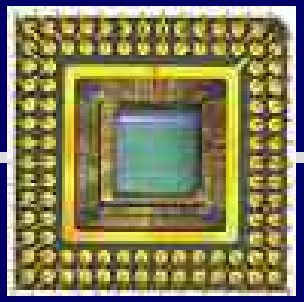


Introduction to Custom Computing

Oskar Mencer

oskar@doc.ic.ac.uk

Objectives

- What is Custom Computing, and Why
- Compilation issues
- Run-time issues
- Systems: putting it all together

Overview of Custom Computing

1. FPGAs: what and why
FPGA, the big picture
FPGA devices

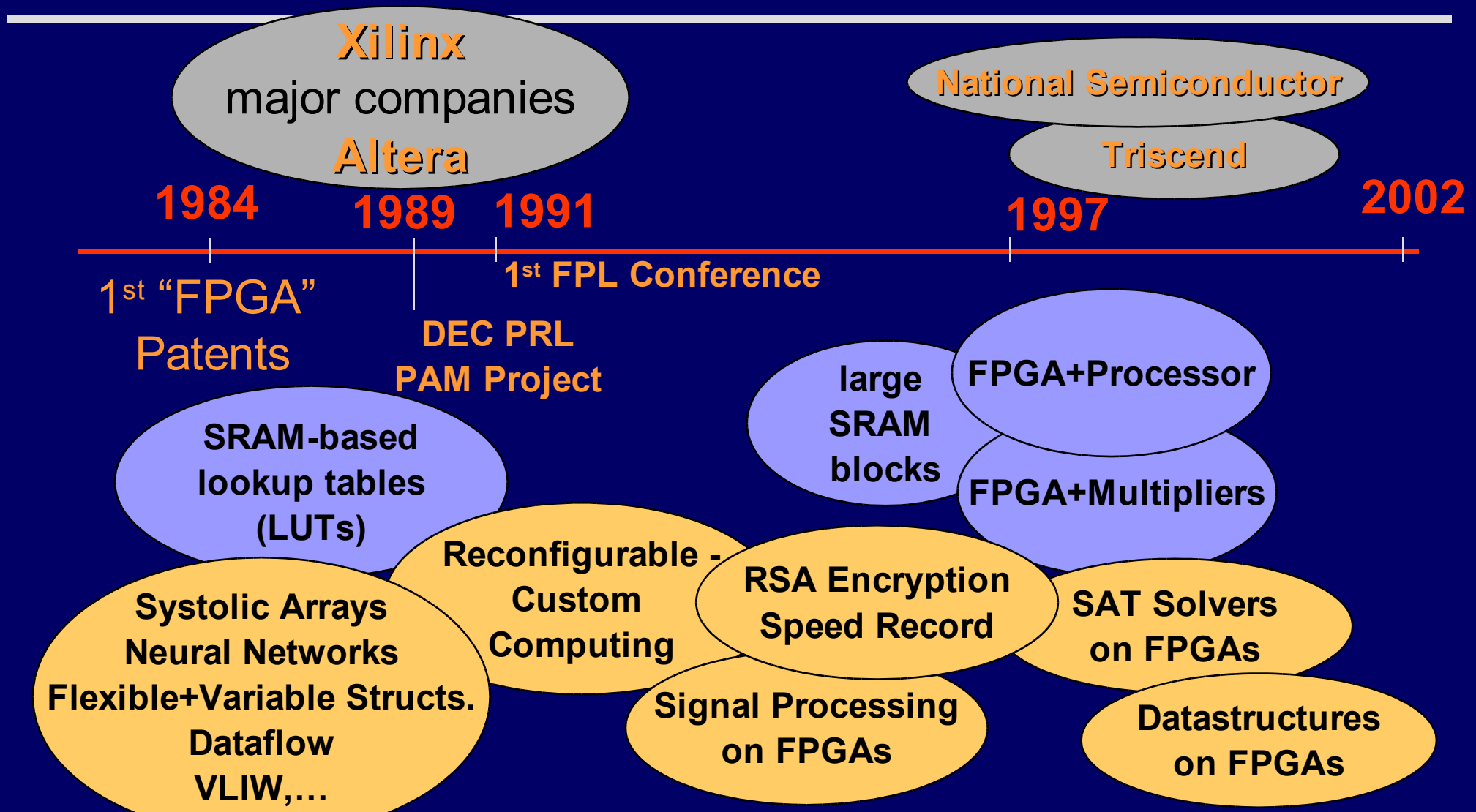
2. Compiling to FPGAs
architecture generation
compiler passes

4. Putting it altogether
system view
PAM: a PC-size system
SONIC: a PCI card

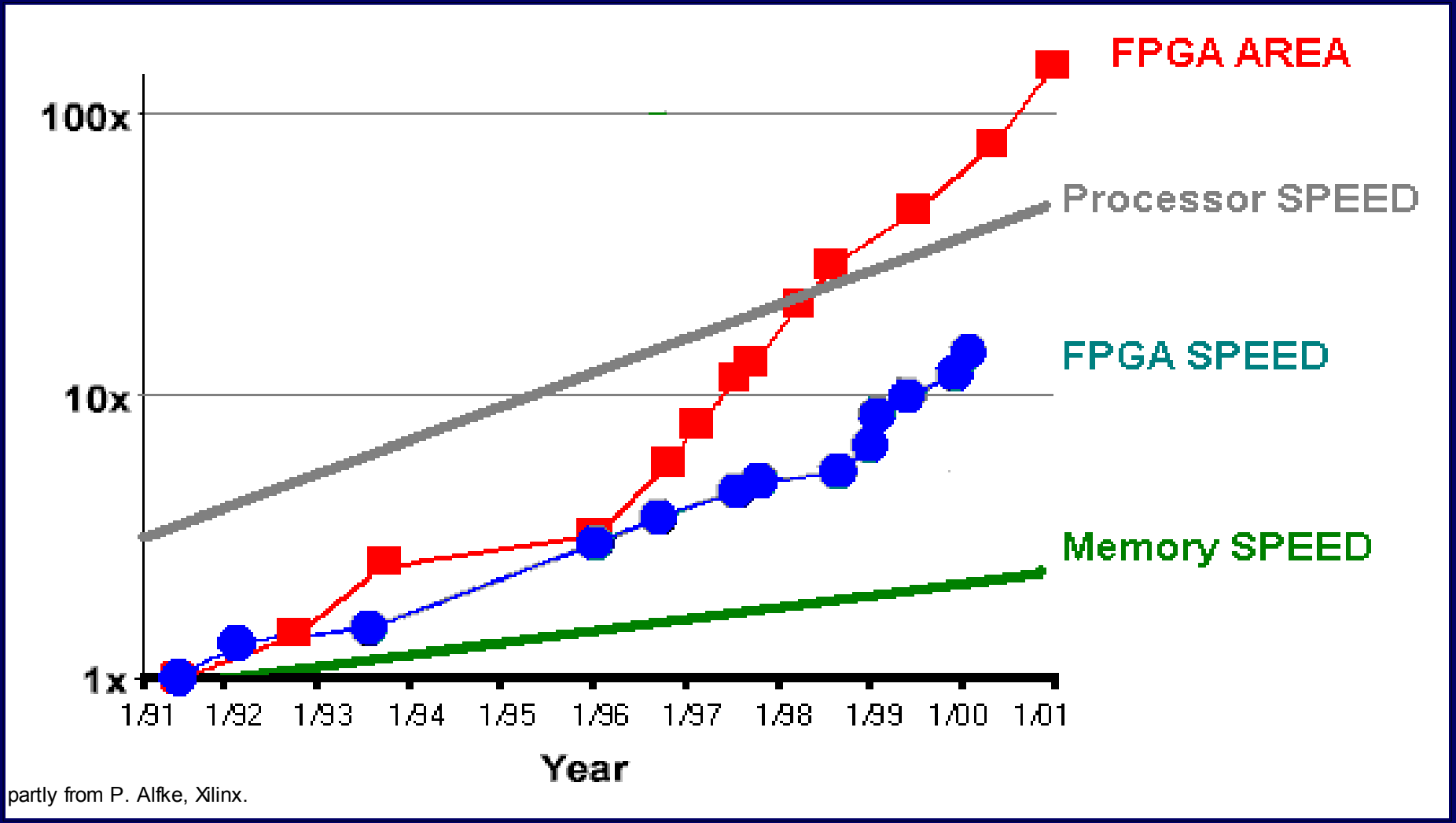
3. Run-time reconfiguration
generating configurations
- at compile time, at run time

scheduling reconfigurations
- control driven, data driven

some History

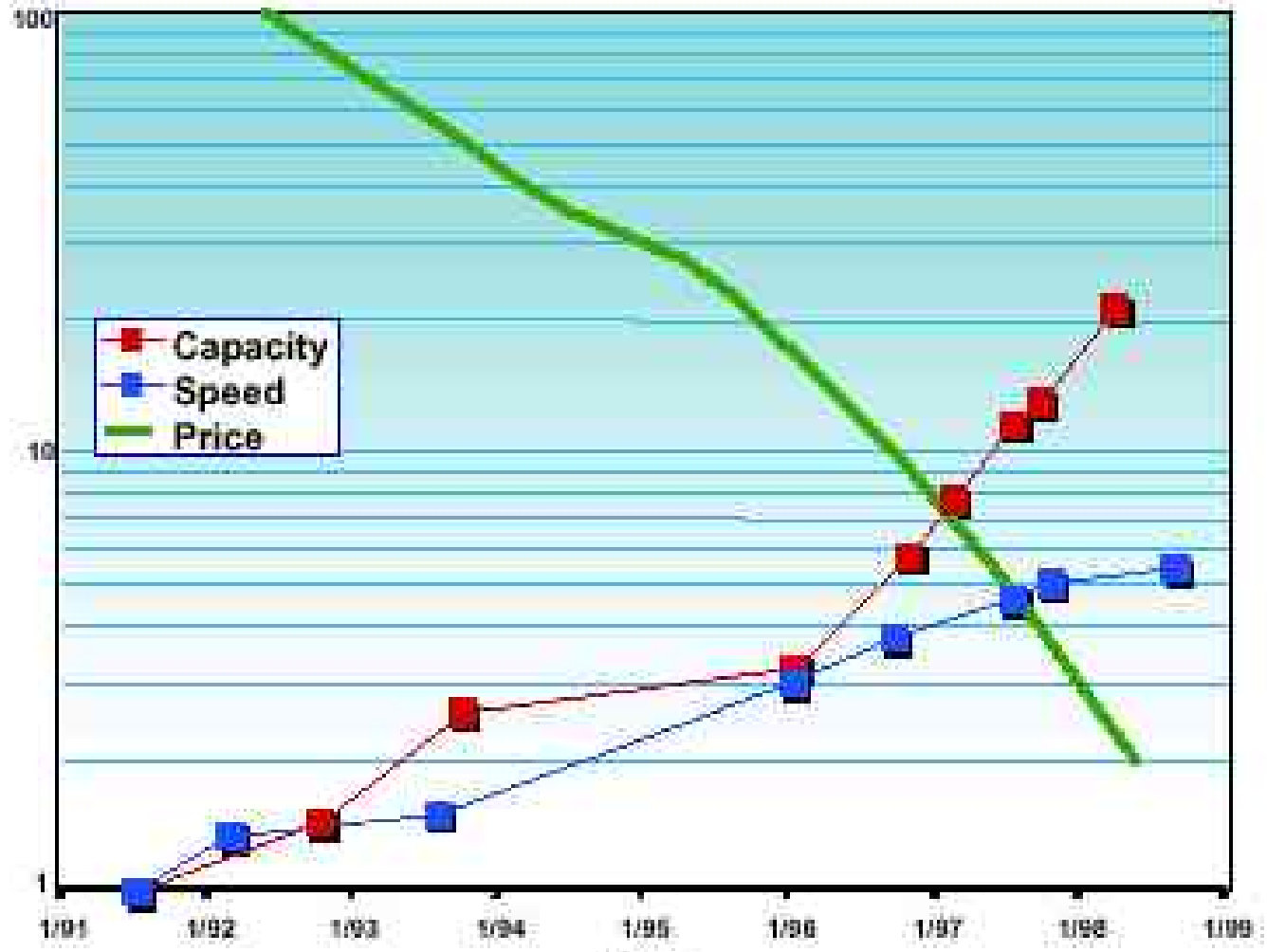


FPGA versus Microprocessor



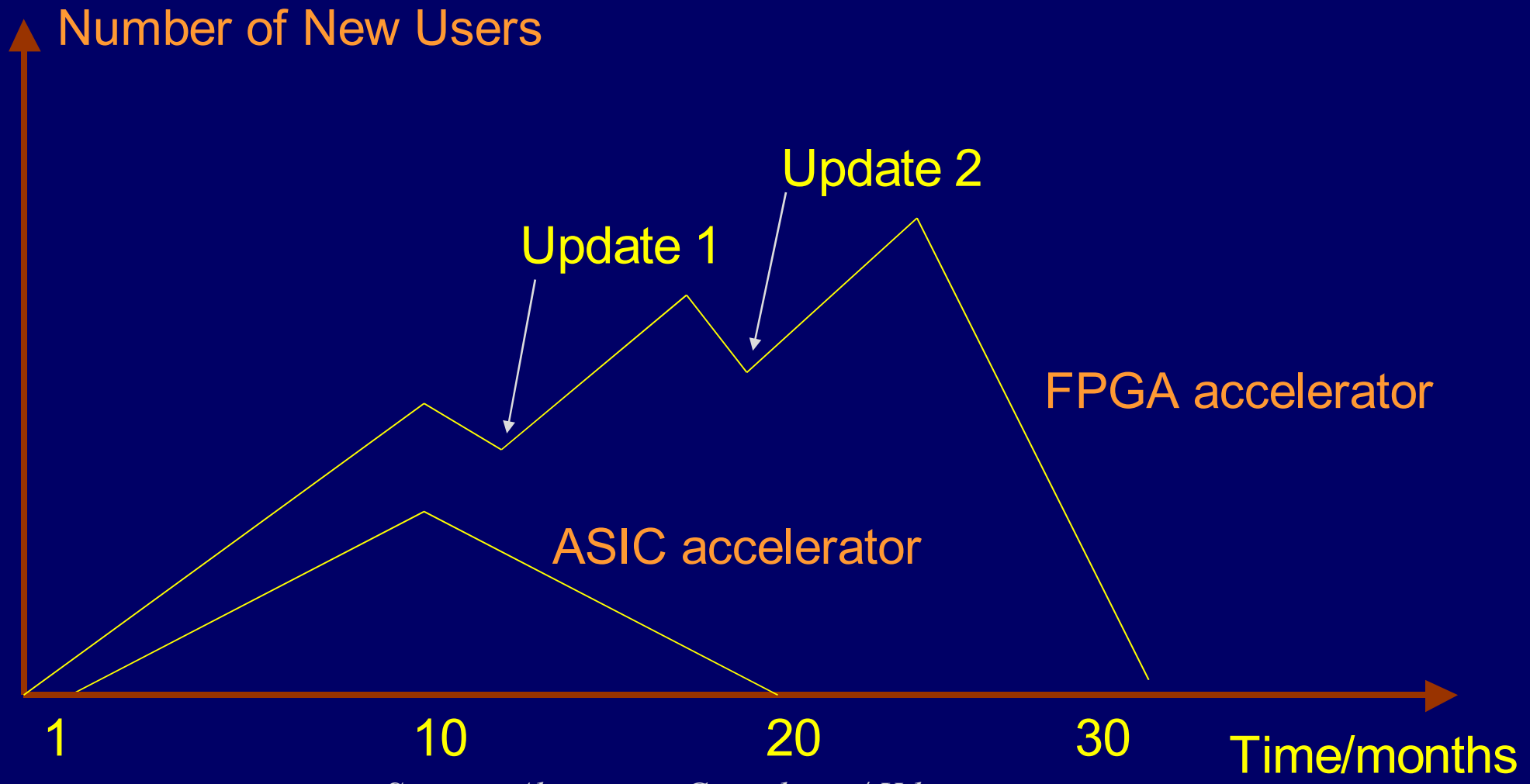
partly from P. Alfke, Xilinx.

… and the price per gate goes down …



from P.Alfke, Xilinx.

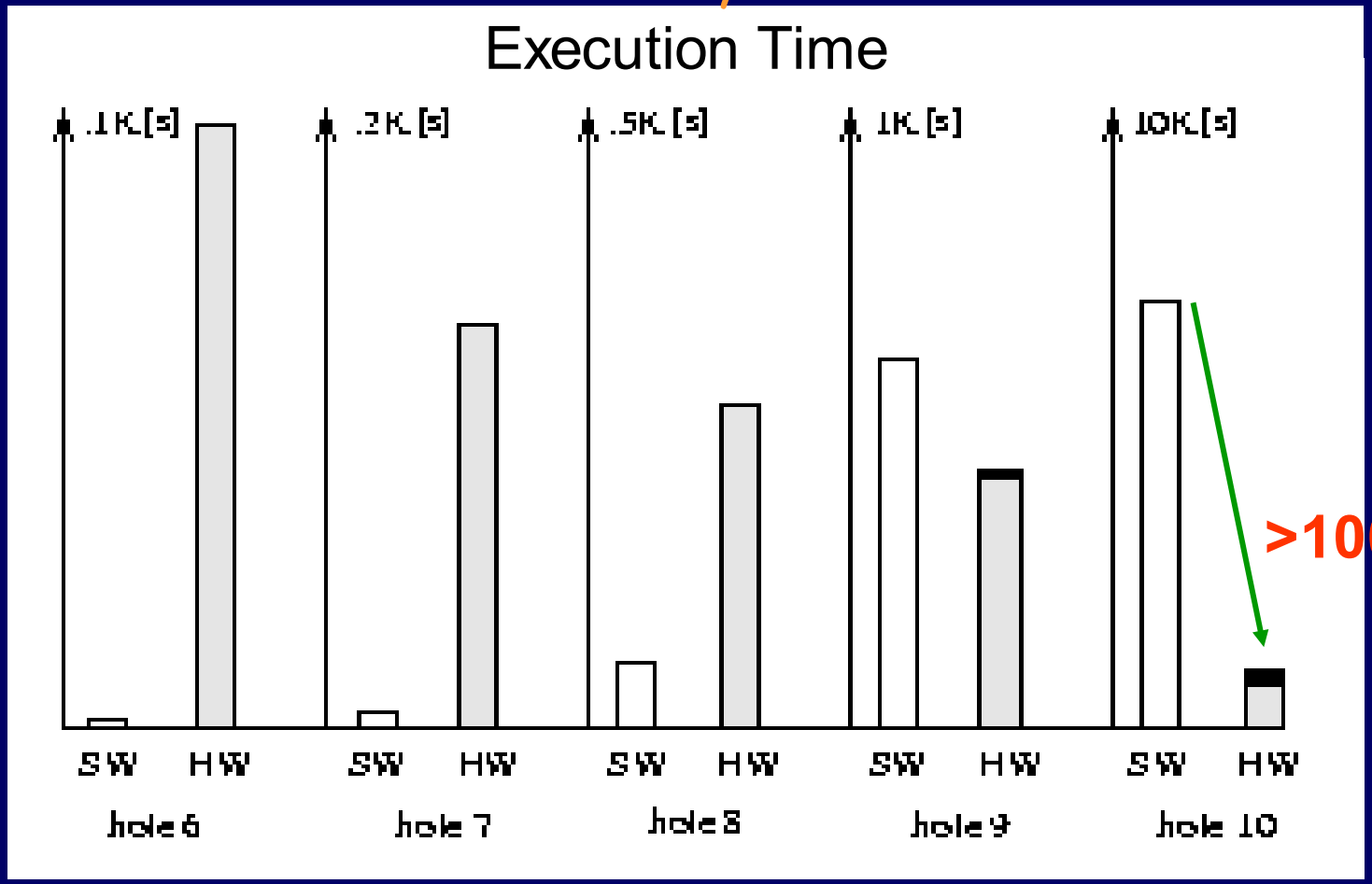
FPGA versus ASIC Accelerator Card



Source: Algotronix Consulting / Xilinx

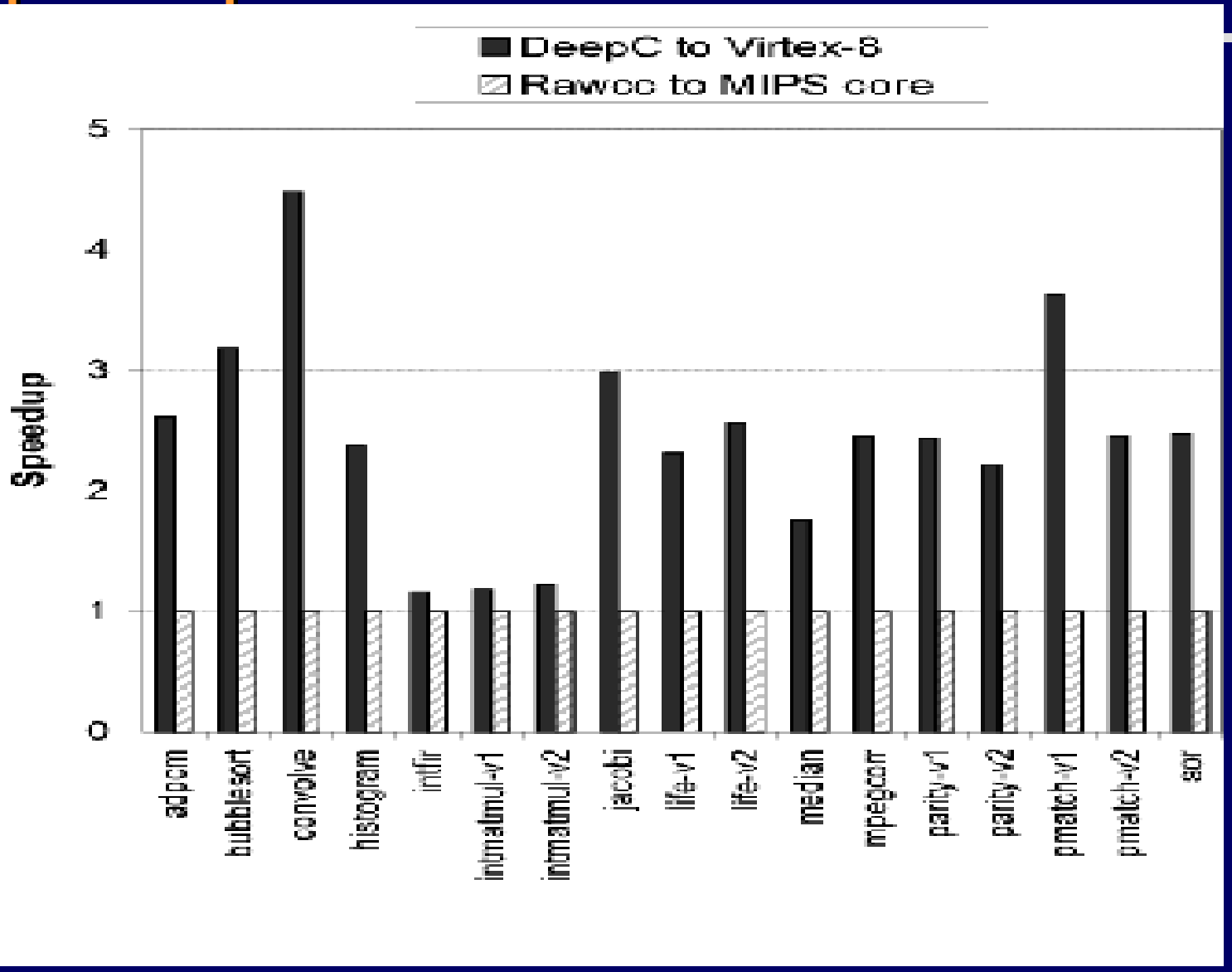
The Speedup Argument

Boolean Satisfiability: FPGA vs Pentium



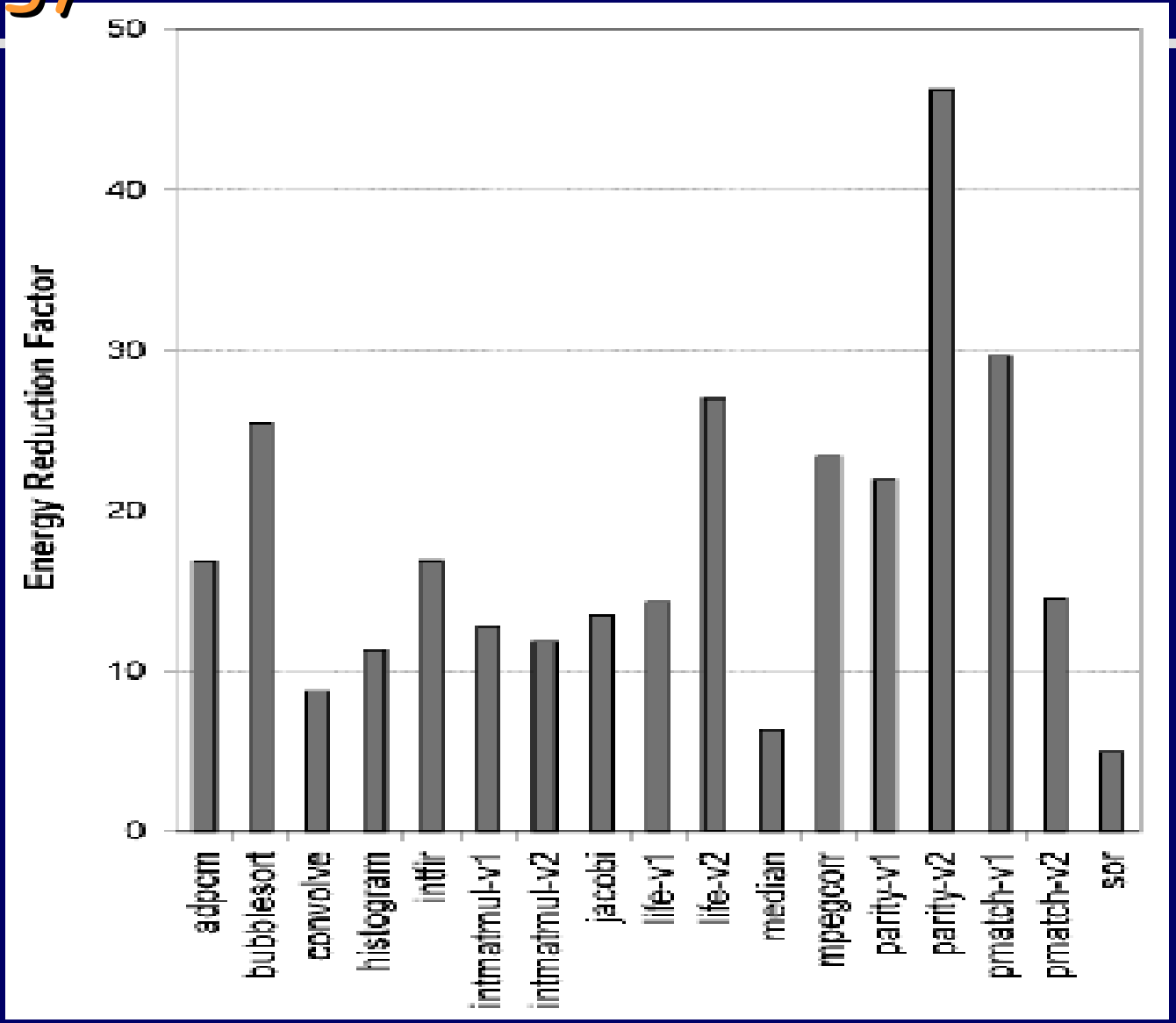
DIMACS Benchmarks, Xilinx FPGAs vs. GRASP Software

Speedup: FPGA vs MIPS Processor



Source: J. Babb PhD thesis, MIT, 2001

Energy Reduction: FPGA vs MIPS Core



Source: J. Babb PhD thesis, MIT, 2001

Overview of Custom Computing

1. FPGAs: what and why

FPGA, the big picture

→ FPGA devices

2. Compiling to FPGAs

architecture generation

compiler passes

4. Putting it altogether

system view

PAM: a PC-size system

SONIC: a PCI card

3. Run-time reconfiguration

generating configurations

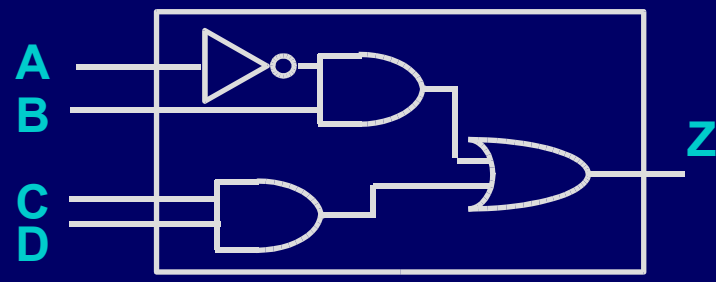
- at compile time, at run time

scheduling reconfigurations

- control driven, data driven

Look-Up Table (LUT): A Universal Gate

- combinatorial logic: stored in Look-Up Tables
- capacity limited by number of inputs not complexity
- LUT delay: independent of logic!

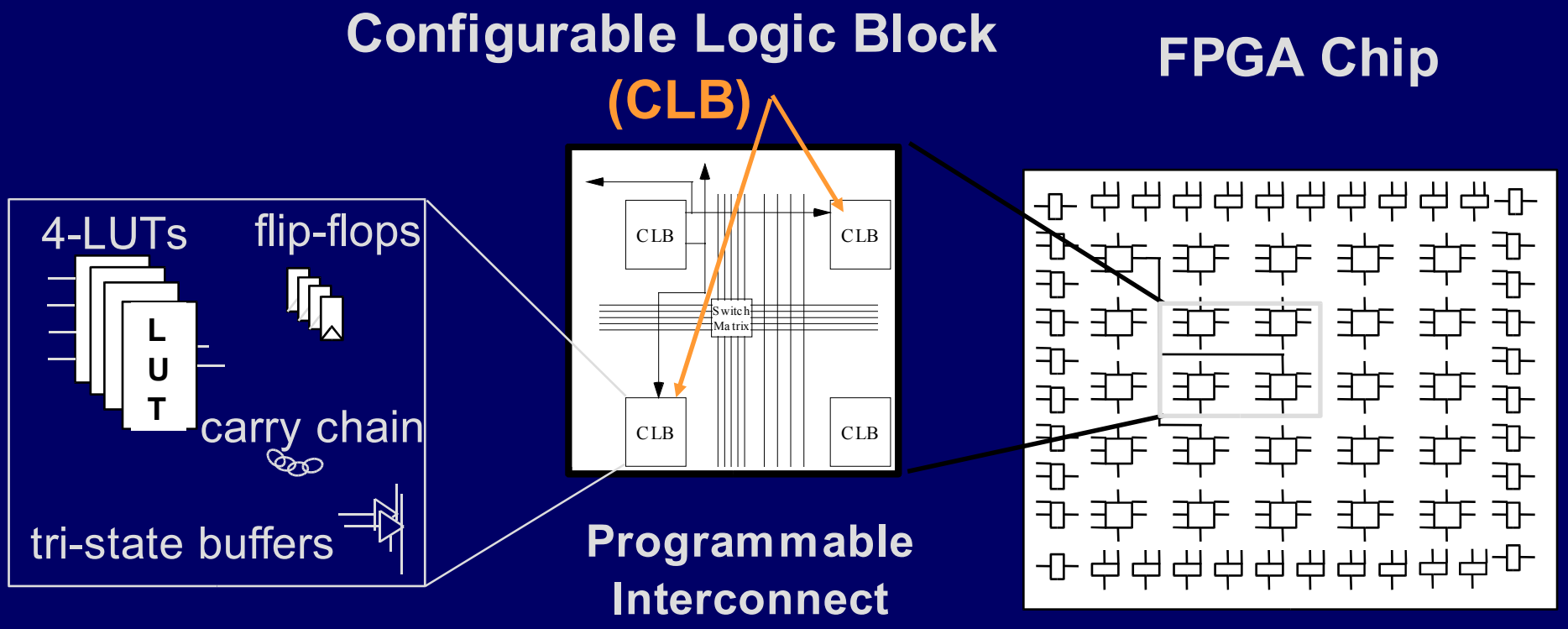


Combinatorial Logic

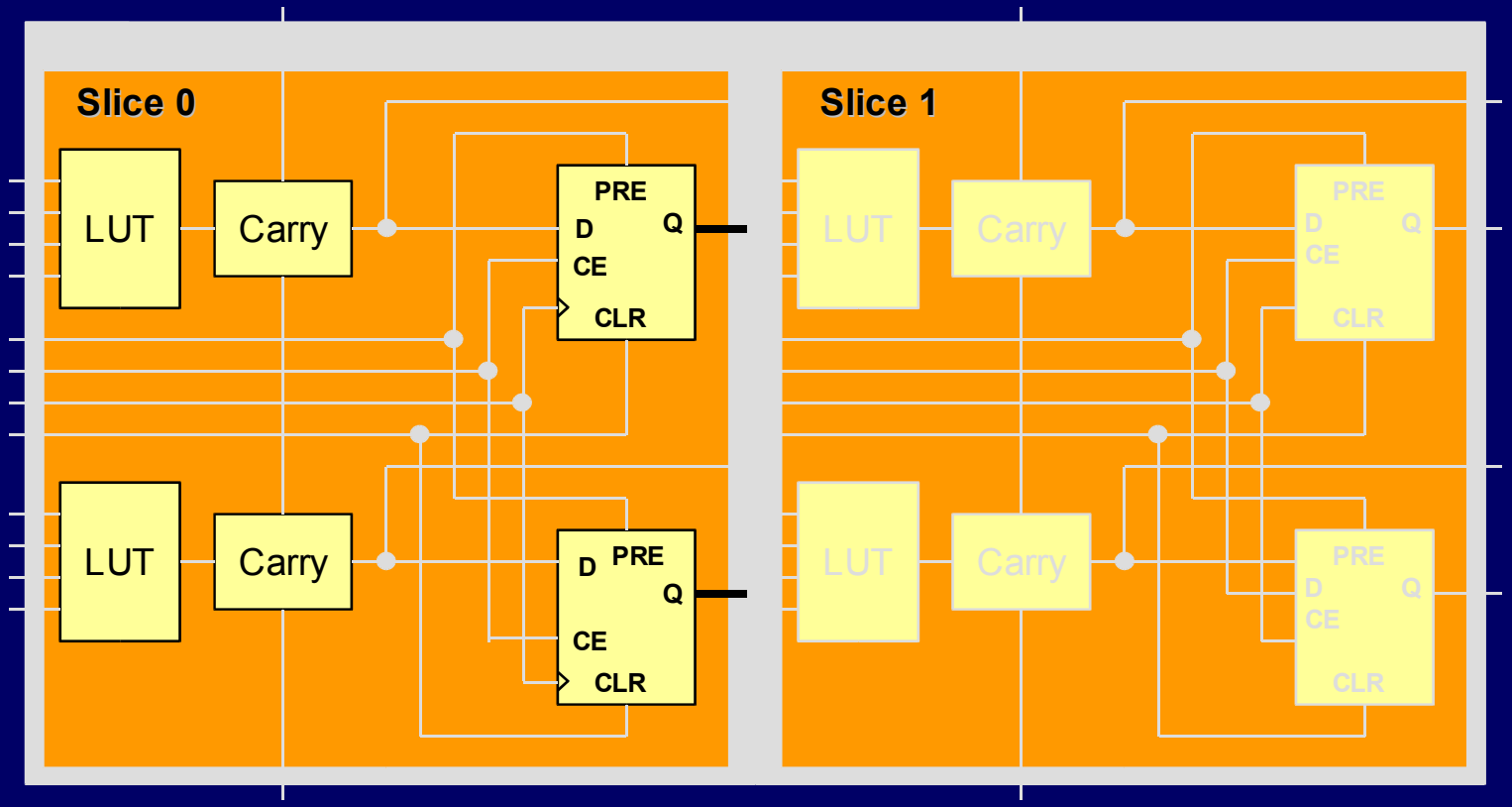
A	B	C	D	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
.	.	.	.	
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

16-bit SRAM

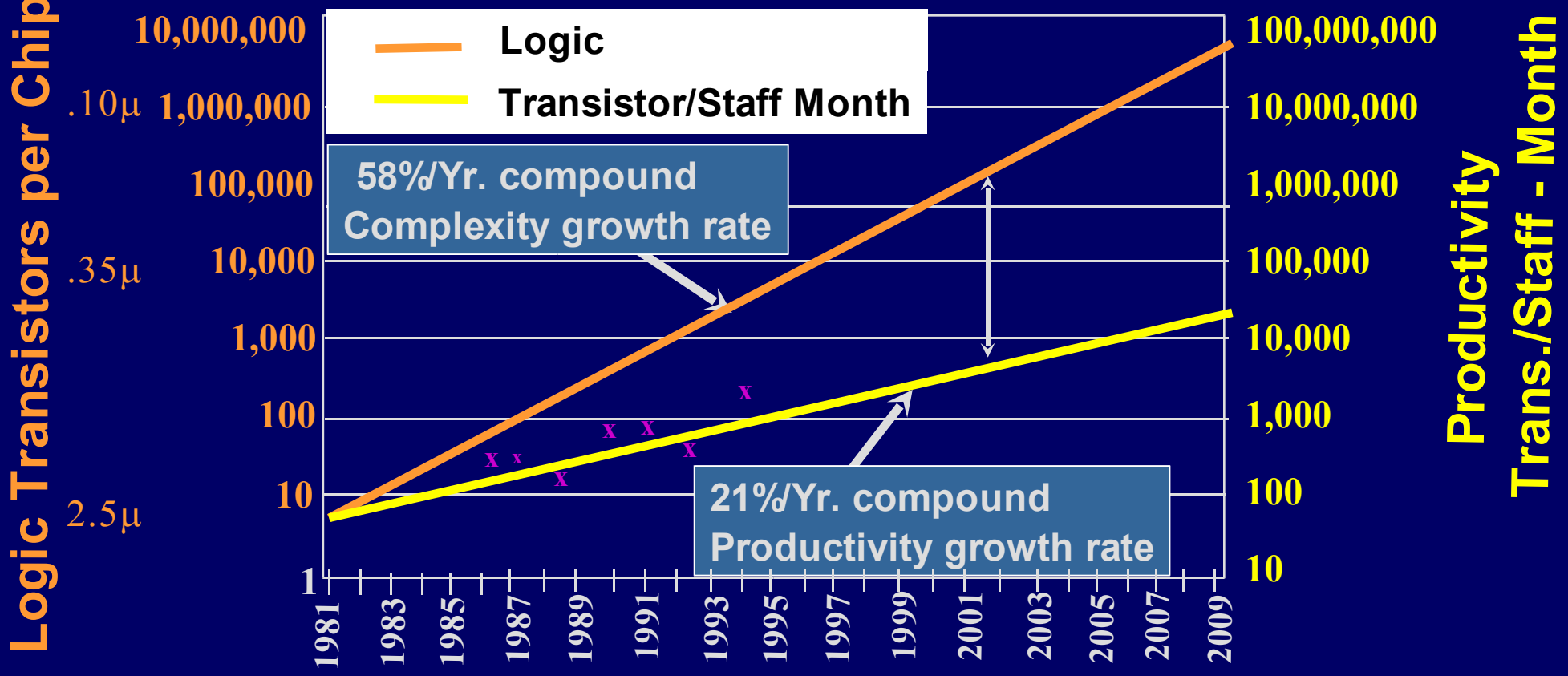
Xilinx FPGAs: XC4000, Virtex, Virtex II



Simplified CLB Structure of Xilinx FPGAs



The VLSI CAD Productivity Gap



Source: SEMATECH

Overview of Custom Computing

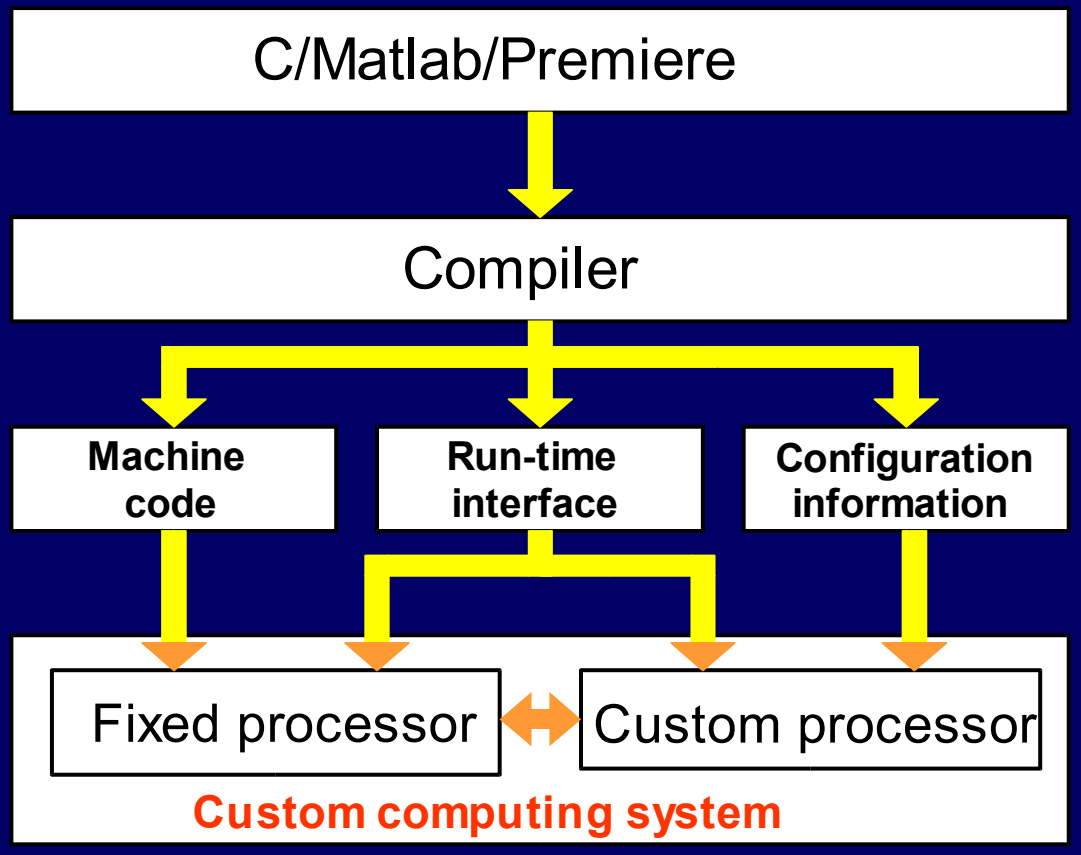
1. FPGAs: what and why
- FPGA, the big picture
- FPGA devices

2. Compiling to FPGAs
- architecture generation
- compiler passes

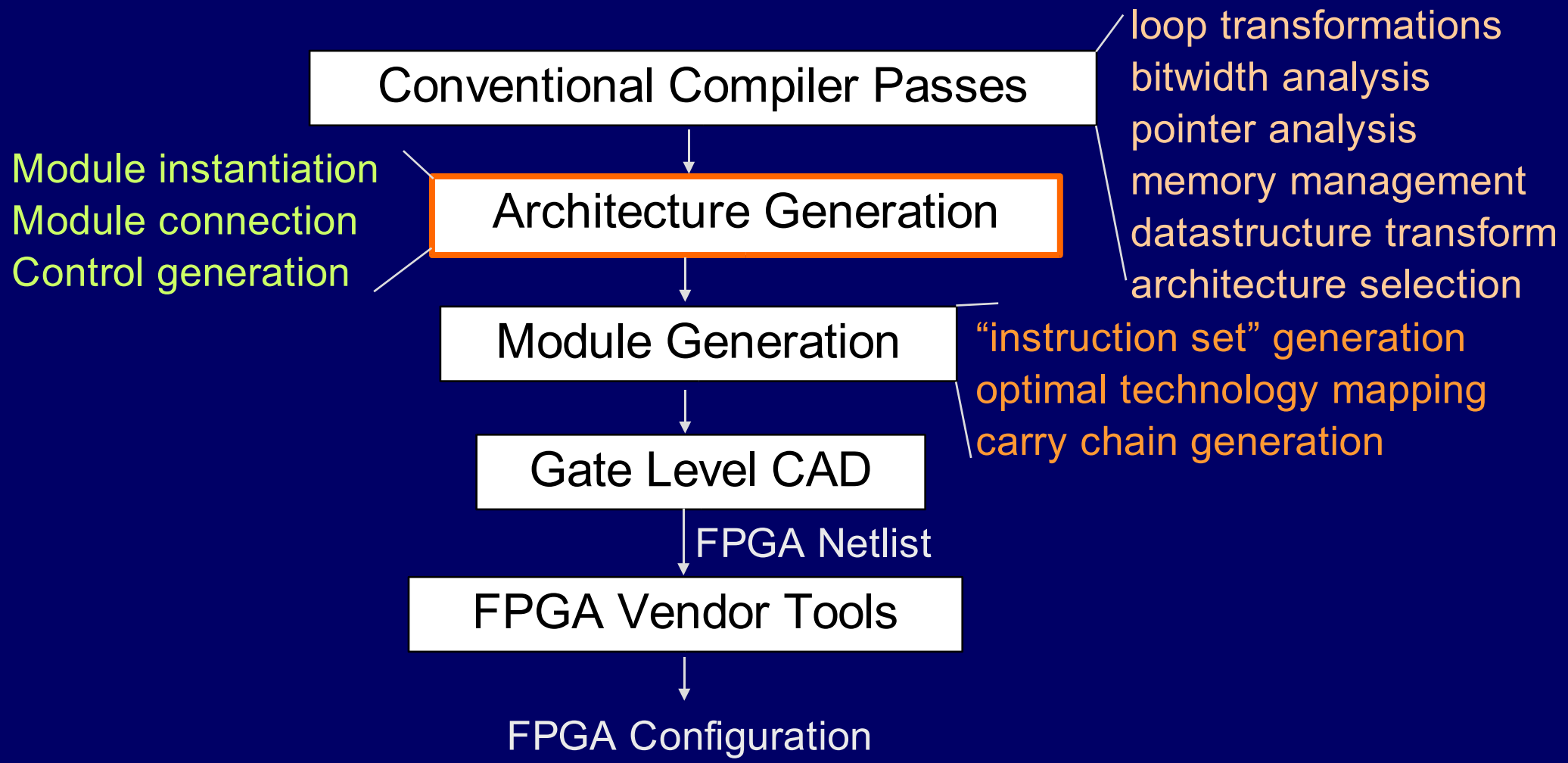
3. Run-time reconfiguration
- generating configurations
  - at compile time, at run time
- scheduling reconfigurations
  - control driven, data driven

4. Putting it altogether
- system view
- PAM: a PC-size system
- SONIC: a PCI card

Programming FPGAs Toolflow



The "Compiler" for FPGAs

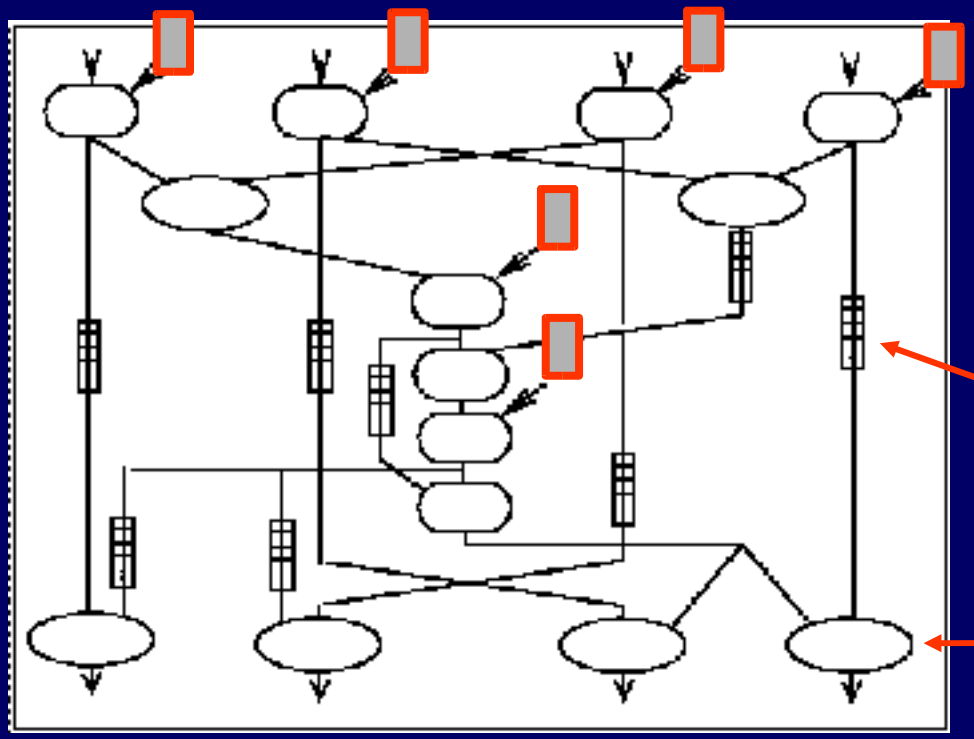


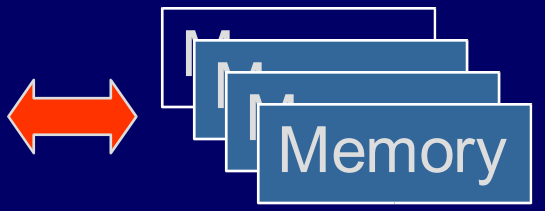
Languages for Architecture Generation

- generate particular instances of a generic architecture, such as a signal processor, a stream architecture, a neural network, a cellular automaton, etc…
- HW Description Languages: VHDL, Verilog
- SW Languages used for Custom Computing
  - C/C++, Java, Ruby, etc.
- C++ example: ASC – A Stream Compiler

ASC - A Stream Compiler for FPGAs

Bell Labs



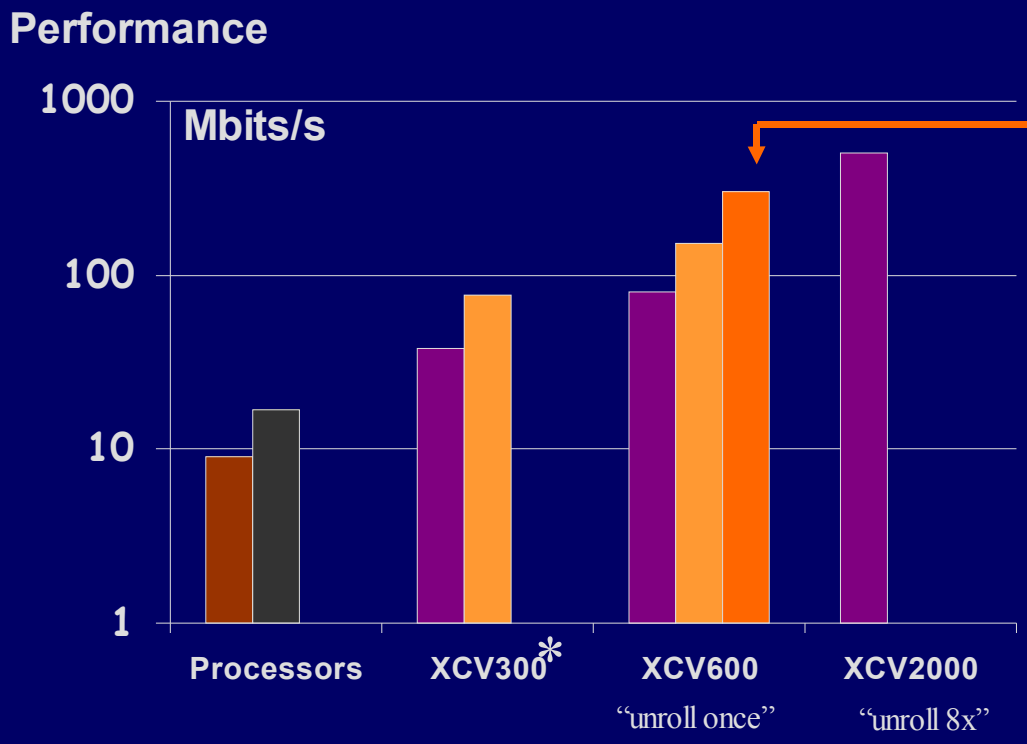


features

- distributed registers
- distributed delay FIFOs
- local memory blocks
- external Memory
- PAM-Blox II modules (pre-pipelined)

ASC generates Stream Architectures based on C++ input.

ASC Example: IDEA Encryption



high $\frac{\text{computation}}{\text{communication}}$ ratio

```
LOOP(8);
  LoopIndex(i);
  OPTIMIZE=REDUNDANT;

  word1 = HWmul(word1, key[i*6+0]);
  word2 = word2 + key[i*6+1];
  word3 = word3 + key[i*6+2];
  word4 = HWmul(word4, key[i*6+3]);

  t2 = word1 ^ word3;
  t2 = HWmul(t2, key[i*6+4]);

  t1 = (t2 + (word2 ^ word4));
  t1 = HWmul(t1, key[i*6+5]);
  t2 = (t1 + t2);

  word1 ^= t1;
  word4 ^= t2;

  t2 ^= word2;
  word2 = word3 ^ t1;
  word3 = t2;
LOOP_END();
```

Overview of Custom Computing

1. FPGAs: what and why

FPGA, the big picture

FPGA devices

programming FPGAs

- module generation

2. Compiling to FPGAs

architecture generation

compiler passes

4. Putting it altogether

system view

PAM: a PC-size system

SONIC: a PCI card

3. Run-time reconfiguration

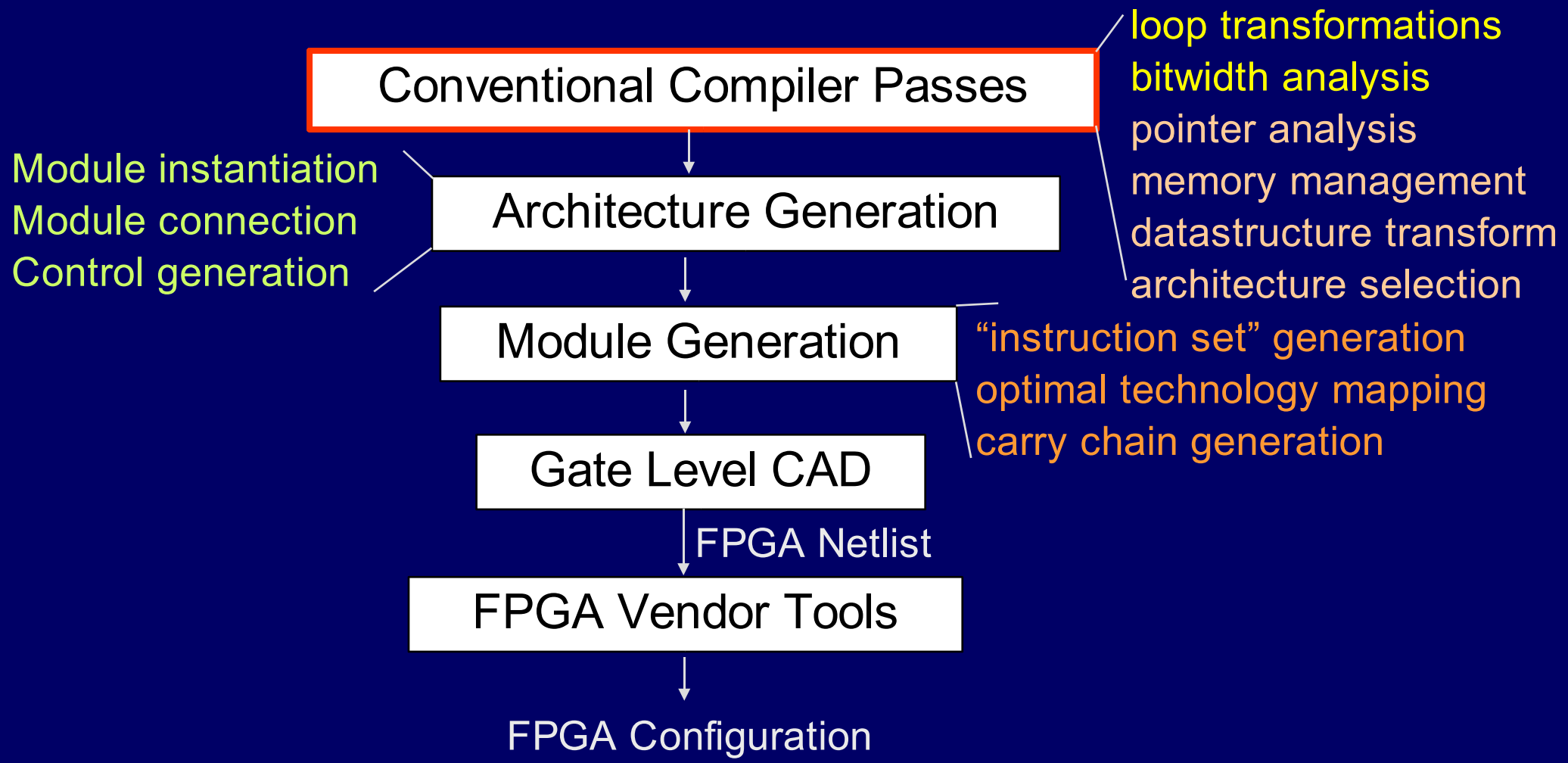
generating configurations

- at compile time, at run time

scheduling reconfigurations

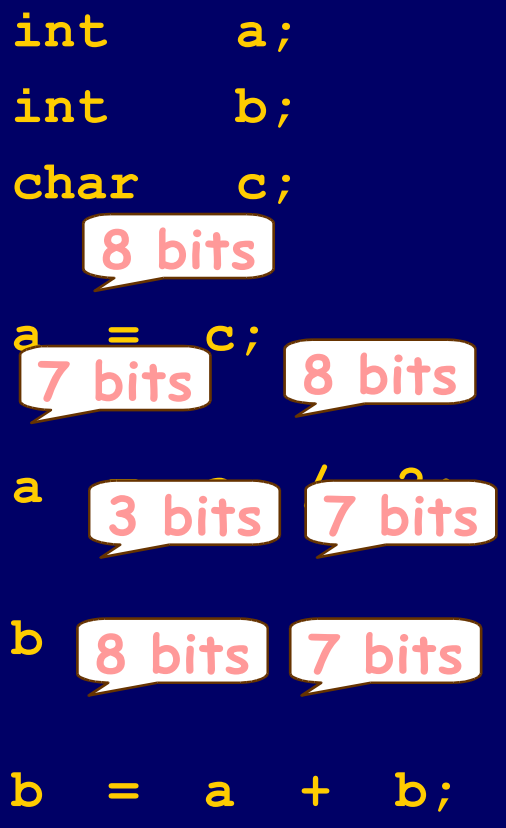
- control driven, data driven

The "Compiler" for FPGAs



Bitwidth Analysis (Type Size Inference)

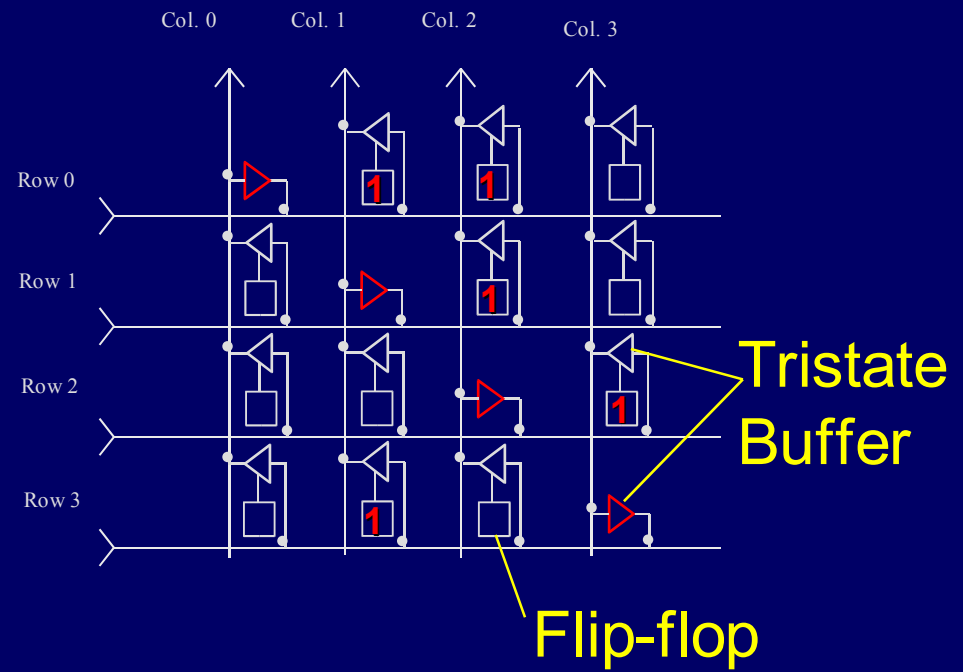
- for programs written in a high-level language
- **minimum bitwidth** required for
  - each variable at every static location of the program
  - each static operation of the program
- reduces memory bandwidth dependency, MBD (slide 15, above)



HAGAR: Hardware Graph Accelerators

Memory Latency Dependent, e.g. data structures, pointers
→ implement data-structure + algorithm in hardware

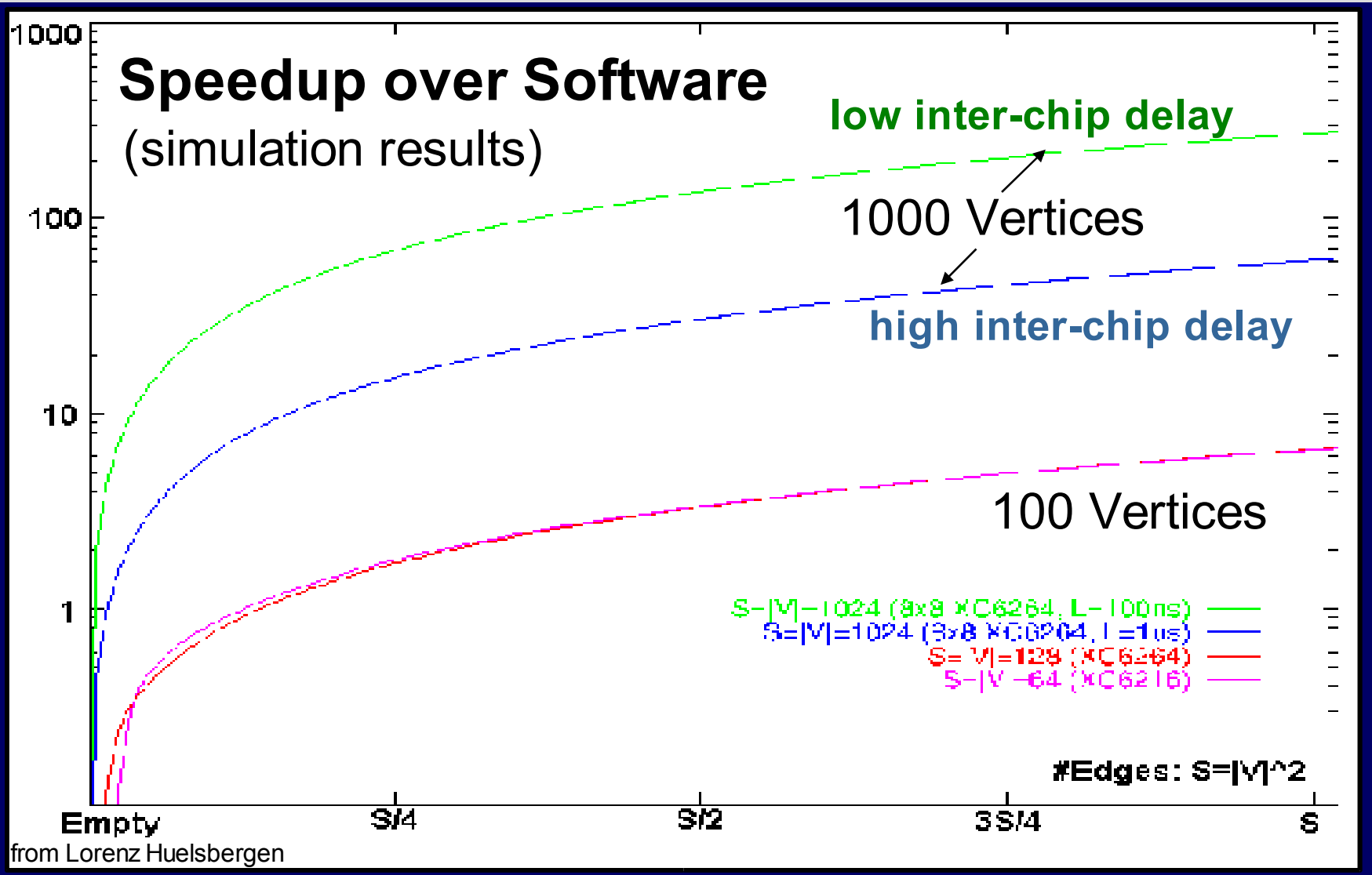
$$\begin{array}{c} \\ v_0 \\ v_1 \\ v_2 \\ v_3 \end{array} \begin{array}{c} \begin{array}{cccc} v_0 & v_1 & v_2 & v_3 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$



operations: insert, delete, reachability, …

reachability speedup over PC: 10× to 1000× for 1K nodes

Speedup of Reachability



from Lorenz Huelsbergen

Overview of Custom Computing

1. FPGAs: what and why

FPGA, the big picture

FPGA devices

programming FPGAs

- module generation

2. Compiling to FPGAs

architecture generation

compiler passes

4. Putting it altogether

system view

PAM: a PC-size system

SONIC: a PCI card

3. Run-time reconfiguration

generating configurations

- at compile time, at run time

scheduling reconfigurations

- control driven, data driven

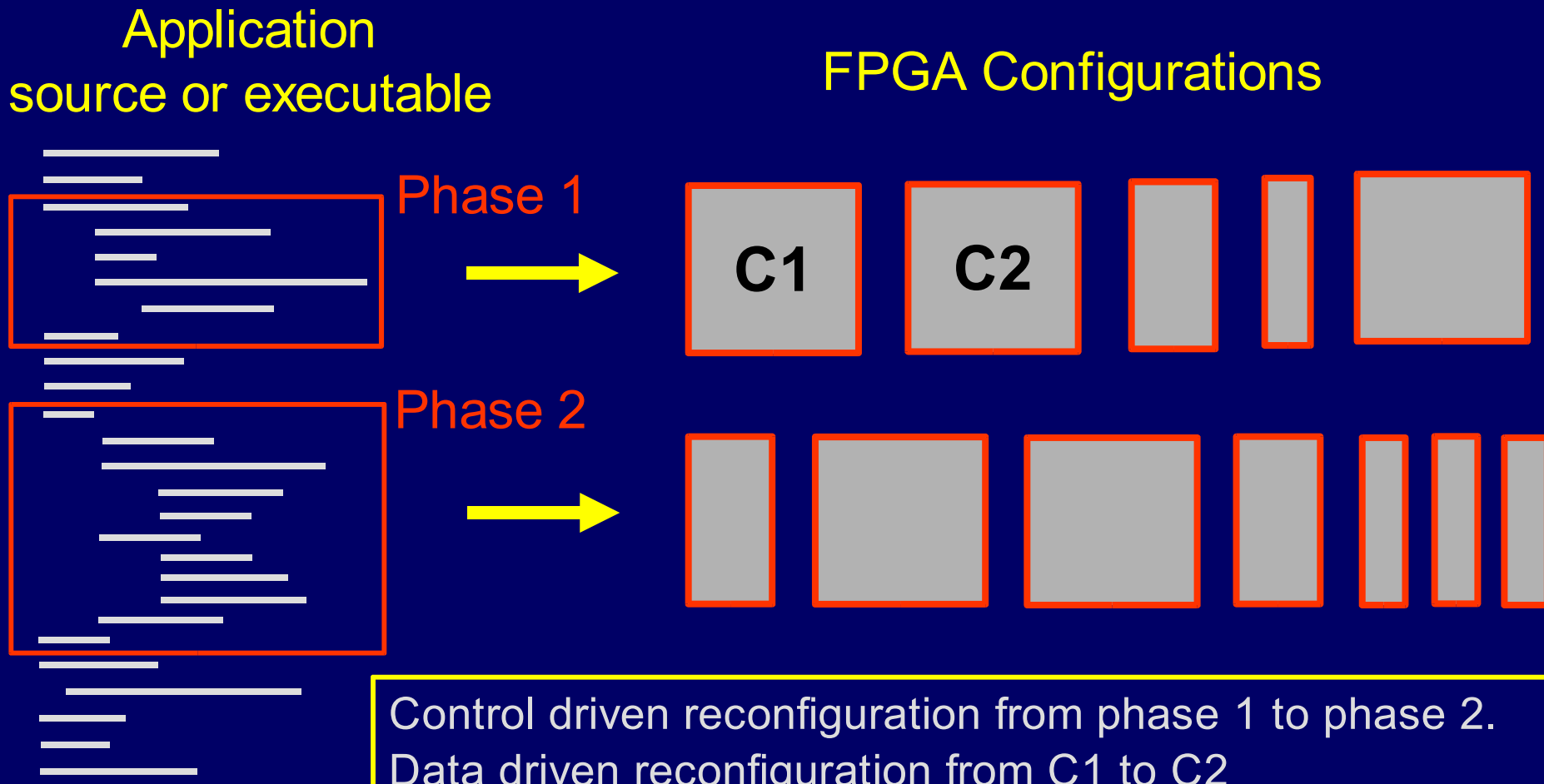
Run-time Reconfiguration

FPGAs are reconfigurable within milliseconds!

1. generating configurations
   - at compile time
   - at run time

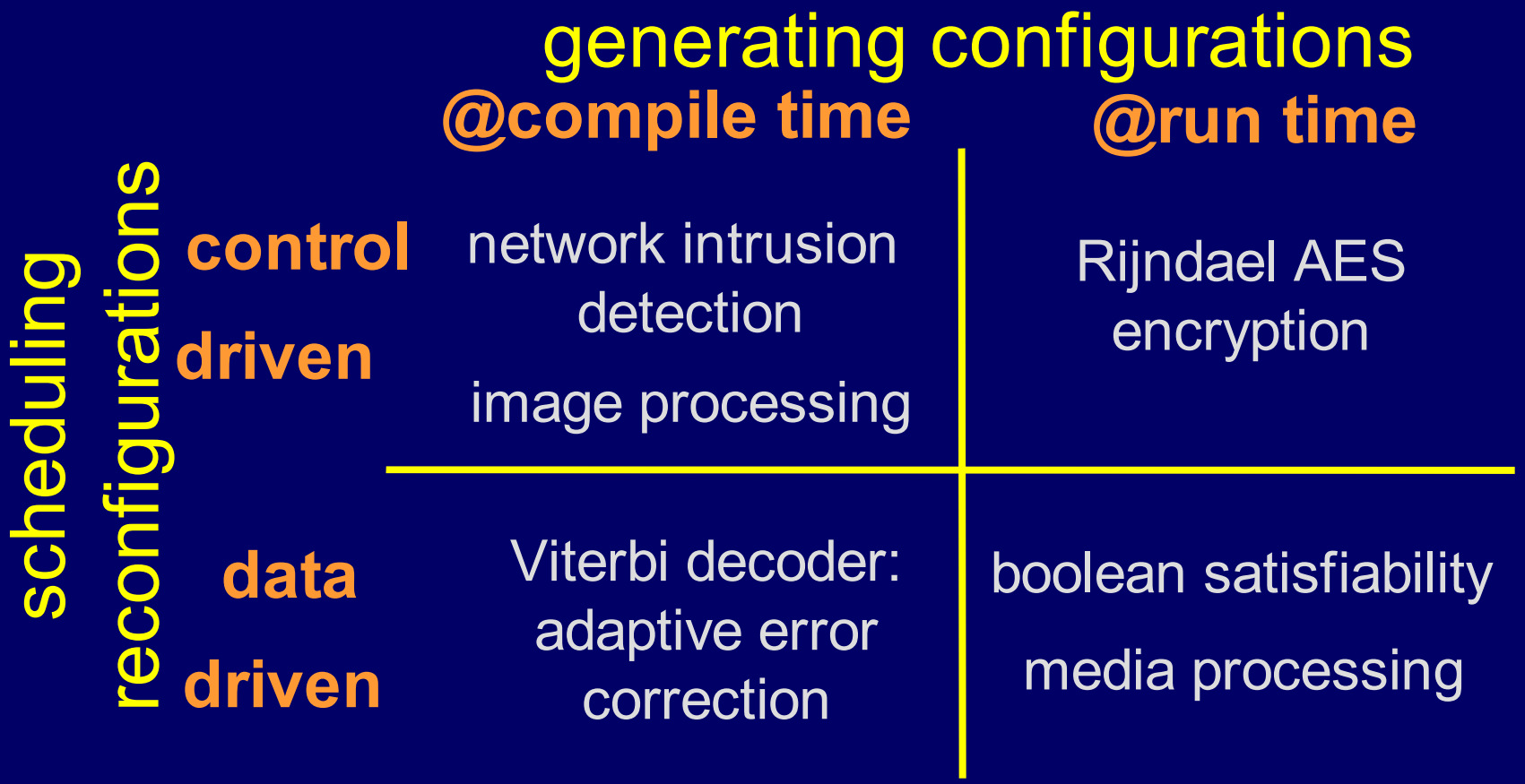
2. scheduling reconfigurations
   - at compile time - control driven
   - at run time – data driven

Generating & Scheduling Reconfigurations



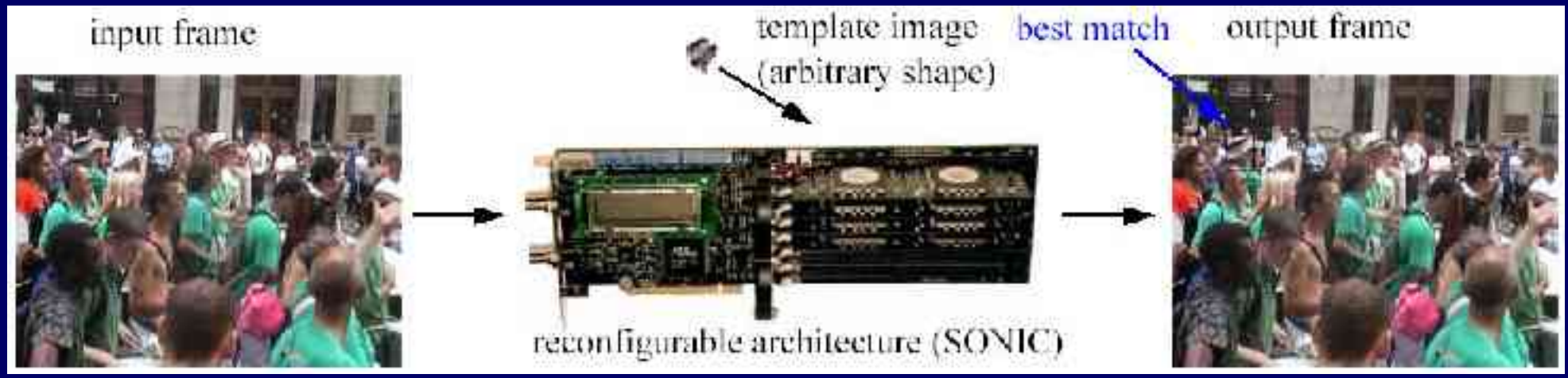
Control driven reconfiguration from phase 1 to phase 2.
Data driven reconfiguration from C1 to C2
=> insert reconfiguration strategy (algorithm) into application

Generating & Scheduling Reconfigurations



Media Processing Example

- shape-adaptive template matching (SATM)
  - MPEG4, MPEG7: find arbitrarily shaped object in video



- template contains object of interest
- Sum of Absolute Distances of template and video

- adapt design to template and search frame

SATM: FPGA versus PC

- HDTV frames: 1920 by 1080 pixels
- for 300 HDTV frames, use PC time as reference

T_{PC} : execution time of PC with 1.4GHz Pentium 4

S_D : speedup of dynamic design

S_{PD} : speedup of partially dynamic design

S_S : speedup of static design

$w \times h$	T_{PC}	S_D	S_{PD}	S_S
10×10	1,015 sec	108	57	36
100×100	88,609 sec	6,970	5,001	3,180

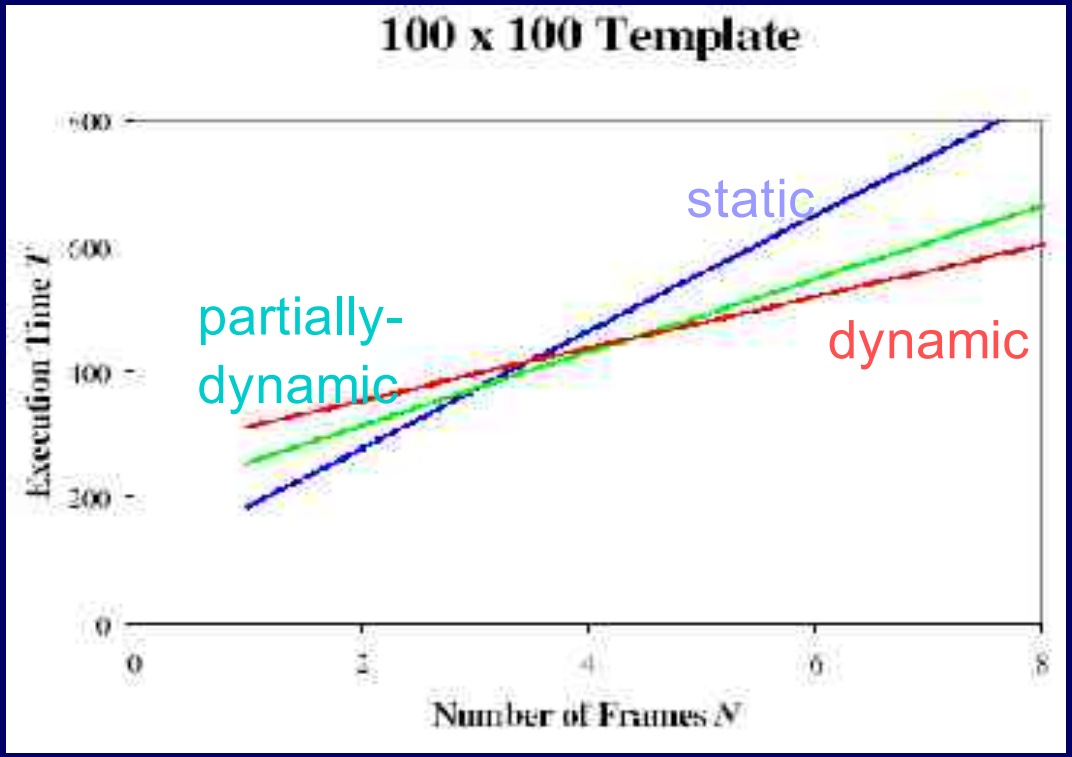
SATM: Performance Results

- dynamic design superior when:
  - a large number of consecutive frames
  - templates do not change often
  - small templates

T_D : time for dynamic design

T_S : time for static design

T_PD : time for partially dynamic design



Overview of Custom Computing

1. FPGAs: what and why

FPGA, the big picture

FPGA devices

programming FPGAs

- module generation

2. Compiling to FPGAs

architecture generation

compiler passes

- loop transformations
- bitwidth analysis

3. Run-time reconfiguration

generating configurations

- at compile time, at run time

scheduling reconfigurations

- control driven, data driven

4. Putting it altogether

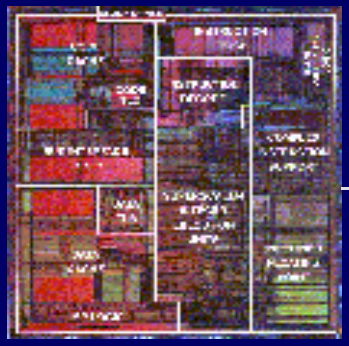
system view

PAM: a PC-size system

SONIC: a PCI card

System View of Custom Accelerators

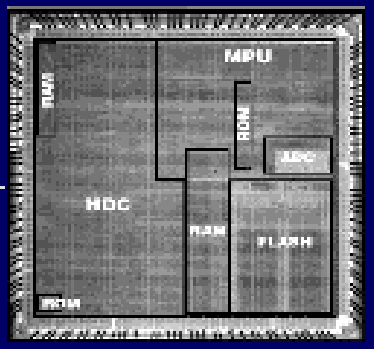
Microprocessor



Intel Pentium

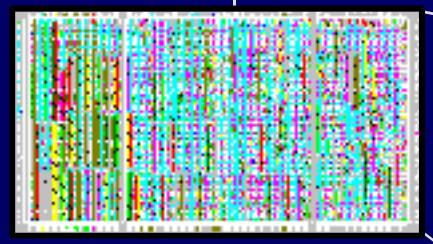
Interconnect

Memory



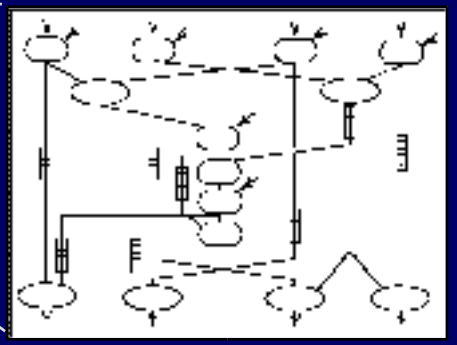
from the IBM website

Local Area Network
PCI Bus
PC Card/Cardbus
Memory Bus / DIMM
Custom Interconnect
On-chip Interconnect

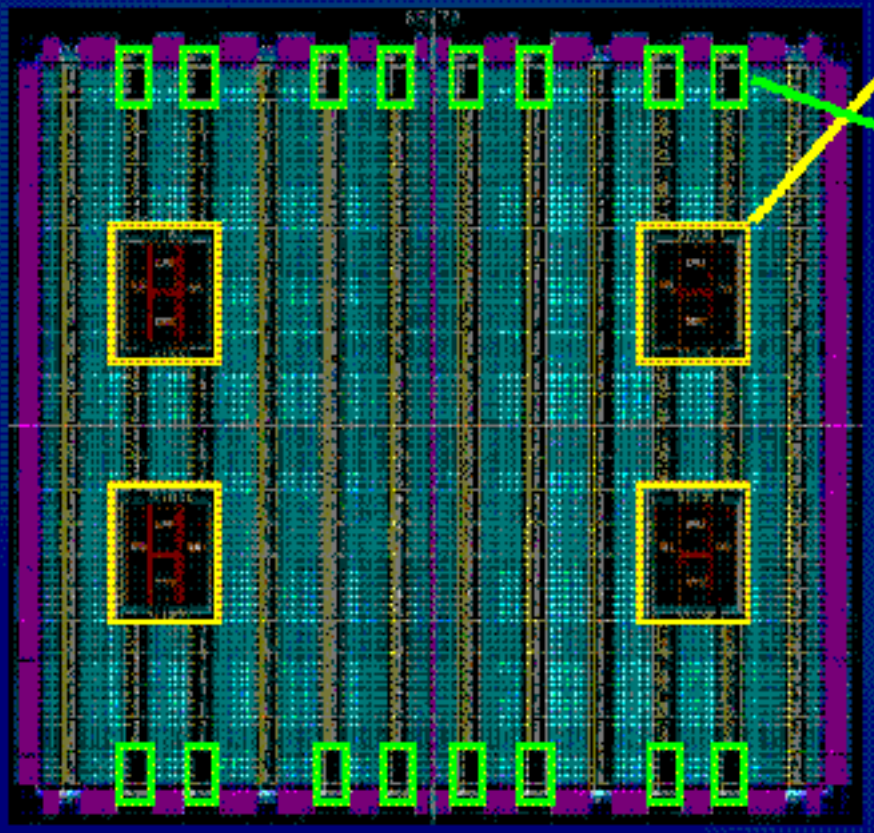


FPGA

"Custom Accelerator"



Xilinx Virtex-II Pro



Virtex-II Pro 2VP50

Source: Xilinx

Virtex-II Base Architecture

- IBM PowerPC CPU
- RocketIO™ MGT
  - Multi-Rate : 3.125, 2.5, 2.0, 1.2, 1.0 Gb/sec
  - Multi-Protocol
    - Gb Ethernet
    - 10Gb Ethernet (XAUI)
    - Intel 3GIO
    - Serial ATA
    - Infiniband
    - FibreChannel
    - RapidIO
    - Serial Backplanes

Case Studies

1 PAM project at DIGITAL PRL (Compaq)
- various applications on multi-FPGA platform

2 SONIC project at Imperial College and Sony
- broadcast-quality video processing PCI card

Case Study 1: The PAM Project

Programmable Active Memories (source: Mark Shand)

People:

Jean Vuillemin
Patrice Bertin
Didier Roncin
Herve Touati
Philippe Boucard
Harry Printz
Mark Shand

Goals:

Maximum performance
Rapid turnaround
Exploring how to build and program PAM
Exploring the application space
(Non-goals: high-level synthesis, platform independence, improving FPGAs)

4 Platforms:

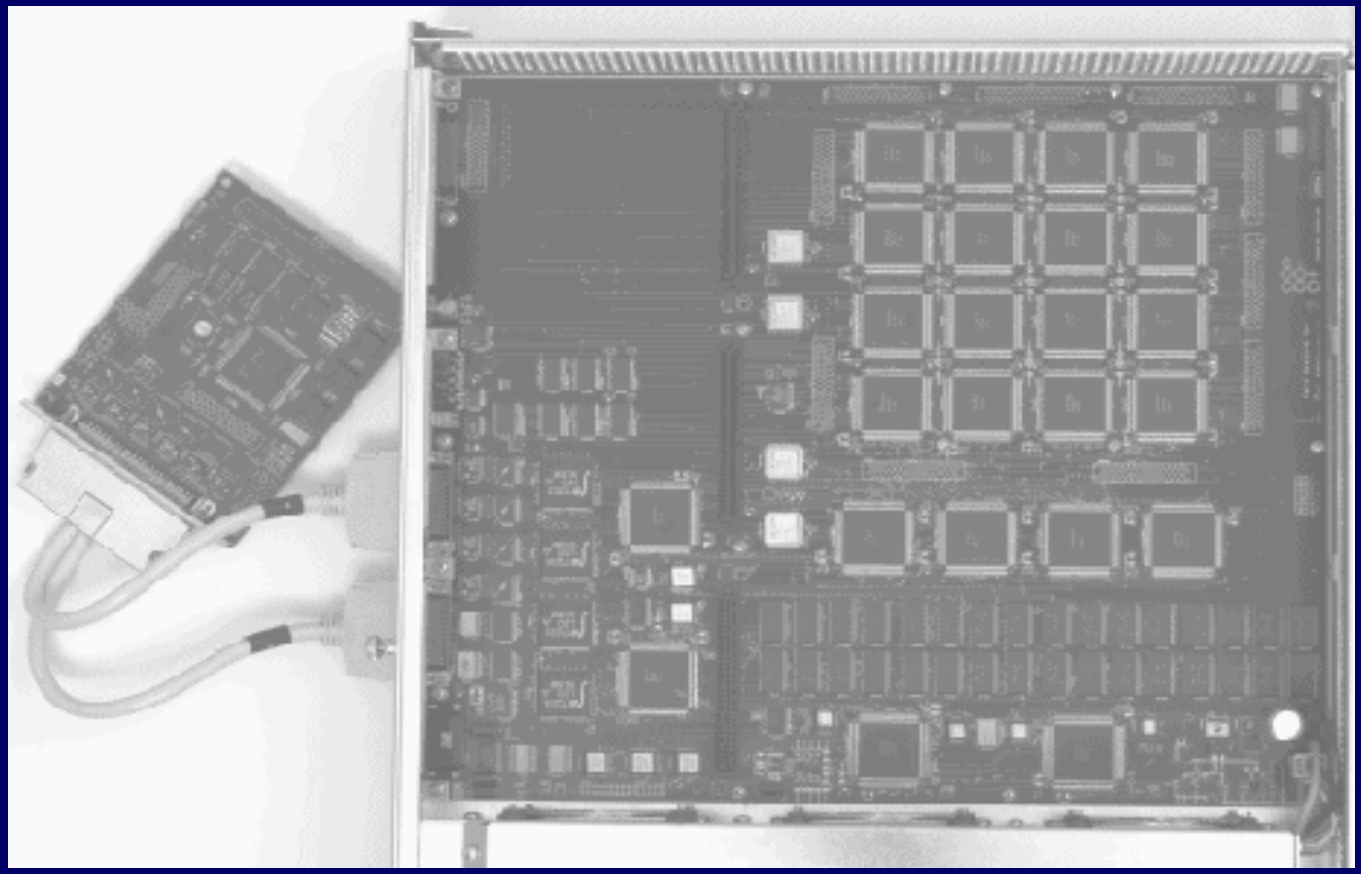
- Perle-0 (1989)
  50k gate reconfigurable VME board
- DECPeRLe-1 (1992)
  200k gates, 20ms turnaround
- TURBOchannel Pamette (1994)
- PCI Pamette (1996)
  PCI card with 40K - 112K gates

Applications:

Long Int. Multiplication
RSA Cryptography
Dynamic Programming
Laplace Heat Equation
Viterbi Decoder
Sound Synthesis
Neural Networks
Stereo Vision
Hough Transform
High Energy Physics
Image Aquisition
Wireless LAN testbed

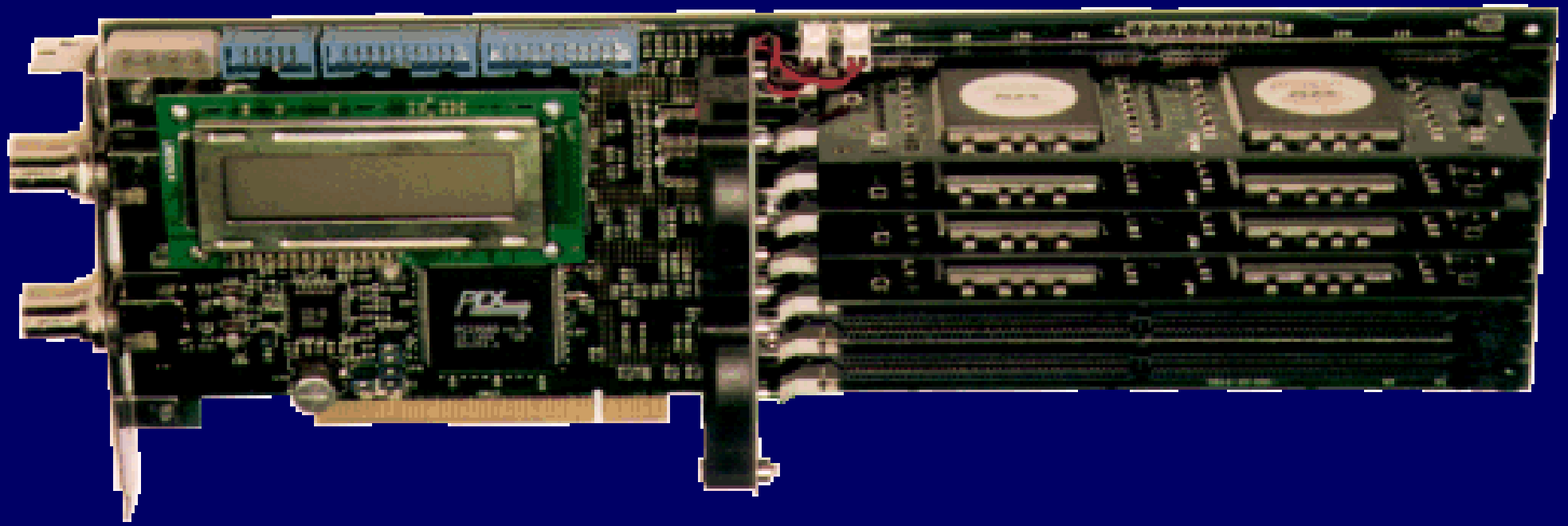
All applications are implemented by hand at the gate level using PamDC.

DECPeRLe-1



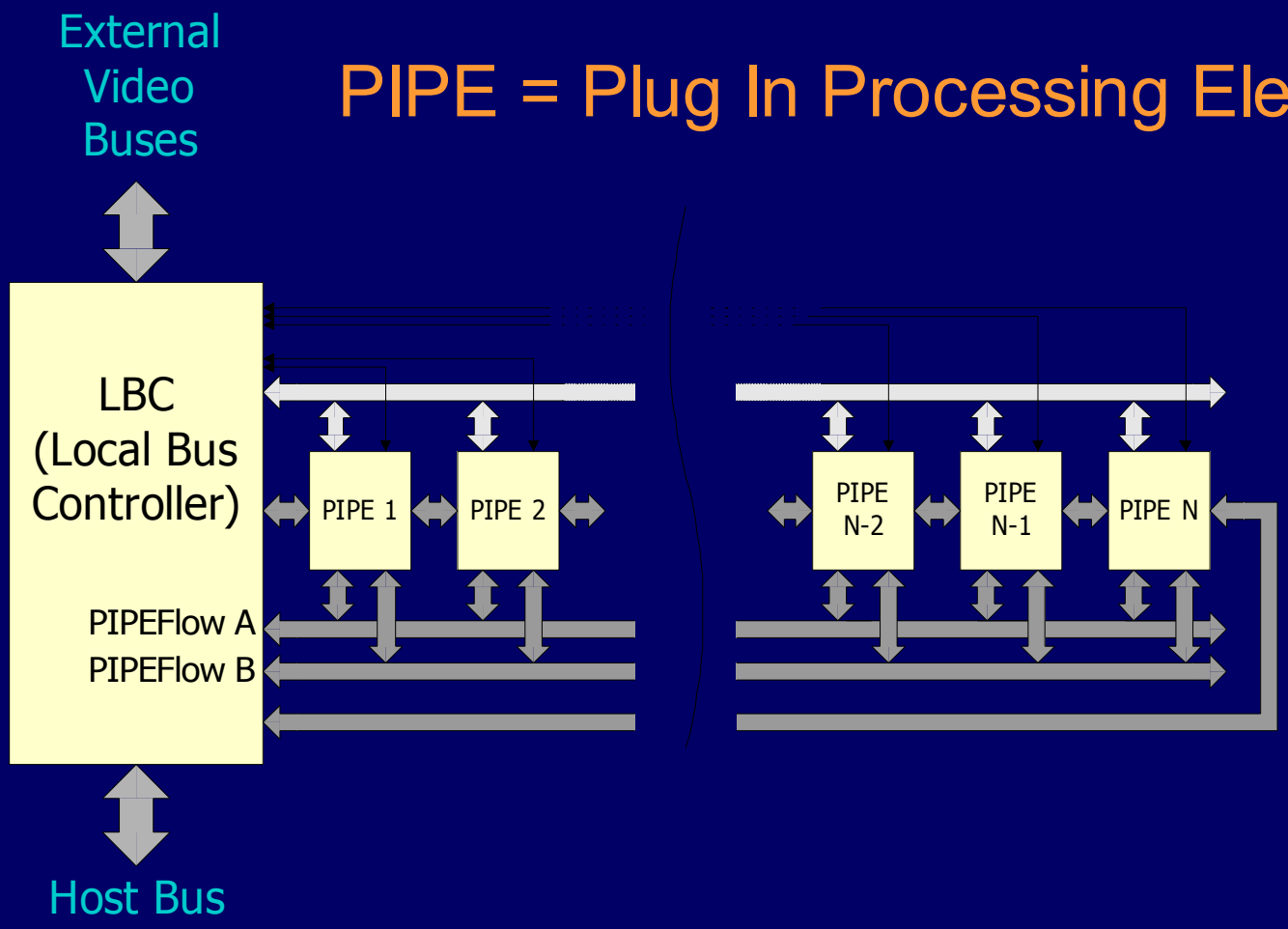
Case Study 2: SONIC

professional video processing at SONY and Imperial College

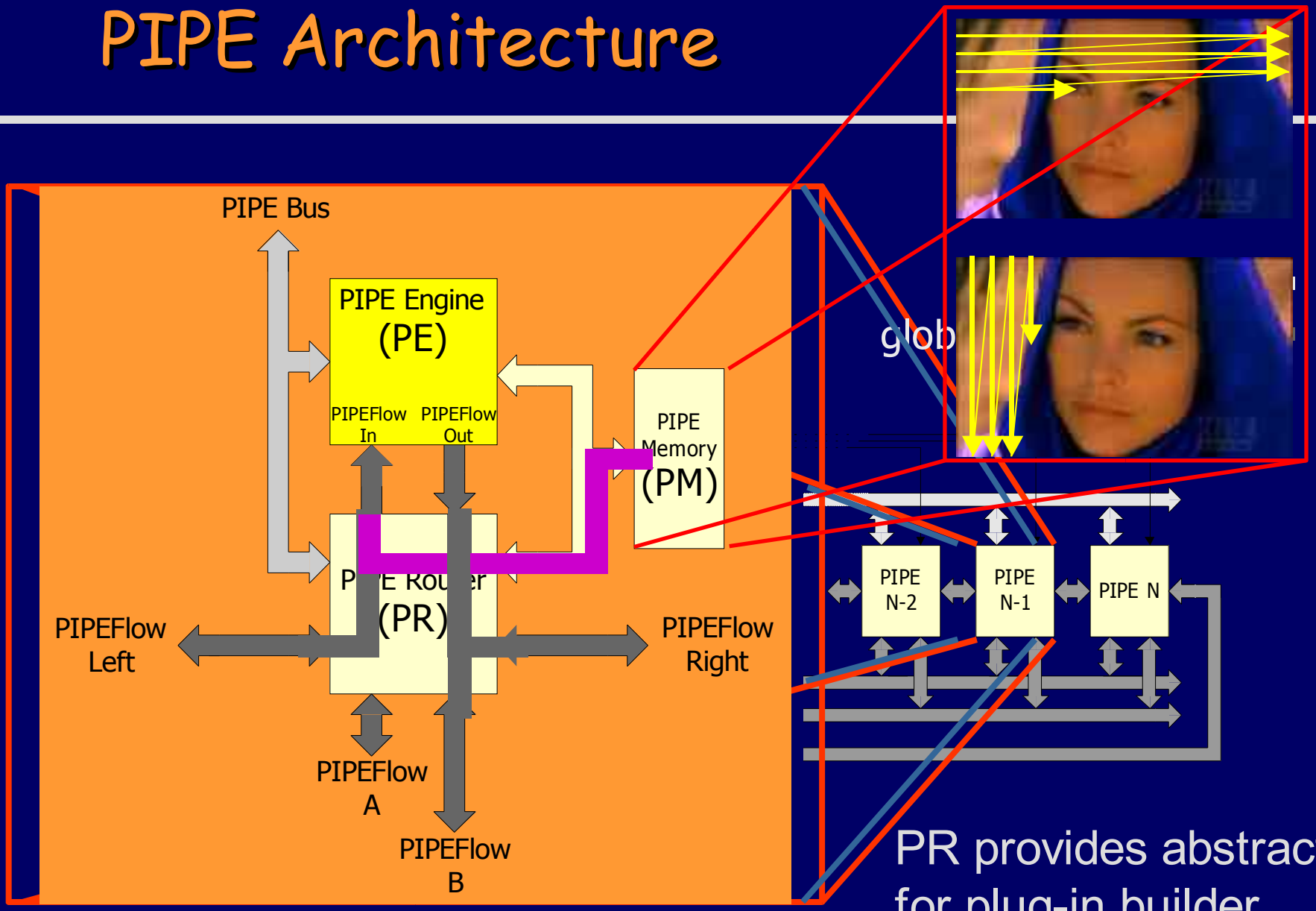


SONIC architecture

PIPE = Plug In Processing Element



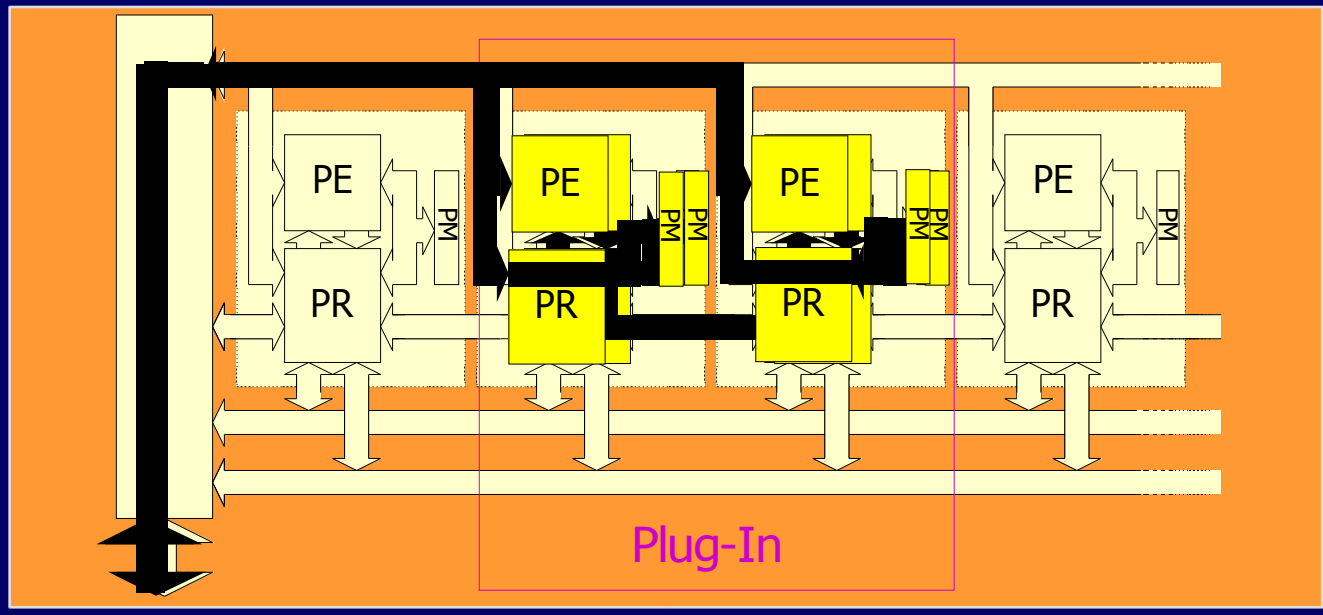
PIPE Architecture



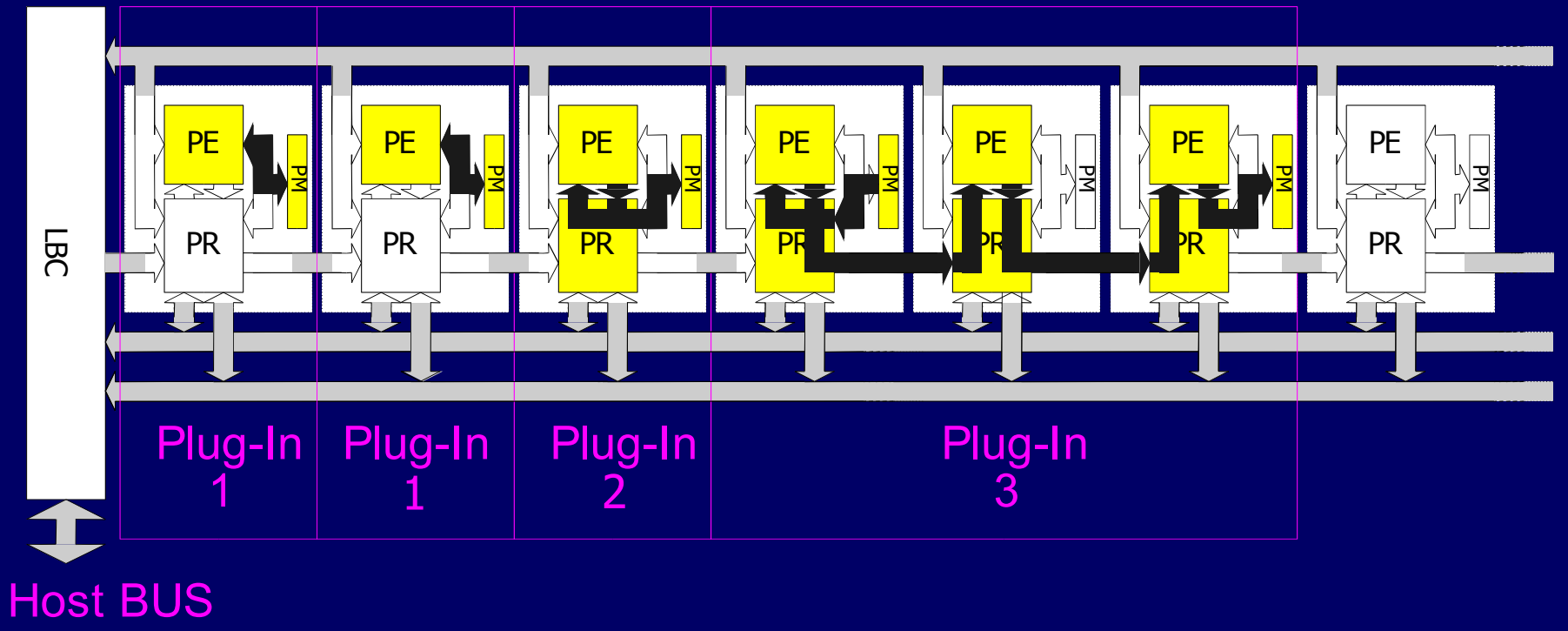
PR provides abstraction for plug-in builder

SONIC Processing an Image

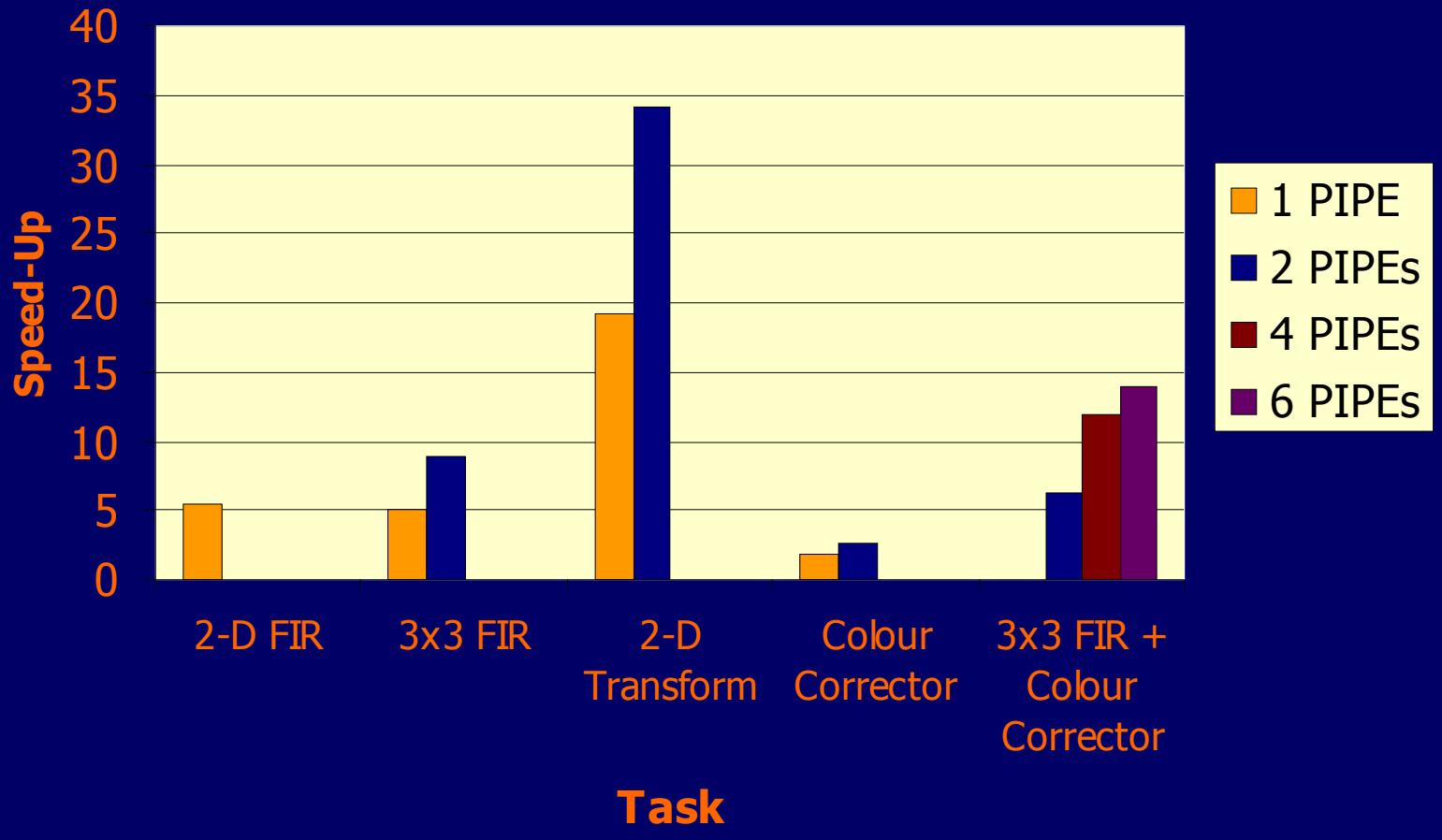
- Configuration of Plug-In
- Image Transfer In
- Processing
- Image Transfer Out



SONIC with Multiple Plug-ins



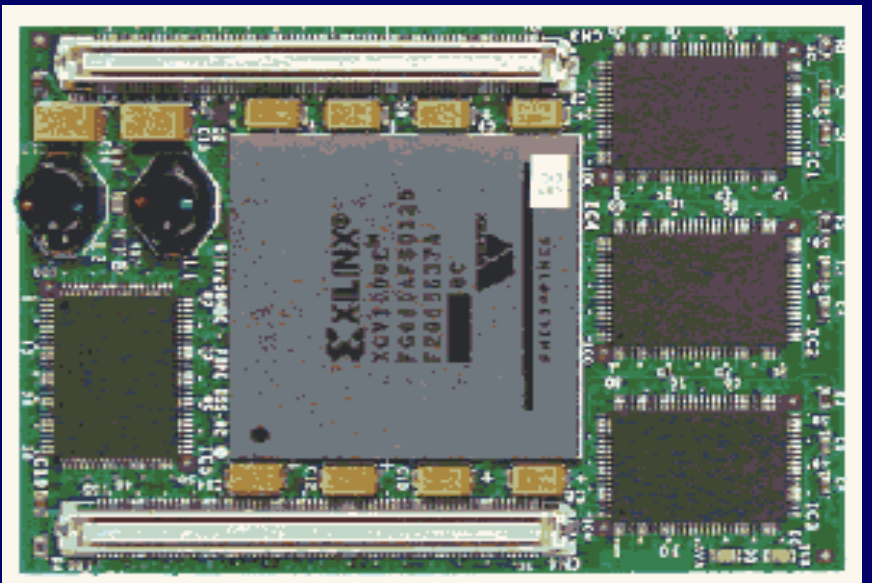
SONIC versus Pentium

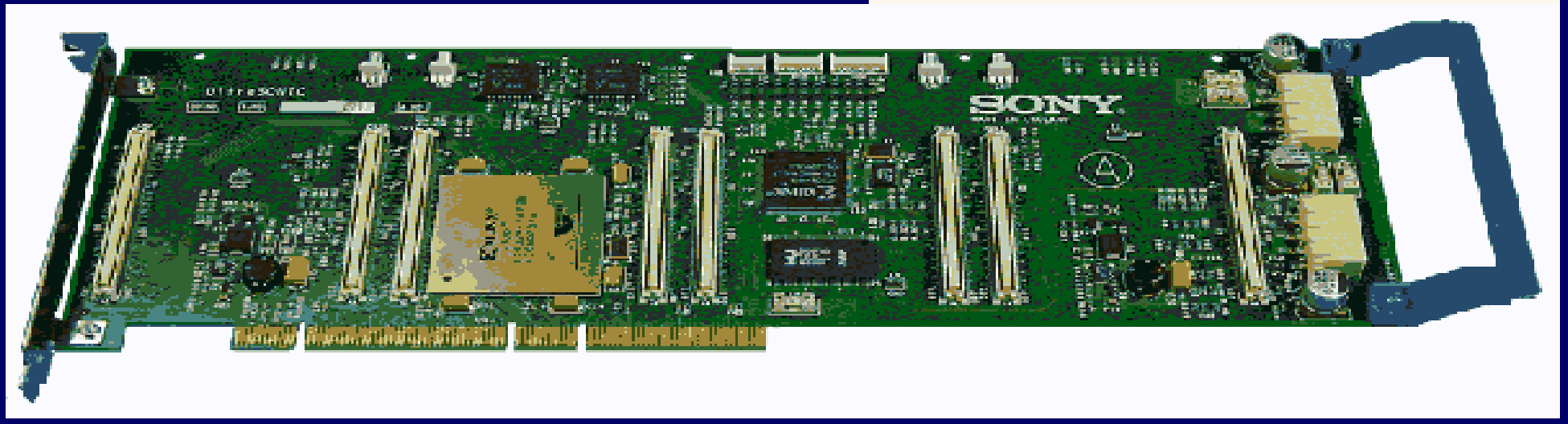


Speedup Compared with an MMX Pentium II 300Mhz

Current Generation: UltraSONIC

- PE, PR on one FPGA
- 8MB SRAM on PIPE
- up to 16 PIPEs
- 64-bit, 66MHz PCI
- real-time image registration





Pointers to Conferences and Journals

FPGA Conferences

- FPGA Conference
- FCCM
- FPL
- FPT
- ERSA

Hardware Design

- ICCAD
- DAC

Computer Architecture

- ISCA, ISC,
- HPCA, ASPLOS,
- MICRO

Journals

IEEE Transactions on Computers

IEEE Transactions on VLSI

IEEE Transactions on CAD

Kluwer Journal on VLSI and Signal Processing

ACM Transactions on Architecture and Code Optimization