A Methodology for Prototyping Flexible Embedded Systems

John Sachs Beeckler and Warren J. Gross
Department of Electrical and Computer Engineering,
McGill University
Montreal, Quebec, Canada
e-mail: john.beeckler@mail.mcgill.ca, warren.gross@mcgill.ca

***Abstract*— A methodology is presented for prototyping flexible, custom embedded systems using low–cost reconfigurable hardware, open–source FPGA–ware, and open–source software. This is done through a case study of the complete design of a custom embedded system. The system is intended as a low–cost, flexible, and reconfigurable platform with multimedia and networking capabilities, for prototyping interactive entertainment applications. The system features an FPGA system–on–chip design supported by a custom board with a versatile set of interfaces including Ethernet, color television, stereo audio and others. The project contains work in three main areas: board design, HDL chip design, and software.**

I. Introduction

This paper examines a methodology for prototyping flexible, custom embedded systems using low–cost reconfigurable hardware, open–source FPGA–ware, and open–source software. This is done through the case study of the complete design of a custom embedded computer system. The system, shown in Figure 1, is intended as a low–cost, flexible, and reconfigurable platform with multimedia and networking capabilities, for prototyping interactive, entertainment applications. The system features a FPGA system–on–chip design based on the open–source LEON microcontroller [1], supported by a custom board with a versatile set of interfaces including Ethernet, color television, stereo audio, analog game controllers, and others. The work concerns three main areas: board design, HDL chip design, and software. The open–source design methodology is described in Section 2. The design specifications are introduced in Section 3. Section 4 discusses the system design. Section 5 presents the FPGA synthesis results, and Section 6 discusses the software configurations. Conclusions are presented in Section 7.

II. A Flexible Design Methodology

We used a flexible, "open–source" design methodology at all three levels. Wherever possible, we used open–source software and VHDL hardware models instead of proprietary solutions. There are several resources for top quality, ASIC proven, open–source HDL designs, or "cores". These range from 8–bit microcontrollers to 64–bit DSPs, and hardware square root units to video compression. OpenCores.org is one such organization providing freely available, freely usable and re–usable open source hardware. OpenCores.org has a selection of verified open–source hardware designs including

Fig. 1. Prototype system.

microprocessors, complete systems–on–chip, arithmetic cores, communication cores, encryption cores, coprocessors, error–correction encoders and decoders, and even board layouts. There are several advantages of development using open–source HDL designs, including:

- Independence from technology (FPGA and ASIC), vendors, boards, and tools
- ASIC prototyping,
- Support and information available from an active community of users participating in continued development,
- Transparency of design,
- Blocks are truly configurable, alterable, and customizable,
- Software support (open–source drivers and operating system ports).

Open–source HDL is generally designed for independence. A perfect example of this is the LEON open–source HDL library from Gaisler Research. All technology dependent features of the LEON are isolated and contained in a special, separate VHDL package. All one has to do to investigate how certain features are implemented for a particular technology is study the corresponding entry of this package. Similarly, porting the LEON to a new FPGA or ASIC technology is as simple as specifying how each entry of this package should be implemented. Open–source libraries such as the LEON or

0840-7789/07/$25.00 ©2007 IEEE

OpenRISC are ideal for prototyping ASIC systems using an FPGA. The logic design itself is independent of any FPGA and can be either directly, or with little effort, modified for ASIC fabrication. This is generally not possible with "IP" (proprietary closed designs) which target a specific FPGA technology. The configurability provided by the open–source nature of these cores greatly simplifies the porting of designs for use on new boards and in new systems and applications. This was made clear to us when attempting to instantiate a Nios I system on the board. The Nios I is a propriety microcontroller system design for Altera FPGAs. Because the Nios I system made certain assumptions about a shared tristate bus on the board, which were true of many other boards but not true of the custom board, it was difficult to use some essential Nios peripherals. When the same problem was encountered with LEON, because we had the source code, the simple addition of three lines of VHDL to the source code solved the problem. Although this may sound like an isolated problem, it illustates the importance of having access to the VHDL of a hardware device. Another aspect to open–source design is configurability. One of the greatest strengths of an FPGA based system is the potential use of its reconfigurability for special, custom hardware. Without access to source code, then the reconfigurability of a system is limited to macroscopic or fixed customizations.

A. An Argument for Proprietary Solutions

The use of proprietary, closed–source technology, both software and hardware, is and continues to be a good solution for many engineering projects. Here we examine a few possible reasons. Proprietary cores made by the manufacturer of an FPGA are highly optimized for the technology used. Such FPGA specific cores can be optimized for area and speed, achieving higher maximum clock frequencies and much smaller FPGA areas than those of technology independent designs. An important aspect to proprietary solutions, both software and HDL cores, is that they offer isolation. In many situations isolation is exactly what is required. It is not realistic for many engineering projects to be directly involved in all aspects of design.

III. Design Specifications

The objective is to create a custom embedded system with multimedia and networking functionality, using reconfigurable hardware and open–source resources. Of particular importance is that the FPGA used be a low–end device because the target applications are consumer market oriented. The design of the system is divided into three phases:

Phase 1 of the project involves designing and building a custom circuit board (Figure 2). The first board prototype was designed, assembled, and experienced several cycles of testing, debugging, and altering for each subsystem it contains. It is a two–layer FPGA board with enough RAM and FLASH for a 32–bit embedded Linux system. It has hardware for interfacing to Ethernet, NTSC output, stereo audio output, and game controller inputs. The board features a low–cost Altera

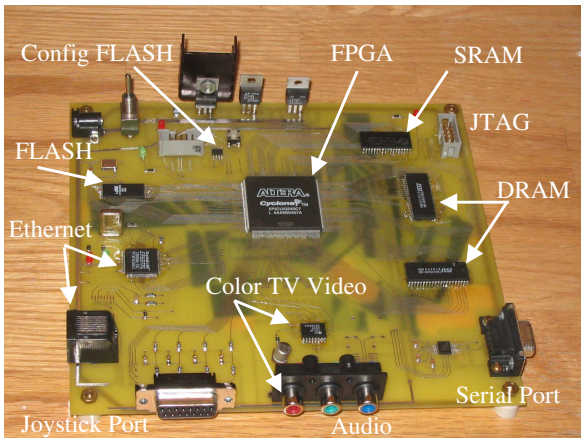

Fig. 2. The board.

Cyclone FPGA with 12,060 logic elements and 239 Kbits of embedded RAM.

Phase 2 of the project is the design of a custom system–on–chip, realized in the reconfigurable logic of the FPGA. It is based on a modified version of LEON 2, a SPARC compatible, open–source VHDL processor design. The LEON has been augmented with custom logic cores and peripherals implementing each of the subsystems and targeting board hardware. These subsystems include a custom bursting memory controller, an on–chip NTSC color video system, an audio controller, and input game controller interfaces.

Phase 3 of the project is embedded software development. Low–level software drivers for the Ethernet, video, audio, input, and flash subsystems are have been developed. The system is currently capable of running RTEMS, an open–source real–time kernel, μCLinux, an embedded Linux distribution for microcontrollers, and bare executables with interrupt handlers. Software applications and operating systems are loaded by a custom boot–loader either from FLASH, Ethernet, or a serial link. C libraries support board hardware and application software such as the fractal graphics software shown in Figure 1 demonstrate the capabilities and functionality of our system.

IV. Systems Design

The system architecture is illustrated in Figure 3. Below, we describe the main components of the design.

A. FPGA

The Altera Cyclone EP1C12 device was chosen for a number of reasons. The Cyclone is a low–cost, high–density, and feature–rich device. Altera advertises volume prices of less than $0.99 per 1,000 logic elements for the Cyclone family and $0.64 per 1,000 logic elements for the Cyclone II family, making it a realistic option for cost sensitive applications. The device chosen, the EP1C12, contains 12,060 logic elements, 239 Kbits of embedded RAM, 2 PLLs, and 249 I/O pins. It was important that the FPGA contain enough logic elements to instantiate a 32 bit microprocessor system, the system's peripheral logic set, coprocessors, and "application acceleration"

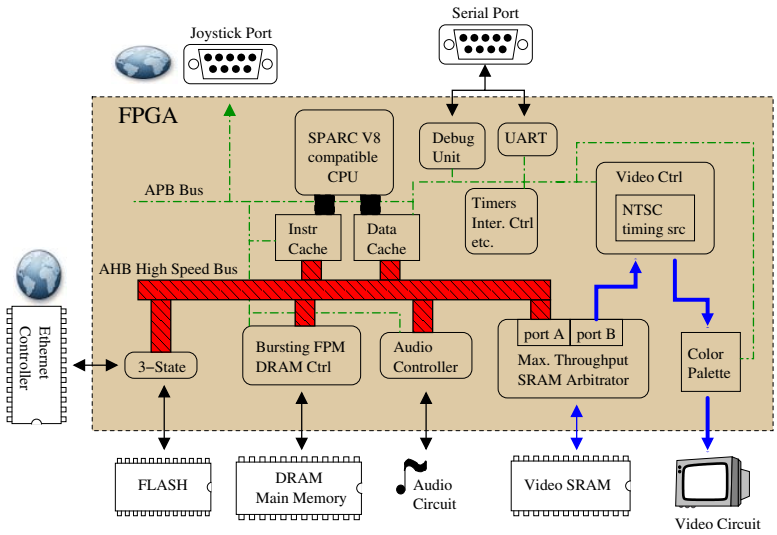

Fig. 3. System architecture overview

logic. The available logic elements and embedded RAM on the EP1C12 have proved to be sufficient. The current LEON based system–on–chip design, with all peripheral hardware uses about 79% of the logic elements. 38% of the embedded RAM is used for microprocessor caches and the video color palette. The on–chip PLLs are used to derive the system clock from the video clock. PLLs in FPGAs bring even more versatility to a design, providing flexible control over the frequencies and phases of multiple clocks domains within one design. This flexible on–chip clock management was useful in the design of a microcontroller with on–chip video logic. Another factor considered when selecting the FPGA was that, at the time, the EP1C12 was the largest Cyclone device available in a package other than a BGA (ball grid array). BGA packages can be difficult to work with manually, making debugging, testing, manual board changes, and assembly problematic. Equally important are the development tools available for an FPGA. The Cyclone family is fully supported by the free edition of the Quartus FPGA development suite. This means that anyone, given a board, can download tools and start developing immediately without purchasing expensive development software. This is critical if a board is intended to be "user–programmable".

B. Memory

A small configuration FLASH stores data for configuring the FPGA with a system design. This configuration FLASH is programmed via a special port. The FPGA can also be programmed directly using the JTAG port. A 512 KB NAND FLASH is used for storing a boot–loader, software, and data. NAND FLASH has the desirable feature of "RAM–style" read accesses, allowing the "in–FLASH" execution of program code. The device used has a lifetime of 10,000 program–erase cycles, making it acceptable for use as a data storage device. 4 MB of 32–bit wide, fast–page–mode DRAM forms the main memory, a sufficient size for μCLinux, or RTEMS applications, and large when compared to some popular hand–held systems. In addition, 512 KB of independently accessible, low–latency SRAM, on a separate bus, make a special memory for use as either a video frame buffer, or a dedicated coprocessor memory. The FPGA internal embedded RAM is used for microprocessor caches, a color palette for the video system, and special peripherals.

C. Video

An essential board–level hardware requirement for video applications is an independently accessible, low–latency memory for use as a frame buffer. This is provided by the SRAM. A logic video system, in the FPGA, is included as part of the microcontroller's on–chip peripherals. However, special board hardware is necessary for synthesizing the analog signal. The composite NTSC color video signal contains three analog levels: luminance Y , and two color difference signals, U and V . The two color difference signals, U and V are used to modulate a high frequency color carrier of 14.318180 MHz, at 0^o and 90^o phase difference. These high frequency color waves are then added to the luminance signal. Finally, synchronization pulses and color carrier pilot bursts are added to the signal. Video system logic in the FPGA creates signal timing and video content, originating from the frame buffer. Output from the FPGA includes digital RGB values (red, blue, and green) and digital synchronization pulses. Analog RGB voltages are synthesized by passive circuity, resistor networks and capacitors, using a pulse–width–modulation technique for higher color resolution. These analog RGB levels and synchronization pulses are fed to the AD724 chip, which performs the necessary RGB–to–YUV conversion, color modulation, sync–injection, and filtering, finally creating a NTSC composite video signal.

D. Audio

Analog levels for both left and right audio channels are created from the FPGA via pulse–width–modulation, using

TABLE I

FPGA RESOURCE UTILIZATION

Logic elements	9627/12060(79%)
Memory bits	91392/239616 (38%)
Registers	3428/12567 (27%)
Combinational Functions	8781
Interconnect Usage	50%

two low–pass RC filters, and buffered by single supply op–amps. A logic audio controller or coprocessor can be included in the FPGA.

E. Ethernet

The trend with highly–integrated, embedded microcontrollers, is to include on–chip Ethernet MAC (media access control) peripherals. The world of FPGA microcontroller systems seems to be following that trend. OpenRISC from OpenCores.org, LEON from Gaisler Research, Microblaze from Xilinx, and Nios from Altera are all FPGA systems–on–chip which include Ethernet controllers as on–chip peripherals. However, when looking at real boards and systems based on these designs, one realizes that within a certain range of systems, the ideal single chip system is not yet a reality. At present, a full–featured embedded system cannot be made without at least 4 major chips. Other than the FPGA or ASIC itself, external RAM, external non–volatile memory for both software and programming an FPGA, and finally Ethernet are required. No matter what logic the system–on–chip contains, todays digital FPGAs require an external interface for any analog subsystem such as Ethernet. In order to save valuable FPGA resources for other systems, an external, all–in–one Ethernet IC is used. The CS8900A is a complete analog and digital Ethernet solution, containing an Ethernet MAC engine, 4 KB of integrated buffer memory, with a complete analog front end for 10Base–T. The FPGA interfaces directly to the CS8900A, which connects to an RJ–45 connector through isolation transformers.

V. FPGA SYNTHESIS

The chip design was synthesized for the Cyclone EP1C12Q240C7 FPGA at a system frequency of 45 MHz. Table I indicates the FPGA resource utilization data for the compiled design. Figure reffig:fpga shows a floorplan view of the FPGA compilation, highlighting areas of about 75% routing congestion. Of the 9,600 logic elements used, approximately 7,900 were used by LEON cores, which includes the processor, buses, caches, and basic peripherals. Of the 7,900 logic elements used by LEON cores, 3,278 were used for the CPU integer unit. Thus far, custom peripherals have only added 1,700 logic elements to the LEON, and we can therefore look forward to using the remaining logic elements, 148,000 bits of embedded RAM, and 9,000 registers to add future hardware blocks.

VI. SOFTWARE

Software development for the system is done using open–source libraries, kernels, and the GNU SPARC tool–chain. Our

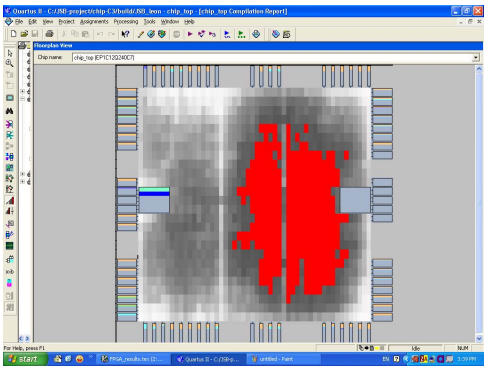

Fig. 4. FPGA routing congestion.

Fig. 5. Multimedia Software Application.

system's hardware is completely supported by custom drivers and libraries for both the μCLinux and RTEMS operating systems. Currently, applications can be developed using the following software environments:

- "Bare Metal"– Stand–alone executables with interrupt handlers, but no kernel or OS,
- RTEMS – An open–source, real–time embedded operating system for embedded and multiprocessor systems,
- μCLinux – A Linux configuration for micro–controllers without virtual memory.

A custom bootloader loads operating systems and applications from FLASH or the network interface using TFTP. Real multimedia software applications, such as the original video game shown in Figure 5, demonstrate the system's capabilities.

VII. CONCLUSIONS

A flexible methodology was explored for prototyping custom embedded systems using low–cost reconfigurable hardware and open–source FPGA–ware. We presented a case study of a embedded system with multimedia and networking capabilities for entertainment applications.

REFERENCES

[1] *LEON2 Processor User's Manual,* http://www.gaisler.com/doc/leon2-1.0.27-xst.pdf