Low Energy Heterogeneous Computing with Multiple RISC-V and CGRA Cores

Luís Fiolhais, Fernando Gonçalves, Rui P. Duarte, Mário Véstias[1], José T. de Sousa
INESC-ID/IST Lisboa, [1]INESC-ID/ISEL, Email: jts@inesc-id.pt

***Abstract*—The idea of combining multiple CPU and CGRA cores is not in itself original but detailed characterizations of such architectures and measurements on compelling applications are difficult to find in the literature. Although commercial CPUs, GPUs and FPGAs are widely available, there are no commercial CGRAs, which may be attributed to the lack of metrics on performance, energy and cost. In this paper, we introduce a heterogeneous computing platform consisting of several RISC-V CPU and Versat CGRA cores. Implementation results for several instances of the architecture are presented. The CPU of choice is the promising open source RISC-V architecture, which has never been featured in CPU/CGRA architectures. This paper presents independent implementations of two RISC-V cores: a minimal one, useful as a simple controller, and a more performant 5-stage pipeline implementation. The RISC-V cores have been designed using the recent Chisel HDL, useful for automating tasks pertaining to the writing of RTL. The selected CGRA is the published Versat architecture, for which 4 different instances have been created. Implementation results for 2 FPGA families and ASIC technology nodes are presented: area, frequency and power. Applications cover digital audio and machine learning, demonstrating the versatility of the proposed platform at competitive area, frequency and energy footprints.**

***Index Terms*—Low power architectures, RISC-V, CGRAs, Heterogeneous computing**

I. Introduction

With the advent of the Internet of Things (IoT) and Artificial Intelligence (AI) algorithms based on neural networks, it is becoming more and more important to reduce the cost and energy consumption of silicon devices. Systems using high performance CPUs, GPUs, FPGAs or combinations of these [1], [2] are interesting but can hardly fit the bill in applications requiring ubiquitous low cost and low power devices.

A high performance CPU can do a few operations in parallel but most of its hardware is dedicated to handling instructions efficiently, which represents a large overhead for the envisioned systems. FPGAs are designed to build arbitrary digital circuits and require a formidable configuration infrastructure which is an overkill for low energy IoT systems. GPUs comprise large numbers of streamlined von Neumann processors, requiring a lot more hardware than a dedicated hardware datapath for the same purpose. Finally, a system entirely implemented in hardware would probably have the best performance but at the cost of a large silicon area (high device cost) and lack of programmability.

Given the drawbacks of the above approaches, an interesting option seems to be a system consisting of several simple CPU cores combined with programmable hardware cores that are simpler than FPGAs. For algorithms that require extensive control, the use of CPUs actually represents an economy of resources as many control structures can be scheduled and executed by a single CPU. This corresponds to time multiplexing of control tasks in a small piece of hardware; for parallel tasks more CPUs can be added. Reasoning in a similar way, the data processing tasks can be time multiplexed using programmable hardware: large hardware circuits can be broken down into smaller pieces and run sequenntially on the programmable hardware.

Because FPGAs are too onerous for most applications, a more suitable type of reconfigurable hardware for embedded devices is the Coarse-Grained Reconfigurable Array (CGRA) [3]. A CGRA is a collection of programmable functional units (FUs) interconnected by programmable switches. The FUs operate on data words of commonly used widths such as 8, 16, 32 or 64-bit words. When the CGRA is programmed it implements hardware datapaths that accelerate computations.

In this work, a heterogeneous computing platform consisting of several RISC-V CPUs [4] and several Versat CGRAs [5] is proposed. The RISC-V cores have been developed under a project code named Adept. A similar approach has been proposed in [6] but here we make the case for the use of RISC-V CPUs and Versat CGRAs.

In order to sustain the IoT revolution and unleash the creativity of millions of engineers, it is indispensable to have an Instruction Set Architecture that is free of Intellectual Property (IP) rights or license fees [7]. Among the few free Instruction Set Architectures (ISAs) available, RISC-V seems to be the most promising one, with wide backing from academia and industry. The other available open source solutions are either dependent on specific technologies, have no parametrization options, or have non-robust development environments [8].

With the emergence of RISC-V [4] and its open source toolchain, with special emphasis on the Chisel Hardware Description Language (HDL) [9] and the Rocket Chip SoC Generator [10], it is likely that the success of open source OS's is extended to open CPU hardware descriptions.

A previous attempt to create a System on Chip (SoC) called Blackbird, and a programming environment for building reconfigurable systems using the OpenRISC open source processor has been made [8]. Blackbird was intended to become an independent SoC derived from ORPSoC [11], the OpenRISC equivalent of the Rocket Chip SoC Generator [10]. Unfortunately, this effort failed as the budget ran out, mainly because there was always dependencies on particular FPGA

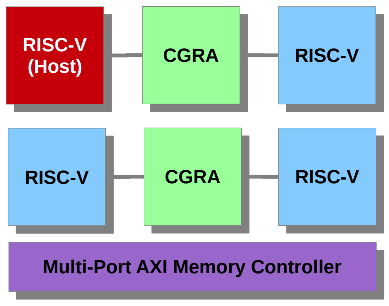

Fig. 1. Block diagram of 4-CPU/2-CGRA instance

boards or proprietary EDA tools which could not be supported in the Blackbird project. Also, the OpenRISC initiative has failed to create a mature enough ecosystem of its own, and the effort revealed itself too complex, as many essential parts of the system were either missing or still being developed by the small existing community. In fact, in spite of the initial enthusiasm, the same may happen to the RISC-V initiative. However, its current impetus is far stronger than OpenRISC has ever enjoyed, and today open source development is facilitated by web platforms such as Github.

The effort reported in [8] has been recently continued using RISC-V processors, with a fresh attempt to build a new SoC called Warpbird [12]. Unfortunately, the team has run into similar difficulties as when using the OpenRISC core, which led to the identification of a new research problem: building and untethred SoC [12]. The existing RISC-V open source projects proved not modular or documented enough so that its parts could easily be reused. In fact, it turned out to be easier to write a RISC-V core from scratch and the Adept project using the Chisel language was started. The system proposed in this paper is portable to any FPGA or ASIC technology as Chisel generates synthesizable Verilog code. The RISC-V CPU is supported by the GNU toolchain and the Versat CGRA has its own compiler and assembler [13].

II. Architecture

The proposed architecture can be used to develop SoCs containing several Adept RISC-V cores and Versat CGRAs. An example system consisting of 4 RISC-V cores and 2 Versat CGRAs is depicted in Fig. 1. The system uses ARM's Advanced eXtensible Interface (AXI) to interconnect the cores, a de facto standard for busing.

In the figure, two CPU cores share the same CGRA, which has a 2-port AXI slave interface connected to the AXI master interfaces of its hosts. The CGRA arbitrates between the two hosts using a round-robin scheme. All CPU and CGRA cores access the external memory by means of their AXI master interfaces connected to an AXI slave multi-port memory controller core, which is a 3rd party core. The external memory must be loaded with programs and data for all cores. This is done by the host CPU, which runs a boot loader program and is able to fetch these data from an external device by means of an AXI or SPI peripheral, not shown in the figure. The next subsections provide details about the RISC-V CPU core and the Versat CGRA core.

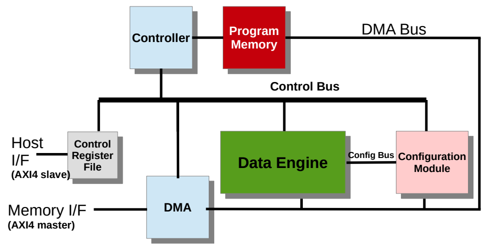

Fig. 2. Versat top-level entity.

A. Adept

Four RISC-V architectures have been studied in order to frame this work: Taiga [14], a high performance RISC-V soft core; Rocket Chip [10], the de facto standard used at UC Berkeley and respective SiFive startup; PicoRV32 [15], a size-optimized architecture; and PULPino [16], a single-core microcontroller system. The proposed SoC in this work had a very specific set of requirements: small size, low power, target FPGAs and ASICs and have a high abstraction level description. Therefore, Rocket Chip was not selected because it had too many features and proved too hard to manipulate; Taiga was not selected because it targets FPGAs only; PULPino and PicoRV32 were not selected because they are written in conventional HDL. In this project, the abstractions present in modern programming languages are a requirement, namely to facilitate parametrization of the RTL. Thus, after considering all 4 architectures, it was decided to implement a new RISC-V core from scratch using the RV32IM ISA and the new Chisel3 HDL [9]. The new architecture was called Adept.

There are two Adept configurations: Adept-3, a 3-stage pipeline low performance configuration which implements the Base Integer (I) instruction set; and Adept-5, a 5-stage pipeline medium performance configuration which implements the Integer Multiplication and Division (IM) instruction sets. Both configurations are optimized for silicon area and only support bare metal applications. Both use instruction and data caches of sizes 8 and 16kB, respectively.

B. Versat

The Versat architecture [5] is shown in Fig. 2. Versat uses the Controller to run programs and the Data Engine (DE) to carry out data intensive computations. The Controller programs are stored in the Program Memory (PM), which has a 1kB boot ROM and an 8kB user RAM. The Controller accesses the various modules in the system using the Control Bus. The boot ROM program can load user programs into the PM from the external memory or other device using the DMA or a special peripheral such as an SPI core. The user program can generate DE configurations for the various acceleration datapaths and store them in the Configuration Module (CM). It can also move configurations between the CM and the external memory using the DMA engine.

The Versat core has a host interface and a memory interface. The host interface (AXI slave) is used by a host system to instruct Versat to load and execute programs. Host and Versat communicate using the shared Control Register File

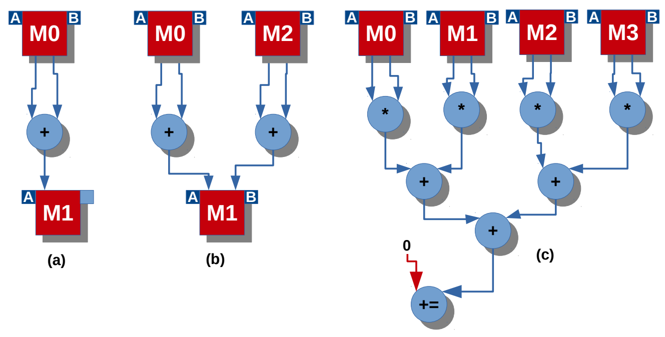

Fig. 3. Data engine datapaths.

(CRF). The CRF is also used by Versat programs as a general purpose (1-cycle access) register file. The memory interface (AXI master) is used by the DMA to access external memory.

Different Versat instances can be generated containing different DE modules. The DE consists of several Functional Units (FUs) interconnected in a full mesh. The current FU types are: dual-port embedded memory, ALU (with and without internal feedback to accumulate results), multiplier and barrel shifter.

The interconnect mesh has 1 clock cycle of delay and the FUs themselves are pipelined for better performance. The FUs are used to build hardware datapaths that work on data streams. Example datapaths are illustrated in Fig. 3. Datapath (a) in the figure is a simple pipeline exploiting Instruction-Level Parallelism (ILP) by performing *load, add* and *store* instructions in parallel; datapath (b) demonstrates ILP plus Data-Level Parallelism (DLP), or Thread-Level Parallelism (TLP) if the two additions are operated as independent threads; datapath (c) involves more FUs thus achieving more parallelism.

The full mesh structure may seem an overkill but it greatly facilitates the programming of Versat, dispensing with complex place and route algorithms. As such, Versat can even be programmed in its assembly language, which is a rare feature among CGRAs. Due to the relatively small number of FUs used, the full mesh interconnect becomes affordable, occupying only 5% of the chip area [5].

When implemented in FPGAs, Versat is an overlay architecture, that is, the reconfigurable FPGA fabric is used to support different Versat instances but the instances themselves are also reconfigurable using their own and much faster reconfiguration infrastructure. When in implemented in ASICs, Versat is a piece of programmable logic that reduces silicon area, lowers the risk of design errors and saves energy while offering hardware-like performance.

III. Toolchain

A. Adept

The Adept RISC-V system has a fully operational GNU toolchain for compilation, debugging and profiling. These are mature tools and most have been merged upstream and their status can be checked in [17] as of version 7.1. There are options to make compilation aware a the floating-point unit, clock frequency for a specific board, etc.

Debug support using GDB is already available for RISC-V but Adept does not yet have hardware support for it. A debug module and corresponding JTAG interface have not yet been incorporated.

When working with the Rocket Chip SoC Generator, the debug module and its software support via OpenOCD [18] were not inter-operable in the Warpbird setup [12], and therefore could not be integrated in Adept. The Rocket Chip debug module and JTAG interface needed to be made usable in a different environment or new ones developed from scratch.

The Adept system is described in Chisel3, a novel hardware description language [9]. Additionally, users can add components to the system using their preferred HDL. However, the authors recommend using Chisel as it automates and eases tasks, such as debugging, connections and interface generation.

B. Versat

Being a CGRA, the Versat development tools are non-standard. Versat can be programmed in assembly language [5] and using a C++ dialect [13].

Since the Versat Controller manages the Data Engine (DE) and the DMA engine via the Control Bus, these modules can be programmed indirectly. To program the DE, datapaths are written to the Configuration Module (CM). To operate the other modules, the Versat Controller accesses their memory mapped registers, as is usual when working with peripherals.

For example, to configure ALU0 to add the outputs of multipliers MUL1 and MUL2, the program needs to store the constant ALU_ADD to the ALU0_FNS function select configuration register, and the constants sMUL1 and sMUL2 to the configuration registers ALU0_selA and ALU0_selB. Versat's full mesh structure makes it possible that inputs of any FU can be connected to the output of any FU. If a partial mesh were used, the programmer would have to keep in mind what can be connected to what, making assembly programming not viable.

A debug tool for Versat has not yet been developed. Debugging a failing datapath can be a difficult task and a combination of techniques must be employed. The Versat Controller is instrumental in this process as it can start the DE and stop it at desired instants, and can read and write to any position of the DE memories. In extreme cases Versat must be debugged using RTL simulation or Integrated Logic Analyzers.

IV. Results

Results have been obtained for 4 different Versat instances described in Table I in terms of the number of ALUs (#ALU), the number of 32x32=64-bit multipliers (#MUL), the number of barrel shifters (#BS) and the number of embedded memories (#MEM).

A. FPGA Implementation Results

The FPGA implementation results for Intel ARRIA V and Xilinx KINTEX-7 devices are shown in Tables II and III, respectively.

As shown by these results, the resource usage is very reasonable. The Adept processors are comparable to standard FPGA

TABLE I
VERSAT INSTANCES

Instance	#ALU	#MUL	#BS	#MEM
Versat-1	6	4	1	4
Versat-2	14	5	5	5
Versat-3	8	4	4	5
Versat-4	8	8	0	5

TABLE II
VERSAT FPGA IMPLEMENTATION RESULTS FOR ARRIA V

Instance	Logic (ALMs)	#REG	RAM (kbit)	#DSP	Fmax (MHz)
Adept-3	670	201	133	0	93
Adept-5	1675	1470	266	4	120
Versat-1	8,607	4,673	351	32	130
Versat-2	15,340	12,620	623	25	120
Versat-3	13,700	11,270	623	21	120
Versat-4	10,680	11,200	623	36	120

processors such as Microblaze (Xilinx) or NIOS (Intel), and the Versat cores allow for compact implementations compared to custom hardware unusable for multiple functions. Adept-3 has not been optimized for frequency; its a critical path goes from the multiplexer in the write-back stage to the branch execute block of the instruction fetch stage and traverses the ALU. The critical path in Adept-5 is in the forwarding path from the memory stage to the register fetch stage.

The results show that Versat cores can be implemented in different sizes with minimal impact on the frequency of operation. However, due to the full mesh structure, the frequency is limited to the 90-130 MHz range, depending on the complexity of the instances.

B. ASIC Implementation Results

Versat and Adept have been implemented in the UMC 130 nm and TSMC 65 nm processes. Table IV shows the results in terms of the technology node (N), silicon area (A), embedded memory (M), frequency of operation (F) and power consumption (P). The frequency and power results have been obtained using the Cadence IC design and simulation tools. The power figure has been obtained using the same tools and the node activity rate extracted from simulation.

The results show that the proposed heterogeneous computing platform is competitive regarding silicon area and power consumption. A few Adept and Versat cores consumes a few hundred milliwatt in a 65nm process, while a *single* ARM Cortex A9 core can consume about the same and occupy a larger silicon area in a 40nm process [5].

1) Applications: The present heterogenous computing platform targets applications for low cost battery operated devices. These applications can benefit from the low energy features of CGRAs. The applications that have been tested in the present architecture are outlined in Table V.

The present platform is compared with an ARM Cortex-A9 processor running the same applications. Unfortunately, it was not possible to compare with other CGRAs [3] because they were not available to be programmed with the same applications. Comparisons using published results were made in [5] and it was concluded that they depend on the resources of the CGRAs being compared. In [19], the multi-dimensional K-Means algorithm was shown to run 3.8x faster while consuming 46.3x less energy running on Versat when compared to the ARM Cortex-A9 system [20], [21]. Here, the results obtained for HE-AAC 5.1 audio encoder and decoder systems are shown in Table VI. The results for the Adept+Versat systems have been normalized to 40nm to facilitate direct comparison.

TABLE III
VERSAT FPGA IMPLEMENTATION RESULTS FOR KINTEX-7

Instance	Logic (LUTs)	#REG	RAM (kbit)	#DSP	Fmax (MHz)
Adept-3	1041	188	133	0	68
Adept-5	2533	1405	266	4	90
Versat-1	12,510	4,396	360	16	102
Versat-2	23,750	13,158	630	25	90
Versat-3	21,184	11,740	630	21	90
Versat-4	16,647	11,699	630	36	90

TABLE IV
ADEPT INTEGRATED CIRCUIT IMPLEMENTATION RESULTS.

Core	N(nm)	A(mm^2)	M(kB)	F(MHz)	P(mW)
Adept-3	130	1.61	16	200	48
Adept-3	65	0.40	16	400	24
Adept-5	130	3.68	32	210	121.8
Adept-5	65	0.92	32	420	58
Versat-1	130	5.20	46	90	132
Versat-1	65	5.20	46	170	132
Versat-2	130	9.52	76	256	355
Versat-2	65	2.38	76	500	179
Versat-3	130	8.54	76	261	330
Versat-3	65	2.18	76	510	167
Versat-4	130	8.30	68	258	321
Versat-4	65	2.10	68	510	160

For the HE-AAC 5.1 audio decoder, the ARM system is 2.6x larger than the proposed implementation and consumes 1.5x more power. For the HE-AAC 5.1 audio encoder, the ARM system is 2.4x larger than the proposed implementation and consumes 1.9x more power. The ARM Cortex A9 system uses the NEON SIMD unit optimized at the assembly level. The proposed system needs to use a slightly higher operation frequency and much more memory than the ARM system. However, this is largely compensated by the use of simple CPUs and data centric CGRAs which consume much less logic resources.

V. CONCLUSION

In this work a low power heterogeneous system is proposed, composed of RISC-V processors and Versat CGRA cores, targeting any design flow (FPGA or ASIC). The designed

TABLE V
AUDIO ENCODERS AND DECODERS

Audio Format	#CPUs	#CGRAs
MPEG1/2 Layers I/II encoder/decoder (2 chs)	1	0
AAC-LC stereo/multichannel decoder (6 chs)	1	0
AAC-LC stereo encoder (2 chs)	1	0
AAC-LC multichannel encoder (6 chs)	3	0
HE-AAC stereo encoder/decoder (2 chs)	1	1
HE-AAC multichannel encoder/decoder (6 chs)	3	1
AC3 stereo encoder/decoder (2 chs)	1	1
AC3 multichannel encoder/decoder (6 chs)	3	1
K-Means Clustering	0	1

TABLE VI
IMPLEMENTATION COMPARISON FOR HE-AAC 5.1 AUDIO CODECS

Core (40nm) / Software	Area (mm^2)	RAM (kB)	Freq. (MHz)	Power (mW)
ARM Cortex-A9 / Decoder	4.6	64	50	31.25
ARM Cortex-A9 / Encoder	4.6	64	70	43.75
3xAdept-5 + Versat-4 / Decoder	1.75	164	80	20.76
3xAdept-5 + Versat-2 / Encoder	1.95	172	80	23.31

RISC-V core, Adept, has two configurations: a small 3-stage pipeline configuration, and a more performant 5-stage pipeline configuration. In both instances, Adept possesses a competitive energy footprint and resource usage. The Versat CGRA allows for a large set of applications to be executed with minimal energy requirements and high performance.

The RISC-V ISA enables the development of SoCs that use one or more processor instances, free of license fees or royalties. Moreover, the RISC-V ISA is supported by a rich open source toolchain and a thriving developer community. In this work, it was concluded that it is still better to develop the RTL in-house as the available open source projects are difficult to manipulate and extend.

When implemented in FPGA, the present platform constitutes an overlay architecture. Applications run slower on it compared to when directly mapped to the FPGA fabric. However, using fast on-the-fly reconfiguration, Versat is able to multiplex large virtual hardware circuits into its small CGRA, and enable applications that otherwise would not fit in the FPGA. The implementation results show that it is possible to instantiate several RISC-V and Versat cores in mid-range FPGAs and run real-world applications such as the audio encoders and decoders presented herein. When implemented as an ASIC, the proposed heterogeneous computing platform consumes close to 3x less silicon area and 2x less energy compared to an ARM Cortex-A9 system equipped with the NEON SIMD unit.

ACKNOWLEDGMENT

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under projects PTDC/EEI-HAC/30848/2017 and UID/CEC/50021/2019.

REFERENCES

[1] Sparsh Mittal and Jeffrey S. Vetter. A Survey of CPU-GPU Heterogeneous Computing Techniques. *ACM Comput. Surv.*, 47(4):69:1–69:35, July 2015.

[2] Jiantao Qiu, Jie Wang, Song Yao, Kaiyuan Guo, Boxun Li, Erjin Zhou, Jincheng Yu, Tianqi Tang, Ningyi Xu, Sen Song, Yu Wang, and Huazhong Yang. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network. In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '16, pages 26–35, New York, NY, USA, 2016. ACM.

[3] Bjorn De Sutter, Praveen Raghavan, and Andy Lambrechts. Coarse-Grained Reconfigurable Array Architectures. In Shuvra S. Bhattacharyya, Ed F. Deprettere, Rainer Leupers, and Jarmo Takala, editors, *Handbook of Signal Processing Systems*, pages 449–484. Springer US, 2010.

[4] Editors Andrew Waterman and Krste Asanović. *The RISC-V Instruction Set Manual, Volume I: User-Level ISA, Document Version 2.2.* RISC-V Foundation, May 2017.

[5] João D. Lopes and José T. de Sousa. Versat, a Minimal Coarse-Grain Reconfigurable Array. In *Proc. of the 12th Int. Meeting on High Performance Computing for Computational Science*, VECPAR, Porto, Portugal, June 2016.

[6] Waqar Hussain, Roberto Airoldi, Henry Hoffmann, Tapani Ahonen, and Jari Nurmi. Design of an Accelerator-Rich Architecture by Integrating Multiple Heterogeneous Coarse Grain Reconfigurable Arrays over a Network-on-Chip. In *Application-specific Systems, Architectures and Processors (ASAP), 2014 IEEE 25th International Conference on*, pages 131–138. IEEE, 2014.

[7] Krste Asanović and David A Patterson. Instruction Sets Should be Free: The Case for RISC-V. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146*, 2014.

[8] José T. de Sousa, Carlos A. Rodrigues, Nuno Barreiro, João C. Fernandes. Building Reconfigurable Systems Using Open Source Components. In *REC*, April 2014.

[9] Jonathan Bachrach, Huy Vo, Brian Richards, Yunsup Lee, Andrew Waterman, Rimas Avižienis, John Wawrzynek, and Krste Asanović. Chisel: Constructing Hardware in a Scala Embedded Language. In *49th Design Automation Conference*, June 2012.

[10] Krste Asanović, Rimas Avižienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, Sagar Karandikar, Ben Keller, Donggyu Kim, John Koenig, Yunsup Lee, Eric Love, Martin Maas, Albert Magyar, Howard Mao, Miquel Moreto, Albert Ou, David A. Patterson, Brian Richards, Colin Schmidt, Stephen Twigg, Huy Vo, and Andrew Waterman. The Rocket Chip Generator. Technical Report UCB/EECS-2016-17, EECS Department, University of California, Berkeley, Apr 2016.

[11] OpenRISC community. OpenRISC Reference Platform System-On-Chip, 2013.

[12] Luís Fiolhais, José T. de Sousa. Warpbird: an Untethered System on Chip Using RISC-V Cores and the Rocket Chip Infrastructure. In *REC*, pages 42–49, January 2018.

[13] Rui Santiago, José T. de Sousa, and João D. Lopes. Compiler for the Versat Architecture. In *XIII Jornadas de Sistemas Reconfiguráveis*, pages 41–48, January 2017.

[14] Eric Matthews and Lesley Shannon. TAIGA: A new RISC-V soft-processor framework enabling high performance CPU architectural features. In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, pages 1–4, Sept 2017.

[15] Clifford Wolf and et. al. PicoRV32, 2015. GIT code repository.

[16] Michael Gautschi, Pasquale D. Schiavone, Andreas Traber, Igor Loi, Antonio Pullini, Davide Rossi, Eric Flamand, Frank K. Grkaynak, and Luca Benini. Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(10):2700–2713, Oct 2017.

[17] RISC-V Foundation. RISC-V Software Ecosystem Overview.

[18] Dominic Rath. Open On-Chip Debugger. Master's thesis, University of Applied Sciences of Augsburg, July 2005.

[19] João D. Lopes, José T. de Sousa, and Horácio Neto. K-Means Clustering on CGRA. In *Proceedings of the 27th International Conference on Field-Programmable Logic and Applications, New Paradigms and Compilers*, FPL 2017, pages 1–4, September 2017.

[20] Wei Wang and Tanima Dey. A Survey on ARM Cortex A Processors. http://www.cs.virginia.edu/skadron/cs8535s11/armcortex.pdf. Accessed 2016-04-16.

[21] ARM. ARM, Dolby and Ittiam Collaborate To Bring Optimized Home Audio Solutions To Market, 2010. Press Release.