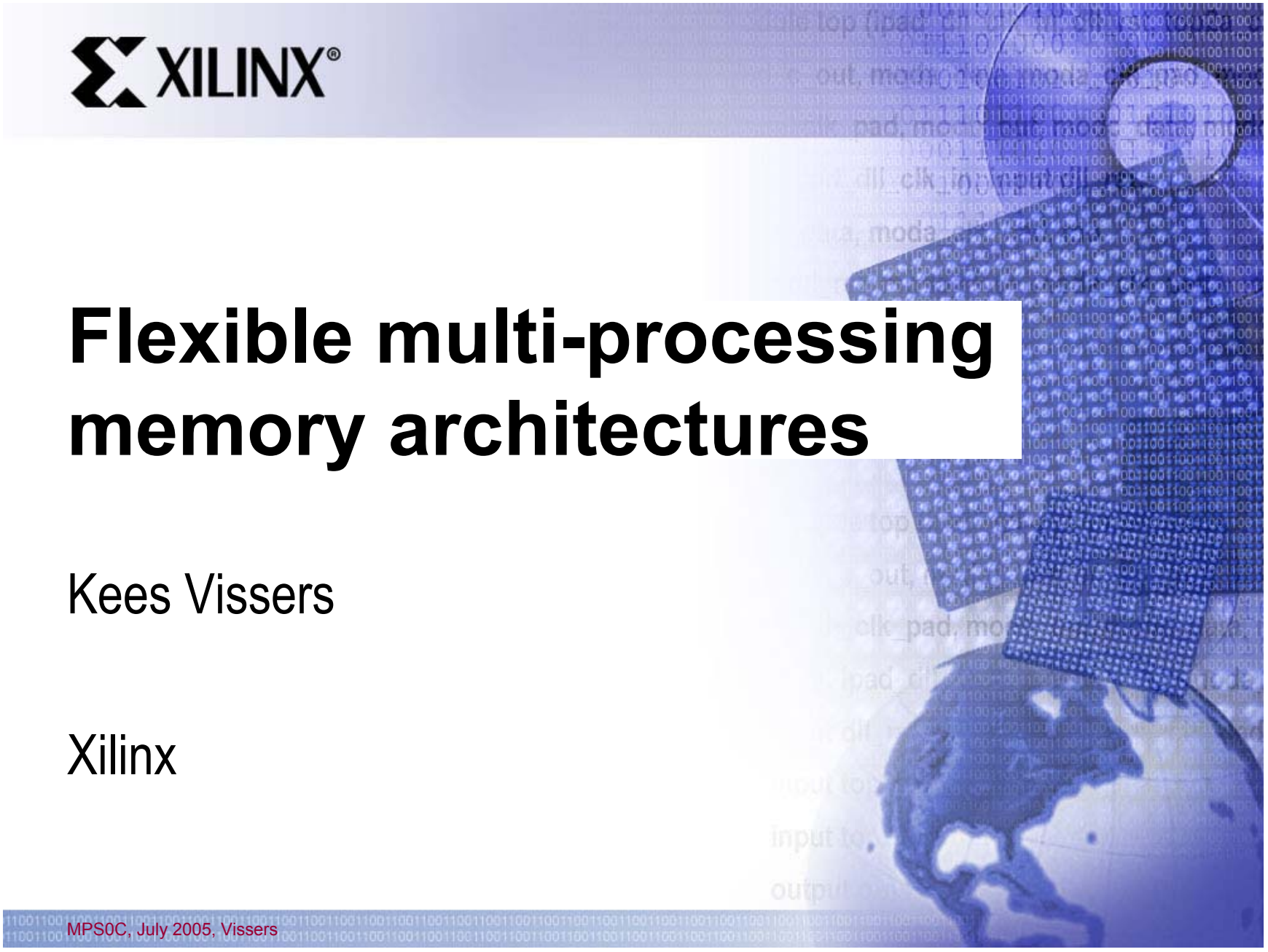
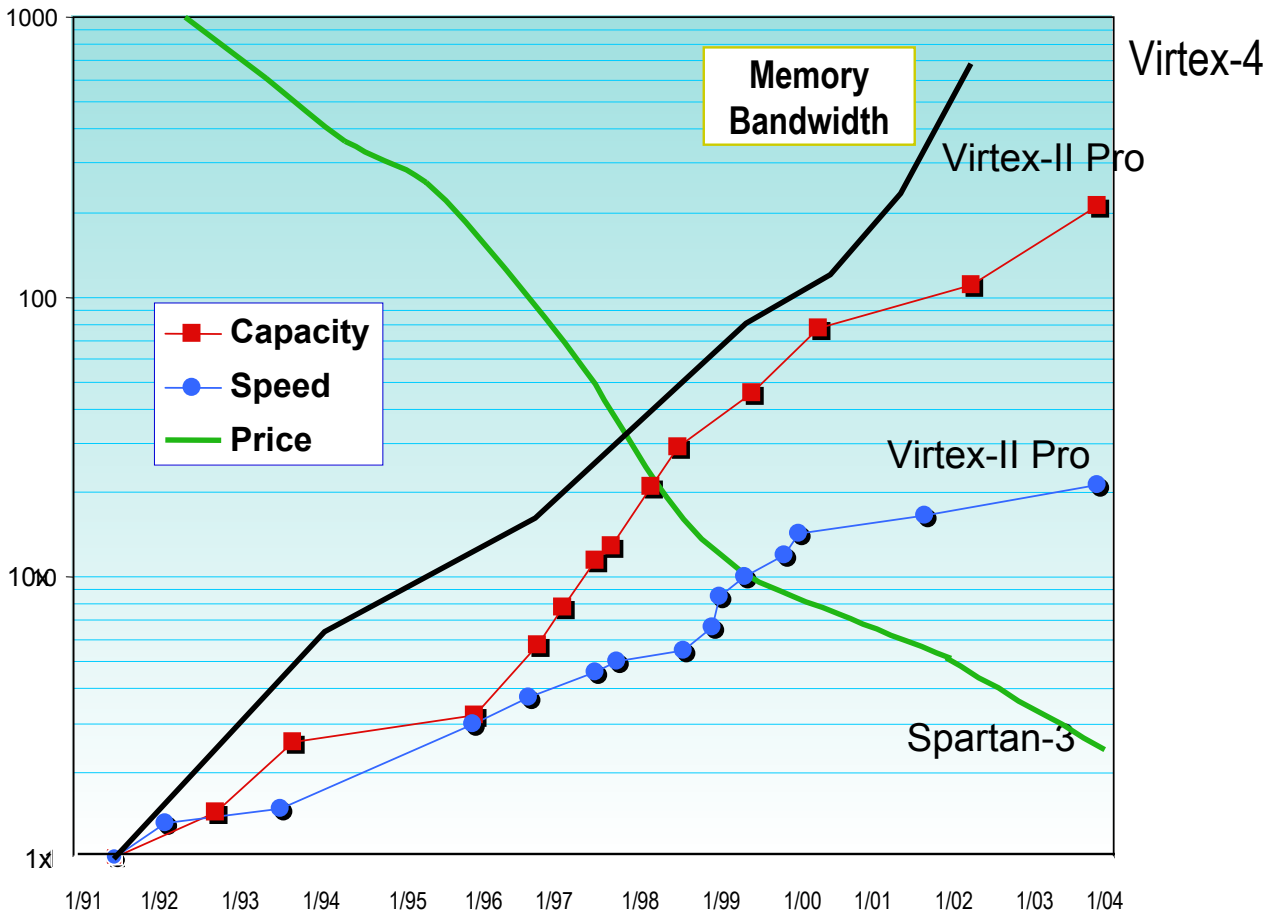


# Flexible multi-processing memory architectures

Kees Vissers

Xilinx

# FPGA, the last 10 years




Price: 300x

Speed: 20x

Capacity: 200x

4000x- 6000x customer value!

# Context

- operand routing is expensive
- Processors and processing is cheap
- More work per instruction makes sense (if it fits):
  - application specific instructions, e.g. media instructions
  - vector processing
- On- chip memory is limited, but easy to get to
- Off-chip memory is unlimited, but hard to get to…

# Single processor

- registers, load and store instruction,
  - typical 16-128 registers,1-5 cycles execute, pipelined
- Level one cache: e.g. D-cache, I-cache often run at processor speed
  - typical 16-64 Kbyte, 5-50 cycles for a cache miss, runs at processor clock speed
- Level two (and three) cache: often unified
  - typical 128K-1Mbyte, needs to go off-chip, hundreds of processor cycles, runs at the interface speed.

# Caches and DMA

Cache:

- hold the most recently used data. Bad for streaming, bad for large program jumps.
- Abstract where what variable is: c= A[i] + B[j]

DMA:

- explicitly program where what data is
- requires synchronization with the program loop
- programmed pre-fetching, small stream-buffers

# Implicit Flow control

Buffer

- full and empty signals, asynchronous behavior

Shared memory

- scheduled access guarantees proper order
- low level semaphores (programmer visible?)

Programming concepts: blocking read or writes, blocking guards (CSP), network of actors.

# Multi-processor models

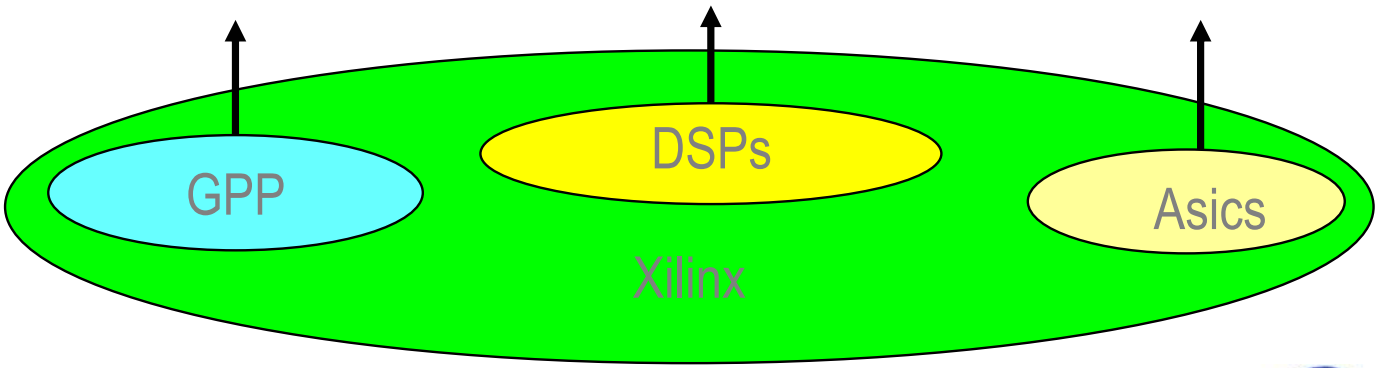
- Sandbridge
- ST multi-processor core
- JPEG2000, micro-blaze network
- Streaming over multi-processors, e.g.  tensilica
- FPGA styles
- SMP, cache coherent ARM, cache coherent media processors
- IBM/Toshiba/Sony cell processor

# FPGA: the ultimate solution

- Customers like: glue-less, risk free, programmable systems, that allow internet updates.
- The ONLY platform that can do it ALL:
  - Programmable I/O, very High Speed I/O
  - Programmable function
  - **Programmable memory hierarchy**
  - Programmable internal interconnect

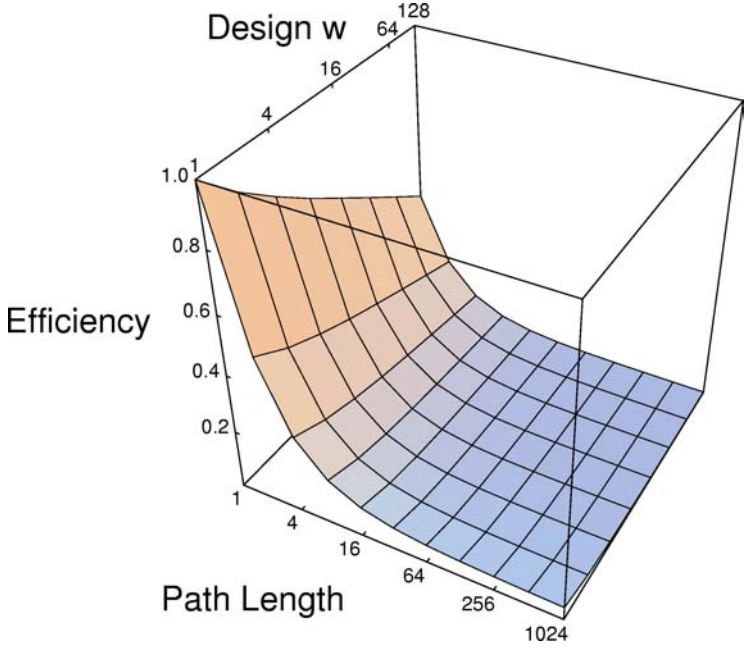
# Processors and Pipelines

| element | PPC | PPC + APU | folding | LUTs DSP48 |
|---|---|---|---|---|
| clock:sample | 1000:1 | 100:1 | 10:1 | 1:1 |
| 500MHz clock | 500Ks/s | 5Ms/s | 50Ms/s | 500Ms/s |
| Memories | 4-256KByte | 4-256KByte | 10KByte 1KByte | |
| Applications | control → audio → | mobile video → HDTV → comms → radar | | |

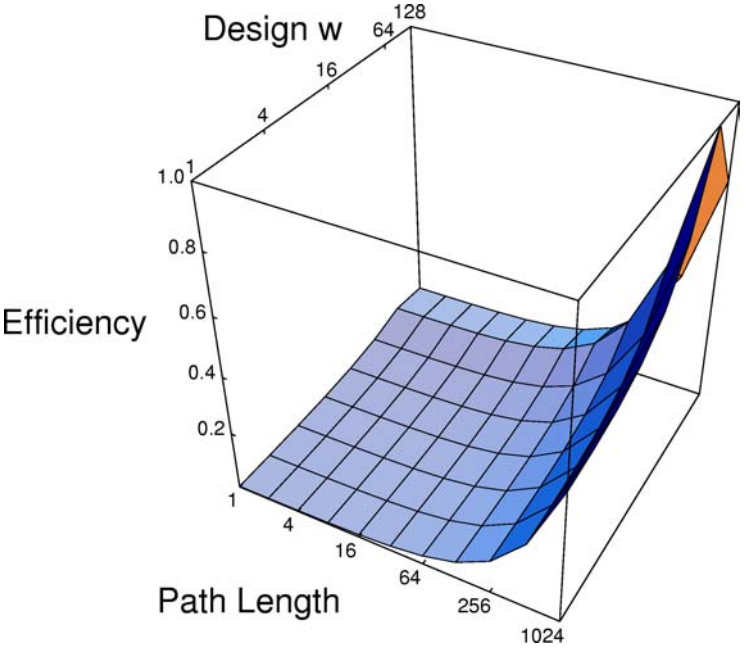


# Architectural Efficiency
## Spatial vs. Temporal Computing

FPGA (*c*=*w*=1)




"Processor" (*c*=1024, *w*=64)



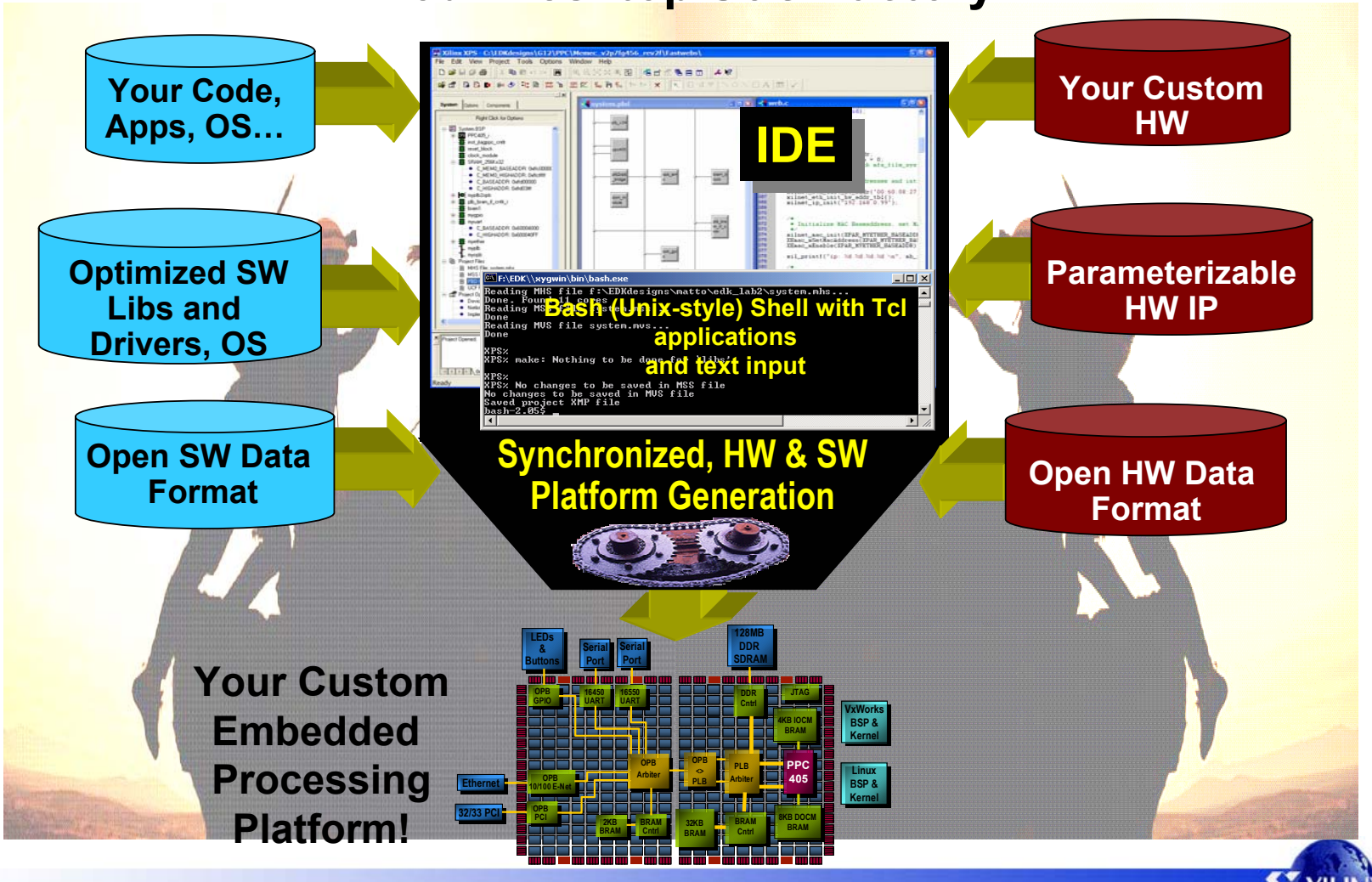

Figures courtesy of André DeHon, California Institute of Technology

# What language for what

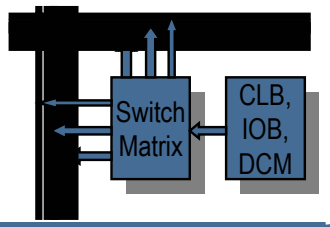
| | Transaction Level | RTL | C/C++ | Matlab/ Simulink | CSP, Click, CAL etc |
|---|---|---|---|---|---|
| Cycle accurate | - | ++ | -- | - | -- |
| Concurrency | ++ | ++ | -- | ++ | ++ |
| Type system | SystemC: ++ SystemVerilog: ++ | VHDL: + Verilog: - | C: + C++: ++ | + | -/+ |
| High Level of Expression | + | -- | ++ | + | ++ |
| Compilation for Processors | - | -- | ++ | + | + |
| Synthesis for FPGAs and Asics | - | ++ | - | + | + |

# Platform Studio

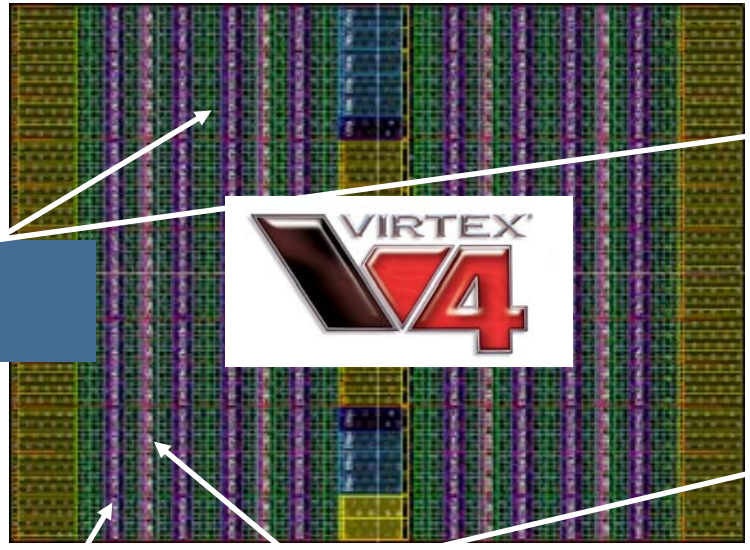
## Your Desktop SoC Factory

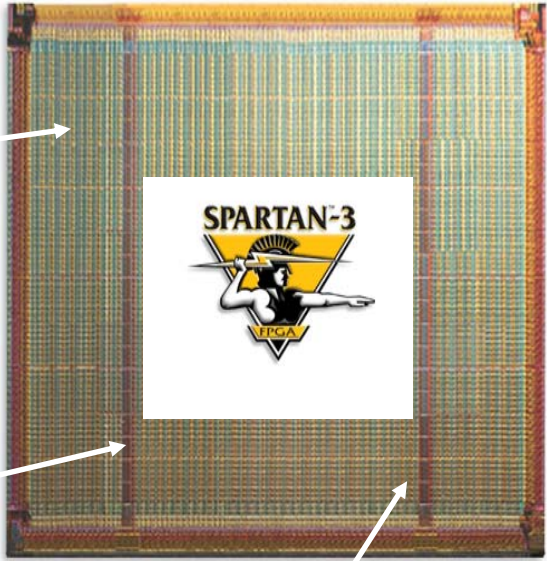


# Xilinx DSP Hardware Platforms

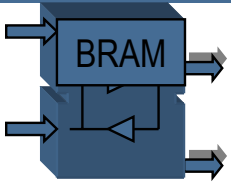



Programmable Arithmetic Fabric

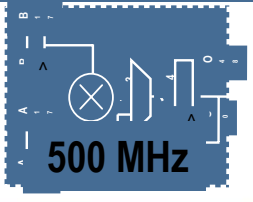



Synchronous Dual-Port RAM




Hundreds of XtremeDSP slices

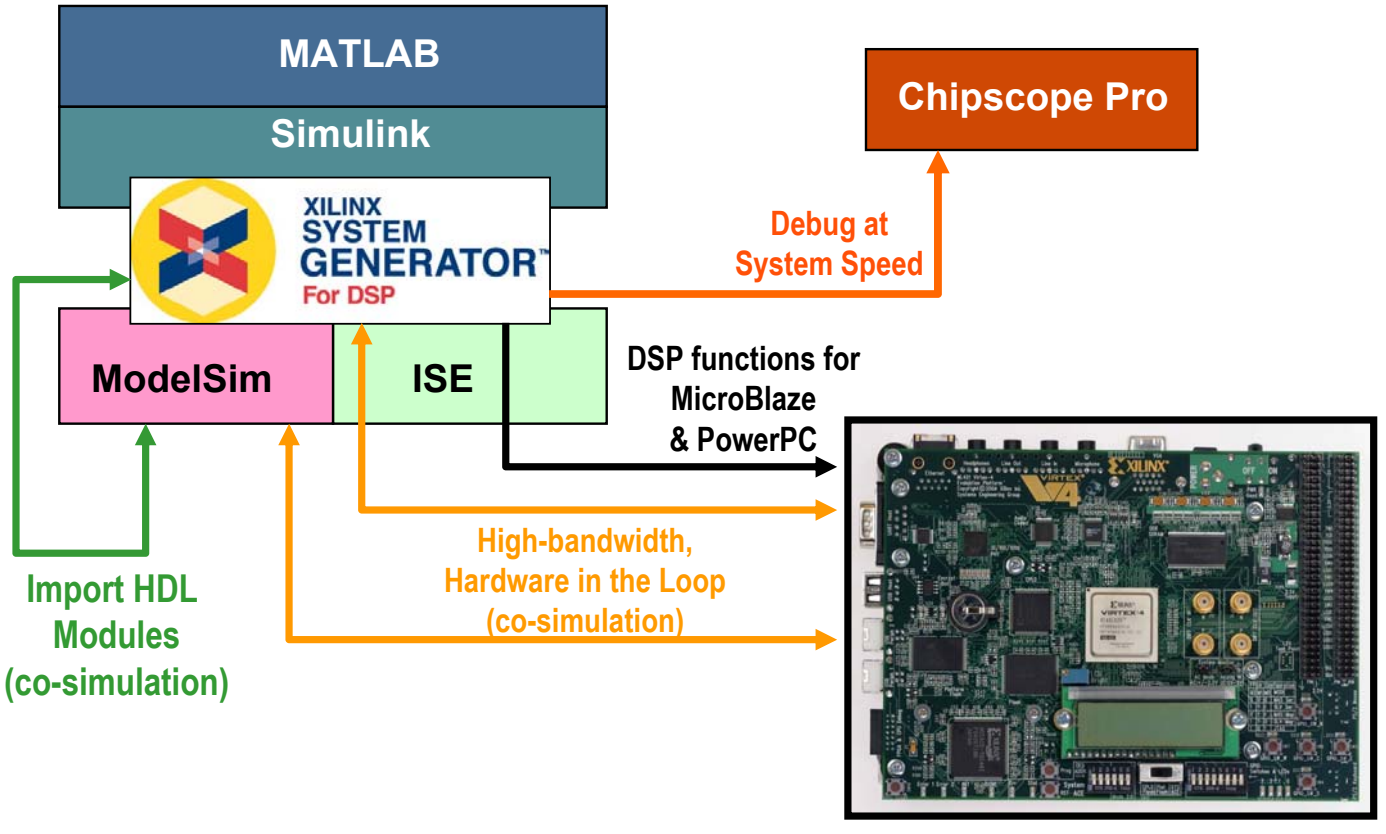



×

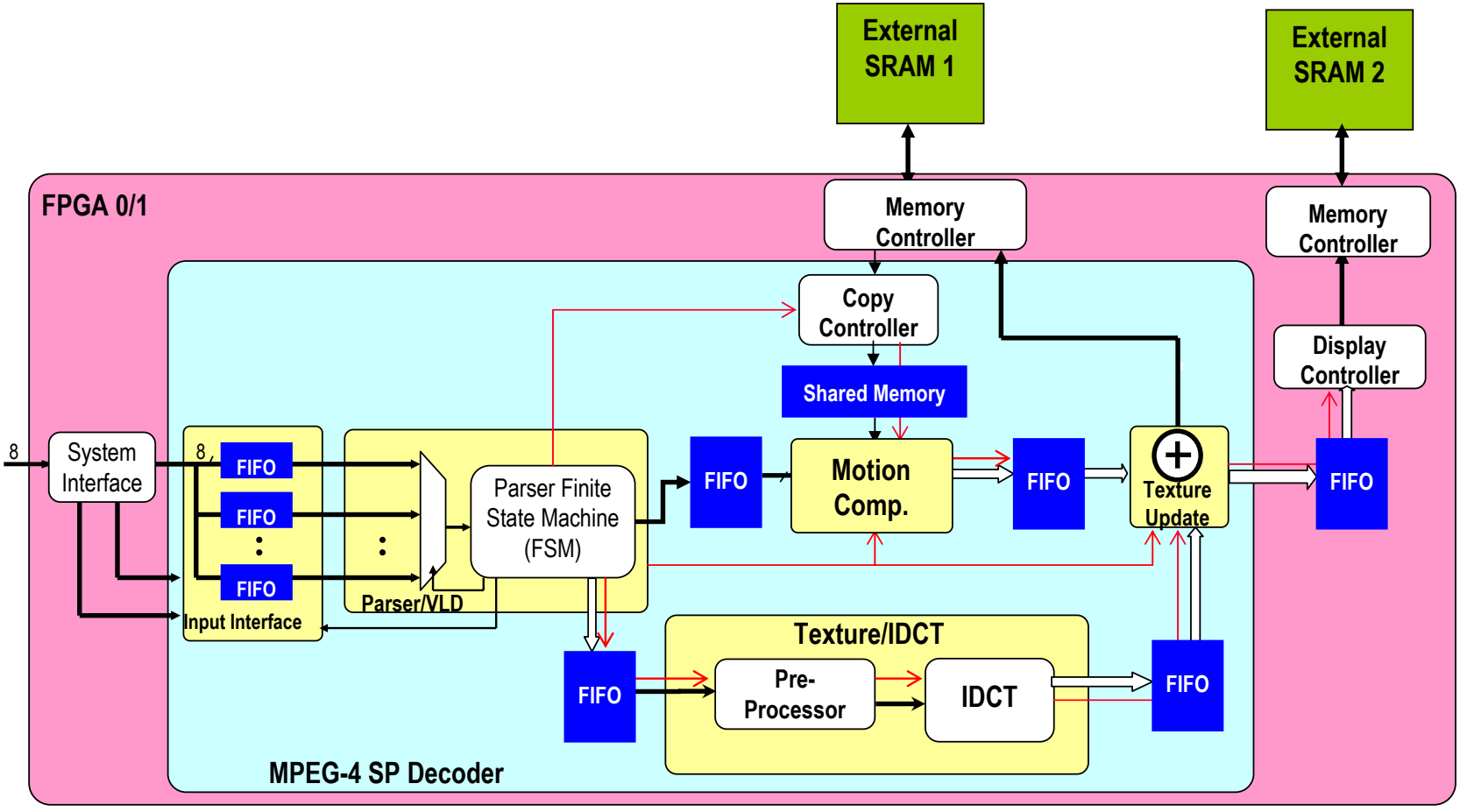
100+ multipliers

High performance through parallelism

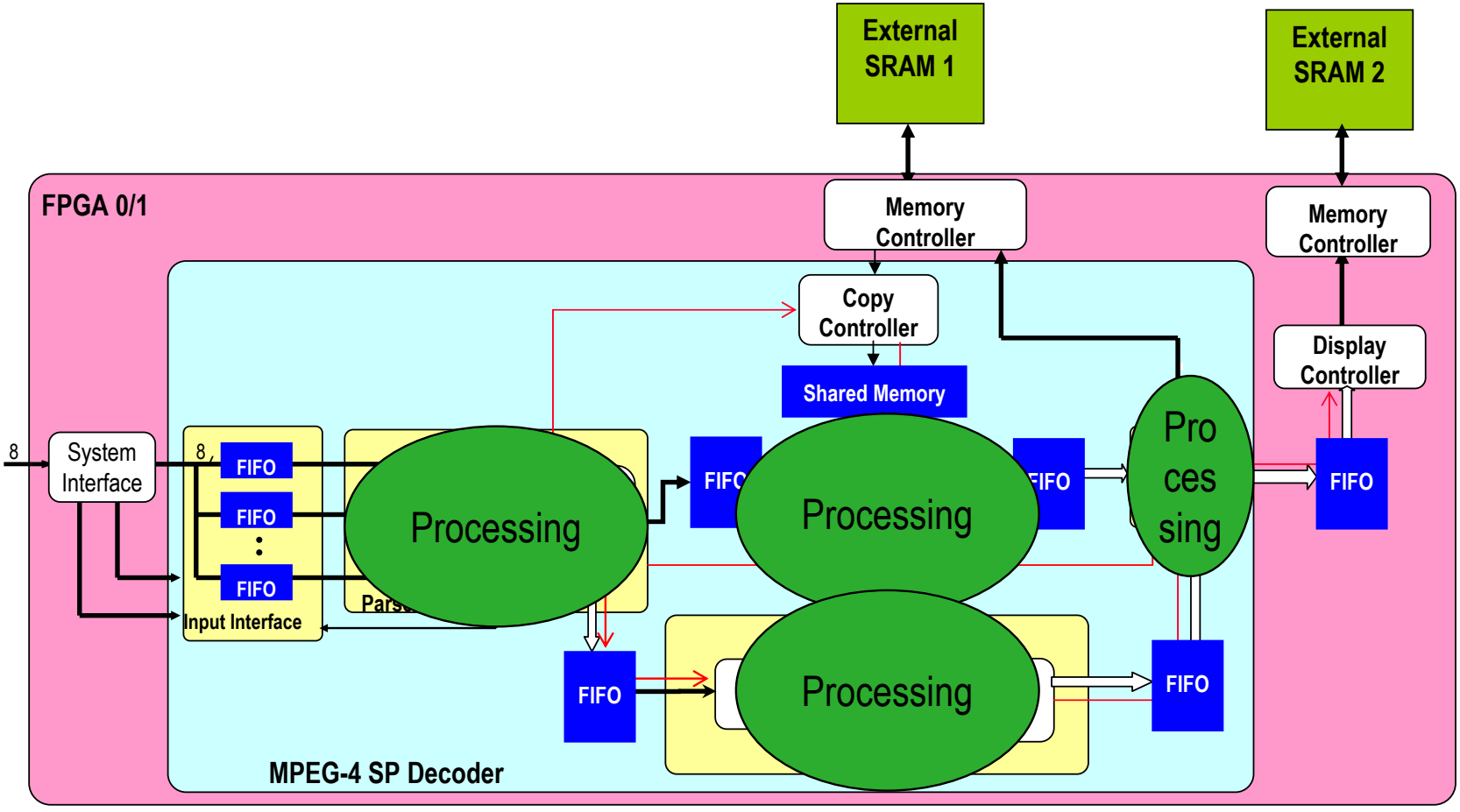
# System Generator for DSP



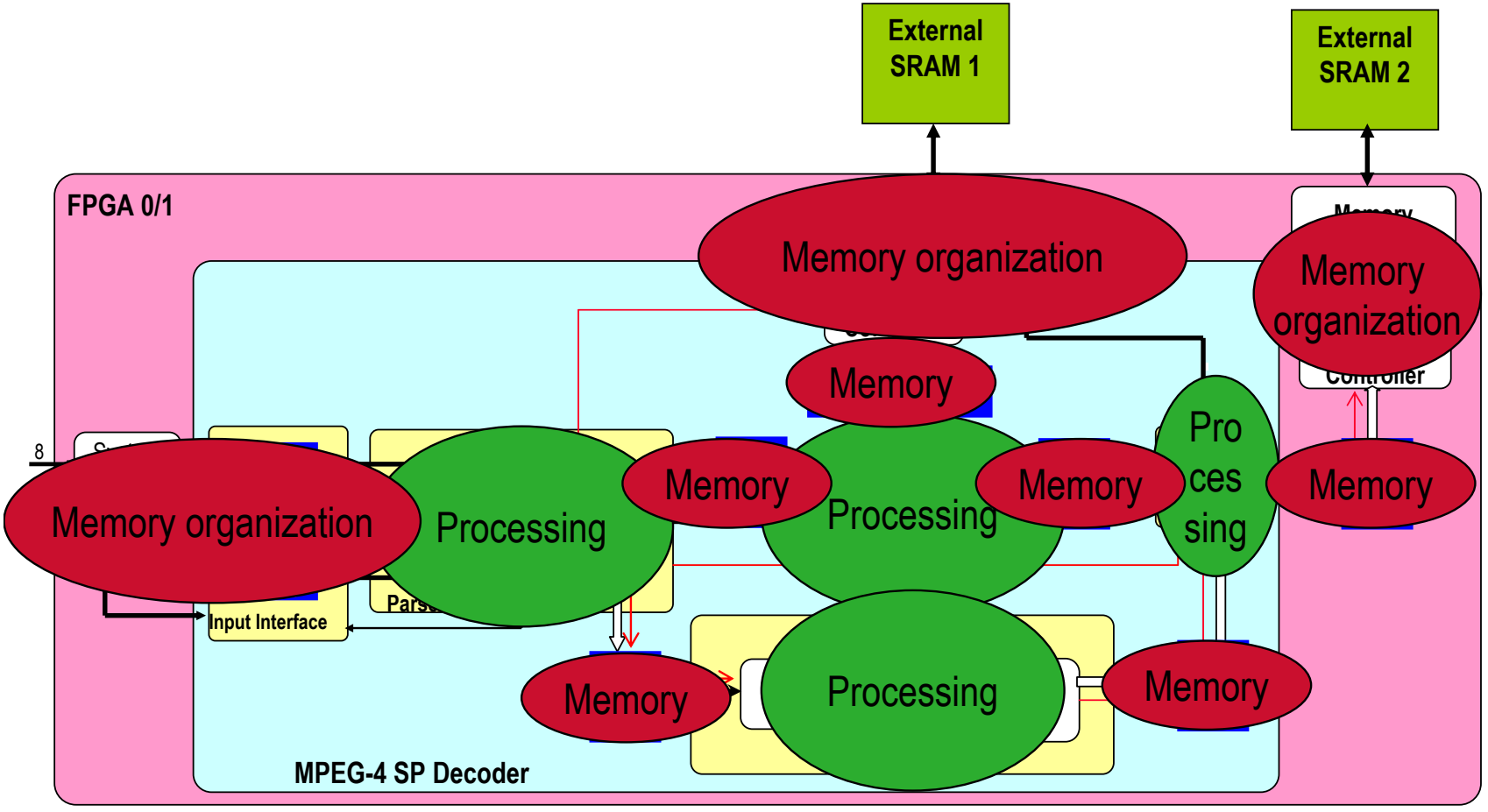
# MPEG-4 Simple Profile Decoder



# MPEG-4 Simple Profile Decoder



# MPEG-4 Simple Profile Decoder



# MPEG-4 Simple Profile Decoder

Abstract from FPGA details or Processor Details

# Memory primitives

Design objective: build a problem specific balanced pipeline that minimizes the on-chip storage.

- Fifo, with flow control
- Buffered fifo, with flow control
- Shared memory, with access control
- Sliding window, with DMA

# Memory primitives low level

Design objective: build a problem specific memory hierarchy out of pre-fabricated primitives:

- Registers,
- Fifo, with flow control (synchronous or asynchronous)
- BRAM: Shared memory (synchronous or asynchronous)
- off-chip memory + memory controller

# MPEG-4 Decoder - Resources

## (Throughput of 50k MB/sec; Single input stream)

| Hardware Block | Lines C | Lines VHDL | BRAMs | Slices | MULTs |
|---|---|---|---|---|---|
| Input Interface | 430 | 100 | 1 | 20 | 0 |
| Parser/VLD | 1,891 | 5,820 | 0 | 1,670 | 2 |
| Copy Controller | 406 | 1,634 | 0 | 700 | 1 |
| Motion Compensation | 265 | 1,527 | 0 | 500 | 2 |
| Texture/IDCT | 943 | 2,969 | 6 | 2,050 | 23 |
| Texture Update | 126 | 401 | 0 | 150 | 2 |
| Auxiliary Controllers | 189 | 1,916 | 16 | 1,470 | 0 |
| Communication/Memory | 0 | 0 | 28 | 250 | 0 |
| Total | 4,250 | 14,367 | 51 | 6,810 | 30 |

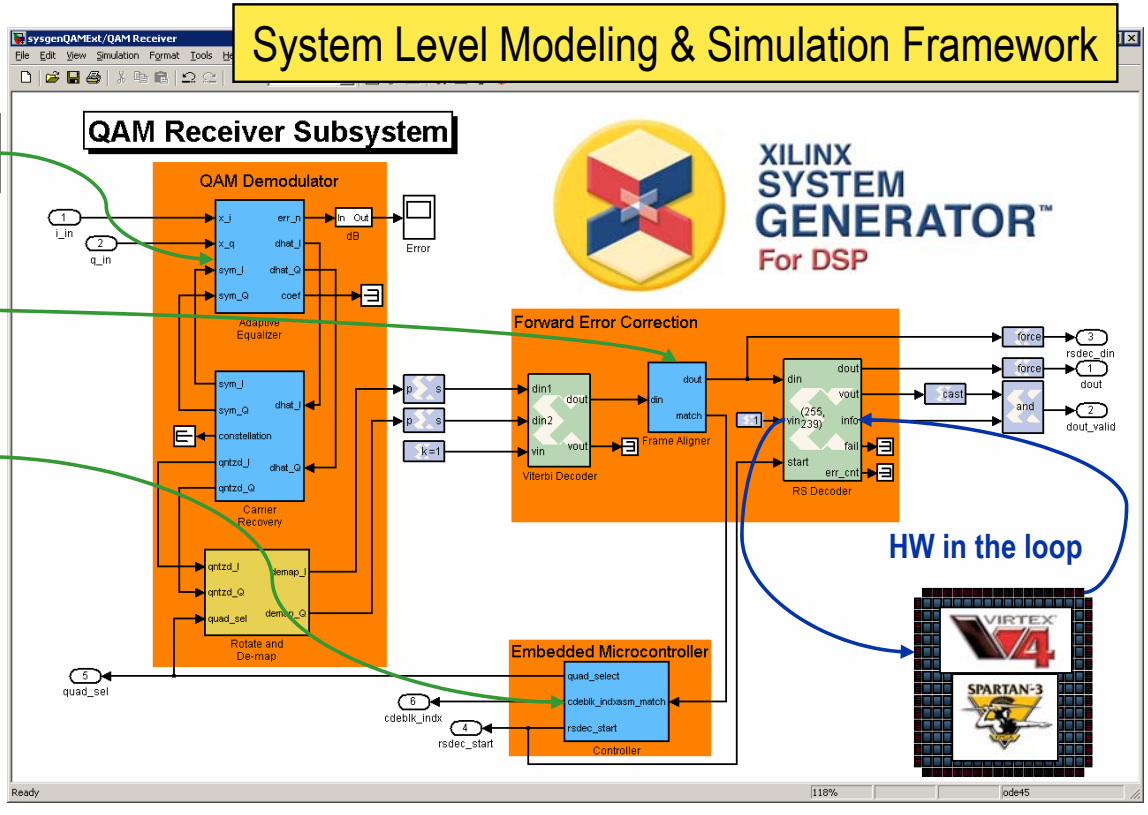
# DSP Programming Model

Work in the language of your problem

MATLAB → AccelChip

HDL

C → Embedded Development Kit



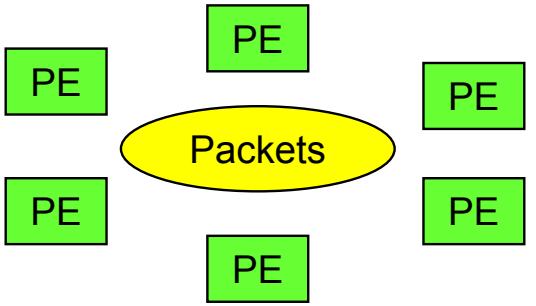

**Methodology re-couples behavior with implementation**

**(while abstracting hardware details *when possible*)**

# Packet processing for the “software guy”

Programmer’s view:
Packets in system are stored in one place

Processing elements (PEs) operate on packets within this communal memory




Actual platform FPGA implementation:
Highly distributed memory architecture

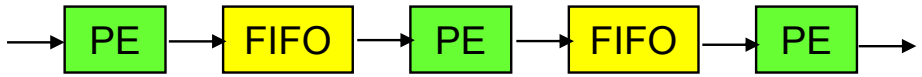
- parts of packets are stored at relevant PEs
- storage encompasses registers, on-chip block memory, off-chip memory

Tools automate the generation of the memory architecture

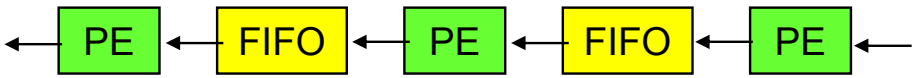
# Packet processing for the “hardware guy”

Programmer’s view:

Packets in system flow through inter-block FIFOs




Represented in *Click Plus* as Queue elements between processing elements (PEs)




Actual platform FPGA implementation:

Shared multiple-port memory controller for external memory

- inter-block FIFOs are mapped to areas of the external memory
- dynamic configuration of the FIFO collection and mappings

Tools automate the generation of the memory architecture

# Challenges

- Build good compilers and synthesis from new concurrent specifications (CSP, Click, CAL)
- Extract parallelism from C programs…..
- Support High-level descriptions
- Integrate a development environment for processors and bit-oriented FPGA design

# Conclusions

- Streaming interfaces and flexible memory architectures are a key benefit.
- FPGAs have the tools to build Multiple Processor systems
- FPGAs enable the programming of total systems consisting of I/O, Memory and processing
- The concurrent programming environment is essential

# Do I need a network on-chip?

- Pro:
  - no bus
  - time-multiplexed wires
- Con:
  - High latency: multi-threading, more on-chip storage

Typical FPGA point to point wires:
10,000 wires at 200MHz = 2000Gbits/s = 250Gbyte/s

# Acknowledgement


- Kristof Denolf, IMEC
- Adrian Chirila Rus, IMEC
- Robert Turney, Xilinx
- Paul Schumacher, Xilinx
- Gordon Brebner, Xilinx