A Fixed Logic RISC Processor with Retargetable Infrastructure

Raul Garcia
The University of Manchester
Email: raul.garcia@manchester.ac.uk

Dirk Koch
The University of Manchester
Email: dirk.koch@manchester.ac.uk

***Abstract*—RISC processors implemented into fixed logic strive to deliver high performance for some compute-intensive applications. Typically, the solution is to extend the instruction set architecture (ISA) with specialized instructions. For instance, the ARM Cortex-A9 added the NEON SIMD engine for the media applications domain. This is a non-scalable solution in embedded systems with tight area and energy budgets. In this paper we propose an architecture where the area occupied by the fixed logic NEON engine is substituted by a retargetable infrastructure using FPGA logic. Arguably, our architecture has two key advantages. First, the parallel processing characteristics of the FPGA logic can be exploited by reconfiguring the infrastructure into a SIMD functional unit (FU). Our experimental results suggest that, for some media applications, a stripped-down version of a FPGA-based SIMD functional unit could match the performance of its fully-featured fixed logic counterpart. Second, the same infrastructure can be used to offload other compute-intensive functions from the processor into specialized FUs, targeted at other application domains. We investigated this idea and evaluated different software functions prone to become hardware-accelerated. We show resource utilization metrics of hardware-accelerated versions of this software functions. We argue that this collection of FUs could be implemented into the reconfigurable infrastructure within the limts of a strict FPGA fabric budget. This budget is estimated from the fixed logic NEON engine area.**

I. Introduction

An Intruction Set Architecture (ISA) comprises all the operations that a processor can perform. Since the ISA of a RISC processor is designed to provide support for a wide variety of application domains, they are known as general purpose processors (GPP). Examples of the different application domains that a GPP can process are security, text processing, multimedia, networking, compression, etc. RISC processors are commonly implemented as hard processors in embedded systems. The term hard processor is used to describe a processor implemented into fixed logic (i.e. ASIC standard cells). As a result, the hardware of a hard processor can not be modified after the fabrication of the chip. For some applications, better perfomance is needed from RISC processors. This problem can be overcome by extending the ISA of the processor with an instruction or a set of instructions specially designed to boost the performance of that particular application. This concep is known as Instruction Set Extension (ISE). Each new ISE is application-driven. For instance, ARM extended the ISA of the ARMv6 with two application-specific instructions: USAD8 and USADA8. Respectively, they are used to calculate the unsigned sum of absolute differences and the accumulated unsigned sum of absolute differences between 8-bit values. This instructions are utilized for motion estimation, object recognition, and graphic compression algorithms such as MPEG. For the next iteration, the ARMv7, ARM introduced a Single Instruction Multiple Output (SIMD) ISE, known as the NEON engine. It was designed to accelerate multimedia and signal processing algorithms such as video encode/decode, 2D/3D graphics, gaming, audio and speech processing. Since this extensions are implemented into fixed logic, computer architects and hardware designers must release a new processor iteration each time a new ISE is implemented. Each new ISE has a cost in terms of chip area and energy consumption. Two disadvantages of a fixed logic ISE become evident. First, implementing a new ISE, impacts the time-to-market of the next product. Second, implementing new ISEs to accelerate more applications is a non-scalable solution, specially in embedded systems limited by strict area and energy consumption budgets. In this paper we present a different approach to deal with this disadvantages. We explore the idea of substituting the fixed logic area originally occupied by the NEON engine with fine-grained reconfigurable fabric. In theory, by making this substitution, we obtain a retargetable infrastructure within the hard processor.

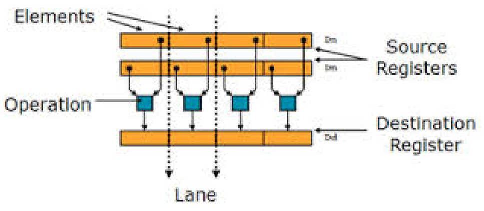

Figure 1. SIMD Architecture. Same operation is perfomed on different elements, generating multiple data.

A. The NEON SIMD Engine

The mobile market demands high performance from multimedia applications. In order to achieve this level of performance, most mobile processors feature a Sigle Input Multiple Output instruction set. A SIMD engine is a parallell architecture, where one single operation is executed by multiple processing units on multiple data elements. Figure 1 illustrates this architecture. ARM introduced on the ARMv7 architecture a dedicated SIMD processing unit known as NEON. The NEON SIMD engine has an independent pipeline and register file and provide 128-bit wide vector operations.

B. Reconfigurable Fabric

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of Configurable Logic Blocks (CLB). This CLBs are connected

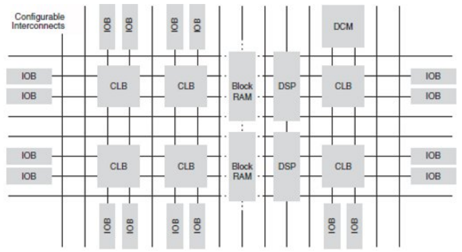

Figure 2. FPGA Fabric. The FPGA fabric is comprised of soft blocks: CLB logic blocks and optimized hard blocks: DSP and BRAM blocks.

through programmable interconnects. An FPGA fabric also have hard blocks highly optimized for specific tasks such as DSP blocks and BRAM blocks. Figure 2 depicts the block structure of a FPGA device. FPGAs allow users to change their designs after the product has been manufactured. This feature eliminates the cost of re-designing and physically replacing the product. FPGA devices are capable of parallel processing. Since multimedia applications requires this type of processing over large sets of data, FPGAs are very suitable for that kind of applications.

C. The Fixed logic and FPGA logic Gap

Since the architecture that we propose involves the substitution of fixed logic by FPGA logic, there are trade-offs that must be faced. Rose et al[?] measured the gap between fixed logic (ASIC standard cells) and reconfigurable logic (FPGA fabric). He found out that for some representative benchmark circuits, the area necessary to implement an algorithm in FPGA logic is, in average 18 times bigger than on Fixed logic. He also found out that a function implemented into FPGA logic is, in average, 3.5 slower that a design implemented into Fixed logic. This may lead to think that a FPGA-based infrastructure would not be able to compete with an efficient fixed logic design. However Kapre et al. [?] demonstrated that under certain conditions, an for some functions from the media applications domain, a vector engine implemented into FPGA fabric can outperform the fixed logic NEON engine by as much as 3.5times. This are encouraging results for our work because it demonstrates that under certain conditions, a functional unit implemented into reconfigurable logic can close the gap compared to a fixed logic implementation.

II. A Retargetable Infrastructure

A retargetable infrastructure implemented into a fixed logic RISC architecture would mean that the pipeline of the processor should integrate a FPGA-based functional unit as show in Figure 3. The infrastructure could exploit the parallel processing characteristics of the FPGAs to implement a specialized SIMD engine capable of delivering a performance close to its fixed logic counterpart. Further, the retargetable infrastructure could also be utilized to accelerate functions from other application domains. We explore this idea in the following subsections.

A. A Light SIMD Engine

The NEON SIMD engine provides a very comprehensive set of instructions, however as we will see in the next section,

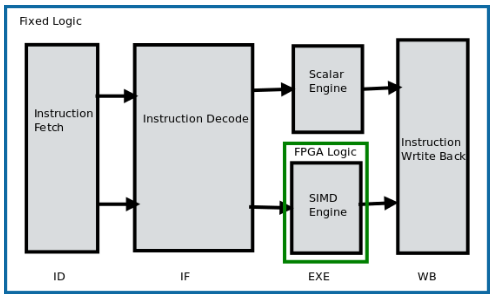

Figure 3. A retargetable infrastructure. The FPGA-based infrastructure is integrated into the RISC pipeline.

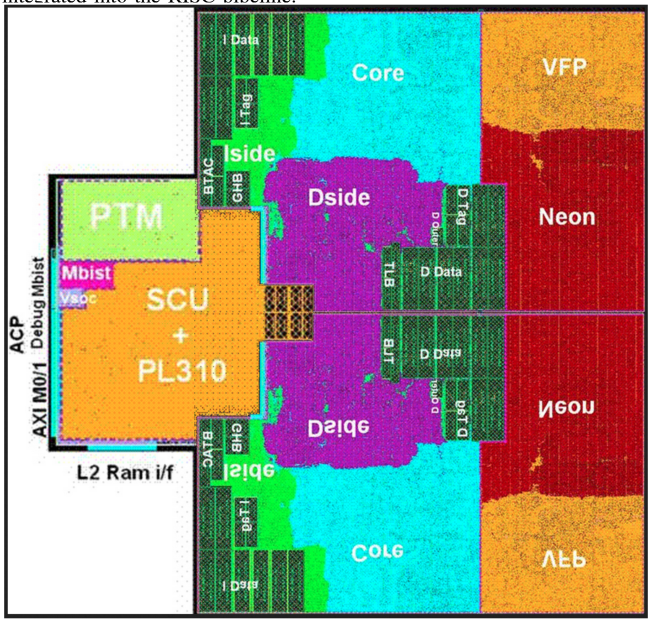

Figure 4. The Xilinx Zynq Chip Floorplan. From the FPGA blocks around the ARM processor it is possible to estimate its equivalence.

it takes a lot of area resources. This also means that the NEON is a power-hungry unit inside the ARM core. As an alternative. We can take advantage of the reconfigurability of the infrastructure to implement lighter, stripped-down versions of the NEON engine. By choosing the correct set of benchmarks that can expose bottlenecks, it is possible to define application-specific implementations that are less power-hungry than the fully-featured NEON engine.

B. FPGA fabric logic budget estimation

From the previous section we learned that there is an area gap between fixed logic and reconfigurable logic. In order to know what functions we would be able to implement into the infrastructure, we must estimate a FPGA fabric budget. From figure 4 we can observe the the NEON engine occupies about 20 percent of the total processor area. This information is useful when we estimate the FPGA budget for the NEON engine area.

1) The Xilinx Zynq Chip: One device that implements the ARM Cortex-A9 IP is the Xilinx Zynq-7000 chip [1]. Figure 5 depicts the general architecture of this device. It combines both 28 nm FPGA fabric and a hard processor IP in the

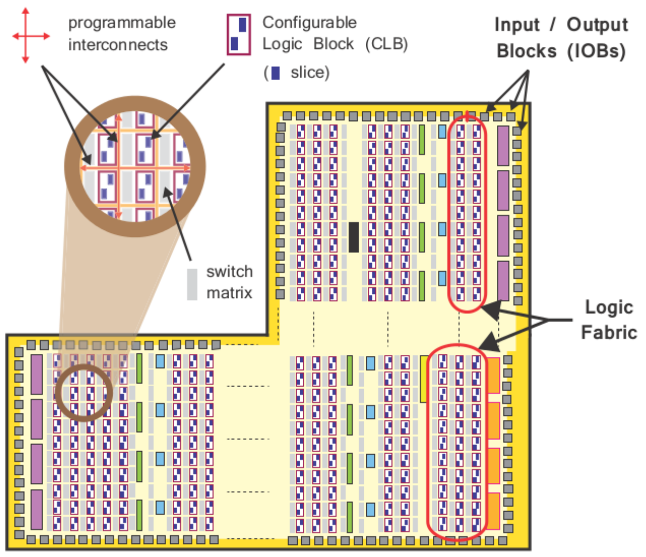
programmable interconnects
Configurable Logic Block (CLB)
(▮ slice)
Input / Output Blocks (IOBs)
switch matrix
Logic Fabric

Figure 5. The Xilinx Zynq Chip Architecture. The reconfigurable logic fabric is built around the fixed logic processor.

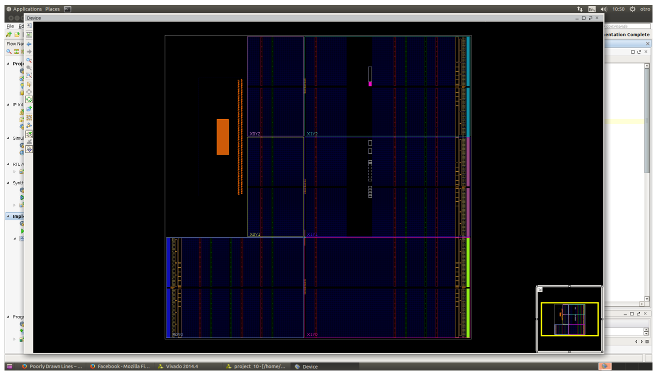

Figure 6. The Xilinx Zynq Chip Architecture. The reconfigurable logic fabric is built around the fixed logic processor.

same chip distributed in two sections: the Programmable Logic (PL), and the Processing system (PS).

2) *Budget estimation methodology:* Since the amount of building blocks (CLBs, Block RAM blocks, DSP slices) that the area corresponding to the PL section is known, it is possible to estimate the equivalence between the area that the PS represents and the amount of reconfigurable resources that could be allocated into that same area. For instance, using Xilinx tools to observe the floorplan of the XC7Z020 device, it is possible to count the number of CLB blocks implemented in Artix-7 FPGA fabric around the PS section.

3) *Results:* From the information obtained in the previous section, the height of the PS section in terms of CLB blocks can be estimated. From figure 6 it can be observed that the height of the PS section is about 50 CLBs (100 slices) high. Similarly, the PS width is about 16 CLB blocks (32 slices) wide, giving a total area of 800 CLBs. In the Xilinx 7 series FPGAs, four LUTs, their flip-flops as well as multiplexers and arithmetic carry logic form a slice, and two slices form a CLB [4], therefore the estimated amount of logic fabric that can be allocated into the PS area is about 6400 LUTs. Also, two columns of 20 BRAM blocks and two columns of 20 DSP slices must be considered. These estimations are summarized in table 1. Now that the amount of reconfigurable resources that the whole area of the PS section can allocate is known, it would be interesting to estimate how many resources the SIMD engine area only would represent. In order to make this estimation, we refer to Figure 1 and calculate the twenty percent of the resources previously estimated. The results represent approximately the amount of reconfigurable resources that can be allocated into the area currently dedicated to the SIMD engine in the ARM Cortex-A9. Since the NEON engine is part of the ARM Cortex-A9 hard IP, after the fabrication of the device, the hardware cannot be modified. However, if the hard IP area dedicated to the NEON engine were replaced with reconfigurable fabric, then this area would not be limited to perform media applications processing only, but it could be reconfigured into high-performance optimized engines for a wide range of application domains. This optimized engines would be bounded to the amount of resources that can be allocated into the same area, the estimation of this resources is presented in table 2. The implementation of this optimized engines would provide more flexibility and higher performance to the final product without adding area overhead.

FPGA Fabric	Artix-7 FPGA
CLBs	800
BRAM(36 Kb Blocks)	40
DSP slices	40

Table I
ARM Processor Resource Estimation

FPGA Fabric	Artix-7 FPGA
CLBs	160
BRAM(36 Kb Blocks)	8
DSP slices	8

Table II
SIMD Engine Resource Estimation

III. ARM Processor Instructions Usage Metrics

We selected a set of representative software benchmarks from the multimedia application domain [2]. Also we selected other software benchmarks commonly used in embedded systems [3]. We used a GCC cross-compiler targeted at the ARMv7 architecture. We compiled with autovectorization enabled to generate SIMD instructions and executed the binaries on the Digilent Zybo board, that features a Xilinx Zynq chip with an ARM Cortex-A9. Then we developed a profiling framework based on the Valgrind dynamic analysis tool. Our primary objective was to study what vector instructions are being generated and which are more utilized. We obtained metrics for applications from multimedia and other application domains. The usage metrics are presented in Table III. From this information

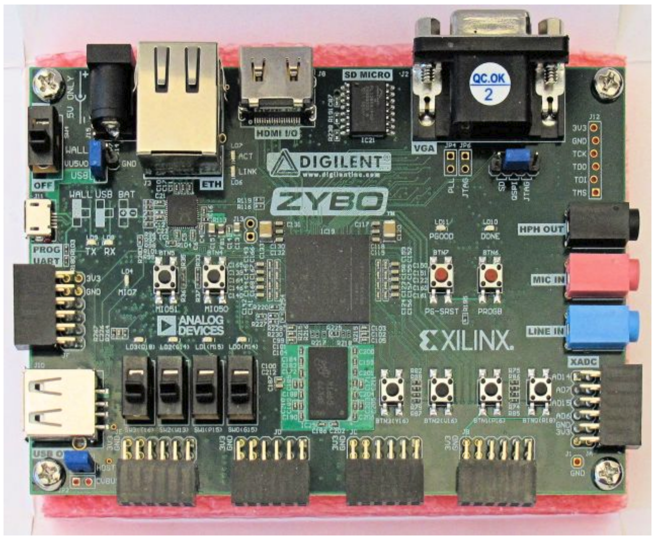

Figure 7. The Xilinx Zynq Chip Architecture. The reconfigurable logic fabric is built around the fixed logic processor.

we can determine which instructions are good candidates to be implemented as functional units into the reconfigurable fabric.

Benchmark	Description	SIMD Instr. Executed
crc32	desc	NVINST1, VINST2
dijkstra	desc	VINST1, VINST2
fft	desc	VINST1, VINST2
lame	desc	VINST1, VINST2
rijndael	desc	VINST1, VINST2
sha	desc	VINST1, VINST2
rgn2gr	desc	VINST1, VINST2
rgn2lm	desc	VINST1, VINST2
vecadd	desc	VINST1, VINST2
poly	desc	VINST1, VINST2
matmul	desc	VINST1, VINST2
dotprod	desc	VINST1, VINST2
l1lin	desc	VINST1, VINST2
l1sat	desc	VINST1, VINST2

Table III
SIMD INTRUCTION USAGE METRICS

IV. FPGA FABRIC RESOURCE UTILIZATION METRICS

In section 2 we estimated a strict resource budget based on the area of the fixed logic NEON engine. Now we will measure the amount of resources that different hardware benchmarks require in terms of reconfigurable fabric. This hardware benchmarks are selected from the study performed in the previous section. A stripped-down implementation of the NEON instruction set was developed. Our SIMD processing unit only supports integer operations and it uses the specialized ”hard” DSP blocks of the FPGA fabric. We use this implementation as the baseline design to generate other metrics. The idea is that different, lighter versions of it can be used for specific functions heavily used in multimedia applications. A lighter implementation means less resource utilization and less energy consumption. Besides the hardware benchmarks targeted at media appplications, we also measured benchmarks for security, networking, and image processing. Table four shows that it is possible t

Benchmark	Description	FPGA resourses utilized
crc32	desc	logic blocks
dijkstra	desc	logic blocks
fft	desc	logic blocks
lame	desc	logic blocks
rijndael	desc	logic blocks
sha	desc	logic blocks
rgn2gr	desc	logic blocks
rgn2lm	desc	logic blocks
vecadd	desc	logic blocks
poly	desc	logic blocks
matmul	desc	logic blocks
dotprod	desc	logic blocks
l1lin	desc	logic blocks
l1sat	desc	logic blocks

Table IV
FPGA FABRIC RESOURCE UTILIZATION METRICS

V. RELATED WORK

The idea of a system that combines the area efficiency and the performance of a Application-Specific Integrated Circuit (ASIC) with the reconfigurability of FPGAs has been around for some time. The GARP architecture [4] is one of the earliest works of this type. This architecture features a hard MIPS processors interconnected with a reconfigurable array. This array is dedicated to accelerate certain funcions. Similarly, the One-Chip architecture [5] presents a system comprised of a RISC processor implemented into fixed logic and a loosely coupled array of FPGA fabric. This works show that it is possible to obtain benefits from coupling an efficient hard processor with reconfigurable functional units. However, this is not an easy task because, there exist a gap [6] between fixed logic and reconfigurable logic in terms of area, energy and speed. Later, as studied by [7], the idea of extending the ISA of RISC processors with custom instructions became popular. This custom instructions or instruction sets are implemented into reconfigurable fabric to hardware accelerate different software functions. The work of [8] and [9] show an interesting approach where a framework to detect hotspots and then automatically generate and implement a custom instruction. Frameworks to generate custom instructions based on comercial tools and popular RISC architectures such as the NIOS II soft processor [10] are part of a current trend that strive to close the gap between fixed logic and FPGA logic. In [11] the authors present the architecture of a framework for custom instruction generation. They build this architecture around the ARM processor [12]. One of the most popular and utilized architectures for embedded systems. They propose a microarchitectural interface to support a plug-and-play model for integrating a wide range of accelerator to a general purpose ARM processor. [13]

[14]

VI. CONCLUSIONS

In this paper we explored the idea of a RISC architecture where the fixed logic SIMD engine is substituted by a FPGA-based retagetable infrastructure. We estimated a FPGA fabric budget for this infrastructure based on the current area of the fixed logic NEON engine. For its characteristics, the Xilinx Zynq chip was used as a base to make this estimation. Once we estimated a stric limit for the insfrastructure budget, we selected a set of software benchmarks from different application domains to study what vector instrictions are beeing generated by the GCC compiler. We also used the Valgrind profiling tool to expose software bottlenecks from those benchmarks. Then we presented resource utilization metrics from hardware functions to accelerate some of the software applications. We show that a collection of this functional units can be implemented into the reconfigurable fabric without exceding the limits of the established budget. The future work will include a more detailed study of the integration of a reconfigurable functional unit to the fixed pipeline of the ARMv7 microarchitecture.

Acknowledgment

The research leading to these results has received funding from...

References

[1] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart, *The Zynq Book. Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC* . Strathclyde Academic Media, 2014.

[2] C. Lee, M. Potkonjak, and W. H. Mangione-Smith, "MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems," *Thirtieth Annual IEEE/ACM International Symposium on Microarchitecture, 1997.*, 2011.

[3] M. R. Guthaus, T. M. A. Jeffrey S. Ringenberg, Dan Ernst, T. Mudge, and R. B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," *2001 IEEE International Workshop on Workload Characterization, 2001.*, 2011.

[4] J. R. Hauser and J. Wawrzynek, "Garp: a MIPS processor with a reconfigurable coprocessor," *The 5th Annual IEEE Symposium on Field-Programmable Custom Computing Machines.*, 1997.

[5] R. D. Wittig and P. Chow, "OneChip: an FPGA processor with reconfigurable logic ," *IEEE Symposium on FPGAs for Custom Computing Machines.*, 1996.

[6] I. Kuon and J. Rose, "Measuring the Gap Between FPGAs and ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2012.

[7] C. Galuzzi and K. Bertels, "The Instruction-Set Extension Problem: A Survey," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 2011.

[8] A. Canis, J. Choi, M. Alham, V. Zhang, A. Kammoona, J. H. Anderson, S. Brown, and T. Czajkowski, "LegUp: high-level synthesis for FPGA-based processor/accelerator systems," *FPGA '11 Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, 2011.

[9] O. A. et al., "An End-to-End Design Flow for Automated Instruction Set Extension and Complex Instruction Selection Based on GCC," *GROW'09: Proceedings of the 1st International Workshop on GCC Research Opportunities.*, 2009.

[10] Altera Corp., "Nios II Custom Instruction. User Guide."

[11] N. Clark, J. Blome, M. Chu, S. Mahlke, S. Biles, and K. Flautner, "An architecture framework for transparent instruction set customization in embedded processors," *32nd International Symposium on Computer Architecture, 2005. ISCA '05. Proceedings.*, 2005.

[12] ARM LTD., "Cortex-A9 Neon Media Processing Engine. Technical Reference Manual."

[13] S. J. Jie and N. Kapre, "Comparing Soft and Hard Vector Processing in FPGA-based Embedded Systems," *2014 24th International Conference on Field Programmable Logic and Applications (FPL)*, 2014.

[14] A. Severance and G. G. Lemieux, "Embedded supercomputing in FPGAs with the VectorBlox MXP Matrix Processor," in *2013 International Conference onHardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2013.