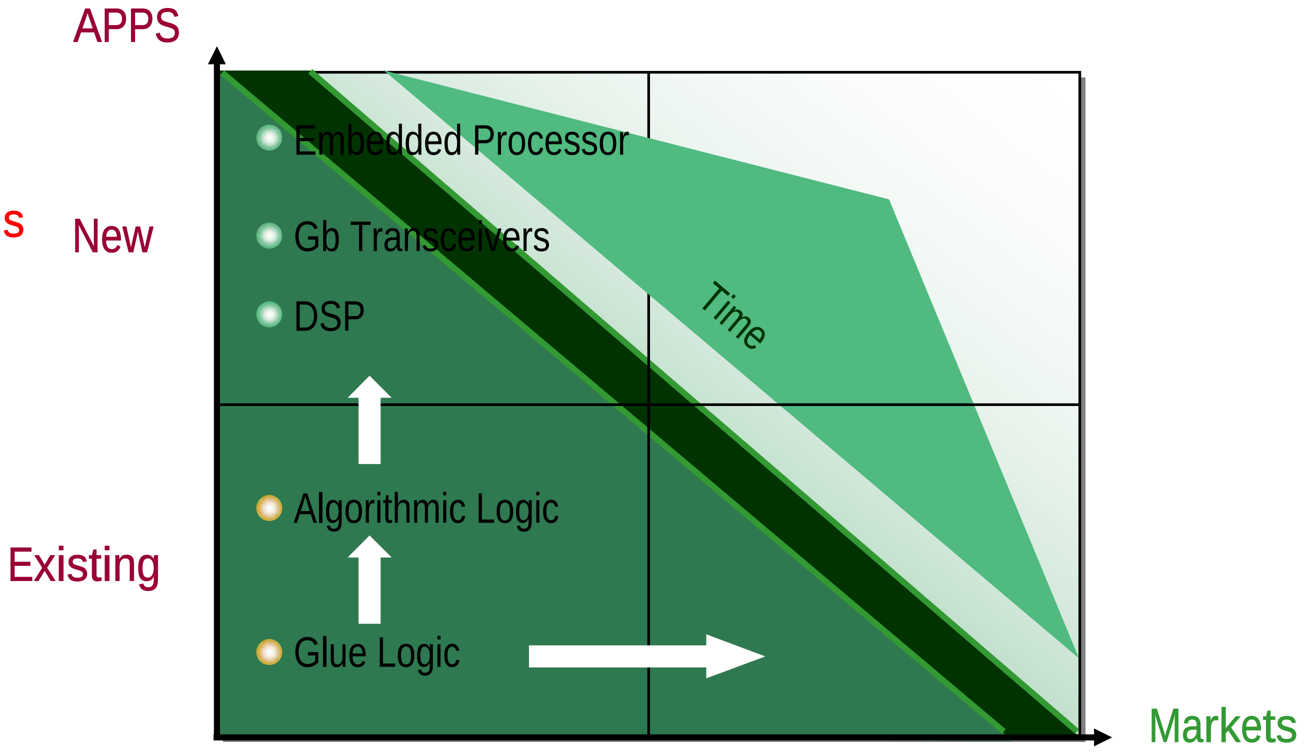
Clock distribution

Crosstalk

Startup init

Xilinx Strategic Directions

Integration
Hard IP
System Tools



APPS

New

Existing

Markets

- Embedded Processor
- Gb Transceivers
- DSP
- Algorithmic Logic
- Glue Logic

Existing

- Network Infrastructure
- Computing Infrastructure
- Industrial, medical
- Military

New

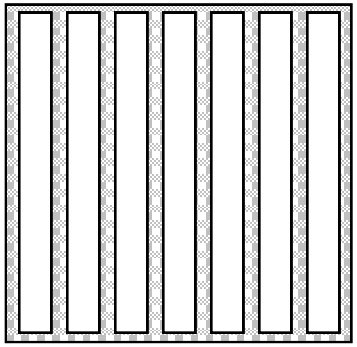
- Consumer Electronics
- Automotive
- Portable

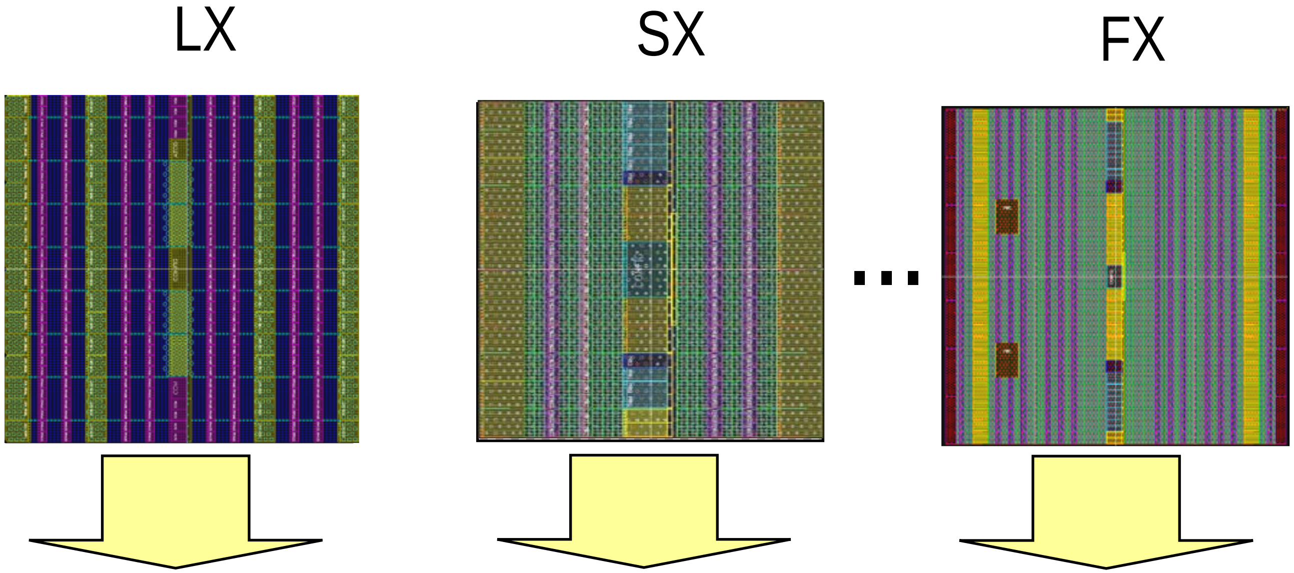
Cost
Power
Quality

Domain Optimized Platforms

One Family – Multiple Platforms

Column based features





🟦 Logic

🟩 Memory

🟧 DSP

🟨 Processing

🟥 High-speed I/O

Logic Domain
Highest logic density

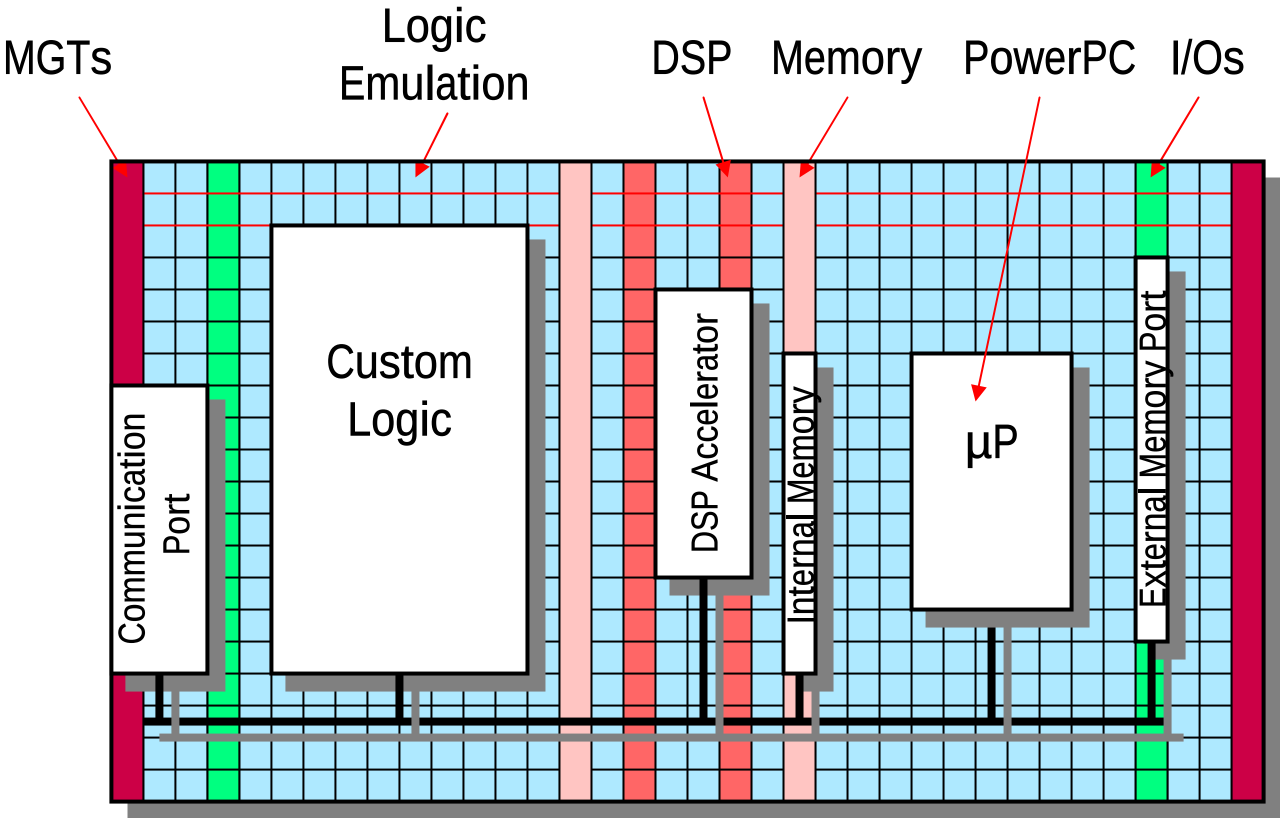
DSP Domain
Highest DSP performance

Connectivity Domain
Embedded Processors
High-speed Serial I/O

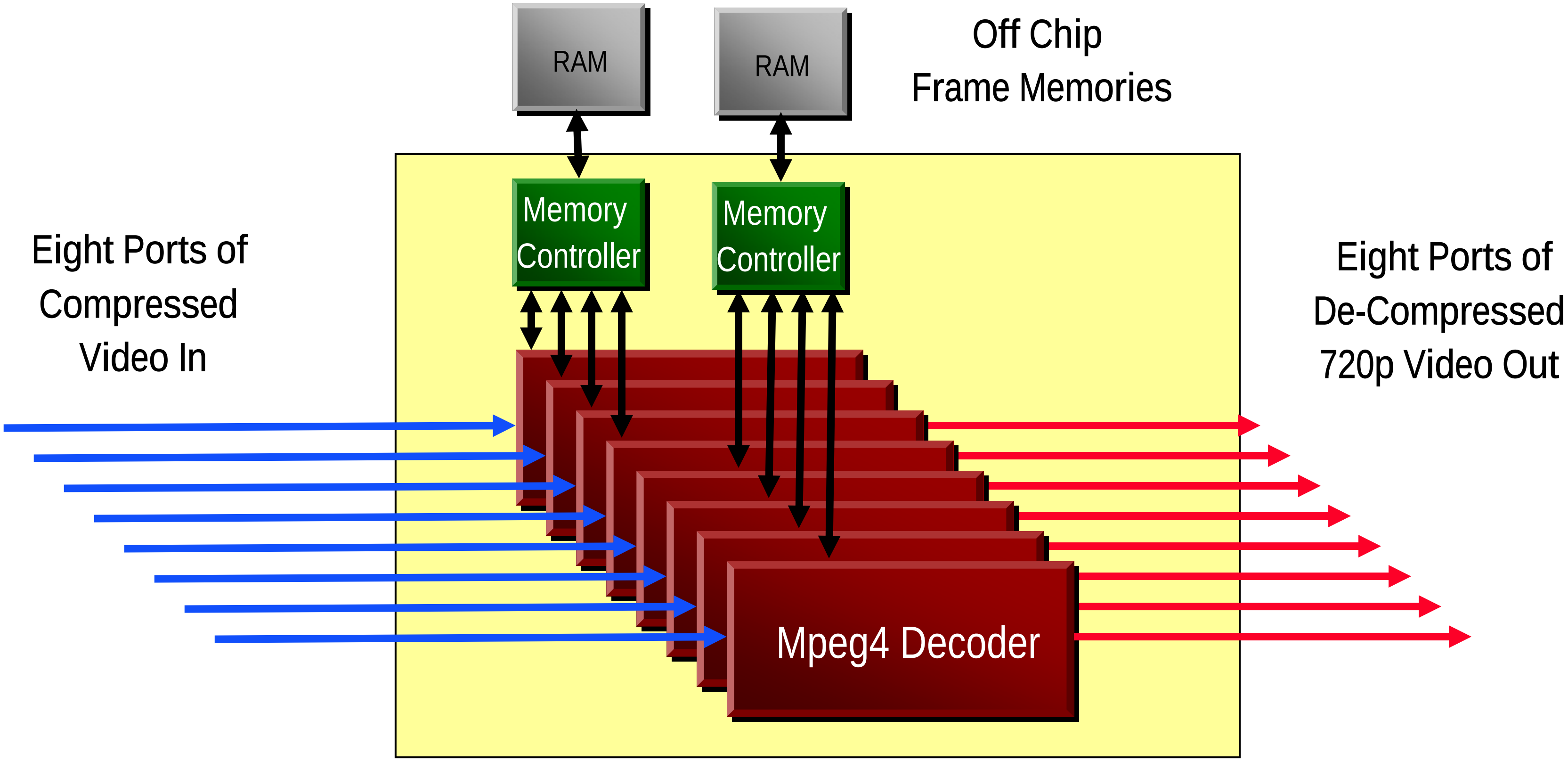
Enables “Dial-In” hard IP Mix
- Logic, DSP, BRAM, I/O, MGT, DCM, PowerPC

Enabled by Flip-Chip Packaging
- I/O Columns Distributed Throughout the Device

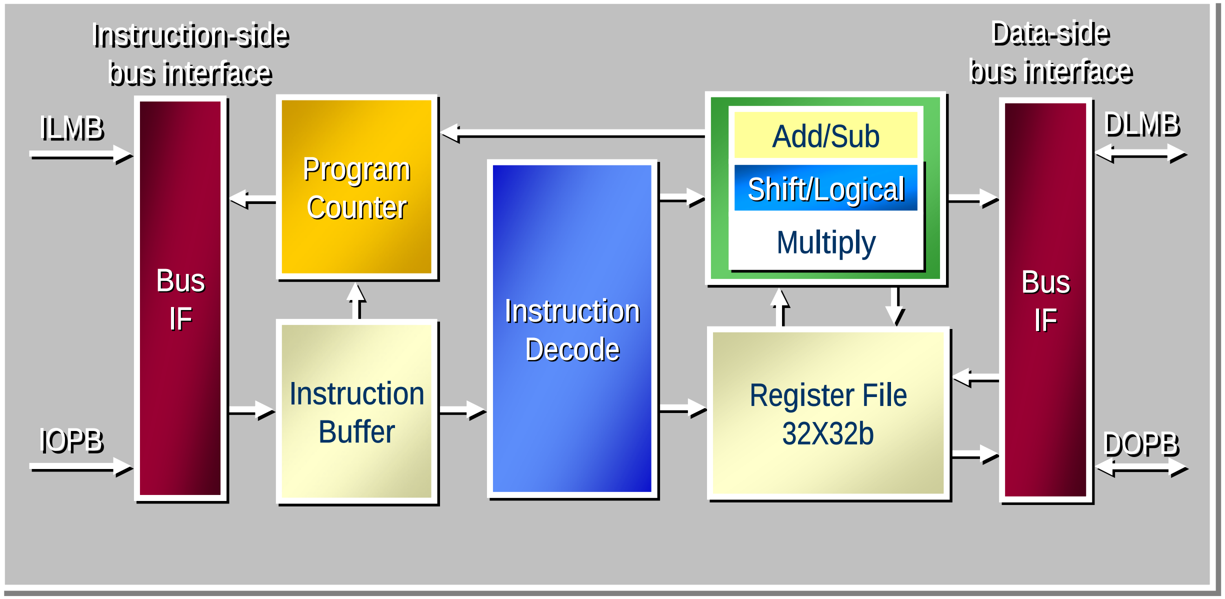
The FPGA System



8 MPEG4 decoders



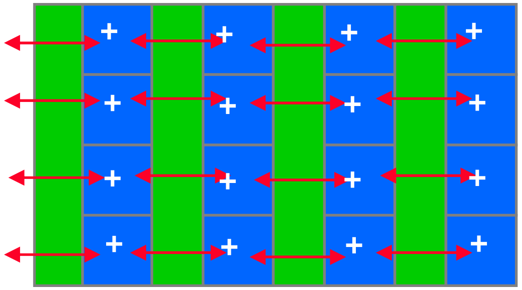
Application Example: MicroBlaze 5.0



- 1400 LUT6
- 230 Dhrystone Mips
- > 200 fit in V5

Future Proof Architecture

- Parallelism
  - Performance & Power
- Distributed Memory
  - Data transfer bottleneck
- Regular
  - Manufacturability
  - Redundancy
- Scalable
  - Future Proof
- 2010
  - 5 cent/32bit MB
  - 2$ for 1 Mgates



Arithmetic/Logic & Memory

"If FPGAs didn't exist today, people would have to invent them…"

FPGA for Embedded Systems

- An embedded system is a system that
  - has a complex <u>concurrent</u> behavior
  - is characterized by <u>stringent timing</u> requirements
  - has <u>non-trivial communication</u> between its components and the rest of the world

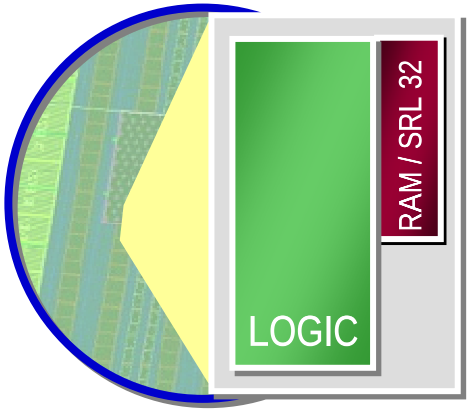
Outline

- Modern FPGA
- FPGA programmable platform
- Programming the FPGA
- Conclusions

FPGA Memory Options

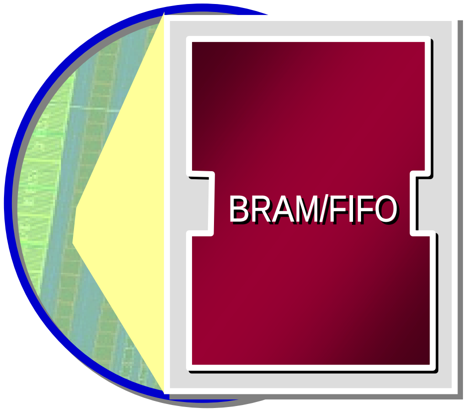
Choose the Right Memory for the Application

Distributed RAM/SRL32



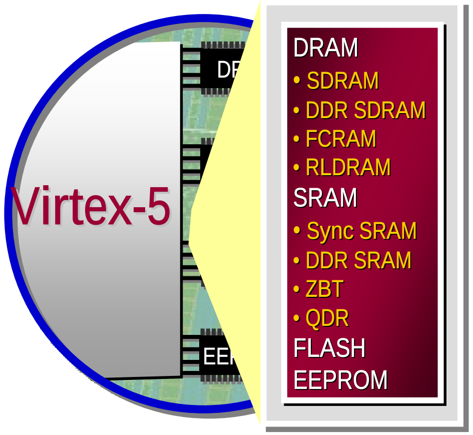
- Very granular, localized memory
- Minimal impact on logic routing
- Great for small FIFOs

On-chip BRAM/FIFO



- Efficient, on-chip blocks
- Flexible + optional FIFO logic
- Ideal for mid-sized FIFOs/buffers

Fast Memory Interfaces



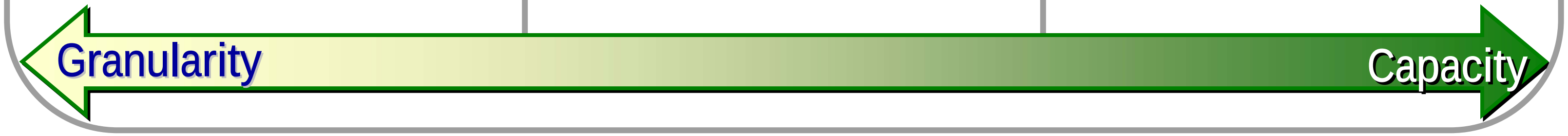
DRAM
- SDRAM
- DDR SDRAM
- FCRAM
- RLDRAM

SRAM
- Sync SRAM
- DDR SRAM
- ZBT
- QDR

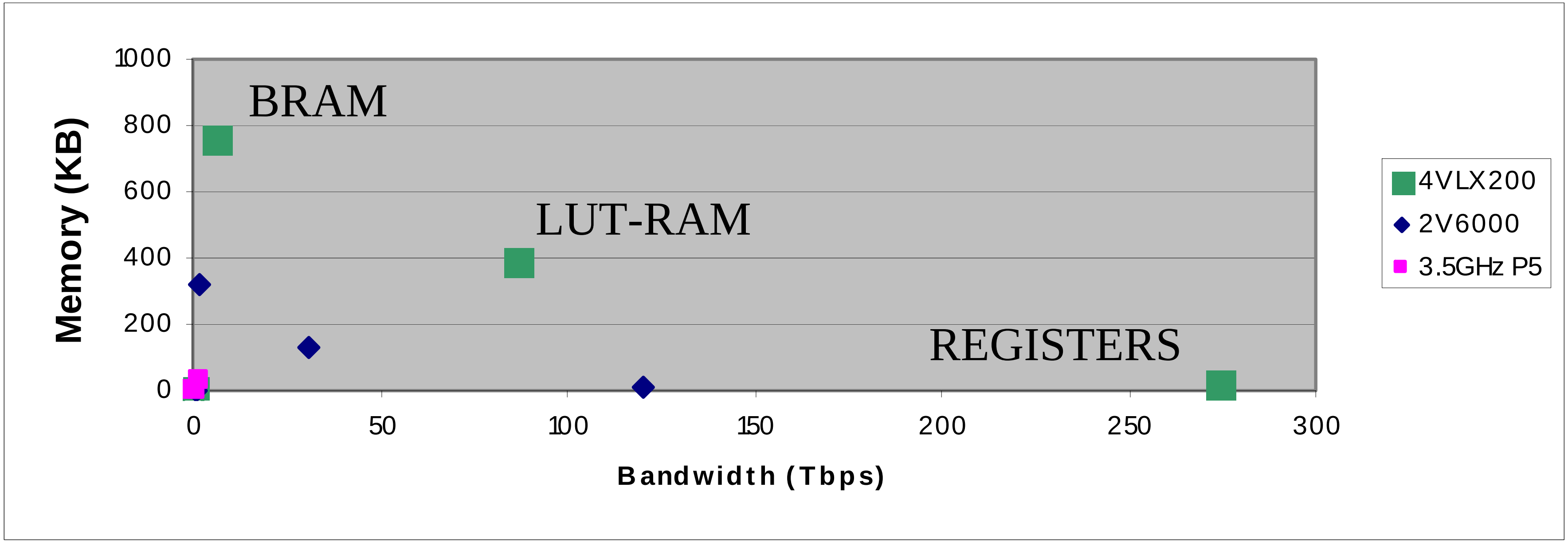
FLASH

EEPROM

- Cost-effective bulk storage
- Memory controller cores
- Large memory requirements



Memory Bandwidth Envelope

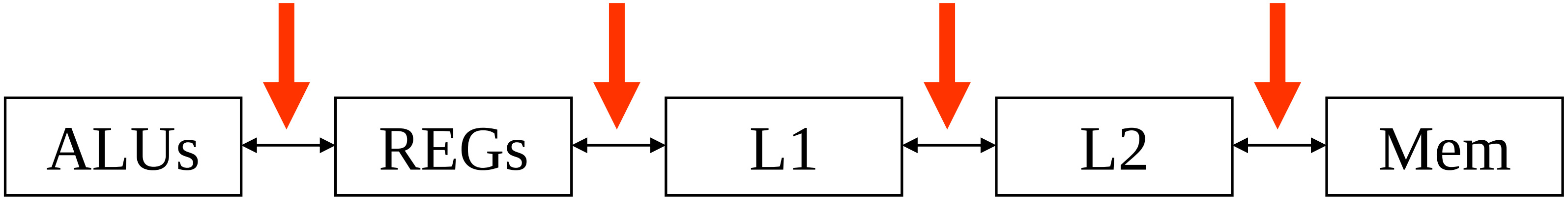


Intel; Xilinx

- Bandwidth to Registers: 500x that of a processor registerfile
- Bandwidth to LUTrams: 50x that of L1 cache of processor
- Bandwidth to BRAMS: 5x that of L1 to L2 cache of a processor

Programmable interconnect

- Can connect compute and registers, small memory and larger memory **arbitrarily**
- 80% of the FPGA resource, but often neglected as the key differentiator
- Contrast this with processors: 4 pre-specified architectural (von Neumann) bottlenecks.

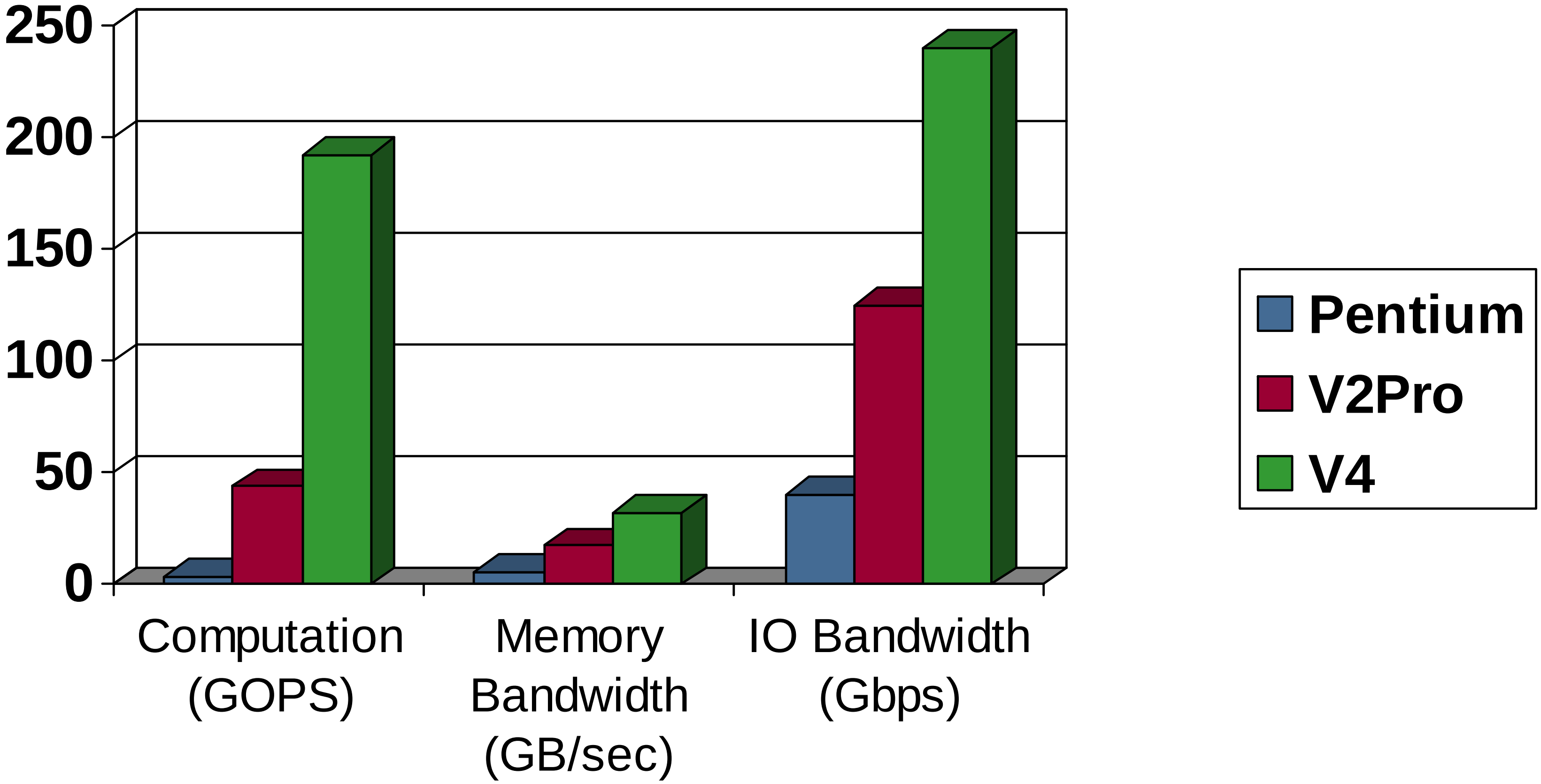


FPGA vs Microprocessor

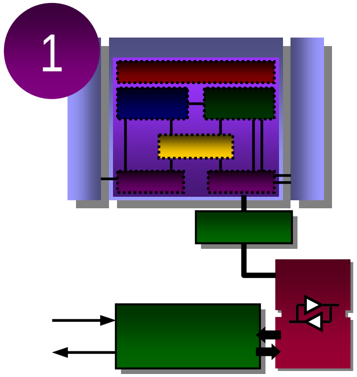
	Microprocessor Itanium 2	FPGA Virtex 2VP100
Technology	0.13 Micron	0.13 Micron
Clock Speed	1.6GHz	180MHz
Internal Memory Bandwidth	102 GBytes per Sec	7.5 TBytes per Sec
# Processing Units	5 FPU(2MACs + 1FPU) + 6 MMU + 6 Integer Units	212 FPU or 300+ Integer Units or ……….
Power Consumption	130 WATTS	15 WATTS
Peak Performance	8 GFLOPs	38 GFLOPS
Sustained Performance	~2 GFLOPs	~19 GFLOPS
I/O / External Memory Bandwidth	6.4 GBytes/sec	67 GBytes/sec

Courtesy Nallatech

High Performance Compute

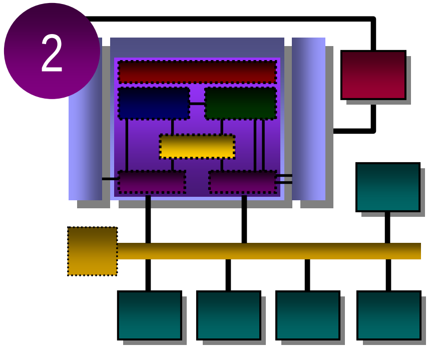


Processor Use Models



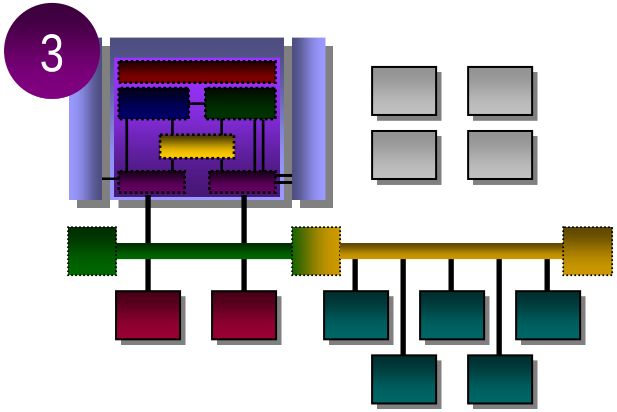
State Machine

- Lowest Cost, No Peripherals, No RTOS & No Bus Structures
- VGA & LCD Controllers
- Low/High Performance



Microcontroller

- Medium Cost, Some Peripherals, Possible Bus Structure
- Control & Instrumentation
- Moderate Performance

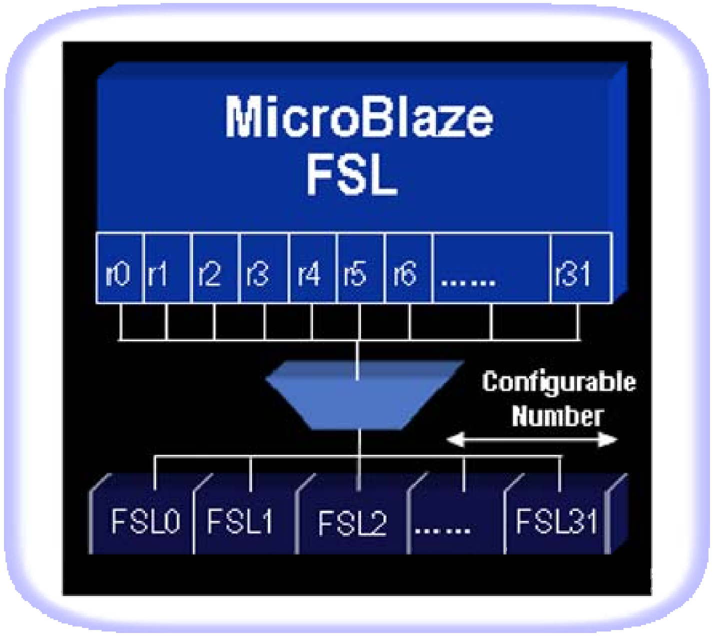


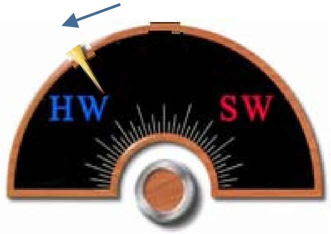
Custom Embedded

- Highest Integration, Extensive Peripherals, RTOS & Bus Structures
- Networking & Wireless
- High Performance

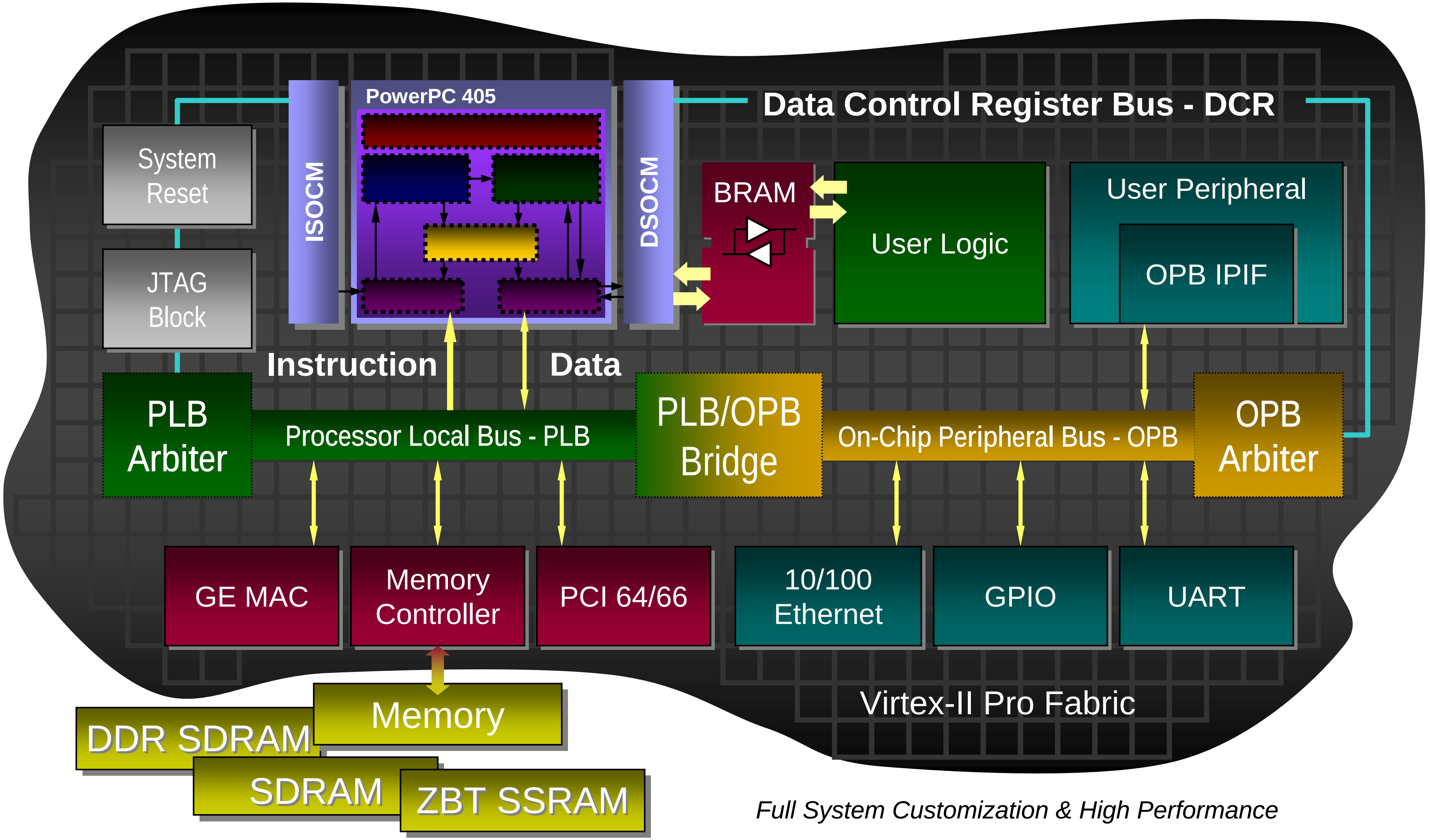
Application-Specific Hardware Acceleration

- When the processor core begins to reach software task capacity, then Fabric Acceleration to the rescue
  - – Use Fast Simplex Link (FSL) to interface to customer-defined accelerators
  - – Enables dramatic improvements in performance



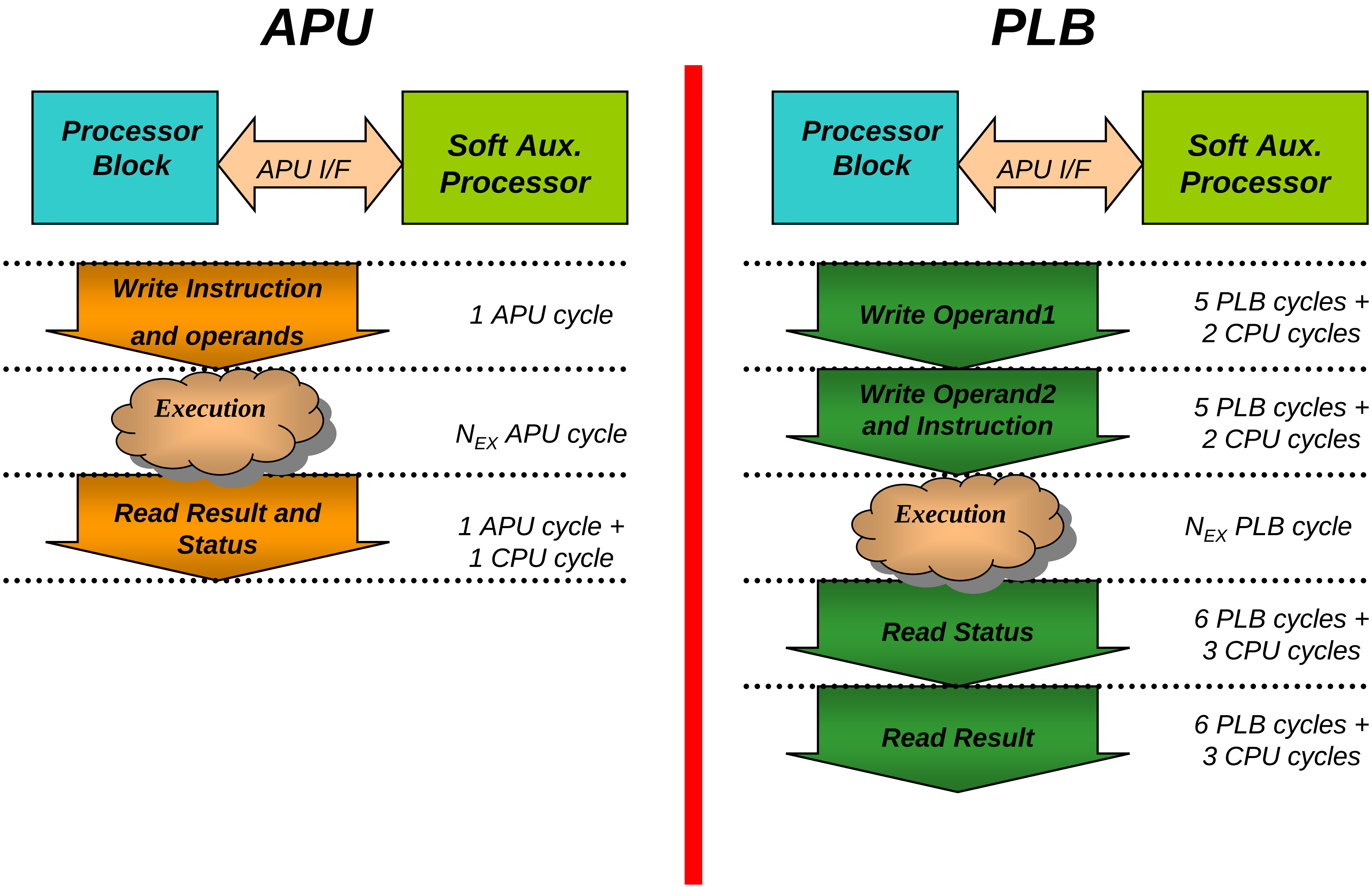


PowerPC Architecture

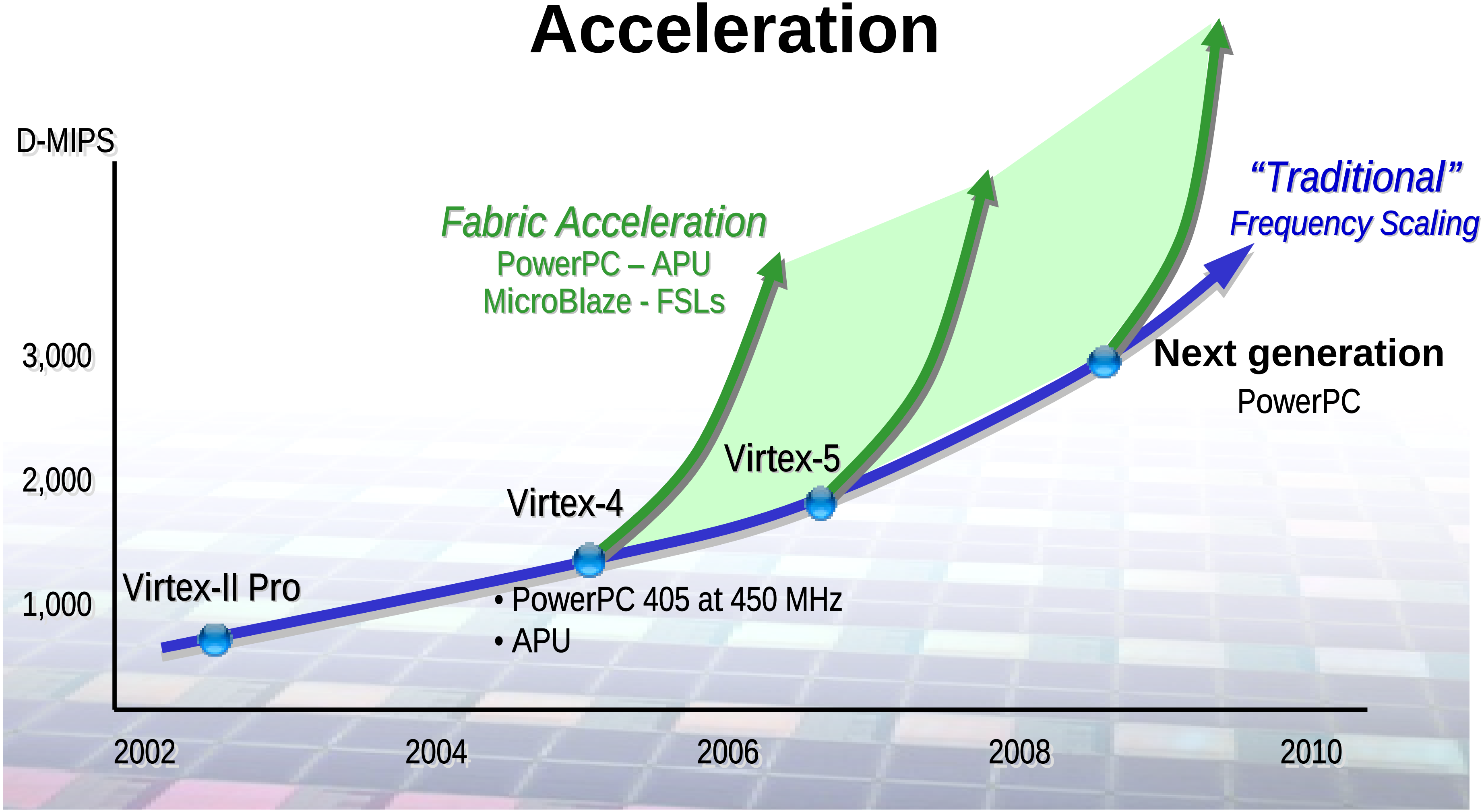


Full System Customization & High Performance

Comparison with Traditional Bus-based

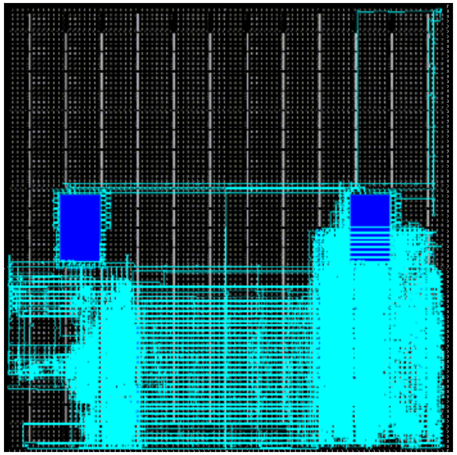


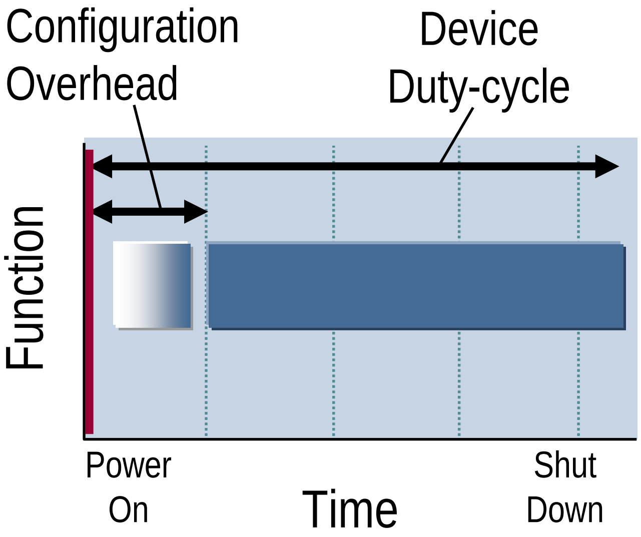
Processor Performance and Fabric Acceleration



Reconfigurable System

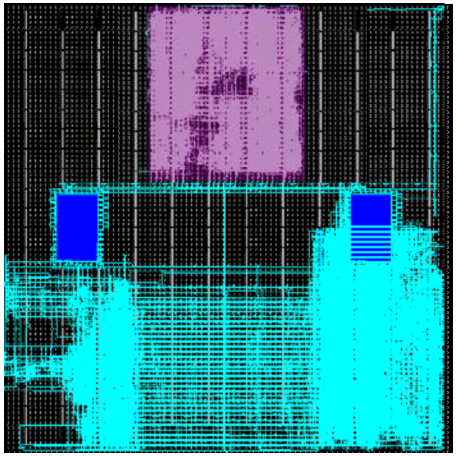
- Fixed configuration
  - Data loads from PROM or other source at power on
  - Configuration fixed until the end of the FPGA duty cycle
- Used extensively during traditional design flow
  - Evaluate functionality of design as it is developed

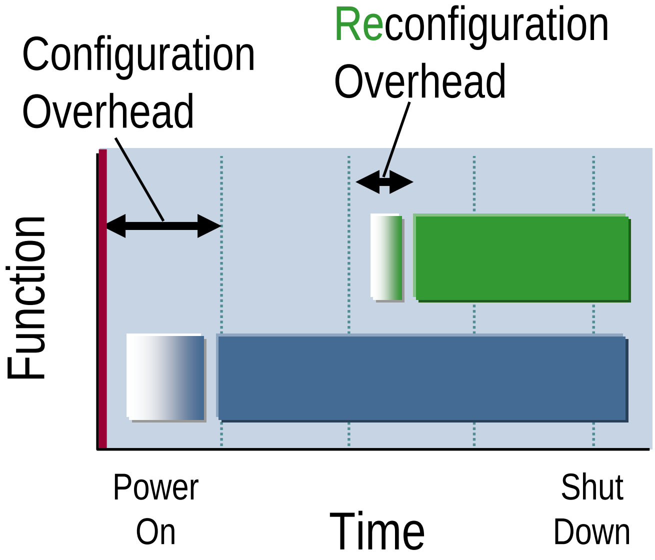




Dynamic Partial Reconfiguration

- A subset of the configuration data changes…
  - But logic layer continues operating while configuration layer is modified…
  - Configuration overhead limited to circuit that is changing…





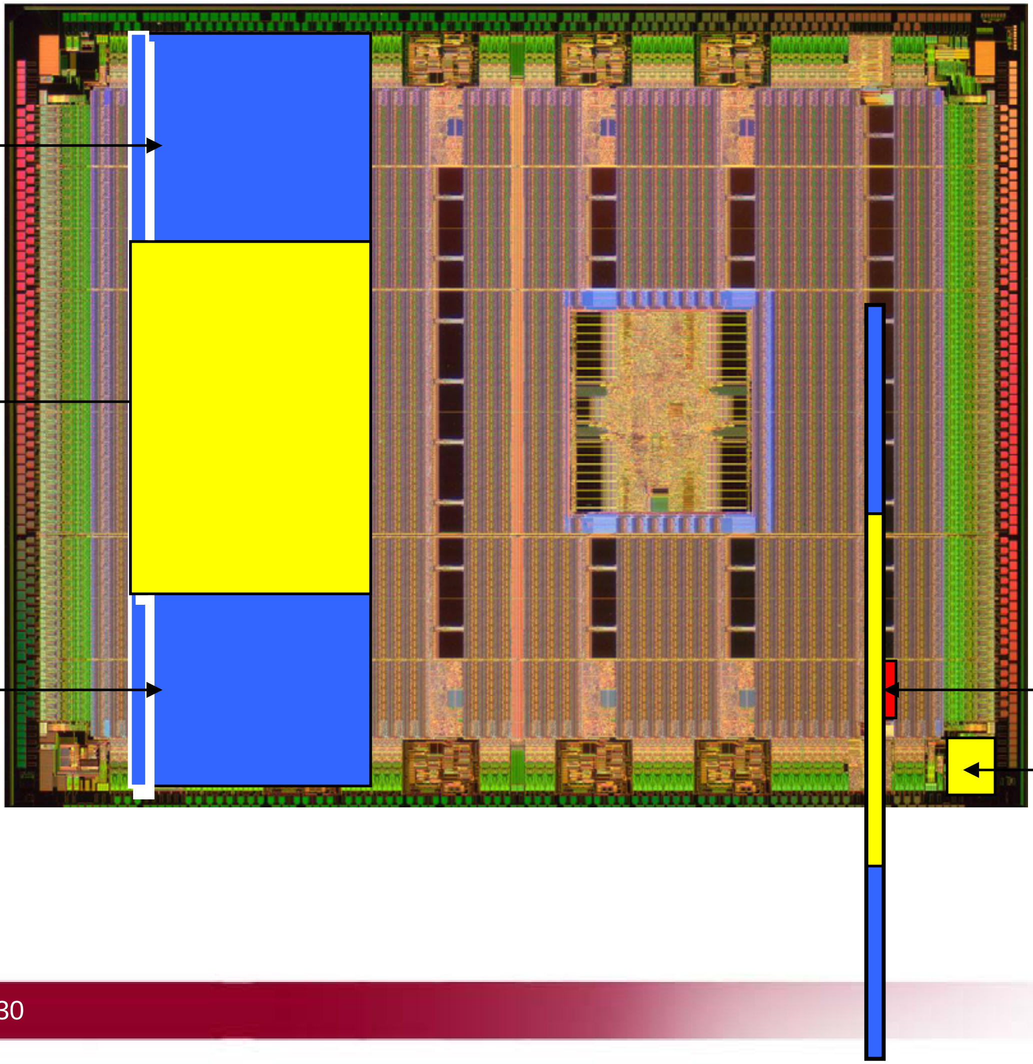
Read / Modify / Write

1. Read back frame and load into BRAM

2. Modify configuration data in BRAM

3. Write modified frame to configuration memory

4. Repeat "Read- Modify-Write" sequence for all frames



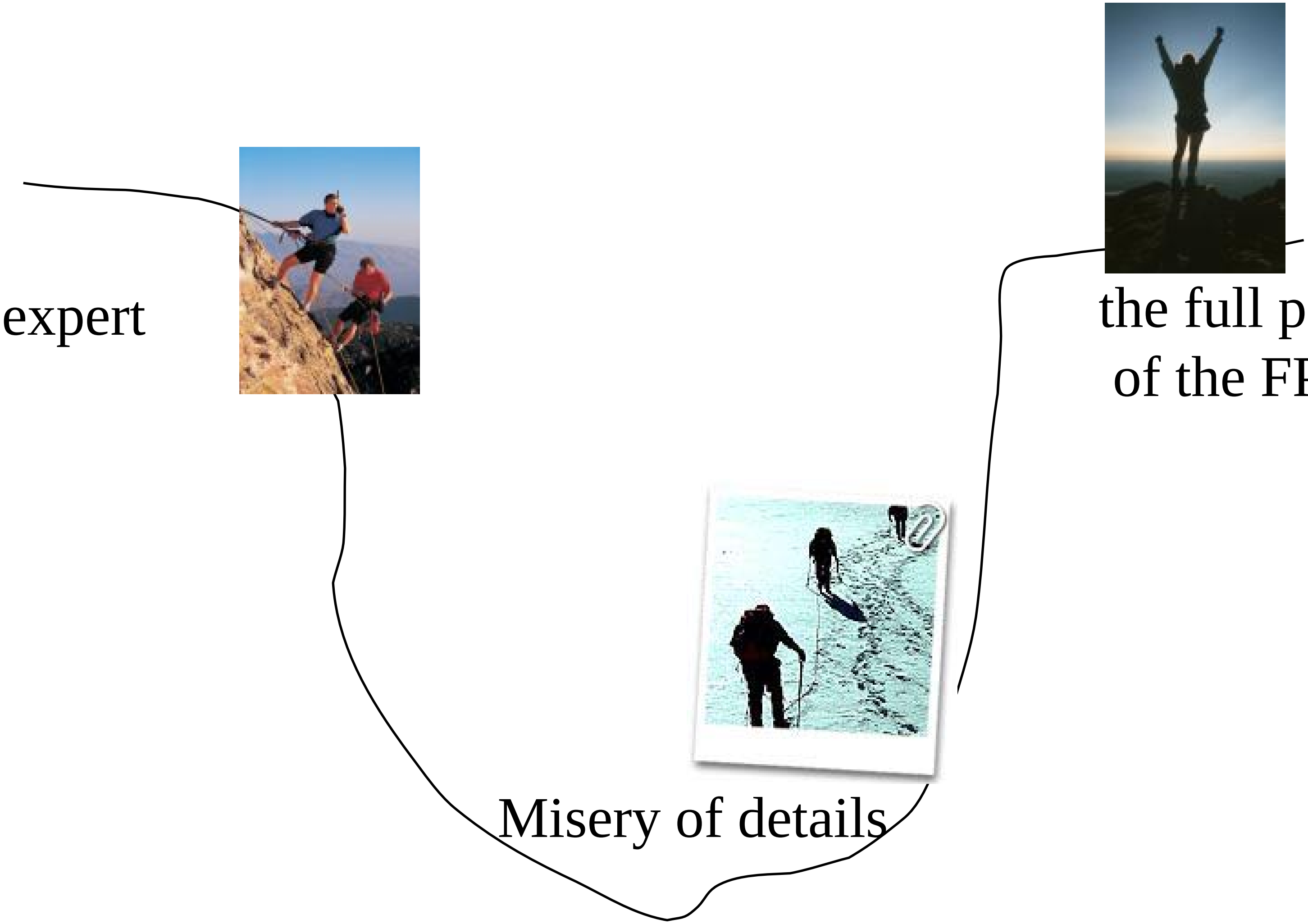
Outline

- Modern FPGA
- FPGA programmable platform
- Programming the FPGA
- Conclusions

Programming FPGAs

Just a Matter Of Software

Bridging the Gap

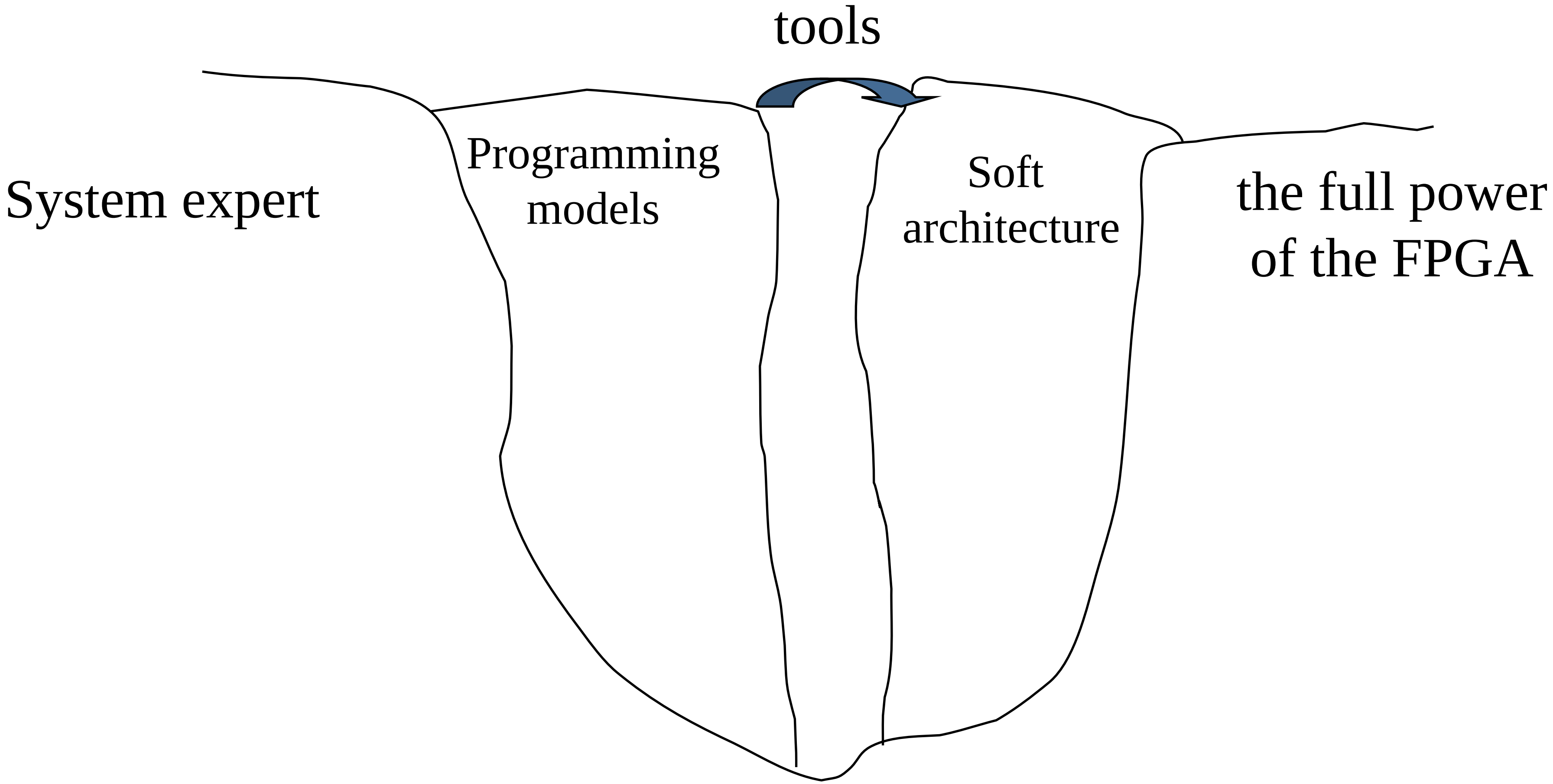


System expert

the full power of the FPGA

Misery of details

Domain Specific Flow



Domain Specific Models

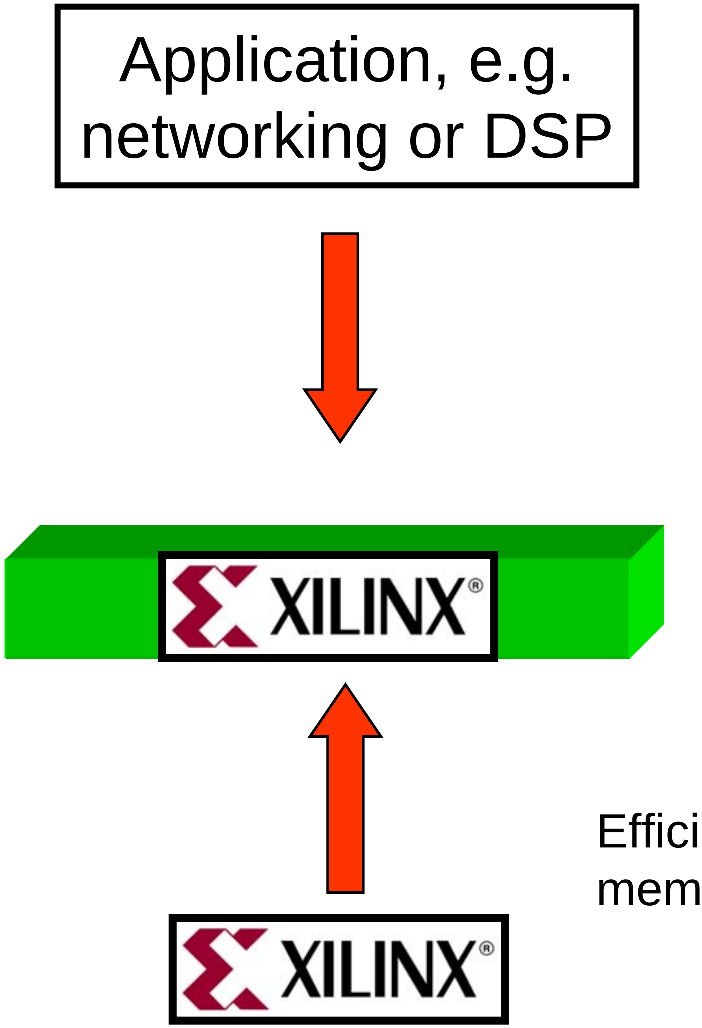
Programming models

- Network Processing: concurrent application of rules to packets
- Digital Signal Processing: concurrent compute on streams of samples, e.g. video pixels
- High Performance Computing: concurrent compute with random access on datasets; compute with floating point and complex numbers

Architecture components

- Network Processing: FSM, micro-coded datapath, processors, pipelines, wide datapaths
- Digital Signal Processing: buffers with flow control, FSM, processors, synthesized expressions, fixed point
- High Performance Compute: Partition the algorithm, specialized instructions, small efficient cache components, floating point units

Bridging the gap



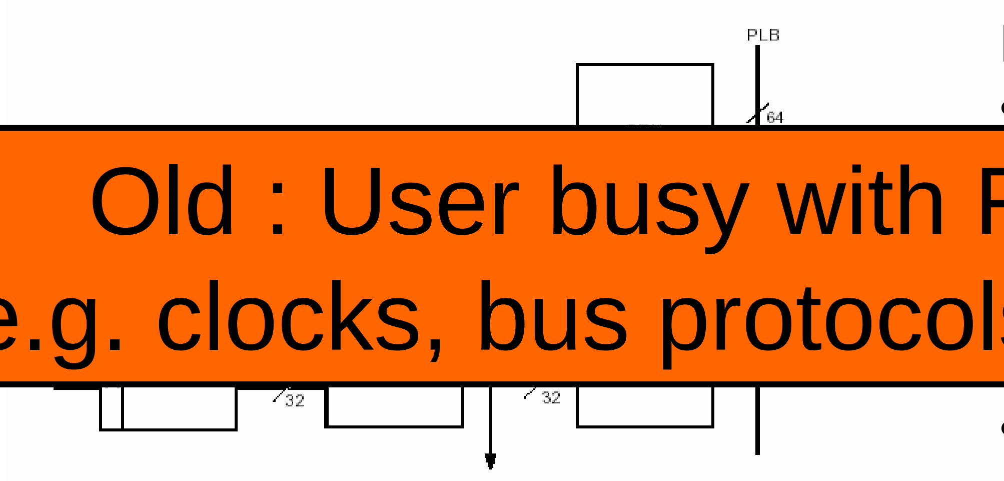
Application, e.g. networking or DSP

- **Domain-specific data model and programming language**
- **API to access features of the domain specific soft architecture**

Hyper-programmed ***soft architecture***

Efficiently exploit logic, immersed IP, processing blocks, memory, interconnection, and programmability of FPGA

Abstracting away FPGA detail



Before:

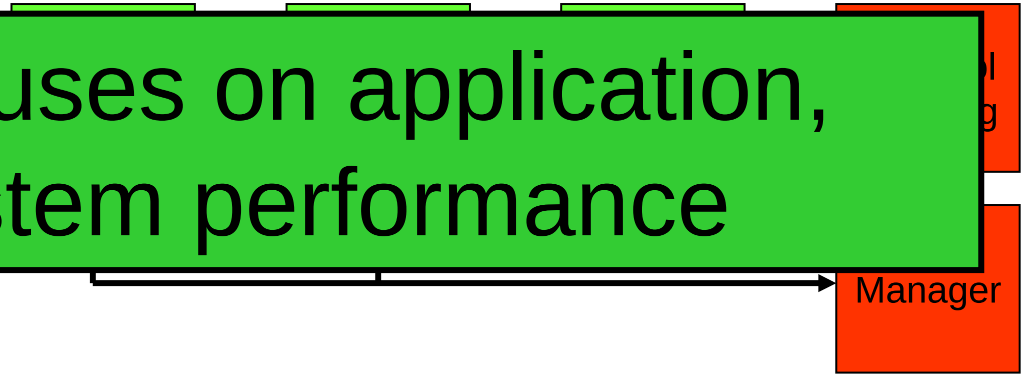
- Complex single C program

nd

- Tailored inter-block interfaces

Old : User busy with FPGA details
e.g. clocks, bus protocols, memory map

A



New : User focuses on application,
including system performance

Challenge

- Specifying complex computational algorithms in a way that...

... is productive,

... permits efficient implementation on FPGAs,

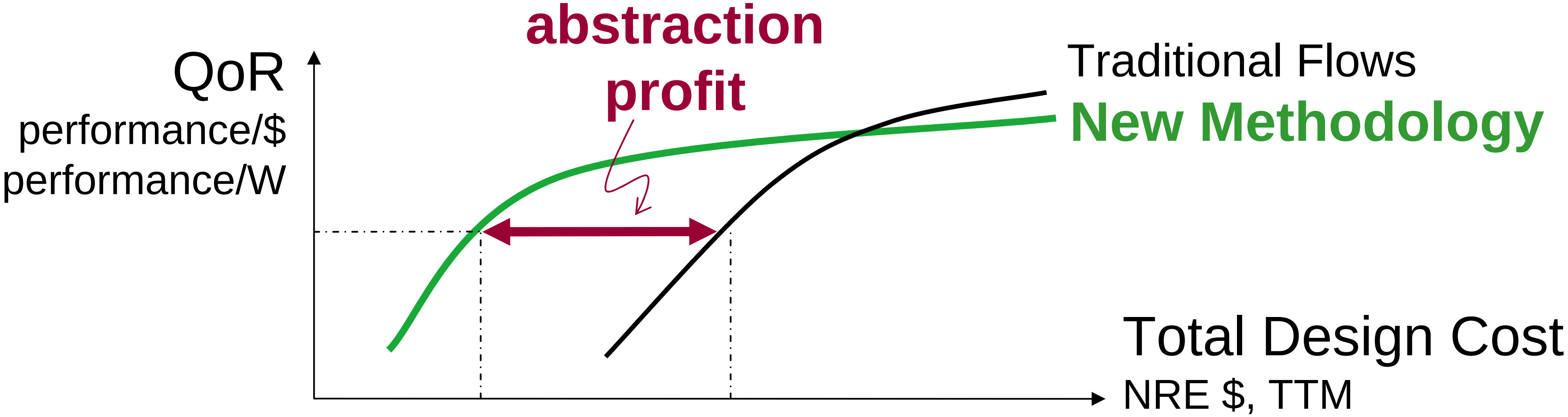
... allows leveraging enormous concurrency of an FPGA,

... provides portability

  - across alternative implementations (e.g. fabric vs processor)
  - across different devices

Productivity

- Quality of result (QoR) is a design constraint
  - Performance, power, cost budgets make QoR a design constraint
- The real problem is to meet the QoR target and minimize:
  - Non-recurring engineering costs (NRE)
  - Time-to-market (TTM)
- Methodology saves design cost by enabling
  - Design of portable, retargetable, composable IP blocks
  - Rapid design space exploration and system composition

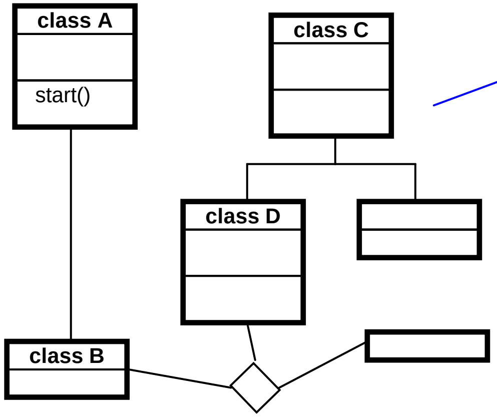


Technical Challenges

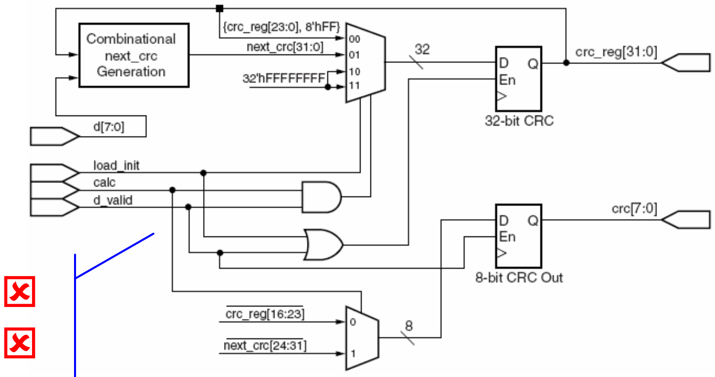
- Methodology must address...
  - implementability, concurrency, portability

- ---a programming model that...
  - ... is simple to understand.
  - ... matches the application domain.
  - ... exposes essential architectural detail, hides the rest.
  - ...induces the programmer to make the right choices.

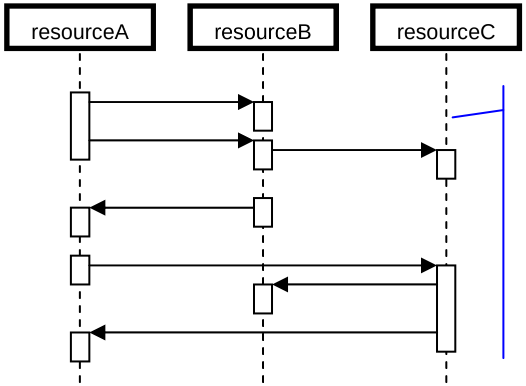
Combine the Best of Both Worlds
Software - Hardware



☑ Encapsulation
☑ Abstraction
☑ Portability
☑ Re-use

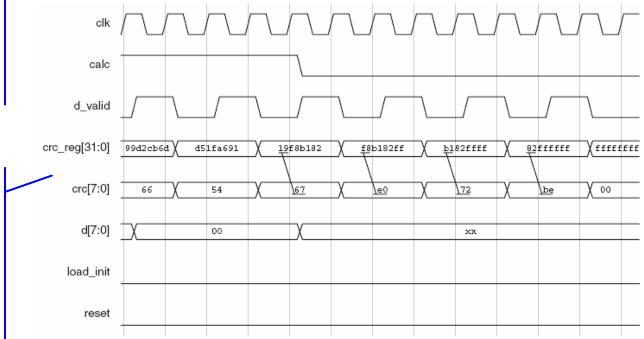


Implementation Detail ☒
Control Logic ☒
Interface Glue ☒
Concurrency ☑
Communication ☑
Architecture ☑



☑ Events
☑ Protocols
☑ Ordering
☒ Sequential execution

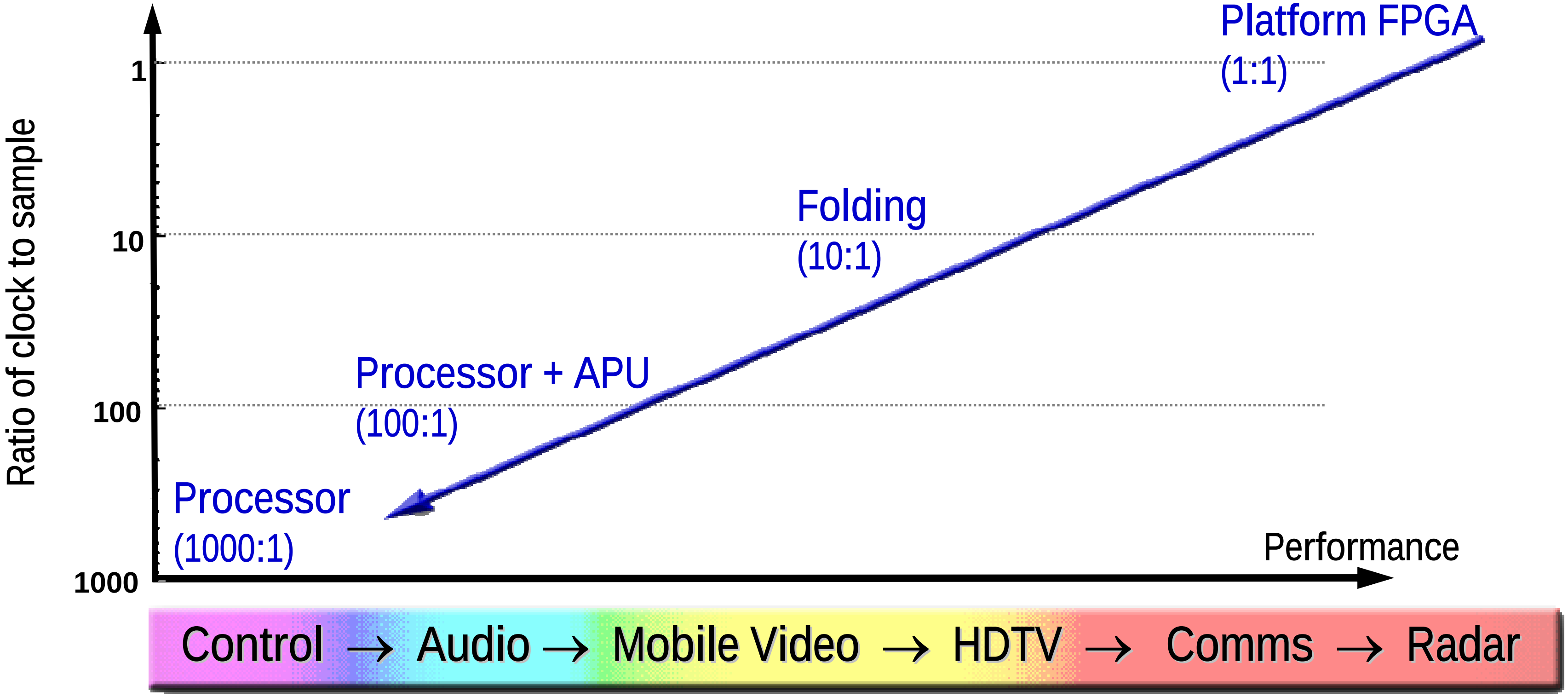
Clocks ☒
Signals ☒
Timing ☒



Combining the strengths of both paradigms results in a radical improvement in hardware/software system design productivity.

DSP solution space

Best Clock-to-Sample Ratio



Spectrum of Applications

"Massive parallelism often allows FPGAs to handle data rates much higher than what DSPs and general-purpose processors can manage, and in today's world of rapidly evolving applications and standards FPGAs' programmability is an advantage over hard-wired solutions." *- Amit Shoham, BDTI, June 15, 2005*

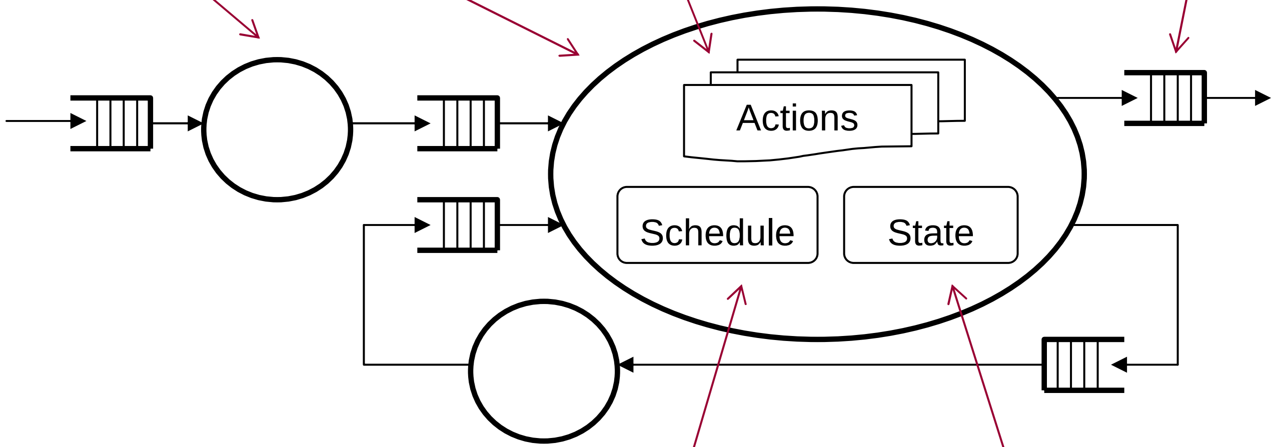
**Inside DSP on Tools: FPGA Tools Bridge Gap Between Algorithm and Implementation, "insidedsp.eetimes.com", June 15, 2005*

DSP : Actor/Dataflow Programming

actors

guarded atomic actions

point-to-point, buffered token-passing connections



autonomous schedule

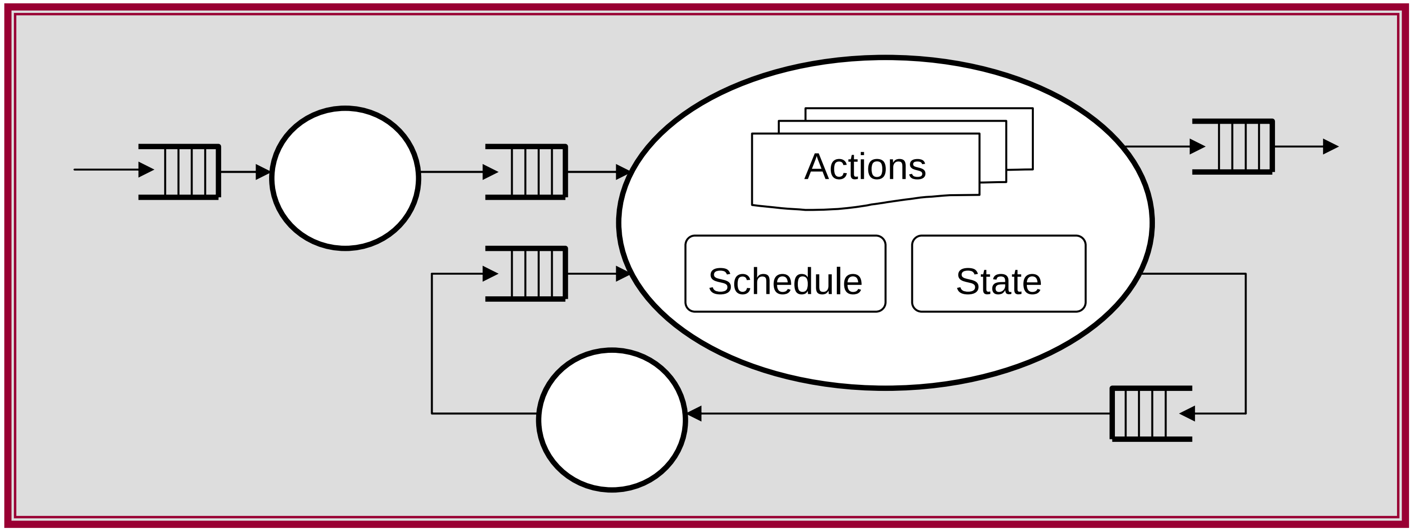
encapsulated state

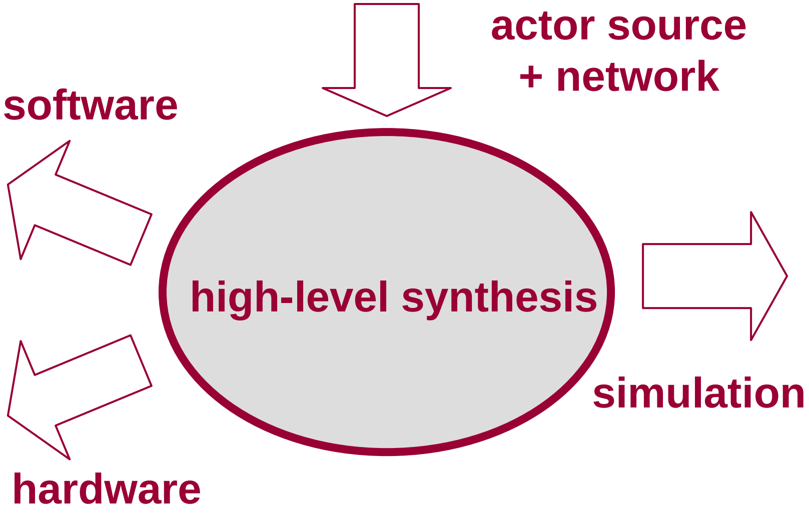
UC Berkeley (Janneck et al)

Benefits of the Actor Model

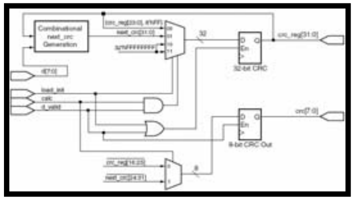
- Dataflow is a natural concept in DSP.
- Explicit description of concurrency and disciplined access to shared state make design and debug of concurrent systems feasible.
- Complete abstraction of time.
- Extensive abstraction of control.
- Same description can target HW and SW implementations.
- Can be visualized easily
- Works naturally with run-time reconfiguration.

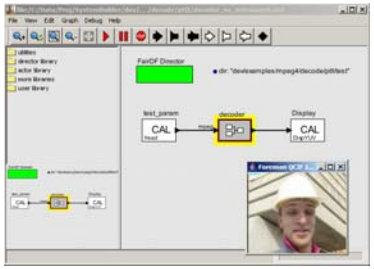
Actor/Dataflow Implementation





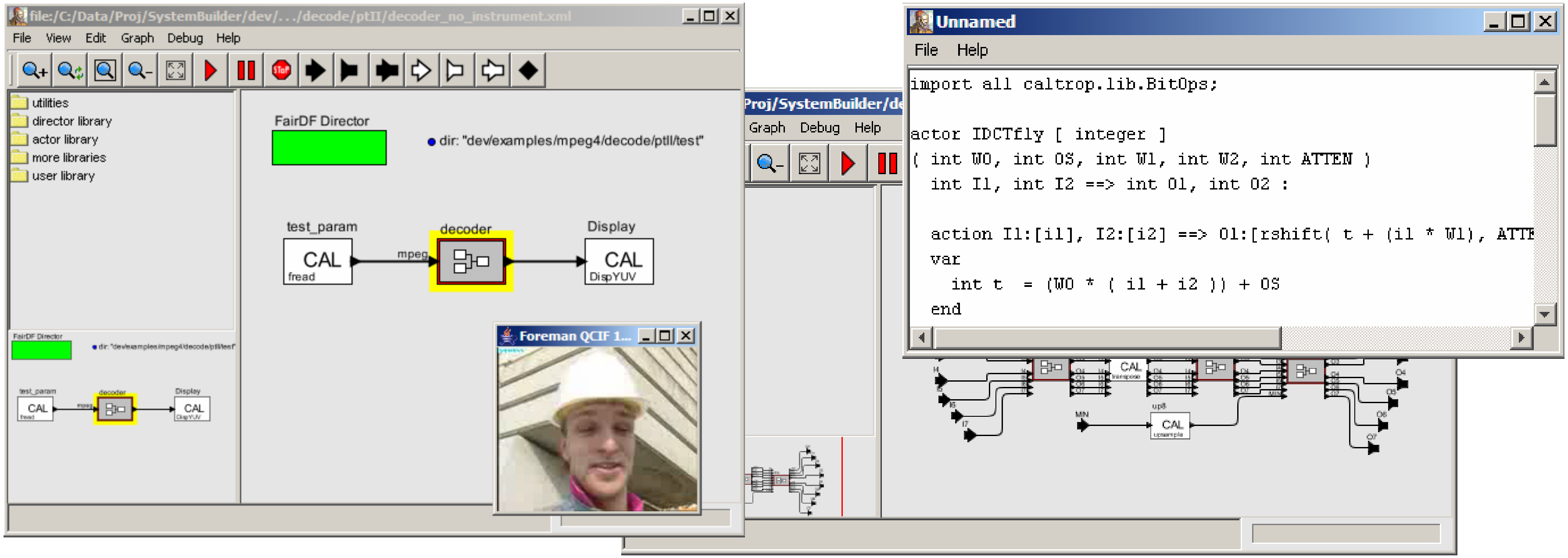
```
class MyActor
{ schedule();
  readPort( portNum );
  writePort( portNum );
}
```





Concurrent Model

- Model entered as
  - Hierarchical, structural composition of actors.
  - Textual code for actor contents.
- Verified with dataflow simulation (Ptolemy-II).

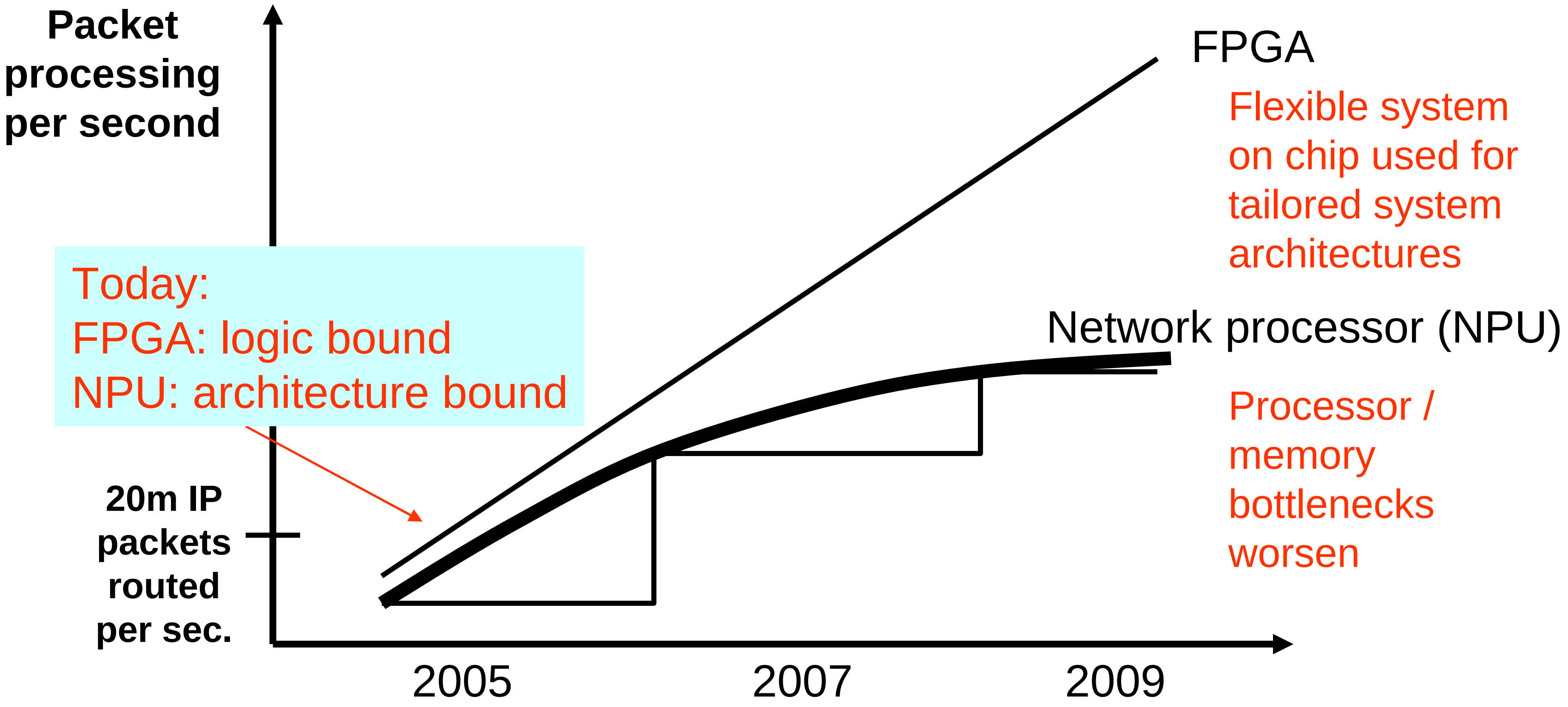


```
import all caltrop.lib.BitOps;

actor IDCTfly [ integer ]
( int W0, int OS, int W1, int W2, int ATTEN )
  int I1, int I2 ==> int O1, int O2 :

  action I1:[i1], I2:[i2] ==> O1:[rshift( t + (i1 * W1), ATTE
  var
    int t  = (W0 * ( i1 + i2 )) + OS
  end
```

Network Processing Solutions

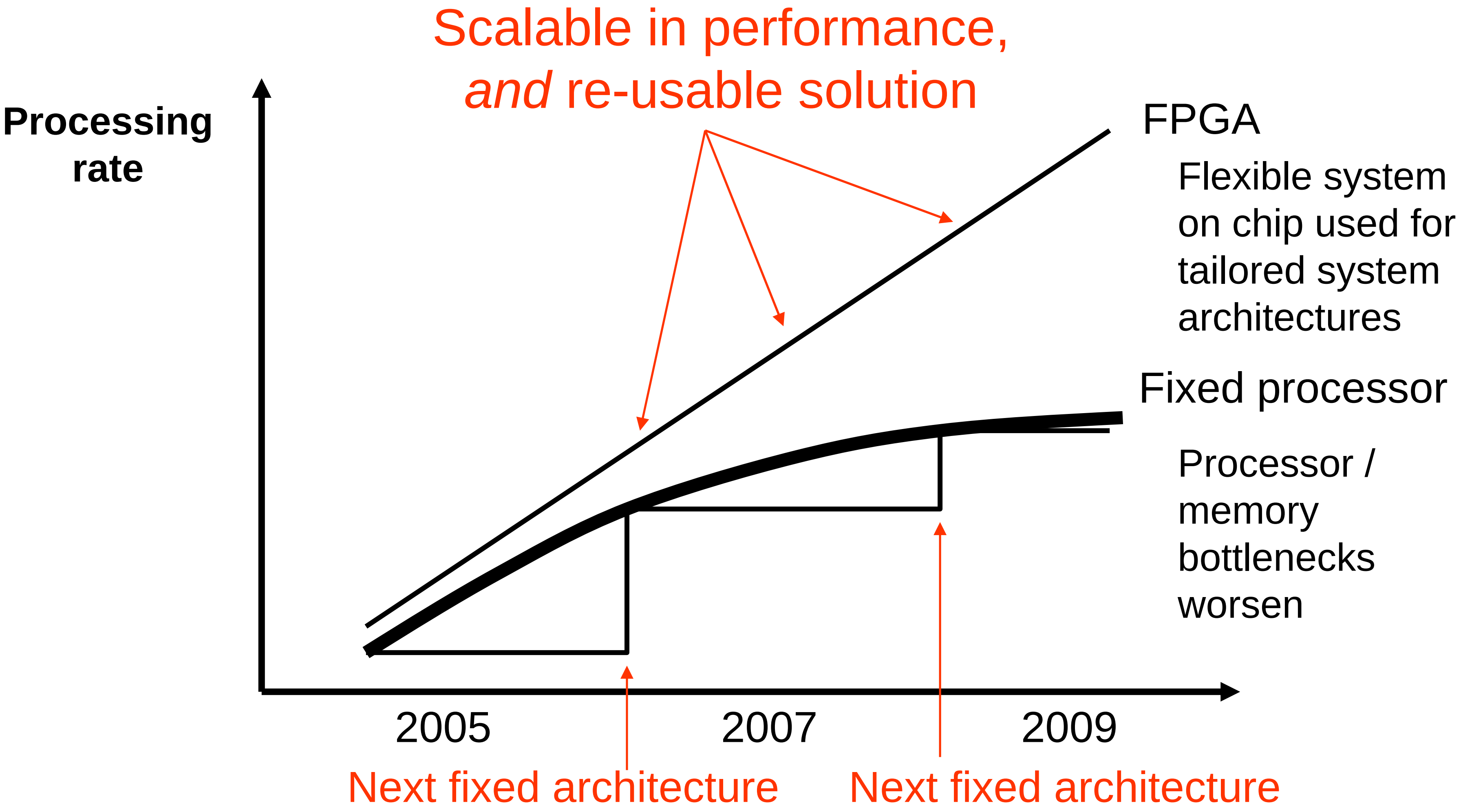


Today:
FPGA: logic bound
NPU: architecture bound

Flexible system on chip used for tailored system architectures

Processor / memory bottlenecks worsen

Flexibility and scalability

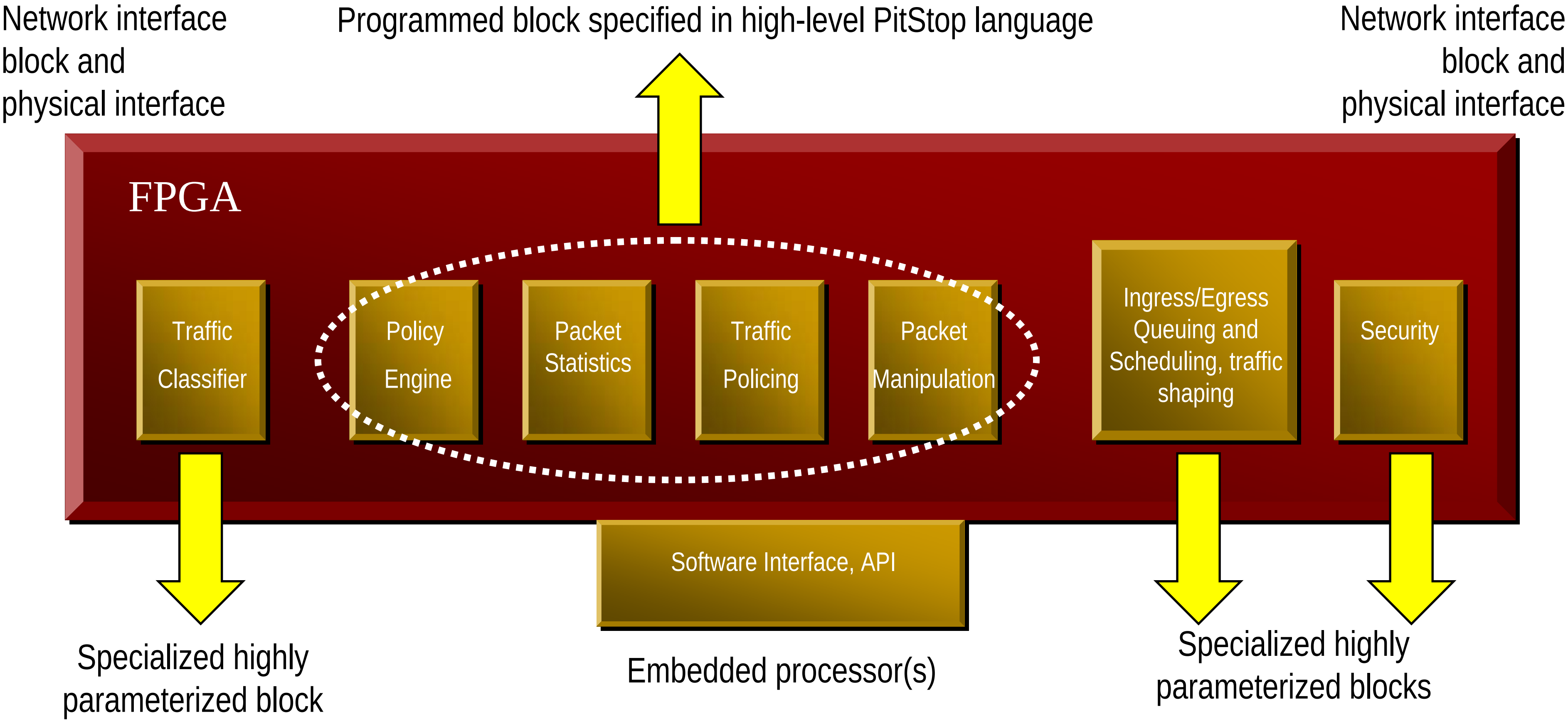


No re-use, architecture dependent

Flexibility opportunity

- “Despite the modest size of the NPU market, the recent trend in this market has surprisingly been segmentation. Vendors are discovering that a single general-purpose network processor cannot meet the needs of a broad set of applications. New products, and even some vendors, now focus on a single segment: access, metro, or enterprise.” - The Linley Group, January 2006

Example: typical line card



Network interface block and physical interface
Programmed block specified in high-level PitStop language
Network interface block and physical interface
FPGA
Traffic Classifier
Policy Engine
Packet Statistics
Traffic Policing
Packet Manipulation
Ingress/Egress Queuing and Scheduling, traffic shaping
Security
Software Interface, API
Specialized highly parameterized block
Embedded processor(s)
Specialized highly parameterized blocks

Blocks (plus other glue) assembled into system, by compilation of high-level Click language description

Network Packet Processing : Object Oriented Programming

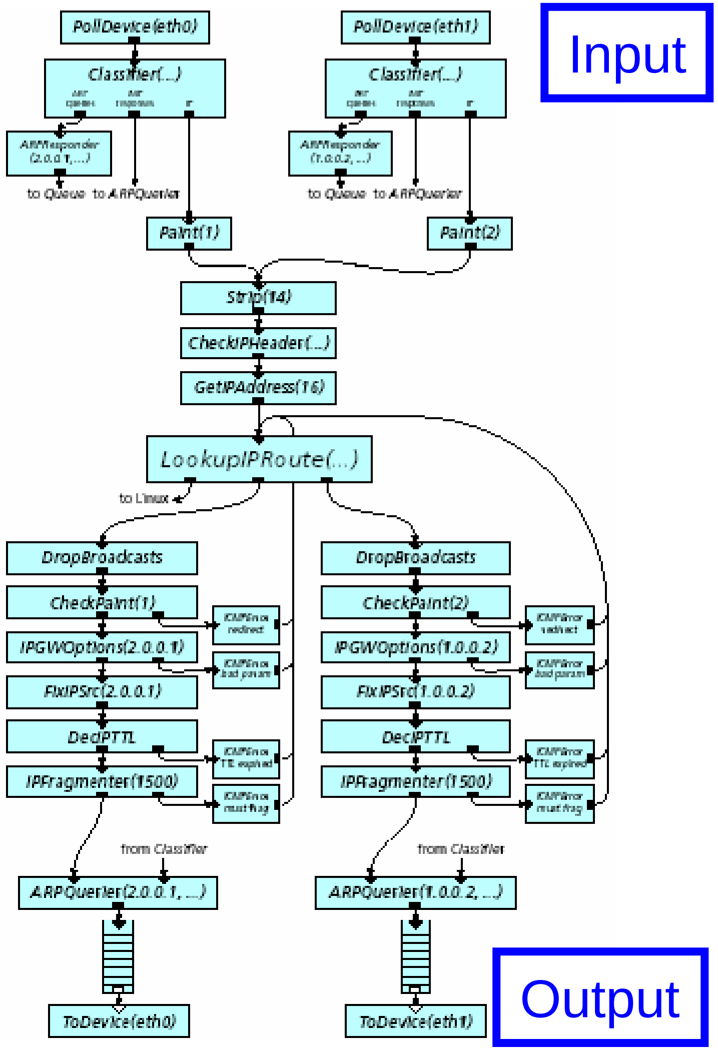
E.g. Click programming

Each block is described as a Click *element*

Connections are made between elements, forming a graph

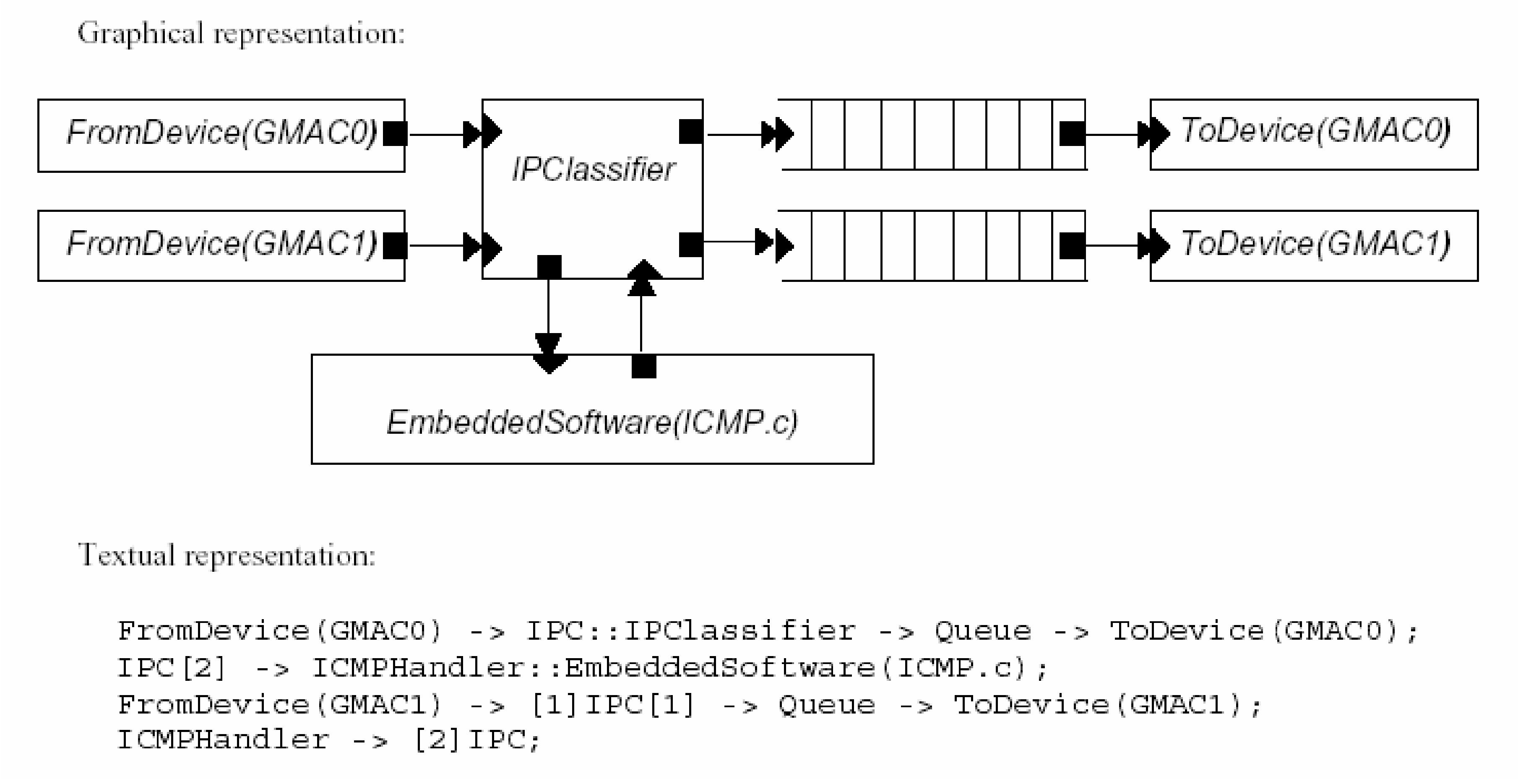
Can be used to describe designs at different granularities:
from coarse-grain blocks to fine-grain blocks

MIT (Kohler et al)



Compact description in Click

IP router with ICMP offload



Graphical representation:

Textual representation:

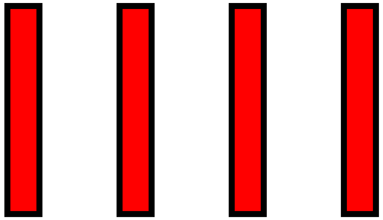
```
FromDevice(GMAC0) -> IPC::IPClassifier -> Queue -> ToDevice(GMAC0);
IPC[2] -> ICMPHandler::EmbeddedSoftware(ICMP.c);
FromDevice(GMAC1) -> [1]IPC[1] -> Queue -> ToDevice(GMAC1);
ICMPHandler -> [2]IPC;
```

Top-level architecture choice

High-level packet processing description language

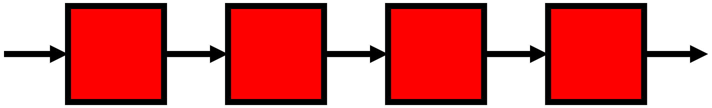
Example 1: Protocol stack handling in end system style setting

Example 2: Layer 2 packet handling in line card style setting



Collection of communicating threads implemented by logic or processor:

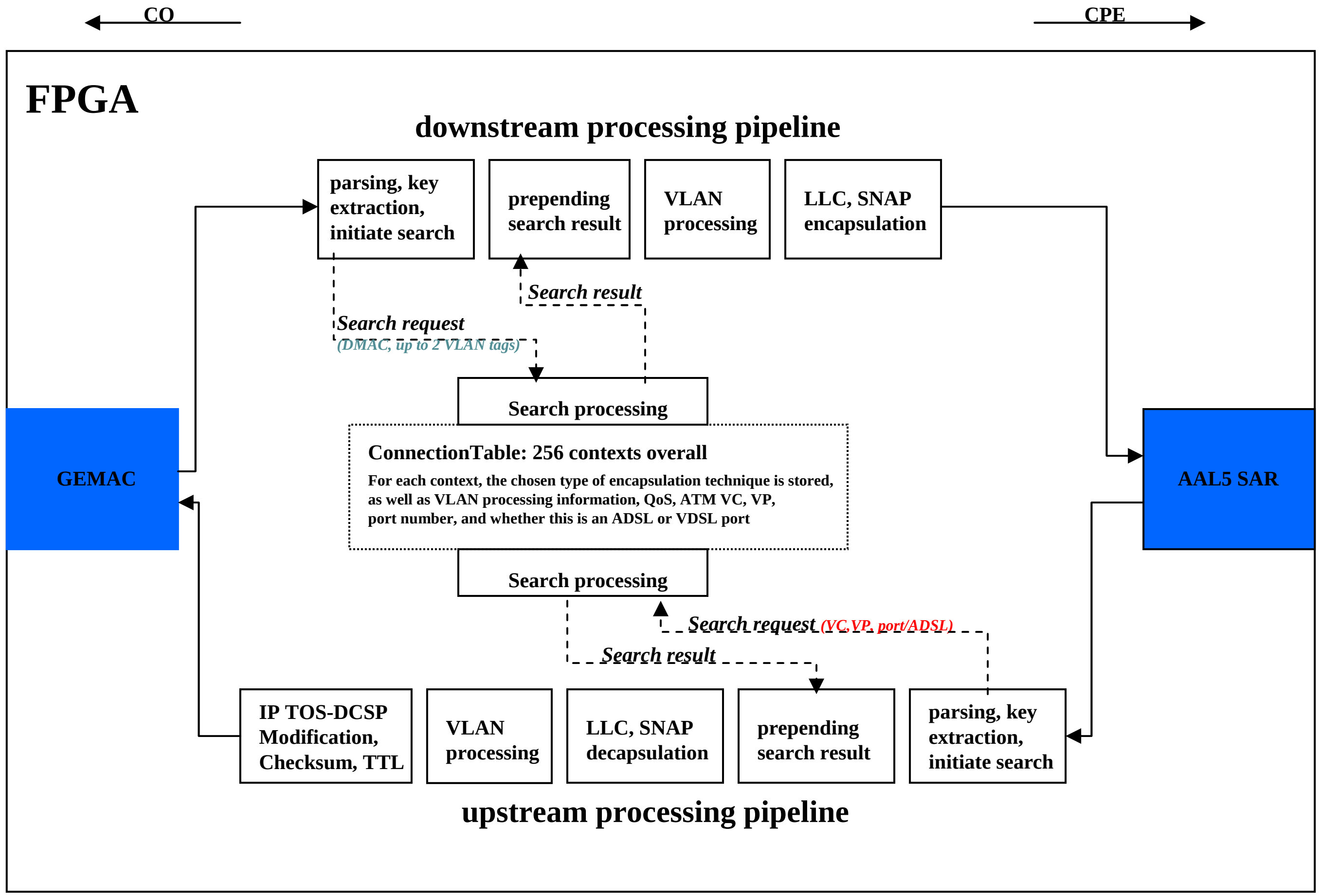
- prioritize low total latency
- … then throughput (say, 1 Gb/s)
- one thread per protocol



Blocks arranged in pipeline implemented by logic:

- prioritize throughput (say, 10+ Gb/s)
- … then latency
- staged according to dependencies

Example : IP-enabled DSLAM



CO

CPE

FPGA

downstream processing pipeline

parsing, key extraction, initiate search

prepending search result

VLAN processing

LLC, SNAP encapsulation

Search result

Search request

(DMAC, up to 2 VLAN tags)

Search processing

GEMAC

AAL5 SAR

ConnectionTable: 256 contexts overall

For each context, the chosen type of encapsulation technique is stored, as well as VLAN processing information, QoS, ATM VC, VP, port number, and whether this is an ADSL or VDSL port

Search processing

Search request (VC,VP, port/ADSL)

Search result

IP TOS-DCSP Modification, Checksum, TTL

VLAN processing

LLC, SNAP decapsulation

prepending search result

parsing, key extraction, initiate search

upstream processing pipeline

Results

Quantifiable:

• silicon cost:	*competitive (comparable to a low end NPU)*
• performance:	*easily achieve 6.4Gbps in a V4 LX25*
• power consumption:	*below 2W*

More qualitative:

- programmability & flexibility of the end solution
  high abstraction level plus FPGA flexibility
- maintainability
  high abstraction level
  hiding of implementation specifics } *offer maintainability*
- development time
  only 6-8 weeks for prototype & simulation

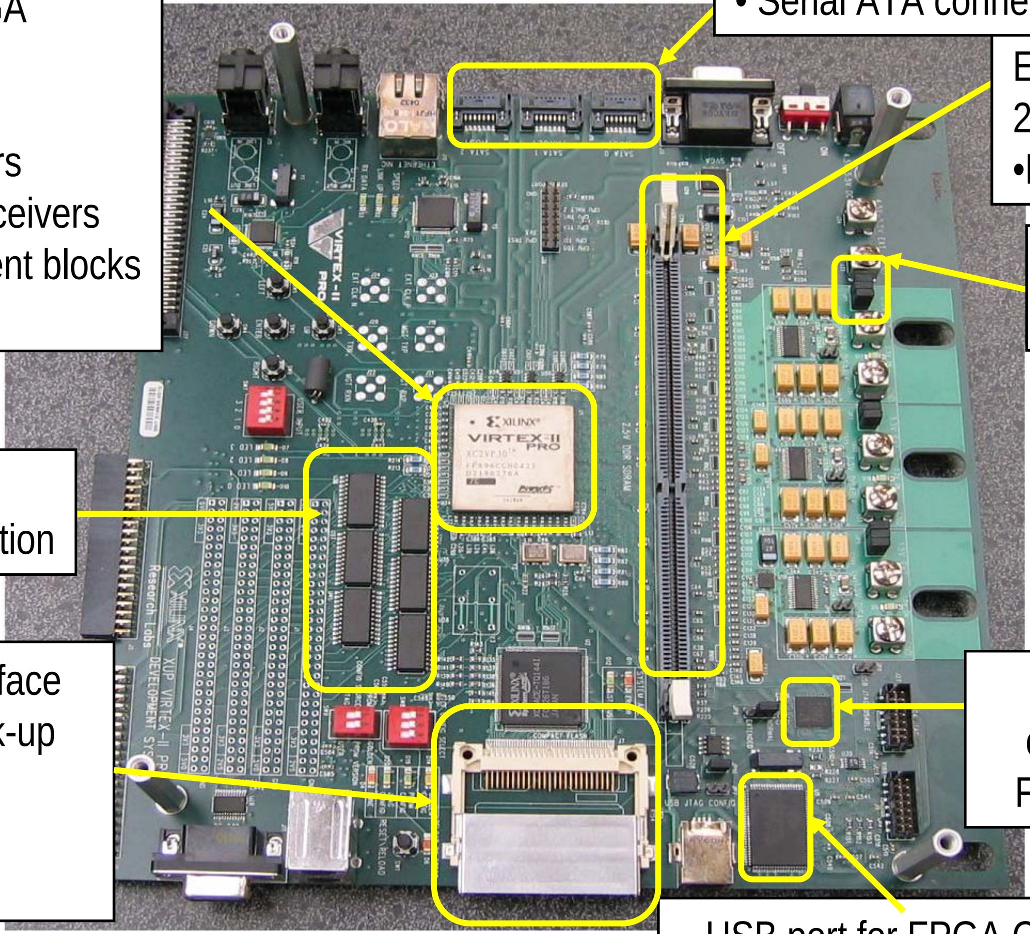
FPGA versus NPU DSLAM implementation

V4 LX25	32-bit datapath	12.5m pps	0.9 W	~$35
	128-bit datapath	50m pps	1.1 W	~$35
Low end NPU	Agere APP300	4m pps	5 W	~$35
	Intel IXP2350	2.5m pps	11 W	~$125
DSLAM specialized NPU	Infineon Convergate-C	0.5m pps	1.5 W	~$35
	Wintegra 717	0.2m pps	2.7 W	~$50
High end NPU	Intel IXP2800	30m pps	25 W	~$400
	Xelerated X11-S200	30m pps	11 W	~$295

Summary

- Domain specialist can get efficient access to FPGAs without being a hardware expert
- Compile/synthesize for a problem-specific optimized combination of logic, embedded processors, and memory
- The 80% routing in the silicon is the secret sauce to outperform fixed processing solutions
- FPGA opportunity requires new thinking and new tools

Xilinx System Workbench for Students



Virtex-II Pro XC2VP30 FPGA
- 30,816 Logic Cells
- 2448 Kbits of BRAM
- 2 PowerPC 405 processors
- 8 MultiGigabit Serial Tranceivers
- 8 Digital Clock Management blocks
- 136 18x18 multipliers

High-speed Gigabit serial I/O
- Serial ATA connectors

Expandable memory up to 2 Gigabytes
•DDR SDRAM DIMM Slot

Support for supply current monitoring

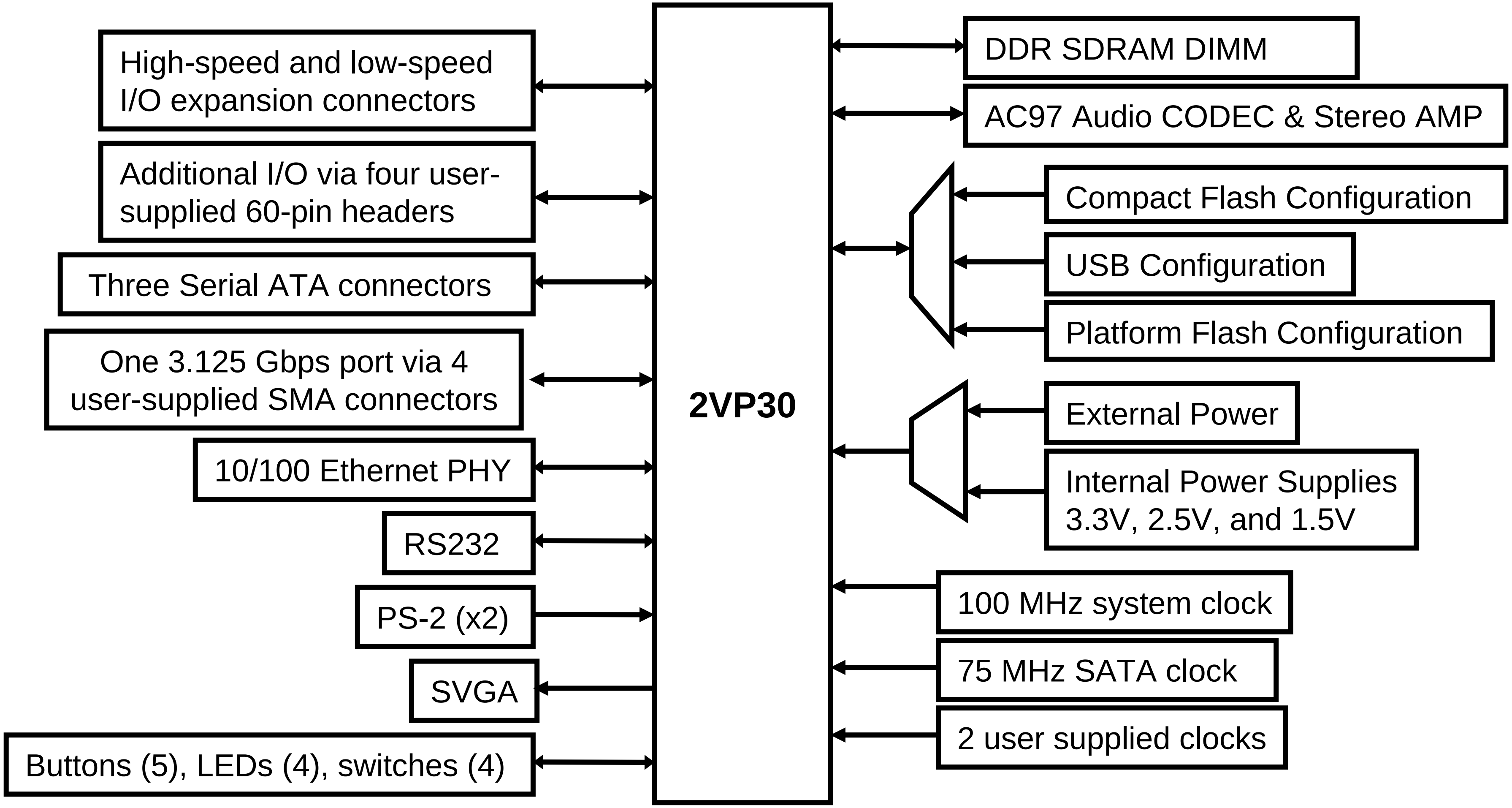
I/O under and over voltage protection

Compact Flash card interface for individual project back-up or IBM Miicrodrives with upto 8Gbit capacity

Self-test / configuration Flash memory

USB port for FPGA Configuration using standard USB cable

Block Diagram



High-speed and low-speed I/O expansion connectors
Additional I/O via four user-supplied 60-pin headers
Three Serial ATA connectors
One 3.125 Gbps port via 4 user-supplied SMA connectors
10/100 Ethernet PHY
RS232
PS-2 (x2)
SVGA
Buttons (5), LEDs (4), switches (4)
2VP30
DDR SDRAM DIMM
AC97 Audio CODEC & Stereo AMP
Compact Flash Configuration
USB Configuration
Platform Flash Configuration
External Power
Internal Power Supplies 3.3V, 2.5V, and 1.5V
100 MHz system clock
75 MHz SATA clock
2 user supplied clocks

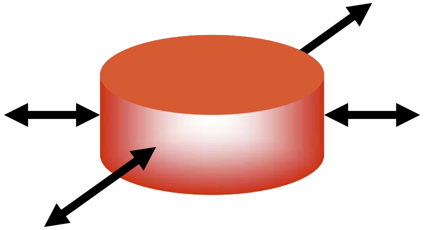
www.xilinx.com/univ

- Online donation forms for Xilinx SW products
- Purchase university boards
- Donations from Xilinx
  - See XUP donation request form at www.xilinx.com/univ
- Educational clip-art

Stanford NetFPGA

http://yuba.stanford.edu/NetFPGA/

NetFPGA is a PCI Board



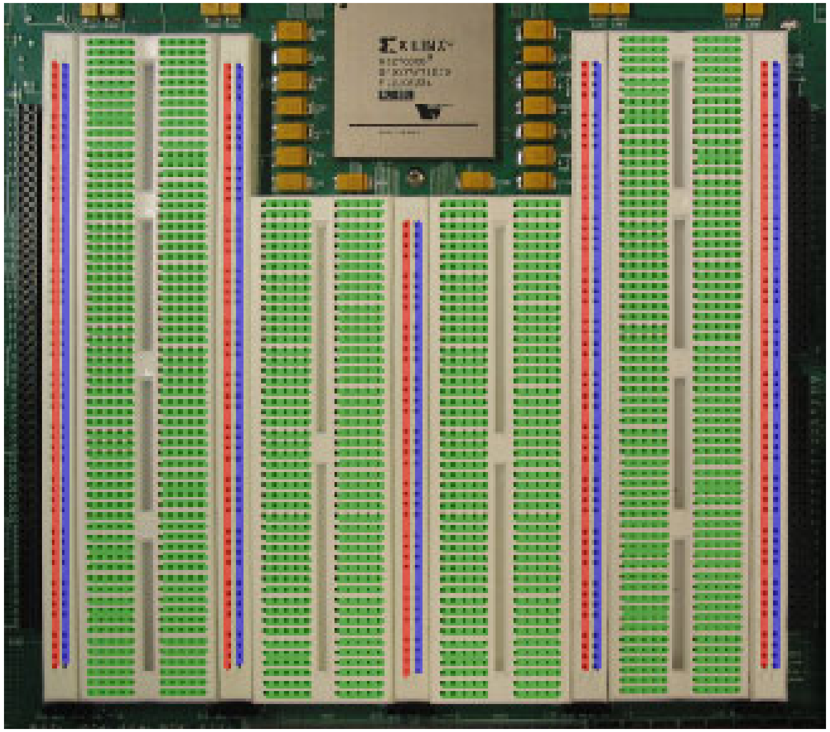
NetFPGA is a Programmable 4 x 1GE “switch” or any packet processor

- Program in Verilog
- Industry-standard design flow
- Contains embedded CPUs

- For classroom & research

MIT Labkit

- http://www-mtl.mit.edu/Courses/6.111/labkit/



Berkeley BEE2

- http://bee2.eecs.berkeley.edu/