ARM Low-power Processors and Architectures

Dan Millett
Verification Enablement
Processor Division
Agenda

- Introduction to ARM Ltd
- ARM Architecture/Programmers Model
- Data Path and Pipelines
- System Design
- Development Tools
ARM Ltd

- Founded in November 1990
  - Spun out of Acorn Computers
  - Initial funding from Apple, Acorn and VLSI

- Designs the ARM range of RISC processor cores
  - Licenses ARM core designs to semiconductor partners who fabricate and sell to their customers
  - ARM does not fabricate silicon itself

- Also develop technologies to assist with the design-in of the ARM architecture
  - Software tools, boards, debug hardware
  - Application software
  - Bus architectures
  - Peripherals, etc
ARM’s Activities

Connected Community
Development Tools
Software IP

Processors
System Level IP:
Data Engines
Fabric
3D Graphics

Physical IP
ARM Connected Community – 700+
Huge Range of Applications

- Intelligent toys
- Utility Meters
- IR Fire Detector
- Exercise Machines
- Energy Efficient Appliances
- Tele-parking
- Intelligent Vending
- Equipment Adopting 32-bit ARM Microcontrollers
How many ARM’s Do You Have?

- Mobile phones: ~100% market share
- Smartphones: 3x 100% market share
- Mobile Computers: 5x 100% market share
- Digital TVs: 35% market share
- Disk Drives: ~75% market share
- PC Peripherals: 40% market share
- Cars: 5x 50% market share
- Microcontrollers: 35% market share
Huge Opportunity For ARM Technology

25+ billion cores to date

100+ billion cores accumulated after next 9 yrs

1998  2011  2020
World’s Smallest ARM Computer?

Wireless Sensor Network

- Sensors, timers
- Cortex-M0 +16KB RAM 65nm UWB Radio antenna
- 10 kB Storage memory ~3fW/bit
- 12µAh Li-ion Battery

Battery

Solar Cells

Wirelessly networked into large scale sensor arrays

Processor, SRAM and PMU

Cortex-M0; 65¢

University of Michigan
World’s Largest ARM Computer?

4200 ARM powered Neutrino Detectors

70 bore holes 2.5km deep
60 detectors per string starting 1.5km down
1km³ of active telescope

Work supported by the National Science Foundation and University of Wisconsin-Madison
From $1\text{mm}^3$ to $1\text{km}^3$
Agenda

Introduction to ARM Ltd

- ARM Architecture/Programmers Model
- Data Path and Pipelines
- System Design
- Development Tools
ARM Cortex Advanced Processors

- **ARM Cortex-A family:**
  - Applications processors
  - Targeted for OS’s, graphics, demanding tasks

- **ARM Cortex-R family:**
  - Embedded processors
  - Real-time signal processing, control applications

- **ARM Cortex-M family:**
  - Microcontroller-oriented processors
  - MCU, ASSP, and SoC applications
Relative Performance*

*Represents attainable speeds in 130, 90, 65, or 45nm processes

<table>
<thead>
<tr>
<th></th>
<th>Cortex-M0</th>
<th>Cortex-M3</th>
<th>ARM7</th>
<th>ARM926</th>
<th>ARM1026</th>
<th>ARM1136</th>
<th>ARM1176</th>
<th>Cortex-A8</th>
<th>Cortex-A9 Dual-core</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Freq (MHz)</td>
<td>50</td>
<td>150</td>
<td>184</td>
<td>470</td>
<td>540</td>
<td>610</td>
<td>750</td>
<td>1100</td>
<td>2000</td>
</tr>
<tr>
<td>Min Power (mW/MHz)</td>
<td>0.012</td>
<td>0.06</td>
<td>0.35</td>
<td>0.235</td>
<td>0.36</td>
<td>0.335</td>
<td>0.568</td>
<td>0.43</td>
<td>0.5</td>
</tr>
</tbody>
</table>

* Represents attainable speeds in 130, 90, 65, or 45nm processes.
Cortex family

**Cortex-A8**
- Architecture v7A
- MMU
- AXI
- VFP & NEON support

**Cortex-R4**
- Architecture v7R
- MPU (optional)
- AXI
- Dual Issue

**Cortex-M3**
- Architecture v7M
- MPU (optional)
- AHB Lite & APB
Data Sizes and Instruction Sets

- The ARM is a 32-bit architecture.

- When used in relation to the ARM:
  - **Byte** means 8 bits
  - **Halfword** means 16 bits (two bytes)
  - **Word** means 32 bits (four bytes)

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

- Jazelle cores can also execute Java bytecode
ARM and Thumb Performance

Dhrystone 2.1/sec @ 20MHz

Memory width (zero wait state)
The Thumb-2 instruction set

- Variable-length instructions
  - ARM instructions are a fixed length of 32 bits
  - Thumb instructions are a fixed length of 16 bits
  - Thumb-2 instructions can be either 16-bit or 32-bit

- Thumb-2 gives approximately 26% improvement in code density over ARM

- Thumb-2 gives approximately 25% improvement in performance over Thumb
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

Banked out Registers

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

The Architecture for the Digital World®
# Program Status Registers

- **Condition code flags**
  - N = Negative result from ALU
  - Z = Zero result from ALU
  - C = ALU operation Carried out
  - V = ALU operation Overflowed

- **Sticky Overflow flag - Q flag**
  - Architecture 5TE/J only
  - Indicates if saturation has occurred

- **J bit**
  - Architecture 5TEJ only
  - J = 1: Processor in Jazelle state

- **Interrupt Disable bits.**
  - I = 1: Disables the IRQ.
  - F = 1: Disables the FIQ.

- **T Bit**
  - Architecture xT only
  - T = 0: Processor in ARM state
  - T = 1: Processor in Thumb state

- **Mode bits**
  - Specify the processor mode

---

<table>
<thead>
<tr>
<th>31</th>
<th>28</th>
<th>27</th>
<th>24</th>
<th>23</th>
<th>16</th>
<th>15</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>Z</td>
<td>C</td>
<td>V</td>
<td>Q</td>
<td>J</td>
<td>Undefined</td>
<td>I</td>
<td>F</td>
<td>T</td>
<td>mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>s</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

The Architecture for the Digital World®
Conditional Execution and Flags

- ARM instructions can be made to execute conditionally by postfixing them with the appropriate condition code field.
  - This improves code density and performance by reducing the number of forward branch instructions.

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

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

- By default, data processing instructions do not affect the condition code flags but the flags can be optionally set by using “S”. CMP does not need “S”.

```
loop
...
SUBS  r1,r1,#1
BNE   loop
```

- decrement r1 and set flags
- if Z flag clear then branch
Classes of Instructions

- **Load/Store**
- **Miscellaneous**
- **Data Operations**
- **Change of Flow**
  - MOV PC, Rm
  - Bcc
  - BL
  - BLX
Branch instructions

- Branch: \( B\{<\text{cond}>\} \) label
- Branch with Link: \( BL\{<\text{cond}>\} \) subroutine_label

- The processor core shifts the offset field left by 2 positions, sign-extends it and adds it to the PC
  - ± 32 Mbyte range
  - How to perform longer branches?
Data processing Instructions

- Consist of:
  - Arithmetic: ADD, ADC, SUB, SBC, RSB, RSC
  - Logical: AND, ORR, EOR, BIC
  - Comparisons: CMP, CMN, TST, TEQ
  - Data movement: MOV, MVN

- These instructions only work on registers, NOT memory.

- Syntax:

  `<Operation>{<cond>}{S} Rd, Rn, Operand2`

- Comparisons set flags only - they do not specify Rd
- Data movement does not specify Rn
- Second operand is sent to the ALU via barrel shifter.
Register, optionally with shift operation

- Shift value can be either be:
  - 5 bit unsigned integer
  - Specified in bottom byte of another register.
- Used for multiplication by constant

Immediate value

- 8 bit number, with a range of 0-255.
  - Rotated right through even number of positions
- Allows increased range of 32-bit constants to be loaded directly into registers
Data Processing Instruction Examples

- MOV r3, r0 ; copies r0 into r3
- MVN r6, r8 ; copies the complement of r8 into r6
- ADD r0, r1, r2 ; r0 = r1 + r2
- ADC r0, r1, r2 ; r0 = r1 + r2 + <carry flag>
- SUB r3, r1, r7 ; r3 = r1 - r7
- RSB r3, r1, r7 ; r3 = r7 - r1
- SBC r3, r1, r7 ; r3 = r1 - (r7 + <carry flag>)
- AND r0, r1, #0xA5 ; r0 = r1 & 0xA5
- BIC r0, r1, #0xA5 ; r0 = r1 with bits 0,2,5,and 7 cleared
- ORR r0, r1, #0xA5 ; r0 = r1 with bits 0,2,5,and 7 set
- CMP r5, r9 ; same as SUBS, but only affects APSR
- CMN r0, r1 ; same as ADDS, but only affects APSR
- TST r0, r1 ; same as ANDS, but only affects APSR
- TEQ r0, r1 ; same as EORS, but only affects APSR
Single / Double Register Data Transfer

- Use to move data between one or two registers and memory
  - LDRD, STRD: Doubleword
  - LDR, STR: Word
  - LDRB, STRB: Byte
  - LDRH, STRH: Halfword
  - LDRSB, LDRSH: Signed byte load, signed halfword load

- Syntax
  - LDR{<size>}{<cond>} Rd, <address>
  - STR{<size>}{<cond>} Rd, <address>

Diagram:
- Memory
- Any remaining space zero filled or sign extended
- Rd
Agenda

Introduction to ARM Ltd
ARM Architecture/Programmers Model
  - Data Path and Pipelines
System Design
Development Tools
The ARM7TDM Core

- **Multiplier**
- **Barrel Shifter**
- **32 Bit ALU**
- **Address Register**
- **Register Bank**
- **Address Incrementer**
- **Decode Stage**
- **Instruction Decoder**
- **Control Logic**

**Inputs:**
- ABE
- A[31:0]
- PC
- MCLK
- nWAIT
- nRW
- MAS[1:0]
- ISYNC
- nIRQ
- nFIQ
- nRESET
- ABORT
- nTRANS
- nMREQ
- SEQ
- LOCK
- nM[4:0]
- nOPC
- nCPI
- CPA
- CPB

**Outputs:**
- DBE
- D[31:0]
Cortex-M3 Datapath

Instruction Decode

Address Incrementer

Address Register

Register Bank

Mul/Div

Barrel Shifter

ALU

Write Data Register

Read Data Register

D_HWDATA

D_HRDATA

I_HRDATA

D_HADDR

I_HADDR

INTADDR

Writeback
Pipeline changes for ARM9TDMI

ARM7TDMI

FETCH

Instruction Fetch

DECODE

Thumb→ARM decompress
ARM decode
Reg Select

EXECUTE

Reg Read
Shift
ALU
Reg Write

ARM9TDMI

FETCH

Instruction Fetch

DECODE

ARM or Thumb Inst Decode
Reg Decode
Reg Read

EXECUTE

Shift + ALU

MEMORY

Memory Access

WRITE

Reg Write
Cortex-M3 Pipeline

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

![Cortex-M3 Pipeline Diagram](image-url)
ARM10 vs. ARM11 Pipelines

**ARM10**

- **Branch Prediction**
- **Instruction Fetch**
- **ARM or Thumb Instruction Decode**
- **Reg Read**
- **Shift + ALU**
- **Memory Access**
- **Reg Write**

**ARM11**

- **Fetch 1**
- **Fetch 2**
- **Decode**
- **Issue**
- **MAC 1**
- **MAC 2**
- **MAC 3**
- **Address**
- **Data Cache 1**
- **Data Cache 2**
- **Shift**
- **ALU**
- **Saturate**
- **Write back**
Full Cortex-A8 Pipeline Diagram

13-Stage Integer Pipeline

10-Stage NEON Pipeline

Branch mispredict penalty
Replay penalty

Instruction Fetch

Instruction Decode

Instruction Execute and Load/Store

Architectural register file

ALU pipe 0

MUL pipe 0

ALU pipe 1

LS pipe 0 or 1

NEON 
Load queue

NEON Instruction Decode

NEON register file

NEON register writeback

Integer ALU pipe

Integer MUL pipe

Integer shift pipe

Non-IEEE FP ADD pipe

Non-IEEE FP MUL pipe

IEEE FP engine

LS permute pipe

Embedded Trace Macrocell

T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13

External trace port
Agenda

- Introduction to ARM Ltd
- ARM Architecture/Programmers Model
- Data Path and Pipelines
  - System Design
- Development Tools
TI OMAP35X SoC
Agenda

Introduction to ARM Ltd
ARM Architecture/Programmers Model
Data Path and Pipelines
System Design

- Development Tools
Development Platforms
Keil Development Tools for ARM

- Includes ARM macro assembler, compilers (ARM RealView C/C++ Compiler, Keil CARM Compiler, or GNU compiler), ARM linker, Keil uVision Debugger and Keil uVision IDE

- Keil uVision Debugger accurately simulates on-chip peripherals (I²C, CAN, UART, SPI, Interrupts, I/O Ports, A/D and D/A converters, PWM, etc.)

- Evaluation Limitations
  - 16K byte object code + 16K data limitation
  - Some linker restrictions such as base addresses for code/ constants
  - GNU tools provided are not restricted in any way

- http://www.keil.com/demo/
Keil Development Tools for ARM
University Resources

- www.arm.com/university/
- University@arm.com
Your Future at ARM…

- **Graduate and Internship/Co-op Opportunities**
  - Engineering: Memory, Validation, Performance, DFT, R&D, GPU and more!
  - Sales and Marketing: Corporate and Technical
  - Corporate: IT, Patents, Services (Training and Support), and Human Resources

- **Incredible Culture and Comprehensive Benefit Package**
  - Competitive Reward
  - Work/Life Balance
  - Personal Development
  - Brilliant Minds and Innovative Solutions

- **Keep in Touch!**
  - www.arm.com/about/careers
TI Panda Board

OMAP4430 Processor
- 1 GHz Dual-core ARM Cortex-A9 (NEON+VFP)
- C64x+ DSP
- PowerVR SGX 3D GPU
- 1080p Video Support

POP Memory
- 1 GB LPDDR2 RAM

USB Powered
- < 4W max consumption (OMAP small % of that)
- Many adapter options (Car, wall, battery, solar, ..)
Fin
Nokia N95 Multimedia Computer

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

Connect. Collaborate. Create.
Targeting community development

- $149
- Personally affordable
- Wikis, blogs, promotion of community activity
- Freedom to innovate
- Instant access to >10 million lines of code
- Free software
- > 1000 participants and growing
- Active & technical community
- Open access to hardware documentation
- Opportunity to tinker and learn
- Open source community needs
- Targeting community development
Fast, low power, flexible expansion

OMAP3530 Processor
- 600MHz Cortex-A8
- NEON+VFPv3
- 16KB/16KB L1$
- 256KB L2$
- 430MHz C64x+ DSP
- 32K/32K L1$
- 48K L1D
- 32K L2
- PowerVR SGX GPU
- 64K on-chip RAM

POP Memory
- 128MB LPDDR RAM
- 256MB NAND flash

Peripheral I/O
- DVI-D video out
- SD/MMC+
- S-Video out
- USB 2.0 HS OTG
- I$^2$C, I$^2$S, SPI, MMC/SD
- JTAG
- Stereo in/out
- Alternate power
- RS-232 serial

USB Powered
- 2W maximum consumption
- OMAP is small % of that
- Many adapter options
- Car, wall, battery, solar, ...

3"
Peripheral I/O
- DVI-D video out
- SD/MMC+
- S-Video out
- USB HS OTG
- I²C, I²S, SPI,
  MMC/SD
- JTAG
- Stereo in/out
- Alternate power
- RS-232 serial

Other Features
- 4 LEDs
  - USR0
  - USR1
  - PMU_STAT
  - PWR
- 2 buttons
  - USER
  - RESET
- 4 boot sources
  - SD/MMC
  - NAND flash
  - USB
  - Serial

On-going collaboration at BeagleBoard.org
- Live chat via IRC for 24/7 community support
- Links to software projects to download

And more…

On-going collaboration at BeagleBoard.org
- Live chat via IRC for 24/7 community support
- Links to software projects to download

And more…
Project Ideas Using Beagle

- **OS Projects**
  - OS porting to ARM/Cortex (TI OMAP)
  - MythTV system
  - “Super-Beagle” – stack of Beagles as compute engine and task distribution
  - Linux applications

- **NEON Optimization Projects**
  - Codec optimization in ffmpeg (pick your favorite codec)
  - Voice and image recognition
  - Open-source Flash player optimizations (swfdec)