ARM7TDMI vs. ARM Cortex-M3 Microcontroller Cores

Paul Roukema
Department of Electrical and Computer Engineering
University of Waterloo
2A Candidate for B.A.Sc. in Computer Engineering
March 22, 2010

Copyright © 2009 by Paul Roukema. All rights reserved.
ARM7TDMI vs. ARM Cortex-M3 Microcontroller Cores

Outline

• In this presentation we will discuss
  » Microcontrollers
  » ARM, architectures vs. implementations
    • ARM7TDMI
    • ARM Cortex-M3
  » Comparison
    • Initialization
    • Interrupt Handling
    • Performance
    • Availability
  » Conclusions
ARM7TDMI vs. ARM Cortex-M3 Microcontroller Cores

Microcontrollers

- Integrates
  - Processor
  - Memory
    - Volatile
    - Non-Volatile
  - Peripherals
    - Serial (UARTs)
    - ADCs
    - Timers
    - Ethernet (sometimes)
  - Oscillator Block
    - Internal
    - External
ARM7TDMI vs. ARM Cortex-M3 Microcontroller Cores

ARM, Architectures vs. Implementations

- ARM Inc. produces
  - Architecture Specifications
  - Implementations of Architectures
  - General IP (UARTS, I2C Controllers, Bus Controllers)

- Architectures ≈ Instruction Sets
  - ARM v4T
  - ARM v7-{M|R|A}

- Implementations
  - ARM7 (This is an ARM v4T implementation)
  - Cortex-M{0|1|3} (ARM v7-M)
  - Cortex-A{5|8|9} (ARM v7-A)
  - xScale (Intel/Marvell)
ARM7TDMI vs. ARM Cortex-M3 Microcontroller Cores

ARM7TDMI

- Introduced in 1994
- Implements ARM v4T Architecture
  - Thumb mode for reduced code size
- Von-Neumann Architecture
- Widely used
  - Nintendo DS (ARM7 + ARM9)
  - iPod
  - Many Others
- Microcontrollers available from many suppliers
  - Atmel, Analog Devices, NXP, STMicroelectronics ...
- Wide software support
ARM7TDMI vs. ARM Cortex-M3 Microcontroller Cores

**ARM Cortex-M3**

- Introduced in 2005
- Implements ARM v7-M architecture
  - Thumb 2, mixed 16/32 bit instructions
- Harvard Architecture
- Branch Prediction
- Few publicized users in commercial products
ARM7TDMI vs. ARM Cortex-M3 Microcontroller Cores

Initialization

• ARM7TDMI *requires* assembly code for initialization
  » Makes writing your own startup code more difficult

• Cortex-M3 allows the use of either C/C++ or Assembly
  » Simpler code to write and debug
  » Easier to understand

<table>
<thead>
<tr>
<th></th>
<th>ARM7</th>
<th>Cortex-M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begins Execution</td>
<td>0x00000000</td>
<td>*(0x00000000)</td>
</tr>
<tr>
<td>Stack Pointer</td>
<td>Undefined</td>
<td>*(0x00000004)</td>
</tr>
<tr>
<td>Language</td>
<td>Assembly</td>
<td>Assembly or C</td>
</tr>
</tbody>
</table>
Interrupt Handling

• Only considering core interrupt handling
  » The usual case for Cortex-M3
  » Usually have a secondary interrupt controller for ARM7
    • Adds latency

<table>
<thead>
<tr>
<th></th>
<th>ARM7</th>
<th>Cortex-M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt Types</td>
<td>FIQ, IRQ</td>
<td>Vectored</td>
</tr>
<tr>
<td># Interrupts</td>
<td>2</td>
<td>Up to 240</td>
</tr>
<tr>
<td>Priorities</td>
<td>2</td>
<td>255</td>
</tr>
<tr>
<td>State saved</td>
<td>FIQ only</td>
<td>YES</td>
</tr>
<tr>
<td>Min Latency</td>
<td>5 Cycles</td>
<td>12 Cycles</td>
</tr>
<tr>
<td>Max Latency</td>
<td>29 Cycles</td>
<td>12 Cycles</td>
</tr>
</tbody>
</table>
Interrupt Handling (cont’d)

• Cortex M3 appears to have slower best-case interrupt latency
  » Have to factor in state saving, interrupt routing
  » As an example, Atmel SAM7S series ARM7 MCUs
    • Features an Advanced Interrupt Controller peripheral
    • Provides 8 priorities, nesting, vectoring
    • Adds 3 - 4.5 cycles latency
    • Still need to save some registers
  » Best case, SAM7S could be faster
    • Requires correct instruction timing
    • Requires reserved interrupt registers or handler in assembly
  » Cortex-M3 provides better guarantee in general case
ARM7TDMI vs. ARM Cortex-M3 Microcontroller Cores

Performance

• Clock speed
  » Instruction Timings
  » Pipeline Stalls

• Power
  » Often the limiting factor

• ARM specific tradeoff
  » ARM vs. Thumb vs. Thumb2 instruction set
    • ARM – 32 bit wide instructions, full set of capabilities
    • Thumb – 16 bit wide instructions, less flexible
    • Thumb2 – 16/32 bit mixed instructions, best of both worlds
### ARM7TDMI vs. ARM Cortex-M3 Microcontroller Cores

#### Performance (cont’d)

<table>
<thead>
<tr>
<th></th>
<th>ARM7</th>
<th>Cortex-M3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Frequency</td>
<td>85 MHz</td>
<td>100 MHz</td>
</tr>
<tr>
<td>DMIPS/MHz</td>
<td>0.94</td>
<td>1.25</td>
</tr>
<tr>
<td>Max DMIPS</td>
<td>80</td>
<td>125</td>
</tr>
<tr>
<td>DMIP/mW</td>
<td>3.36</td>
<td>3.75</td>
</tr>
<tr>
<td>Area (mm²)</td>
<td>0.62</td>
<td>0.37</td>
</tr>
</tbody>
</table>

- **Speed optimized 180nm Process**
- **Important consideration not represented**
  - ARM7 can only reach full performance running ARM code
  - Requires more code space, faster instruction bus
  - Thumb mode ARM7 gives max. 63 DMIPS
  - Cortex-M3 is exclusively Thumb2
### ARM7TDMI vs. ARM Cortex-M3 Microcontroller Cores

#### Availability

<table>
<thead>
<tr>
<th>Vendors</th>
<th>ARM7</th>
<th>Cortex-M3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Atmel, Analog Devices, NXP, STMicro, Freescale, Texas Instruments, Samsung</td>
<td>Atmel, NXP, STMicro, Texas Instruments</td>
</tr>
</tbody>
</table>

- Market availability for Cortex-M3 has improved significantly over the last year
- Some vendors targeting low cost markets, 8/16 bit pricing
- ARM7 still has an edge in specialized applications
- Software ecosystem still maturing on Cortex-M3 front
  - Compilers OK, RTOS support could be better
Conclusions

- Cortex-M3 microcontrollers represent an easier target from a software perspective
- Improved interrupt handling
- Higher peak performance
- Better power consumption
- ARM7 benefits from wider availability and larger support ecosystem
- Largest benefit in small, low cost situations replacing dsPIC, HC12 or even PIC and AVR
- One set of development tools and rules over a wider range of applications
References