Microprocessor Soft-Cores:
An Evaluation of Design Methods and Concepts on FPGAs

Pieter Anemaet (1159100), Thijs van As (1143840)
{P.A.M.Anemaet, T.vanAs}@student.tudelft.nl

Computer Architecture (Special Topics), ET4 078
Department of Computer Engineering
Faculty of Electrical Engineering, Mathematics & Computer Science
Delft University of Technology

Abstract

Reconfigurable computer architectures are becoming increasingly popular for many applications. This paper presents an evaluation of design methods and concepts of one particular reconfigurable architecture: soft-core processors. Soft-core processors provide a lot of options for system designers. System flexibility is heavily increased, while a high execution speed can still be attained. A detailed overview of the Xilinx MicroBlaze soft-core is given, as well as soft-core implementations of established fixed-core processors like the Intel Pentium and the Z80.

1 Introduction

FPGA design saw an increasing growth in popularity the last decade. This is because of several reasons, but mainly because of the low non-recurring engineering costs and the short design times compared to the ASIC design process.

In this paper, characteristics of soft-core processors are presented. An overview is given of design aspects, and some dedicated soft-core processors are mentioned. The Xilinx MicroBlaze [5] processor is talked about in more detail. Furthermore, soft-core implementations of fixed-core processors are dealt with. An implementation of the Intel Pentium [9] processor on an FPGA is reviewed in terms of performance and scalability. Eventually, advantages and disadvantages of cored and non-cored FPGA designs are evaluated.

The remainder of this paper is organized as follows. Section 2 presents general concepts about FPGA design, and reasons for its popularity. Subsequently, Section 3 gives an overview of the characteristics of a soft-core processor. In Section 4, examples of dedicated soft-core processor designs are given, and an architectural overview of the Xilinx MicroBlaze soft-core processor is presented. Section 5 deals with soft-core implementations of fixed-core processors, and an implementation of the Intel Pentium is reviewed. Hereafter, advantages and disadvantages of cored FPGA designs are evaluated in Section 6. Finally, Section 7 summarizes the evaluated design aspects.

2 Background

For the design of systems on chip (SoCs), the usage of FPGAs is increasing more and more. In many situations, where normally an ASIC would have been designed, an FPGA design has more advantages. Different aspects of FPGA design are presented in this Section.

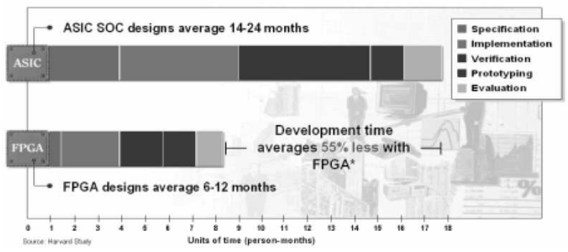


Figure 1: FPGA VS ASIC design times

2.1 Advantages of FPGA design

Designing for FPGAs brings lower non-recurring engineering (NRE) costs than designing the same SoC for an ASIC. This is mainly caused by the fact that no external NRE costs have to be made; the IC itself is already manufactured (and payed for). The manufacturing process of the IC is responsible for a large part of the costs. A mask set for an ASIC in the 90 nm process cost about $1M [11].

Concurrent development of hardware and software is possible when designing SoCs on FPGAs. Because it is possible to instantly make design and debug changes, hardware and software development teams can cooperate more effectively.

Because of this concurrent development of hardware and software, and the fact that the physical ICs are already fabricated, the product lead times have become very short compared to ASIC designs. Average FPGA and ASIC design times are given Figure 1 [10].

Due to its pre-designed circuitry, there will not occur any issues related to crosstalk (XT) in an FPGA design. These issues include capacitive, inductive or conductive coupling between different circuit elements.

When designing for an FPGA, the designer does not have to deal a lot with timing issues. This is because extra routing and timing (like digital clock managers) are implemented on FPGAs.

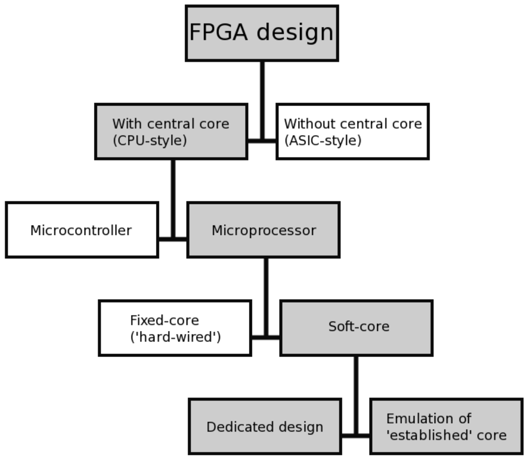

FPGA design
With central core (CPU-style)
Without central core (ASIC-style)
Microcontroller
Microprocessor
Fixed-core ('hard-wired')
Soft-core
Dedicated design
Emulation of 'established' core

Figure 2: FPGA design space

2.2 Disadvantages of FPGA design

When a design has to be mass-produced, the costs will be very high for the physical chips alone. Of course there are low NRE costs, but these are one-time only costs, and will be factored out by the number of ICs needed. This is probably the most important reason for not implementing a design on an FPGA. An FPGA can cost in the order of 100 times more than an ASIC [11].

The operation speed of a design on an FPGA slower than the same design on an ASIC. This varies , depending on the specific design.

When implementing the same design on an FPGA and an ASIC, the total area usage will be higher on an FPGA. This is mainly because of the routing logic needed.

Every time an FPGA-based system is powered on, the FPGA needs to be reconfigured (there are some exceptions, like the recently released Xilinx Spartan-3AN [5]). This means that there has to be some extra non-volatile memory in the system, and a controller that is able to configure the FPGA with its design.

2.3 Design space within FPGA development

The design space within FPGA development can be divided into different sections. Figure 2 presents an overview of this division.

First, the design space is split up in designs that have an ASIC-like architecture and designs that have a CPU-like architecture. The ASIC-like designs are able to perform a fixed number of tasks, in specific implementation. The CPU-like designs can perform multiple tasks, making use of the instruction set of a general purpose processing unit. One can write software for this, granted that there is a tool-chain (compiler, linker, etc) available. This paper focusses on the CPU-like designs.

Subsequently, the design space is separated into microprocessor and microcontroller designs. Microcontroller refers to a complete SoC, mostly including some peripherals, random access memory, program and data memories. Usually a lot slower than a microprcessor. This paper elaborates about microprocessor implementations.

Practically, there are two ways of embedding a microprocessor in an FPGA design. The first one is embedding a hard-wired core in an FPGA. An example of such an implementation is the Virtex-II Pro [5] FPGA by Xilinx. This FPGA has up to two PowerPC cores embedded. The other possibility is a so-called soft-core, which is practically an implementation of a processing core which is dynamically (re-)configured on the FPGA. This remainder of this paper will deal with soft-cores.

The implementation of a soft-core can take two forms. The first one is in the form of an already established core. For example an implementation of a MIPS or Pentium architecture. Section 5 elaborates on these cores. The second implementation is in the form of an architecture that is designed as a dedicated (reconfigurable) core. Section 3 gives examples of such cores.

3 Soft-core processors

This Section presents information about soft-core processors in general. General concepts, and speed/flexibility trade-offs are presented in this Section.

3.1 What is a soft-core processor?

As stated earlier, a soft-core processor is a dynamically (re-)configurable processing core (on an FPGA). When the source of a soft-core is available, it is extremely easy to adapt this core to the designer's own needs and wishes. Extra peripherals like UARTs can be added or removed easily.

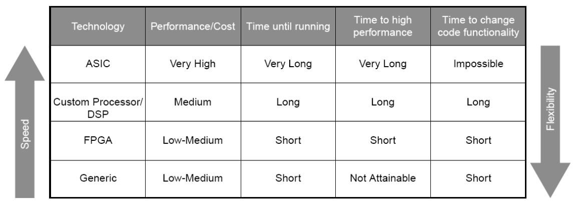
One of the main advantages of soft-core processors is, that they can be used as general purpose processors like known in many (personal) computer systems. When a tool-chain is available, a programmer can write applications for these cores in a high-level programming language like C or C++.

Combining multiple processing cores into one FPGA design is relatively easy. Section 4.2 will present a specific design method for implementing multiple MicroBlaze cores into one FPGA.

Using a general purpose processor based design, sufficient 'headroom' is ensured. Headroom is used as a term to indicate that there is space available for future system elements that don't have reason for existence at the time of the initial release. It is for example easy to add new peripherals to an existing bus architecture. A soft-core can also be used as a reconfigurable co-processor.

3.2 Speed/flexibility trade-off

When designing computer systems, the designer faces a trade-off between system speed and flexibility. Figure 3 depicts an overview of different technologies, organized according to this speed/flexibility trade-off. At one end of the spectrum resides the ASIC, which has a relative high execution speed, but a very low flex-


Speed
Flexibility

Technology	Performance/Cost	Time until running	Time to high performance	Time to change code functionality
ASIC	Very High	Very Long	Very Long	Impossible
Custom Processor/ DSP	Medium	Long	Long	Long
FPGA	Low-Medium	Short	Short	Short
Generic	Low-Medium	Short	Not Attainable	Short

Figure 3: Speed/flexibility trade-off

ibility. After all, an ASIC is designed to execute (just) one function. At the other end of the spectrum resides the general purpose processor (GPP). A GPP has a relative slow execution of tasks, because instructions have to be fetched and decoded from memory before execution can start. Also, most (complicated) functions are composed of more than one instruction. The Figure shows that FPGA design and GPP design only differ in the 'Time to high performance' criterion. With the use of soft-cores we want to fill the gap between ASIC design and GPP design; we want to be able to design systems which are very flexible, and at the same time are able to perform very fast at specific tasks.

4 Dedicated soft-core processors

There are several soft-core processors designed for FPGAs, both commercially and freely. Examples are the Altera NIOS II [1] core, the OpenSPARC [4] core and OpenRISC 1000 [3]. Xilinx, the world's largest manufacturer of FPGAs designed two soft-cores, the PicoBlaze [5] and the MicroBlaze [5] cores. The PicoBlaze core is actually an 8 bit microcontroller. MicroBlaze is a full-featured 32 bit RISC core. This Section presents more information about the MicroBlaze soft-core.

4.1 Xilinx MicroBlaze architecture

The MicroBlaze soft-core processor is a 32 bit RISC processor, with a maximum clockspeed of 150MHz. A Harvard-style bus architecture is used, so there are separate instruction and data

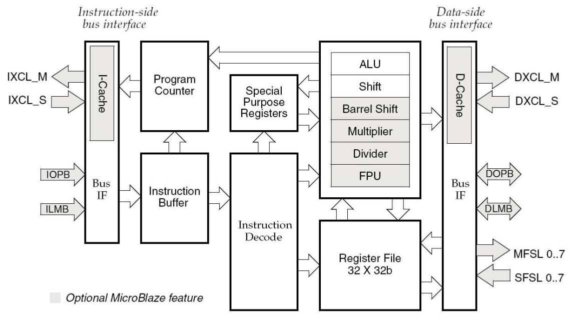


Figure 4: MicroBlaze block diagram

busses. The IBM CoreConnect [2] bus architecture is used for connecting peripherals to the the MicroBlaze.

Xilinx released its Embedded Development Kit (EDK) software to easily modify the MicroBlaze core. Extra peripherals like UARTS, ethernet controllers or other IP cores can be easily configured using the EDK. It also offers the capability of generating an ASIC design out of the the MicroBlaze FPGA design.

A fully featured toolchain is available for the MicroBlaze core. Versions of the GNU C Compiler, GNU Debugger, assembler, linker and other tools have been adapted to work with the MicroBlaze.

A default configuration of the MicroBlaze core uses 920 CLBs. On a Spartan-II FPGA, it uses approximately 80% of the available resources. On a Virtex-II Pro FPGA, it uses approximately 1% [10].

Figure 4 depicts a block diagram of the MicroBlaze core [5]. The OPB busses are the On-chip Peripheral Busses, as defined by the IBM CoreConnect architecture. The Local Memory Bus (LMB) are used to communicate with local memories. The Fast Simplex Link (FSL) interfaces are used to connect accelerating hardware like co-processors to the MicroBlaze architecture. Up to 8 interfaces are available. The Xilinx CacheLink (XCL) interface provides an interface to cache memories.

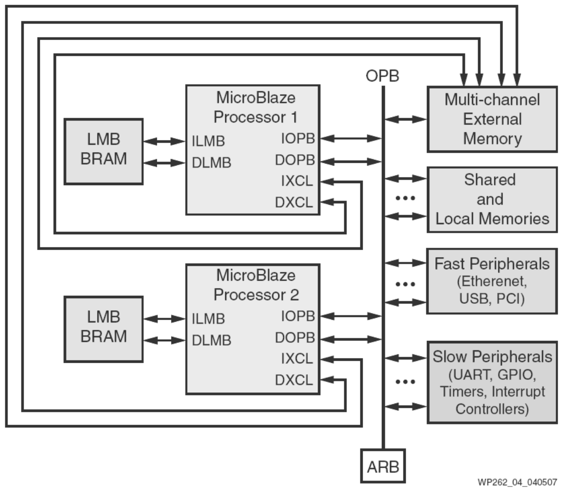

OPB
MicroBlaze Processor 1
LMB BRAM
ILMB
DLMB
IOPB
DOPB
IXCL
DXCL
MicroBlaze Processor 2
LMB BRAM
ILMB
DLMB
IOPB
DOPB
IXCL
DXCL
Multi-channel External Memory
Shared and Local Memories
Fast Peripherals (Etherenet, USB, PCI)
Slow Peripherals (UART, GPIO, Timers, Interrupt Controllers)
ARB
WP262_04_040507

Figure 5: MicroBlaze multi-processing topology

4.2 A MicroBlaze multi-processor environment

Figure 5 depicts a multi-processor topology using MicroBlaze cores [6]. The cores and some fast peripherals are connected to the OPB bus. An arbiter takes care of arbitrating the bus. All MicroBlaze cores have independent access to an external memory. Slower peripherals are usually connected to another bus, that is connected with a bridge to the OPB.

4.3 When to choose for MicroBlaze?

When designing computer systems, it is in many cases useful to use a general purpose processor in the design. Section 3 elaborates more on the question why to use a soft-core processor. The MicroBlaze core is available for every developer who has a license for the EDK software. Because the EDK software is widely used in industry, and the MicroBlaze core is supported on a wide range of popular Xilinx FPGAs it is a very cost-effective solution in many cases. The table below presents the performance of the MicroBlaze core on different FPGAs. Performance is given in Dhrystone MIPS (DMIPS).

FPGA	Size (LUTs)	DMIPS
Virtex-5	1010	240
Virtex-4	1809	184
Spartan-3	1843	115

5 Soft-core implementations of fixed-core processors

Developing new processors is a time consuming task. Not only have the number of transistors per chip increased exponentially, the introduction of techniques such as pipelining increase the complexity of the models dramatically. The complexity of simulation and testing is increasing in line with the chip complexity.

A new way to test and experiment on processor concepts, is to implement them on an FPGA. This way, they can be tested and simulated in an environment that achieves almost the same performance as the real implementation.

Another reason for implementing processor cores on FPGAs, is for education. Having a microprocessor implementation running on an FPGA can give the student a much better idea of the architecture of a microprocessor. This in contrast to a software emulator; changes to the design can be made easily, and the effects can be ‘visualized’ on a real chip.

5.1 MIPS implementation

In [7], a 32-bit MIPS CPU was designed for the use in SoCs and for educational purposes. 25 of the most used instructions were implemented in this design (mostly load/store, boolean, shift and branch instructions). A bottom-up design approach was used, starting with the design of the memories, boolean functions and register files.

The processor was implemented on a Xilinx XCS200 FPGA (200K gates). The processor was tested with an image processing implementation.

It was also demonstrated to embed this design in a SoC, with the image processing device and the MIPS processor in the same FPGA (that has more than 600K gates).

5.2 Z80 implementation

An implementation of a Z80 processor [8], was solely designed for educational purposes. A number of goals to be understood by students after studying the Z80 design are stated below:

1. What does actually take place on the hardware?
2. How does communication over busses take place ?
3. What bus contention problems take place when organising micro-operations of a machine-cycle?

These goals are visualized by presenting the states, buses and other hardware graphically. By actually presenting it visually and by creating a possibility to iterate through a program step-by-step, the student obtains a better insight in the microprocessor.

5.3 Intel Pentium implementation

Developers at Intel implemented a Pentium 75 [9] processor on a Xilinx Virtex4 LX200 [5] FPGA. The FPGA was made pin-compatible with a standard Socket 7 motherboard. It is able to boot and run Windows XP and Fedora Core 4. Tests proved the FPGA implementation was able to execute all instructions according to expectations.

The mainboard was clocked down from 75 MHz to 25 MHz. Although no obvious reason is stated in [9], this probably has been done because the routing tools for the FPGA weren't able to successfully rout the design for use with a 75 MHz clock. The used technology was 90 nm, while the original Pentium was fabricated on 600 nm technology. This FPGA implementation has a slow-down of 3x compared to the original model. When the generation gap (of 12 years) is taken into account, it can be stated that this Pentium core could be clocked at 750 MHz using current silicon technology (90 nm). This is just a rough analysis, using the fact that a current desktop processor has approximately 5x the number of pipeline stages (and hence 5x the clock rate).

5.4 Why implement a Pentium on an FPGA?

The Pentium simulation was extended with some special features, exploiting the possibilities of the FPGA. This gives the designer the possibility to explore the design space.

The following modifications were made to the core:

- The branch target buffer was expanded from 256 entries to 512 entries.
- The L1 Caches were expanded from a 8 KB 2-way set associative with 32 bytes per cache line to 32 KB 8-way set associative.
- Cryptography engines were integrated, extending the system with some rather special functionality.

In the next two Sections, implications caused by the second modification are presented. The first modification did not have large effects on performance, and the third modification was merely an addition to the system.

5.4.1 Implications on area

Figure 6 depicts an overview of the area increase after changing the cache properties. As can be seen from the figure, there was an increase of more than 50% on the RAM blocks.

5.4.2 Implications on speed

Figure 7 depicts an overview of the performance increase after changing the cache properties. The speed increased with an average of 15.75%, with crafty reaching as high as 40%.

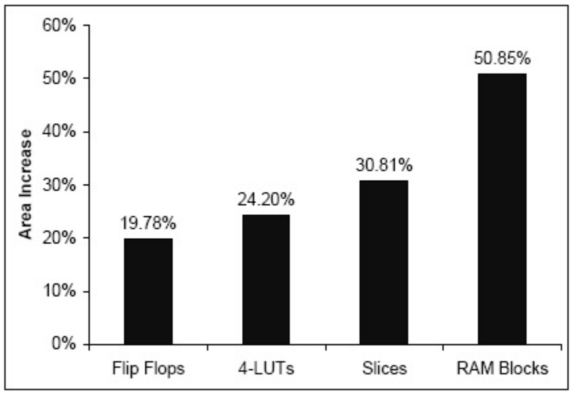


Figure 6: Area increase of 32KB 8-way L1 caches VS 8KB 2-way L1 caches

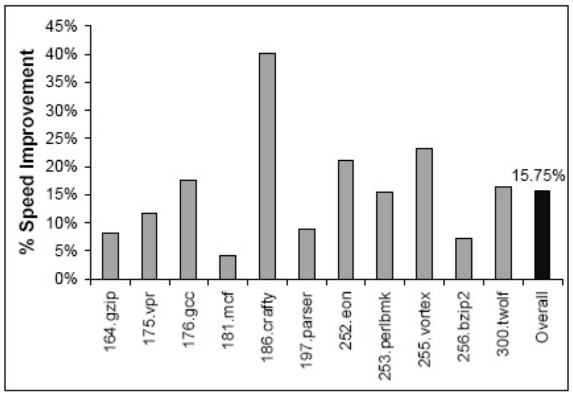


Figure 7: Performance increase of 32KB 8-way L1 caches VS 8KB 2-way L1 caches

6 Advantages and disadvantages of (soft-)cores

In this chapter, the advantages and disadvantages will be discussed of (soft-)cores. First, in section 6.1, there will be a discussion between FPGAs equipped with and those without a core. Next, a discussion in section 6.2 will be on soft versus hard-cores.

6.1 FPGAs with GPP core VS FPGAs without GPP core

Advantages of FPGAs with a GPP core:

- FPGAs with a core are software programmable. A high-level programming language like C or C++ can be used to determine the system function(s).
- FPGAs with a core are easily updated/upgraded with new functions, because they can be programmed in software. This saves a lot of money.
- FPGAs with a core can easily communicate with peripherals next to the system. When a software library for these peripherals is available, software functions can be used for programming.

Disadvantages of FPGAs with a GPP core:

- The core usually takes a fair amount of resources on the FPGA. This depends on the host FPGA and on the core, but not every situation is suited for a core-based design.
- When using a GPP core, a compiler/linker/debugger toolchain should be available for the given architecture. This is not always the case (to some extend). Designing a toolchain takes a lot of time.
- Because each instruction for the processor has to be fetched and decoded from a memory, a speed degradation takes place.

6.2 FPGAs with a soft-core VS FPGAs with a hard-core

Advantages of FPGAs with a soft-core:

- A soft-core can be removed from the design when it is not needed. This saves a lot of area, that can be used for other elements.
- A soft-core can be customized to the demands of the designer. Heterogeneous functions like large multipliers can be excluded, when not needed.
- A soft-core can be reconfigured at run-time to meet other constraints.

Disadvantages of FPGAs with a soft-core:

- A soft-core uses many resources from the FPGA. Heterogeneous functions on the FPGA are in many cases taken by the soft-core. A fixed-core has its own hard-wired hardware.

- For most hard-wired cores, a toolchain is already available. This is not always the case for soft-cores.

- A fixed-core is a lot more optimized for the silicon technology. Manually placed and overthought blocks are inside, instead of an implementation routed by software.

7 Summary

FPGA design saw an increasing growth in popularity the last decade. This is mainly because development times are drastically lower than designing ASIC systems. NRE costs are also much lower than with ASIC design.

When designing FPGA based systems with a microprocessor architecture, one can choose for a soft-core processor, or an FPGA with a built-in fixed-core processor. A soft-core processor is a general purpose processor described by an HDL, which can be modified by the designer. This processor is programmable using a (high-level) programming language, making use of the processor's instruction set.

With a soft-core processor, a design becomes more flexible, while still keeping high-performance parts inside the FPGA.

The Xilinx MicroBlaze 32 bit RISC soft-core processor is available for most Xilinx FPGAs. It can be easily adapted to the designer's needs. A full-featured GNU toolchain is available for software development.

Developing new microprocessors is a time consuming task. To test and experiment on these processor-concepts, they can be emulated on an FPGA. Recently, a Pentium processor was implemented on an FPGA. The Pentium simulation was extended with some special features to benchmark the changes.

References

[1] *Altera*. http://www.altera.com/.

[2] *IBM CoreConnect*. http://www-03.ibm.com/chips/products/coreconnect/.

[3] *Opencores.org*. http://www.opencores.org/.

[4] *OpenSPARC.net*. http://www.opensparc.net/.

[5] *Xilinx*. http://www.xilinx.com/.

[6] V. Asokan. *Designing Multiprocessor Systems in Platform Studio*. Xilinx, April 2007.

[7] C. Chang, C. Huang, Y. Lin, Z. Huang, and T. Hu. FPGA Platform for CPU Design and Applications. In *Proceedings of 2005 5th IEEE Conference on Nanotechnology*, July 2005.

[8] H. Diab and I. Demashkieh. A reconfigurable microprocessor teaching tool. In *IEEE PROCEEDINGS*, volume 137, pages 287–292, September 1990.

[9] SL. Lu, P. Yiannacouras, T. Suh, and R. Kassa. An FPGA-Based Pentium in a Complete Desktop System. In *FPGA'07*, February 2007.

[10] K. Parnell and R. Bryner. *Comparing and Contrasting FPGA and Microprocessor System Design and Development*. Xilinx, July 2004.

[11] D. Pramanik, H. Kamberian, C. Progler, M. Sanie, and D. Pinto. Cost effective strategies for asic masks. *Cost and Performance in Integrated Circuit Creation*, 5043:142 – 152, February 2003.