Trends in Programmable Solutions

SoC FPGAs for Embedded Applications and HW/SW Co-Design

Mike Hutton
Principal Investigator
Office of the CTO

Misha Burich
Sr. VP and Altera CTO

EDPS Apr 5, 2012

Agenda

- Market Drivers and New Enabling Technologies
- Embedded Computing with FPGAs
  - The Emergence of SoC FPGAs
  - Programmable Silicon Substrate
- Software-Programmable Hardware
  - OpenCL for Parallel Programming
  - System Integration
- Summary

Market Drivers and Enabling Embedded Technologies

Most Markets Rely on Embedded Processing

Consumer and Automotive

Communications and Broadcast

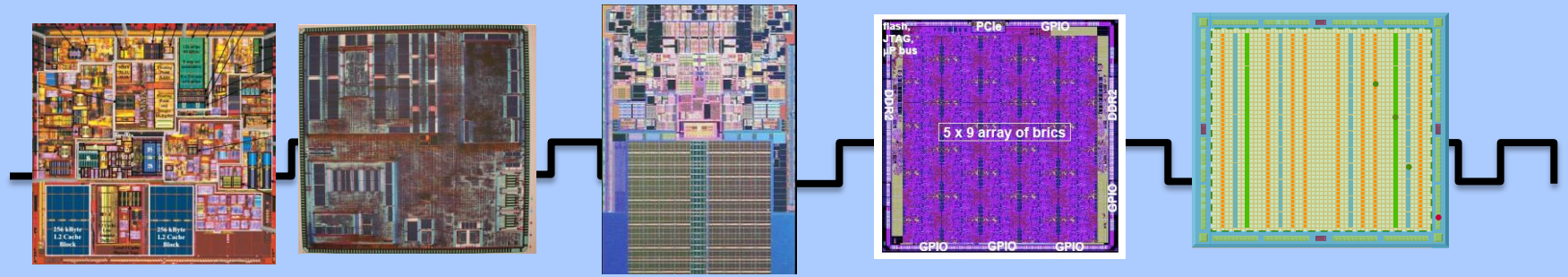
Computer and Storage

Test and Medical

Military and Industrial

Enabling Technologies: 200X-201X

- Moore's law enables high density programmable platforms



CPUs DSPs Multi-Cores Arrays FPGAs

Single Cores

Multi-Cores
Coarse-Grained
CPUs and DSPs

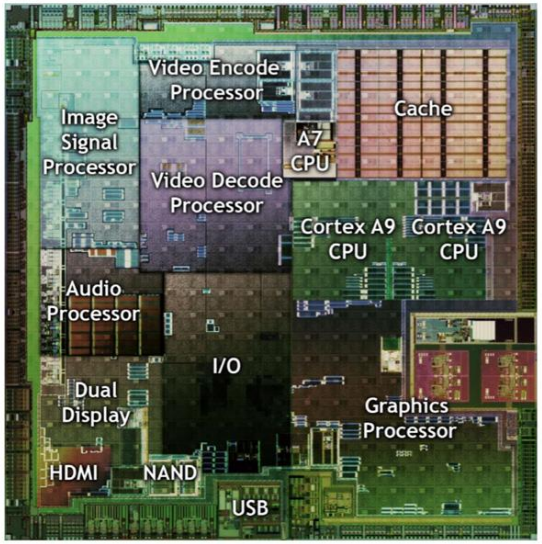
Coarse-Grained
Massively
Parallel
Processor
Arrays

Fine-Grained
Massively
Parallel
Heterogeneous
Arrays

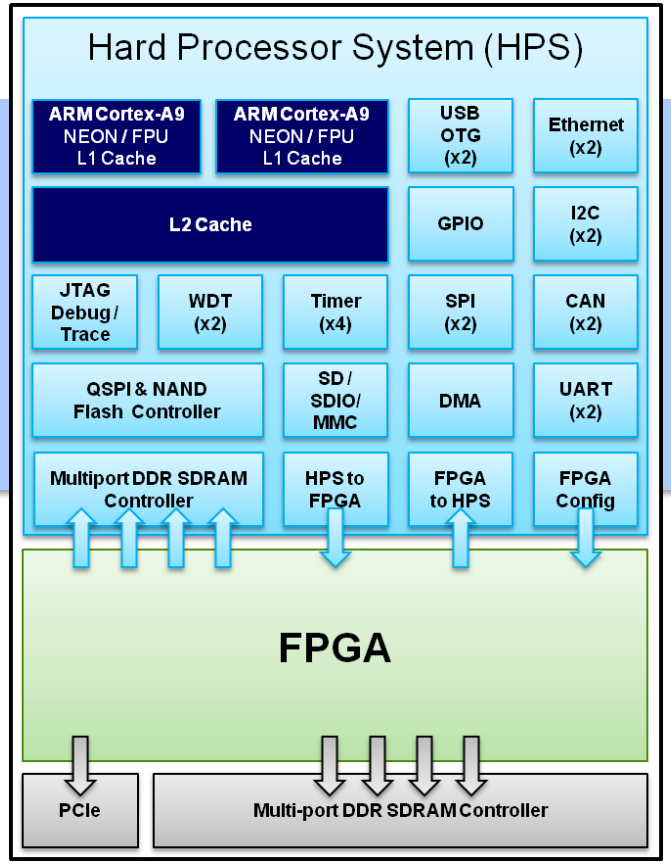
The Age of Systems-On-Chip

- ARM based SoCs by

Apple, Nvidia, Qualcomm, TI, others

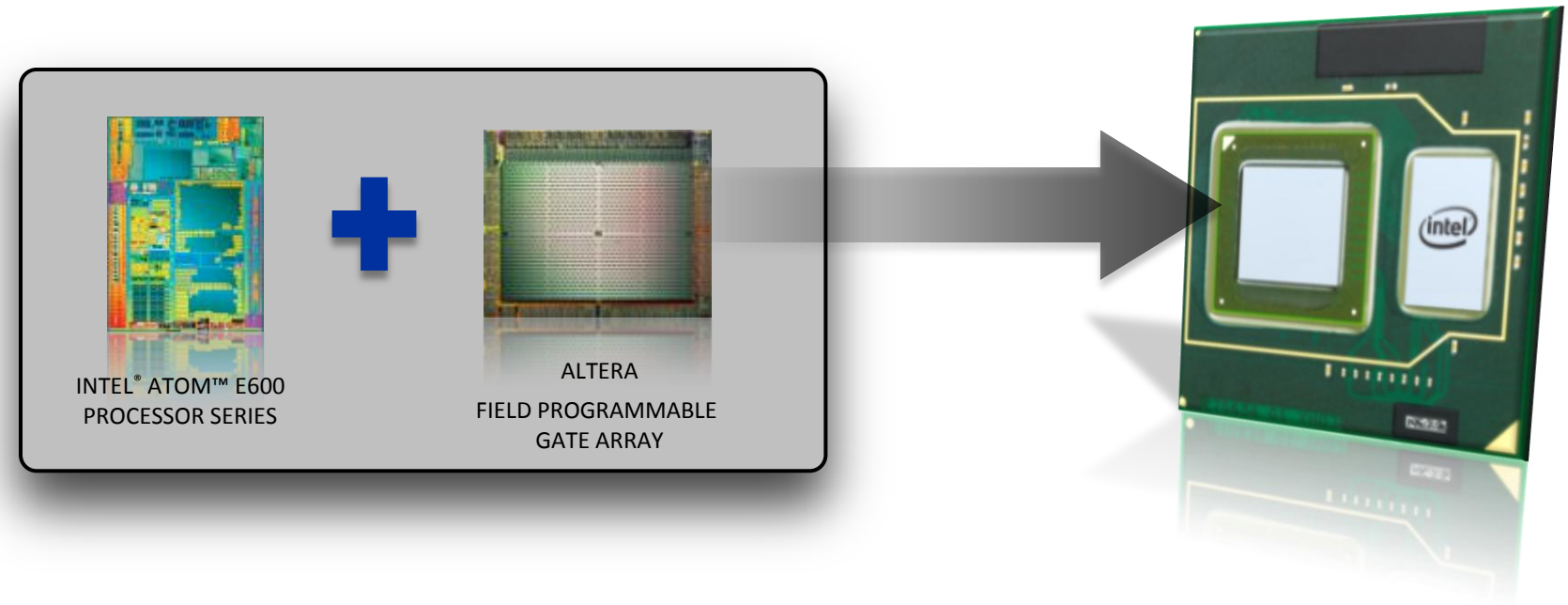


- SoC FPGAs by Altera, Xilinx, others



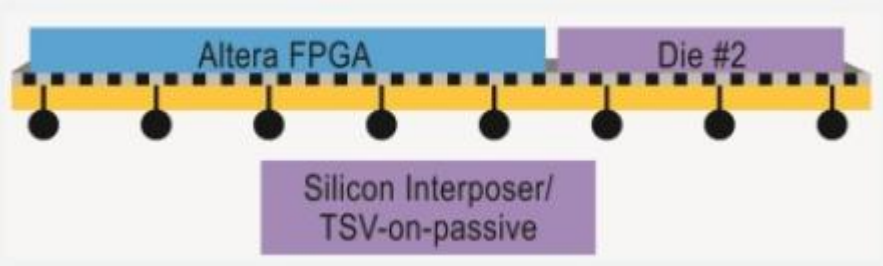
More-Than-Moore

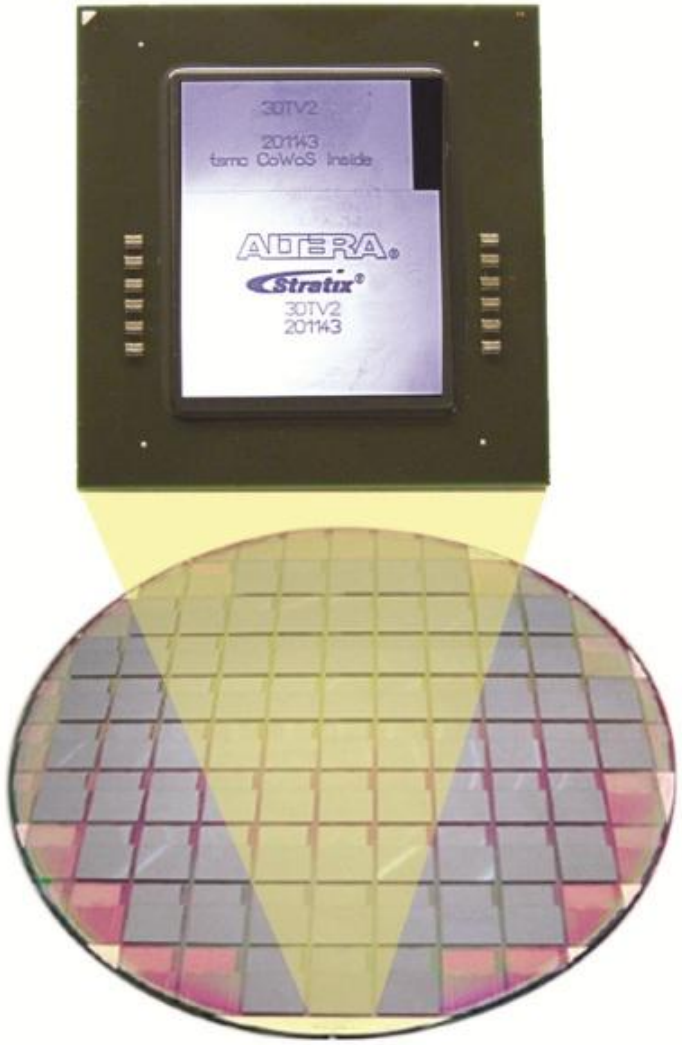
- System-in-package multi-die integration
  - POP, 2.5D, 3D, micro-bumps, through-silicon-vias (TSV)
- Example: Intel's integration of Atom processor with Altera's FPGA
  - E600C



First Heterogeneous 3D IC

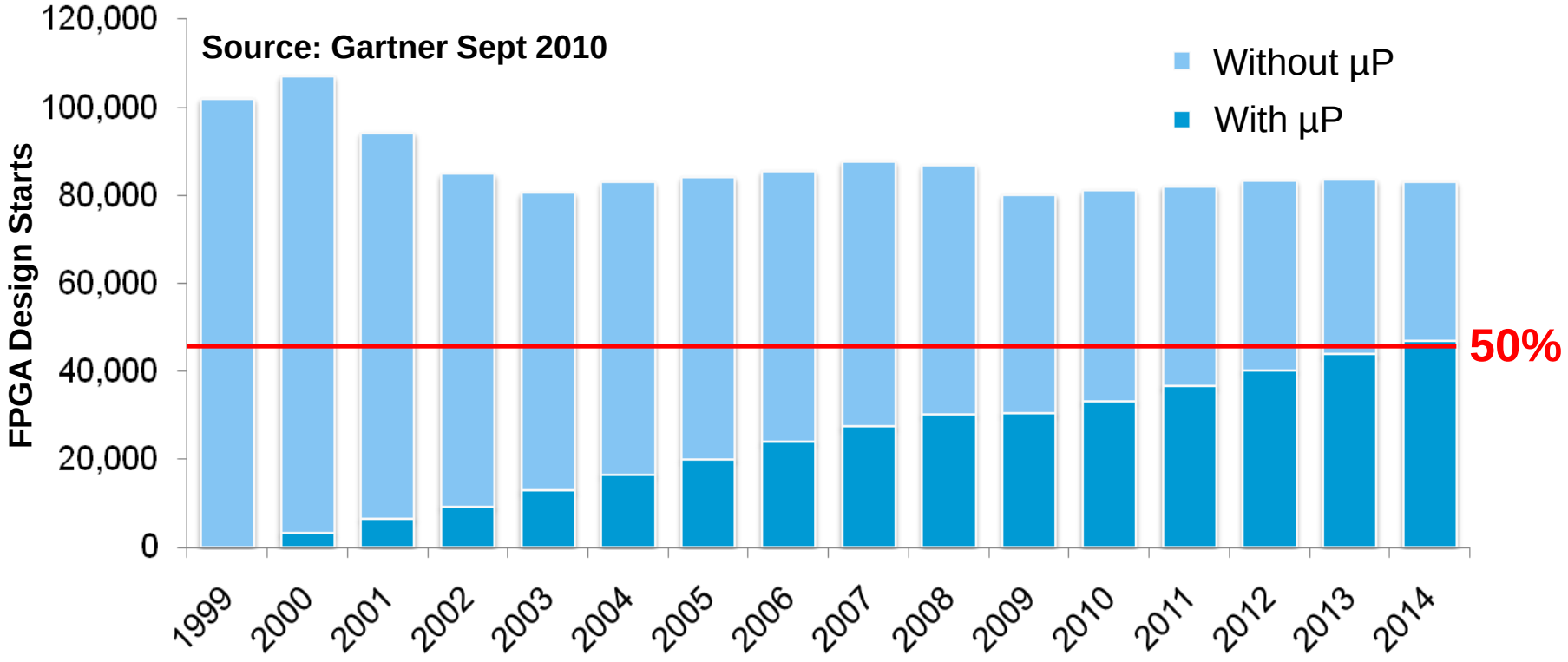
(Announced 3/22/2012)





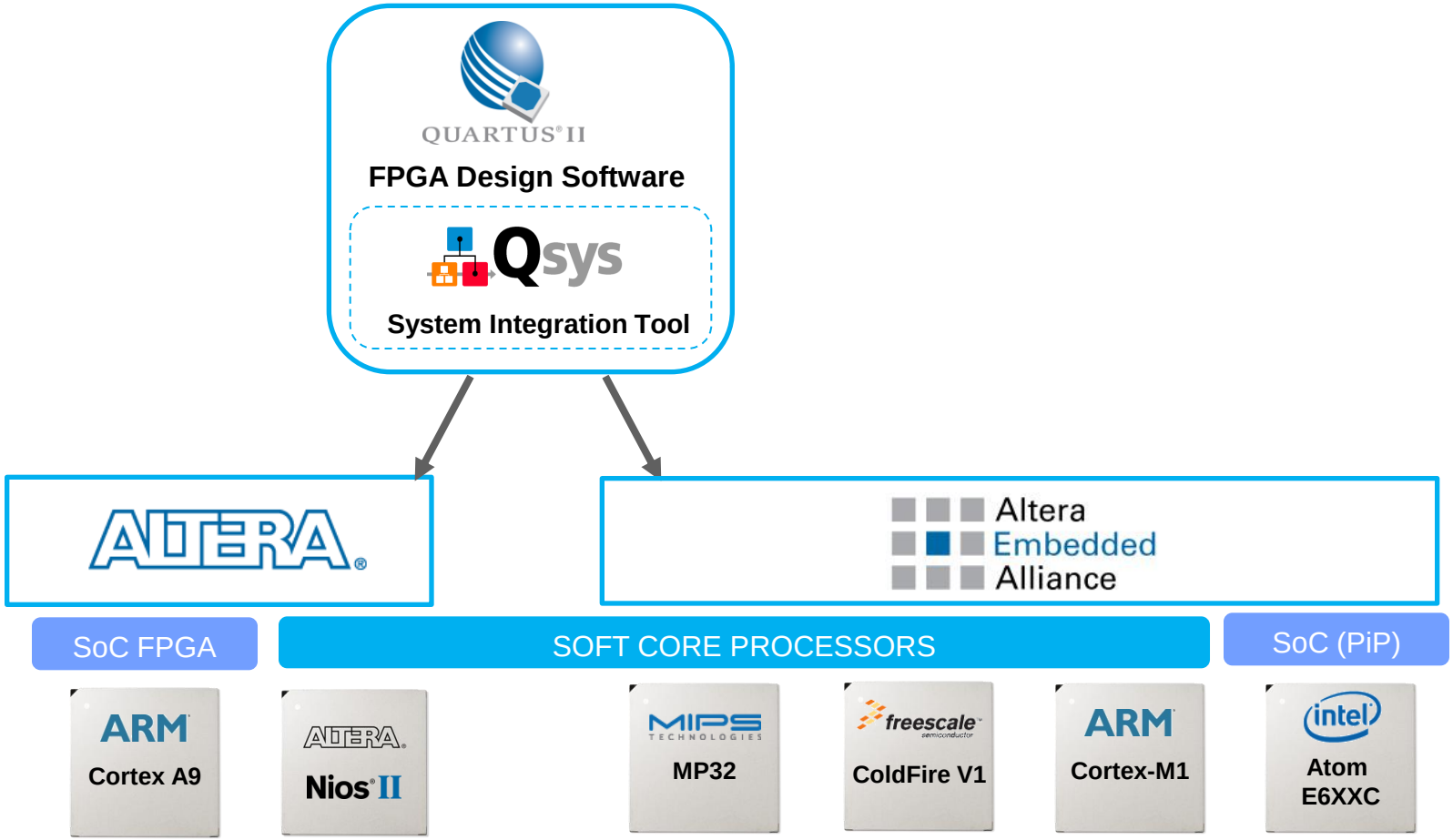
TSMC CoWoS
(Chip-on-Wafer bonding)

Embedded Computing on FPGAs



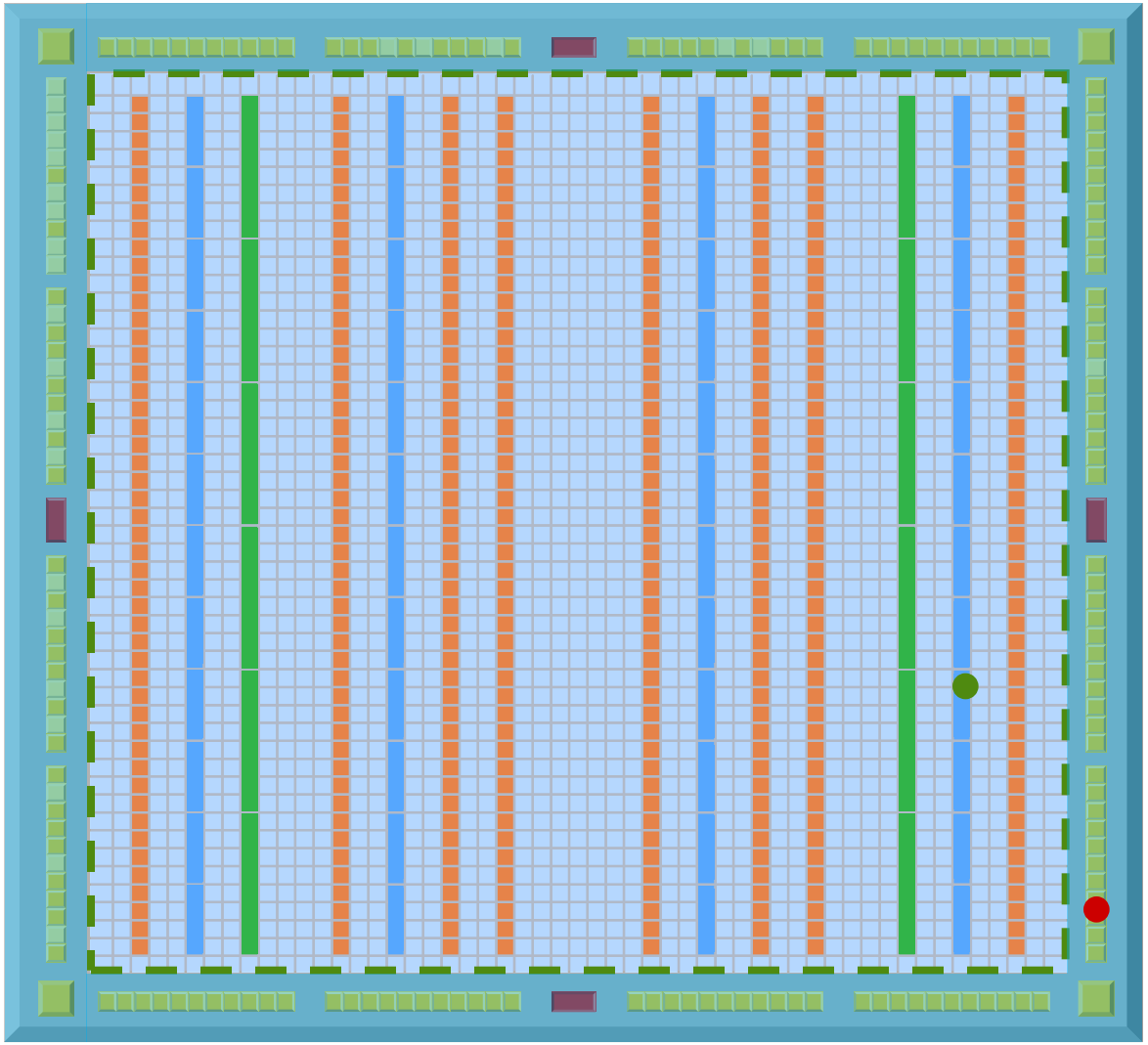
Soon, a majority of FPGA designs will feature an embedded processor

Altera's Embedded Initiative



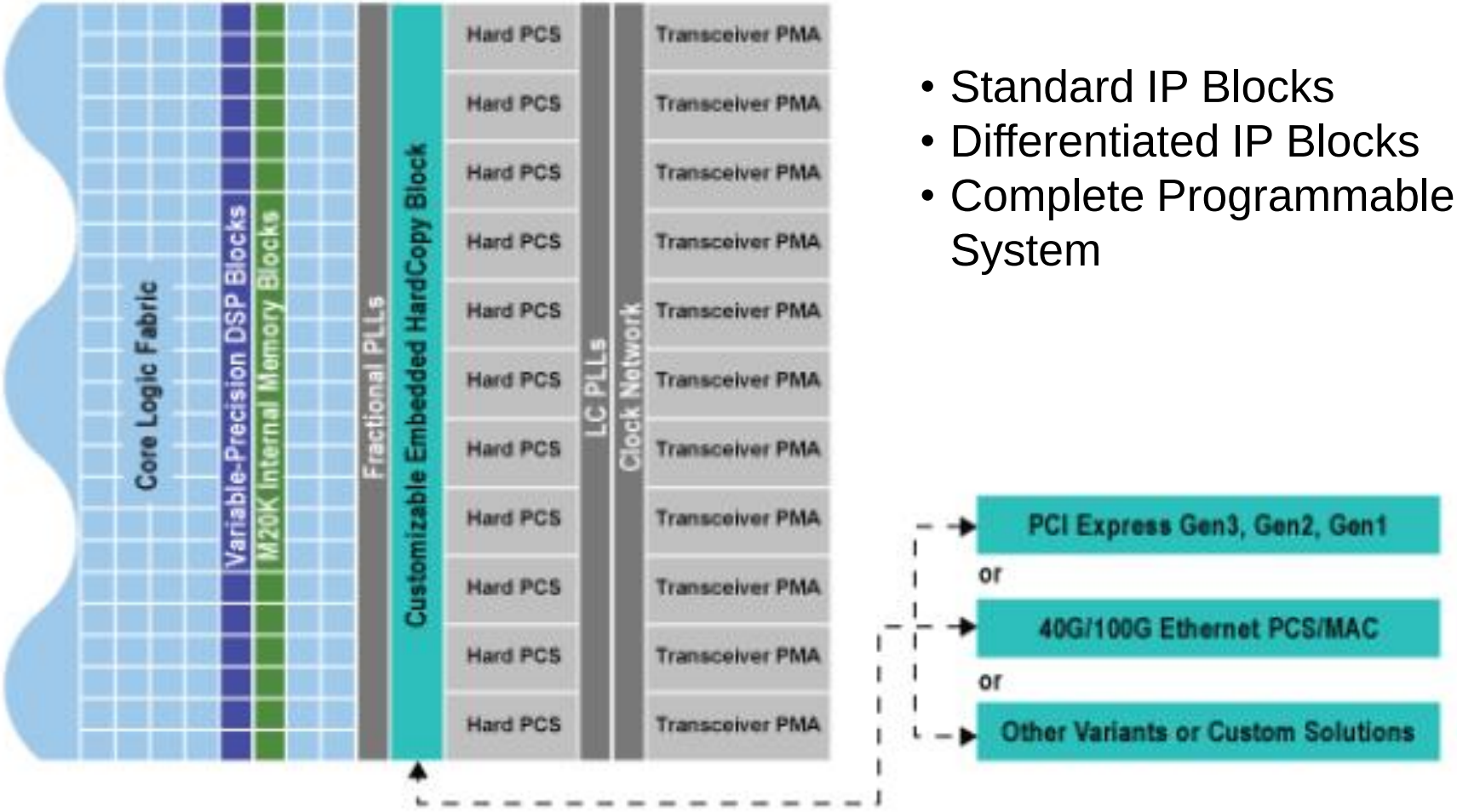
Single FPGA Design Flow for a Broad Array of Industry-Leading Embedded Processors

Modern FPGA: Massively Parallel



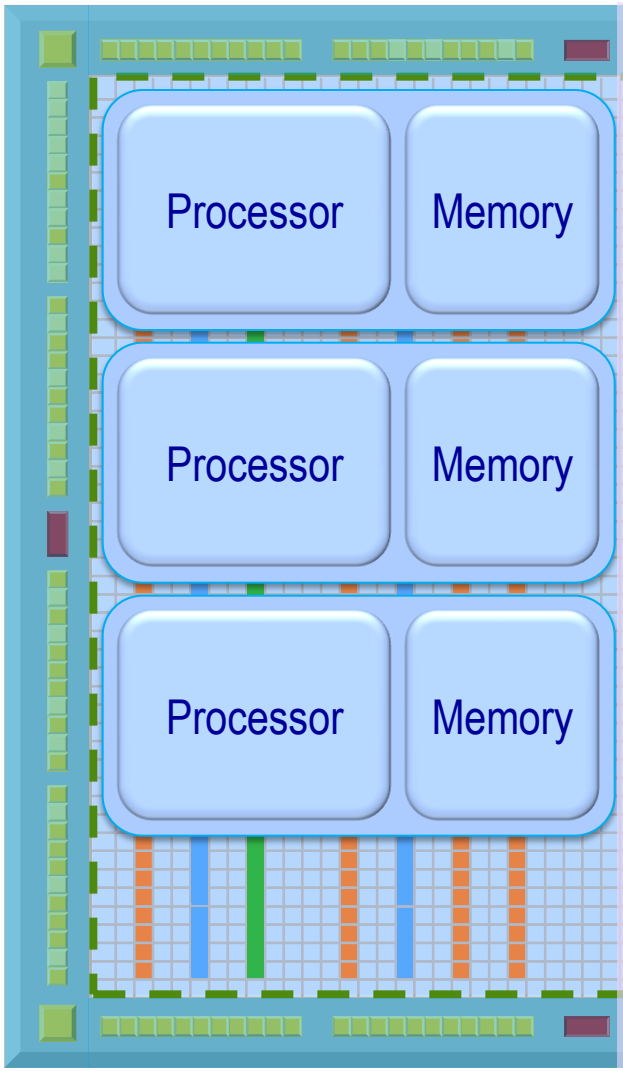
- Million + Logic Elements
- Multi-billion Transistors
- Thousands of 20-Kbit Memory Blocks
- Thousands of MAC DSP Blocks
- Many External Memory, LVDS, And Transceivers IO Blocks

Hardened System Protocols and IP



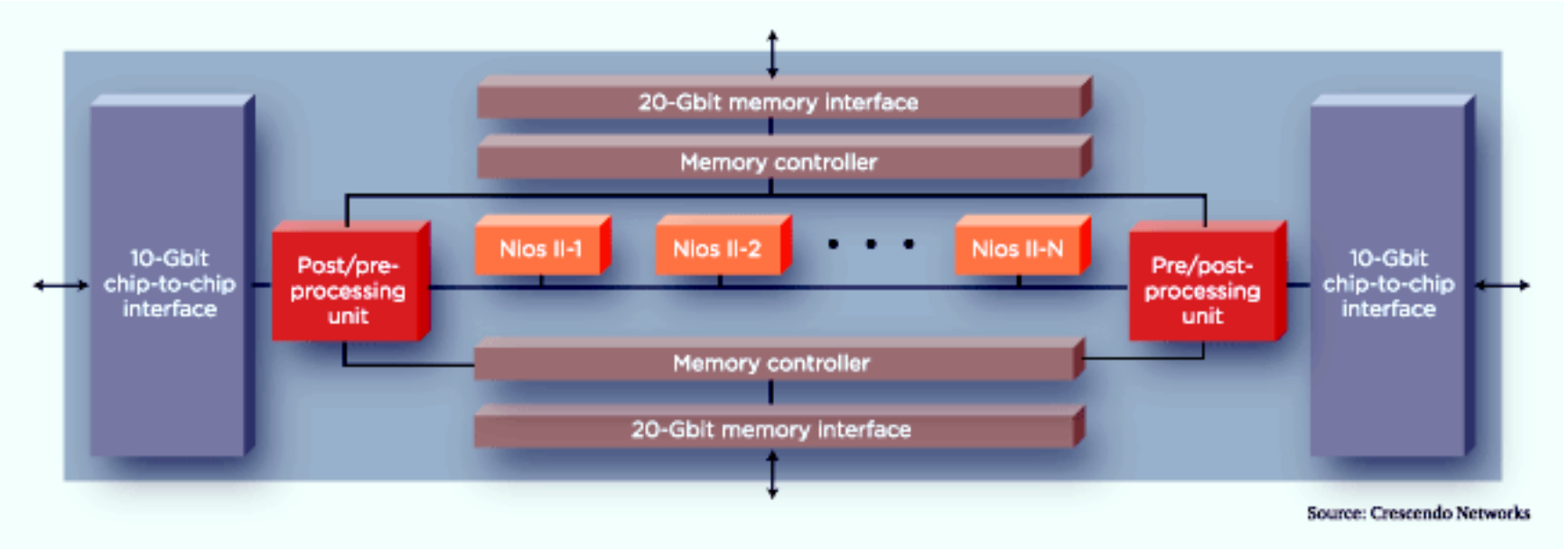
- Standard IP Blocks
- Differentiated IP Blocks
- Complete Programmable System

FPGA : Fully Configurable Multicore



- Can implement many coarse-grained processors
- Exploiting the fine grained parallelism of the FPGA for acceleration
- Possibly heterogeneous
  - Optimized for different tasks
- Customizable to suit the needs of a particular application

An Example: Crescendo Networks



- A Programmable Eight-Processor Architecture
  - Using Embedded Soft Processor Nios II
  - On a Single FPGA
  - For Layer 4/7 Applications

Following the Silicon Trajectory

- Convergence in hardware solutions
  - Heterogeneity – on-chip, interposer, 3D and SIP
  - Embedded processing with dedicated programmable hardware
  - Programmable substrate for full systems-on-chip

- Enabling technologies
  - HW/SW Co-design
  - Ecosystem, Infrastructure and IP
  - Differentiation through programmability

Hardware-Software Co-Design for SoC FPGAs

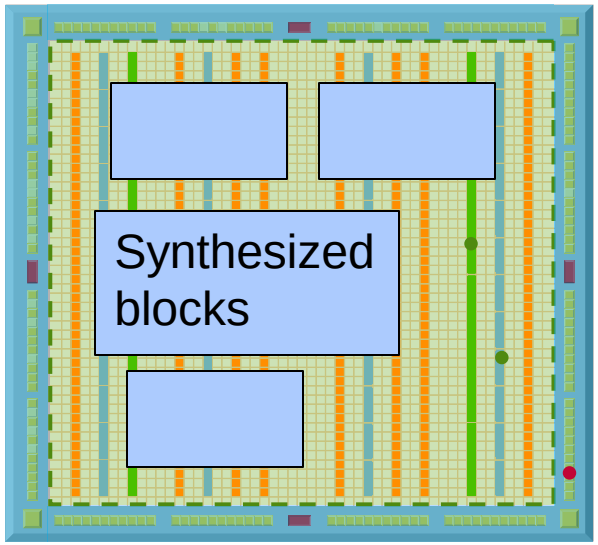
SoC FPGA Programming Models

- Traditional FPGA's programming model is RTL
  - State machines, datapaths, arbitration, buffering, etc.
- Also need a programming model that represents an FPGA as:
  - Processor with hardware accelerators
  - Configurable multi-core device

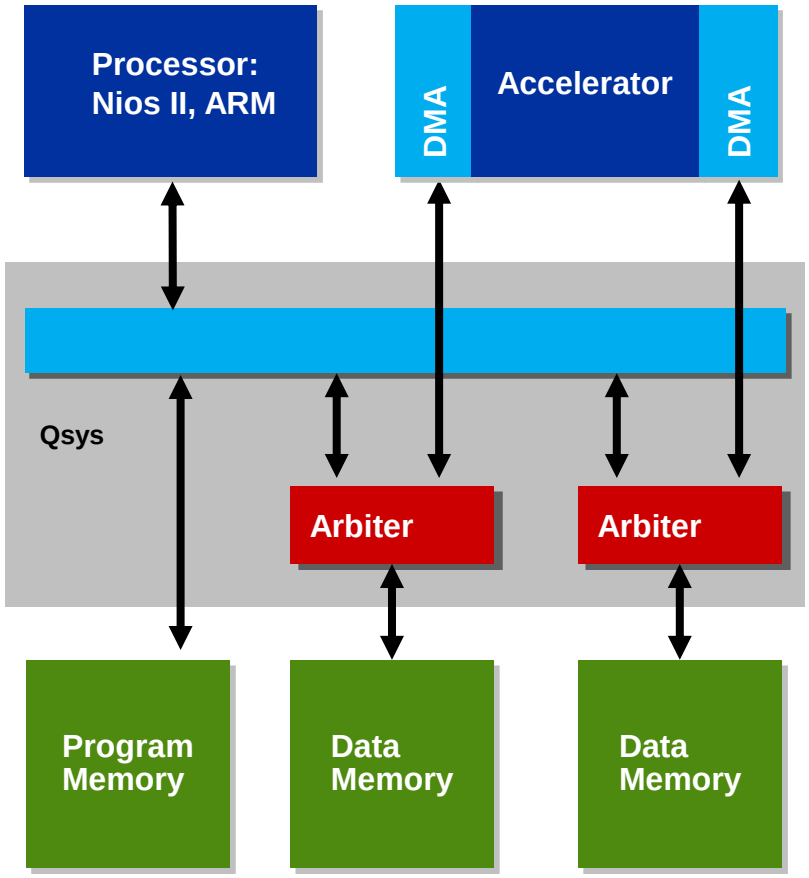
C/C++ Compilers for FPGAs

- Several commercial offerings
  - C/C++/System C High-Level-Synthesis to RTL
    - C-to-Silicon Compiler (Cadence)
    - Impulse C (Impulse Accelerated Technologies)
    - Catapult C (Mentor)
    - SynphonyC (Synopsys)
    - AutoESL (now Xilinx)
    - Cynthesizer (Forte),
    - NEC

```
d_transform (int *t, int *p)
{
   …setup code…
   for (i = 0; i < Buf_Size; i++)
   {
        …loop overhead…
        *t = transform (*p);
        t++;
        p++;
   }
   …exit code…
}
```



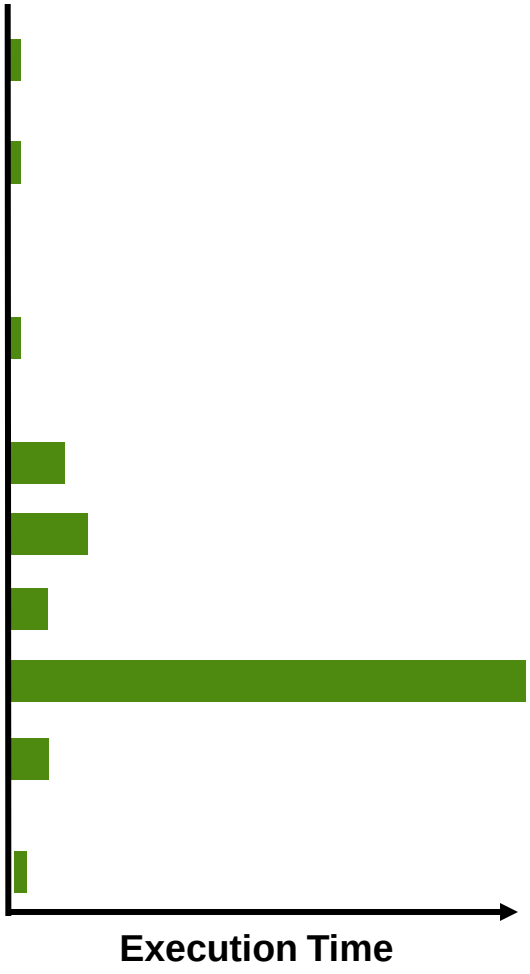
Integration of Embedded Processors and Hardware Accelerators in FPGAs: C2H



Identify the Software Bottleneck

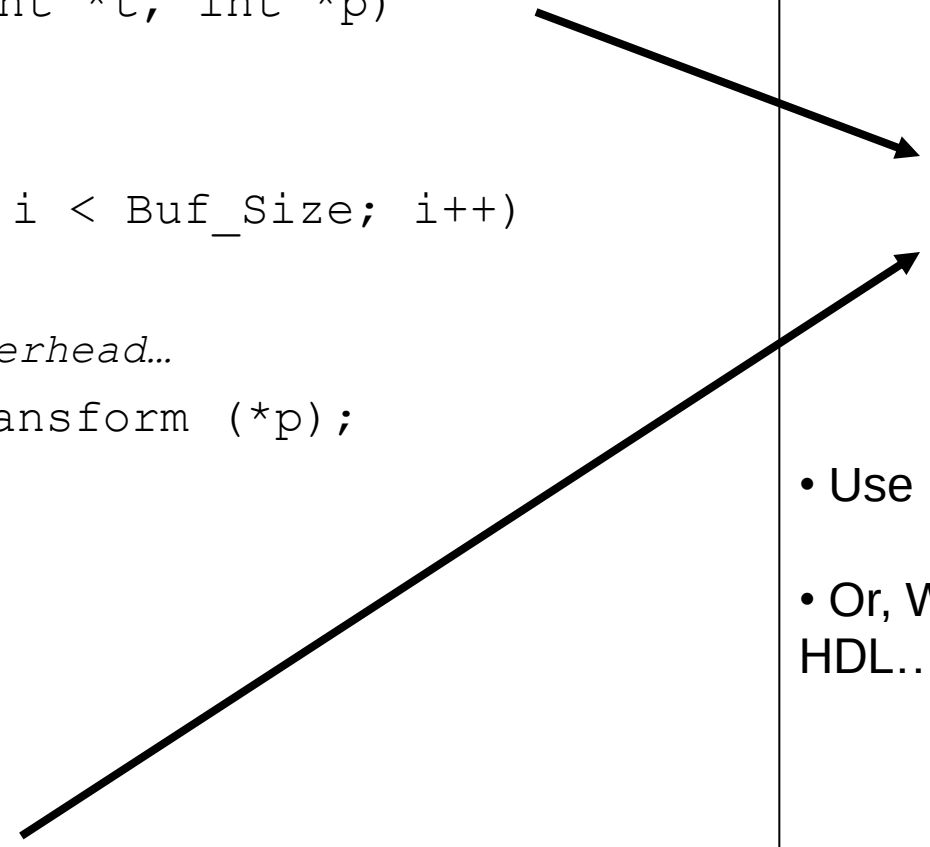
```
main ()
{ …variable declarations…
  init();

  while (!error && got_data())
  {
    do_user_interface();
    gather_statistics();
    if (got_new_data())
      d_transform(in_buf, out_buf);
    check_for_errors();
  }
  cleanup();
}
```



Compile C Function Into Hardware

```
d_transform (int *t, int *p)
{
   …setup code…
   for (i = 0; i < Buf_Size; i++)
   {
        …loop overhead…
        *t = transform (*p);
        t++;
        p++;
   }
   …exit code…
}
```



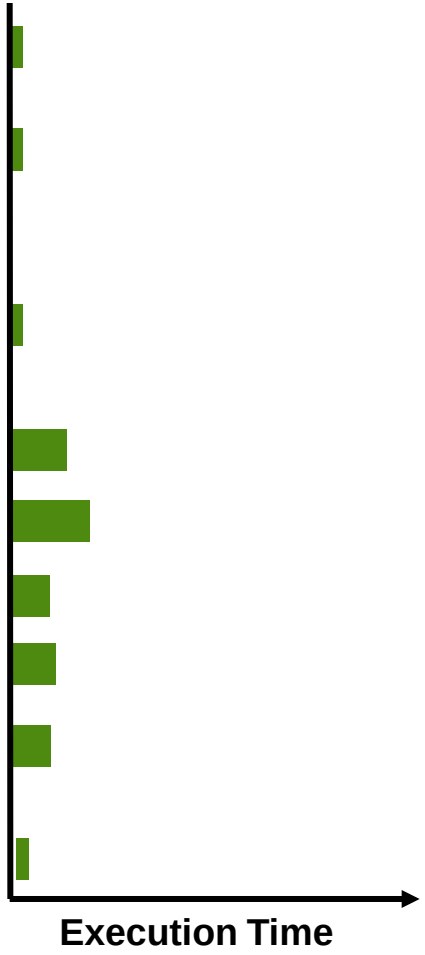
Hardware Accelerator

- Use HLS Compiler
- Or, Write the Function in HDL…

The Result

```
main ()
{ …variable declarations…
  init();

  while (!error && got_data())
  {
    do_user_interface();
    gather_statistics();
    if (got_new_data())
      d_transform(in_buf, out_buf);
    d_transform
  }
  cleanup();
}
```

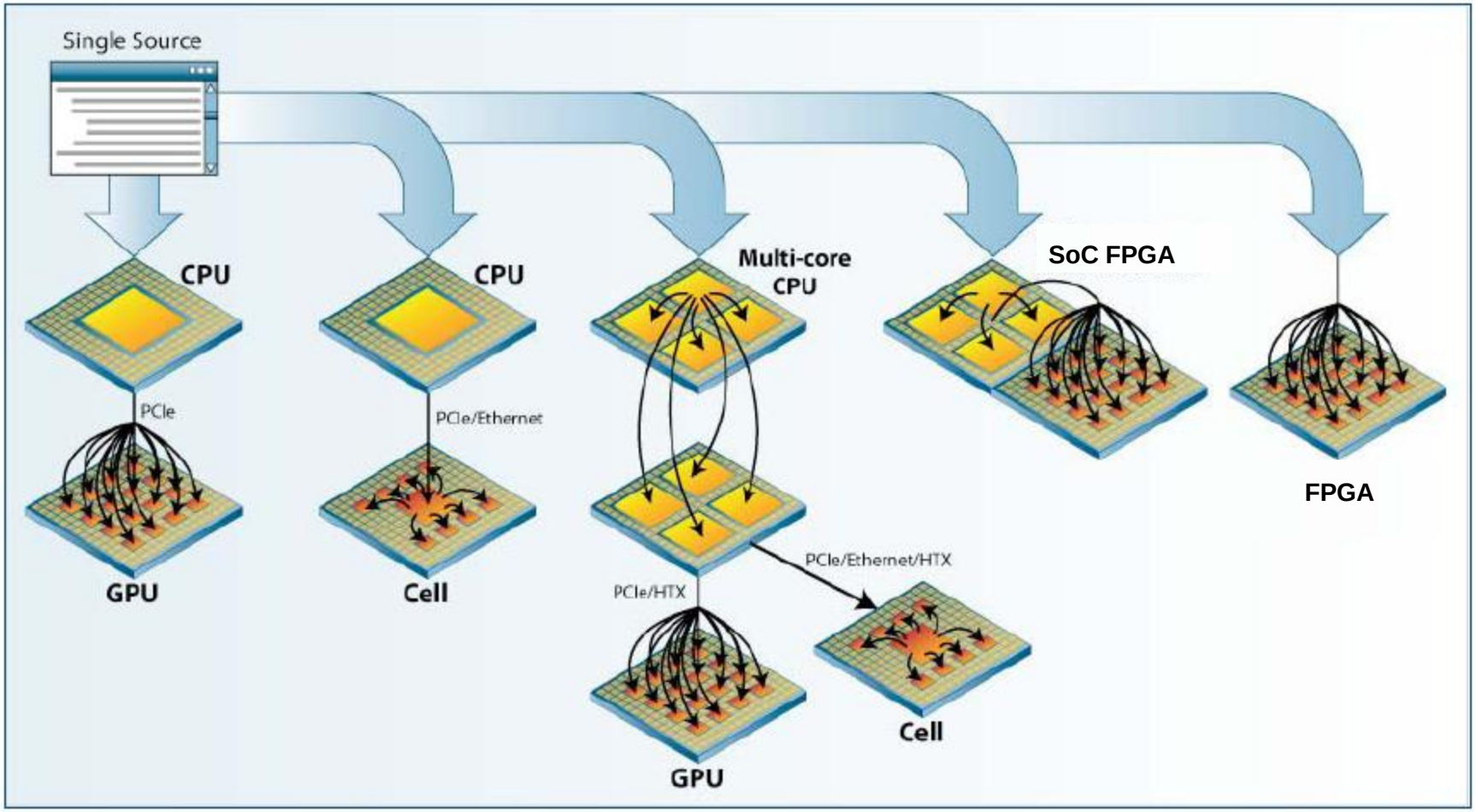


OpenCL Approach to SoC FPGA Programming

What is OpenCL?

- OpenCL is a programming model developed by the Khronos group to support multi-core programming and silicon acceleration
- An industry consortium creating open API standards
- Commitment to royalty-free standards

OpenCL Portability



Source: RapidMind

OpenCL Structure

- Natural separation between the code that runs on accelerators and the code that manages those accelerators
  - The “host” code is pure software that can be executed on any sort of conventional microprocessor
    - Soft processor, Embedded hard processor, external x86 processor
  - The kernel code is ‘C’ with a minimal set of extensions that allows for the specification of parallelism and memory hierarchy

OpenCL Host Program

- Pure software written in standard ‘C’
- Communicates with the Accelerator Device via a set of library routines which abstract the communication between the host processor and the kernels

Copy data from Host to FPGA

Ask the FPGA to run a particular kernel

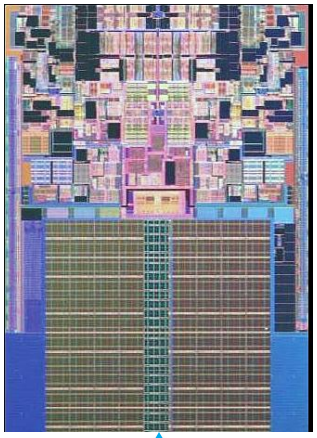
Copy data from FPGA to Host

```
main()
{
   read_data_from_file( … );
   maninpulate_data( … );

   clEnqueueWriteBuffer( … );
   clEnqueueTask(…, my_kernel, …);
   clEnqueueReadBuffer( … );

   display_result_to_user( … );
}
```

OpenCL SoC FPGA Target



DDR*

PCIe

Embedded Soft or Hard Processor

External Memory Controller & PHY

Switch Fabric

Interface IP

Host Program

Kernels

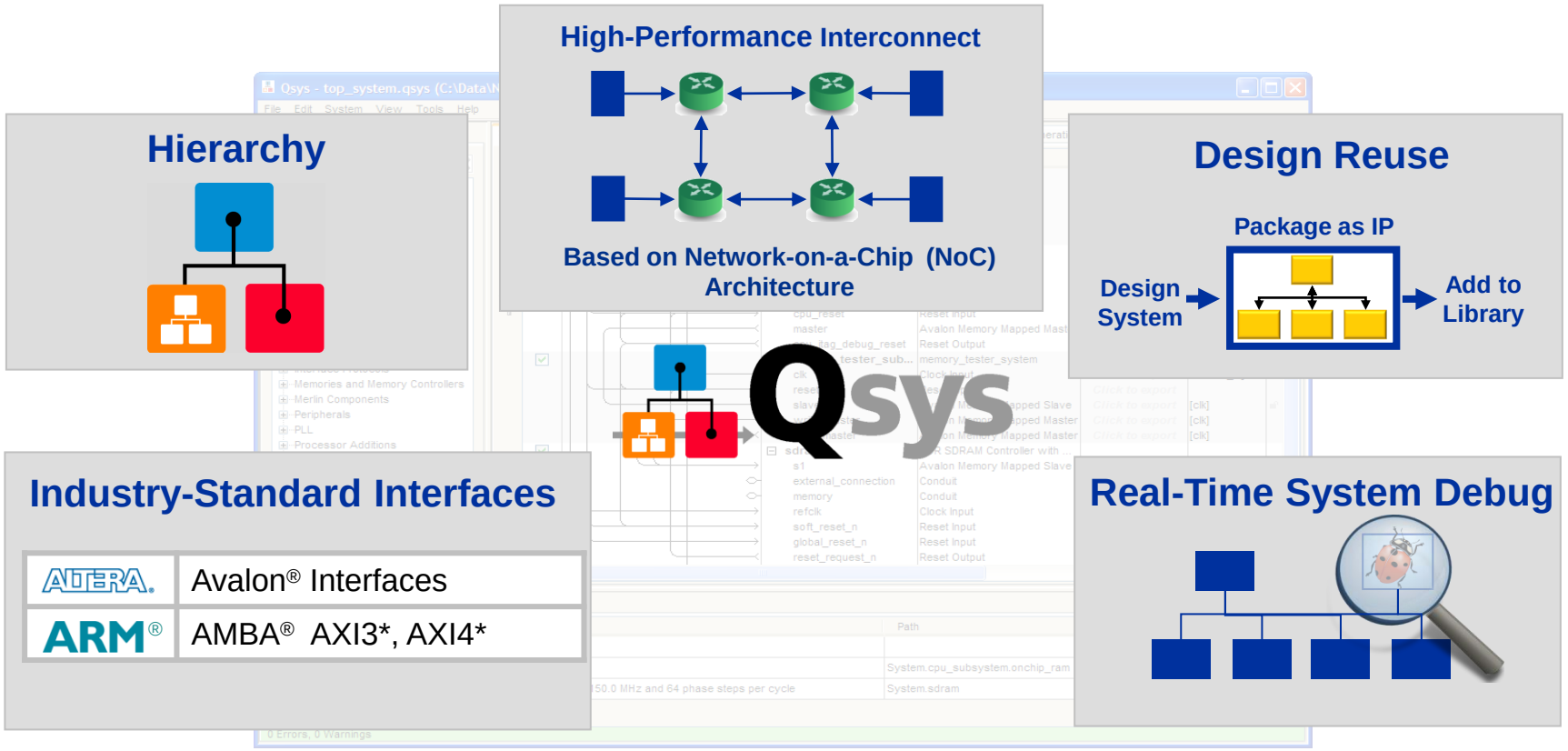
Processor

Memory

Application Specific External Protocols

System Integration

Qsys System Integration Platform

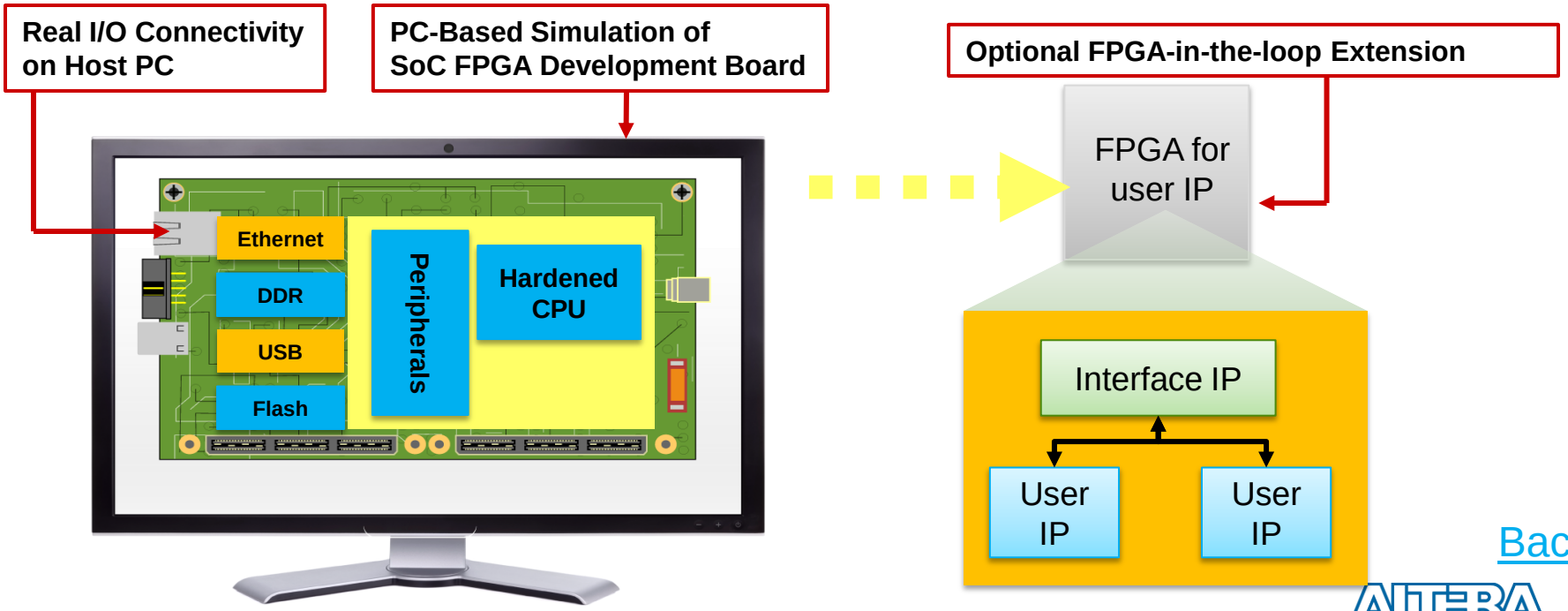


- Qsys is Altera's design environment for
  - Deployment of IP
  - Deployment of reference designs and example designs
  - Development platform for Altera custom solutions
  - Design platform for customers to quickly create system designs

Back

SoC Virtual Target

- Immediate device-specific software development
  - Binary- and register-compatible
- Virtual prototyping technology
- Linux and VxWorks enabled, compatible with ARM tools



Back

Summary

- The Era of Silicon Convergence
  - SoC FPGAs and Programmable Substrate
- The Era of New Embedded Software
  - HW/SW co-design and parallel programming
- The Era of Innovation
  - Exploiting heterogeneous, customizable devices
  - Massively parallel embedded programmable solutions

Thank You