CprE 488 – Embedded Systems Design

Lecture 2 – Embedded Platforms

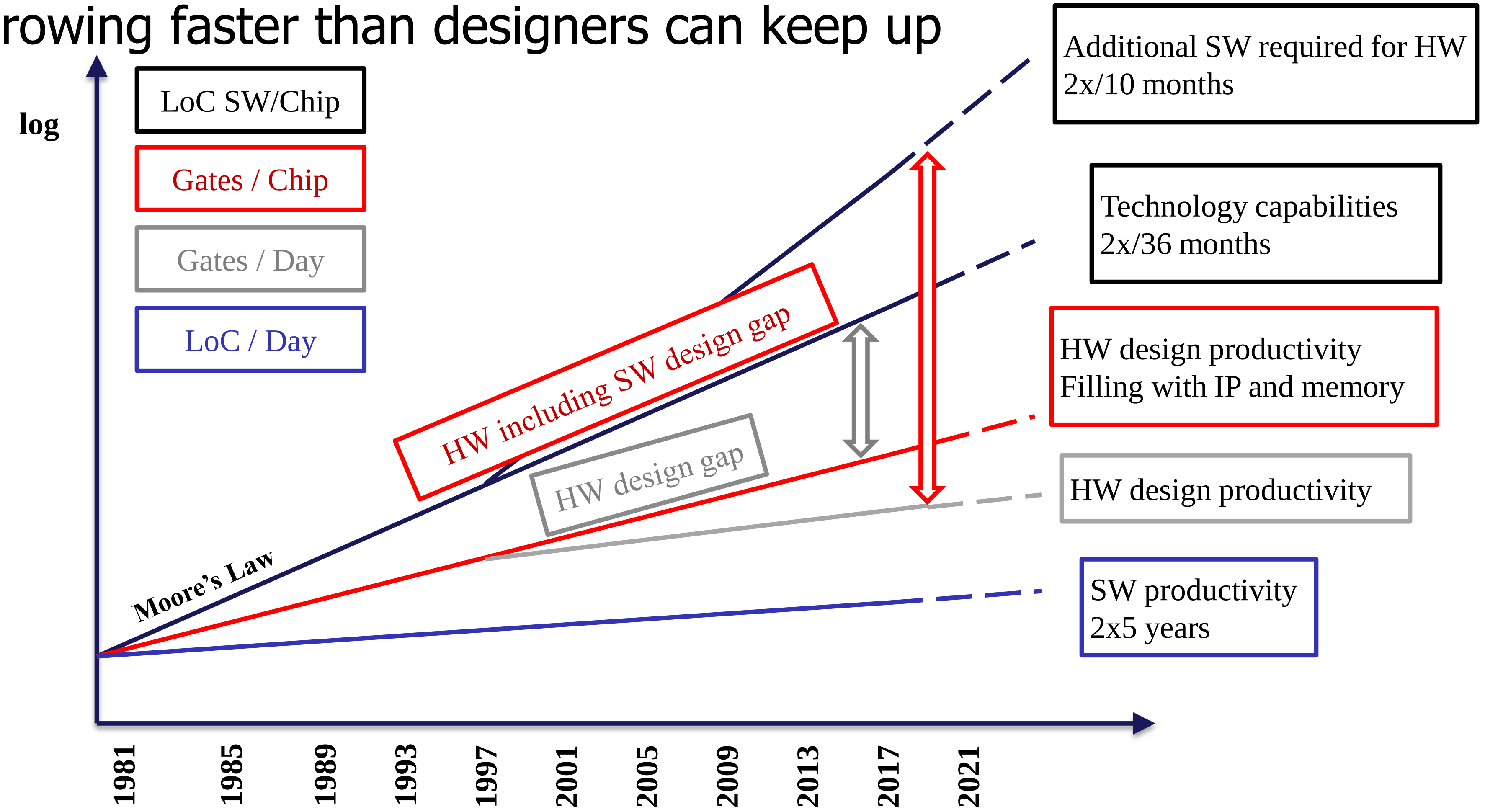
Joseph Zambreno
Electrical and Computer Engineering
Iowa State University

www.ece.iastate.edu/~zambreno
rcl.ece.iastate.edu

Don't reinvent the wheel, unless you plan on learning more about wheels. – Jeff Atwood

The Growing Designer Productivity Gap

- Embedded systems today are characterized by rapidly expanding functionality coupled with shrinking time to market
- Hardware capabilities (and accompanying software needs) are growing faster than designers can keep up

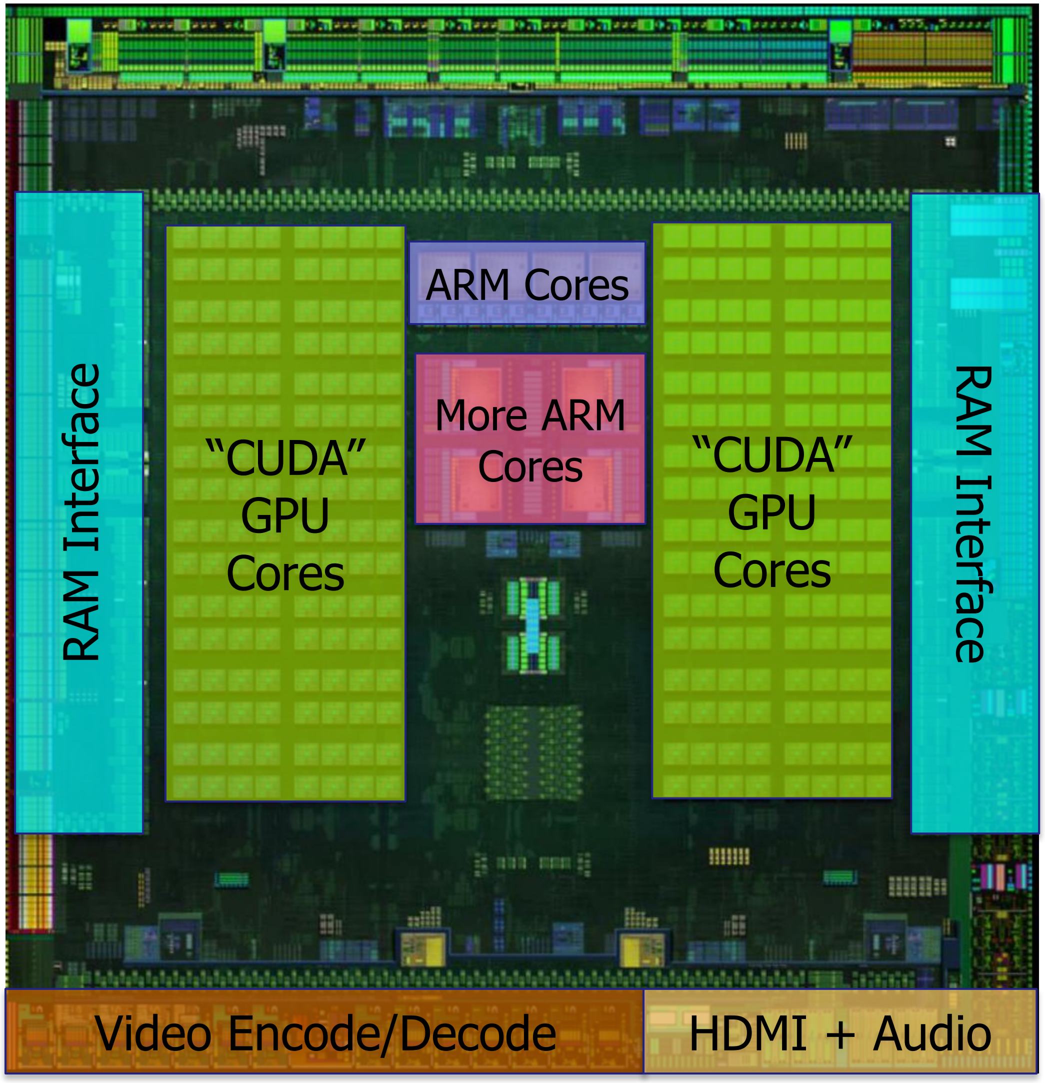


Coping with Complexity

- Intellectual Property (**IP**) reuse:
  - Pre-designed hardware modules – either soft cores (e.g. source, netlists) or hard cores (transistor or other layout)
  - Range in functionality from simple multipliers, to memories, to interfacing logic, to processors
  - Pre-designed software as well (e.g. drivers, OS)
  - Typically follow some set interfacing standards
- **Platform**-based design:
  - Design philosophy that focuses on IP-centric design and reuse
  - Platform: a customizable design for a particular type of system, consisting of embedded processors, peripherals, and software

An Example Platform

- Tegra X1 – NVIDIA's latest mobile "processor"
- Main processing, I/O, and memory hardware IP cores integrated onto a single chip:
  - System-on-Chip (SoC)
  - Still part of a larger embedded platform (e.g. a tablet)
- Significant portion of the chip is dedicated to specialized hardware
- Previous Tegra iterations were not especially successful – limited software ecosystem



SoC Requirements

- Typically a few CPU cores with other specialized functions, connected via an on-chip *network*
- Optimized for a particular application domain (e.g. mobile phone, tablet, desktop)
- SoCs make sense only in high-volume, multi-generational applications, where they are mandatory for low cost

- To use SoCs effectively, a manufacturer must have:
  - A reliable source of IP (CPU/cache, on-chip networks, device and memory controllers)
  - Design tools to aggregate and interconnect the IP
  - Product designers to specify the functions and employ the tools
- What isn't needed?
  - A silicon fabrication facility
  - A final product assembly plant

This Week's Topic

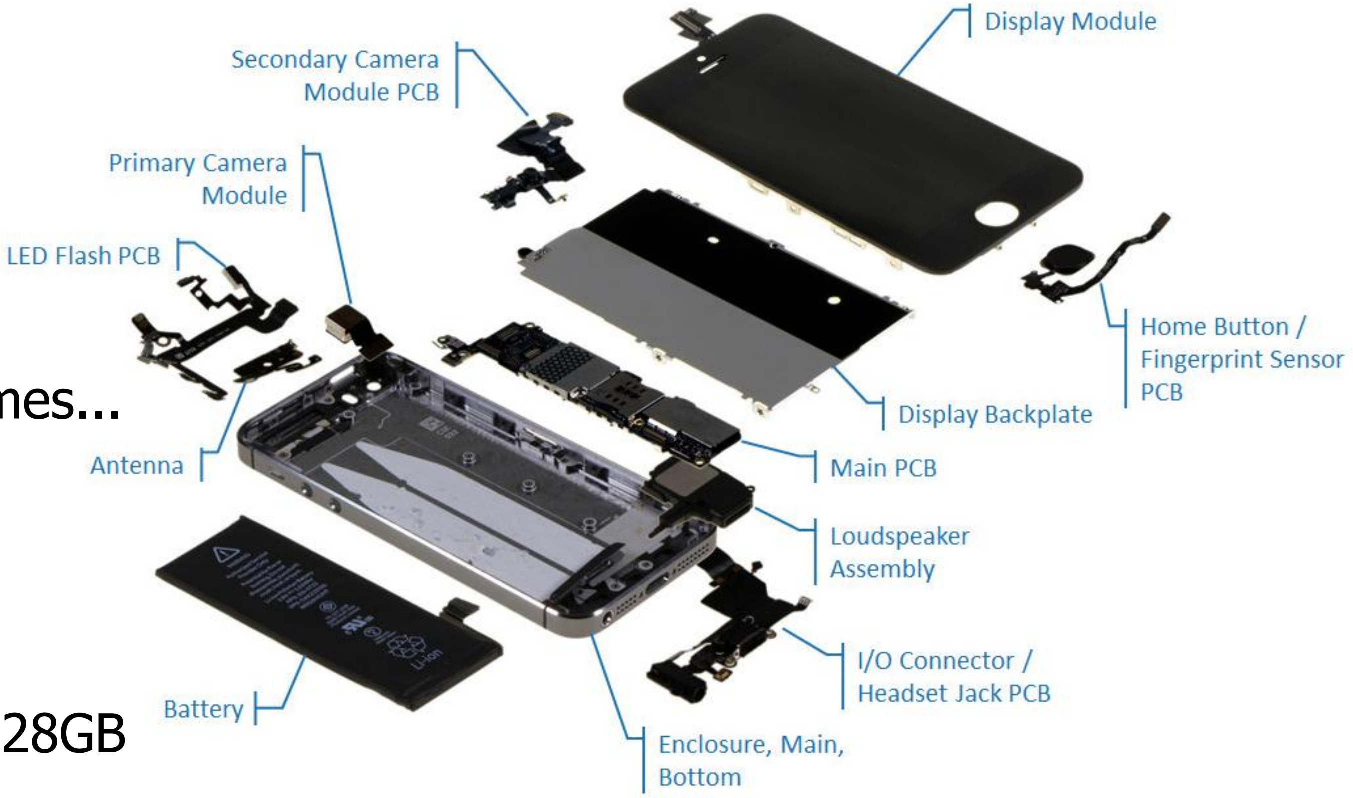
- Platform-based design
  - Platform case study: Apple iPhone 6 plus
  - Platform case study: Digilent ZedBoard
- Bus-based communication architectures
  - AMBA
- Direct Memory Access

- Reading: Wolf chapter 4

Case Study: The Apple iPhone 6 Plus

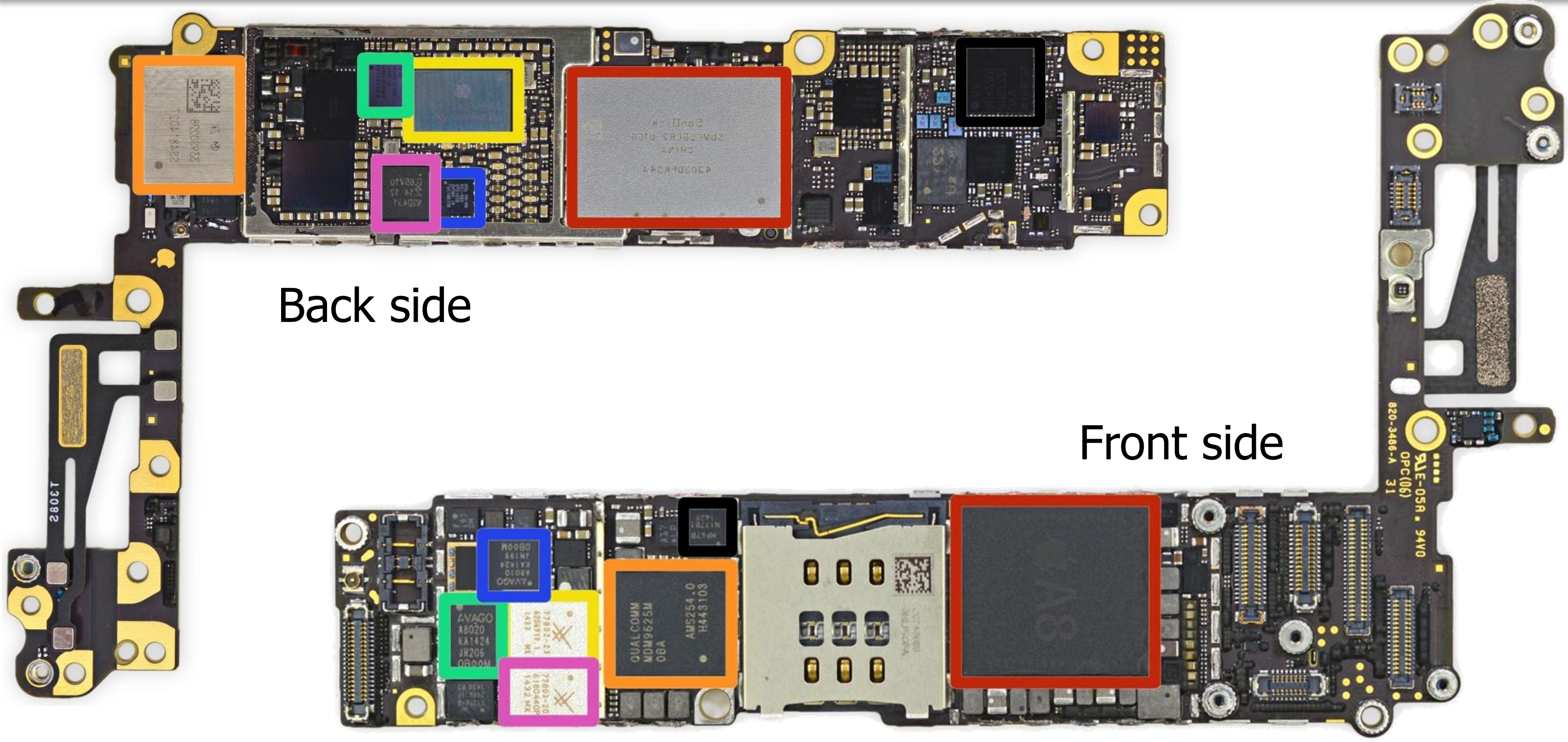
- In competitive markets, low-level platform details are typically not provided by manufacturers
- Fortunately, there exist companies that specialize in tearing apart platforms (e.g. IHS) and reverse-engineering SoCs (e.g. Chipworks)

becomes…



Apple iPhone 6 Plus 128GB

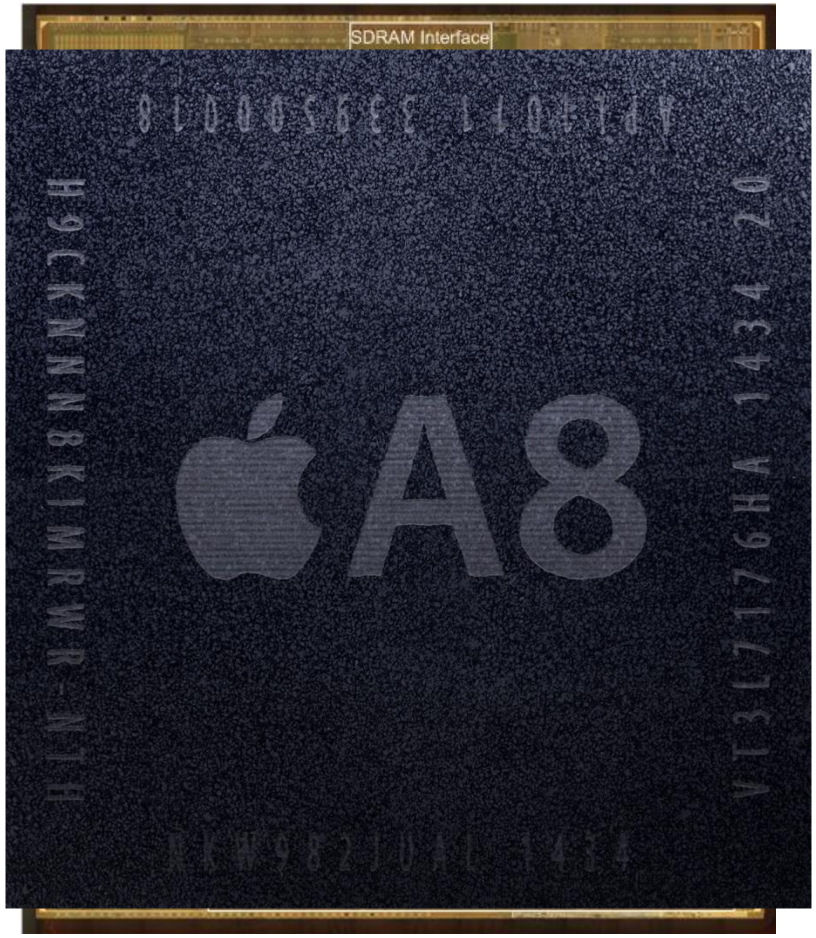
iPhone 6 Plus – Main PCB Components



- Main processor
- DRAM memory
- Sensor coprocessor
- Accelerometers, gyros, and compass
- WiFi and Bluetooth
- 4G LTE Modem
- RF and power amplifiers
- Audio codec

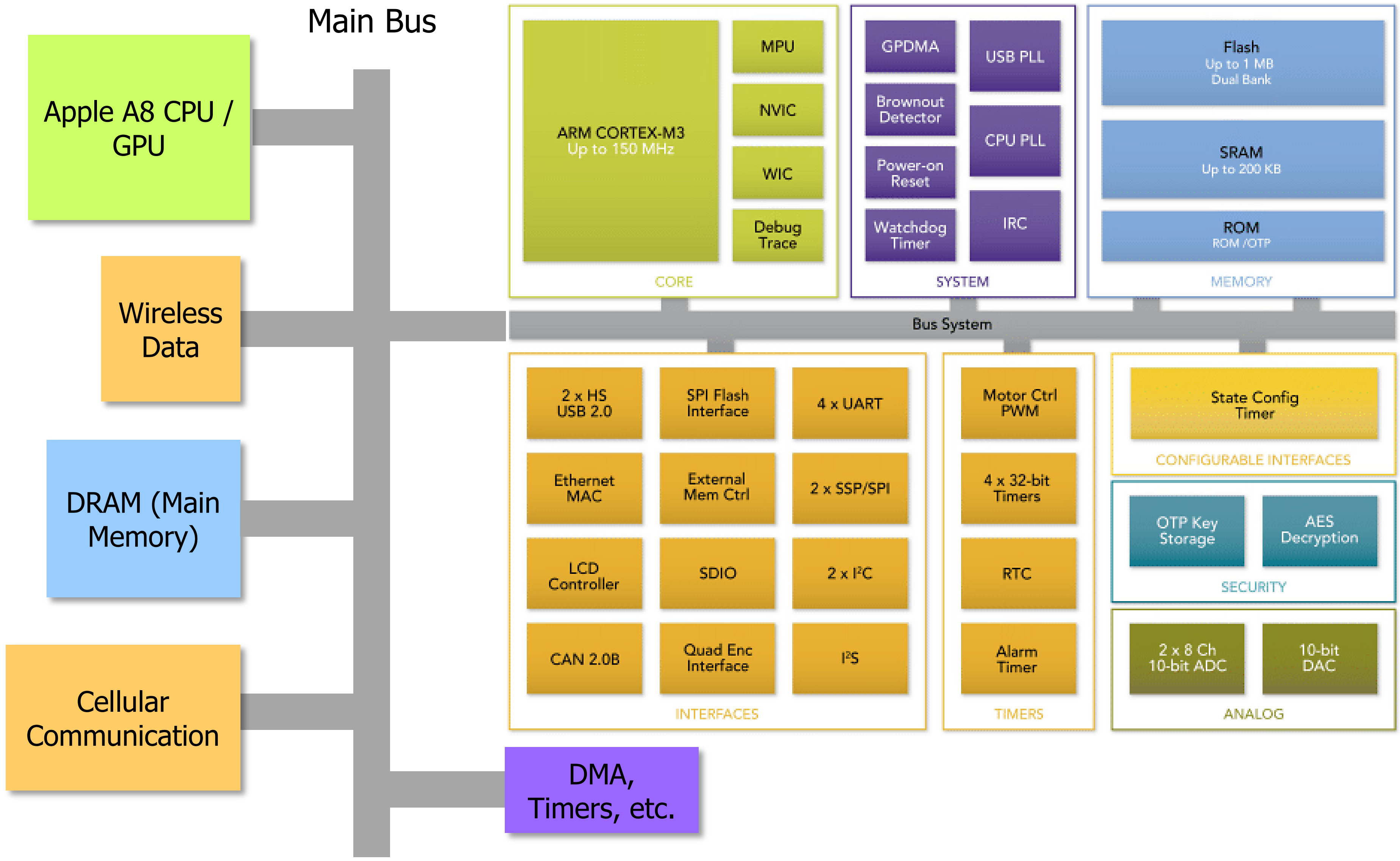
iPhone 6 – Apple A8 SoC

- Apple has a "full" IP license from ARM, meaning they can create any ARM ISA-compatible CPU core
- Dual-ARM cores (some ARMv8 variant with 64-bit instructions)
  - Clock speed – 1.38 GHz
  - 64 KB L1 instruction and data caches, 1 MB L2 cache, 4 MB L3 cache (shared between CPU and GPU)
- PowerVR Series 6XT GPU – integrated as IP core from Imagination Technologies
- Various IP for DRAM interfacing, other peripherals



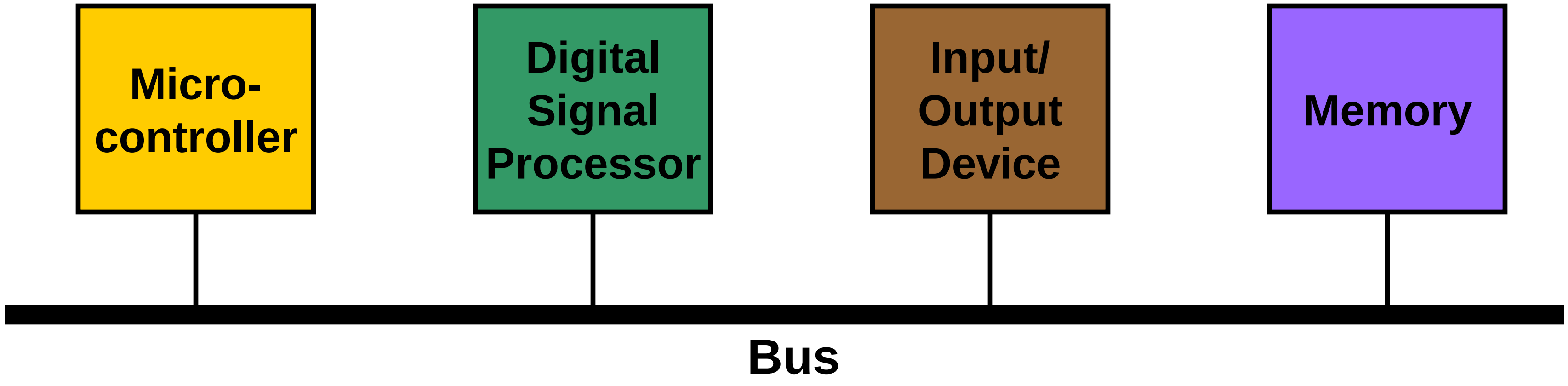
Apple A8 Processor Floorplan (image courtesy Chipworks)

iPhone 6 – Platform View

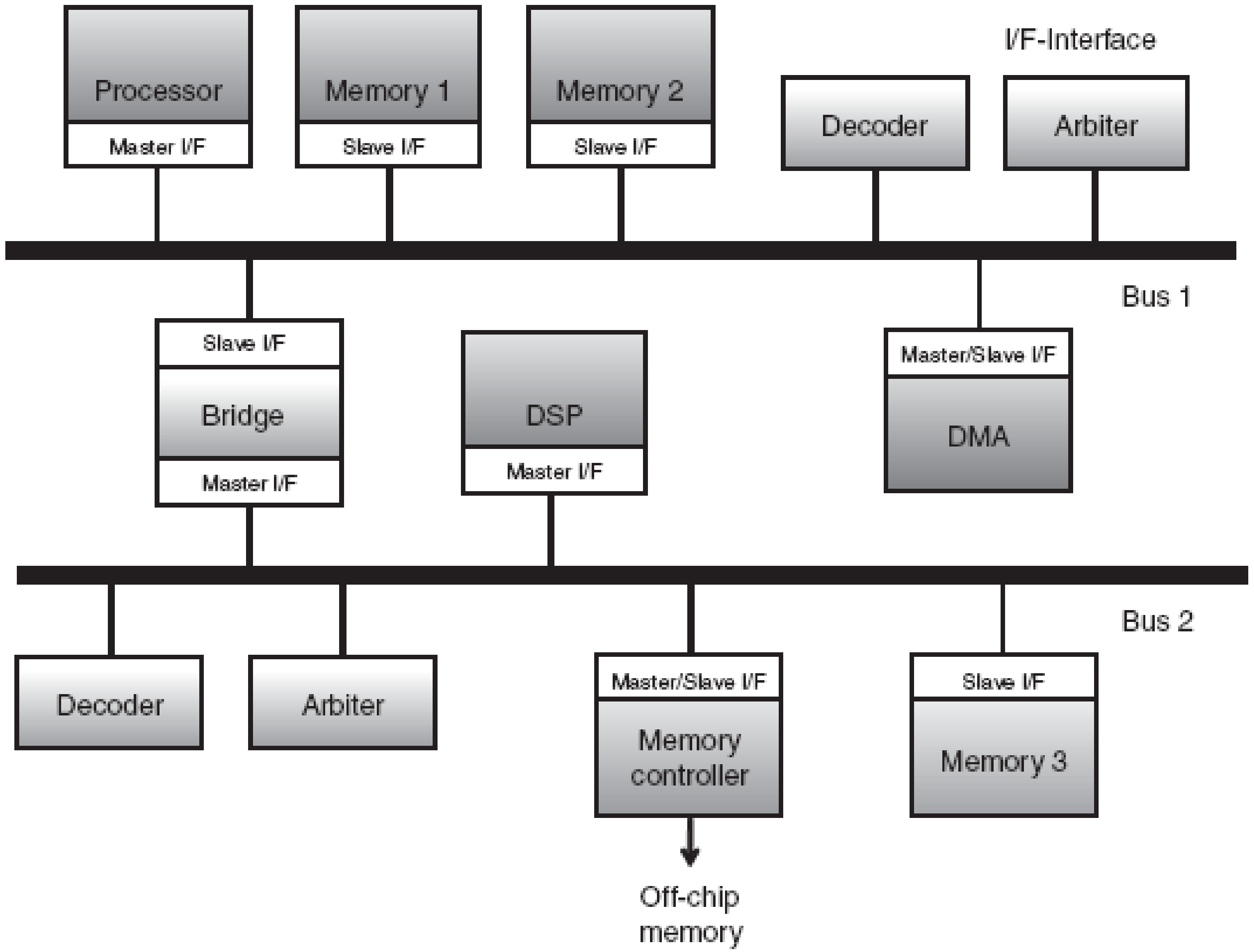


On-Chip Communication Architectures

- Buses are the simplest and most widely used SoC interconnection networks
- Bus:
  - A collection of signals (wires) to which one or more communicating IP components are connected
  - A protocol associated with that communication
- Only one IP component can transfer data on the shared bus at any given time



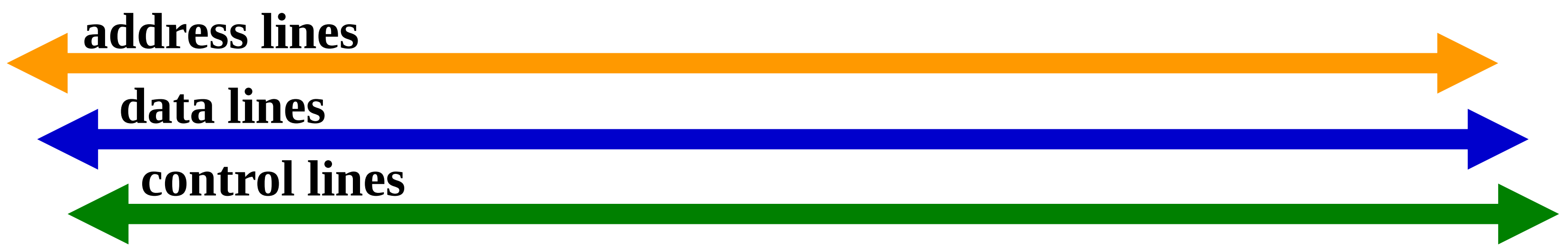
Bus Terminology



Bus Terminology (cont.)

- Master (or Initiator)
  - IP component that initiates a read or write data transfer
- Slave (or Target)
  - IP component that does not initiate transfers and only responds to incoming transfer requests
- Arbiter
  - Controls access to the shared bus
  - Uses an arbitration scheme to select master to grant access to bus
- Decoder
  - Determines which component a transfer is intended for
- Bridge
  - Connects two buses
  - Acts as slave on one side and master on the other

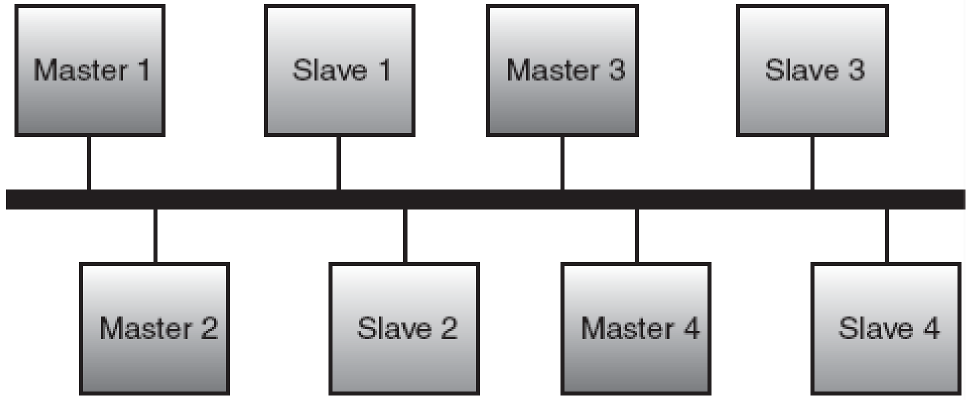
Bus Signal Lines

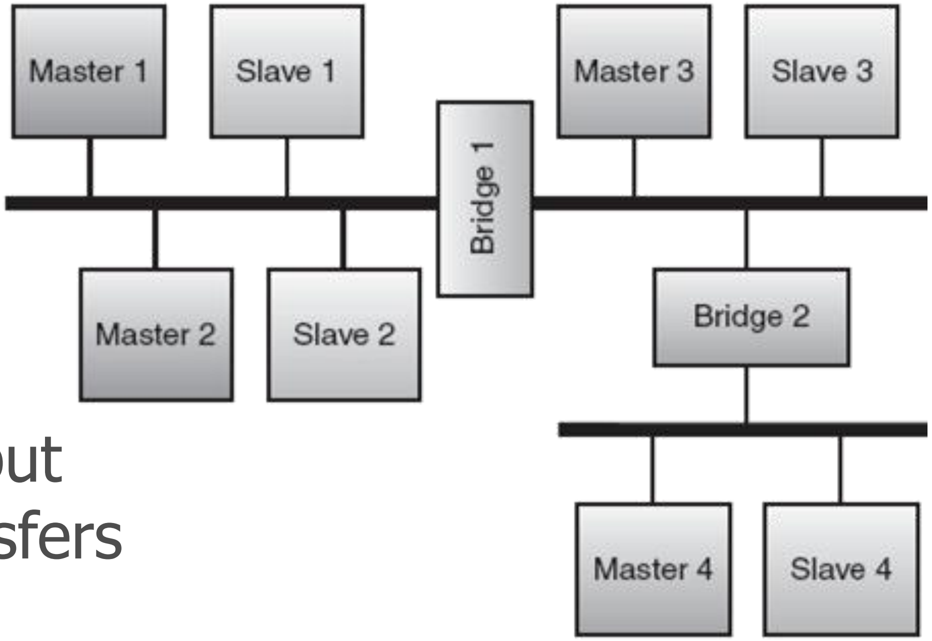


- A bus typically consists of three types of signal lines:
  - Address
    - Carry address of destination for which transfer is initiated
    - Can be shared or separate for read, write data
  - Data
    - Carry information between source and destination components
    - Can be shared or separate for read, write data
    - Choice of data width critical for application performance
  - Control
    - Requests and acknowledgements
    - Specify more information about type of data transfer
    - Ex: byte enable, burst size, cacheable/bufferable, write-back/through, …

Types of Bus Topologies

- Shared bus

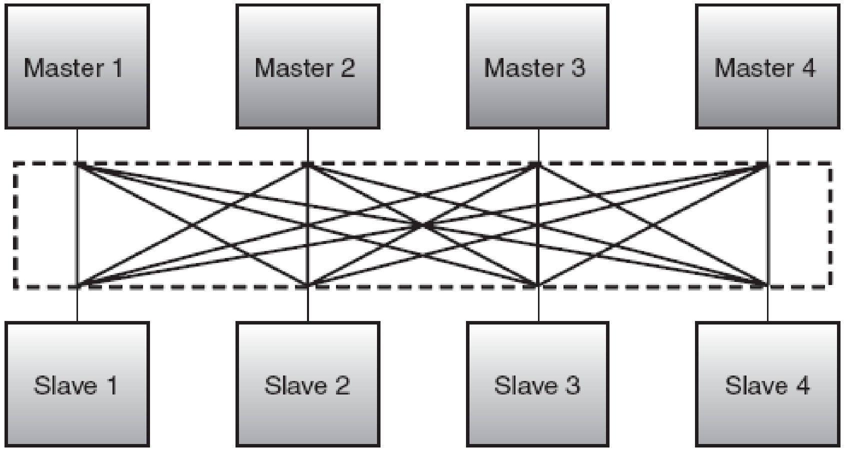




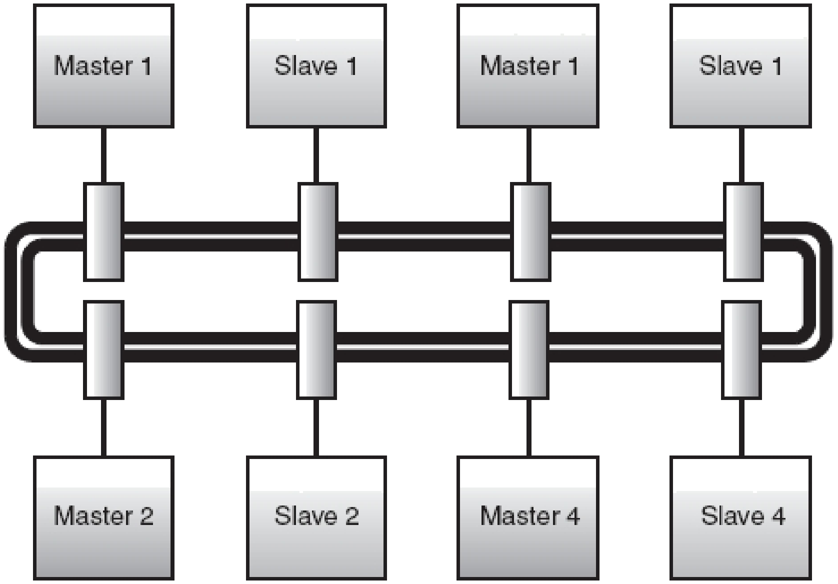
- Hierarchical shared bus
  - Improves system throughput
  - Multiple simultaneous transfers

Types of Bus Topologies (cont.)

- Full crossbar bus



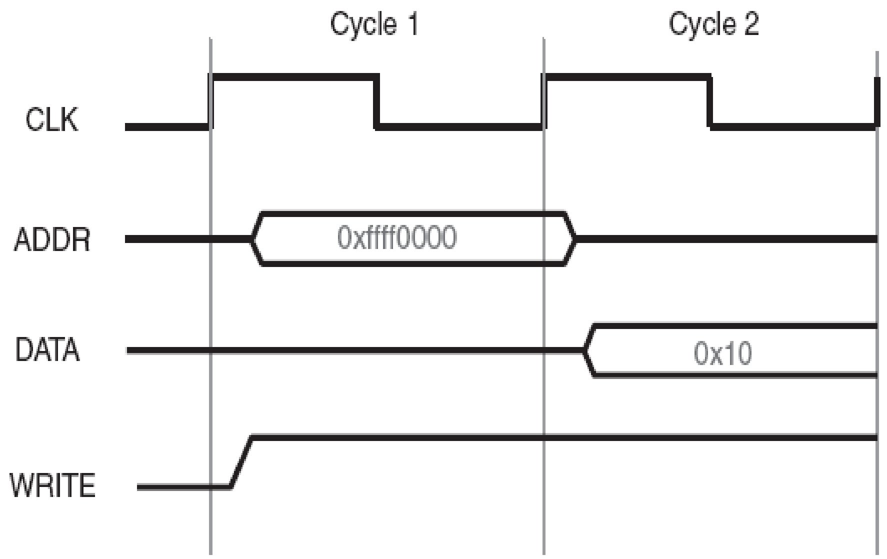
- Ring bus

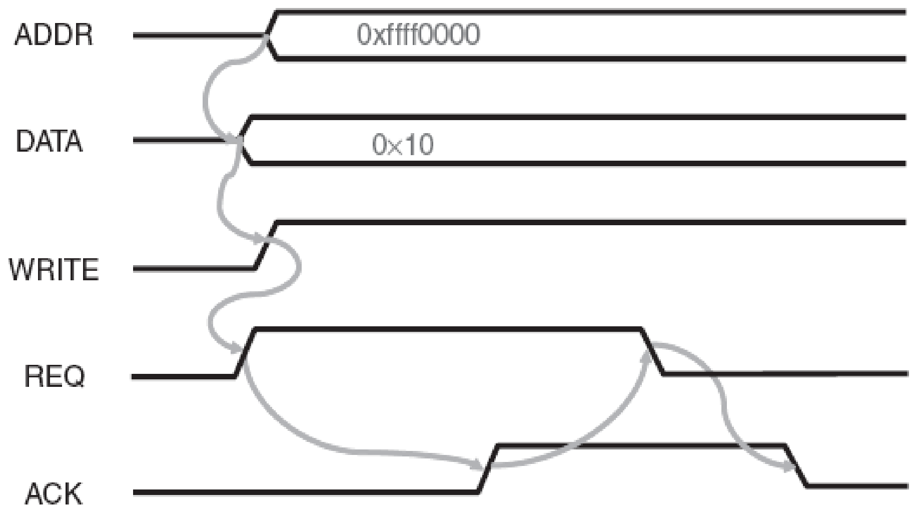


- Advantages? Disadvantages?

Bus Clocking

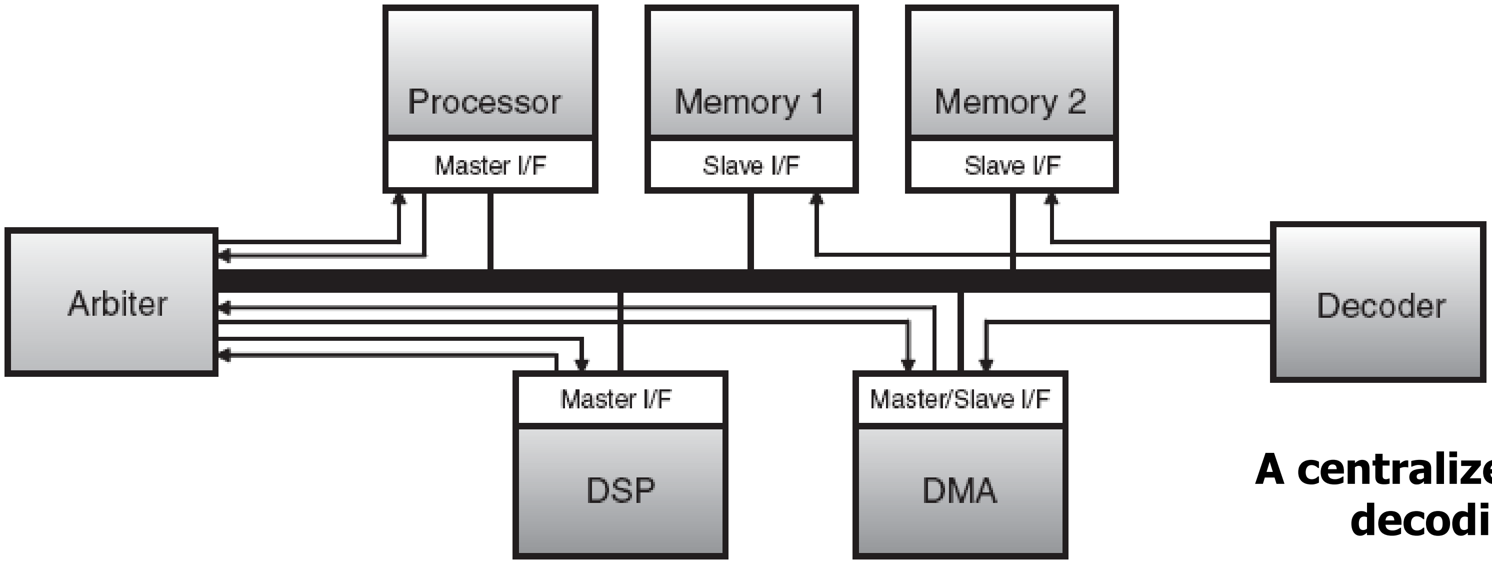
- Synchronous Bus
  - Includes a clock in control lines
  - Fixed protocol for communication that is relative to clock
  - Involves very little logic and can run very fast
  - Requires frequency converters across frequency domains
- Asynchronous Bus
  - Not clocked
  - Requires a handshaking protocol
    - performance not as good as that of synchronous bus
    - No need for frequency converters, but does need extra lines
  - Does not suffer from clock skew like the synchronous bus





Decoding and Arbitration

- A bus implementation includes logic for both decoding and arbitration (either distributed or centralized):
  - Decoding – determining the target for any transfer initiated by the master
  - Arbitration – deciding which master can use the shared bus if more than one master requested bus access simultaneously



A centralized arbitration / decoding scheme

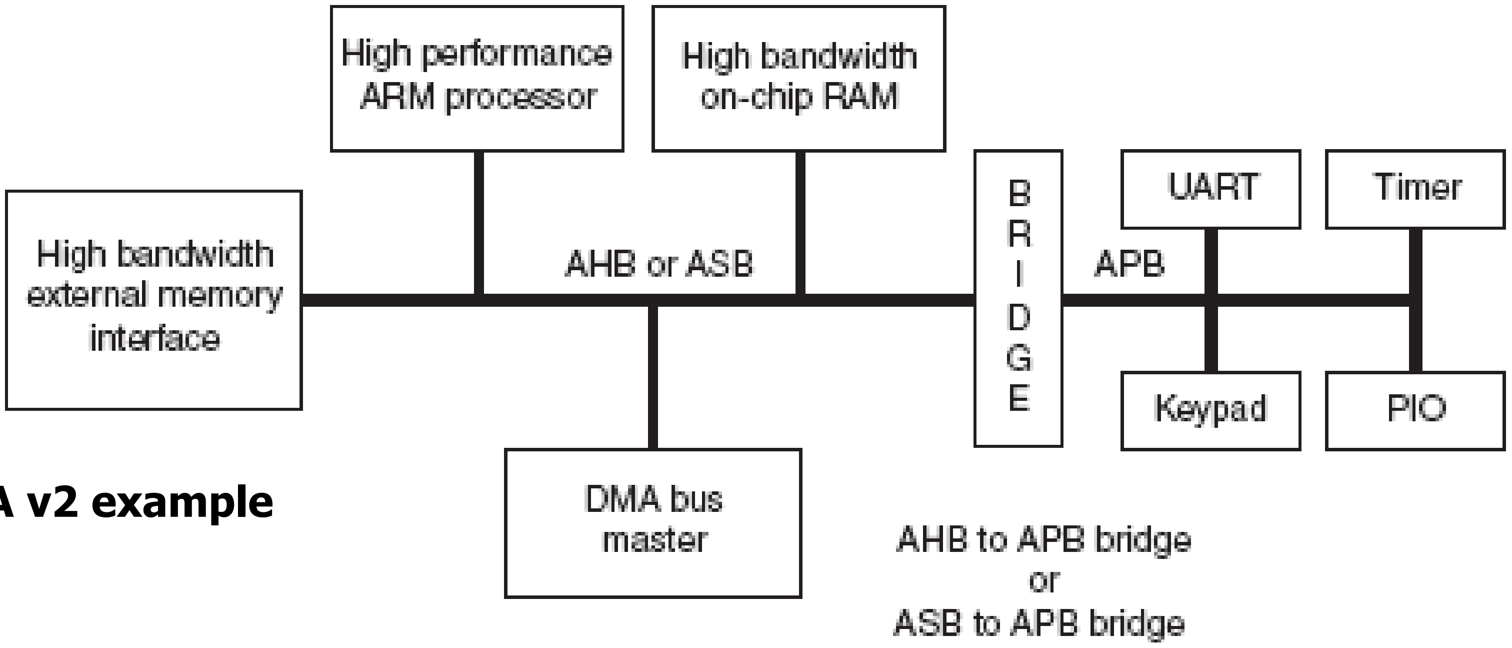
Bus Standards

- Bus standards are useful for defining a specific interface and data transfer protocol

- Ideally, all IP in a design are compatible
- In reality, several competing standards for SoC design:
  - CoreConnect (IBM)
  - AMBA (ARM)
  - Wishbone (OpenCores)
  - Avalon (Altera)
  - Quick Path (Intel)
  - HyperTransport (AMD)
  - STBus (STMicroelectronics)

AMBA

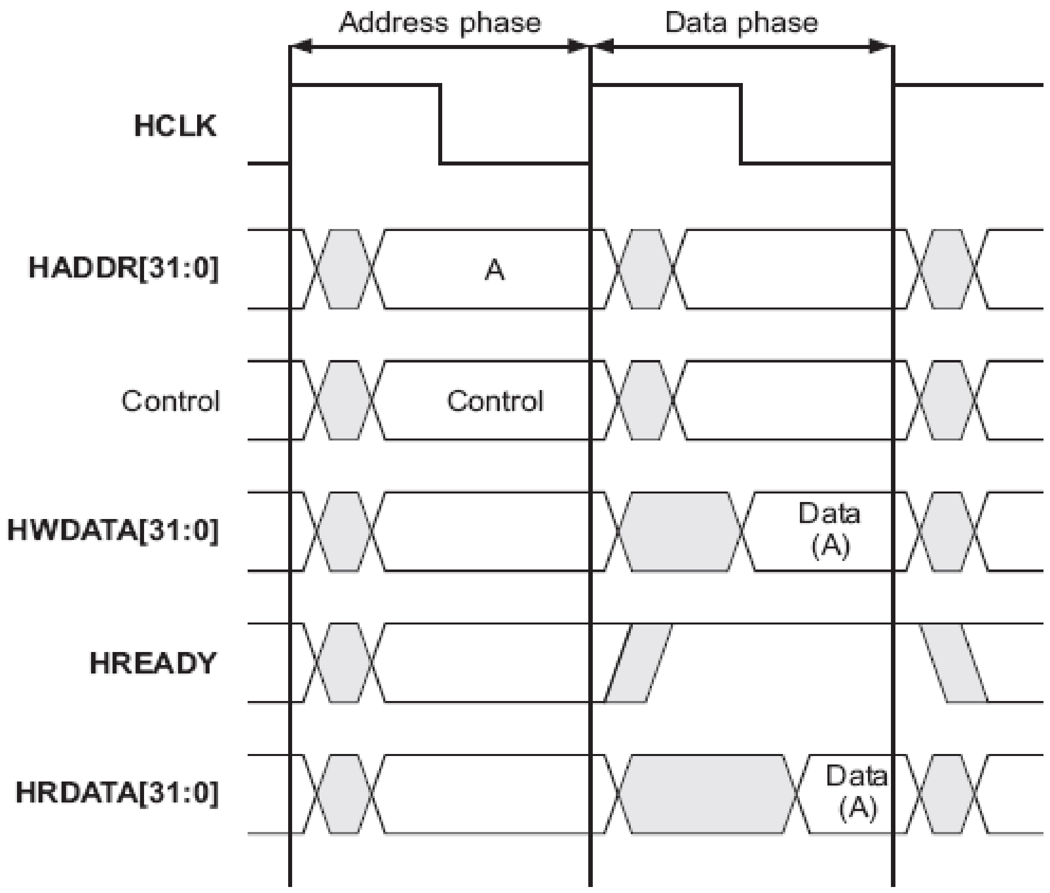
- Advanced Microcontroller Bus Architecture (AMBA)
- Multiple versions (typically dependent on choice of processor IP)
  - v1 – Advanced System Bus (ASB), Advanced Peripheral Bus (APB)
  - v2 – ASB, APB, Advanced High-performance Bus (AHB)
  - v3 – Advanced eXtensible Interface (AXI), APB, Advanced Trace Bus (ATB)
  - v4 – AXI Coherency Extensions (ACE), AXI, AXI-Stream, APB, ATB
  - v5 – Coherent Hub Interface (CHI), ACE


High performance ARM processor
High bandwidth on-chip RAM
High bandwidth external memory interface
AHB or ASB
BRIDGE
UART
Timer
APB
Keypad
PIO
DMA bus master
AHB to APB bridge or ASB to APB bridge

AMBA v2 example

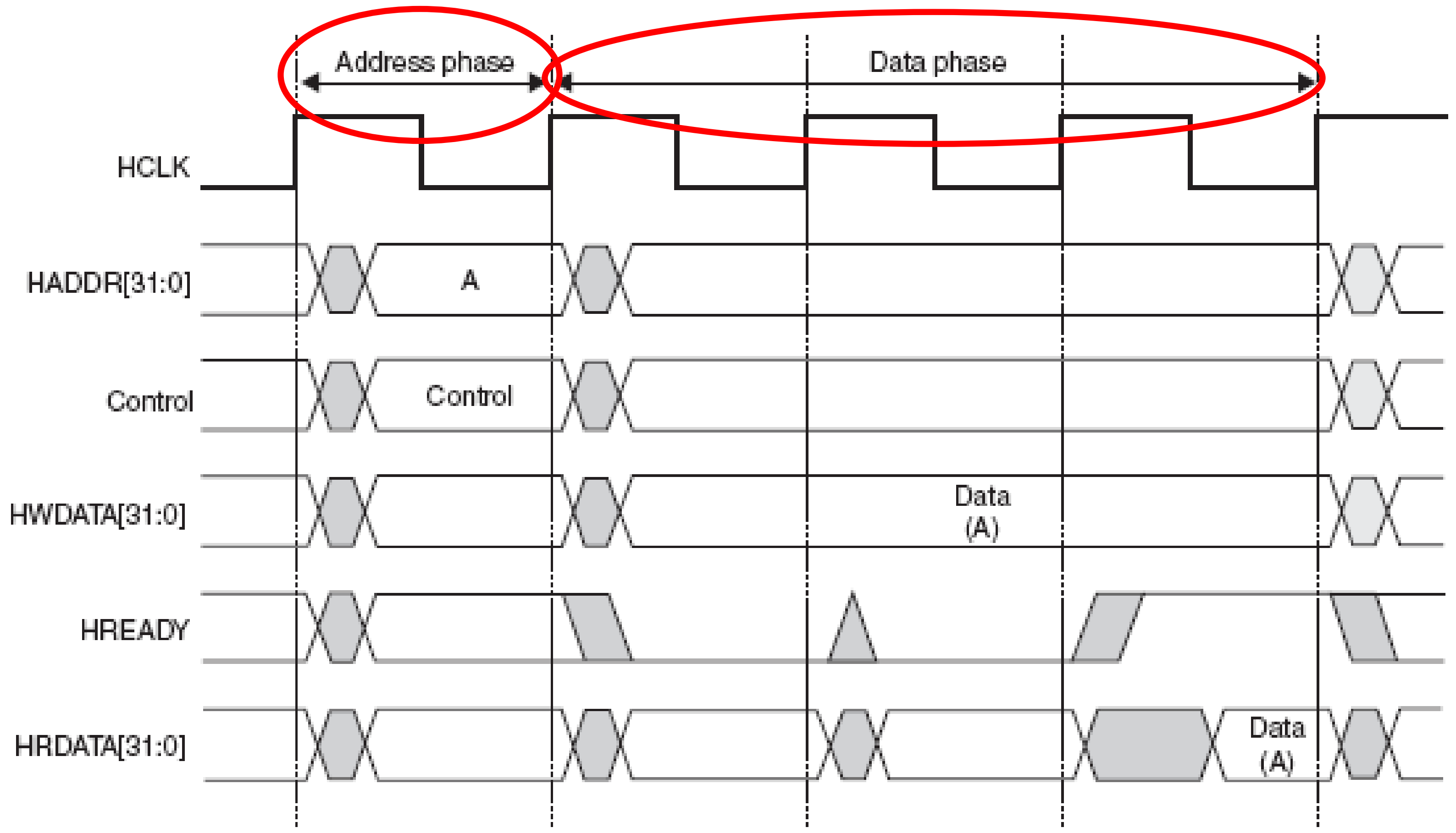
AMBA AHB Operation

- Split ownership of Address and Data bus



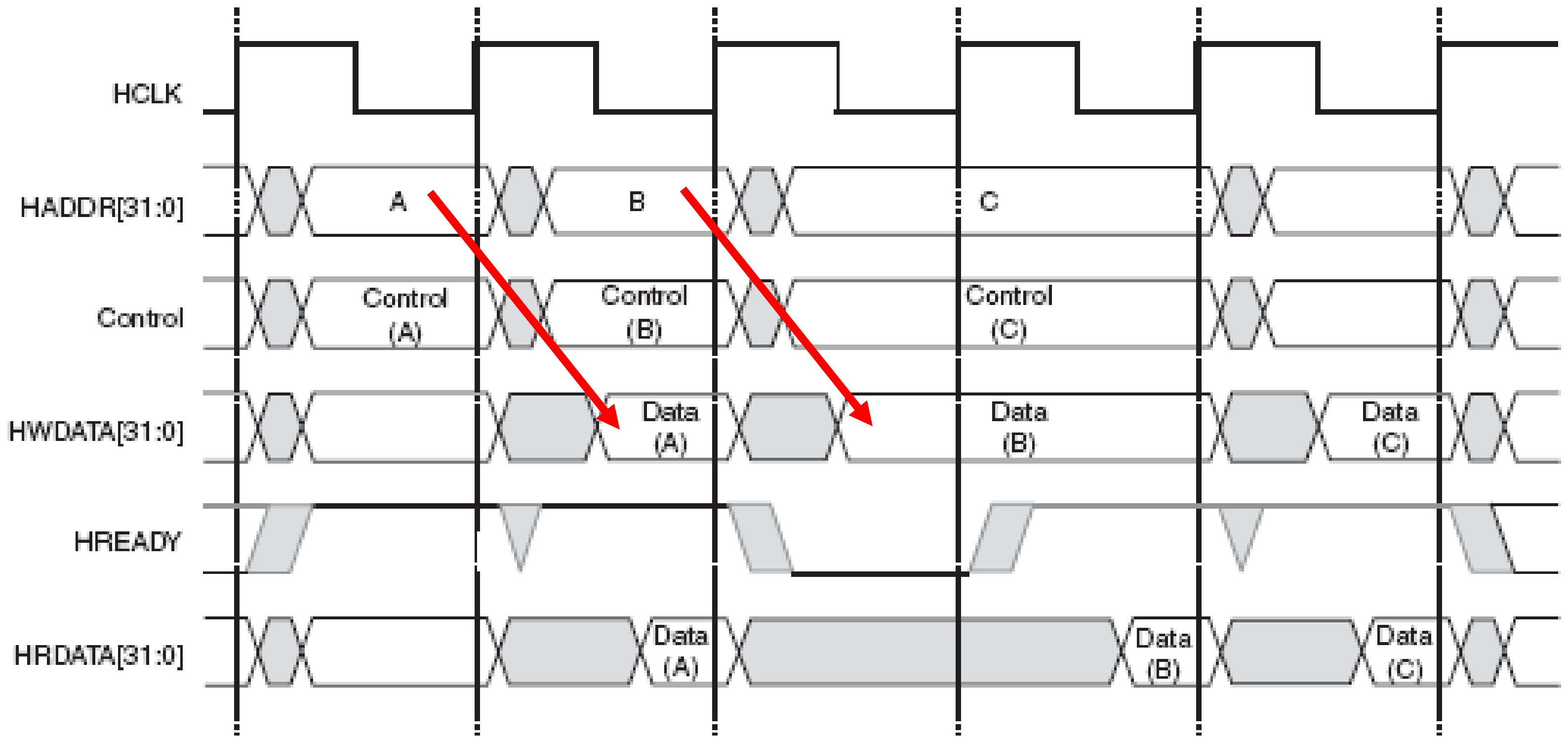
AMBA AHB Operation (cont.)

- Data transfer with wait states

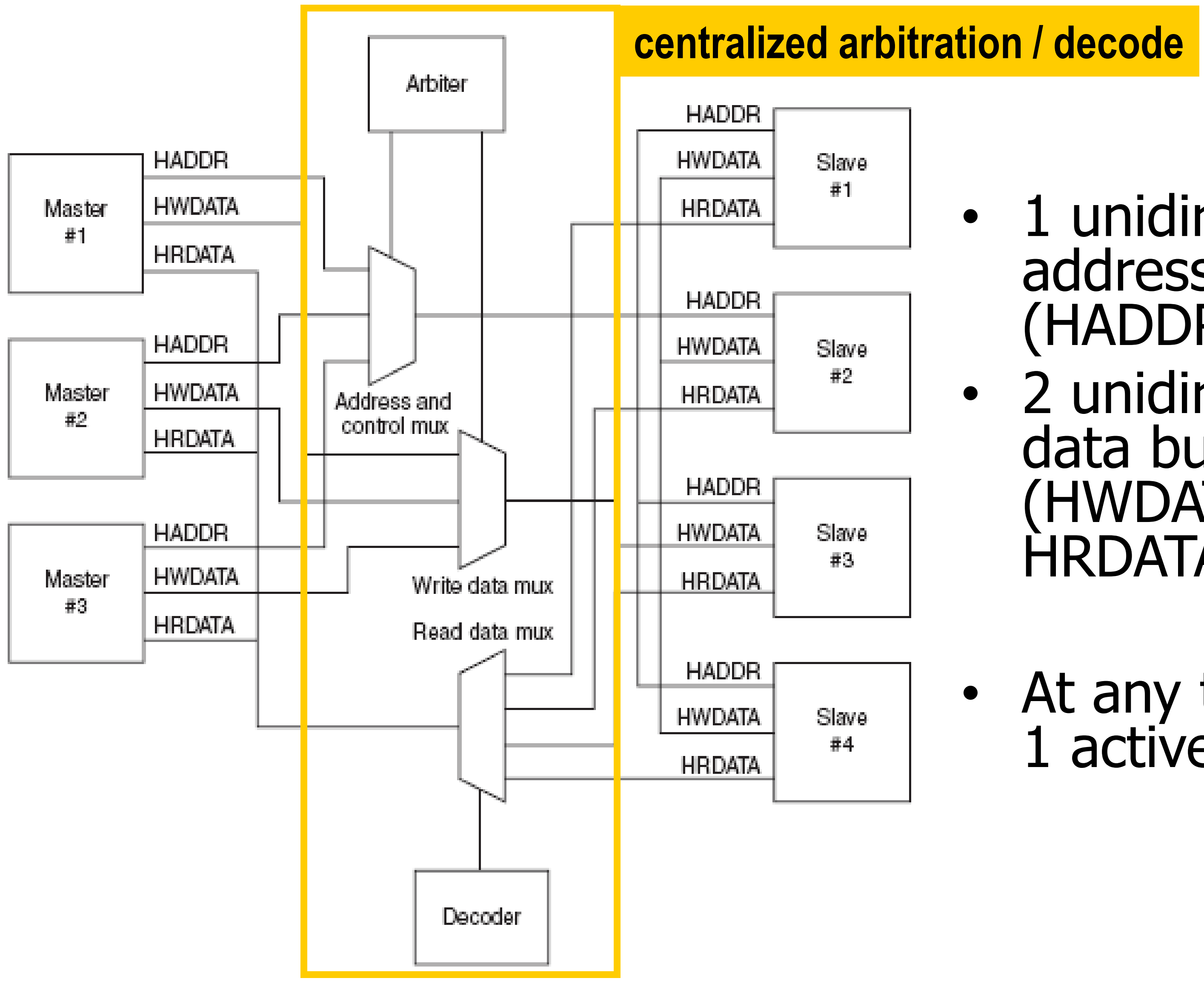


AHB Pipelining

- Transaction pipelining increases bus bandwidth



High-Level AHB Architecture

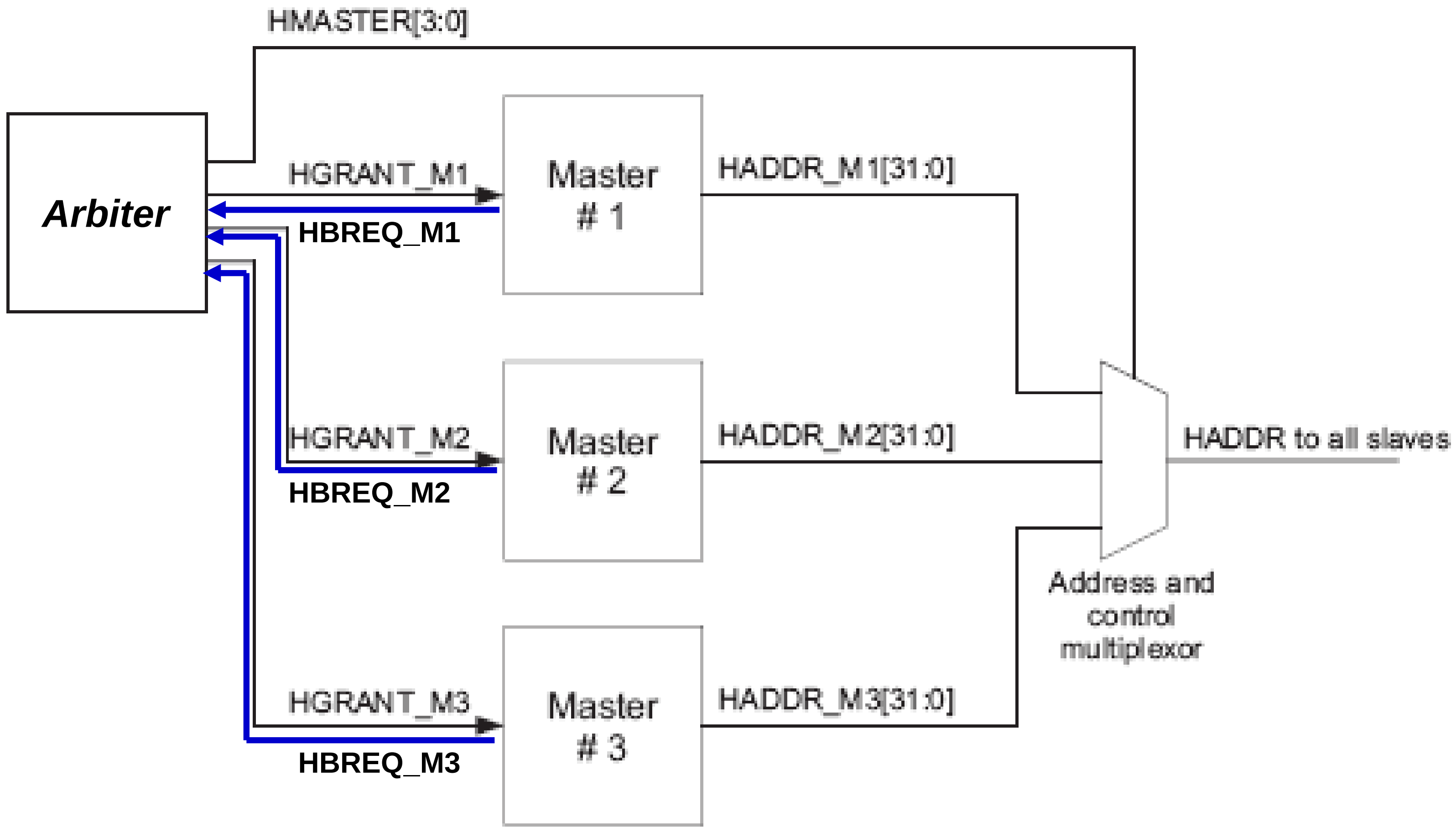


- 1 unidirectional address bus (HADDR)
- 2 unidirectional data buses (HWDATA, HRDATA)

- At any time only 1 active data bus

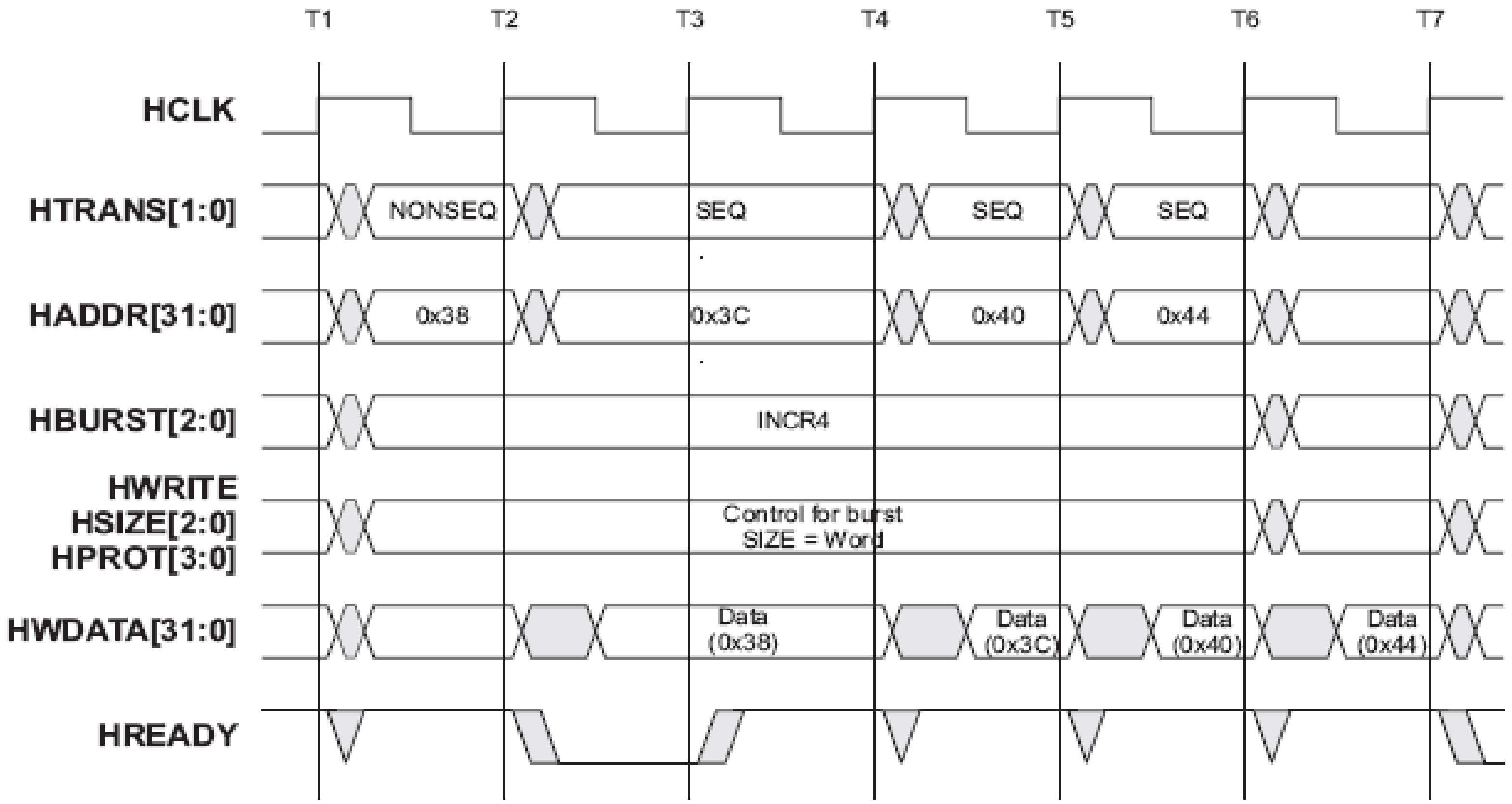
AHB Arbitration

- Arbitration protocol specified, but not the arbitration policy



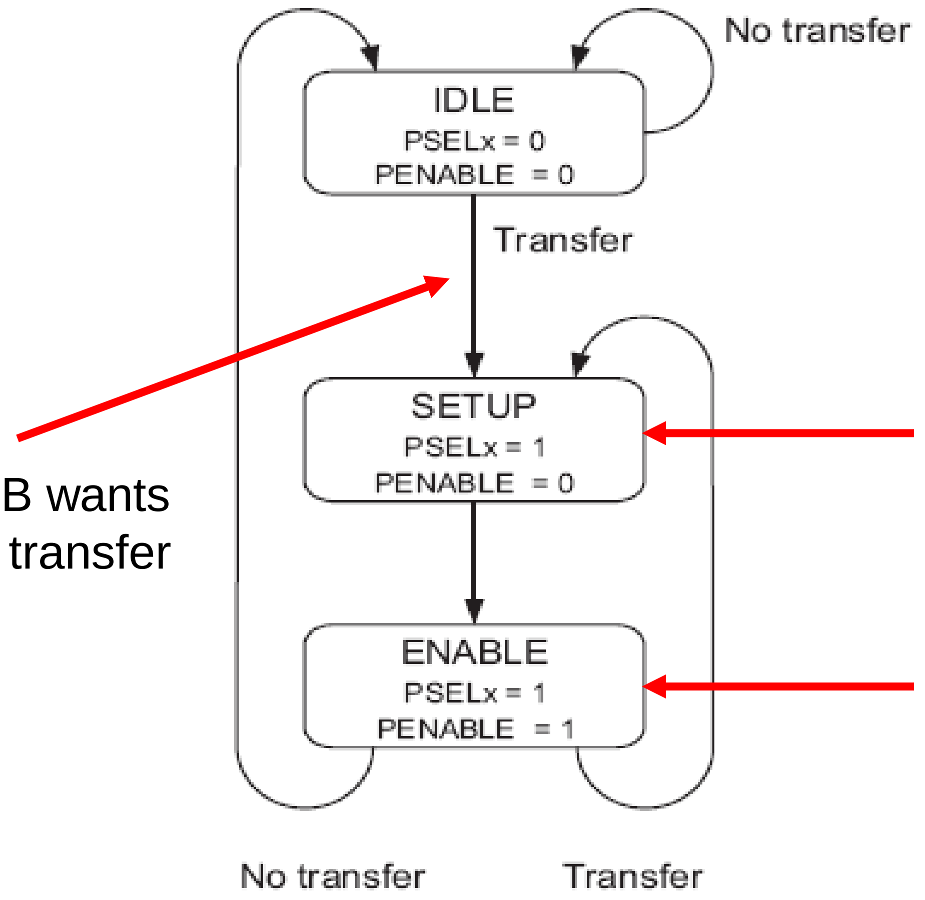
AHB Burst Transfers

- Bursts cut down on arbitration, handshaking time (improving performance)



APB State Diagram

- No (multi-cycle) bursts or pipelined transfers

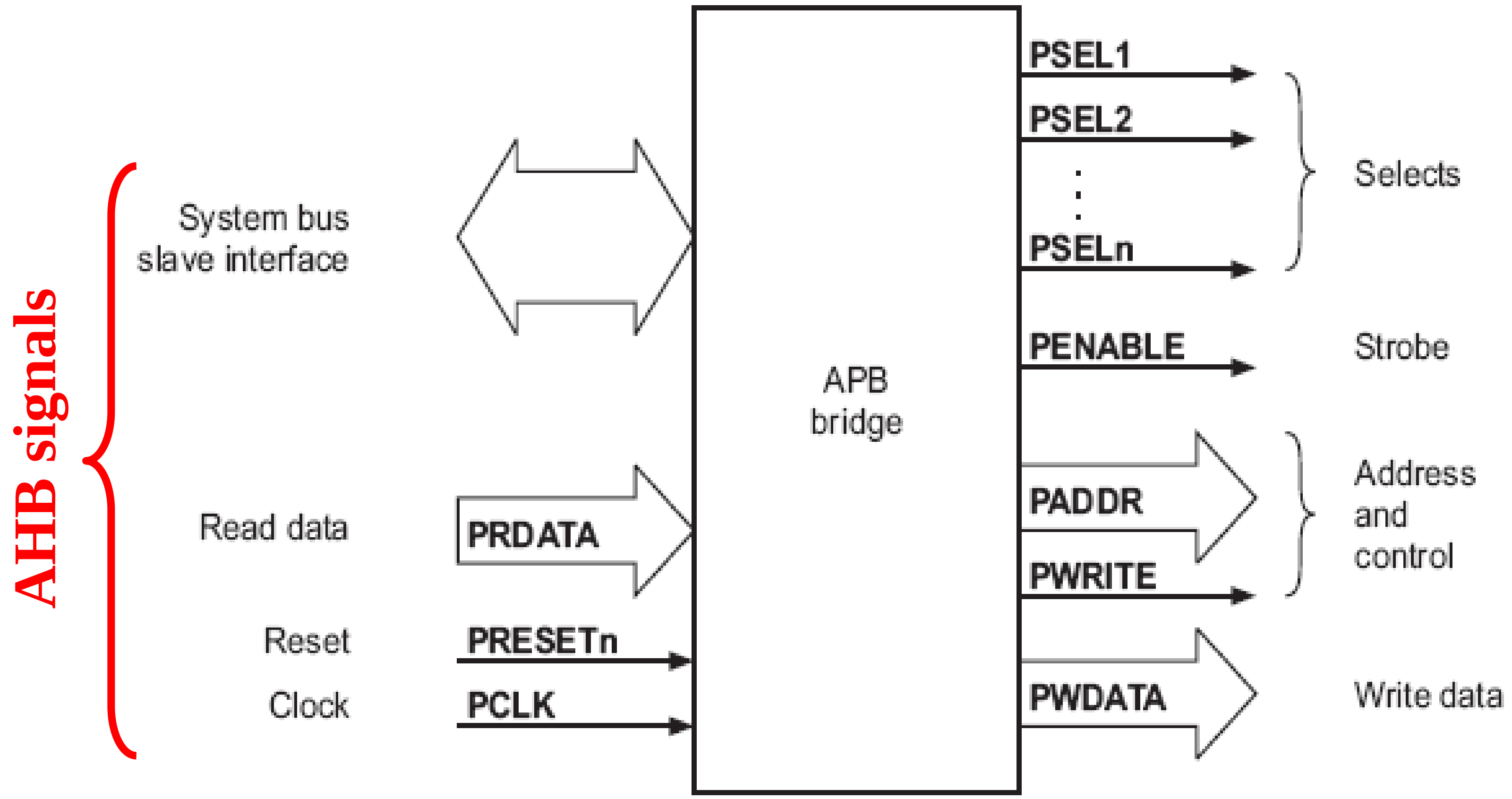


When AHB wants to drive a transfer

One cycle penalty for APB peripheral address decoding

Transfer occurs here

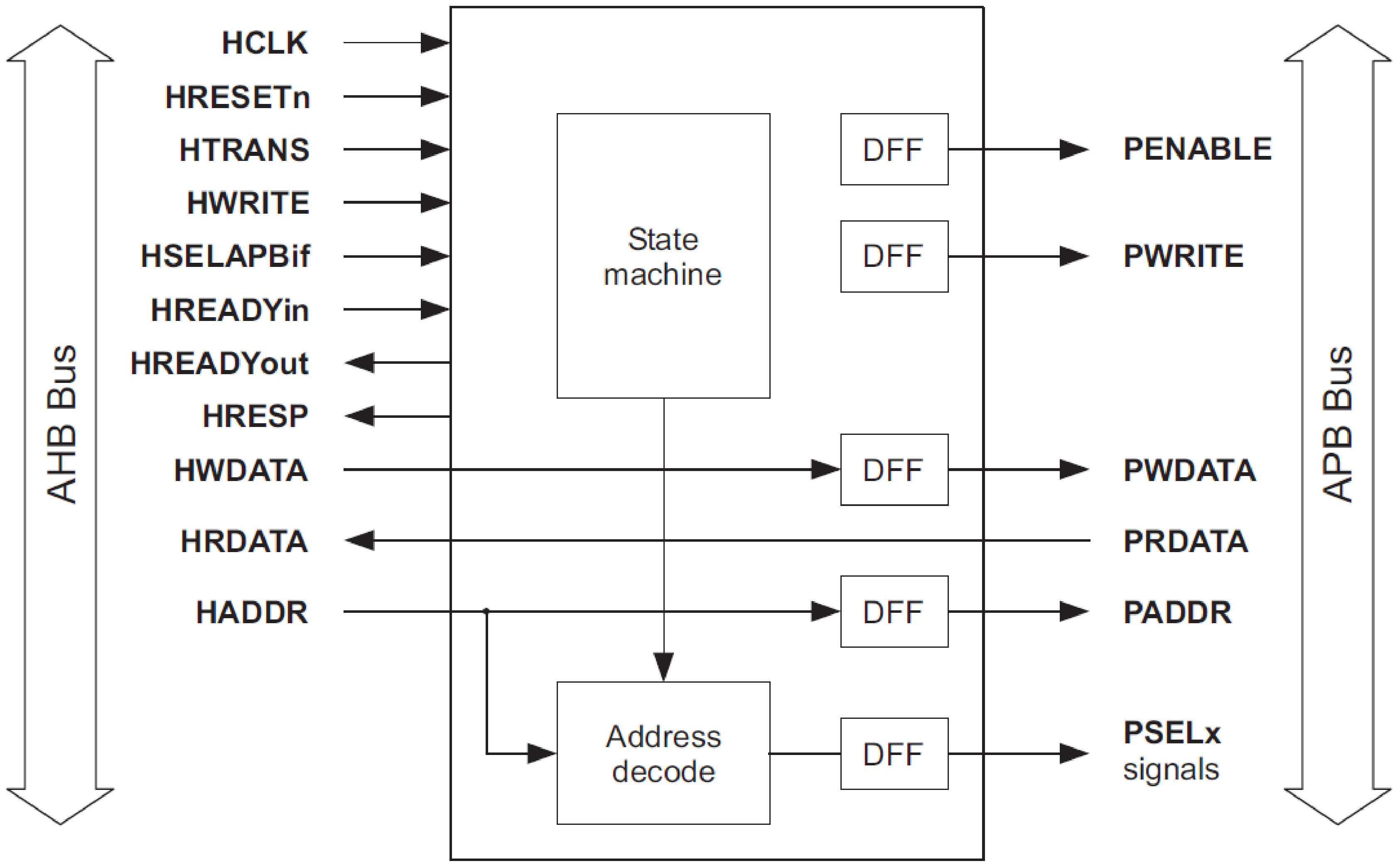
AHB-APB Bridge



High performance

Low power (and performance)

AHB-APB Bridge



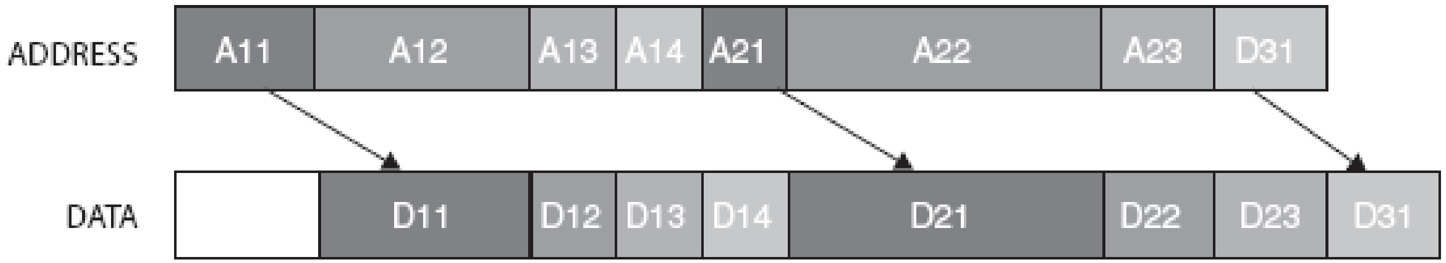
High performance

Low power (and performance)

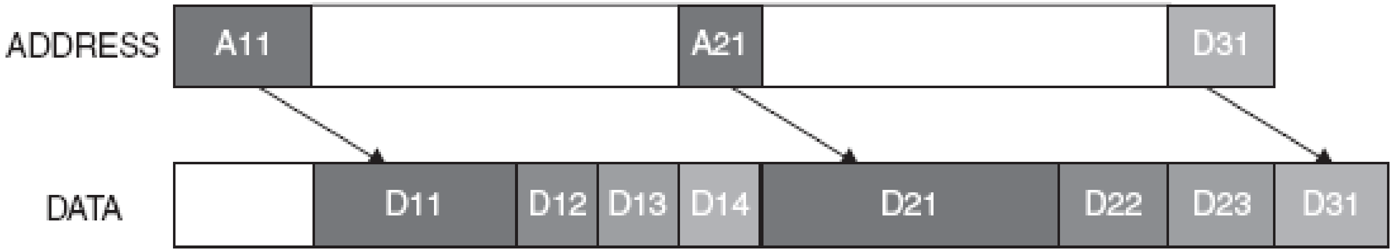
AHB vs. AXI

- Incremental improvements to AMBA since v2
- AMBA AXI – introduces separate read/write address, out-of-order operation

- AHB burst:
  - Address and data are locked together (single pipeline stage)
  - HREADY controls intervals of address and data

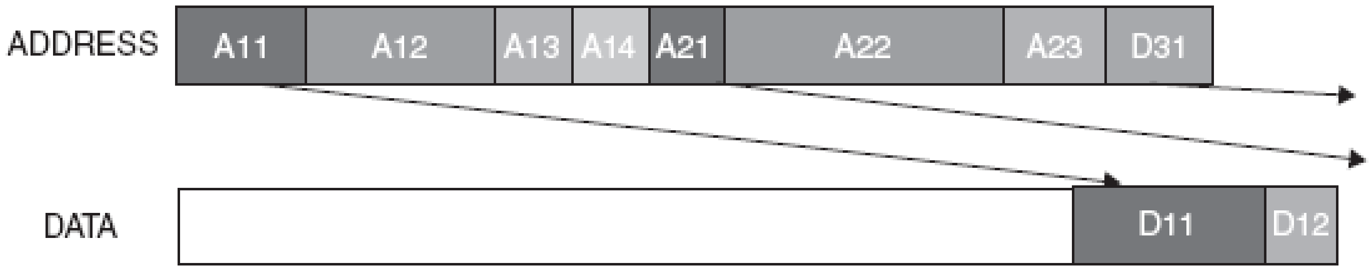


- AXI burst:

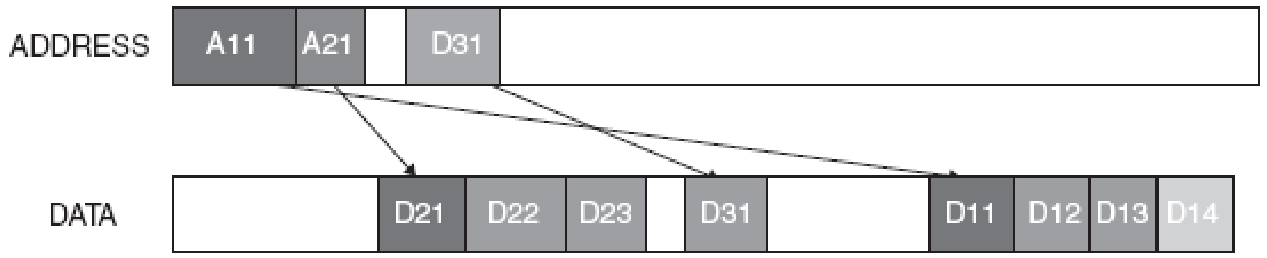


AHB vs. AXI (cont.)

- Out of order completion
- With AHB:
  - If one slave is very slow, all data is held up
  - SPLIT transactions provide very limited improvement

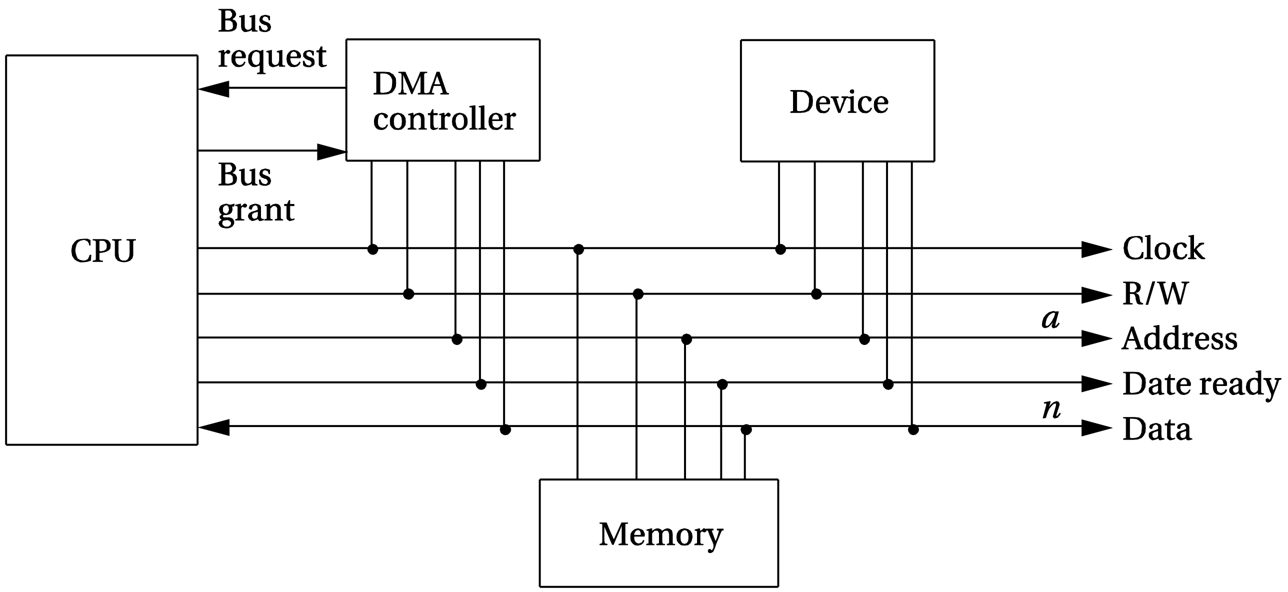


- With AXI:
  - Multiple outstanding addresses, out of order (OO) completion allowed



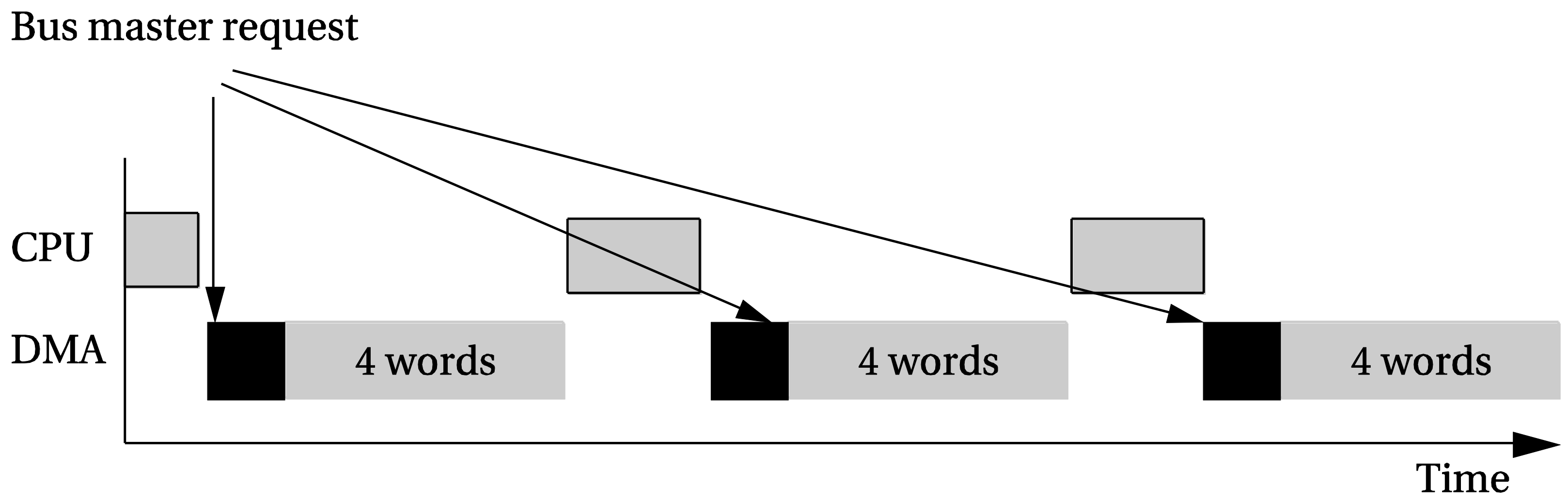
Buses and DMA

- Direct Memory Access (DMA) performs data transfers without executing instructions
  - CPU sets up transfer
  - DMA engine fetches, writes
- DMA controller is a separate unit on the bus



DMA Operation

- CPU sets DMA registers for start address, length
- DMA status register controls the unit
- Once DMA is bus master, it transfers automatically
  - May run continuously until complete
  - May use every nth bus cycle
- In the meantime, CPU carries on with other work

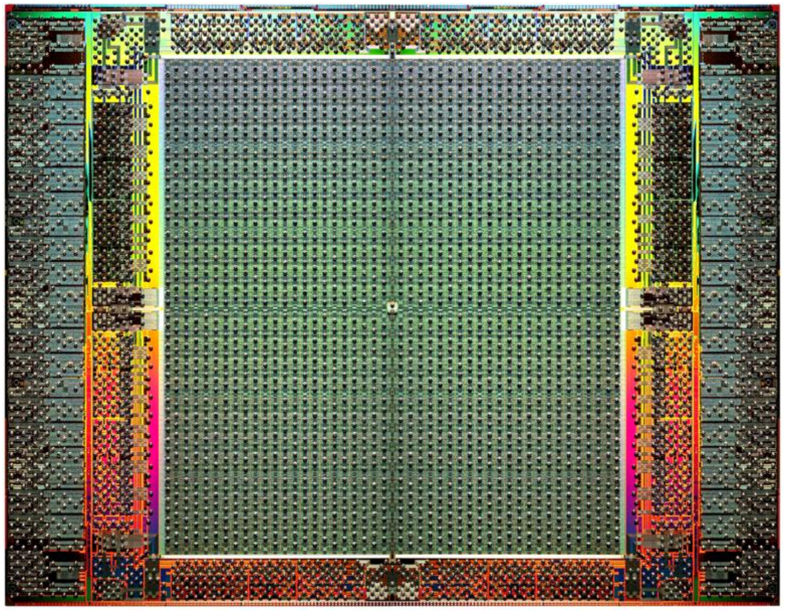


Another Approach to System Design

- Field-Programmable Gate Arrays (FPGAs)
  - Clean-slate design at the gate (logic) level
  - High volumes, since differentiated designs can be produced with the same chips
  - Simple silicon, so can be introduced rapidly in a new process
- Energy efficiency is not as great as with full-custom or ASIC designs
  - But is frequently good enough, particularly for algorithms that don't work well in software running on a general purpose CPU (e.g. compression, crypto)
  - So modern FPGAs contain "hard" functions that can be interconnected (and powered down when not used)
    - CPU cores (Xilinx Zynq family has 2 ARM cores with caches)
    - DSPs (hundreds)
    - Embedded RAM (megabits)
    - External DRAM controllers
    - Controllers for common I/O standards (e.g. Ethernet)
- Lots of logic, but:
  - Can't use all the logic; wires are usually the thing that limits a design
  - You pay for logic that you don't use

Introduction to the FPGA (cont.)

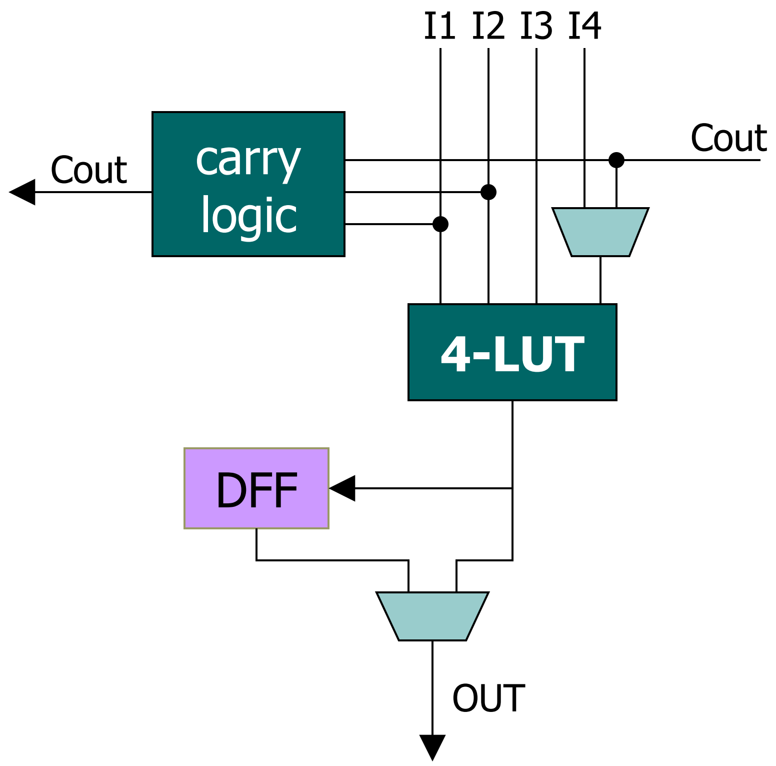
- Major players in the FPGA industry:
  - Chipmakers – device families
    - Xilinx – Spartan [II/3/6], Virtex [E/II/II-Pro/4/5/6/7]
    - Microsemi – eX, MX, SX, Axcelerator, Fusion ProASIC
    - Altera – APEX, FLEX, Arria [II/V], Cyclone [II/III/IV/V], Stratix, Stratix [II/III/IV/V]
    - Lattice – ECP3, SC/M, XP2
  - Software developers – CAD tools
    - 1st-party tools from Xilinx, Altera, etc.
    - Synopsys – Synplify Pro, Synplify Premier
    - Mentor Graphics – HDL Designer, Precision RTL, ModelSim


Altera Stratix IV EP4S40G5

FPGA Architecture

- FPGAs are composed of the following:
  - Configurable Logic Blocks (CLBs)
  - Programmable interconnect
  - Input/Output Buffers (IOBs)
  - Other stuff (clock trees, timers, memory, multipliers, processors, etc.)
- CLBs contain a number of Look-Up Tables (LUTs) and some sequential storage
  - LUTs are individually configured as logic gates, or can be combined into *n* bit wide arithmetic functions
  - Architecture specific

LUT-based Logic Element

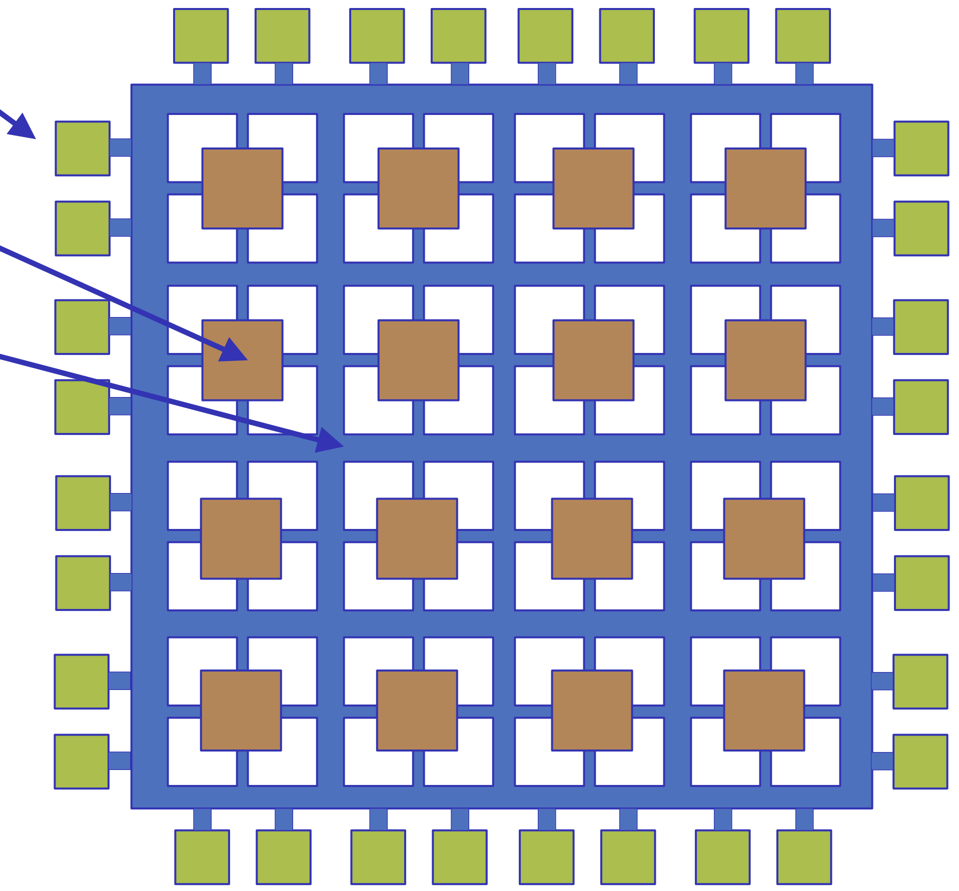


- Each LUT operates on four one-bit inputs
- Output is one data bit
- Can perform any Boolean function of four inputs
- 2^{2^4} = 65536 functions (4096 patterns)

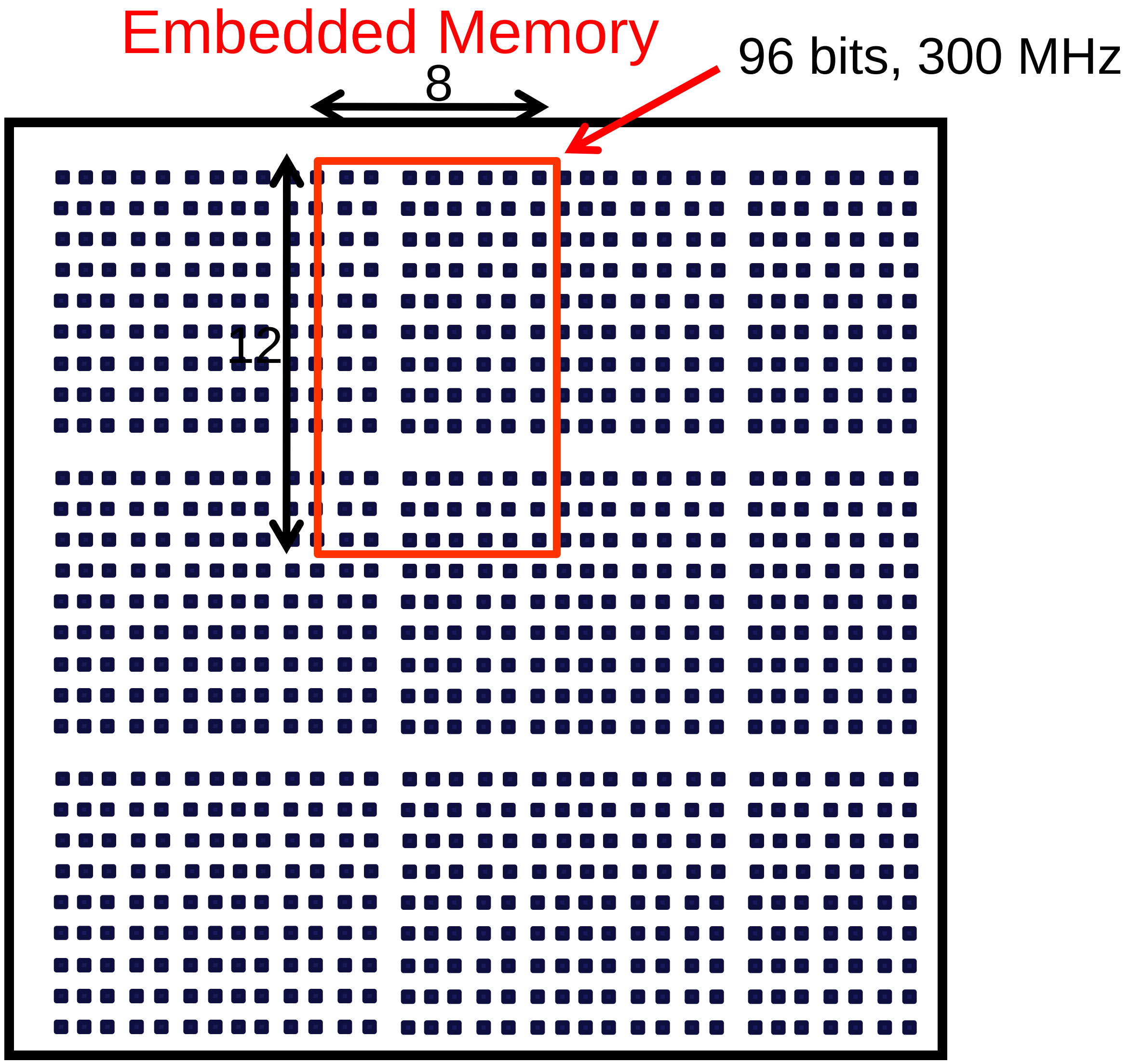
- The basic logic element can be more complex (multiplier, ALU, etc.)
- Contains some sort of programmable interconnect

FPGA Architecture (cont.)

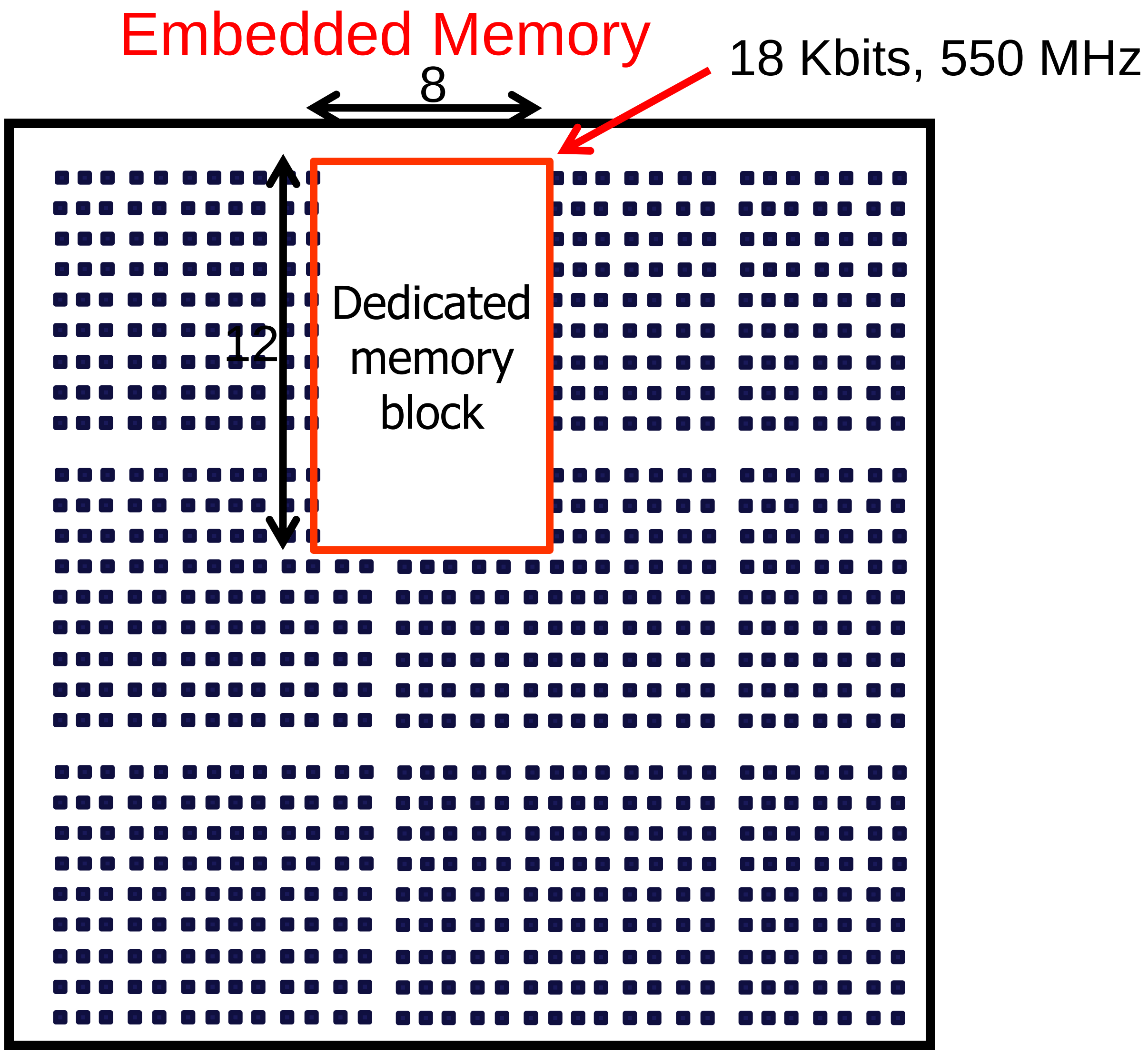
- Input/Output Buffers (IOBs)
- Configurable Logic Blocks (CLBs)
- Programmable interconnect mesh
- Generic *island-style* FPGA architecture



Optimized Resources: Dedicated Logic



Optimized Resources: Dedicated Logic



Optimized Resources: Dedicated Logic

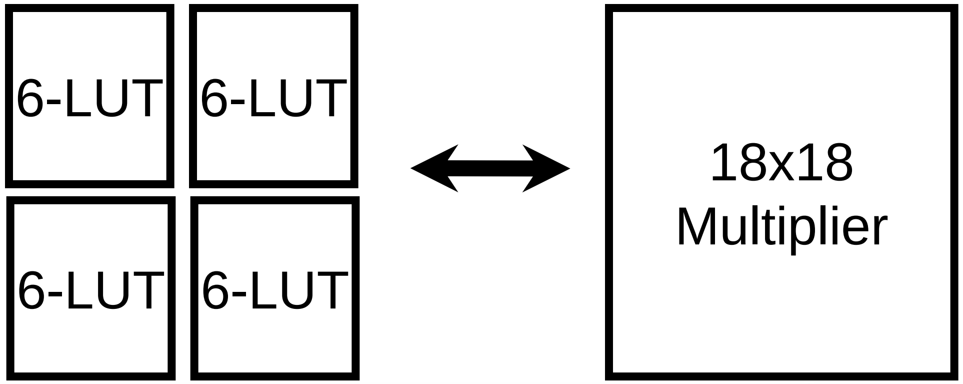
Multiplication

18x18 multiply

Type	# LUTs	Latency	Speed
LUT	~400	5 clks	380 MHz
Dedicated 18x18 Multiplier	0	3 clks	450 MHz

Virtex-5 (6-LUTs)

Very rough estimate of Silicon area comparison
(assuming SX95 andLX110 have about the same die size)

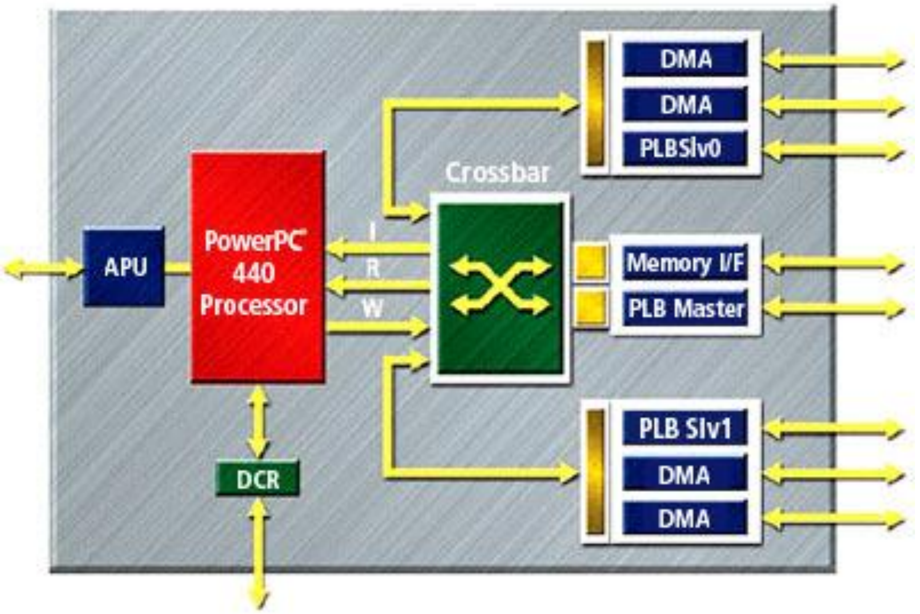


In other word you can replace one LUT based 18x18 multiplier With 100 dedicated 18x18 Multipliers!!!

Optimized Resources: Dedicated Logic

Processor

PowerPC hard-core



MicroBlaze soft-core

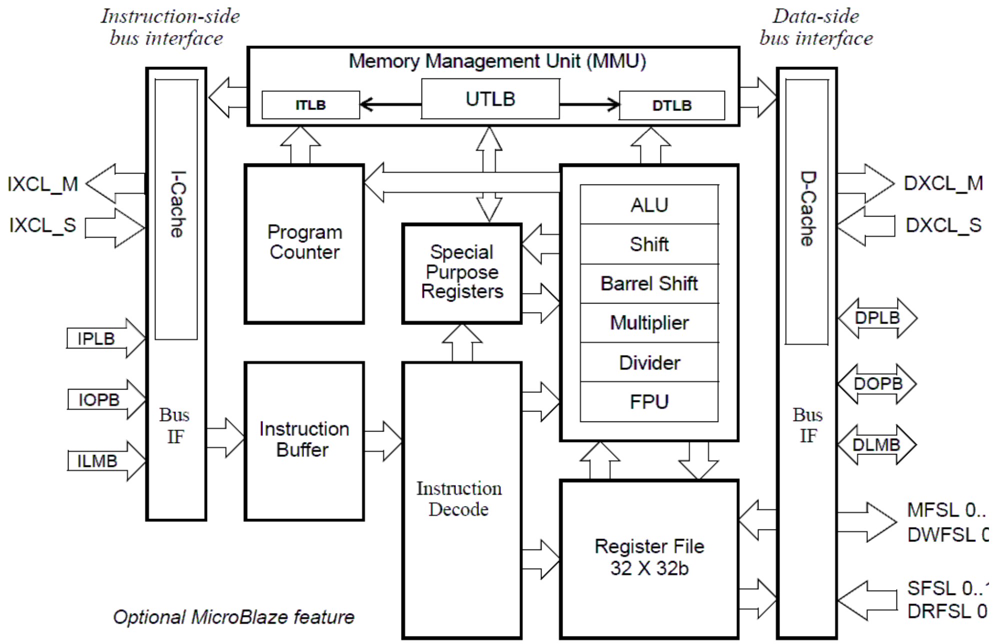


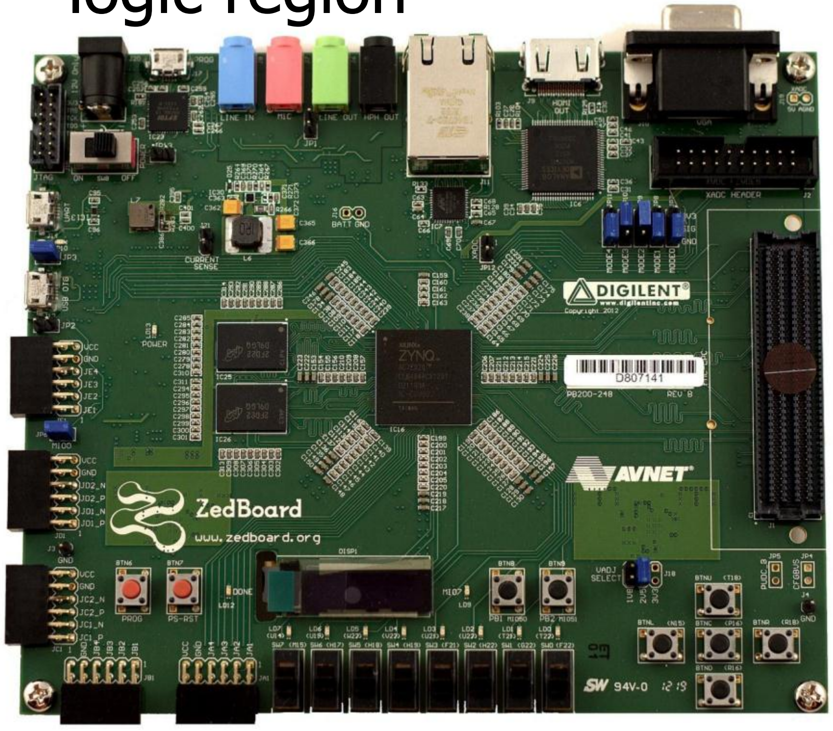
Figure 1-1: **MicroBlaze Core Block Diagram**

- 500 MHz
- Super scalor
- Highspeed 2x5 switch fabric

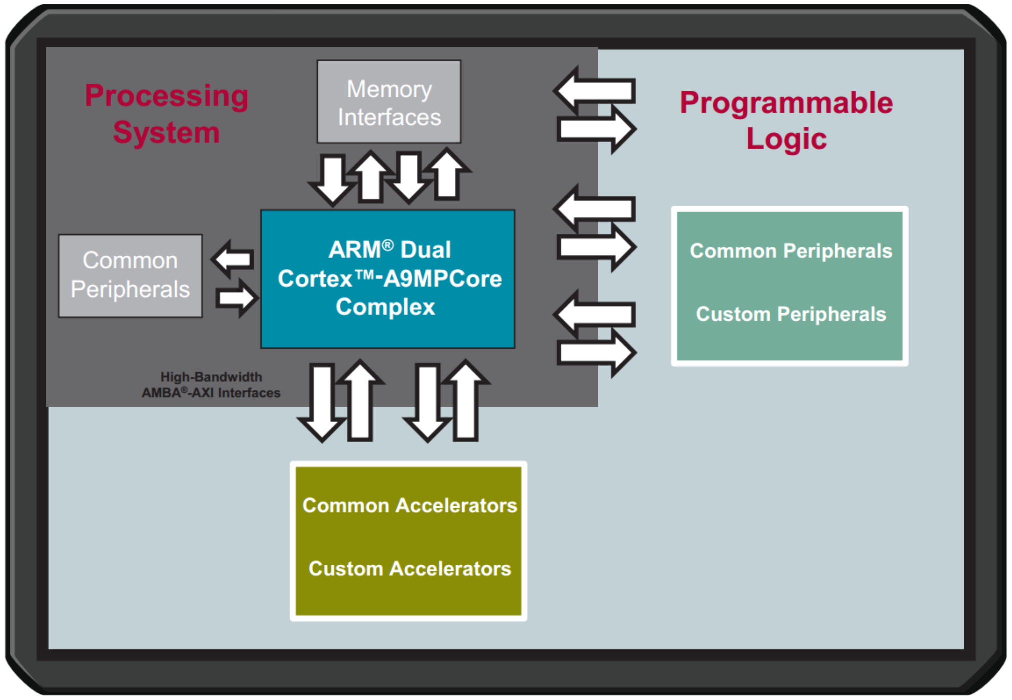
- 250 MHz
- Simple scalar

Xilinx Extensible Processing Platform

- Coupling of dual-core ARM Cortex-A9 with reconfigurable logic
- "Processing System" is fully integrated and hardwired, and the platform can behave like a typical processor by default
- ARM CPU can control reconfiguration of programmable logic region



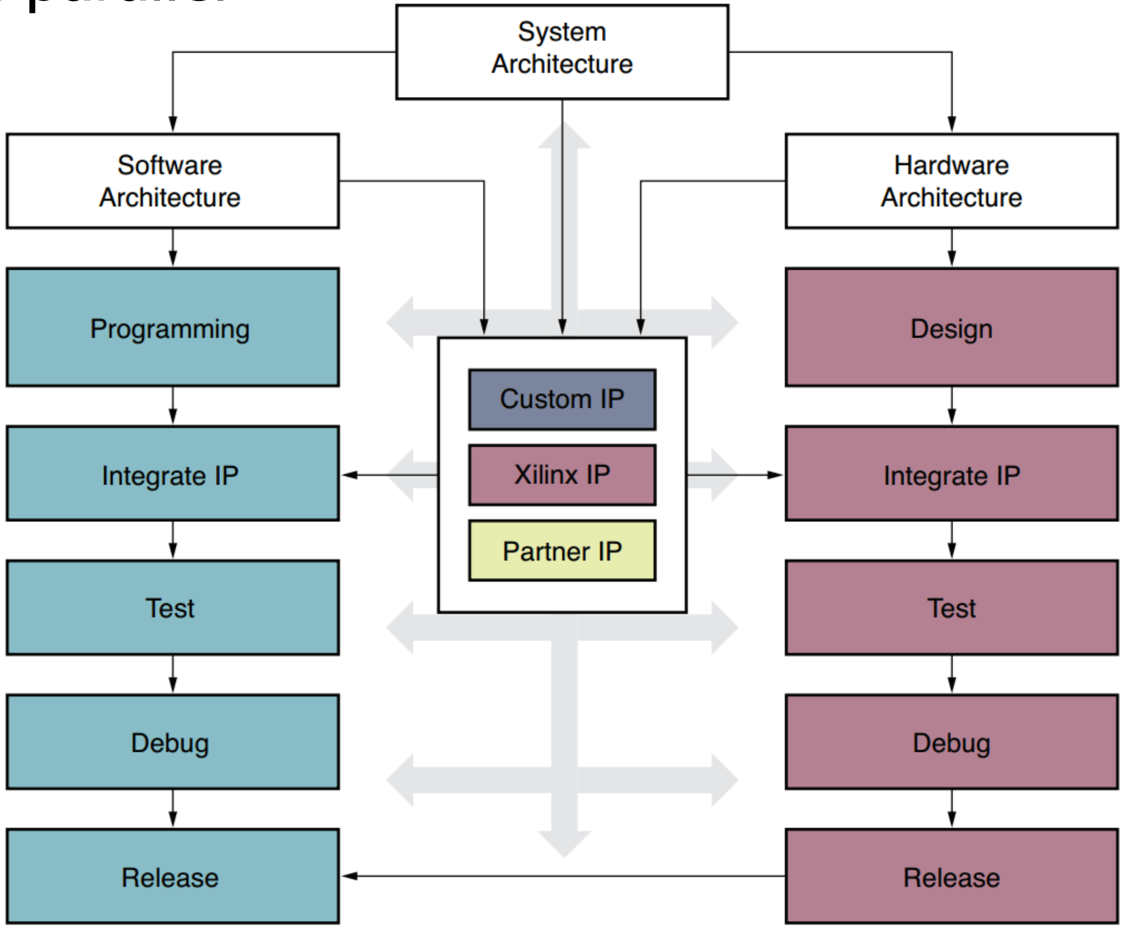
Digilent ZedBoard



Xilinx Zynq Extensible Processing Platform

Software Development Flow

- IP-centric flow (both HW and SW)
- System Architect, Logic Designer, and Software Developer can work in parallel



Acknowledgments

- These slides are inspired in part by material developed and copyright by:
  - Marilyn Wolf (Georgia Tech)
  - Sudeep Prasicha (Colorado State)
  - Nikil Dutt (UC-Irvine)