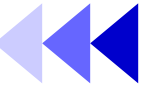
Introduction to Lab & Homework

Kun-Bin Lee 李坤儐
kblee@yankees.ee.nctu.edu.tw

Department of Electronics Engineering
National Chiao Tung University

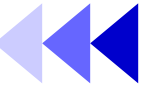
Oct. 29, 2001

Outline



- System-Level Design and Modeling
- IP Core Design
- Lab & Homework

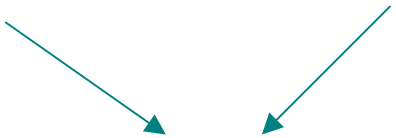
System-level Design



How designs can be
specified?
partitioned?
verified?

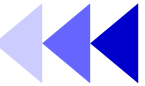
Software
C/C++

Hardware
Verilog/VHDL



A language and modeling platform?

An Opportunity To Do It Right !



System Level

RTL Level

Gate Level

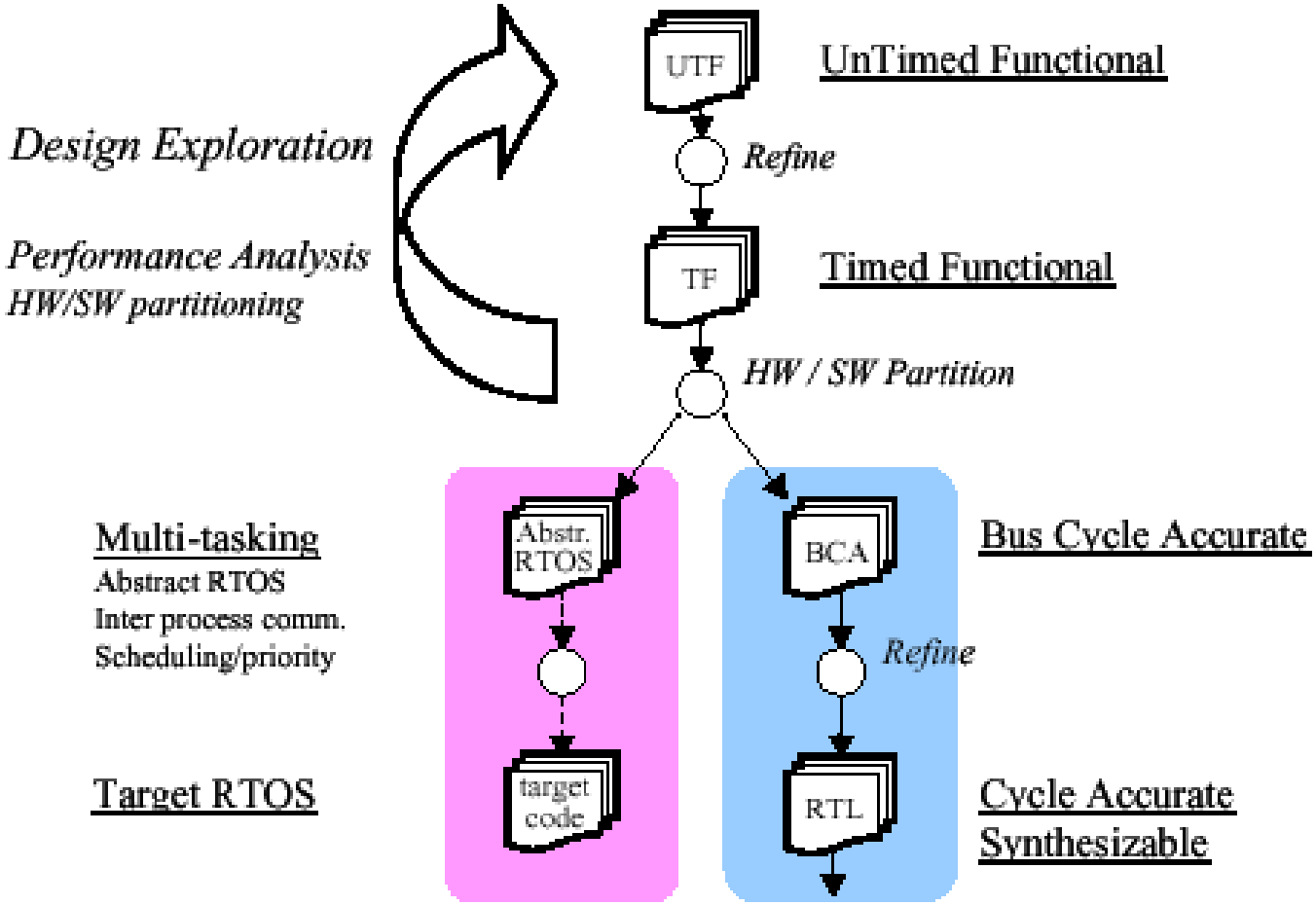
1 SYSTEMC™

2 **Verilog**
VHDL

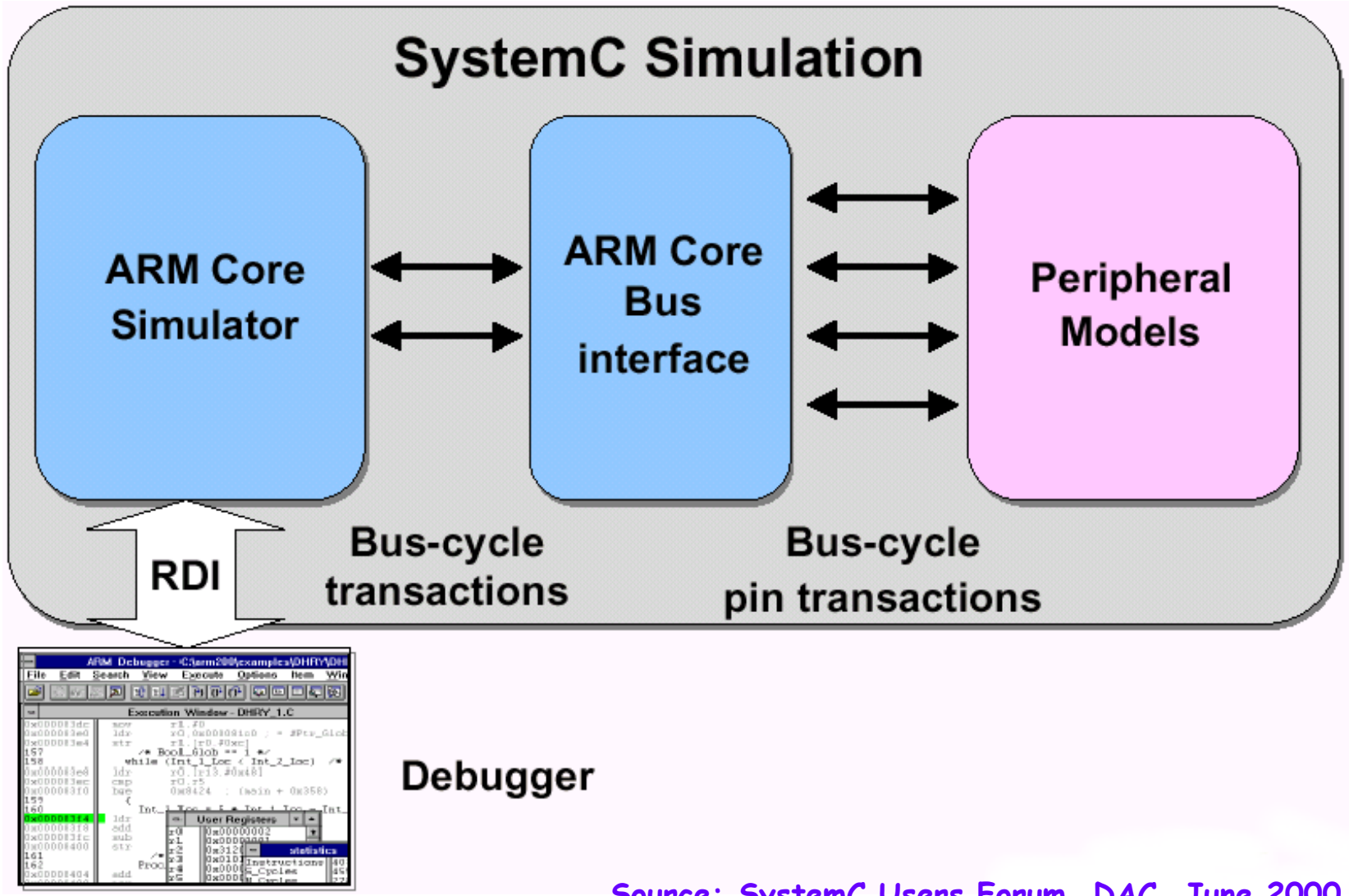
3 **Daisy**
Mentor
Valid

1980s 1990s 2000s

Level of abstraction in SystemC

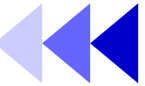


Example

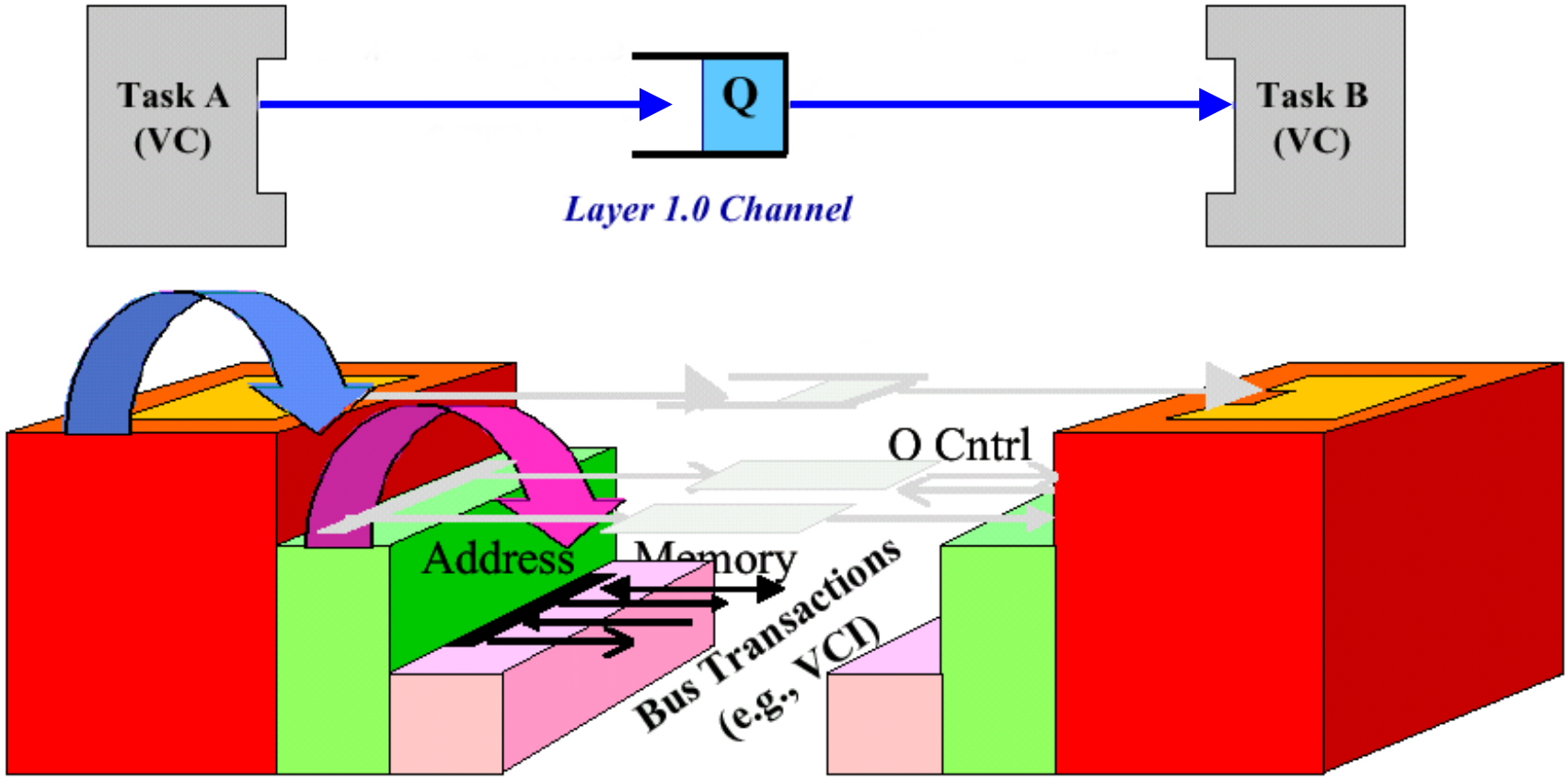


Source: SystemC Users Forum, DAC, June 2000

Communication Refinement Based on SystemC 2.0

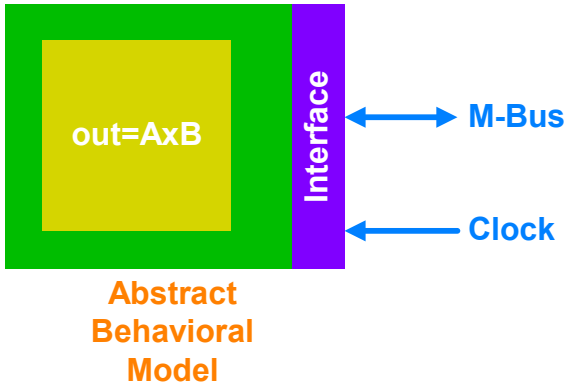


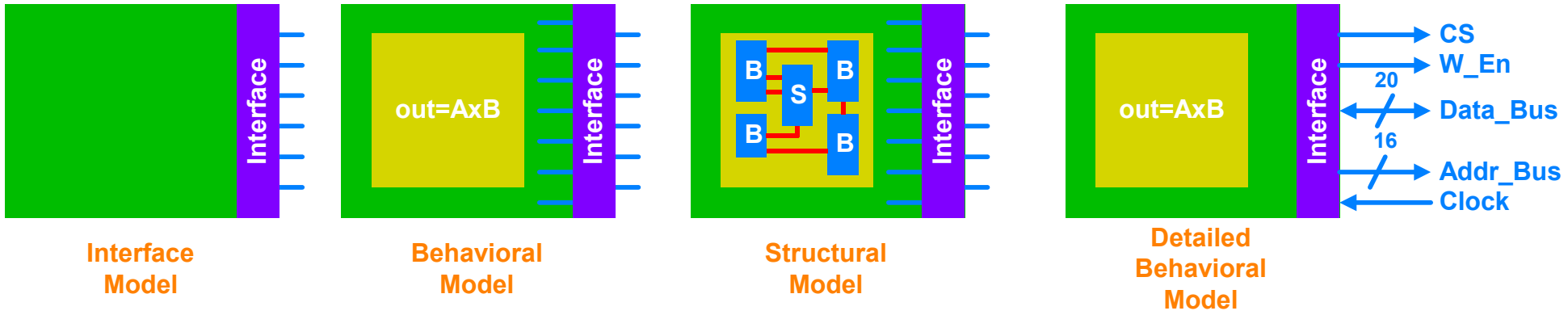
Layer approach
to
guarantee consistency of communication during refinement



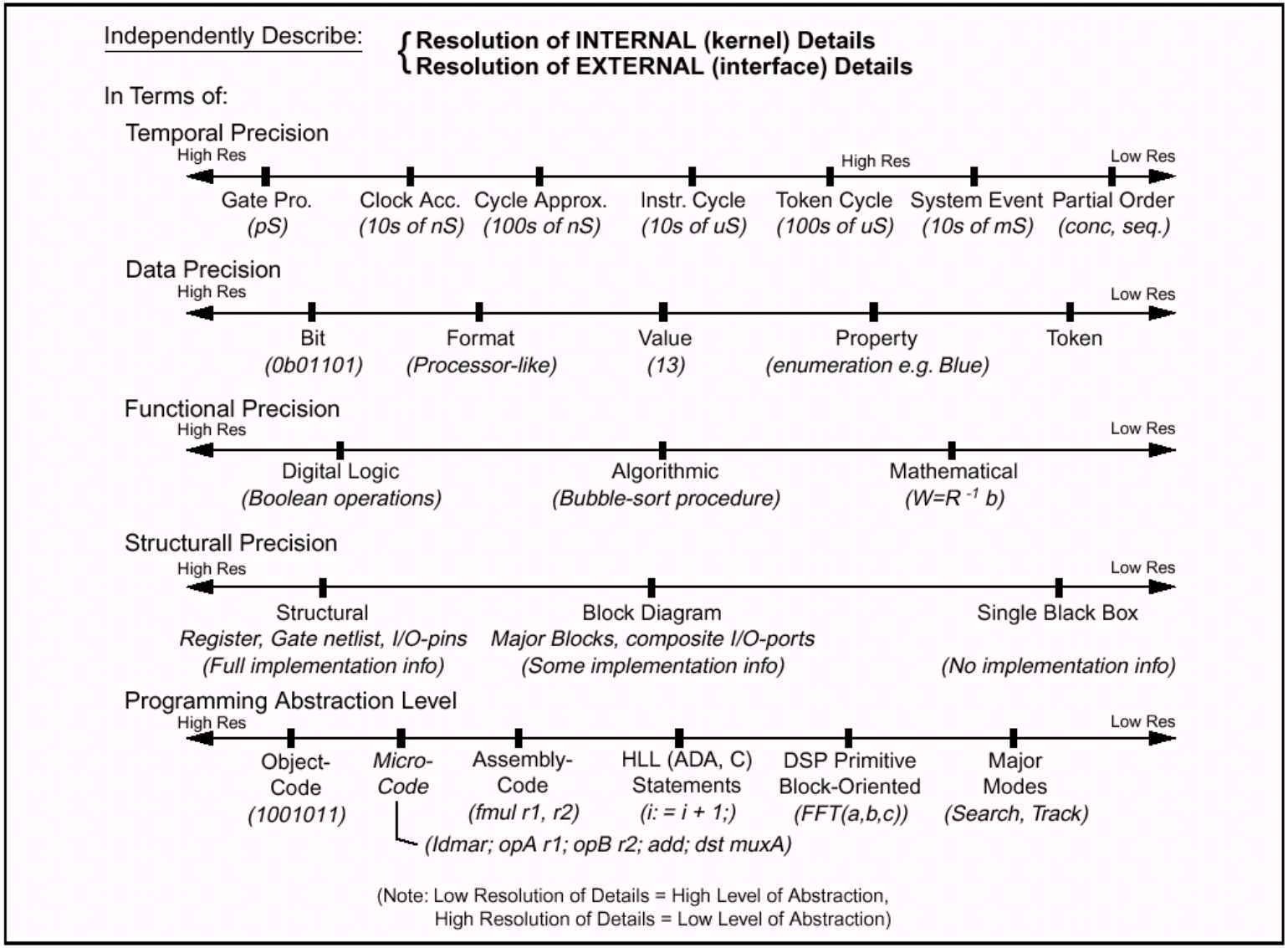
General Modeling Concepts

- Interface model
  - Synonym: bus functional, interface behavioral
- Behavioral model
  - Behavior = function with timing
  - Abstract behavioral model
  - Detailed behavioral model
- Structural model

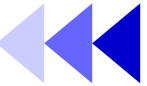




Taxonomy Definition in Different Precision

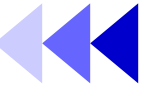


The Intent of Different Level of IP Model



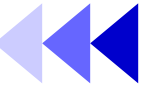
- Design exploration at higher level
  - Import of top-level constraint and block architecture
  - Hierarchical, complete system refinement
  - Less time for validating system requirement
  - More design space of algorithm and system architecture
- Simple and efficient verification and simulation
  - Functional verification
  - Timing simulation/verification
  - Separate internal and external (interface) verification
  - Analysis: power and timing
- Verification support

Issues of IP Modeling

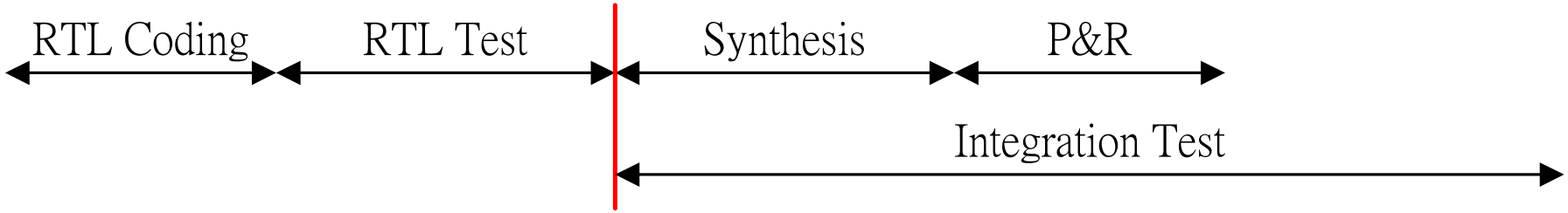


- Attributes
  - What is a sufficient set of model attributes?
  - How are these model attributes validated?
  - How is the proper application of an abstract model specified?
- Two important dimensions of time
  - Model development time is labor intensive: model reusability
  - Simulation time depends upon strategy chosen for mixed domain simulations

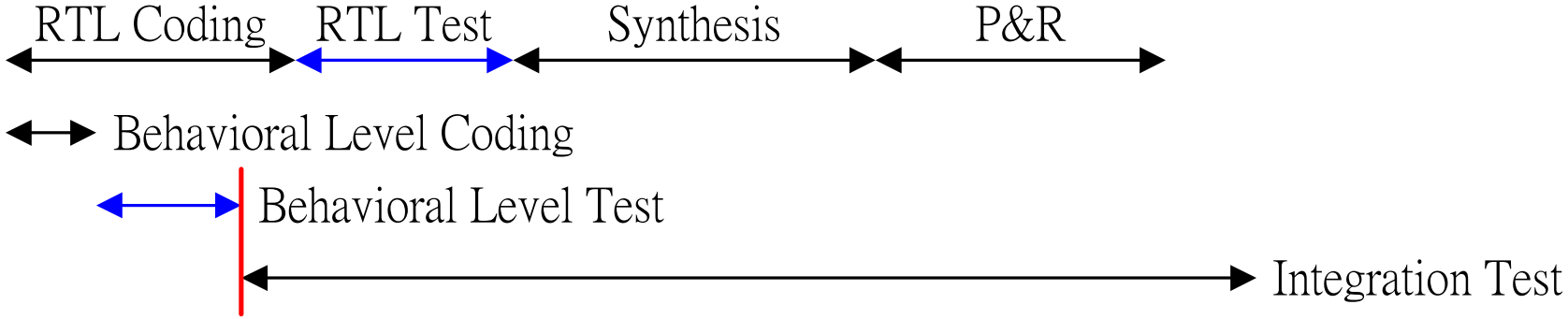
Example: Manage Size and Run-Time



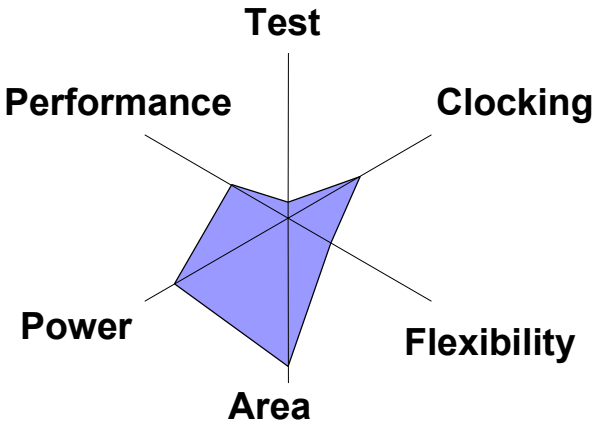
Start at RTL



Start at behavioral level

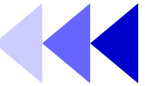


IP Design



- Flexible
  - Programmable
  - Configurable
  - Re-configurable
  - Reusable pattern for both verification and testing
- Robust
  - Verification
  - Test strategy: scan/ATPG, BIST, isolation (MUX). etc
- Friendly
  - Documentation
  - Comment on source code, if any
  - Script

Interface

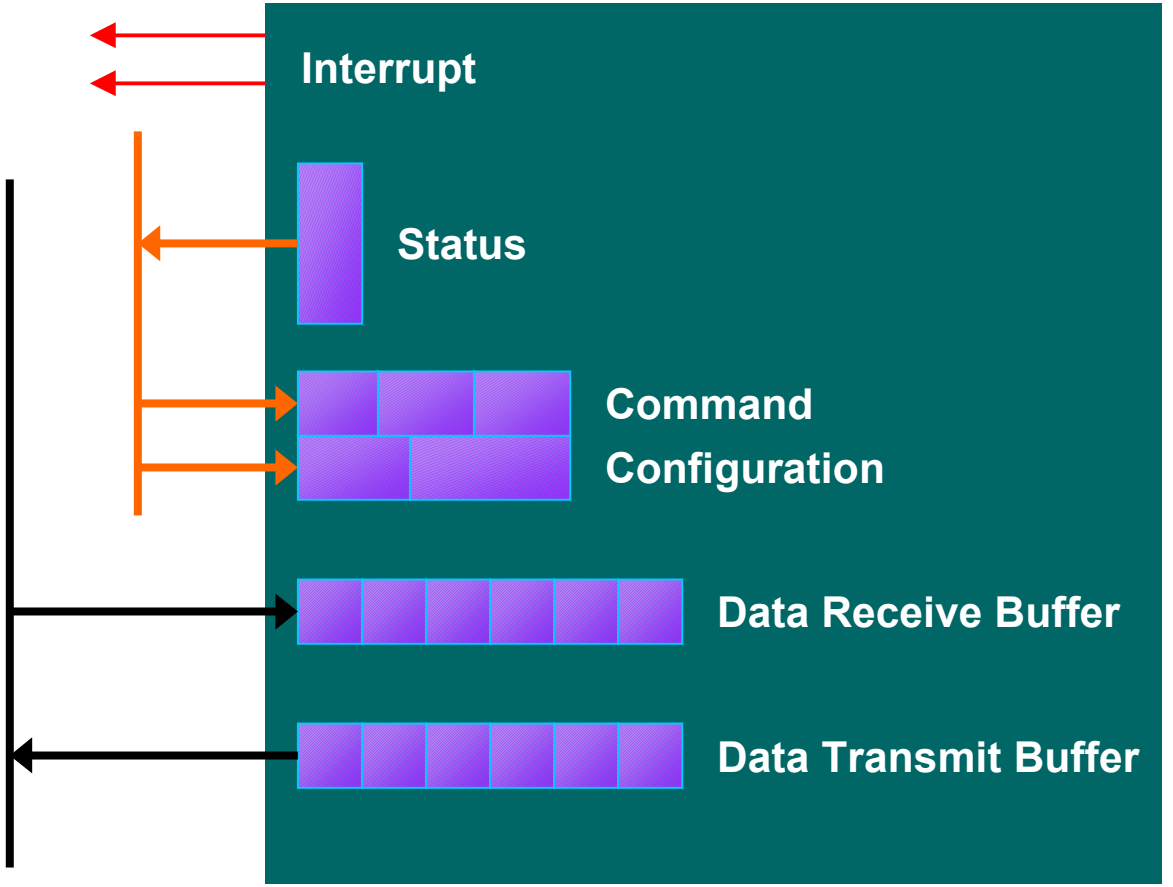


- OCB decides the interface of integrated IPs
- Why VSIA gave up to define OCB standard?
  - It is impractical to define a single common bus architecture and signal protocol given the wide range of SOC applications and performance requirement
  - Too many commercial OCBs available already
    - ARM: AMBA
    - IBM: OPB & PLB
    - Philips: PI & PI2 Bus
    - Palmchip: Mbus & Palmbus
    - Mentor Graphic - FISP bus
- Instead of it, VSIA defined
  - On-chip bus attributes Specification (OCB 1 2.0)
  - Virtual Component Interface Standard (OCB 2 2.0)

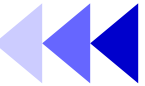
Control Scheme Model

Interrupts

Status Polling (timer)

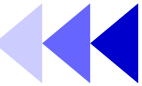


OCB De Facto Standard - AMBA



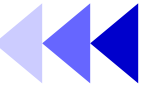
- ARM dominated 70% of embedded processor market
  - AMBA 2.0 is adopted by a lot of companies, including Synopsys, Samsung, ARC, inSilicon, MIPS, etc
  - E.g., AMBA-based DesignWare IP
- AMBA is open and free
  - you have to license it from ARM and ARM owns patent on it
- AMBA Design Kit (ADK) and AMBA Compliance Testbench (ACT) are available from ARM
- Multi-layer AHB is define by ARM from complex multi-master system

VSIA VCI



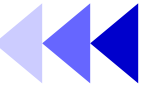
- Some vendors are developing VCI wrapper generator
- Some vendors begin to provide VCI-AMBA wrapper
- Some vendors begin to provide design with VCI interface

Parameterized IP Design



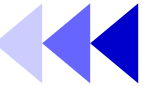
- Why to parameterize IP?
  - Provide flexibility in interface and functionality
  - Facilitate verification
- Parameterizable types
  - Logic/Constant functionality
  - Structural functionality
    - Bit-width、depth of FIFO、regulation and selection of sub-module
  - Design process functionality (mainly in test bench)
    - Test events
    - Events report (what, when and where)
    - Automatic check event

Verification



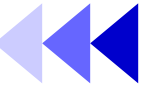
- Function
- Architecture functional verification
- Control algorithm and protocol verification
- Hardware RTL verification
- hardware and software interface certification (via HW-SW Co-sim)
- Hardware and software coverification (via rapid prototyping)
- Netlist verification

Reusable Design - Test Suite



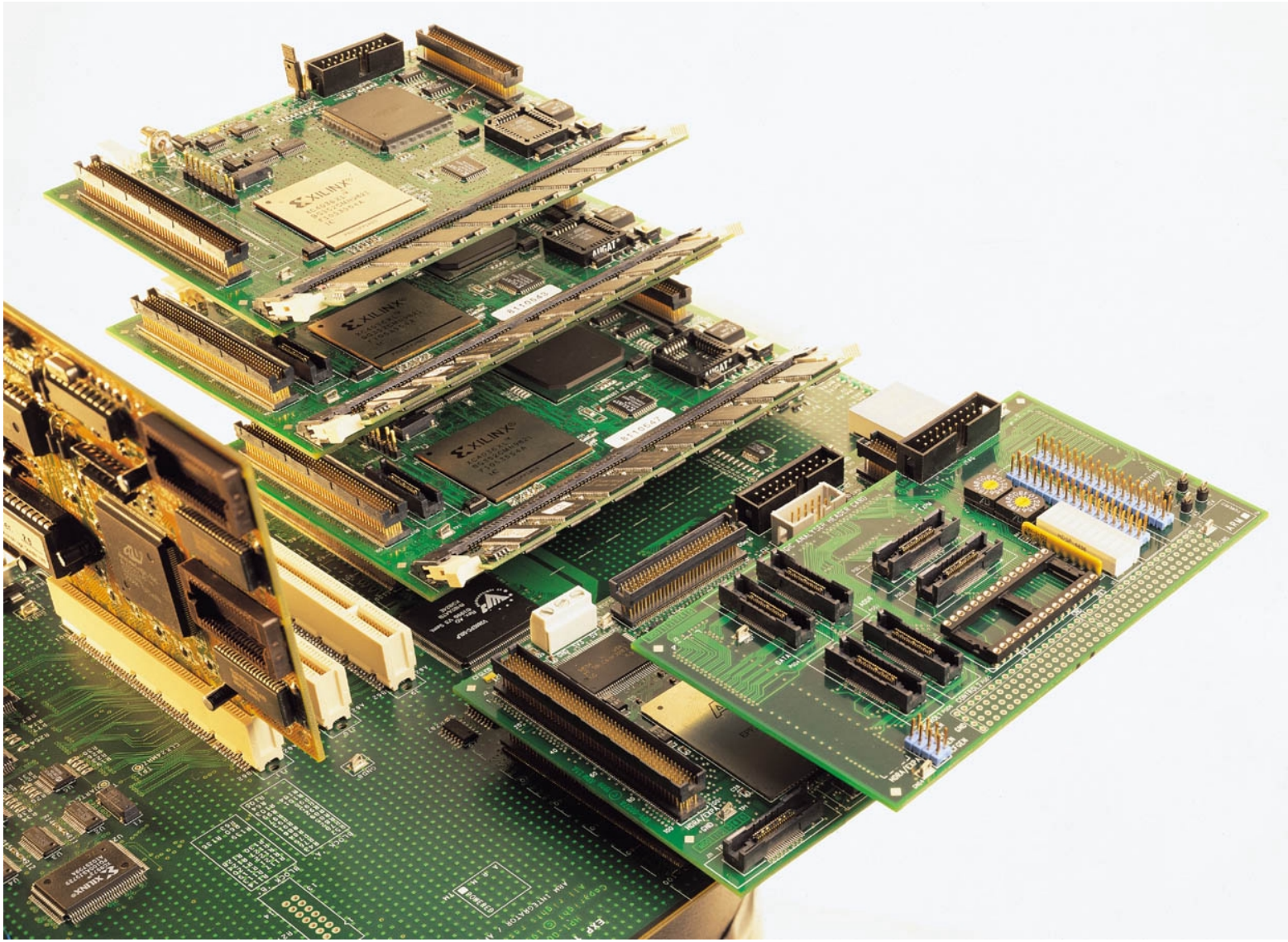
- Test events
  - Automatically adjusted when IP design is changed
  - Partition test events to reduce redundant cases when test for all allowable parameter sets at a time
- Debug mode
  - Test for the specific parameter set at a time
  - Test for all allowable parameter sets at a time
  - Test for the specific functionality
  - Step control after the specific time point
- Display mode of automatic checking
  - display[0]: event current under test
  - display[1]: the time error occurs
  - display[2]: expected value and actual value
  - ...

Equipment

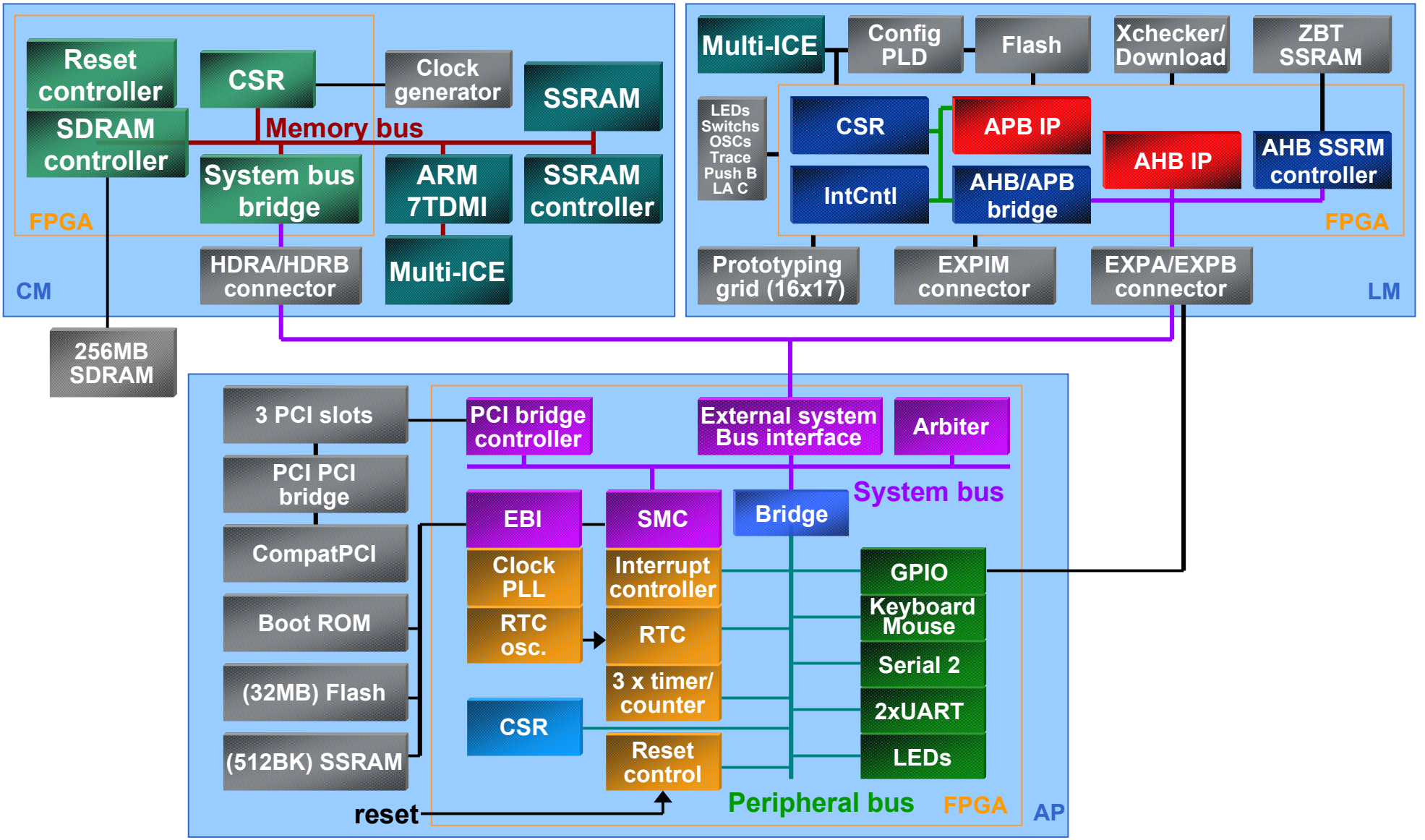


- Software tools
  - ARM Developer Suite v1.1 10 + 1
- Development boards
  - Core Module: 4 7TDMI + 2 720T
  - Logic Module (XCV2000E Xilinx VirtexE) 6
  - ASIC Development Platform (Integrator/AP AHB) 6
  - Multi-ICER Interface v2.0 2 ➔ 6

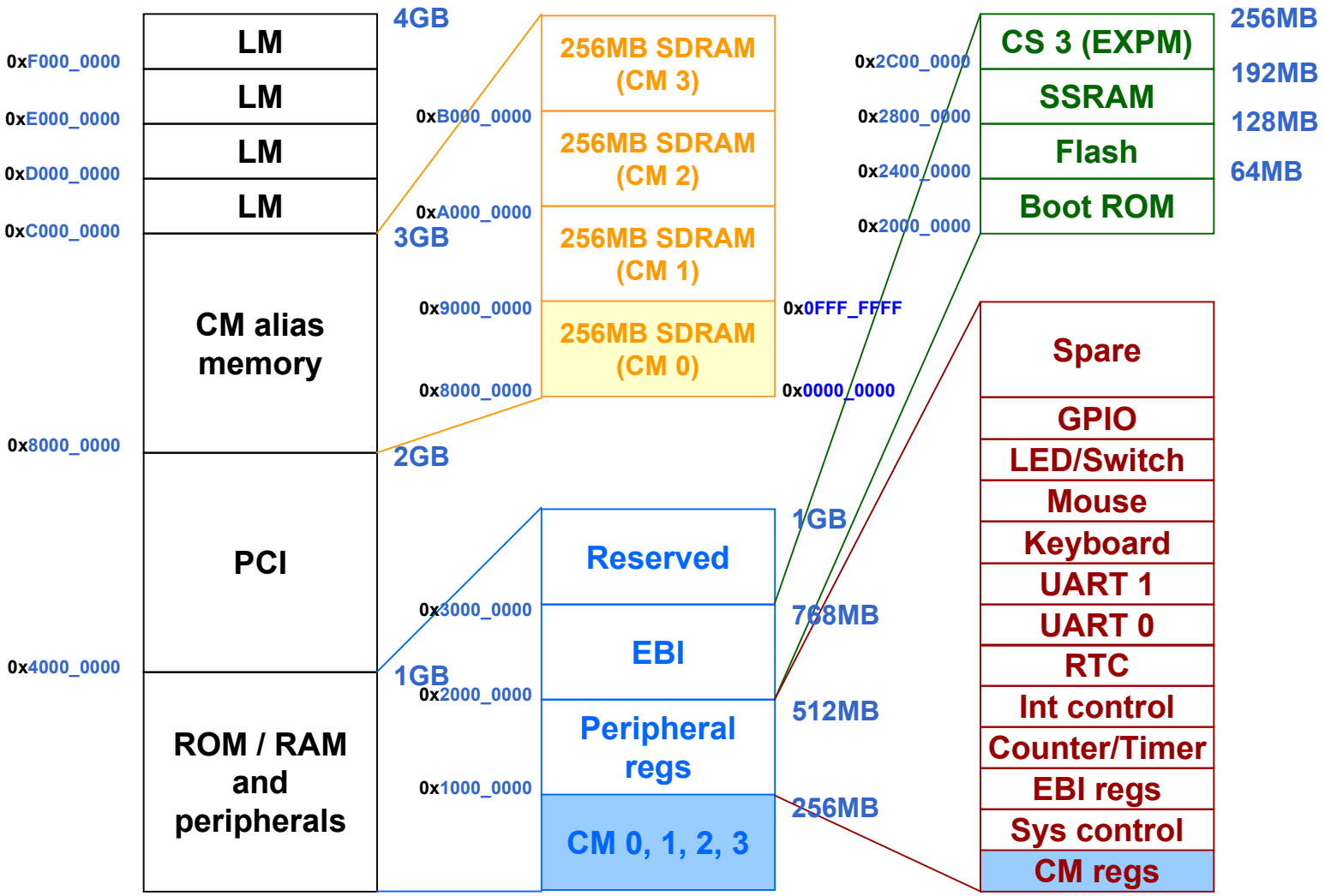
Integrate All The Modules in The Integrator



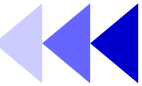
ARM Integrator



System Memory Map

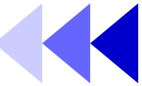


In the labs



- Lab 1
  - Be familiar with software development environment
  - Write code (driver) for ARM-based platform design
  - Software cost estimation
- Lab 2
  - Target system environment and resources
  - HW/SW communication: polling and Interrupt
- Lab 3
  - Modeling
  - High-level HWSW co-verification
- Lab 4
  - Interface design: AMBA-compliant IP
  - HW/SW interaction

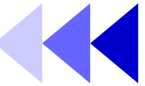
Homework



- JPEG encoder IP for applications such as
  - Scanner (low cost, encoder only)
  - Digital still camera (low power, encoder and decoder)
- Be aware of the differences between the software and final target platform.
  - I/O interface, Data structure, Partition of Functional modules

- HW 1
  - Understanding JPEG ➔ Limitation of programmable processor
- HW 2
  - Port to ARM Integrator environment
- HW 3
  - Model hardware portion IP of JPEG in C level
  - HW/SW co-verification
- HW 4
  - AMBA-compliant IP
  - FPGA-proven design

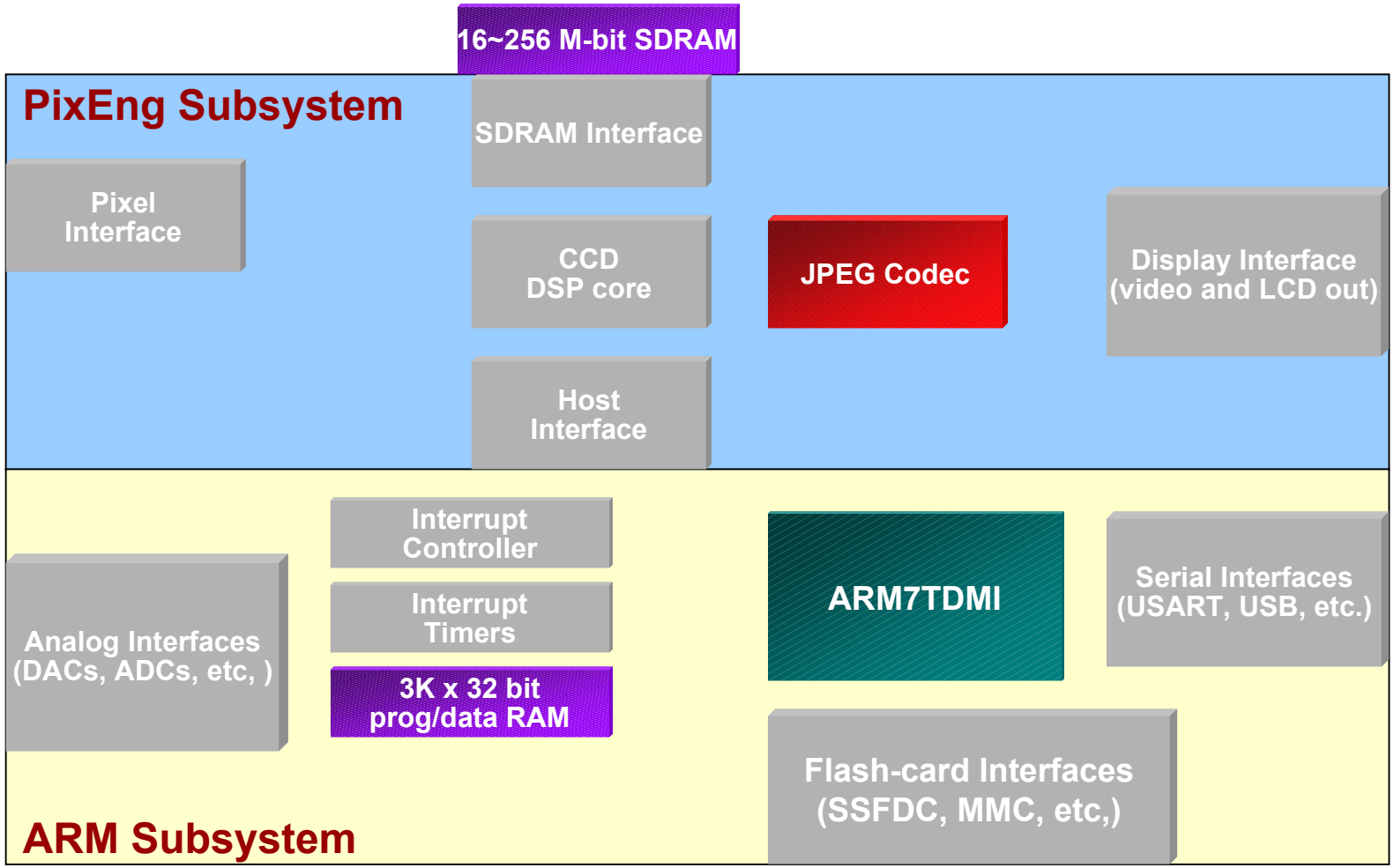
Examples for JPEG related IPs



- JPEG encoder IP
  - http://www.amphion.com/cs6100.html
  - http://cadal3.cie.nsysu.edu.tw/news/IP_contest_2000/S05.html
- DCT engine
  - http://www.amphion.com/cs6310.html
  - http://www.xilinx.com/ipcenter/dct_lounge/

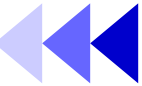
and more: www.google.com

JPEG in the Integrated Chip



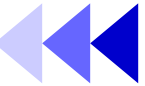
FujiFilm MD2305 High-performance Digital Camera Processor

Final Due



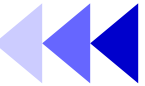
- Design
  - One of the following design
    - Components for JPEG encoder, e.g., DCT
    - Baseline JPEG encoder
    - Feature-enhanced JPEG encoder, e.g.,
      - color conversion
      - design for both JPEG encoder and decoder
  - FPGA-proven design
- Deliverables
  - http://vlsi.ee.ccu.edu.tw/~ip/Specification.htm
  - http://vlsi.ee.ccu.edu.tw/~ip/news/IP_contest_2001/deliver.doc

More Information



- JPEG FAQ and reference code information
  - http://www.faqs.org/faqs/jpeg-faq/
- IP measurement for reusability
  - http://www.openmore.com/
- Reuse guides
  - Reuse Methodology Manual for System-on-A-Chip Designs, by Michael Keating and Pierre Bricaud, 2ed. 1999
  - http://www.xilinx.com/ipcenter/designreuse/xrfg.htm

In IP Core Design, We Focus on



- HW/SW co-design
  - HW/SW partition and scheduling
  - HW/SW communication and synchronization
  - HW/SW co-verification
  - More realistic environment
- Interface design
  - Industry de facto standard: ARM AMBA
  - Optional: VCI + AMBA bus wrapper
  - HW/SW interface
- Cost of the hole IP
  - Hardware (gate count)
  - Software driver
  - Memory (software code + hardware buffer)
- Under the performance constraint, optimize for cost and/or power consumption