Cortex-M0 Implementation on a Xilinx FPGA

Pedro Ignacio Martos y Fabricio Baglivo

Laboratorio de Sistemas Embebidos
Facultad de Ingeniería - UBA
Buenos Aires, Argentina
pmartos@fi.uba.ar / baglivofabricio@gmail.com

ABSTRACT

This paper presents an implementation of the recently available ARM Cortex ^{TM}-M0 soft core processor on a small Xilinx FPGA. These kinds of processors are oriented to mixed-signal and micro- controller devices with low cost and low power consumption. At present, Xilinx does not offer a Cortex-M0 soft core solution. By contrast, Actel and Altera offer Cortex^{TM}-M1 cores in FPGA solutions. The aim of this work is to connect the ARM Cortex-M0 soft core with a synthesized code memory using the Advanced Microcontroller Bus Architecture (AMBA$^{®}$) AHB-Lite interface in a Xilinx FPGA to evaluate how much fabric resources are needed to implement this soft core processor. For this purpose, we have designed an small system that receives the data request from the processor, sends it to the synthesized memory, and returns the data obtained to the processor. In this implementation, the proper work of the system is monitored with Chipscope-Pro, a Xilinx in-system debugging tool.

1. INTRODUCTION

Within the 32-bit ARM processor family, shown in Figure 1, Cortex-processors support high performance applications with embedded operating systems and real time applications. Cortex^{TM}-R processors are oriented toward low-cost controllers, with deterministic, fixed-latency and interrupt handling. These processors are also intended for high-performance, real-time applications. Cortex-M is the lowest member in Cortex family. This family is an option for simple, low-cost, low-power and low-performance designs. The Cortex-M0 is, nowadays, an alternative to 8-bit microcontrollers, with the advantage of high processing capacity. It is built on a high-performance processor core, with a 3-stage pipelined Von Neumann architecture. This processor implements the ARMv6-M architecture, which supports the ARMv6-M Thumb$^{®}$ instruction set, including

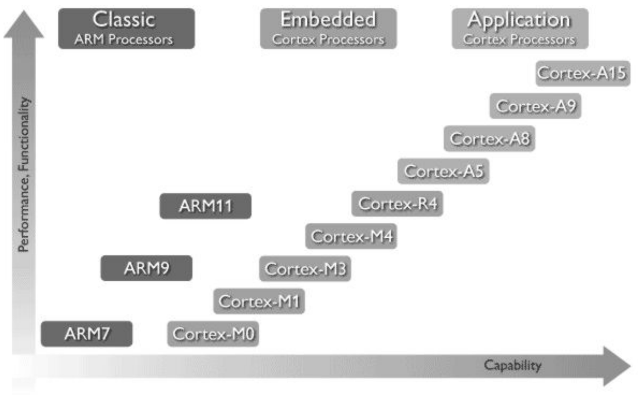


Figure 1 ARM Cortex Family

Thumb-2 technology. This provides the exceptional performance expected of a modern 32-bit architecture, with a higher code density than other 8-bit and 16-bit microcontrollers, being capable of achieving performance figures around 0.9 DMIPS/MHz..

1.1. Cortex-M0 DesignStart

The Cortex-M0 DesignStart (M0DS) processor, shown in Figure 2, is a fixed configuration of the Cortex-M0 (M0) processor. It is delivered by ARM as a pre-configured and obfuscated netlist, but it is synthesizable, Verilog version of the full Cortex-M0 processor. The main differences between them are: a) M0 has a full AMBA interface that provides MASTER and SLAVE support, while the M0DS provides an AMBA Lite interface with only MASTER support; b) M0 can be implemented with either a high speed multiplier (1 clock cycle) or a slow-speed multiplier (32 clock cycles), while the M0DS only allows the slow-speed multiplier; c)M0 can handler 32 interrupts while M0DS can handler 16 interrupts; and d) M0 includes an optional wake-up interrupt controller, architectural clock gating and hardware debugging. M0DS does not include these options.

The M0DS processor is distributed as a single zipped tar file bundle, containing the release notes, synthesized

Verilog, and test-bench code. The test-bench instantiates the Cortex-M0 DS module and connects it in a minimal way to a memory model and clock and reset generators. It also provides a means of outputting information from the processor to the Verilog simulator's console output.

The aim of this work is: synthesize the Cortex-M0_DS in a real FPGA, connect it to a memory with a small program inside using the AMBA-Lite interface, evaluate how much FPGA fabric resources are needed to do it, and see its applicability in small footprint systems.

1.2. AHB AMBA-Lite overview

This protocol defines the data and address buses, and all the control signal for high performance synthesizable designs requiring high bandwidth. First, the address bus, HADDR[31:0], is a MASTER output to a SLAVE device. Data transfers are performed using two buses: a read one, which is a SLAVE output to a MASTER input, called HRDATA[31:0], and a write one, which is a MASTER output to a SLAVE, called HWDATA[31:0]. In this work we use only HRDATA. Finally the protocol specifies control signals. HBURST[2:0] indicates if the transfer is a single transfer or forms part of a burst. HMASTLOCK indicates if the current transfer is part of a locked sequence. HPROT[3:0] provides additional information about a bus access and is primarily intended for use by any module that wants to implement some level of protection. HSIZE[2:0] indicates the size of the transfer, e.g. ,byte, half word, or word. HTRANS[1:0] indicates the transfer type of the current transfer (IDLE, BUSY, NON SEQUENTIAL, SEQUENTIAL). The HWRITE signals indicates the transfer direction, when HIGH this signal indicates a write transfer, and when LOW, a read transfer.

2. THE IMPLEMENTATION

2.1. The board

For this paper we used a Digilent Spartan 3E Starter FPGA board. This digital system design platform is built around a Xilinx Spartan 3E FPGA. It has 16 megabytes of fast SDRAM and 16 megabytes of flash ROM. It also has a 50MHz oscillator, plus a socket for a second oscillator. This board contains a USB2 port that provides board power, programming, and data transfers. Some peripherals are also included on the board like an LCD display, a set of LEDs, switches, etc. In particular, we use LEDs as event indicators. One of the major advantages of this board is that is possible to use Xilinx software tools (Impact, Chipscope Pro, xmd, etc.), downloading software (Adept) from the Digilent web page. This is very important for our purpose because it is possible to program the board and see the state of it easily.

The Xilinx S3E500-4 is the FPGA included on the board. It has 500K gates, 10,500 logic cells, 20 hardware multipliers, 360Kbits of dedicated RAM, 73 Kbits of distributed RAM, 4 clock handlers, and a maximum clock frequency of 300MHz .

2.2. Software tools

We used the Xilinx® Integrated Software Environment (ISE™) as our design suite software. The ISE project navigator allowed us to manage the project and synthesize. Core Generator is the tool we used to generate the ROM memory and the reset generator. ISIM was used for simulation purpose. Also we used Chipscope Pro to perform online debugging. This program let us see the state of the system. For C code compiling, we used the ARM Microcontroller Development Kit by Keil™.

The ARM deliverables package contains a logical folder with synthesizable code and test-bench code. The test-bench project has a Verilog implementation of the processor, Cortexm0ds_tb.v, prepared for simulation. It also includes a HelloWorld.c program with a make file containing the compilation parameters. As result of this compilation a .bin file is obtained. This file is a memory

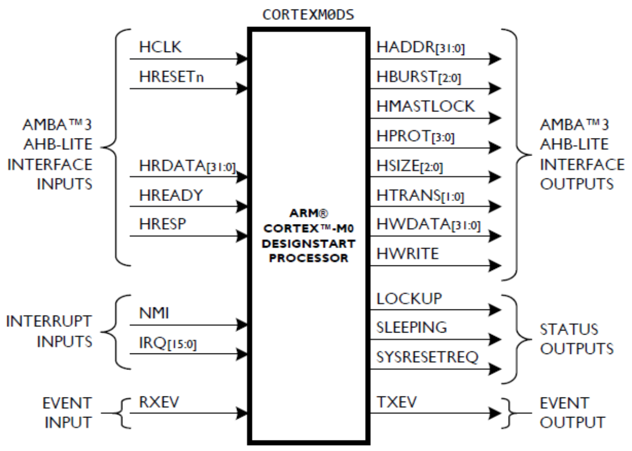


Figure 2 Cortex-M0 implementation

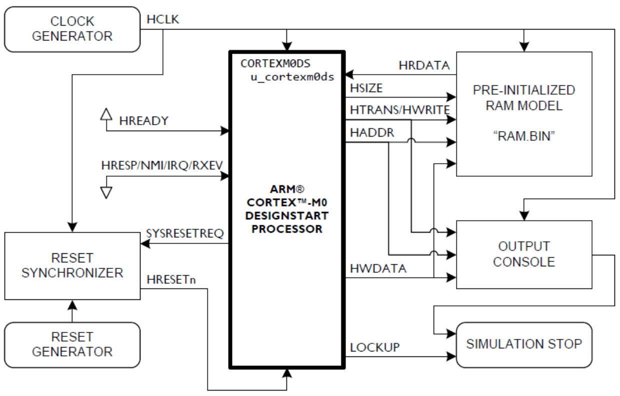


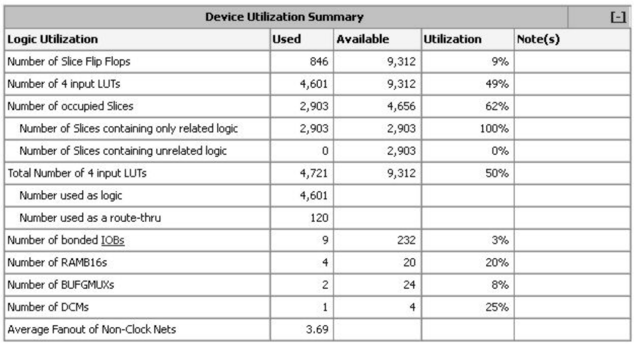
Figure 3 Cortex M0 Design test-bench Schematic

image that is loaded by the Verilog test-bench at the beginning of the simulation.

2.3. System Implementation

The first step of the implementation process was the replication of the Hello World project. We realized that the makefile did not work correctly with Keil or IAR, so we decided to begin a new project with Keil. We made the compilation process and we compared the .bin file obtained with the one ARM provide in the test-bench package. After that, we began the simulation with ISIM. In it we saw that the data bus did not contain valid values. The .bin file was not correctly loaded by the Verilog code because it was assumed that the Verilog instruction $fread() made double words accesses, where it actually made byte accesses. After this modification the test-bench was working as expected.

The next step was the implementation of the testbench code into a synthesized VHDL code (see Figure 3). An external 50MHz oscillator was used as the external clock. We used a synthesized DCM to generate the 10Mhz system clock. The reset generator was implemented using the Xilinx architecture wizard. A two clock-cycles reset pulse is needed to initialize the processor. The pre initialized 1Kx32bit RAM was created using the Core Generator tool. The processor does 32 bits data access, so we had to shift the RAM address bus in 2 positions, so processor addr[0] is connected to RAM addr[2]. Some LEDs were connected to internal signals, namely, LOCKED and SLEEPING. The address and data buses of the AMBA-Lite interface and HWRITE were connected directly to the memory. HREADY was fixed to '0' and HRESP was fixed to '1'. Others signals of the interface were connected to internal signals for debugging purpose. A bus signal detector was developed to compare HWRITE bus information with two patterns, one to turn the LED on and the other one to turn it off. The user constraint file (.ucf) was defined using the Plan Ahead tool



Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Flip Flops	846	9,312	9%	
Number of 4 input LUTs	4,601	9,312	49%	
Number of occupied Slices	2,903	4,656	62%	
Number of Slices containing only related logic	2,903	2,903	100%	
Number of Slices containing unrelated logic	0	2,903	0%	
Total Number of 4 input LUTs	4,721	9,312	50%	
Number used as logic	4,601			
Number used as a route-thru	120			
Number of bonded IOBs	9	232	3%	
Number of RAMB16s	4	20	20%	
Number of BUFGMUXs	2	24	8%	
Number of DCMs	1	4	25%	
Average Fanout of Non-Clock Nets	3.69			

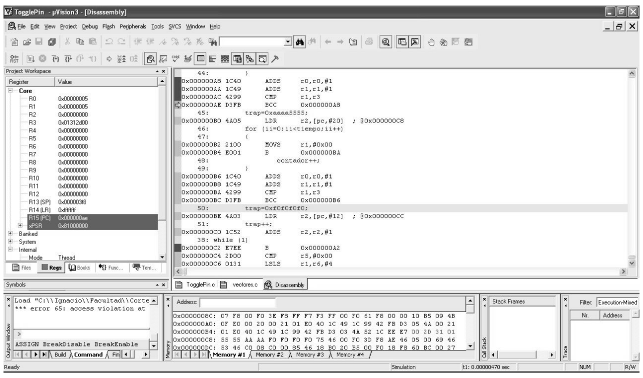
Figure 4 FPGA use with the Project Implementation.

A "Toggle LED" project was created. Its aim was to turn on and off one of the board´s LEDs with the Cortex processor. The .bin file was generated using the Keil tools. The Core Generator tool allowed us to fill the memory with an image. The file format required was .coe. A script was generated to transform the .bin file in a .coe. Once the project was synthesized, IMPACT was used to program the board.

3. RESULTS

As mentioned before, the Chipscope Pro package was used to see the transitions on the AMBA AHB-Lite interface signals and for debugging. With this tool, we could verify that the interface worked as expected. All memory accesses were correctly synchronized and we saw that the LED on the board blinks. So we conclude that the implementation is correct and functional.

In Figure 4 we show some results using the FPGA with the project implementation. Figure 5 shows the simulation of the memory accesses in Keil software. Figure 6 shows the Chipscope Pro capture of the AMBA bus. Timing reports (post place & route) gave a maximum clock speed of 40MHz. This value could be improved by implementing time and placement constraints.


TogglePin - µVision3 - [Disassembly]
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Project Workspace
Files Regs Books Func... Tem...
Symbols
TogglePin.c vectores.c Disassembly
Build Command Fin
Memory #1 Memory #2 Memory #3 Memory #4
Stack Frames
Filter: Execution-Mixed
Nr. Address
Ready Simulation NUM R/W

Register	Value
Core	
R0	0x00000005
R1	0x00000005
R2	0x00000000
R3	0x01312d00
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x000003f8
R14 (LR)	0xffffffff
R15 (PC)	0x000000ae
xPSR	0x81000000
Banked	
System	
Internal	
Mode	Thread

```
    44:          }
0x000000A8 1C40      ADDS     r0,r0,#1
0x000000AA 1C49      ADDS     r1,r1,#1
0x000000AC 4299      CMP      r1,r3
0x000000AE D3FB      BCC      0x000000A8
    45:          trap=0xaaaa5555;
0x000000B0 4A05      LDR      r2,[pc,#20]  ; @0x000000C8
    46:          for (ii=0;ii<tiempo;ii++)
    47:          {
0x000000B2 2100      MOVS     r1,#0x00
0x000000B4 E001      B        0x000000BA
    48:              contador++;
    49:          }
0x000000B6 1C40      ADDS     r0,r0,#1
0x000000B8 1C49      ADDS     r1,r1,#1
0x000000BA 4299      CMP      r1,r3
0x000000BC D3FB      BCC      0x000000B6
    50:          trap=0xf0f0f0f0;
0x000000BE 4A03      LDR      r2,[pc,#12]  ; @0x000000CC
    51:          trap++;
0x000000C0 1C52      ADDS     r2,r2,#1
    38: while (1)
0x000000C2 E7EE      B        0x000000A2
0x000000C4 2D00      CMP      r5,#0x00
0x000000C6 0131      LSLS     r1,r6,#4
```

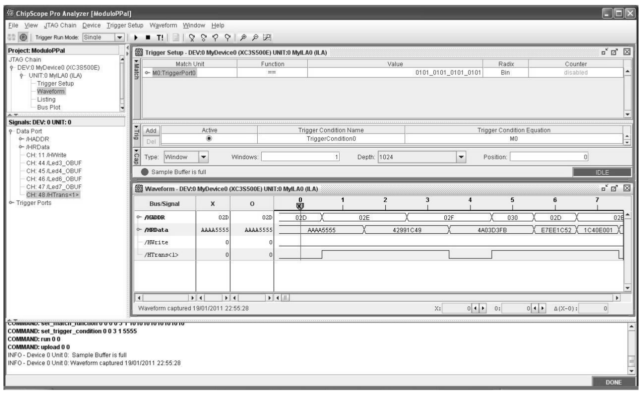
```
Load "C:\\Ignacio\\Facultad\\Corte
*** error 65: access violation at

>
ASSIGN BreakDisable BreakEnable
```

```
Address:
0x0000008C: 07 F8 00 F0 3E F8 FF F7 F3 FF 00 F0 61 F8 00 00 10 B5 09 4B
0x000000A0: 0F E0 00 20 00 21 01 E0 40 1C 49 1C 99 42 FB D3 05 4A 00 21
0x000000B4: 01 E0 40 1C 49 1C 99 42 FB D3 03 4A 52 1C EE E7 00 2D 31 01
0x000000C8: 55 55 AA AA F0 F0 F0 F0 75 46 00 F0 3D F8 AE 46 05 00 69 46
0x000000DC: 53 46 C0 08 C0 00 85 46 18 B0 20 B5 00 F0 18 F8 60 BC 00 27
```

```
t1: 0.00000470 sec
```

Figure 5 Simulation in Keil



```
COMMAND: set_match_function 0 0 0 0 3 1 1010101010101010
COMMAND: set_trigger_condition 0 0 3 1 5555
COMMAND: run 0 0
COMMAND: upload 0 0
INFO - Device 0 Unit 0:  Sample Buffer is full
INFO - Device 0 Unit 0: Waveform captured 19/01/2011 22:55:28
```

Figure 6 Capture of the AMBA Lite bus

4. CONCLUSIONS

The most remarkable conclusion is that it is possible to implement the M0DS in a low range FPGA. With this result, Xilinx, Actel and Altera (the three most important FPGA manufacturers) can support this core making it a considerable alternative when portability between these three FPGA types is a requirement for a design.

As an improvement, it would be useful to have a complete test bench that allows us to generate the bin file from the source code. That is not possible right now, and it would accelerate the development time.

In future work, other peripherals will be connected to the AMBA bus in order to increase the processors capacity. Also, going further, a Linux operating system can be investigated on this processor, obtaining a Linux implementation in a small footprint design.

I. REFERENCES

[1] ARM Ltd, "ARM DDI 0419C ARMv6-M Architecture Reference Manual", September 2010.

[2] ARM Ltd,, "ARM IHI 0033A AMBA 3 AHB-Lite Protocol V.1 Specification", June 2006.

[3] ARM Ltd, "AT510-DC-80001-r0p0-00-rel0 ARM Cortex M0 DesignStart Release Note" August 2010.

[4] ARM Ltd, "ARM DDI 0432C Cortex M0 Revision r0p0 Technical Reference Manual", November 2009.

[5] ARM Ltd, "ARM DUI 0497A Coertex M0 Devices Generic User Guide", October 2009.

[6] Xilinx, "DS312 Spartan-3E FPGA Family: Datasheet", August 2009.

[7] Digilent, "Digilent Spartan 3E Starter Kit Reference Manual", June 2008.

II. ACKNOWLEDGES

To William Hohl, Joe Bungo, Fiona Cole, the people at the AR University Program and the people at the Xilinx University Program (XUP) for their support and cooperation.

III. TRADEMARKS AND COPYRIGHTS

The information about ARM processor families was mainly extracted from ARM Ltd web site (www.arm.com), as published on October, 2010.