Embedded Processors on FPGA: Soft vs Hard

Vivek Jayakrishnan
School of Engineering
Grand Valley State University
Grand Rapids, MI, 49504
Email: vivekjay14@gmail.com

Chirag Parikh
School of Engineering
Grand Valley State University
Grand Rapids, MI, 49504
Email: parikhc@gvsu.edu

Abstract

In the past, Field Programmable Gate Arrays (FPGAs) were primarily used for prototyping and debugging purposes. However, with their increased popularity, many commercial products now incorporate FPGAs. In the late 1990s, FPGA vendors introduced System-on-chip (SoC) devices that included one or more hard-core processors and an FPGA fabric on a single integrated circuit to allow for more complex designs that involved hardware and software co-integration. While this approach provides advantages of running your design at much higher speeds it does not provide the flexibility of modification to suit the application. Because of this, many FPGA vendors provide the solution of using soft-core processors that is configured from logic resources inside the FPGA. Depending on several factors, a designer can choose either Hard or Soft processor in his design depending on the application. This paper presents an in-depth performance analysis and direct one-on-one comparison between the two. For this task two different platforms, one housing a hard-core processor and another housing a soft-core processor is chosen to run a digital oscilloscope application. This is then used to measure the FPGA resource utilization, execution speed and power consumption on each.

Keywords — FPGA, System-on-Chip, ARM, VHDL, Verilog

I. Introduction

FPGA (Field Programmable Gate Arrays) are reprogrammable silicon chips that rewire themselves to implement user's functionality rather than just run a software application from memory like a processor. The term *field programmable* in the name implies that the user in the field can reconfigure the chip's hardware for specific applications. FPGAs consist of mixes of configurable static Random Access Memory (SRAM or Flash), high-speed input/ output pins (I/O), logic blocks, and routing. Programmable blocks called Configurable Logic Blocks (CLB) along with reconfigurable interconnects, that allow CLBs to be physically connected to one another, are the main components of an FPGA.

The development and drop in price of semiconductors and electronics in general has slowly blurred the lines between FPGAs and microprocessors by combining the two in a single package with more flexibility. FPGAs offer several advantages over ASICs speed, reliability and flexibility. However, we face a trade-off by only using an FPGA for processing and I/O connectivity in the system. FPGAs do not have the driver ecosystem and code/IP base that microprocessors and Operating Systems (OS) do. Microprocessors coupled with OS provide the foundation for file structures and communication to peripherals used for essential tasks. To tackle this a hybrid architecture has emerged in which a microprocessor is paired with an FPGA.

This can be done in two ways. The first one is embedding a hard core, by having a dedicated block on the FPGA silicon. The second one is the so-called soft-core where the implementation of a processing core is dynamically configured on the FPGA.

FPGA designers face a dilemma in choosing either Hard-core or Soft-core processor for their design. Each approach comes with its own pros and cons. In this paper, We talk about an application that was developed and implemented on both; a hard-core processor based FPGA system and a soft-core processor based FPGA system. The paper then compares the two approaches based on several important factors such as performance, power consumption and resource utilization.

The primary objective of this paper is to compare and contrast both the approaches and suggest conditions for choosing one approach over the other. In the next section, we talk about other studies on this topic. Then in section III software design approach is described in brief followed by a section on hardware design. Section V is where the experimental results are discussed. The last section concludes the paper with our findings and suggestions.

II. Related Works

Before we present the results from our experiments, let us look at some of the work published in the area of comparison between hard processor and a soft processor. In their paper, Martos and Baglivo[2] showed the result of implementing the Cortex M0 Design Start soft-core processor on a low-end FPGA from Xilinx. The processor was simulated on test bench and then successfully tested with an LED toggling application. Mondragon and Christman in their paper[3] compared a soft-core processor with an actual micro controller. The paper highlights the trade-off that both methodologies can offer. Both methods are compared on the basis of environment, visibility to internal signal behavior, testability, design flexibility, cost and availability, power consumption etc. Three different control systems are implemented on both soft-core and hard-core based FPGAs and compared by Weber and Chin in their paper[4].

Anemaet & As[5] presented an evaluation of design methods and concepts of soft-core processors. A detailed overview of Xilinx Microblaze soft-core is given as well as soft-core implementations of established fixed-core processors like Intel and Pentium Z80. Also discussed are the pros and cons of FPGAs over ASICs. In the white paper by Sandia National Laboratories[6], the author compared three reconfigurable FPGA based soft-core processors – the Microblaze, the open sourced Leon3, and the licensed Leon3. Using two different benchmarking applications, the resource utilization was measured for each. Miney & Kukenska[7] study the implementation of soft-core processors in FPGA and some of the decisions and design trade-offs which must be made during the design process. It looks at the operational performance as well as the power required to implement the design system functionality. Salem, Othman & Saoud[8] implemented a Real Time Operating System on both Hard-core and Soft-core processors and used them to control a DC motor drive. In his paper, Prado[9] presented a comparison in speed, power, flexibility and cost between a micro-controller and its soft-core version. Soft-core developed by university of Massachusetts is compared against a hard PIC16F84 micro-controller. The soft-core was found to outperform the microcontroller by a speed factor of 6.9 and in power consumption by a factor of 28.

III. Software Design

A digital Oscilloscope application was designed to analyze the performance of both the development boards used in this paper. Majority of the processing that forms the backbone of this application is implemented in hardware, which is discussed in the section Hardware Implementation. The software for the application is written in C. First, the FPGA is programmed with a specific bit file that allows us to upload this C code onto the non-volatile external PSRAM memory in Nexys-4 development board through UART. Following this, we program the FPGA with the bit file for the Oscilloscope application. The main component in the C application is a while loop. Figure 1 graphically represents the control flow inside the while loop. The loop repeats again after each refresh period of the display.

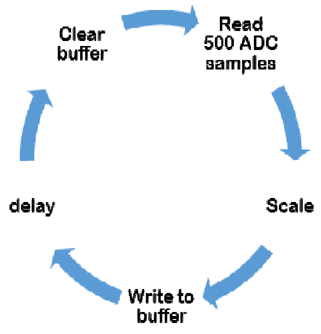


Figure 1. An overview of the contents of the while loop

Initially 500 ADC values are read from the XADC port of the development board. These values are then processed and scaled to values that correspond to the location on the VGA display. For this, an equation was developed which transformed the analog voltage value (which ranged from 0-1 Volt) to a function of the X and Y axes on the display where X is the sample number and Y is the amplitude.

$$x = x\_plotarea\_start + index \quad (1)$$

$$y = (y\_plotarea\_stop+1) - analog\_value \quad (2)$$

where:

- **x** and **y** are the coordinates on the display area (within a 640x480)
- **x_plotarea_start** is the offset from the right border on the screen and is set as 70
- **y_plotarea_stop** is the midpoint of the screen from where the positive y axis starts and is set as 239
- **index** is the position of the element in the 500 long array and **analog_value** is the value of the element

After this the memory location in the video buffer corresponding to these X and Y values are calculated and written into. Since the hardware implementation of the VGA module in FPGA enables independent refreshing of the display, the buffer values which have been written would be visible on the display as colored pixels. Following this step, a delay loop is executed so that the waveform snapshot appears stable enough for the user to see it clearly. Immediately after this, the buffer is cleared so that the next set of ADC values can be written to it. To increase the refresh rate of the screen, only the memory locations that were written are cleared. These set of steps are repeated in each iteration thus giving rise to a continuously refreshing Digital oscilloscope.

IV. Hardware Design

Hardware Implementation has been done on two separate development boards. The first one is the Digilent Nexys-4 board that houses an Artix-7 FPGA and a softcore ARM cortex M0. The second one is the Digilent Zybo board that houses both a Zynq 7010 FPGA and a dual core ARM Cortex A9 processor. The implementations for both are slightly different because of the difference in hardware and bus architectures.

A. Soft-core based design

On the Nexys-4 development board, we are using soft-core ARM Cortex M0 processor distributed by ARM as open source, as the embedded processor. Following this different custom IP modules were designed for the peripherals and their related functionalities in Verilog and VHDL. All the custom IPs along with those provided by ARM for their AMBA (AHB Lite) bus were combined to generate a Block diagram named *System Wrapper* which was then connected to the Cortex M0 DS processor to make up the entire top level. The Cortex M0 DS processor communicates to the peripherals in the system wrapper module using the AMBA 3 AHB-Lite bus. Figure 2 shows a block diagram representation of how the top level module looks like. The top level consists of two main modules that are the Cortex soft-core module and the System Wrapper module.

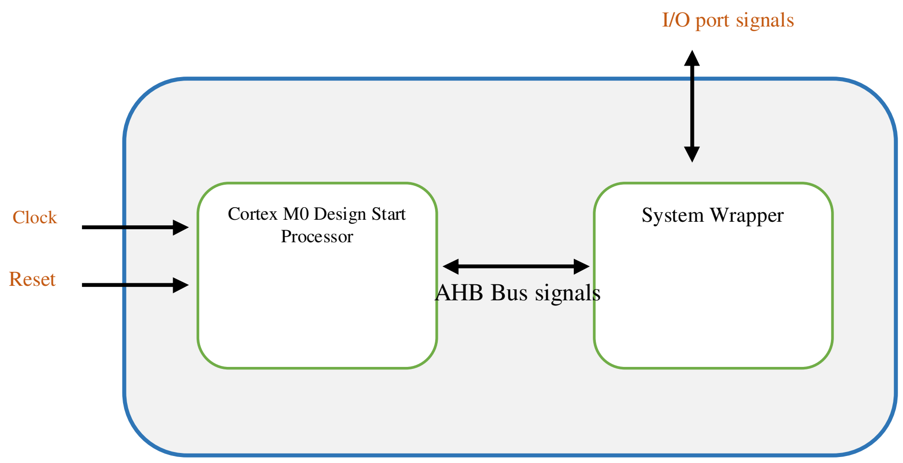


Figure 2. Top level block for the Soft-Core based design

B. Hard-core based design

For the hard-core based design, Digilent Zybo FPGA development board was used. This board houses an ARM Cortex A9 dual core processor inside the Zynq series FPGA logic. The IP block provided by Xilinx for the Zynq processor was used to form the block diagram for the top-level design in Vivado. Compared to Soft-core based design, no system wrapper was needed in the top level, as AXI Interconnect IP block added along with Zynq Processing block performs Address and Data Multiplexing. The main components of the top level design (shown in Figure 3) are the Zynq processing system, the Bus interfacing IPs, VGA module and the I/O blocks.

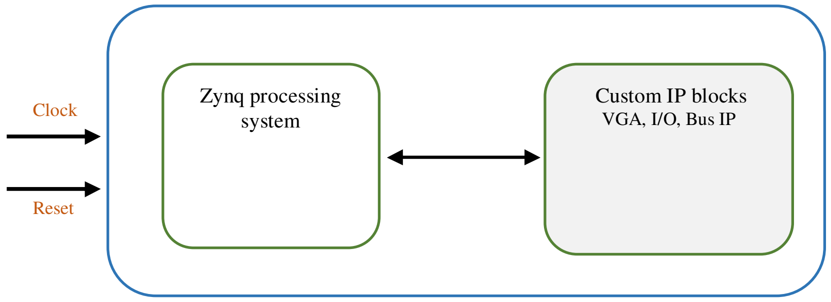


Figure 3. Top level for the Hard-core based design

V. Experimental Results

A. Experimental Setup

In this section we will discuss how both the implementations were tested on hardware and the results that were obtained. Figure 4 shows what a typical testing set up involves. An Analog voltage function is given to the FPGA through the XADC port. The FPGA does the required processing on the data and sends the necessary VGA signals to the monitor which displays the dynamically refreshing waveform on an oscilloscope background. An Oscilloscope can be optionally connected between the function generator and the FPGA board so as to verify the wave function.

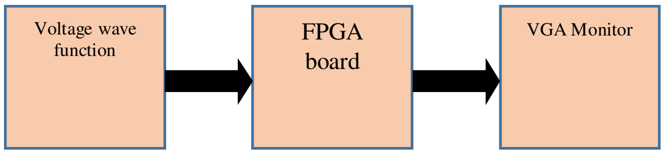


Figure 4. An overview of the hardware Setup

B. Results

a) Speed

The speed of the system is calculated by looking at the time taken for each core to write a single pixel data to the display buffer. First, sine wave is chosen as the wave function for easiness in measuring. Then for each design (Hard-core/Soft-core), we find a frequency that results in a single period of the wave fitting the screen perfectly. As we know that each waveform on the monitor has maximum of 500 pixels, we can calculate the approximate time taken to write one such pixel from the 500 long array (in which we store the ADC values) into the display buffer.

For the Soft-core based design, this frequency was found to be 75 Hz which corresponds to 26.6 µs to write a single data point to the display buffer. For the Hard-core based design, the frequency was found to be 1.5 KHz which corresponds to 1.33 µs. Thus the Hard-core was found to have faster read-write speed by this method.

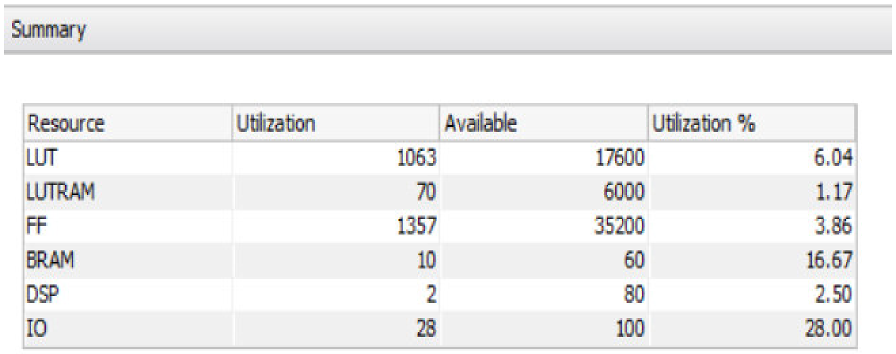
b) Power

Xilinx Vivado tool gives an opportunity to perform power analysis and it generates report with the results. The data in this report is compared for both designs to learn more about the power consumption in each. According to the Vivado power analysis report, hard-core was predicted to consume much more power (1.443 W) compared to the soft-core (0.170 W).

However, this is in contrast with what most studies report. For instance work done by Mundragon and Christman[3] found that hard-core processors generally consume lower power compared to its soft-core counterparts. This discrepancy in our testing results was attributed to the black box nature of the Cortex M0 Design Start processor provided by ARM. We are under the assumption that the processor might have a feature where most of the logic and peripherals are put into sleep mode when inactive and wake up when a service is requested from the processor.

c) Resource Utilization

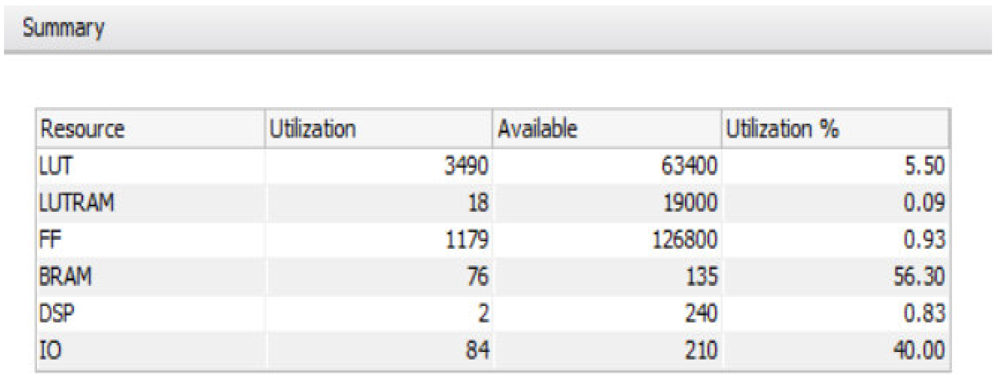
Xilinx Vivado also provides the user with the Resource Utilization data for the design after the Implementation phase. This data was compared for both the designs. And as expected, hard-core based design utilized less resources (Figure 5) than the Soft-core (Figure 6) one since the Cortex M0 Design Start processor is completely simulated using FPGA logic.



Summary

Resource	Utilization	Available	Utilization %
LUT	1063	17600	6.04
LUTRAM	70	6000	1.17
FF	1357	35200	3.86
BRAM	10	60	16.67
DSP	2	80	2.50
IO	28	100	28.00

Figure 5. Logic utilization for the Hard-core based design


Summary

Resource	Utilization	Available	Utilization %
LUT	3490	63400	5.50
LUTRAM	18	19000	0.09
FF	1179	126800	0.93
BRAM	76	135	56.30
DSP	2	240	0.83
IO	84	210	40.00

Figure 6. Logic utilization for the Soft-core based design

VI. Conclusion

The cores were compared based on three factors – Speed, power consumption and resource utilization. The hard-core outperformed the soft-core in both speed and resource utilization categories. The hard-core processor is not limited by the FPGA fabric speed as in the case of soft-core. Also unlike the soft-core, the hard-core exists as an independent component on the same chip separate from the FPGA logic, resulting in lower utilization stats. However, in case of power consumption soft-core comes out on top because the ARM Cortex A9 processor consumes more power in our testing. Table 1 shows a side-by-side comparison of hard-core and soft-core processors depending on the results from out testing. A check mark and a cross is used in the table to denote which type of processor is better in terms of the differentiating factor listed on the leftmost column.

The high speed of Hard-core based FPGA make them perfect for time intensive applications. Also the spare FPGA logic that would be used for soft-core processor could be used to further add more functionality or memory. An example of a real world application is Microsoft's internet search tool Bing, which swapped Microprocessors for FPGAs with embedded ARM processors on its data centers that drive its search algorithm and deep learning neural networks[10]. The same algorithms performed 40 times faster on the new chips compared to previous ones.

Table 1. Soft-core vs Hard-core test results

	Soft-core	Hard-core
Power consumption	✓	✗
Speed	✗	✓
Resource utilization	✗	✓
Flexibility for design	✓	✗

The soft-core meanwhile is better suited for applications that has the potential to undergo constant improvement in design. Its re-configurable aspect empowers the user to change the design requirements on the go. Peripherals can be added or removed from the design at ease in very few steps resulting in a custom processor that only contains the functionality needed. Soft-cores could also be used to test the functionalities of a design during the prototyping phase so they can eventually be replaced by a hard-core processor in the final design. In short, we can observe that the hard-core processor is best suited for applications were speed and resource minimization is of prime concern, whereas soft-core processor should be preferred where flexibility of application is main priority. Analysis presented in this paper does not account for all the aspects of a processor core and is mainly geared towards comparison of two specific architectures for a discrete set of metrics. More in-depth analysis can be performed to get a thorough comparison between the two cores.

Bibliography

[1]. Vazhoth Kanhiroth, V.J., "Embedded Processors on FPGAs – Hard-core vs Soft-core", Master's Thesis, April 2017.

[2]. Martos, P., Baglivo, F. "Implementing the Cortex-M0 Design Start processor on a low end FPGA", 2011

[3]. Mondragon, A. F., & Christman, J. "Hard Core vs. Soft Core: A Debate". In American Society for Engineering Education, 2012

[4]. Weber, J. M., & Chin, M. J. "Using FPGAs with embedded processors for complete hardware and software systems". In T. S. Meyer, & R. Webber (Eds.), AIP Conference Proceedings (Vol. 868, No. 1, pp. 187-192). AIP, November 2006

[5]. Anemaet, P., & As, T. V. "Microprocessor Soft-Cores: An Evaluation of Design Methods and Concepts on FPGAs". part of the Computer Architecture (Special Topics) course ET4078, Department of Computer Engineering, 2003.

[6]. Learn, M. "Evaluation of Soft-Core Processors on a Xilinx Virtex-5". Sandia National Laboratories. SAND2011-2733.

[7]. Minev, P. B., & Kukenska, V. S., "Implementation of soft-core processors in FPGAs". In UNITECH'07 International Sceintific Conference, November 2007.

[8]. Salem, A. K. B., Othman, S. B., & Saoud, S. B. "Hard and soft-core implementation of embedded control application using RTOS". In Industrial Electronics, 2008. ISIE 2008. IEEE International Symposium on (pp. 1896-1901). IEEE, June 2008

[9]. Prado, D. F. G., "Embedded micro-controller and FPGA soft-cores". In ELECTRÓNICA – UNMSM (No. 18). Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, USA, December 2006

[10]. McMillan, Robert. "Microsoft Supercharges Bing Search With Programmable Chips". WIRED. N.p., 2017. Web. 31 Mar. 2017.