ARM Based Systems

Géza Lőre

Research Engineer, ARM Ltd
Geza.Lore@arm.com

Agenda

- Introduction to ARM Ltd
- ARM Architecture / Programmers Model
- Microarchitectures
- AMBA
- Example ARM based SoC

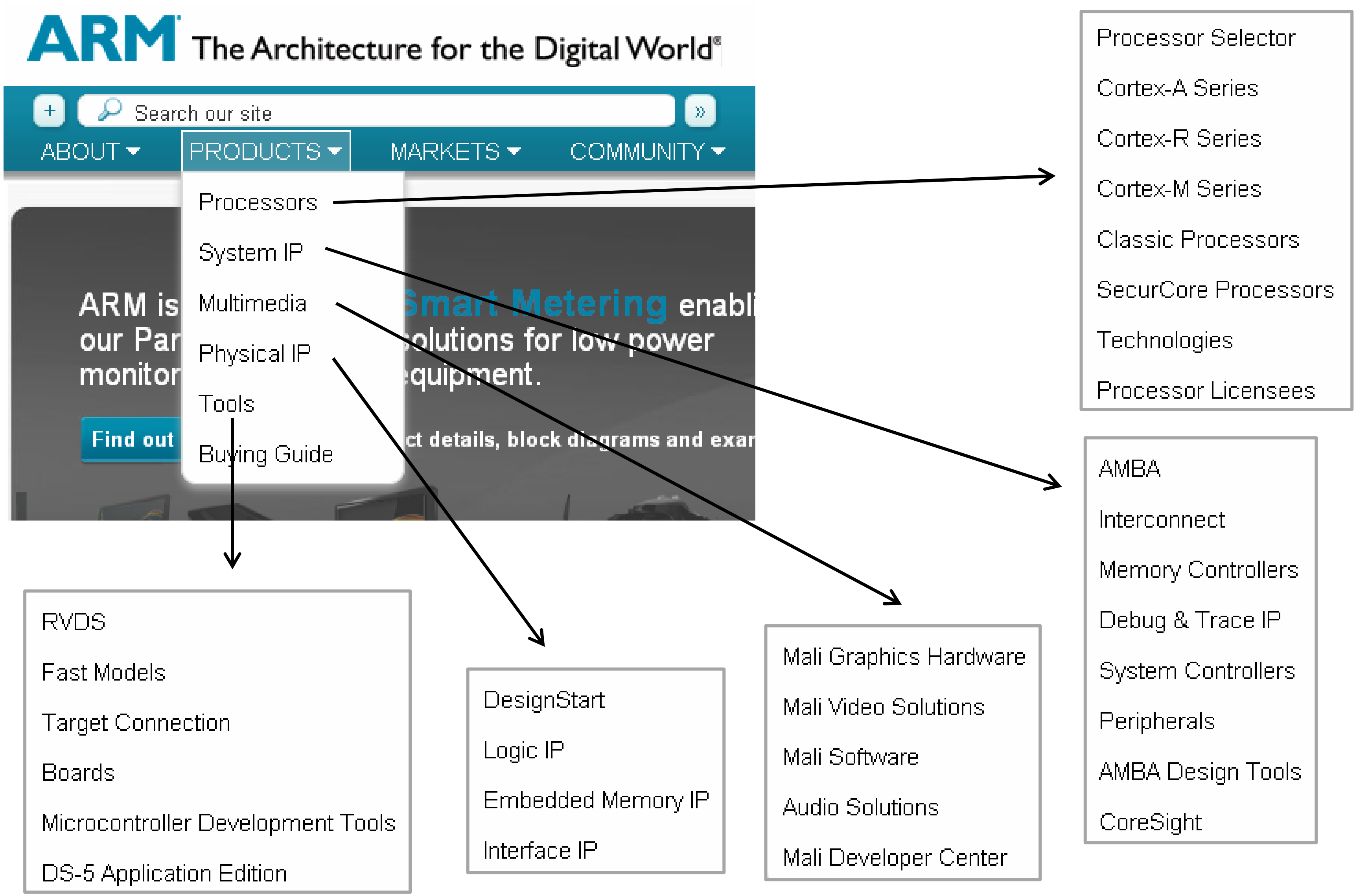
ARM Ltd

- Founded in 1990
- Spun out of Acorn Computers
- Primary product: RISC processor cores

- ARM is an ***Intellectual Property*** (IP) company
  - We do not fabricate or sell silicon
  - We license technology to our partners
  - Our partners design and fabricate products, which they sell to their customers

- Develops technologies to assist with the design-in of the ARM architecture
  - Software tools, application software
  - Development boards, debug hardware
  - Bus architectures, peripherals, etc.

ARM's Activities



- Processor Selector
- Cortex-A Series
- Cortex-R Series
- Cortex-M Series
- Classic Processors
- SecurCore Processors
- Technologies
- Processor Licensees

- AMBA
- Interconnect
- Memory Controllers
- Debug & Trace IP
- System Controllers
- Peripherals
- AMBA Design Tools
- CoreSight

- Mali Graphics Hardware
- Mali Video Solutions
- Mali Software
- Audio Solutions
- Mali Developer Center

- DesignStart
- Logic IP
- Embedded Memory IP
- Interface IP

- RVDS
- Fast Models
- Target Connection
- Boards
- Microcontroller Development Tools
- DS-5 Application Edition

ARM Connected Community – 550+

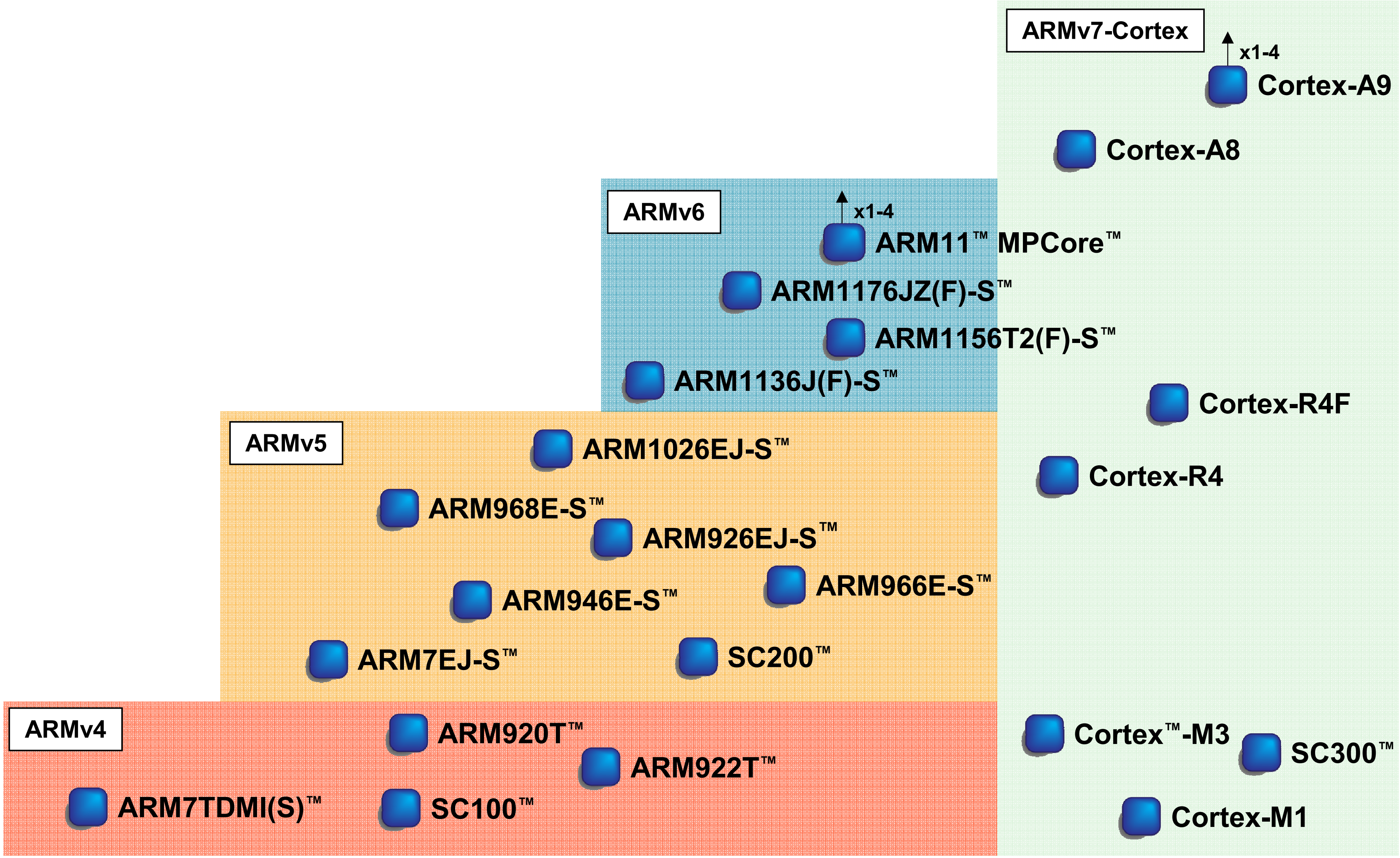
Applications

You're never more than 6ft away from an ~~SPIDER~~

Agenda

- Introduction to ARM Ltd
- ARM Architecture / Programmers Model
- Microarchitectures
- AMBA
- Example ARM based SoC

Architecture Versions



ARMv7-Cortex

x1-4

Cortex-A9

Cortex-A8

Cortex-R4F

Cortex-R4

Cortex™-M3

SC300™

Cortex-M1

ARMv6

x1-4

ARM11™ MPCore™

ARM1176JZ(F)-S™

ARM1156T2(F)-S™

ARM1136J(F)-S™

ARMv5

ARM1026EJ-S™

ARM968E-S™

ARM926EJ-S™

ARM966E-S™

ARM946E-S™

ARM7EJ-S™

SC200™

ARMv4

ARM920T™

ARM922T™

ARM7TDMI(S)™

SC100™

Data Sizes and Instruction Sets

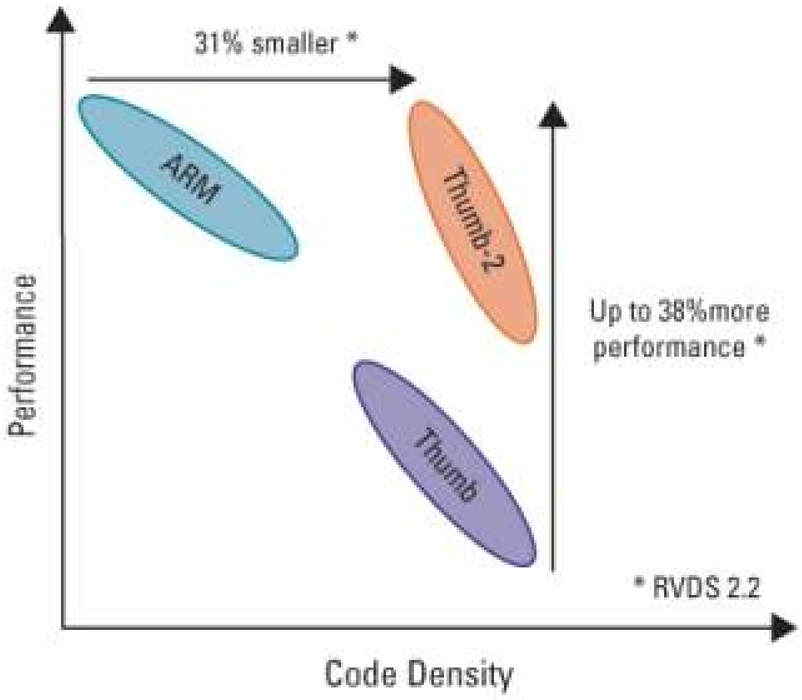
- The ARM is a 32-bit Load/Store architecture
  - Registers are 32 bit wide
  - Operations carried out on registers
  - Memory accessed only with Load and Store operations

- Most ARM's implement two instruction sets
  - 32-bit ARM Instruction Set
  - 16-bit Thumb Instruction Set

- Multiple optional extensions are available
  - Jazelle - execute Java bytecode in hardware
  - NEON - Advanced SIMD engine
  - TrustZone - security extensions

ARM vs Thumb vs Thumb-2

- ARM Instruction set
  - High performance
  - All instructions are 32 bit
  - All instructions can be conditional
- Thumb
  - Re-encoded subset of the ARM ISA
  - All instructions are 16 bit
  - Restricted functionality
- ***Thumb-2***
  - Extension of Thumb
  - 16 and 32 bit instructions mixed
  - Full functionality available



- Instruction sets can be mixed in the same program: *Interworking*

Intsruction set details

- Conditional execution of ARM instructions

```
    CMP    r3,#0
    BEQ    skip
    SUB    r0,r1,r2
skip
```

```
CMP    r3,#0
SUBNE  r0,r1,r2
```

- Barrel shifter in the data path

```
    ADD    r1,r2,r2,LSL #3    ; r1 = 9*r2
    SUB    r4,r4,r5,LSR #2    ; r4 = r4 - 0.25*r5
    RSB    r3,r3,r3,LSL #2    ; r3 = 3*r3
```

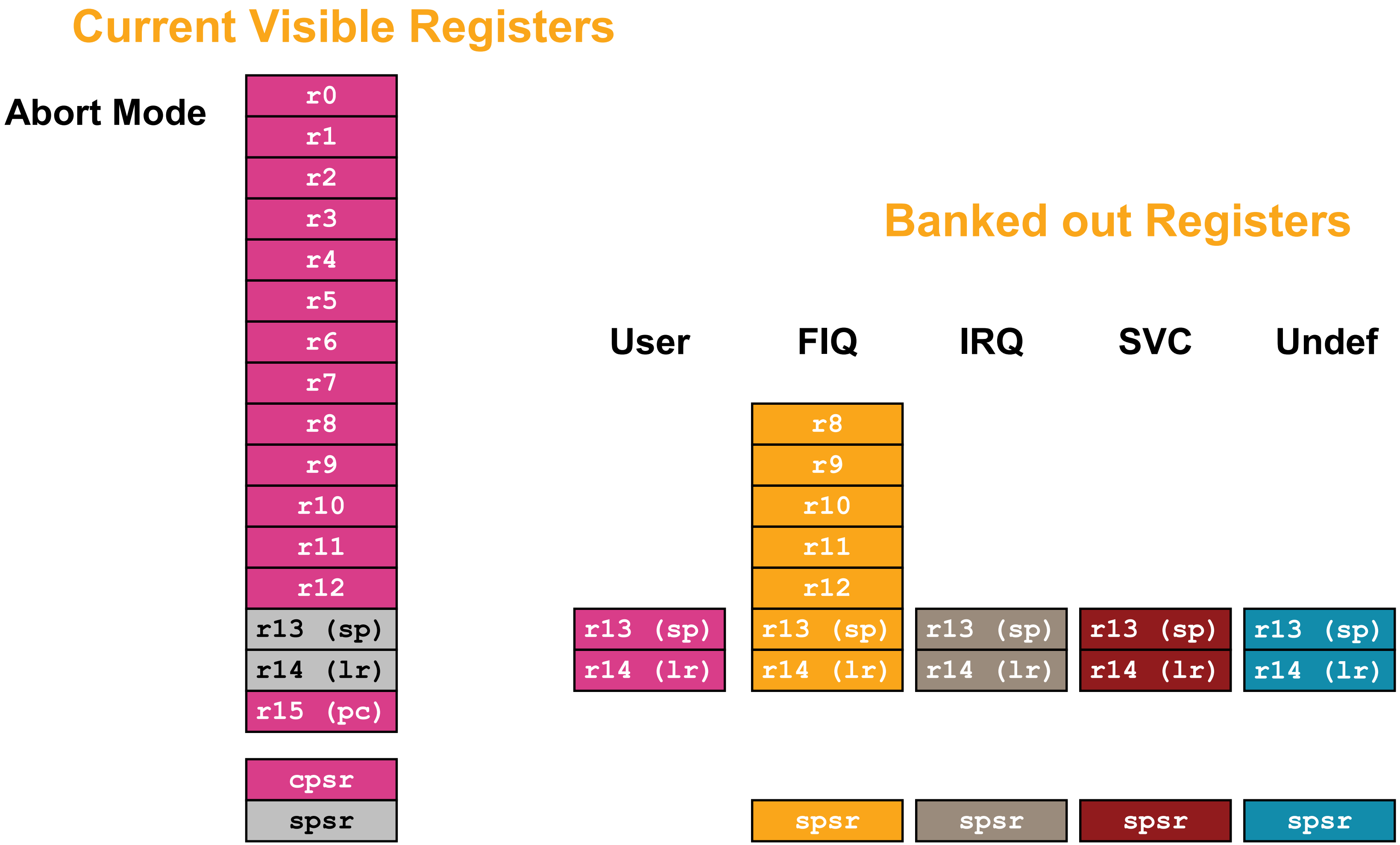
- Check the appropriate ARM Architecture Reference Manual (ARM ARM)

  http://infocenter.arm.com/

Processor Modes (Classic)

- The ARM has seven basic operating modes:
  - **User** : unprivileged mode under which most tasks run
  - **FIQ** : entered when a high priority (fast) interrupt is raised
  - **IRQ** : entered when a low priority (normal) interrupt is raised
  - **Supervisor** : entered on reset and when a Software Interrupt instruction is executed
  - **Abort** : used to handle memory access violations
  - **Undef** : used to handle undefined instructions
  - **System** : privileged mode using the same registers as user mode

The ARM Register Set



Current Visible Registers

Abort Mode

- r0
- r1
- r2
- r3
- r4
- r5
- r6
- r7
- r8
- r9
- r10
- r11
- r12
- r13 (sp)
- r14 (lr)
- r15 (pc)

- cpsr
- spsr

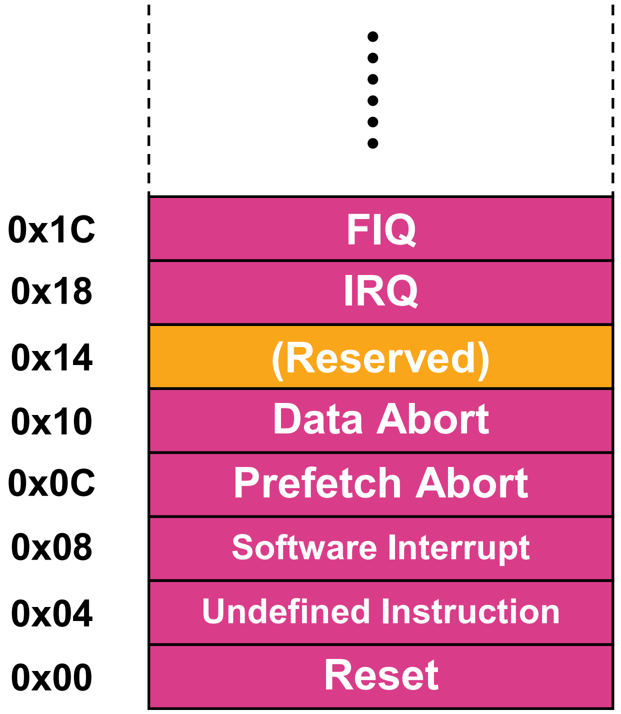
Banked out Registers

User	FIQ	IRQ	SVC	Undef
	r8			
	r9			
	r10			
	r11			
	r12			
r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)
r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)
	spsr	spsr	spsr	spsr

Exception Handling

- When an exception occurs, the ARM:
  - Copies CPSR into SPSR_<mode>
  - Sets appropriate CPSR bits
    - Change to ARM state
    - Change to exception mode
    - Disable interrupts (if appropriate)
  - Stores the return address in LR_<mode>
  - Sets PC to vector address
- To return, exception handler needs to:
  - Restore CPSR from SPSR_<mode>
  - Restore PC from LR_<mode>

This can only be done in ARM state.

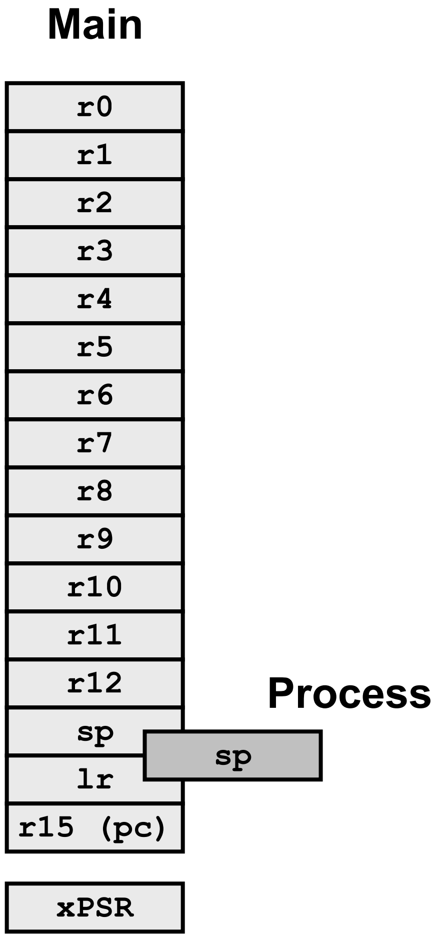


	⋮
0x1C	FIQ
0x18	IRQ
0x14	(Reserved)
0x10	Data Abort
0x0C	Prefetch Abort
0x08	Software Interrupt
0x04	Undefined Instruction
0x00	Reset

Vector Table

Cortex-M Programmer's Model

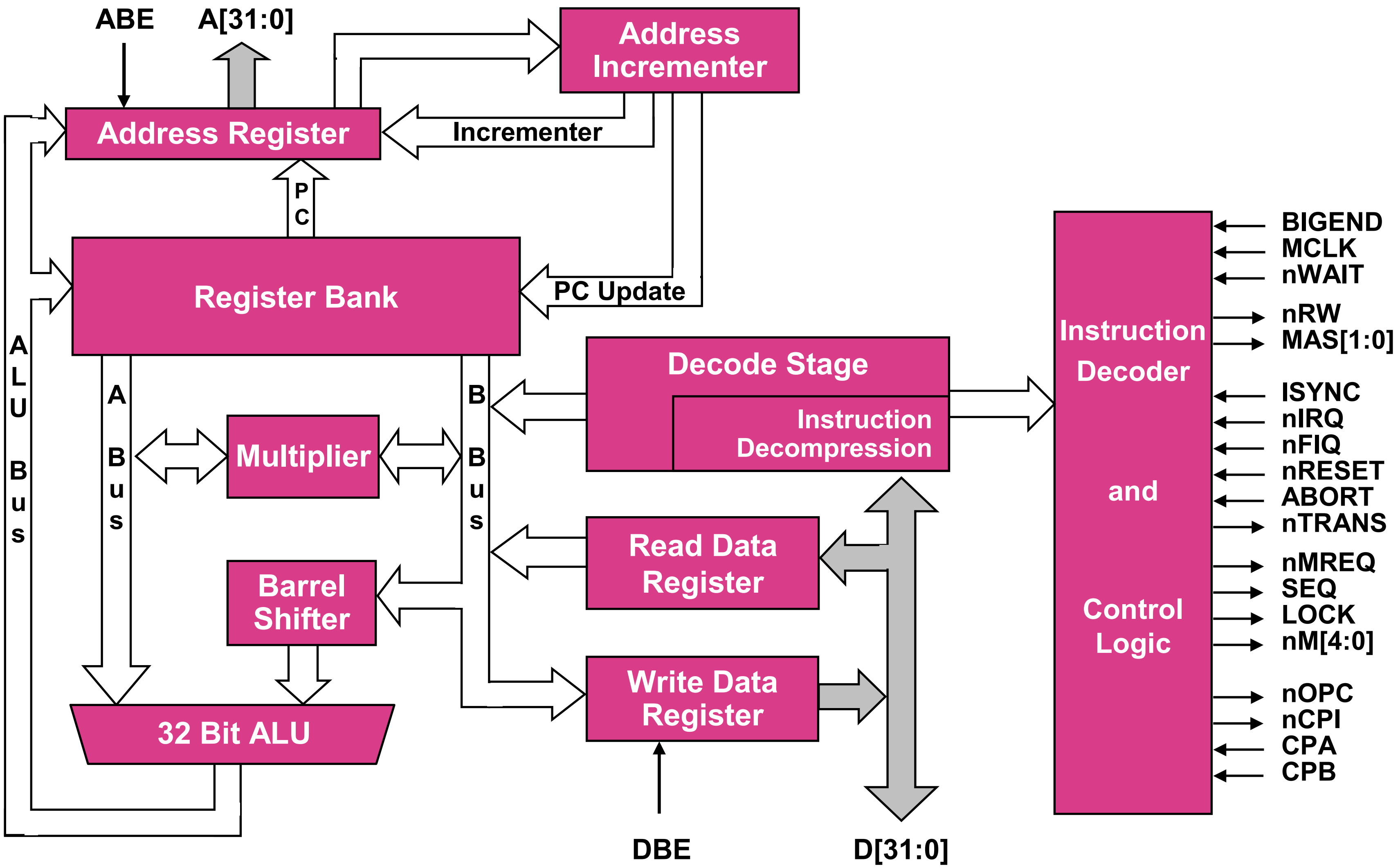
- ARMv7-M (Cortex-M class processors) have a different programmer's model
- Fully programmable in C
- Stack-based exception model
- Only two processor modes
  - Thread Mode for User tasks
  - Handler Mode for OS tasks and exceptions
- Vector table contains addresses



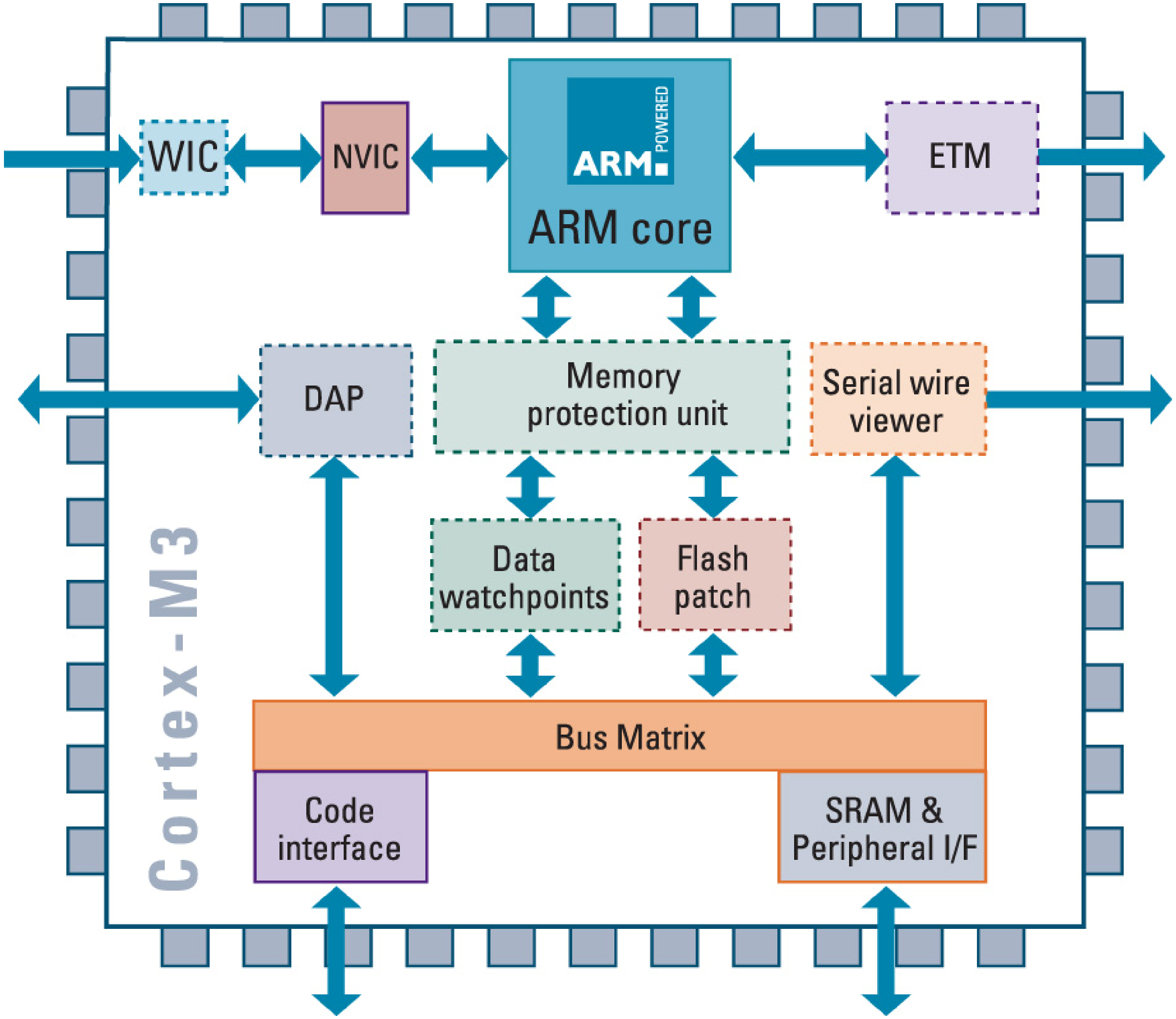
Agenda

- Introduction to ARM Ltd
- ARM Architecture / Programmers Model
- Microarchitectures
- AMBA
- Example ARM based SoC

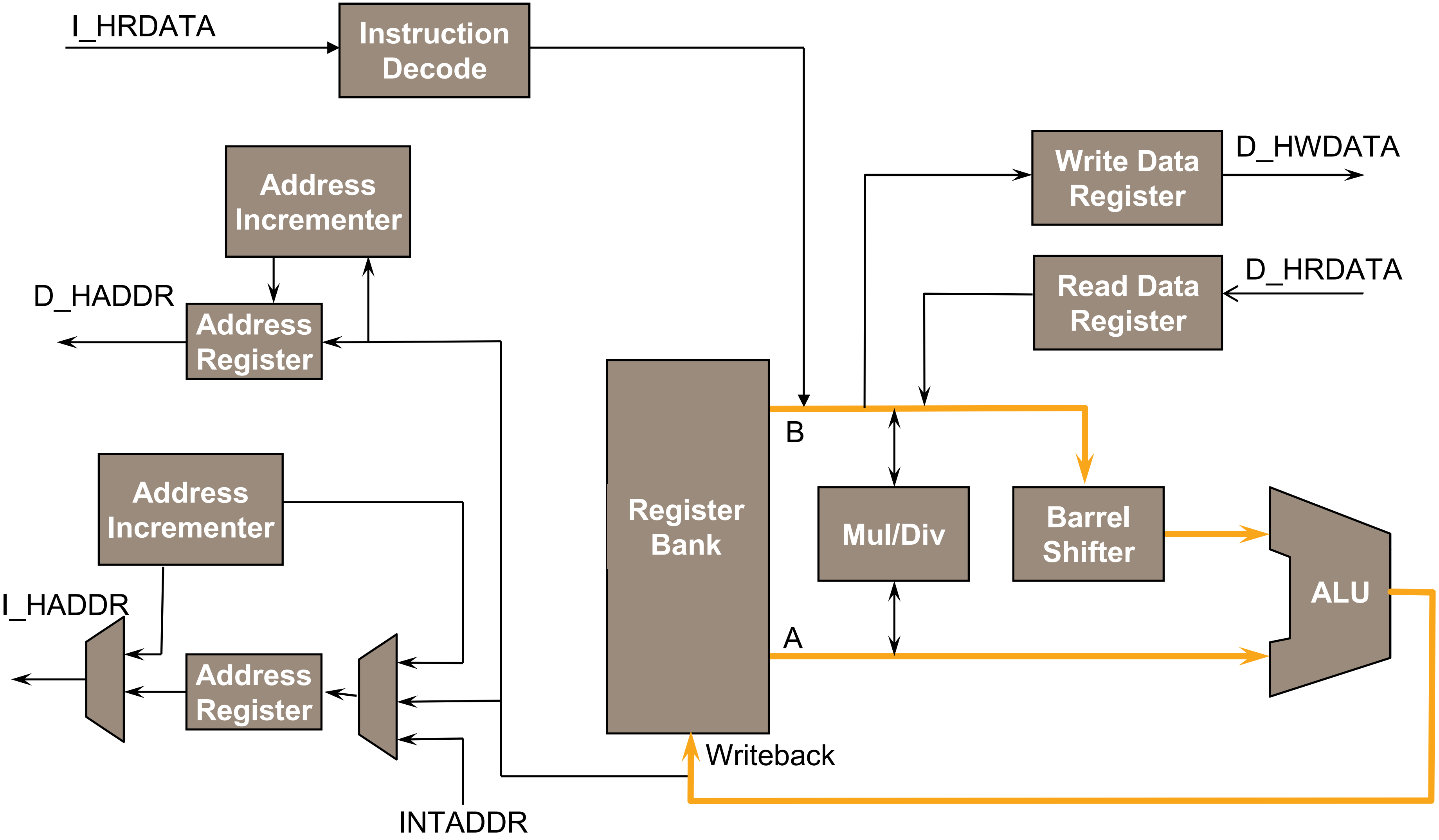
The ARM7TDMI Core (ARMv4)



The Cortex-M3 processor (ARMv7-M)

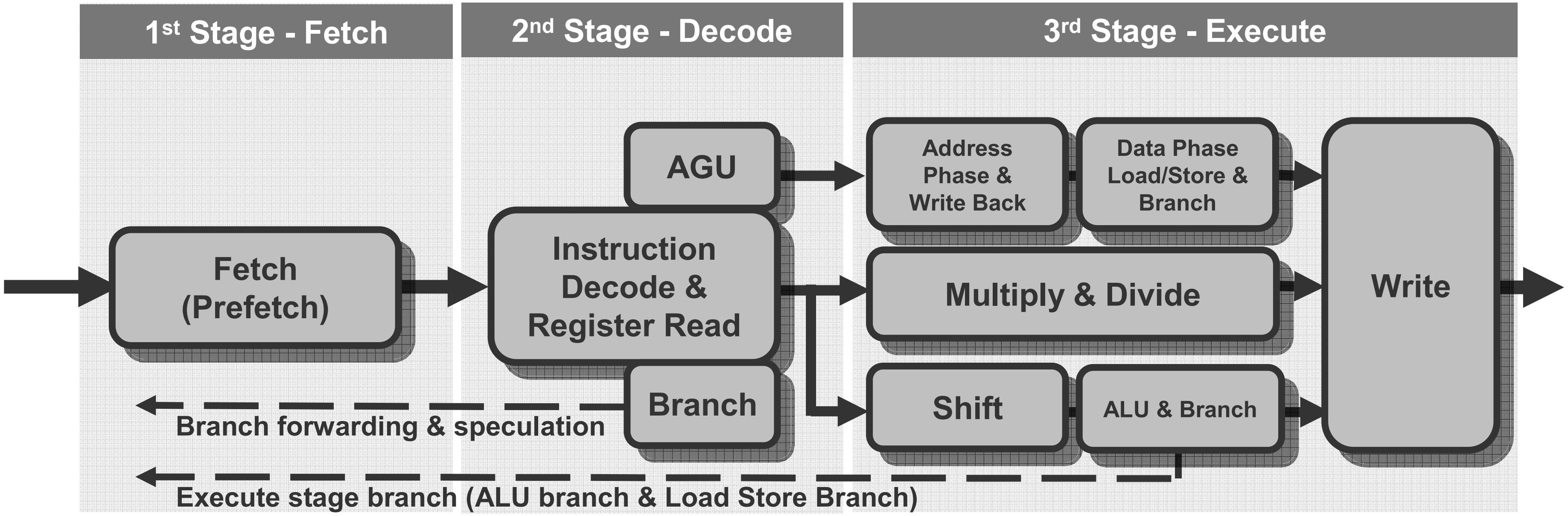


Cortex-M3 Datapath



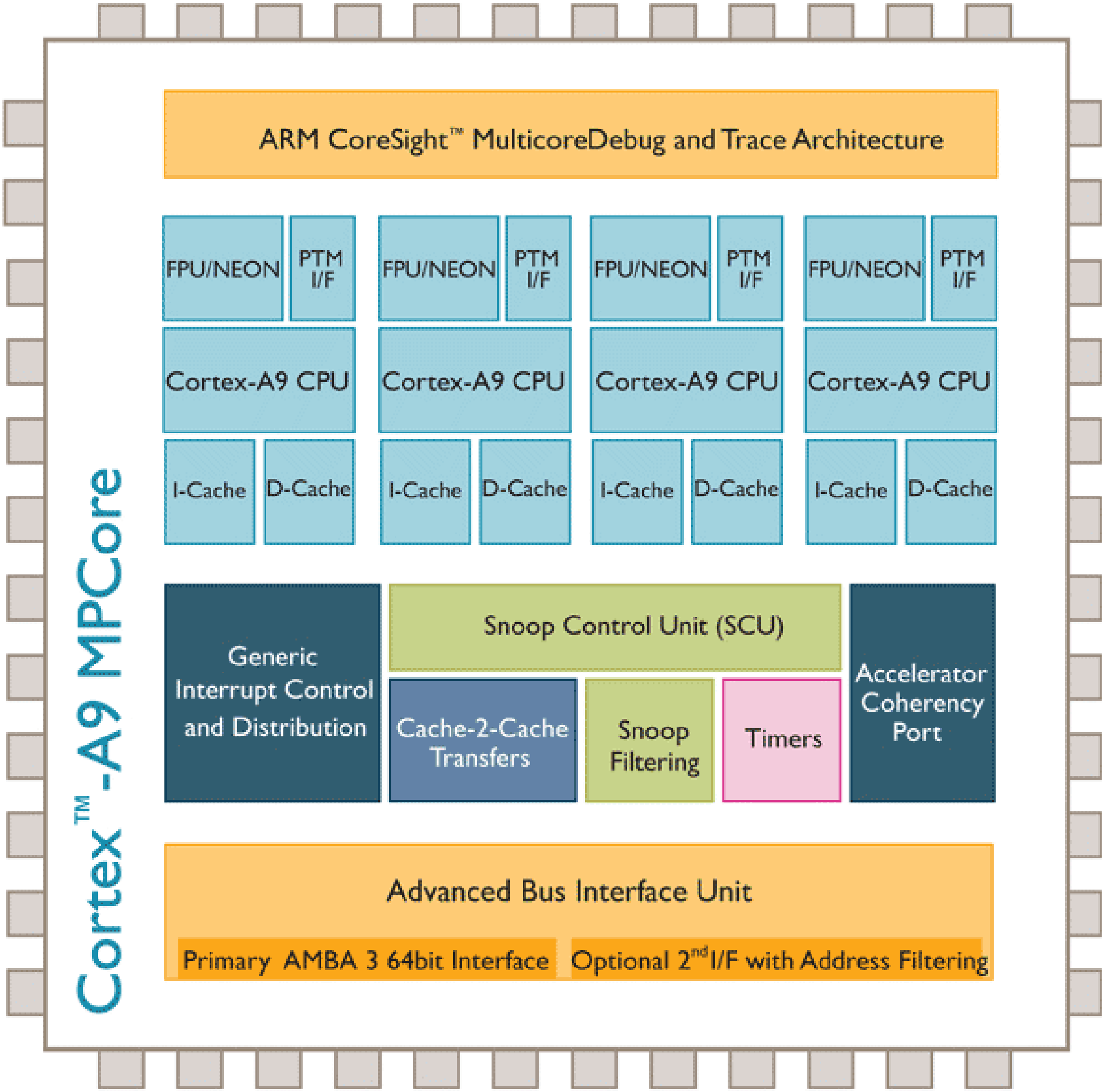
Cortex-M3 Pipeline

- Cortex-M3 has 3-stage fetch-decode-execute pipeline
  - Similar to ARM7
  - But Cortex-M3 does more in each stage to increase overall performance



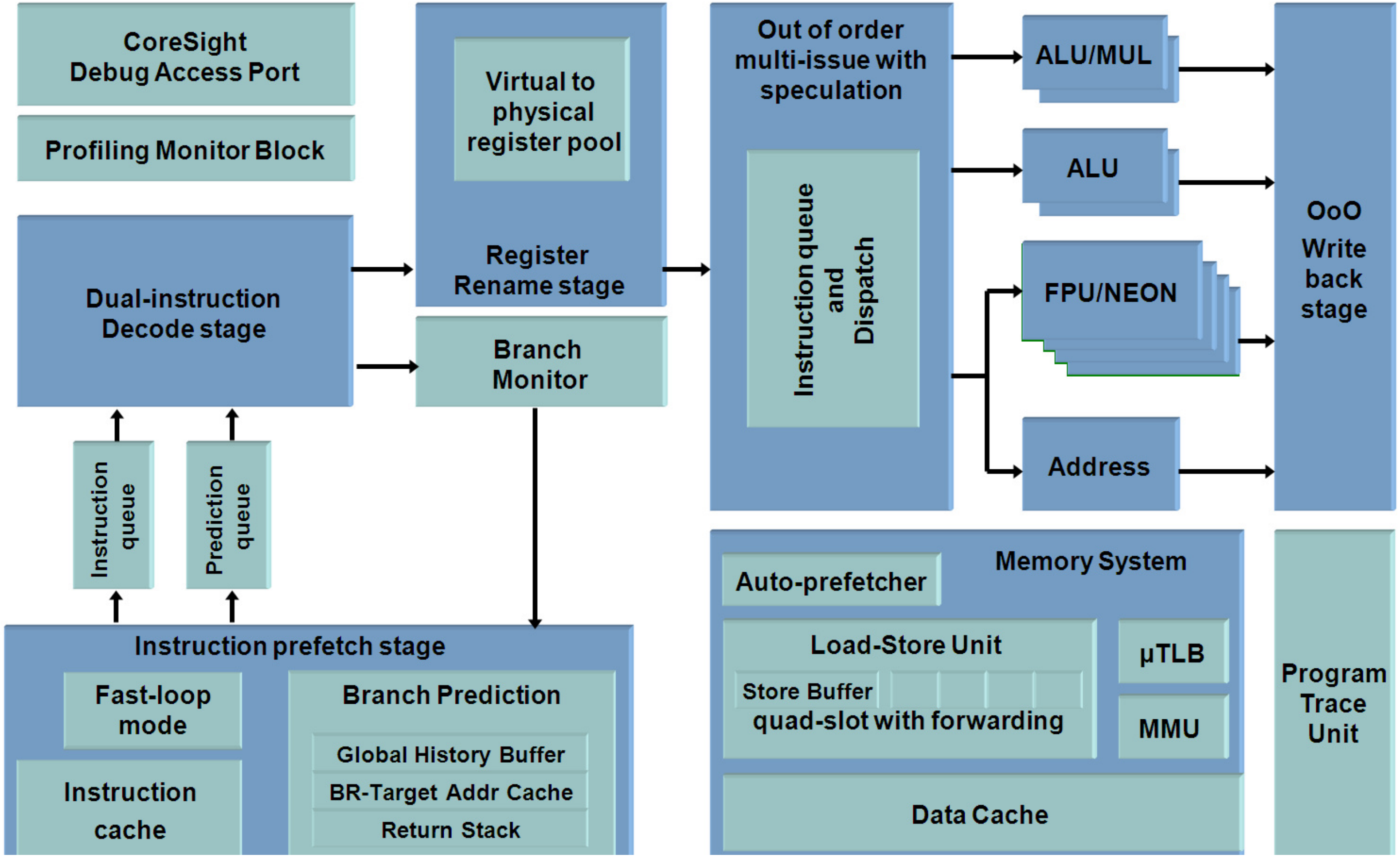
1st Stage - Fetch
2nd Stage - Decode
3rd Stage - Execute
AGU
Address Phase & Write Back
Data Phase Load/Store & Branch
Fetch (Prefetch)
Instruction Decode & Register Read
Multiply & Divide
Write
Branch
Shift
ALU & Branch
Branch forwarding & speculation
Execute stage branch (ALU branch & Load Store Branch)

The other end: Cortex-A9 (ARMv7-A)



Cortex-A9 Pipeline

- 8-stage, Out-of-Order, Multi-issue superscalar



Agenda

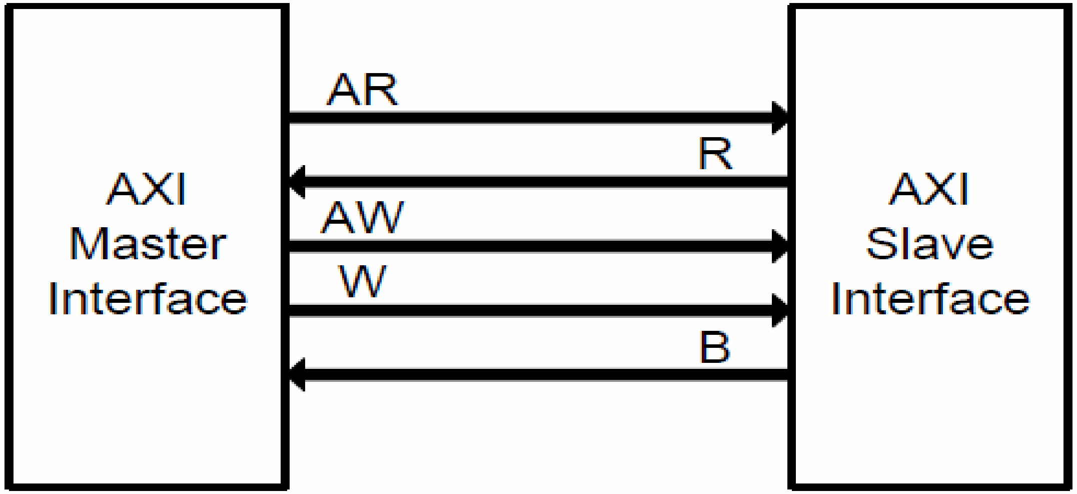
- Introduction to ARM Ltd
- ARM Architecture / Programmers Model
- Microarchitectures
- AMBA
- Example ARM based SoC

AMBA

- AMBA is an open standard owned and maintained by ARM
- AMBA 3 contains 4 main interface specifications:
  - APB – Advanced Peripheral Bus
  - AHB – Advanced High-performance Bus
  - AXI – Advanced eXtensible interface
  - ATB – Advanced Trace Bus
- Newest version: AMBA 4 since March 2010
  - Some modifications to AXI
  - Introducing
    - AXI Stream
    - AXI-Lite

AXI 3

- Original version of AXI, defined in AMBA 3
- Used by current IP
- AXI is a simple *point-to-point* link
- 5 independent channels
- Each channel is a set of signals going in the same direction

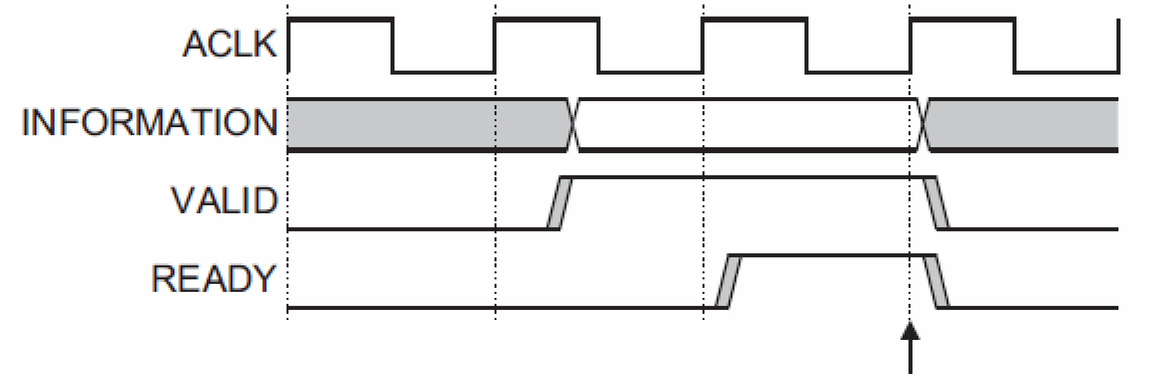


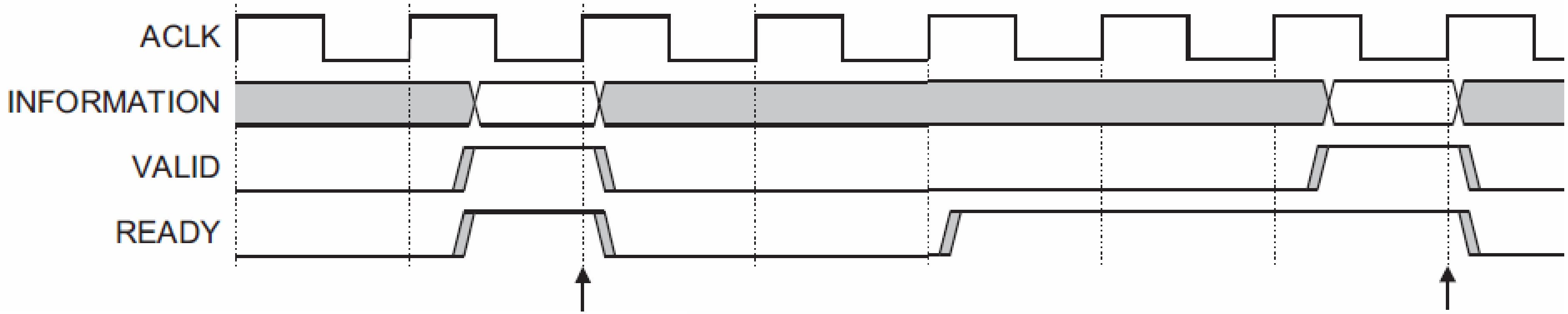
AXI channels

- The 5 channels are
  - AR: Read Address channel
  - R: Read Data channel
  - AW: Write Address channel
  - W: Write Data channel
  - B: Write Response channel
- Address channels carry control information and 32 bit addresses
- Data channels carry powers of 2 wide data (between 8-1024) together with some control signals
- Each channel contains
  - ID signals
  - VALID and READY handshake signals

AXI handshake

- The source generates the **VALID** signal to indicate when the information is available
- The destination generates the **READY** signal to indicate that it accepts the information
- Transfer occurs only when both the **VALID and READY** signals are HIGH.



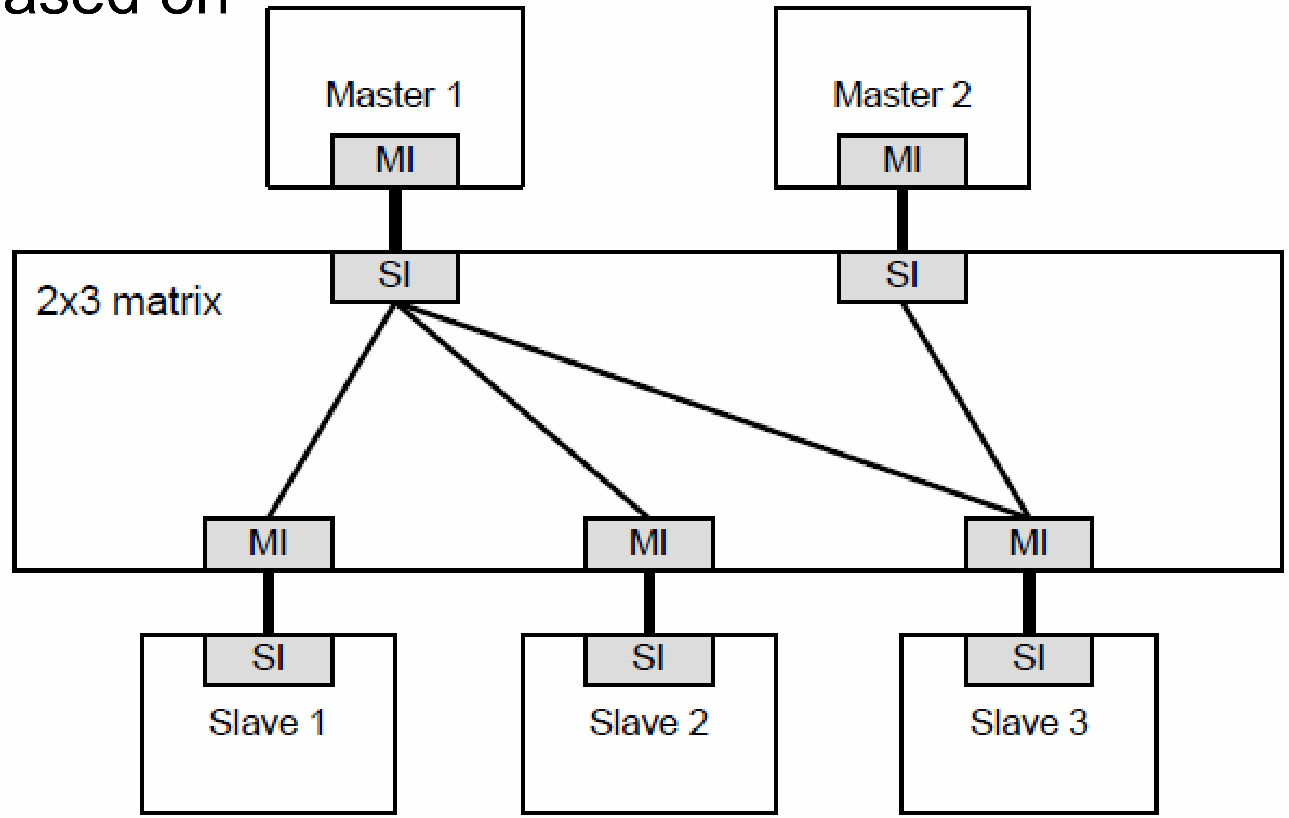


AXI transactions

- Parts of a transaction are identified by the transaction ID
- Read transaction
  - Master sends Read Address on AR channel
  - Slave sends Read Data and Response on R channel
- Write transaction
  - Master sends Write Address on AW channel
  - Master sends Write Data on W channel
  - Slave sends Write Response on B channel
- Transactions can have different (among others)
  - Burst length (1-16 beat)
  - Burst type (FIXED, INCR, WRAP)
  - Protection properties
- AXI supports
  - Multiple outstanding transactions
  - Out of order transaction completition

AXI based interconnects

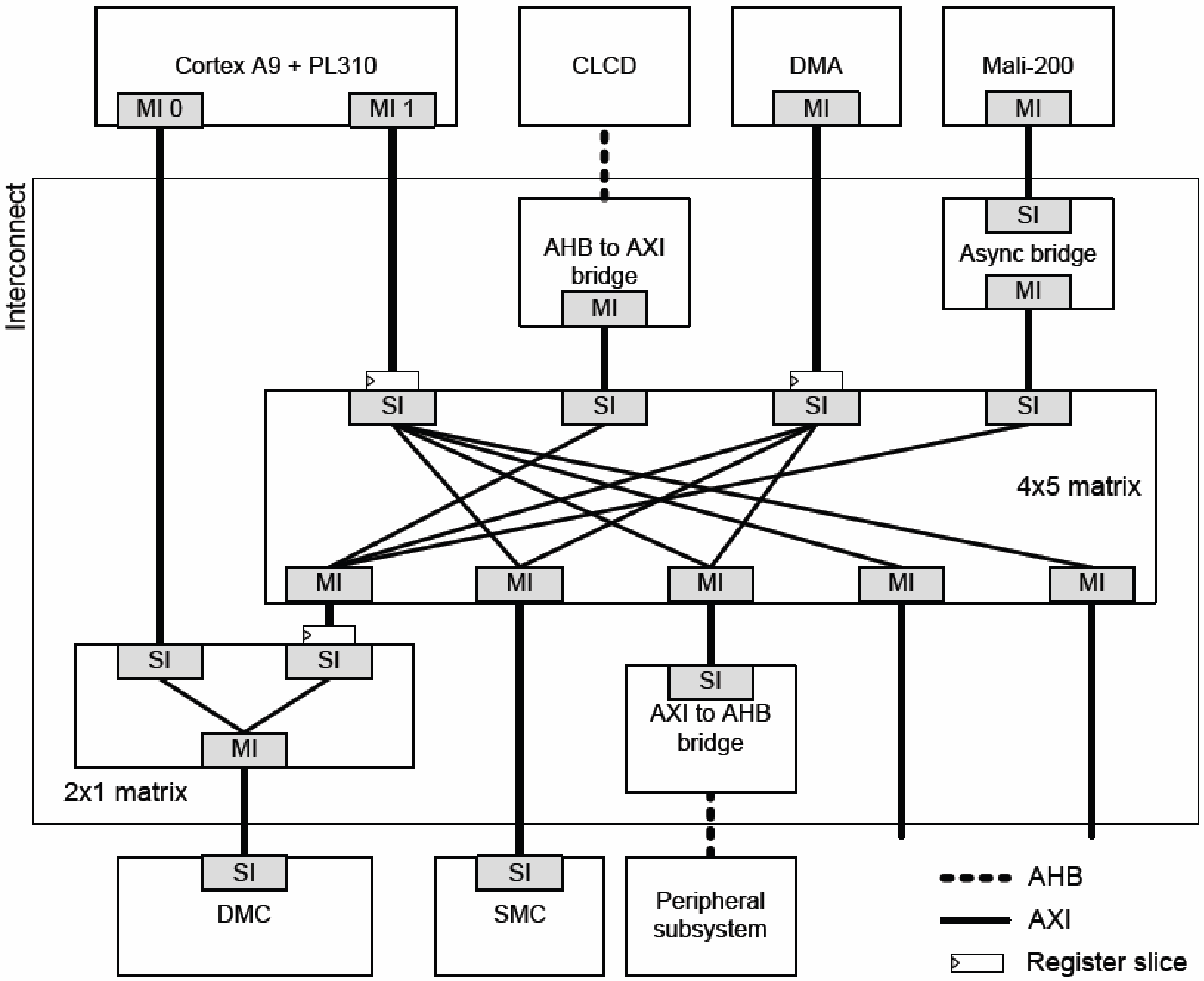
- AXI is a point-to-point link
- In practice, we build cross bar switches (or matrices) to route transactions between multiple masters and slaves
- Routing is done based on
  - Transaction Address
  - ID



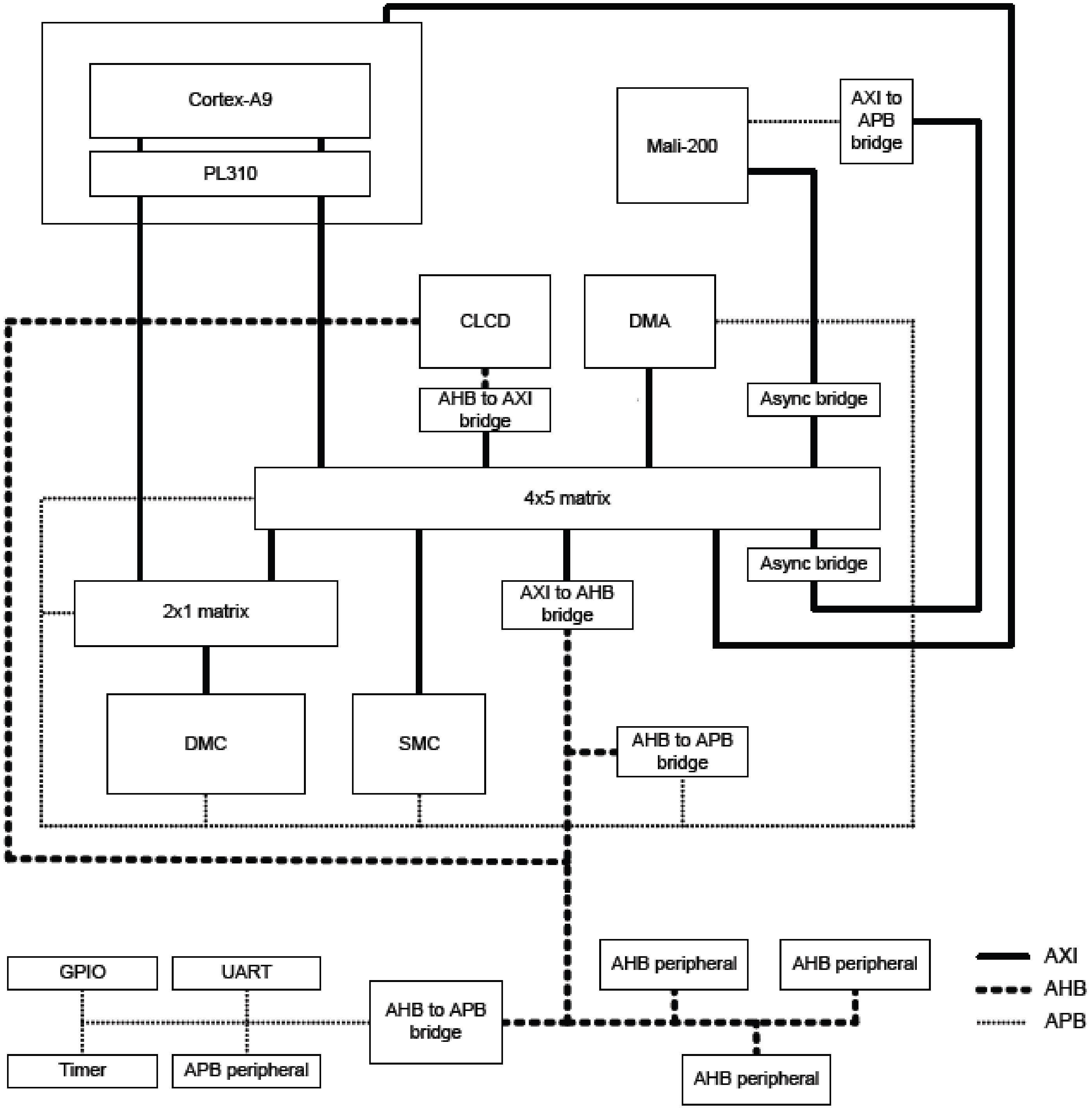
Agenda

- Introduction to ARM Ltd
- ARM Architecture / Programmers Model
- Microarchitectures
- AMBA
- Example ARM based SoC

Example ARM based system

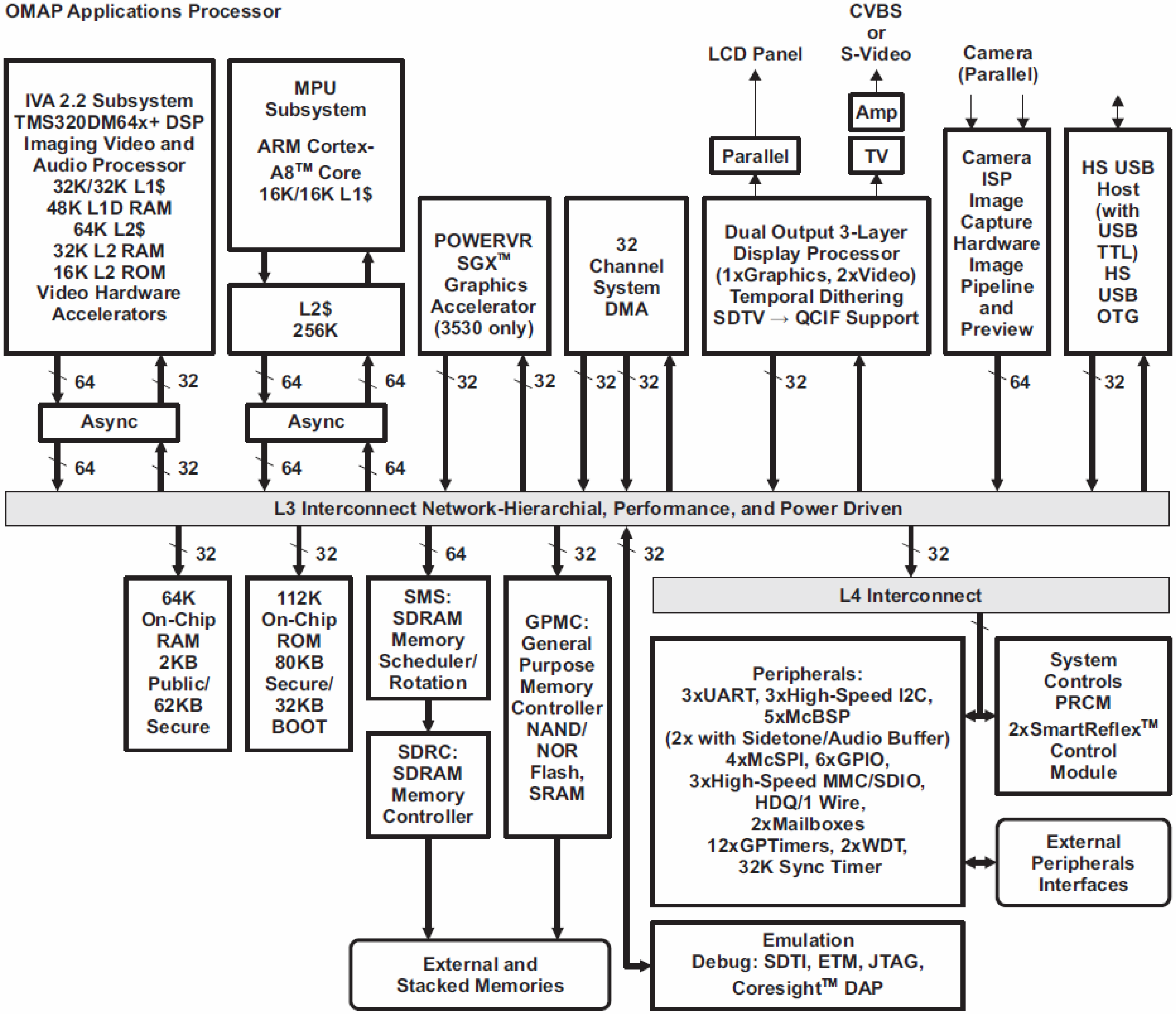


Example ARM based system 2



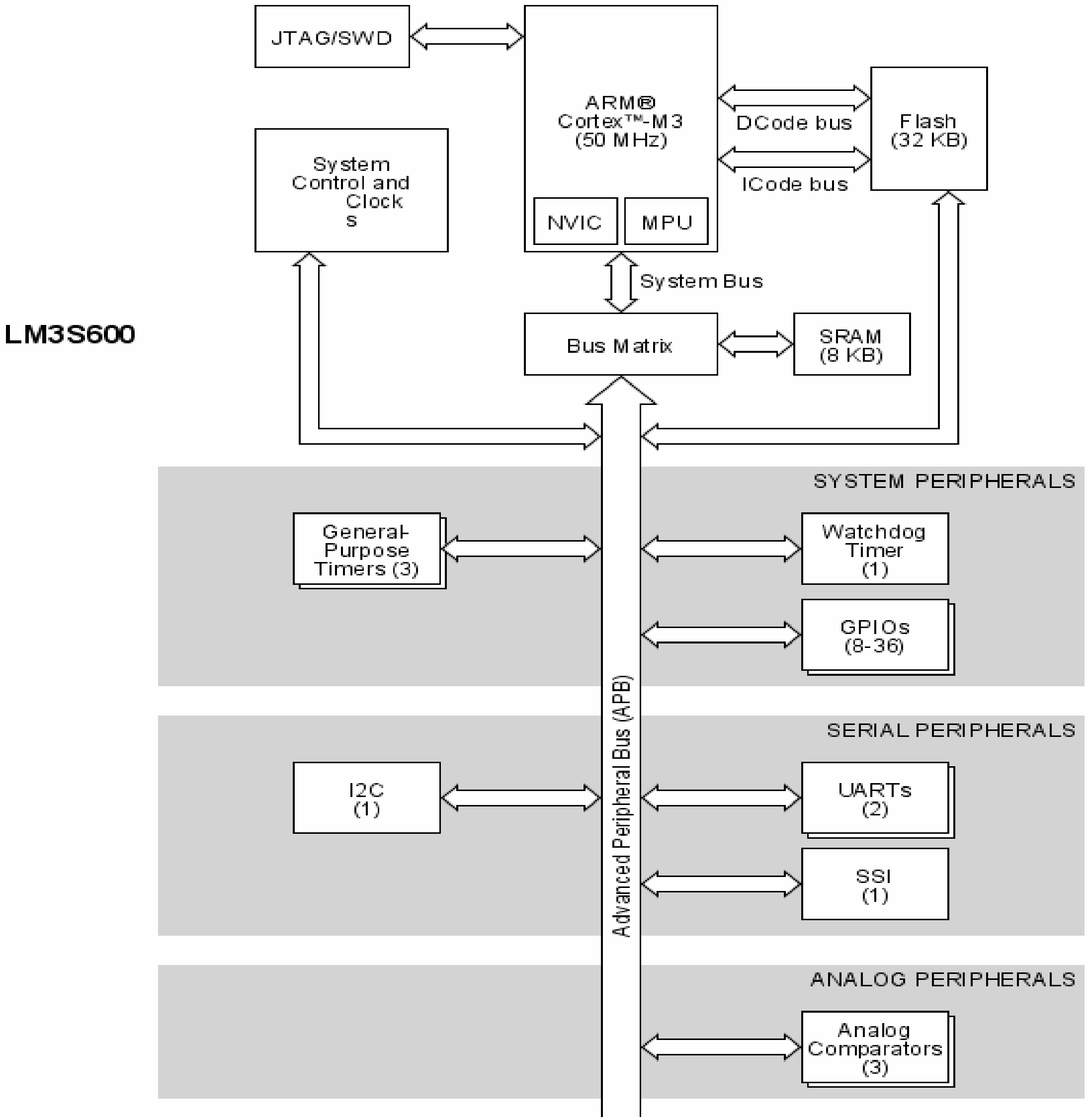
A real SoC: TI OMAP3530

- 720 MHz
  Cortex-A8
- $32
- Nokia N
  Series:
  OMAP2



Back to the low end

- Stellaris® LM3S600 Microcontroller
- 50 MHz Cortex-M3
- $2



Putting it all together: Nokia N95

OMAP™ 2420 Applications Processor

ARM1136™ processor-based SoC, developed using Magma ® Blast® family and winner of 2005 INSIGHT Award for 'Most Innovative SoC'

Symbian OS™ v9.2

Operating System supporting ARM processor-based mobile devices, developed using ARM® RealView® Compilation Tools

S60™ 3rd Edition

S60 Platform supporting ARM processor-based mobile devices

Mobiclip™ Video Codec

Software video codec for ARM processor-based mobile devices

ST WLAN Solution

Ultra-low power 802.11b/g WLAN chip with ARM9™ processor-based MAC

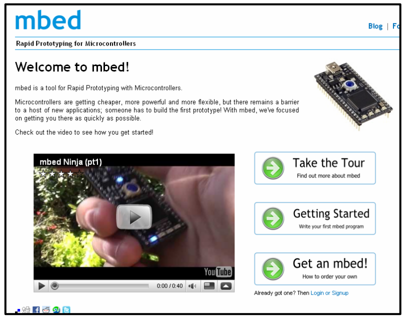
NOKIA
Connecting People

Connect. Collaborate. Create.

Conclusions

If you want to play ☺

- **Getting Started and Rapid Prototyping with ARM MCUs**
  - Complete Targeted Hardware, Software and Web 2.0 Platform



mbed

Blog | Fo

Rapid Prototyping for Microcontrollers

Welcome to mbed!

mbed is a tool for Rapid Prototyping with Microcontrollers.

Microcontrollers are getting cheaper, more powerful and more flexible, but there remains a barrier to a host of new applications; someone has to build the first prototype! With mbed, we've focused on getting you there as quickly as possible.

Check out the video to see how you get started!

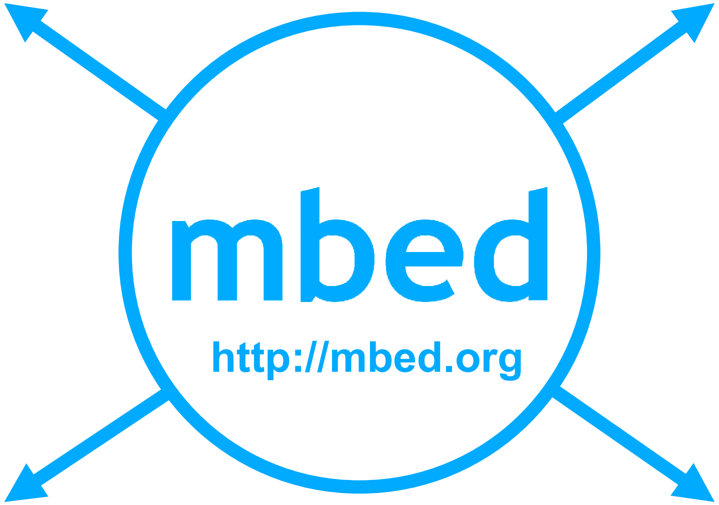
Take the Tour
Find out more about mbed

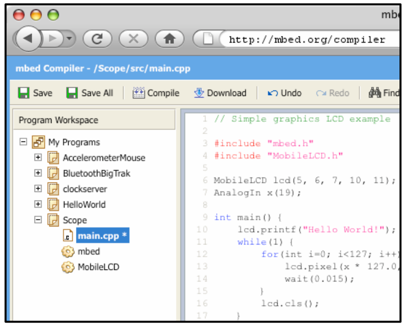
Getting Started
Write your first mbed program

Get an mbed!
How to order your own

Already got one? Then Login or Signup

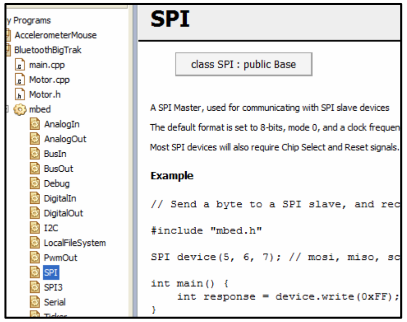
Dedicated Developer Web Platform





```
// Simple graphics LCD example

#include "mbed.h"
#include "MobileLCD.h"

MobileLCD lcd(5, 6, 7, 10, 11);
AnalogIn x(19);

int main() {
    lcd.printf("Hello World!");
    while(1) {
        for(int i=0; i<127; i++
            lcd.pixel(x * 127.0
            wait(0.015);
        }
        lcd.cls();
    }
```

Lightweight Online Compiler



SPI

class SPI : public Base

A SPI Master, used for communicating with SPI slave devices

The default format is set to 8-bits, mode 0, and a clock frequen

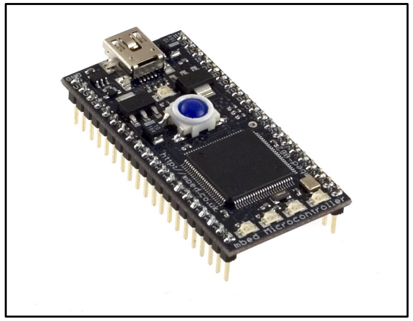
Most SPI devices will also require Chip Select and Reset signals.

Example

```
// Send a byte to a SPI slave, and rec

#include "mbed.h"

SPI device(5, 6, 7); // mosi, miso, sc

int main() {
    int response = device.write(0xFF);
}
```

High-level Peripheral APIs

Rapid Prototyping
for Microcontrollers



Cortex-M3 MCU in a Prototyping Form-Factor

Fin