Product Migration from FPGA (Cortex-M1) to a Standard ARM Based Microcontroller

Jens Stapelfeldt
Doulos Ltd.
jens.stapelfeldt@doulos.com

Dr. Gerd von Cölln
Technical University OOW
coelln@technik-emden.de

Abstract

With the Cortex-M processors architecture ARM has moved further into the microcontroller space than ever before. Moreover ARM is offering with the Cortex-M1 the first ARM processor specifically designed for the FPGA world. Microcontrollers for FPGAs have been in the past, and are still, dominated by proprietary soft-core designs from the FPGA vendors (eg. NIOS for Altera and MicroBlaze for Xilinx). The new Cortex-M architecture from ARM provides an interesting and challenging opportunity for merging of the FPGA and Embedded Systems markets.

This paper will introduce the Cortex-M1 and Cortex-M3 architecture with the focus on their similarities and differences, and will show how to develop and migrate a product from FPGA to a standard ARM based controller or even a Cortex-M3 based microcontroller. This paper will give an independent overview of the design flow and tool possibilities arising out of this unique situation.

1 Intro

With the strong increase of FPGA capacity on the one hand and falling unit cost on the other, FPGAs are no longer used only for prototyping and validation. Depending on the application and the projected sales volume, FPGAs can directly compete with ASICs, as cost for ASIC mask production strongly rises for newer technologies. With this, FPGAs are becoming more and more interesting for SoC-designs, which usually contain a microprocessor as central processing unit.

Using synthesizable, configurable microprocessor cores (soft-cores) in a FPGA has several advantages over using hard-cores, even if hard-cores usually run faster and leave more programmable elements free for customer logic. Soft processor cores can be customized or tailored for a specific application. They can easily be modified to fix problems, add features, optimize designs or even adapt the design to different products. Providing the necessary infrastructure, it will also be possible to modify designs which are already in use. For FPGA vendors, it is no longer necessary to deliver multiple versions of a FPGA, i.e. with and without processor core, which reduces costs.

In the past, two trends have been seen: 1) The usage of processor cores that have been optimized for ASICs and 2) the usage of proprietary cores (e.g. NIOS, MicroBlaze). Using these solutions, designers have been faced with a number of problems. On the one side, ASIC-optimized cores often showed bad performance in terms of processing speed or area efficiency when mapped to FPGAs. On the other hand, FPGA optimized cores (e.g. Altera´s NIOS or Xilinx´s Microblaze) cause problems because they are proprietary. This means that designers are stuck with particular design tools and suffer the problem of porting software originally coded for industry standard processors.

A third way was to offer industry standard architectures (e.g. ARM7 and PowerPC) as hard-cores. These cores only had few possibilities in terms of adapting processors to specific applications or requirements. With this, one main advantage of FPGAs, i.e. flexibility, is strongly impaired. Thus, these kind of FPGA have often been used as evaluation platform only.

ARM's new Cortex-M1 will overcome most of these drawbacks. The Cortex-M1 is an FPGA-optimized, industry standard microprocessor which is provided as a configurable soft-core. The main advantages compared to other FPGA-cores are: 1) The huge ecosystem of tools and design support, 2) the huge amount of existing software and hardware IP for ARM processors, and 3) upward compatibility to the Cortex-M3. These advantages help to reduce risk and get products to the market sooner.

The rest of this paper will be organized as follows: Section 2 focuses on the organisation of the Cortex-M, giving an overview of the core's organisation and the instruction set architecture.

Differences between M1 and M3 will be pointed out and discussed. In section 3, we give an overview of Actel's and Altera's Cortex-M1 FPGA solutions. A comparison with other FPGA soft-cores will be given in Section 4. In section 5 we conclude with a brief summary.

2 Architecture of Cortex-M

The Cortex-M architecture will over time replace the successful ARM7TDMI architecture, which – even though this was not its original intent – has been used in a large number of microcontroller designs. In the following, we will give a short overview on the organisation of the Cortex-M3 and Cortex-M1 architecture and point out some major differences to the ARM7TDMI. We will start with the Cortex-M3 and point out differences with M1, which is a tailored version of the M3.

2.1 Core Organisation

One of the biggest changes to the traditional ARM architectures is that for the Cortex-M the memory map is predefined and almost identical for the M1 and the M3. The complexity of the AHB interface matrix is different for the M1 and M3. The Cortex-M3 has different interfaces to access instructions and data memory space simultaneously, making the M3 core look like a Harvard architecture.

The Cortex-M1 has a less complex AHB matrix (AHB-Lite interface) and is therefore more like a von Neumann architecture. The optional TCMs (tightly-coupled-memory) for the Cortex-M1 are only for a maximum size of 1Mbyte. If this is implemented the Cortex-M1 can access instructions and data simultaneously from that memory space and would behave like a Harvard architecture within that memory space.

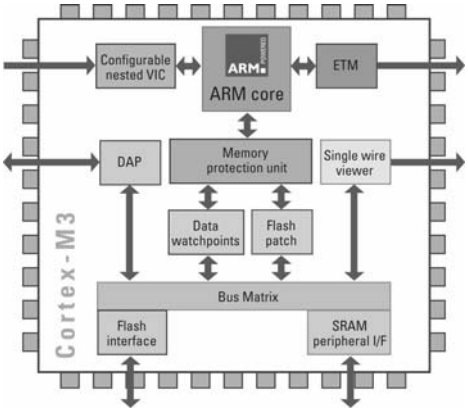

Figure 1: Block diagram of the Cortex-M3

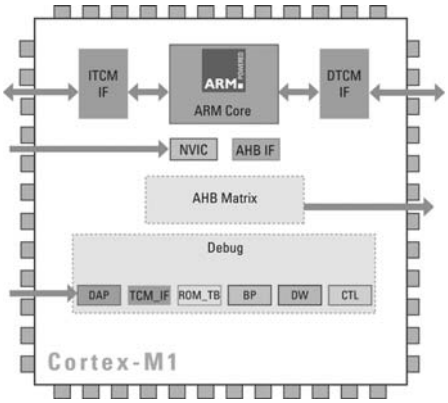

Figure 2: Block diagram of the Cortex-M1

Both cores have three pipeline-stages: instruction fetch, decode and execution. During the execution phase, registers are read, an appropriate operation is executed and results are written back to the register bank.

The Cortex-M3 has a pre-fetch buffer and some easy branch prediction included in the core. When a branch instruction is encountered, the decode stage also fetches the branch destination instruction. In the following execute stage, the branch is resolved and it is known which instruction is to be executed next. If the branch is not to be taken, the next sequential instruction is already available. If the branch is to be taken, the branch instruction is made available at the same time as the decision is made, restricting idle time to just one cycle. This pre-fetch of the branch destination instruction is different from the handling of branches in the ARM7TDMI.

The Cortex-M contains 13 general purpose registers (r0 – r12), the stack pointer (SP), the link register (LR), the program counter (PC) and the program status register (xPSR). Physically the core has fewer registers but, from the programmer's point of view, they are the same registers as for any other ARM processor if you follow the AAPCS (ARM Architecture Procedure Calling Standard).

Compared to previous ARM architectures, the number of banked out registers has been dramatically reduced, as the handling of exceptions is significantly different. In previous ARM architectures "banked out" registers are used for exception handling. When the core enters the FIQ-Mode due to a signal change at the fast interrupt input, register r8 up to the link register are

“exchanged”. Thus, it is not necessary to save these registers in an interrupt service routine that handles the FIQ. The interrupt mechanism of the Cortex-M is completely different. Entering an interrupt, the Cortex-M will finish the current instruction and automatically save PC, r0-r3, r12, r14 and xPSR (following the ARM Architecture Procedure Calling Standard). On an interrupt return the status will be automatically restored. We will describe the interrupt mechanism in more detail in Section 2.3.

The Cortex-M3 offers an enhanced multiplier and is the first ARM core offering a hardware divider unit as part of the instruction set (v7M ISA). Additionally, this core offers some interesting power saving features and instructions. Detailed discussion of these features is outside the scope of this paper. The Cortex-M1 offers different implementations for the multiplier depending on the resources available on the FPGA architecture.

Shown in Figure 1 is the debug subsystem, which is optional for the Cortex-M1. Its main components are the BreakPoint Unit (BPU), Data Watchpoint (DW) Unit, the Debug TCM interface and the Debug Control Register. The BPU has four instruction comparators that can individually be configured to perform hardware breakpoints. The DW Unit has two comparators that can be configured as hardware watchpoints. Access to the ITCM and DTCM is realized by two corresponding debug interfaces. If the interfaces are used, the FPGA must support dual ported memory to connect both the debug memory interface and the core memory interface. If dual ported memory is not supported, arbitration logic must be added. Debug access is provided via a Serial Wire JTAG Debug Port (SWJ-DP). The SWJ-DP is a standard CoreSight debug port that combines JTAG-DP and Serial Wire Debug Port (SW-DP). The SWJ-DP is designed to permit pin sharing of JTAG-TDO and JTAG-TDI when they are not being used for JTAG debug access. The advantage of this optional logic is that one can build the first systems with the debug logic and develop and debug the software. When the system is stable a smaller version (perhaps even in a smaller and thus cheaper FPGA) could be used.

The following table summarizes the main differences between the ARM7TDMI and Cortex-M1.

	ARM7TDMI	**Cortex-M1**
Architecture	Von Neumann	Von Neumann (Harvard (with TCMs for max. 1MByt)
Pre-fetch buffer	No	E.g. used for branch instructions
Cache Support	No	No
Interrupt controller	No interrupt controller, two interrupts	NVIC, 1-32 interrupt, with automatic processor state saving and restoration
Register set	37 registers with several register banks	Simplified register set
Debug	JTAG debug interface	Optimized for microcontroller application
External interface	Native ARM7 – not AMBA	AMBA AHB-lite
Vector table	Instructions	Adresses
Size	6100 tiles, 29 MHz (Actel)	4300 tiles, 72 MHz (Actel)

Table 1: Comparision to ARM7TDMI

2.2 Instruction Set Model

The ARM7TDMI core was the first ARM core implementing the ARMv4T instruction set architecture, including a 32-bit ARM ISA and a 16-bit Thumb ISA. The processor can be switched between ARM ISA mode and Thumb ISA mode dynamically. The main reason for the Thumb ISA is to achieve higher code density and reduced bus width and activity. Internally the processor decompresses the instruction to 32-bit ARM instruction, so the datapath remains unchanged (except the Thumb decompressor).

Reducing instructions to 16-bit results in some limitations: 1) Most Thumb instructions are unconditional, while all ARM instructions are conditional. 2) Many Thumb data processing instructions use a 2-address format, while ARM data processing instructions use a 3-address format. 3) The number of directly accessible registers is reduced to r0 to r7.

One major drawback of the Thumb ISA is that accessing the coprocessors or handling of exceptions requires 32-bit ARM instructions. With this, time consuming switches from Thumb to ARM mode are necessary. With the new ARM Thumb2 instruction set ARM makes up for that and with the new compilers one can dynamically choose between performance and code density. The first ARM core implementing that instruction set, with several other improvements, was the ARM1156 core.

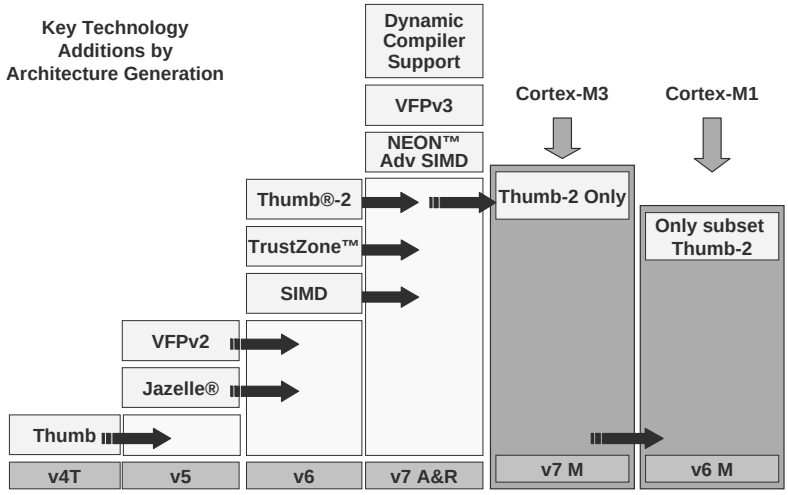

Figure 3: ARM instruction set evolution

Figure 3 shows the development of the different ARM instruction set architectures over recent years. The ARMv7M instruction set follows from all the improvements needed for the micro-controller world. The Cortex-M3 is consequently only running with the Thumb and not supporting the ARM (32 bit) instruction set.

The Cortex-M1 runs with the ARMv6-M ISA and makes use of that functionality just to run the exceptions in Thumb mode without the overhead for the rest of the architecture. The ARMv6-M is a subset of the Thumb-2 (ARMv7M), consisting of all 16-Bit Thumb instructions and some Thumb2 32-bit instructions, that can be executed in Thumb processor mode.

Operating states and modes

Operating modes have strongly been simplified compared to modes from other ARM processors. The ARM7TDMI as other ARM processors distinguishes between User mode for normal code execution and FIQ, IRQ, SVC and Undef Modes for privileged code execution.

The Cortex-M processor has two modes of operation only:

- Handler mode – The handler mode is entered as the result of an exception (interrupt, fault, etc.). Running in handler mode, the processor is executing an exception handler or interrupt service routine. The handler mode always provokes a privileged execution.
- Thread mode – The thread mode is entered on Reset, and can be re-entered as the result of an exception return. The thread mode is used for normal code execution, privileged or unprivileged. Privileged operation allows access to all processor resources, while the access is limited for unprivileged operation. On Reset the privileged Thread mode is entered.

Besides the operation modes, two operating states are available:

- Thumb state – This is normal execution, running the set of 16-bit, halfword-aligned Thumb and Thumb-2 instructions; as well as the 32-bit BL, MRS, MSR, ISB, DSB, and DMB instructions.
- Debug state – This is the state when halting for debug.

Registers

As already mentioned r0 to r12 are general purpose registers. r13 to r15 are used as stack pointer, link register and program counter. r0 to r7 are named as "low registers" which can directly be accessed by all instructions. r8 to r12 are the "high registers" and are not accessible by all 16-bit instructions. Thus, additional move operations are sometimes required to access the data in the high registers.

In contrast to ARM7TDMI program status register (PSR), different parts of the Cortex-M program status register can be accessed as individual registers (application PSR, interrupt PSR and execution PSR). They can be accessed as individual registers or combinations using MSR and MRS instructions. The application PSR provides access to the condition code (N/Z/C/V) flags. The

interrupt PSR contains the interrupt service routine (ISR) number of the current exception activation. The execution PSR contains the Thumb state bit. For the Cortex-M1 this bit is always 1. Figure 4 shows the PSR and corresponding processor status bits.

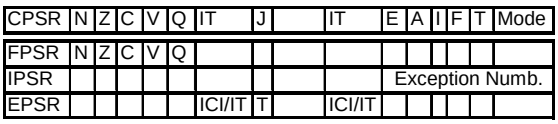

Figure 4: CPSR from the ARM11 and the new format for the Cortex-M

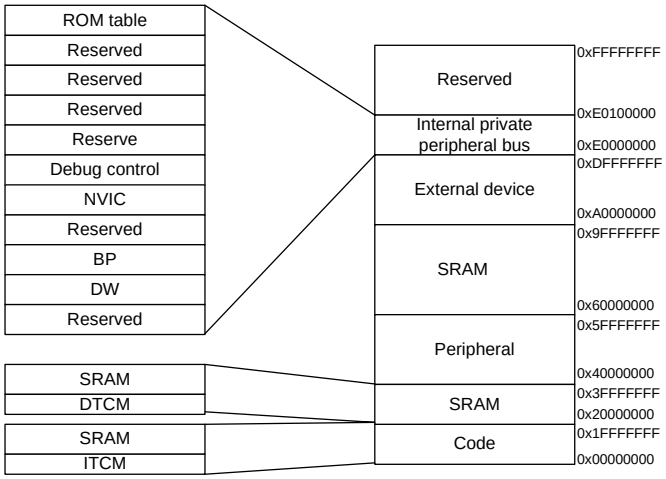

Figure 5: Cortex-M1 memory map

Memory formats and map

The Cortex-M1 addresses the memory as a linear collection of bytes. Since caches are not supported, the processor assumes zero wait-state memory on chip, at least for the TCMs. Supported data types are 32-bit words, 16-bit halfwords and 8-bit bytes. Data can be accessed in little endian (default) or big-endian format (instructions are always accessed in little-endian).

In contrast to the ARM7TDMI the Cortex-M1 has a fixed memory map (cf. Figure 5). Code can be accessed on the ITCM or external memory via the AHB-lite interface. Data access can appear on the corresponding DTCM interface or the AHB-lite interface to off-chip memory. The private peripheral bus area provides an access to the interrupt controller, the breakpoint and watchpoint unit. As the SRAM area, the private peripheral bus area is marked as Execute Never (XN) area. Thus, instruction fetches are prevented by the processor hardware.

Cortex-M3 Bit Banding

Another interesting feature implemented in the bus interface of Cortex-M3 is the so called “Bit Banding”. Microcontrollers often perform bit accesses, e.g. to memory addresses that grant access to registers of internal or external peripherals. C allows the definition of variables to access a group of bits, although not in a compiler independent way [4]. Usually, access is performed using bit-masks and logical operations to read or write corresponding bits. This leads to difficulties if access has to be done in an atomic manner.

Cortex-M3 bit banding eliminates this problem. Writes to a word address in the bit band alias area affect a single bit in the bit band region. In detail, Bit 0 of the stored word is written to the appropriate bit. The write is translated to an atomic read-modify-write on the Cortex-M1 bus. With this, shared memory problems can be avoided. The Cortex-M1 has two 1 Mbit areas, one in the lower part of the SRAM area and one in the lower part of the Peripheral area. The appropriate 32MByte bit band alias areas are located directly above these bit band areas.

2.3 Interrupt Mechanism

In various ARM7 based microcontrollers the different semiconductor companies have built their own version of an interrupt controller connected to the core via AHB Bus. NXP (Philips) is using an ARM IP (PrimeCell PL-190 VIC Vector Interrupt Controller), whereas STM and Atmel use their own implementation in their ARM-based controller families. In some cases this can make a significant difference in the performance when running interrupt intensive applications. The variety of interrupt controllers can cause problems when porting software even from an ARM7 based system to another ARM7 based system.

The new Cortex-M architecture alleviates that problem by making the interrupt controller part of the core itself. In this case the NVIC (nested vector interrupt controller) is tightly coupled to the core and has its own interface. The number of an active interrupt is visible in the program status register in the core. This, together with some other changes to the architecture, makes the interrupt behaviour of the ARM core much faster and exactly predictable (deterministic) unlike previous ARM architectures.

In the Cortex-M3 the NVIC can support up to 240 external interrupts. The Cortex-M1 can support up to 32 interrupts. The configuration of the number of available interrupts is done during core configuration at synthesis time. The first 15 interrupt lines and addresses in the vector table (not including the 240/32 external interrupts) are reserved for internal use. They are used for the NMI and internal exceptions like the BusFaults, hard Faults, MPU Fault and the DebugMonitor only to name some. Some of them are implemented in the Cortex-M1 as well.

Interrupt Registers

The NVIC has several configuration and control registers and can be accessed like a memory mapped peripheral from address 0xE000E000. Each interrupt has several registers to control it. The Enable Bit (SETENTAx Register 0xE000E100) can enable or mask the interrupt. Writing 1 to the bit sets it and thus enables the interrupt, and a read indicates the value of the register. Writing 0 to the register has no effect.

To clear the enable of the register one must write a 1 to the “Clear Enable Register” (CLRENAx Register 0xE000E180). Reading that register indicates the current value of the enable status. For some people that may be new since some microcontroller users are used to doing that with only one register. Depending on the numbers of the implemented interrupts, there is the need for an additional Enable and Clear Enable Register for an additional set of 32 interrupts (e.g. if you have implemented 64 external interrupts you will have SETENTA0 and SETENTA1). For the Cortex-M1 there is only the SETENTA0 and CLRENA0 at the specified address since we have at most 32 interrupts.

If the Pending Bit (SETPENDx register 0xE000E200) is set the interrupt is pending. That bit can also be used to set the interrupt via software a very useful feature to test interrupt service software. Again the bit will be cleared in a different register named the “Clear Pending register” (CLRPENDx Register 0xE000E280) by writing a 1. A read of either register indicates the current pending status.

Priority Level Register

The NVIC has the possibility to pre-empt active interrupts depending on the individual priority of an interrupt. The priority of an interrupt can be set in the Interrupt Priority Level register (0xE000E400 – 0xE000E4EF). The options here are max. 8 bits and min. 3 bits for the Cortex-M3. This enables rather complex possibilities for priority levels and subpriority level groups. The Cortex-M1 uses the same registers but only 2 bits are used to indicate the priority; the remaining bits are reserved. So the porting of the software to a later Cortex-M3 is very easy.

Active Status Register (0xE000E300-0xE000E31C)

Each interrupt has its own active bit in the Interrupt Active Status Register indicating if an interrupt is executing or “active – stacked”. Active-stacked means in this case, the interrupt was executing, but was pre-empted by another higher-priority interrupt.

There are some more special registers like the Exception-masked register (PRIMASK), Fault-masked register (FAULTMASK) and the based priority register (BASEPRI) which can affect the interrupt processing. Discussion of these features would require too much detail for this paper, and they are not all implemented in the Cortex-M1 architecture.

2.4 Interrupt Behaviour

As mentioned earlier the Cortex-M has implemented an “micro-code” mechanism which saves and restores the 8 relevant registers as required by the AAPCS. This is done automatically by the core, unlike previous cores. The core also updates the stack pointer and link register, and sets the program counter to the relevant exception address. The exception address is taken from the vector table. Please note that for the Cortex-M the vector table holds an actual program address, unlike previous ARM architectures.

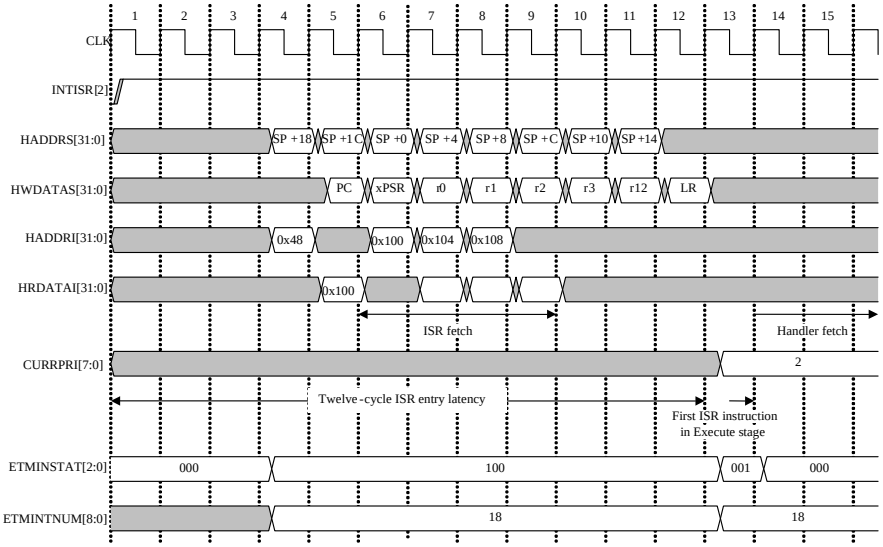

Figure 6: Interrupt timing diagram

Figure 6 shows a timing diagram of the possible interrupt scenario for a Cortex-M3 core. In this case the diagram shows the data and address bus of the SRAM- (HADDRS and HWDATAS) and the Code- (HADDRI and HRDATAI) interface. The PC value is pushed as the first item to the stack. This enables the code to set up the address of the corresponding interrupted handler in the PC register and the instruction interface while still pushing the register r0 to r3 and r12 to the stack. Before the core begins to execute the first instruction for that ISR (in this case at clock cycle 13) an interrupt with higher priority could intervene, reset the PC and start fetching that one instead. This would not change anything on the behaviour. The lower-priority interrupt would stay pending until the higher priority interrupt has been serviced [2].

Interrupt tail-chaining

Another important difference to previous ARM architectures is that Cortex-M can tail-chain interrupts. In previous ARM architectures it was necessary to pop all registers back and push them on the stack again before to start with the next interrupt handler. In a worst case scenario this can make a difference of up to 30 clock cycles as can be seen in Figure 7 below.

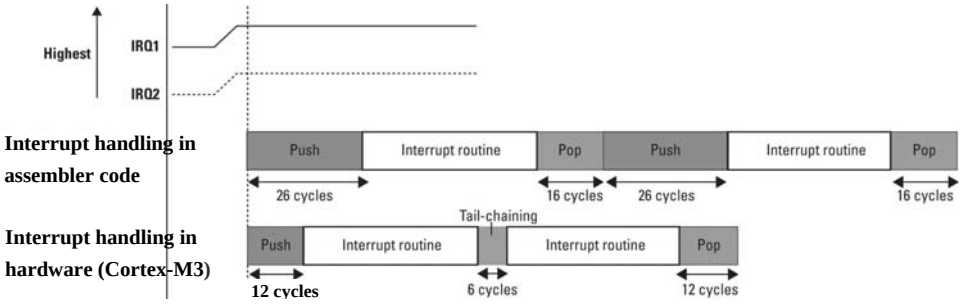

Figure 7: Tail-chain interrupt

3 Existing Cores, Design Flows and License Policies

ARM has announced support for many FPGA devices, including those from Actel, Altera, Lattice and Xilinx. As FPGA technologies strongly differ among the vendors (SRAM vs. Flash, LUT4 vs. LUT5 or LUT6, etc.) ARM provides optimized processor cores for each FPGA vendor. This is done by delivering the core in two parts: 1) A high-level generic model of the processor and 2) a lower-level device-specific layer. During synthesis the higher level model is mapped to a device-dependent lower-level design.

Up to now, optimized lower level designs are available for Actel and Altera devices. In the following we will give a short overview of the existing hardware and design environment.

3.1 Actel

Actel provides the Cortex-M1 for their ProASIC, FUSION and IGLOO devices. As other soft-cores, the Cortex-M1 has several options for configuration, which are summarized in Table 2. The actual release (v2.0) has been configured for minimum area and debug, i.e: 1) TCMs are not available, 2) small multiplier is used, 3) no OS support extension, 4) little endian is supported.

Having this configuration and synthesizing the core to run at 70 MHz, about 4400 tiles are used for the core and an additional 5000 tiles are required if debug is included. These values remain nearly constant for the different technologies. The CoreMP7 (ARM7-TDMI), running at 30 MHz maximum, uses about 6400 tiles, which is about 50% larger than the Cortex-M1. As an example, Table 3 compares the performance and area utilization for the different devices.

70 MHz is the maximum achievable speed for Actel's Flash based devices. Actel decided to use the small multiplier as this allows synthesizing the core to run at higher clock rates. Thus the overall throughput is higher than using the fast multiplier.

TCM	Can be varied in steps of to 2 kByte from 0 to 1024 kBytes.
NVIC	1, 8, 16 or 32 reprioritizable interrupts are supported
OS extension	OSs can be supported by including the SVCall instruction, two stack pointers (main and process) are present and the timer module
Endianess	Big endian (BE8) and little endian are supported
Multiplier	A standard 32 Bit h/w multiplier (three cycles) or a smaller, lower performance multiplier (33-cycles) can be chosen
Debug	Debug subsystem can be excluded

Table 2: Options for configuration

Device	Performance MHz	Size (Tiles)	Size (Tiles) RVDS	Size (Tiles) Flash Pro	RAM Blocks	Device	Utilization
M1 Fusion							
No debug	67,5	4,452	-	-	4	M1AFS600	31%
With debug	66,9	-	9,272	9,462	4	M1AFS600	68%
M1 IGLOO 1.2 V							
No debug	24	4,435	-	-	4	M1AGL600	31%
With debug	24	-	9,250	9,440	4	M1AGL600	68%
M1 ProASIC3/E							
No debug	69,9	4,435	-	-	4	M1A3PE1500	31%
With debug	66,5	-	9,250	9,440	4	M1A3PE1500	68%

Table 3: Comparison of performance and tiles utilization

Design environment and development boards

Actel offers its CoreConsole IP Deployment Platform, SoftConsole program development environment and Actel LiberoIDE. These tools can be download from the Actel webpage and be used without fee. CoreConsole is used as platform builder for integrating and configuring of IP-blocks. CoreConsole provides a couple of IP-blocks royalty-free, e.g. timer, watchdog, PWM and GPIO modules. Actel uses AMBA as bus standard, thus a huge range of available third party components can easily be integrated. Libero is Actel's FPGA development tool for Windows and Linux. It supports code simulation (ModelSIM AE), sythesis (Synplify) and back-end tools (Designer). SoftConsole is an Eclipse based software development environment. At present Code Sourcery GNU tools are supported, but newer releases will support ARM RealView tools, too.

At the moment, Actel offers two evaluation boards. The M1 Fusion Development Kit includes a board with an M1AFS600 device, Actel Libero IDE Gold, CoreConsole, SoftConsole, and optional FlashPro3 programmer. It enables application development with Cortex-M1 in mixed-signal M1 Fusion devices. The board provides an USB interface as a PC interface, which is used for download and debugging. Also included are 16MByte Flash and 1MByte SRAM. In addition, the board provides current measurement functionality and two 40-pin GPIO headers.

The M1 ProASIC3 Development Kit includes a board with an M1A3P1000 device and appropriate development tools. The board enables application development with Cortex-M1 in Actel's nonvolatile M1 ProASIC3 devices, and supports ISP, device serialization, and FlashLock on-chip

system security. 2 MByte Flash and additional SRAM are available. In addition several interfaces as PCI, Ethernet, RS232, RS485, SMBus and LCD can be used for prototyping development.

For a quick start, Actel provides several web-casts and design examples, which explain the steps for soft-processor design using Actel design tools. Nevertheless, some of these examples use the CoreMP7 and have to be modified for the usage of the Cortex-M1. Additional examples will be found on Doulos' Know How web-pages [6].

As the CoreMP7, the Cortex-M1 is free of license fees or royalties. This clearly distinguishes Actel from other vendors.

3.2 Altera

ARM is offering a development kit, which uses Altera's Cyclone III FPGAs. The kit has been announced for Q4/2007 and will be distributed by Arrow Electronics. As Altera has a SRAM based technology that is inherently faster than Actel's flash-based technology the processor will run at higher clock rates. Altera announced that the Cortex will run at 100 MHz using 2600 Logic Elements. The used core configuration is not fully explained, but it can be assumed that a minimal configuration was used.

The development kit will include Quartus II and SOPC Builder, Altera's FPGA design tools. The core is provided as a SOPC Builder megafunction, that supports the SOPC Builder's system interconnect fabric (formerly known as Avalon switch fabric) for quick and easy integration. Thus, the Cortex will be connected via an appropriate bridge to Altera's system bus.

The development kit includes the ARM RealView Microcontroller Development Kit. Thus ARM RealView Compilation Tools or the Keil µVision 3 development environment can be used for software development. Version 3.1 of the RealView Kit provides a complete instruction set system model of the Cortex-M1. In addition it is fully compatible with the RealView CREATE family of electronic system level design tools and models, enabling concurrent hardware/software development based on cycle accurate virtual prototypes. For a quick-start, a tutorial and example hardware and software design projects are provided.

Altera users have to obtain an ARM license (click-through End User License Agreement). The license gives a 1-year right to use the Cortex-M1 processor in new designs, and a free 1,000 unit royalty grant.

4 Comparison to Other Cores

ARM's Cortex M1 is competing with other FPGA-optimized embedded processors. Altera has the 32-bit Nios II processor family, Xilinx the new MicroBlaze v5.0. In terms of performance or throughput newer versions of these processors outperform ARM's Cortex-M1 (cf. Table 4). The Cortex-M1 isn't really a huge improvement over the 12 year old ARM7TDMI. The Cortex-M1 still uses a three-stage pipeling and does not support branch prediction, for example. In contrast, Altera Nios II/f has a six-stage pipeline, supports caches and local memory, has dynamic branch prediction and can be synthesized to run at up to 205 MHz. Thus, the Nios clearly has the more sophisticated architecture.

Comparing the pricing and license model, Nios and Microblaze also seem to be the better alternative. License cost of these processors is $495, including software-development tools. Additionally, they are royalty free.

Summing up, performance and pricing are not the reasons for choosing ARM Cortex-M1. The main reason is that there is a comfortable migration path to other Cortex processors which are optimized for ASIC designs. Software written and optimized for the M1 will run on other Cortex processors without modification. Although it's possible to synthesize a Nios or Microblaze for an ASIC, those processors aren't designed for that purpose and will show a limited performance compared to Cortex processors. In addition ARM's 32-bit processors are the industry standard. Thus, a number of high quality development tools, operating systems and IP-blocks are available, which help to reduce development time. It has also been mentioned that ARMs Cortex-M1 is available for all industry relevant FPGA technologies. Synthesizing Altera's NIOS or Xilinx MicroBlaze to other FPGAs is not allowed, so developers are locked to a specific FPGA vendor.

Feature	ARM Cortex-M1	Altera Nios II/f	Altera Nios II/s	Altera Nios II/e	Xilinx MicroBlaze v5.0	Xilinx MicroBlaze v4.0
Architecture	ARMv6-M	Nios II	Nios II	Nios II	MicroBlaze	MicroBlaze
Primary FPGA Targets	Fusion ProASIC, Stratix, Virtex-4/5, Cyclone, Spartan	Statix, Cyclone, HardCopy	Statix, Cyclone, HardCopy	Statix, Cyclone, HardCopy	Virtx-5	Virtex-4, Spartan-3E
Configurable ISA	No	Yes	Yes	Yes	No	No
Pipeline Depth	3-stages	6-stages	5 stages	1 stage	5 stages	3 stages
I/D-Cache	No	0-64K	0-64K	No	0-64K	0-64K
Local Memory	0 or 2 1K-1024k each	0-8 configurable	0-4 configurable	No	0-2 256K each	0-2 128K each
32-Bit Multiplier	Two options	Optional	Optional	No	Optional	Optional
32-Bit Divider	No	Optional	Optional	No	Optional	Optional
Barrel Shifter	Yes	Optional	Optional	No	Optional	Optional
FPU	No	Optional 32-Bits	Optional 32-Bits	Optional 32-Bits	Optional 32-Bits	Optional 32-Bits
Branch Prediction	No	Dynamic	Static	No	No	No
Privilege Levels	1	2	2	2	1	1
Max. Core Frequ.	72 MHz (Actel), > 170 MHz (Xilinx, Altera)	205 MHz	165 MHz	200 MHz	220 MHz	205 MHz
Max. Int. Perf.	0.8 DMIPS/MHz	225 DMIPS	127 DMIPS	31 DMIP	240 DMIPS	166 DMIPS
Max. FP Perf.	n/a	n/a	n/a	n/a	50 MFLOPS	33 MFLOPS
FPGA Logic Cells	4300+ LUT3 tiles (~1900 LUT4 cells)	1800	1150	600	960-1700	950-2400
Price	Free (Actel)	$495	$495	$495	$495	$495

Table 4: Comparison of Cortex-M1 to Nios II, LEON3 and Xilinx (taken from [3])

5 Summary

ARM's new Cortex-M1 processor follows a trend in the semiconductor industry. Costs for ASICs are rising while unit prices for FPGAs are falling and gate counts are growing. Up to now, FPGA developers were limited to using Altera Nios or Xilinx MicroBlaze processors. Migration to ASIC design is difficult for these processors. ARM's Cortex-M1 is supported by multiple development tools, operating systems and IP modules. The core is available for different FPGA technologies. As described in the paper, migration from Cortex-M1 to M3 is possible with minimal effort. With this, ARM's Cortex-M processors allow a simple upgrade of designs and migration to ASICs.

In addition, migration from earlier ARM7 based microcontrollers to Cortex-M based design is simple. C code and Thumb assembler will run on the Cortex-M without modification. ARM assembler code (usually used for exception handlers) can not be executed on the Cortex-M. Nevertheless, exception handlers have to be re-written anyway, as the exception handling has been modified.

We suppose that other competitors will follow ARM. Meanwhile, Altera and Xilinx may provide versions of the Nios and MicroBlaze core optimized for ASIC design. Altera has already started its HardCopy initiative, allowing developers to port the Nios II to a structured ASIC. On the other side, competitors like ARC, MIPS and Tensilica can be expected to follow ARM in providing FPGA optimized versions of their cores.

Literature

[1] ARM, Technical Reference Manual Cortes M1, M3
[2] Yiu, The Definitive Guide to the Arm Cortex-M3, Butterworth Heinemann, 2007
[3] Halfhill, Tom R., ARM Blesses FPGAs, Microprocessor Report, March 19, 2007
[4] Micheal Barr, Programming Embedded Systems in C and C++, O'Reilly, 2006
[5] Actel web portal, http://www.actel.com
[6] Doulos web portal, http://www.doulos.com/ARM
[7] ARM FPGA web porta, http://www.arm.com/fpga
[8] Altera web portal, http://www.altera.com/products/ip/processors/32_16bit/m-arm-cortex-m1.html