Chapter 3
VCI Interface, AMBA Bus and Platform-based Design

C. W. Jen 任建葳

cwjen@twins.ee.nctu.edu.tw

Outline

- VCI Interface Standards
- AMBA - On Chip Buses
- Platform-based SoC Design
- SoC Design Flow

Outline

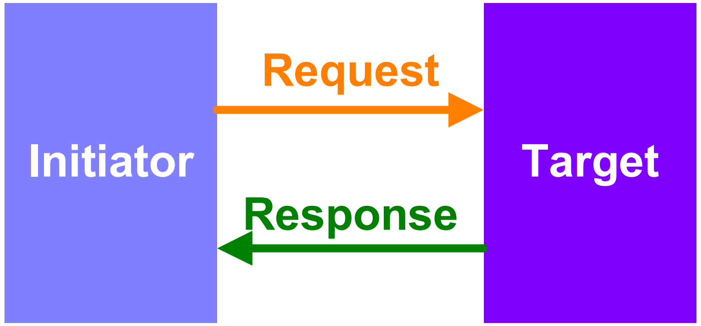
- **Single VCI Interface Standards**
- AMBA - On Chip Buses
- Platform-based SoC Design
- SoC Design Flow

Virtual Component Interface - VCI

- What is VCI
  - A request-response protocol, contents and coding, for the transfer of requests and responses
- Why VCI
  - Other IP blocks not available ‘wrapped’ to the on-chip communications may work with IP wrappers. VCI is the best choice to start with for an adaptation layer
- VCI specifies
  - Thee levels of protocol, compatible each other
    - Advanced VCI (AVCI),
    - Basic VCI (BVCI)
    - Peripheral VCI (PVCI)
  - Transaction language

VCI Point-to-Point Usage

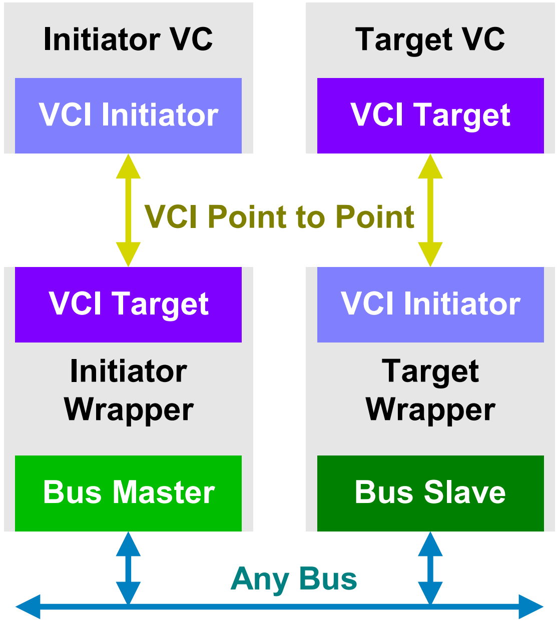
- Simplicity: small footprint and high bandwidth
  - Initiator only request
  - Target only respond
  - If a VC needs both, implement parallel initiator and target interfaces
- Star topology



Point-to-point usage

VCI Usage with a Bus

- Used as the interface to a wrapper (a connection to a bus)
  - OCB suppliers provide VCI wrappers.
  - EDA vendors provide tools to create wrapper automatically



VCI usage with a bus

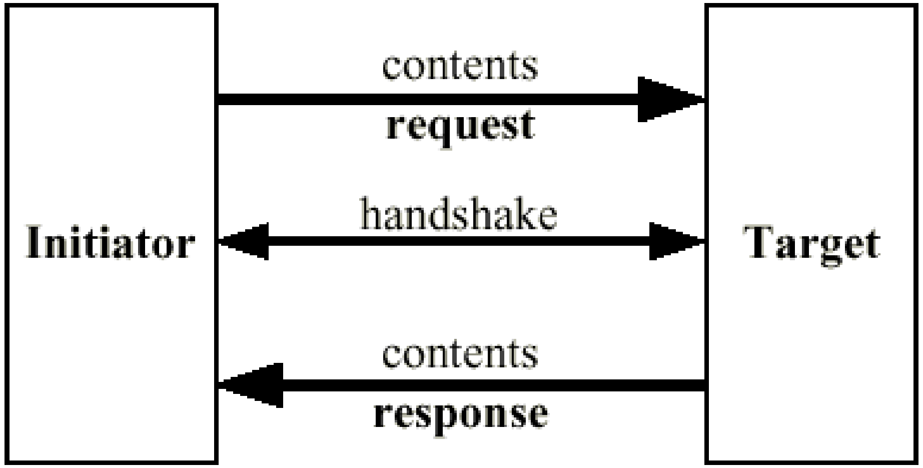
Split Protocol

- The timing of the request and the response are fully separate. The initiator can issue as many requests as needed, without waiting for the response.

BVCI	order kept
AVCI	request tagged with identifiers, allow different order
PVCI	no split protocol
	each request must be followed by a response before the initiator can issue a new request

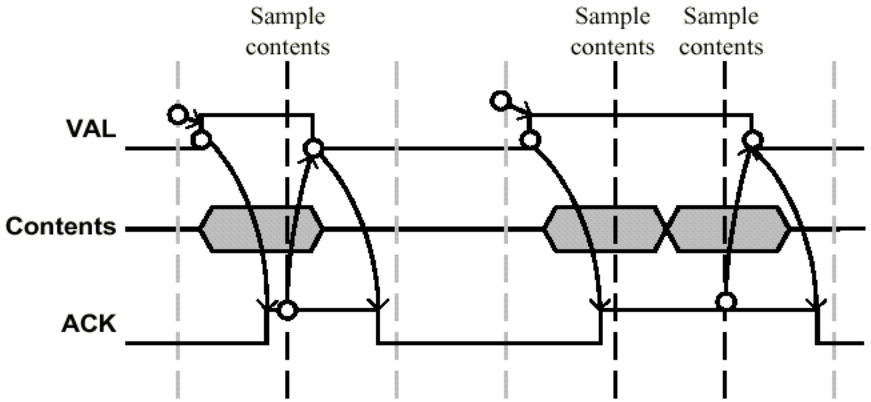
Initiator – Target Connection (PVCI)

- The request contents and the response contents are transferred under control of the protocol: 2-wire handshake Valid (VAL) and Acknowledge (ACK)

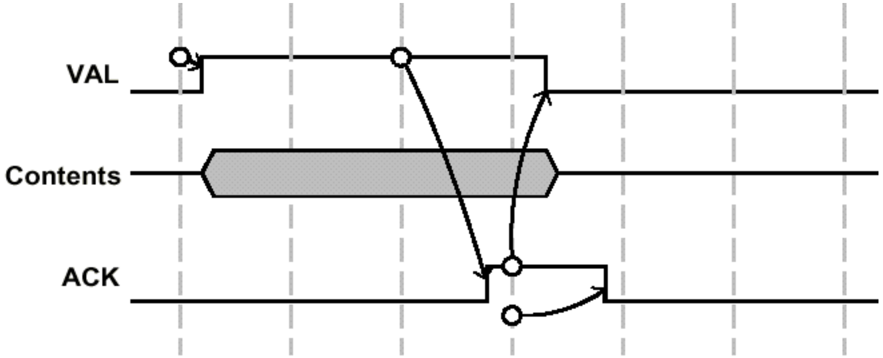


Control Handshake

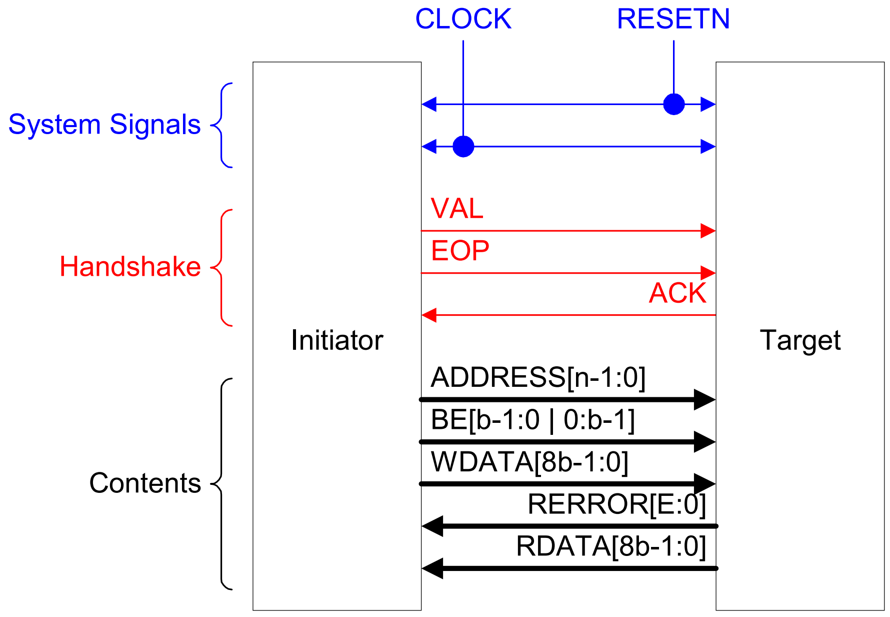
- Asynchronous



- Synchronous



Request and Response Contents



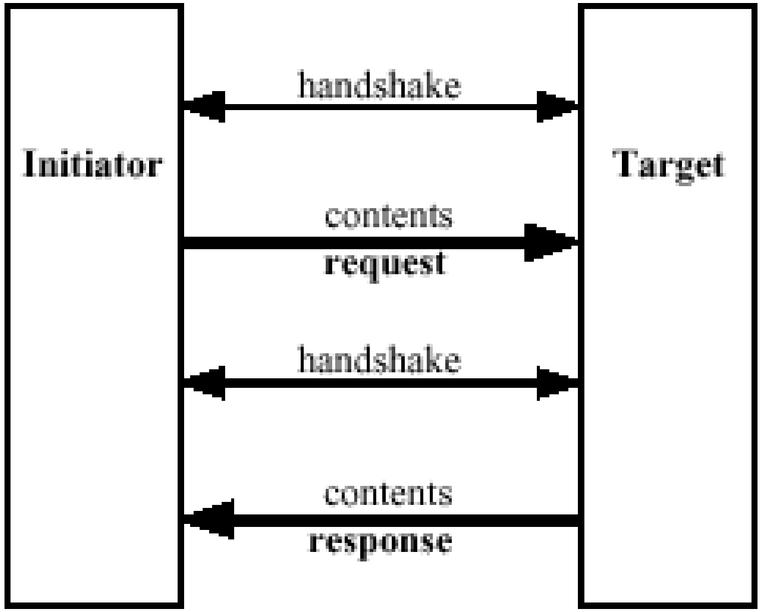
- Main PVCI features
  - Up to 32-bit Address
  - Up to 32-bit Read Data
  - Up to 32-bit Write Data
  - Synchronous
  - Allows for 8-bit, 16-bit, and 32-bit devices
  - 8-bit, 16-bit, and 32-bit Transfers
  - Simple packet, or ‘burst’ transfer

PVCI Protocol

- Transfer Request
  - Read8, Read16, Read32, Read N cells
  - Write8, Write16, Write32, Write N cells
- Transfer Response
  - Not Ready
  - Transfer Acknowledged
  - Error
- Packet Transfer
  - The packet (burst) transfer makes is to transfer a block of cells with consecutive addresses
  - While the EOP signal is de-asserted during a request, the address of the next request will be ADDRESS+cell_size

Initiator – Target Connection (BVCI)

- The request and response handshakes are independent of each other
  - Request handshake: CMDVAL and CMDACK
  - Response handshake: RSPVAL and RSPACK



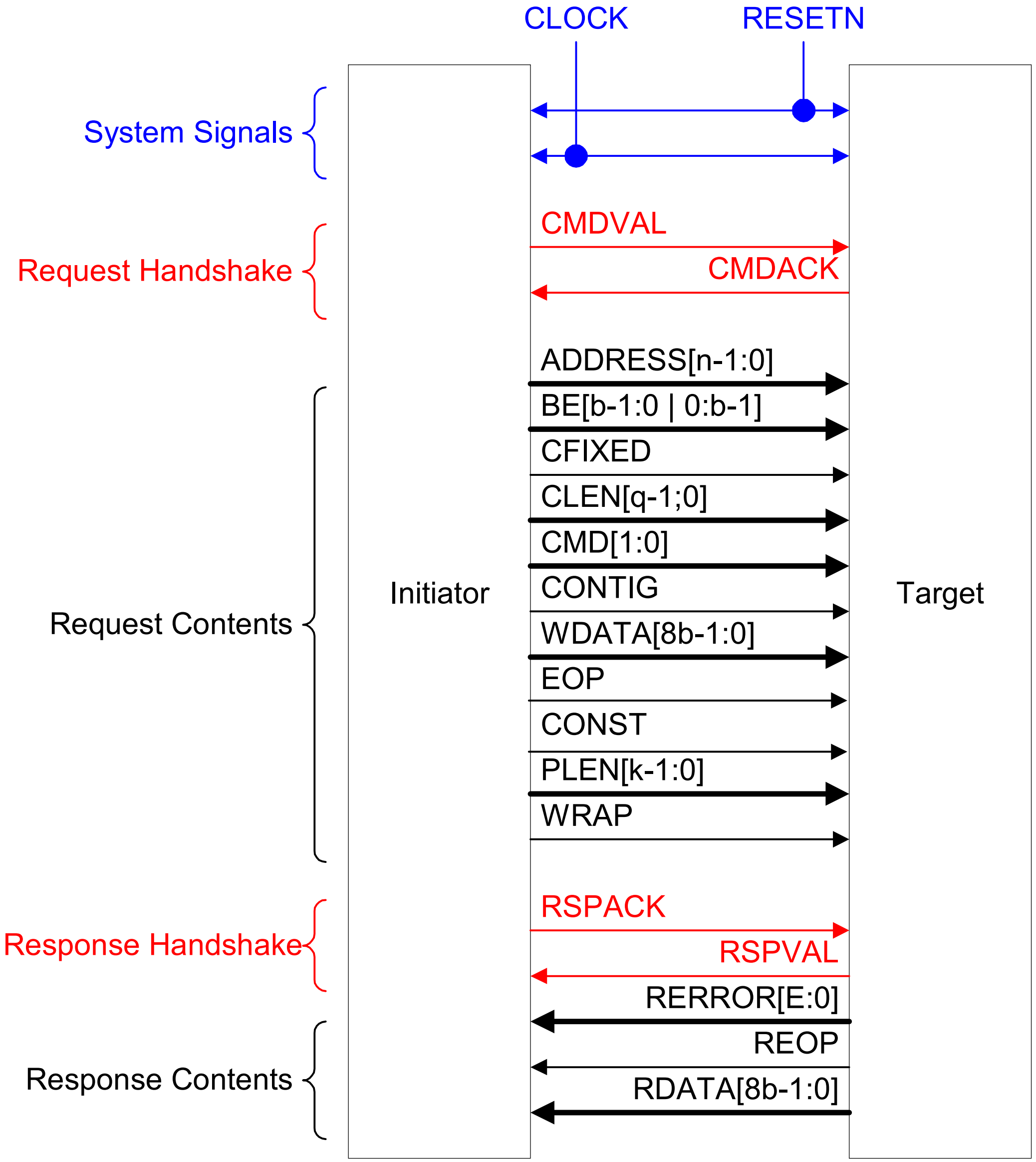
Cells, Packets, and Packet Chains

- Each handshake transfers a cell across the interface. The cell size is the width of the data passing across a VCI.
  - 1, 2, 4, 8, or 16 bytes for BVCI
  - 1, 2, 4, bytes for PVCI
- Cell transfers can be combined into packets, which may map onto a burst on a bus.
  - A VCI operation consists of a request packet and a response packet
  - Packets are atomic
  - Packets are similar in concept to “frames” in PCI
- Packets can be combined into chains, to allow longer chains of operations to go uninterrupted.

Request and Response Contents

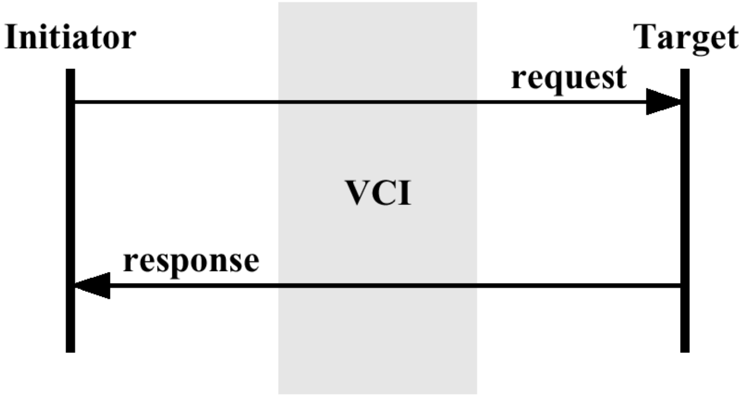
- Request contents are partitioned into three signal groups and validated by the CMDVAL signal
  - Opcode, specify the nature of the request (read or write)
  - Packet Length and Chaining
  - Address and Data
- Response contents validated with the RSPVAL. Each request has its response.
  - Response Error
  - Read Data

BVCI Signals



BVCI Protocol

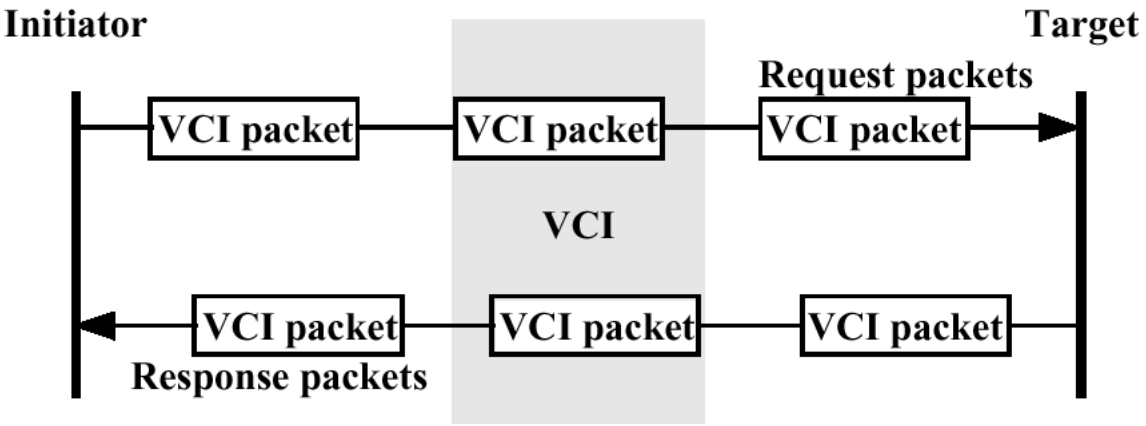
- The protocol has three stacked layers: transaction layer, packet layer, and cell layer
- Transaction layer: A pair of request and response transfers



  - Above hardware implementation
  - A series of communicating objects that can be either hardware or software modules
  - The information exchanged between initiator and target nodes is in the form of a request-response pair

Packet Layer

- The packet layer adds generic hardware constraints to the system model
- In this layer, VCI is a bus-independent interface, just physically point-to-point



- A transaction is called a “VCI operation” if the information is exchanged using atomic request and response transfers. In a packet layer, a VCI transaction decomposes into one or more operations.

Packet

- Packet is the basic unit of information that can be exchanged over the VCI in an atomic manner.
- Multiple packets can be combined to form larger, non-atomic transfer units called packet chains.
- A VCI operation is a single request-response packet pair.
- Packet length is the number of bytes transferred
- The content of a packet depends on whether it is a request or response packet and the type of operation being carried out - such as read, write, etc.

Cell Layer

- The cell layer adds more hardware details such as interface width, handshake scheme, wiring constraints, and a clock to the system.
- A cell is the basic unit of information, transferred on rising CLOCK edges under the VAL-ACK handshake protocol, defined by the cell layer. Multiple cells constitute a packet.
- Both request and response packets are transferred as series of cells on the VCI. The number of cells in a packet depends on the packet length and the interface width.

BVCI Operations

- The basic transfer mechanism in VCI is packet transfer. A packet is sent as a series of cells with the EOP field in the last cell set to value 1. Each cell is individually handshaken under the VAL-ACK handshake. Either the initiator or the target can insert wait cycles between cell transfers by de-asserting VAL or ACK.

- Transfer Requests
  - Read/Write a cell
  - Read/Write a packet from random/contiguous addresses
  - Read/Write a packet from one address
  - Issue a chain of packets

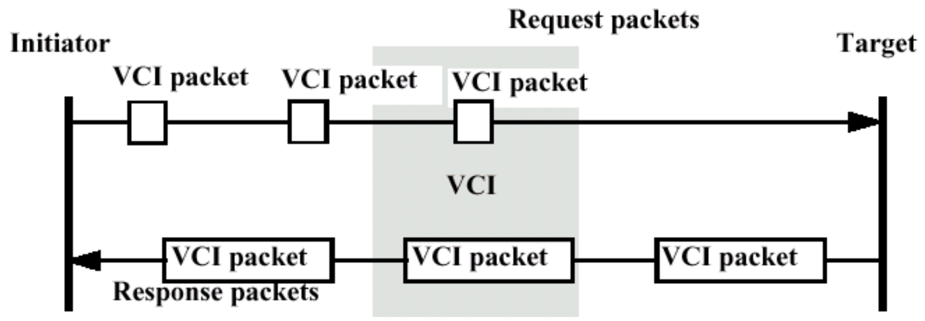
- Transfer Responses
  - Read/Write cell/packet successful
  - Read/Write packet general error
  - Read/Write bad data error
  - Read/Write Abort disconnect

Advanced VCI

- AVCI supports out-of-order transactions and an advanced packet model
- Advanced Packet Model
  - – Request and response packets do not have the same size
  - – Need
    - request packet: one cell, set the start address and address behavior
    - response packet: many cells, read data return
- Arbitration hiding
  - – pipelines of both the request and response packets
- Source Identification
  - – a unique identifier for each initiator

AVCI Protocol

- Still 3 layers similar to BVCI. No difference in the transaction layer, slightly differ in the packet and cell layers
- Packet layer



- Cell layer
  - AVCI cell layer differs from BVCI with some additional fields, with side band signals for arbitration hiding
  - Arbitration hiding signals are separately handshaken

Concept of the Bus

- A group of lines shared for interconnection of the functional modules by a standard interface
  - E.g., ARM AMBA, IBM CoreConnect
- Interconnection structure
  - Point-to-Point
  - On-chip bus
  - On-chip network

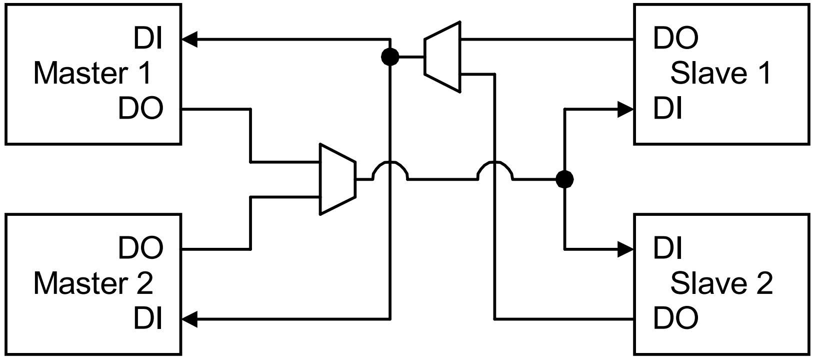
Differences Between Traditional Bus/OCB

- The root: I/O pins are limited and fixed
- The characteristics of a traditional bus
  - Shared I/O
  - Fixed interconnection scheme
  - Fixed timing requirement
  - Dedicated address decoding
- For a OCB
  - Routing resource in target device (e.g. FPGA, ASIC)
  - Bandwidth and latency are important

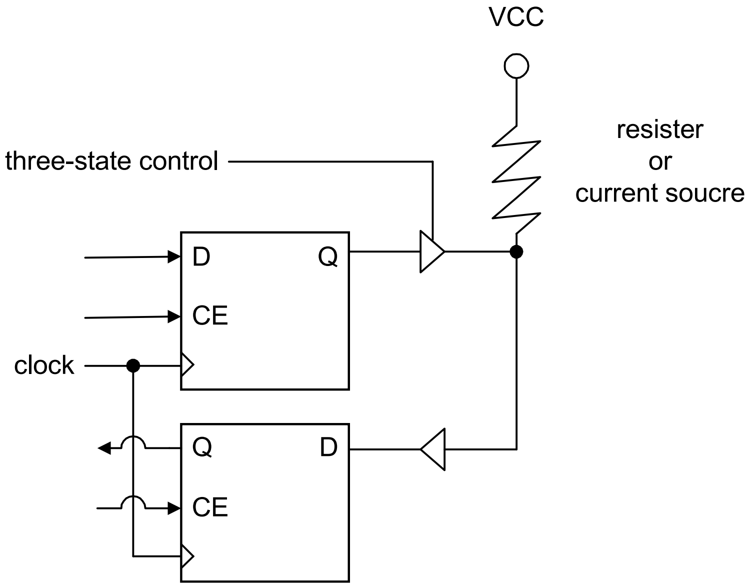
Shared I/O

- Three-state I/O. E.g. multiple masters, input/output
  - Slower than direct interconnection
  - Limited by bus keeper or quality of routing resource in the target device
  - Solution in OCB: multiplexer logic interconnection
  - Xilinx design guideline: We recommend using multiplexer-based buses when designing for reuse since they are *technology-independent* and more portable.
- Multiplexed functional I/O. E.g. address/Data.
  - Need more time to transfer the same amount of data
  - Solution in OCB: separate buses

Physical View of Shared I/O



Multiplexer-based buses



Three-state I/O

Physical Constraints

- Fixed Interconnection Scheme
  - Traditional buses usually routed across a standard backplane
  - OCB allowed a variable interconnection scheme that can be defined by the system integrator at the "*tool level*"
- Fixed Timing Requirement
  - Traditional buses have fixed timing requirements:
    - They are both tested as sub-assemblies
    - They have highly capacitive and inductive loads
    - They are designed for the *worst-case* operating conditions when unknown bus modules are connected together
  - OCB has a variable timing specification that
    - Can be enforced by place & route tools (tool level)
    - Usually does not specify absolute timing
    - Possibly only specifies a single timing specification（WISHBONE, Silicore）

Address Decoding

- Standard microcomputer buses usually use the full address decoding technique
  - That's because the interconnection method does not allow the creation of any new signals on the interface
- OCB can only use partial address decoding
  - Higher speed address decoder
  - Less redundant address decoding logic
  - Integrator must define part of the address decoder logic for each IP core (disadvantage)

Bus Components

- Switch or node
  - arbitration,routing
- Converter or bridge (type converter)
  - from one protocol to another
- Size converter
  - buffering capacity

Bus Transaction

- Bus cycle
  - one bus clock period
- Bus transfer
  - read or write operation, 1 or more bus cycles
  - terminated by a completion response from the addressed slave
- Burst operation
  - one or more data transaction, initiated by a bus master

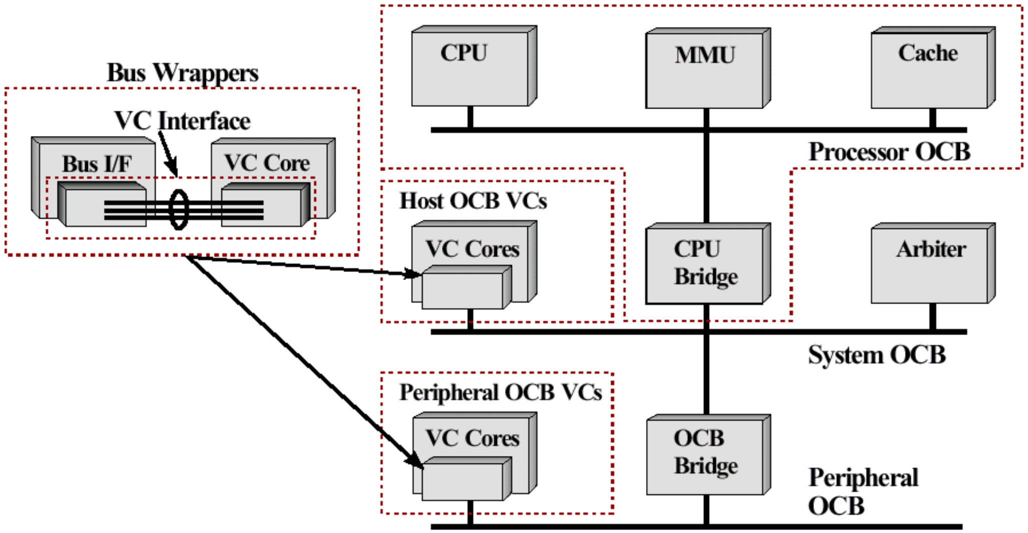
Bus Transfer

- A means to transfer data on the shared communication lines between VCs
- Protocol: guarantee the correct transfer
  - request arbiter to use bus
  - request sender to send data → sender ACK →send data → receiver ack to receipt
  - if error, re-send
  - release bus
- Transfer modes
  - read or write
  - asynchronous or synchronous
  - transfer size 8, 16, 32, 64, 128 bits
  - transfer operations

Bus Signals

- Address and data
- Interface controls
- Arbitration
- Interrupt
- Error reporting
- System level
- Test/Boundary scan
- Others

Bus Hierarchy



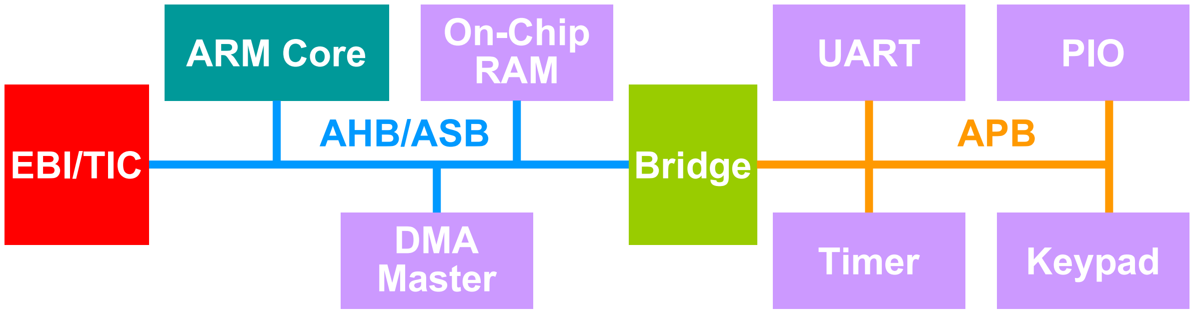
- The structure of multiple buses within a system, organized by bandwidth
- Local processor bus
  - highly processor-specific
  - processor, cache, MMU, coprocessor
- System bus (backbone)
  - RISC processor, DSP, DMA (masters)
  - Memory, high resolution LCD peripheral
- Peripheral bus
  - Components with other design considerations (power, gate count, etc.)
  - Bridge is the only bus master

Outline

- VCI Interface Standards
- **AMBA - On Chip Buses**
- Platform-based SoC Design
- SoC Design Flow

ARM OCB - AMBA

- Advanced Microcontroller Bus Architecture (AMBA)
- AMBA 2.0 specifies
  - the Advanced High-performance Bus (AHB)
  - the Advanced System Bus (ASB)
  - the Advanced Peripheral Bus (APB)
  - test methodology



A typical AMBA system

Features of AMBA

- AHB is superior to ASB in
  - performance and synthesizibility and timing verification

Advanced High-performance Bus (AHB)
High performance
Pipelined operation
Multiple bus master
Burst transfers
A single centralized decoder

Split transactions
single-cycle bus master handover
single-clock edge operation
non-tristate implementation
wider data bus configurations (8/16/32/64/128 bits)

Advanced System Bus (ASB)
High performance
Pipelined operation
Multiple bus master
Burst transfers
A single centralized decoder

Advanced Peripheral Bus (APB)
Low power
Simple interface

APB access MUST take 2 PLCK cycles

Notes on the AMBA Specification

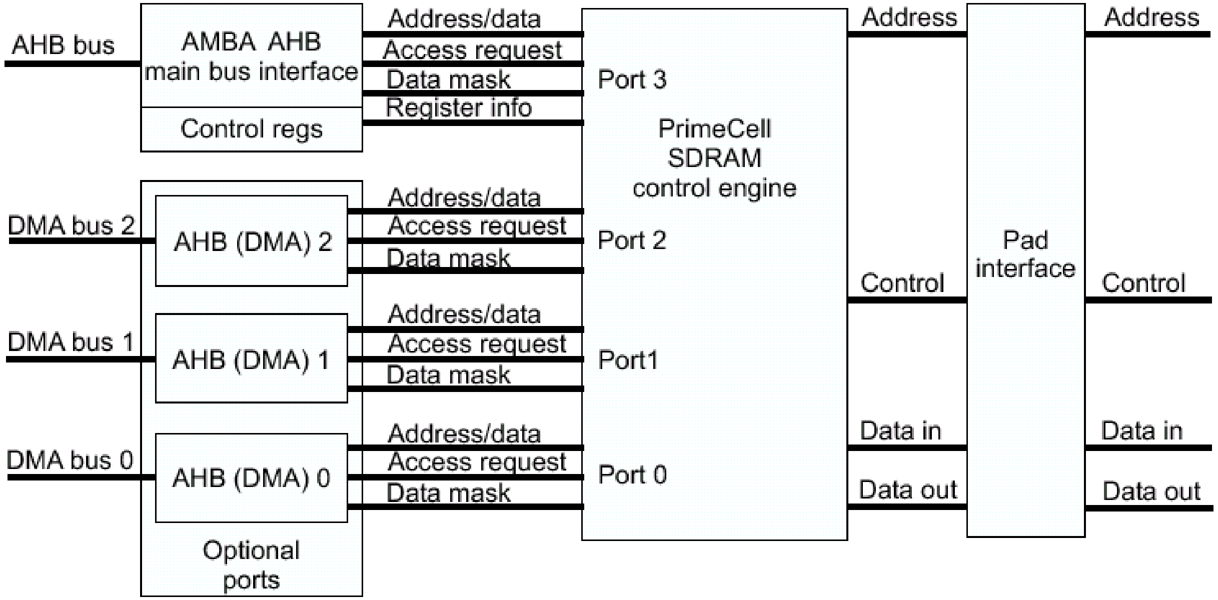
- Technology independence
  - The specification only details the bus protocol at the clock cycle level
- Electrical characteristics
  - No information regarding the electrical characteristics is supplied
- Timing specification
  - The system integrator is given maximum flexibility in allocating the signal timing budget amongst the various modules on the bus
  - More free, but may also be more danger and time-consuming

Notes on AMBA (1/3)

- Split transaction
  - NOT truly split transaction - the arbiter only masks the access of the master which gets a SPLIT transfer response
  - Master does not need extra slave interface
  - Only allows a single outstanding transaction per bus master
- NOT support Sideband signals
  - Sideband signals: reset, interrupts, control/status, generic flags, JTAG test interface, etc.
  - Require the system integrator to deal with them in an ad-hoc way for each system design.
  - Good references of sideband signals: VSIA VCI or Sonics OCP

Notes on AMBA (2/3)

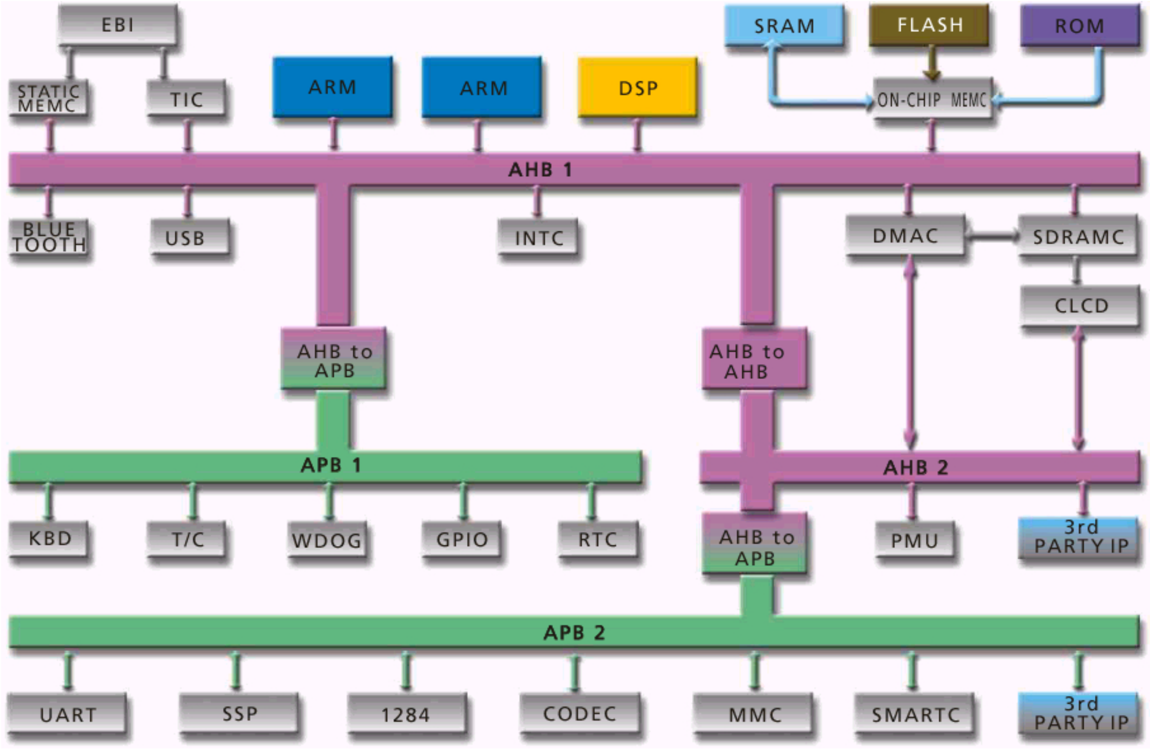
- DMA channels
  - Use AHB protocol
    - E.g. PrimeCell SDRAM Controller
    - Easy to connect to another AHB bus



  - Adopt user defined protocol
    - Lower the complexity of the DMA interface

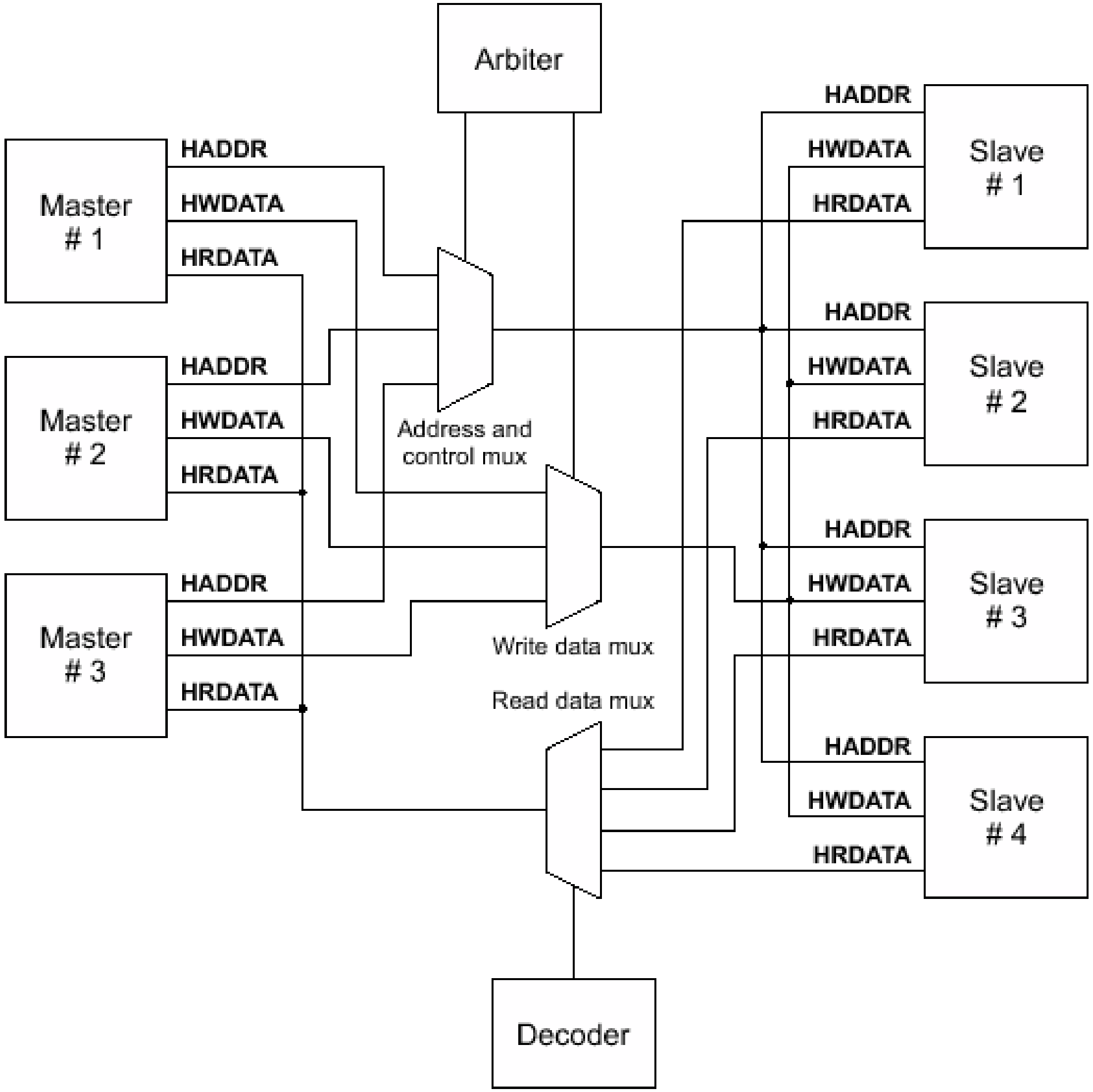
Notes on AMBA (3/3)

- APB does not support WAIT transaction
  - Access status register first, then access data register
  - Alternative: designed as AHB slaves
  - Multiple AHB/APB to reduce loading



Wipro's SOC-RaPtor^{TM} Architecture

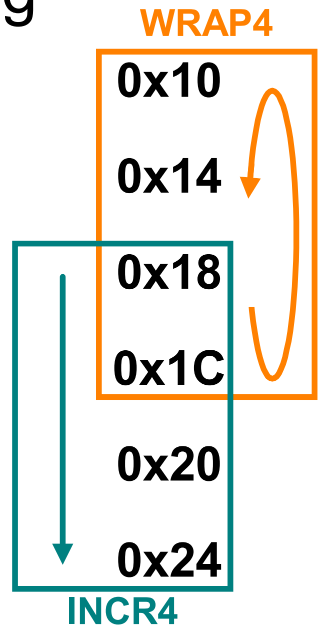
AHB Interconnect



- **Bus master drives the address and control**
- **Arbiter selects one of the master**

AHB Operation (1/2)

- Master asserts a request signal to the arbiter. Arbiter then gives the grant to the master.
- A granted bus master starts an AHB transfer by driving address and control signals:
  - address
  - direction
  - width
  - burst forms
    - Incrementing burst: not wrap at address boundaries
    - Wrapping burst: wrap at particular address boundaries
- Write data bus: move data from the master to a slave
- Read data bus: move data from a slave to the master



Address wrap in 4-word boundary

AHB Operation (2/2)

- All slaves sample the address

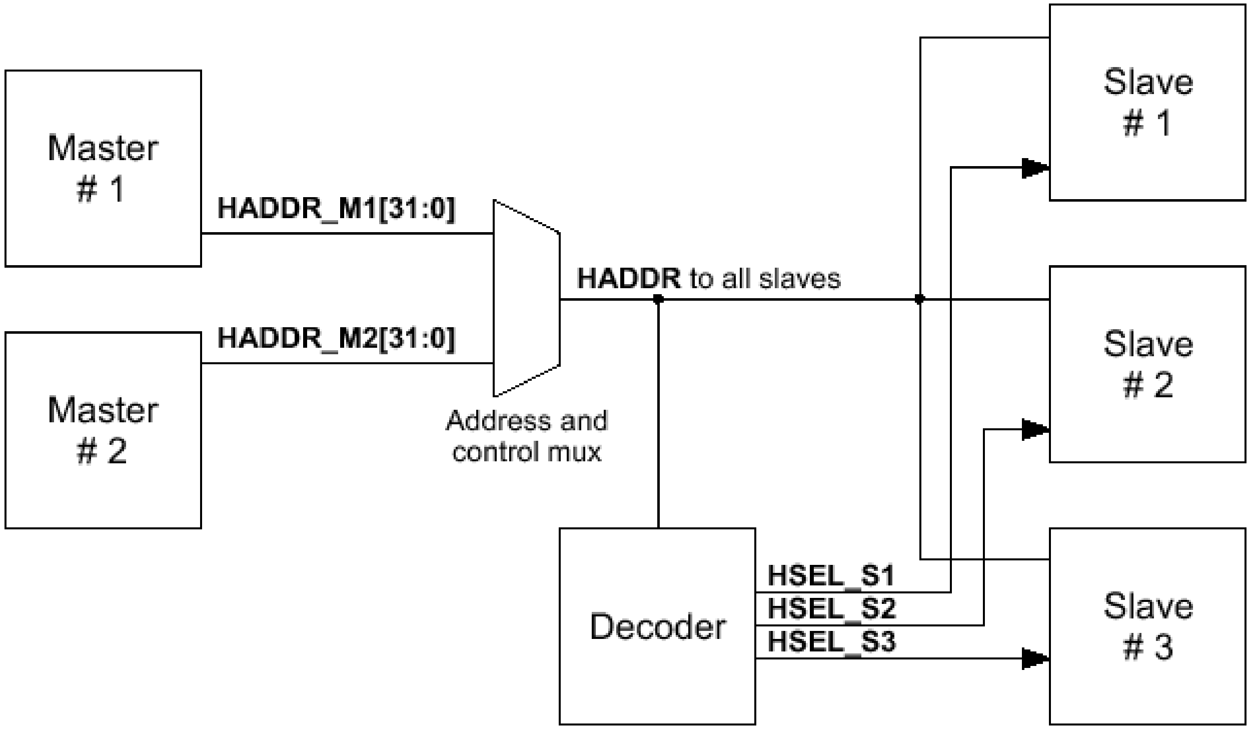
  Data can be extended using the HREADY signal, when LOW, wait states be inserted and allow extra time for the slave to provide or sample data
- During a transfer the slave shows the status using the response signals HRESP[1:0]
  - OKAY: transfer progressing normally

    when HREADY is HIGH, transfer has completed successfully
  - ERROR: transfer error
  - RETRY and SPLIT: transfer can't complete immediately, but the bus master should continue to attempt the transfer
- As burst transfer, the arbiter may break up a burst and in such cases the master must re-request for the bus.

Institute of Electronics, National Chiao Tung University

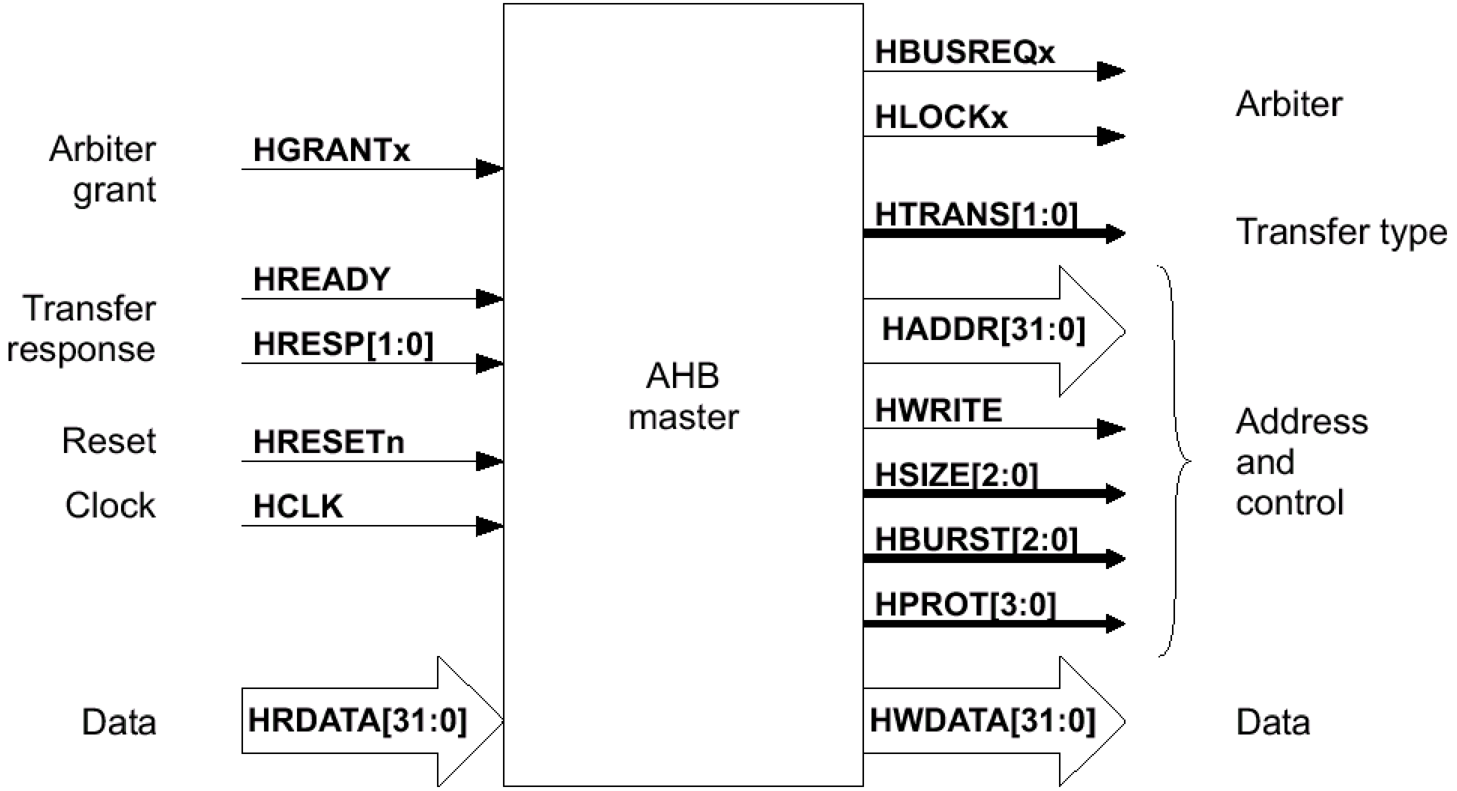
Address Decoding

- A central address decoder provides HSELx for each slave
- Minimum address space that can be allocated to a single slave is 1K Byte
  - No incrementing transfers can over a 1K Byte boundary



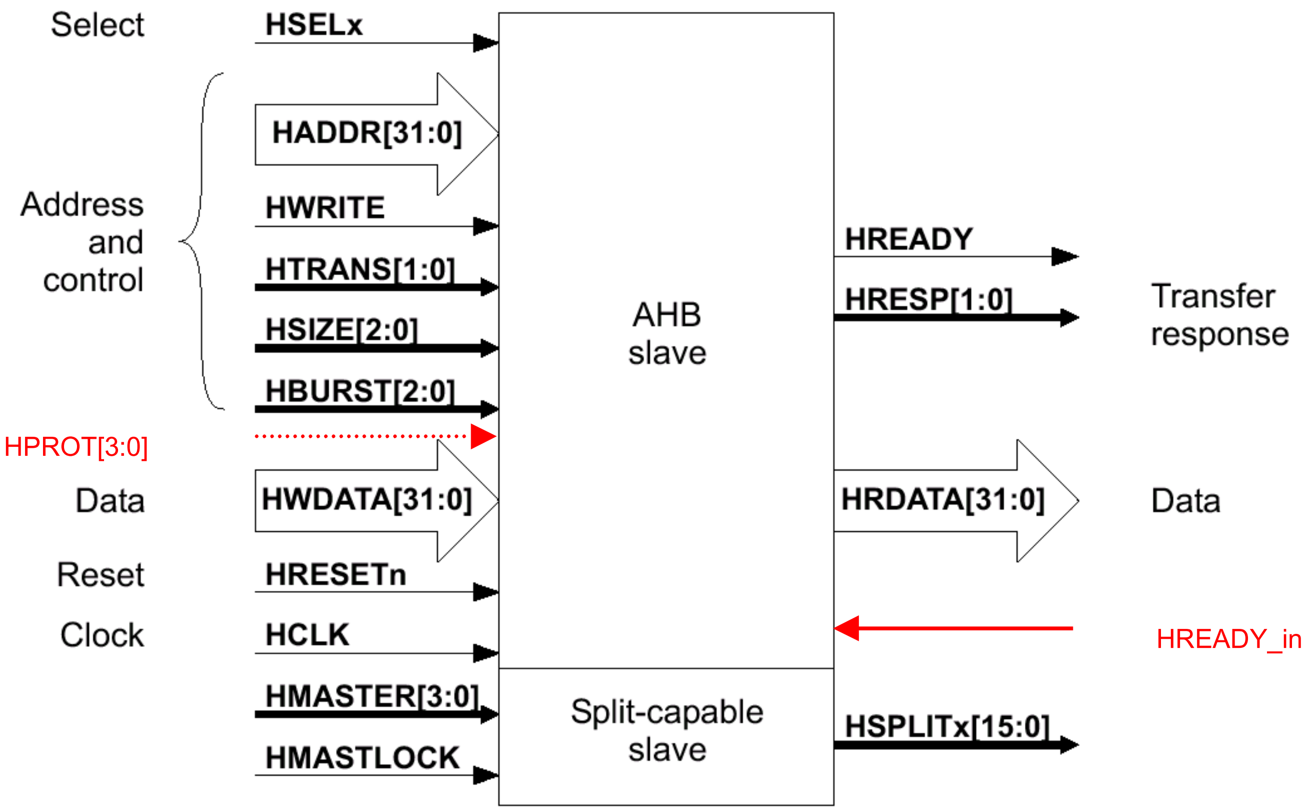
AHB Master

- Initiate read and write by providing an address and control interface
- Processor, DMA, DSP test interface

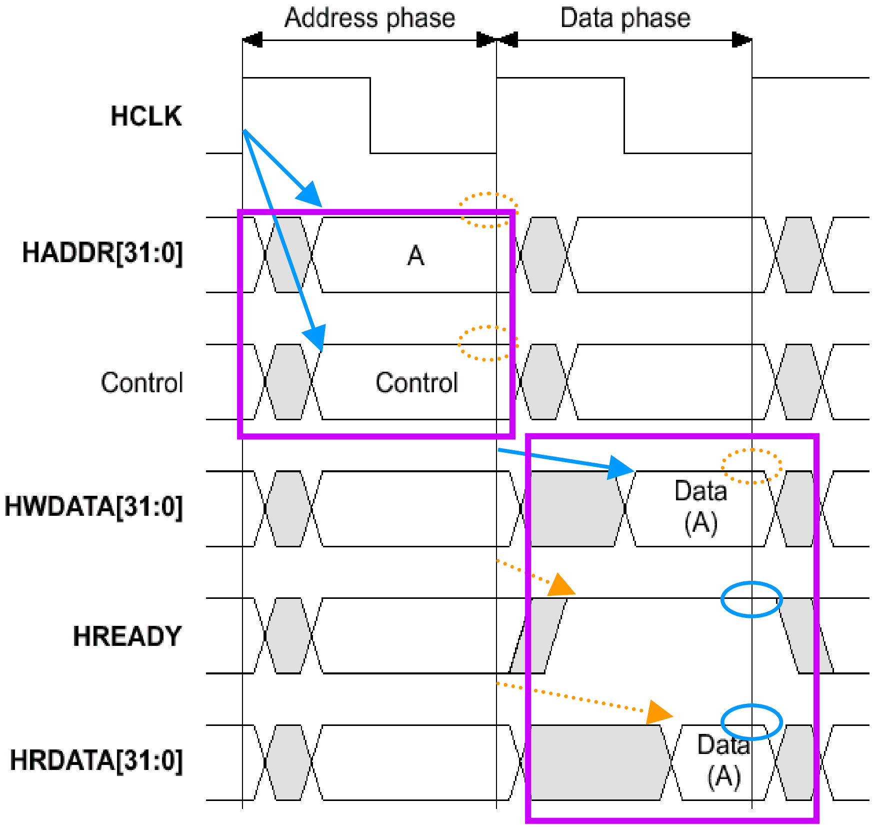


AHB Slave

- Respond to a read or write operation within a given address-space range
- Back to the master the success, failure or waiting



Basic Transfer

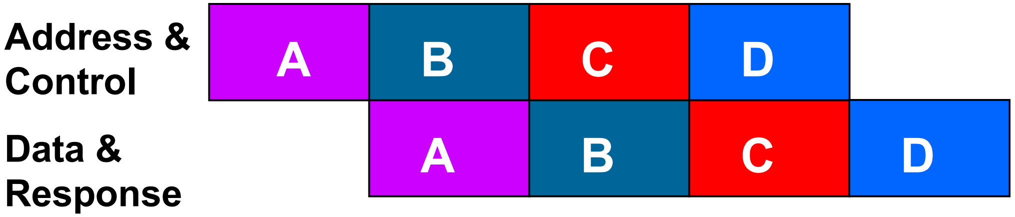


— Master action

⋯ Slave action

→ Drive action

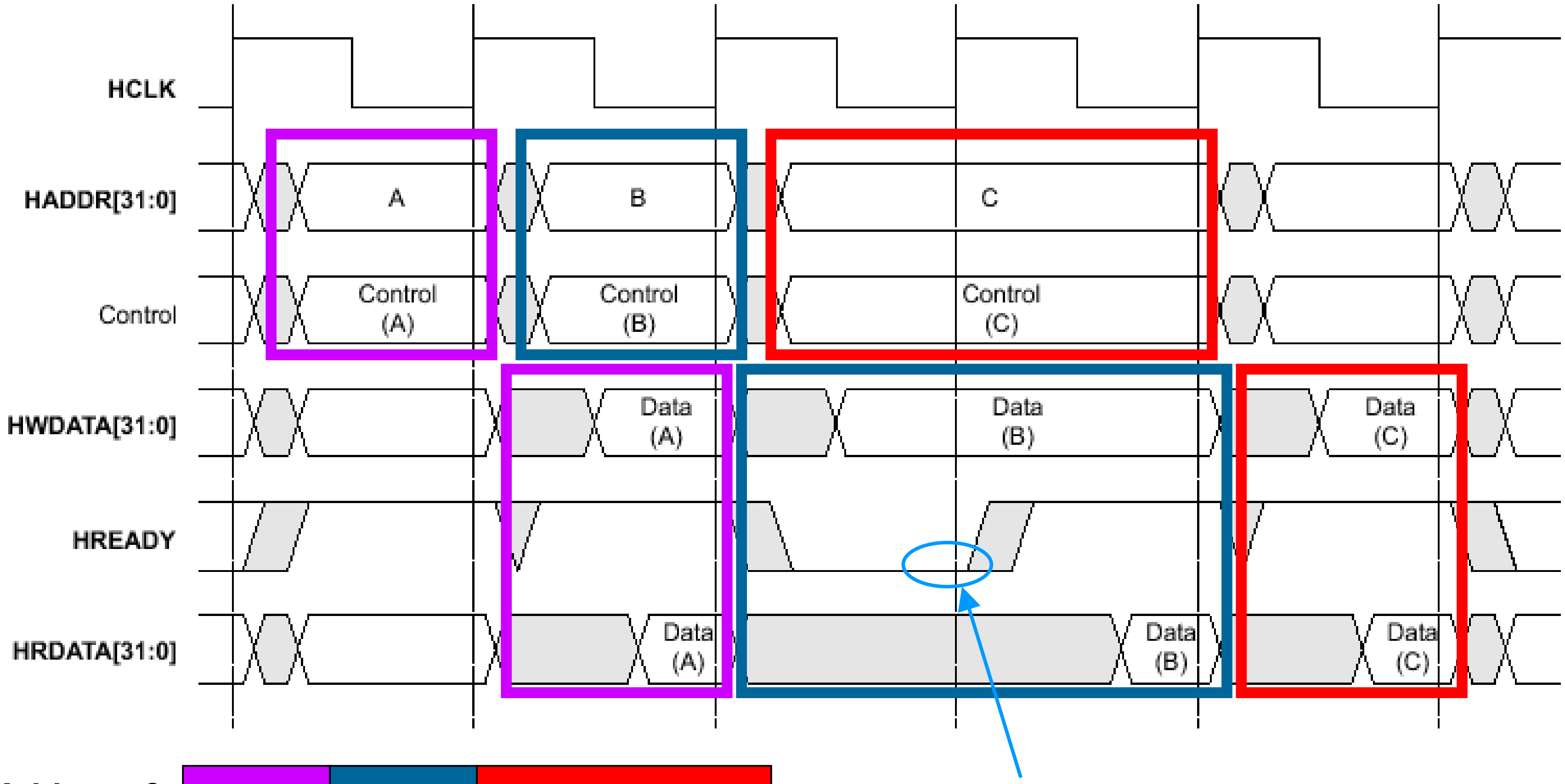
○ Sample action



- Address phase : one cycle
  Data phase : one or several cycles
- 1st clock : master drives address and control
- 2nd clock : slave samples address and control
- 3rd clock : bus master sample the slave’s response

Multiple Transfers

- Three transfers to run related address A, B, and C



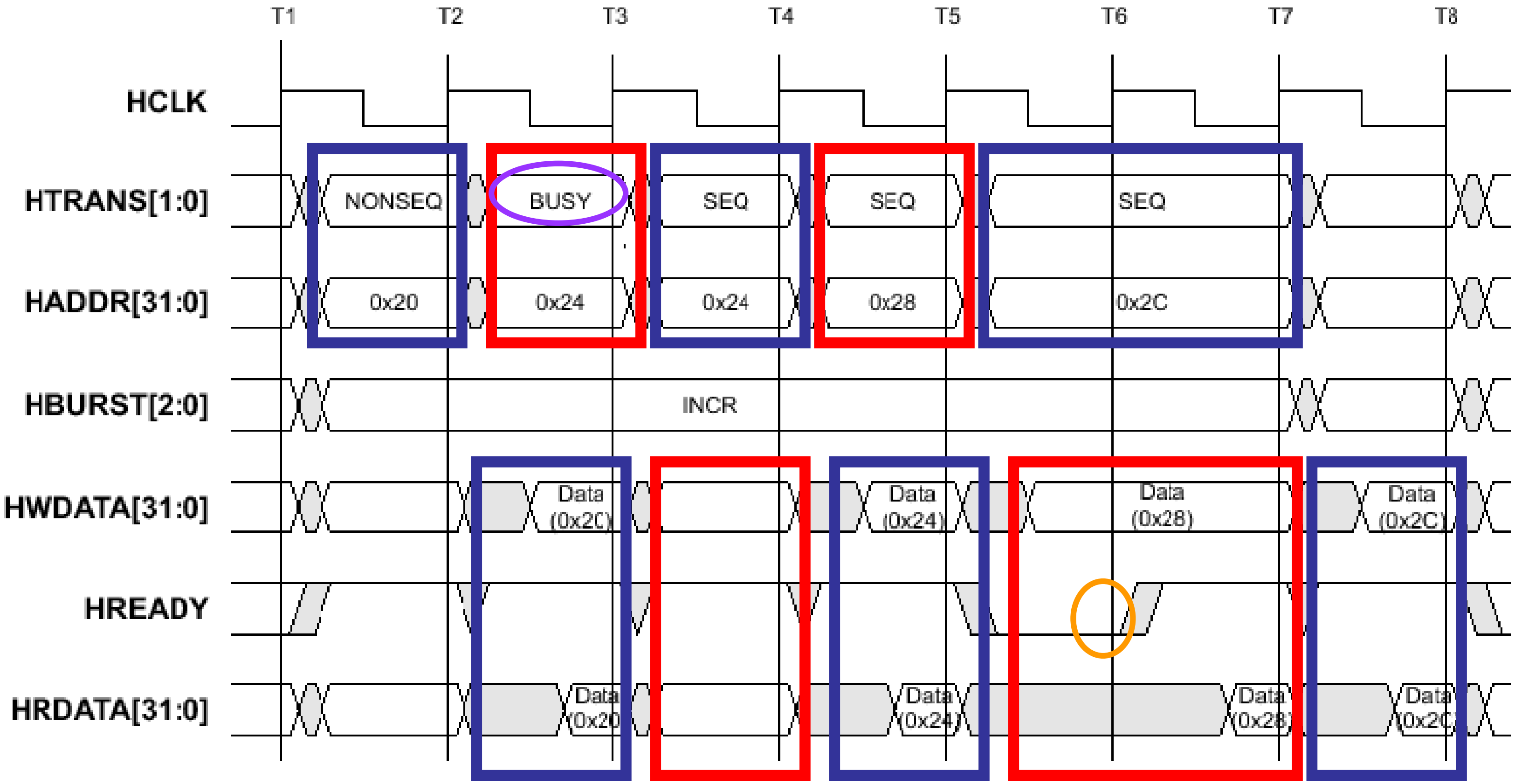
1 wait cycle for address B

Address & control	A	B	C	
Data & Response		A	B	C

Transfer Type (1/2)

HTRANS[1:0]	Type	Descripton
00	IDLE	Slaves must always provide a zero wait state OKAY response to IDLE transfers and the transfer should be ignored by the slave
01	BUSY	Masters cannot take next trnsfer place immediately during a burst transfer. Slaves take actions as they take for IDLE.
10	NONSEQ	Indicates the first transfer of a burst or a single transfer
11	SEQ	The remaining transfers in a burst are SEQUENTIAL. The control information is identical to the previous transfer.

Transfer Type (2/2)



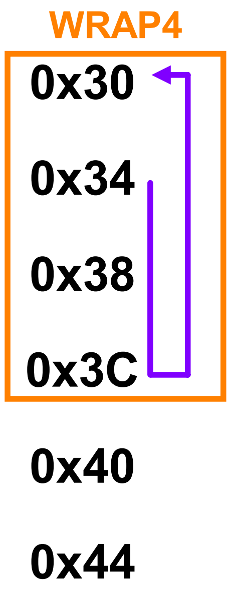
- During T2-T3, **master** is unable to perform the second transfer of burst immediately and therefore the master uses **BUSY** transfer to delay the start of the next transfer.
- During T5-T6, **slave** is unable to complete access immediately, and uses **HREADY** to insert a single wait state.

Burst Operation

- 4-, 8-, 16-beat
- e.g., 4-beat, start address 0x34, wrapping burst
  ➔four transfers: 0x34, 0x38, 0x3C, 0x30
- Burst length

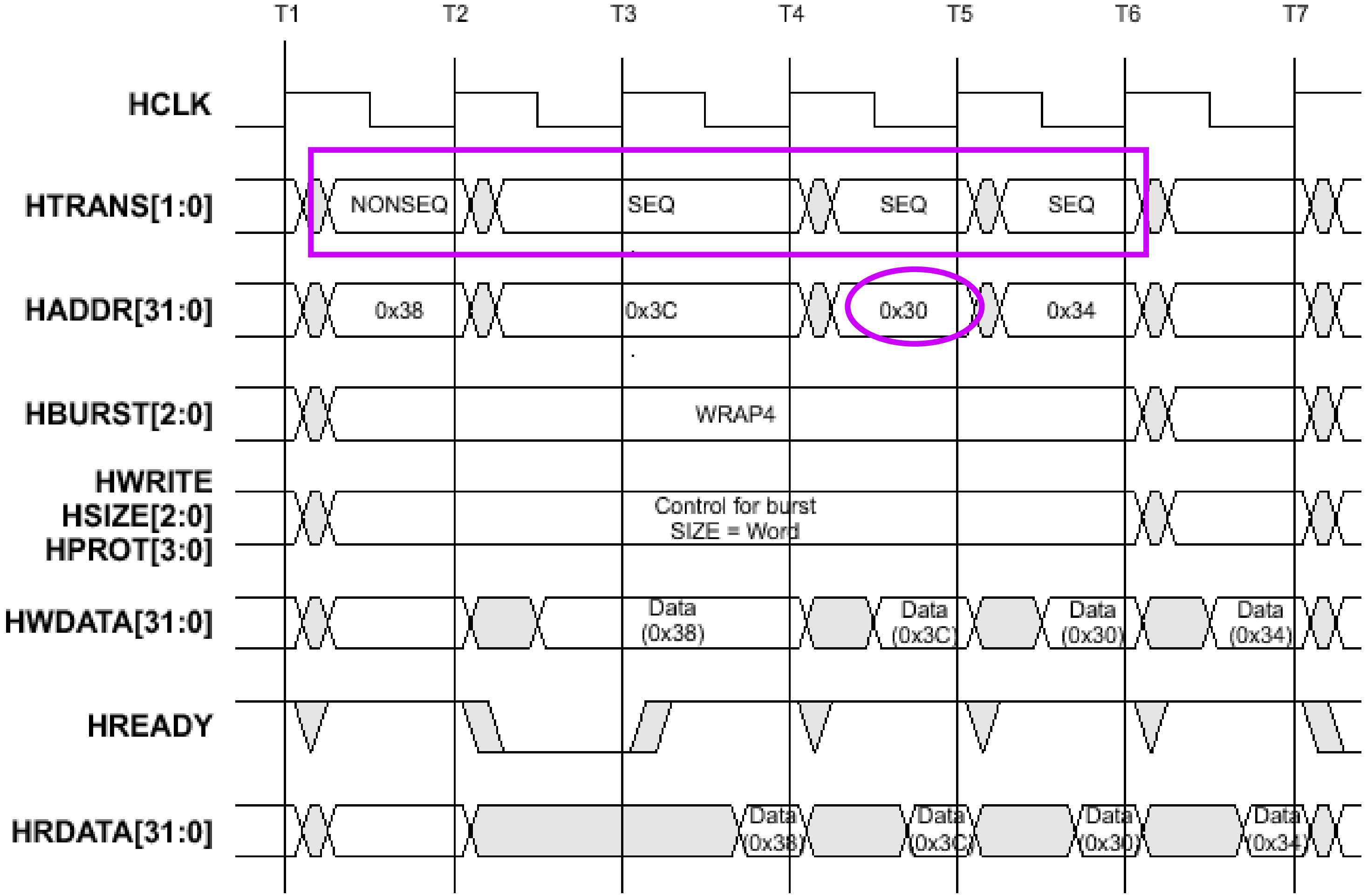
HBURST[1:0]	Type	Descripton
000	SINGLE	Single Transfer
001	INCR	Incrementing burst of unspecified length
010	WRAP4	4-beat wrapping burst
011	INCR4	4-beat incrementing burst
100	WRAP8	8-beat wrapping burst
101	INCR8	8-beat incrementing burst
110	WRAP16	16-beat wrapping burst
111	INCR16	16-beat incrementing burst

- Limitation: bursts must not cross a 1k Byte address boundary



Institute of Electronics, National Chiao Tung University

Four-beat Wrapping Burst

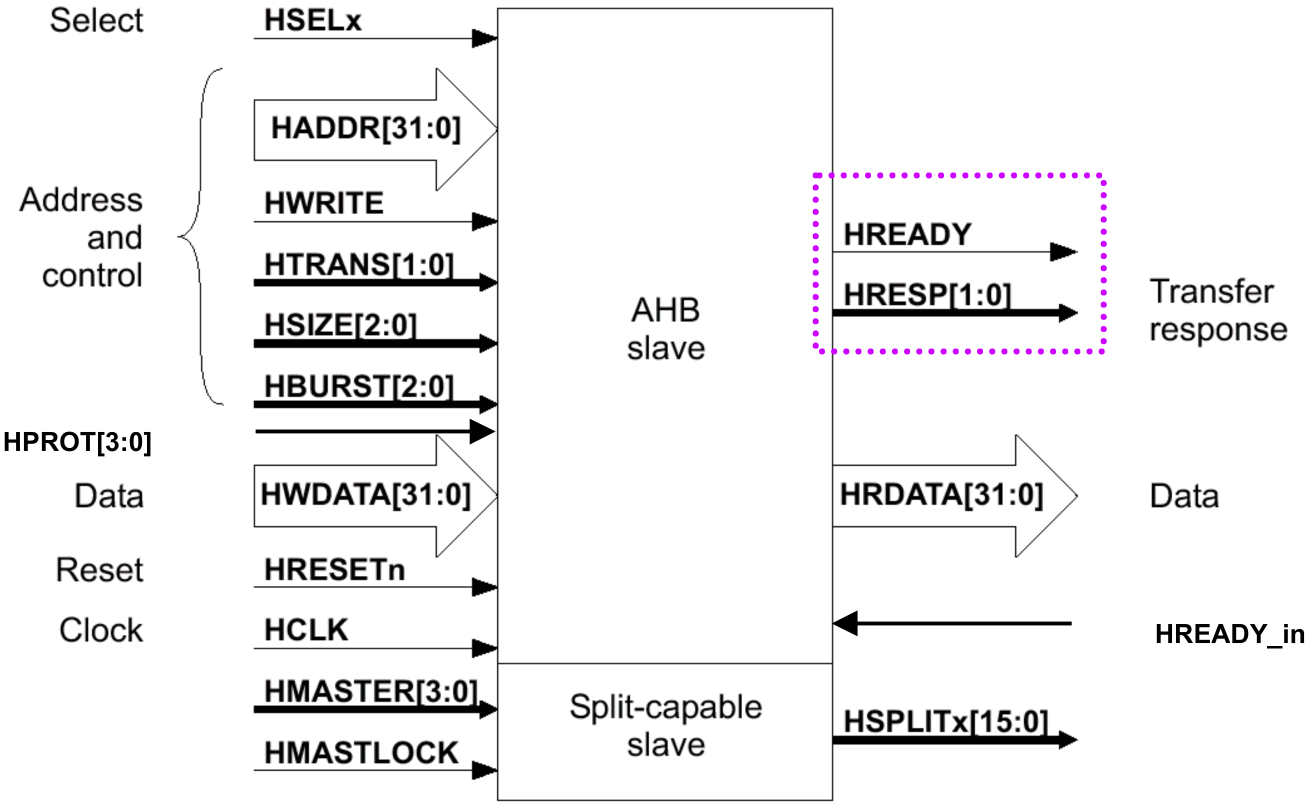


Control Signals

- Have exactly the same timing as the address bus
- Must remain constant throughout a burst of transfers
- Types
  - HWRITE : Transfer direction
  - HSIZE[2:0] : Transfer size
  - HPROT[3:0]) : Protection control

  indicate if the transfer is:
    - An opcode fetch or data access
    - A privileged mode access or user mode access
    - Access is cacheable or bufferable (for bus masters with a memory management unit)

Transfer Responses (from slave)



Transfer Responses

- HREADY
- HRESP[1:0] Response

HRESP[1:0]	Response
00	OKAY
01	ERROR
10	RETRY
11	SPLIT

- Two-cycle response
  - ERROR & RETRY & SPLIT
  - To complete current transfer, master can take following action

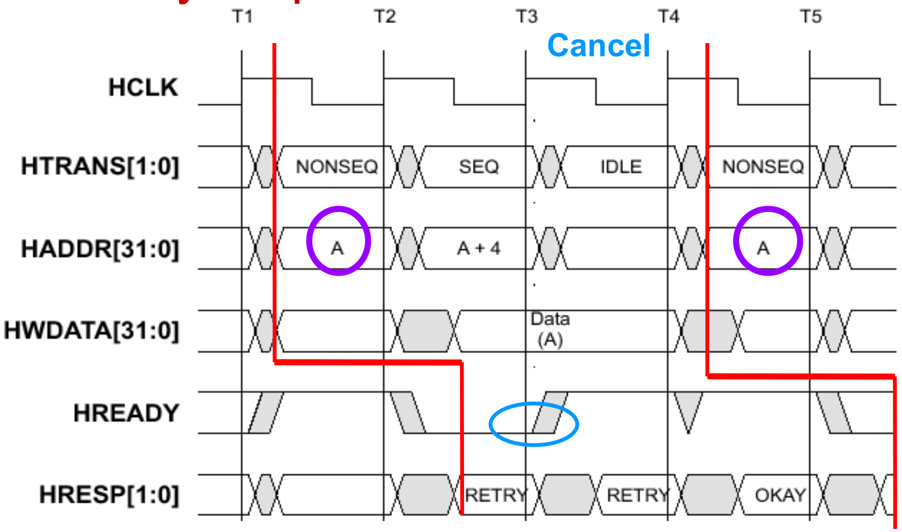
    Cancel for RETRY

    Cancel for SPLIT

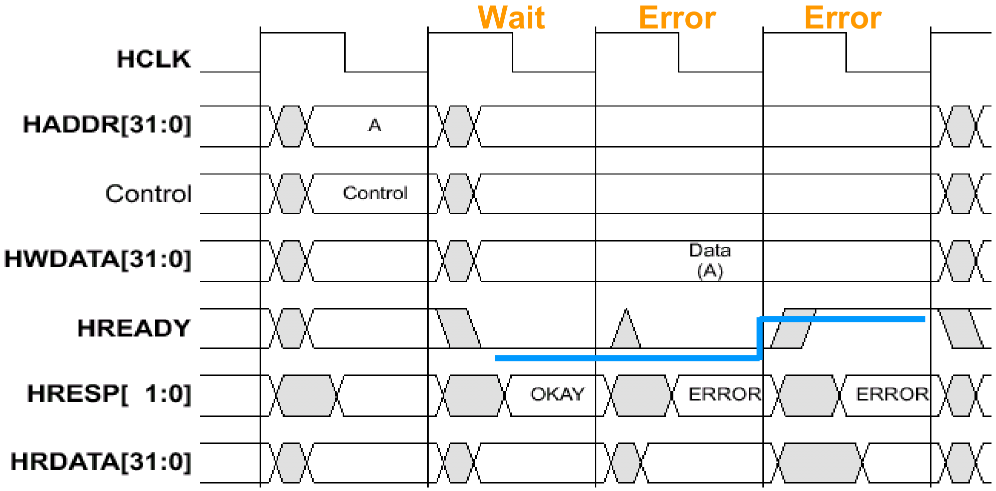
    Either cancel or continue for ERROE

Examples of Two-cycle Response

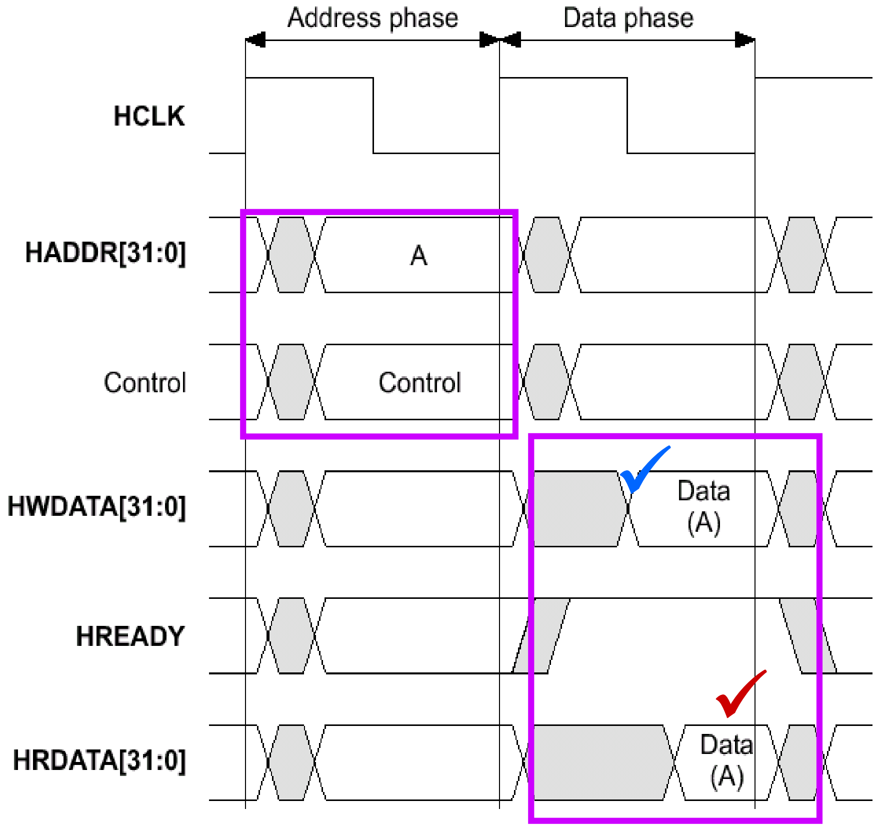
- Retry response



- Error response

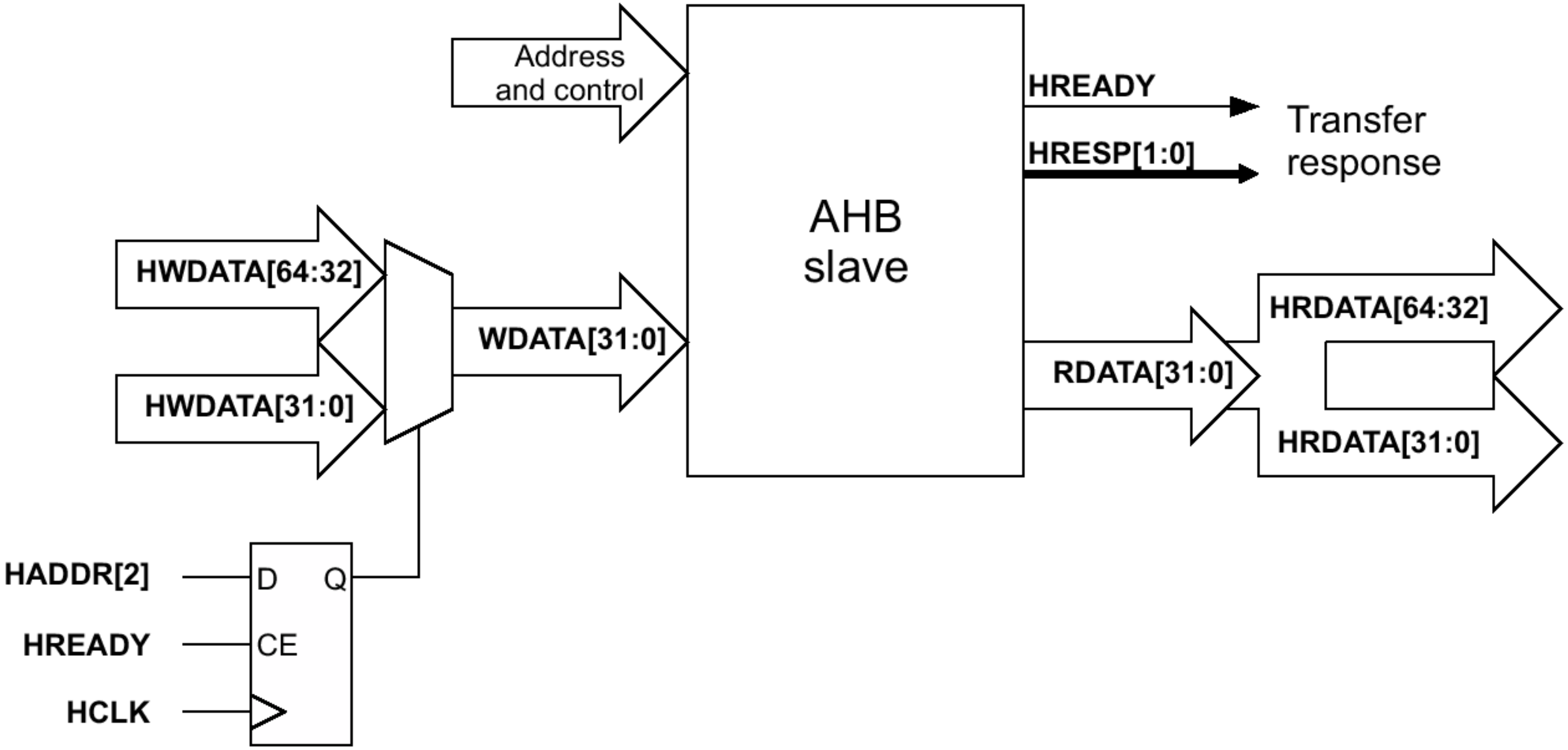


Data Buses

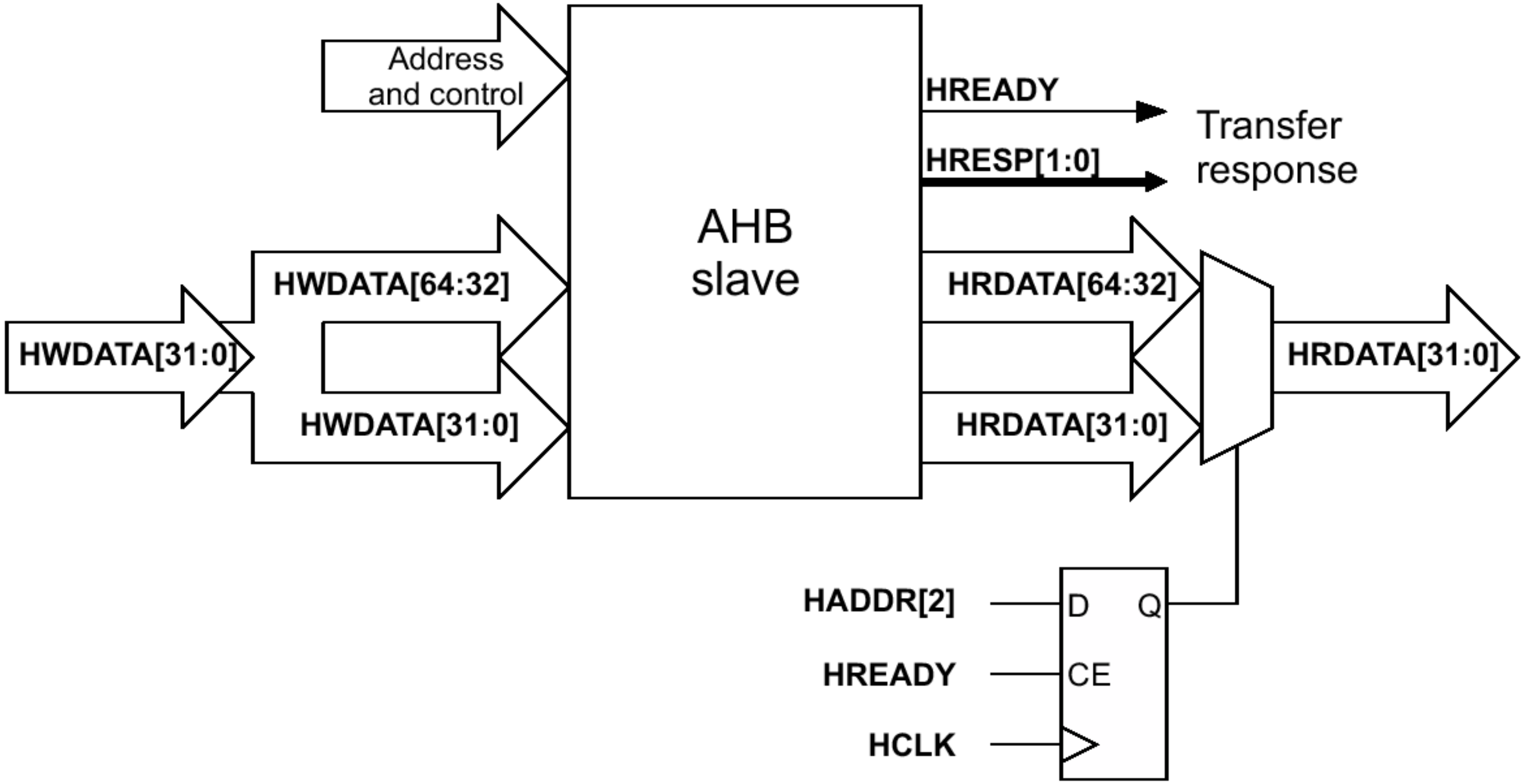


- HWDAA : 32 bits
- RDATA : 32 bits
- Endianness : fixed, lower power; higher performance

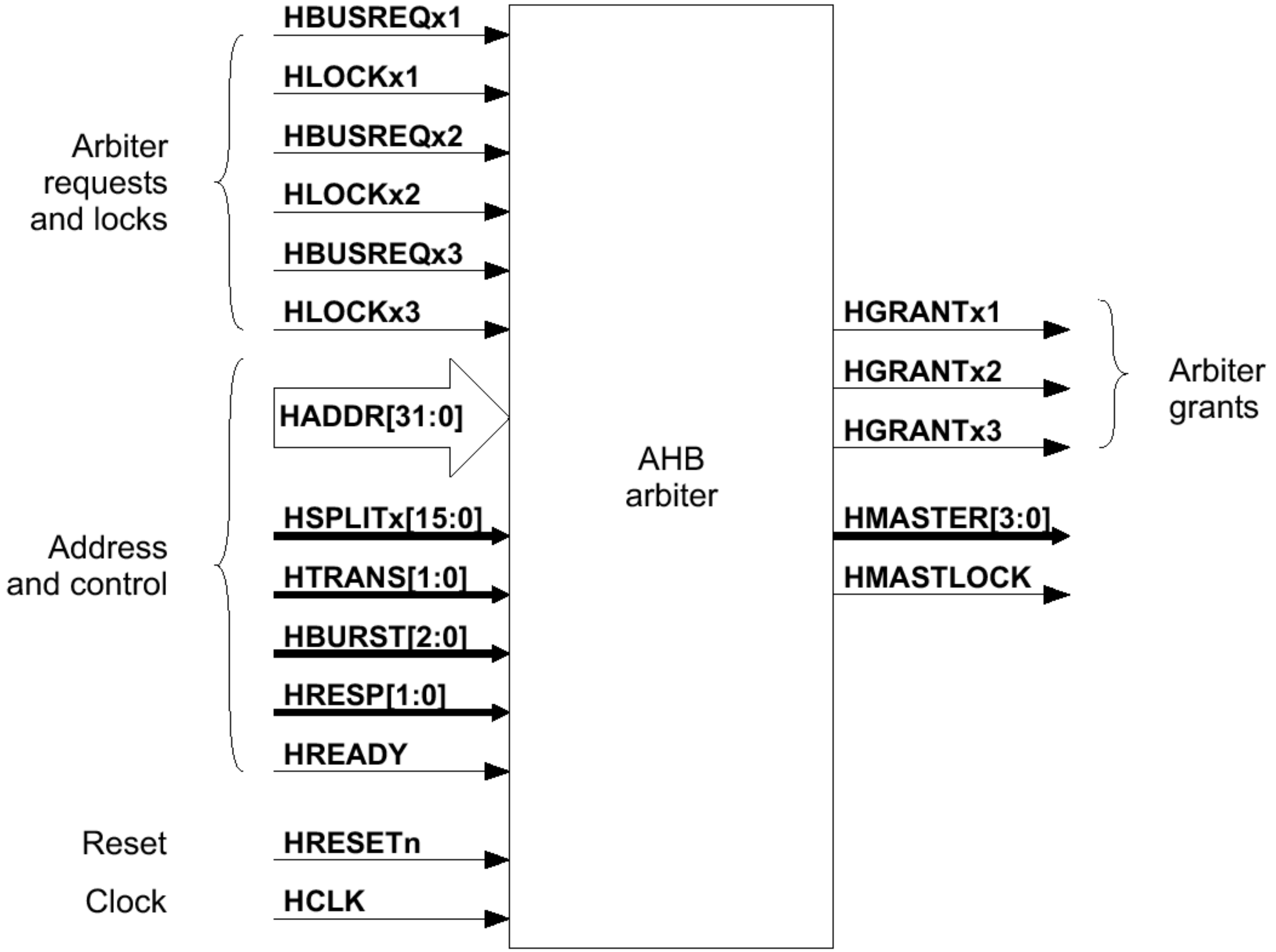
Narrow Slave on A Wide Bus



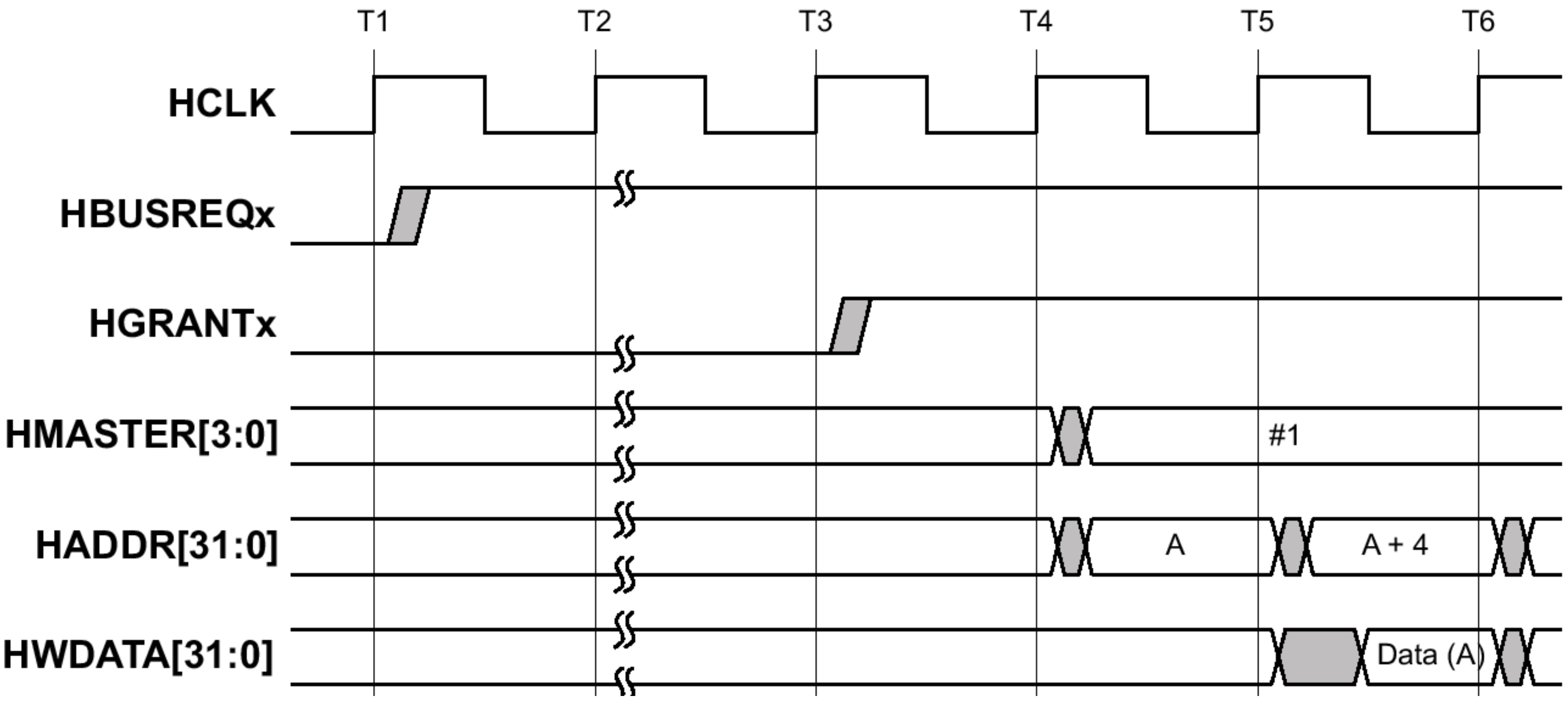
Wide Slave on A Narrow Bus



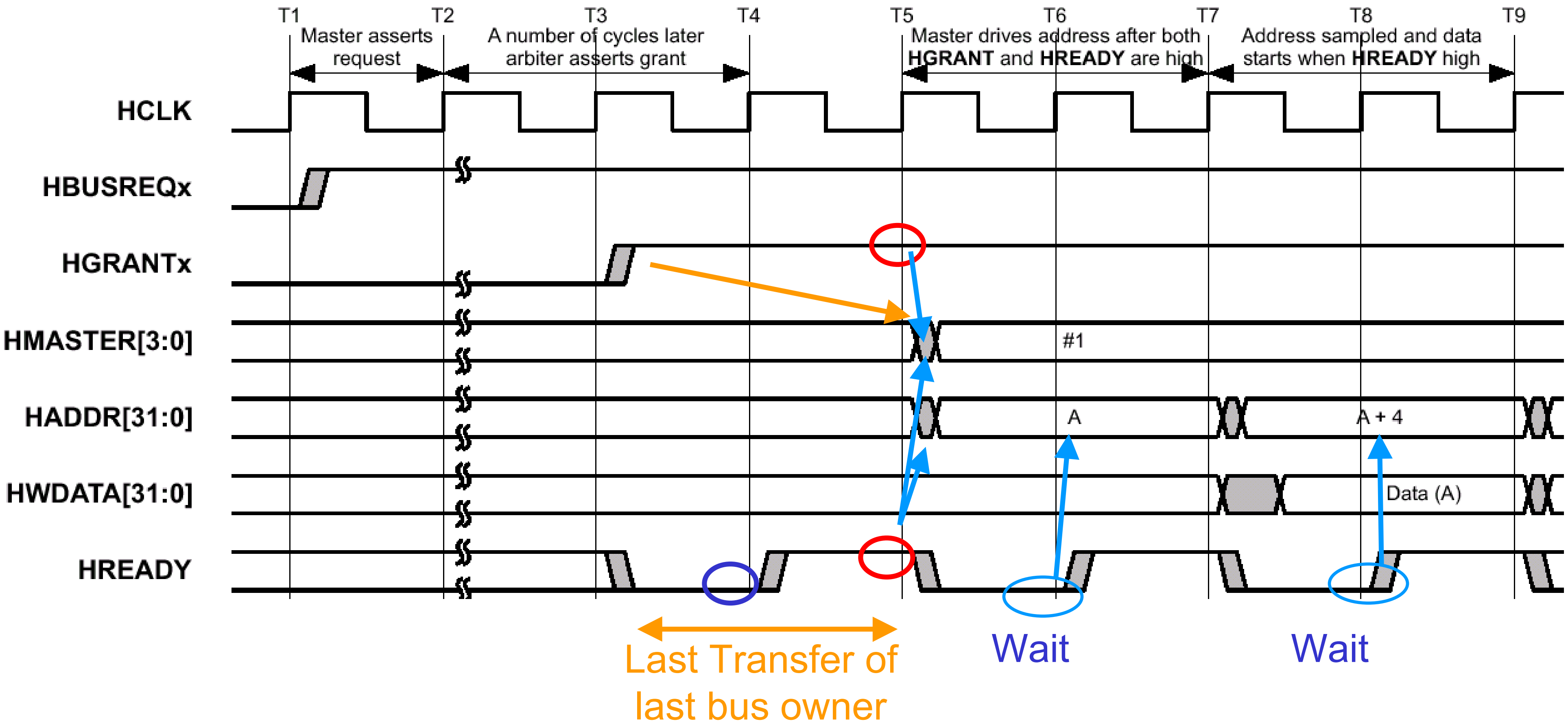
Arbitration



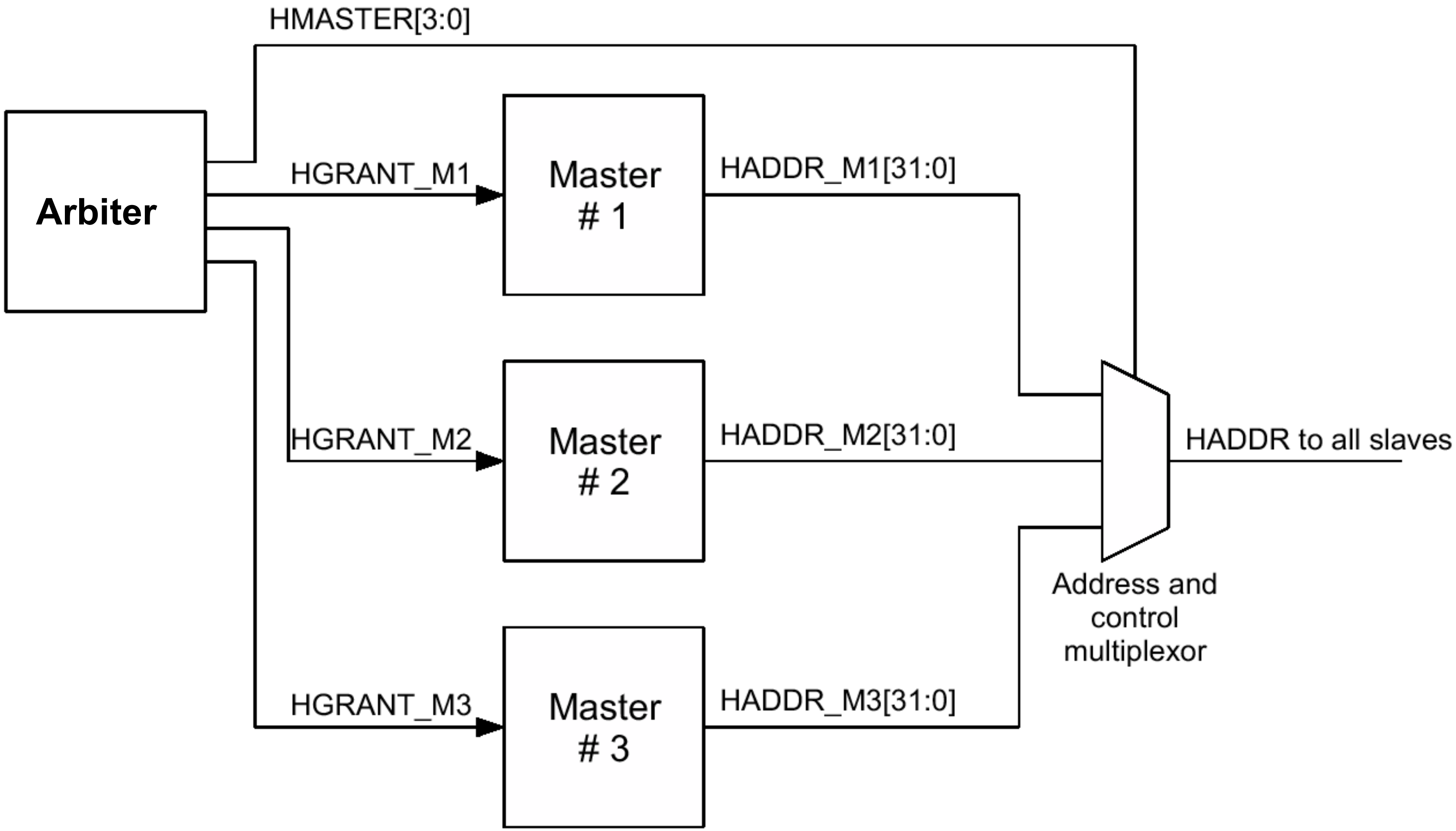
Granting Bus Access With No Wait States



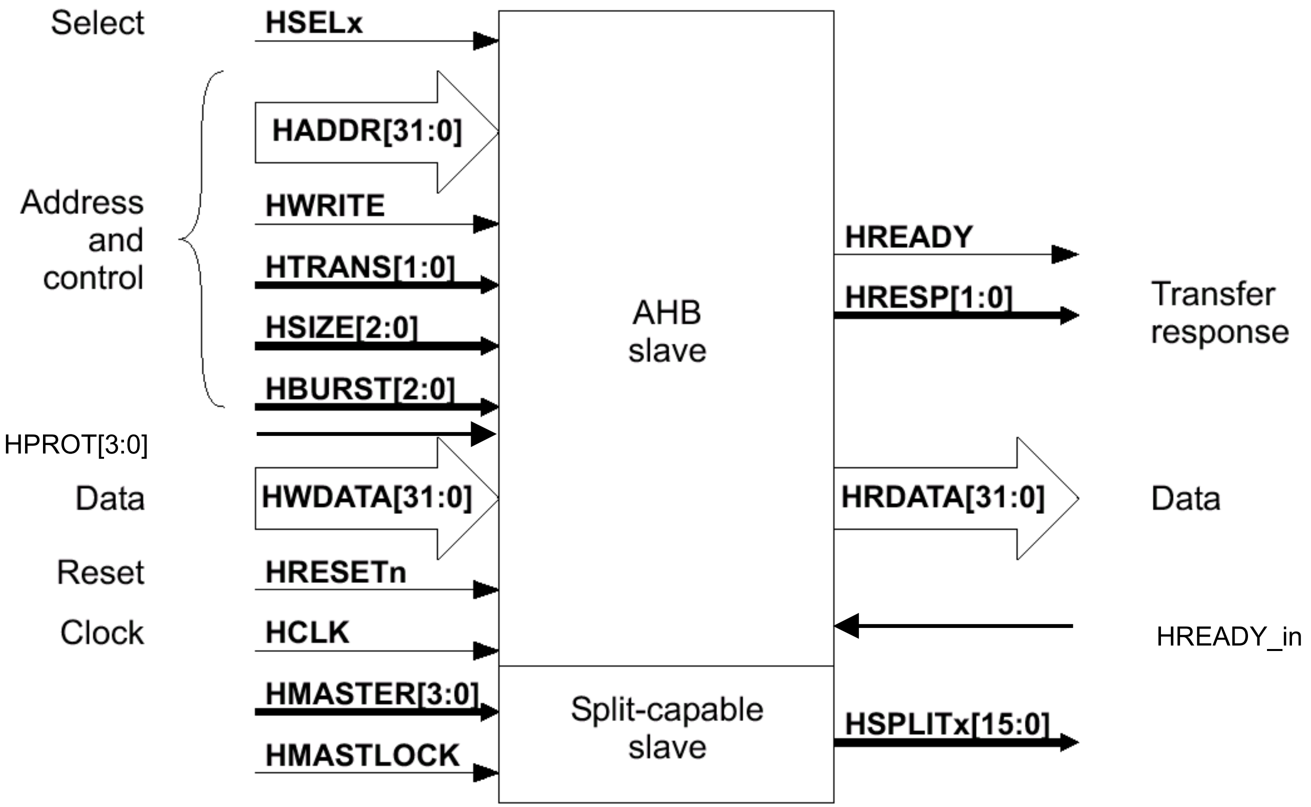
Granting Bus Access


T1
T2
T3
T4
T5
T6
T7
T8
T9
Master asserts request
A number of cycles later arbiter asserts grant
Master drives address after both HGRANT and HREADY are high
Address sampled and data starts when HREADY high
HCLK
HBUSREQx
HGRANTx
HMASTER[3:0]
#1
HADDR[31:0]
A
A + 4
HWDATA[31:0]
Data (A)
HREADY
Last Transfer of last bus owner
Wait
Wait

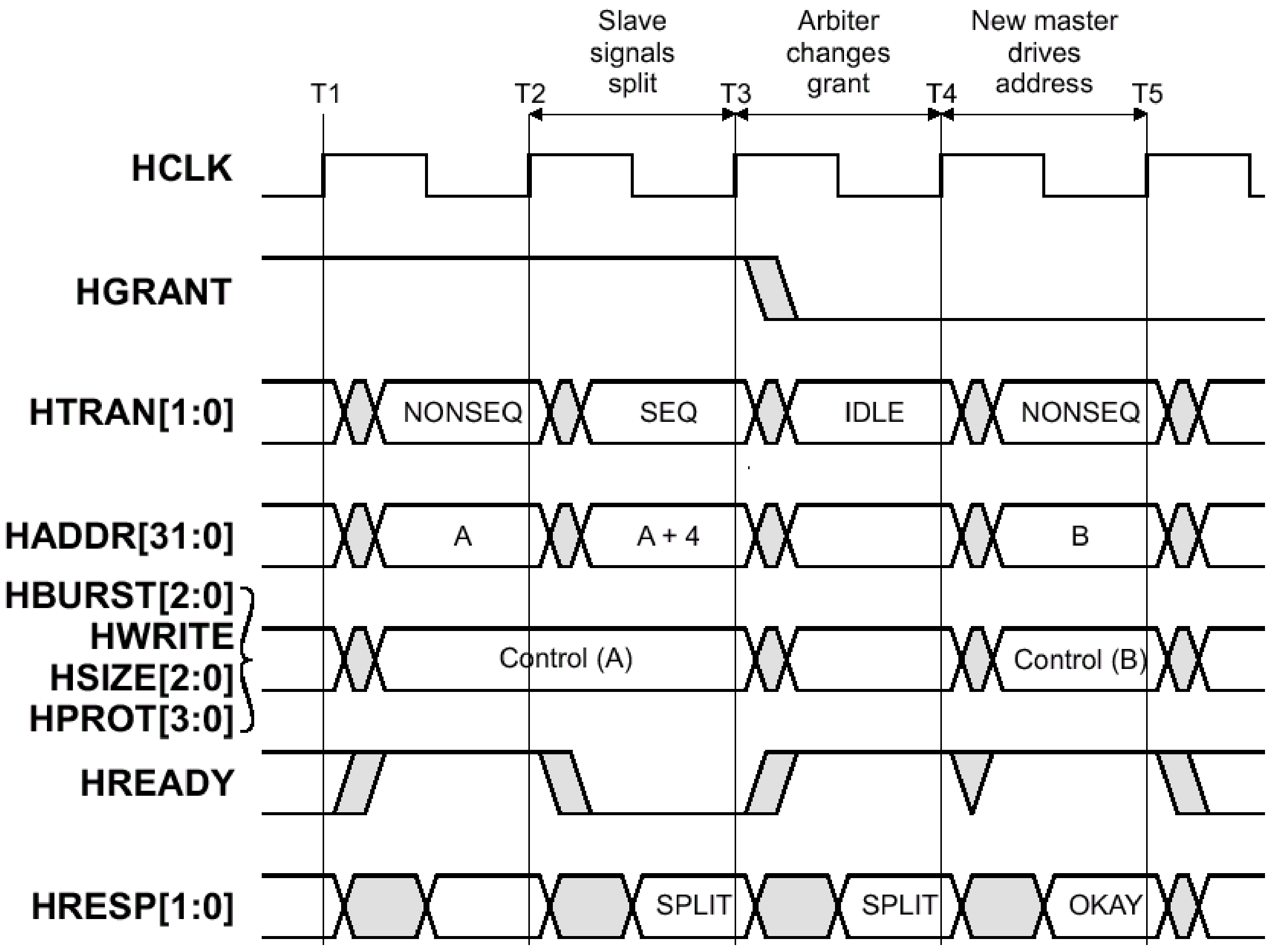
Bus Master Grant Signals



Split Transfer



Split Transfer

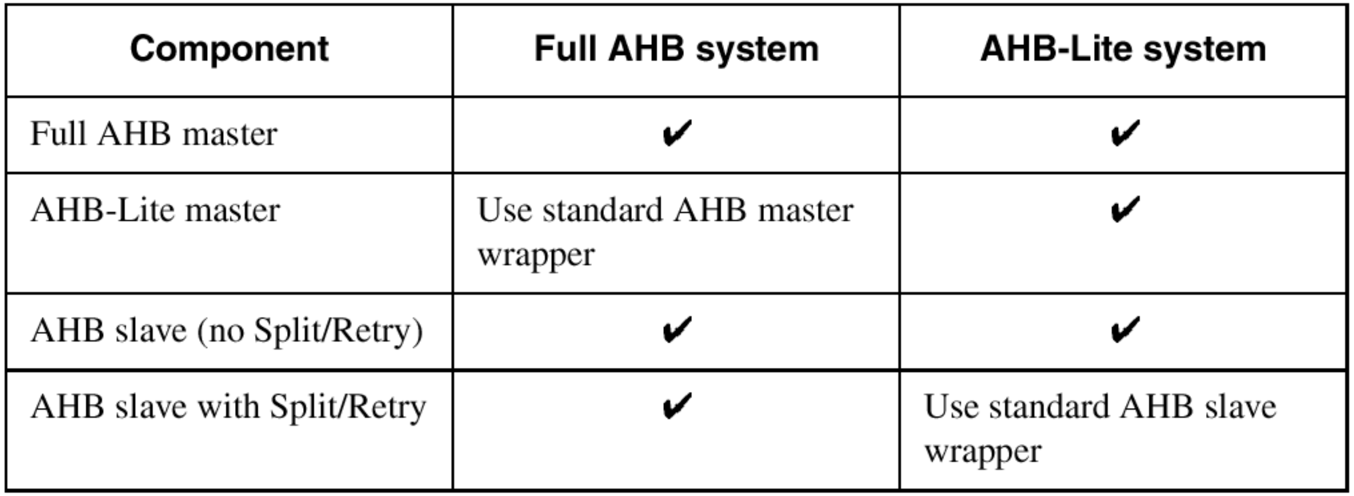


Institute of Electronics, National Chiao Tung University

AHB-Lite

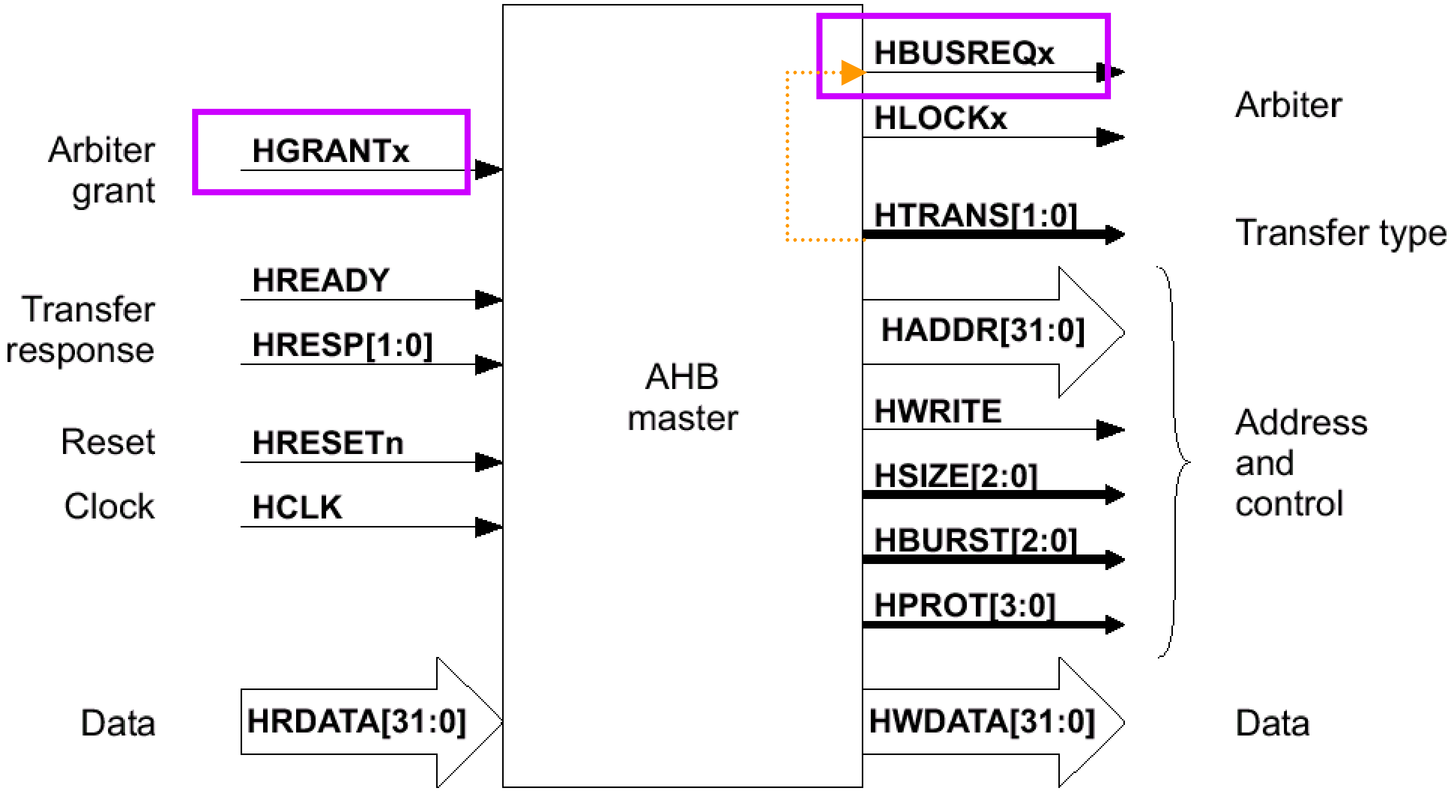
- Requirement
  - Only one master
  - Slave must not issue Split or Retry response
- Subset of AHB Functionality
  - Master: no arbitration or Split/Retry handling
  - Slave: no Split or Retry responses
- Standard AHB masters can be used with AHB-Lite
- Advantage
  - Master does not have to support: the following cases:
    - Losing bus ownership
    - Early bus termination
    - Split and Retry response
  - No arbiter
  - No Master-to-slave mux
  - Allows easier module design/debug

AHB-Lite Interchangeability

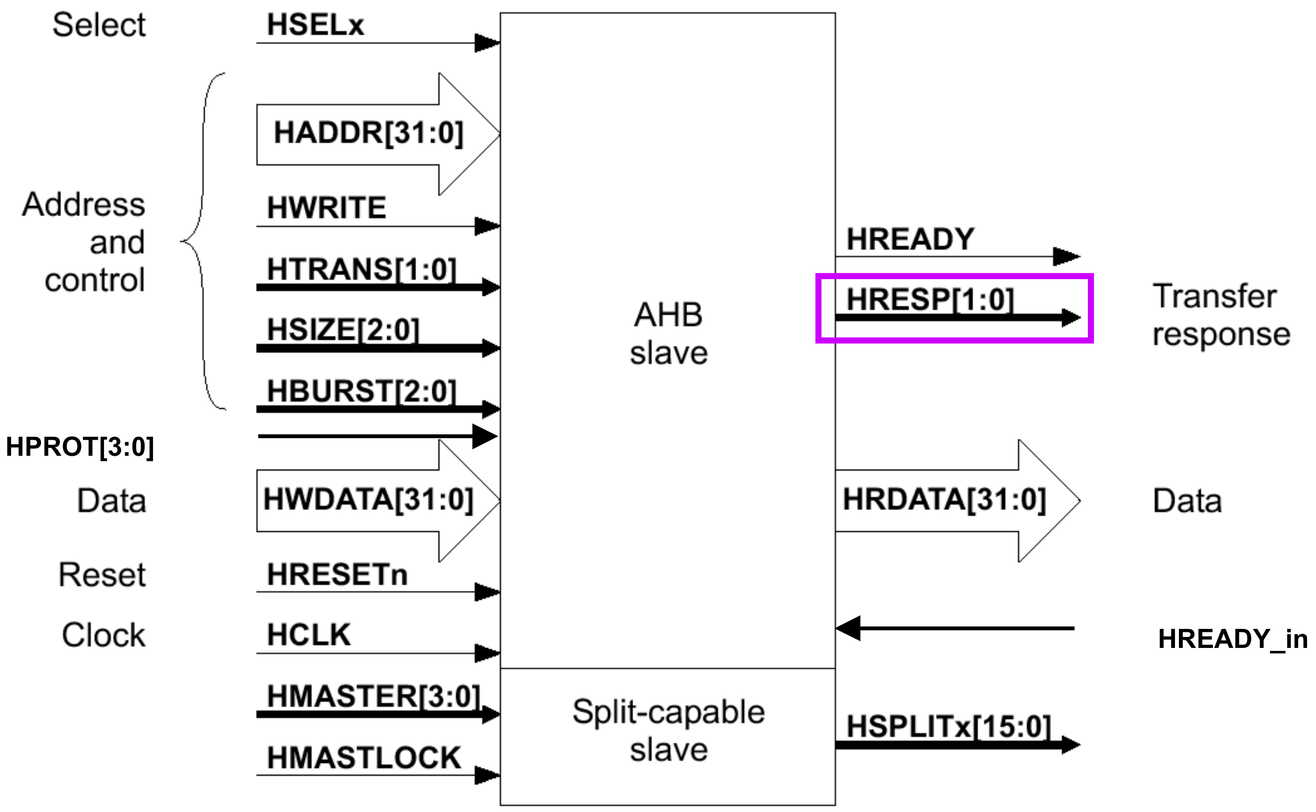


Component	Full AHB system	AHB-Lite system
Full AHB master	✔	✔
AHB-Lite master	Use standard AHB master wrapper	✔
AHB slave (no Split/Retry)	✔	✔
AHB slave with Split/Retry	✔	Use standard AHB slave wrapper

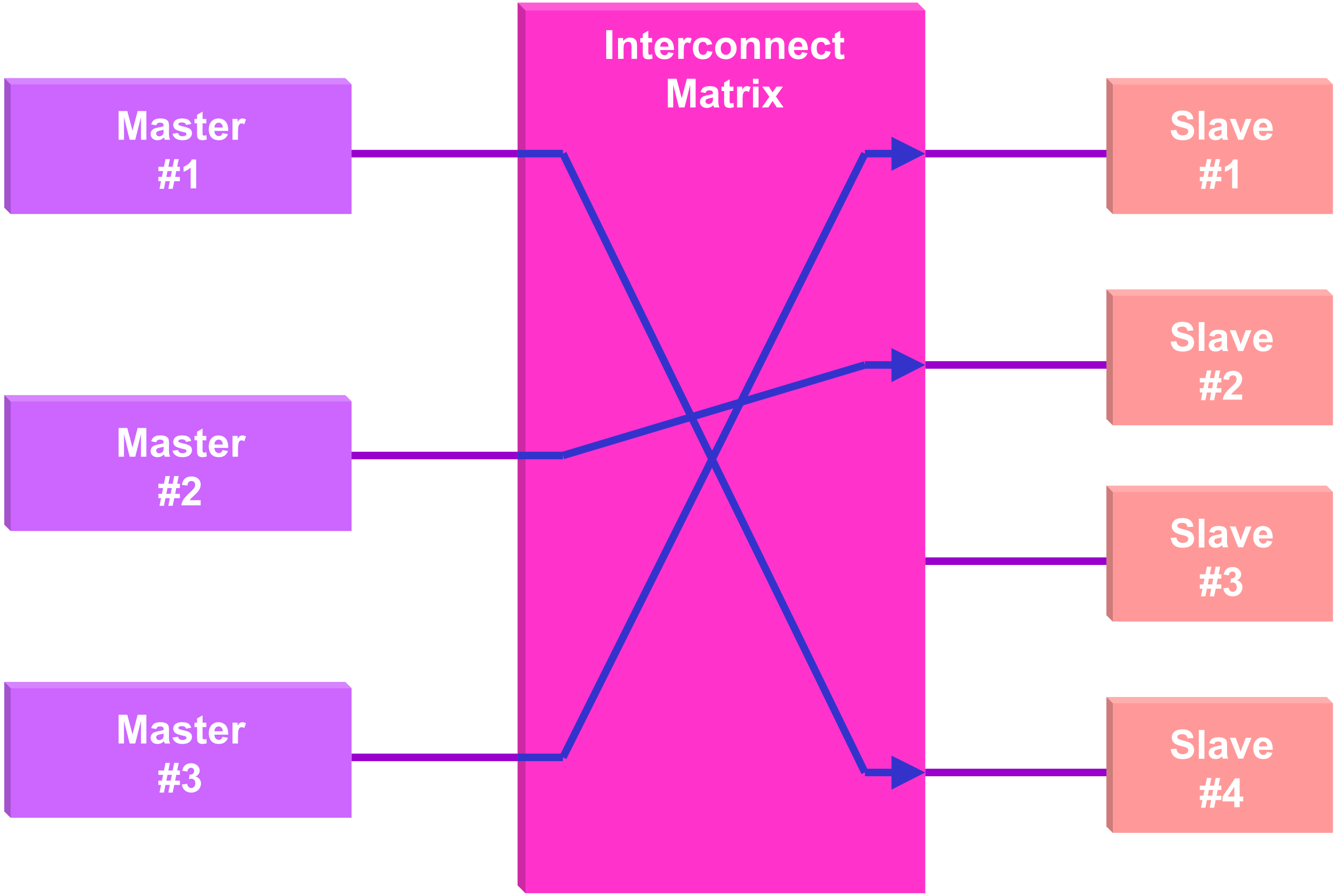
AHB-Lite Master



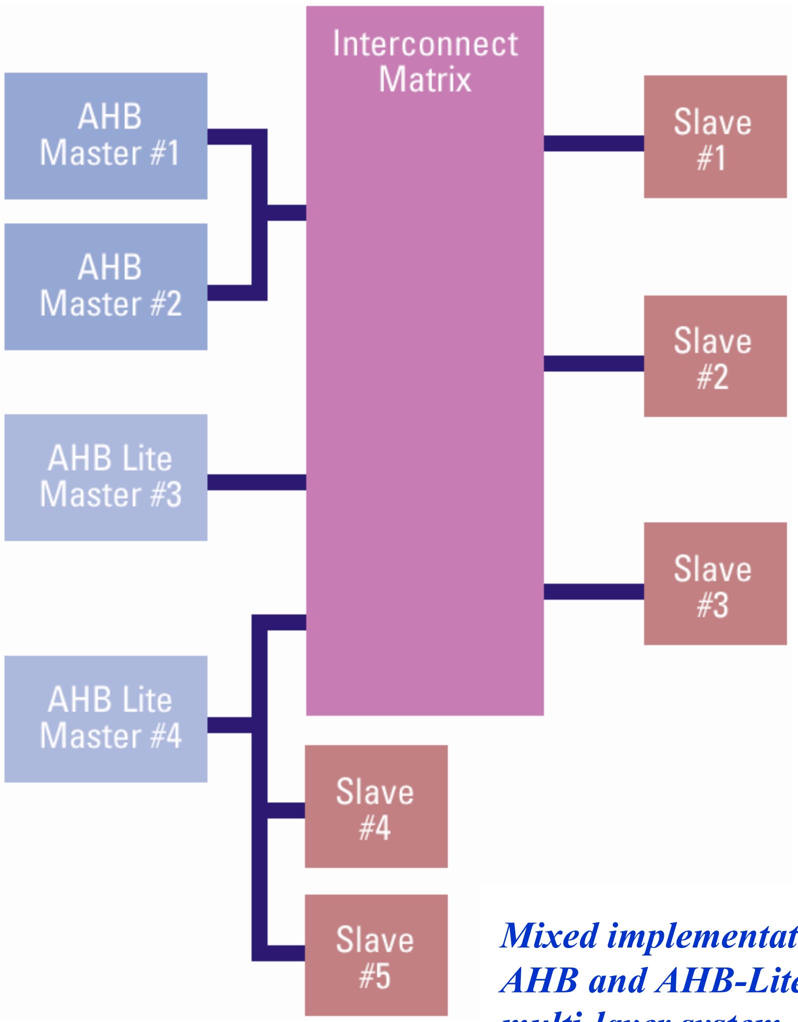
AHB-Lite Slave



Multi-layer AHB (1/2)



Multi-layer AHB (2/2)



- Local slaves
- Multiple slaves on one slave port
- Multiple masters on one layer

Mixed implementation of AHB and AHB-Lite in a multi-layer system.

Comparison among AMBA and other OCBs

	OPB	PLB	APB	ASB	AHB	PIbus	PIbus2	Mbus	PalmBus	FISPbus
Width (bits)	8, 16, 32	32 2.9 GB/s, 183 MHz 128 bit	up to 32 bit	8, 16, 32	2^n, n=3~10	8, 16, 32	8, 16, 32,64	8, 16, 32,64	8, 16, 32	
Peak Bandwidth (size/per cycle)	1 Data Tranfer	2 Data Tranfer	4 bytes 0.5 bus width	4 bytes 1 bus width	1 bus width	1 Data Tranfer	1 Data Tranfer	Data Bus Width	Data Bus Width	0.5/1 Data Tranfer (v1/v2)
Timing Guidelines	%	%	Symbolic term	Symbolic term	Symbolic term	early, middle. late	early, middle. late			
Synchronous			rising clock edge	falling clock edge*	rising clock edge	rising clock edge	rising clock edge			
Data Bus Implementation	Distributed And-Or/ Multiple xor	Multiple xor			Multiple xor	Tristate	Tristate			

Source - Black : OCB 1 1.0
- Other colors : Update

ARM Cores and Their Bus Interfaces

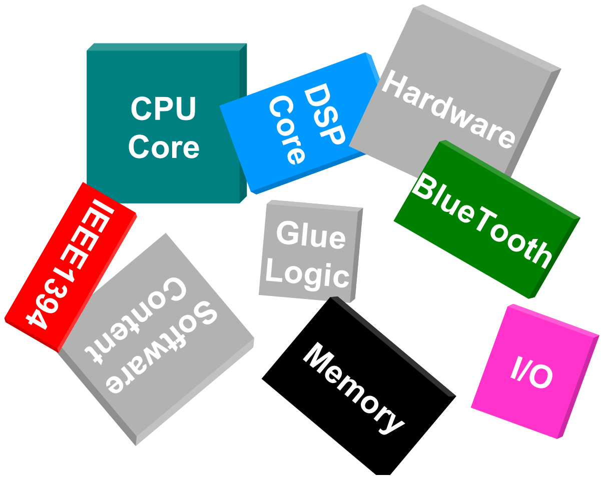
ARM	ARM7TDMI	ARM8	ARM9	ARM1020E
Transistors	74,209	124,554	111,000	7,000,000
Process Technology	0.35u	0.5u	0.25u	0.18u
Clock Rate	66MHz	72MHz	200MHz	400MHz
Vdd	3.3V	-	2.5V	1.5
MIPS	60	-	220	500
Data Bus	32bits	32bits	32bits	32-bit A 64bit W 64-bit R External memory bus interface is AMBA AHB compliant

✍ ARM System-on-Chip Architecture, by Steve Furber, Addison-Wesley, 2000

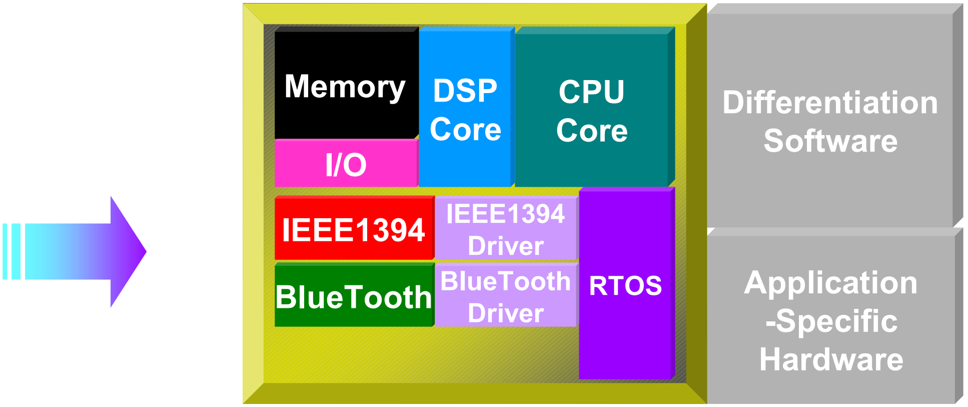
Outline

- VCI Interface Standards
- AMBA - On Chip Buses
- **Platform-based SoC Design**
- SoC Design Flow

The New System Design Paradigm



Block-Based Design



Platform-Based Design

Orthogonalization of concerns: the separation
of function and architecture,
of communication and computation

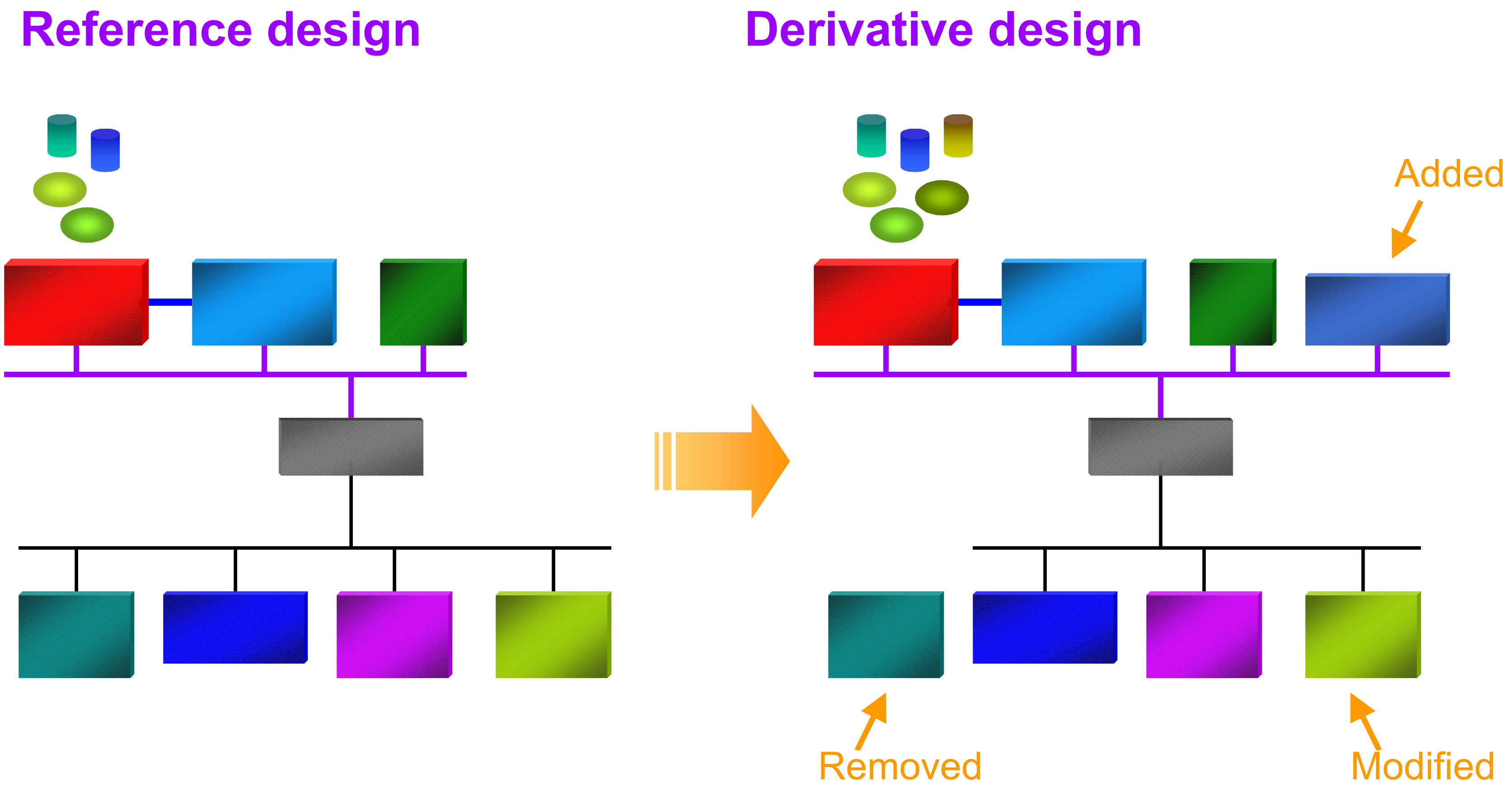
Terms

- Function
  - A function is an abstract view of the behavior of the system.
  - It is the input/output characterization of the system with respect to its environment.
  - It has not notion of implementation associated to it.
- Architecture
  - An architecture is a set of components, either abstract or with a physical dimension, that is used to implement a function.
- Architecture platform
  - A fixed set of components with some degrees if variability in the performance or dimensions of one or more of its components

Communication

- Communication provides for the transmission of data and control information between functions and with the outside world.
- Communication layers
  - Transaction: Point-to-point transfers between VCs. Covers the range of possible options and responses (VC interface).
  - Bus Transfer: Protocols used to successfully transfer data between two components across a bus.
  - Physical: Deal with the physical wiring of the buses, drive, and timing specific to process technology.

How Platform-Based Design Works?



Platform-based integration

- A fully defined architecture with
  - Bus structure
  - Clocking/power distribution
  - OS
- A collection of IP blocks
- Architecture reuse

The definition of a hardware platform is the result of a trade-off process involving
reusability, production cost and performance optimization.

Ingredients of A Platform

- Cores
  - Processor IP
  - Bus/Interconnection
  - Peripheral IP
  - Application specific IP
- Software
  - Drivers
  - Firmware
  - (Real-time) OS
  - Application software/libraries
- Validation
  - HW/SW Co-Verification
  - Compliance test suites
- Prototyping
  - HW emulation
  - FPGA based prototyping
  - Platform prototypes (i.e. dedicated prototyping devices)
  - SW prototyping

How to Build A Platform

- Architecture constraints for an integration platform:
  - first pick your application domain
  - then pick your on-chip communications architecture and structure (levels and structure of buses/private communications)
  - then pick your Star IP (e.g. processors) – processors ‘drag’ along detailed communications choices e.g. processor buses,
  - dedicated memory access, etc. - ARM-AMBA, etc. Also limit e.g. RTOS
  - pick application specific HW and SW IP
  - other IP blocks not available ‘wrapped’ to the on-chip communications may work with IP wrappers. VSI Alliance VCI is the best choice to start with for an adaptation layer

Pros & Cons of Platform-based Design Design

- Advantages
  - Can substantially shorten design cycles
  - Large share of pre-verified components helps address the validation bottleneck for complex designs
  - Enables quick derivative designs once the basic platform works
  - Rapid prototyping systems can be used to quickly build physical prototypes and start S/W development
- Limitations
  - Limited creativity due to predefined platform components and assembly
  - Differentiation more difficult to achieve, needs to be primarily in application software

Platform-summary

- What is a platform - a shortcut to time-to-market
  - Object
    - Architecture reuse
    - HW/SW co-design
  - Accessory: tools, design and test methodologies
- How to differentiate a platform
  - Programmability, Configurability, Scalability, Robustness
  - Performance, Area, Power
  - Application softwares
- Intention
  - Prototyping, product

Types of Platform

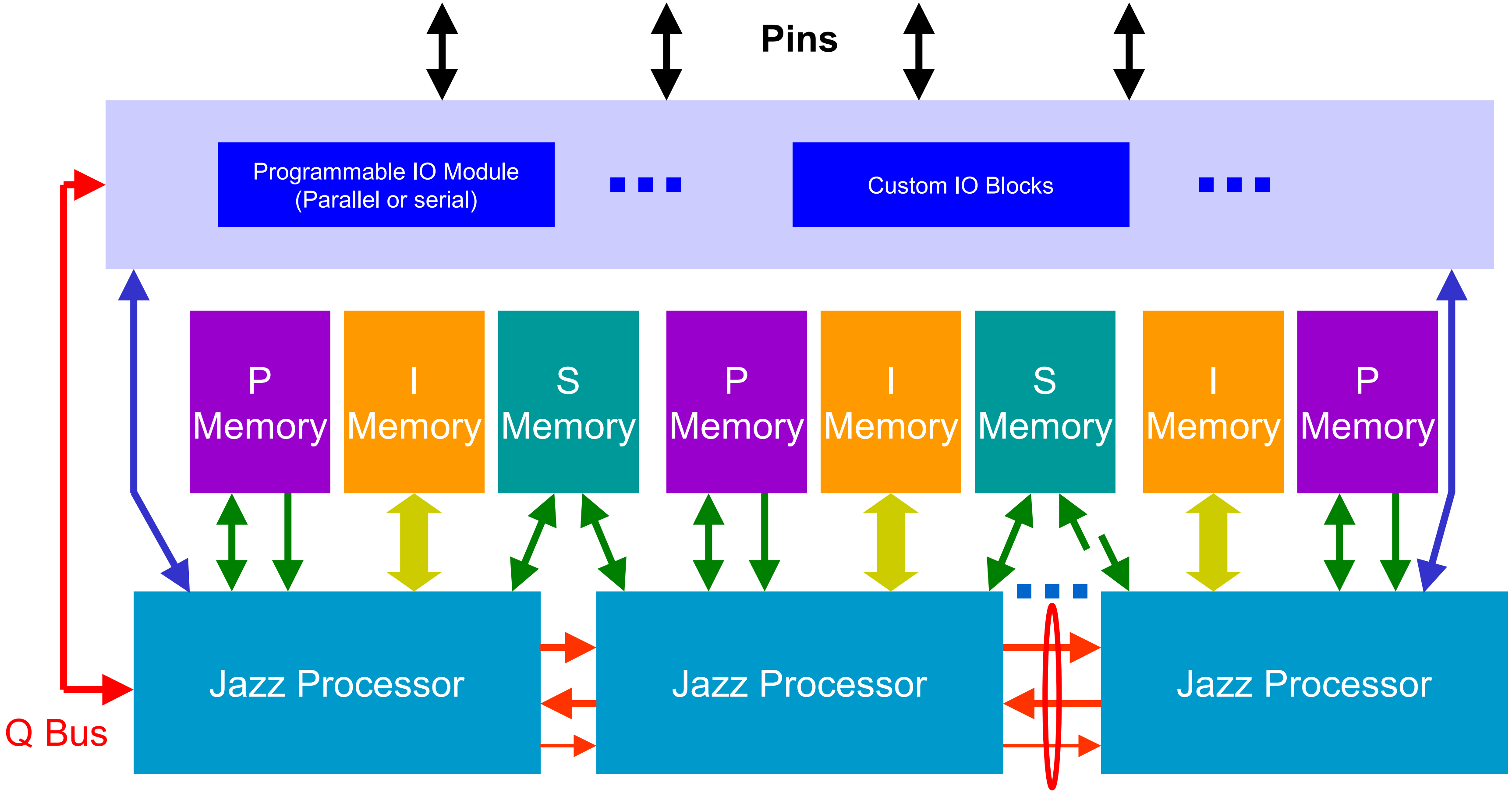
- According to the strength of constraints on hardware

stronger

  - Fixed Platforms
    - Software-oriented: TI's OMAP^{TM}, Philips Nexperia™.
    - Application-specific: Ericsson’s BCP,
  - Configurable platforms
    - Bus structure, multiple processor, programmable logic device
    - E.g.: Altera's Excalibur^{TM}, Triscend’s CSoC, Philips RSP, Cypress MicroSystems’ PSoC^{TM}, E.g.: Palmchip’s PalmPak^{TM}, Wipro’s SOC-RaPtor^{TM} , Tality’s ARM-based SoC.
  - Programmable platform
    - Improv’s - PSA^{TM} Jazz

weaker

Improv - PSA^{TM} Jazz Platform

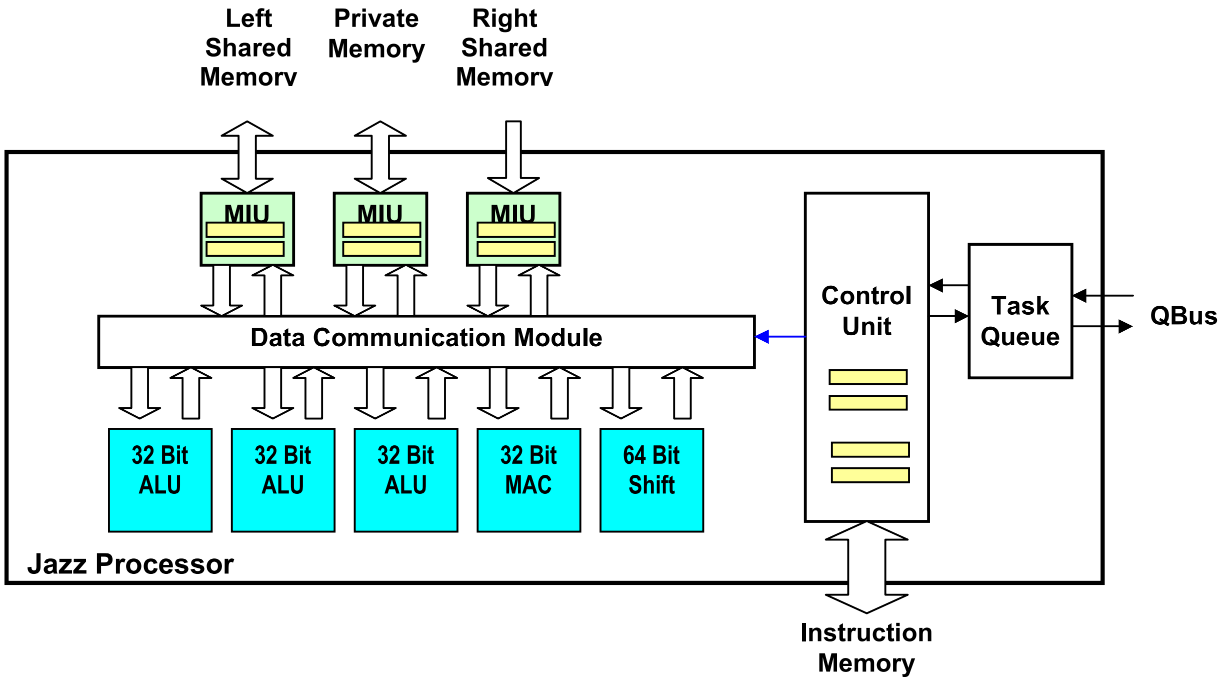


Acronym
I : Instruction
P: Private:
S: Shared

Q Bus (Queue Bus)
QBus-A →
Qbus-B ←
Arb →

Acronym - PSA: Programmable Systems Architecture

Jazz VLIW Processor - A Sample

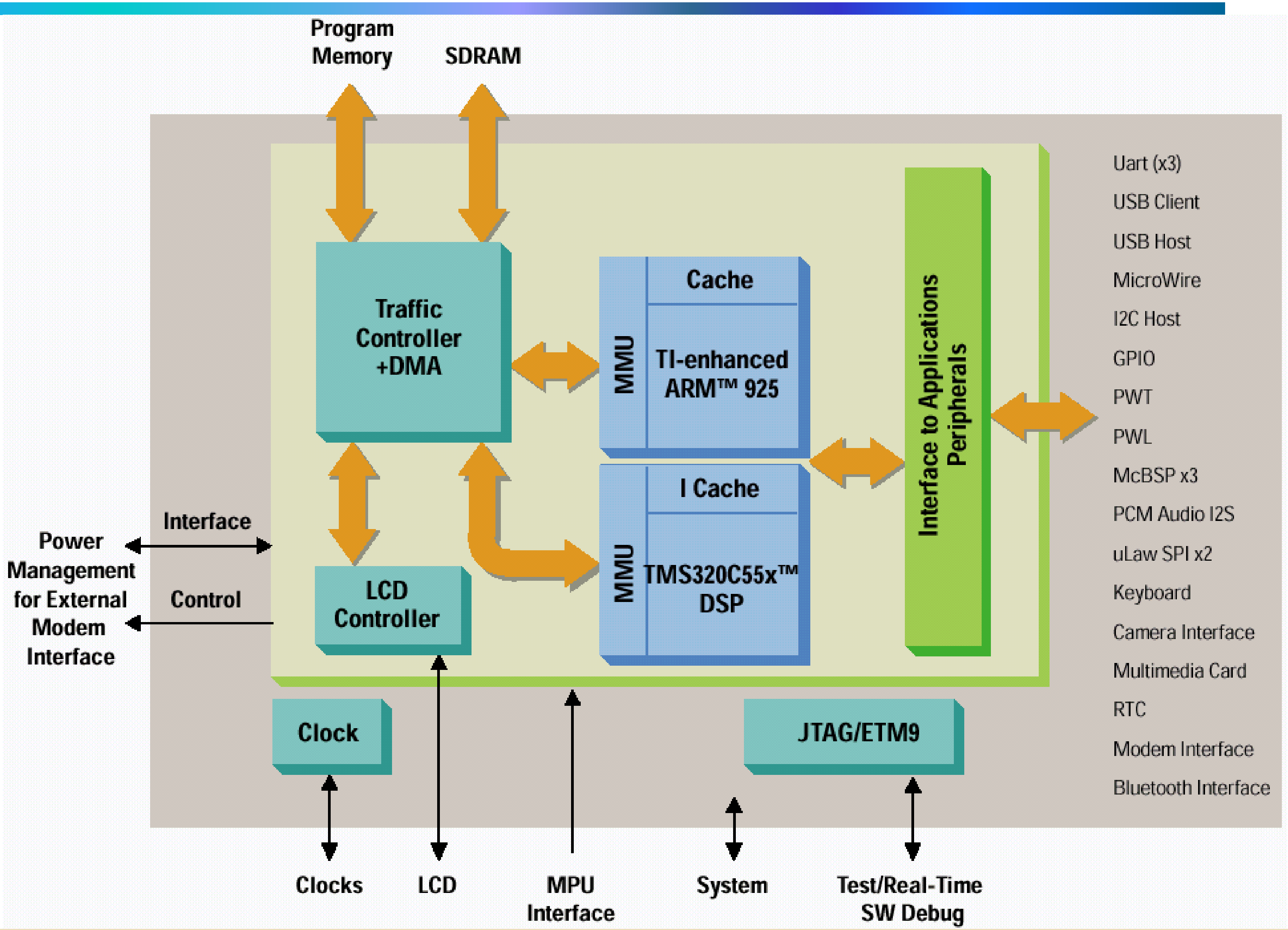


- 3 ALUs, 1 MAC, 1 SHIFT, 1CNT (built into control unit)
- 240 bit instruction width (memory image lower using instruction compression)
- 32-bit datapath, 16-bit address width
- 32 deep Task Queue
- 1.3 BOPS at 100 MHz (5 CU ops, counter, 7 MIU ops)
- ~100K gates

Features

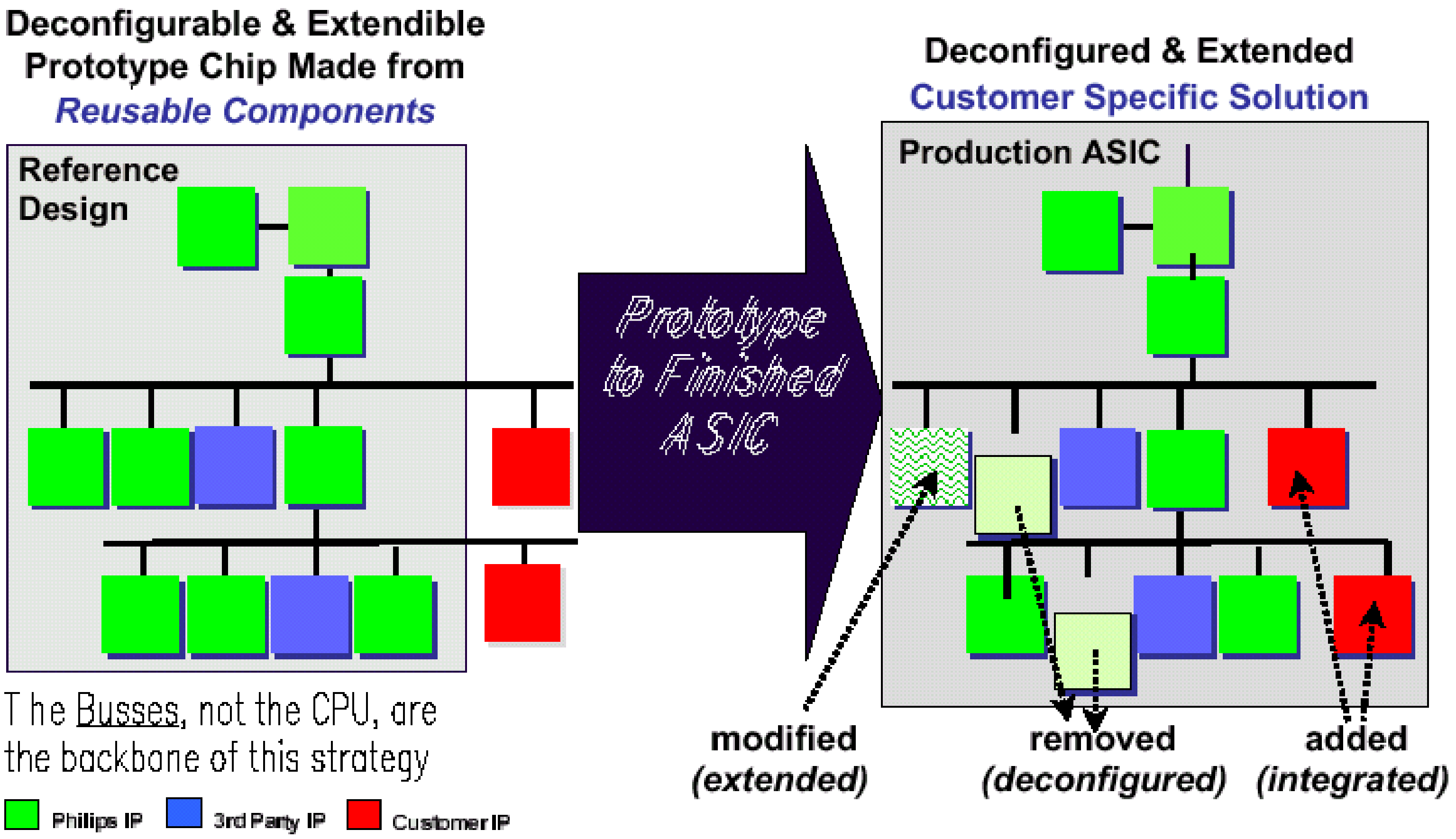
- State-of-the-art compilation technology that supports both
  - Task level parallelism (with the multiple processors)
  - Instruction level parallelism (through the Jazz VLIW processors).
- Designer start at the Java level
- No OS required
- Configuration at three levels
  - Platform - Collection of processors, data/instruction memory and I/O resource
  - Processor - Computation units and memory interfaces
  - Instruction - User can create custom logic computation units

TI's OMAP^{TM} Platform

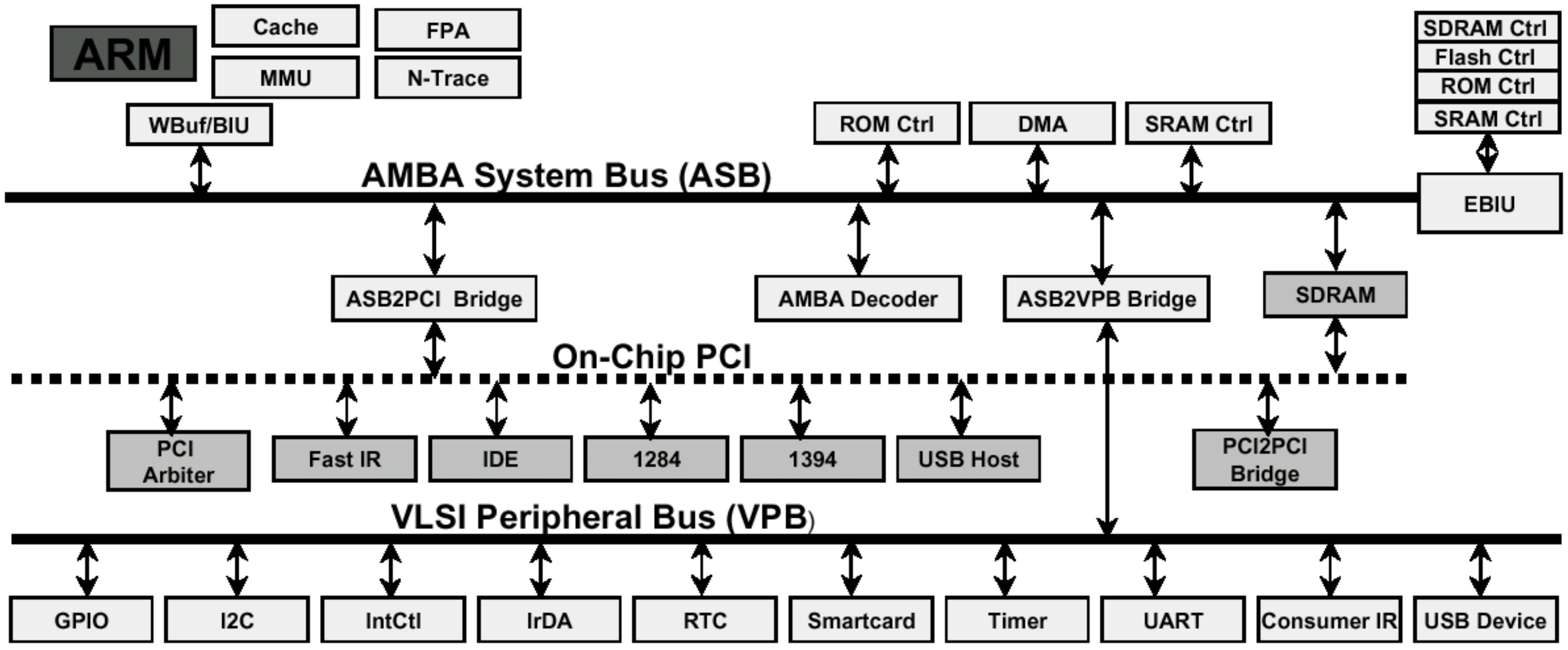


Acronym - OMAP: Open Multimedia Applications Platform

Philips - Rapid Silicon Prototyping (RSP)



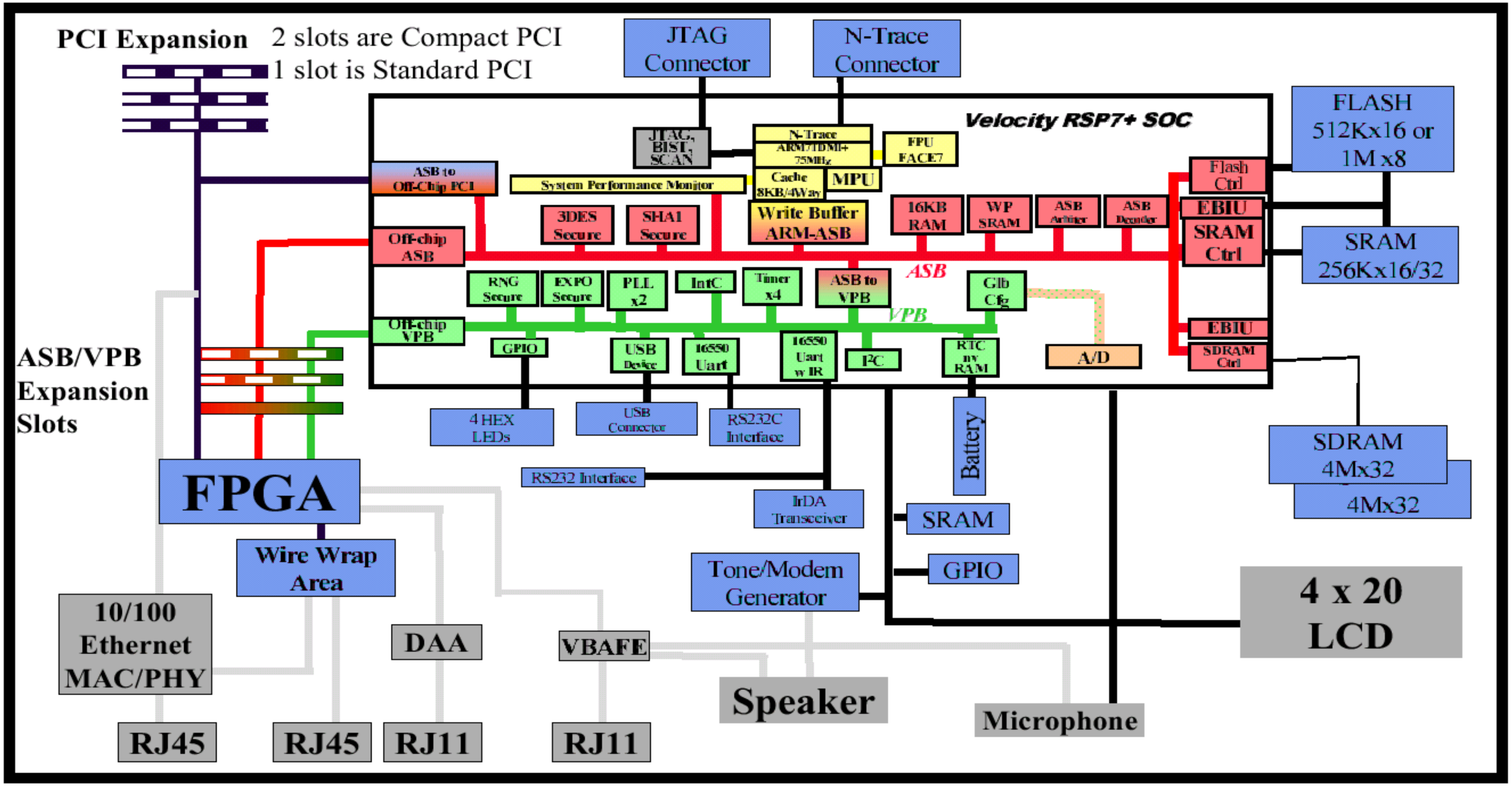
RSP7 ASIC Block Diagram



RSP7+ is targeted at customer designing SOC ASICs for:

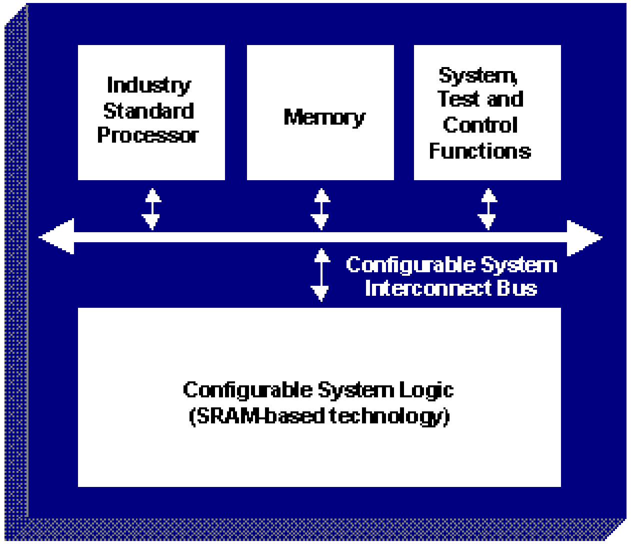
- Networking Peripherals
- Virtual Private Networks
- Systems Requiring ARM-based Control and Wired Connectivity

RSP7+ Emulation Board

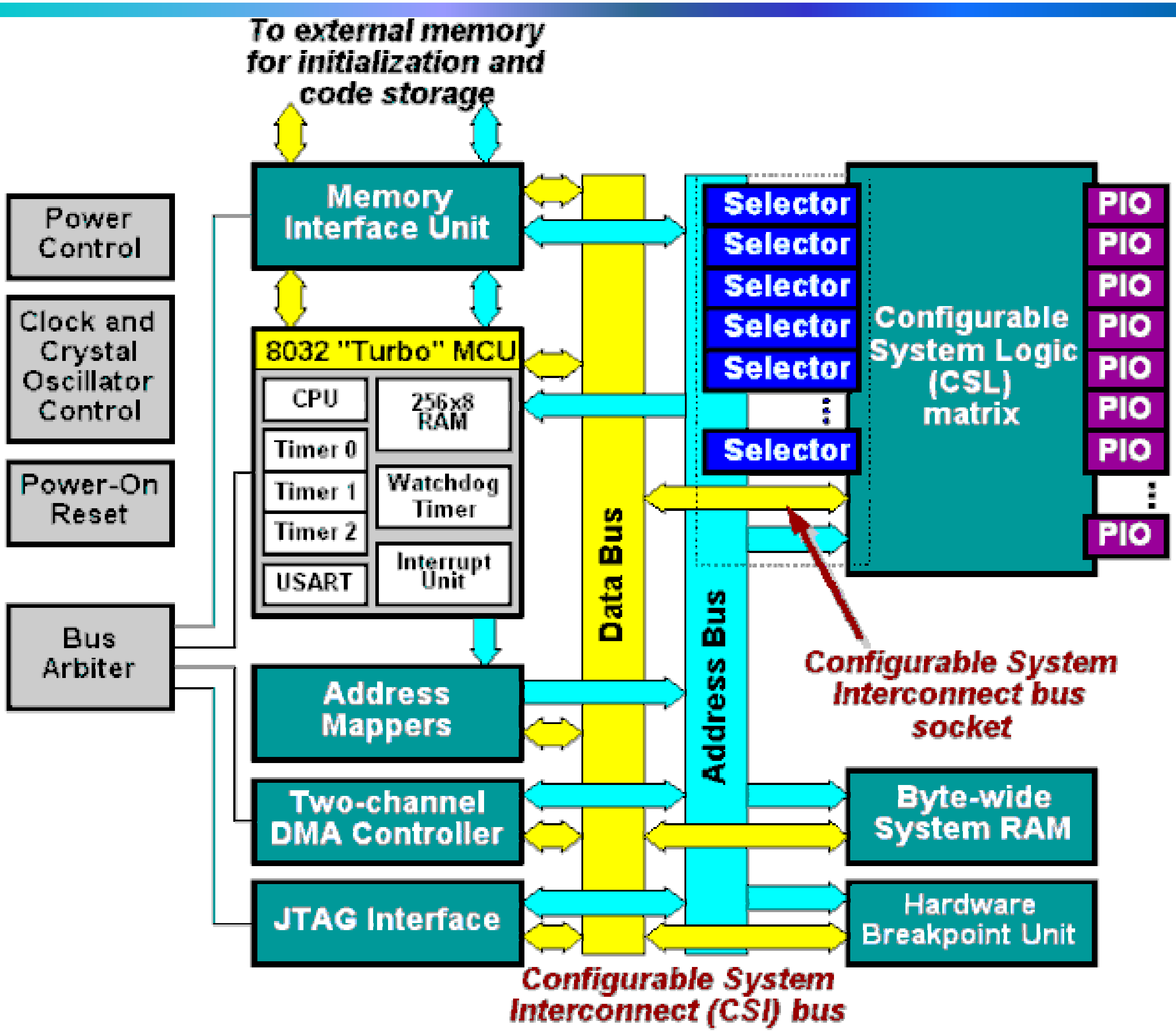


Triscend - Configurable System-on-Chip

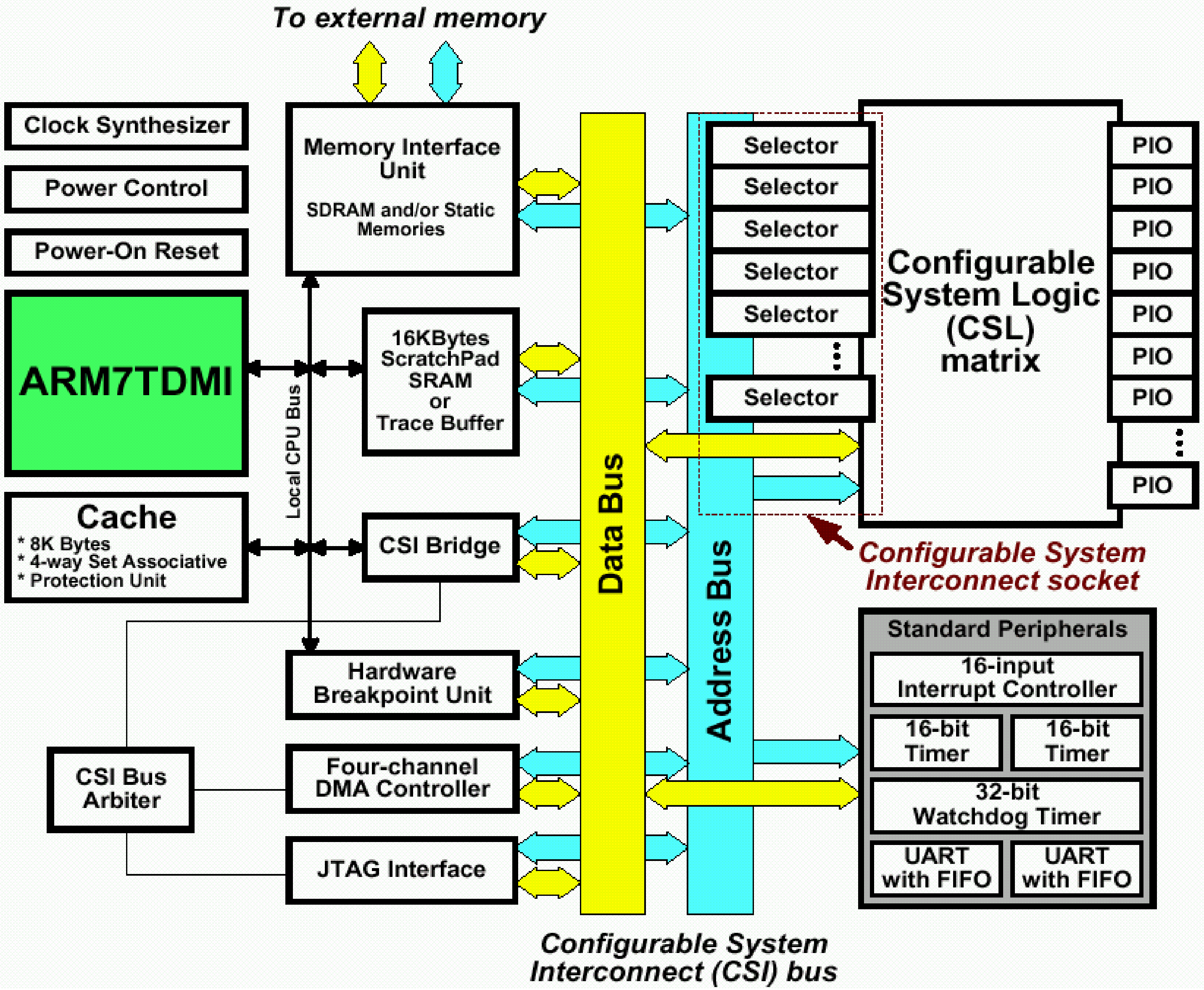
- A configurable system-on-chip (CSoC) is a single device consisting of:
  - A dedicated, industry-standard processor
    - 8051-based E5a
    - ARM-based for A7 device
    - SuperH for the future (2001.1.22 announced, 2002 available)
  - An open-standard, dedicated, on-chip bus
  - Configurable logic
  - Memory
  - Other system logic


Industry Standard Processor
Memory
System, Test and Control Functions
Configurable System Interconnect Bus
Configurable System Logic (SRAM-based technology)

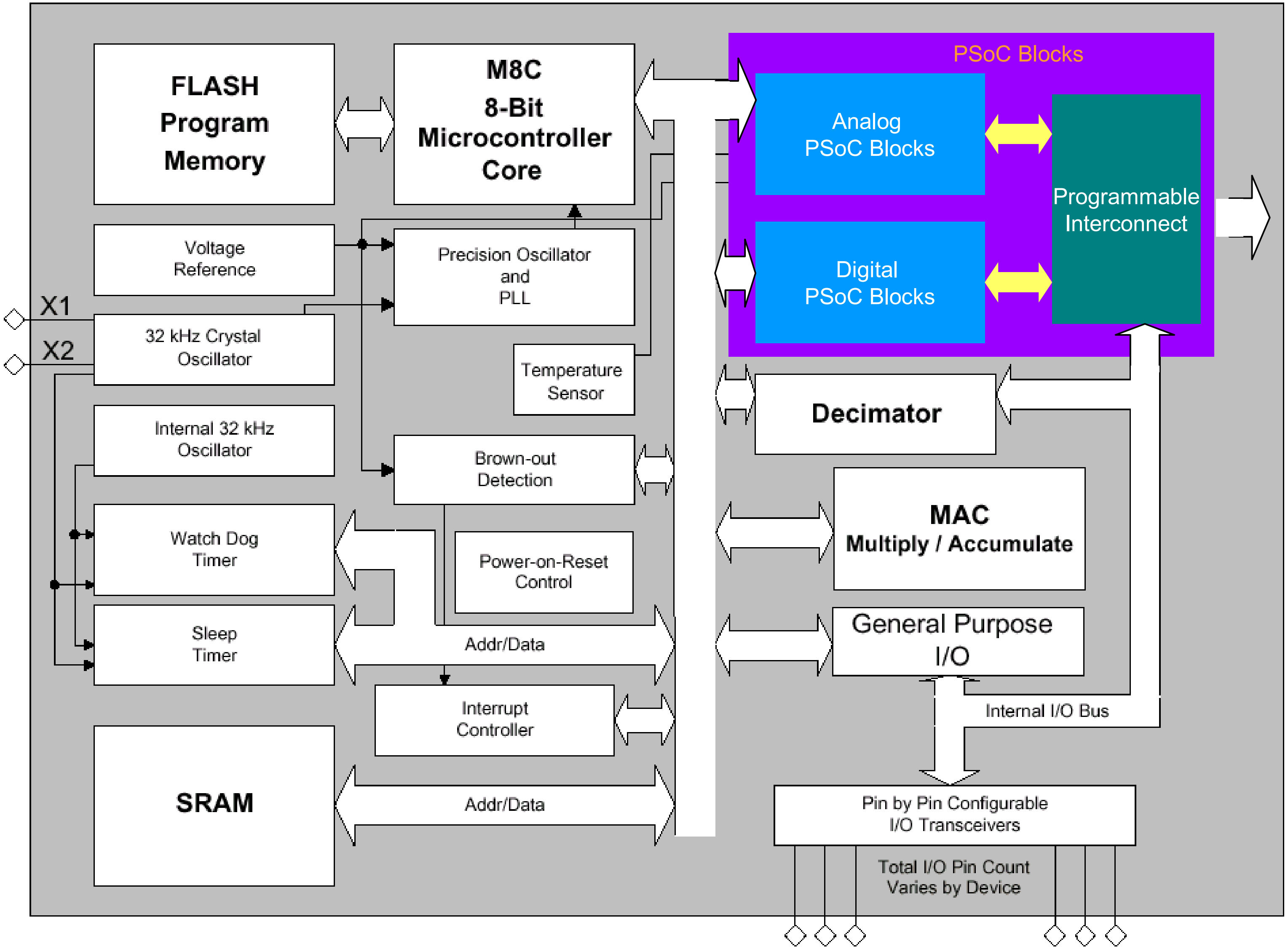
Triscend E5 System Highlights



Triscend A7 System Highlights



Cypress MicroSystems - PSoC^{TM}

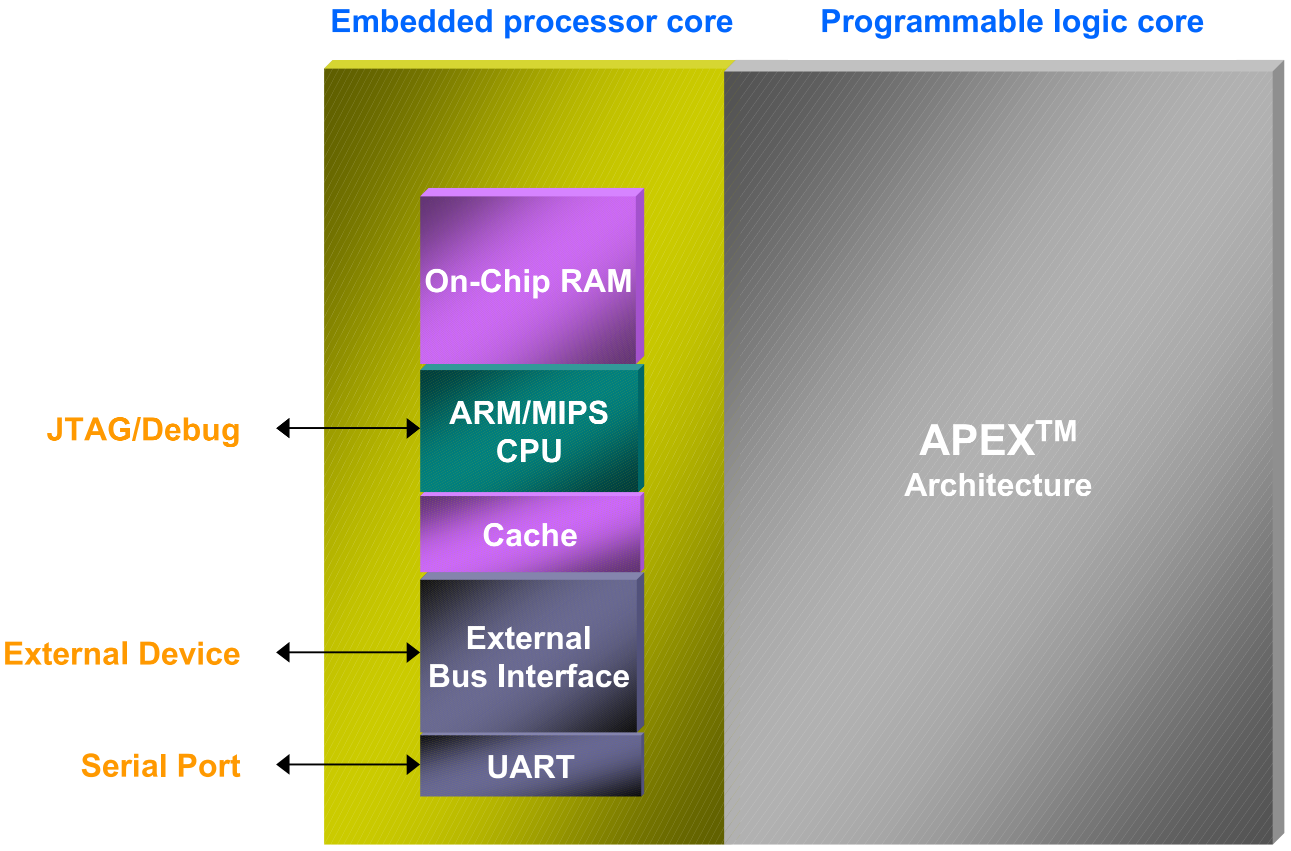


PSoC Blocks

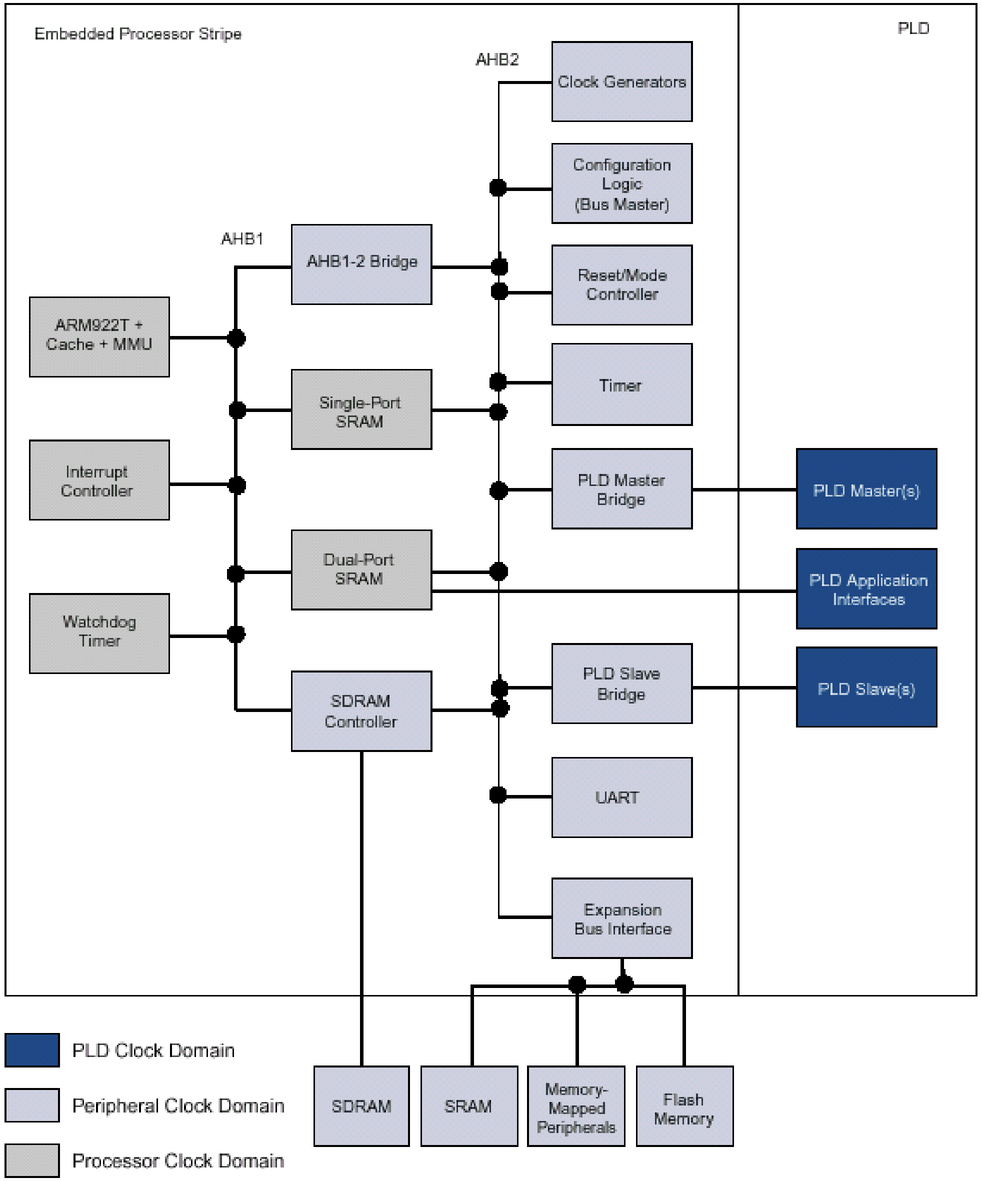
- Eight 8-bit digital PSoC blocks
  - Four Digital Basic Type A blocks:
    - Timer/Counter/Shifter/CRC/PRS/Deadband functions
  - Four Digital Communications Type A blocks:
    - Timer/Counter/Shifter/CRC/PRS/Deadband functions
    - Full-duplex UARTs and SPI master or slave functions
- Twelve analog PSoC blocks
  - Three types: ContinuousTime (CT) blocks, and type 1 and type 2 Switch Capacitor (SC) blocks that support
  - 14 bit Multi-Slope and 12 bit Delta-Sigma ADC, successive approximation ADCs up to 9 bits, DACs up to 9 bits, programmable gain stages, sample and hold circuits, programmable filters, differential comparators, and temperature sensor.

Altera - Excalibur^{TM} Embedded Processors

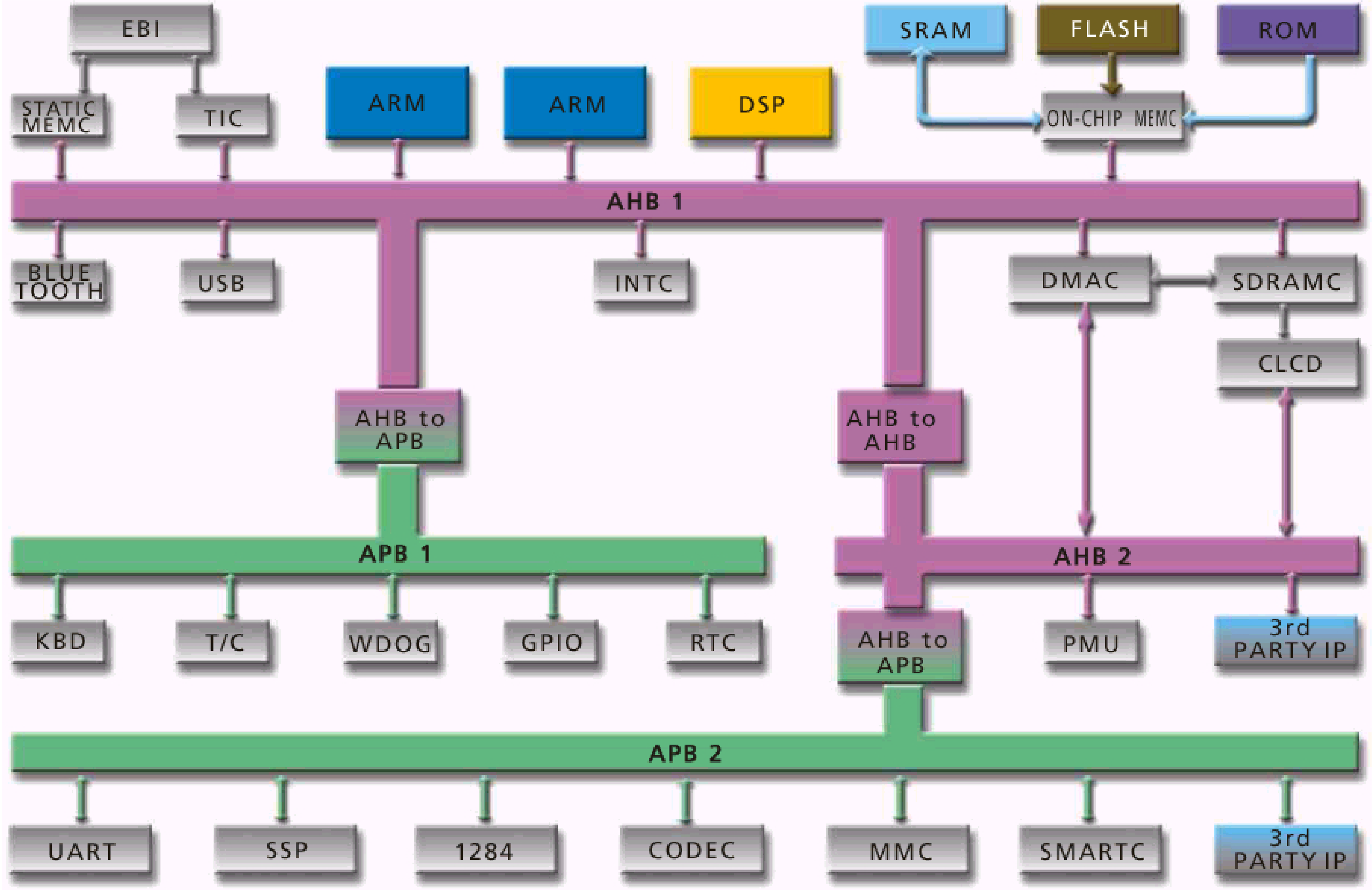
- Processors
  - ARM, MIPS



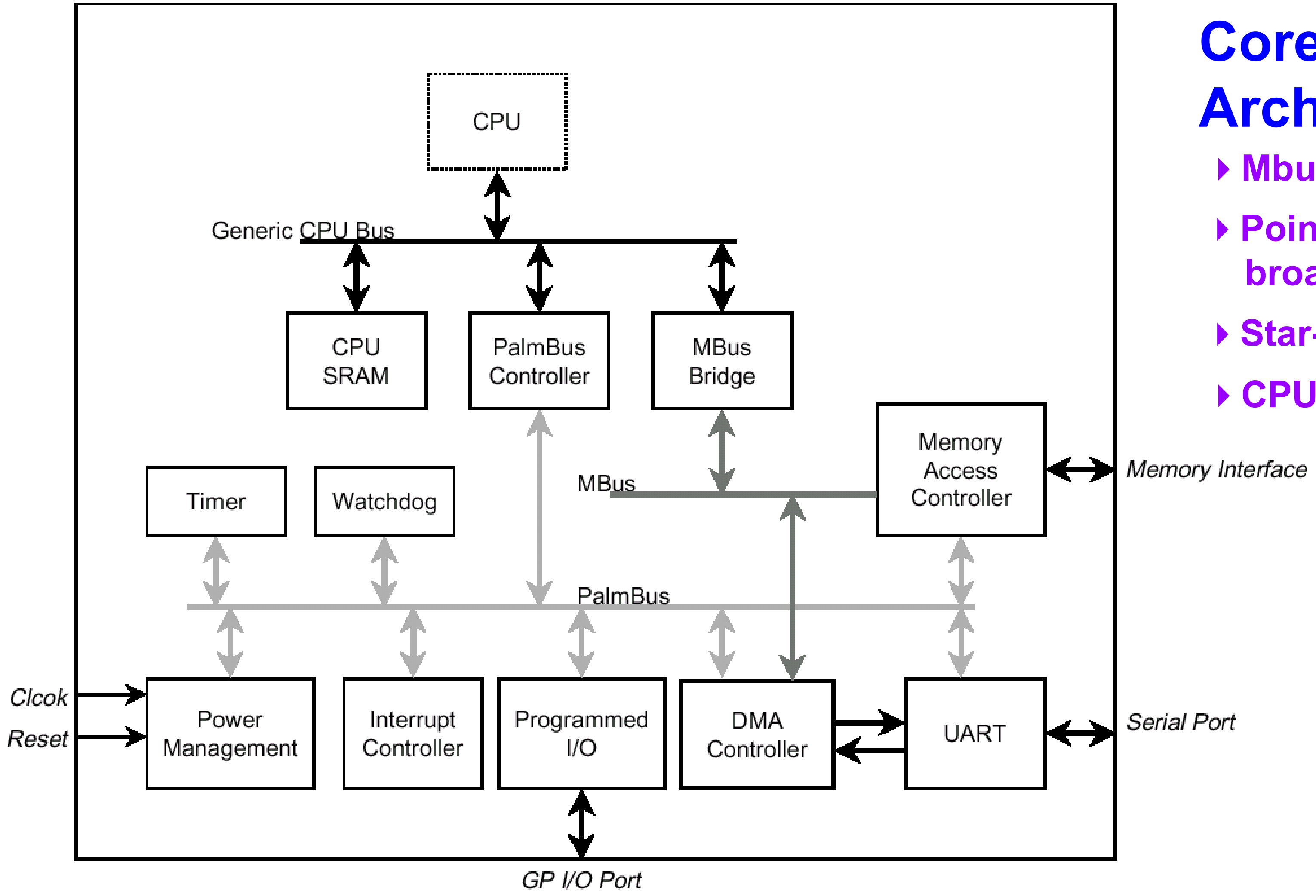
ARM-Based System Architecture



Wipro's SOC-RaPtor^{TM} Architecture



Palmchip's PalmPak^{TM} SoC Platform

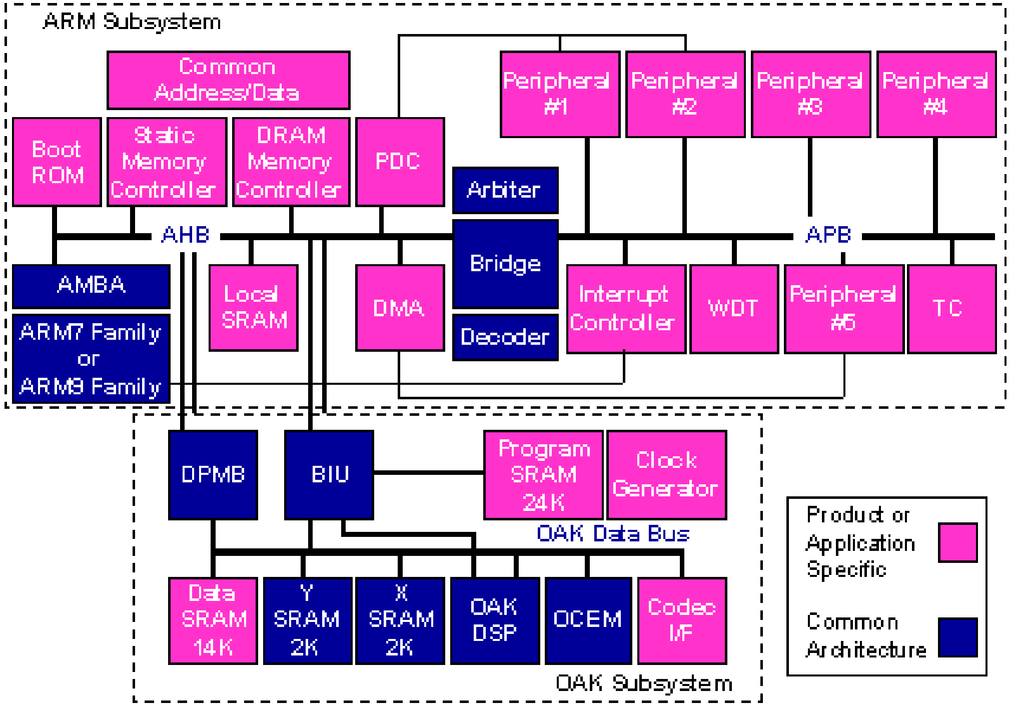


CoreFrame^{TM} Architecture

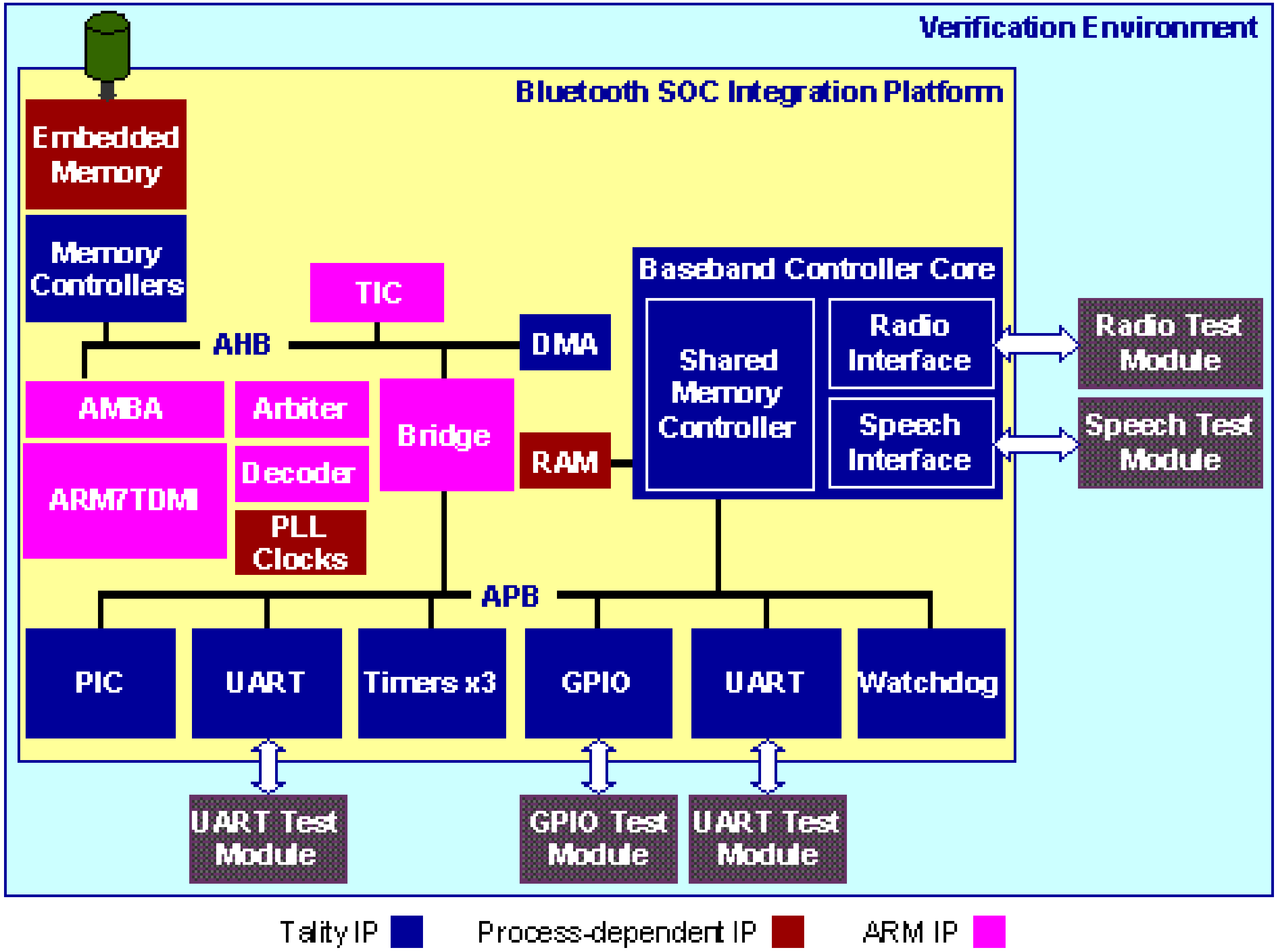
- Mbus and PalmBus
- Point-to-point and broadcast connections
- Star-shaped topology
- CPU Subsystem

Tality’s ARM/OAK-based SoC Platform

- Used as the development vehicle for multiple application-specific Integration Platforms.
  - for Bluetooth, xDSL and Cable Modems.
  - “Socketizes” the IP to make it AMBA 2.0-compliant.



Example of Tality's Derived Design - Bluetooth



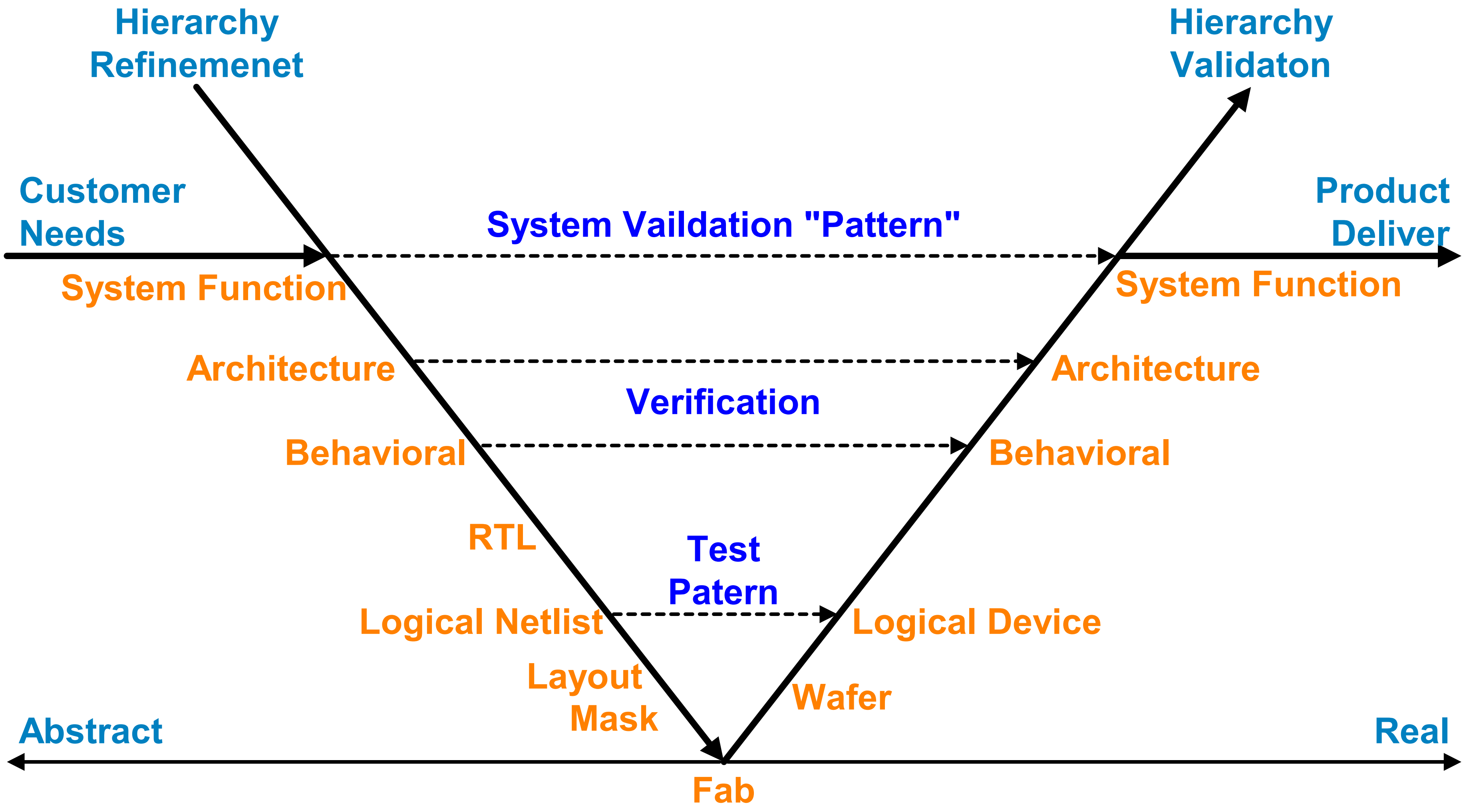
Outline

- VCI Interface Standards
- AMBA - On Chip Buses
- Platform-based SoC Design
- **SoC Design Flow**

Challenges of SoC Era

- Deign complexity
  - Validation & Verification
  - Design space exploration
  - Integration
  - Timing & power
  - Testing
  - Packaging
- Time to market
- The cost

From Requirement to Deliverables



Five SoC Design Issues

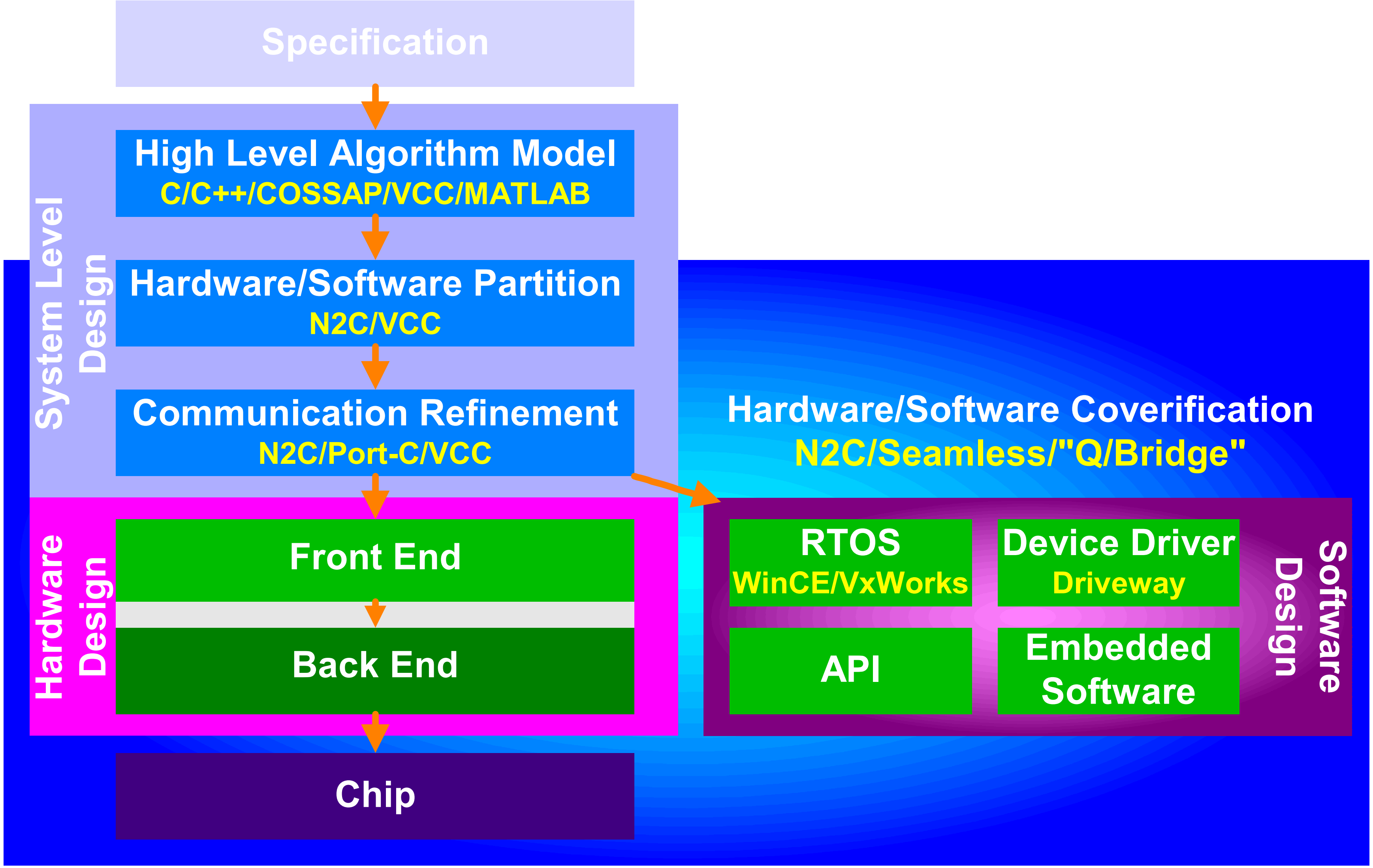
- To manage the design complexity
  - Co-design
  - IP modeling
  - Timing closure
  - Signal Integrity
  - Interoperability

How to Conquer the Complexity

- Use a known real entity
  - A pre-designed component (reuse)
  - A platform
- Partition
  - Based on functionality
  - Hardware and software
- Modeling
  - At different level
  - Consistent and accurate

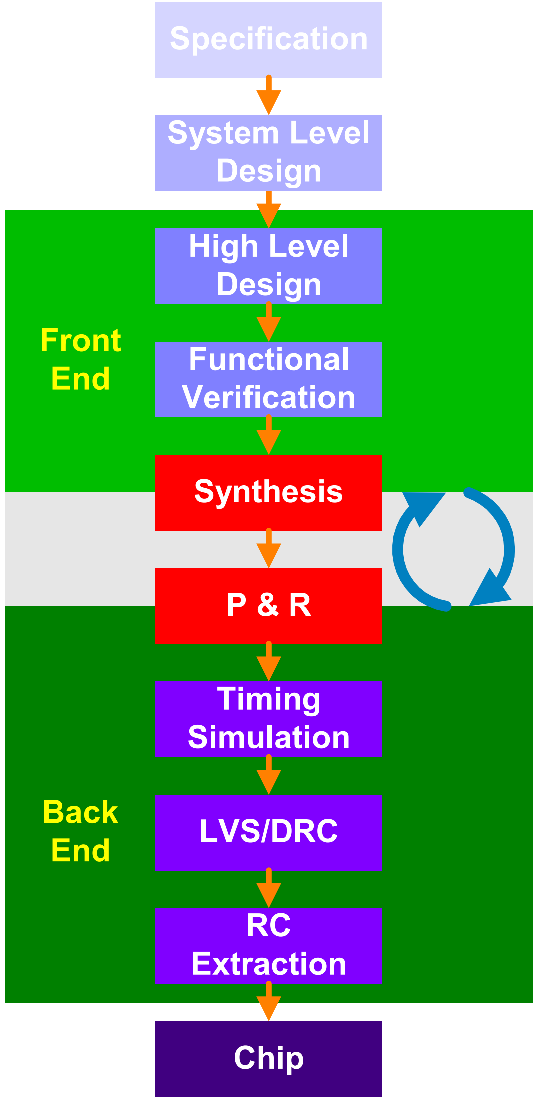
Institute of Electronics, National Chiao Tung University

SoC Design Flow



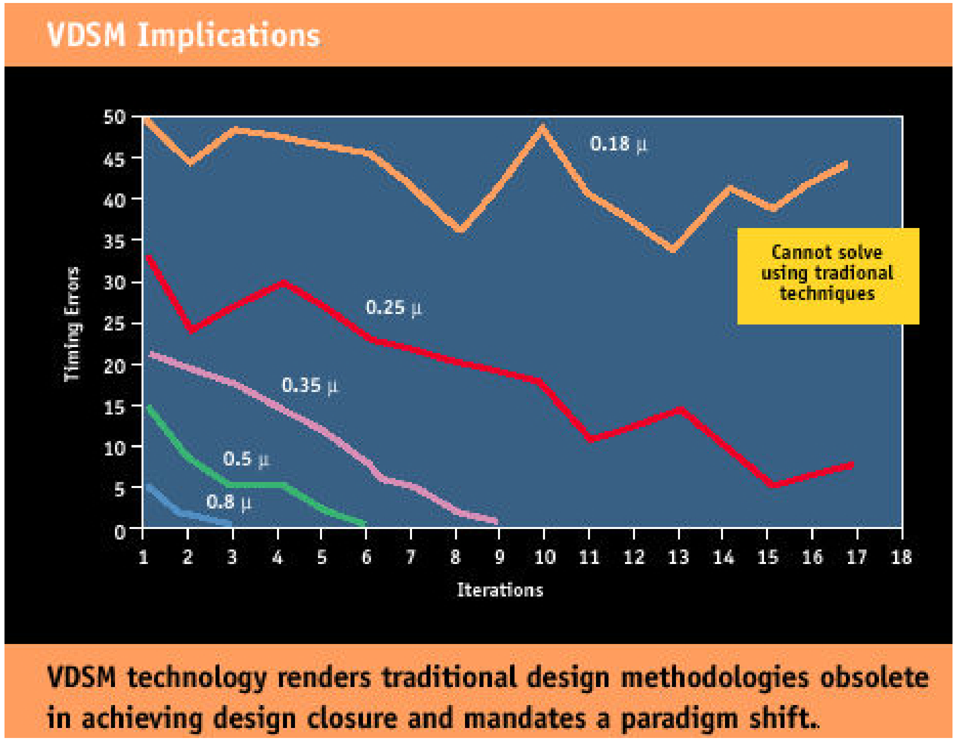
Physical Design Flow

- In VDSM
  - Interconnect dominates delay
  - Timing closure
  - Signal integrity
- Traditional design flow
  - Two-step process
  - Physical design is performed independently after logic design
- New design flow
  - Capture real technology behaviors early in the design flow
  - Break the iteration between physical design and logic design



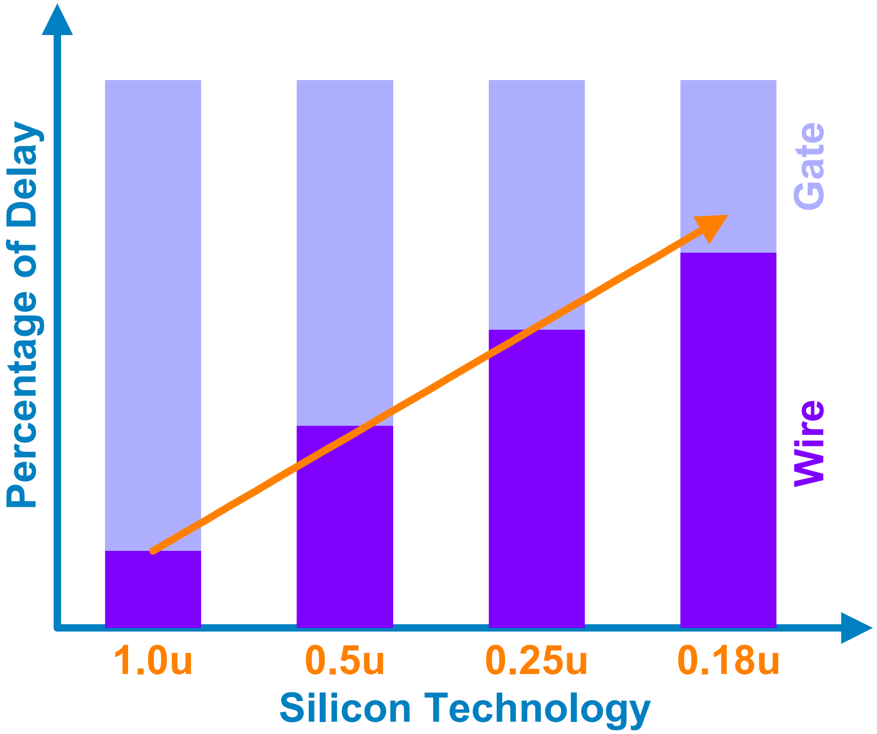
Making Sense of Interconnect

- At 0.35u, timing convergence started to become a problem.
- At 0.25u, it started to significantly impact the work of the designer.
- At 0.18u, if not accounted for, it actually causes designs to fail.



VDSM technology renders traditional design methodologies obsolete in achieving design closure and mandates a paradigm shift.

Source: Avant!

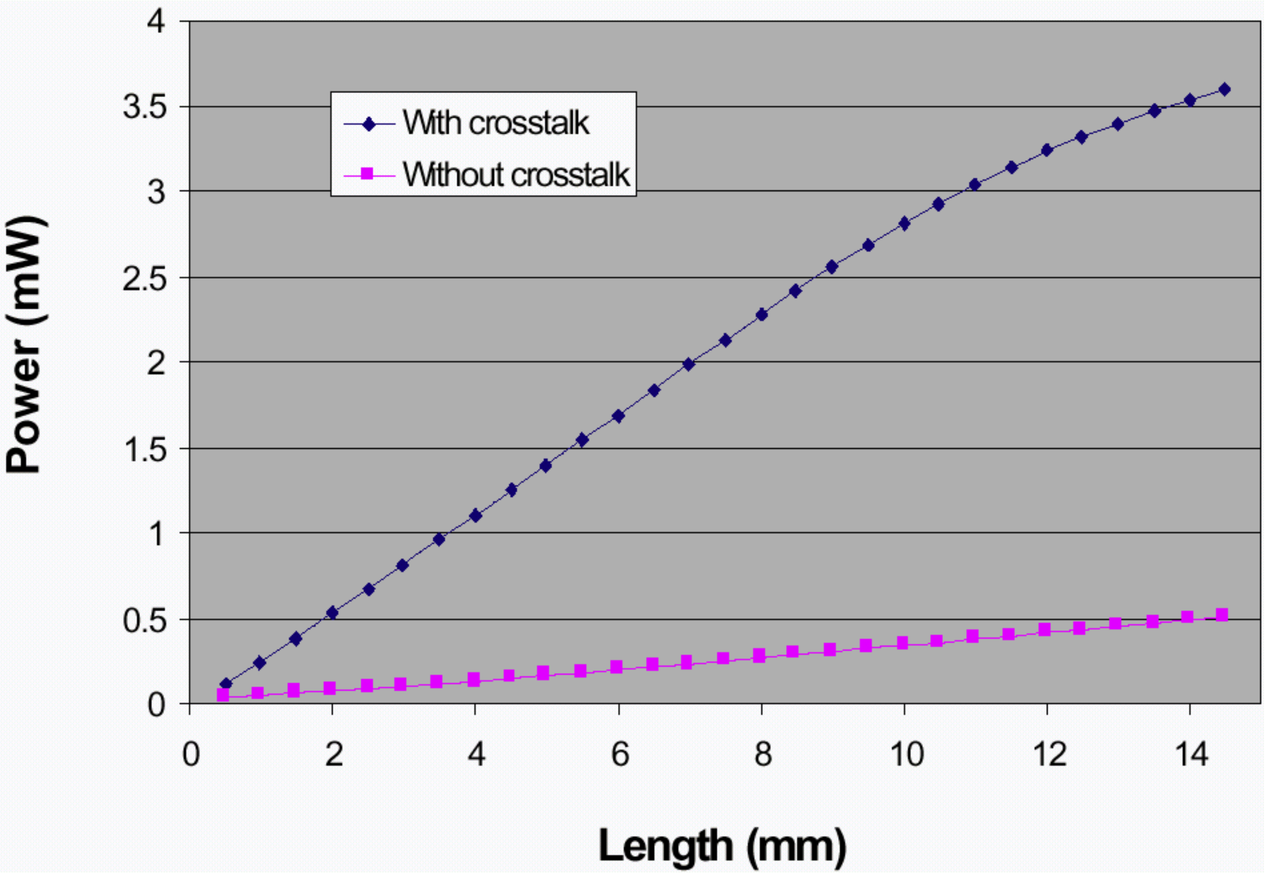


Source: Synopsys

Interconnect Power Consumption in DSM

- DSM effects in energy dissipation:

  cross-coupling capacitances



Source: Y. Zorian, S. Dey, and M. Rodgers, "*Test of Future System-on-Chips*," Proceeding of the 2000 International Conference on Computer-Aided Design, 392-398

Signal Integrity and Timing Closure

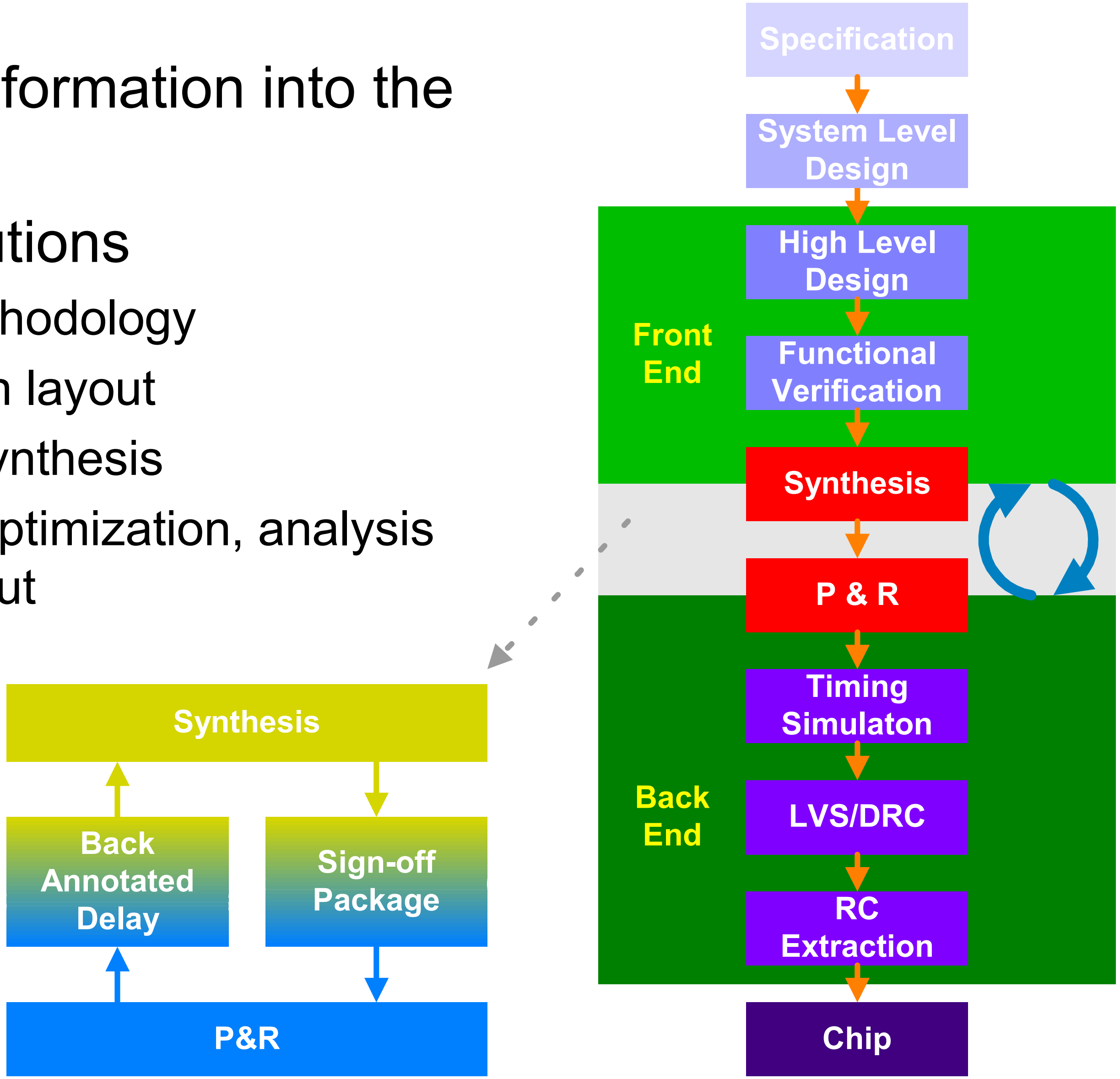
- Root causes of both Signal Integrity and Timing Closure
  - Inadequate interconnect modeling techniques
  - No effective design methodology
- Synthesis timing does not correlate with physical timing
  - Factors
    - Coupling capacitance increases
    - Interconnect resistance increases
    - Device noise margins decrease
    - Higher frequencies result in on-chip inductive effects
  - Problems
    - Signal electromigration
    - Antenna effects
    - Crosstalk delay
    - Crosstalk noise

Example - Crosstalk Delay

- Net-to-net coupling capacitance dominates as a percentage of total capacitance in VDSM.
- The coupling capacitance can be multiplied by the *Miller Effect*
  - Wire capacitance can be off by 2X if the adjacent wires are switching in the opposite direction.
  - The coupling capacitance can be much less than expected if the wires are switching in the same direction
- Both have to be considered during timing analysis to fully account for setup and hold constraints.

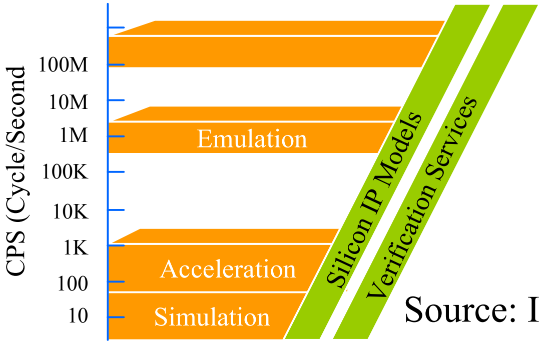
New Physical Design Flow Needed

- Bring physical information into the logical design
- Overview of solutions
  - Single pass methodology
  - Synthesis-driven layout
  - Layout-driven synthesis
  - All-Integrated (optimization, analysis and layout) layout



HW/SW Cosimulation Through Emulation

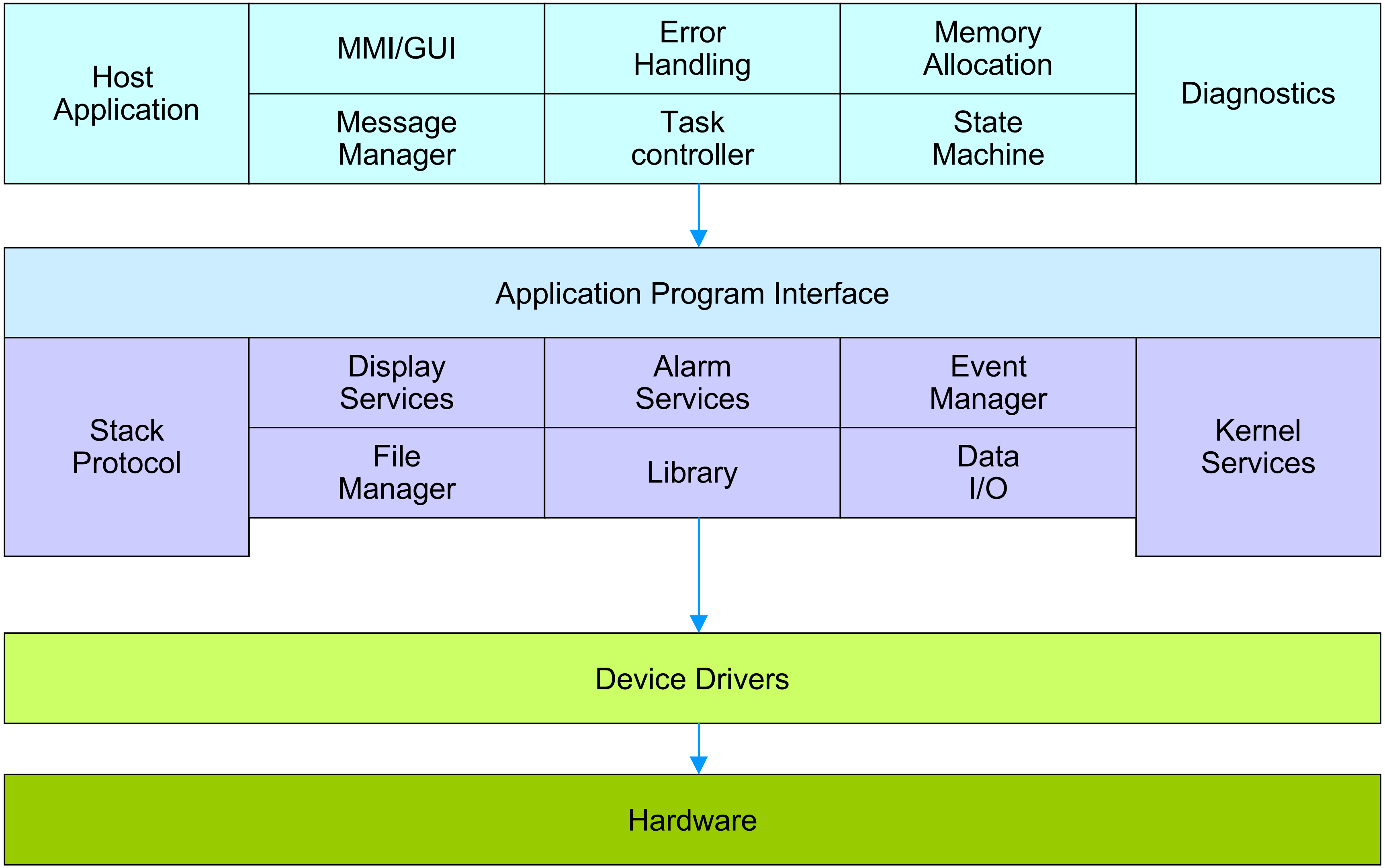
- Emulation in “virtual silicon”
  - Complete functional simulation of the chip at close to real time
  - Run real software



Source: IKOS Systems Inc

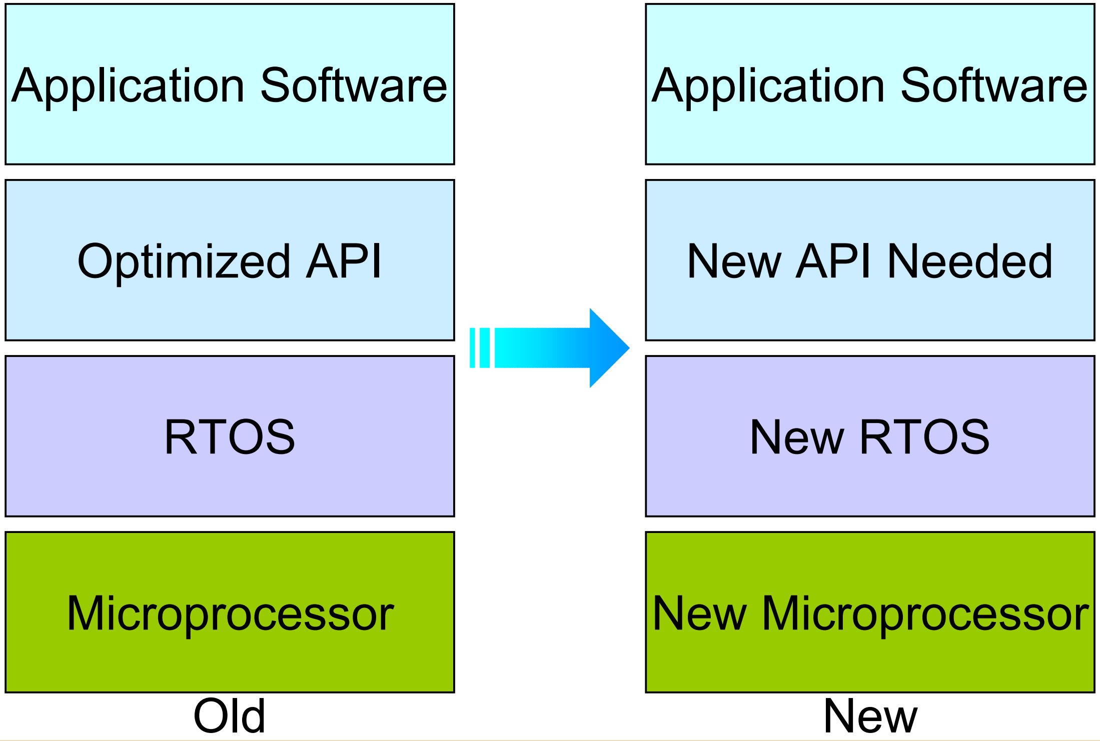
- Tools to enable simulation between EDAs and emulators
  - Cycle-based simulators
  - Full-timing simulators
  - Instruction set simulators
  - E.g. Quickturn Q/Bridge
- Expensive, long learning curve and set-up time

Embedded Software Architecture for SoC Design



Software Development

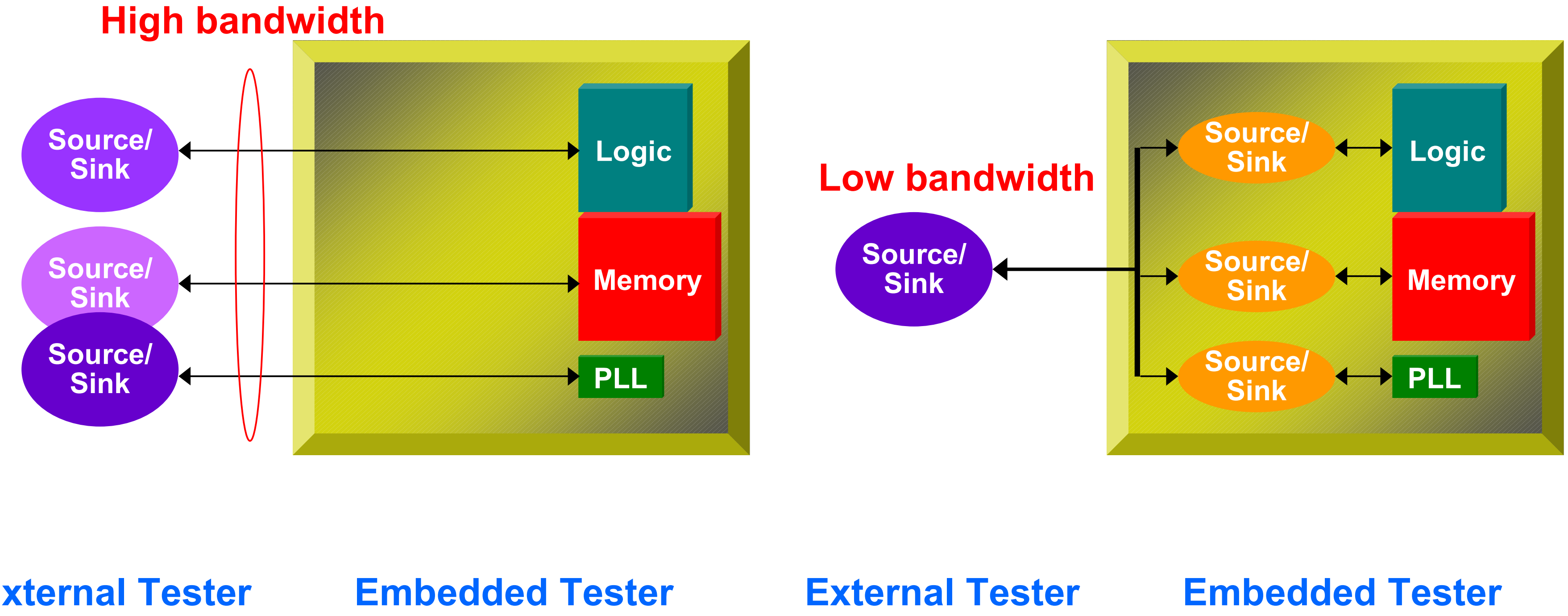
- Porting software to a new processor and RTOS
  - Using a common RTOS abstraction layer
- The evolution of embedded system in the future
  - An standard RTOS



Software Performance Estimation

- Have to take the following into account
  - Instruction set
  - Bus loading
  - Memory fetching
  - Register allocation
- Example: Cadence VCC technology
  - CPU characterized as Virtual Processor Model
  - Using a Virtual Machine Instruction Set
  - SW estimation using annotation into C-Code
  - Good for early system scheduling, processor load estimation
    - Two orders of magnitude faster than ISS
    - Greater than 80% accuracy

Tester Partitioning

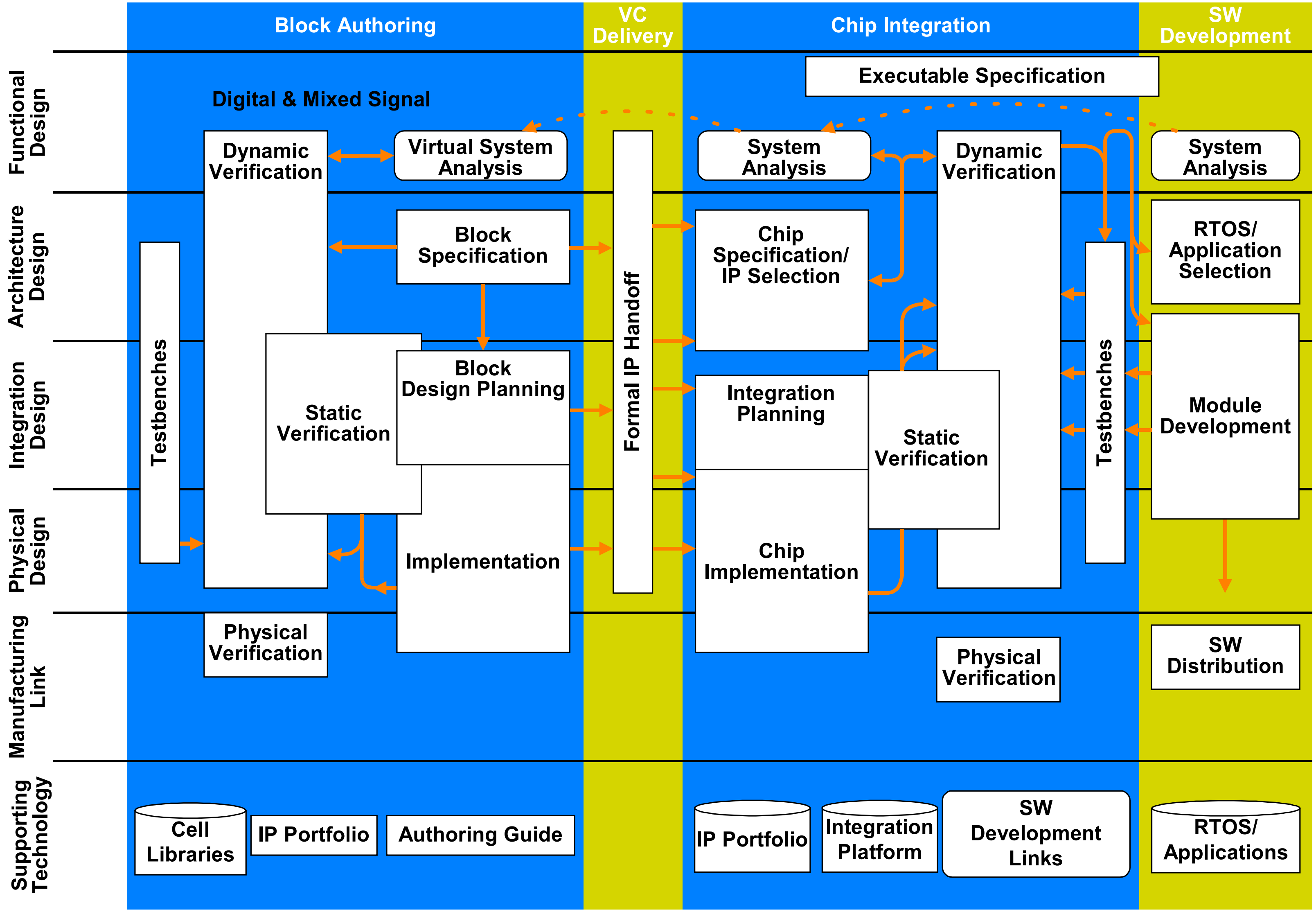


Source: Y. Zorian, S. Dey, and M. Rodgers, "*Test of Future System-on-Chips*," Proceeding of the 2000 International Conference on Computer-Aided Design, 392-398

Self-Testing of Embedded Processor Cores

- Logic BIST
  - Based on the application of pseudo random test patterns generated by on-chip test pattern generators like LFSRs.
  - Cannot always achieve very high fault coverage for processor cores.
- Instruction-based self-test techniques
  - Rely on generating and applying random instruction sequences to processor cores.
  - The approach determines the structural test needs of processor components
  - Advantage: programmability and flexibility

Platform-based Design



✍ Source: "Surviving the SOC Revolution - A Guide to Platform-Based Design" by Henry Chang et al, Kluwer Academic Publishers, 1999

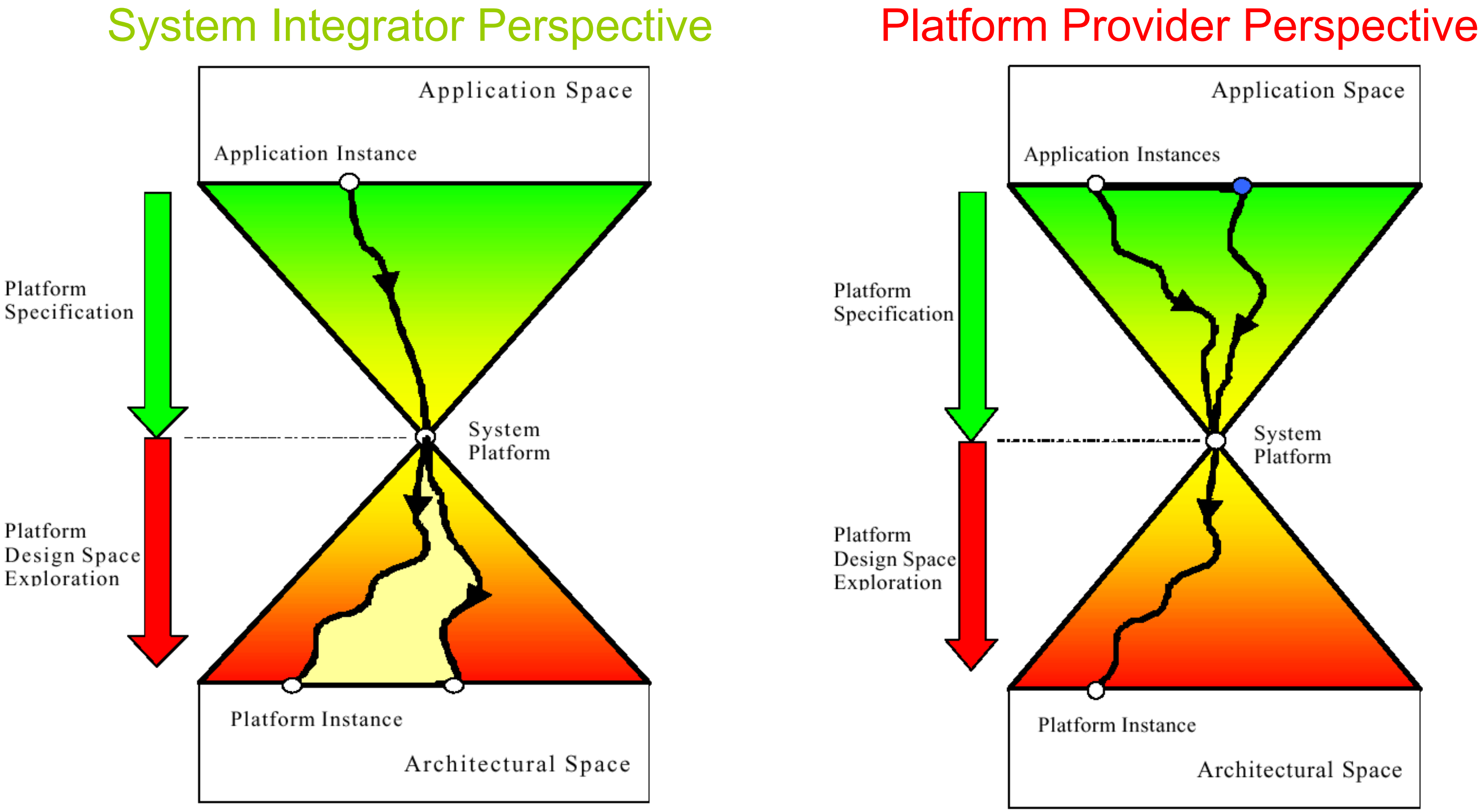
Design Entry

Gate level Truth table FSM Waveform	1K~10K
RTL level	10K~100k
System level modeling	100K~100M

Manage Size and Run-Time

Hardware Platform-Based Design

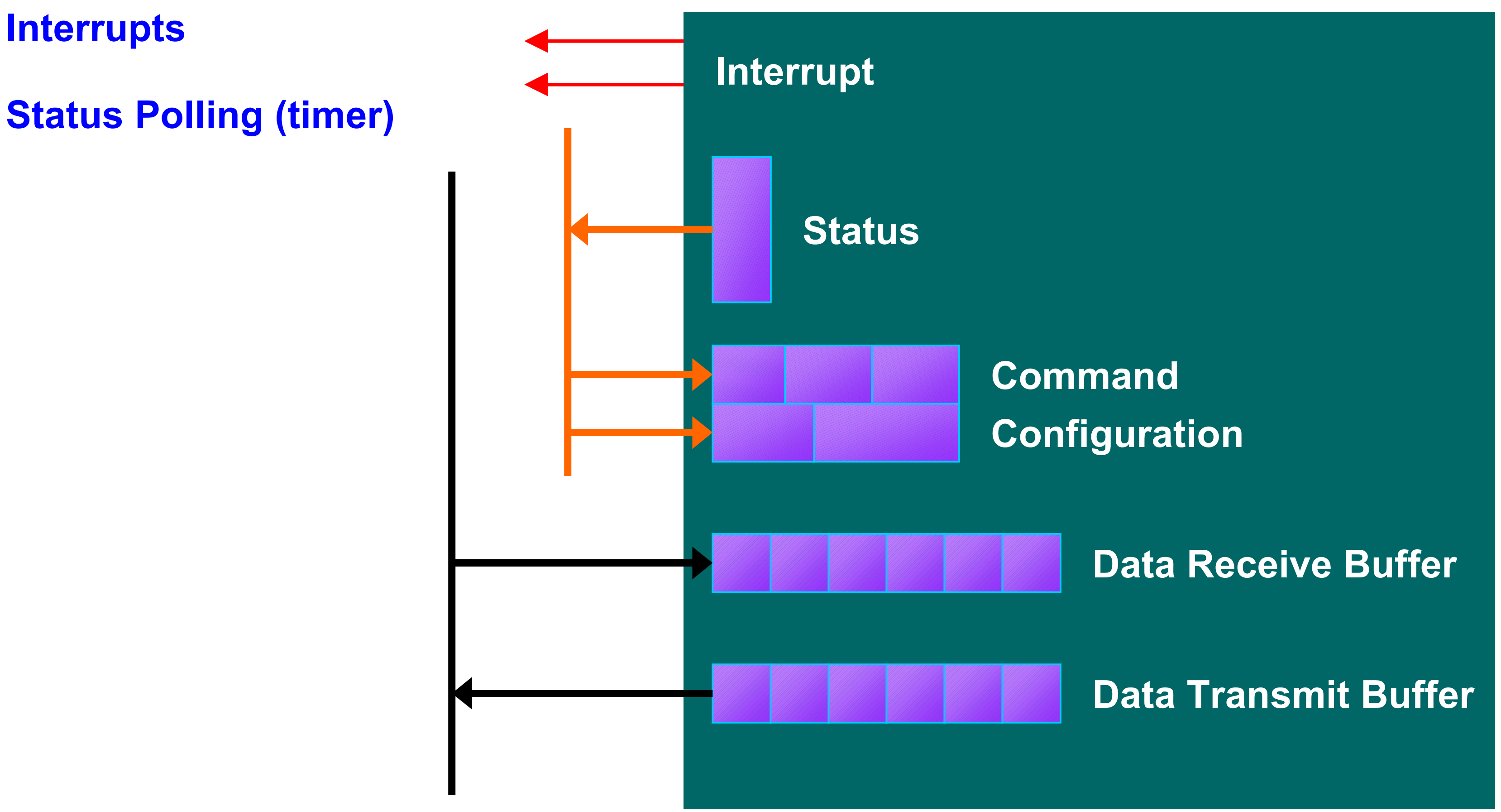
It is a "meet-in-the-middle" approach.



System-Level Design

- Goal
  - To define the platform that satisfies the system functions with performance/cost tradeoff
- Platform design
  - Bus structure
  - IP and their function design
    - **Customized instructions**
    - **Parallelism**
    - **Command parameters**
    - **Configurable parameters**
    - **IP parameters**
  - Control scheme
  - Data communication (bandwidth)

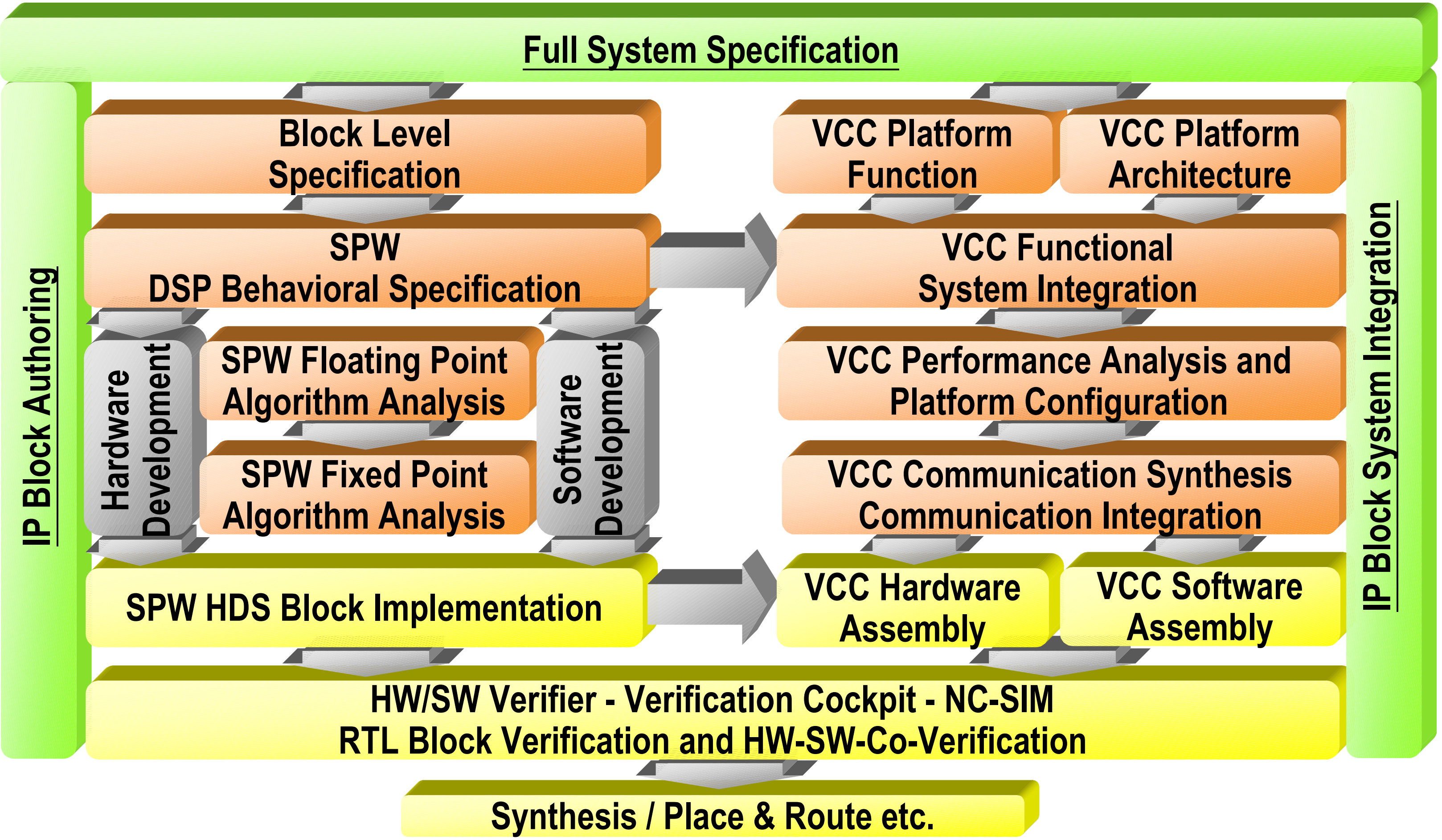
Control Scheme Model



Some Helps in System-Level Design

- Cadence VCC (Virtual Component Codesign, from Cadence Berkely Labs)
  - Performance simulation
  - Communication refinement technology
- Vast Systems Technology
  - VPM (Virtual Processor Model)
  - HW/SW codesign
- CoWare N2C (Napkin-to-chip)
  - Interface synthesis
- SystemC

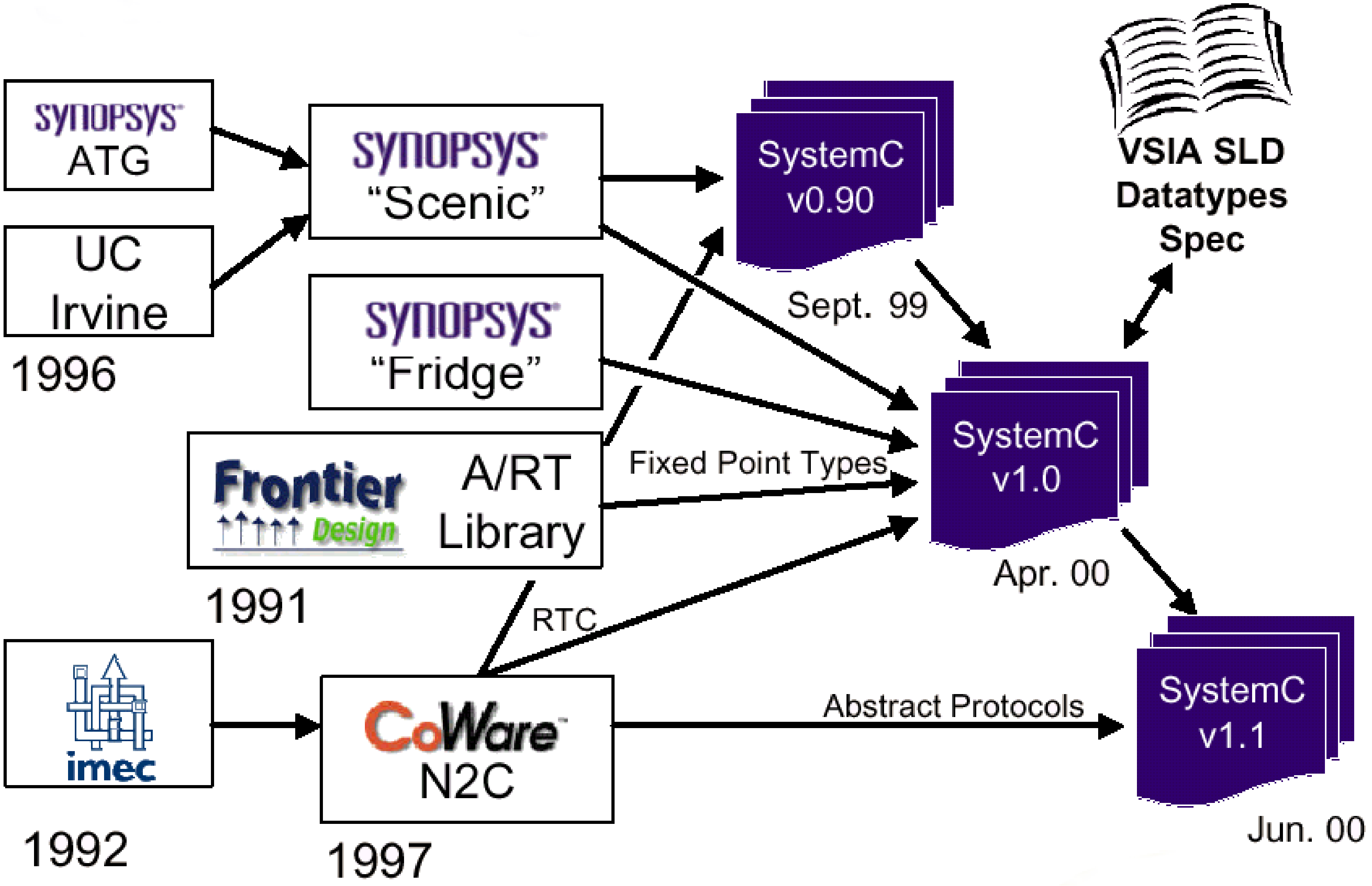
Cadence’s VCC


Full System Specification
IP Block Authoring
IP Block System Integration
Block Level Specification
VCC Platform Function
VCC Platform Architecture
SPW DSP Behavioral Specification
VCC Functional System Integration
Hardware Development
SPW Floating Point Algorithm Analysis
Software Development
VCC Performance Analysis and Platform Configuration
SPW Fixed Point Algorithm Analysis
VCC Communication Synthesis Communication Integration
SPW HDS Block Implementation
VCC Hardware Assembly
VCC Software Assembly
HW/SW Verifier - Verification Cockpit - NC-SIM
RTL Block Verification and HW-SW-Co-Verification
Synthesis / Place & Route etc.

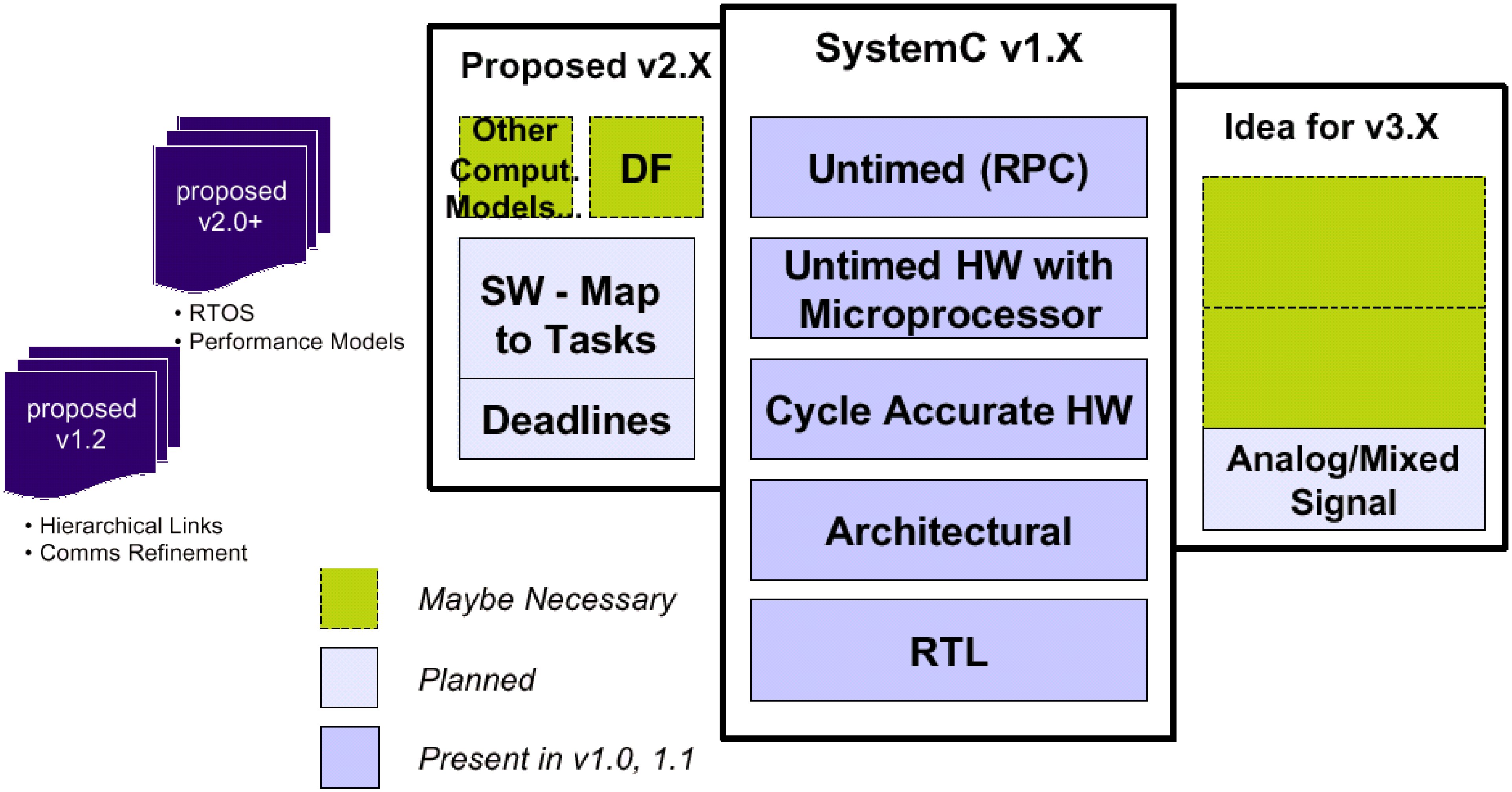
An Opportunity To Do It Right !

	1980s	1990s	2000s
System Level			1 SYSTEMC™
RTL Level		2 **Verilog** **VHDL**	
Gate Level	3 **Daisy** **Mentor** **Valid**		

SystemC Heritage



SystemC Roadmap



proposed v2.0+
- RTOS
- Performance Models

proposed v1.2
- Hierarchical Links
- Comms Refinement

Proposed v2.X
- Other Comput. Models...
- DF
- SW - Map to Tasks
- Deadlines

SystemC v1.X
- Untimed (RPC)
- Untimed HW with Microprocessor
- Cycle Accurate HW
- Architectural
- RTL

Idea for v3.X
- Analog/Mixed Signal

🟩 *Maybe Necessary*

□ *Planned*

▨ *Present in v1.0, 1.1*

The Intent of Different Level of Model

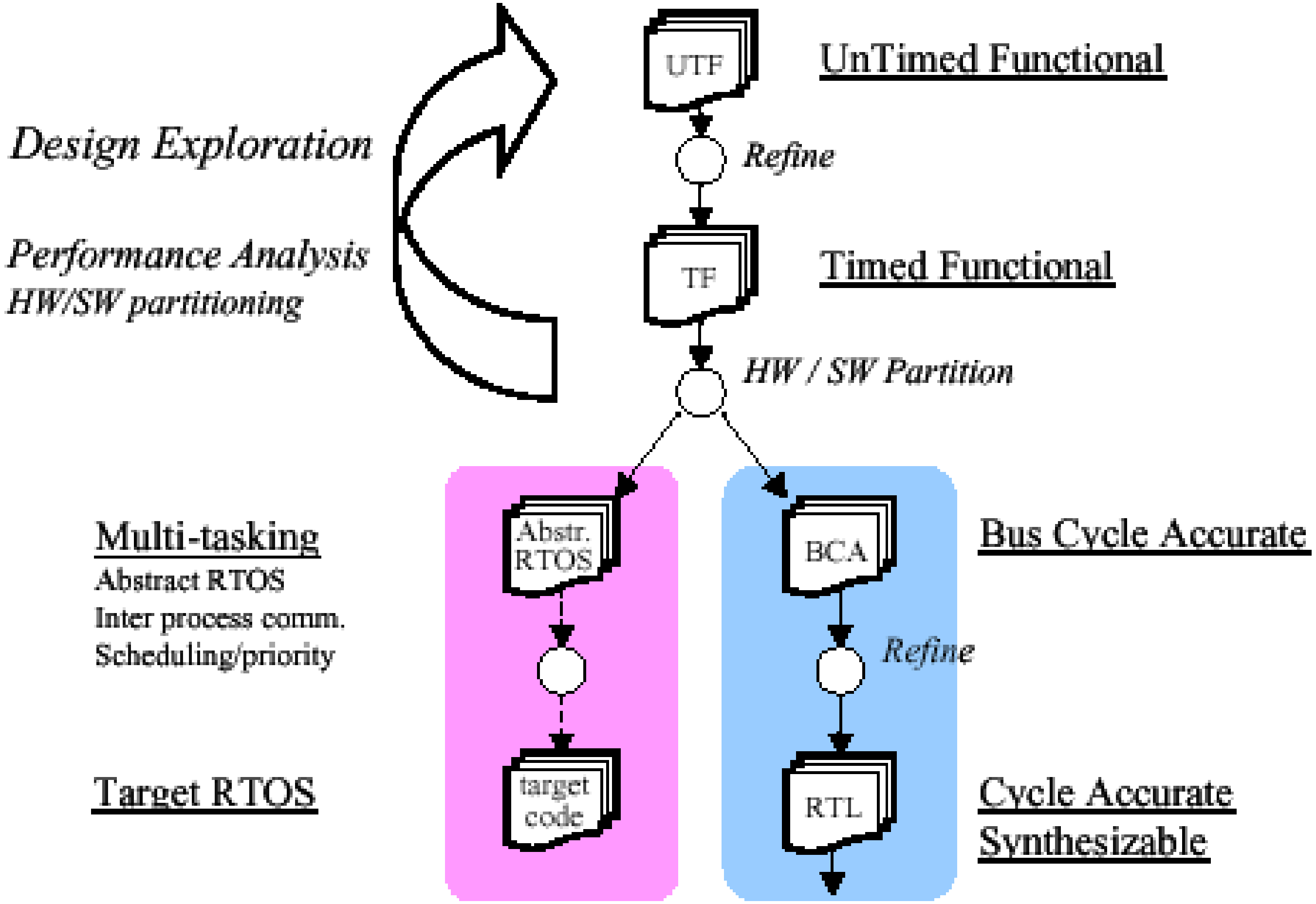
- Design exploration at higher level
  - Import of top-level constraint and block architecture
  - Hierarchical, complete system refinement
  - Less time for validating system requirement
  - More design space of algorithm and system architecture
- Simple and efficient verification and simulation
  -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

SystemC

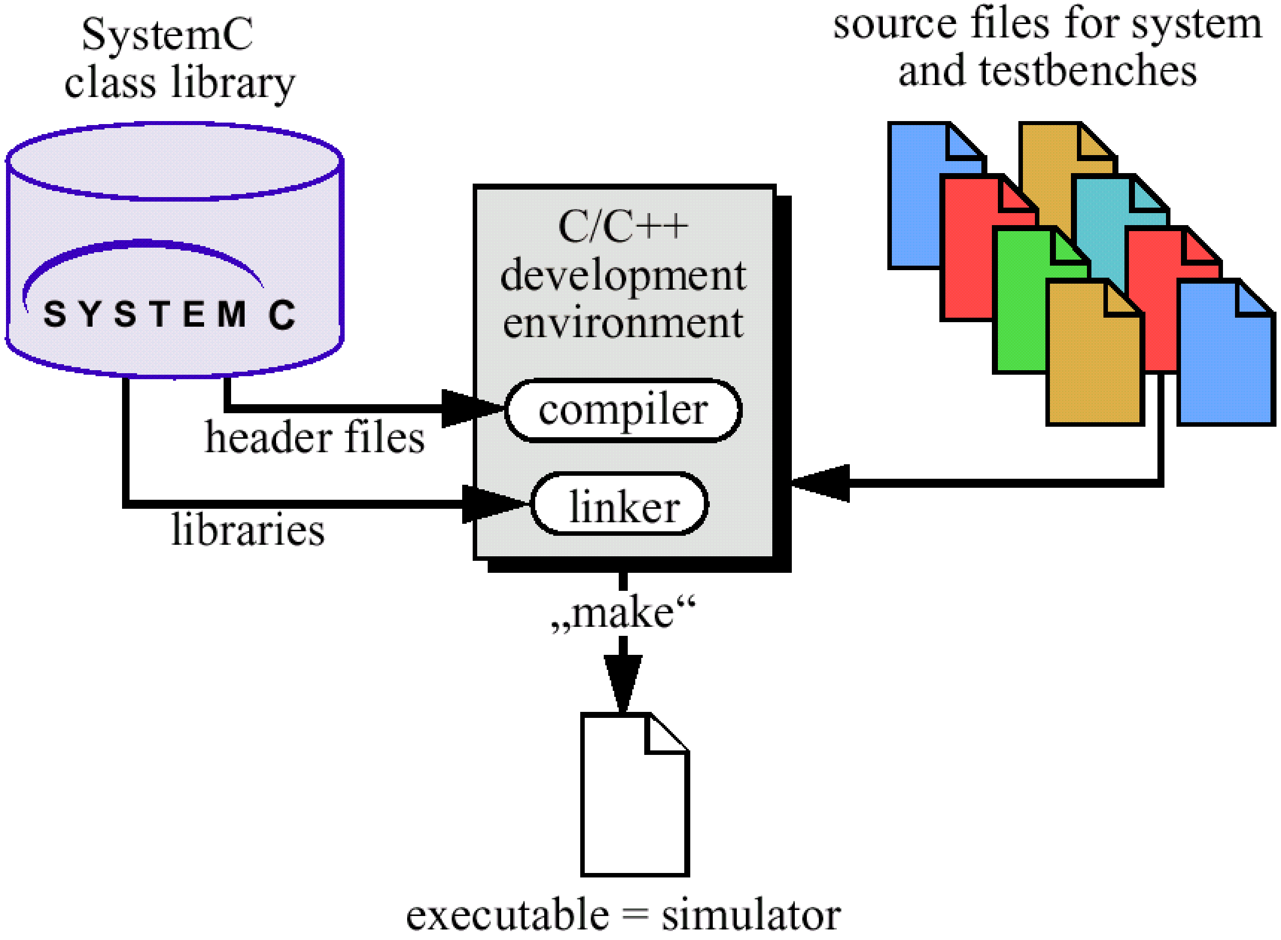
- SystemC is a modeling platform
  - C++ extensions to add hardware modeling constructs
  - a set C++ class library
  - simulation kernel
  - supports different levels of abstraction

Good Candidate for Task Level Mapping

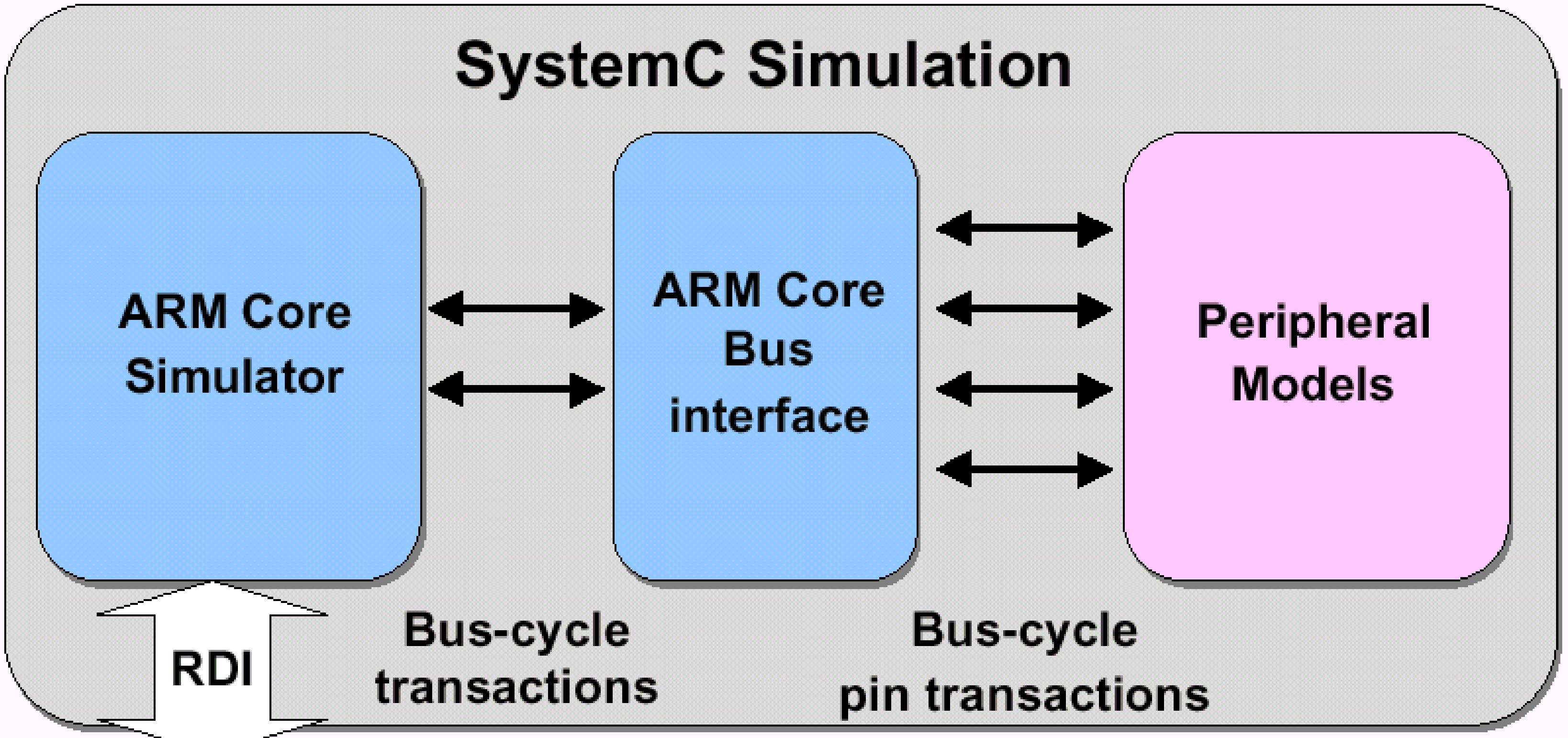
Level of abstraction in SystemC

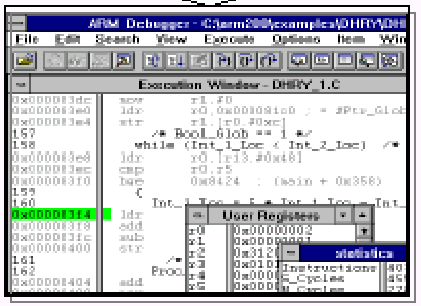


SystemC Design Flow



Example




ARM Debugger - C:\arm200\examples\DHRY\DH
File Edit Search View Execute Options Item Win
Execution Window - DHRY_1.C
User Registers
statistics

```
0x000083dc   mov    r1,#0
0x000083e0   ldr    r0,0x000081c0 ; = #Ptr_Glob
0x000083e4   str    r1,[r0,#0xc]
157              /* Bool_Glob == 1 */
158            while (Int_1_Loc < Int_2_Loc)  /*
0x000083e8   ldr    r0,[r13,#0x48]
0x000083ec   cmp    r0,r5
0x000083f0   bge    0x8424  ; (main + 0x358)
159            {
160              Int_1_Loc = 5 * Int_1_Loc - Int_
0x000083f4   ldr
0x000083f8   add
0x000083fc   sub
0x00008400   str
161              /*
162              Proc_
0x00008404   add
```

```
r0   0x00000002
r1   0x00000001
r2   0x312
r3   0x010
r4   0x000
r5   0x000
```

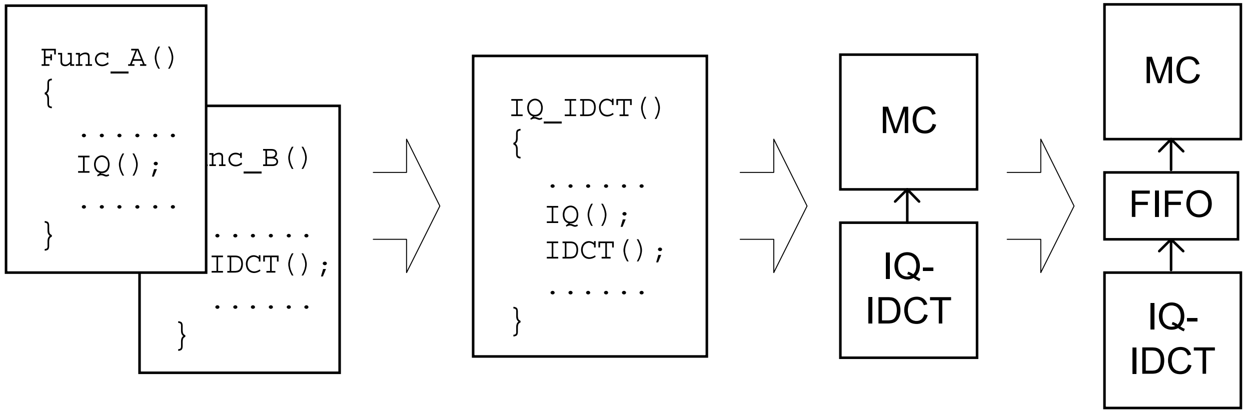
```
Instructions   40
S_Cycles       45
N_Cycles       27
```

Debugger

Source: SystemC Users Forum, DAC, June 2000

Implementing Virtual Prototypes

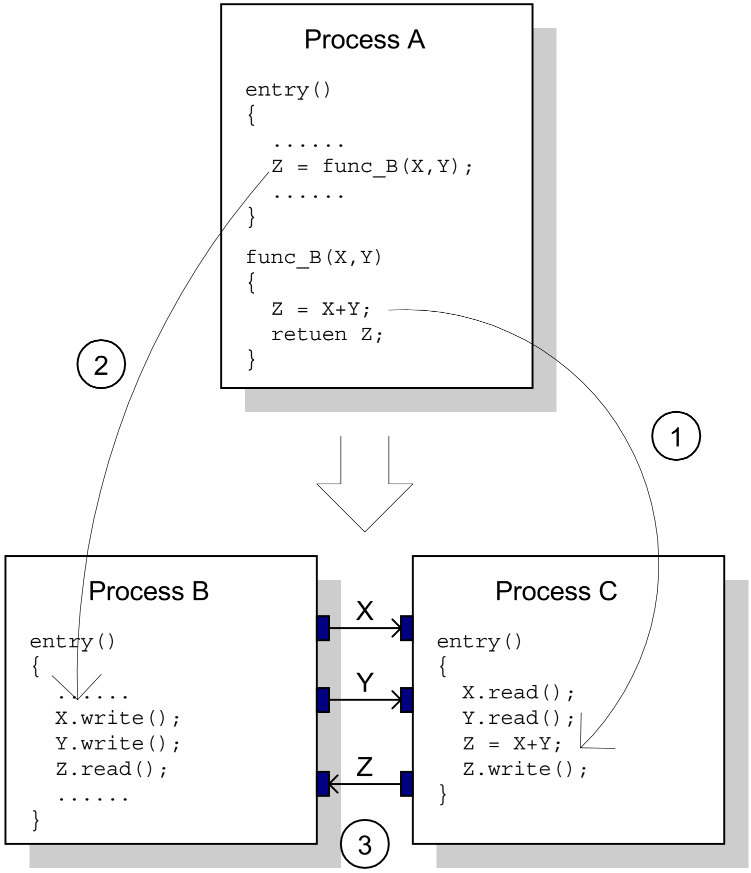
- Functionality partition
- Module specification
- Communication refinement



Functionality Partition

- Separating communication and computation
- Using hierarchy to group related functionality
- Choosing the granularity of the basic parts

Module Specification (1/2)



```
entry()
{
  ......
  Z = func_B(X,Y);
  ......
}

func_B(X,Y)
{
  Z = X+Y;
  retuen Z;
}
```

```
entry()
{
  ......
  X.write();
  Y.write();
  Z.read();
  ......
}
```

```
entry()
{
  X.read();
  Y.read();
  Z = X+Y;
  Z.write();
}
```

1. Pull out functionality into new created process
2. Replace function call with inter-process communcation.
3. Instantiate new process and define channels to connect them.

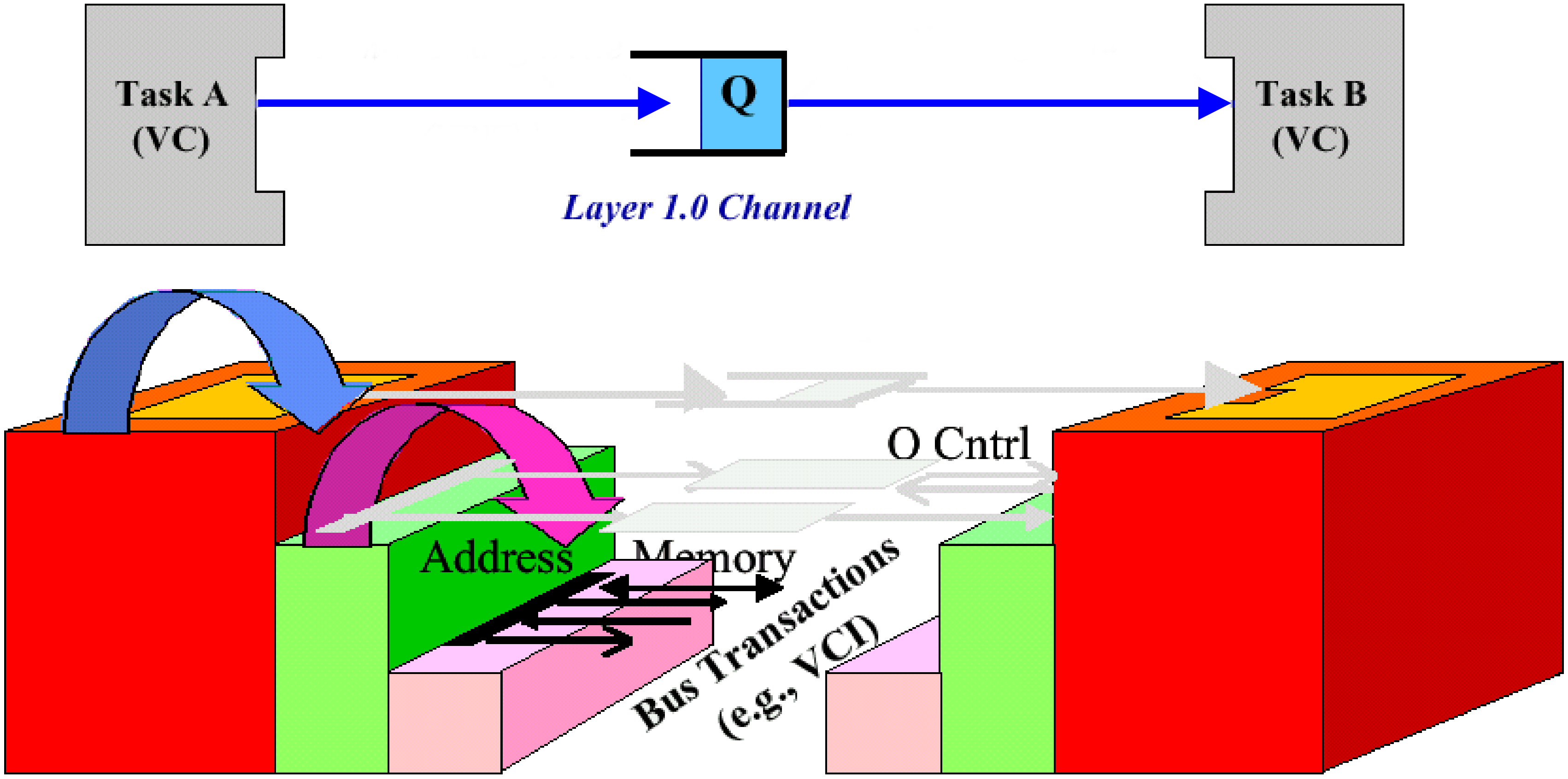
Module Specification (2/2)

- Abstraction Levels
  - Untimed Functional Level
    - Processes execute in zero time but in order
  - Timed Functional Level
  - Bus-Cycle Accurate Level
    - Transaction on bus are modeled cycle accurate
- Cycle Accurate Level
  - Behavior is clock cycle accurate

Communication Refinement

Key

Guarantee consistency of communication during refinement



Software Performance Estimation

- Have to take the following into account
  - Instruction set
  - Bus loading
  - Memory fetching
  - Register allocation
- Example: Cadence VCC technology
  - CPU characterized as Virtual Processor Model
  - Using a Virtual Machine Instruction Set
  - SW estimation using annotation into C-Code
  - Good for early system scheduling, processor load estimation
    - Two orders of magnitude faster than ISS
    - Greater than 80% accuracy

Discussion: Commonality and Differentia

- Differentiae
  - Processor core (e.g., customized inst. set)
  - IP parameterized
  - IP add/move
- Design methodology of platform
  - System-level
  - Platform-level design methodology
    - Design flow
    - Models
    - Tools (EDA venders, 3rd party or home-made)

Summary

- Platform-based design
  - From board design to SoC design
  - From executable spec., i.e., C/C++, to SystemC
- Modeling
  - Performance evaluation
  - Task mapping
  - Communication refinement