Soft-NEON

DOI:
10.1109/ASAP.2016.7760802

Document Version
Accepted author manuscript

Published in:
2016 IEEE 27th International Conference on Application-Specific Systems, Architectures and Processors, ASAP 2016

soft-NEON: A Study on Replacing the NEON Engine of an ARM SoC with a Reconfigurable Fabric

Jose Raul Garcia Ordaz
School of Computer Science
The University of Manchester
Oxford Road, Manchester
United Kingdom
Email: raul.garcia@manchester.ac.uk

Dirk Koch
School of Computer Science
The University of Manchester
Oxford Road, Manchester
United Kingdom
Email: dirk.koch@manchester.ac.uk

***Abstract*—Power is a limiting factor in the design of embedded processors. For this reason adding more instruction extensions is not a scalable option. To overcome this issue, we study the effects of replacing the NEON unit of an ARM SoC with an FPGA-like reconfigurable fabric. We measure the gap between the conventional hard-NEON and a soft-NEON implementation. We found that the soft-NEON has an overhead of 25.17× and 6.23× for area and latency, respectively. This overhead is reduced by exploiting the reconfigurability of the fabric by incorporating FPGA-specific optimization techniques. Moreover, we show that instead of implementing the predefined NEON instruction set, custom instructions can be loaded to the reconfigurable fabric by using a HLS compilation flow. With this approach performance gains of over 2.8× have been obtained for some kernels.**

I. Introduction

To enhance the performance of critical applications, new functionalities have been introduced to several embedded processors. These new functionalities are exposed as ISA extensions, for example ARM's NEON. This functional unit is located in parallel to the scalar ALU and the vector floating-point unit (VFP). With a rich instruction set of over 100 hard-wired SIMD instructions, NEON can be seen as a hardened general-purpose media coprocessor. The ample variety of NEON data-processing instructions translates into a large amount of real-estate used on an ARM core. For example, a visual inspection of a ARM Cortex-A9 floorplan shows that NEON takes approximately 20% of the SoCs real-estate. An alternative to provide the functionality provided by the hardened NEON without having to physically implement it at the same time would be to substitute it with a reconfigurable fabric. With this approach, depending on the current application, an appropriate SIMD instruction subset could be loaded and executed by the fabric. Unfortunately this approach comes at a cost. According to [1], a functional unit implemented in an FPGA fabric takes more area, is slower, and consumes more energy than the same function implemented into an ASIC. Although a similar gap is to be expected between a hard- and a soft-NEON, FPGA-specific optimization techniques can be used to close this gap.

II. A customizable soft-NEON Functional unit

A. Measuring the gap between soft-NEON and hard-NEON

We analysed the floorplan of a Zynq Z-7020 chip which features an ARM Cortex-A9 SoC. We establish that the area occupied by the hardened ARM CPU is equivalent to 10400 LUTs, 20800 FFs, 80 DSPs, and 40 BRAM blocks. Note that this SoC is a dual-core CPU (each single core containing a NEON unit). Consequently, the real-state corresponding to each single core is 5200 LUTs, 40 DSPs, and 20 BRAMs. To estimate the real-state equivalent to the two NEON units we take 20% of the total area of the hardened ARM CPU. This is equivalent to 2080 LUTs, 16 DSPs, and 8 BRAMs. The equivalent of a single NEON unit is half of this amount. To measure the amount of resources that the NEON unit would consume on an FPGA target, a design compatible with the NEON ISA was developed in HDL. The design was based on the specifications described on the architecture reference manual of the ARMv7-A architecture [2]. According to our results, the area ratio between the soft-NEON unit and a hardend-NEON unit implementation is 25.17×. This area gap, measured on a 28 nm fabrication technology, is in the range of 17-27× previously measured in [3]. The maximum frequency that the ARM CPU can achieve on a Zynq Z-7020 device is 866 MHz. Therefore the delay ratio between the soft-NEON unit and a hardend-NEON unit implementation is 6.23×. This ratio is slightly lower than the delay ratio measured in the same work (18-26×) [3].

B. Closing the gap between soft-NEON and hard-NEON

Given the low utilization of the NEON ISA (18% for the Parsec Benchmark), ISA subsetting is used to close the gap between a soft-NEON and a hard-NEON implementation. On average, the area-cost of implementing an application-specific soft-NEON subset is 9476 LUTs and 32 DSPs. Our profiling data shows that most of the NEON instructions used by an application doesn't perform operations on all the vector widths supported by NEON (i.e. 8-bit vector elements, 16-bit vector elements, 32-bit-vector elements, and 64-bit vector elements). In this case, we can further take advantage of the reconfigurability of the fabric by applying another FPGA-specific technique introduced in [4] *called vector width customization*. We applied this approach to the soft-NEON vector

TABLE I
EXAMPLES OF CUSTOM INSTRUCTIONS FOR THE SOFT-NEON FABRIC

Benchmark	Application Domain	Time-Consuming Kernel	% of total Execution Time	Speedup Estimated		Resource Utilization	
				Kernel	Overall	LUT	DSP
blackscholes	Financial Analysis	CNDF( )	33.33%	406.27	1.50	2914	13
fluidanimate	Animation	GetLengthSq( )	13.47%	44.75	1.15	1223	8
streamcluster	Data Mining	dist( )	94.45%	4.42	2.77	888	5
swaptions	Financial Analysis	CumNormalInv( )	20.54%	114.81	1.17	5184	75

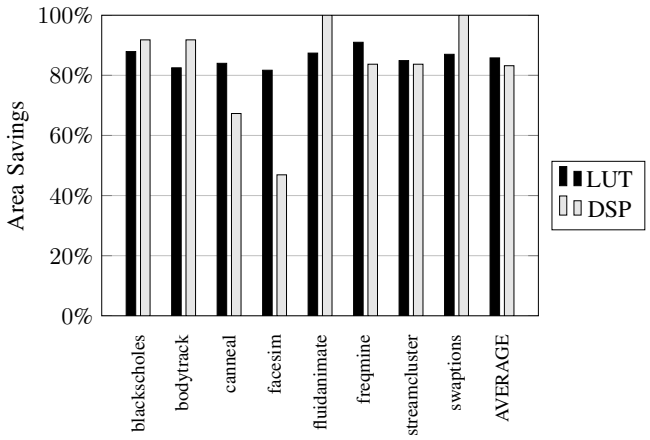

Fig. 1. Area savings obtained by applying ISA subsetting and vector width customization to soft-NEON.

unit to gain further area savings. In this case, the area-cost of implementing a soft-NEON ISA subset is on average 3719 LUTs, 8 DSPs, and 4 BRAMs. Figure 1 shows the area savings for application-specific soft-NEON subsets with vector width customization.

TABLE II
RESOURCE UTILIZATION BREAKDOWN FOR APPLICATION-SPECIFIC NEON ISA SUBSETS WITH VECTOR WIDTH CUSTOMIZATION

NEON Subset	Not Optimized			Optimized		
	LUT	DSP	Fmax	LUT	DSP	Fmax
blackscholes	7528	31	155	3167	4	172
bodytrack	11143	34	155	4575	4	172
canneal	9484	31	151	4196	16	168
facesim	14444	34	155	4787	26	172
fluidanimate	8888	34	155	3294	0	172
freqmine	6637	27	170	2353	8	187
streamcluster	8696	31	155	3963	8	172
swaptions	8989	34	155	3414	0	172
AVERAGE	9476	32	156	3719	8	173

C. Customization beyond the NEON SIMD Instruction Set

The fabric that implements soft-NEON could also be used to extend the capabilities of the architecture not only with SIMD instructions but with custom instructions targeted at specific applications. Dynamic reconfiguration can be used to load application-specific instructions at runtime to boost the performance of the processor. We used profiling data and the Vivado HLS tool to generate some examples of custom instructions for the Parsec Benchmark. Table I shows some examples of benchmarks examined, the application domain to which they belong, the name of the time-consuming kernel that we detected inside the application, and the fraction of time that it contributes to the benchmarks total execution time. The table also shows the overall speedup estimated per benchmark as described above and the count of FPGA primitives needed to implement those kernels in custom hardware.

III. CONCLUSION

In this work we explored the idea of substituting the hardened NEON of an ARMv7 processor with a reconfigurable FPGA fabric. We measured the gap between the conventional NEON and a soft-NEON implementation and we found that a soft-NEON takes 25.17× more LUTs and 6.13× more DSPs primitives than the FPGA resource equivalent of the hardened NEON. In addition the soft-NEON was 6.23× slower. We narrowed this gap with the help of *ISA subsetting* and *vector width customization* down to 3.6× for LUTs, 1.0× for DSPs, and 5.0× for latency. By considering customized ISA extensions beyond ARM and NEON, we demonstrated substantial performance boosts for some kernels using a high level synthesis approach.

ACKNOWLEDGMENT

The first author acknowledge the support from the Mexican National Council for Science and Technology (*CONACyT*) through scholarship number 381920 received to undertake doctoral studies. Dirk Koch is supported by the project *Reconfigurable Tera Stream Computing* funded by the Defence Science and Technology Laboratory under grant DSTLX10000092266.

REFERENCES

[1] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 2, pp. 203–215, 2007.

[2] ARM Holdings. (2014) ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition. [Online]. Available: https://silver.arm.com/download/ARM_and_AMBA_Architecture/AR570-DA-70000-r0p0-00rel2/DDI0406C_C_arm_architecture_reference_manual.pdf

[3] H. Wong, V. Betz, and J. Rose, "Comparing FPGA vs. custom CMOS and the impact on processor microarchitecture," in *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*. ACM, 2011, pp. 5–14.

[4] P. Yiannacouras, J. G. Steffan, and J. Rose, "VESPA: portable, scalable, and flexible FPGA-based vector processors," in *Proceedings of the 2008 international conference on Compilers, architectures and synthesis for embedded systems*. ACM, 2008, pp. 61–70.