ARM CortexM3 – Programmers view and Development Environment

Hardware Software CoDesign

November 2011

Agenda

ARM CortexM3 – Programmers view and Development Environment

1. Basic information on ARM CortexM3
2. Programmers view of CortexM3 (refer as M3)
3. Discussion on the Answering Machine Assignment (Group wise 1, 2, 3, 4)
4. Mid Semester Paper distribution

ARM CortexM3 – Programmers view

Introduction to ARM CortexM3

1. Primarily designed to target the 32bit Microcontoller market

2. Great performance at low cost and many new features available only in high-end processors

3. Enhanced determinism, guaranteeing that critical tasks and interrupts are serviced as quickly as possible, but in a "known" number of cycles.

4. Improve code density, ensuring that code fits even the smallest memory footprints

5. Ease of use, providing debugability and easy programmability for those applications which are migrating from 8, 16bit to 32bit.

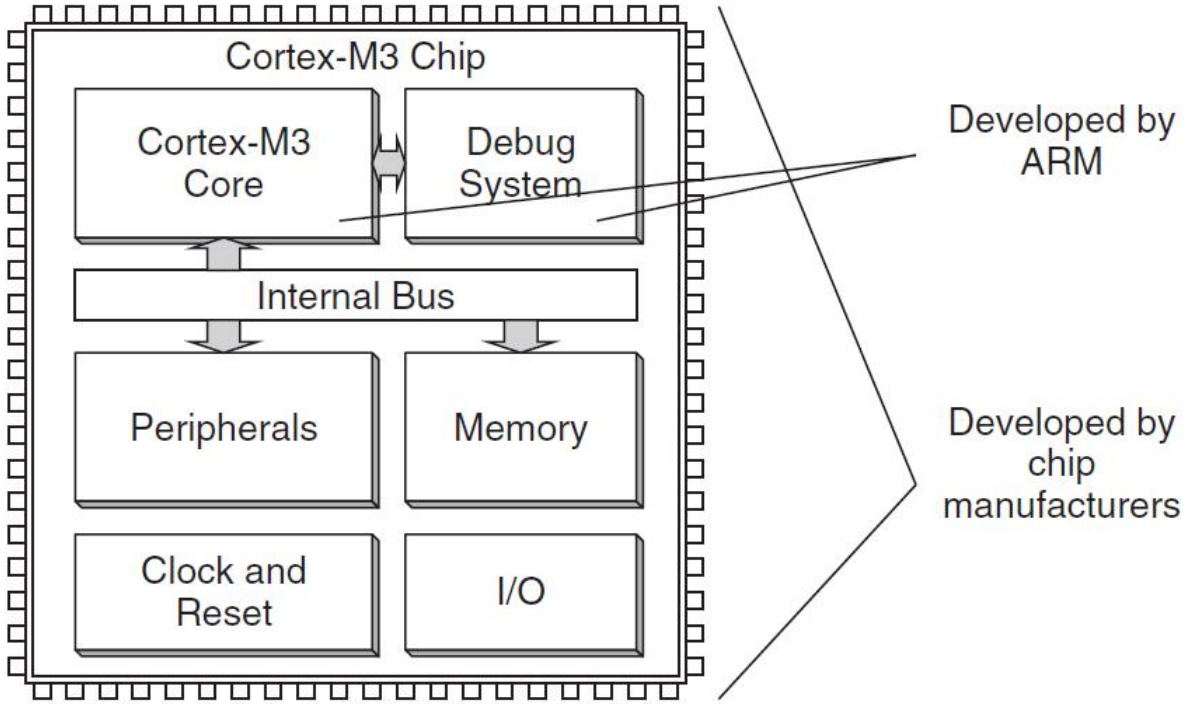
6. Can be used in "device aggregation", where multiple traditional 8bit devices can get replaced by a single 32bit high performance device.

7. Through the compilers, the amount of code reuse across other ARM systems can take place.

ARM CortexM3 – Programmers view

Introduction to ARM CortexM3

1. The use model for ARM and other integrators is :-



2. In general, ARM has come up with Cortex family like :-
   - **A family** :: designed for high-performance *application* platforms
   - **R family** :: designed for high-end embedded systems in which *Real-Time* performance is needed
   - **M family** :: designed for deeply embedded *Microcontroller* systems

ARM CortexM3 – Programmers view

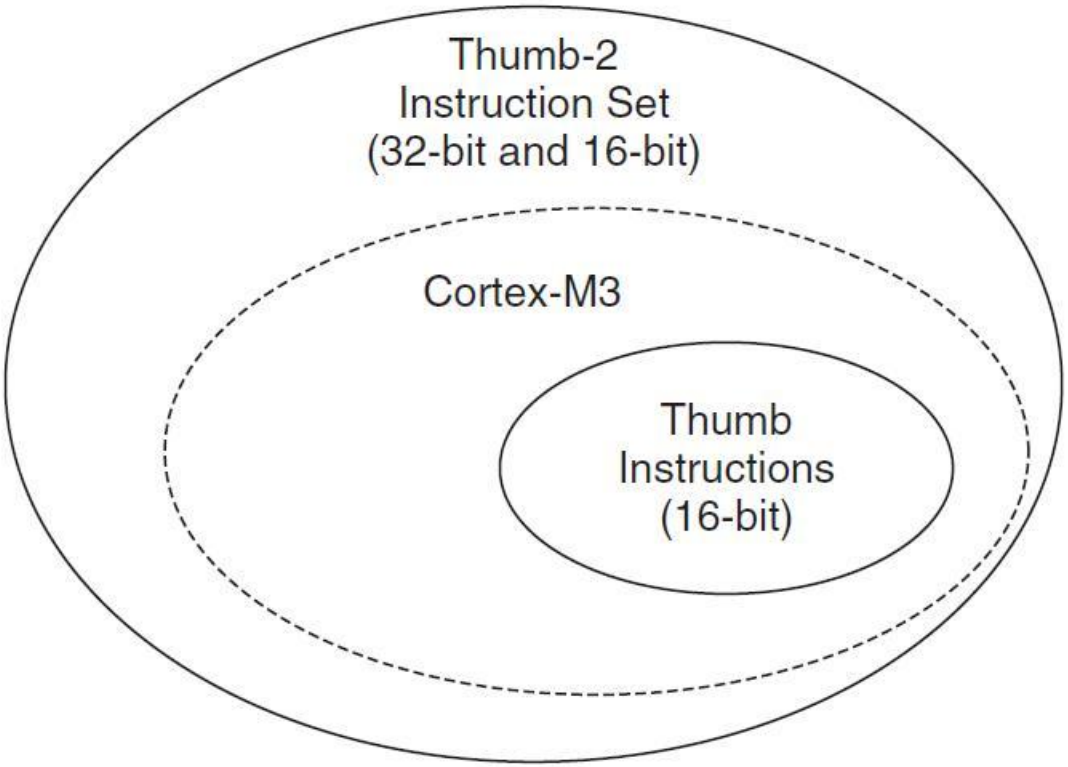
Introduction to ARM CortexM3 – ARM and Thumb Instruction Sets

1. There are two different types of instruction sets
   (a) 32bit called *ARM* instruction set
   (b) 16bit called *Thumb* instruction set

2. During program execution, the processor can be dynamically switched between the ARM state or Thumb state to use either of the instruction sets

3. The Thumb instruction set provides only a subset of the ARM instructions, but can provide high code density.

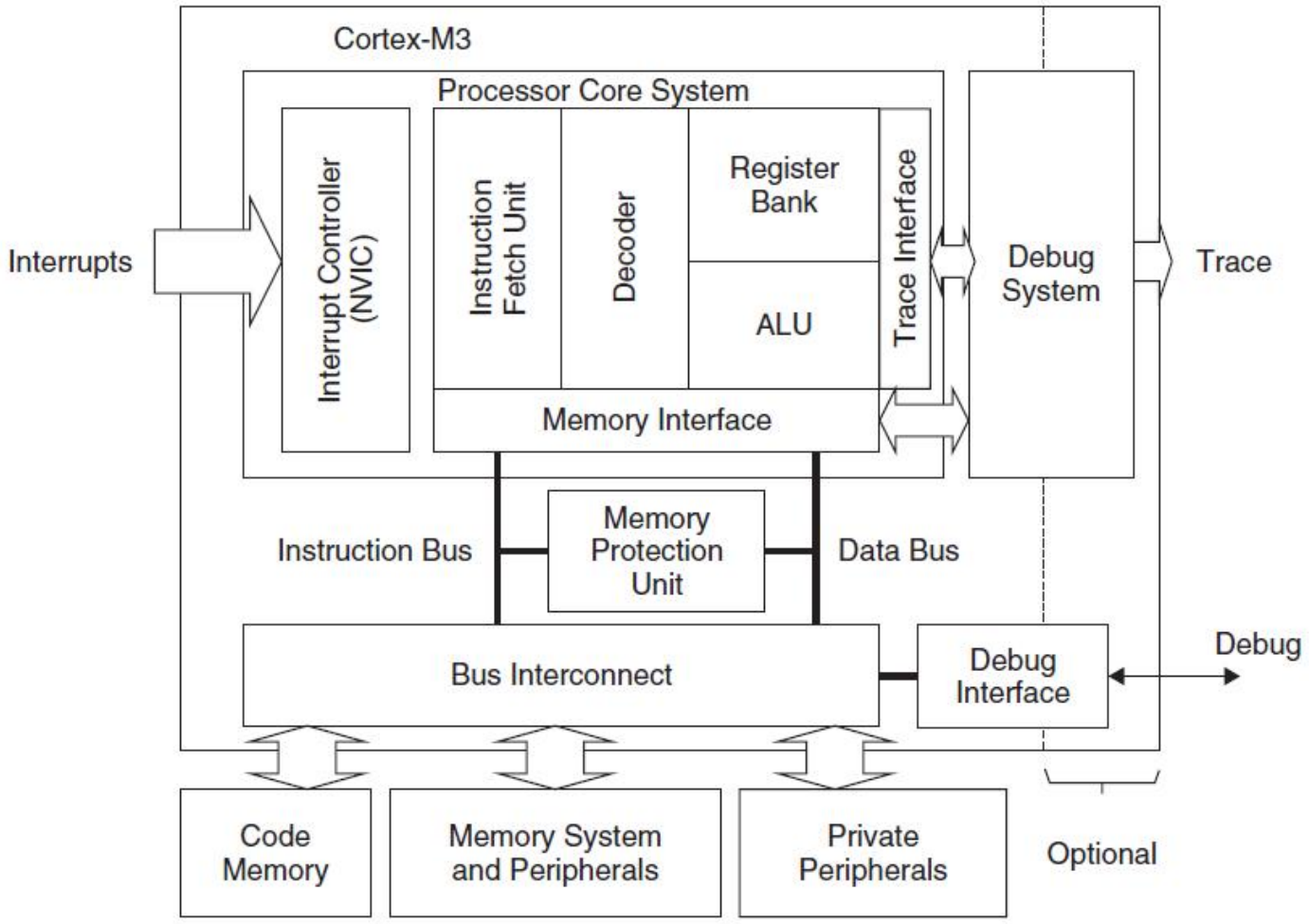
4. M3 processor supports only the Thumb-2 (and traditional Thumb) instruction set. It uses Thumb-2 instruction set for all operations



The Relationship Between the Thumb-2 Instruction Set and the Thumb Instruction Set

ARM CortexM3 – Programmers view

Introduction to ARM CortexM3 – Basic Block Diagram



A Simplified View of the Cortex-M3

1. The processor has a Harvard architecture, ie., has a separate instruction bus and data bus. However, the instruction and data buses share the same memory space (unified memory system). i.e, Cannot get 8GB space just because there are separate bus interfaces.

2. *GROUP DISCUSSION* :: How does Harvard architecture help M3 ?

ARM CortexM3 – Programmers view

Introduction to ARM CortexM3 – General Purpose Register Set

1. The M3 has GP registers *R0 to R15*

2. *General Purpose Registers* :: R0 to R12

3. *Stack Pointer* :: R13 $\rightarrow$ banked R13 register

   (a) *Main Stack Pointer MSP* - Default StackPointer, used by the OS kernel and exception handlers

   (b) *Process Stack Pointer PSP* - Used by the user application code

4. *Link Register* :: R14 $\rightarrow$ When a subroutine is called, the return address is stored in the link register.

5. *Program Counter* :: R15 $\rightarrow$ The current program address. This register can be written to control the program flow

ARM CortexM3 – Programmers view

Introduction to ARM CortexM3 – Operation Modes

1. The M3 has 2 Operation modes and 2 privilege levels

2. *Operation Modes* :: What kind of operation the M3 is doing. Whether the processor is running a normal program or running an exception handler like an interrupt handler or system exception handler.

   (a) *Thread Mode* :: Thread mode is entered on reset, and can be entered as a result of an exception return. Privileged and User code can run in Thread mode.

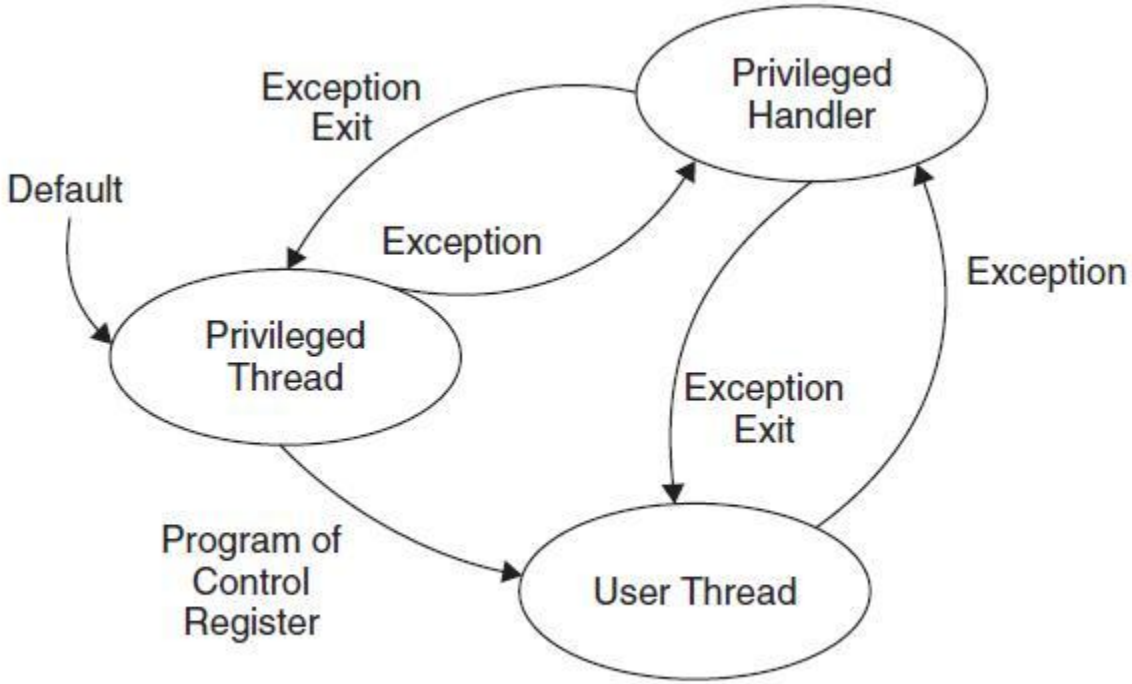
   (b) *Handler Mode* :: Handler mode is entered as a result of an exception. All code is privileged in Handler mode.

3. *Privilege Levels* :: provide a mechanizm for safeguarding memory acces to critical regions as well as provide a basic security model.

   (a) *Privilege Level* ::

   (b) *User Level* ::

	Privileged	*User*
When running an exception	Handle Mode	
When running main program	Thread Mode	Thread Mode



Allowed Operation Mode Transitions

ARM CortexM3 – Programmers view

Introduction to ARM CortexM3 – BuiltIn Nested Vectored Interrupt Controller

1. The M3 Processor includes an Interrupt Controller called the *NVIC (Nested Vectored Interrupt Controller* with following main features :-

   (a) *Nested Interrupt Support* :: All interrupts and most of the system exceptions can be programmed to different priority levels. When an interrupt occurs, the NVIC compares the priority of this interrupt to the current running priority level. If the priority of the new interrupt is higher than the current level, the interrupt handler of the new interrupt will override the current running task.

   (b) *Vectored Interrupt Support* :: When an interrupt is accepted, the starting address of the ISR is located from a vector table in memory. There is no need to use software to determine and branch to the starting address of the ISR. Thus it takes less time to process the interrupt request.

   (c) *Dynamic Priority changes Support* :: Priority levels of interrupts can be changed by software during run time. Interrupts that are being serviced are blocked from further activation until the ISR is completed, so their priority can be changed without risk of accidental re-entry.

   (d) *Reduction of Interrupt Latency* :: M3 includes automatic saving and restoring some register contents, reducing delay in switching from one ISR to another and handling late arrival interrupts.

   (e) *Interrupt Masking* :: Interrupts and system exceptions can be masked based on their priority level or masked completly using interrupt masking registers BASEPRI,

PRIMASK, FAULTMASK. They can be used to ensure that time-critical tasks can be finished on time without being interrupted.

Cortex-M3 Exception Types

Exception Number	Exception Type	Priority (Default to 0 if Programmable)	Description
0	NA	NA	No exception running
1	Reset	−3 (Highest)	Reset
2	NMI	−2	Nonmaskable interrupt (external NMI input)
3	Hard fault	−1	All fault conditions, if the corresponding fault handler is not enabled
4	MemManage fault	Programmable	Memory management fault; MPU violation or access to illegal locations
5	Bus fault	Programmable	Bus error (Prefetch Abort or Data Abort)
6	Usage fault	Programmable	Exceptions due to program error
7–10	Reserved	NA	Reserved
11	SVCall	Programmable	System service call
12	Debug monitor	Programmable	Debug monitor (break points, watchpoints, or external debug request)
13	Reserved	NA	Reserved
14	PendSV	Programmable	Pendable request for system device
15	SYSTICK	Programmable	System tick timer
16	IRQ #0	Programmable	External interrupt #0
17	IRQ #1	Programmable	External interrupt #1
...	...	...	...
255	IRQ #239	Programmable	External interrupt #239

ARM CortexM3 – Programmers view

Introduction to ARM CortexM3 – Memory Map

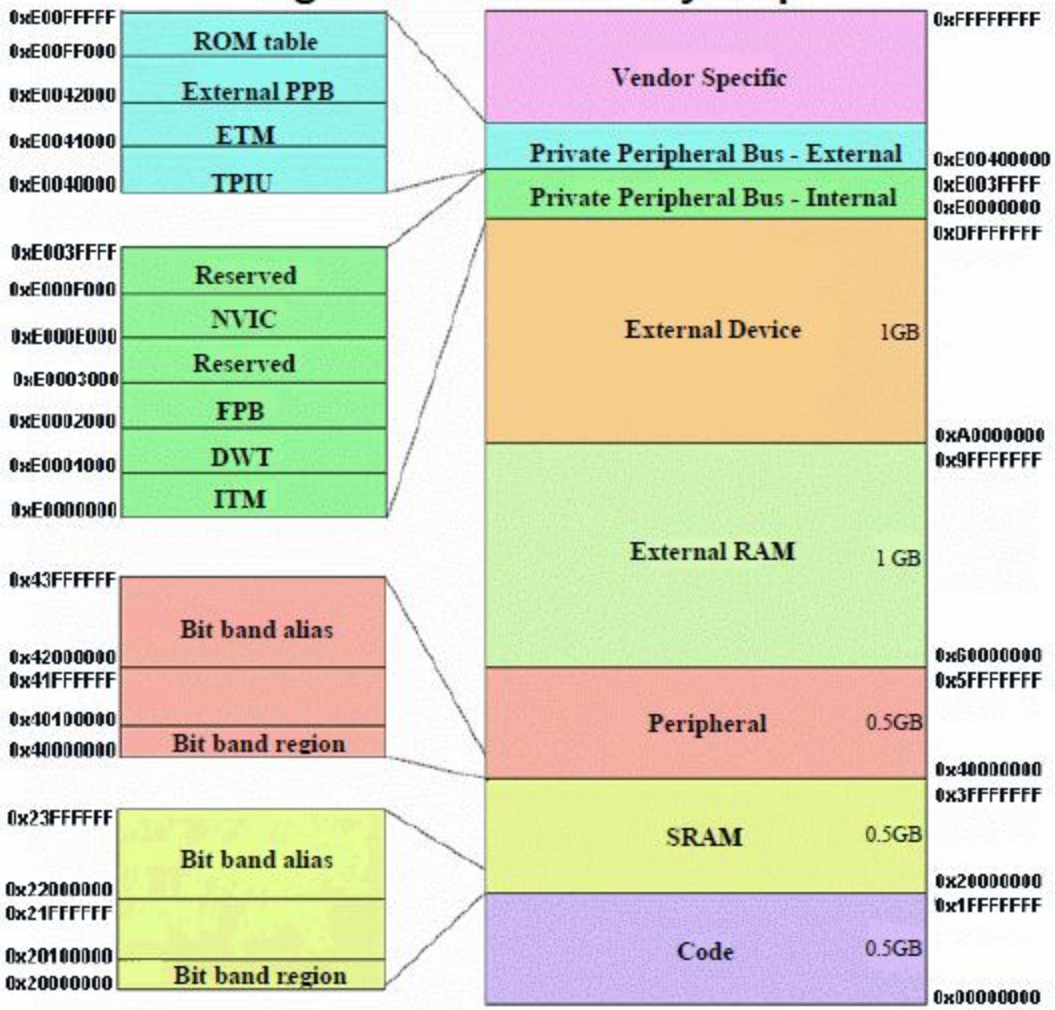
1. M3 has a predefined memory map. This allows the built-in peripherals, such as NVIC, and debug components to be accessed by simple memory access instructions.

2. This allows most system features through normal C program code.

3. Allows optimization for ease of integration and re-use

4. M3 has an optional Memory Protection Unit (MPU). The MPU is setup by an OS, allowing data used by privileged code to be protected from User programs. The MPU can be used to make memory regions ReadOnly to prevent accidental erasing of data, or to isolate memory regions between different tasks in a multi-tasking system. Overall, helps in making systems more robost and reliable.

Figure 4. The memory map

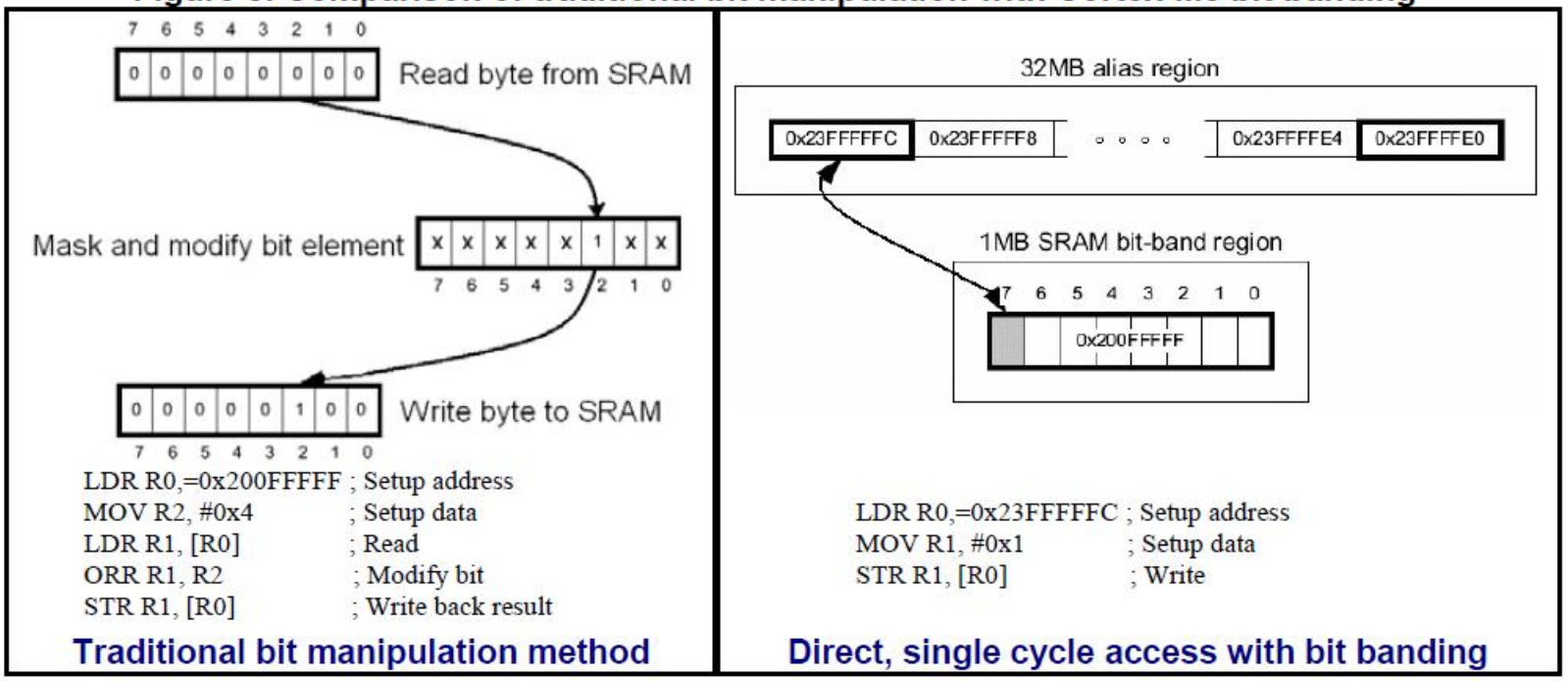


ARM CortexM3 – Programmers view

Introduction to ARM CortexM3 – Memory Map – Bit Banding

The Cortex-M3 processor enables direct access to single bits of data in simple systems by implementing a technique called bit-banding (Figure 5). The memory map includes two 1MB bit-band regions in the SRAM and peripheral space that map on to 32MB of alias regions. Load/store operations on an address in the alias region directly get translated to an operation on the bit aliased by that address. Writing to an address in the alias region with the least-significant bit set writes a 1 to the bit-band bit and writing with the least-significant bit cleared writes a 0 to the bit. Reading the aliased address directly returns the value in the appropriate bit-band bit. Additionally, this operation is atomic and cannot be interrupted by other bus activities.

Figure 5. Comparison of traditional bit manipulation with Cortex-M3 bit-banding



ARM CortexM3 – Programmers view

Introduction to ARM CortexM3 – Instruction Set

1. Basic syntax used is :: `opcode operand1, operand2, ...  ; Comments`
2. Bringup the TRM of M3 or GuideToArmCortexM3 (pg 71). Following categories :-
3. 16-bit Data processing instructions
4. 16-bit Branch instructions
5. 16-bit Load and Store instructions
6. Other 16-bit instructions
7. Same as above with 32-bit instructions

ARM CortexM3 – Programmers view and Development Environment

Acknowledgements

1. ARM Assembler Reference :: www.arm.com

2. The Definitive Guide To ARM Cortex-M3 :: Joseph Yiu