FAU Erlangen-Nueremberg

Seminar

Selected Topics in Embedded Systems

The ARM Microprocessor and ARM-based Microcontrollers

Nguatem William

May 17, 2006

Contents

1 Introduction

1.1 ARM History

ARM(Advanced RISC Machines)Ldt. is a semiconductor IP[1] supplier with headquarter in Cambridge,UK. ARM Ltd. pioneered in the early 1990s the concept of openly-licensable IPs for the development of a 32-bit RISC processor-based SoC[2]. The company licenses its products(processor cores, peripheral IP, development tools, application software, EDA[3] tools and design services) to a large network of partner companies. These licensees utilize ARM technology as essential building blocks

[1] acronym for **I**intellectual **P**roperty
[2] System-On-Chip
[3] Electronic Design Automation

for the microprocessors, peripherals and SoCs they develop and manufacture (e.g Sony, Intel, Nokia etc).

1.2 ARM Basics

The ARM is a 32-bit RISC machine with the following main features:

- a load-store architecture (register-to-register),
- 3-address instructions,
- conditional execution of all instruction,
- the ability to perform a general shift operation and a general ALU in a single instruction that executes in a single clock cycle,
- a large set of registers, all of which can be used for most purposes,
- extensible instruction set through coprocessor instruction set.

2 Programmer's Model

2.1 Registers and Operating Modes

ARM machines have 16 general-purpose user-accessible registers named r0 through r15 (see figure 1). Registers r15 is dedicated as the program counter (PC = r15).There is also a current program status register (CPSR) and 5 saved program status registers (SPSR). By convention registers r14 and r13 of the user-accessible registers are the link register (LR, saves subroutine return address) and stack pointer (SP) respectively.

The content of the CSPR-register(see Figure 2) can briefly be summarizes as follows:

- *N,Z,C,V* contains the negative, zero, carry and overflowflags respectively.
- The *T* flag is been used only in the THUMB-architecture(to be explained later)
- Bits *I* and *F* are used to disable Interrupt und fast-Interrupt.
- The *mode bits* defines the five different privilleged operating modi(*M0,M1,M2,M3,M4*)

ARM processors support 7 operating modes. This is advantageous by providing a clear cut difference between user and system levels, facilitate the handling of memory violations. The presence of many registers facilitate processing in this different modes. Usually, program execution in the ARM operates in user mode. However, the handling of exceptions and supervisor calls (which are sometimes called Software Interrupts) takes place at other well defined privileged operating modes. The CPSR[4:0] (bottom five bits of the CPSR, see figure 2) defines the different operating modes.

A summary of the different operating modes and their respective accessible registers is given in figure 3.

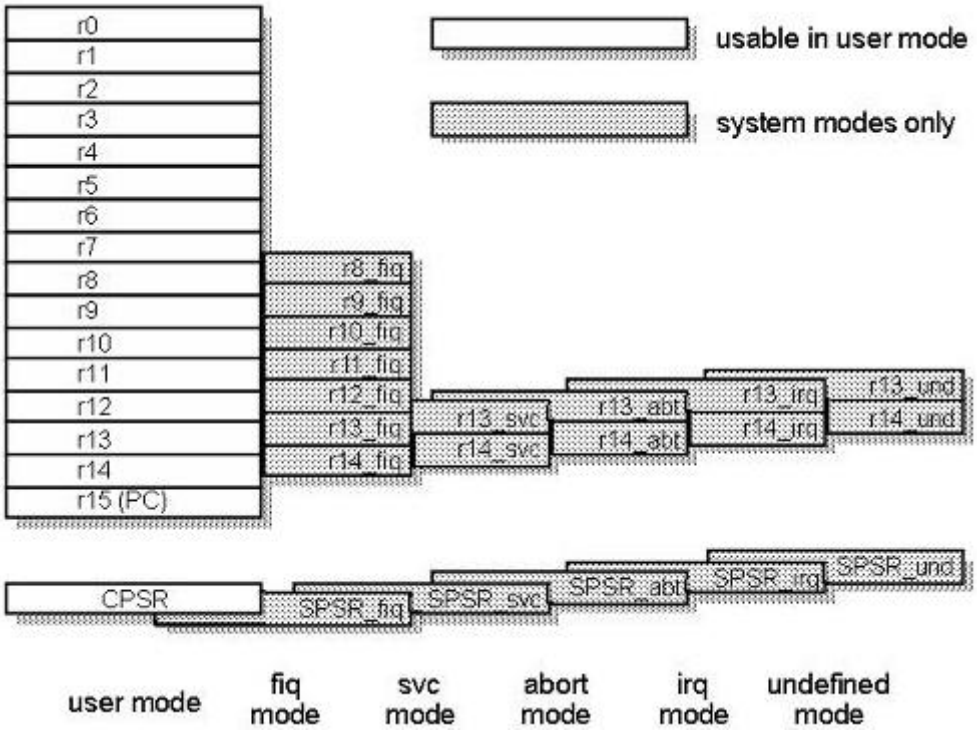

Figure 1: ARMs visible registers

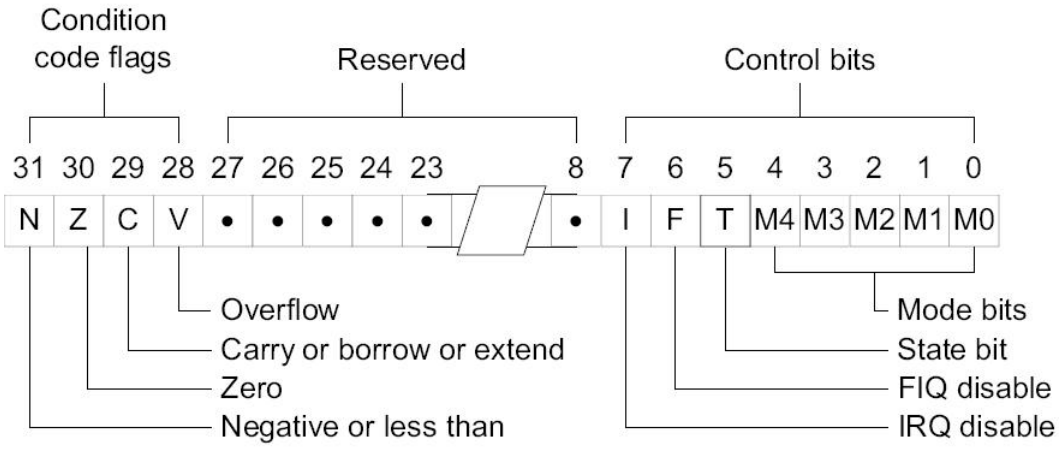

Figure 2: processor status register

SPRS [4:0]	Mode	Use	Accessible register set	
10000	User	Unprivileged, normal execution mode	PC, r14…r0	CPRS
10001	FIQ	High priority (fast) Interrupt is raised	PC, r14_fiq …r8_fiq	CSPSR, SPSR_fiq
10010	IRQ	Low priority Interrupt Occurred	PC , r14_irq...r13_irq, r12…r0	CPSR, SPSR_irq
10011	SVC	Software interrupt(SWI) is executed	PC, r14_svc…r13_svc, r12…r0	CPSR, SPSR_svc
10111	Abort	Handling of memory Access violations	PC, r14_abt…r13_abt, r12…r0	CPRS, SPSR_abt
11111	System	Run privileged task	PC, r14…r0	CPSR

Figure 3: ARM's operating modes

2.2 Exception Handling

In ARM exceptions may be divided into the following main groups:

1. Exceptions generated as direct effect of executing an instruction. Software Interrupts, undefined instructions (including coprocessor instructions in the absence of the coprocessor).
2. Exceptions generated as side-effects of an instruction. Example being a Data abort(memory fault during a load or store data access)
3. Exceptions generated externally, asynchronous to the normal program execution. IRQ and FIQ as examples.

The general exception handling mechanism is as follows:

- Saving of the current program state by copying of the PC into r14_exc and the CPSR into SPSR(exc:denotes the exception type).
- SP may used to save other user register if these are needed by the exception handler.
- The processor operating mode is changed to the appropriate exception mode.
- The return address is stored in LR_exc.
- Disabling of IRQs by setting bit 7 of the CPRS and, if the exception is a fast interrupt, bit 6 of CPRS is equally set.
- The PC is forced to take a value between 0x00 (vector address[4] of the Reset exception) and 0x1c (vector address of the FIQ exception), the particular value depends on the type of exception.

[4]vector address contains a branch to the relevant routine, though the FIQ code can start immediately since it occupies the highest vector address

To return to normal program execution:

- user registers are restored from the stack (provided they were modified)
- restoring of CPRS from SPSR
- restoring of the PC from LR_exc

All exception modes have banked registers for r13 (to provide private stack pointer) and r14(to store LR, see figure 1). The presence of a fast interrupt mode in addition to the normal interrupt (IRQ) gives a better means of differentiating between different priority interrupts that could arise. The efficiency of the fast interrupt mechanism is due to its additional private registers (r8-r12) providing a better handling of its routine without the need to save or restore these registers.

2.3 Memory

Like in most other processors, the ARM views memory as a linear collection of bytes numbered upwards from zero. Words could be stored in such a byte-arranged memory in two ways depending on two memory alignment schemes (big endian and little endian). ARM processors are generally neutral to any memory alignment scheme, that is ARM cores support both big endian and little endian. This neutrality allows for a more efficient integrition to the possible different peripheral components. Though the majority of this components are little endian accounting for the default aligment in all ARM cores being equally little Endian. As an example, in the ARM926EJ-S CPU with the ARM9 core, endianess is set through the BIGENDINIT input pin (0-for little endian, which is default and 1 for big endian) and the current endianess is indicated by the CFGBIGEND output pin.

2.4 The ARM Coprocessor Interface

The ARM architecture supports a general-purpose extension of its instruction set by adding hardware co-processors. Typical examples of this being the Coprocessor 15 responsible for the on-chip control of MMU and Cache in the ARM720, Coprocessors 10 responsible for floating point arithmetic and many other application specific co-processors (e.g piccolo for DSP). As an alternative to coprocessors is a software-based implementation. However, this approach is not very efficient.

3 ARM Internal

3.1 ARM Processor Organization

Based on the pipelined nature of ARM processor architecture, one can differentiate between several families of ARM processor organizations:

- the 3-stage, an example is the ARM7,
- the 5-stage pipeline, example being the strongARM and ARM9 cores,
- the 6-stage pipeline , example ARM10 core and,
- the 8 stage pipelined architecture organizations, e.g ARM11

The above mention architectures having more than 3-stage pipelines also possesses a Harvard architecture in order to overcome the von Neumann bottleneck(single Memory for Instruction and data).

3.2 ARM Instruction Set Architecture

Generally there are 3 basic variations in instruction length encoding: variable length (e.g,VAX, Intel 80x86), fixed length(e.g Motoroller) and hybrid (e.g Thumb). The ARM has fix length instruction encoding with a variation(Thumb) having hybrid encoding.

ARM's ISA can be divided into six broad class of instructions:

- Data movement or data processing instruction
- Branch instructions
- Status register transfer instruction
- Load and store instructions
- Coprocessor instructions
- Exception-generating instructions

In ARM all instructions are conditionally executed according to the state of the CPSR condition codes. This reduces the branch overhead and compensates for the lack of a branch predictor.
ARM also has a built-in barrel shifter which provides a limited degree of parallelism by carrying out a shifts as part of other instructions. The shift is applied to the second operand rather than the result.

Example, Consider the ARM instruction:

$$ADDr1, r2, r3, LSL\#4$$

This shifts the 32-bit operand in register r3 left by four places before adding it to the contents of register r2 and depositing the result in register r1 all in the same clock cycle. In RTL[5] terms this instruction is defined as

$$[r1] \leftarrow [r2] + [r3] \text{ x } [16]$$

4 ARM Cores and Versions

4.1 IP cores

ARM Ldt. offers to its partners a variety of 16/32-bit embedded RISC cores that are grouped into a range of families: ARM1 through ARM6, the ARM7, strongARM (in collaboration with Intel), ARM8, ARM9, ARM10, ARM11, SecurCore and the ARM Cortex families of microprocessor cores. Some of these cores are superseded by others, ARM1-ARM6 by ARM7 and ARM8 by both ARM9 and ARM10.
Within the different families of cores, ARM also offers different extensions which are denoted on a particular cores name. As an example in the ARM7 family, there is an ARM7EJ-S, ARM720T and ARM7TDMI-S with T, E, J, S, M and D representing the respective extended funtionality, where,
T: 16-bit encoded instruction set (see section 6 on THUMB),
D: on-chip Debug support, enabling the processor to halt in response to a debug request,
M: enhanced Multiplier, 64-bit result

[5]Register Transfer Level

Family	Core	Feature	Cache (I/D)/MMU	in Application
ARM7TDMI	ARM7TDMI(-S)	3-stage pipeline	none	Game Boy Advance, iPod
	ARM710T	MMU		Psion 5 series
	ARM720T		8KB unified, MMU	
	ARM7EJ-S	Jazelle DBX	none	
ARM9TDMI	ARM9TDMI	5-stage pipeline	none	
	ARM920T		16KB/16KB, MMU	Motorola i. MX1
	ARM940T		4KB/4KB, MPU	GP2X (second core)
ARM9E	ARM946E-S		variable, tightly coupled memories, MPU	Nintendo DS, Nokia N-Gage, Conexant 802.11 chips
	ARM926EJ-S	Jazelle DBX	variable, TCMs, MMU	Siemens and Beng mobile phones (x65 series and newer)
ARM10E	ARM1020E	(VFP)	32KB/32KB, MMU	
	ARM1026EJ-S	Jazelle DBX	variable, MMU or MPU	
ARM11	ARM1136J(F)-S	SIMD, Jazelle DBX, (VFP)	variable, MMU	
	ARM1156T2(F)-S	SIMD, Thumb-2, (VFP)	variable, MPU	
	ARM1176JZ(F)-S	SIMD, Jazelle DBX, (VFP)	variable, MMU+TrustZone	
Cortex	Cortex-M3	Microcontroller profile	no cache, (MPU)	Luminary Micro[2] microcontroller family
	Cortex-A8	NEON, Jazelle RCT, Thumb-2	variable (L1+L2), MMU+TrustZone	

Figure 4: ARM Cores

I: embeddedICE hardware , give on-chip breakpoint and wachtpoint.
J: Jazelle enhancement is present (see section 6 on ARM's special properties),
S: complette Synthesizable.

A summary of the main features of some of the cores is given on figure 4. On this figure, one should note the general trend of increasing functionality as the time increases (time increases downwards on the figure).

4.2 Versions

From the first processors produced by Acorn RISC Machines[6] to latter cores produced by ARM Ldt., several changes have been made on the ISA over this years of development. These changes allows code compatibility between versions of the ARM Cores. The current version of the ARM core is v7[7] denoted ARMv6 (ARM11 family) with their main features being JazelleDBX, NEON(extension of SIMD for media processing) TrustZone, Thumb-2 and, an 8-stage pipeline architecture. A summary of some versions is tabulated below.

[6] former name of ARM Ldt.
[7] release

Core	Architecture Versions
ARM1	v1
ARM2	v2
ARM2as, ARM3	v2a
ARM6, ARM600, ARM610	v3
ARM7, ARM700, ARM710	v3
ARM7TDMI, ARM710T, ARM720T, ARM740T	v4T
StrongARM, ARM8, ARM810	v4
ARM9TDMI, ARM920T, ARM940T	v4T
ARM9ES	v5TE
ARM10TDMI, ARM1020E	v5TE
ARM11	v6
ARMCortex	v7

Figure 5: ARM architecture Versions

5 ARM support for System Developtment

5.1 AMBA on-chip bus architecture

The availability of the AMBA interface specifications on ARM processors permits, IP developers implement and test modules without prior knowledge of the system into which the component will be finally integrated. This also facilitate communication between the ARM core and other on-chip macrocells (peripheral components to the ARM core)in an SoC. Three diffrent busses are defined within the AMBA specification:

1. The Advanced High-performance Bus (AHB) is used to connect high performance system modules.

2. The Advanced System Bus (ASB) is used to connect system modules. It supports only burst mode data transfer.

3. The Advance Peripheral Bus (APB) offers a very simple interface for low-performance peripherals.

A typical AMBA-based microcontroller will support either AHB and APB or ASB and APB. The latest generation of the AMBA specification is known as AMBA 3 AXI (Advanced eXtensible Interface). AMBA 3 AXI Specification delivers a better performance and support massively parallel systems.

5.2 The ARM debug architecture

ARM cores provides hardware facilities for debugging comparable to those offered by an In-Circuit-Emulator (ICE) approach. This is archived by extending the functionality of the JTAG test port and

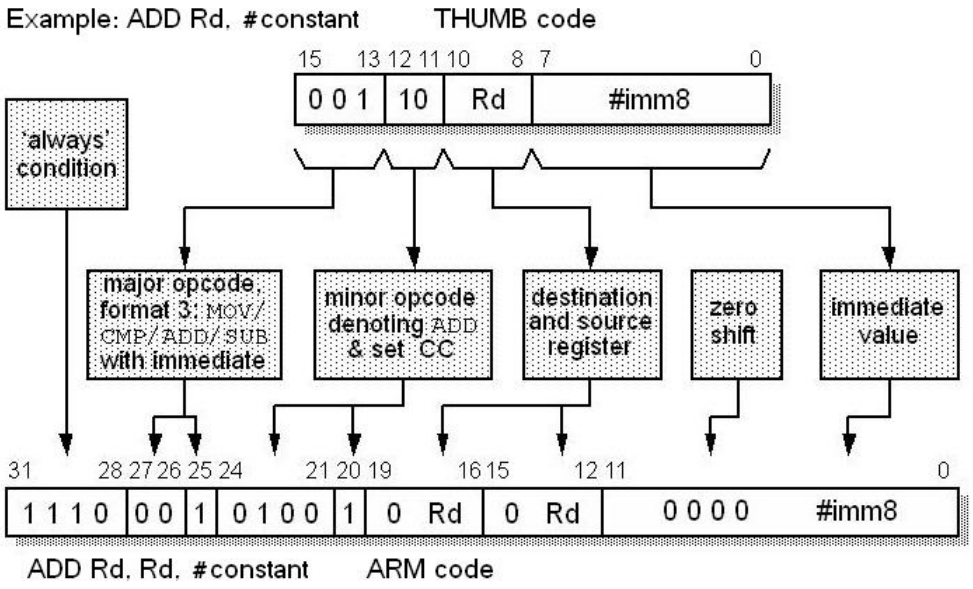

Figure 6: THUMB-ARM Instruction Mapping

adding new breakpoint and watchpoint registers. These registers are accessed by JTAG scan chains which also forces instructions into the processor to access it and the system state. The extra overhead by adding of the new registers on the core is overwhelm by its resulting advantages. Cores having this functionality are called EmbeddedICE modules.

Most recent ARM cores also support an embedded trace mechanism capable of doing a realtime debugging. This is archived by limiting the interface bandwidth using a trace compression techniques.

6 ARM Special Properties

6.1 Thumb (Encoding an Instruction set)

With the introduction of RISC computers in embedded applications, the 32-bit fixed instruction format became a liability since cost and hence smaller code are important. Several manufacturers offered a new hybrid version of their RISC instruction sets which archives a 32-bit performance at an 8/16-bit system cost. ARMs approach in encoding of the most commonly used 32-bit instructions into 16-bit opcodes is known as THUMB. The resulting 16-bit instructions are then decompose in the instruction pipeline into full normal 32-bit ARM instructions in real-time without any performance lost prior to execution. It is necessary to note that THUMB is not a complete ISA but a subset of the normal 32-bit found in some ARM cores. These THUMB enabled cores are given the label "T" (eg ARM7**T**DMI). Figure 6 illustrates an example of THUMB's instruction compression Scheme.

Advantages and Disadvantages of the THUMB

+ Excellent code density for minimal system size.

- In Thumb state there isn't any direct access of registers cpsr/spsr (i.e must switch to ARM state to access these registers)

- Only branch instructions are executed conditionally whereas all ARM instructions are conditionally executed

- Many Thumb data processing instructions use a 2-address format.

The success of Thumb has lead to the development of Thumb-2.This extension of the normal Thumb instructions set encodes new groups of 32-bit instructions. Thumb-2 also extends both the ARM and Thumb instruction set with yet more instructions, including bit-field manipulation, table branches, and conditional execution.

6.2 Jazelle

Jazelle is a hardware-accelerated java code mechanism. ARM processor core requires an extra decoder (Jazelle instruction decoder) between the instruction cache and the CPU pipeline which decodes the java bytecodes into normal ARM instructions which are immediately executed.Unlike THUMB, there is no direct mapping of instructions to the ARM instruction set.

This direct hardware execution of Java bytecode on a single core compared to a more memory-demanding and slower software alternative with a JVM[8] or even a more-expensive solution with coprocessors and dual-core processors[9] has a number of advantages,

- + provides the reuse of existing architectural components leading to lower system cost .
- + lesser power needed due to an in-core integrated solution.
- + single core for both java and native code execution giving platform developers the freedom to run Java applications alongside established operating systems, middleware and application code on a single processor.
- - additional overhead for realizing the hardware decoder circuit.

In the java state, bytecodes are divided into three groups, executed, emulated and undefined. Though most of the java bytecodes are executed while at the java state, the remainder (bytecodes emulated)[10] of the code are cracked into small pieces and interpreted as short sequences of normal ARM instructions.

Besides the Jazelle DBX[11] mentioned above that supports only java bytecodes, there is also Jazelle RCT(Runtime Compilation) capable of supporting different ”bytecodes” languages like .Net MSIL(Microsoft Intermediate Language),Python and Perl. Jazelle RCT provides a small modification to Thumb-2, making the instruction set particularly suited to code generated at runtime (example, by JIT[12] compilation) by adding 12 new 16-bit instructions. This new architecture are named THUMB-2EE and their operating states THUMB-EE. Thumb-2EE is aimed at accelerating different Virtual Machines, and allows JIT compilers to output smaller compiled code without impacting performance. Jazelle RCT is fully intergrated from all cores from v7 (ARM Cortex family of cores see figure 5).

[8] Java virtual machine
[9] double-core parallel operating processors
[10] Typical examples are functions like complex divisions, method invocation, and floating-point operations.
[11] DBX for Direct Bytecode eXecution
[12] Just In Time

6.3 TrustZone

This is ARMs hardware-based security mechanism. This is archived by providing a clear hardware partition between secure code and data from non-secure information. Transitions between this two worlds is being coordinated by the present of a new mode called *Secure Monitor*, acting like a gate keeper. Such an address space separation enables secure code and data to run alongside an OS securely and efficiently. This intrinsic device Security feature, gives room for developers to build additional security (e.g Cryptography), on top of the secured hardware.

6.4 Neon Technology

The increasing demands for new media applications on handheld (embedded devices) acts as a driving force for new technology. It is a combined 64-bit and 128bit hybrid SIMD [13] instruction set that provides hardware acceleration for media and signal processing applications. It works independently by having a private pipeline and register files for independent execution . The key features of this technology are:

- aligned and unaligned data access,
- support for integer and floating point data types. This ensures adaptability to a broad range of signal processing application, from compression, encoding/decoding to even 3D graphics.
- the presence of a large register file. This enables the efficient handling of data and minimizes memory accesses leading to a very high performance increase.

6.5 Other Extensions

Besides the above mentioned extensions of the ARM, there exist a bunch of other extension solutions like,
ARM DSP-Enhanced Extension(v5TE): The adding of new DSP instructions to ARM instruction set in order to accelerate applications demanding a DSP oriented processor. This is advantageous over DSP-based implementations due to a reduction in power consumption and chip area since processing all the processing is done on a single processor.
ARM Intelligent Energy Manager (IEM) solution: Implementation of advanced algorithms to optimally balance processor workload and energy consumption. OptimoDE, etc.

7 Arm Based systems

7.1 ARM CPU

ARM CPU is a combination of ARM Core + Cache + MMU. An example is the Intel's (licensee of ARM Ldt.) ARM-based solution known as strongARM SA110 CPU. The StrongARM CPU has the following features :

- Seperate instruction(ICache) and data(DCache) caches (Harvard architecture) both being 16K.
- Seprete 32-entry instruction and data TLB[14].
- An 8-entry write buffer with up to 16 bytes per entry.

[13] Single Instruction Multiple Data
[14] translation look-aside buffer

- A system control coprocessor controlling the memory system.
- a five stage pipeline:

  0 Fetch stage: Fetch instruction from Icache or memory.

  1 Decode stage: Decode instruction, read input values from register file.

  2 Execute stage: Shifts and arithmetic (except multiplies).

  3 Buffer stage: Data cache or memory access, multiplies, and system coprocessor access.

  4 Writeback: Write output values to register file.

The strongARM design has now been replaced by the Intel XScale family which are widely used in personal digital assistants.
ARM-based microcontroller is a combination of ARM-CPU and Periphery (I/O-controllern, Real-time-clock, etc).

8 References

1. **ARM System-on-Chip Architecture (2nd Edition) by Steve Furber published by Addison Wesley ISBN 0-201-67519-6**
2. Jazelle DBX White Paper Ref: N/A, Issued: 01 September 2004, www.arm.com/documentation/White_Papers/index.html
3. Jazelle RCT White Paper Ref: N/A, Issued: 01 May 2005, www.arm.com/documentation/White_Papers/index.html