

The ARM Architecture

Seng Lin Shee

20th May 2004

Outline

- Introduction / History
- ARM architecture
- Interesting ARM ISA Features
 - OS support
 - exceptions
- ISA Extensions
 - Thumb
 - Jazelle
 - DSP Instructions
- Architecture Implementations
 - ARM
 - Intel
 - Manchester University

History

- Started in 1983 – Acorn Ltd
- Build own machine – refused access to Intel 80286
- First sample, ARM1 (1985)
 - led by Roger Wilson and Steve Furber
 - based on 6502
- ARM2
 - 32-bit bus, 26-bit addressing
 - Simplest, 30K transistors compared to 68000 (Motorola)
 - no microcode, no cache
 - Low power, but better than 286 (Intel)
- Advanced RISC Machines
 - Working with Apple on newer versions of core
 - Spun off as separate company by 1990
- ARM6 created
 - Basis for Apple Newton PDA
 - Full 32-bit CPU

Introduction

- Leading provider of 32-bit embedded RISC microprocessors, 75% of market
 - Common architecture
 - High performance
 - Low power consumption
 - Low system cost
- Solutions for
 - Embedded real-time systems for mass storage, automotive, industrial and networking applications
 - Secure applications – smartcards and SIMs
 - Open platforms running complex operating systems

Licencing ARM Technology

- Implementation License
 - most popular
 - Complete information to design & manufacture integrated circuits containing ARM core
 - hard or soft core (macro cells)
 - Plan to be used in several products
- Foundry License
 - For fab-less semiconductor vendors to develop & sell ARM core-based products manufactured by licensed companies
- Architecture License
 - Develop own CPU implementations
- Academic License
 - Basic building blocks of the core to allow simulation and design of prototypes parts for academic research
 - Enables a core simulation environment to be created

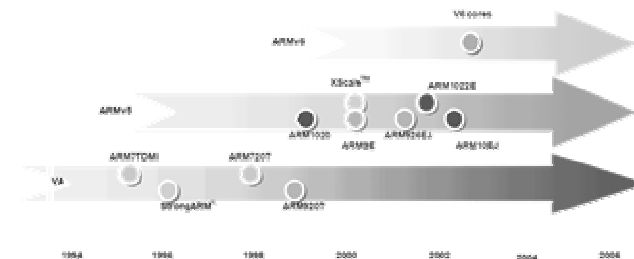
Architecture Definition

- The rules for how the microprocessor will behave, but without constraining or specifying how it will be built.
- Specification for the interface with the outside world
 - enabling operating system, application and development support to be planned and implemented.
- Instruction set (ISA)
 - concurrent ISA
 - media instructions
- Programmer's model
 - register sets etc
- Processor Interfaces
 - co processors
 - bus interfaces etc

ARM Architecture Variants

- ARMv1
 - First version of ARM processor
 - 26-bit addressing, no multiply / coprocessor
- ARMv2
 - ARM2, First commercial chip
 - Included 32-bit result multiply instructions / coprocessor support
- ARMv2a
 - ARM3 chip with on-chip cache
 - Added atomic load and store
 - Coprocessor 15 (cache management)
- ARMv3
 - ARM6, first processor after being independent
 - 32-bit addressing, separate CPSR, SPSR, virtual memory support

ARM Architecture



ARM Architecture Variant

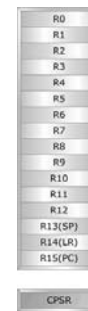
- ARMv4
 - added half word load and store.
- ARMv5
 - improved ARM and Thumb interworking, count leading zeroes (CLZ) instruction, and architecture variants:
 - E – enhanced DSP instructions including saturated arithmetic operations and 16 bit multiply operations
 - J – support for new Java state, offering hardware and optimized software acceleration of bytecode execution.
- ARMv6
 - Include 'TEJ' enhancements
 - Memory management, multiprocessing, SIMD instructions
 - 6 new status bits (GE[3:0], E, A bits)

ARM ISA Features

ARM Instruction Set Architecture

- Based on Berkeley RISC design
- Features used
 - Load-store architecture
 - Fixed-length 32-bit instructions
 - 3-address instruction formats
- Features rejected
 - Register windows
 - Delayed branches
 - Single-cycle execution of all instructions

Programmer's Model



- 32 bit RISC processor core (32 bit instructions)
- 37 pieces of 32 bit integer registers
- 17 visible registers
 - 15 general purpose
 - PC
 - CPSR – condition and mode bits
- Pipelined
- Cached (depending on implementation)
- Von Neumann / Harvard
- 8 / 16 / 32 bits data type
- 7 modes of operations (usr, fiq, irq, svc, abt, sys, und)

The ARM coprocessor interface

- Supports a general-purpose extension of its instruction set through the addition of hardware coprocessors
- Support for up to 16 logical coprocessors
 - 16 private registers of any width
 - Coprocessors use load-store architecture
 - Uses handshaking to perform instructions

Conditional Execution



- Every instruction has a 4 bit condition code
- Every code can be made conditional
- Extend conditional execution to all of its instructions, including supervisor calls and coprocessor instructions (excluding Thumb instructions)
- Each instruction mnemonic may be extended by appending two letters defined. (EQ, NE, GE, LT, GT etc)

Conditional Execution

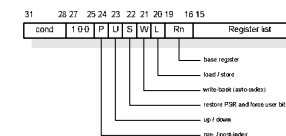
- Cuts down significantly on the space available for displacement memory access
- Avoid branch instructions when generating code for small if statements

```

int gcd(int i, int j)
{
    while (i != j) {
        if (i > j)
            i -= j;
        else
            j -= i;
    }
    return i;
}

b      test
loop  subgt  R1,R1,Rj
      suble Rj,Rj,Ri
      test  cmp   R1,Rj
      bne  loop
    
```

Multiple Register Transfer Operation



- Allow any subset (or all) of the 16 registers visible in the current operating mode to be loaded from or stored to memory
- Used on procedure entry and return to save and restore workspace registers and are useful for high-bandwidth memory block copy routines

Fold shifts / rotates into ALU operation

- The operand can be shifted before being processed and stored into a destination register
- Operations include arithmetic, logical, and register-register move
- $a += (j < 2)$ can be rendered as a single instruction on the ARM

Architectural Support for Operating Systems

- Coprocessor Number 15
 - On-die system control coprocessor
 - Controls the operation of the on-die cache, memory management or protection unit, write buffer, prefetch buffer, branch target cache and system configuration signals.
- MMU architecture
 - Translates virtual addresses into physical addresses.
 - Control memory access permission, aborting illegal accesses.
 - Uses a 2-level page table with table-walking hardware
 - A TLB which stores recently used page translations
 - All access made to the CP15 registers
 - 16 domains – each protected from one another while using the same TLB

Architectural Support for Operating Systems

- Synchronization
 - Require mutually exclusive access to a data structure.
 - Only one process can access this at any one time.
 - Must wait until no other process is accessing the data
 - Some sort of lock to prevent another process from accessing it until it has finished the operation.
 - ARM architecture supports synchronization by providing a 'SWAP' instruction.
 - Instruction is atomic
 - A 'test and set' instruction
 - A register is set to the 'busy' value and swapped with the memory location containing the Boolean.
 - If loaded value is 'free' the process can continue if 'busy' process must wait by on the lock

Architectural Support for Operating Systems

- Context switching
 - A process runs in a context
 - States includes the values of all the processor's registers, including the program counter, stack pointer etc
 - When a process switch takes place, the context of the old process must be saved and that of the new process restored.
 - 'architectural support' for register saving and restoring in privileged mode
 - special forms of the load and store multiple instructions
 - allow code running in a non-user mode to save and restore the user registers from an area of memory addressed by a non-user mode register

AMBA Interface

- De-facto standard for on-chip bus
- Open standard
- Framework for System-on-Chip designs
- Strategy for interconnection and management of functional blocks (SOC)
- One or more CPUs with multiple peripherals
- Maximum confidence in peripheral reuse
- IP Developers develop own products without worrying about connectivity

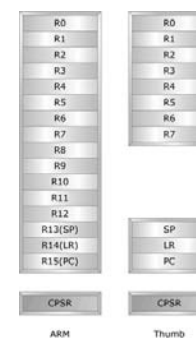
Architecture Overview

| Architecture | Thumb® | DSP | Jazelle | Media | TrustZone | Thumb-2 |
|--------------|--------|-----|---------|-------|-----------|---------|
| v4T | √ | | | | | |
| v5TE | √ | √ | | | | |
| v5TEJ | √ | √ | √ | | | |
| v6 | √ | √ | √ | √ | | |
| v6Z | √ | √ | √ | √ | √ | |
| v6T2 | √ | √ | √ | √ | | √ |

T: Thumb
 J: Jazelle
 E: DSP instructions
 T2: Thumb2

ARMv4

Thumb Instruction Set



- 16-bit instructions mapped to 32-bit ARM instructions
- Does not affect performance because expansion done via dedicated hardware within the chip
- Only 8 visible registers (LO registers)
- CPSR register determines mode of operation; switch mode by executing a Branch and Exchange instruction (BX)
- Thumb-ARM differences
 - Most Thumb instructions are executed unconditionally
 - Data processing instructions use a 2 address format
 - Instruction formats are less regular than ARM instruction formats
- Downside : overhead to switch from 32-bit to 16-bit unlike Xtensa Processors

ARMv5

ARM DSP Extensions

- broaden the suitability of the ARM CPU family to applications that require intensive signal processing
- retain the power and efficiency of a high-performance RISC microcontroller
- Features
 - Single-cycle 16x16 and 32x16 MAC implementations
 - Zero overhead saturation extension support
 - New instructions to load and store pairs of registers, with enhanced addressing modes
 - New CLZ instruction improves normalization in arithmetic operations and improves divide performance
 - Full support in the ARMv5TE and ARMv6 architecture
- Applications
 - Audio encode/decode (MP3: AAC, WMA)
 - MPEG4 decode
 - Voice and handwriting recognition
 - Embedded control
 - Bit exact algorithms (GSM-AMR)

Jazelle

- Adding a third instruction set to the ISA
 - Java Byte Code
 - Additional instruction for supporting Jazelle
- Architectural extensions to execute Java Byte Code directly
- Reuse all existing processor resources without the need to re-engineer existing architecture or add cost, power or memory resources
 - J bit set in CPSR
 - All processor state related to Java execution are stored in normal ARM register set.
 - Any interrupt routine which saves on entry and restores on exit are compatible with Jazelle
- Hardware logic contribute to only 12K gates

ARMv6

Key features of ARMv6

- Media processing extensions
 - 2x faster MPEG4 encode/decode
 - 2x faster audio DSP
- Improved cache architecture
 - Physically addressed caches
 - Reduction in cache flush/refill
 - Reduced overhead in context switches
- Improved exception and interrupt handling
 - Important for improving performance in real time tasks
- Unaligned and mixed-endian data support
 - Simpler data sharing, application porting and saves memory

Programmer's Model

- Six new status bits have been added to the programmer's model
 - GE[3:0] bits
 - SIMD status bits- greater than or equal to for each 8/16 bit slice
 - E-bit
 - Indicates the current load/store endian setting of the core can be set/cleared with the SETEND instruction
 - A-bit
 - Indicates if imprecise data abort exceptions are masked

Media Instructions

- Enables more efficient software implementation of high-performance media applications.
- Over 60 SIMD instructions
- Uses the GE-bits added to the programmer's model
- Support four 8-bit and two 16-bit operations, parallel add and subtract, selection, packing and unpacking
- Supports dual 16-bit multiply add/subtract
- Expected performance between 2x and 4x

Media Instructions

| |
|--|
| ARMv5TE: 5 cycles in a single-cycle implementation |
| SMULTT Real,Ra,Rb ;Real = Ra.real*Rb.real |
| SMULBB Temp,Ra,Rb ;Temp = Ra.imag*Rb.imag |
| SUB Real,Real,Temp ;Real = Ra.real*Rb.real - Ra.imag*Rb.imag |
| SMULTB Imag,Ra,Rb ;Imag = Ra.real*Rb.imag |
| SMLABT Imag,Ra,Rb ;Imag = Ra.real*Rb.imag + Ra.imag*Rb.real |
| ARMv6: 2 cycles in a single-cycle implementation |
| SMUSD Real,Ra,Rb ;Real = Ra.real*Rb.real - Ra.imag*Rb.imag |
| SMUADX Imag,Ra,Rb ;Imag = Ra.real*Rb.imag + Ra.imag*Rb.real |

Example 16-bit Complex Multiply

| Architecture | Cycles/4 pixels |
|--------------|-----------------|
| ARMv5TE | 18 cycles |
| ARMv6 | 3 cycles |

Implementing Sum of Absolute Differences

Thumb 2

- although a single Thumb instruction is equivalent to a single ARM instruction, more 16-bit Thumb instructions are needed to accomplish the same overall function.
- combination of ARM and Thumb code within an application
 - balance the cost, performance and power characteristics of the system.
- ARM Thumb 2 core technology
 - New 16-bit thumb instruction for improve program flow
 - New 32-bit thumb instruction derived from ARM instruction equivalent
 - Co-processor access, privilege instructions, special functions (SIMD)
 - ARM 32-bit ISA has been improved
- Performance
 - Performance similar to instruction based on ARM ISA (98%)
 - 5 percent smaller than Thumb high density code
 - 2-3 percent faster than Thumb high density code

Architecture Implementations

ARM Processor Cores

| Core | Architecture |
|--|--------------|
| ARM1 | v1 |
| ARM2 | v2 |
| ARM2aS, ARM3 | v2a |
| ARM6, ARM600, ARM610, AMULET1, AMULET2 | v3 |
| ARM7, ARM700, ARM710 | v3 |
| ARM7TDMI, ARM710T, ARM720T, ARM740T | v4T |
| Strong ARM, ARM8, ARM810 | v4 |
| ARM9TDMI, ARM920T, ARM940T, AMULET3 | v4T |
| ARM9E-S | v5TE |
| ARM10TDMI, ARM1020E, XScale | v5TE |
| ARM11 | v6 |

ARM7TDMI

- Evolved from first ARM core to implement the 32-bit address space, ARM6
- A 3 volt compatible reword of the ARM6 32-bit integer core
 - Thumb 16-bit compressed instruction set
 - On-chip Debug support
 - An enhanced Multiplier
 - EmbeddedICE hardware (breakpoint and watchpoint)
- Using a 3 stage pipeline
 - Implements Arm architecture version 4T
 - Fetch – instruction is fetched from memory and placed in instruction pipeline
 - Decode – instruction is decoded and datapath control signal signals prepared
 - Execute – register bank is read, operand shifted, ALU result generated and written back in a destination register

ARM7TDMI

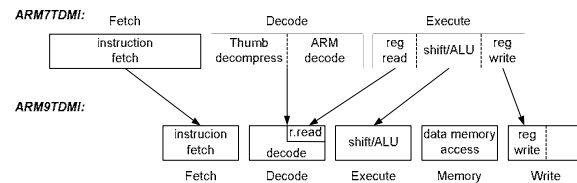
- Two read ports and one write port
- One additional read port and an additional write port that give special access to r15, the program counter
- Interfaces
 - Memory interface
 - MMU interface
 - Coprocessor interface
 - Debug interface
 - JTAG interface
- ARM7TDMI characteristics

| | | | | | |
|--------------|---------|-------------|---------------------|--------|-------|
| Process | 0.35 um | Transistors | 74,209 | MIPS | 60 |
| Metal layers | 3 | Core area | 2.1 mm ² | Power | 87 mW |
| Vdd | 3.3 V | Clock | 0-66 MHz | MIPS/W | 690 |

ARM9TDMI

- Adopts a 5-stage pipeline to increase the maximum clock rate
- Use a separate instruction and data memory ports to allow an improved CPI
- Owes a lot to the StrongARM pipeline
 - StrongArm has a dedicated branch adder which operates in parallel with the register read stage
 - ARM9TDMI uses the main ALU (gives additional clock cycle penalty for a taken branch; smaller / simpler core; avoid critical timing path)
 - StrongARM designed for a particular process technology
 - ARM9TDMI is readily portable to new processes
- Thumb instruction decoding
 - Uses hardware to decode both ARM and Thumb instructions directly
- Static branch prediction
- Can also connect to single unified memory but doing so requires a complex high-speed memory subsystem.
- ARM9E-S is a synthesizable version of the ARM9TDMI core (30% larger than the ARM9TDMI)

ARM9TDMI

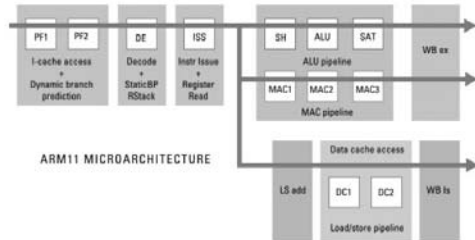


| | | | | | |
|--------------|---------|-------------|---------------------|--------|--------|
| Process | 0.25 um | Transistors | 111,000 | MIPS | 220 |
| Metal layers | 3 | Core area | 2.1 mm ² | Power | 150 mW |
| Vdd | 2.5 V | Clock | 0-200 MHz | MIPS/W | 1500 |

ARM11

- First implementation of the ARMv6 architecture
- Target applications:
 - Wireless
 - Consumer
 - Networking
 - Automotive
- Supports 4 K cache sizes
- Developed for both synthesizable and semi-custom hard macrocell implementations
- ARM11 cores have synthesis friendly pipeline structure, designed to work with commercially available synthesis tools
- Features
 - Thumb – code compression
 - 'E' – DSP processing
 - Jazelle – Java acceleration

ARM11



- 8 pipeline stages (40% higher throughput)
- Extensive forwarding
- Branch prediction
 - Dynamic branch predictor (64-entry, 4 state branch target address cache) – Strongly taken, Weakly taken, Strongly not taken, Weakly not taken
 - Static prediction

ARM11

- Pipeline parallelism
 - Separate datapaths for the ALU, multiply-accumulate (MAC) and Load-Store (LS) instructions.
- Improved memory access
 - Non-blocking, hit-under-miss operation of memory system
 - Pipeline stalls only when 3 successive misses occurs
- 64-bits datapaths
 - 64-bit data buses between the processor integer unit and the instruction and data caches, and between coprocessors and the integer unit.
 - Fetch two instructions in a single cycle
 - Load- and store-multiple instructions every cycle
 - 64-bit effective performance, but at a 32-bit cost

| | | | |
|---------|---------|-----------|---------------------|
| Process | 0.13 um | | |
| | | Core area | 2.7 mm ² |
| | | Power | 150 mW |
| Vdd | 1.2 V | Clock | 0-500 MHz |
| | | | mW/MHz |
| | | | 0.4 |

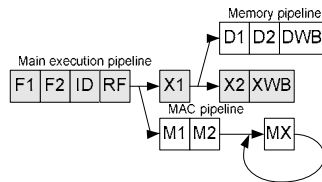
XScale

- ARM Architecture Implementations
 - Fully implements the ARMS-TE instruction set
 - Thumb
 - DSP instructions (but without floating point)
- 7-stage integer pipe (vs 5 in StrongARM)
- 32 KB data and 32 KB instruction caches
- Write buffer (8 entry)
- 128-entry branch target buffer (helps predict branches and reduce pipeline stalls)
- 2 KB mini-cache (data) which can support a reading number of data streams without thrashing the data cache
- Full MMU (i.e. supports Unix)
- "hit-under-miss" feature allows execution to continue even when a cache miss is being processed
- Debug unit for use with Multi-ICE (v2.2 or later)
 - this was missing from StrongARM

XScale

- Extensions to ARM architecture
 - A DSP coprocessor (CP0) has been added that contains a 40 bit accumulator and 8 new operations in coprocessor space, hereafter referred to as new instructions.
 - New page attributes were added to the page table descriptors. The C and B page attribute encoding was extended by one more bit to allow for more encodings: write allocate and mini data cache.
 - Additional functionality has been added to coprocessor 15. CP15 configures the MMU, caches, buffers and other system attributes.
 - Coprocessor 14 was created. CP14 contains the performance monitor registers and the trace buffer registers
 - Enhancements were made to the Event Architecture, instruction cache and data cache parity error exceptions, breakpoint events, and imprecise external data aborts.

XScale



- Longer pipeline has disadvantages
 - Longer branch prediction penalty (4 cycles to 1 cycle-StrongArm)
 - Large load use delay – need optimizing compiler to fill in the gap
 - LDM, STM incur a few extra clock cycles delay
 - Decode and register file lookup take up 2 clock cycles

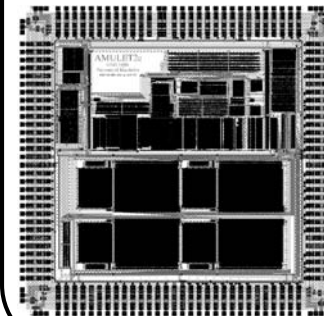
Comparisons

| Feature | ARM9E™ | ARM10E™ | Intel® XScale™ | ARM11™ |
|-----------------------------|------------------|------------------|------------------|------------------------------|
| Architecture | ARMv5TE(J) | ARMv5TE(J) | ARMv5TE | ARMv6 |
| Pipeline Length | 5 | 6 | 7 | 8 |
| Java Decode | (ARM926EJ) | (ARM1026EJ) | No | Yes |
| V6 SIMD Instructions | No | No | No | Yes |
| MIA Instructions | No | No | Yes | Available as coprocessor |
| Branch Prediction | No | Static | Dynamic | Dynamic |
| Independent Load-Store Unit | No | Yes | Yes | Yes |
| Instruction Issue | Scalar, in-order | Scalar, in-order | Scalar, in-order | Scalar, in-order |
| Concurrency | None | ALU/MAC, LSU | ALU, MAC, LSU | ALU/MAC, LSU |
| Out-of-order completion | No | Yes | Yes | Yes |
| Target Implementation | Synthesizable | Synthesizable | Custom chip | Synthesizable and Hard macro |
| Performance Range | Up to 250MHz | Up to 325MHz | 200MHz – >1GHz | 350MHz – >1GHz |

AMULET

- Fully asynchronous circuit
- Implementation of ARM processor using Micropipeline design style
- Demonstrate that an asynchronous microprocessor can offer a reduction in electrical power consumption over a synchronous design in the same role
- Developed in the University of Manchester, England
- Motivation:
 - No clock skew problems
 - Asynchronous design only causes transitions in the circuit in response to a request to carry out useful work
 - Less electromagnetic radiation due to less coherent internal activity
 - Deliver typical rather than worst-case performance since its timing adjusts to actual conditions rather than worst-case conditions (clocked)

AMULET2e



- Four phase micropipeline control circuits
- Power saving latch designs
- Load and register forwarding
- Branch target prediction
- "Sleep" mode
- Self timed cache
- Dynamic external bus sizing
- Direct interface to commodity memory devices
- 0.5µm, 3 layer metal process
- Core size 5mm x 5mm; die size 6.5mm x 6.5mm

