The Reality of System Design Today: Do Theory and Practice Meet?

Grant Martin

Fellow, Cadence Berkeley Labs

ACSD, Portugal, 18 June 2003: 0900-1000

Point of view

ACSD
Third international conference on application of concurrency to system design

- The history of system-level design as a viable commercial concern for the EDA tools industry is littered with false starts:
  - ESDA: Electronic System Design Automation
  - Behavioural synthesis
  - ESL: Electronic System-Level design
- In 2003, EDA is continuing to shrink its focus to primarily physically-related SoC design problems at 130-90-sub-90 nm processes.
- Yet many embedded systems today present profound problems of specification, design and verification:
  - Wireless and wired communications terminals and infrastructure
  - Multimedia consumer devices
  - Large complex systems

Point of view, continued

ACS
Third international conference on application of concurrency to system design

- Is the problem one of:
  - The system design community is too small and diverse, thus making a commercial marketplace for tools permanently unviable?
  - Have we espoused incorrect theories about how these systems should be designed and verified?
  - Have we been premature in trying to ‘industrialise’ system-level design?
  - Have we been looking in the wrong place for a large enough community of designers with compelling problems which can be solved on a commercial basis?
  - Or a combination of all of the above…….?
- We will examine a number of issues in this talk

Outline

- SoC and System Level Design
- Key Requirements
- An existential view of HW-SW Codesign
- Algorithmic Design and Implementation
- Modelling and Design of SoCs
- System Virtual Prototypes of SoCs for ESW
- What about Behavioural Synthesis?
- New SoC Architectures
- Conclusion

SoC

- “System” is more important than “Chip”
- Today’s chipset = tomorrow’s chip or SiP
- The system must be designed as an entity with tradeoffs across boundaries: HW-SW, analogue-digital, chip-package-board

System-Level Design:

- Always tomorrow’s methodology
- EDA’s focus is shrinking to IC Physical design
- SoC may be the best place to see system design applied

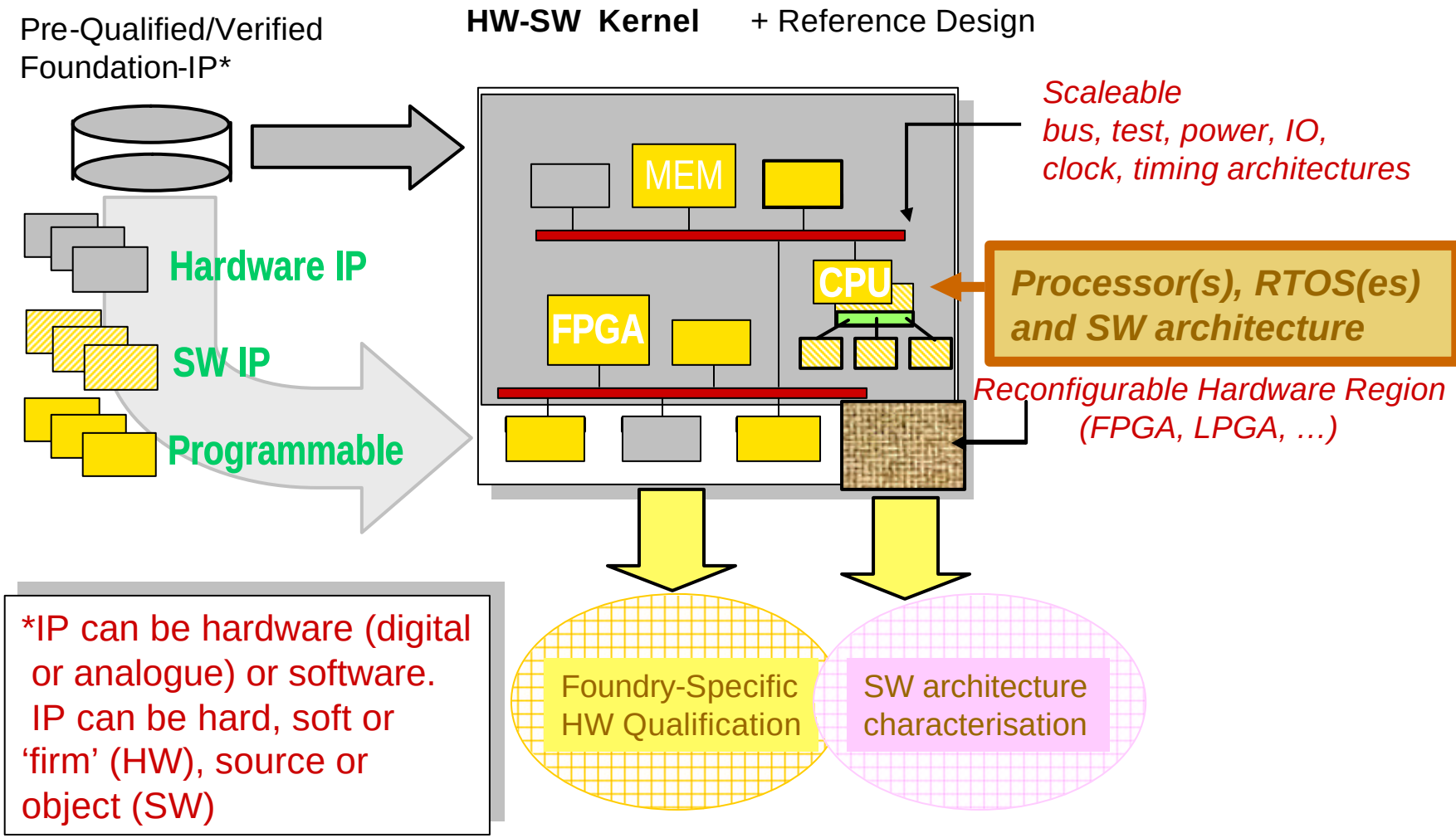
Concurrency:

- Most interesting embedded systems are fairly concurrent and becoming more so:
  - Multiple threads of control
  - Multiple dataflow processing streams
  - Multiple RISC + DSP (1+1 $\Rightarrow$ n+m)
  - But designers are afraid of concurrency at the system level

Key SoC/System Level Design Requirements

- Algorithmic design and implementation
- Modelling of SoCs and SoC platforms at the system level
  - Integration of SoC and configuration of designs
  - Build a platform model and verify HW-SW interfaces
  - Performance analysis and Design Space Exploration
- “System Virtual Prototypes” for embedded SW:
  - Hardware-dependent SW (HdS)
  - ESW application development

SoC Platforms



*IP can be hardware (digital or analogue) or software. IP can be hard, soft or 'firm' (HW), source or object (SW)

An existential view of HW-SW Codesign

- Does it exist?....i.e. as a delayed implementation choice
- Tradeoffs of HW vs. SW – not very relevant for most designs
  - Legacy: most tradeoffs are known or dead obvious
  - Processors are changed only very rarely
    - The SW legacy is enormous
  - Specifications easily drive an obvious choice of HW implementation when needed
- More relevant?
  - “SW-SW Co-Design” – mapping functions to multiple programmable or configurable computation and communications resources
  - Obvious need for concurrency-based design methods and tools!

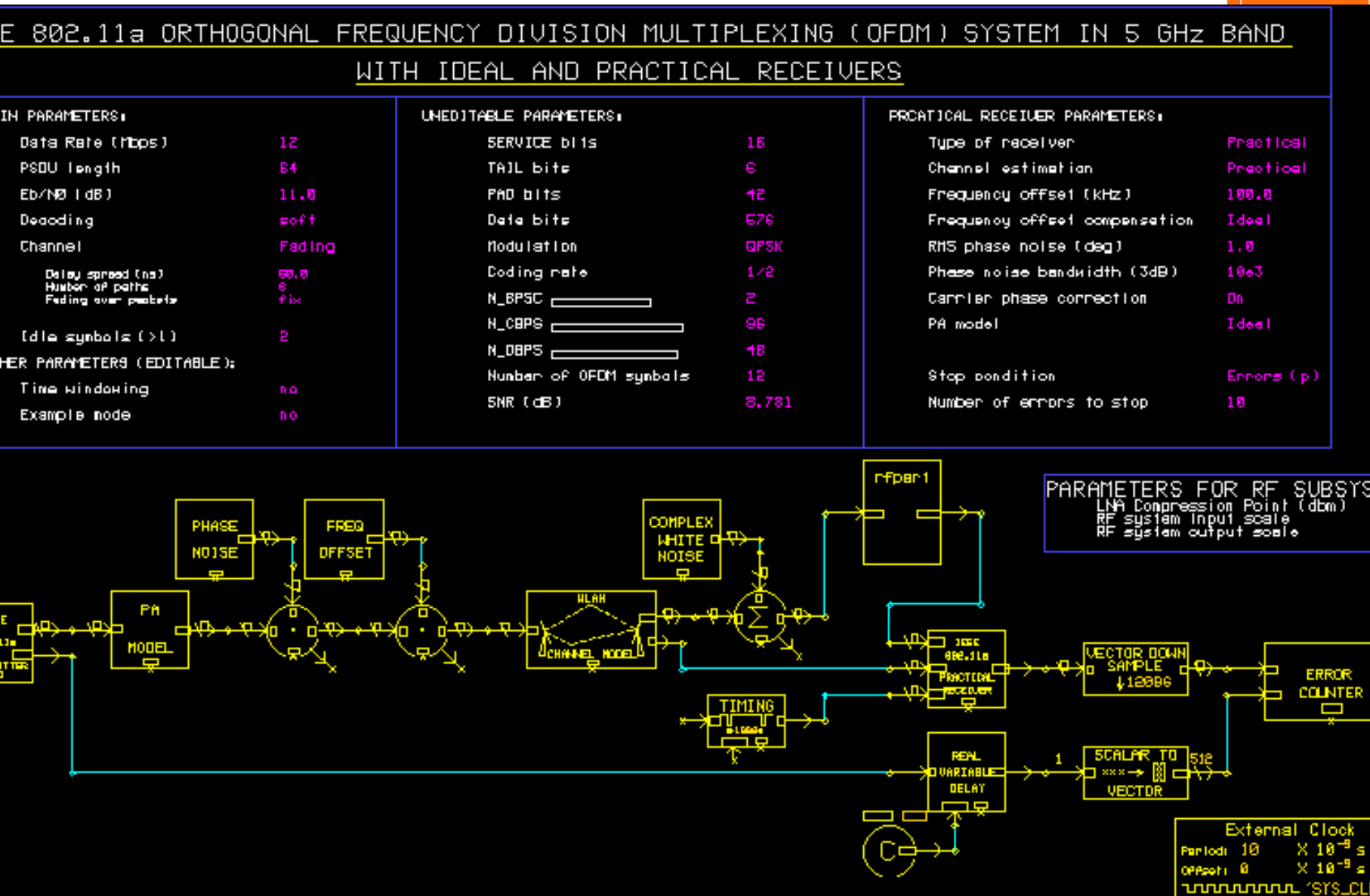
Algorithmic Design and Implementation

- Design and Implementation of complex control and dataflow algorithms in HW, SW or a combination
- Today's best practices use system level design tools
- Well established for many years
  - Dataflow is better handled than mixes of dataflow and control
- Used both for less integrated systems as well as SoC

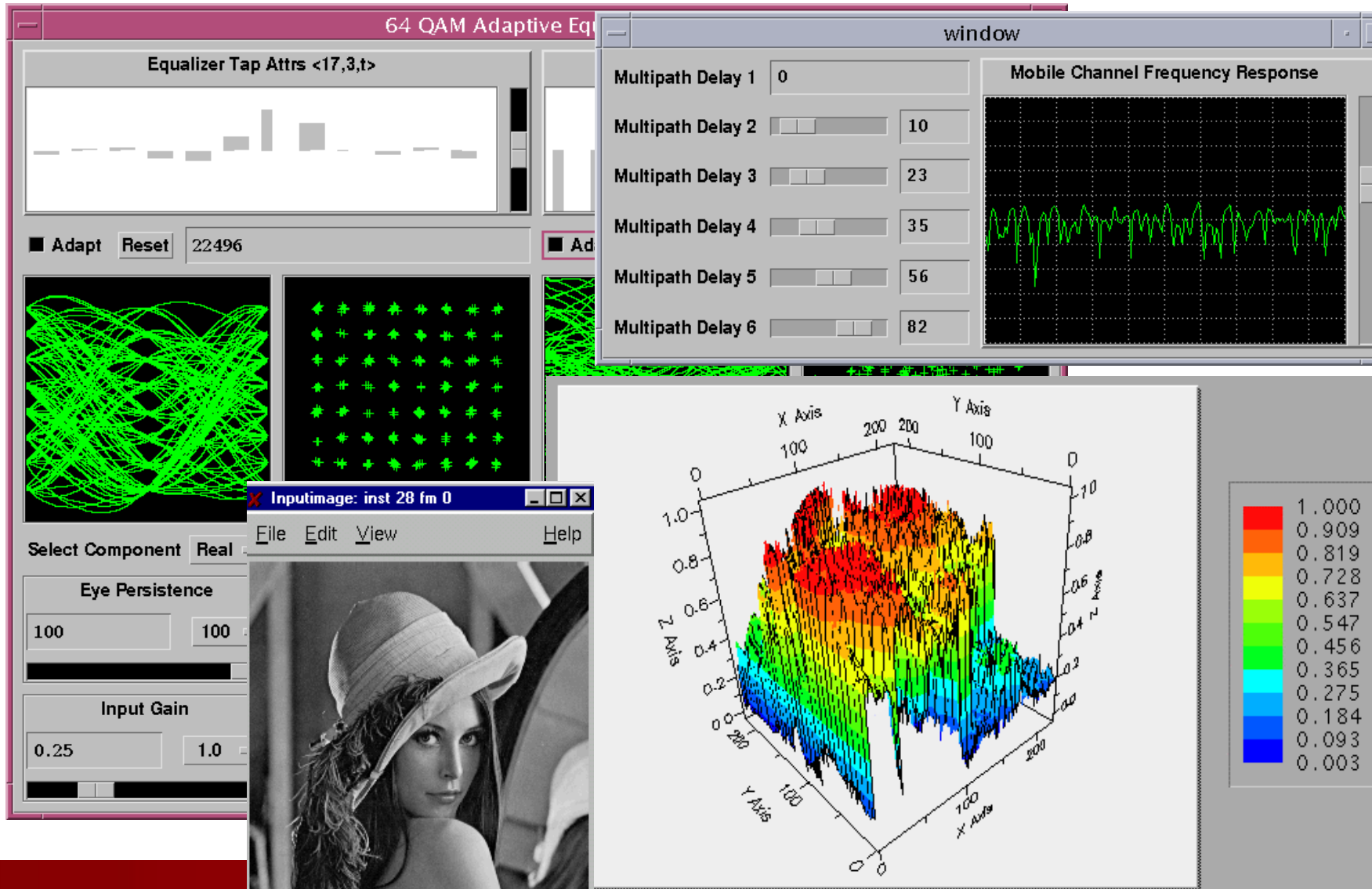
Dataflow algorithms

- Classic and well-established tools exist
  - Mathworks: Matlab, Simulink
  - Cadence: SPW
  - Research: Ptolemy I/II
  - Synopsys: COSSAP/CoCentric System Studio

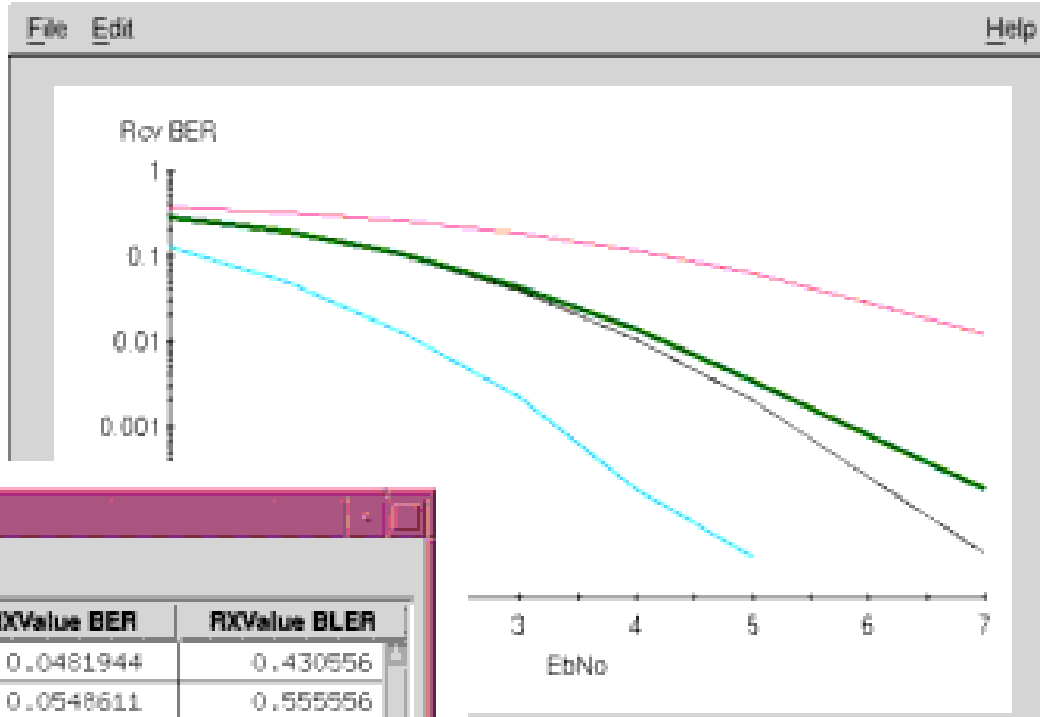
A Dataflow tool example



A Dataflow Tool: Interactive Simulation



Bit Error Rate Analysis



Multisimulation Probe Results

ber_sim/bler.csv

EbNo	Metric	phase	RXValue Blocks	RXValue BER	RXValue BLER
3	Hard	3	72	0.0481944	0.430556
1	Soft	3	72	0.0548611	0.555556
2	Soft	3	72	0.0122222	0.208333
1	Hard	3	72	0.164722	0.944444
3	Soft	3	72	0.000277778	0.0138889
4	Hard	3	72	0.00975	0.194444

Metric=Hard, phase=2
Metric=Hard, phase=3
Metric=Soft, phase=2
Metric=Soft, phase=3

lay = # of Pts =
Freq. = Point =
lue = Y Value =

ot Type Y = F(X)

X: RXValue BER

Y: RXValue BLER

Plot

New Trace for Each:

Plot Points Where:

Control algorithm capture and analysis

ACS

Third international conference on application of concurrency to system design

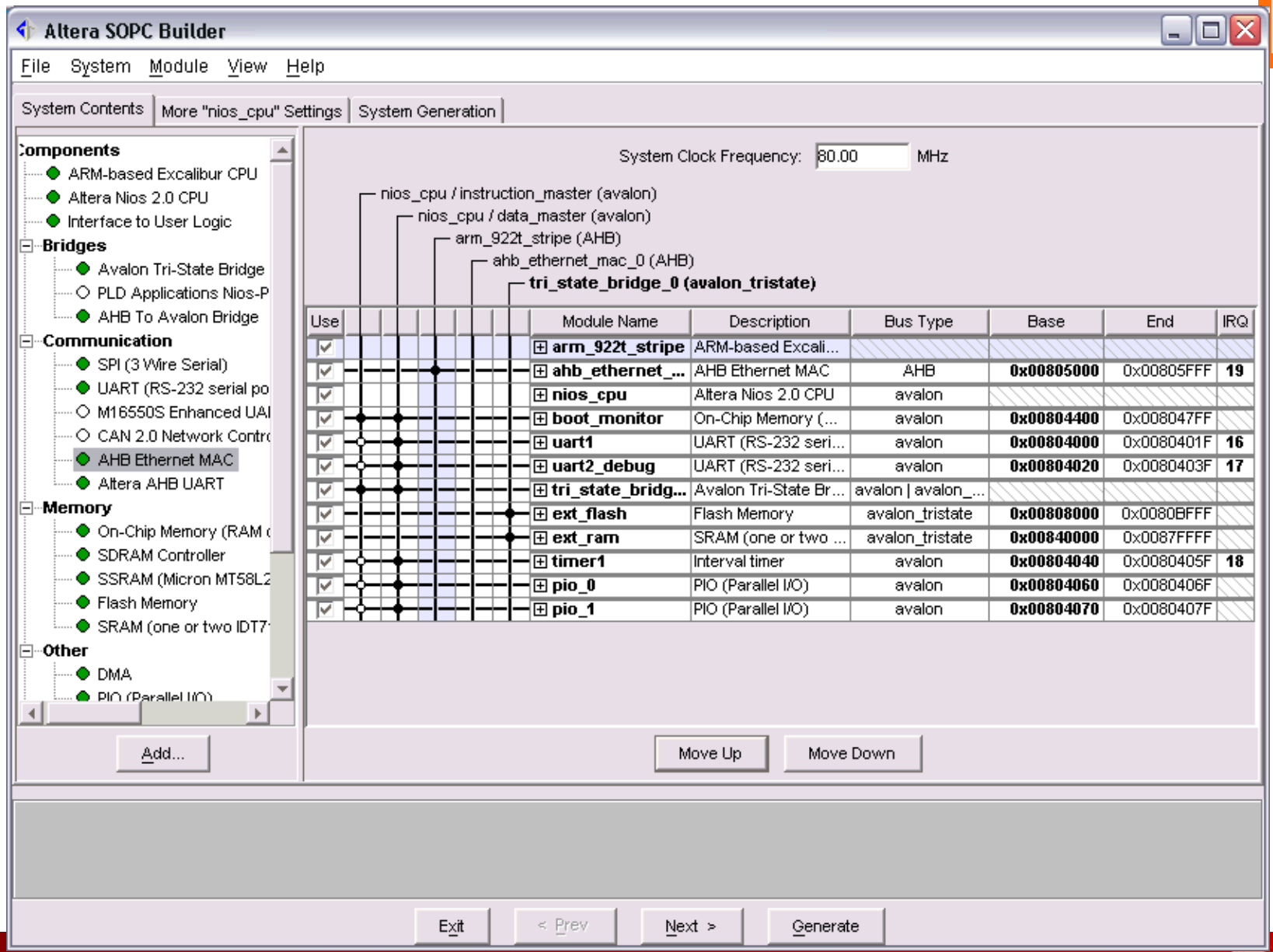
- State Machine capture and simulation
  - Links to HW and SW generation
- A possible place for adapting SW development flows into the system space
  - SDL, UML
    - Statecharts, state diagrams, message sequence charts
  - Esterel
    - Synchronous Reactive systems

Modelling and Design of SoCs and their architectures

ACSD
Third international conference on application of concurrency to system design

- Capturing SoC architecture and providing SoC configurators
- SoC model integration and developing verification models
  - Especially HW/SW integration models
  - “Golden models” for implementation verification
- Design Space Exploration

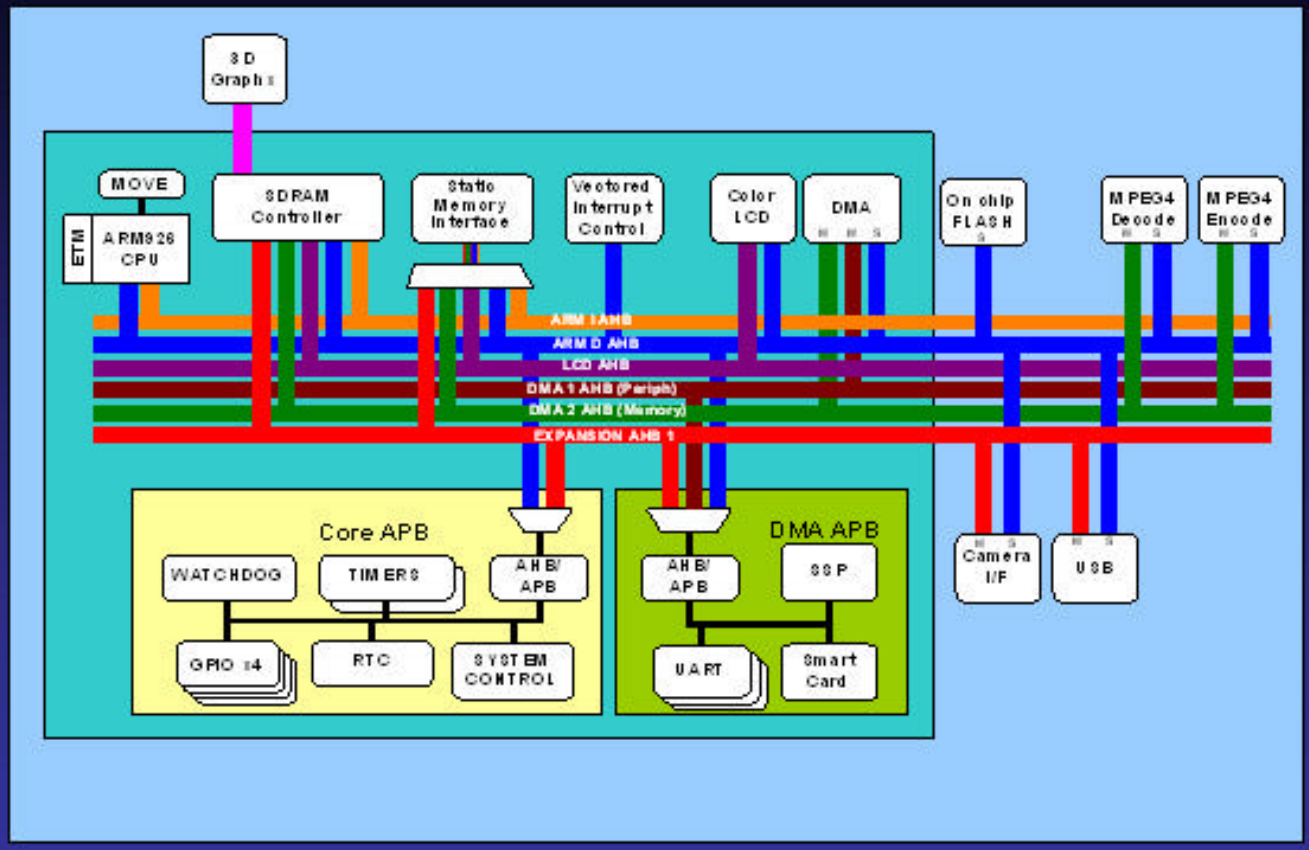
SoC Configurators – Example: Altera SOPC Builder



Source: Altera web site www.altera.com

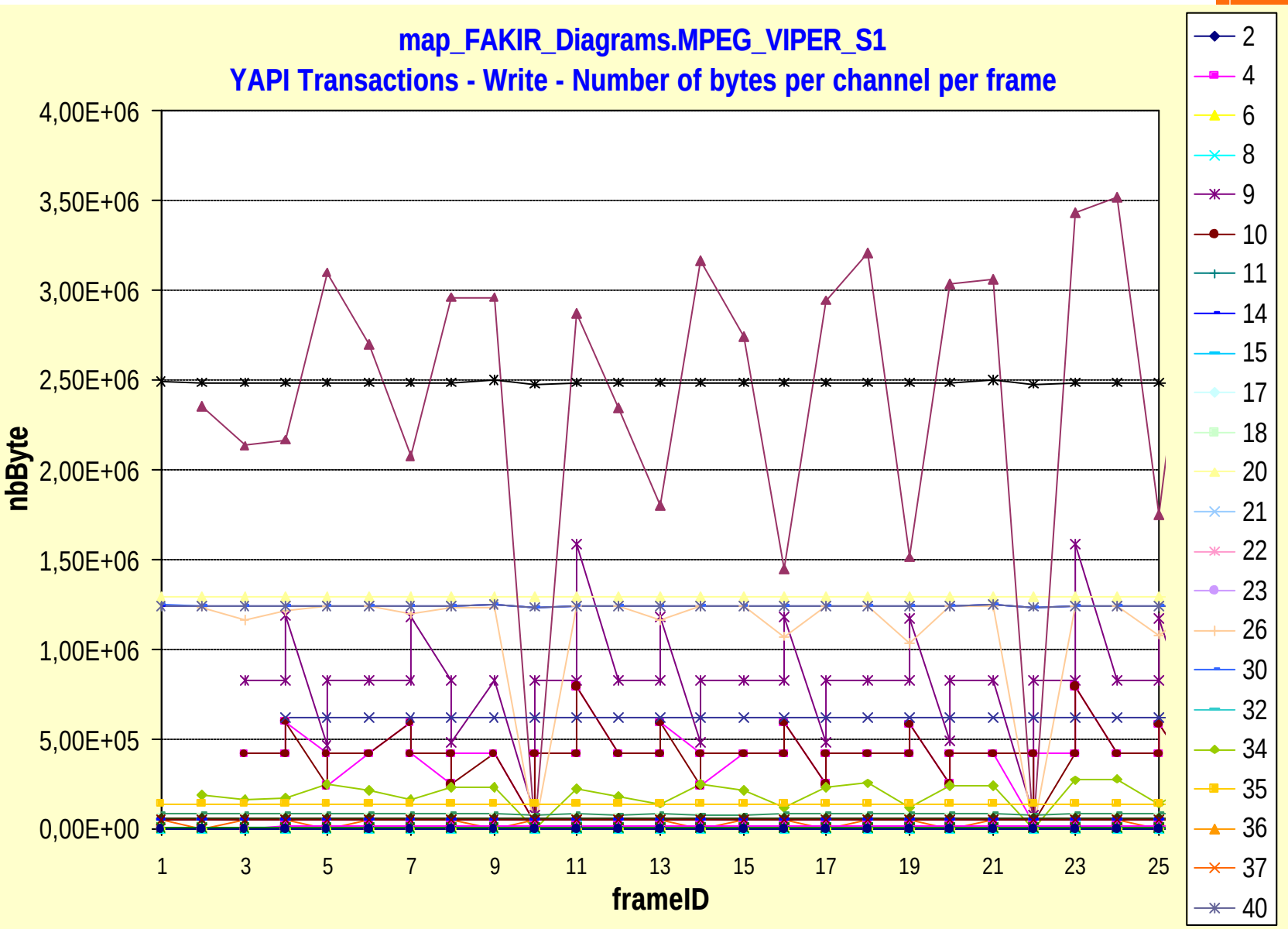
SoC Model Integration: SystemC

Platform design problem



ARM THE ARCHITECTURE FOR THE DIGITAL WORLD™

Design Space Exploration - example



System Virtual Prototypes of SoCs for ESW

- Hardware-dependent SW developers
  - Need cycle and phase accuracy, bit-level precision
- Applications SW developers
  - Need functional correctness and fast execution

A Scenario for Embedded Systems

Applications SW Designers

Platform Abstraction:

Link between the communities

m Platform AP

Systems Architects

HdS SW Designers

HdS API

HdS Spec

HW Designers

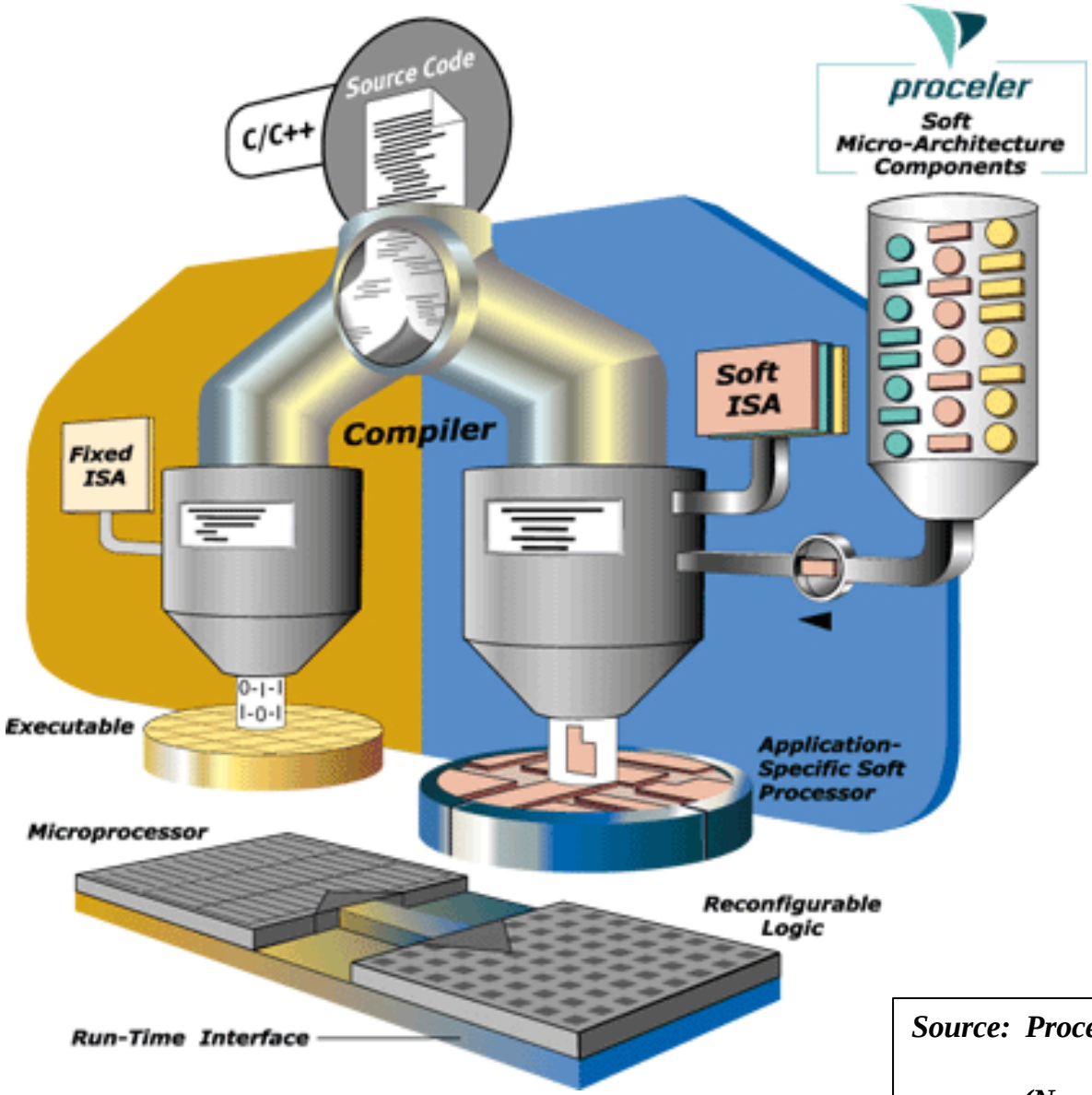
System Level Virtual Prototypes

- Develop HdS Models of your HW-SW platform
  - Using, for example, SystemC
  - SystemC 2.0 adds abstract communications modelling
  - SystemC 3.0 will add abstract RTOS modeling
    - Scheduling, communications services
- Use SystemC (2.0-3.0) as a simulation and analysis “Backbone” for analysing the complete HW-SW embedded system
  - An interoperable, Integration, Infrastructure
- Back-annotate the results INTO the SW developer’s development environment via fast-execution, functional System Virtual Prototypes
  - SW modeling tools
    - UML, SDL, …
  - SW Integrated Development Environments

What about Behavioural Synthesis?

- Not been a major success for commercial tools
  - Quality of results uncompetitive with general RTL synthesis
- Used in specialised contexts
  - E.g. dataflow algorithm to implementation
- May resurface via ***SW implementation optimisation*** to processor+HW combination
  - Emerging “co-processor synthesis”

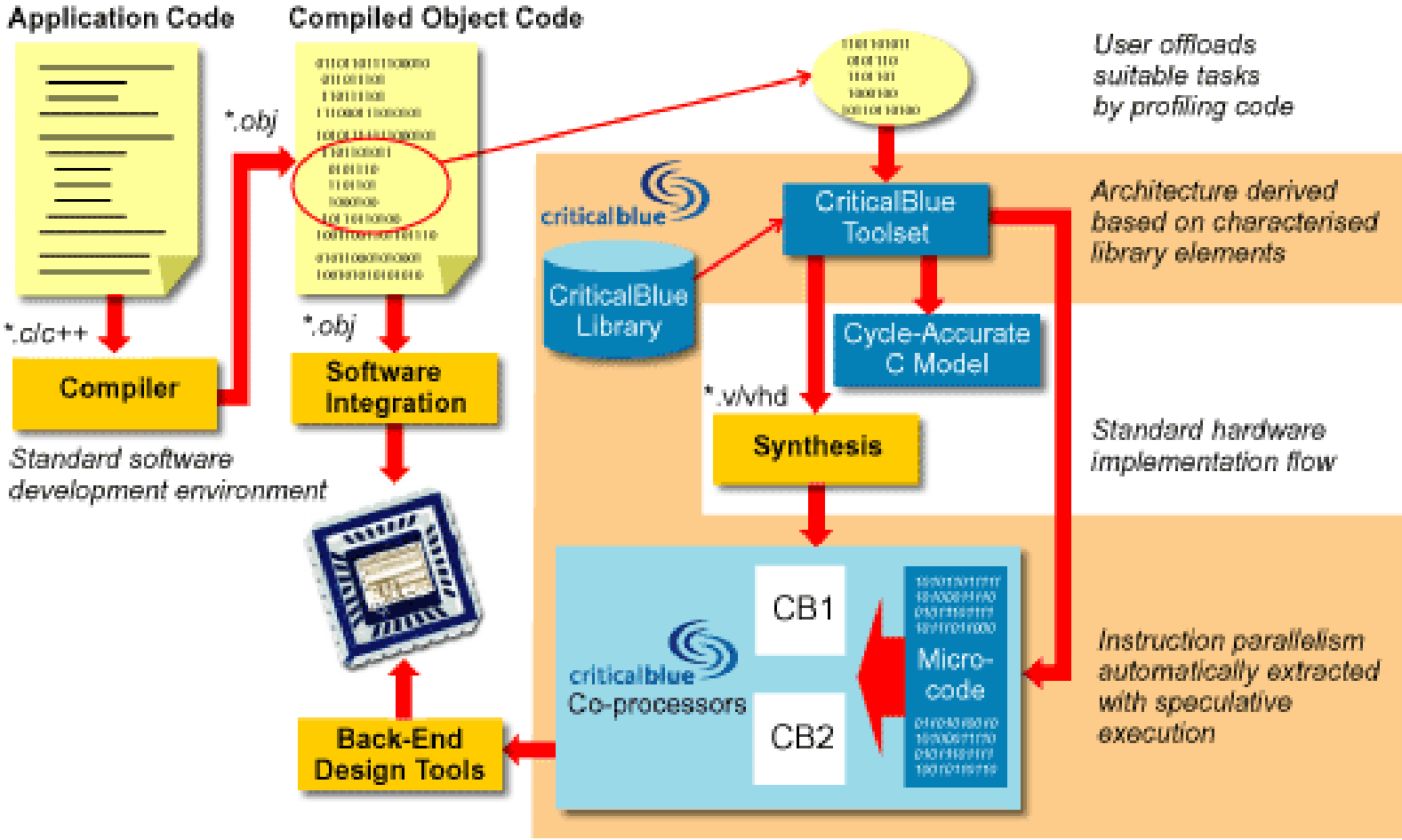
Compile to SW+HW: The Proceler Example



Source: Proceler Web Site *www.proceler.com*

(Now Unfortunately Defunct)

From the defunct to the emergent: Co-processor Synthesis - CriticalBlue



Source: CriticalBlue Web Site *http://www.criticalblue.com/technology3.htm*

New SoC Architectures

- A ‘sea’ of Flexible, configurable computational resources
- Using flexible, configurable, on-chip communications networks
- New architectures require new thinking
- ***This*** may be the real opportunity for system-level design
- But it may come with a real SW focus
- Current interest in compiling SW models to processor+accellerating HW (often using reconfigurable logic)

- Obvious need here for Methods and Tools involving concurrency!
  - Mapping computation to sea of resources
    - Optimising configuration
  - Mapping communications to network
  - Making sure it all works together harmoniously

ne Software Washing Machine

Software Washing Machine [Catthoor, IMEC]

Target independent source to source transformations for data-intensive applications save power!

IN: Dirty C++ software IP's + scenario

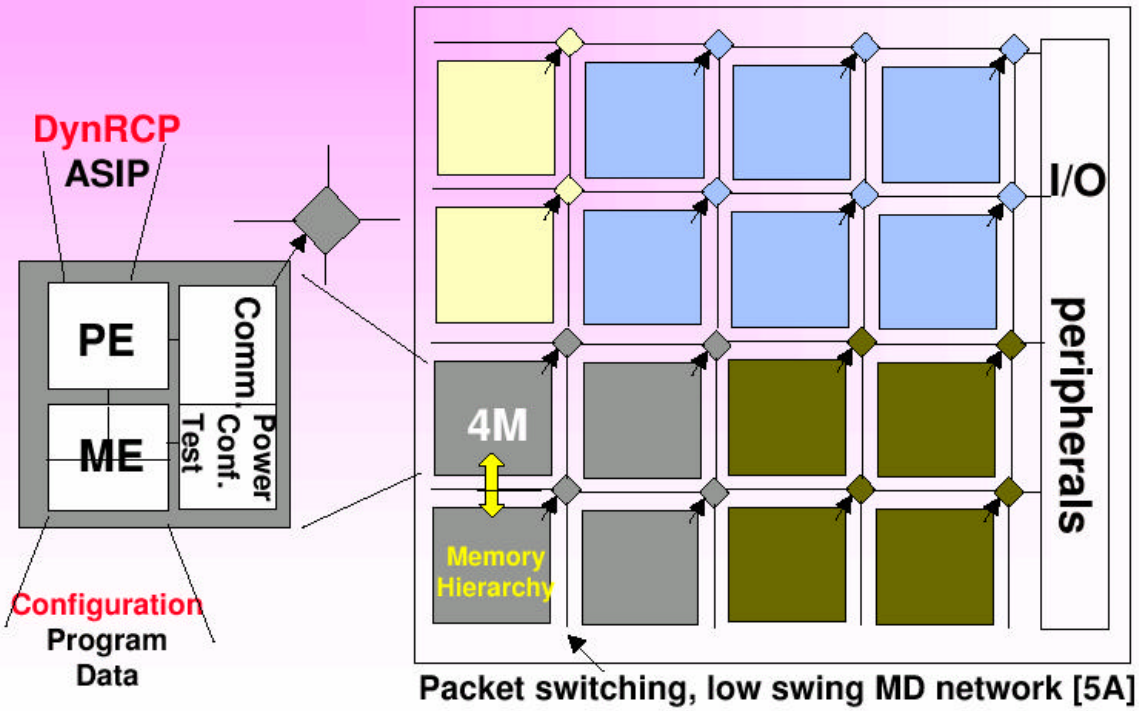
OUT: Cleaned Concurrent C++ tasks OUT (SYSTEMC)

Platform specific compiler

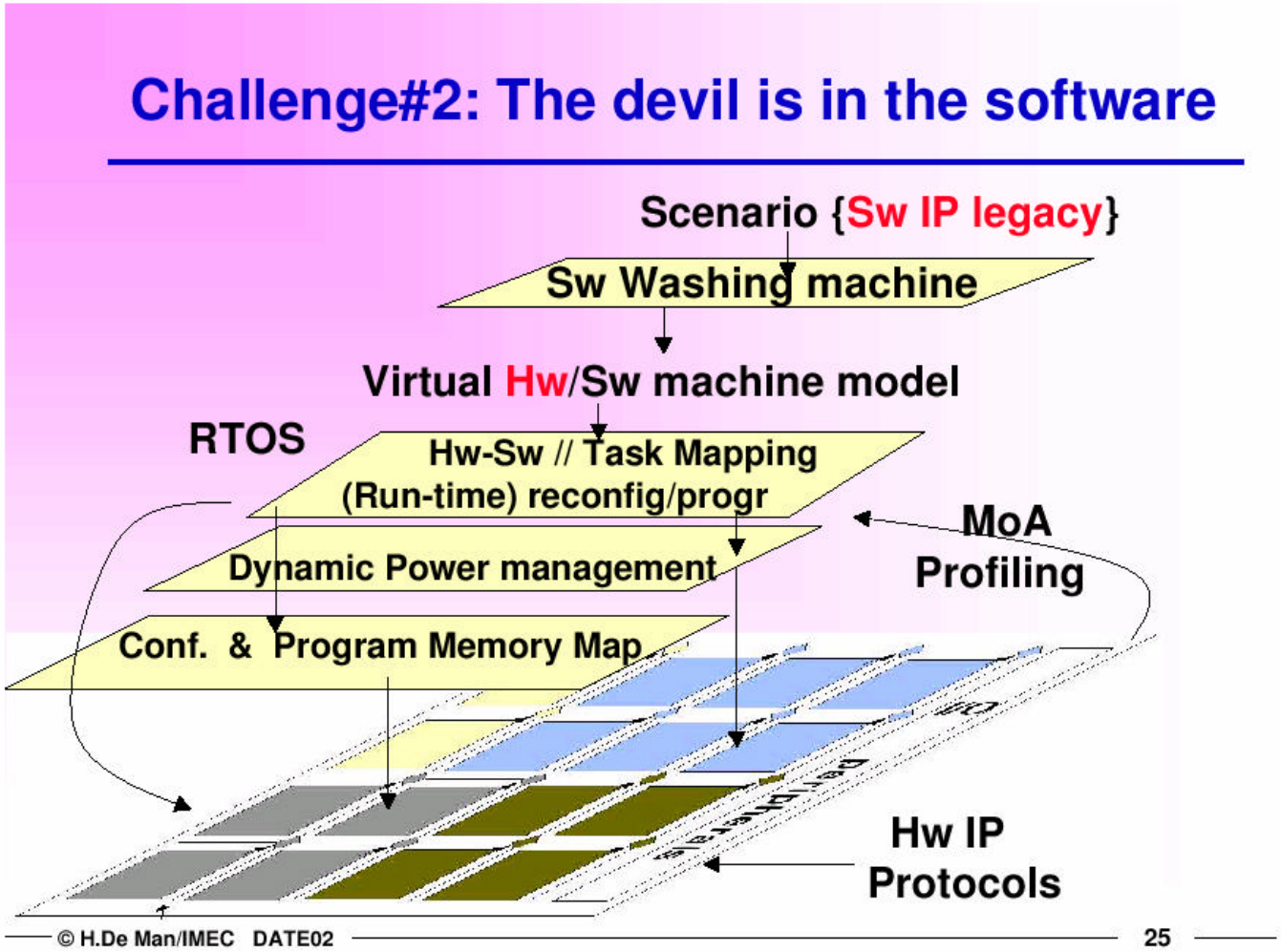
Networks of Adaptive Computing

Networked Adaptive Computing Machines

Subnetworks for compute intense concurrent tasks (brain zones)



The Mapping Problem



Conclusion

- Is this just a SW problem?
- Will systems design = SW design and implementation?
- Why should a system designer care about HW at all?
  - Except inasmuch that it gives him or her ***choices*** about implementation ***tradeoffs***
  - ***This is a job for a compiler with optimisation options***
- What are the implications for the research community?
  - Can advanced tools and methods overcome designers fear of concurrency?
  - Can we also unify the HW and SW design community into a new one: ***system designers*** who, armed with the right tools, boldly, reliably and quickly can implement highly concurrent applications on networks of configurable processors?