ARM SoC Platform

Shao-Yi Chien

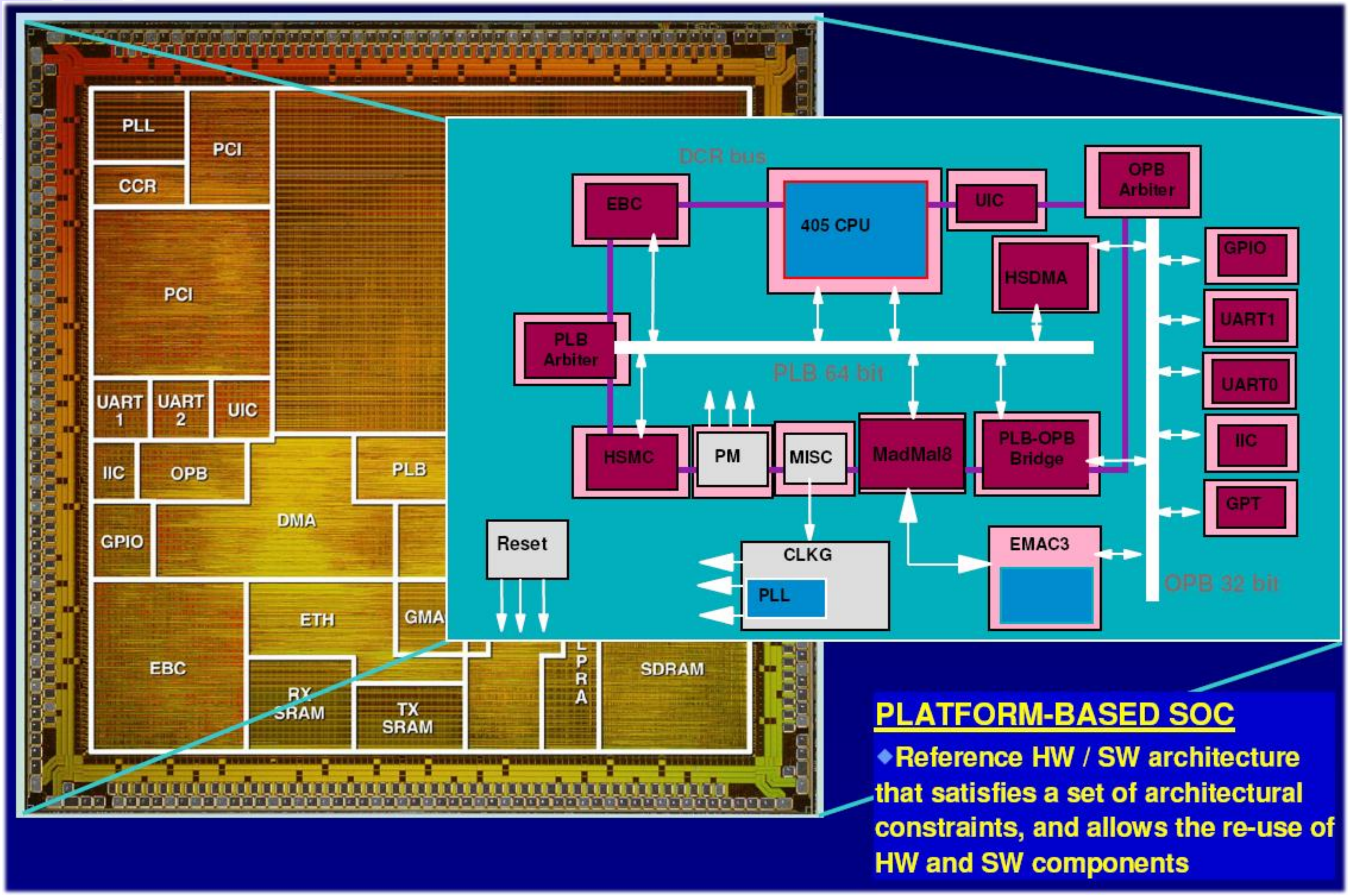
Outline

- Platform based design
- ARM processors
- Introduction to AMBA
- Communications between different IPs
- Introduction to AXI

Platform Based Design

- Precise definition of platform-based design
  - An organized method to reduce the time required and risk involved in designing and verifying a complex SoC, by heavy reuse of combinations of hardware and software IP. Rather than looking at IP reuse in a block by block manner, platform-based design aggregates groups of components into a reusable platform architecture.
- System platform
  - A coordinated family of hardware-software architectures, satisfying a set of architectural constraints that are imposed to allow the reuse of hardware and software components

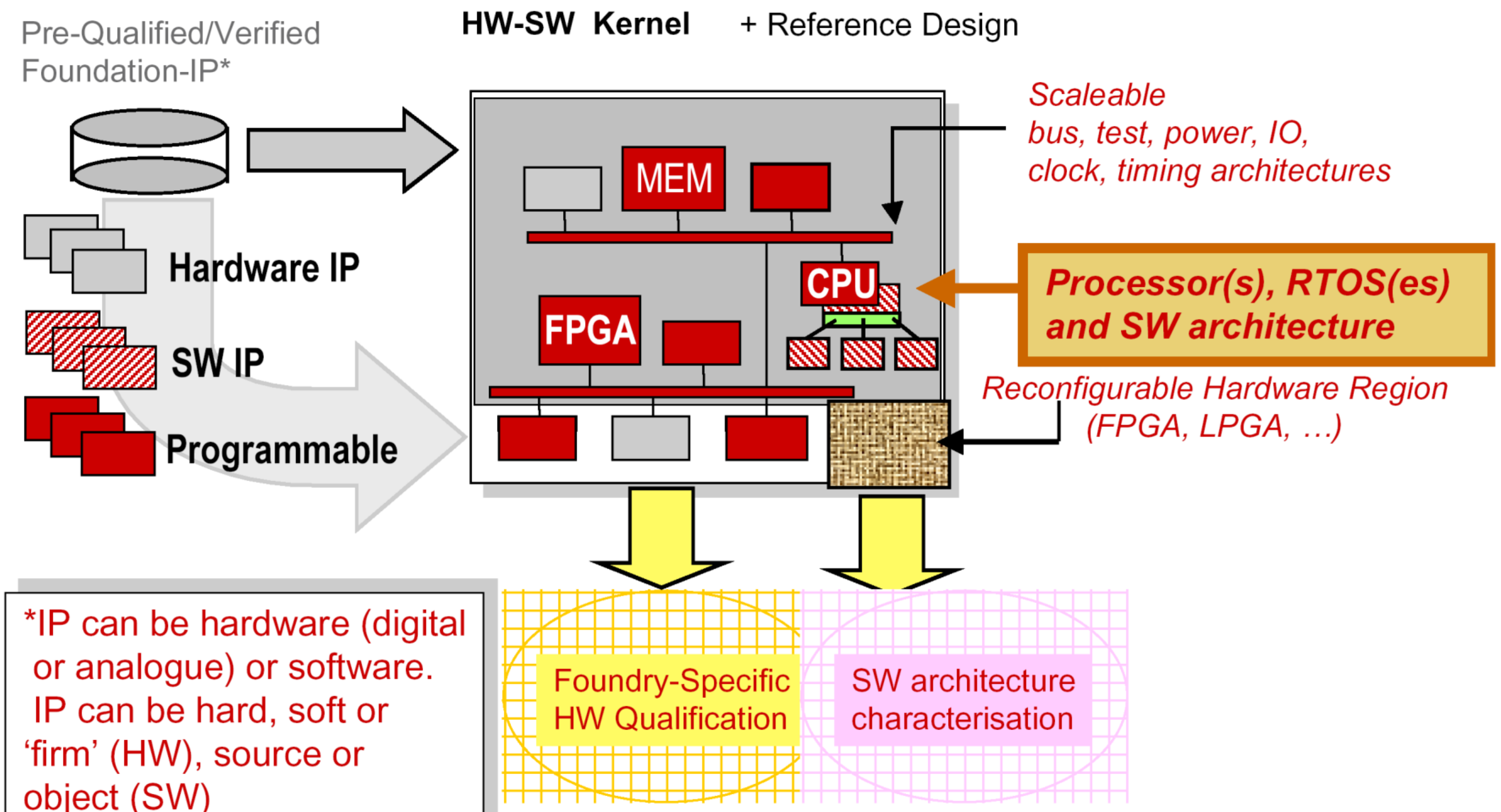
Platform Based Design



PLATFORM-BASED SOC

- Reference HW / SW architecture that satisfies a set of architectural constraints, and allows the re-use of HW and SW components

A Hardware-centric View of a Platform

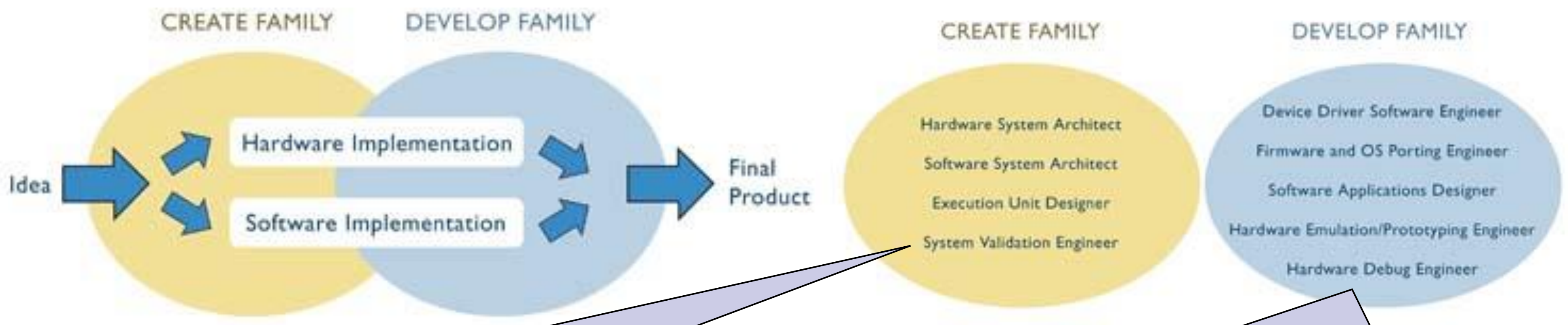

Pre-Qualified/Verified Foundation-IP*
HW-SW Kernel
+ Reference Design
Scaleable bus, test, power, IO, clock, timing architectures
MEM
CPU
FPGA
Hardware IP
SW IP
Programmable
Processor(s), RTOS(es) and SW architecture
Reconfigurable Hardware Region (FPGA, LPGA, …)
Foundry-Specific HW Qualification
SW architecture characterisation

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

Source: Grant Martin and Henry Chang, ISQED 2002 Tutorial

ARM – SoC Platform Provider

- Integrate all system design tools and platform in ARM realview family (old system)


CREATE FAMILY
DEVELOP FAMILY
Idea
Hardware Implementation
Software Implementation
Final Product
CREATE FAMILY
Hardware System Architect
Software System Architect
Execution Unit Designer
System Validation Engineer
DEVELOP FAMILY
Device Driver Software Engineer
Firmware and OS Porting Engineer
Software Applications Designer
Hardware Emulation/Prototyping Engineer
Hardware Debug Engineer

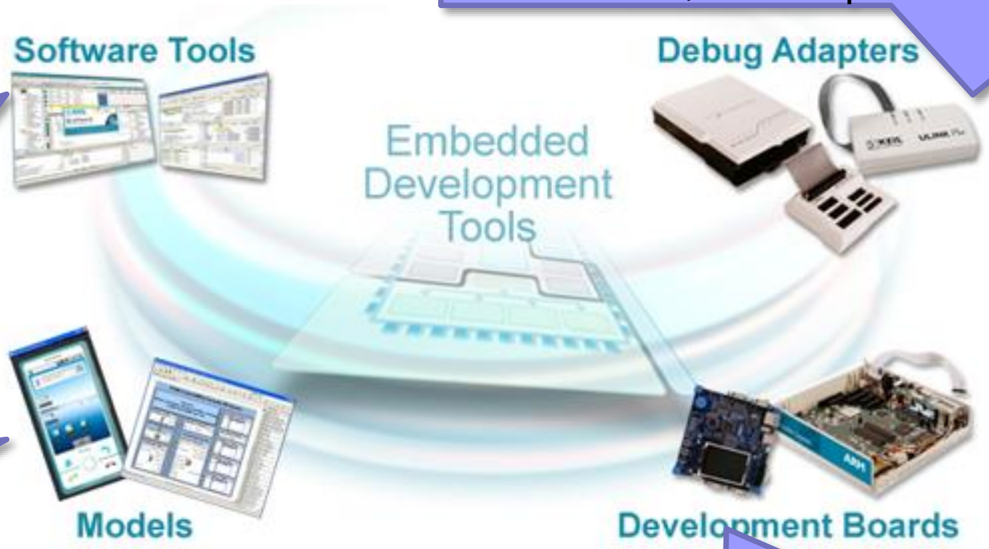
• SoC Designer with MaxSim Technology
~~• Core Generator with MaxCore Technology~~
• Model Library for SoC Designer

• RealView Developer Suite (RVDS)
• RealView ICE
• RealView Trace
• Hardware Platforms

ARM – SoC Platform Provider

- Development Studio 5 (DS-5)
- MDK-ARM Microcontroller Development Kit
- RVDS

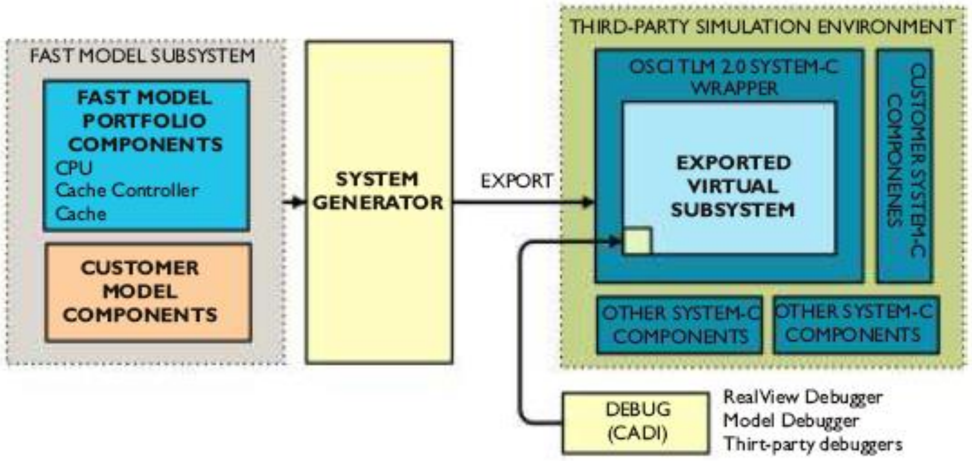
- DSTREAM
- RVI
- RVT2
- VSTREAM
- Keil ULINK, ULINK pro

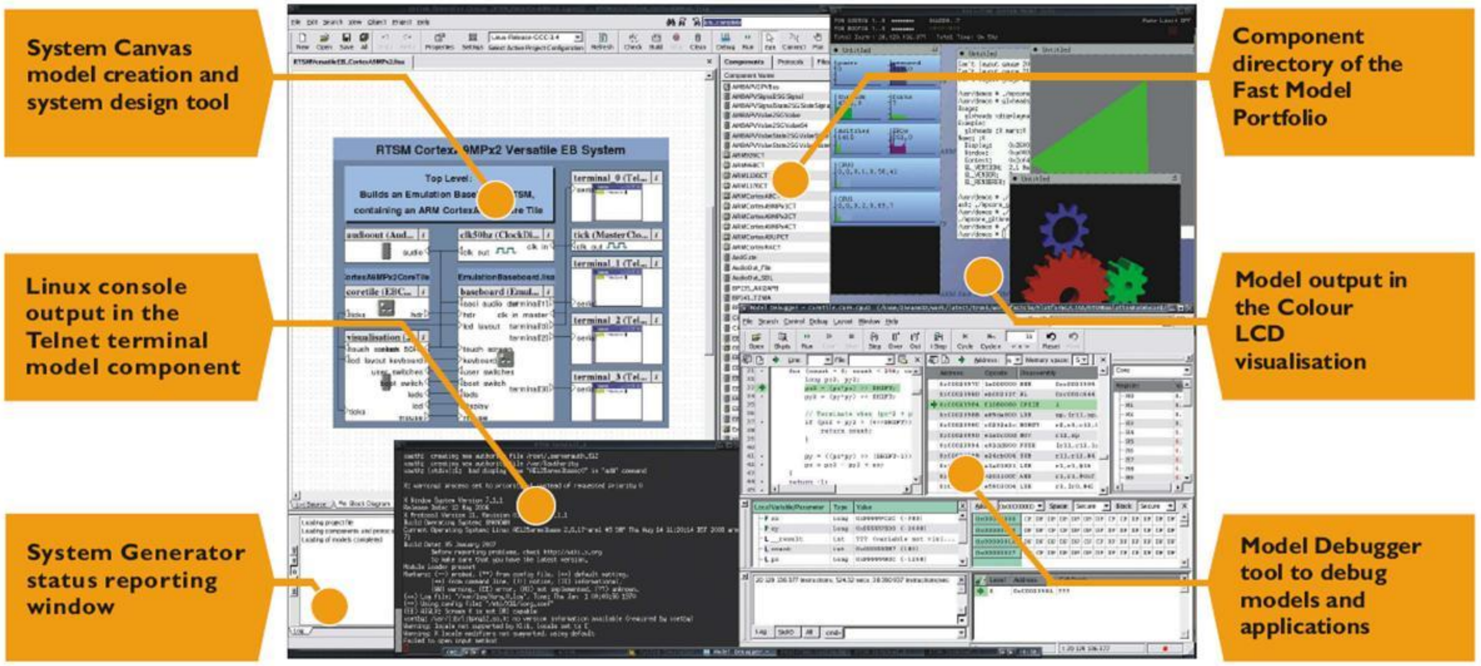


- Fast Model
- Real-Time System Model

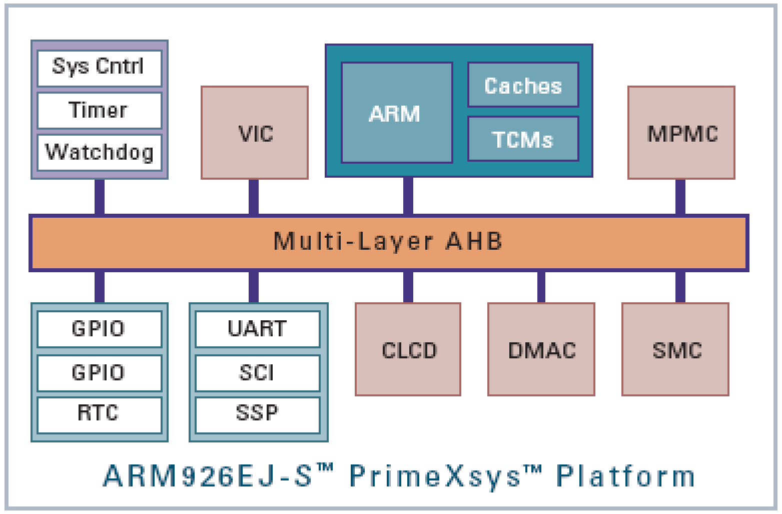
- Versatile™ Express
- Versatile Platform Baseboard
- Keil MPS board
- Keil evaluation boards

Fast Model


FAST MODEL SUBSYSTEM
FAST MODEL PORTFOLIO COMPONENTS
CPU
Cache Controller
Cache
CUSTOMER MODEL COMPONENTS
SYSTEM GENERATOR
EXPORT
THIRD-PARTY SIMULATION ENVIRONMENT
OSCI TLM 2.0 SYSTEM-C WRAPPER
EXPORTED VIRTUAL SUBSYSTEM
CUSTOMER SYSTEM-C COMPONENES
OTHER SYSTEM-C COMPONENTS
OTHER SYSTEM-C COMPONENTS
DEBUG (CADI)
RealView Debugger
Model Debugger
Thirt-party debuggers


System Canvas model creation and system design tool
Linux console output in the Telnet terminal model component
System Generator status reporting window
Component directory of the Fast Model Portfolio
Model output in the Colour LCD visualisation
Model Debugger tool to debug models and applications
RTSM Cortex-A9MPx2 Versatile EB System

ARM926EJ-S PrimeXsys Platform



- **Hardware**
  - **ARM**
  - **AMBA**
- Software
  - OS and RTOS
  - Software development model (SDM)
  - Instruction set simulator (ISS)
- Development tools
  - RealView developer suite
  - RealView ICE and Trace
  - Hardware platform

ARM

Ref:

1. Slides of “IP Core Design,” National Chiao-Tung University.
2. Slides of “SoC Lab,” MOE S&IP Consortium.
3. S.Furber, *ARM System-on-Chip Architecture*, Addison Wesley, 2000.
4. http://www.arm.com

Outline

- Overview
- ARM core family
- Introduction to several ARM processors

Outline

- Overview
- ARM core family
- Introduction to several ARM processors

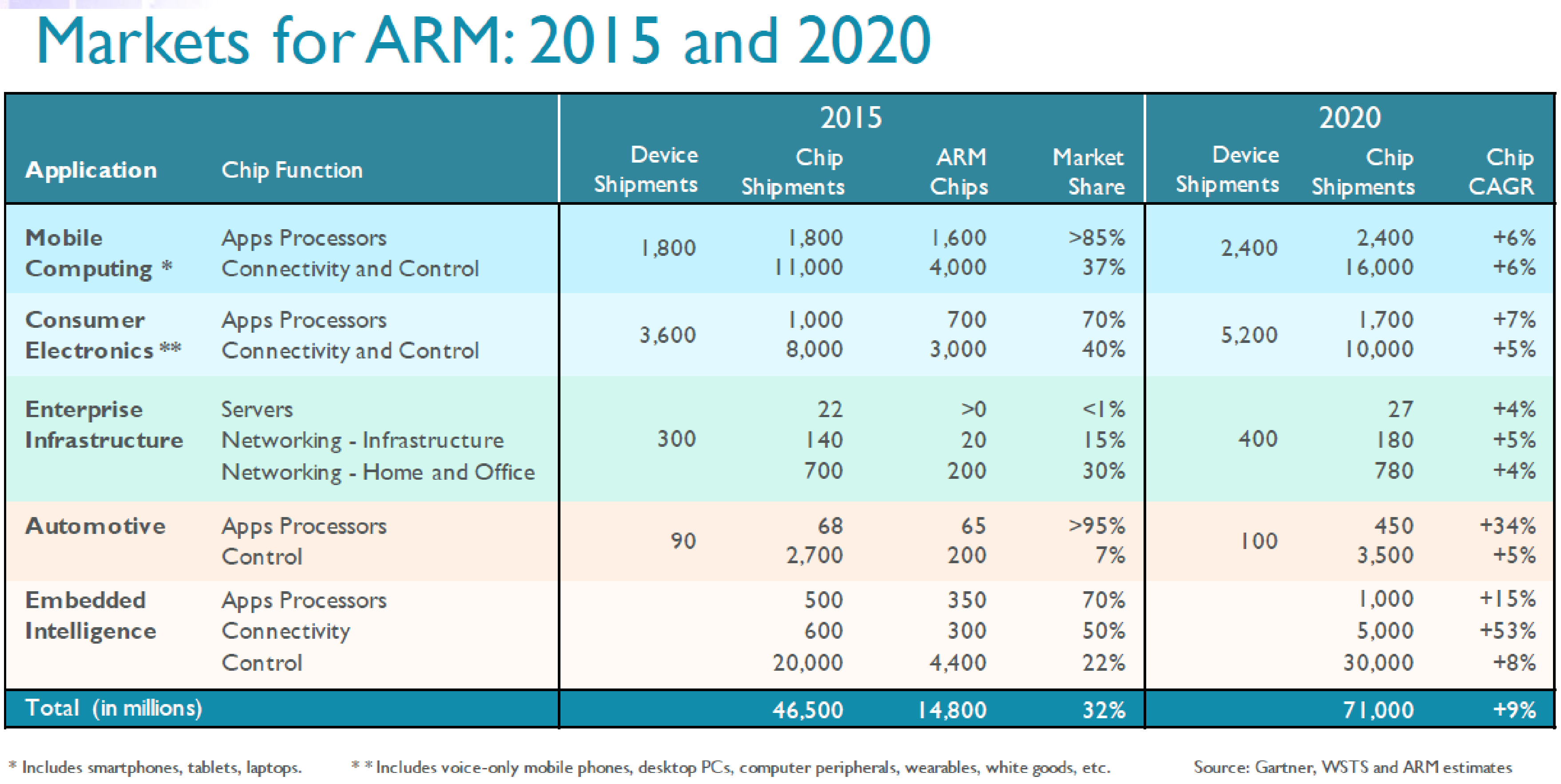
ARM Ltd

- ARM was originally developed at Acorn Computer Limited, of Cambridge, England between 1983 and 1985.
  - 1980, RISC (Reduced Instruction Set Computer) concept at Stanford and Berkeley universities.
  - First RISC processor for commercial use
- 1990 Nov, ARM Ltd was founded
- ARM cores
  - Licensed to partners who fabricate and sell to customers.
- Technologies assist to design in the ARM application
  - Software tools, boards, debug hardware, application software, bus architectures, peripherals etc…
- Modification of the acronym expansion to **Advanced RISC Machine**.

ARM Ltd

- ARM is the industry's leading provider of 32-bit embedded RISC microprocessors with almost **75%** of the market!
- ARM expects 50% mobile chip market share by 2015
- 95% in smartphone market
- 80% in digital camera market
- 40% in TV market

ARM Ltd



Markets for ARM: 2015 and 2020

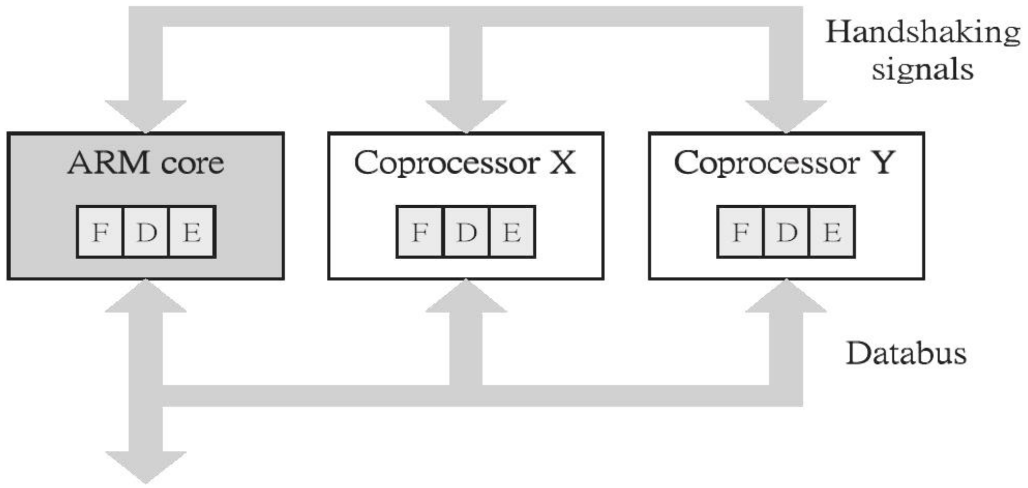
		2015				2020		
Application	Chip Function	Device Shipments	Chip Shipments	ARM Chips	Market Share	Device Shipments	Chip Shipments	Chip CAGR
Mobile Computing *	Apps Processors	1,800	1,800	1,600	>85%	2,400	2,400	+6%
	Connectivity and Control		11,000	4,000	37%		16,000	+6%
Consumer Electronics **	Apps Processors	3,600	1,000	700	70%	5,200	1,700	+7%
	Connectivity and Control		8,000	3,000	40%		10,000	+5%
Enterprise Infrastructure	Servers	300	22	>0	<1%	400	27	+4%
	Networking - Infrastructure		140	20	15%		180	+5%
	Networking - Home and Office		700	200	30%		780	+4%
Automotive	Apps Processors	90	68	65	>95%	100	450	+34%
	Control		2,700	200	7%		3,500	+5%
Embedded Intelligence	Apps Processors		500	350	70%		1,000	+15%
	Connectivity		600	300	50%		5,000	+53%
	Control		20,000	4,400	22%		30,000	+8%
Total (in millions)			46,500	14,800	32%		71,000	+9%

* Includes smartphones, tablets, laptops. * * Includes voice-only mobile phones, desktop PCs, computer peripherals, wearables, white goods, etc. Source: Gartner, WSTS and ARM estimates

Features of the ARM Instruction Set

- Load-store architecture
  - Process values which are in registers
  - Load, store instructions for memory data accesses
- 3-address data processing instructions
- Conditional execution of every instruction
- Load and store multiple registers
- Shift, ALU operation in a single instruction
- Open instruction set extension through the coprocessor instruction
- Very dense 16-bit compressed instruction set (Thumb)

Coprocessors



- Up to *16* coprocessors can be defined
- Expands the ARM instruction set
- Each coprocessor can have up to 16 private registers of any reasonable size
- Load-store architecture

Thumb

- Thumb is a 16-bit instruction set
  - Optimized for code density from C code
  - Improved performance form narrow memory
  - Subset of the functionality of the ARM instruction set
- Core has two execution states – ARM and Thumb
  - Switch between them using ***BX*** instruction
- Thumb has characteristic features:
  - Most Thumb instruction are executed unconditionally
  - Many Thumb data process instruction use a 2-address format
  - Thumb instruction formats are less regular than ARM instruction formats, as a result of the dense encoding.

I/O System

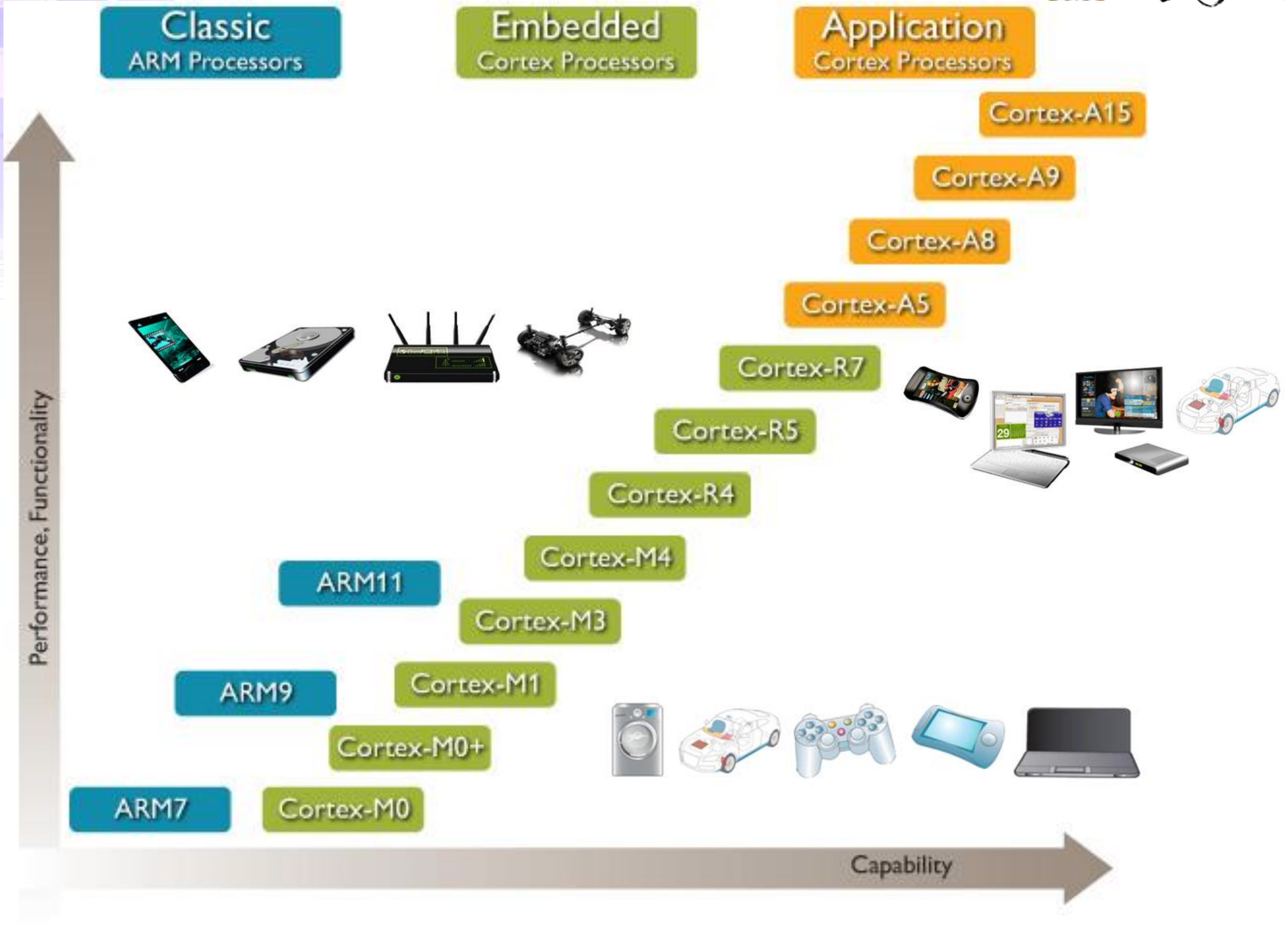
- ARM handles input/output peripherals as ***memory-mapped*** with interrupt support
- Internal registers in I/O devices as addressable locations with ARM's memory map read and written using load-store instructions
- Interrupt by normal interrupt (*IRQ*) or fast interrupt (*FIQ*)
- Input signals are *level-sensitive* and *maskable*
- May include Direct Memory Access (DMA) hardware

Outline

- Overview
- ARM core family
- Introduction to several ARM processors

ARM Core Family

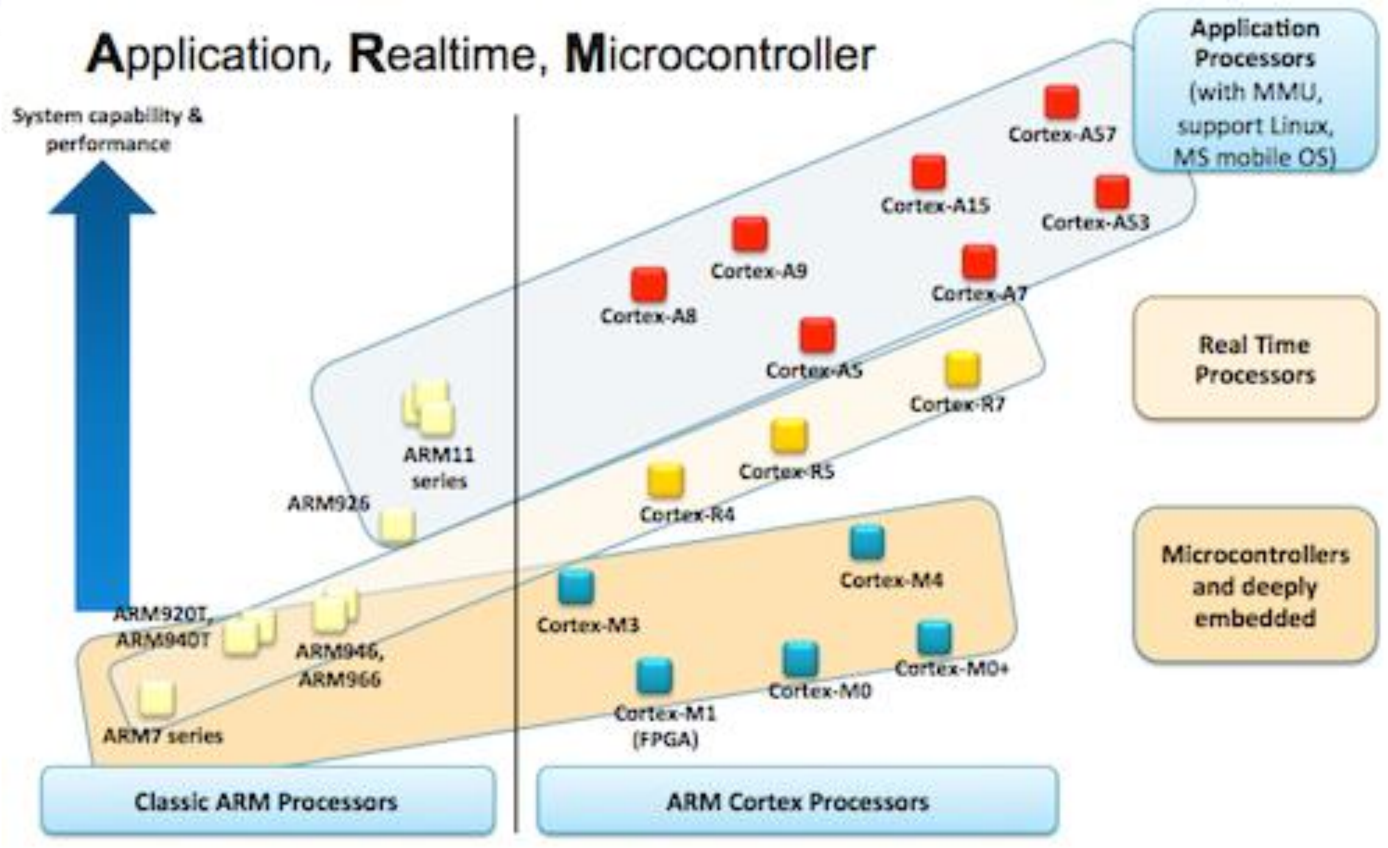
Application Cores	Embedded Cores	Secure Cores
ARM Cortex-A53, A55, A57,A65AE, A72, A73, A75, A76AE, A76	ARM Cortex-M23, M33	SecurCore SC100
ARM Cortex-A32, A35	ARM Cortex-M10	SecurCore SC110
ARM Cortex-A12, A15, A17	ARM Cortex-M0, M1, M3, M4, M7	SecurCore SC200
ARM Cortex-A9 MPCore	ARM Cortex-R4(F), R5, R7	SecurCore SC210
ARM Cortex-A9 Single Core	ARM1156T2(F)-S	SecurCore SC300
ARM Cortex-A5, A7, A8	ARM7EJ-S	
ARM11 MPCore	ARM7TDMI	
ARM1136J(F)-S	ARM7TDMI-S	
ARM1176JZ(F)-S	ARM946E-S	
ARM720T	ARM966E-S	
ARM920T	ARM968E-S	
ARM922T	ARM996HS	
ARM926EJ-S		



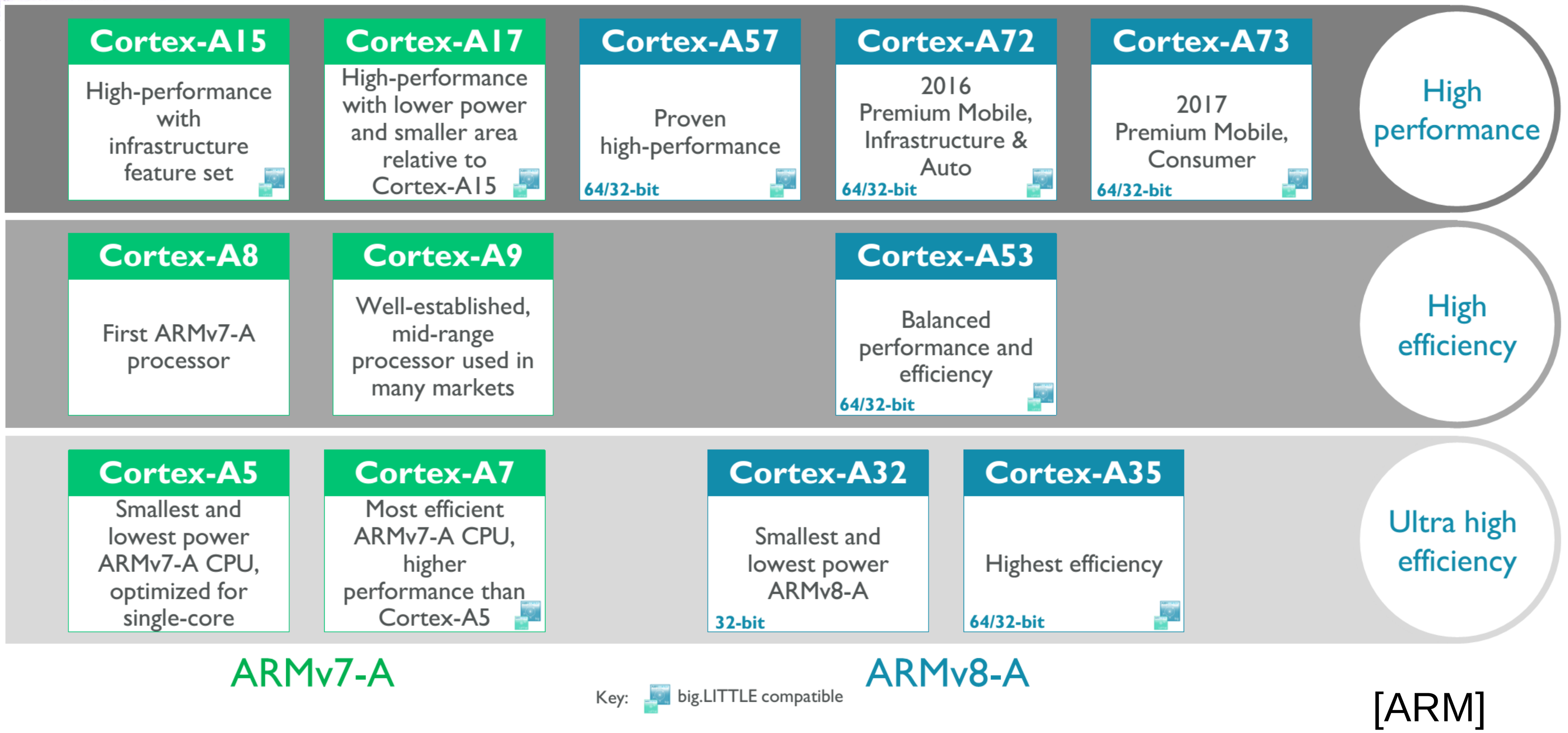
Classic
ARM Processors
Embedded
Cortex Processors
Application
Cortex Processors
Cortex-A15
Cortex-A9
Cortex-A8
Cortex-A5
Cortex-R7
Cortex-R5
Cortex-R4
Cortex-M4
ARM11
Cortex-M3
ARM9
Cortex-M1
Cortex-M0+
ARM7
Cortex-M0
Performance, Functionality
Capability

[ARM]

ARM Core Family: 3 Categories



Cortex-A



Cortex-A15 High-performance with infrastructure feature set ▨	**Cortex-A17** High-performance with lower power and smaller area relative to Cortex-A15 ▨	**Cortex-A57** Proven high-performance 64/32-bit ▨	**Cortex-A72** 2016 Premium Mobile, Infrastructure & Auto 64/32-bit ▨	**Cortex-A73** 2017 Premium Mobile, Consumer 64/32-bit ▨	High performance
Cortex-A8 First ARMv7-A processor	**Cortex-A9** Well-established, mid-range processor used in many markets		**Cortex-A53** Balanced performance and efficiency 64/32-bit ▨		High efficiency
Cortex-A5 Smallest and lowest power ARMv7-A CPU, optimized for single-core	**Cortex-A7** Most efficient ARMv7-A CPU, higher performance than Cortex-A5 ▨	**Cortex-A32** Smallest and lowest power ARMv8-A 32-bit	**Cortex-A35** Highest efficiency 64/32-bit ▨		Ultra high efficiency

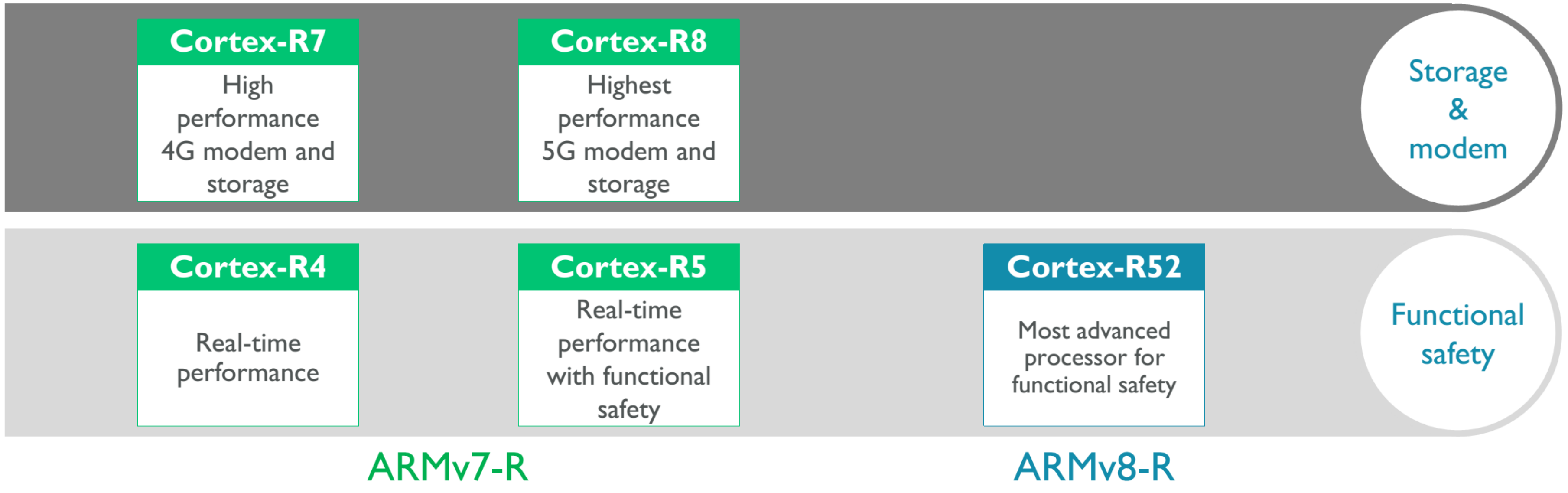
ARMv7-A

ARMv8-A

Key: ▨ big.LITTLE compatible

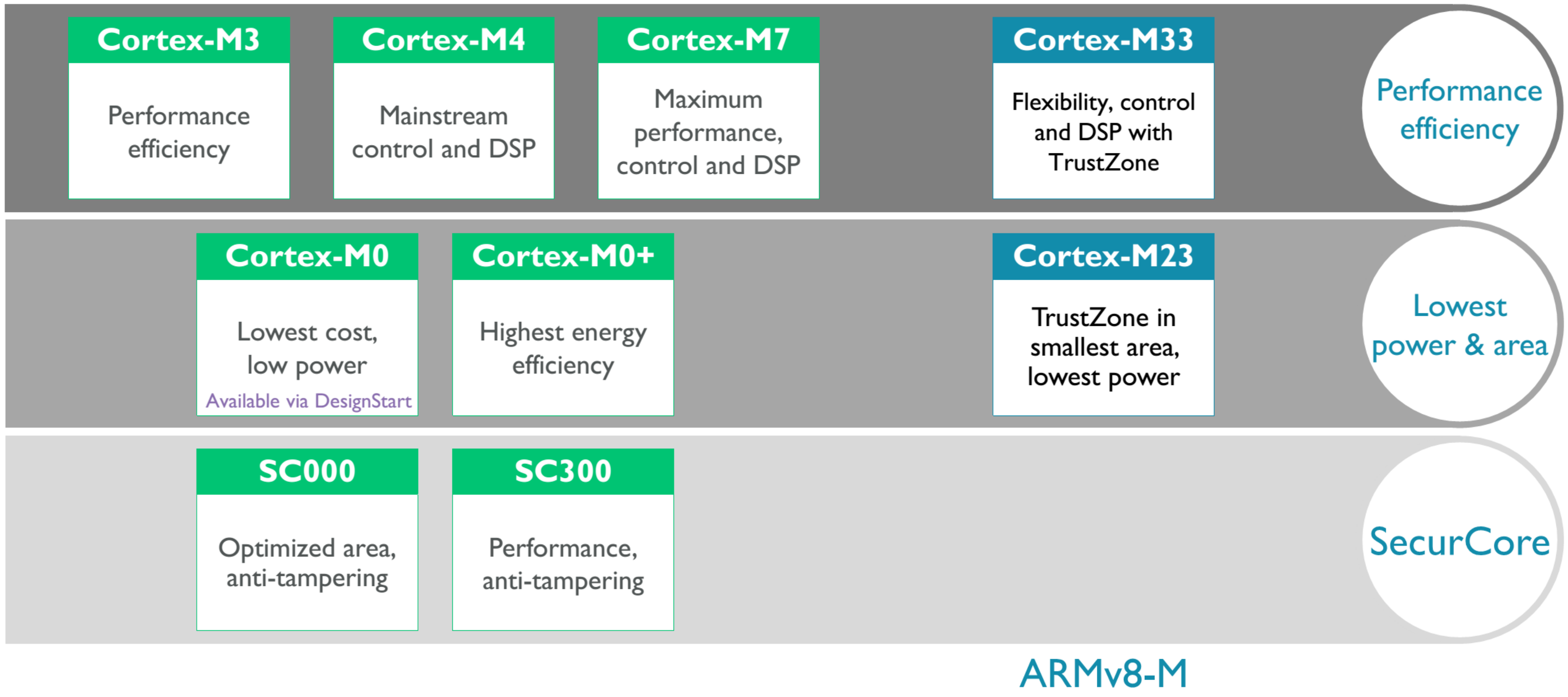
[ARM]

Cortex-R


Cortex-R7
High performance 4G modem and storage
Cortex-R8
Highest performance 5G modem and storage
Storage & modem
Cortex-R4
Real-time performance
Cortex-R5
Real-time performance with functional safety
Cortex-R52
Most advanced processor for functional safety
Functional safety
ARMv7-R
ARMv8-R

[ARM]

Cortex-M



Cortex-M3
Performance efficiency
Cortex-M4
Mainstream control and DSP
Cortex-M7
Maximum performance, control and DSP
Cortex-M33
Flexibility, control and DSP with TrustZone
Performance efficiency
Cortex-M0
Lowest cost, low power
Available via DesignStart
Cortex-M0+
Highest energy efficiency
Cortex-M23
TrustZone in smallest area, lowest power
Lowest power & area
SC000
Optimized area, anti-tampering
SC300
Performance, anti-tampering
SecurCore
ARMv8-M

[ARM]

Product Code Demystified

- T: Thumb
- D: On-chip Debug support
- M: Enhanced Multiplier
- I: Embedded ICE hardware
- T2: Thumb-2
- S: synthesizable code
- E: Enhanced DSP instruction set
- J: JAVA support, Jazelle
- Z: TrustZone
- F: Floating point unit
- H: Handshake, clockless design for synchronous or asynchronous design

ARM Processor Cores (1/4)

- ARM processor core + cache + MMU
  → ARM CPU cores
- ARM6 → ARM7
  - 3-stage pipeline
  - Keep its instructions and data in the same memory system
  - **T**humb 16-bit compressed instruction set
  - on-chip **D**ebug support, enabling the processor to halt in response to a debug request
  - enhanced **M**ultiplier, 64-bit result
  - Embedded **I**CE hardware, give on-chip breakpoint and watchpoint support

ARM Processor Cores (2/4)

- ARM8 → ARM9
  → ARM10
- ARM9
  - 5-stage pipeline (130 MHz or 200MHz)
  - Using separate instruction and data memory ports
- ARM 10 (1998. Oct.)
  - High performance, 300 MHz
  - Multimedia digital consumer applications
  - Optional vector floating-point unit

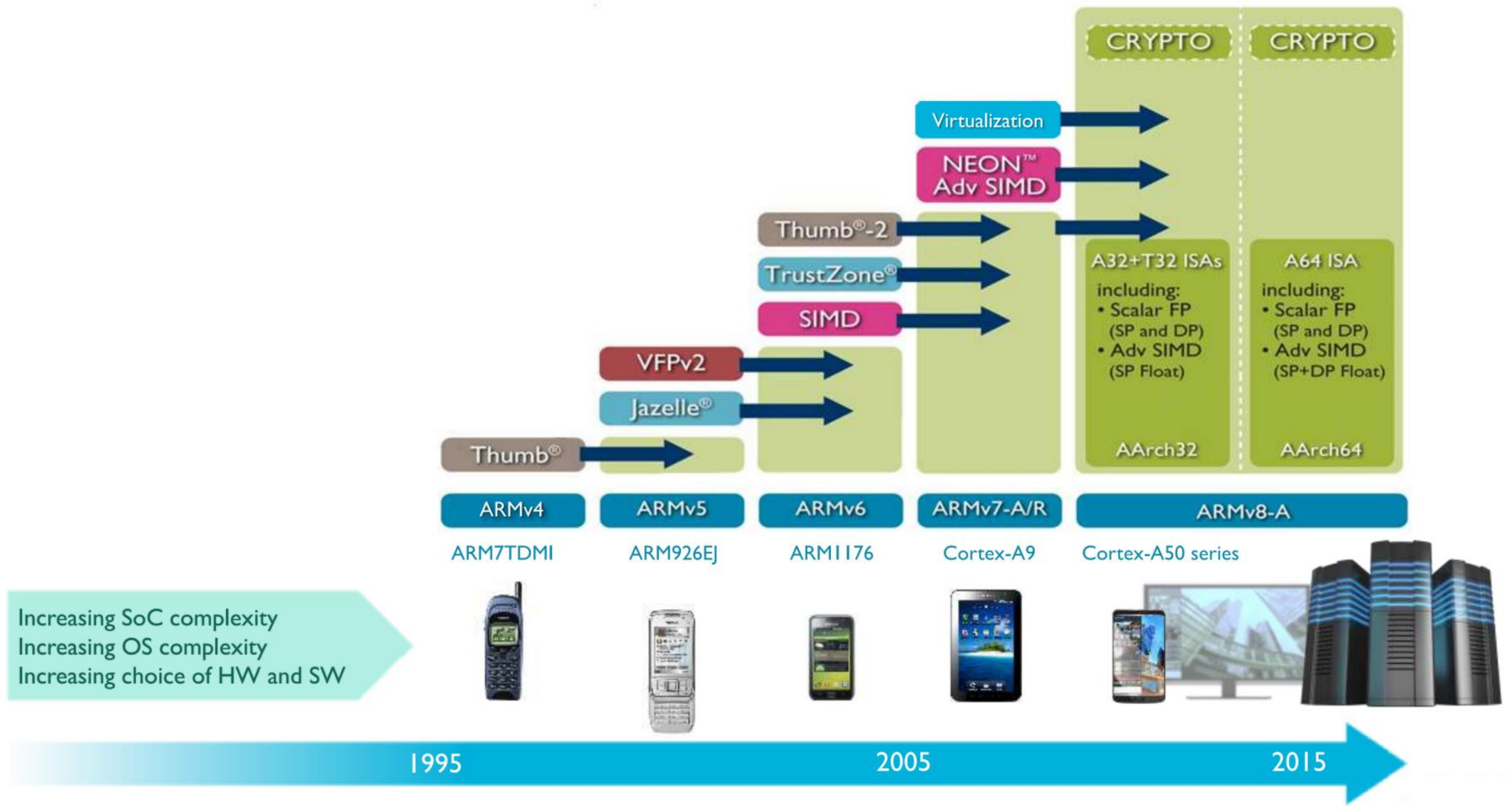
ARM Processor Cores (3/4)

- ARM11 (2002 Q4)
  - 8-stage pipeline
  - Addresses a broad range of applications in the wireless, consumer, networking and automotive segments
  - Support media accelerating extension instructions
  - Can achieve 1GHz
  - Support AXI
- SecurCore Family
  - Smart card and secure IC development

ARM Processor Cores (4/4)

- Cortex Family
  - Provides a large range of solutions optimized around specific market applications across the full performance spectrum
  - ARM Cortex-A Series, applications processors for complex OS and user applications.
    - Supports the ARM, Thumb and Thumb-2 instruction sets
  - ARM Cortex-R Series, embedded processors for real-time systems.
    - Supports the ARM, Thumb, and Thumb-2 instruction sets
  - ARM Cortex-M Series, deeply embedded processors optimized for cost sensitive applications.
    - Supports the Thumb-2 instruction set only

ARM Architecture



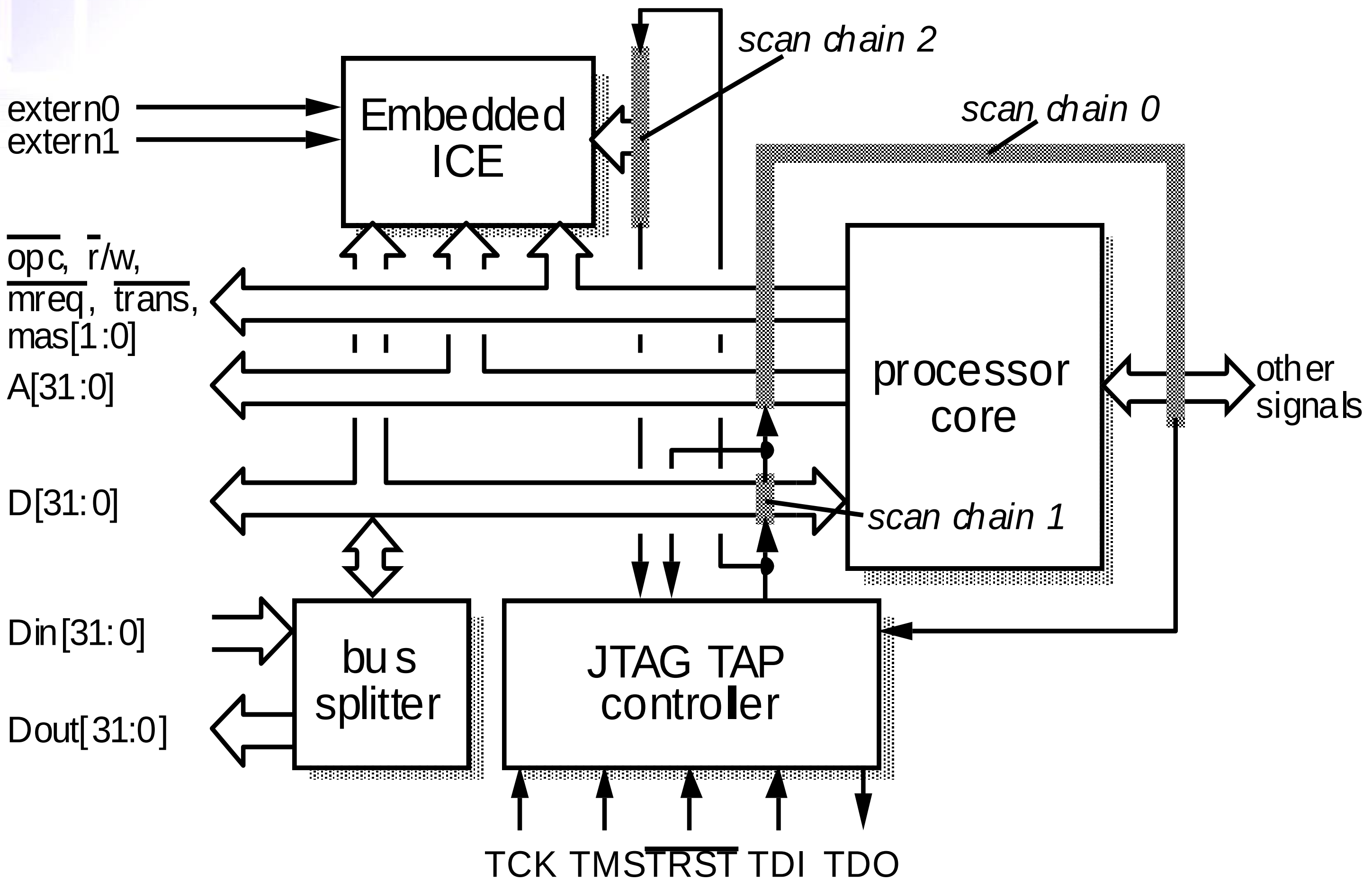
Outline

- Overview
- ARM core family
- Introduction to several ARM processors

ARM7TDMI Processor Core

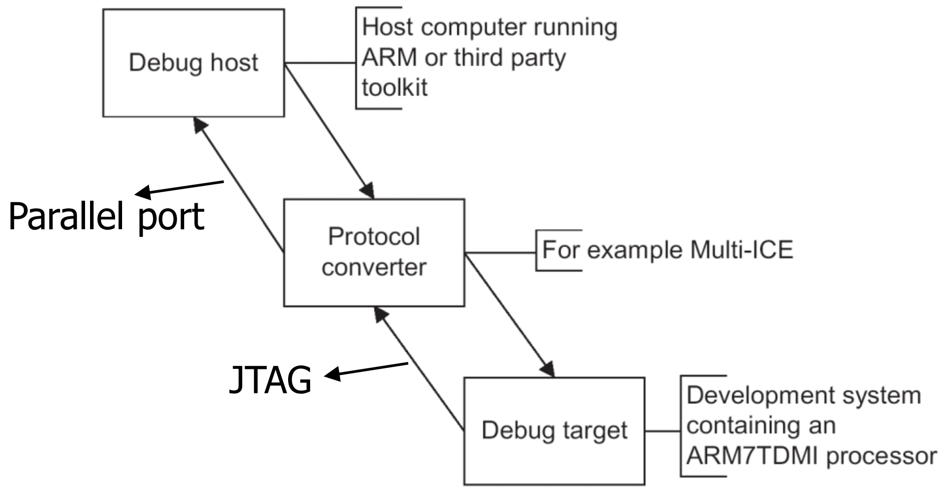
- Low-end ARM core for applications like digital mobile phones
- TDMI
  - **T**: Thumb, 16-bit compressed instruction set
  - **D**: on-chip Debug support, enabling the processor to halt in response to a debug request
  - **M**: enhanced Multiplier, yield a full 64-bit result, high performance
  - **I**: Embedded ICE hardware
- Von Neumann architecture
- 3-stage pipeline, CPI ~ 1.9

ARM7TDMI Block Diagram

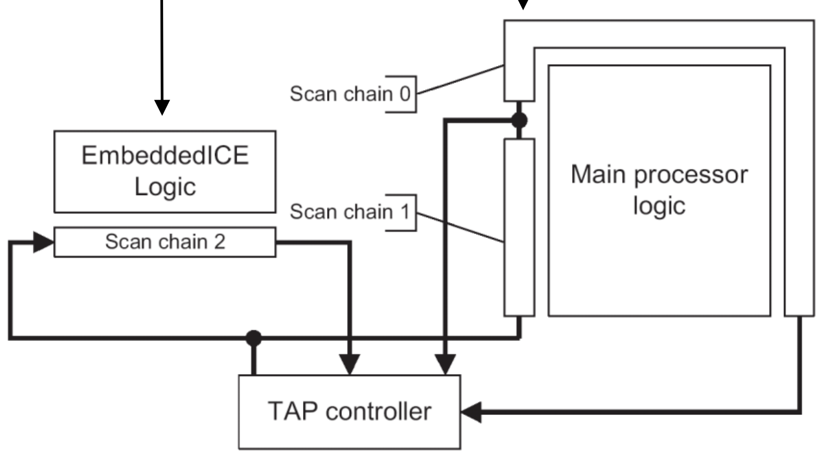


Debug Support

- D: debug extension
  - Internal content observation
  - External content observation
- I: EmbeddedICE logic extension
  - Breakpoint and watchpoint detection


Debug host
Host computer running ARM or third party toolkit
Parallel port
Protocol converter
For example Multi-ICE
JTAG
Debug target
Development system containing an ARM7TDMI processor

Typical debug architecture



Hardware support for debug

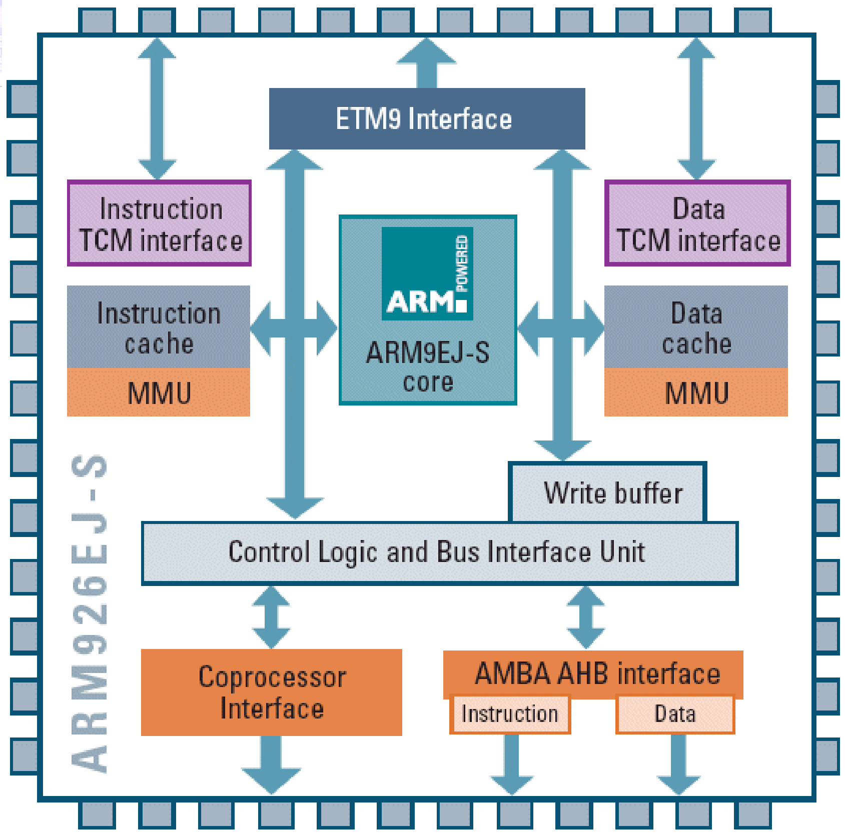
ARM7TDMI Performance Characteristics

	0.13um	0.18um
Area with cache (mm^2)	-	-
Area w/o cache (mm^2)	0.26	0.53
Frequency (MHz)	133	100-80
Typical mW/MHz with cache	-	-
Typical mW/MHz w/o cache	0.06	0.23

ARM9TDMI

- Harvard architecture
  - Increases available memory bandwidth
    - Instruction memory interface
    - Data memory interface
  - Simultaneous accesses to instruction and data memory can be achieved
- 5-stage pipeline
- Changes implemented to
  - Improve CPI to ~1.5
  - Improve maximum clock frequency

ARM926EJ-S



- ARMv5TEJ architecture (ARMv5TEJ)
- 32-bit ARM instruction and 16-bit Thumb instruction set
- DSP instruction extensions and single cycle MAC
- ARM Jazelle technology
- MMU which supports operating systems including Symbian OS, Windows CE, Linux
- Flexible instruction and data cache sizes
- Instruction and data TCM interfaces with wait state support
- EmbeddedICE-RT logic for real-time debug
- Industry standard AMBA bus AHB interfaces
- ETM interface for Real-time trace capability with ETM9
- Optional MOVE Coprocessor delivers video encoding performance

ARM926EJ-S Performance Characteristics

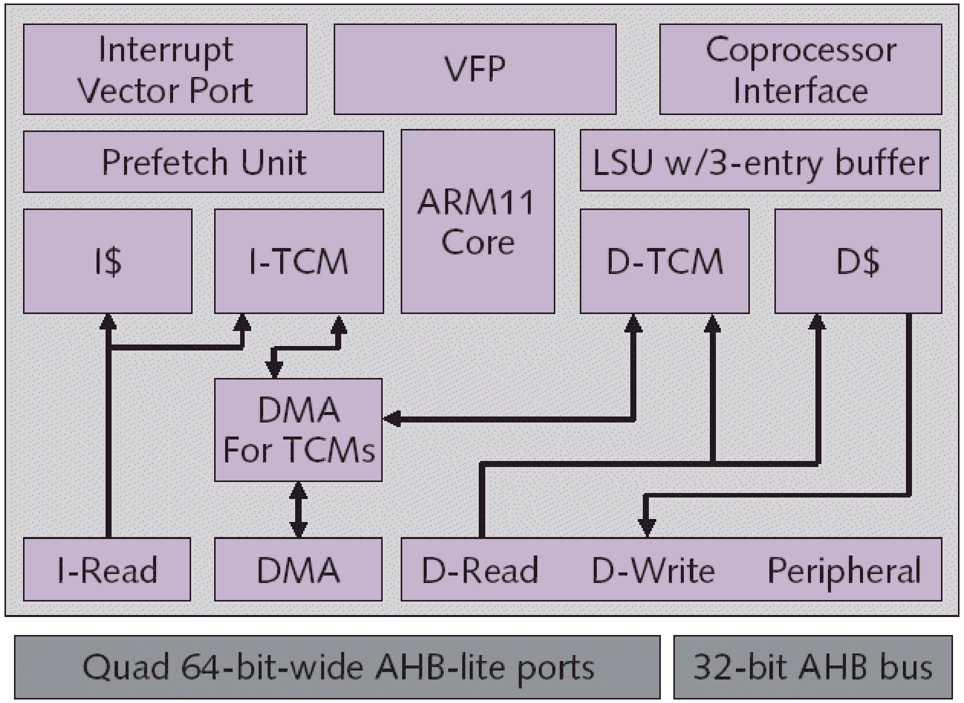
	0.13um	0.18um
Area with cache (mm^2)	3.2	8.3
Area w/o cache (mm^2)	1.68	4.0
Frequency (MHz)	266	200-180
Typical mW/MHz with cache	0.45	1.40
Typical mW/MHz w/o cache	0.30	1.00

ARM1176JZ(F)-S

- Powerful ARMv6 instruction set architecture
  - Thumb, Jazelle, DSP extensions
  - SIMD (Single Instruction Multiple Data) media processing extensions deliver up to 2x performance for video processing
- Energy-saving power-down modes
  - Reduce static leakage currents when processor is not in use
- High performance integer processor
  - 8-stage integer pipeline delivers high clock frequency
  - Separate load-store and arithmetic pipelines
  - Branch Prediction and Return Stack
  - Up to 660 Dhrystone 2.1 MIPS in 0.13µ process
- High performance memory system
  - Supports 4-64k cache sizes
  - Optional tightly coupled memories with DMA for multi-media applications
  - **Multi-ported AMBA 2.0 AHB bus interface speeds instruction and data access**
  - ARMv6 memory system architecture accelerates OS context-switch

ARM1176JZ(F)-S

- Vectored interrupt interface and low-interrupt-latency mode speeds interrupt response and real-time performance
- Optional Vector Floating Point coprocessor (ARM1136JF-S)
  - Powerful acceleration for embedded 3D-graphics



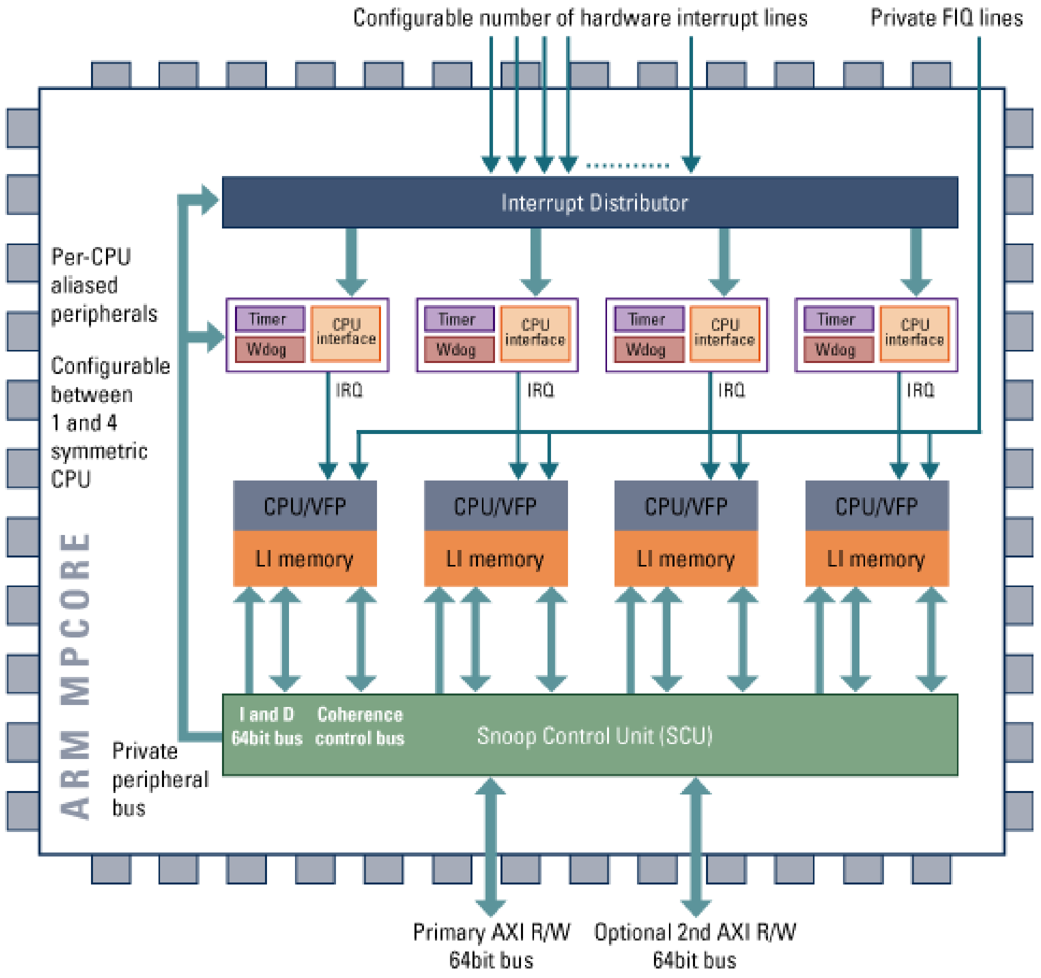
ARM1176JZ(F)-S Performance Characteristics

	0.13um	0.18um
Area with cache (mm^2)	5.55	-
Area w/o cache (mm^2)	2.85	-
Frequency (MHz)	333-550	-
Typical mW/MHz with cache	0.8	-
Typical mW/MHz w/o cache	0.6	-

ARM11 MPCore

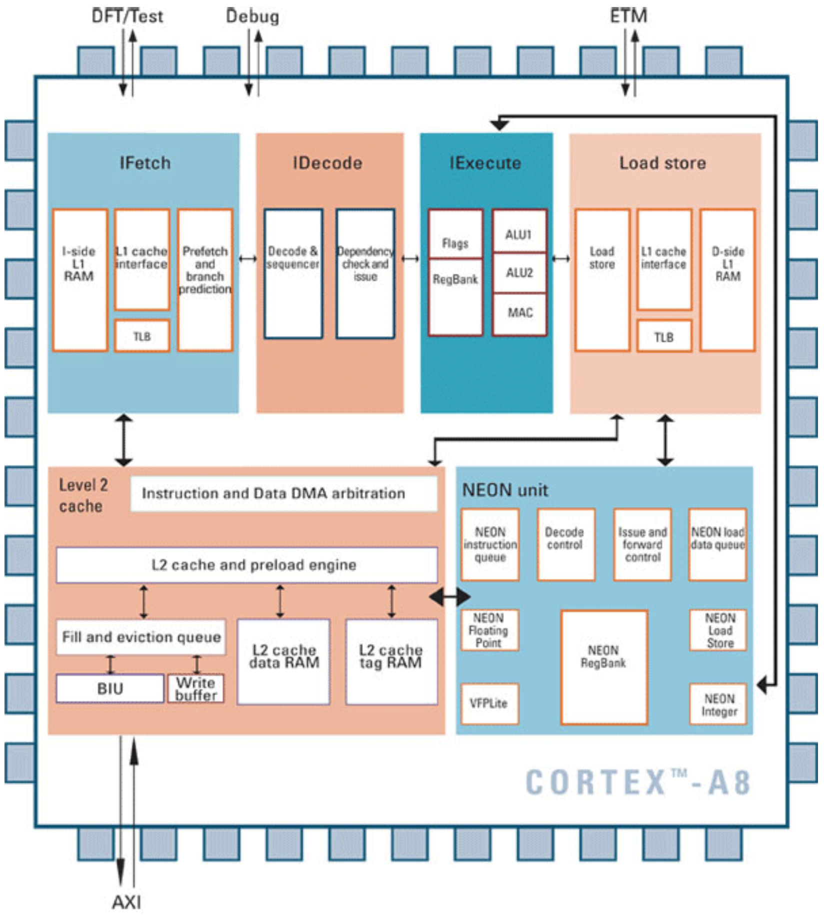
- Highly configurable
  - **Flexibility of total available performance from implementations using between 1 and 4 processors.**
  - Sizing of both data and instruction cache between 16K and 64K bytes across each processor.
  - Either dual or single 64-bit AMBA 3 AXI system bus connection allowing rapid and flexibility during SoC design
  - Optional integrated vector floating point (VFP) unit
  - Sizing on the number of hardware interrupts up to a total of 255 independent sources

ARM11 MPCore



ARM Cortex-A8

- Used for applications including mobile phones, set-up boxes, gaming consoles, and automotive navigation/entertainment systems
- High performance with low power consumption

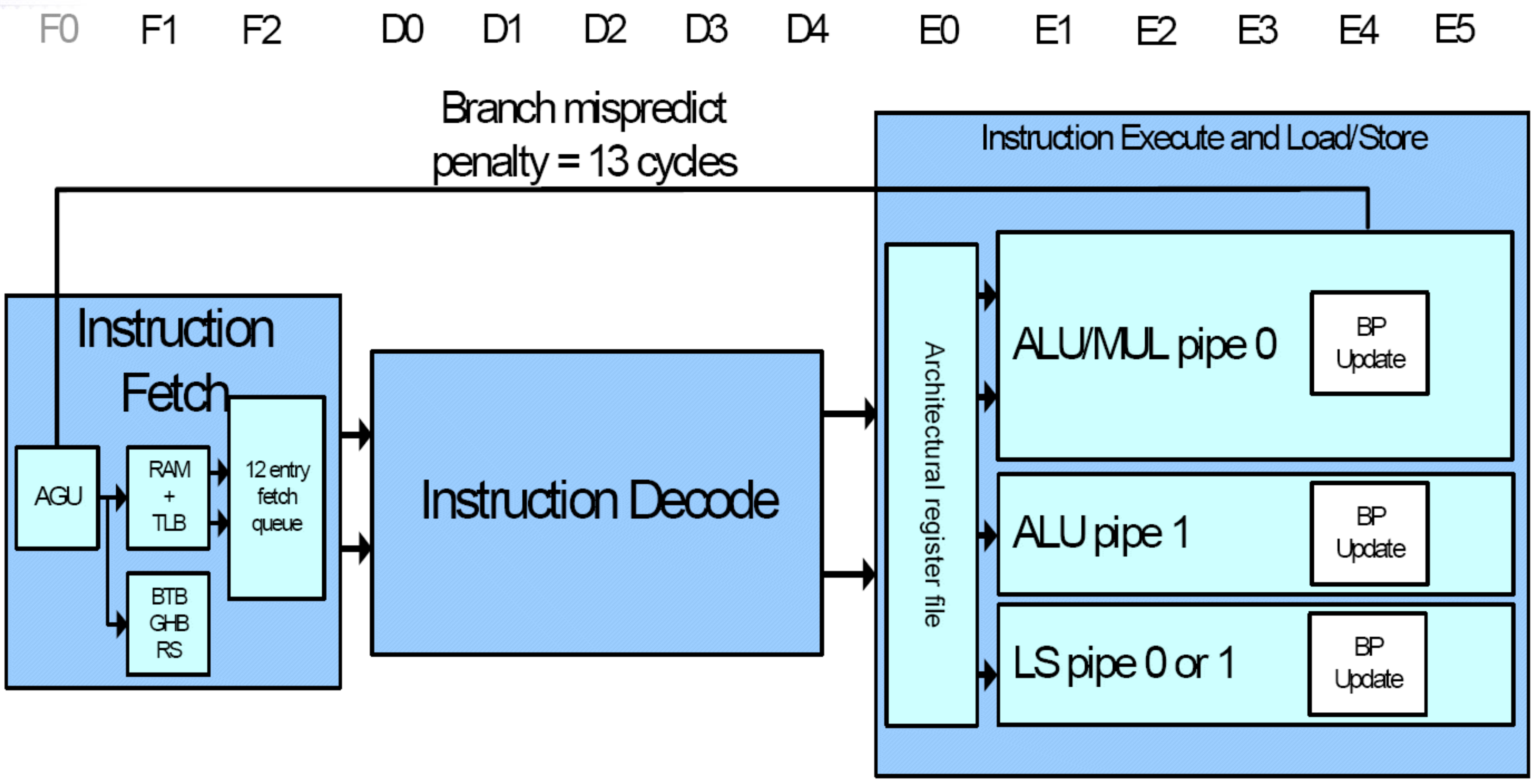


ARM Cortex-A8

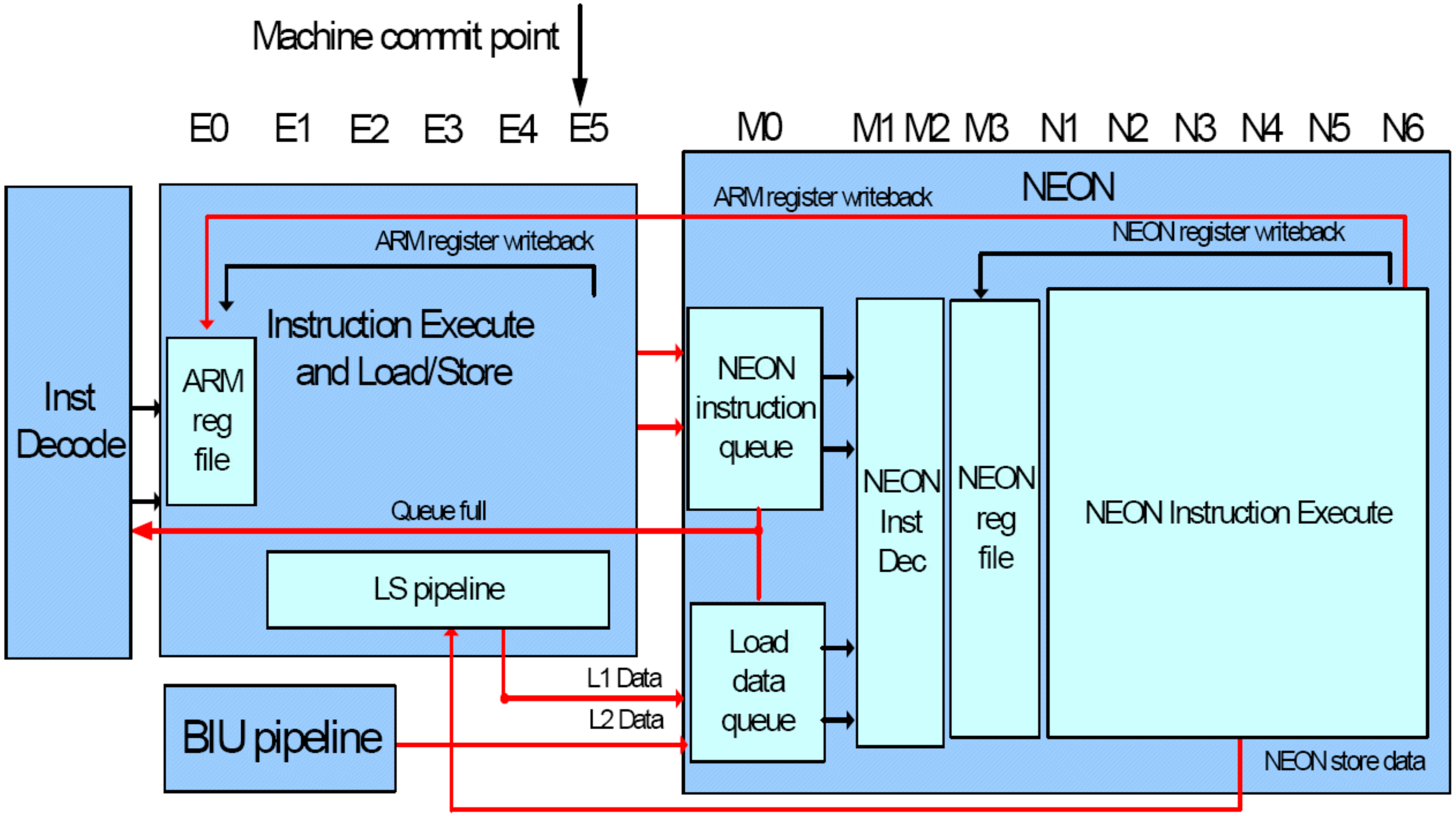
- Architecture features
  - Thumb-2 instruction
    - Add 130 additional instructions to Thumb
    - High density, high performance
  - NEON media and signal processing technology
    - For audio, video, and 3D graphics
    - Decode MPEG-4 VGA 30fps @ 275MHz and H.264 video @ 350MHz
  - TrustZone technology
  - VFPv3

ARM Cortex-A8

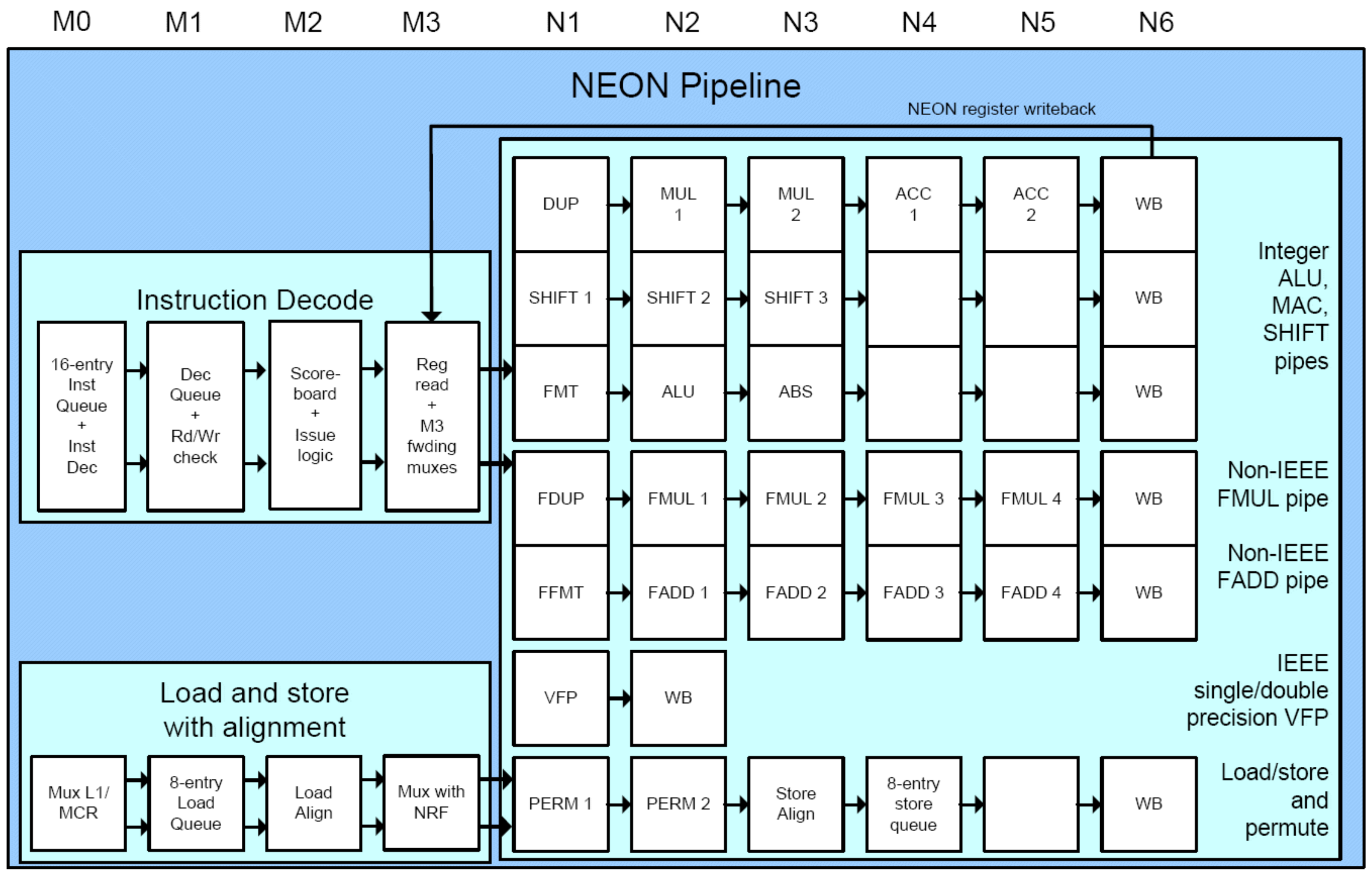
- Superscalar pipeline



ARM Cortex-A8



ARM Cortex-A8



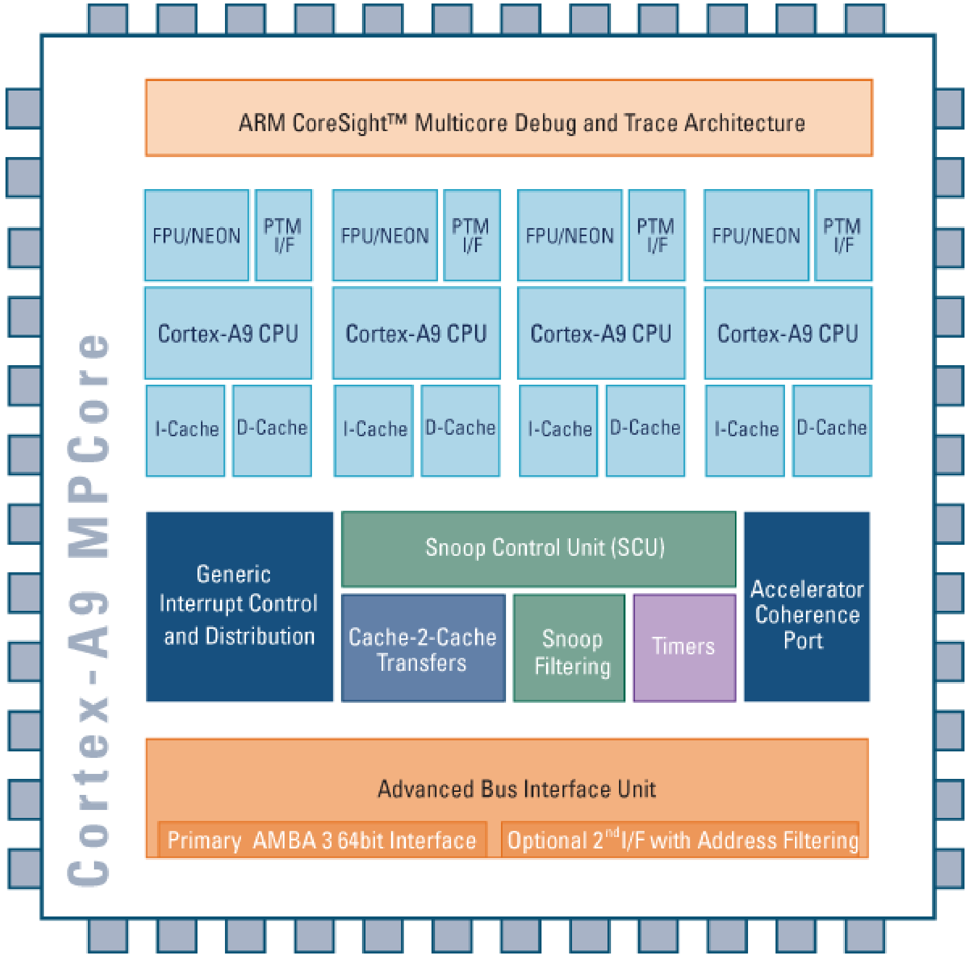
ARM Cortex-A8

Process	65nm (LP)	65nm (GP)
Frequency (MHz)	650+	1100+
Area with cache (mm^2)	<4	<4
Area without cache (mm^2)	<3	<3
Power with cache (mW/MHz)	<0.59	<0.45

ARM Cortex-A9

- Unrivalled performance with 2GHz typical operation with the TSMC 40G hard macro implementation
- Low power targeted single core implementations into cost sensitive devices
- Scalable up to four coherent cores with advanced MPCore technology
- Optional NEON™ media and/or floating point processing engine
- Dhrystone Performance: 2.50 DMIPS/MHz per core (1-4 cores)
- ISA Support
  - ARM, Thumb®-2 / Thumb, Jazelle, DSP extenstion, Advanced SIMD NEON™ unit (Optional), Floating Point Unit (Optional)

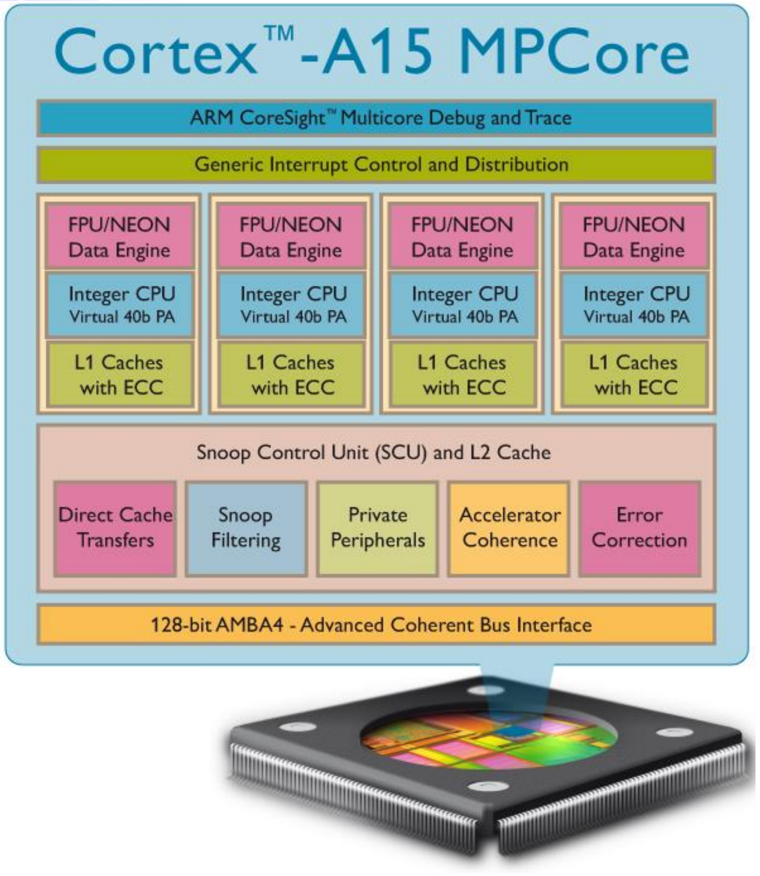
ARM Cortex-A9


Cortex-A9 MPCore
ARM CoreSight™ Multicore Debug and Trace Architecture
FPU/NEON
PTM I/F
Cortex-A9 CPU
I-Cache
D-Cache
Generic Interrupt Control and Distribution
Snoop Control Unit (SCU)
Cache-2-Cache Transfers
Snoop Filtering
Timers
Accelerator Coherence Port
Advanced Bus Interface Unit
Primary AMBA 3 64bit Interface
Optional 2nd I/F with Address Filtering

ARM Cortex-A9

Architecture	Single Core	Dual Core	Dual Core
Process	TSMC65G	TSMC40G	TSMC40G (Power Optimized)
DMIPS	2075	10000	4000
Frequency (MHz)	830	2000	800
Energy Efficiency (mW/DMIPS)	5.2	5.26	8
Power	0.4W	1.9W	0.5W
Area (mm²)	1.5 (excludes caches)	6.7 (including L1 parity and all DFT/DFM)	4.6 (including all DFT/DFM)

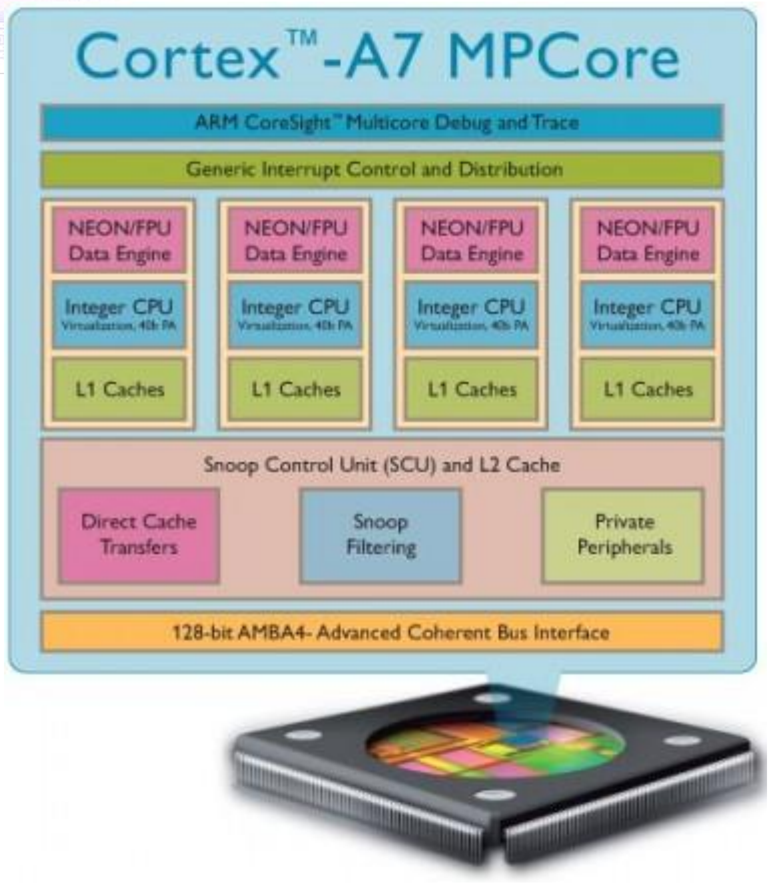
ARM Cortex-A15



Cortex™-A15 MPCore
ARM CoreSight™ Multicore Debug and Trace
Generic Interrupt Control and Distribution
FPU/NEON Data Engine
Integer CPU Virtual 40b PA
L1 Caches with ECC
FPU/NEON Data Engine
Integer CPU Virtual 40b PA
L1 Caches with ECC
FPU/NEON Data Engine
Integer CPU Virtual 40b PA
L1 Caches with ECC
FPU/NEON Data Engine
Integer CPU Virtual 40b PA
L1 Caches with ECC
Snoop Control Unit (SCU) and L2 Cache
Direct Cache Transfers
Snoop Filtering
Private Peripherals
Accelerator Coherence
Error Correction
128-bit AMBA4 - Advanced Coherent Bus Interface

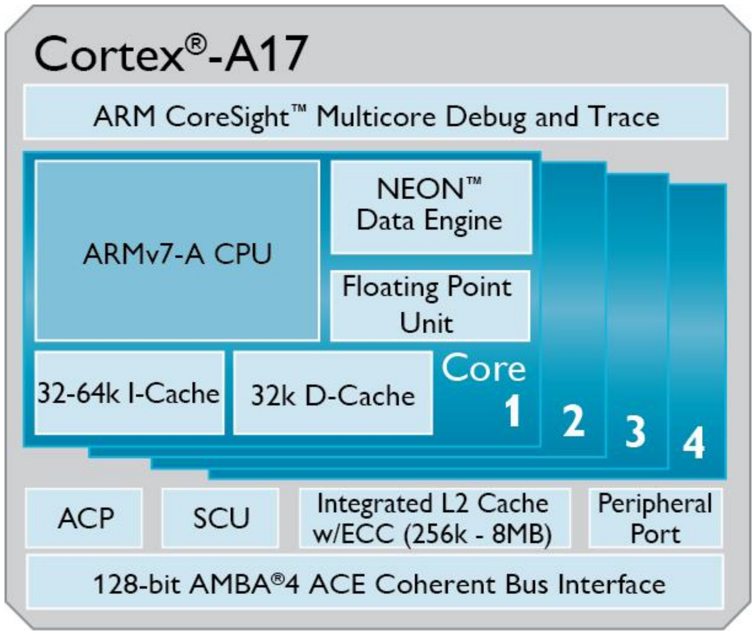
- Working frequency: 1GHz—2.5GHz
- 1—4 Cores
- Out-of-order superscalar pipeline with a tightly-coupled low-latency level-2 cache which can be up to 4MB
- Full hardware virtualization, Large Physical Address Extensions (LPAE) addressing up to 1TB of memory as well as error correction capability for fault-tolerance and soft-fault recovery

ARM Cortex-A7


Cortex™-A7 MPCore
ARM CoreSight™ Multicore Debug and Trace
Generic Interrupt Control and Distribution
NEON/FPU Data Engine
Integer CPU
Virtualization, 40b PA
L1 Caches
Snoop Control Unit (SCU) and L2 Cache
Direct Cache Transfers
Snoop Filtering
Private Peripherals
128-bit AMBA4- Advanced Coherent Bus Interface

- Designed for low-cost, fully featured entry-level smart phones and other low-power applications
- Best power-efficiency and footprint as a standalone applications processor
  - More performance than 2011 mainstream smartphone CPU
  - Up to 20% more performance while consuming 60% less power
- Companion CPU to Cortex-A15 to enable big.LITTLE Processing

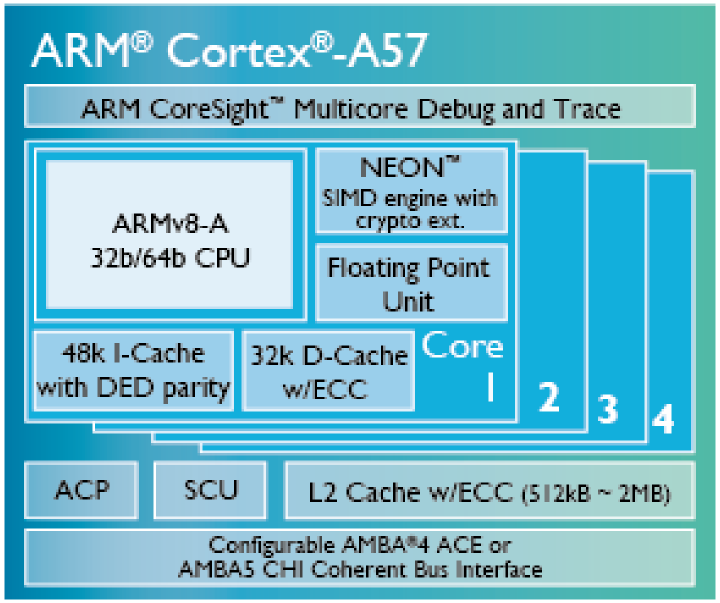
ARM Cortex-A17



Cortex®-A17
ARM CoreSight™ Multicore Debug and Trace
ARMv7-A CPU
NEON™ Data Engine
Floating Point Unit
32-64k I-Cache
32k D-Cache
Core 1 2 3 4
ACP
SCU
Integrated L2 Cache w/ECC (256k - 8MB)
Peripheral Port
128-bit AMBA®4 ACE Coherent Bus Interface

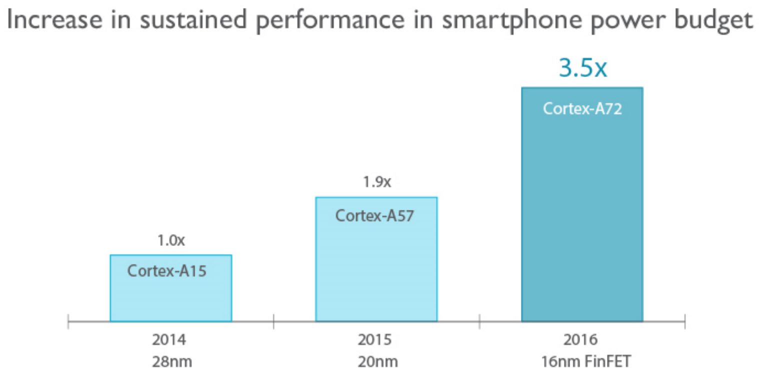
- Architecture
  - Cortex ARMv7-A processor
  - Out-of-order CPUs with a 11+ stage pipeline
- Multicore
  - 1-4X SMP within a single processor cluster
  - Multiple coherent SMP processor clusters using AMBA® 4 ACE technology
  - Compatible with CCI-400 for up to two clusters
- ISA Support
  - ARM and Thumb-2
  - TrustZone® security technology
  - NEON™ Advanced SIMD
  - DSP & SIMD extensions
  - VFPv4 Floating point
  - Hardware virtualization support
  - Large Physical Address Extensions (LPAE)
  - Integer Divide
  - Fused MAC
  - Hypervisor debug instructions

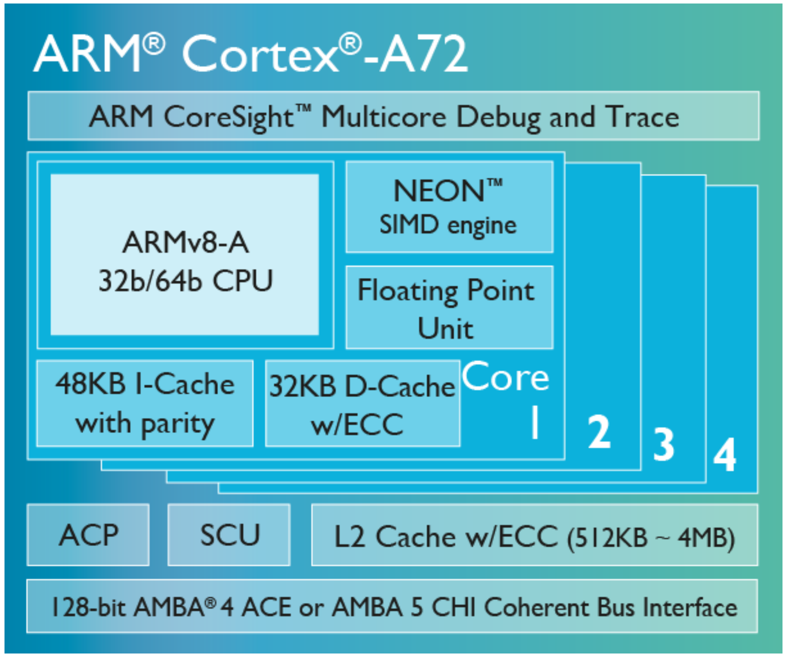
ARM Cortex-A57


ARM® Cortex®-A57
ARM CoreSight™ Multicore Debug and Trace
ARMv8-A 32b/64b CPU
NEON™ SIMD engine with crypto ext.
Floating Point Unit
48k I-Cache with DED parity
32k D-Cache w/ECC
Core 1
2
3
4
ACP
SCU
L2 Cache w/ECC (512kB ~ 2MB)
Configurable AMBA®4 ACE or AMBA5 CHI Coherent Bus Interface

- ARMv8-A
- Multicore 1-4x Symmetrical Multiprocessing (SMP) within a single processor cluster, and multiple coherent SMP processor clusters through AMBA® 5 CHI or AMBA 4 ACE technology
- ISA Support
  - AArch32 for full backward compatibility with ARMv7
  - AArch64 for 64-bit support and new architectural features
  - TrustZone® security technology
  - NEON Advanced SIMD
  - DSP & SIMD extensions
  - VFPv4 Floating point
  - Hardware virtualization support
- big.LITTLE with A53

ARM Cortex-A72

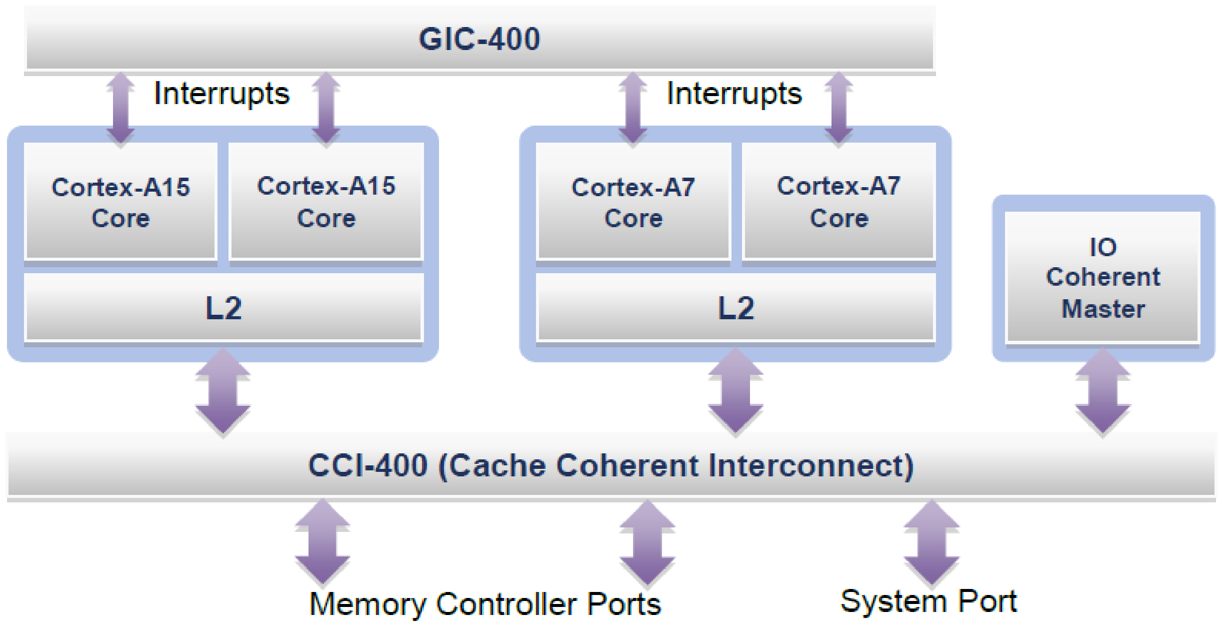




- ARMv8-A Architecture
- 1-4x SMP within a single processor cluster, and multiple coherent SMP processor clusters through AMBA® 5 CHI or AMBA 4 ACE technology
- ISA Support
  - AArch32 for full backward compatibility with ARMv7
  - AArch64 for 64-bit support and new architectural features
  - TrustZone® security technology
  - NEON™ Advanced SIMD
  - DSP & SIMD extensions
  - VFPv4 Floating point
  - Hardware virtualization support

Big.LITTLE Processing Architecture

- A15 and Cortex-A7 systems have hardware cache coherency
- CCI-400 provides cache coherency between clusters
- GIC-400 provides transparent Interrupt control

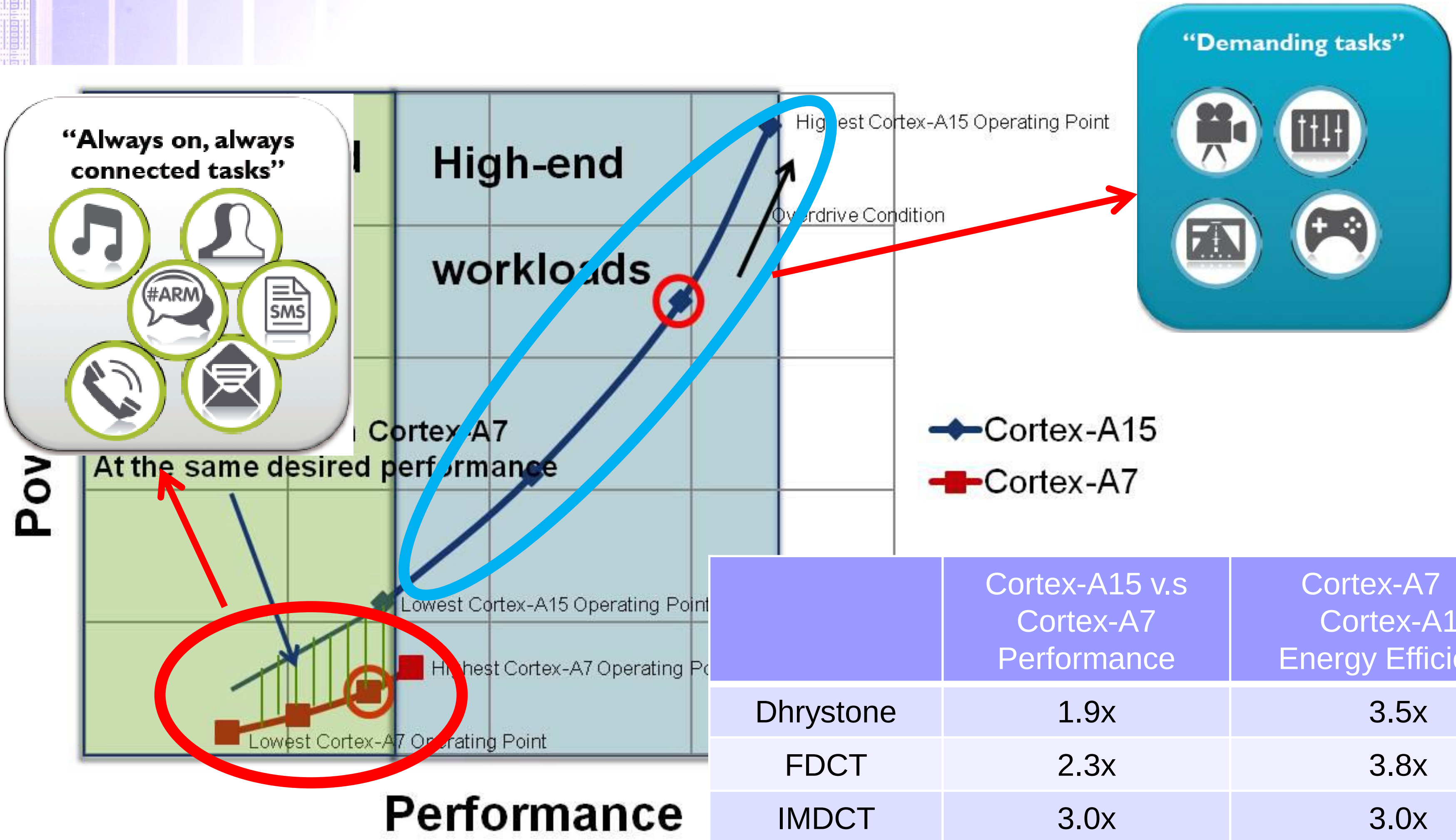


Full task migration in less than 20us

Cache coherency managed in hardware

128-bit system transactions

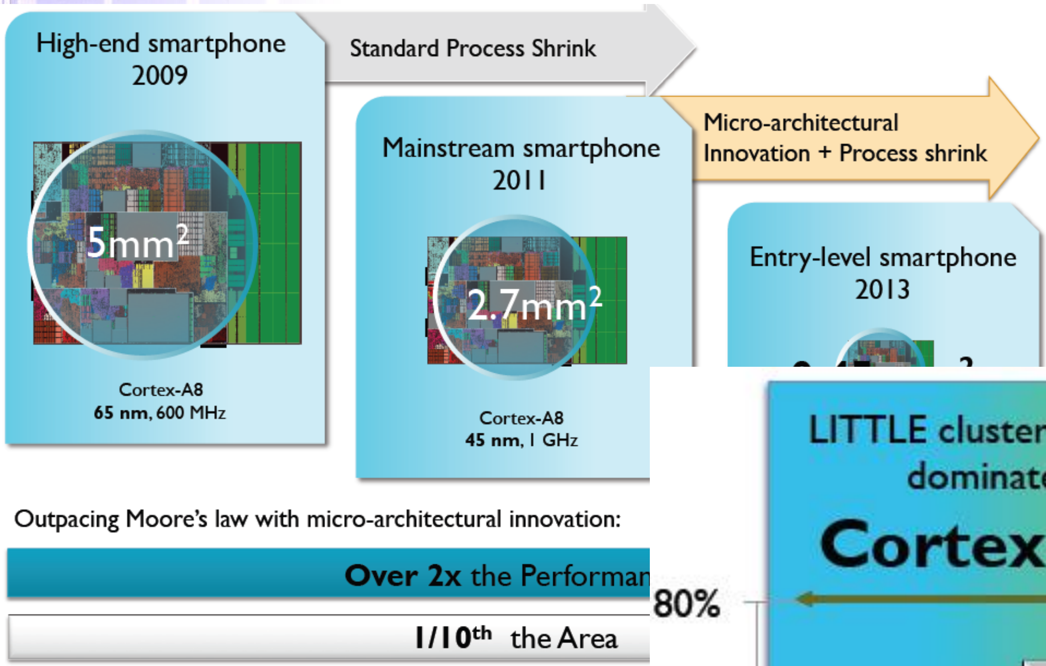
big.LITTLE Processing Architecture



	Cortex-A15 v.s Cortex-A7 Performance	Cortex-A7 v.s Cortex-A15 Energy Efficiency
Dhrystone	1.9x	3.5x
FDCT	2.3x	3.8x
IMDCT	3.0x	3.0x
MemCopy L1	1.9x	2.3x
MemCopy L2	1.9x	3.4x

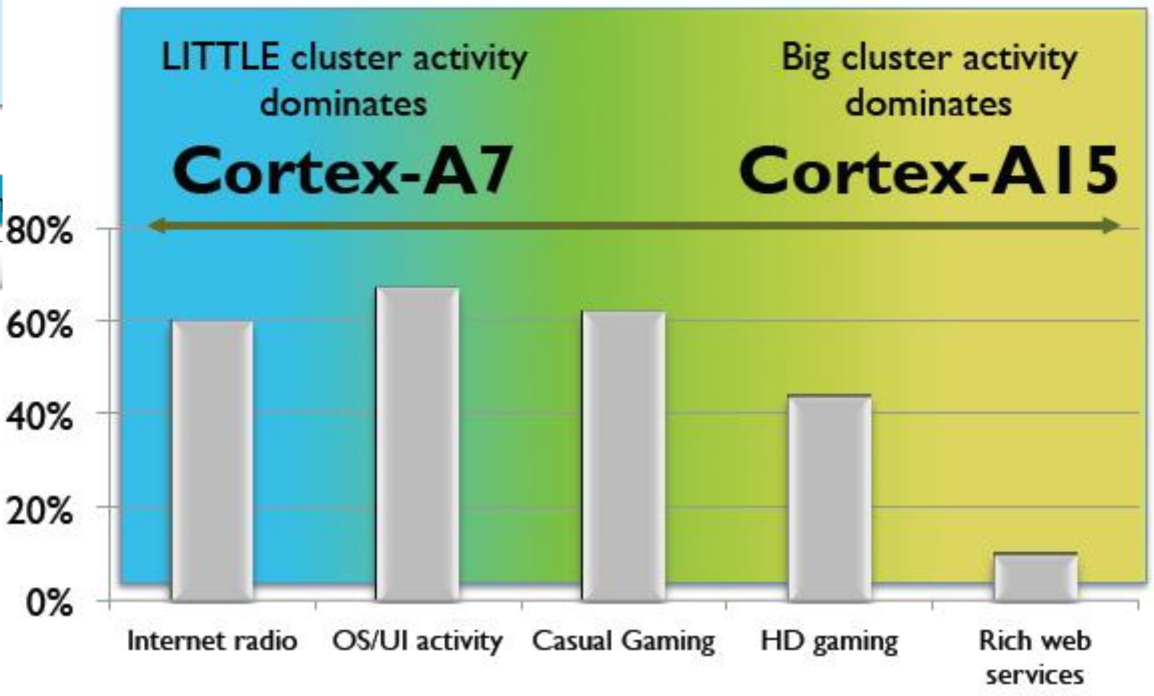
big.LITTLE Processing Architecture

- Why Choose Cortex-A7


High-end smartphone 2009
5mm²
Cortex-A8
65 nm, 600 MHz
Standard Process Shrink
Mainstream smartphone 2011
2.7mm²
Cortex-A8
45 nm, 1 GHz
Micro-architectural Innovation + Process shrink
Entry-level smartphone 2013
Outpacing Moore's law with micro-architectural innovation:
Over 2x the Performan
1/10th the Area

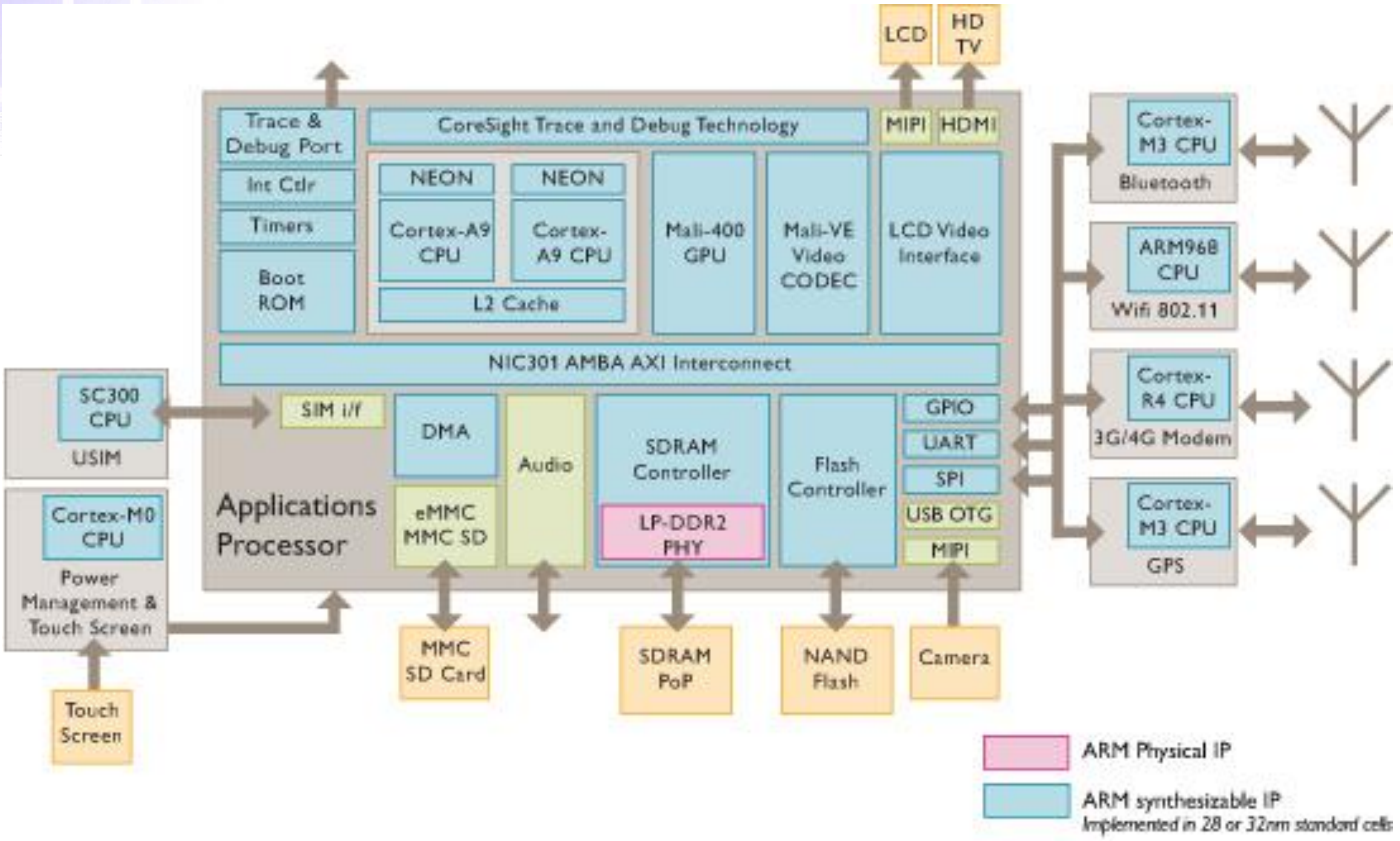
Power Saving Performance

A7+A15 v.s A9



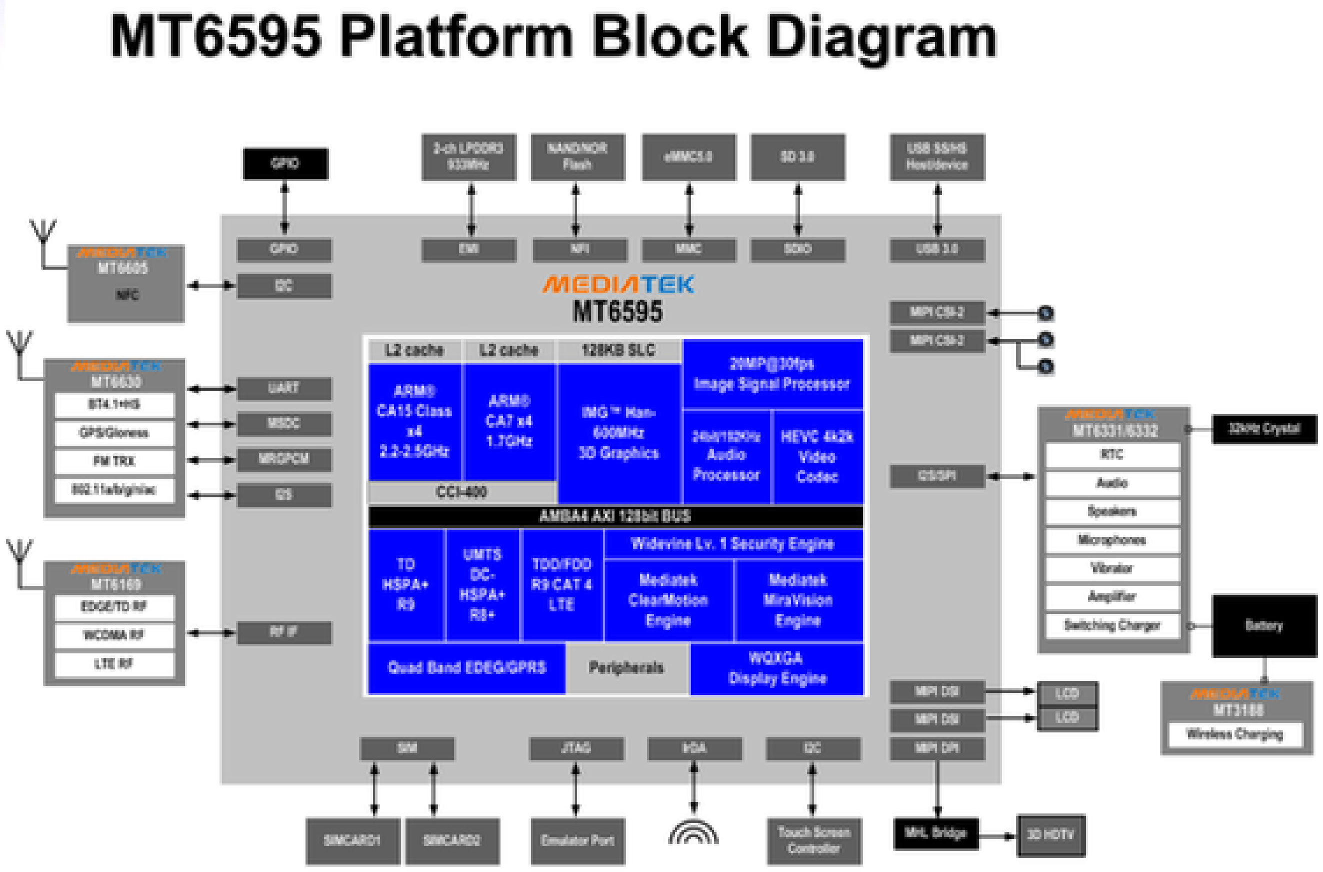
Multimedia SoC Design

One Example



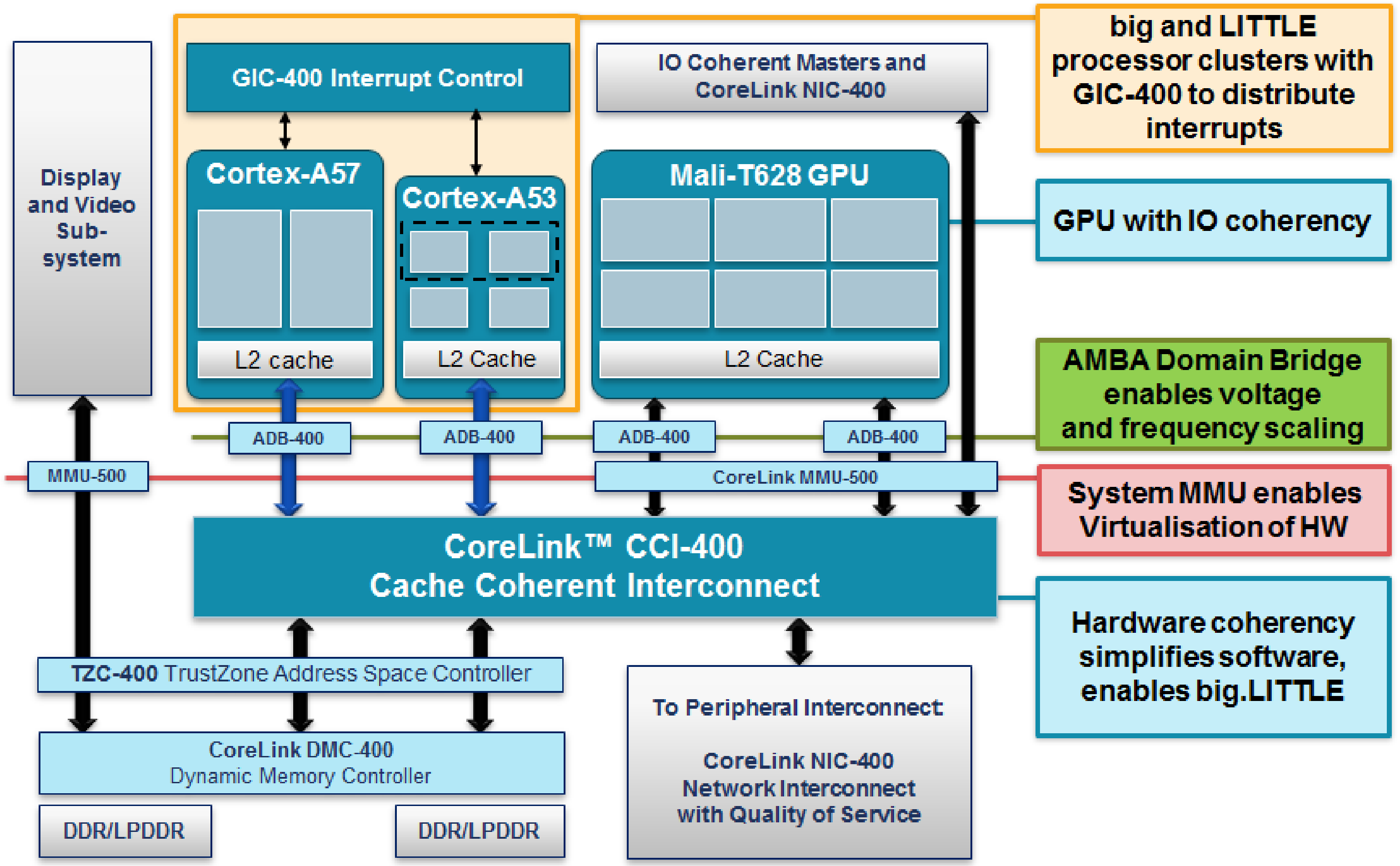
LCD
HD TV
Trace & Debug Port
CoreSight Trace and Debug Technology
MIPI
HDMI
Int Ctlr
Timers
Boot ROM
NEON
NEON
Cortex-A9 CPU
Cortex-A9 CPU
L2 Cache
Mali-400 GPU
Mali-VE Video CODEC
LCD Video Interface
Cortex-M3 CPU
Bluetooth
ARM968 CPU
Wifi 802.11
Cortex-R4 CPU
3G/4G Modem
Cortex-M3 CPU
GPS
NIC301 AMBA AXI Interconnect
SC300 CPU
USIM
SIM i/f
DMA
Audio
SDRAM Controller
LP-DDR2 PHY
Flash Controller
GPIO
UART
SPI
USB OTG
MIPI
Cortex-M0 CPU
Power Management & Touch Screen
Applications Processor
eMMC MMC SD
MMC SD Card
SDRAM PoP
NAND Flash
Camera
Touch Screen
ARM Physical IP
ARM synthesizable IP
Implemented in 28 or 32nm standard cells

Another Example



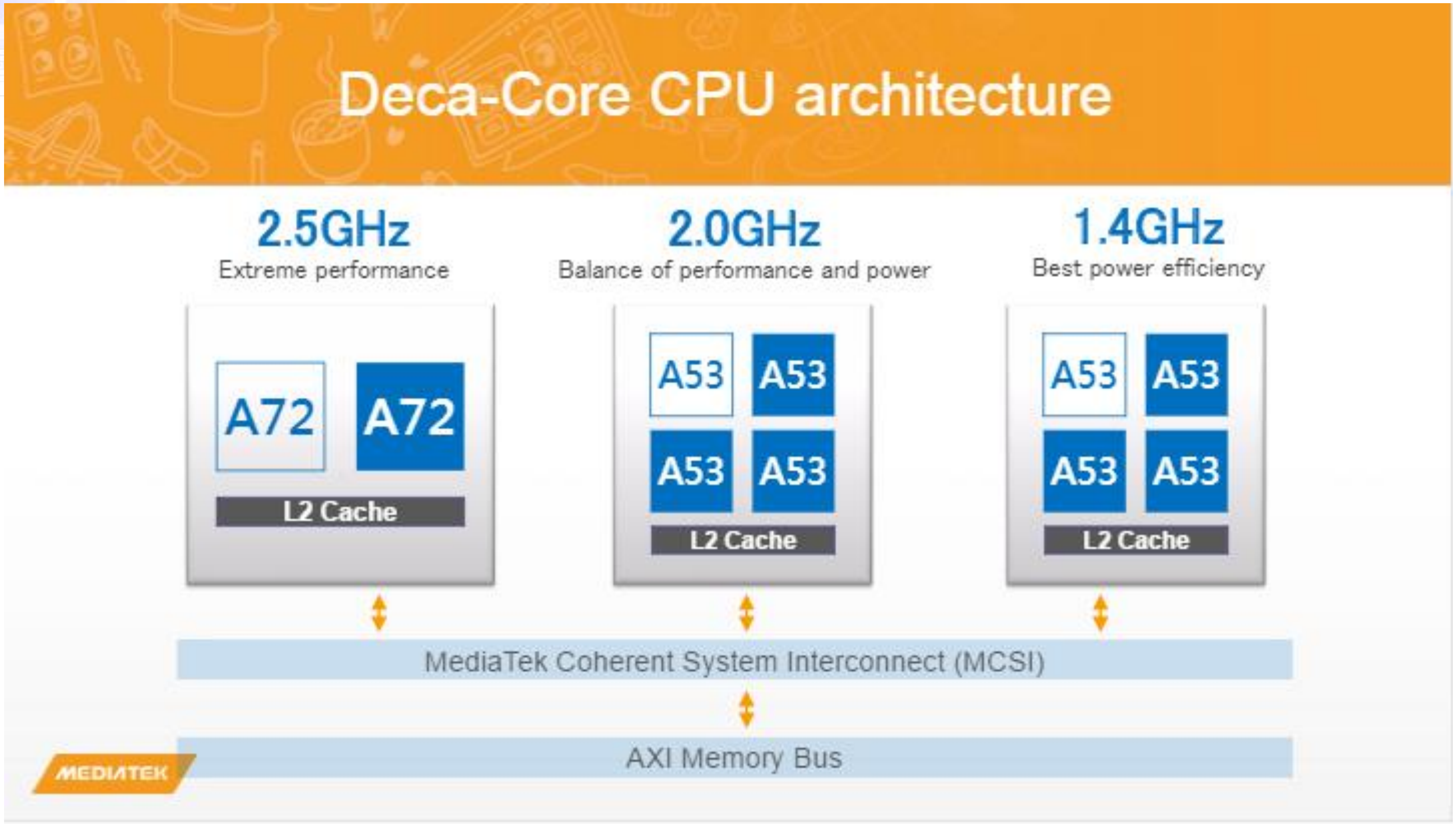
http://event.mediatek.com/_en_octacore/index.html

Another Example



Display and Video Sub-system
GIC-400 Interrupt Control
Cortex-A57
L2 cache
Cortex-A53
L2 Cache
IO Coherent Masters and CoreLink NIC-400
Mali-T628 GPU
L2 Cache
ADB-400
ADB-400
ADB-400
ADB-400
MMU-500
CoreLink MMU-500
CoreLink™ CCI-400 Cache Coherent Interconnect
TZC-400 TrustZone Address Space Controller
CoreLink DMC-400 Dynamic Memory Controller
DDR/LPDDR
DDR/LPDDR
To Peripheral Interconnect
CoreLink NIC-400 Network Interconnect with Quality of Service
big and LITTLE processor clusters with GIC-400 to distribute interrupts
GPU with IO coherency
AMBA Domain Bridge enables voltage and frequency scaling
System MMU enables Virtualisation of HW
Hardware coherency simplifies software, enables big.LITTLE

Another Example: Helio X20



AMBA

Ref: AMBA Specification Rev. 2.0

Outline

- Overview
- AHB

Outline

- Overview
- AHB

BUS Brief

- In a system, various subsystems must have interfaces to one another
- The bus serves as a shared communication link between subsystems
- Advantages
  - Low cost
  - Versatility
- Disadvantage
  - Performance bottleneck

AMBA Introduction

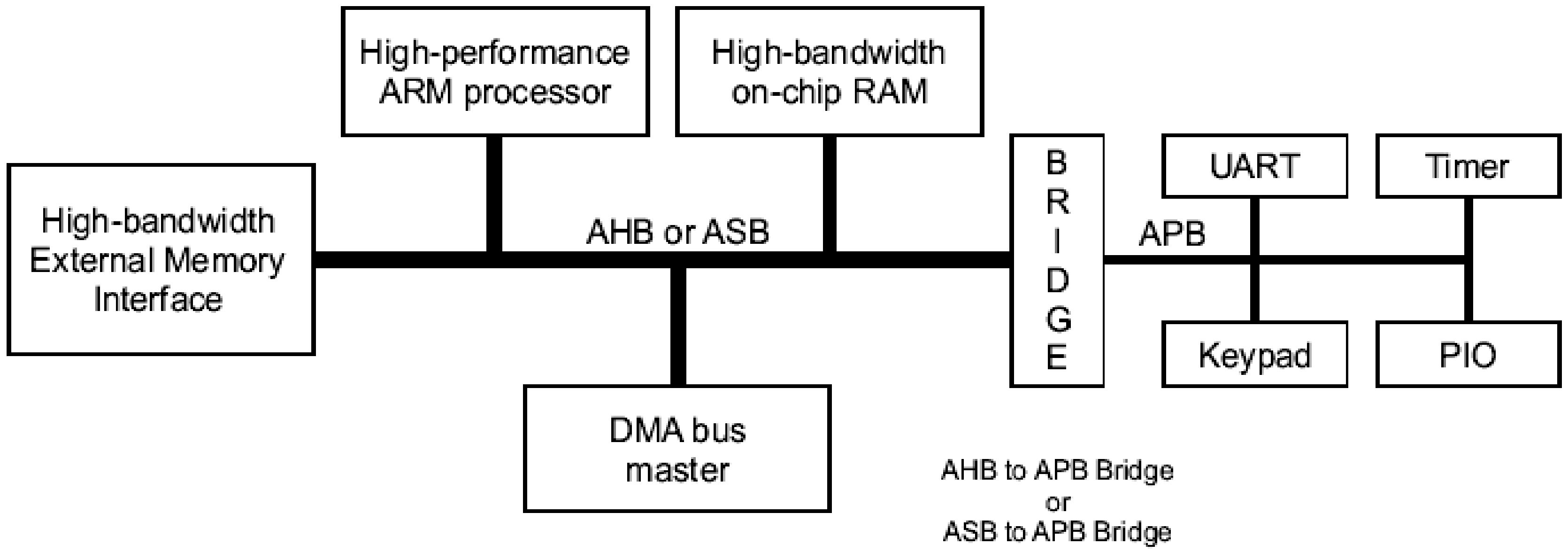
- Advanced Microcontroller Bus Architecture
  - An on-chip communication standard
- Three buses defined
  - AHB (Advanced High-performance Bus)
  - ASB (Advanced System Bus)
  - APB (Advanced Peripheral Bus)

AMBA History

- AMBA 1.0
  - ASB and APB
  - Tri-state implementation
- AMBA 2.0
  - AHB, ASB, and APB
  - Multiplexer architecture to eliminate timing problem
- AMBA 3.0
  - AMBA Advanced eXtensible Interface (AXI)
- AMBA 4.0
  - Minor changes to AXI
  - AXI4-Lite, AXI4-Stream
  - In phase 2: ACE (AXI Coherency Extensions), ACE-Lite
- AMBA 5.0
  - AHB
  - Coherent Hub Interface (CHI)

A Typical AMBA 2.0 System

- Processors or other masters/slaves can be replaced


High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth External Memory Interface
AHB or ASB
DMA bus master
BRIDGE
APB
UART
Timer
Keypad
PIO
AHB to APB Bridge or ASB to APB Bridge

AHB

- High performance
- Pipelined operation
- Multiple bus masters (up to 16)
- Burst transfers
- Split transactions
- Bus width: 8, 16, 32, 64, 128 bits
- Mux-type bus
- Single clock edge (rising edge) design
- Recommended for new designs

ASB

- High performance
- Pipelined operation
- Multiple bus masters
- Burst transfers
- Bus width: 8, 16, 32 bits
- Tristate-type bus
- Falling edge design

APB

- Lower power
- Latched address and control
- Simple interface
- Suitable for many peripherals
- Single clock edge (rising edge) design
- Appears as a local secondary bus that is encapsulated as a single AHB or ASB slave devices

AHB Components

- AHB master
  - Initiate a read/write operation
  - Only one master is allowed to use the bus
  - uP, DMA, DSP, …
- AHB slave
  - Respond to a read/write operation
  - Address mapping
  - External memory I/F, APB bridge, internal memory, …
- AHB arbiter
  - Ensure that which master is active
  - Arbitration algorithm is not defined in ABMA spec.
- AHB decoder
  - Decode the address and generate select signal to slaves

APB Components

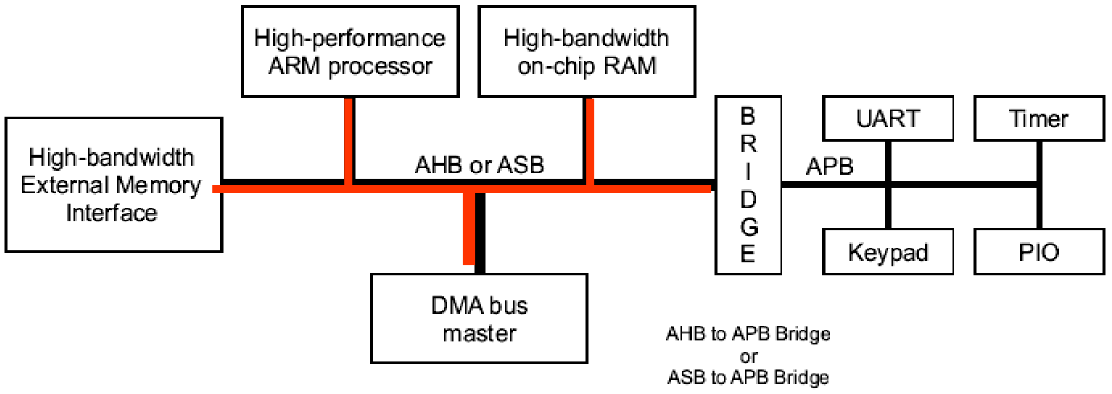
- AHB2APB Bridge
  - Provides latching of all address, data, and control signals
  - Provides a second level of decoding to generate slave select signals for the APB peripherals
- All other modules on the APB are APB slaves
  - Un-pipelined
  - Zero-power interface
  - Timing can be provided by decode with strobe timing
  - Write data valid for the whole access

Notes on the AMBA Specification

- Technology independent
- Not define electrical characteristics
- Timing specification only at the cycle level
  - Exact timing requirements will depend on the process technology used and operation frequency

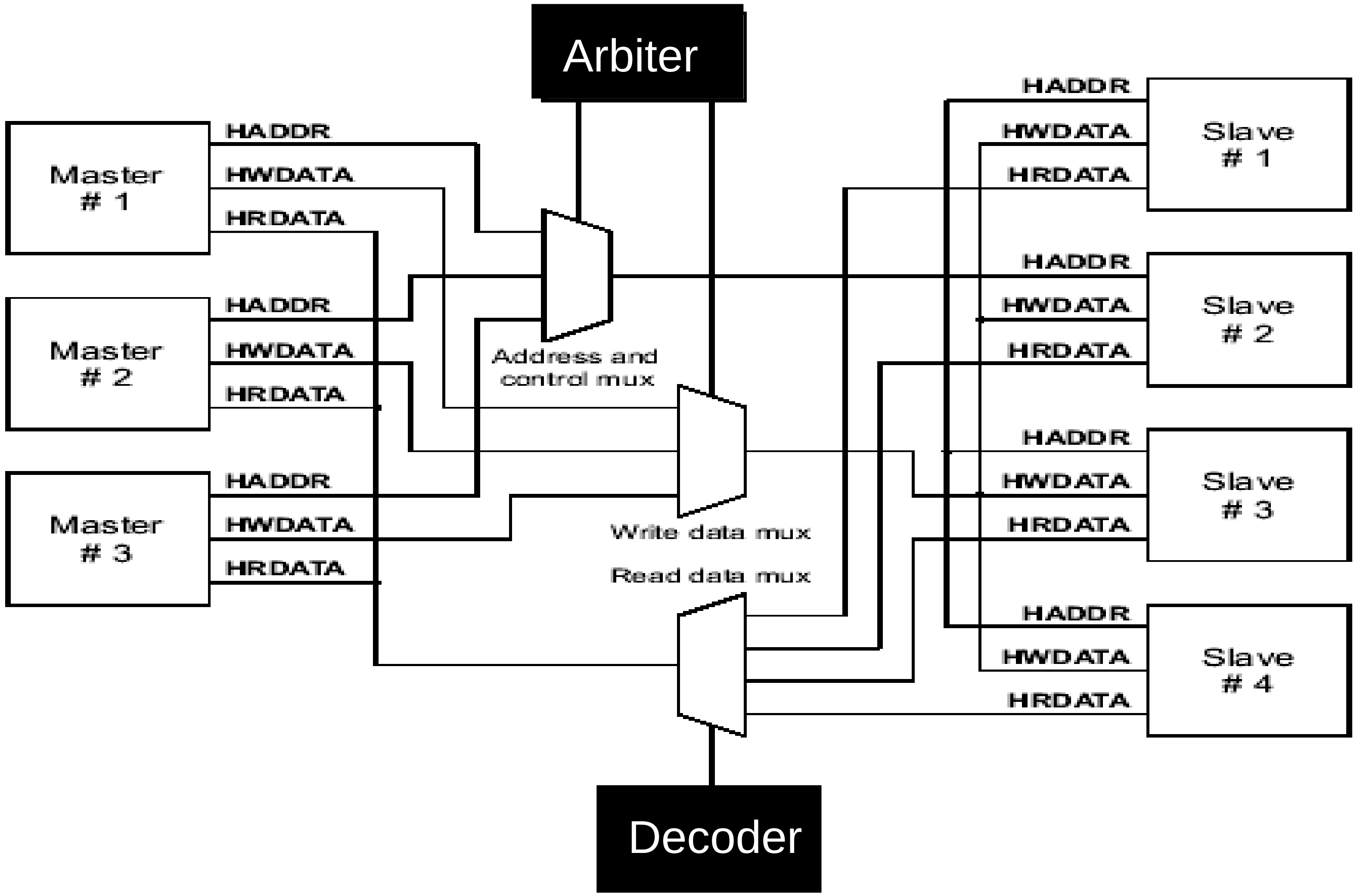
Outline

- Overview
- AHB

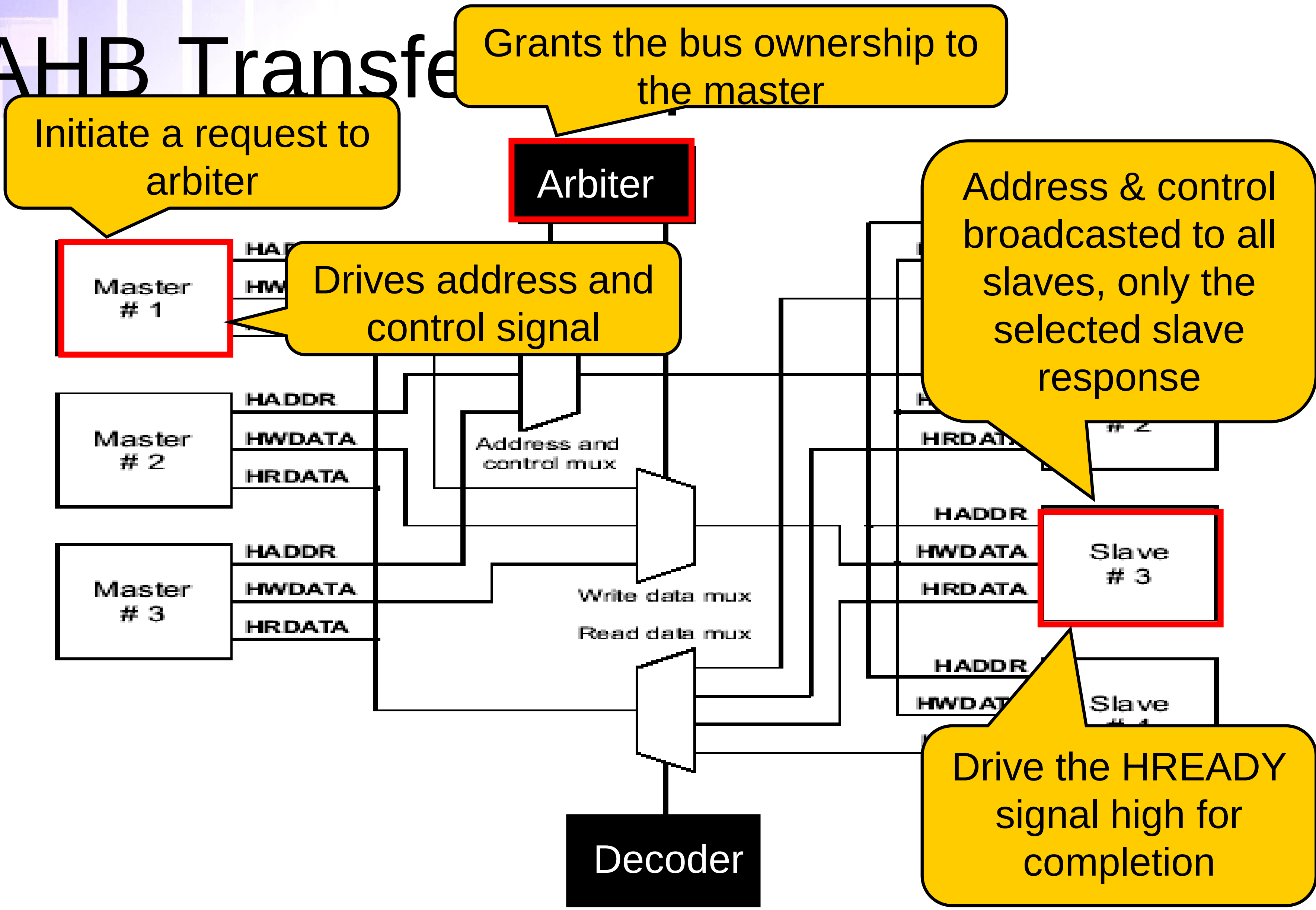


High-performance ARM processor
High-bandwidth on-chip RAM
High-bandwidth External Memory Interface
AHB or ASB
DMA bus master
BRIDGE
APB
UART
Timer
Keypad
PIO
AHB to APB Bridge or ASB to APB Bridge

AHB Bus Interconnection



AHB Transfe



Grants the bus ownership to the master
Initiate a request to arbiter
Arbiter
Master # 1
HA
HW
Drives address and control signal
Address & control broadcasted to all slaves, only the selected slave response
Master # 2
HADDR
HWDATA
HRDATA
Master # 3
HADDR
HWDATA
HRDATA
Address and control mux
Write data mux
Read data mux
HRDATA
HADDR
HWDATA
HRDATA
Slave # 3
HADDR
HWDAT
Slave
Drive the HREADY signal high for completion
Decoder

AHB Signals

Name	Source	Description
HCLK	Clock source	Bus clock
HRESETn	Reset controller	Reset
HADDR[31:0]	Master	Address bus
HTRANS[1:0]	Master	Transfer type
HWRITE	Master	Transfer direction
HSIZE[2:0]	Master	Transfer size
HBURST[2:0]	Master	Burst type
HPROT[3:0]	Master	Protection control
HWDATA[31:0]	Master	Write data bus
HSELx	Decoder	Slave select
HRDATA[31:0]	Slave	Read data bus
HREADY	Slave	Transfer done
HRESP[1:0]	Slave	Transfer response (status)

Basic AHB Signals

- HRESETn
  - Active low
- HADDR[31:0]
  - The 32-bits system address bus
- HWDATA[31:0]
  - Write data bus, from master to slave
- HRDATA[31:0]
  - Read data bus, from slave to master

Basic AHB Signals (cont.)

- HTRANS
  - Indicates the type of the current transfer
    - NONSEQ, SEQ, IDLE, or BUSY
- HSIZE
  - Indicate the size of the transfer
- HBURST
  - Indicate the burst type of the transfer
- HRESP
  - Shows the status of bus transfer, from slave to master
    - OKAY, ERROR, RETRY, or SPLIT

Basic AHB Signals (cont.)

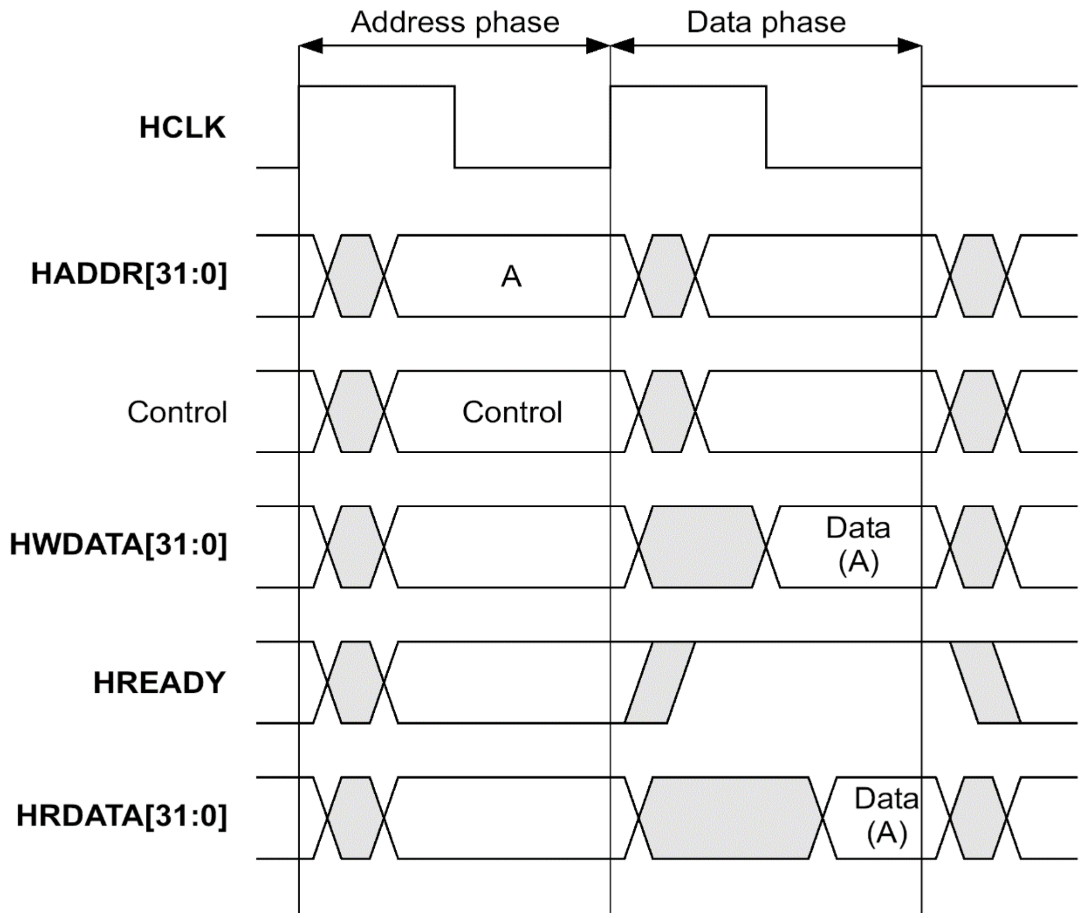
- HREADY
  - High: the slave indicate the transfer done
  - Low: the slave extend the transfer
- HREADY_IN
  - From decoder
  - Decoder tell the slave that the bus is available
  - Not explained in AMBA spec. 2.0 !!

Basic AHB Transfers

- Two distinct sections
  - The **address phase**, only one cycle
  - The **data phase**, may require several cycles, achieved by HREADY signals
- Pipeline transaction
  - Address phase is before the data phase

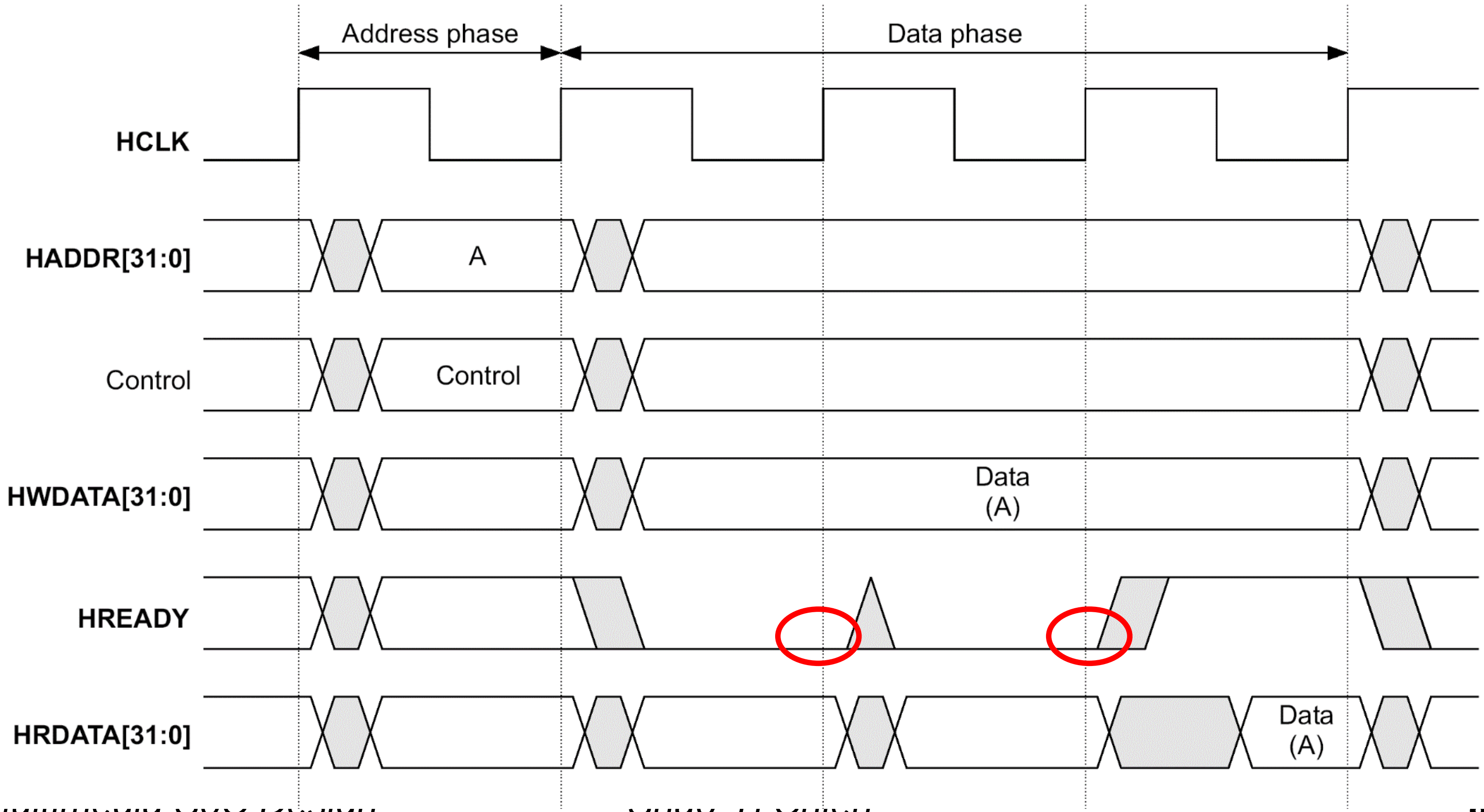
Basic AHB Transfers (cont.)

- A simple transfer with no wait state



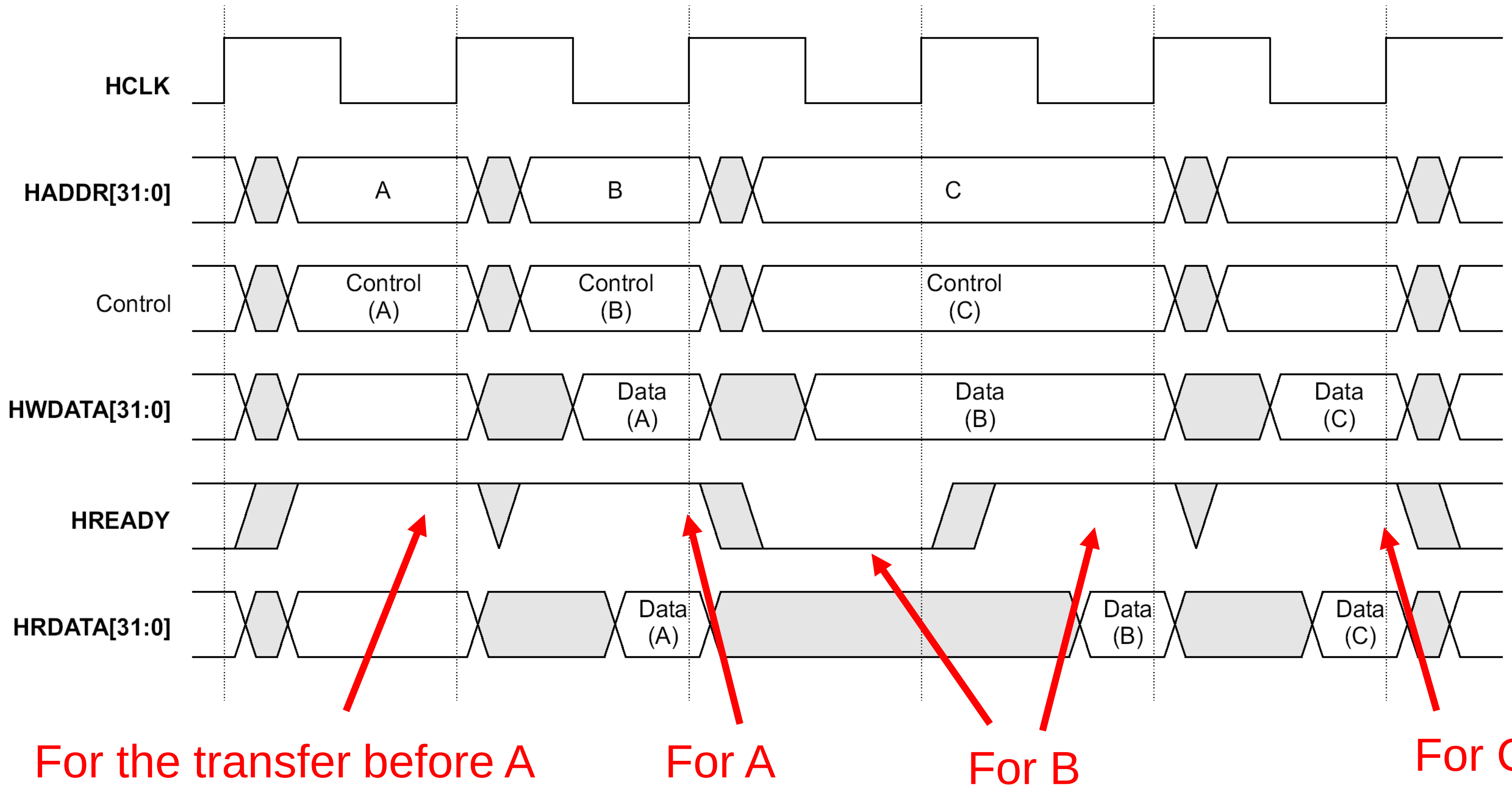
Basic AHB Transfers (cont.)

- A simple transfer with two wait states



Basic AHB Transfers (cont.)

- Multiple transfers



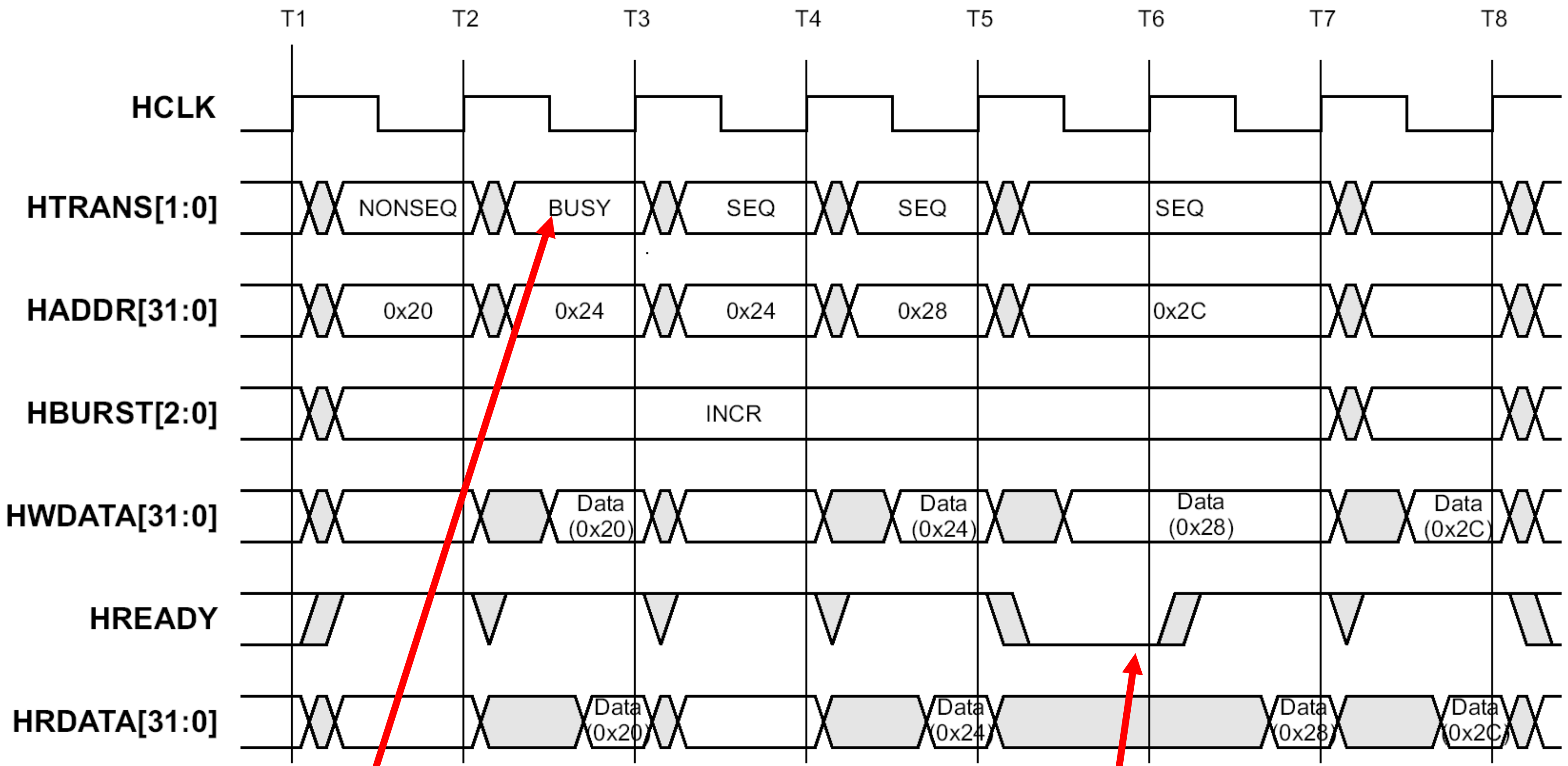
Transfer Type

- HTRANS[1:0]: transfer type
  - Four types, IDEL, BUSY, NONSEQ,SEQ
- 00 : IDLE
  - No transfers
  - When master grant bus, but no transfer
- 01 : BUSY
  - Allow the master to insert IDLE cycle during transfers

Transfer Type (cont.)

- 10 : NOSEQ
  - Indicate a single transfer
  - or the first transfer of a burst
  - The address & control signals are unrelated to the previous transfer
- 11 : SEQ
  - Indicate the following transfers
  - The address is related to the previous transfer

Transfer Type Example



Delayed by the master

Delayed by the slave

AHB Control Signals

- HWRITE
  - High : write
  - Low : read

- HSIZE[2:0]
  - 000 : 8 bits
  - 001 : 16 bits
  - 010 : 32 bits
  - 011 : 64 bits
  - 100 : 128 bits
  - 101 : 256 bits
  - 110 : 512 bits
  - 111 : 1024 bits
  - The max is constrained by the bus configuration
  - 32 bits (010) is often used

Burst Operation

- AHB burst operations
  - 4-beat, 8-beat, 16-beat, single transfer, and undefined-length transfer
  - Both incrementing & wrapping burst
- Incrementing burst
  - Sequential, the address is just the increment of the previous one
- Wrapping burst
  - If the start address is not aligned (size x beats), the address will wrap when the boundary is reached
  - Ex: 4-beat wrapping burst of word (4-byte): 0x34→0x38→0x3C→0x30

Address Calculation Example

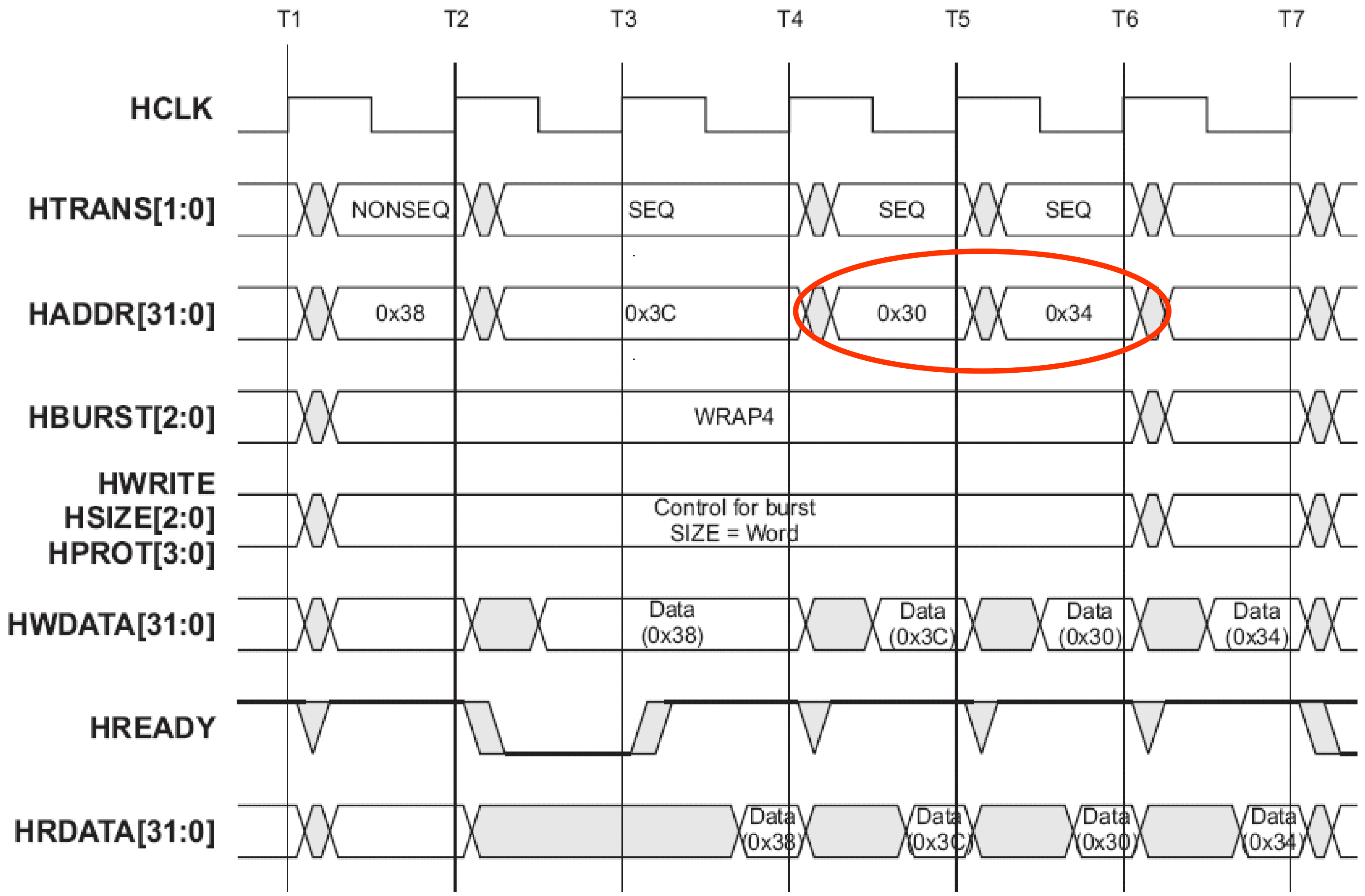
- The address calculation is according to HSIZE and HBURST
- Example: HSIZE = 010 (32 bits) with starting address = 0x48

HBURST	Type	Address
000	SINGLE	0x48
001	INCR	0x48, 0x4C, 0x50,… The most useful
010	WRAP4	0x48, 0x4C, 0x40, 0x44
011	INCR4	0x48, 0x4C, 0x50, 0x54
100	WRAP8	0x48, 0x4C, 0x50, 0x54, 0x58, 0x5c, 0x40, 0x44
101	INCR8	0x48, 0x4C, 0x50, 0x54, 0x58, 0x5c, 0x60, 0x64
110	WRAP16	0x48, 0x4C,…, 0x7c, 0x40, 0x44
111	INCR16	0x48, 0x4C,…, 0x7c, 0x80, 0x84

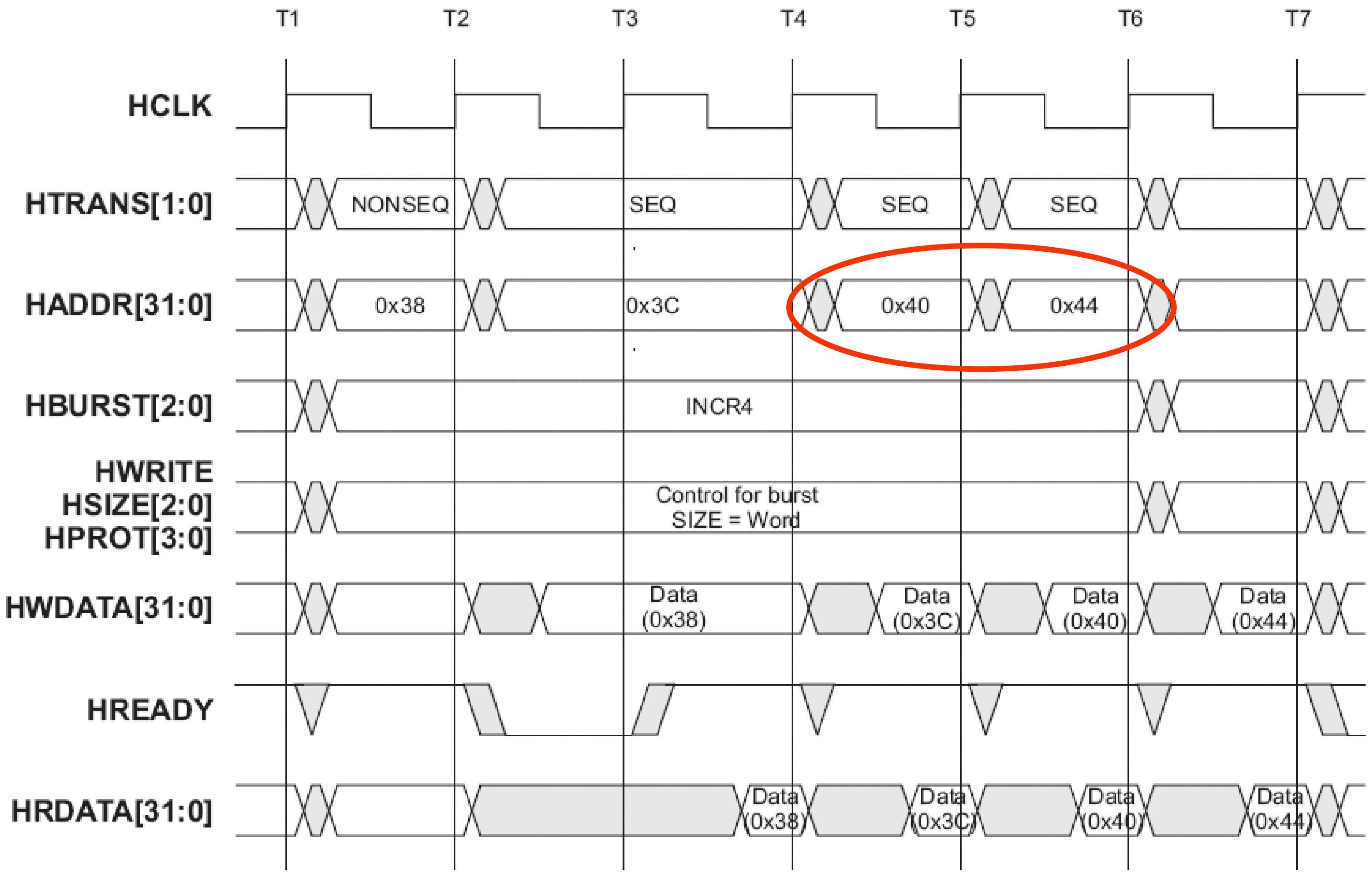
Important!!

- Burst transfer can't cross the 1K boundary
  - Because the minimal address range for a slave is 1 KB
  - NONSEQ → SEQ → 1KB Boundary → NONSEQ → SEQ…
- The master should not attempt to start a fixed-length incrementing burst which would cause this boundary to be crossed

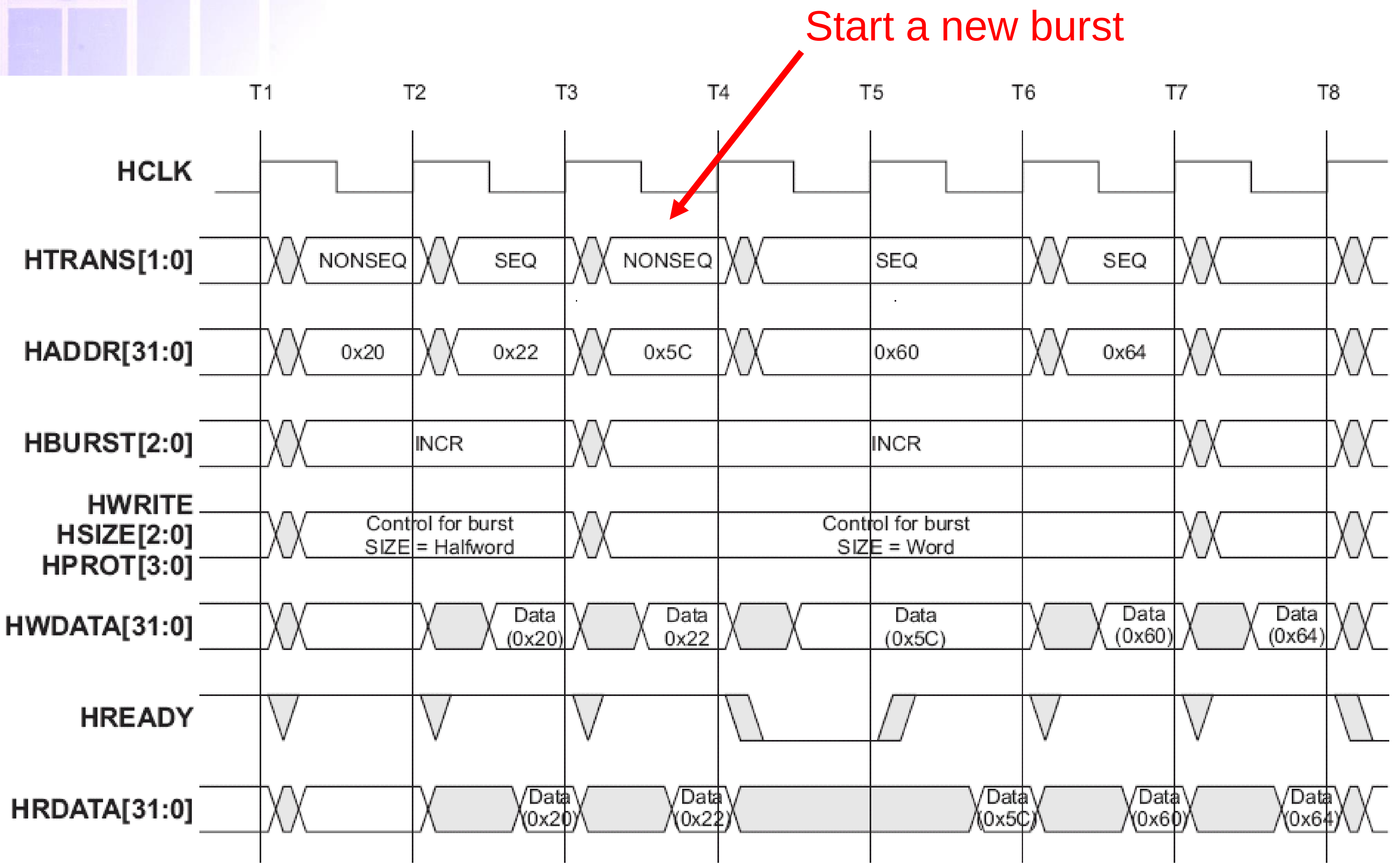
Example: 4-Beat Wrapping Burst



Example: 4-Beat Incrementing Burst



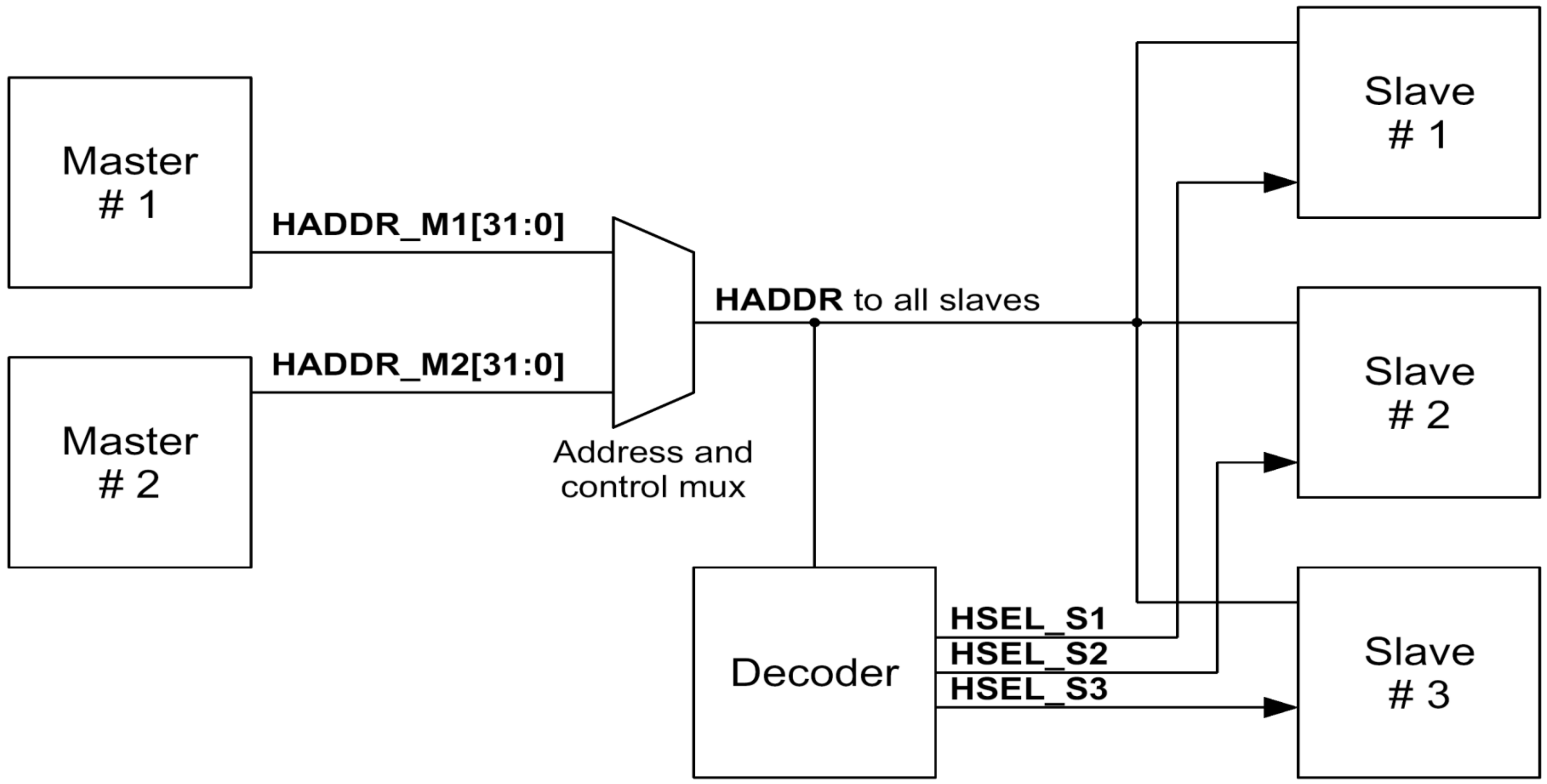
Example: Undefined-Length Burst



Address Decoding

- HSELx : slave select
  - Indicate the slave is selected by a master
- A central address decoder is used to provide the select signal
- A slave should occupy at least 1KB of memory space
- An additional default slave is used to fill up the memory map

Address Decoding (cont.)



Slave Response

- The slave accessed must respond the transfer
- The slave may
  - Complete the transfer
  - Insert wait state
  - Signal an error to indicate the transfer failure
  - Delay the transfer, leave the bus available for other transfer (split)

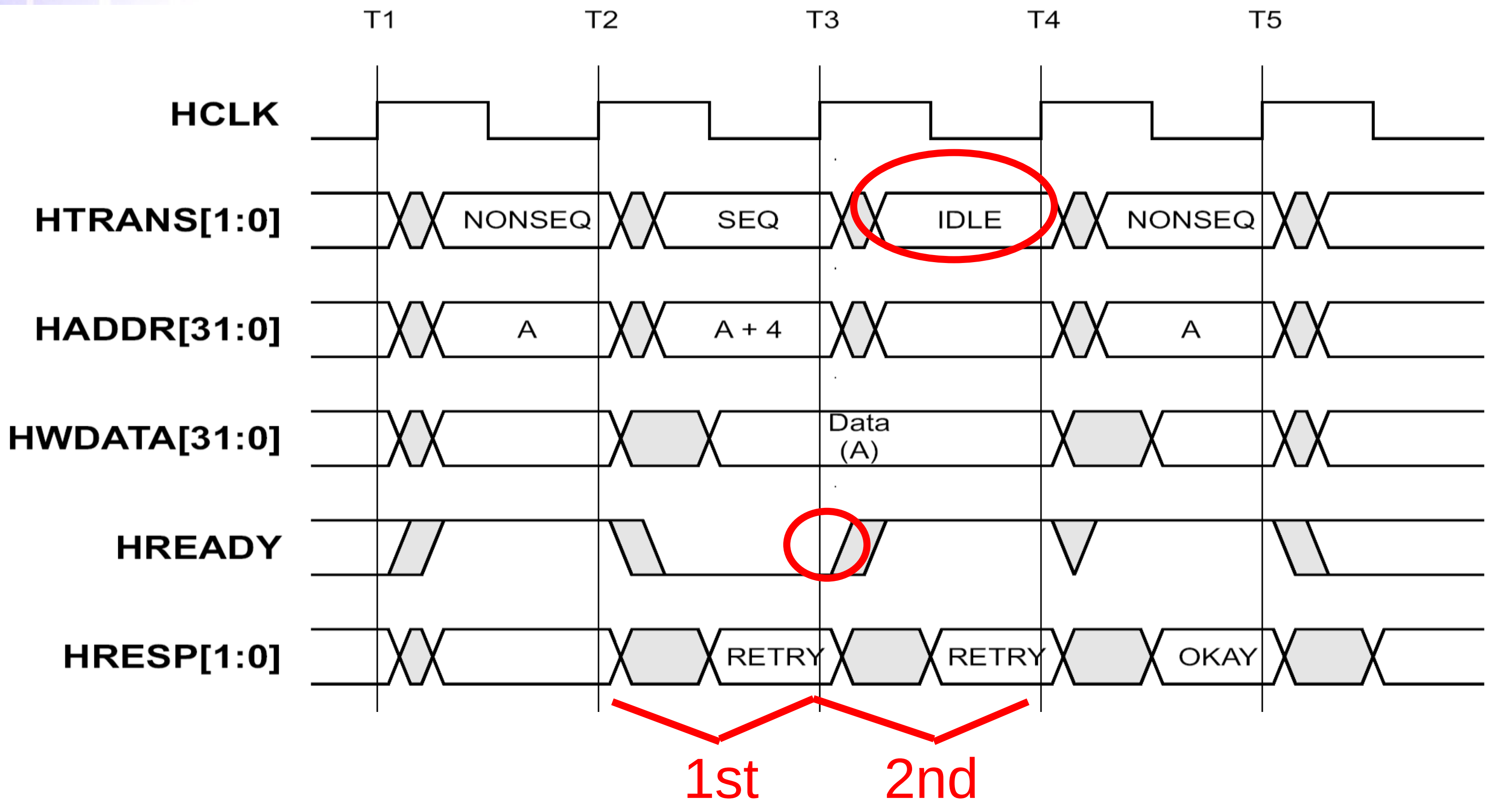
Slave Response Signals

- HREADY : transfer done
- HRESP[1:0] : transfer response
- 00 : OKAY
  - Successful
- 01 : ERROR
  - Error
- 10 : RETRY
  - The transfer is not completed
  - Ask the master to perform a retry transfer
- 11 : SPLIT
  - The transfer is not completed
  - Ask the master to perform a split transfer

Two-cycle Response

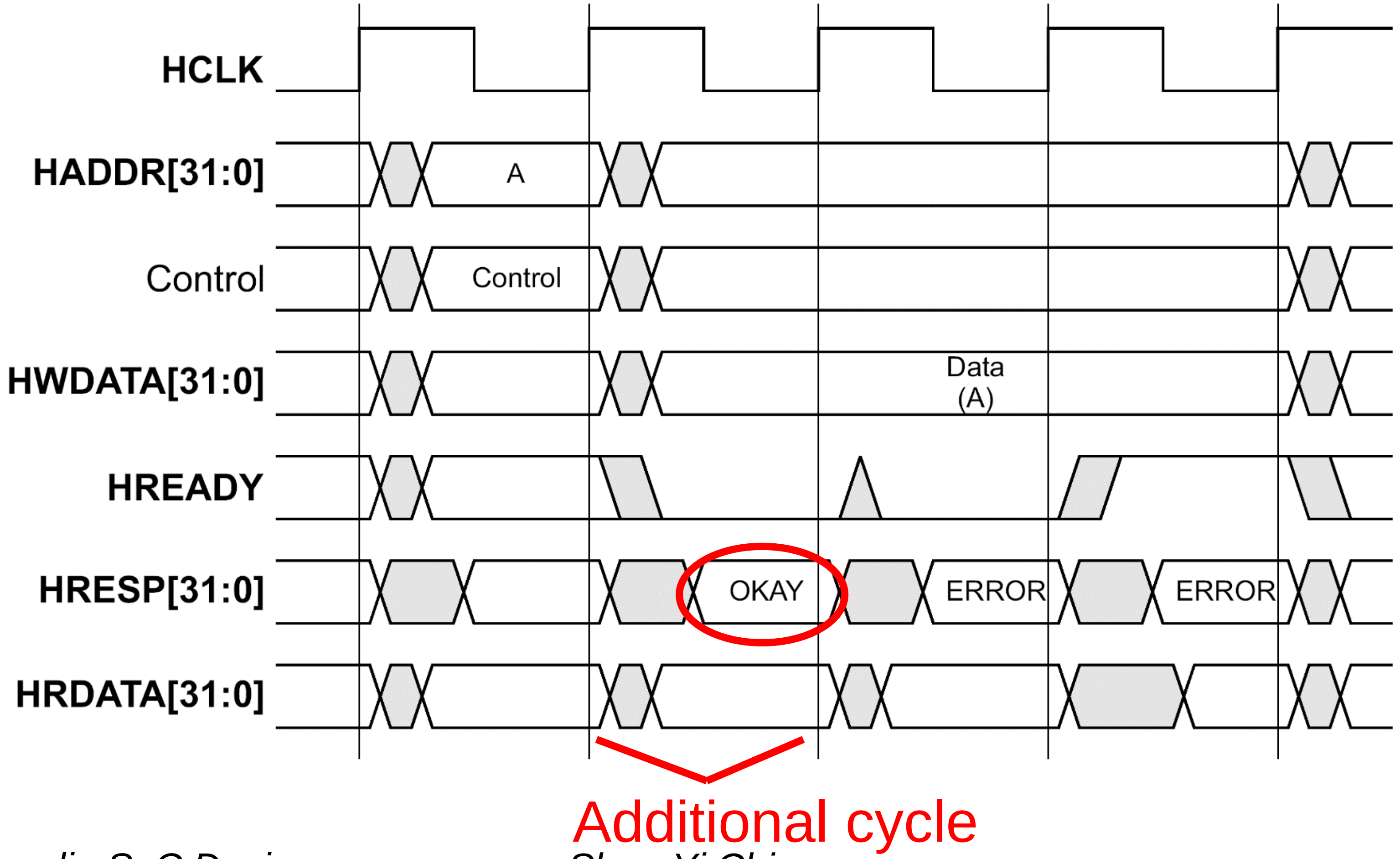
- HRESP[1:0]
  - OKAY: single cycle response
  - ERROR : two-cycle response
  - RETRY : two-cycle response
  - SPLIT : two-cycle response
- Two-cycle response is required because of the pipeline nature of the bus. This allows sufficient time for the master to handle the next transfer

Retry Response Example



ERROR Response Example

- An error response which needs three cycles



Different Between Retry and Split

- The major difference is the way of arbitration
  - RETRY: the arbiter will continue to use the normal priority
  - SPLIT: the arbiter will adjust the priority scheme so that any other master requesting the bus will get access
    - Requires extra complexity in both the slave and the arbiter
- A bus master should treat RETRY and SPLIT in the same manner

Data Bus

- Because of non-tri-state, separate read & write buses
- Endianness
  - Not specified in the AMBA spec.
  - **All the masters and slaves should of the same endianness**
  - Dynamic endianness is not supported

- For IP design, only IPs which will be used in wide variety of applications should be made bi-endian

Active Bytes Lanes for a 32-bit Little-Endian Data Bus

Transfer size	Address offset	DATA [31:24]	DATA [23:16]	DATA [15:8]	DATA [7:0]
Word	0	✓	✓	✓	✓
Halfword	0	-	-	✓	✓
Halfword	2	✓	✓	-	-
Byte	0	-	-	-	✓
Byte	1	-	-	✓	-
Byte	2	-	✓	-	-
Byte	3	✓	-	-	-

Active Bytes Lanes for a 32-bit Big-Endian Data Bus

Transfer size	Address offset	DATA [31:24]	DATA [23:16]	DATA [15:8]	DATA [7:0]
Word	0	✓	✓	✓	✓
Halfword	0	✓	✓	-	-
Halfword	2	-	-	✓	✓
Byte	0	✓	-	-	-
Byte	1	-	✓	-	-
Byte	2	-	-	✓	-
Byte	3	-	-	-	✓

AHB Arbitration Signals

Name	Source	Description
HBUSREQx	Master	Bus request
HLOCKx	Master	Locked transfers
HGRANTx	Arbiter	Bus grant
HMASTER[3:0]	Arbiter	Master number
HMASTLOCK	Arbiter	Locked sequence
HSPLITx[15:0]	Slave (SPLIT-capable)	Split completion request

Arbitration Signals (cont.)

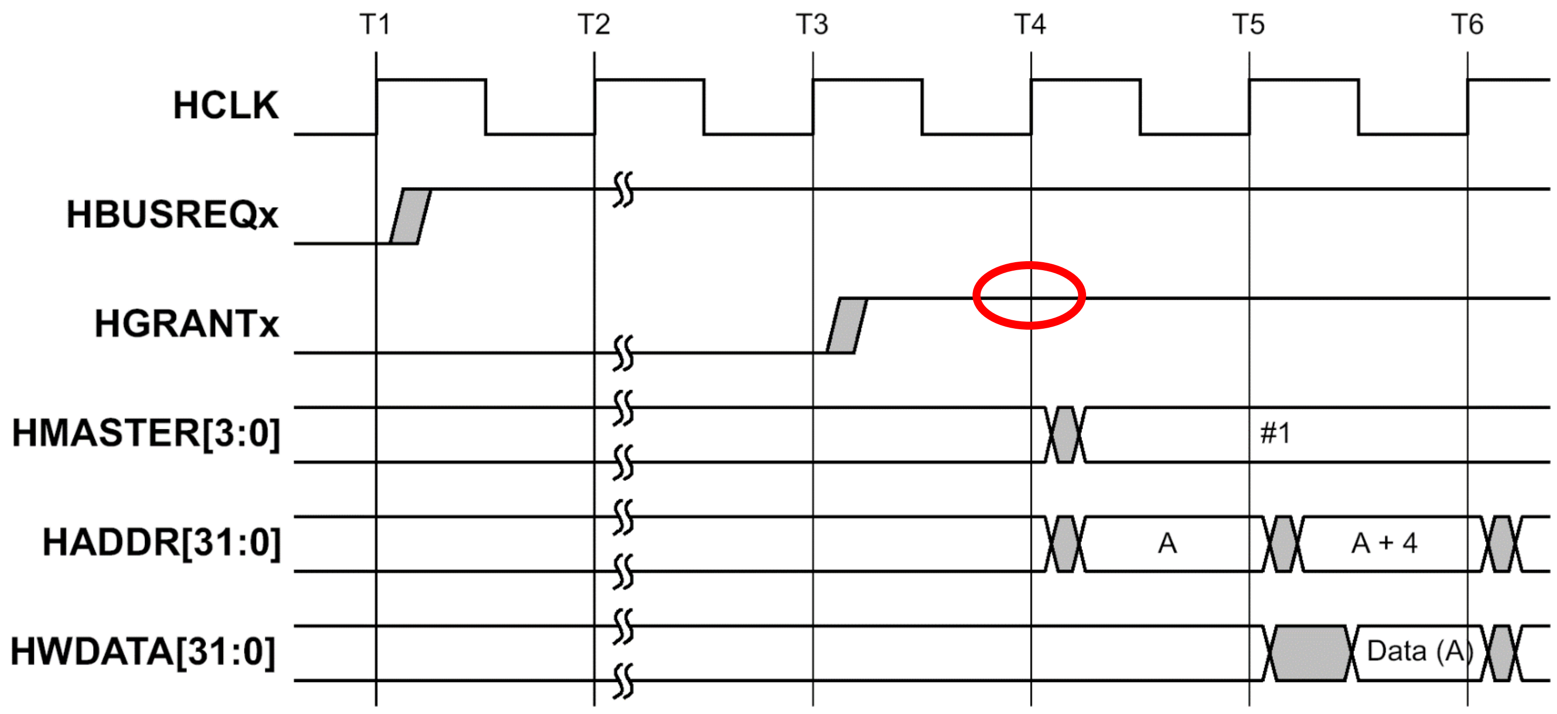
- HBUSREQ
  - Bus request
- HLOCKx :
  - High: the master requires locked access to the bus
- HGRANTx
  - Indicate the master x accessible to the bus
  - Master x gains ownership: HGRANTx=1 & HREADY=1

Arbitration Signals (cont.)

- HMASTER[3:0]
  - Indicate which master is transferring, information for splitting
- HMASTLOCK
  - Indicate the master is performing a locked transfer
- HSPLITx[15:0]
  - Used by the slave to indicate the arbiter which master should be allowed to re-attempt a split transaction
  - Each bit corresponds to a single master

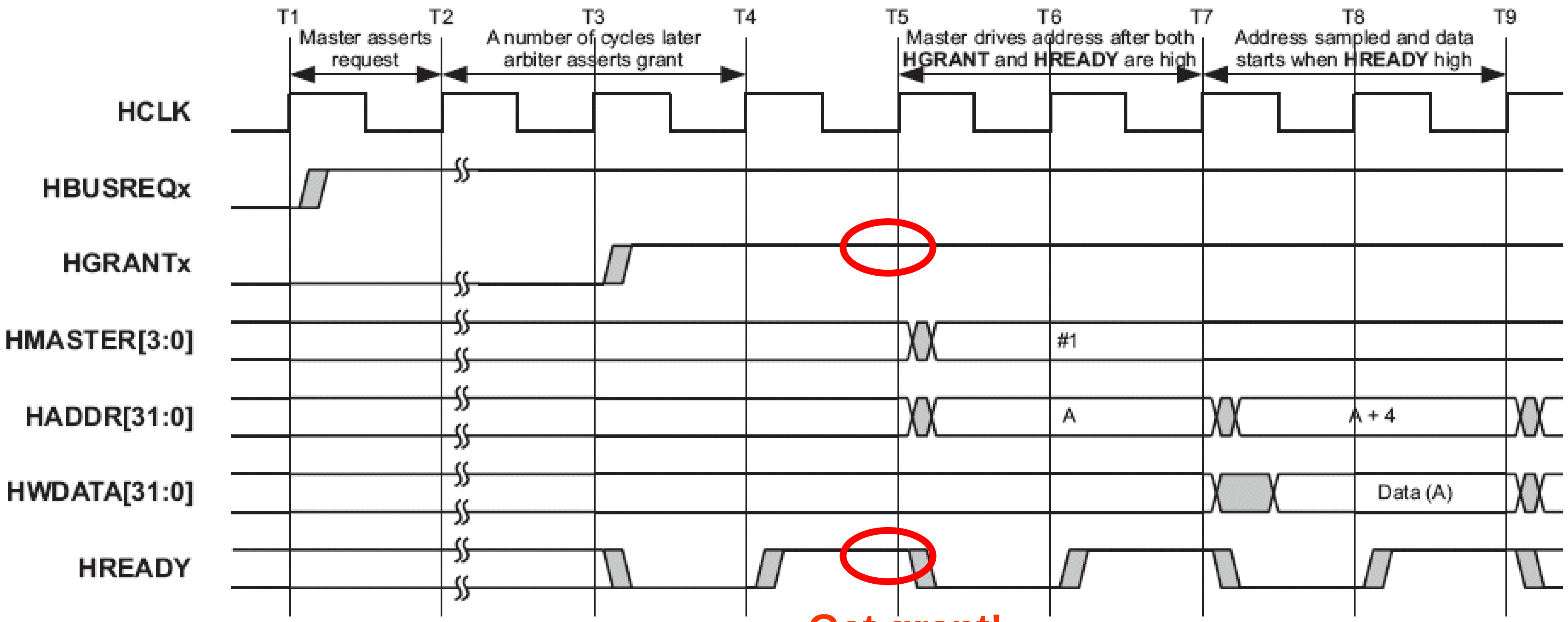
Arbitration Example (1)

- Granting access with no wait state



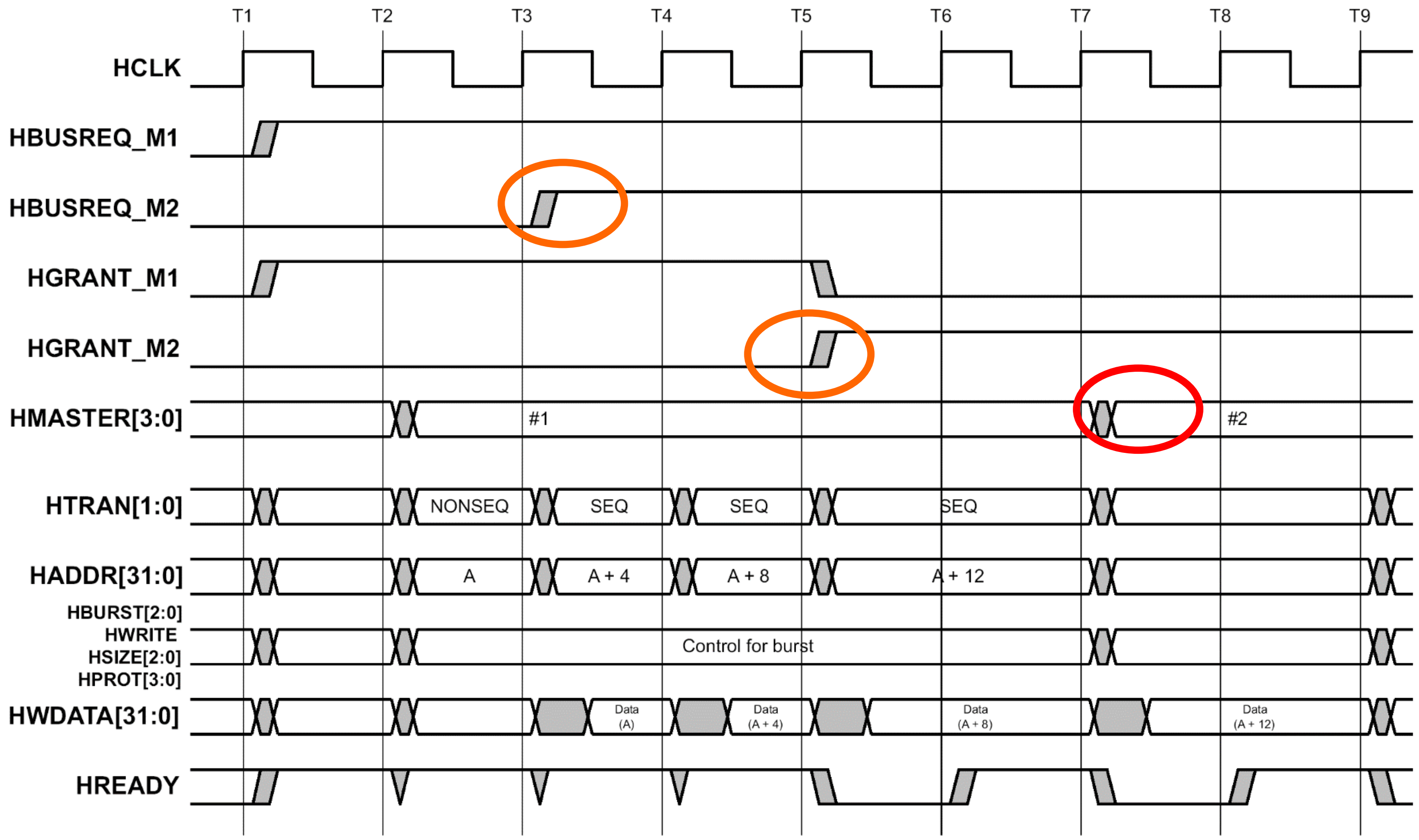
Arbitration Example (2)

- Granting access with wait states

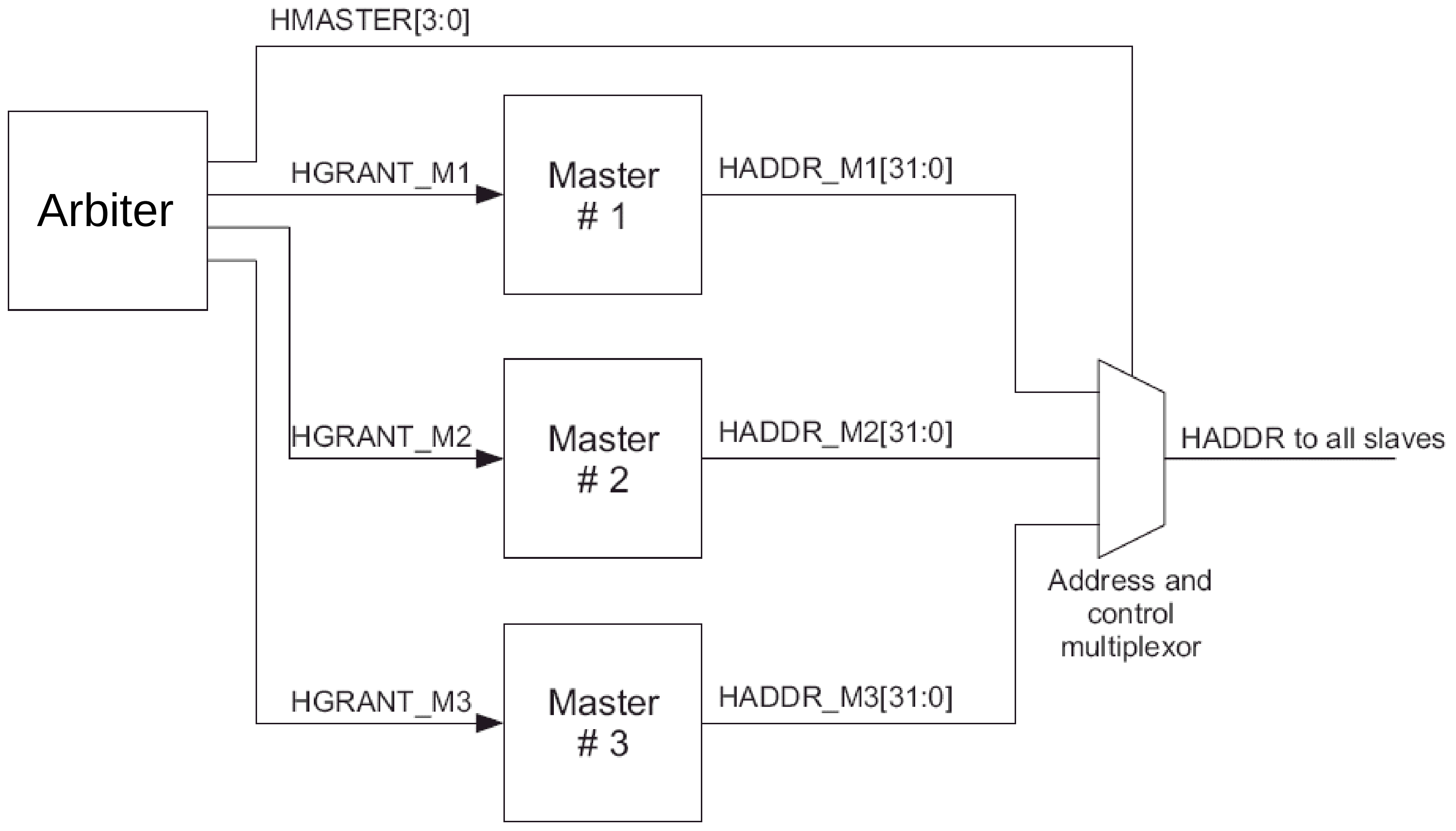

T1
T2
T3
T4
T5
T6
T7
T8
T9
Master asserts request
A number of cycles later arbiter asserts grant
Master drives address after both HGRANT and HREADY are high
Address sampled and data starts when HREADY high
HCLK
HBUSREQx
HGRANTx
HMASTER[3:0]
HADDR[31:0]
HWDATA[31:0]
HREADY
#1
A
A + 4
Data (A)

Get grant!

Arbitration Example (3)



Bus Master Grant Signals



Notes

- For a fixed length burst, it is not necessary to continue request the bus
- For a undefined length burst, the master should continue to assert the request until it has started the last transfer
- If no master requests the bus, grant to the default master with HTRANS=IDLE
- It is recommended that the master inserts an IDLE transfer after any locked sequence to provide the opportunity for changing arbitration

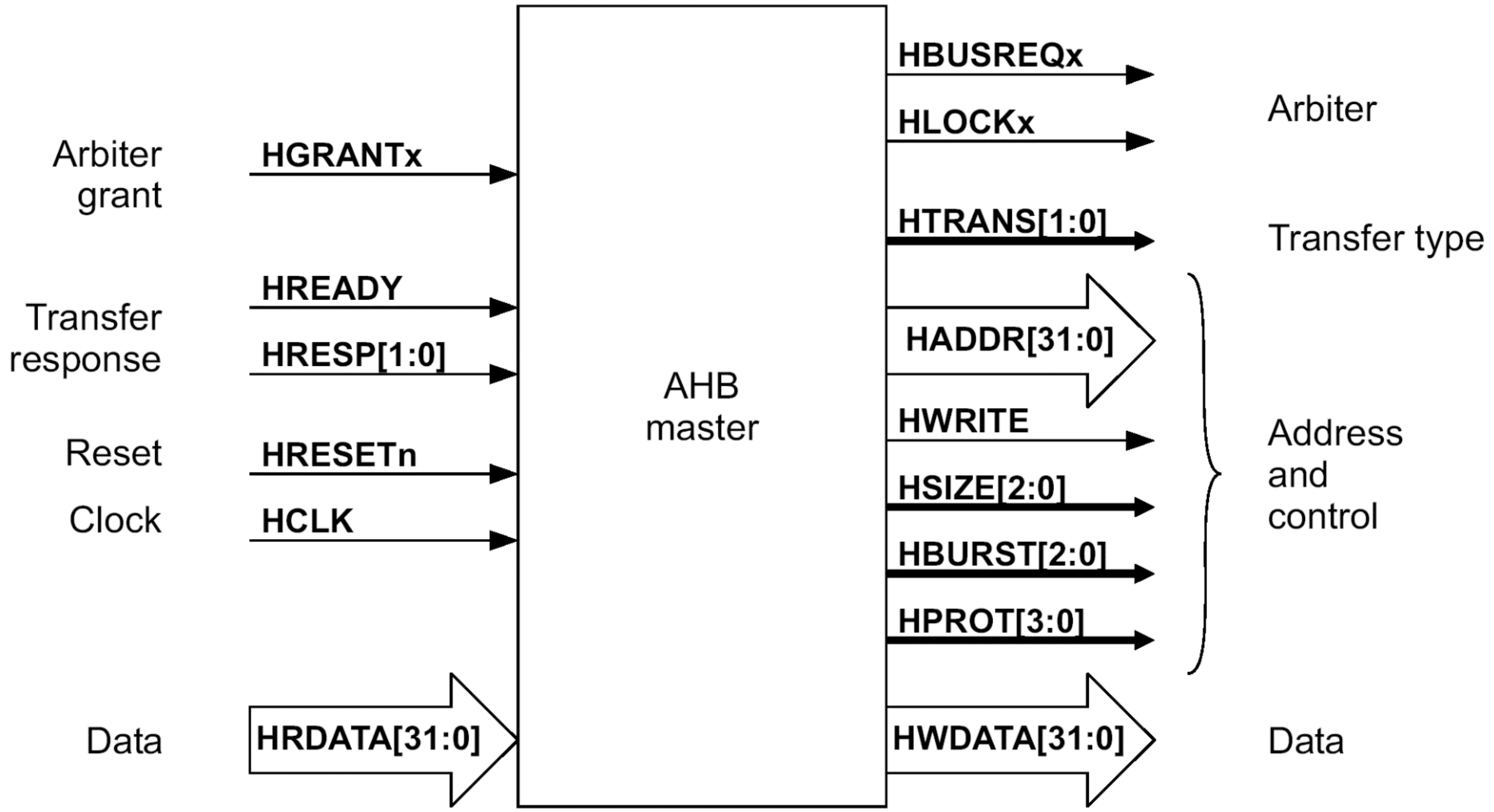
Split Transfer Sequence

- The master starts the transfer.
- If the slave decides that it may take a number of cycles to obtain the data, it **gives a SPLIT transfer response**. The slave **record the master number**, HMASTER. Then the arbiter **change the priority** of the masters.
- The arbiter grants other masters, bus master handover.
- When the slave is ready to complete the transfer, it asserts the appropriate bit of the HSPLITx bus to the arbiter.
- The arbiter restores the priority
- The arbiter will grant the master so it can re-attempt the transfer
- Finish

Preventing Deadlock

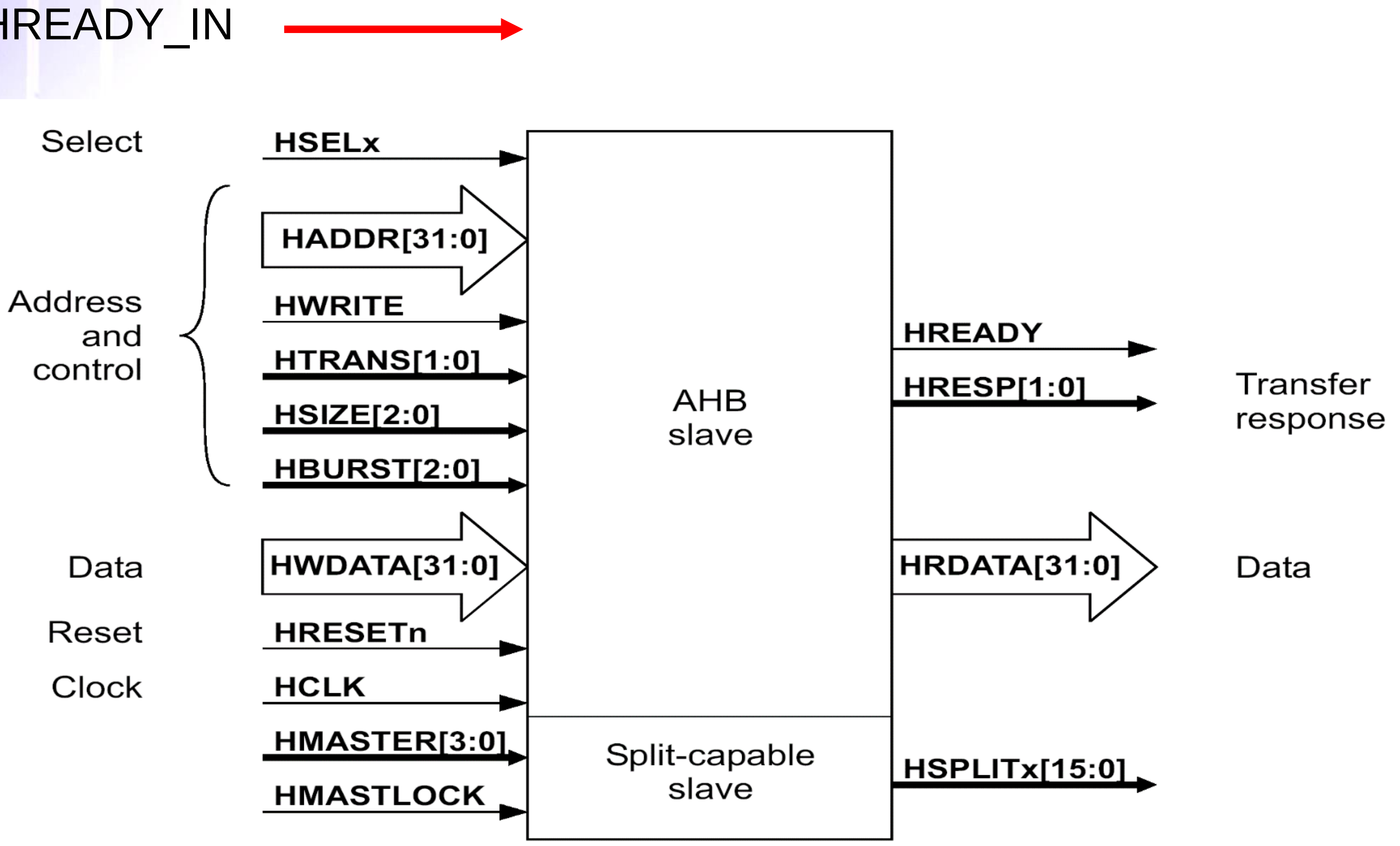
- It is possible for deadlock if a number of different masters attempt to access a slave which issues SPLIT or RETRY
- **The slave can withstand a request from every master in the system, up to a maximum of 16.** It only needs to record the master number. (can ignore address and control)
- **A slave which issues RETRY responses must only be accessed by one master at a time.**
  - Some hardware protection mechanisms, such as ERROR message, can be used.

AHB Master Interface

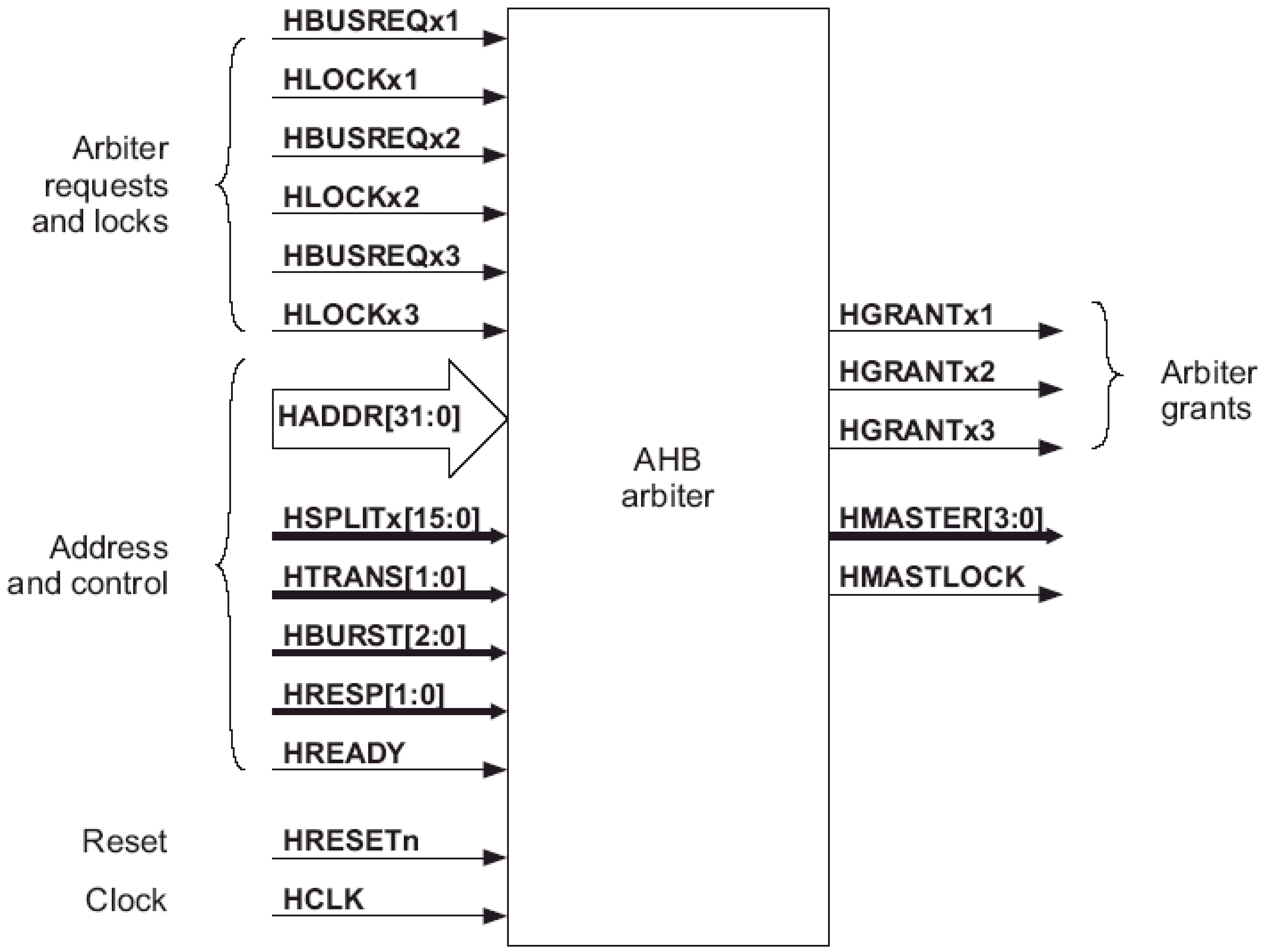


AHB Slave Interface

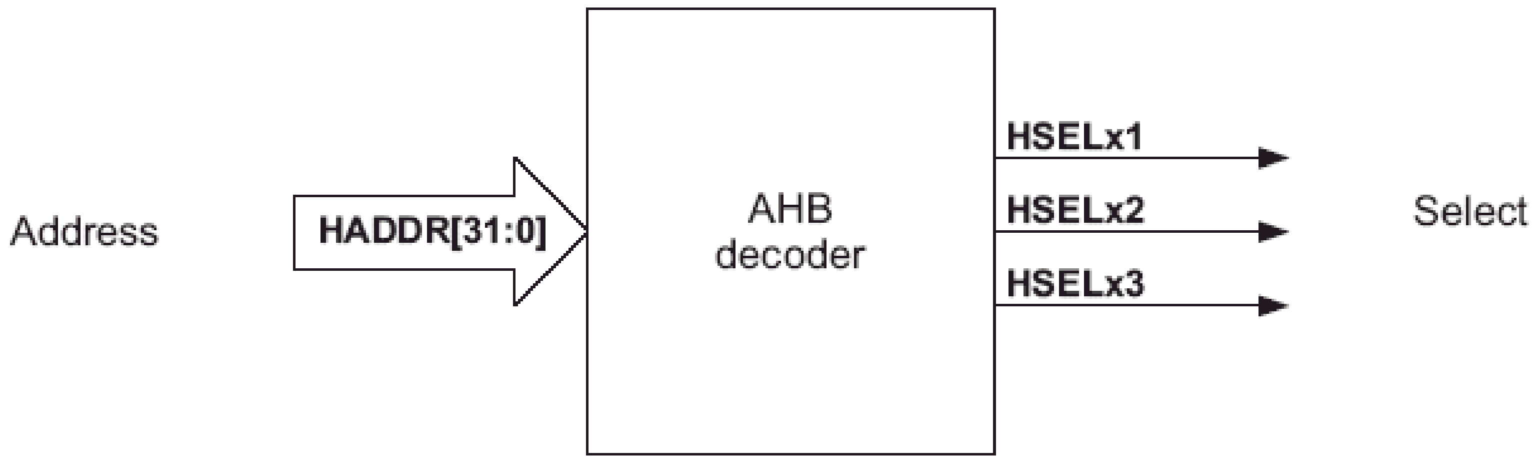
HREADY_IN



AHB Arbiter


Arbiter requests and locks
HBUSREQx1
HLOCKx1
HBUSREQx2
HLOCKx2
HBUSREQx3
HLOCKx3
Address and control
HADDR[31:0]
HSPLITx[15:0]
HTRANS[1:0]
HBURST[2:0]
HRESP[1:0]
HREADY
Reset
HRESETn
Clock
HCLK
AHB arbiter
HGRANTx1
HGRANTx2
HGRANTx3
Arbiter grants
HMASTER[3:0]
HMASTLOCK

AHB Decoder



Review on AHB

- Main components
  - Master, slaves, arbiter, decoder
- How the transfer progress
  - The pipelined scheme
- How to increase the performance
  - Burst read/write
- Arbitration
  - Bus ownership handover

Other Topics

- AHB-Lite
- Multi-layer AHB

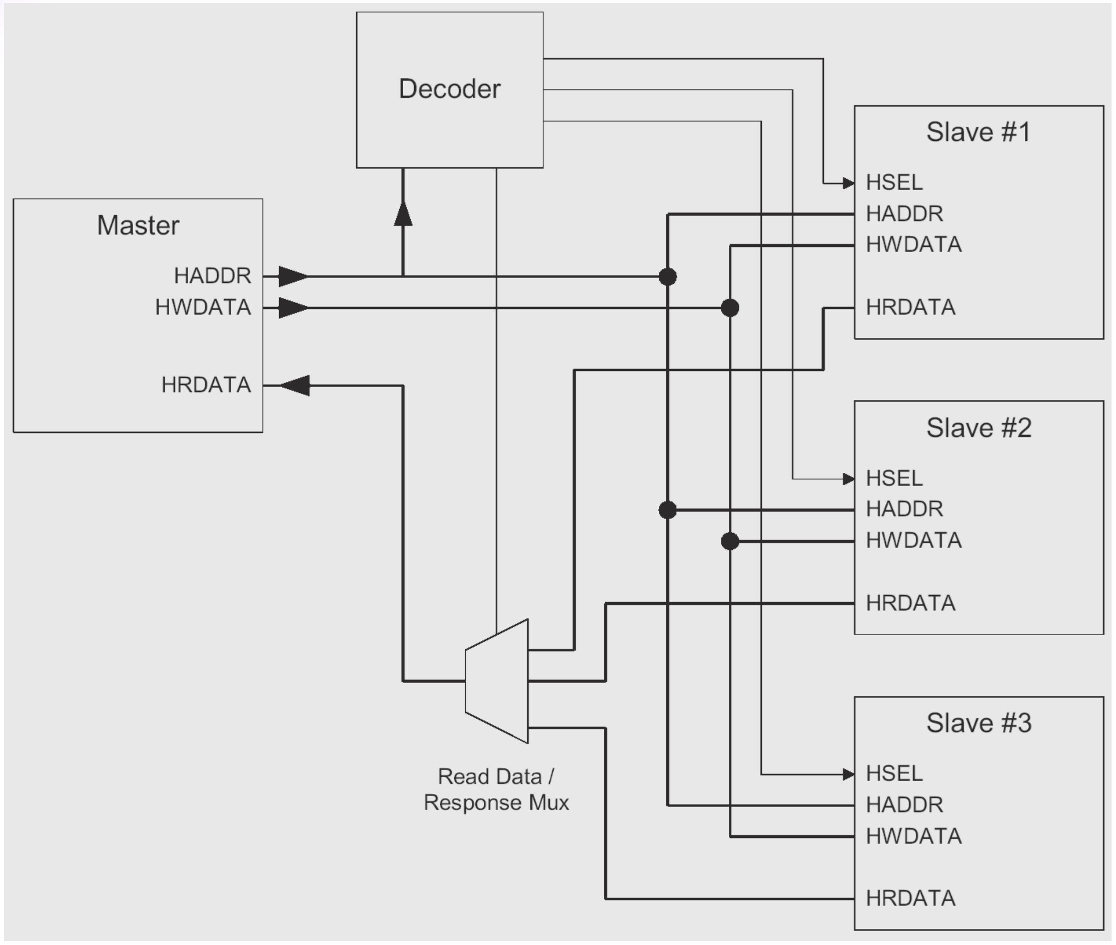
AHB-Lite

- AHB-Lite
  - A subset of the full AHB specification
  - Only one single bus master used
  - No need of the request/grant protocol to the arbiter
  - No arbiter
  - No Retry/Split response from slaves
  - No master-to-slave multiplexor

AHB-Lite Interchangeability

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

AHB-Lite Block Diagram

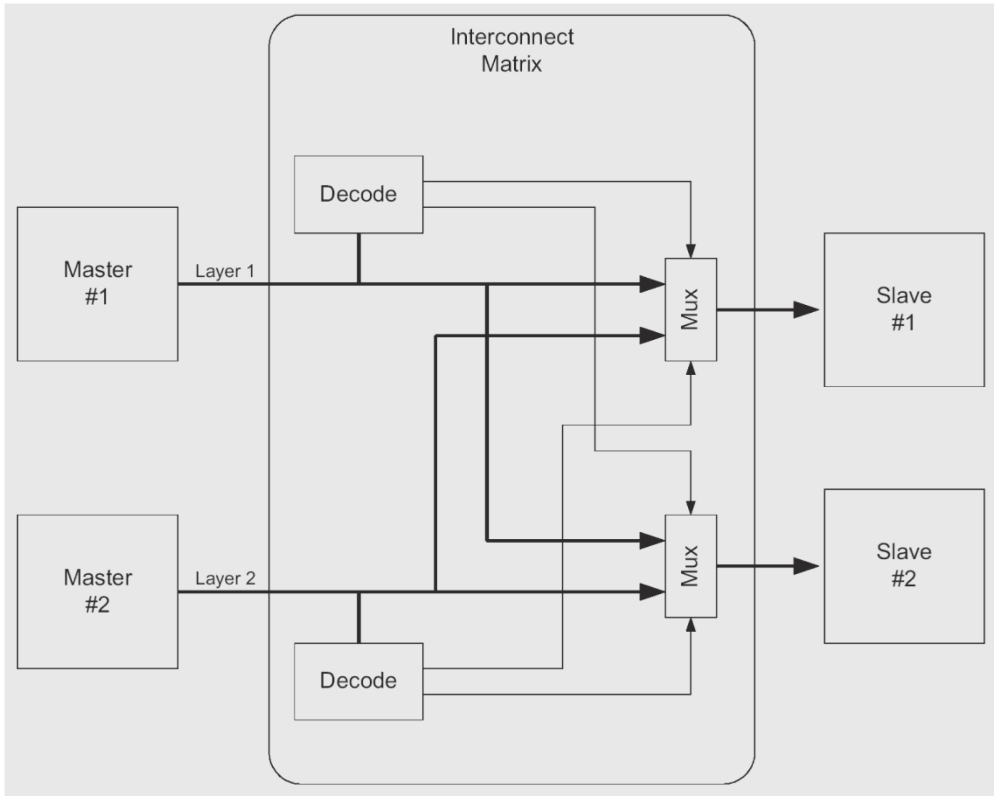


Multi-layer AHB

- Multi-layer AHB
  - Enables parallel access paths between multiple masters and slaves by an interconnection matrix
  - Increase the overall bus bandwidth
  - More flexible system architecture
    - Make slaves local to a particular layer
    - Make multiple slaves appear as a single slave to the interconnection matrix
    - Multiple masters on a single layer

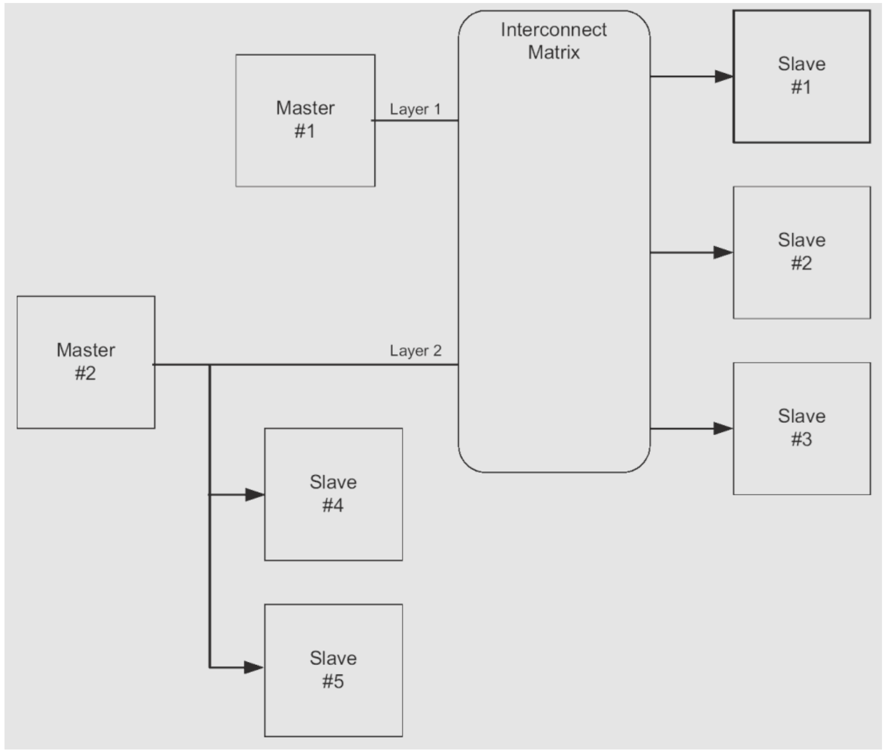
Multi-layer AHB

- A simple multi-layer system



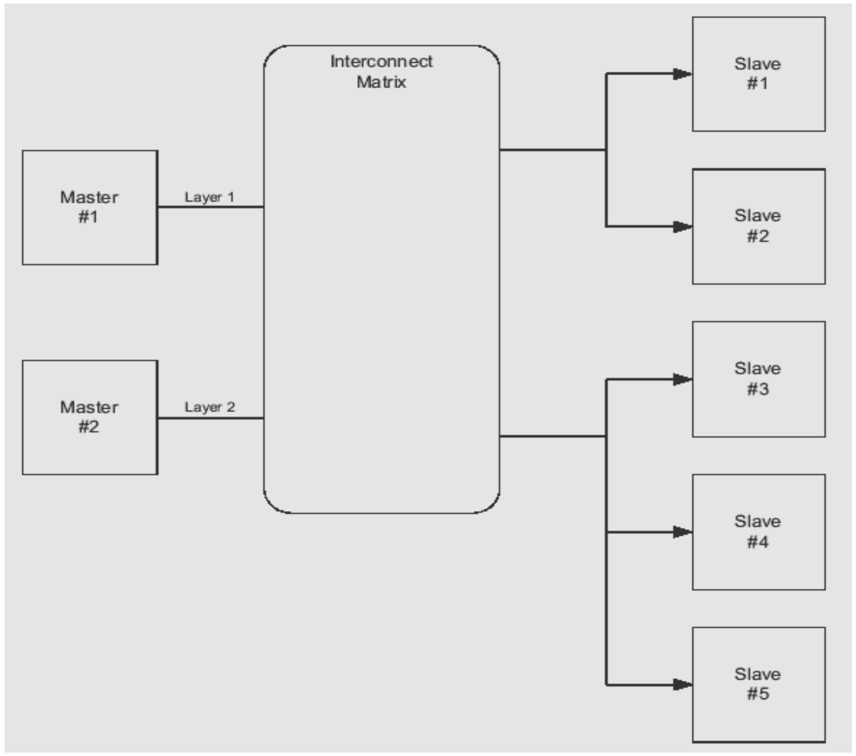
Multi-layer AHB

- Local slaves
  - Slave #4 and Slave #5 can only be accessed by Master #2



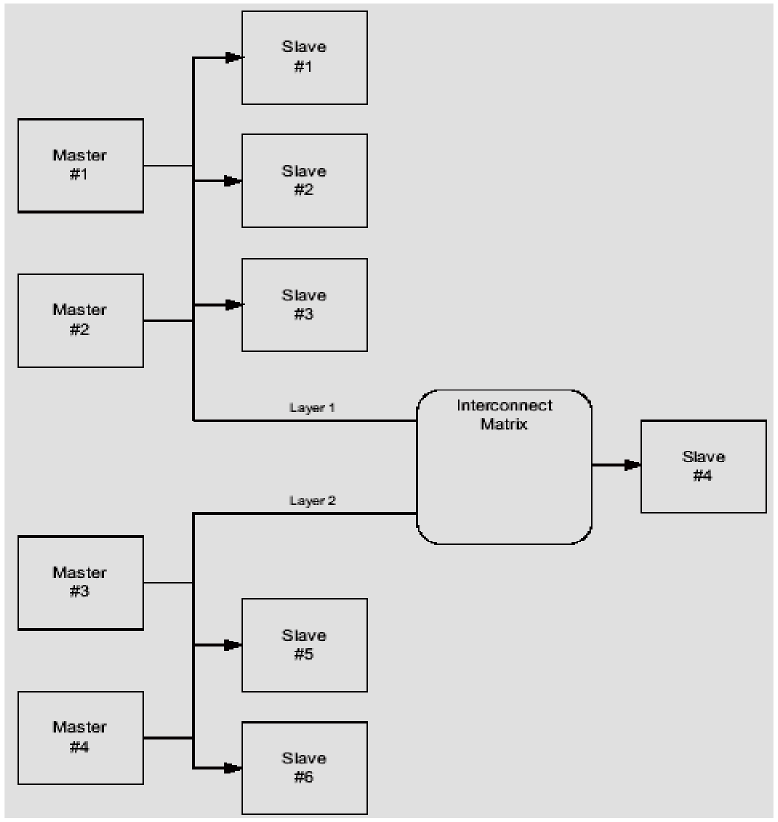
Multi-layer AHB

- Multiple slaves on one slave port
  - Combine low-bandwidth slaves together
  - Combine salves usually accessed by the same master together



Multi-layer AHB

- Multiple masters on one layer
  - Combine masters which have low-bandwidth requirements together
  - Combine special masters together



Communications between Different IPs

Communications

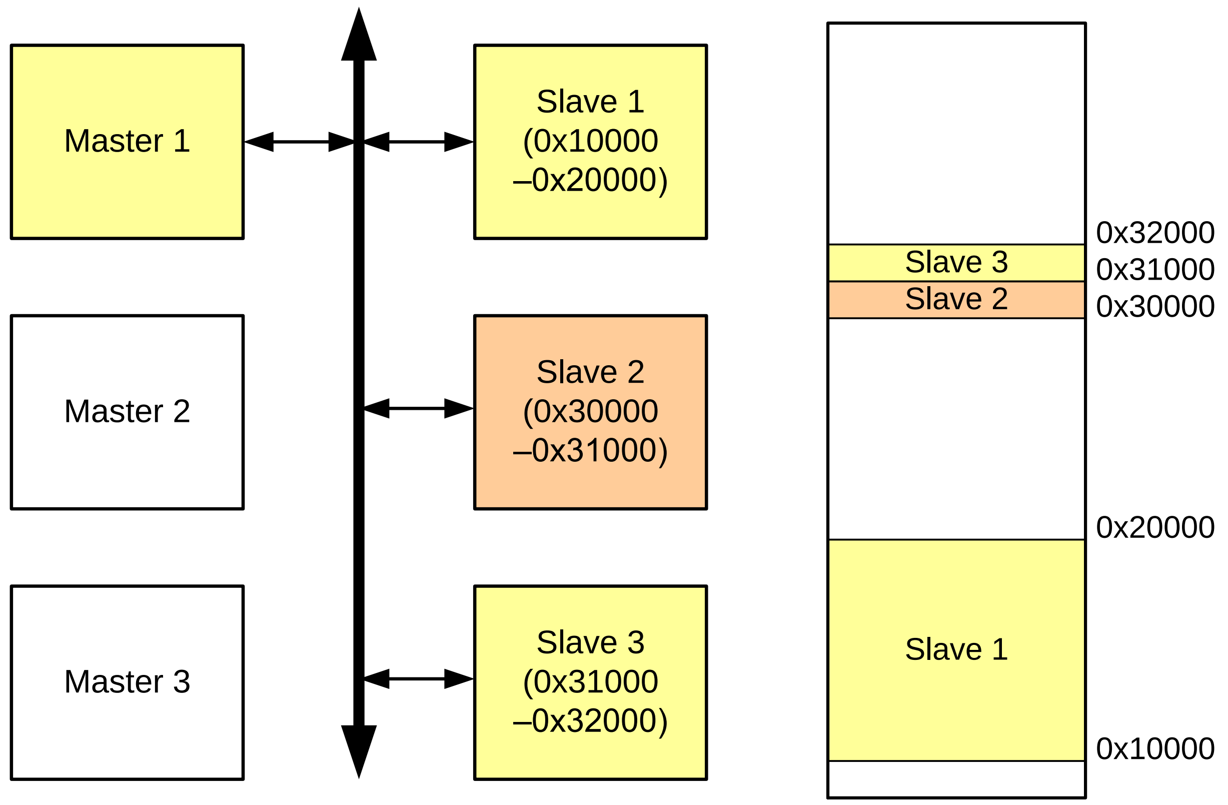
- CPU (master) ←→ IP (slave)
- IP (master) ←→ IP (slave)

Memory Mapped I/O

- Each **slave** occupies a range of (>1KB) address space in the system
- All the slaves are **addressable**
- Memory mapped register/memory
- CPU/IP and read/write data to other IP **as read/write data from/to memory**

Communication between IPs

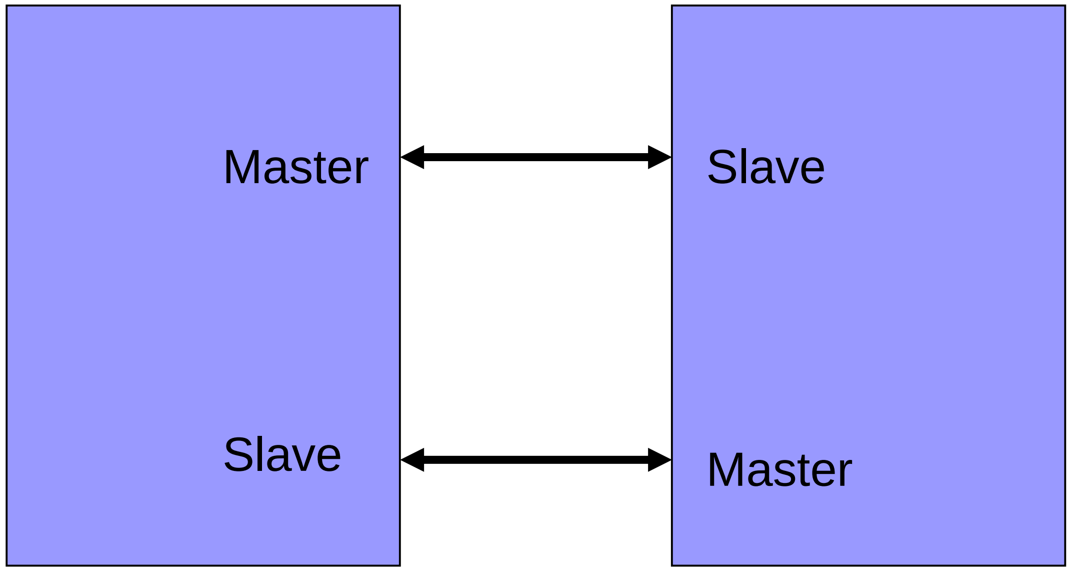
- After the master is granted by the arbiter, it can access all the slaves on the bus



Write(address, data)
Read(address, data)

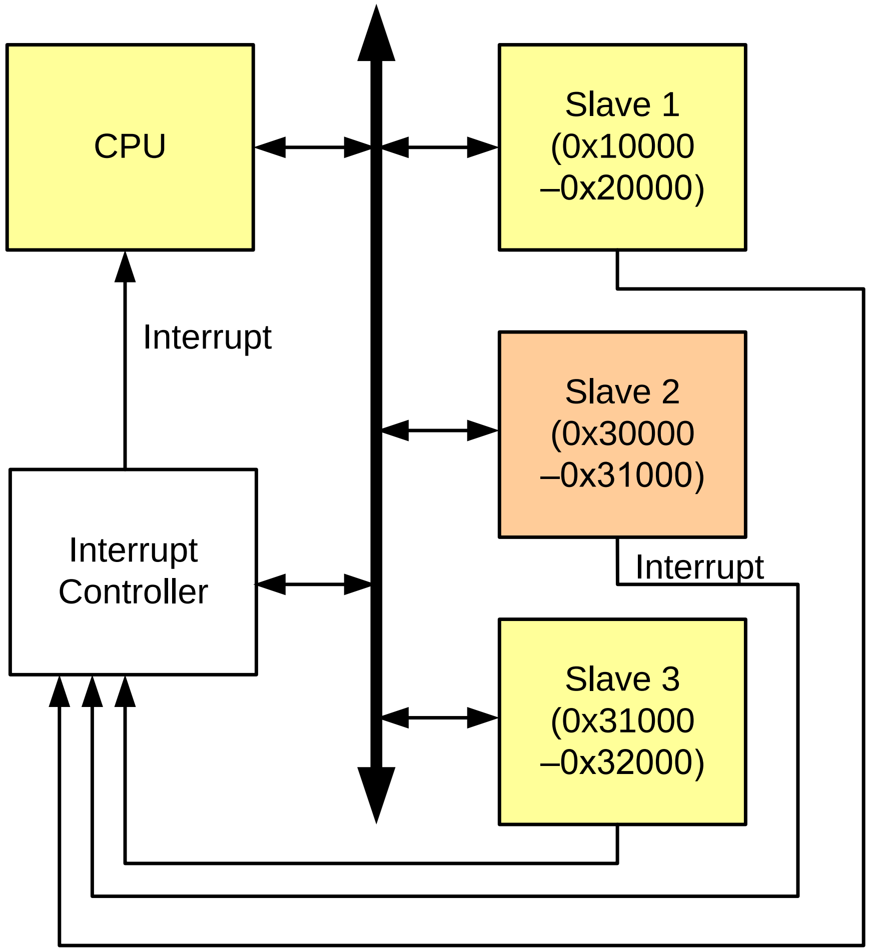
Ex:
Write(0x30020, 0x0)
Read(0x30100, &temp)

An IP Can Have Both Master and Slave I/F

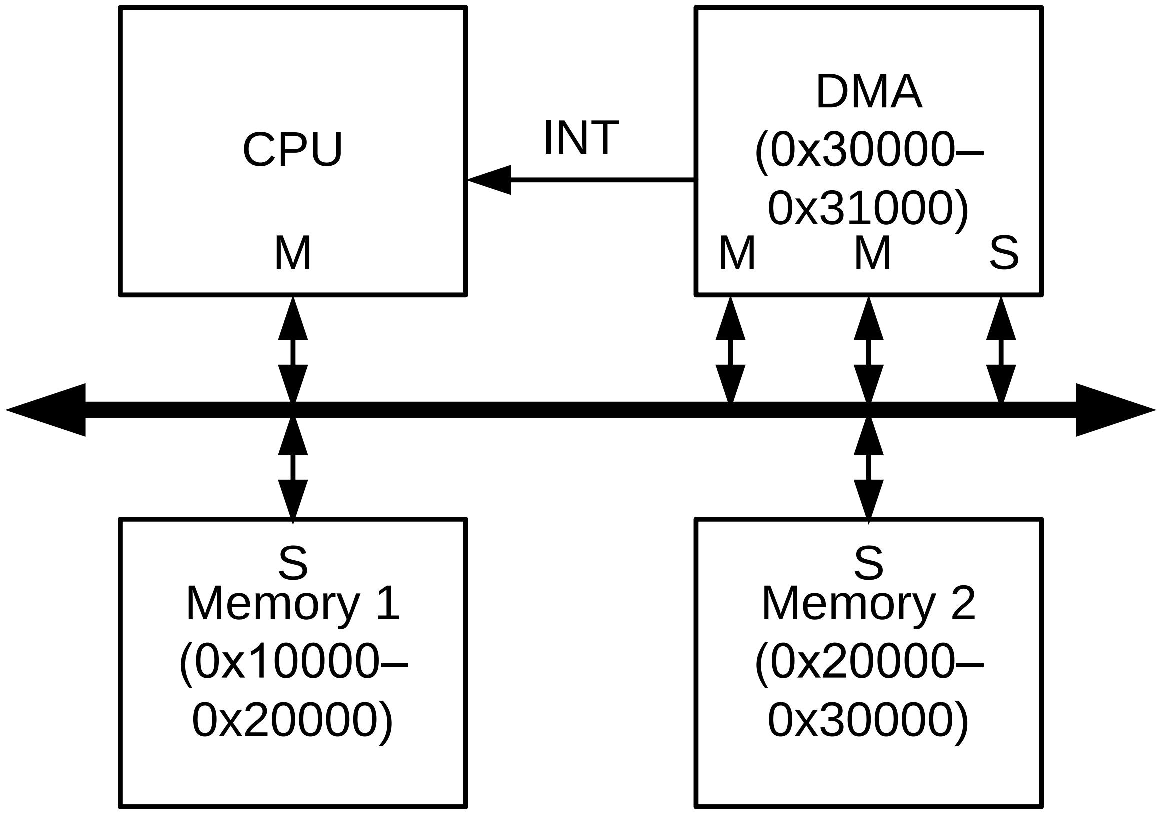


Communication between CPU and IP

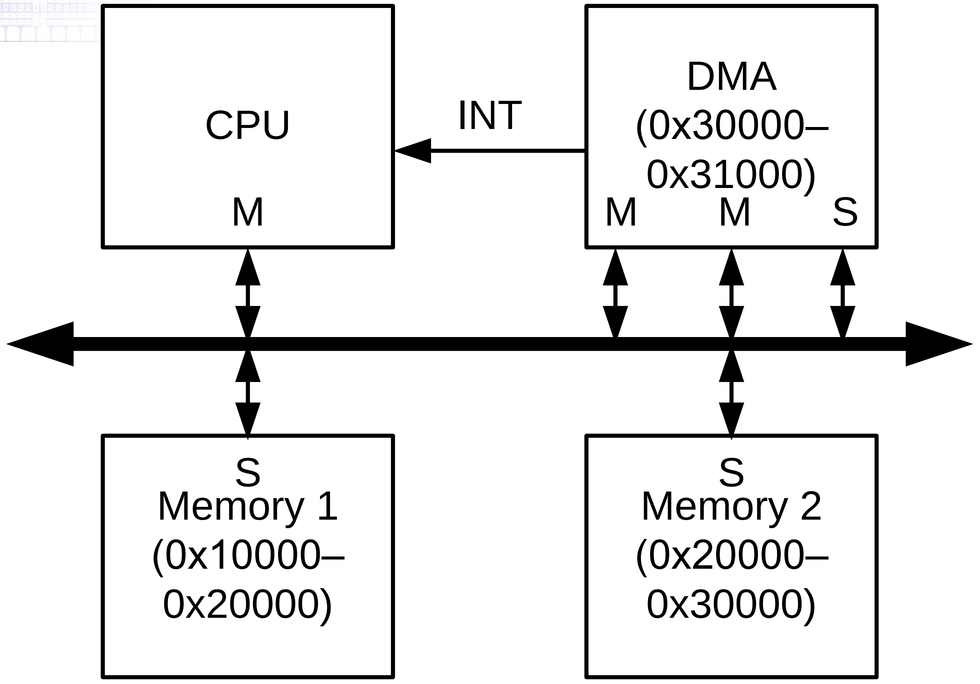
- CPU is always the master
- The IP is always the slave
- The IP can initiate the feedback with **interrupt**
- After interrupt, the CPU enters interrupt mode, and the interrupt is handled with **interrupt service routine (ISR)**


CPU
Slave 1 (0x10000 –0x20000)
Interrupt
Slave 2 (0x30000 –0x31000)
Interrupt Controller
Interrupt
Slave 3 (0x31000 –0x32000)

Example: DMA


CPU
M
INT
DMA
(0x30000–
0x31000)
M
M
S
S
Memory 1
(0x10000–
0x20000)
S
Memory 2
(0x20000–
0x30000)

Example: DMA

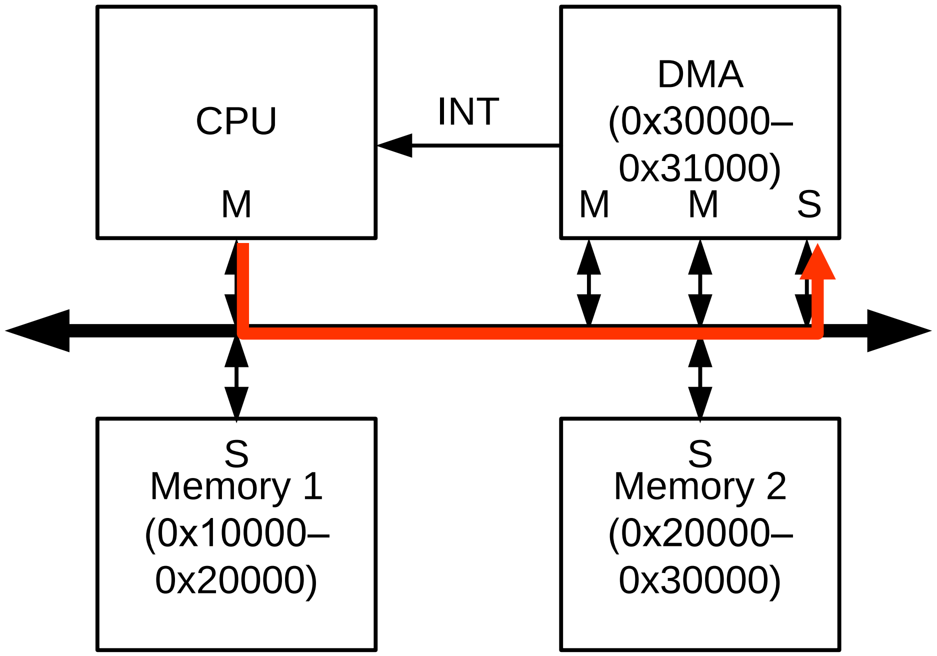


Address	Function
0x00	1: start, 0: stop
0x04	Status. 0: ready; 1: busy
0x08	Source address
0x0C	Destination address
0x10	Size

Example: DMA

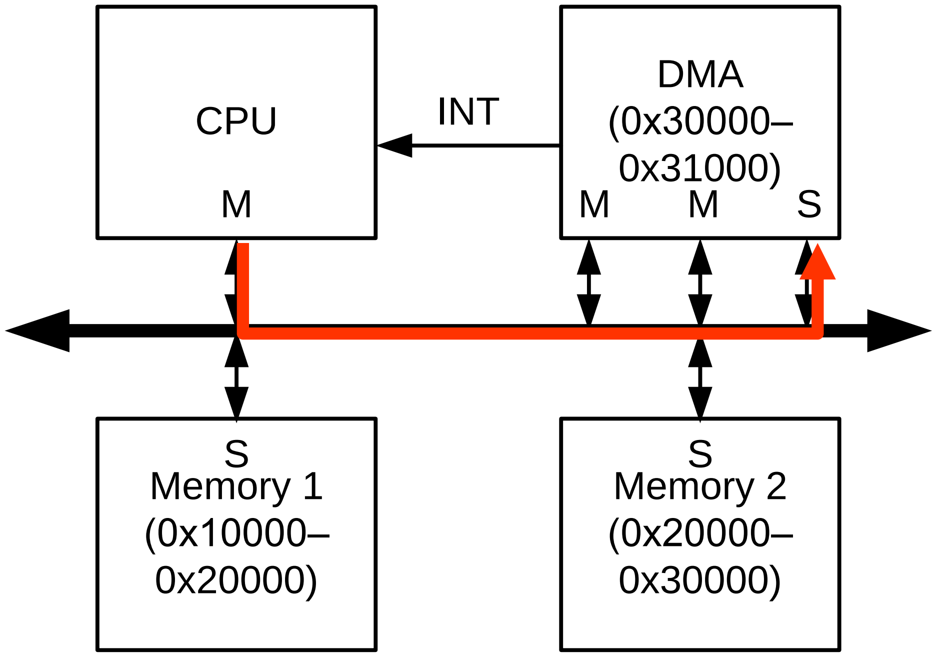
- Step 0: CPU check the status of DMA to make sure it is ready to be used

```
While(1)
{
    Read(0x30004, &status)
    if(status == 0)
        break;
}
```



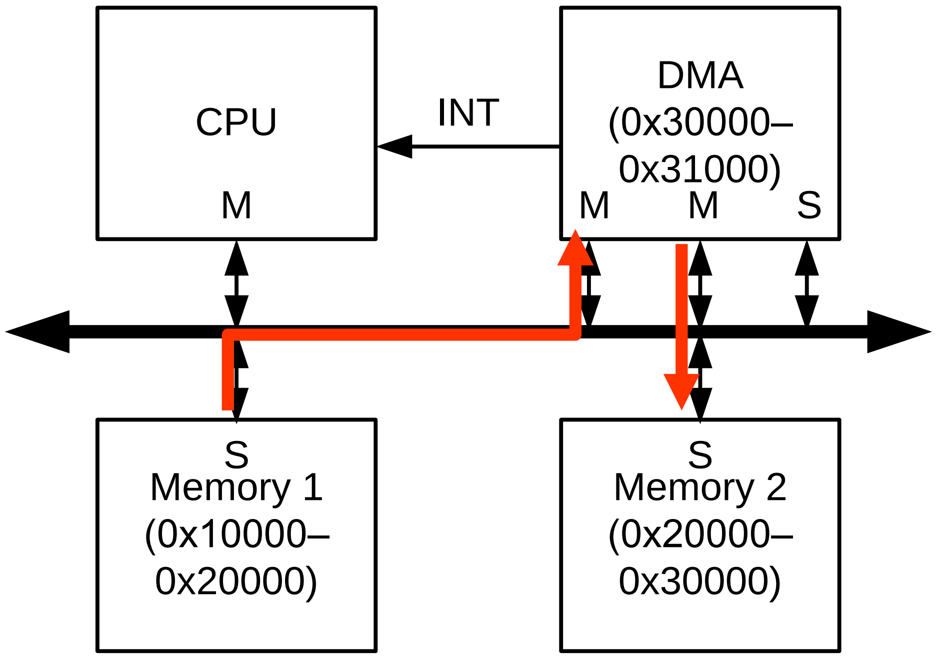
Example: DMA

- Step 1: CPU sets the (source address), (destination address), and (size) with the slave I/F
  - Write(0x30008, 0x10000)
  - Write(0x3000C, 0x20000)
  - Write(0x30010, 0x100)
- Step 2: starts DMA
  - Write(0x30000, 0x1)



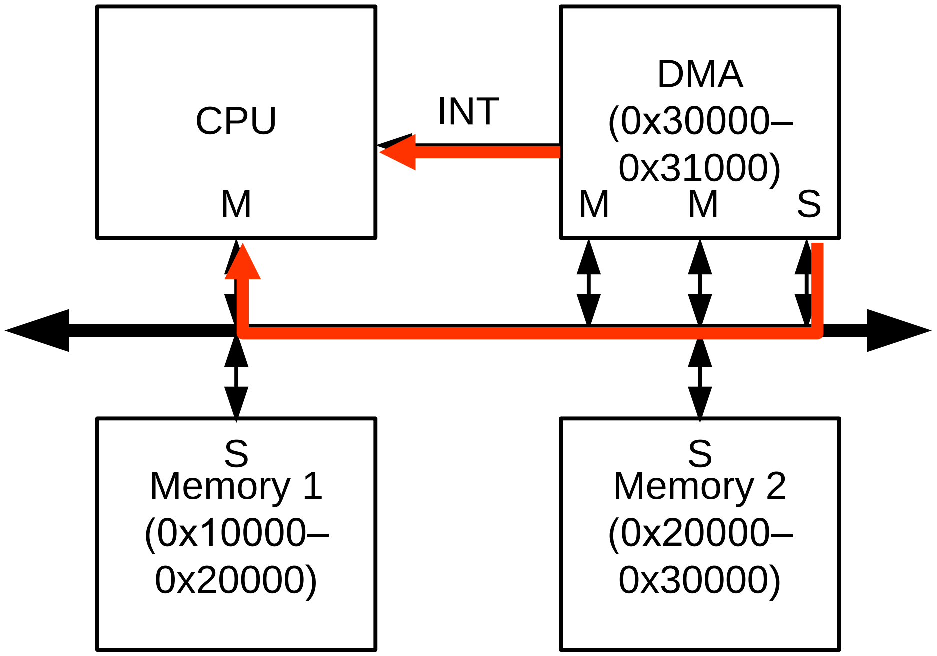
Example: DMA

- Step 3: DMA moves data from memory 1 to memory 2 with the two master I/F



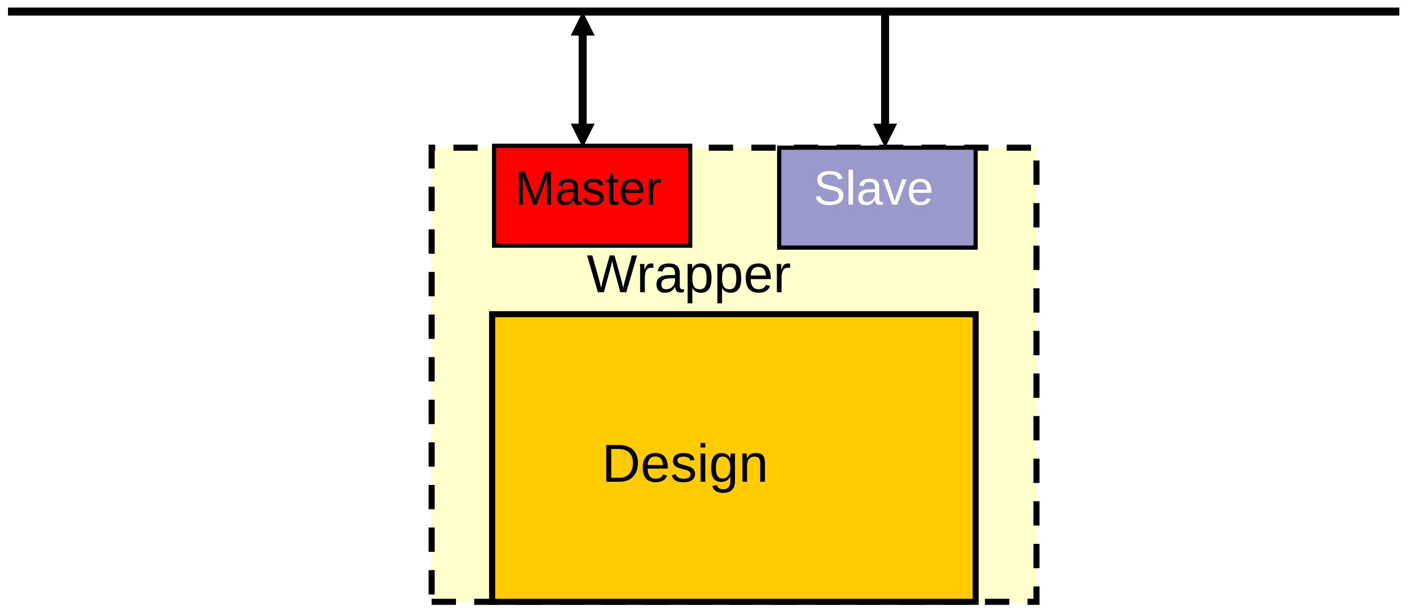
Example: DMA

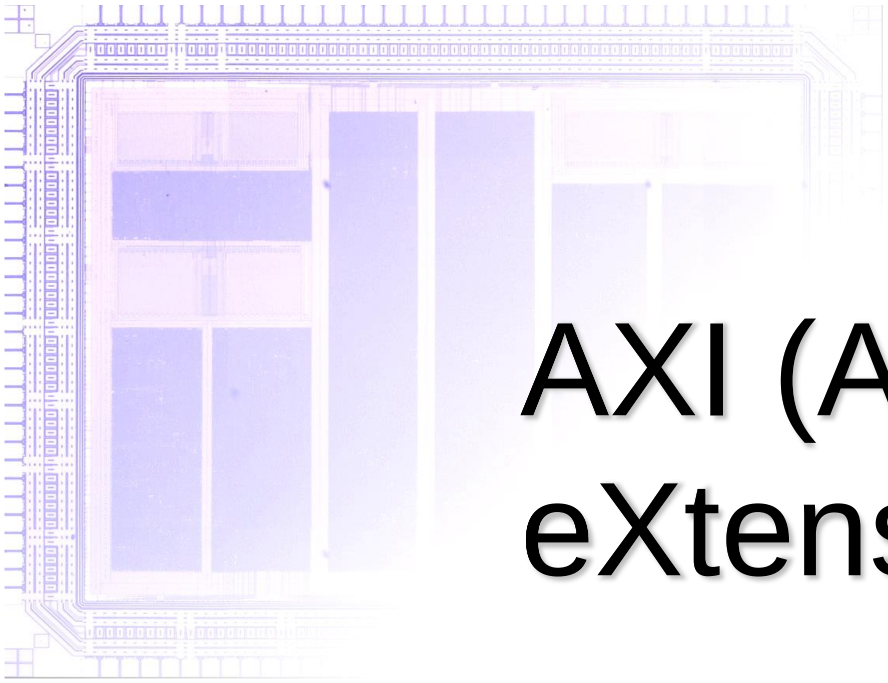
- Step 4: DMA interrupts CPU
- Step 5: CPU checks the status of DMA
  - Read(0x30004, &status)



Example: Hardware Accelerator with both Master and Slave I/F

- Both master port & slave port in your design
  - Read/write data via master port
  - Configured by the others via slave port
  - Such as some parameters set by the processor





AXI (Advanced eXtensible Interface)

Ref: AMBA AXI Protocol Specification v1.0

Objectives

- Be suitable for high-bandwidth and low-latency designs
- Enable high-frequency operation without using complex bridges
- Meet the interface requirements of a wide range of components
- Be suitable for memory controllers with high initial access latency
- Provide flexibility in the implementation of interconnect architectures
- Be backward-compatible with existing AHB and APB interfaces.

Key Features

- Separate address/control and data phases
- Support for unaligned data transfers using byte strobes
- Burst-based transactions with only start address issued
- Separate read and write data channels to enable low-cost *Direct Memory Access* (DMA)
- Ability to issue multiple outstanding addresses
- Out-of-order transaction completion
- Easy addition of register stages to provide timing closure
- Include optional extensions that cover signaling for low-power operation

Architecture

- AXI is burst-based
- Each transaction has address and control information on address channel that describes the nature of the data to be transferred
- Five channels: read and write address channels, read data channel, write data channel, write response channel
  - Consists of a set of information signals and uses a two-way **VALID** and **READY** handshake mechanism
  - The read data channel and write data channel also include a **LAST** signal to indicate when the transfer of the final data item within a transaction takes place.

Channels (1)

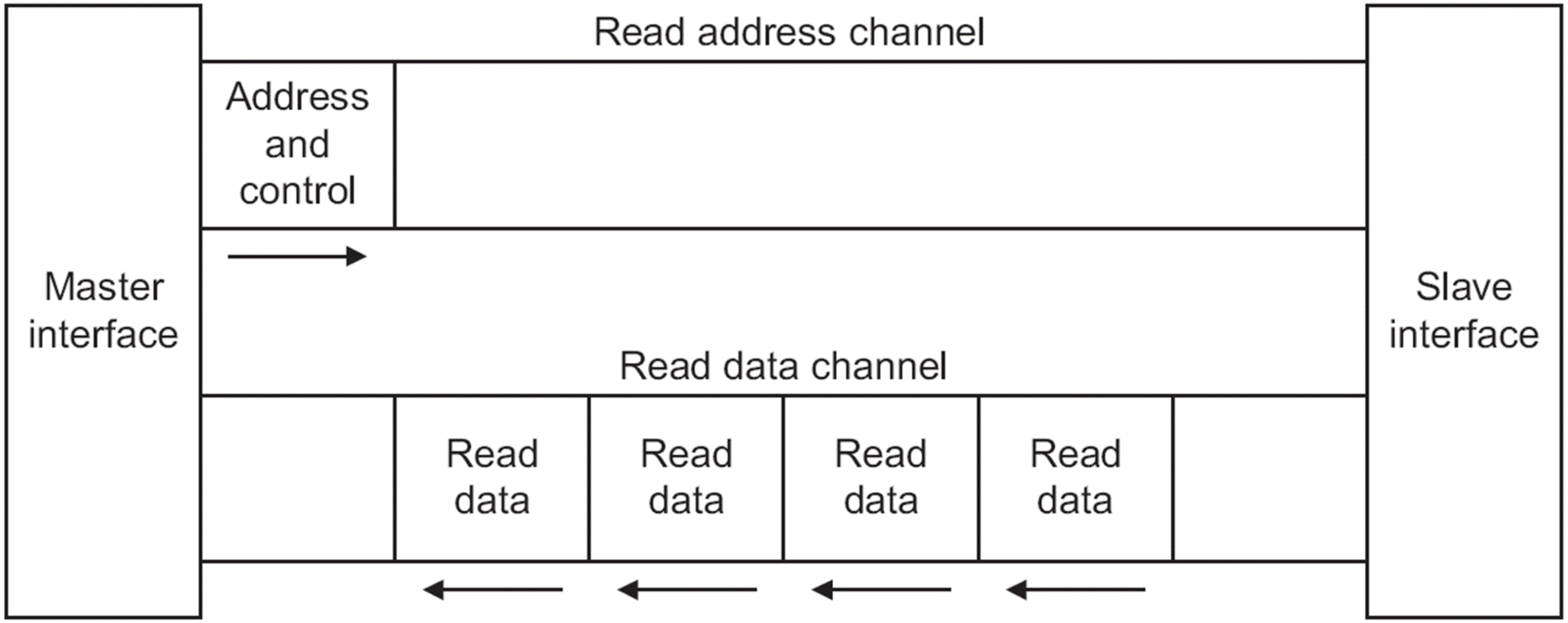
- Read and write address channels (AXXX)
  - Carry address and control information
  - Variable-length bursts, from 1 to 16 data transfers per burst
  - Bursts with a transfer size of 8-1024 bits
  - Wrapping, incrementing, and non-incrementing bursts
  - Atomic operations, using exclusive or locked accesses
  - System-level caching and buffering control
- Read data channel (RXXX)
  - Convey read data and read response
  - The data bus, which can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide
  - A read response indicating the completion status of the read transaction

Channels (2)

- Write data channel (WXXX)
  - The data bus, which can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide
  - **One byte lane strobe** for every eight data bits, indicating which bytes of the data bus are valid
  - Is always treated as buffered
- Write response channel (BXXX)
  - Completion signaling which occurs once for each burst

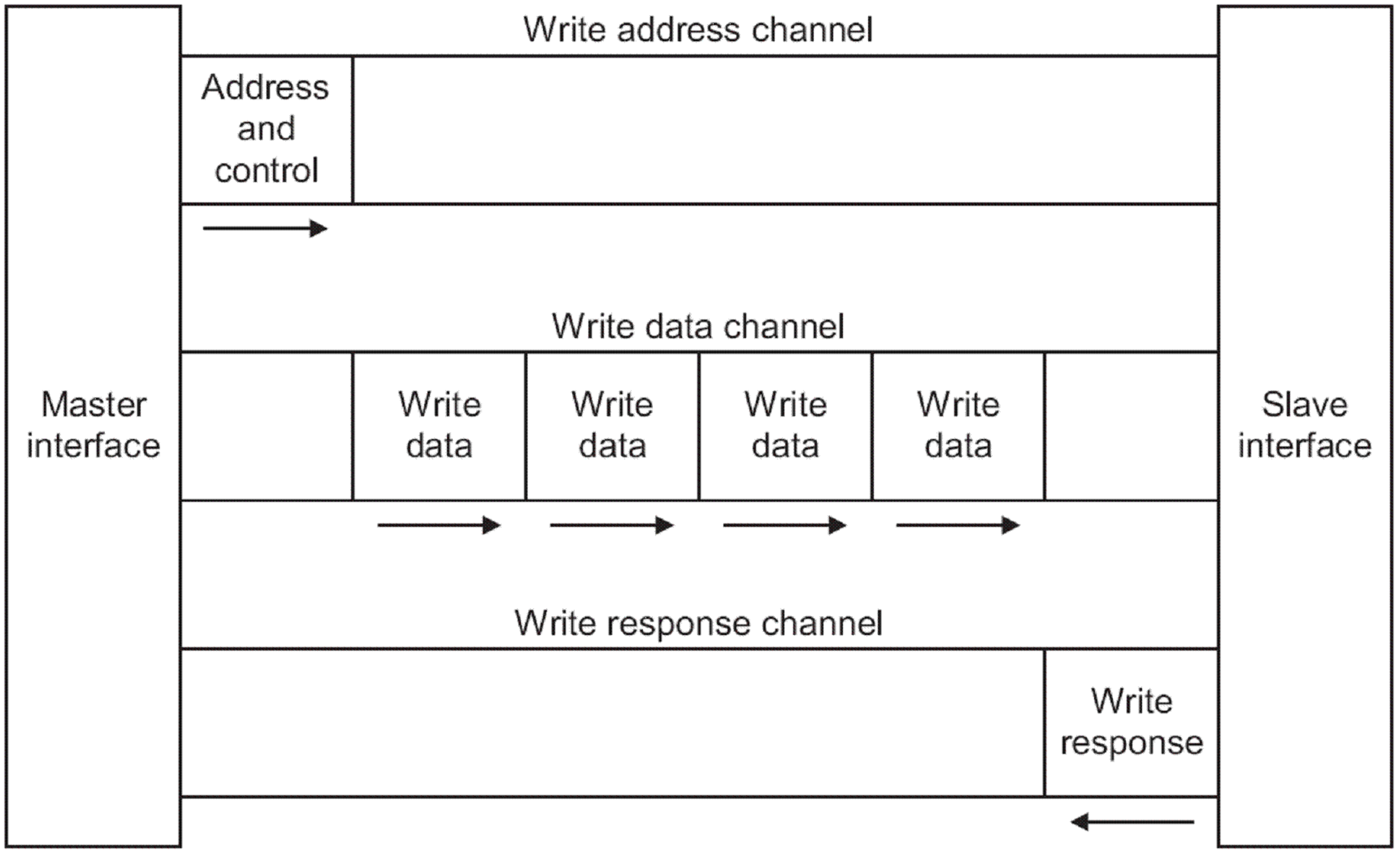
Channels (3)

- Channel architecture of reads

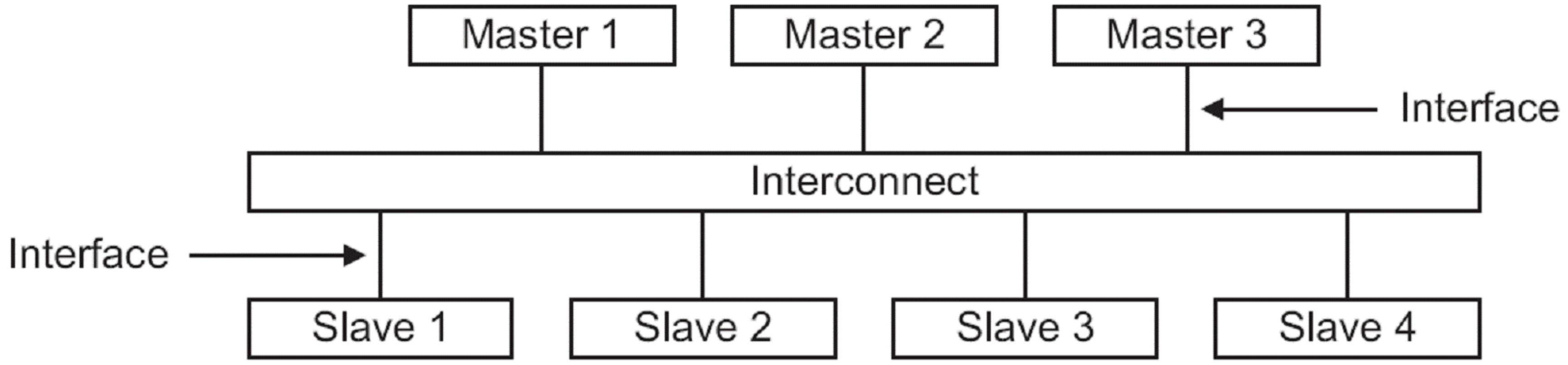


Channels (4)

- Channel architecture of writes



Interface and Interconnection



- Enables a variety of different interconnection implementations
  - Shared address and data buses
  - **Shared address buses and multiple data buses**
  - Multilayer, with multiple address and data buses

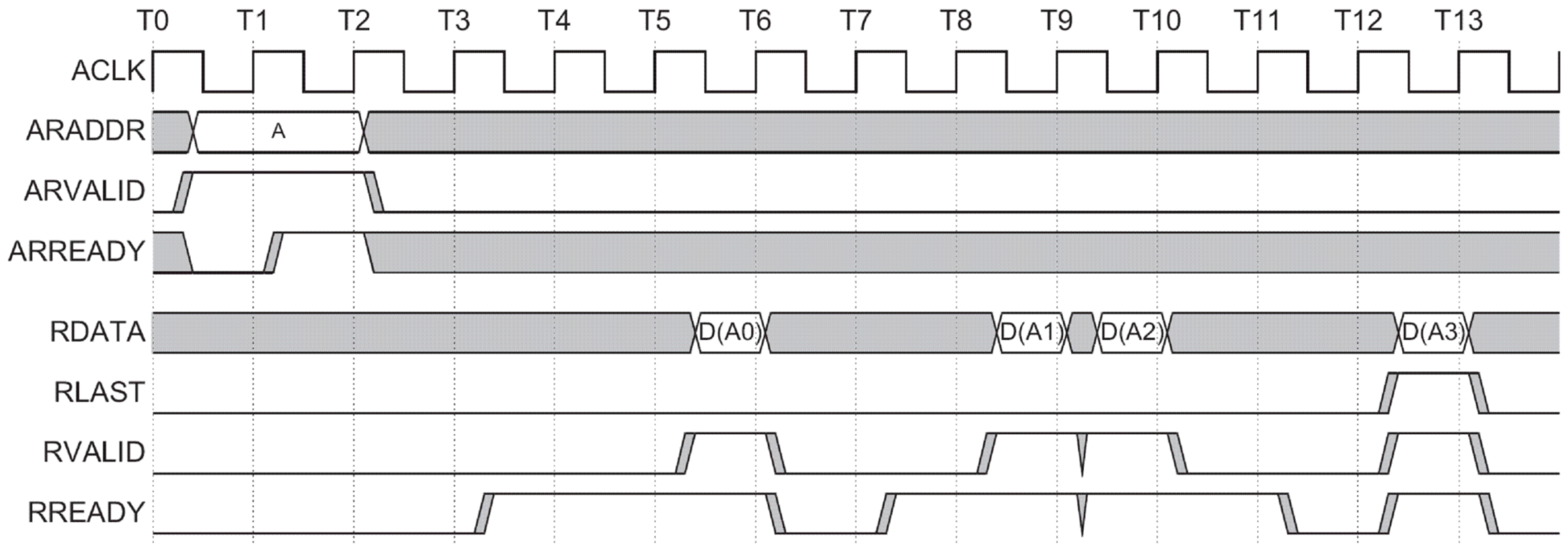
Register Slices

- AXI channels transfers information in only one direction, and there is no requirement for a fixed relationship

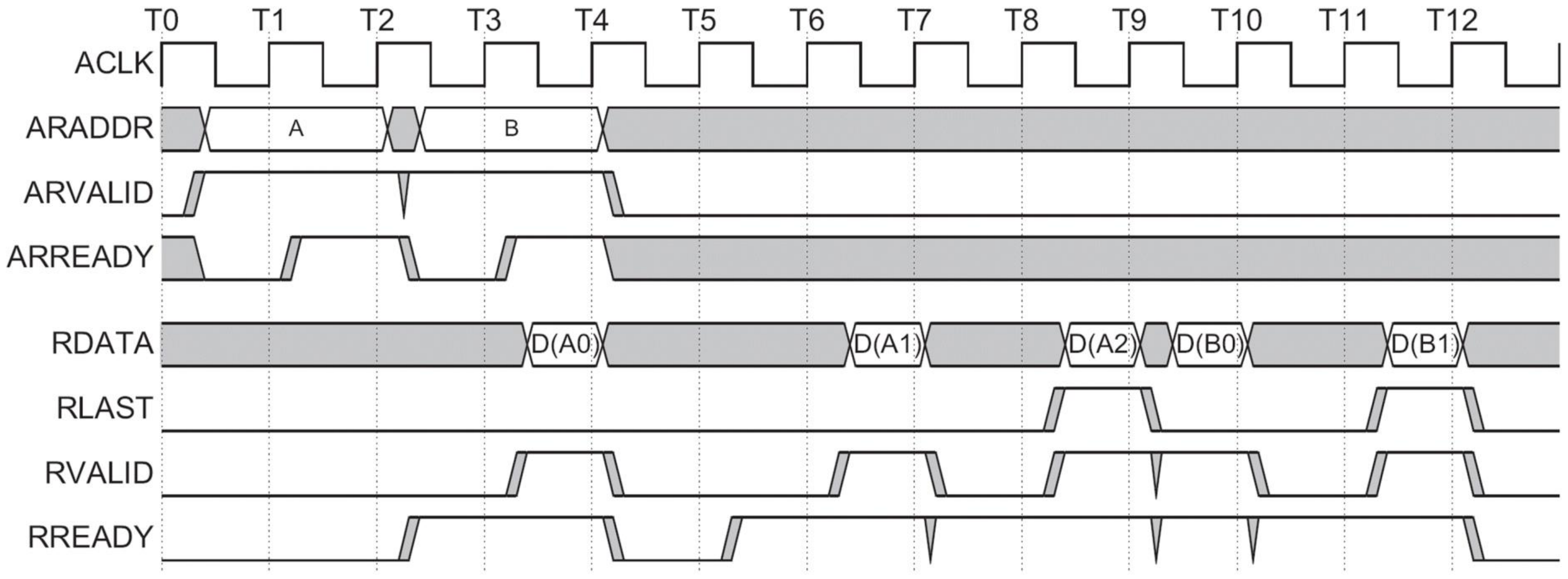
➔ Enables the insertion of a **register slice** in any channel

- Trade-off between cycles of latency and maximum frequency of operation

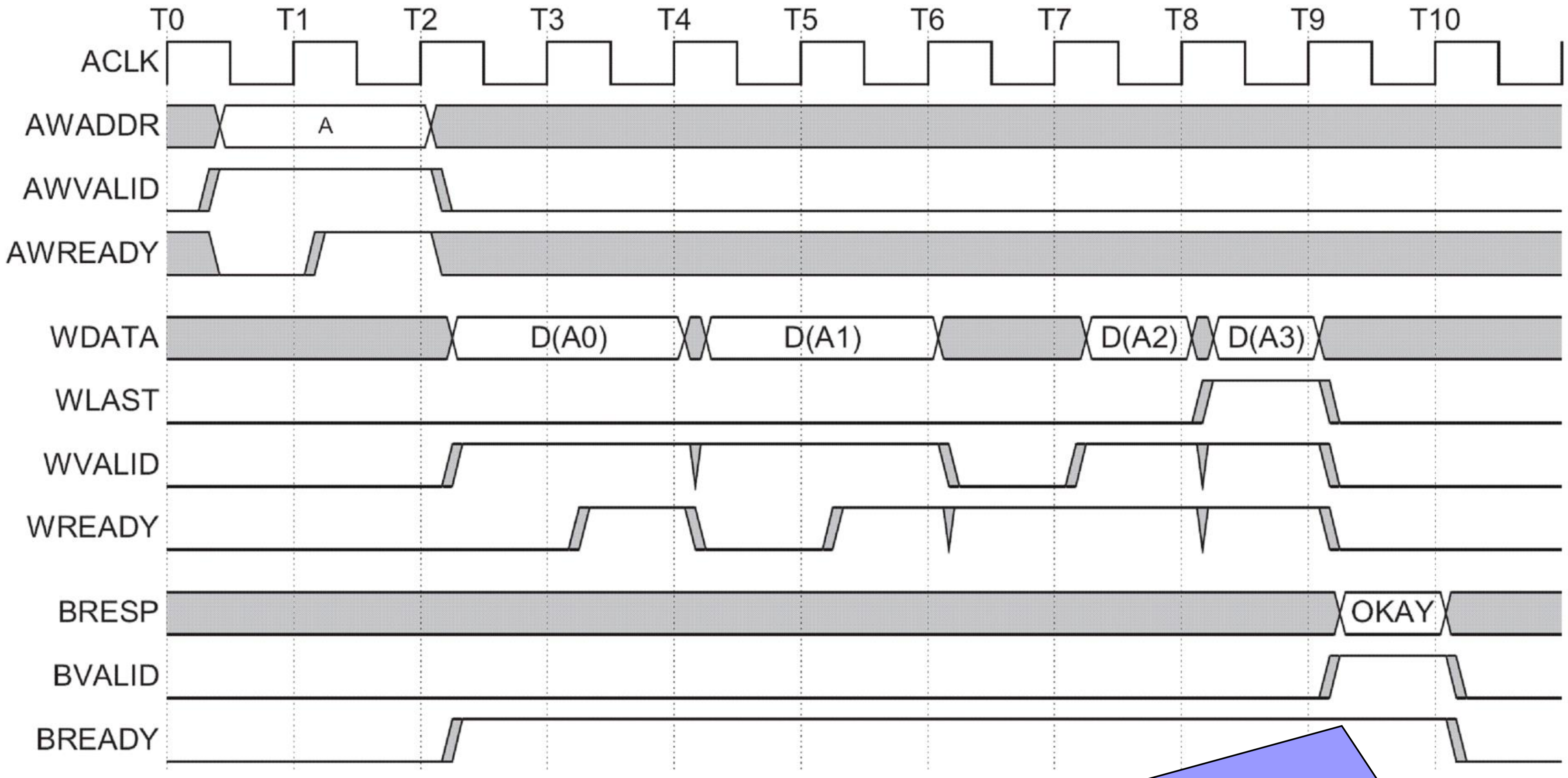
Example: Read Burst



Example: Overlapping Read Burst



Example: Write Burst



When the slave has accepted all the data items, it drives a write response back to the master to indicate that the write transaction is complete.

Transaction Ordering

- Enables out-of-order transaction completion
- Give an ID tag to every transaction
  - Transaction with the same ID ➔ in-order
  - Transaction with different ID ➔ can be completed out-of-order
- The ID tag is similar to a master number, but each master can implement multiple virtual masters by supplying different ID tags (virtual master number)
- Simple masters can issue every transaction with the same ID tag, and simple slaves can respond to every transaction in order, irrespective of the ID tag.

Signal Descriptions (1)

- Global signals

Signal	Source	Description
ACLK	Clock source	Global clock signal
ARESETn	Reset source	Global reset signal

Signal Descriptions (2)

- Write address channel signals (1)

Signal	Source	Description
AWID[3:0]	Master	Write address ID
AWADDR[31:0]	Master	Write address (the first address)
AWLEN[3:0]	Master	Burst length: 0(1)—15(16)
AWSIZE[2:0]	Master	Burst size: 2^{AWSIZE}
AWBURST[1:0]	Master	Burst type (00: FIXED, 01: INCR, 10: WRAP, 11: Reserved)
AWLOCK[1:0]	Master	Lock type (00: normal, 01: exclusive, 10: locked, 11: reserved)
AWCACHE[3:0]	Master	Cache type

Signal Descriptions (3)

- Write address channel signals (2)

Signal	Source	Description
AWPROT[2:0]	Master	Protection type: normal, privileged, or secure
AWVALID	Master	Write address valid. Remain stable until the AWREADY signal goes high
AWREADY	Slave	Write address ready

Signal Descriptions (4)

- Write data channel signals

Signal	Source	Description
WID[3:0]	Master	Write ID tag, must match to AWID
WDATA[31:0]	Master	Write data (up to 1024 bits)
WSTRB[3:0]	Master	Write strobes. Indicate which byte lanes to update in memory. 1 bit for each eight bits.
WLAST	Master	Write last
WVALID	Master	Write valid
WREADY	Slave	Write ready

Signal Descriptions (5)

- Write response channel signals

Signal	Source	Description
BID[3:0]	Slave	Response ID, must match to AWID
BRESP[1:0]	Slave	Write response. OKAY, EXOKAY, SLVERR, and DECERR
BVALID	Slave	Write response valid
BREADY	Master	Response ready

Signal Descriptions (6)

- Read address channel signals (1)

Signal	Source	Description
ARID[3:0]	Master	Read address ID
ARADDR[31:0]	Master	Read address (the first address)
ARLEN[3:0]	Master	Burst length: 0(1)—15(16)
ARSIZE[2:0]	Master	Burst size: 2^{AWSIZE}
ARBURST[1:0]	Master	Burst type (00: FIXED, 01: INCR, 10: WRAP, 11: Reserved)
ARLOCK[1:0]	Master	Lock type (00: normal, 01: exclusive, 10: locked, 11: reserved)
ARCACHE[3:0]	Master	Cache type

Signal Descriptions (7)

- Read address channel signals (2)

Signal	Source	Description
ARPROT[2:0]	Master	Protection type: normal, privileged, or secure
ARVALID	Master	Read address valid. Remain stable until the ARREADY signal goes high
ARREADY	Slave	Read address ready

Signal Descriptions (8)

- Read data channel signals

Signal	Source	Description
RID[3:0]	Slave	Read ID tag, must match to ARID
RDATA[31:0]	Slave	Write data (up to 1024 bits)
RRESP[1:0]	Slave	Read response. OKAY, EXOKAY, SLVERR, and DECERR
RLAST	Slave	Read last
RVALID	Slave	Read valid
RREADY	Master	Read ready

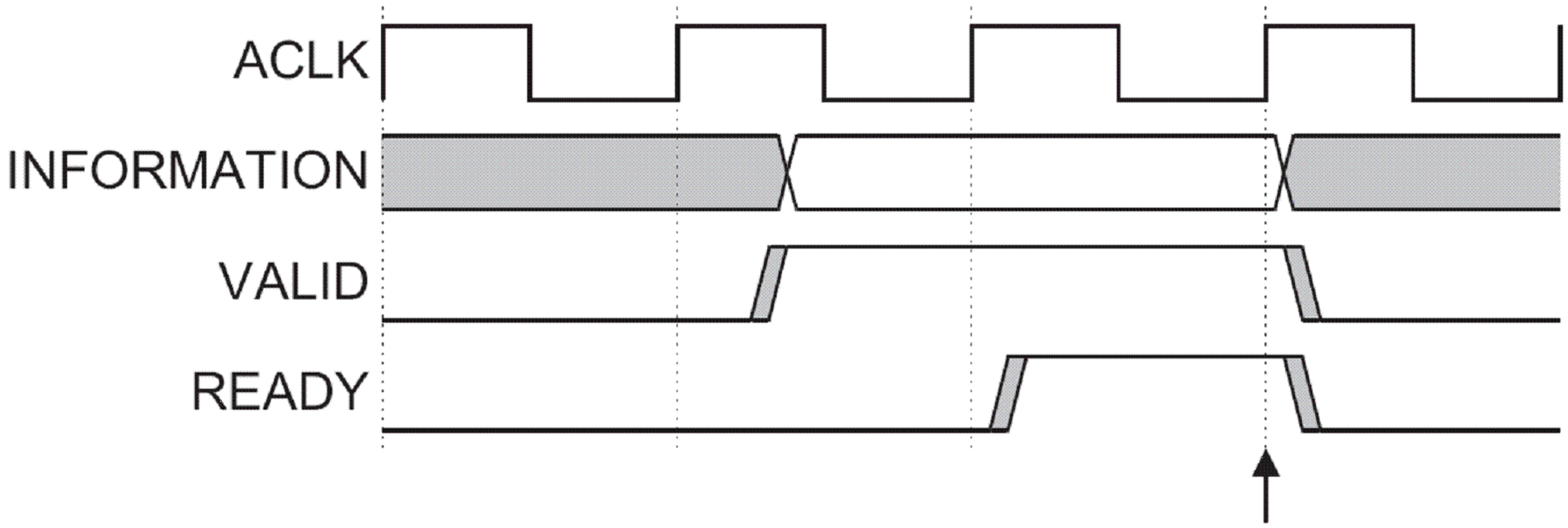
Signal Descriptions (9)

- Low-power interface signals

Signal	Source	Description
CSYSREQ	Clock controller	System low-power request
CSYSACK	Peripheral device	Low-power request acknowledgement
CACTIVE	Peripheral device	Clock active: indicates that the peripheral requires its clock signal

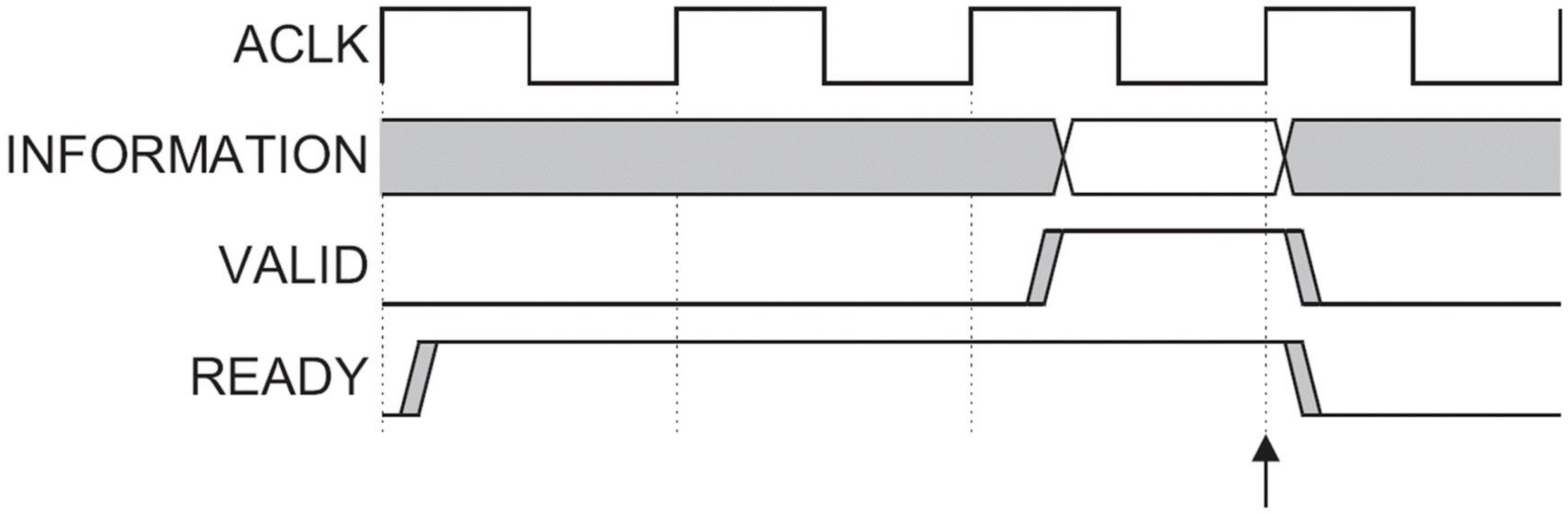
Channel Handshake (1)

- Normal case



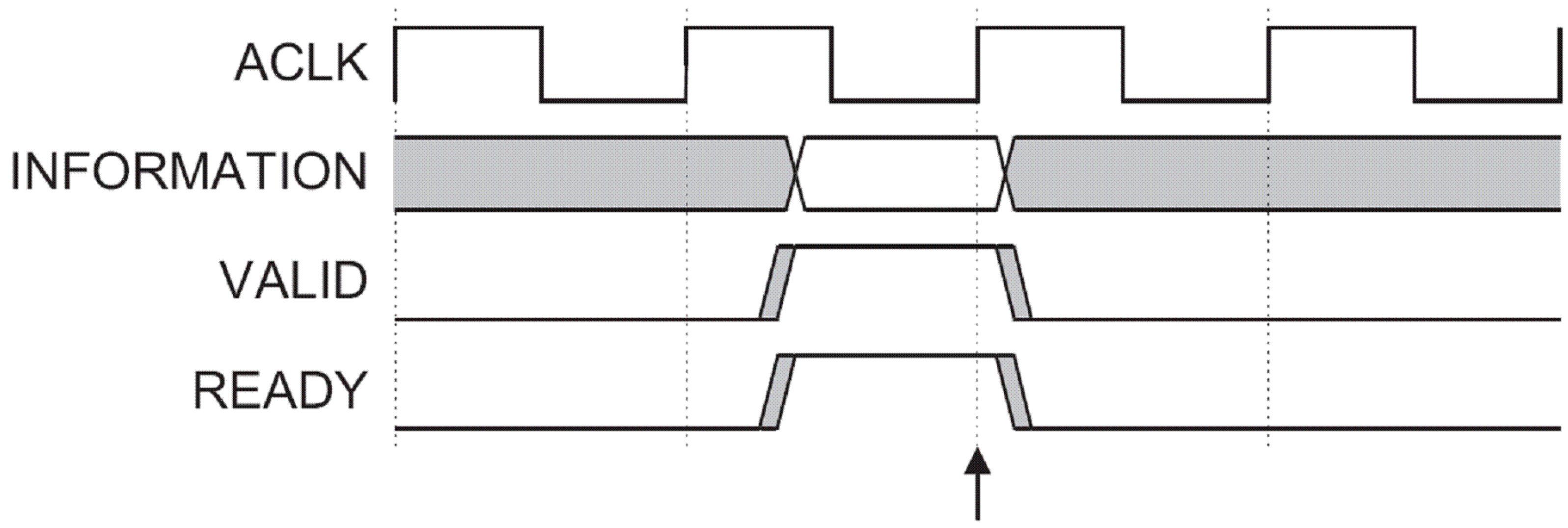
Channel Handshake (2)

- The destination can accept the data or control information in a single cycle as soon as it becomes valid ➔ high efficiency

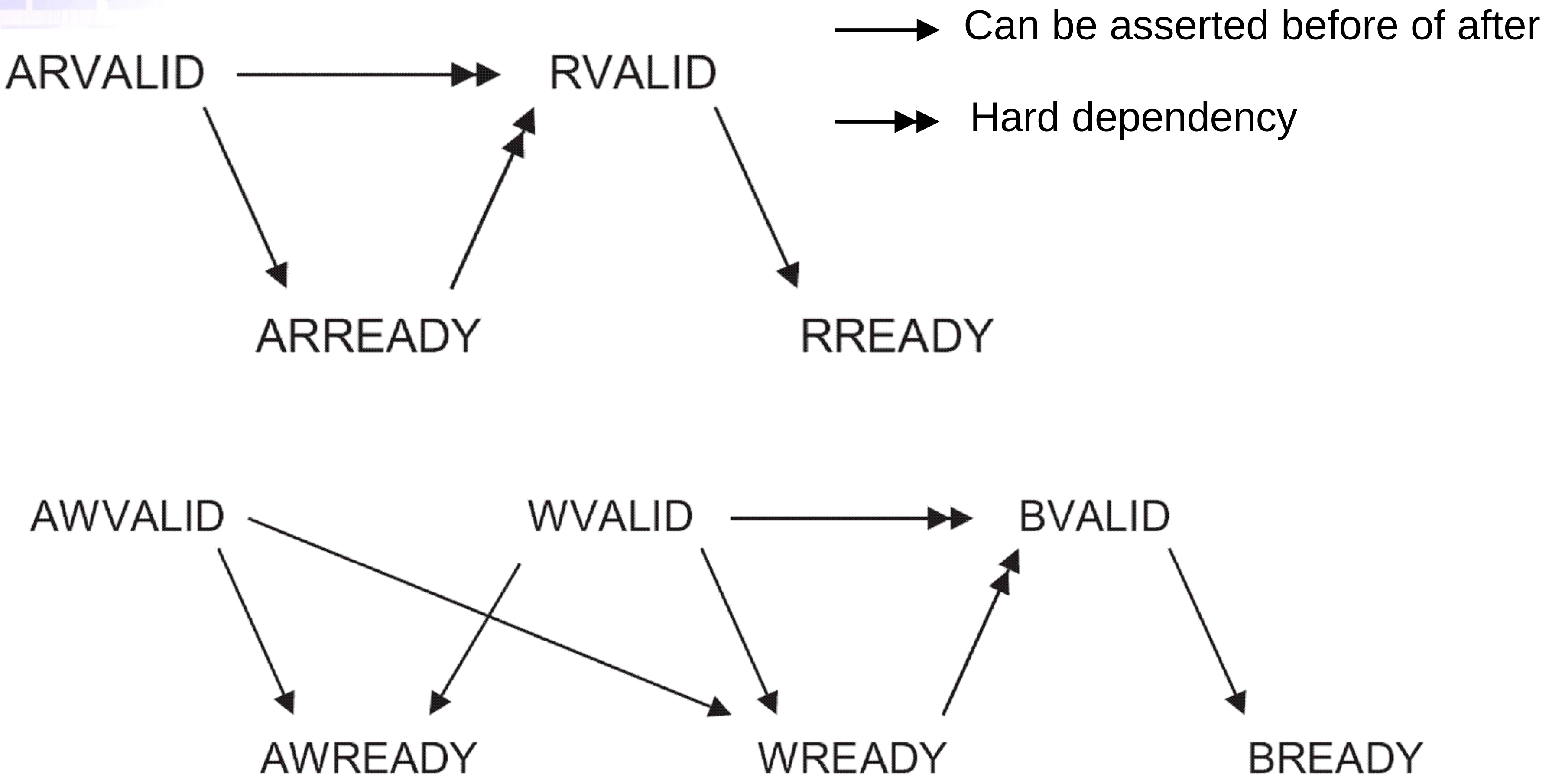


Channel Handshake (3)

- The transfer occurs immediately



Dependencies between Channel Handshake Signals


Can be asserted before of after
Hard dependency
ARVALID
RVALID
ARREADY
RREADY
AWVALID
WVALID
BVALID
AWREADY
WREADY
BREADY