TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

M. Waqar Hussain

Design and Development from Single Core Reconfigurable Accelerators to a Heterogeneous Accelerator-Rich Platform

Julkaisu 1263 • Publication 1263

Tampere 2014

Tampereen teknillinen yliopisto. Julkaisu 1263
Tampere University of Technology. Publication 1263

M. Waqar Hussain

Design and Development from Single Core Reconfigurable Accelerators to a Heterogeneous Accelerator-Rich Platform

Thesis for the degree of Doctor of Science in Technology to be presented with due permission for public examination and criticism in Tietotalo Building, Auditorium TB109, at Tampere University of Technology, on the 27th of November 2014, at 12 noon.

Tampereen teknillinen yliopisto - Tampere University of Technology
Tampere 2014

ISBN 978-952-15-3406-5 (printed)
ISBN 978-952-15-3422-5 (PDF)
ISSN 1459-2045

Contact Information

M. Waqar Hussain
Department of Electronics and Communications Engineering
Tampere University of Technology
P.O. Box 553
FIN-33101 Tampere
Finland
tel: +358 40 198 1872 (office)
+358 40 465 2116 (mobile)
e-mail: waqar.hussain (at) tut dot fi

Supervisor:

Jari Nurmi

Professor, D.Sc. (Technology)
Department of Electronics and Communications Engineering
Tampere University of Technology
Finland
Email: jari.nurmi(at)tut dot fi

Pre-examiners:

Gerard J. M. Smit

Professor, Ph.D.
Department of Electrical Engineering, Mathematics and Computer Science
University of Twente
The Netherlands
Email: g.j.m.smit(at)utwente dot nl

Pasi Liljeberg

Adjunct Professor, D.Sc. (Technology)
Computer Systems Laboratory
Department of Information Technology
University of Turku, Finland
Email: pakrli(at)utu dot fi

Opponents:

Dr.-Ing. Mario Porrmann

Professor, Ph.D.
Faculty of Technology, 33594 Bielefeld, Germany
Universität Bielefeld
Germany
Email: mporrman(at)techfak.uni-bielefeld.de

Pasi Liljeberg

Adjunct Professor, D.Sc. (Technology)
Computer Systems Laboratory
Department of Information Technology
University of Turku, Finland
Email: pakrli(at)utu dot fi

Dedicated to those in Muslim countries who are struggling for Rationality, Democracy, Freedom and Discovery...

ACKNOWLEDGMENTS

This research work was partially conducted at the **Department of Computer Science, University of Chicago, Illinois, USA** and at the **Institute for Communication Technologies and Embedded Systems, Rheinisch-Westfaelische Technische Hochschule, Aachen, Germany** under the supervision of Prof. Dr. Henry Hoffmann and Prof. Dr.-Ing. Gerd Ascheid, respectively. Their contribution for providing financial and on-site resources is greatly acknowledged.

The major part of the thesis was conducted under the supervision of Prof. Dr. Jari Nurmi at the **Department of Electronics and Communications Engineering, Tampere University of Technology, Tampere, Finland**.

TABLE OF CONTENTS

PREFACE

At first, I acknowledge my mother Anisa Zahid and my father Zahid Hussain for their patience and consistent struggle, love and support to bring me up to a stage that I became able enough to conduct this research and write this manuscript.

I am really thankful to my supervisor Prof. Jari Nurmi to believe in my abilities and for providing finance throughout my research in Tampere University of Technology, Finland and abroad. I am also thankful to Prof. Dr.-Ing Gerd Ascheid and Dr. Henry Hoffmann for being my supervisors at RWTH, Aachen, Germany and University of Chicago, IL, USA. I express my thanks to the reviewers of this thesis, Prof. Dr. Gerard J. M. Smit, University of Twente, Netherlands and Prof. Dr. Pasi Liljeberg, University of Turku, Finland for providing valuable comments to the draft of this manuscript.

I thank Dr. Fabio Garzia, Dr. Tapani Ahonen and Dr. Roberto Airoldi for being my helpful colleagues, introducing new research topics to me and collaborating for joint research work. It also extends to Prof. Dr. João M. P. Cardoso, University of Porto, Portugal, Prof. Dr. Guy Gogniat, Universitè de Bretagne-Sud - UEB, France and Prof. Dr.-Ing. Diana Göhringer, Ruhr Universität, Bochum, Germany for their support in organizing conference special-sessions.

Dr. Nadeem Lehrasab, a brilliant academic who selflessly spends days and nights in scientific research. I can surely grant him the credit to introduce me the world of science and technology in the early days of my professional carrier. Dr. Zafar Mahmood has been my time tested friend, I truly acknowledge his wisdom, honestly, sense in understanding the world and enthusiasm

in every field of life.

The most beautiful days of my life in Finland were spent in friendship with Andrea Milanti. I am thankful to him for being an honest and lovely friend. Matteo Maggioni has been my very caring and amazing friend. He was like my life support in Chicago when we were there together in autumn, winter and spring of 2013-14. Davide Fantozzi is a true gentleman and a fine friend. Dennis Surmann, a great friend from Munich. The future generations will be proud of his scientific research. To my most loving friend Rizwan Fazal, the time we spent together cannot simply be forgotten. I greatly admire Suvi Jokinen for her consistent friendship. I thank Bruno Di Buó for being an interesting and honest friend. I wish Khashayar Khanlari to be happy and successful forever. It was a great happiness to be friends with Serena, Romain, Ugur, Marco and Juho Lauri. It was a pleasure to know Cliff Magori Ogutu and his nice music collection that I always enjoyed during the weekends in Finland. I thank Michal Novak for such nice memories in Tampere, Finland. Wajid Ali, Waseem Haider, Saadi Rehman, Moeed Khan Pasha, Asif Nawaz and Usman Rahim, thank you for your long support and friendship, it will always be remembered. I also extend my thanks to Fawad Mazhar for great memories in Riga, Latvia.

My friends from Chicago USA, Chiara, Marco, Graziano, Connor, Sean, Gökalp, Huazhe and Severin, thank you for your friendship and socializing with me. Living in Chicago would have been too colorless without you.

Apart from loving friends, visit to different countries was always a pleasure. Contributing to different conferences provided me an opportunity to see important places in the world that I always wondered. I believe, the recreational activities are important to gain back enough energy to address the latest unsolved problems in science. I remember the beautiful MuseumsQuartier in Vienna, Austria when I visited there for my first conference trip. The evenings spent at Gaslamp Quarter, San Diego, California - a memorable drive towards Hollywood, Los Angeles and a pleasant walk at the Hollywood Walk of Fame. On return, I had an opportunity to visit Times Square, Manhattan at the New York City. The York Minster in England is the most beautiful cathedral I

have ever seen followed by a short visit to Oxford Street in London. The most breath taking moment in my life was the sunset that I witnessed at Santorini island in Greece. I lived in Aachen, Germany for most of the time during my research visit at RWTH University but my memories are attached to Karow, Pankow, Alexander Platz and Schönhauser Allee in Berlin. Meanwhile, I visited many cities and countries for leisure. Apart from Berlin, I have beautiful memories from Riga, Latvia. Nevertheless, Kaunas and Vilnius of Lithuania are very amazing cities. The list of leisure trips is long but Stockholm-Sweden and Tromsø-Norway trips with friends were the most memorable.

For all these wonderful experiences, I paid a cost - my brothers Awais and Bilal, my sister Fozia, my brother-in-law Ali Zeeshan, my niece and nephews Zainab, Karrar and Abbas, I love you all and staying away from you has been a big compromise in my life to bring substance to this research work.

Waqar Hussain

15th October 2014

ABSTRACT

The performance of a platform is evaluated based on its ability to deal with the processing of multiple applications of different nature. In this context, the platform under evaluation can be of homogeneous, heterogeneous or of hybrid architecture. The selection of an architecture type is generally based on the set of different target applications and performance parameters, where the applications can be of serial or parallel nature. The evaluation is normally based on different performance metrics, e.g., resource/area utilization, execution time, power and energy consumption. This process can also include high-level performance metrics, e.g., Operations Per Second (OPS), OPS/Watt, OPS/Hz, Watt/Area etc. An example of architecture selection can be related to a wireless communication system where the processing of computationally-intensive signal-processing algorithms has strict execution-time constraints and in this case, a platform with special-purpose accelerators is relatively more suitable than a typical homogeneous platform.

A couple of decades ago, it was expensive to plant many special-purpose accelerators on a chip as the cost per unit area was relatively higher than today. The utilization wall is also becoming a limiting factor in homogeneous multicore scaling which means that all the cores on a platform cannot be operated at their maximum frequency due to a possible thermal meltdown. In this case, some of the processing cores have to be turned-off or to be operated at very low frequencies making most of the part of the chip to stay underutilized. A possible solution lies in the use of heterogeneous multicore platforms where many application-specific cores operate at lower frequencies, therefore reducing power dissipation density and increasing other performance parameters. However, to achieve maximum flexibility in processing, a general-purpose

flavor can also be introduced by adding a few Reduced Instruction-Set Computing (RISC) cores. A power class of heterogeneous multicore platforms is an accelerator-rich platform where many application-specific accelerators are loosely connected with each other for work load distribution or to execute the tasks independently.

This research work spans from the design and development of three different types of template-based Coarse-Grain Reconfigurable Arrays (CGRAs), i.e., CREMA, AVATAR and SCREMA to a Heterogeneous Accelerator-Rich Platform (HARP). The accelerators generated from the three CGRAs could perform different lengths and types of Fast Fourier Transform (FFT), real and complex Matrix-Vector Multiplication (MVM) algorithms. CREMA and AVATAR were fixed CGRAs with eight and sixteen number of Processing Element (PE) columns, respectively. SCREMA could flex between four, eight, sixteen and thirty two number of PE columns. Many case studies were conducted to evaluate the performance of the reconfigurable accelerators generated from these CGRA templates. All of these CGRAs work in a processor/coprocessor model tightly integrated with a Direct Memory Access (DMA) device. Apart from these platforms, a reconfigurable Application-Specific Instruction-set Processor (rASIP) is also designed, tested for FFT execution under IEEE-802.11n timing constraints and evaluated against a processor/coprocessor model. It was designed by integrating AVATAR generated radix-(2, 4) FFT accelerator into the datapath of a RISC processor. The instruction set of the RISC processor was extended to perform additional operations related to AVATAR.

As mentioned earlier, the underutilized part of the chip, now-a-days called *Dark Silicon* is posing many challenges for the designers. Apart from software optimizations, clock gating, dynamic voltage/frequency scaling and other high-level techniques, one way of dealing with this problem is to use many application-specific cores. In an effort to maximize the number of reconfigurable processing resources on a platform, the accelerator-rich architecture HARP was designed and evaluated in terms of different performance metrics. HARP is constructed on a Network-on-Chip (NoC) of 3×3 nodes where with

every node, a CGRA of application-specific size is integrated other than the central node which is attached to a RISC processor. The RISC establishes synchronization between the nodes for data transfer and also performs the supervisory control. While using the NoC as the backbone of communication between the cores, it becomes possible for all the cores to address each other and also perform execution simultaneously and independently of each other.

The performance of accelerators generated from CREMA, AVATAR and SCREMA templates were evaluated individually and also when attached to HARP's NoC nodes. The individual CGRAs show promising results in their own capacity but when integrated all together in the framework of HARP, interesting comparisons were established in terms of overall execution times, resource utilization, operating frequencies, power and energy consumption. In evaluating HARP, estimates and measurements were also made in some advanced performance metrics, e.g., in MOPS/mW and MOPS/MHz. The overall research work promotes the idea of heterogeneous accelerator-rich platform as a solution to current problems and future needs of industry and academia.

LIST OF PUBLICATIONS AND AUTHOR'S CONTRIBUTION

This thesis is mainly based on the following publications.

[P1] W. Hussain, T. Ahonen F. Garzia and J. Nurmi, "Application-Driven Dimensioning of a Coarse-Grain Reconfigurable Array", in Proc. of NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2011), pp. 234-239, San Diego, California, USA.

[P2] W. Hussain, F. Garzia, T. Ahonen, J. Nurmi, "Designing Fast Fourier Transform Accelerators for Orthogonal Frequency-Division Multiplexing Systems", Journal of Signal Processing Systems for Signal, Image and Video Technology, Springer, Vol. 69, pp. 161-171, November 2012.

[P3] W. Hussain, T. Ahonen, R. Airoldi and J. Nurmi, "Energy and Power Estimation of Coarse-Grain Reconfigurable Array based Fast Fourier Transform Accelerators", in Proc. of 2012 International Symposium on Reconfigurable Communication-centric Systems-on-Chip, pp. 1-4, York, England, July 2012.

[P4] W. Hussain, T. Ahonen and J. Nurmi, "Effects of Scaling a Coarse-Grain Reconfigurable Array on Power and Energy Consumption", in Proc. of SoC 2012, Tampere, Finland, Oct 2012.

[P5] W. Hussain, X. Chen, G. Ascheid, J. Nurmi, "A Reconfigurable Application-specific Instruction-set Processor for Fast Fourier Transform Processing", in Proc. of 2013 IEEE 24th International Confer-

ence on Application-Specific Systems, Architectures and Processors (ASAP), pp. 339-345, 5-7 June 2013, Washington, D.C., USA.

[P6] W. Hussain, R. Airoldi, H. Hoffmann, T. Ahonen, J. Nurmi, "Design of an Accelerator-Rich Architecture by Integrating Multiple Heterogeneous Coarse Grain Reconfigurable Arrays over a Network-on-Chip", 2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors (ASAP), pp. 131-138, 18-20 June 2014, Zurich, Switzerland.

The author of this thesis has the major contribution in the above mentioned publications that includes establishing and conducting the entire experimental work followed up by writing of the papers. The co-authors contributed to reviewing and providing the feedback.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ALM	Adaptive Logic Module
ALUT	Adaptive Look-Up Table
ASIP	Application Specific Instruction-set Processor
CGRA	Coarse Grain Reconfigurable Array
COFFEE	COre For FrEE
CREMA	Coarse-grain REconfigurable array with Mapping Adaptiveness
DCT	Discrete Cosine Transform
DMA	Direct Memory Access
DSP	Digital Signal Processing
FF	Flip Flop
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array

FU	Functional Unit
GOPS	Giga Operations Per Second
GPP	General Purpose Processor
GUI	Graphical User Interface
HARP	Heterogeneous Accelerator-Rich Platform
I/O	Input/Output
ISA	Instruction-Set Architecture
LE	Logic Element
LUT	Look-Up Table
MAC	Multiply ACcumulate
MFC	Multi-Function logic Cell
MIMO	Multiple Input Multiple Output
MOPS	Millions of Operations Per Second
MPSoC	Multi-Processor System-on-Chip
MVM	Matrix-Vector Multiplication
NoC	Network-on-Chip
OFDM	Orthogonal Frequency-Division Multiplexing
PE	Processing Element

PLL	Phase Looked Loop
rASIP	reconfigurable Application Specific Instruction-set Processor
RC	Reconfigurable Cell
RDU	Reconfigurable Datapath Unit
RF	Register File
RISC	Reduced Instruction-Set Computing
RTL	Register Transfer Level
SRAM	Synchronous Random Access Memory
WCDMA	Wideband Code-Division Multiple Access
VHDL	Very high-speed integrated circuit Hardware Description Language
VLIW	Very Long Instruction Word

1. INTRODUCTION

The human society is influenced by and relies on computers than ever before for an organized living. Its use ranges from everyday life to discovering sub-atomic particles, fields and events that led to the creation of our universe. One side of the picture is the telecommunication industry promoting their smart-phones and competing on basis of operating system, processing power and store of applications. Smart-phones fall into the category of *Embedded Systems* that are roughly defined as systems with specialized hardware and software. Since embedded systems offer a deterministic execution time for every task, there is an increasing demand to maximize the number of such tasks that can be supported on an embedded device. In single core architectures, there was a trend for many years to increase the system performance by scaling voltage and frequency of the system. This trend led to a constraint i.e., maximum limit of power density per unit area of a chip on an available technology. In the past few years, the vendors have come up with multicore systems to avoid power-density/area constraint. Currently, there is an apparent problem with multicore scaling in terms of *Dark Silicon*. In a recent study, it is shown that on a single chip, all cores cannot be clocked at their maximum possible operating frequency due to a possible meltdown threat posed by thermal power dissipation [1]. As a result, most of the chip has to be powered-off or forced to operate at very low frequency. The dark silicon area can be replaced by planting many application-specific accelerators that can operate at very low frequencies and reduce the execution time of a kernel significantly. As motivated, this work focuses on the design of template-based Coarse-Grain Reconfigurable Arrays (CGRA) to generate special-purpose accelerators for use in single core and multicore platforms. The CGRAs consume relatively

lower power as they are designed to be efficient to process signal processing related algorithms.

1.1 Objective and Scope of Research

The utilization-wall issue raises many questions about the limitations in a platform for maximum performance. The theoretical understanding of utilization-wall provides generalization of a specific group of architectures that will help to reduce its effects. Examples of such architectures can be realized by implementing many application-specific accelerators on a chip, use of multiple template-based CGRAs, maximizing the number of computational resources and intercore communication using a Network-on-Chip. In this context, substantial research work has been conducted to design scalable CGRAs and design of a heterogeneous accelerator-rich platform in order to maximize the number of processing resources in a platform. These platforms are thoroughly evaluated against several performance metrics. The next chapters contain detailed description about their design, development and evaluation. The research work also focuses on some relatively general issues, e.g., the effects of loosely and tightly coupling accelerator(s) to a processor. In this regard, accelerators were designed from the same template-based CGRAs (using a process that will be discussed in Chapter 3) and then used as coprocessors. The overall development process passes through an evolution from a relatively smaller CGRA towards large scale CGRAs. It then moved to designing many heterogeneous CGRA-based accelerators that are loosely coupled with each other and to a Reduced Instruction-Set Computing (RISC) core over a Network-on-Chip. All of the architectures are C programmable. The scope of this thesis is wide and extends to a comparative analysis based on the measurement and estimation of different performance metrics and mapping of computationally-intensive signal processing algorithms.

1.2 Main Results

The main results from thesis can be highlighted as follows

1. Introducing application-driven scalability in CGRAs.
2. Implementation of different lengths and types of Fast Fourier Transform (FFT) algorithms considering timing constraints of IEEE-802.11x standards.
3. Measurement and Estimation of different performance metrics, i.e., power, energy, resource utilization, operating frequency and execution time of cores prototyped on Field Programmable Gate Array (FPGA).
4. Designing a heterogeneous accelerator-rich architecture by integrating multiple CGRAs over an NoC.
5. Constraint-driven frequency scaling in CGRAs.

1.3 Thesis Outline

The thesis consists of chapters that highlight selected contributions made in form of scientific publications. Chapter 2 presents important literature related to the work presented in the next chapters. Chapter 3 explains the structure of CGRAs and in particular the CGRAs designed by the author and also a case-study related to a reconfigurable Application-Specific Instruction-set Processor (ASIP). Chapter 4 is about an accelerator-rich platform supported by a Network-on-Chip (NoC) infrastructure. Chapter 5 presents details related to the platforms designed and their evaluation in terms of different performance metrics. In Chapter 6, conclusions and the future work is presented.

2. LITERATURE REVIEW

Computationally intensive tasks are often assigned to Multi-Processor System on Chip (MPSoC) or accelerators working in a processor/coprocessor model. One powerful class of accelerators is Coarse-Grain Reconfigurable Array (CGRA) which is ideal for signal processing applications as it provides high parallelism and throughput. General-purpose CGRAs occupy an area of a few million gates and their presence in the system cannot be justified unless they are heavily utilized most of the time. One of the examples of general-purpose CGRA is BUTTER [14] which was developed at Tampere University of Technology, Finland followed by CREMA [38] which was a template-based CGRA. Some other examples of CGRAs are Morphosys [16], ADRES [20] and PACT-XPP [23]. Furthermore, the CGRAs working as coprocessor or in stand-alone can be integrated to form a heterogeneous multicore platform to process many applications simultaneously and independently of each other unless inter-core communication is required for a certain application.

2.1 Processor/Coprocessor Models

Single core processors have been in use for decades. They started with general-purpose approach targeting many applications. However, for some computationally intensive applications, dedicated accelerators were used, e.g., audio, video and internet streaming. As the time passed, different kinds of processors emerged for a different set of requirements. For large-scale parallel applications, Very Long Instruction Word (VLIW) machines were developed and for

Digital Signal Processing (DSP) applications, DSP processors were designed with one or more Multiply and Accumulate (MAC) units. VLIW machines are dated back to early 1980s from the work by Josh Fisher in Yale University [51]. To support high-end mobile communication applications, a mix of DSP and VLIW architectures were developed which should stream multiple applications simultaneously [52]. In the last decade, the processors are generally designed with integrated accelerators which operate either in loose or tight coupling. Both of the cases are discussed below.

In tight coupling, the accelerators have high bandwidth for communication with the processor. It enables faster data transfer and synchronization. A large bandwidth interface is expensive and therefore multi and many accelerator cores working all together will not be feasible unless otherwise required. The tight integration can be achieved using a Direct Memory Access (DMA) device engine or directly integrating the accelerator in the datapath of the processor. For example, in [53] and [54], a reconfigurable engine is integrated into the datapath of a processor by extending its instruction set. As another example, a CGRA is tightly integrated with a processor using a network of switched interconnections [55].

Loosely Coupled accelerators have a low bandwidth to communicate with the processor but in this way multiple accelerators can be connected to the processor. Loose Coupling can be achieved by interfacing accelerator to the system bus or on the node of a local or remote network. In this way, multiple accelerators can work simultaneously and also exchange data with each other. An example of a Network-on-Chip (NoC)-based loosely coupled architecture is P2012, which is a platform of four clusters and each cluster consists of four cores [28].

2.2 Reconfigurable Devices

Reconfigurability is the ability of digital hardware to switch its functionality at run-time for different applications. In simplest form, it can be established

if a 2-to-1 multiplexer is used to produce an output with inputs from two different combinational circuits and the select line of the multiplexer switches as required. This approach is used to build-up many complicated devices to perform many different tasks. For example, FPGAs are developed by a few vendors and they have a large number of applications in different industries and in academia. The reconfigurable devices can be roughly classified based on their level of granularity into three different categories; Fine ($<$ 4-bits), Middle ($\leq$ 8-bits) and Coarse-Grain ($>$ 8-bits) [26].

2.2.1 Fine Grain Devices

An example of a Fine-Grain device is an FPGA which has been in the market for a few decades. The smallest unit of processing in an FPGA is called a Logic Element (LE) which contains a Look-Up Table (LUT), few logic gates, 2-to-1 multiplexers and also a few Flip-Flops (FFs). The neighboring LEs are connected using local interconnections while LEs which are relatively far from each other are connected using global interconnections. The two well known FPGA vendors are Xilinx [56] and Altera [57]. Xilinx focuses on resource utilization while Altera targets higher synthesis frequencies in their tools [49].

GARP [6] is a fine-grain device that offers a low granularity, i.e., 2-bit and 4-input LUT. The connectivity on its fabric is limited to maintain a fixed operating frequency [5]. It acts as a reconfigurable coprocessor that has access to processor's data memory while the processor has the control of reconfiguration and execution by the coprocessor. Another fine-grain device is FlexEOS which is SRAM-based fabric, re-programmable using VHDL and Verilog [7]. It was designed using standard CMOS technology for integration into a System-on-Chip (SoC). The building block of FlexEOS is a Multi-Function logic Cell (MFC) which has 7 inputs, 1 output and contains a 4-input LUT and a D-type FF. MOLEN is another example of fine-grain reconfigurable unit that acts a coprocessor to a General-Purpose Processor (GPP) ([8], [9]). The Instruction-Set Architecture (ISA) of the GPP is extended

to add special instructions to operate on the reconfigurable unit while some registers are also shared between the GPP and MOLEN.

2.2.2 Middle Grain Devices

In middle-grain reconfigurable devices, the word length is less than or equal to eight which favors seamless mapping of algorithms with the same processing word length. However as the processing width increases, the difficulty in mapping the algorithm also increases due to irregular subword-length computing. Middle-grain devices are a good compromise between area and performance. PiCoGA-III is a mid-grain reconfigurable datapath unit composed of a matrix of Reconfigurable Datapath Unit (RDU), where each RDU contains a 4-bit LUT, a 4-bit ALU and a 4-bit integer and Galois field multiplier ([10], [11]). DART is a mixed-grain reconfigurable processing engine supporting both 8-bit and 16-bit processing capability ([12], [13]). The mixed-grain paradigm is to support different word length applications while minimizing power dissipation and maximizing performance.

2.2.3 Coarse Grain Devices

Coarse-Grain Reconfigurable Arrays (CGRA) are one of the most successful platforms in industrial and academic research. This is mostly because they have a high-level of granularity and therefore a number of different applications can be easily targeted on them. CGRAs have an academic track-record of processing many data parallel applications, e.g., Fast Fourier Transform (FFT) processing while satisfying execution-time constraints of IEEE-802.11a/g standard [3], Wideband Code Division Multiple Access (WCDMA) cell search [41], image and video processing [14], [59], Turbo Codes [61] and Finite Impulse Response (FIR) filtering [60]. CGRAs offer a large bandwidth and high throughput processing. However, they also occupy a large area of a few million gates and can have a potentially high transient power dissipation. Some of the examples of CGRAs found in literature are BUTTER

([14], [15]), Morphosys ([16], [17], [18], [19]), ADRES ([20], [21], [22]) and PACT-XPP ([23], [24], [25]).

ADRES

ADRES is considered as a widely experimented CGRA presented in literature by different research groups across the world. It is a reconfigurable array of 8×8 elements where the elements consists of functional units, Register Files (RFs) and routing resources. The routing resources include wires, buses and networks. Specific instances from ADRES can be generated using an XML-based architecture specification language. At the top level, ADRES tightly integrates with a Very Long Instruction Word (VLIW) processor. In ADRES, only fixed-point operations are supported which can process multimedia and telecommunication application's word length. The RFs are for temporary data storage. There are 1-bit RFs that store the predicate signal to support iterative scheduling. The other RFs are used for intermediate data storage. ADRES is a flexible platform performing at 40 Mega Operations Per Second (MOPS)/mW with 90nm technology.

MorphoSys

The MorphoSys is a 8×8 array of processing units called Reconfigurable Cell (RC). The array is partitioned into four quadrants where the RCs within a quadrant are more densely connected than the quadrants are to each other. It could operate on up to 16-bit of data and supports dynamic reconfiguration where the configuration-memory can store up to 32 different configurations. The MorphoSys CGRA was tightly integrated into the processor core and the main memory. Special instructions were added into the processor core to carry out MorphoSys related operations e.g., control operations, configuration and data transfer between the array and the main memory. MorphoSys could show a performance of 25.6 Giga Operations Per Second (GOPS) while executing Discrete Cosine Transform (DCT) and Inverse-DCT.

PACT-XPP

PACT-XPP is a reconfigurable data processing engine consisting of adaptive computing elements that communicate over a network. It has run-time partial reconfiguration capability which means that a part of computing elements can be reconfigured for a new functionality while the others can keep computing data without any interruption. PACT-XPP supports different kinds parallelism, for example, pipelining, instruction level, data flow, task level and therefore it is well suited for streaming applications. XPP is a self-reconfigurable data processing engine that includes externally-triggered event-based reconfiguration. In this context, the trigger depends on special signals originating from within the array. PACT-XPP peak performance was estimated to be 57.6 GOPS at 150.0 MHz.

BUTTER

BUTTER acts as a coprocessor for COFFEE RISC core [39] and processes many computationally-intensive kernels. It is a CGRA of Processing Elements (PEs) where each PE functions as an ALU. All the PEs could communicate with each other using point-to-point connections and exchange data with each other. The configuration data is loaded from the main memory of the system into the CGRA using a DMA device. It is also used to load the data to be processed into the local memories of the CGRA. The DMA device allows tight integration between the CGRA and the data memory of the system. BUTTER was instantiated for a 8×4 matrix of PEs for the mapping of a 2D low-pass image filter, a noise reduction filter and a de-blocking filter used in H.264 decoder. The instance required 34,277 ALUTs while achieving an operating frequency of 300.0 MHz when the synthesis was optimized for resource utilization. These figures are slightly different if synthesis is optimized for speed.

2.2.4 Compiler for Coarse-Grain Reconfigurable Array

As a CGRA is an array of ALUs that exchange data with each other while performing a specific arithmetic or logic operation, any compiler tasked to map a certain algorithm will perform at least three steps; *Placement, Routing and Scheduling*. The profiling of the software is performed in advance to identify the candidate loops to be mapped on the CGRA. The architecture of the CGRA can be described, for example, in XML-based language which is considered in conjunction with the data flow graph of the algorithm. The compiler can finally perform placement and routing using a modulo-scheduling algorithm. The algorithm performs placement and routing while considering the available parallelism in terms of computing/routing resources and the throughput constraints of the algorithm. A well known example of a CGRA (ADRES) compiler is DRESC [66].

2.3 Multicore Platforms

Multicore platforms are composed of either homogeneous or heterogeneous cores. The homogeneous platform cores are generally RISC processors loosely coupled with each other while in heterogeneous platforms it is not necessarily the case. The heterogeneous platforms exist both in loose and tight coupling of constituent cores. The homogeneous platforms are generally programmable in C where the algorithm code is distributed almost equally on all the processor cores while heterogeneous platforms may require additional customized tool support to program the cores at the data flow level. Some of the multicore platforms are described as follows.

2.3.1 MORPHEUS

Morpheus ([26], [27]) is a platform consisting of a fine, a middle and a coarse-grain reconfigurable accelerator. The overall system is dynamically

reconfigurable and has a regular infrastructure (communication and memories) to enable regularity in control among the heterogeneous accelerators. The platform is accompanied by software including operating system and design tools for efficient use. The fine-grain, middle and coarse-grain devices are called FlexEOS, DREAM and XPP-III. All these devices communicate with each other over an NoC. DREAM is a reconfigurable DSP core composed of a 32-bit RISC processor and PiCoGA reconfigurable fabric. FlexEOS is an SRAM-based scalable FPGA fabric built on high-density multi-function logic cells. It can be programmed using VHDL and Verilog. XPP-III is a CGRA which is integrated into the datapath of a VLIW processor.

2.3.2 P2012

P2012 is a platform consisting of four clusters communicating with each other using a NoC ([28], [29]) where each cluster consists of 16 general-purpose processors. The processors are locally synchronous and globally asynchronous. P2012 is tested for image processing related algorithms. This platform can be considered in the category of homogeneous Multi-Processor System-on-Chip (MPSoC) platforms.

2.3.3 NineSilica

NineSilica ([30], [31]) platform is developed at Tampere University of Technology, Finland. It is a network of nine nodes where each node contains a 32-bit COFFEE RISC processor. The central node acts as a master node while the others act as slave nodes. Each core has its own instruction and data memory. All the nodes can exchange data with each other over the network using packet switching technique. NineSilica is another example of a homogeneous MPSoC and is completely programmable in C language. A larger version, Quad-Ninesilica also exists with 36 computing nodes.

2.3.4 RAW

Reconfigurable Architecture Workstation (RAW) consist of 16 pieces of 32-bit MIPS2000 processors arranged as an array of order 4×4 ([32], [37]). The processors communicate with each other using a NoC. RAW allows both static and dynamic scheduling. Its behavior is similar to a reconfigurable fabric under static scheduling while the dynamic scheduling mechanism between the cores classify it as a multicore platform. The programmable NoC in RAW targets mainly the wire-delay problem.

2.3.5 CRISP

The Cutting edge Reconfigurable ICs for Stream Processing (CRISP) was a European Union funded project that presents a General Stream Processor (GSP) [33]. The GSP is a coarse-grained core-level highly scalable platform for a wide range of streaming DSP applications. Its architecture was designed on the principle of locality-of-reference, so there are different levels of storage in this heterogeneous multicore architecture. Its instance, the GSP demodulator is built as an integrated platform composed of a General Purpose processor Device (GPD) and Reconfigurable Fabric Devices (RFD). The GSP instance contains one GPD and five RFDs. The overall system contains a ARM core and forty-five Xentium DSP cores. All the cores are connected via an NoC. The Xentium tile processor can operate at least at 200.0 MHz (90 nm CMOS technology, worst-case conditions). The GPD chip is manufactured in UMC 130 nm CMOS technology and also has an operating frequency of 200.0 MHz.

2.3.6 Intel's Single-Chip Cloud Computer

Intel designed a 48 X86-based multicore single-chip system to provide opportunities to system designers to explore its design space using their own custom applications and evaluate future needs and design challenges [34]. Intel named it as Single-chip Cloud Computer (SCC). The SCC also provides a

mesh network to establish communication between cores, four memory controllers and power management functions. It is fabricated on a 45 nm technology and it is a non-cache-coherent homogeneous platform. All the processors can independently operate between 100.0 to 800.0 MHz. The network operates using a separate clock source running at a frequency of 800.0 MHz or 1.6 GHz. The SCC has been benchmarked several times by different research groups across the world. For example, [35] mapped cosmological N-body simulator Gadget2, the NAS Parallel Benchmarks bt and lu as well as a self-written communication kernels while exploring the effects of power management functions.

2.3.7 TILE64^{TM}

TILE64^{TM} is a 8×8 array of processors connected through a 2D mesh network targeting a wide area of embedded applications [36]. It is a homogeneous MPSoC where each core is capable of running SMP Linux. Each processor is a 64-bit instruction word VLIW machine. The integer datapaths are 32-bit wide and also supports subword computations. The TILE64 processor runs at 750.0 MHz and achieves a maximum performance of 384 GOPS. The communication network provides 120 GB/s bandwidth. The system has L1, L2 and L3 caches and there is also an autonomous DMA engine in every tile of the processor.

2.4 Comparative Differences

The main difference between the different devices mentioned in the above subsections is the *Level of Granularity*. Fine-grain devices like FPGAs, GARP, FlexEOS, Molen have the lowest level of granularity, which means the processing bit-width is less than or equal to four. These devices are the most optimal in resource utilization as the granularity level is very fine, but the placement and routing would be most difficult. In this case, the compilers for these devices are more difficult to design and implement. The mid-grain

devices like PiCoGA and DART are a compromise between fine and coarse-grain devices. The compilers for placement and routing mechanism are relatively simpler than the fine-grain ones but for higher bit-width processing, the compilation time is relatively longer. Coarse-grain devices i.e., ADRES, MorphoSys, PACT-XPP and BUTTER have the highest level of granularity. They mostly deal at word-level processing and have a wide range of applications. Their compilers are the simplest.

All of these architectures are mentioned as they describe the evolution of reconfigurable devices starting from fine-grain to coarse-grain. Their brief description with examples is important for the general understanding of the advanced reconfigurable architectures designed and implemented in this research work.

The architectures like MORPHEUS, P2012, NineSilica and RAW are described to analyze the general concept used to design homogeneous and heterogeneous multicore architectures. The heterogeneous multicore architectures like MORPHEUS are generally reconfigurable. P2012 and NineSilica are made up of general purpose processors and demand the programmer to efficiently distribute the algorithmic computational load over the processors and also manage the data transfer in the most efficient way.

2.5 The Design Approach

The reconfigurable fabrics designed and implemented in this research are CGRAs and a heterogeneous CGRA-based platform programmable in C. It is easier to place and route algorithms on CGRAs as they support processing on the word level which also leads to faster prototyping of algorithms. Differently scaled CGRAs were designed for comparative performance evaluation. The scaled versions of CGRAs were integrated all together over a Network-on-Chip to maximize the number of reconfigurable processing resources on a platform. The above mentioned CGRAs and multicore platforms are of fixed sizes, however, the research work evolves from a relatively smaller template-

based CGRA to large-scale CGRAs and then constituting a multicore platform by integrating them all together over an NoC. The integrated multicore platform offers the highest number of processing resources in comparison to the other multicore platforms. Furthermore, all the CGRAs and the multicore platform designed are highly scalable and offer a high parallelism therefore favoring custom computation of highly parallel signal processing algorithms.

3. COARSE GRAIN RECONFIGURABLE ARRAYS AS ACCELERATORS TO PROCESSORS

The demand for the support and simultaneous execution of multiple applications on mobile devices is increasing every day, posing a greater challenge to energy sources. Especially with the introduction of 3G in mobile devices, there is a growing desire by the end-user to run multiple applications without having to care about the energy consumption. There are expectations for real-time performance governed by strict constraints imposed by many radio standards, for example IEEE.802.11 a/g/n [2].

In the recent past, hardware was fixed and the emphasis was more on the scalability of the software. For example, in Fast Fourier Transform (FFT) processing, it can be desirable to process 64, 128, 256 or 512 points of data at different times. The signal flow graphs of these structures are different from each other but if one radix-2 butterfly is employed, we can scale the processing of this set of data. As described in [3] and [42], 64 and 1024 points have been processed on the same radix-4 FFT accelerator by scaling only the software kernel. In Multiple-Input-Multiple-Output (MIMO) Orthogonal-Frequency-Division-Multiplexing (OFDM) demodulators, the throughput requirements may change as the environment changes and the processing capabilities have to adapt to the changes. One of the examples in MIMO-OFDM baseband domain is FFT processing where the software is scaled for optimal power dissipation [4].

Computationally intensive kernels are often assigned to Multi-Processor System on Chip (MPSoC) or accelerators working in a processor/coprocessor model. As discussed in previous chapter, a powerful class of accelerators is

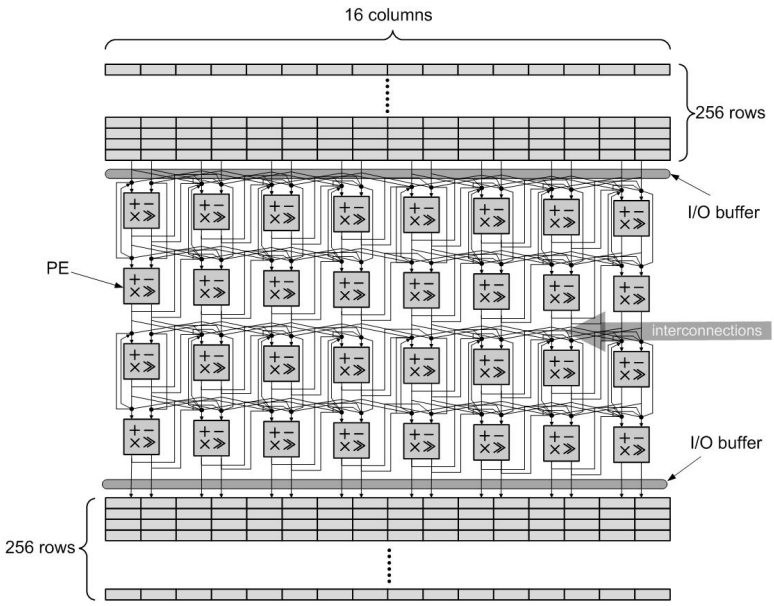


Fig. 1. *CREMA, a 4×8 PE template-based CGRA.*
© 2009 IEEE

Coarse-Grain Reconfigurable Array (CGRA) which is ideal for signal processing applications because it provides high parallelism and throughput. General-purpose CGRAs occupy an area of a few million gates and their presence in the system cannot be justified unless they are heavily utilized most of the time. In this chapter, different types of CGRAs are described which are developed as part of this research work, i.e., CREMA, AVATAR and SCREMA plus a reconfigurable Application Specific Instruction-set Processor (rASIP) .

3.1 CREMA

CREMA is a 4×8 PE CGRA template which is equipped with two 32-bit local memories of 16×128 size. The local memories are provided to avoid memory bottlenecks, therefore increasing the throughput of the CGRA. Each PE has two inputs, two outputs and is capable to perform 32-bit integer and

[1] The designs and implementations presented in this chapter have been published in articles [38], [3], [45] and [54].

floating-point (IEEE-754) based operations. All PEs can receive data from the neighboring PEs in point-to-point fashion. The interconnections among PEs are of local or global nature. Local interconnections can be established with the adjacent PEs. One type of global interconnections are with the right-most PEs in any row of the generated accelerator. The other type of global interconnection is used to connect output(s) of the I/O-buffer to any of the processing element's input. The type of interconnections to be used to map an application is dependent on the user. While specifying interconnections, the user also specifies which type of operation(s) each PE should perform at any clock cycle. The pattern of interconnections and the operation to be performed by each PE at any clock cycle is called a *context*. A context can be switched at run-time to enable different specific tasks. A new configuration stream defining additional contexts can also be loaded at run time using the Direct Memory Access (DMA) device [40]. In between the local memories and the processing array, there are I/O-buffers that are used to provide interleaving to the data stored in the local memories before they are supplied to the processing array. Each I/O-buffer consists of 16 of 16×1 multiplexers and 16 of 32-bit registers. The output of each multiplexer is delayed one cycle by the register. The user can load the data in any of the local memories of the generated accelerator using the DMA device, enable a context and process the data over the accelerator array. As required, the user can enable another context and keep processing by switching to different contexts until the final results are obtained. The program flow controlling the execution is written in C and is compiled by the gcc compiler tailored for a COFFEE RISC processor. Based on the program flow, specific tasks can be assigned to the CGRA using special function calls written for CGRAs. COFFEE is responsible for the cycle-accurate processing of CREMA-generated accelerator by writing control words in the control registers of the accelerator. Once the user finishes designing contexts using the graphical tool which is a custom tool made for CREMA to design contexts, the graphical tool in return generates the .h configuration files which contain the configuration words. At the system start-up time, these configuration words are fetched from the main memory of the system by the DMA device then distributed over the PE array

using a pipelined infrastructure [43]. These configuration words select the operation to be performed by each PE and also the interconnections among the PEs. CREMA-generated accelerator can have eight configurations words at most in each PE. If the application demands, more configuration words can be fetched from the main memory using the DMA device. All these capabilities make CREMA generated accelerator array *dynamically reconfigurable*. CREMA is shown in Fig. 1, equipped with two local memories each consisting of 256 rows.

3.2 AVATAR

A scaled-up version of CREMA was designed and published as AVATAR [3] as shown in Fig. 2. Scaling up CREMA to AVATAR required hardware alterations including additional configuration memories, multiplexers, encoders and registers. AVATAR was a 4×16 PE CGRA and had more computational power than CREMA because of its larger size. Both CREMA and AVATAR are capable of processing different computationally intensive kernels, for example, MVM [38], correlation [41] and Fast Fourier Transform (FFT). FFT execution time constraints for IEEE-802.11a/g and 3GPP-LTE standard were satisfied by CREMA-generated FFT accelerators presented in [42] and [44]. To satisfy FFT execution time constraints for MIMO-OFDM IEEE-802.11n [2] standard, a larger accelerator array was required. CREMA-generated FFT accelerator was composed of three different contexts to map a radix-4 FFT butterfly that had to be enabled in a sequence to process a single stage of FFT algorithm [44]. However, AVATAR-generated radix-4 FFT accelerator was large enough to do that in a single context. AVATAR generated radix-(2, 4) FFT accelerator can process 64- and 128-point FFT algorithms while satisfying the execution time constraints of IEEE-802.11n standard.

In CREMA, as there are 16 inputs in the first row of PEs, so there are 16 of 16×1 multiplexers in each I/O-buffer. In this way, each and every input of all PEs have their own multiplexer to access each and every memory bank in the local memory, therefore providing maximum bandwidth. In case of

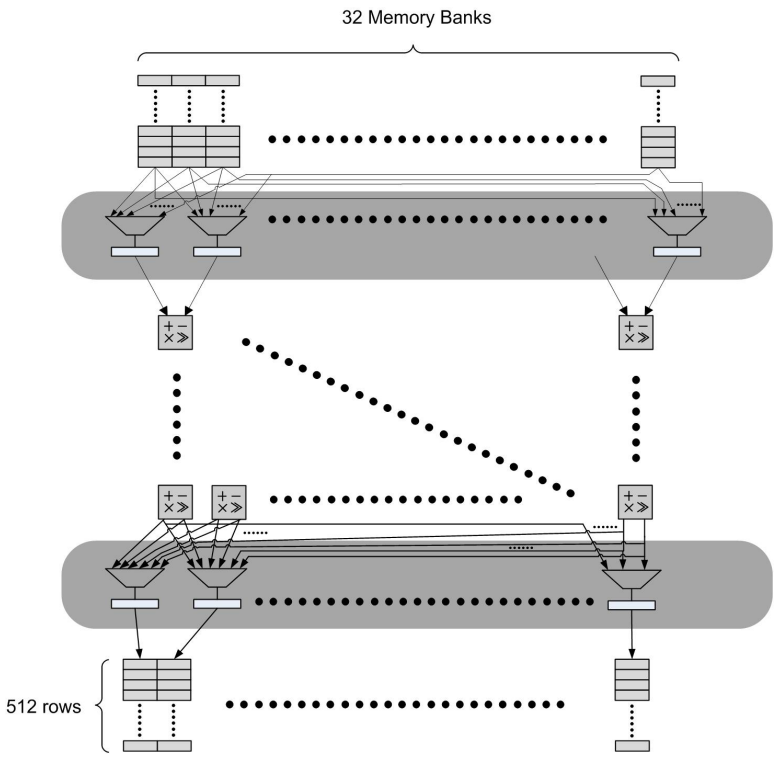


Fig. 2. *AVATAR, a 4×16 PE template-based CGRA*

AVATAR which is a *non-scalable* 4×16 PE CGRA, there are 32 memory banks in each local memory, so there are 32 of the 32×1 multiplexers in each I/O-buffer. If we consider the next version of AVATAR, it will be a 4×32 PE CGRA and there will be 64 of the 64×1 multiplexers in each I/O-buffer. This continuous scaling-up will explode in resource utilization compared to the speed-up it offers. We developed SCREMA to avoid this explosion and removed the I/O-buffers to provide seamless scalability in the architecture and reduced resource consumption for FPGA synthesis.

3.3 Scalable-CREMA

SCREMA is a CGRA template scalable both in rows and columns of PEs but the number of columns can only be scaled to 4, 8, 16 and 32. After 32, the next

size of the scaled version of the CGRA in terms of number of PE columns is 64, such a large CGRA will stay underutilized for the type of algorithms under consideration for mapping. For example, the maximum order of matrix-vector multiplication considered for mapping is 32. It is for this major reason, the scaling limit for PE columns was not increased further. SCREMA is based on the architecture of CREMA so they have many structural properties in common. For example, functionality of the PE is the same and also the way algorithms are mapped. However, SCREMA has an advantage as it can flex between CGRA templates of different sizes by changing a few parameters in the definition file of its VHDL model. It means that scaling is performed statically at the compilation time, however, the accelerator generated from the scaled SCREMA template is dynamically reconfigurable at run-time. The digital hardware due to its binary characteristics scales in regular fashion by the factor of base 2, so for example, if at present the number of PE columns is equal to eight then the next scaled-up version will have the number of PE columns equal to sixteen. As the number of PE columns increases, the size of memory banks also increases correspondingly. Fig. 3 shows the general flow for generating the accelerator from a scaled CGRA template. The number of rows and columns of PEs for the CGRA template are decided by the user in the VHDL definition package file. The configuration file generated by the GUI tool and the VHDL definition package file are compiled with the VHDL model of SCREMA, which can generate accelerators of four different sizes depending on the user input. The accelerators can work as coprocessor to COFFEE. The configuration bit stream (*.h file) generated by the GUI tool is loaded at run time by COFFEE with the help of a DMA device depending on the program flow written manually by the user.

Fig. 4 shows 4×16 PE SCREMA based processing model, where COFFEE does the general-purpose processing and the computationally intensive tasks are executed by a SCREMA generated accelerator. 4×16 SCREMA is in the figure just as an example. The figure remains the same except that SCREMA can flex between 4×4, 4×8, 4×16 and 4×32 PEs. It can also be noticed, SCREMA is equipped with two local memories, their sizes scale as the PE

array scales. Main memory of the system, COFFEE, I/O Peripherals, Control Unit, DMA/SCREMA exchange data with each other using a network of switched interconnections. This network provides dedicated connections between different modules for faster exchange of data.

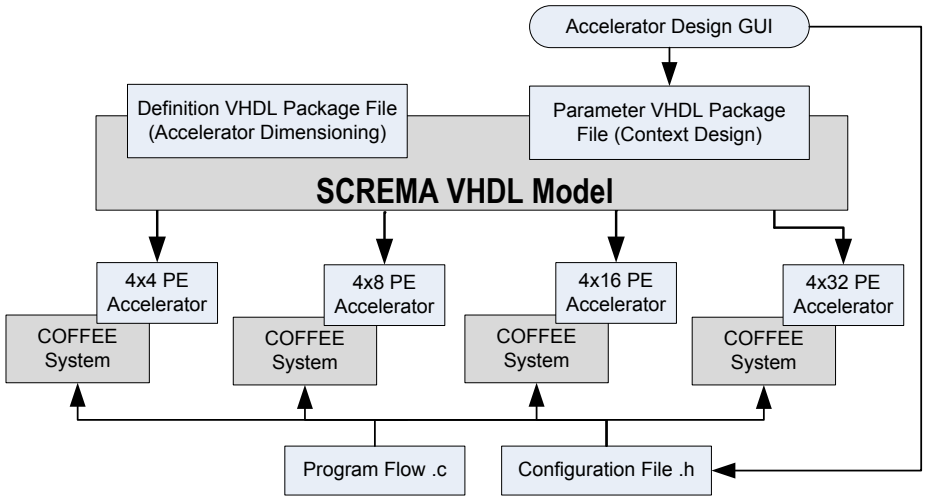


Fig. 3. *SCREMA based Accelerator Generation Flow.*

3.4 Processing Elements

As described earlier, the unit of processing for CREMA, AVATAR and SCREMA is called a PE which can perform both integer and floating-point operations. Each PE has two input and one immediate register where the operand to be processed can be stored. Fig. 5 shows the LUT, adder, multiplier, shifter and floating-point logic as computational resources inside the PE. The configuration words decide the output of the decoder inside each PE which in return enables the operator among the PE computing resources. The results from the computational resources is selected by a multiplexer and is supplied to the output. The PE appears to be a template to the user at system design time. Based on the requirements of the application, the user can se-

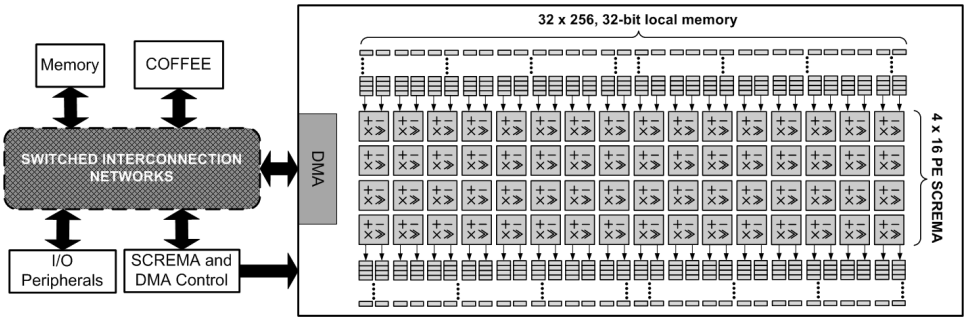


Fig. 4. SCREMA based Embedded Processing Model.

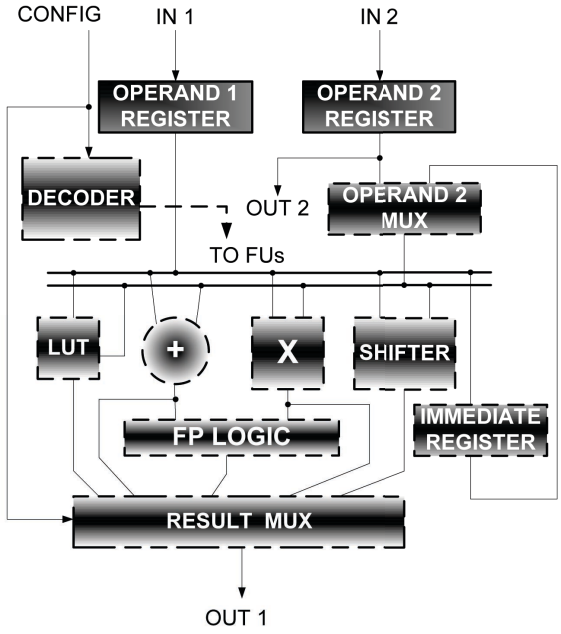


Fig. 5. Processing Element used in CREMA, AVATAR and SCREMA.

lect which resources are required to be instantiated while designing contexts. The size of the decoder and multiplexer inside each PE increases as the number of contexts designed by the user increases and vice-versa. In Fig. 5, all the modules with dashed border are user-specific for instantiation thus saving resource/area utilization.

3.5 Design of Contexts and Application Mapping

The contexts are designed while considering the computational requirements of the algorithm and the constraint in terms of the number of rows and columns of PEs in the CGRA. The design of contexts and application mapping can be described while explaining how an integer Matrix-Vector Multiplication (MVM) application can be mapped onto SCREMA. The application mapping process is the same for CREMA and AVATAR except that these CGRAs are not scalable column-wise.

Considering, if a matrix $a = [a_{i,j}]$ is an N^{th}-order matrix which is supposed to be multiplied by vector $\overrightarrow{b} = [b_i]$ to produce a product vector $\overrightarrow{p} = [p_i]$ then the multiplication process can be defined as

$$[p_i] = \sum_{j=1}^{N} [a_{i,j}] \times [b_j] \tag{1}$$

where i, j = 1, 2, 3,...,N.

Consider a $R \times C$ PE SCREMA; each local memory can be denoted as M consisting of $2 \times C$ memory banks that can store up to 2^m words, where R, C are the number of rows, columns of SCREMA respectively and m can be any positive integer. We can express a location in the data memory of SCREMA as $M[r][c]$ where c represents the memory bank number and r the location number in that memory bank. As sets, r and c can be written as $r = \{r_1, r_2, r_3, ..., r_{2^m}\}$ and $c = \{c_1, c_2, c_3, ..., c_{2C}\}$.

Suppose $N = 4$ in Eq. 1, then the vector $\overrightarrow{b} = [b_1, b_2, b_3, b_4]$ will be multiplied with matrix A to produce the product vector $\overrightarrow{p} = [p_1, p_2, p_3, p_4]$, where the matrix A can be written as

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \tag{2}$$

The next step is to load the matrix A and vector $\overrightarrow{b}$ into the data memory of SCREMA. The pattern of loading the data will be as follows.

for $i = 1 : i \leq 4 : i++$ **do**
  for $j = 1 : j \leq 4 : j++$ **do**
    $M[2 \times i - 1][j] \leftarrow [a_{i,j}]$
    $M[2 \times i][j] \leftarrow [b_j]$
  end for
end for

The MVM accelerator generated by a 4×4 PE SCREMA needs four multiplication operators in the first row to process 4^{th} order MVM in the first context. After the multiplication process, there has to be a shift operation which is carried out in the second row of the first context and finally the addition operations in the third and fourth row. As all the PEs have operand registers, therefore the overall processing is performed in a pipelined fashion. The first context for the 4×4 PE MVM accelerator is shown in Fig. 6. It needs eight SCREMA clock cycles to process the data from the first four rows of the local memory and stores the resultant vector in the second local memory. Out of eight clock cycles, four are used in processing the data and the other four are the latency of the context.

The matrix and the vector data words are read by the DMA from the main memory of the system and written on the local memories of MVM accelerator. The DMA starts from location number $M[1][1]$ and $M[1][2]$ to write the matrix words and the vector words respectively. When the matrix word is written to the memory location $M[r][2C-1]$ then the next matrix word is written at $M[r+1][1]$. Similarly, when the DMA writes the vector word to the location $M[r][2C]$ then the next vector word is written to $M[r+1][2]$. After every matrix word is written, there will be the corresponding vector word written next to it as they are supposed to be multiplied. To generalize the loading pattern of a matrix and a vector of any order $N = 2^n$ in one of the local memories of SCREMA generated accelerator of C number of PE columns where $n \in Z^+$ and $n \geq 1$, the following algorithm can be used to align the data for the context shown in Fig. 6 or a group of those contexts working in parallel.

Algorithm 1 Data Placement Algorithm in the Local Memory of SCREMA

$c \leftarrow 0$
$r \leftarrow 0$
for $i = 1 : i \leq N : i++$ **do**
  $r \leftarrow r + 1$
  $c \leftarrow 1$
  for $j = 1 : j \leq N : j++$ **do**
    $M[r][c] \leftarrow [a_{i,j}]$
    $c \leftarrow (c+1) \mod 2C$
    $M[r][c] \leftarrow [b_j]$
  end for
end for

It is considered that the context shown in Fig. 6 is the most optimal implementation as only five of the sixteen PEs are not used. The problem with MVM is about the data processing path through the PEs that finally narrows down to only a single PE which is supposed to produce the product vector. If the number of rows of SCREMA are increased, there will be many of the PE which will not be used at all and occupy unnecessary resources. In this particular case, the number of contexts shown in Fig. 6 can be increased to work in parallel for faster processing. To explain the implementation details further, the mapping of MVM for $N = 8$ on 4×8 PE SCREMA is discussed as follows. Using the data placement algorithm above, the data is loaded in one of the local memories of SCREMA in alignment as shown in Fig. 9. The overall computation requires two stages of processing. The first stage produces the partial products that requires eight SCREMA clock cycles plus four cycles of latency as there are four rows of SCREMA. In the next stage, the partial products will be added to produce the final product vector. To do so, another context is employed as shown in Fig. 10. For implementation of N^{th}-order MVM for different C, other than the context required for summing up the partial products, there are additional contexts that are required at different times for reordering of the partial products. Reordering is required to add the related partial products. This kind of reordering is always required in cases when $N > 2C$, meaning if number of columns (memory banks) of the local memory are less than the order of the matrix to be processed.

The mapping of $N = 4, 8, 16$ and 32 MVM on each of the 4×4, 4×8, 4×16 and 4×32 PE accelerators is similar to the examples of mapping $N = 4$ and 8 MVM on 4×4 and 4×8 PE SCREMAs.

In case of MVM, there is a preprocessing step required between two processing steps if the number of memory banks in a local memory is less than the order of MVM. The processing step is the context shown in Fig. 6 and multiples of it work in parallel as C increases by an order of 2^m. The preprocessing contexts are 'L' shaped as shown in Fig. 7. As every PE is registered, each preprocessing or processing context adds to latency. The processing context has a constant latency of four clock cycles. The preprocessing context is

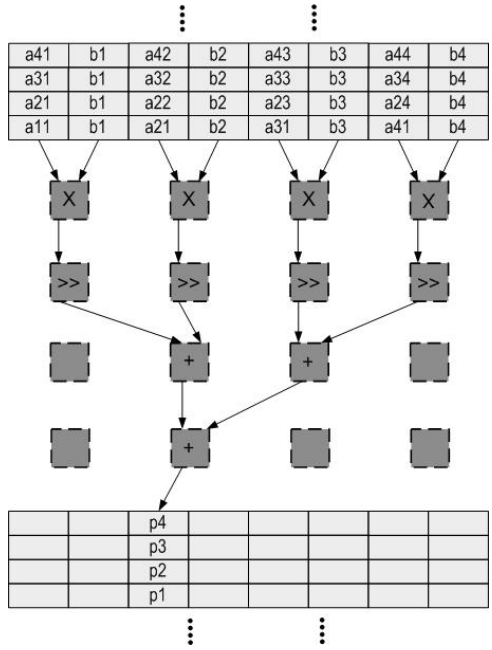


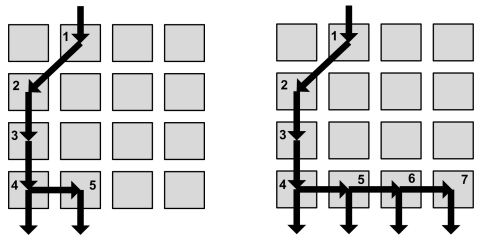
Fig. 6. *4×4 PE SCREMA for $N = 4$ MVM.*

designed to have latency of five or seven clock cycles. An 'L' shaped context with a latency of five clock cycles shown in Fig. 7(a) which is used in the second step of preprocessing of $N = 8$ on 4×4 PE SCREMA, also mentioned in Table 1. In the contents of the table, Pro, ctx.swh and Pre.Pro stand for Processing, context switch and PreProcessing. Table 1 shows execution (Exe) of different orders of MVM on different scaled versions of SCREMA. In the table, N shows the order of MVM, S/No shows the order in which a processing or preprocessing step is performed. The latency mentioned in the table depends on the context designed for processing or preprocessing while a fixed addition of +2 is due to two local memories causing one cycle delay each. CC is the number of clock cycles required to perform a processing or a preprocessing step. The CC. Total is the overall number of clock cycles required to process an MVM of order N.

Let's suppose that $pp_1, pp_2, pp_3, ..., pp_{16}$ were the partial products that were generated by context shown in Fig. 6. From the context, it is apparent that these partial products will be stored in a single memory bank. In Fig. 8(b),

it is shown that for $C = 4$ and $N = 8$, the first column of the matrix and the vector to be processed will fill the first two rows of the local data memory of SCREMA. This means that the partial products produced from the data in the first two rows have to be added but as the result of the execution of the context shown in Fig. 6, the partial products are stacked on each other and hence cannot be added together without a preprocessing step. A preprocessing context shown in Fig. 7(a) will be enabled and partial products will get aligned in two different memory banks and then they will be added by a third context to produce the final product vector. Similarly, it is shown in Table 1 that the execution of $N = 16$ MVM kernel on a 4×4 PE MVM accelerator requires a preprocessing latency of seven clock cycles in an 'L' shaped context shown in Fig. 7(b). At first, the data words belonging to the first column of the matrix and vector are loaded using Algorithm 1 in the local data memory of MVM accelerator. This loading will fill the first four rows of one of the local data memories. Then the execution starts by enabling the context shown in Fig. 6. As a result of execution of this context, all the partial products $pp_1, pp_2, pp_3, ..., pp_{16}$ stack on each other. The partial products consecutive to each other can only be added in a group of four at maximum. While executing the preprocessing context shown in Fig. 7(b), pp_1 will be in PE numbered as seven after a latency of seven clock cycles. The data will be written for one clock cycle and will be stalled for the next three so that in the next write operation, pp_5, pp_6, pp_7 and pp_8 could be written in the same row and then be added all together by the third context. Once the overall processing completes, the results of MVM can be transferred back to the main memory using special DMA operations.

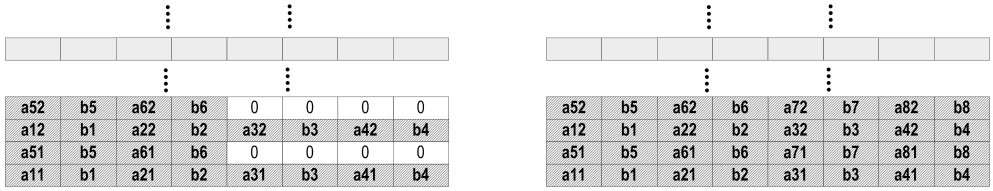
It is to be noticed that in Fig. 8(a) and 8(b), the vector $\overrightarrow{b}$ is loaded multiple times along with every column of the matrix. In the design of CGRAs, the local memories are made only line readable in an effort to make the address decoding logic simpler and faster. It is only when the data is read to be processed over CGRA. Therefore, for correct execution of the MVM algorithm, the vector $\overrightarrow{b}$ is loaded along with every column of the matrix.



(a) Latency of five clock cycles

(b) Latency of seven clock cycles

Fig. 7. *Preprocessing Contexts used for Data Reordering.*



a52	b5	a62	b6	0	0	0	0
a12	b1	a22	b2	a32	b3	a42	b4
a51	b5	a61	b6	0	0	0	0
a11	b1	a21	b2	a31	b3	a41	b4

(a) $N = 6$

a52	b5	a62	b6	a72	b7	a82	b8
a12	b1	a22	b2	a32	b3	a42	b4
a51	b5	a61	b6	a71	b7	a81	b8
a11	b1	a21	b2	a31	b3	a41	b4

(b) $N = 8$

Fig. 8. *Data Organization in Local Memory of* 4×4 *PE SCREMA for the Processing of MVM.*

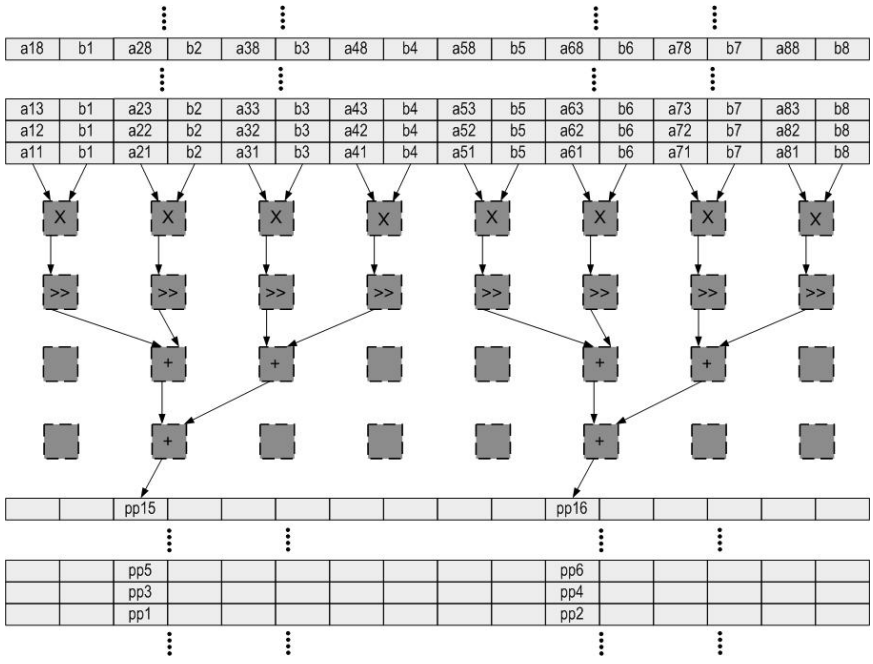


Fig. 9. 4×8 *PE SCREMA for* $N = 8$ *MVM.*

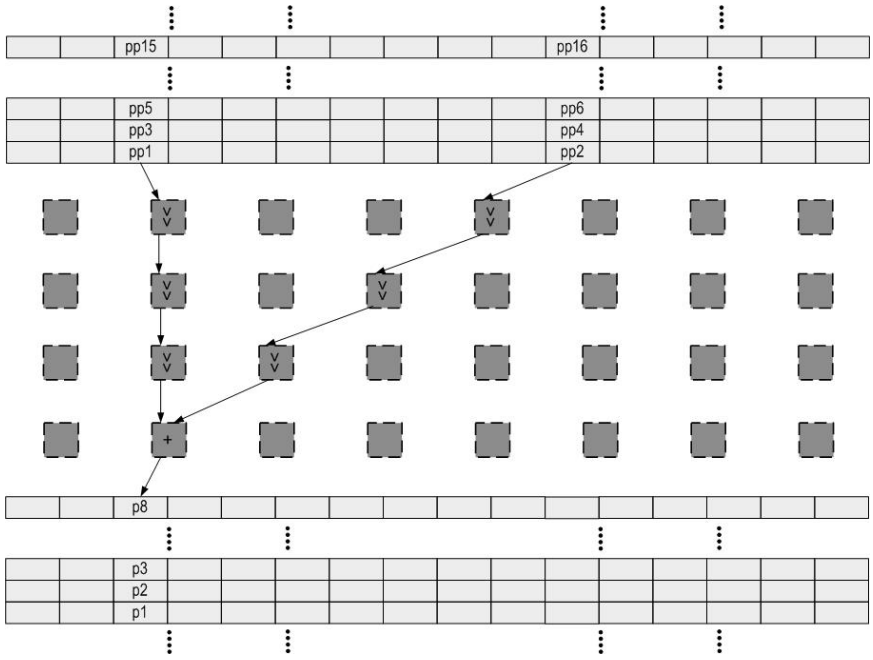


Fig. 10. 4×8 *PE SCREMA, Second Context for* $N = 8$ *MVM Second Stage.*

Table 1: Cycle-by-Cycle Measurement of MVM Kernels on SCREMAs.

SCREMA	N	S/No	Exe. Type	Latency	CC	CC. Total
4×4	4	1	Pro	4+2	4	10
4×4	8	1	Pro	4+2	16	85
			ctx.swh		13	
		2	Pre.Pro	5+2	16	
			ctx.swh		13	
		3	Pro	4+2	8	
4×4	16	1	Pro	4+2	64	237
			ctx.swh		5	
		2	Pre.Pro	7+2	64	
			ctx.swh		51	
		3	Pro	4+2	32	
4×4	32	1	Pro	4+2	256	811
			ctx.swh		5	
		2	Pre.Pro	7+2	256	
			ctx.swh		42	
		3	Pro	4+2	64	
			ctx.swh		5	
		4	Pre.Pro	5+2	64	
			ctx.swh		53	
		5	Pro	4+2	32	
4×8	4	1	Pro	4+2	2	8
4×8	8	1	Pro	4+2	8	49
			ctx.swh		21	
		2	Pro	4+2	8	
4×8	16	1	Pro	4+2	32	193
			ctx.swh		5	
		2	Pro	4+2	32	
			ctx.swh		46	
		3	Pre.Pro	5+2	32	
			ctx.swh		5	

Continued on next page

Table 1 – *Continued from previous page*

SCREMA	N	S/No	Exe. Type	Latency	CC	CC. Total
		4	Pro	4+2	16	
4×8	32	1	Pro	4+2	128	688
			ctx.swh		5	
		2	Pro	4+2	128	
			ctx.swh		49	
		3	Pre.Pro	5+2	128	
			ctx.swh		5	
		4	Pro	4+2	64	
			ctx.swh		42	
		5	Pre.Pro	5+2	64	
			ctx.swh		5	
		6	Pro	4+2	32	
4×16	4	1	Pro	4+2	1	7
4×16	8	1	Pro	4+2	4	44
			ctx.swh		24	
		2	Pro	4+2	4	
4×16	16	1	Pro	4+2	16	90
			ctx.swh		12	
		2	Pro	4+2	16	
			ctx.swh		12	
		3	Pro	4+2	16	
4×16	32	1	Pro	4+2	64	419
			ctx.swh		5	
		2	Pro	4+2	64	
			ctx.swh		45	
		3	Pro	4+2	64	
			ctx.swh		5	
		4	Pre.Pro	5+2	64	
			ctx.swh		45	
		5	Pro	4+2	32	
4×32	4	1	Pro	4+2	1	7

Continued on next page

Table 1 – *Continued from previous page*

SCREMA	N	S/No	Exe. Type	Latency	CC	CC. Total
4×32	8	1	Pro	4+2	2	40
			ctx.swh		24	
		2	Pro	4+2	2	
4×32	16	1	Pro	4+2	8	81
			ctx.swh		19	
		2	Pro	4+2	8	
			ctx.swh		20	
		3	Pro	4+2	8	
4×32	32	1	Pro	4+2	32	295
			ctx.swh		5	
		2	Pro	4+2	32	
			ctx.swh		54	
		3	Pro	4+2	32	
			ctx.swh		5	
		4	Pro	5+2	32	
			ctx.swh		41	
		5	Pro	4+2	32	

3.6 Case Study: Reconfigurable Application-Specific Instruction-Set Processors

Application-Specific Instruction-Set Processors (ASIPs) provide an optimal solution to accelerate specific applications while carrying a general-purpose flavor. ASIP's customization to specific applications is achieved by adding dedicated hardware in the datapath of the processor and using it by introducing special instructions in the instruction-set of the processor. The dedicated hardware can have an additional feature of reconfigurability to accelerate multiple applications. In Chapter 2, many fine and coarse-grain reconfigurable

accelerators are discussed which are integrated into the datapath of a processor. In this case study, a CGRA template mentioned earlier in this chapter 'AVATAR' is tightly integrated with a RISC processor thus making an rASIP. A radix-(2, 4) FFT accelerator was generated from AVATAR-template that processes 64- and 128-point stream while considering execution-time constraint of IEEE-802.11n standard. AVATAR needs to performs following three essential basic steps for the successful execution of a kernel.

1. load the configuration stream
2. load the data to be processed
3. write the control registers to enable execution

During the system startup time, the configuration stream and the data to be processed can be loaded into AVATAR's respective memories from the main memory. It is achieved by adding special load-word instructions into the Instruction-Set Architecture (ISA) of the RISC processor. Additional instructions were added in the ISA of RISC processor to pass control words to the control registers of the CGRA's control unit for cycle accurate processing. Once the CGRA receives the control words, it performs multiple cycles to process a task and during this process, the RISC does not interrupt it. The CGRA has its own driving clock, therefore allows it to run faster than the RISC processor. Once the CGRA completes the processing, it returns the results back to the main memory using special store-word instruction added into the ISA of the RISC processor. The tight integration of AVATAR in the datapath of the RISC processor allows to reduce overhead caused by the DMA engine and other logic resources in comparison if AVATAR works in a standard processor/coprocessor model. Furthermore, while saving resource utilization in the same comparison, the power dissipation and energy consumption remained almost un-compromised. The integration of AVATAR into the datapath of RISC processor using an interface register is depicted in Fig. 11.

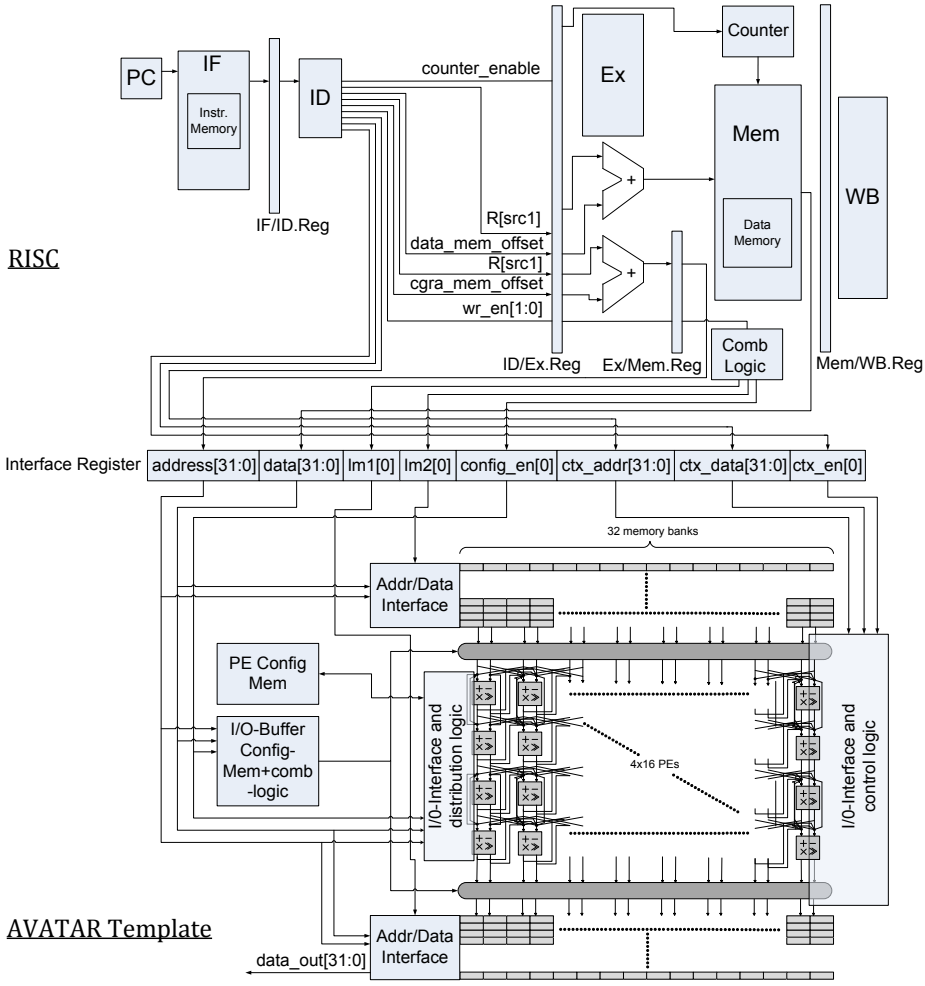


Fig. 11. *Integration of AVATAR into the datapath of a RISC processor.*
© 2013 IEEE [46]

3.7 Relation to Existing Work

In this chapter, CGRAs like CREMA, AVATAR, SCREMA and a reconfigurable ASIP are discussed which relate to the latest research in coarse-grain devices. The modern trend is focused more towards coarse-grain devices as they support word level processing while the placement and routing is relatively easier in comparison to finer grained devices. The CGRAs presented in

this work collectively show the increase in computational-resource parallelism by offering more number of PE resources. It can be observed, the number of PE columns in AVATAR increased to 16 in comparison to CREMA with 8 PE columns, therefore making it more computationally powerful. The SCREMA which is a scalable CGRA shows that the number of columns of PEs can be increased to 32 for a generated instance. The scalability trend in this particular fashion is not directly visible in the CGRAs mentioned in Section 2.2.3. All of this design effort is to maximize the number of reconfigurable processing resources on a platform. In the end, the design of an rASIP is presented which shows the advantages in terms of reduced logic utilization by integrating a CGRA (AVATAR) in the datapath of a RISC processor.

4. AN ACCELERATOR-RICH PLATFORM WITH CGRAS AS PROCESSING ENGINES

The accelerator-rich architectures are targeted to maximize the number of available processing resources on a platform. As an advantage, the platform becomes capable to accelerate massively parallel applications simultaneously. The accelerator cores can either be loosely or tightly coupled with each other. The loose coupling allows all cores to be able to address each other but the communication between them is relatively slow in comparison if they are tightly coupled as discussed in the previous chapters. Apart from CREMA, AVATAR and SCREMA, this research work also introduces Heterogeneous Accelerator-Rich Platform (HARP) as shown in Fig. 12. HARP is a template based architecture, composed of nine nodes where each node can contain a CGRA based accelerator while only the central node contains the RISC processor. All the cores (CGRAs + RISC) communicate with each other using an NoC. HARP is composed of different and multiple CGRAs of specific sizes, where the size of the CGRA is in terms of order (rows $\times$ columns) of PEs. The specific sizes of CGRAs are tailored for specific applications. These heterogeneous CGRAs are together running a proof-of-concept test-case to verify the functionality of the overall design. The overall supervision in terms of control and communication is provided by COFFEE RISC core [1].

[1] The design and implementation presented in this chapter has been published in article [47].

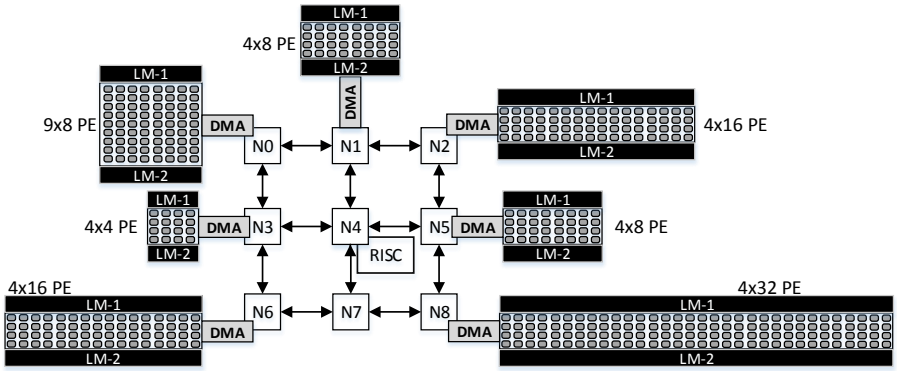


Fig. 12. *Heterogeneous Accelerator-Rich Platform.*

4.1 Motivation

The multicore scaling is threatened by recently known utilization wall. In a case-study, it is shown that only 7% of a 300mm^2 die can by used at maximum possible frequency under a power dissipation constraint of 80W [1]. This underutilized part of the chip is called *Dark Silicon*. The un-utilized part can be operated at very low frequency or has to be switched off to avoid thermal breakdown of the chip. It means the performance on a chip can not be increased by scaling up the number of cores on the chip. The un-utilized part of the chip can be replaced with special-purpose accelerators operating at lower frequencies. Another motivation to propose such a system is to increase the processing capability on a chip by maximizing the number of computational resources. In total, HARP provides 408 PE on chip as an example maxima.

4.2 Integration of CGRAs

As the backbone of communication between the cores in HARP is an NoC, therefore the cores are loosely coupled and they can address each other. The central node of the NoC contains the RISC processor which acts as master

while the other nodes act as slaves. The central node also contains a data and an instruction memory besides a RISC processor. The slave nodes contain a DMA device and also a data memory besides a CGRA device. The RISC processor can write data to the NoC in form of packets. The packets are then routed over the NoC and reaches their destination node. The data memory and the RISC processor are integrated with each other using request and response switches. The DMA device in the slave nodes has a slave and a master interface. The slave interface of the DMA receives data from the NoC and as a result either the DMA master is activated or a CGRA related operation is performed. If the DMA's master is activated, it can either access the node's internal data memory or write to the network. In the whole process, the request and response switches are activated by the arbiter as required. Fig. 13 shows a combined view of the master and the slave nodes of HARP.

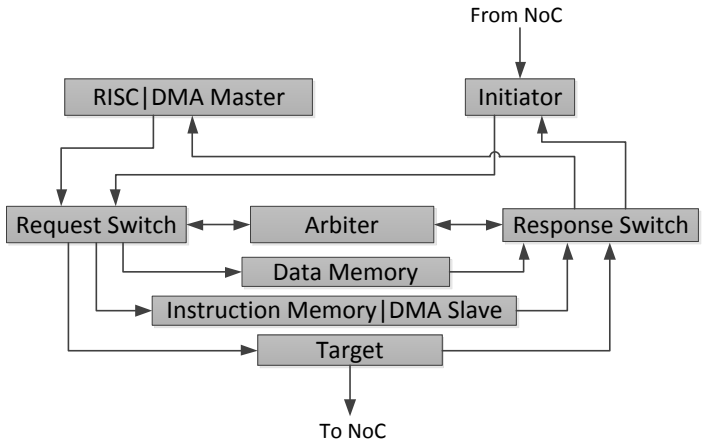


Fig. 13. *A detailed view of master and slave nodes of NoC.*

The type of CGRA used for each node, the order of the CGRAs in terms of rows×columns of PEs and the kernels mapped are demonstrated only for the proof-of-concept. The details are shown in Table 2.

Node	PEs	CGRA Type	Kernel
N0	8×9	CREMA	radix-4 64-point FFT
N1	4×8	CREMA	complex MVM 64^{th}-order
N2	4×16	AVATAR	radix-(2, 4) 128-point FFT
N3	4×4	SCREMA	real MVM 32^{nd}-order
N5	4×8		
N6	4×16		
N8	4×32		

Table 2. *Different CGRAs connected to HARP nodes*
The orders of the CGRA in terms of rows×columns of PEs and the kernels implemented, respectively. © 2014 IEEE [47]

4.3 Synchronization

The RISC core in the master node has a shared memory space which is used for synchronization. The data transfer between the memories within a slave node and in between the slave nodes need synchronization. It is also true for any exchange of data between the slave and master node. The data transfer within a node is carried out by the DMA device, such that a DMA transfer needs to be completed before the next one starts. The synchronization is established in two steps. At first the RISC sets its shared memory location corresponding to the destination node and as a second step transmits the control packet over the NoC. The NoC routes the control packet towards the destination node which is received at its DMA slave interface. As a result the DMA master starts the data transfer between the node's data memory and the local memory of the same node's CGRA. Once the transfer is complete, the DMA transports an acknowledgment over the network towards the master node destined to its corresponding shared memory location which was originally set by the RISC core to establish synchronization. The acknowledgment resets that memory location and after that, the RISC can order another DMA

transfer within the same slave node. The synchronization for data transfer between slave nodes and also between slave and master node is established using a similar protocol. The synchronization for data transfer does not interrupt the data transfer between other nodes as the synchronization is established using shared memory mechanism.

4.4 Application Mapping and Simulation

The application mapping is only for testing and to demonstrate the functional capability of HARP. It is performed in a way that the data sharing could be demonstrated between the master and slave nodes and also between the slave nodes. During the system startup time, the RISC processor loads the configuration stream and the data to be processed from its local memory to the local memories of the respective nodes using the NoC. The RISC then sends the control words to the slave node's DMAs for intra-node data transfer. Upon receiving the control words, the DMAs fetch data from the slave node's local memory and writes it to one of the local memories of the CGRA. After this, the computation process starts. At first, the RISC sends control and processing information to node-0 which performs a 64-point FFT using a radix-4 butterfly implemented on a 9×8 PE CGRA. Once the FFT is computed, the DMA of node-0 sends the result to the data memory of the node-1. The RISC then activates the DMA of node-1 for inter-node data transfer between the data memory and the local memory of the CGRA of node-1. In the second phase, the RISC activates the CGRA of node-1 where a 64^{th}-order complex MVM is performed. After the completion of the task, the results of MVM are transported back to the data memory of the RISC processor using the DMA of node-1. This whole process shows the supervisory control of the RISC processor over the other nodes. Node-2 is processing 128-point FFT using a radix-(2, 4) FFT accelerator generated by a 4×16 PE CGRA. It is processed independently and irrespective of the processing by other nodes. The node-3, node-5, node-6 and node-8 are processing 32^{nd}-order integer MVM using accelerators that were generated using 4×4, 4×8, 4×16 and 4×32 PE SCREMA CGRAs. All of

these nodes are processing simultaneously and independently of each other. The overall design in HARP is operated by a single clock source at 100.0 MHz. The clock cycles required for the data transfer from the data memory of one node to the data memory of another node (D. Mem to D. Mem), data transfer from the data memory of a slave node to the local memory of the CGRA (D. Mem to CGRA) and the clock cycles required for kernel execution (Exe. Total) are shown in Table 3. Clock cycles with * sign represent data transfer from CGRA's local memory to a Node's data memory.

Node-to -Node	D. Mem to D. Mem	D. Mem to CGRA	Trans. Total	Exe. Total
N4-N0	1017	659	1676	420
N0-N1	1036	446*	1482	457
N1-N4	-	448*	-	-
N4-N2	2042	1033	3075	571
N4-N3	75622	-	75622	728
N4-N5				609
N4-N6				340
N4-N8				211

Table 3. *Clock cycles required for different stages of data transfer and processing in HARP.*

5. MEASUREMENTS, ESTIMATIONS AND OPTIMIZATIONS

Almost all of the architectures presented in this research work are synthesized for Stratix FPGAs. One of the reasons using FPGA is their lower costs in prototyping and testing. The proposed architectures were evaluated for different performance metrics including operating frequency, execution time, resource utilization, energy and power dissipation. For simulation purposes, the cycle accurate simulator (ModelSim [67]) was used at Register Transfer Level (RTL) and at gate-level. The resource utilization was measured in terms of total numbers of Adaptive Look-Up Tables (ALUT), Adaptive Look-up Modules (ALMs), Logic Registers and 18-bit DSP elements. For a 32-bit multiplier instantiated in a CGRA, two 18-bit DSP elements are required. Altera's Stratix-IV and Stratix-V devices provide ALUTs and ALMs for logic synthesis, respectively. The size of an ALM is twice as ALUT [58]. The operating frequencies were calculated for different operating temperatures i.e., 0°C and 85°C. The power dissipation was estimated for the gate-level netlists of the designs by using the signal toggle-rate data generated for a specific or a set of application.

5.1 Comparative Analysis

As this research work proposes the design of CREMA, AVATAR, SCREMA, rASIP and HARP, it is important to present details related to different performance metrics to establish comparisons and summarize the outcome from the entire experimental work. To conduct comparisons, the measurements and

estimations of different performance metrics are provided as follows.

5.1.1 Resource Utilization

CGRA	Application	rows × columns	ALUT	Logic Registers	18-bit DSP
CREMA	MVM	6 × 8	11,757	9,606	64
AVATAR	MVM	4 × 16	21,131	10,518	64
CREMA	Radix-4 FFT	9 × 8	12,475	10,997	48
AVATAR	Radix-4 FFT	4 × 16	29,361	8,760	48

Table 4. *Resource Utilization by Accelerators Generated using* $9 \times 6/8$ *PE CREMA and* 4×16 *PE AVATAR.*

Table 4 shows that differently scaled CGRA but carrying the same amount of DSP resources and executing the same application may not require similar amount of other resources for logic implementation. In the table, it can be observed that the number of ALUTs and Logic Registers is not similar in application-specific cases. However, advantages are different from case to case, a CGRA providing higher parallelism (number of PE columns) provides shorter execution time at the cost of more logic resources in comparison to a CGRA with lower parallelism. In Table 5, it can be observed that the AVATAR System requires twice as many PEs to provide twice as much parallelism as CREMA, therefore the resource utilization almost doubles. The same is the case for a comparison between AVATAR and its dual version. Table 6 shows, when SCREMA is scaled-up from 4×4 to a 4×8 PE MVM accelerator, the resource utilization also doubles. The resource utilization keeps doubling as the CGRA scales-up.

It is shown in Table 7 that there is a considerable saving of ALUTs and Logic Registers when using rASIP in comparison to AVATAR and COFFEE RISC system which are working in a typical processor/coprocessor model. The only compromise is the data transfer bottle-neck which is caused by removing the DMA device and relying on load/store instructions of the processor.

Systems	ALUT	Logic Registers	DSP	%
CREMA Sys	19,463	13,613	64	7%
AVATAR Sys	40,227	19,313	128	15%
Double AVATAR Sys	80,263	37,609	256	31%

Table 5. *Resource Utilization Summary of AVATAR for Radix-(2, 4) FFT Accelerator.*

MVM Accelerator Size	Comb ALUTs	Logic Registers	DSPs
4×4 PE	2,566	2,766	16
4×8 PE	4,805	3,820	32
4×16 PE	8,259	6,784	64
4×32 PE	15,522	12,057	128

Table 6. *Resource utilization by MVM accelerators generated using SCREMA on Stratix-IV device.*

Table 8 shows the resource utilization by HARP on a Stratix-V FPGA device. The motivation behind HARP was to maximize the number of computational resources on a chip as shown by 92% utilization of FPGA DSP resources.

5.1.2 Operating Frequencies and Execution Times

This subsection provides a mixed description of performance based on processing time related parameters, e.g., execution time, operating frequency and number of clock cycles required for the execution of a kernel.

Table 9 shows overall execution times for processing a 32-bit integer 4^{th}-order MVM and a 64-point radix-4 FFT. It can be observed that as the CGRA provides more parallelism, the execution time decreases. In case of processing FFT in radix-4 scheme, a 9×8 PE CGRA generated accelerator performs almost equal to the one generated by 4×16 PE CGRA and for performing MVM, 4×16 PE CGRA generated MVM accelerator clearly outperforms the other in comparison. In this case, 4×16 PE CGRA is considered to be more

System	Comb ALUTs	Logic Registers	DSPs
C/A [3]	40,227	19,313	128
rASIP	36,083	14,429	112
Savings [%]	10.3	25.2	12.5

Table 7. *Resource utilization by COFFEE/AVATAR (C/A) System and rASIP on Stratix-IV FPGA device.*

ALMs	78,845 / 158,500	50%
Registers	64,436 / 634,000	10%
Memory Bits	21,000,928 / 38,912,000	54%
DSP	236 / 256	92%

Table 8. *Resource Utilization by the overall architecture of HARP on a Stratix-V FPGA Device.*

favorable as it can accommodate both of the applications in the same size. In Table 9, Slow and Fast relate to the operating frequencies achieved at the slow and fast corner-timing models for the FPGA device, respectively.

Table 10 shows the synthesis frequencies achieved for COFFEE, CREMA and AVATAR on Stratix-IV FPGA device using slow and fast timing models. The regularity in the structures of CREMA and AVATAR allows only small deviations in operating frequencies during the synthesis process.

The performance of rASIP is shown in Table 11. The table shows that rASIP is capable of satisfying execution time constraints imposed by IEEE-802.11n standard for FFT processing at the demodulator side.

5.1.3 Energy and Power Estimations

As a CGRA with high parallelism offers faster processing, it means there are more signals ready to be toggled at any particular time instant contributing to increase in dynamic power dissipation which can be observed from Table

App.	rows × columns	COFFEE (Slow)	ACR (Slow)	COFFEE (Fast)	ACR (Fast)	ACR. Cycles	Exe. Time
MVM	6×8	123.41	252.53	200.6	418.59	149	0.74
	4×16	119.89	228.0	198.57	327.44	91	0.45
Radix-4 FFT	9×8	112.47	191.02	182.52	320.92	221	1.22
	4×16	127.65	211.28	210.93	295.86	191	1.29

Table 9. *Synthesis Frequencies (MHz) and Execution Time (Exe.) in µs of COFFEE and CGRA generated Accelerators (ACR.).*

Timing Model	COFFEE	CREMA	AVATAR
CREMA Sys (Slow)	125.96	165.02	×
CREMA Sys (Fast)	199.28	273.15	×
AVATAR Sys (Slow)	120.63	×	177.34
AVATAR Sys (Fast)	195.96	×	278.32
D. AVATAR Sys (Slow)	118.41	×	166.42
D. AVATAR Sys (Fast)	194.59	×	256.81

Table 10. *Synthesis Frequencies (MHz) of COFFEE RISC and AVATAR generated accelerators in Single and Dual AVATAR System (D. AVATAR Sys).*

12. However, there is almost no difference in the static power dissipation as it should be a linear function of the resources used on the FPGA device. It is mainly because the un-utilized part of FPGA device can not be switched off and due to this reason, a large portion of static power adds an offset to the actual static power required by the design.

The energy consumption by rASIP is shown in Table 13. The overall energy consumption by rASIP is almost equal to COFFEE/AVATAR based system. Therefore while keeping energy consumption almost the same, a 25% saving in logic utilization shows advantages of using rASIPs over a standard processor/coprocessor model.

Another relationship of dynamic power dissipation and parallelism can be observed in HARP as shown in Table 14. A CGRA generated accelerator con-

FFT Algo	Freq. (MHz) Slow (100°C)	Freg. (MHz) Fast (0°C)	Cycles	Exe. Time
64-point	159.16	280.11	357	1.27μs
128-point			575	2.05μs
4×64			884	3.15μs

Table 11. *Operating Frequency and execution time by rASIP for processing 64 and 128-point FFT algorithms.*

MVM Accelerator	Static Power	Dynamic Power	I/O Power	Total Power
4×4 PE	428.48 mW	127.27 mW	58.07 mW	613.82 mW
4×8 PE	430.41 mW	208.63 mW	56.53 mW	695.57 mW
4×16 PE	435.11 mW	387.21 mW	56.95 mW	879.27 mW
4×32 PE	448.51 mW	728.25 mW	51.04 mW	1227.80 mW

Table 12. *Power consumption by MVM accelerators of different sizes of SCREMA.*

nected to a node of HARP dissipates more power if the number of PE columns is larger. Dynamic energy consumption is based on individual execution-time window of the kernel.

In Table 14, the dynamic power is estimated by simulating the gate-level netlist of the overall instance of HARP. For estimation purposes, the operating conditions are set as a temperature of 25°C and a frequency of 100.0 MHz. The number of clock cycles presented in Table. 3 approaches the timing information presented in Fig. 14, considering the frequency of 100.0 MHz. There are some calculation differences when relating Fig. 14 and Table 3, as in the table, clock cycles consumed during system start-up time are not considered. Additionally there can be some errors as measurements for clock cycle count are made while observing cursors on simulation time window.

HARP is also compared to other state-of-the-art as shown in Table 15 where Exe, dyn.pwr, freq stand for Execution, Dynamic Power and Frequency, respectively. In comparison to a general-purpose homogeneous MPSoC, it

System	FFT Algorithm	Frequency (MHz)	Clock Cycles	Execution Time (ns)	Power (mW)	Energy μJ
C/A	64-point	166.6	355	2,130	985.90	2.09
rASIP	64-point	160.0	357	2,231	1024.55	2.28
C/A	128-point	166.6	570	3,420	985.90	3.37
rASIP	128-point	160.0	575	3,593	1024.55	3.68

Table 13. *Energy consumption by C/A Accelerator and rASIP for processing 64 and 128-point FFT algorithms.*

showed a speed-up of 2.5X at the cost of 2X additional logic resources while processing 64-point FFT. The comparisons were also established against heterogeneous platforms, i.e., Multiple-Instruction Multiple-Data (MIMD) based platform of several microprocessors and P2012 which is a four-cluster architecture where each cluster contains four microprocessors. Another interesting platform for comparison is MORPHEUS which is a large computing platform of heterogeneous nature. In some cases, HARP is found to perform multiple times better than other platforms in units of Giga Operations per Second (GOPS) and GOPS/mW.

5.2 Case Study: Using Feedback Control for Power Efficiency

[1]This case study provides details related to a self-optimizing processor/coprocessor model that uses a control system to accelerate applications under user specified performance constraints, i.e., execution time while minimizing power dissipation. It is important to consider power dissipation as the savings can be supplied to other modules to increase system performance. COFFEE RISC acts as a processor while AVATAR as an accelerator. The RISC processor provides the overall supervisory control while AVATAR accelerates the targeted kernels. The control system is implemented in software in a way that in each control loop iteration the RISC processor counts the total number of

[1] The design and implementation of this case-study has been published in article [48].

Node CGRA Size Other HW	Algorithm (Type)	Dynamic Power (mW)	Active Time (μs)	Dynamic Energy (μJ)
N0 8×9 PE	Radix-4 FFT (64-point)	12.98	3523.3	45.7
N1 4×8 PE	Complex MVM (64-Order)	11.54	2270.2	26.2
N2 4×16 PE	Radix-(2, 4) FFT (128-point)	21.21	2458.6	52.1
N3 4×4 PE	Real MVM (32nd Order)	6.16	7631.7	47.0
N4 RISC	General Flow Control	5.72	13382.4	76.5
N5 4×8 PE	Real MVM (32nd Order)	9.96	5752.7	57.3
N6 4×16 PE	Real MVM (32nd Order)	17.57	3832.1	67.3
N7	-	0.03	-	-
N8 4×32 PE	Real MVM (32nd Order)	32.64	1896.7	61.9
NoC	-	0.67	13382.4	8.97
Integration Logic	-	31.54	-	-
Total	-	150.02	-	442.97

Table 14. *Dynamic Energy Estimation for each CGRA Node and the NoC.*

clock cycles it requires for AVATAR to complete the functional execution of the kernels. It then calculates the difference between the user-stated goal and the current number of clock cycles and based on the difference it tunes the frequency of the Phase Locked Loop (PLL) devices driving the computational and memory units of AVATAR. As soon as the desired operating frequency is achieved within the allowed tolerance levels, the control loop lock is achieved and further tuning of the operating frequency stops. It has been observed that

Platform / Technology	Performance Metric	Platform's Value	HARP's Value	Gain
NineSilica [62] / FPGA 40 nm	FFT Exe. Time (μs)	10.3	4.2	2.5X
TTA-MPSoC / [63] CMOS 130 nm	dyn.pwr/freq (mW/MHz)	0.52	1.5	0.34X
[64] / FPGA 90 nm	GOPS	19.2	40.8	2.0X
DREAM / [65] CMOS 90 nm	GOPS/mW	0.2	0.032	0.16X
P2012 / [28] CMOS 90 nm	GOPS/mW	0.04	0.032	0.8X
MORPHEUS / [26] CMOS 90 nm	GOPS/mW	0.02	0.032	1.6X

Table 15. *Comparisons based on different performance metrics with HARP implementation at 100.0 MHz on a 28 nm FPGA.*

when the control loop is active, there has been significant reduction in the average dynamic power dissipation.

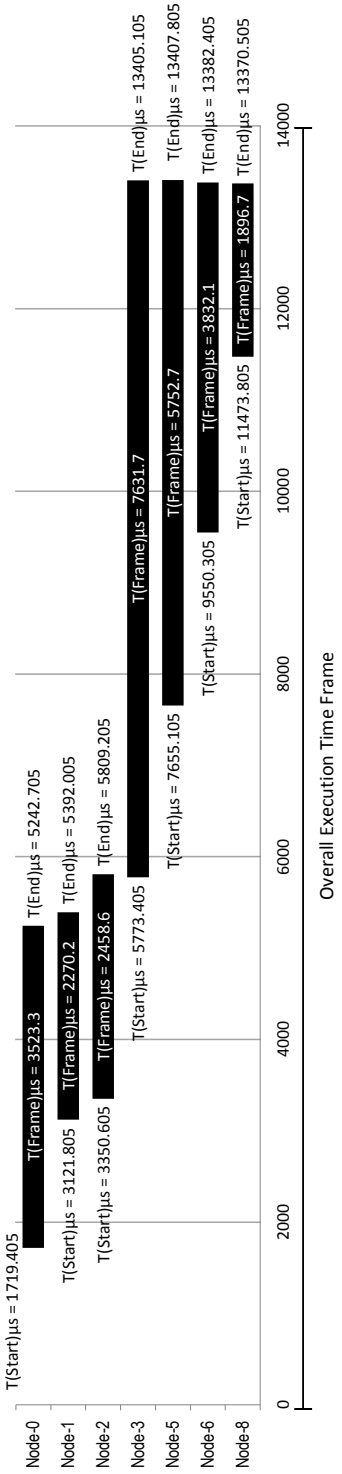


Fig. 14. *The activation and timing window of each kernel on a CGRA node in HARP.*

6. CONCLUSION

This thesis work expands from the design and development of different types CGRAs to a multicore heterogeneous architecture. It presents architectures of CREMA, AVATAR, SCREMA and HARP. The first three are template-based partially-scalable CGRAs while the later one is also a template based but a Heterogeneous Accelerator-Rich Platform. From CREMA, SCREMA and AVATAR, accelerator of sizes 4×4, 4×8, 6×8, 9×8, 4×16 and 4×32 PE were generated. These accelerators could perform integer or complex matrix-vector multiplication or Fast Fourier Transform processing while satisfying the execution time constraints of IEEE-802.11a/g/n standard. Furthermore, these accelerators were synthesized on Stratix-IV/V FPGAs and evaluated for different performance metrics including resource utilization, operating frequency, overall execution time, power dissipation and energy consumption. From the estimations, it could be observed that the dynamic power dissipation increases as the parallelism offered by the CGRA increases.

An rASIP was also designed as part of this research work by integrating AVATAR template into the datapath of a RISC processor. The instruction-set of the RISC core was extended by introducing additional instructions to enable basic and advanced operations on the AVATAR generated accelerator. The rASIP was performing 64 and 128-point FFT in radix-(2, 4) scheme while partially satisfying execution-time constraints for IEEE-802.11n standard. Measurement and estimation of different performance metrics proved the advantages of using rASIP over typical processor/coprocessor based system.

HARP's design was motivated by the Dark Silicon problem and to replace the under utilized part of the chip with special-purpose accelerators. Another

motivation was to maximize the number of PEs available on a chip to meet the ever increasing demand for throughput. HARP was a template-based architecture consisting of nine nodes of a NoC where each node contains a CGRA generated accelerator except the central node which contains a RISC processor. The RISC processor performs the overall supervisory control and establishes synchronization between the accelerator cores. With the help of the NoC, all the accelerator cores can address each other while working independently and simultaneously. The accelerator nodes were performing different lengths and types of FFT algorithms, real and complex MVM of different orders. HARP was compared to other state-of-the-art platforms in terms of many performance metrics. The HARP case-study shows the advantages of using a loosely-coupled heterogeneous accelerator-rich platform while maximizing the number of reconfigurable computational resources on a platform.

6.1 Future Work

CGRAs can be an interesting solution to the Dark Silicon problem of multicore scaling. They can be programmed at the data-flow level. However, in the current situation, the industry and academia do not have an appropriate compiler support for them. It would be a great opportunity for the research community to investigate into this area.

A standard model of homogeneous multicore scaling can be developed as a reference platform where the thermal breakdown is known at a certain operating frequency. Some of the cores can be replaced with special-purpose accelerators and then the performance of the platform can be re-evaluated.

The accelerator-rich platform has the potential to be a suitable candidate for IEEE-802.11ac, 4G and 5G radio standards. It would be interesting to evaluate HARP and its future derivatives for these standards.

Heterogeneous platforms exist in different definitions. A platform like HARP is using only CGRAs of different types and sizes for application-specific needs. A platform like MORPHEUS has an FPGA, a middle grain device

integrated with a VLIW processor and a CGRA. The homogeneous and heterogeneous platforms have their own advantages and disadvantages. It would be interesting to evaluate a blend of both, i.e., a partial-homogeneous integrated with a partial-heterogeneous platform.

BIBLIOGRAPHY

[1] G. Venkatesh, J. Sampson, N. Goulding, S. Gracia, V. Bryksin, J. L. Martinez, S. Swanson, M. B. Taylor, "Conservation cores: reducing the energy of mature computations", ASPLOS 10, pp. 205218, 2010.

[2] "IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput", IEEE, 3 Park Avenue, NY 10016-5997, USA, Oct 2009, E-ISBN : 978-0-7381-6046-7, Print ISBN: 978-0-7381-6047-4.

[3] W. Hussain, F. Garzia, T. Ahonen, J. Nurmi, "Designing Fast Fourier Transform Accelerators for Orthogonal Frequency-Division Multiplexing Systems", Journal of Signal Processing Systems for Signal, Image and Video Technology, Springer, Vol. 69, pp. 161-171, November 2012.

[4] B. Fu and P. Ampadu, "An Area Efficient FFT/IFFT Processor for MIMO-OFDM WLAN 802.11n", J Sign Process Syst, ISSN: 1939-8018, vol. 56, pp. 59-68, 2009

[5] H. Wawrzinek, "GARP: A MIPS processor with reconfigurable coprocessor". Proceedings of the Symposium on Field Programmable Custom Computing Machines, 1997.

[6] T. J. Callahan, J. R. Hauser, J. Wawrzynek, "The GARP Architecture and C Compiler", Computer, Vol. 33, pp. 62-69, Apr. 2000, doi: 10.1109/2.839323.

[7] N. S. Voros, A. Rosti, M. Hubner, ”Flexeos Embedded FPGA Solution”, in Dynamic System Reconfiguration in Heterogeneous Platforms, Lecture Notes in Electrical Engineering, Springer Netherlands, pp 39-47, vol. 40, ISBN 978-90-481-2426-8.

[8] S. Vassiliadis, S. Wong, G. N. Gaydadjiev, K. Bertels, G. Kuzmanov, E. M. Panainte, ”The Molen Polymorphic Processor”, IEEE Transactions on Computers, vol. 53, pp. 13631375, November 2004.

[9] E. M. Panainte, K. Bertels, S. Vassiliadis, ”The Molen Compiler for Reconfigurable Processors”, ACM Trans. Embed. Comput. Syst., Vol. 6, ISSN: 1539-9087, New York, USA, February 2007.

[10] A. Lodi, C. Mucci, M. Bocchi, A. Cappelli, M. De Dominicis, L. Ciccarelli, ”A Multi-Context Pipelined Array for Embedded Systems”, International Conference on Field Programmable Logic and Applications, 2006. FPL06, pp. 18, Aug. 2006.

[11] A. Lodi, M. Toma, F. Campi, A. Cappelli, R. Canegallo, R. Guerrieri, ”A VLIW Processor with Reconfigurable Instruction Set for Embedded Applications”, IEEE Journal of Solid-State Circuits, Vol. 38, pp. 1876-1886, Nov. 2003, doi: 10.1109/JSSC.2003.818292.

[12] S. Pillement, O. Sentieys, R. David, ”DART: a functional-level reconfigurable architecture for high energy efficiency”, EURASIP J. Embedded Syst. 2008, Article 5 (January 2008), 13 pages, DOI=10.1155/2008/562326, http://dx.doi.org/10.1155/2008/562326.

[13] D. Raphael, C. Daniel, P. Sebastien, S. Olivier, ”A Compilation Framework for a Dynamically Reconfigurable Architecture”, in Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream, Editors: M. Glesner and P. Zipf and M. Renovell, Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 1058-1067, Vol. 2438, ISBN: 978-3-540-44108-3, 2002.

[14] C. Brunelli, F. Garzia, and J. Nurmi, "A Coarse-Grain Reconfigurable Architecture for Multimedia Applications Featuring Subword Computation Capabilities", in Journal of Real-Time Image Processing, Springer-Verlag, 2008, 3 (1-2): 21-32. doi:10.1007/s11554-008-0071-3.

[15] C. Brunelli, F. Garzia, D. Rossi, J. Nurmi, "A Coarse-Grain Reconfigurable Architecture for Multimedia Applications Supporting Subword and Floating-point Calculations", Journal of Systems Architecture, Volume 56, Issue 1, January 2010, Pages 38-47, ISSN 1383-7621, http://dx.doi.org/10.1016/j.sysarc.2009.11.003.

[16] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. C. Filho, "Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications", IEEE Trans. Computers, vol. 49, no. 5, pp. 465-481, 2000.

[17] M-H. Lee, H. Singh, L. Guangming, N. Bagherzadeh, F. J. Kurdahi, E. M. C. Filho, C. A. Vladimir, "Design and Implementation of the MorphoSys Reconfigurable Computing Processor", The Journal of VLSI Signal Processing-Systems for Signal, Image, and Video Technology, Kluwer Academic Publishers, pp. 147-164, Vol. 24, March 2000.

[18] L. Guangming, H. Singh, L. Ming-Hau, N. Bagherzadeh, F. J. Kurdahi, E. M. C. Filho, V. Alves, "The MorphoSys Dynamically Reconfigurable System-on-Chip", Proceedings of the First NASA/DoD Workshop on Evolvable Hardware, pp. 152-160, 1999.

[19] H. Singh, L. Ming-Hau, L. Guangming, F. J. Kurdahi, N. Bagherzadeh, E. M. C. Filho, "MorphoSys: A Reconfigurable Architecture for Multimedia Applications", Proc. XI Brazilian Symposium on Integrated Circuit Design, pp. 134-139, Oct. 1998.

[20] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix", Field-Programmable Logic and

Applications, vol. 2778, pp. 61-70, September 2003, ISBN 978-3-540-40822-2.

[21] F. Bouwens, M. Berekovic, A. Kanstein, G. Gaydadjiev, P. Diniz, E. Marques, K. Bertels, M. Fernandes, J. Cardoso, "Architectural Exploration of the ADRES Coarse-Grained Reconfigurable Array", Reconfigurable Computing: Architectures, Tools and Applications, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, vol. 4419, pp. 1- 13, ISBN: 978-3-540-71430-9, 2007.

[22] Mei Bingfeng, F.-J. Veredas, B. Masschelein, "Mapping an H.264/AVC decoder onto the ADRES reconfigurable architecture", International Conference on Field Programmable Logic and Applications, pp. 622-625, Aug. 24-26, 2005.

[23] V. Baumgarte, G. Ehlers, F. May, A. Nuckel, M. Vorbach, and M. Weinhardt, "PACT-XPP A Self-Reconfigurable Data Processing Architecture", The Journal of Supercomputing, vol. 26, no. 2, pp. 167-184, September 2003.

[24] J. M. P. Cardoso, and M. Weinhardt, "XPP-VC: A C Compiler with Temporal Partitioning for the PACT-XPP Architecture", in Field-Programmable Logic and Applications: Reconfigurable Computing Is Going Mainstream, Editors: M. Glesner and P. Zipf and M. Renovell, Lecture Notes in Computer Science, Springer Berlin Heidelberg, pp. 864-874, Vol. 2438, ISBN: 978-3-540-44108-3, 2002.

[25] F. Campi, R. Konig, M. Dreschmann, M. Neukirchner, D. Picard, M. Juttner, E. Schuler, A. Deledda, D. Rossi, A. Pasini, M. Hubner, J. Becker, R. Guerrieri, "RTL-to-layout Implementation of an Embedded Coarse Grained Architecture for Dynamically Reconfigurable Computing in Systems-on-Chip", SOC 2009. International Symposium on System-on-Chip, 2009, pp.110,113, 5-7 Oct. 2009.

[26] N. S. Voros, M. Hübner, J. Becker, M. Kühnle, F. Thomaitiv, A. Grasset, P. Brelet, P. Bonnot, F. Campi, E. Schler, H. Sahlbach, S. Whitty, R. Ernst,

E. Billich, C. Tischendorf, U. Heinkel, F. Ieromnimon, D. Kritharidis, A. Schneider, J. Knaeblein, W. Putzke-Rming, "MORPHEUS: A Heterogeneous Dynamically Reconfigurable Platform for Designing Highly Complex Embedded Systems". ACM Trans. Embed. Comput. Syst. 12, 3, Article 70 (April 2013), 33 pages.

[27] F. Thoma, M. Kuhnle, P. Bonnot, E. M. Panainte, K. Bertels, S. Goller, A. Schneider, S. Guyetant, E. Schuler, K. D. Muller-Glaser, J. Becker, "MORPHEUS: Heterogeneous Reconfigurable Computing", International Conference on Field Programmable Logic and Applications, FPL 2007, pp. 409-414, 27-29 Aug. 2007.

[28] D. Melpignano, L. Benini, E. Flamand, B. Jego, T. Lepley, G. Haugou, F. Clermidy, D. Dutoit, "Platform 2012, a Many-Core Computing Accelerator for Embedded SoCs: Performance Evaluation of Visual Analytics Applications". In Proc. 49th Annual Design Automation Conference (DAC '12). ACM, New York, NY, USA, 1137-1142.

[29] F. Conti, C. Pilkington, A. Marongiu and L. Benini, "He-P2012: architectural heterogeneity exploration on a scalable many-core platform", in Proc. of the 24th edition of the great lakes symposium on VLSI (GLS-VLSI '14), pp. 231-232, ACM, New York, NY, USA.

[30] R. Airoldi, F. Garzia, O. Anjum, J. Nurmi, "Homogeneous MPSoC as baseband signal processing engine for OFDM systems", International Symposium on System on Chip (SoC), 2010, pp. 26-30, Sept. 2010, doi: 10.1109/ISSOC.2010.5625562.

[31] R. Airoldi, F. Garzia, J. Nurmi, "FFT Algorithms Evaluation on a Homogeneous Multi-processor System-on-Chip", 2010 39th International Conference on Parallel Processing Workshops (ICPPW), pp 58-64, 13-16 Sept. 2010.

[32] M. B. Taylor, J. Kim, J. Miller, D. Wentzlaff, F. Ghodrat, B. Greenwald, H. Hoffman, P. Johnson, Jae-Wook Lee, W. Lee, A. Ma, A. Saraf,

M. Seneski, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, A. Agarwal, ”The Raw microprocessor: a computational fabric for software circuits and general-purpose programs,” Micro, IEEE , vol. 22, no. 2, pp. 25,35, Mar/Apr. 2002.

[33] T. Ahonen, T. D. ter Braak, S. T. Burgess, R. GeiBler, P. M. Heysters, H. Hurskainen, H. G. Kerkhoff, A. B. J. Kokkeler, J. Nurmi, J. Raasakka, G. K. Rauwerda, G. J. M. Smit, K. Sunesen, H. van Zonneveld, B. Vermeulen, X. Zhang, ”CRISP: Cutting Edge Reconfigurable ICs for Stream Processing”, in Reconfigurable Computing, Joao M. P. Cardoso and M. Hubner, Ed. Springer New York, 2011, pp. 211-237, ISBN: 978-1-4614-0060-8.

[34] T. G. Mattson, M. Riepen, T. Lehnig, P. Brett, W. Haas, P. Kennedy, J. Howard, S. Vangal, N. Borkar, G. Ruhl, S. Dighe, ”The 48-core SCC Processor: the Programmer’s View”. In Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC ’10). IEEE Computer Society, Washington, DC, USA.

[35] P. Gschwandtner, T. Fahringer, R. Prodan, ”Performance Analysis and Benchmarking of the Intel SCC”, 2011 IEEE International Conference on Cluster Computing (CLUSTER), pp. 139-149, 26-30 Sept. 2011.

[36] S. Bell, B. Edwards, J. Amann, R. Conlin, K. Joyce, V. Leung, J. MacKay, M. Reif, Bao Liewei, J. Brown, M. Mattina, Miao Chyi-Chang, C. Ramey, D. Wentzlaff, W. Anderson, E. Berger, N. Fairbanks, D. Khan, F. Montenegro, J. Stickney, J. Zook, ”TILE64 - Processor: A 64-Core SoC with Mesh Interconnect”, IEEE International Solid-State Circuits Conference, 2008. ISSCC 2008. Digest of Technical Papers. pp. 88,598, 3-7 Feb. 2008.

[37] M. B. Taylor, W. Lee, J. Miller, D. Wentzlaff, I. Bratt, B. Greenwald, H. Hoffmann, P. Johnson, J. Kim, J. Psota, A. Saraf, N. Shnidman, V. Strumpen, M. Frank, S. Amarasinghe, and A. Agarwal. ”Evaluation of

the Raw Microprocessor: An Exposed-Wire-Delay Architecture for ILP and Streams", SIGARCH Comput. Archit. News 32, March 2004.

[38] F. Garzia, W. Hussain and J. Nurmi, "CREMA, A Coarse-Grain Reconfigurable Array with Mapping Adaptiveness", in Proc. 19th International Conference on Field Programmable Logic and Applications (FPL 2009). Prague, Czech Republic: IEEE, September 2009.

[39] J. Kylliäinen, T. Ahonen, and J. Nurmi, "General-purpose embedded processor cores - the COFFEE RISC example", In Processor Design: System-on-Chip Computing for ASICs and FPGAs, J. Nurmi, Ed. Kluwer Academic Publishers / Springer Publishers, June 2007, ch. 5, pp. 83-100, ISBN-10: 1402055293, ISBN-13: 978-1-4020-5529-4.

[40] C. Brunelli, F. Garzia, C. Giliberto and J. Nurmi, "A Dedicated DMA Logic Addressing a Time Multiplexed Memory to Reduce the Effects of the System Buss Bottleneck", in Proc. 18th International Conference on Field Programmable Logic and Applications, (FPL 2008), Heidelberg, Germany, 8-10 September 2008, pp. 487-490.

[41] F. Garzia, W. Hussain, R. Airoldi, J. Nurmi, "A Reconfigurable SoC tailored to Software Defined Radio Applications", in Proc of 27th Norchip Conference, Trondheim (NO), 2009.

[42] W. Hussain, F. Garzia and J. Nurmi, "Exploiting Control Management to Accelerate Radix-4 FFT on a Reconfigurable Platform", in Proc. International Symposium on System-on-Chip 2010. Tampere, Finland: IEEE, pp. 154-157, September 2010, ISBN: 978-1-4244-8276-4.

[43] F. Garzia, C. Brunelli and J. Nurmi, "A pipelined infrastructure for the distribution of the configuration bitstream in a coarse-grain reconfigurable array", in Proceedings of the 4th International Workshop on Reconfigurable Communication-centric System-on-Chip (ReCoSoC'08). Univ Montpellier II, July 2008, pp. 188-191, ISBN:978-84-691-3603-4.

[44] W. Hussain, F. Garzia, and J. Nurmi, ”Evaluation of Radix-2 and Radix-4 FFT Processing on a Reconfigurable Platform,” in Proceedings of the 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'10). IEEE, pp. 249-254, April 2010, ISBN 978-1-4244-6610-8.

[45] W. Hussain, T. Ahonen and J. Nurmi, ”Effects of Scaling a Coarse-Grain Reconfigurable Array on Power and Energy Consumption”, to appear in Proc. SoC 2012, Tampere, Finland, Oct 2012.

[46] W. Hussain, X. Chen, G. Ascheid, J. Nurmi, ”A Reconfigurable Application-specific Instruction-set Processor for Fast Fourier Transform processing”, 2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors (ASAP), pp. 339-345, 5-7 June 2013, Washington, D.C., USA.

[47] W. Hussain, R. Airoldi, H. Hoffmann, T. Ahonen, J. Nurmi, ”Design of an Accelerator-Rich Architecture by Integrating Multiple Heterogeneous Coarse Grain Reconfigurable Arrays over a Network-on-Chip”, in Proc. of 2014 IEEE 25th International Conference on Application-Specific Systems, Architectures and Processors (ASAP), 18-20 June 2014, Zurich, Switzerland.

[48] W. Hussain, H. Hoffmann, T. Ahonen, J. Nurmi, ”Constraint-Driven Frequency Scaling in a Coarse Grain Reconfigurable Array”, in Proc. of the International Symposium on System-on-Chip 2014, Tampere, Finland.

[49] Altera(2006) Stratix II vs. Virtex-4 Density Comparison. Consulted on 16 February 2007. Altera white paper at http://www.altera.com

[50] W. Hussain, T. Ahonen F. Garzia and J. Nurmi, ”Application-Driven Dimensioning of a Coarse-Grain Reconfigurable Array”, in Proc. NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2011), San Diego, California, USA.

[51] Hennessy JL, Patterson DA (1990) Computer Architecture: A Quantitative Approach. 3rd edn. Elseview Morgan Kaufmann, San Francisco.

[52] C. Panis, "VLIW DSP Processor for High-End Mobile Communication Applications", In Processor Design: System-on-Chip Computing for ASICs and FPGAs, J. Nurmi, Ed. Kluwer Academic Publishers / Springer Publishers, June 2007, ch. 5, pp. 83-100, ISBN-10: 1402055293, ISBN-13: 978-1-4020-5529-4.

[53] D. Vassiliadis, N. Kavvadias, G. Theodoridis, and S. Nikolaidis, "A RISC architecture extended by an efficient tightly coupled reconfigurable unit". In Proc. ARC, 2005.

[54] W. Hussain, X. Chen, G. Ascheid, J. Nurmi, "A Reconfigurable Application-specific Instruction-set Processor for Fast Fourier Transform processing", 2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors (ASAP), pp. 339-345, 5-7 June 2013, Washington, D.C., USA.

[55] F. Garzia, T. Ahonen, J. Nurmi, "A switched interconnection infrastructure to tightly-couple a RISC processor core with a coarse grain reconfigurable array," Research in Microelectronics and Electronics, 2009. PRIME 2009. Ph.D. , vol., no., pp.16,19, 12-17 July 2009.

[56] www.xilinx.com

[57] www.altera.com

[58] Altera, "Logic Array Blocks and Adaptive Logic Modules in Stratix-IV Devices", Chapter 2, Stratix-IV Device Handbook, Vol. 1, February 2011, Altera Corporation.

[59] Chia-Cheng Lo, Shang-Ta Tsai, Ming-Der Shieh; , "A reconfigurable architecture for entropy decoding and IDCT in H.264," VLSI Design, Automation and Test, 2009. VLSI-DAT '09. International Symposium on , vol., no., pp.279-282, 28-30 April 2009, doi: 10.1109/VDAT.2009.5158149, ISBN: 978-1-4244-2781-9.

[60] P. Kunjan, C. Bleakley, "Systolic Algorithm Mapping for Coarse Grained Reconfigurable Array Architectures", Reconfigurable Computing: Architectures, Tools and Applications, Lecture Notes in Computer Science, 2010, Springer Berlin Heidelberg, pp 351-357, Vol. 5992, DOI: 10.1007/978-3-642-12133-3_33.

[61] Y. Kishimoto, S. Haruyama, H. Amano, "Design and Implementation of Adaptive Viterbi Decoder for Using A Dynamic Reconfigurable Processor" in Proc. Reconfigurable Computing and FPGAs, 2008. ReConFig '08, pp 247-252, doi=10.1109/ReConFig.2008.39, ISBN: 978-1-4244-3748-1.

[62] R. Airoldi, F. Garzia, O. Anjum, J. Nurmi, "Homogeneous MPSoC as baseband signal processing engine for OFDM systems", International Symposium on System on Chip (SoC), 2010, pp. 26-30, Sept. 2010, doi: 10.1109/ISSOC.2010.5625562.

[63] O. Anjum, T. Ahonen, J. Nurmi, "MPSoC based on Transport Triggered Architecture for baseband processing of an LTE receiver", Journal of Systems Architecture, Volume 60, Issue 1, January 2014, Pages 140-149, ISSN 1383-7621.

[64] P. Bonnot, F. Lemonnier, G. Edelin, G. Gaillat, O. Ruch, P. Gauget, "Definition and SIMD implementation of a multi-processing architecture approach on FPGA". In Proc. of Design, Automation and Test in Europe (DATE '08). ACM, New York, NY, USA, 610-615.

[65] F. Campi, A. Deledda, M. Pizzotti, L. Ciccarelli, P. Rolandi, C. Mucci, A. Lodi, A. Vitkovski, L. Vanzolini, "A dynamically adaptive DSP for heterogeneous reconfigurable platforms". In Proc. of Design Automation and Test in Europe (DATE '07). EDA Consortium, San Jose, CA, USA, 9-14.

[66] S. Vassiliadis, D. Soudris, "ADRES & DRESC: Architecture and Compiler for Coarse-Grain Reconfigurable Processors", In Fine- and Coarse-

Grain Reconfigurable Computing, B. Mei, M. Berekovic, J-Y, Mignolet, Springer Netherlands, 2007, pp. 255-297, ISBN: 978-1-4020-6504-0.

[67] http://www.mentor.com/

PUBLICATIONS

PUBLICATION 1

Application-Driven Dimensioning of a Coarse-Grain Reconfigurable Array

Waqar Hussain, Tapani Ahonen, Fabio Garzia, Jari Nurmi
Department of Computer Systems, Tampere University of Technology
P. O. Box 553, FIN-33101, Tampere, Finland
Email: firstname.lastname@tut.fi

***Abstract*— In this paper, we have scaled-up a Coarse-Grain Reconfigurable Array (CGRA) in specific widths to evaluate matrix-vector multiplication and radix-4 64-point Fast Fourier Transform (FFT) algorithms. The evaluation of different CGRAs is based on complexity in application mapping, performance and area utilization. Reducing the product development time and achieving higher reliability has always been the need of industry and system integrators. The low-level designers and application developers need higher performance with minimum resources. Based on our evaluation, we propose a choice of array sizes for fulfilling the needs of both system integrators and low-level designers.**

I. Introduction

Scaling an algorithm and maintaining a performance level while increasing or decreasing the number of processors has always been a matter of concern for designers. Reducing the product development cycle and maintaining a higher reliability especially for mission critical applications require system integration and scalability has always been important in this domain. On the other side the Non-Recurring Engineering (NRE) cost of the project increases due to the complexity in application mapping. To use a Coarse-Grain Reconfigurable Array (CGRA) is attractive for application engineers as it offers a high level of parallelism, throughput and run-time reconfigurability. These features make them ideal to support different applications and switch between them at run-time. Examples of some of the CGRAs are Morphosys[1], ADRES[2], PACT-XPP[3], BUTTER[6] and CREMA[4]. However, it has always been challenging to decide which kind of CGRA to be used for a set of applications. The mapping of applications depends on the order of a CGRA in terms of rows and columns. The selection of the order is also complicated as it directly effects the performance, area utilization and NRE cost of the project.

A CGRA is an array of processing elements (PE) and each PE is capable to perform a set of different arithmetic, algebraic and logical tasks. The PEs can exchange data with each other using different types of interconnections, for example multiplexed point-to-point or network infrastructure. A CGRA can be a stand-alone device or working as an accelerator with a Reduced Instruction Set Computing (RISC) processor. The RISC processor can perform general-purpose processing while the computationally intensive kernels can be processed by the CGRA. Many successful case studies related to CGRAs are meeting the timing constraints for FFT processing in Orthogonal Frequency-Division Multiplexing (OFDM) based demodulators [5], Wideband Code Division Multiple Access (WCDMA) cell search [7], Finite Impulse Response Filter [14], Turbo Codes [15] and Image and Video processing applications [6], [16].

In this paper, we have selected CREMA which is a 4×8 PE CGRA and scaled it to a 6×8, 9×8 and 4×16 PE array required for the mapping of $(1 \times 4) \times (4 \times 4)$ matrix-vector multiplication and radix-4 64-point FFT kernels. We have analyzed the mapping of these kernels, performed simulations and obtained synthesis results. Based on our analysis, we observed the influence of dimensioning on two different design strategies i.e.,

1) Rapid Prototyping and System Integration
2) Global Optimum Implementation for Area and Speed

The paper is organized as follows. Section II discusses the architecture of CREMA and its scaling to generate 6×8, 9×8 and 4×16 PE arrays. In the next section, we give a brief introduction to matrix-vector multiplication and FFT. In Section IV we present the mapping of $(1 \times 4) \times (4 \times 4)$ matrix-vector multiplication and radix-4 64-point FFT on the scaled arrays. Section V discusses the results and finally we draw some conclusions.

II. CREMA and its Scalability

CREMA is a 32-bit template-based CGRA which means it can generate special-purpose accelerators on the specified mapping. The template mechanism allows only those hardware resources to be generated that are required for a specific application. CREMA generated accelerator works with COFFEE RISC processor [8] in a processor/co-processor model. The overall system comprises also a memory and interface for the peripherals. All the system components interact with each other using a network of switched interconnections. CREMA consists of a matrix of 4×8 PEs and each PE can be connected to the other PEs using local and global connections. The details about the structure of CREMA and its local and global connections can be found in [4]. Each PE is capable to perform a set of different arithmetic and logical operations in integer and IEEE-754 floating point format. CREMA is equipped with two 32-bit local memories each of size 16×256. In between the local memories, there are I/O buffers that provide interleaving to the data during the transfer from the local memories to the processing array. The I/O buffers consist of registers and multiplexers. The multiplexers interleave the data

and then the data is stored in the registers. The I/O buffers and the PEs are supported by configuration memories that are loaded with configuration words. These words select a pattern of interleaving among the I/O buffers and the operation to be performed by the PEs at a particular time. The combination of I/O buffers interleaving pattern and the set of operations to be performed by the PEs define a context. The context is designed and implemented by the user using a graphical platform or a C-based programming language. The accelerator generated can support at maximum eight different set of operations for the PE array and 32 different interleaving patterns for the I/O buffers. However more sets of operations can be designed and loaded replacing the previous ones as required. The configuration words are loaded in the configuration memories of the PEs from the main memory using the using the Direct Memory Access (DMA) device [9]. Similarly using the DMA, the data to be processed are loaded from the main memory of COFFEE RISC processor to one of the local memories of CREMA. The configuration words for the PE array are distributed using a pipeline infrastructure which reduces the overhead for this task [11]. The control flow of the program is written in C and COFFEE RISC performs the control operations on CREMA generated accelerators by a polling mechanism. A typical control flow performed by these accelerators is explained in [10].

In addition to the template-based structure of CREMA which allows resource-aware mapping of applications, scaling required by an application can be performed to keep the number of execution contexts to only one. Scaling can be achieved by increasing or decreasing the number of rows or columns as required by the application. For example, in [10], the 4×8 PE size of CREMA allows two radix-2 FFT butterflies to be mapped in parallel. If such a parallelism is not required and the task could be performed by only one butterfly then half of the array remains useless. In this case, it is better to instantiate only four columns than the fixed size of eight. In contrast to this, if higher throughput or parallelism is required, addition of rows or columns on demand can increase the scope of application mapping. Fig. 1 shows the overall system in use and scalability of CREMA in terms of rows and columns.

CREMA was designed using VHDL and the whole structure is parametric. It requires only changing a few parameters in the definition of CREMA that can generate more rows as required. However, it is not simple to add more columns. The I/O buffers that provide interleaving need to be modified as they are supposed to provide access to each input of the PE in the first row to every bank in one of the local memories. By increasing the array size by one column requires additional two memory banks in local memories to be added as the PE has two inputs. This would also require the I/O buffers to expand. The expansion would include the addition of two registers and two multiplexers. Since the number of inputs of regular multiplexers is a power of two, some of the multiplexer inputs will be wasted. This additional hardware will increase the area utilization and propagation delays. To avoid wasting these inputs, it is better to scale-up the number of columns

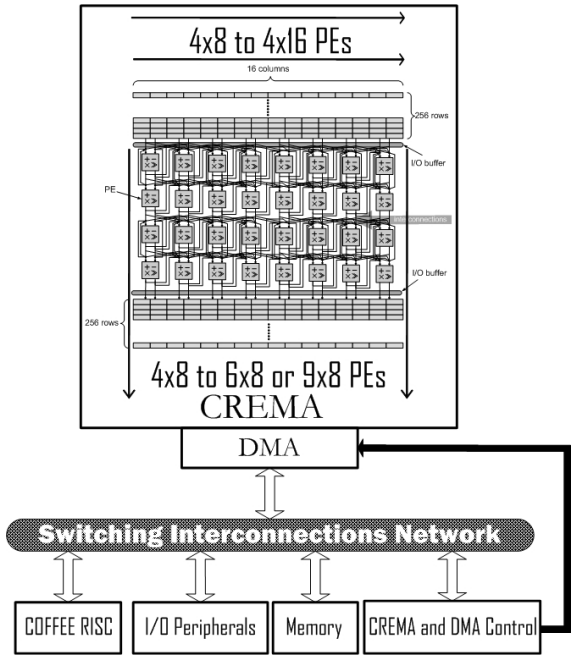


Fig. 1. System Architecture and Scaling in CREMA

in powers of two. For example, scaling of 4×8 PE array to 4×16 would need upgrading the multiplexer size one step up, i.e., from 16×1 to 32×1 and the number of memory banks would increase to 32. In this way, each input of a multiplexer will get connected to one of the 32 memory banks and none of the inputs will be wasted.

Scaling up or down a CGRA depends on the optimal mapping of an application in terms of performance, area utilization, power and development time. We have selected two important applications to scale-up CREMA: matrix-vector multiplication and radix-4 64-point FFT. Matrix-vector multiplication is of key importance in almost all disciplines of science and engineering. Algorithms from computational fluid dynamics to genetics involve exhaustive computations depending on matrix-vector multiplication. An FFT calculation is required many times during the analysis of signals. These days different communication standards using OFDM demodulation require FFT processing with strict timing constraints [13]. For both of these applications, we have scaled up CREMA to a 6×8, 9×8 and a 4×16 PE array. The 4×16 PE array allows mapping of both of these applications that are otherwise performed by 6×8 and 9×8 individually.

The extension of CREMA to 6×8 and 9×8 PEs CGRA was the only possible dimensioning to map matrix-vector multiplication and radix-4 FFT processing in a single context while restricting the number of columns to eight. A one step-up increase in the number of columns is 16 from 8 as the columns increases by a power of two. This is because of the

use of regular multiplexers in the I/O buffers. Both of the matrix-vector multiplication and radix-4 FFT butterfly can be mapped using four rows while keeping the number of columns equal to sixteen.

III. Matrix-Vector Multiplication and FFT Algorithm

The matrix-vector multiplication process can be defined as follows. Consider $A = [a_{i,j}]$ is an $N \times N$ matrix and $\overrightarrow{b} = [b_i]$ is a vector to be multiplied to produce a product vector $\overrightarrow{c} = [c_i]$, where

$$c_i = \sum_{j=0}^{N-1} a_{i,j} b_j \tag{1}$$

FFT is a technique to compute the discrete Fourier Transform of a signal in digital systems. Mathematically it can be defined as

$$X[k] = \sum_{n=0}^{N-1} x[k] W_N^{nk} \tag{2}$$

where n is any sample among N samples to be processed and $W_N^{nk} = exp(-j2\pi nk/N)$ is a twiddle factor.

The theory behind radix-$2^m, m \in Z^+$ FFT algorithm is well known [12]. The basic unit of FFT structure is called a 'butterfly' and is shown for $m = 1, 2$ with their mathematical representation and explanation in [10].

IV. Implementation on CGRA

The implementation of matrix-vector multiplication requires six rows if the number of columns are restricted to eight and N is considered equal to four in Eq. 1. A total number of sixteen multiplications need to be performed that require all the PEs of the first two rows. The sixteen elements of matrix A are stored in the immediate registers of the PEs in the first two rows while the vector $\overrightarrow{b}$ is loaded in the first local memory. To store the elements of matrix A in immediate registers, two execution contexts are required to address each row separately. Since we are performing integer processing, there has to be a shift operation after each multiplication. The shift amount can be loaded into the immediate registers of the third and fourth row using additional contexts. Time required to load the immediate registers with the shift amount is not considered part of the overall execution time since they can be loaded during the system start-up time. The fourth and fifth row is used to carry out the addition operation. The context used for the multiplication, shift and addition operations is shown in Fig. 2.

As another case study, we have implemented integer radix-4 FFT algorithm by increasing the number of rows to nine while restricting the number of columns to only eight. A total number of 16 inputs provided by eight columns can receive all the data required in parallel in a single cycle. The main processing context is the implementation of mathematical equations representing the radix-4 butterfly in [10]. The implementation of radix-4 butterfly requires twelve multiplication

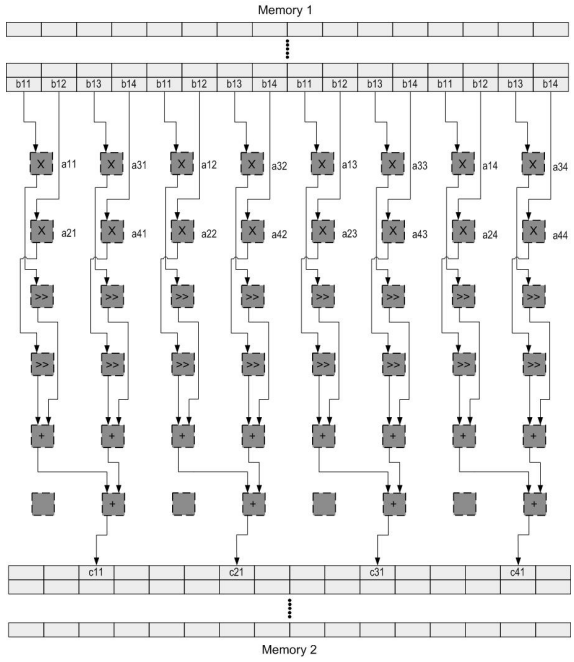


Fig. 2. Matrix-Vector Multiplication using 6×8 PE Array

operations and therefore twelve shift operations. The rest of the processing elements perform the required number of additions and subtractions. The shift operations require loading the shift amount by an additional context. The time required to load the shift amount is not considered in the overall execution since it can be considered in the system start-up time. A radix-4 64-point FFT algorithm requires sixteen butterflies in each processing stage to give a maximum processing speed but that is an expensive choice consuming more FPGA resources than implementing one butterfly and processes the 64 points of data over it. This method requires preprocessing between every processing stage to reorder data for the next stage processing. For this, we have implemented additional contexts based on delay-chains that reorder the data in time-division multiplexing fashion. The main processing context carrying the radix-4 butterfly is implemented using a 9×8 PE array and is shown in Fig. 3.

Both the applications discussed above require different number of rows while keeping the number of columns fixed. Contrary to increasing the number of rows, the number of columns was increased to sixteen and number of rows was fixed to four while scaling CREMA. The designed 4×16 PE array has a definite structural supremacy over the two 6×8 and 9×8 PE arrays as it can process both matrix-vector multiplication and radix-4 64-point FFT with a better or almost the same speed. Due to its ability to take 32 inputs at a time, it can easily receive sixteen words of the matrix and four words of the vector required for seamless mapping

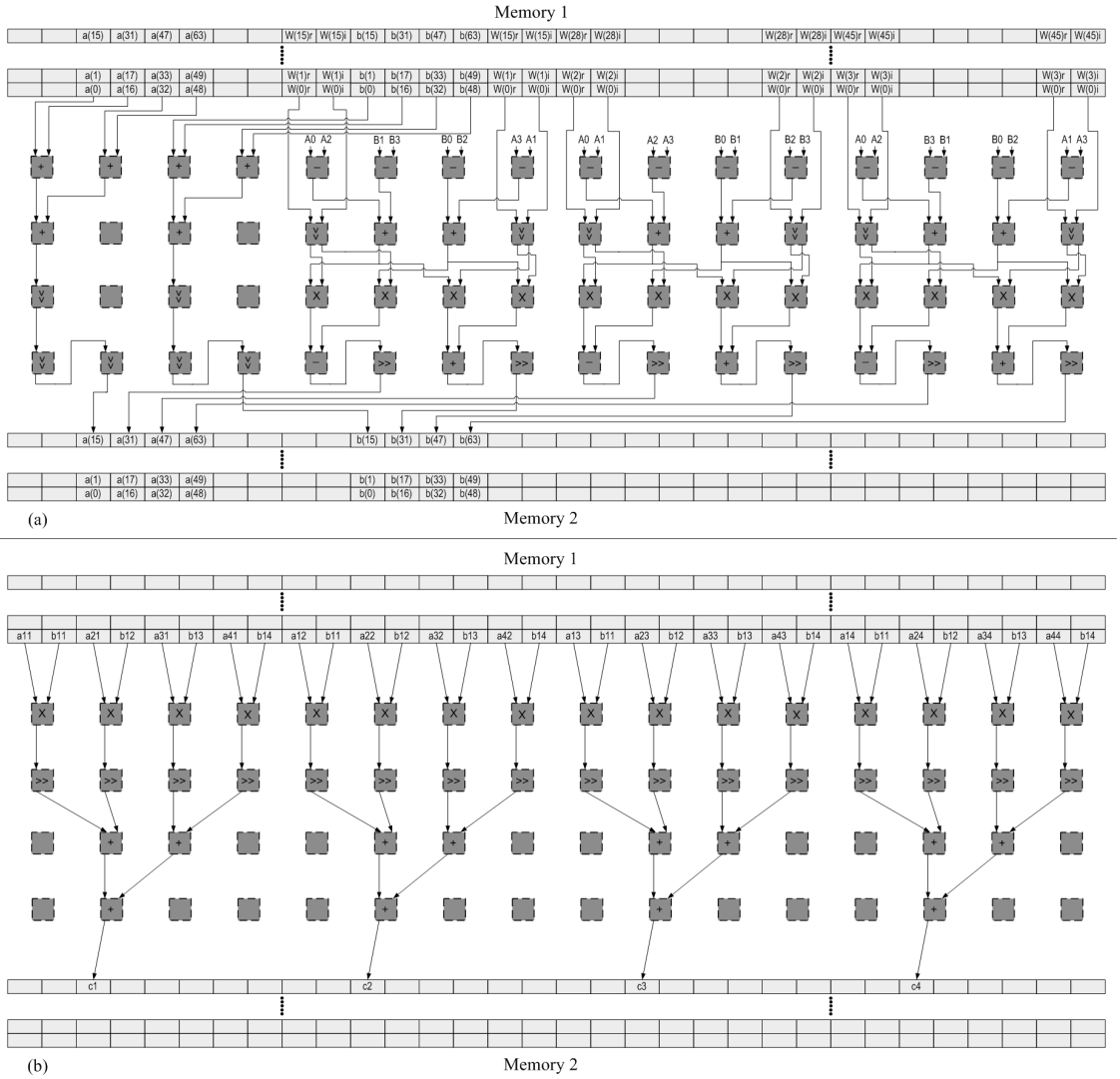


Fig. 4. Radix-4 FFT Butterfly and Matrix-Vector Multiplication Mapping on 4×16 PE Array

Application	rows × columns	ALUT	Memory ALUT	Logic Registers	18-bit DSP
Matrix-Vector Multiplication	6×8	11,757	48	9,606	64
Matrix-Vector Multiplication	4×16	21,131	0	10,518	64
Radix-4 FFT	9×8	12,475	448	10,997	48
Radix-4 FFT	4×16	29,361	0	8,760	48

TABLE I

RESOURCE UTILIZATION BY ACCELERATORS GENERATED USING CGRAS OF DIFFERENT SIZES

of $(1 \times 4) \times (4 \times 4)$ matrix-vector multiplication kernel. So in this case, only one context is required to implement the multiplication task. In case of FFT mapping on 4×16 PE array, all the equations shown in [10] characterizing a radix-4 FFT butterfly can be mapped in a single context. The preprocessing stages to reorder the data have been implemented using delay-chain contexts. Fig. 4/a shows radix-4 64-point FFT and Fig. 4/b shows matrix-vector multiplication mapping on 4×16 PE

Application	rows × columns	COFFEE (Slow)	ACR (Slow)	COFFEE (Fast)	ACR (Fast)	ACR Cycles	Exe Time
Matrix-Vector Multiplication	6×8	123.41	252.53	200.6	418.59	149	0.74 μs
Matrix-Vector Multiplication	4×16	119.89	228.0	198.57	327.44	91	0.45 μs
Radix-4 FFT	9×8	112.47	191.02	182.52	320.92	221	1.22 μs
Radix-4 FFT	4×16	127.65	211.28	210.93	295.86	191	1.29 μs

TABLE II

SYNTHESIS FREQUENCIES (MHZ) OF COFFEE AND ACCELERATORS (ACR) GENERATED

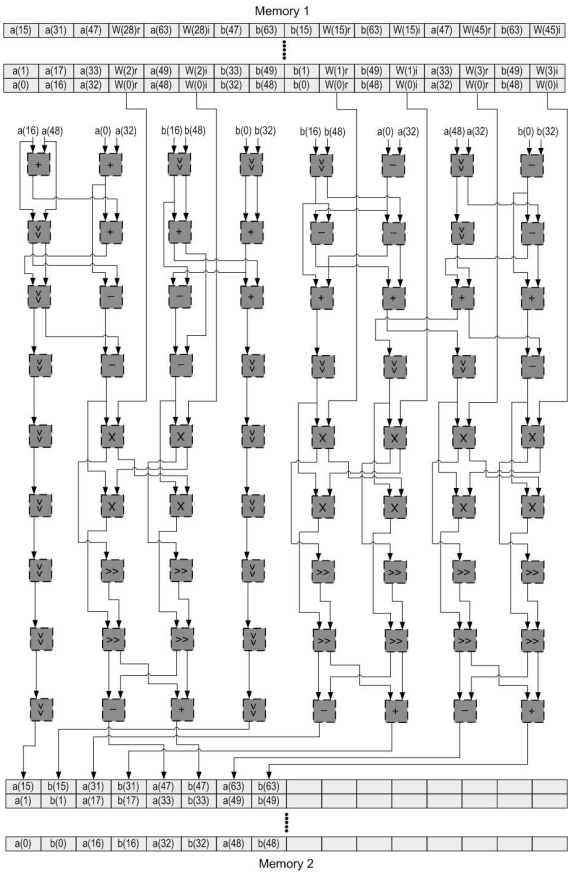


Fig. 3. Radix-4 64-point FFT Processing Context in a 9×8 PE Array

array.

V. RESULTS

The accelerators that we generated from different CGRA templates made by scaling-up CREMA were synthesized for Altera's Stratix-IV device. Simulations were performed using ModelSim and the number of clock cycles required for each kernel was measured in terms of COFFEE clock cycles since the generated accelerators always get a higher synthesis frequency than COFFEE, and on the other hand the comparison point is a software implementation executing on the host processor. During the simulations, the ratio between COFFEE and accelerator clock cycles was set according to the synthesis frequencies obtained.

Table I shows the area utilization summary of different accelerators generated. The consumption of ALUTs on the FPGA device is almost twice in case of 4×16 PE array than for the other arrays in comparison. This is mainly because of the increase in the multiplexer size from 16×1 to 32×1 in the I/O buffers. The consumption of DSP resources are the same. Since CREMA is a 32-bit CGRA, the result of multiplication of two 32-bit numbers would result in a 64-bit number. To deal with a 32-bit multiplication, four 18-bit elements on the FPGA device are required. In Fig. 2 and Fig. 4/b, it can be observed that the total number of 32-bit multipliers in the context are sixteen and in Table I, a total of $16 \times 4 = 64$ 18-bit DSP elements were used.

The structure of 4×16 PE array has a reduced application development time compared to the other arrays. However it is important to know the execution time required by different kernels on the accelerators generated. Table II shows the synthesis frequencies in the fast and slow timing models of the Stratix-IV device. The fast and slow timing models are based on different operating condition. Higher frequencies can be obtained while keeping the temperature low during the synthesis. From the table, it can be observed that the clock cycles required by 4×16 PE array are less than required by the other array sizes and it has a better or almost equal execution time. Considering the reduced application development time and the numerical figures from Table II, the 4×16 PE array will be the system integrator's choice. The compact mapping of radix-4 FFT butterfly in a 9×8 PE array requires almost half of ALUTs as shown in Table I and requires a bit less execution time than its comparative 4×16 PE array mapping making it a better choice for low-level designers concentrating on the characteristics of a single application.

The motivation for choosing those specific CGRA dimensions came from the natural parallelism of the algorithms evaluated. The time required to develop an application reduces if a higher bandwidth (number of columns) CGRA is used. A higher bandwidth allows step-by-step implementation of operations and in order placement of data in local memories related to a mathematical equation. On the other side, it is complex to design an accelerator with more pipeline registers, in other words increasing the number of rows of a CGRA. To implement an application with a limited bandwidth CGRA, the number of rows has to be increased to implement the operations required. This results in out of order placement

of data in local memories and irregular implementation of operations in a CGRA.

VI. Conclusion

A wrong decision in project management may lead to design failure. Decisions are mostly based on the objectives of the project and financial limitations. In this paper, we focused on the scalability of Coarse-Grain Reconfigurable Array (CGRA) devices and scaled-up a 4×8 processing element (PE) CGRA to 6×8, 9×8 and 4×16 PE CGRAs. We evaluated these CGRAs for development and execution time plus resource utilization. The evaluation was performed for two important and frequently used applications in science and engineering that are matrix-vector multiplication and Fast Fourier Transform algorithms. Based on our evaluation, we would recommend 4×16 (rows $\times$ columns) PE CGRA for rapid prototyping of applications and system integration. The 6×8 and 9×8 PE CGRA would be advisable for designers searching for global optimum implementation with respect to resource utilization and execution time.

References

[1] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. C. Filho, "Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications". IEEE Trans. Computers, vol. 49, no. 5, pp. 465-481, 2000.

[2] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix", Field-Programmable Logic and Applications, vol. 2778, pp. 61-70, September 2003, ISBN 978-3-540-40822-2.

[3] V. Baumgarte, G. Ehlers, F. May, A. Nuckel, M. Vorbach, and M. Weinhardt, "PACT XPP-A Self-Reconfigurable Data Processing Architecture", The Journal of Supercomputing, vol. 26, no. 2, pp. 167-184, September 2003.

[4] F. Garzia, W. Hussain and J. Nurmi, "CREMA, A Coarse-Grain Reconfigurable Array with Mapping Adaptiveness", in Proc. 19th International Conference on Field Programmable Logic and Applications (FPL 2009). Prague, Czech Republic: IEEE, September 2009.

[5] W. Hussain, F. Garzia and J. Nurmi, "Exploiting Control Management to Accelerate Radix-4 FFT on a Reconfigurable Platform", in Proc. International Symposium on System-on-Chip 2010. Tampere, Finland: IEEE, pp. 154-157, September 2010, ISBN: 978-1-4244-8276-4.

[6] C. Brunelli, F. Garzia, and J. Nurmi, "A Coarse-Grain Reconfigurable Architecture for Multimedia Applications Featuring Subword Computation Capabilities", in Journal of Real-Time Image Processing, Springer-Verlag, 2008, 3 (1-2): 21-32. doi:10.1007/s11554-008-0071-3.

[7] F. Garzia, W. Hussain, R. Airoldi, J. Nurmi, "A Reconfigurable SoC tailored to Software Defined Radio Applications", in Proc of 27th Norchip Conference, Trondheim (NO), 2009.

[8] J. Kylliainen, T. Ahonen, and J. Nurmi, "General-purpose embedded processor cores - the COFFEE RISC example", In Processor Design: System-on-Chip Computing for ASICs and FPGAs, J. Nurmi, Ed. Springer Publishers, June 2007, ch. 5, pp. 83-100, ISBN-10: 1402055293, ISBN-13: 978-1-4020-5529-4.

[9] C. Brunelli, F. Garzia, C. Giliberto and J. Nurmi, "A Dedicated DMA Logic Addressing a Time Multiplexed Memory to Reduce the Effects of the System Buss Bottleneck", in Proc. 18th International Conference on Field Programmable Logic and Applications, (FPL 2008), Heidelberg, Germany, 8-10 September 2008, pp. 487-490.

[10] W. Hussain, F. Garzia, and J. Nurmi, "Evaluation of Radix-2 and Radix-4 FFT Processing on a Reconfigurable Platform," in Proceedings of the 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'10). IEEE, pp. 249-254, April 2010, ISBN 978-1-4244-6610-8.

[11] F. Garzia, C. Brunelli and J. Nurmi, "A pipelined infrastructure for the distribution of the configuration bitstream in a coarse-grain reconfigurable array", in Proceedings of the 4th International Workshop on Reconfigurable Communication-centric System-on-Chip (ReCoSoC'08). Univ Montpellier II, July 2008, pp. 188-191, ISBN:978-84-691-3603-4.

[12] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series", Math. Comp., vol. 19, pp, 297-301, April 1965.

[13] J. C. Kuo, C. H. Wen and A. Y. Wu, "Implementation of a programmable 64-2048-point FFT/IFFT processor for OFDM based communication systems", in Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on Circuits and Systems, vol. 2, May 2003, pp. 121-124.

[14] P. Kunjan, C. Bleakley, "Systolic Algorithm Mapping for Coarse Grained Reconfigurable Array Architectures", Reconfigurable Computing: Architectures, Tools and Applications, Lecture Notes in Computer Science, 2010, Springer Berlin Heidelberg, pp 351-357, Vol. 5992, DOI: 10.1007/978-3-642-12133-3_33.

[15] Y. Kishimoto, S. Haruyama, H. Amano, "Design and Implementation of Adaptive Viterbi Decoder for Using A Dynamic Reconfigurable Processor" in Proc. Reconfigurable Computing and FPGAs, 2008. ReConFig '08, pp 247-252, doi=10.1109/ReConFig.2008.39, ISBN: 978-1-4244-3748-1.

[16] Chia-Cheng Lo, Shang-Ta Tsai, Ming-Der Shieh; , "A reconfigurable architecture for entropy decoding and IDCT in H.264," VLSI Design, Automation and Test, 2009. VLSI-DAT '09. International Symposium on , vol., no., pp.279-282, 28-30 April 2009, doi: 10.1109/VDAT.2009.5158149, ISBN: 978-1-4244-2781-9.

PUBLICATION 2

J Sign Process Syst (2012) 69:161–171
DOI 10.1007/s11265-011-0642-6

Designing Fast Fourier Transform Accelerators for Orthogonal Frequency-Division Multiplexing Systems

Waqar Hussain · Fabio Garzia · Tapani Ahonen · Jari Nurmi

Received: 2 December 2010 / Revised: 17 August 2011 / Accepted: 14 November 2011 / Published online: 4 December 2011

Abstract Designing accelerators for the real-time computation of Fast Fourier Transform (FFT) algorithms for state-of-the-art Orthogonal Frequency-Division Multiplexing (OFDM) demodulators has always been challenging. We have scaled-up a template-based Coarse-Grain Reconfigurable Array device for faster FFT processing that generates special purpose accelerators based on the user input. Using a basic and a scaled-up version, we have generated a radix-4 and mixed-radix (2, 4) FFT accelerator to process different length and types of algorithms. Our implementation results show that these accelerators satisfy not only the execution time requirements of FFT processing for Single Input Single Output (SISO) wireless standards that are IEEE-802.11 a/g and 3GPP-LTE but also for Multiple Input Multiple Output (MIMO) IEEE-802.11n standard.

Keywords FFT · CREMA · AVATAR · MIMO · OFDM

W. Hussain (✉) · F. Garzia · T. Ahonen · J. Nurmi
Department of Computer Systems, Tampere University of Technology, P.O. Box 527, 33101 Tampere, Finland
e-mail: waqar.hussain@tut.fi

F. Garzia
e-mail: fabio.garzia@tut.fi

T. Ahonen
e-mail: tapani.ahonen@tut.fi

J. Nurmi
e-mail: jari.nurmi@tut.fi

1 Introduction

Multifunction devices are gaining popularity in industrial and consumer markets. The competition increases among different vendors to produce low-cost high-end products with a vast number of appealing features. A well-known example of multifunction devices is a cellular phone. Today's users want their cell phones to work on different communication standards as well as with audio/video streaming features, as a digital camera and a navigator. All of this is possible with embedded technology. An embedded system can be defined as a special purpose computing system with a balanced mixture of software with flexibility and hardware for application performance. A good example of this is a Reduced Instruction Set Computing (RISC) processor working with accelerators in a processor/co-processor model.

Coarse Grain Reconfigurable Arrays (CGRA) are a powerful class of accelerators that offer high throughput and parallelism by their structure. Due to its ability to perform several tasks, a CGRA occupies an area of a few million gates which makes it expensive on the chip and its presence in the system can not justified unless they are heavily utilized. Examples of such general-purpose CGRAs are BUTTER [1], Morphosys [2], ADRES [3] and PACT-XPP [4]. The basic building block of a CGRA is a reconfigurable Processing Element (PE) which is capable of performing a set of arithmetic and logical tasks on word-sized data (16–64 bits depending on the system). Each PE can be connected to the neighbouring PEs in point-to-point fashion or using a network on chip. The user defines the operations to be formed by each PE and also the interconnections at compile time which together make up the

configuration patterns for the CGRA. These patterns are loaded in the configuration memories of the CGRA and are enabled one at a time by the RISC core. On enabling a pattern, the data start getting executed over the CGRA. In this process, the result from each PE gets passed on to the next PE depending on the interconnection configuration. In many cases only a few hardware operators are used for an application rather than the over-all resources carried by each PE. To overcome this problem, we developed a template-based CGRA called CREMA (Coarse-grain REconfigurable with Mapping Adaptiveness) [5]. Using a template-based CGRA, only those resources can be instantiated that are required for a particular application. CREMA has been proven efficient in many case studies like matrix-vector multiplication [5], Fast Fourier Transform (FFT) [6] and Wideband Code Division Multiple Access (WCDMA) cell search [7]. In Wireless Local Area Network (WLAN) 802.11n standard, there are 1–4 simultaneous data sequences that can be transmitted and received using 4×4 antenna configurations [8]. The transceiver has four Radio Frequency (RF) front ends followed by Analog to Digital (A/D) converters, Automatic Gain Control (AGC), time synchronization and frequency synchronization. Then there are four FFT processing blocks and then channel estimation and MIMO detection. The next cascaded blocks are a de-spatial parser, de-puncturer and channel decoder. The IEEE-802.11n transceiver can work with two lengths (64 or 128-points) of data sequences with different base-band frequency bandwidths, 20–40 MHz. As an option, one of them can be selected at a time depending on the throughput need. The block diagram of IEEE-802.11n WLAN receiver is shown in Fig. 1 [9, 10]. In all of these processing blocks, FFT blocks are among the most computation intensive. The IEEE-802.11n standard requires the FFT processing of 1–4 simultaneous data sequences with different timing constraints depending on their length. In particular four sets of 64 and 128-point FFT should get executed within 3.6 and 4.0 μs respectively [8]. The traditional approach is to increase the number of FFT accelerators as required but this increases the on-chip area proportionally. In this paper, we have treated four parallel streams serially and tried to use one FFT accelerator to process them within the given time constraints.

FFT is a special class of discrete Fourier transform (DFT). Mathematically the discrete Fourier transform can be defined, if the input samples to a system are x_n, where $n = 0, 1, 2, ..., N-1$ as

$$X[k] = \sum_{n=0}^{N-1} x[k] W_N^{nk} \tag{1}$$

where n is any sample among N samples to be processed and $W_N^{nk} = exp(-j2\pi nk/N)$ is a twiddle factor. The theory behind radix-2^m FFT algorithms is well known [11], where $m \in Z^+$ and its structural unit is called a butterfly. As m increases, the number of execution stages decreases for FFT processing but increases the arithmetic resources required by the butterfly in addition to the complexity of the FFT structure. The details regarding the butterfly and FFT structures can be found in [6]. A 64-point FFT requires six stages to be executed by a radix-2 butterfly but the same task can be performed in three stages if a radix-4 butterfly is used.

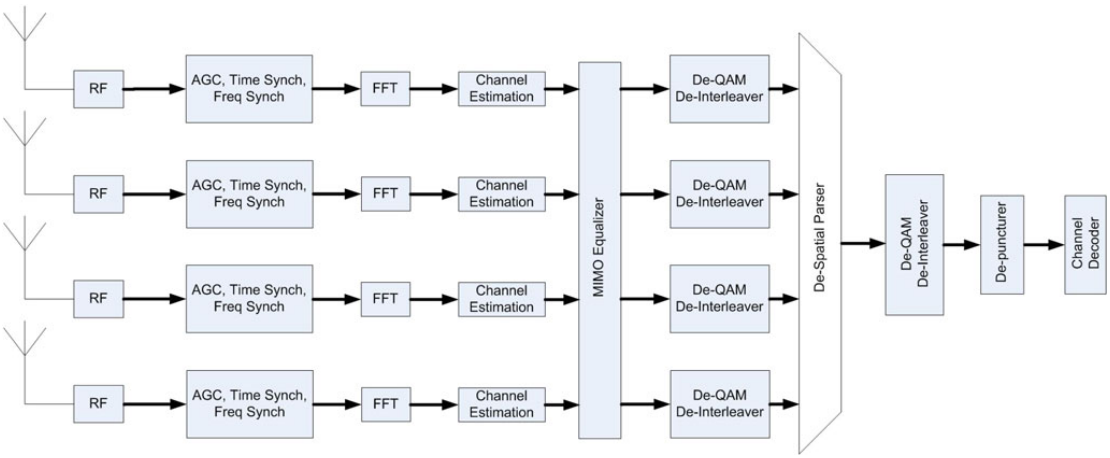


Figure 1 IEEE-802.11n receiver block diagram.

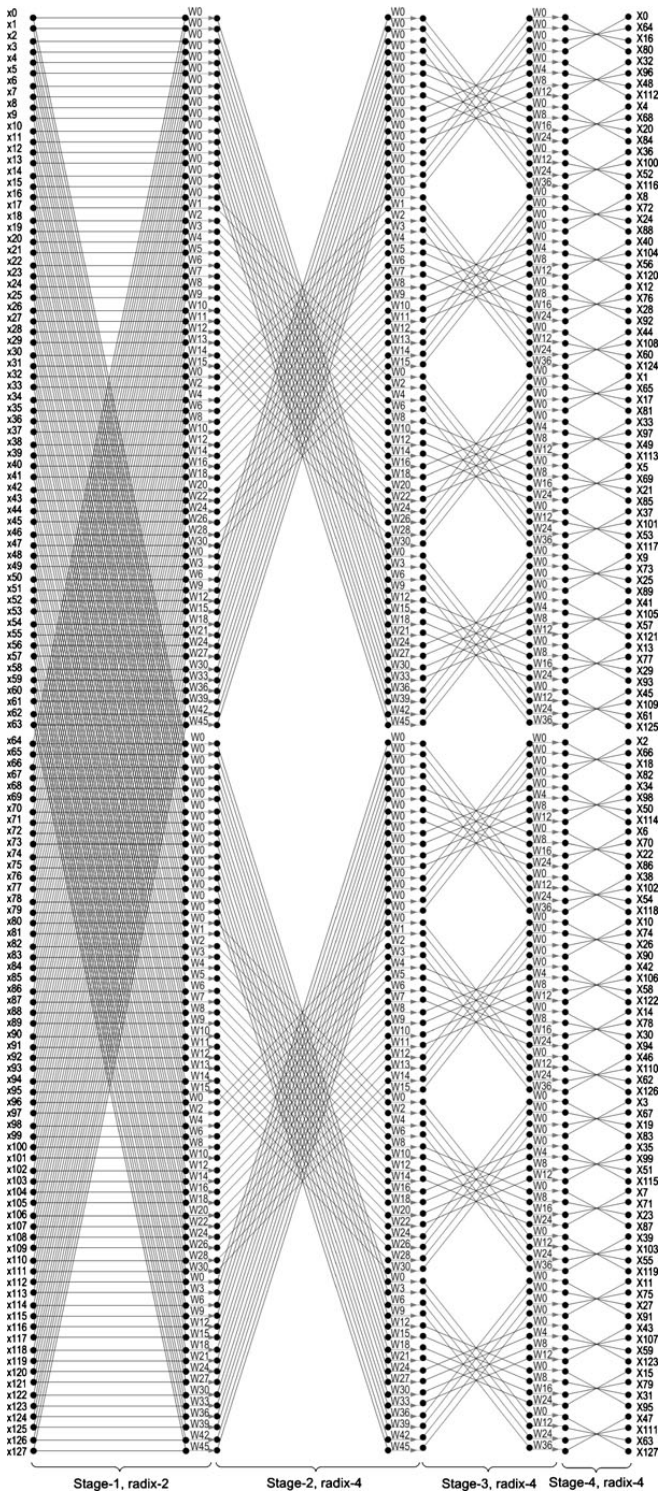


Figure 2 Mixed-radix (2, 4) FFT structure.

However a 128-point FFT can not be processed by a radix-4 butterfly and the radix-2 one requires expensive seven stages to process it. However, if a mixed-radix scheme is used, the 128-point FFT can be solved in four stages. This can be performed by processing the first stage with a radix-2 butterfly and as a result the 128-point FFT structure splits in two halves. Each half will be a 64-point FFT which can be processed by a radix-4 butterfly in three stages. The signal flow graph for a mixed-radix, 128-point FFT structure is shown in Fig. 2.

The accelerators generated using CREMA can process 64 and 1024-points FFT for SISO OFDM standards that are IEEE-802.11a/g and 3GPP Long Term Evolution (LTE) respectively [12]. To achieve higher FFT processing speed, we performed hardware and software upgrades on CREMA to create its scaled-up version called AVATAR (Advanced Value-Added Template Array with Reconfigurability) and implemented 4×64-point and 4×128-point FFT for MIMO OFDM IEEE-802.11n standard. The work in this paper carries the novelty for achieving FFT execution time constraints for IEEE-802.11n standard that has never been implemented before using a CGRA. In recent past, IEEE-802.11n performance was achieved by ASIC implementations of single and multipath delay feedback FFT processing hardwares mentioned in [9, 10].

In Section 2, we discuss the architecture of CREMA and the next section is dedicated to the structure, operational features and capabilities of AVATAR. Section 3 discusses the design of mixed-radix FFT accelerator using AVATAR and mapping of different length and types of FFT algorithm. Section 5 presents the simulation and synthesis results and finally we present the conclusion.

2 CREMA

CREMA is a template-based CGRA that allows to select at compile time only those hardware resources that are to be used by a specific application. In this way it generates special-purpose accelerators tailored for the specific mapping. CREMA is a 32-bit CGRA composed of 4×8 PEs and works with COFFEE [13] RISC in a processor/co-processor model. Each PE can perform integer and IEEE-754 floating-point arithmetic operations. The PE has two operand inputs and two outputs and can interact with the inputs and outputs of the neighbouring PEs in local and global fashion. The local connections are up-left (A,B), up-right (A,B), up (A,B), left (A,B) and loop (A), where A and B are the left and right outputs of PE. The local connection, loop(A) creates the feed-back path from the left or right output of a PE to any of the inputs of the PE above it. The global connections include horizontal and vertical connections. The horizontal connection connects any of the selected inputs of a PE to one of the outputs of the right-most PE in the immediate top row. The vertical connection connects any of the input of a PE to the I/O buffers which will be discussed later. A pattern of interconnections and operations to be performed by each PE in the CGRA defines a 'context'. The context switching is based on the configuration words stored in 32 memories, each associated to its corresponding PE. These configuration memories are loaded with words with the help of Direct Memory Access (DMA) [14] device which works in a tight coupling with CREMA. Once the configuration words are fetched from the main memory of COFFEE by the DMA, they are distributed over the CGRA with the help of a pipelined infrastructure [15]. Each configuration word is composed of an address field and data field. The pipelined infrastructure distributes the configuration words based on their address field. Each context is defined with the help of a graphical platform called Firetool (FIeld programming and REconfiguration management Tool) which is based on drop-down menu selection [16]. The contexts can also be defined using a C-based environment. Using the Firetool, the operation to be performed by each PE and the interconnections with neighbouring PEs are selected and saved. Additional contexts can be designed and implemented from time to time if required. Firetool then generates configuration files that are used for mapping. CREMA is equipped with two local memories and the data to be processed is loaded in one of the local memories. Once that data gets processed over the CGRA, it is stored in the other memory. The direction of data flow between the local memories can be reversed and the context can be switched as required unless the final results for an application have not been obtained yet.

3 System Architecture and AVATAR

To achieve higher FFT processsing speed, we wanted to develop a CGRA that could carry the whole radix-4 FFT butterfly in a single context. For this purpose, we developed AVATAR which is a 32-bit, 4×16 PEs CGRA. Many of the features of AVATAR have been inherited from CREMA. The system architecture is mainly composed of COFFEE RISC processor and AVATAR which is a highly parallel template-based CGRA. AVATAR is tightly coupled in the system with the help of a DMA device [14]. The general purpose

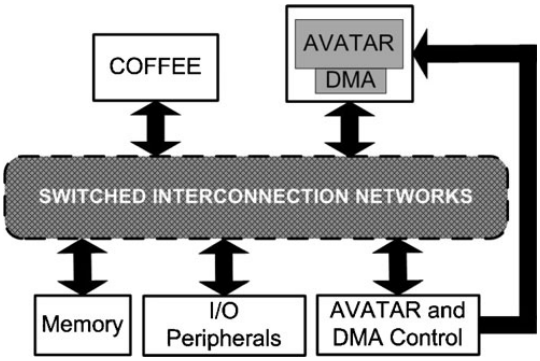


Figure 3 The embedded processing model.

processing is performed by COFFEE which is programmable in C while the computation intensive kernels are executed by the accelerators generated by AVATAR. The DMA interacts with COFFEE, system memory and I/O peripherals with the help of a matrix of switched interconnections that enable a higher data transfer rate. The control logic block allows the DMA and AVATAR to run with the timing parameters defined by the user. The over-all system diagram is shown in Fig. 3

AVATAR, depicted in Fig. 4, is a scaled-up version of CREMA with a higher operating frequency than COFFEE. AVATAR is equipped with two memories for the processing of data. Each memory has 32 banks that are dual-port and can store 512 × 32-bit words. The data to be processed is loaded from the main memory of the system to any of the local memories of AVATAR by using the DMA device. In between the local memories and AVATAR, there are I/O buffers that provide data interleaving in a way that every memory bank can be accessed by every input of the PEs in the first row. In CREMA, the I/O buffers were composed of 16 of 16 × 1 multiplexers and registers as to be able to take data from any of the 16 memory banks and to provide to the input of PEs in the first row. Each of the 16 multiplexers is associated to only one of the inputs in the first row of CREMA. Since CREMA was scaled up to be AVATAR, a total of 32 instances of 32 × 1 multiplexers were required with 32 registers to provide data from 32 memory banks to any of the PE inputs in the first row of AVATAR. The I/O buffer configuration words are loaded from the main memory to the configuration memories by the DMA transfers that are used to select different configurations for the I/O buffers. As there are 16 columns in AVATAR and there are 32 memory banks constituting each local memory, each

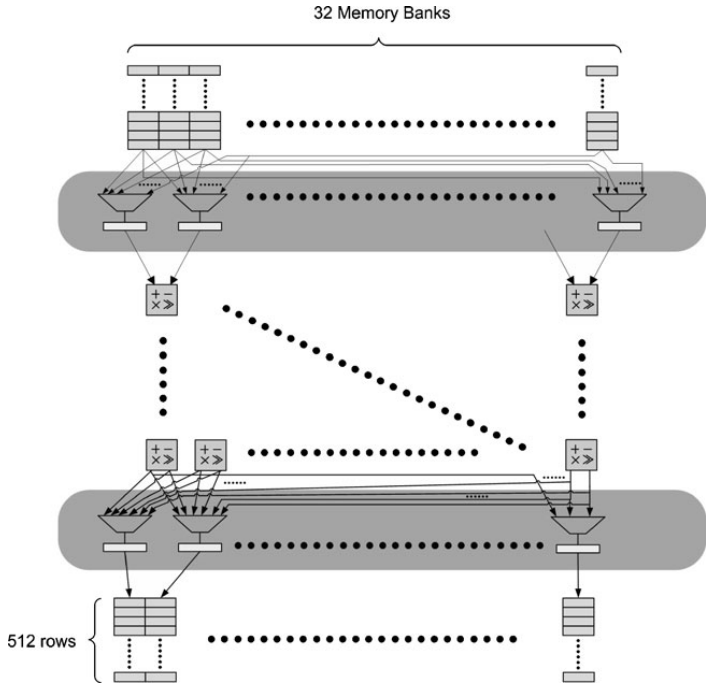


Figure 4 AVATAR (Advanced Value-Added Template Array with Reconfigurability).

input of a PE has a 1-to-1 correspondence to a memory bank in the local memories. The operations to be performed and interconnections among the PEs in AVATAR are similar to CREMA with the only difference in the vertical connections. The vertical connections in AVATAR bypasses the I/O buffers and connects the input of a PE directly with the corresponding memory bank as to provide maximum possible interleaving for the I/O buffers.

Every PE contains arithmetic and logical operators. The arithmetic operations can be performed in both integer and floating point (IEEE-754) format that include addition, subtraction, multiplication and also of immediate type. The logical operations include shift and the other types can be performed by using a look-up table inside each PE. As AVATAR is also a template like CREMA, only those arithmetic and logical resources are selected at compile time that are to be used at run-time to save as much on-chip area as possible. In this way only those resources are instantiated that the user selects. If a PE has to do more than one task, a configuration memory and a multiplexer are also generated inside a PE to select between output of different operators at run-time. The size of the configuration memory and multiplexer depends on the number of resources generated. The configuration words for memories inside a PE are also loaded by the DMA. The Firetool was also made flexible to support operations for AVATAR as well as for CREMA.

After the configuration words are loaded for the I/O buffers and the PEs, the data to be processed can be loaded in one of the local memories of AVATAR. A context for I/O buffers and PEs array can be enabled and data can be processed over the array. The processed data stored in the second memory after the interleaving provided by the second I/O buffer. The direction of data processing can be reversed and another context can be enabled as required.

4 Design of FFT Accelerator

A mixed-radix accelerator was designed using AVATAR by mapping four different contexts. The first two contexts contain the radix-2 and radix-4 butterflies respectively, the other two were used for data reordering. These contexts were designed and implemented using Firetool. Using the mixed-radix scheme, the accelerator can process different lengths of FFT structures efficiently. Every FFT structure requires a certain number of processing stages, for example radix-2, 64-point FFT structure requires six processing stages and will executed in minimum number of clock cycles if 32 butterflies are working in parallel. Increasing the number of butterflies for FFT processing increases the arithmetic resources required, so the conventional way is to use only one butterfly and process the data in a time-multiplexed manner. This requires preprocessing to re-order the data between the two consecutive stages of FFT.

The 4×16 array size of AVATAR allows four radix-2 butterflies to be mapped to work in parallel in a single context. This context is used to process the first stage of the 128-point FFT structure shown in Fig. 2. The word length for data to be processed and twiddle factors is selected to be 12 bits for both real and imaginary parts to meet IEEE-802.11n system requirements. The 128-point data followed by twiddle factors is loaded in the first local memory of the FFT accelerator using the DMA transfers. The sample points and twiddle factors are loaded in such a way so the task of processing is distributed among the four radix-2 butterflies equally. The 128 points are processed in 16 avatar clock cycles (cc) with a latency of four clock cycles. From Table 1, radix-2 FFT butterfly requires four multiplications and therefore four shift operations. Since we are processing 12-bit numbers into 32-bit register elements, the numbers do not overflow even if we do an addition or subtraction after multiplications. As shown in Fig. 5, a shift operation is performed after an addition or subtraction. The shift operation is performed according to a fixed value (12 bits) loaded in the immediate registers of the PE over the vertical connections. During the processing of data, the interleaving provided by the first I/O buffer distributes the data to the four radix-2 butterflies. The second I/O buffer interleaves the results from the array to be stored in the second local memory in a way that it can be used by the second context without any data re-ordering.

The second context is a radix-4 butterfly that processes the second, third and fourth stage as regular 64 point FFT processing by a radix-4 butterfly. To process 64 points FFT, only the second context is used to process a stage. The radix-4 FFT butterfly has four

Table 1 Stages, arithmetic and logical operators required by different length and types of FFT.

FFT type	Length	Stages	Adds	Subs	Muls	Shifts
Radix-2	64	6				
	128	7	3	3	4	4
	4×128	7				
Radix-4	64	3				
	1,024	5	15	15	12	12
	4×64	3				
Radix-2,4	4×128	4	18	18	16	16

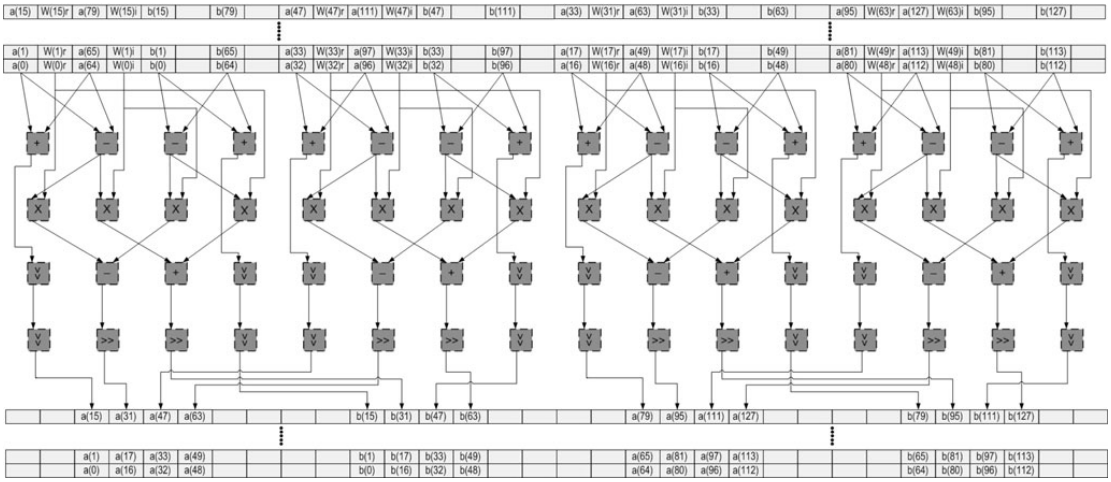


Figure 5 The first context consisting of four radix-2 butterflies.

complex samples and three twiddle factors as its inputs. This means fourteen integer inputs in total but the design of the second context (radix-4) requires redundant data loading of twiddle factors. The twiddle factors are supplied to the array by the vertical connections that are not flexible as they by-pass the I/O buffers. The samples processed by the first context (radix-2) have to be distributed to the four radix-2 butterflies in a way that the results generated by the context are following the splitting of radix-2 FFT structure in two equal parts. The two parts of the results generated by the first context are stored in left and right half of the second memory simultaneously. The direction of data processing now gets reversed and the second context

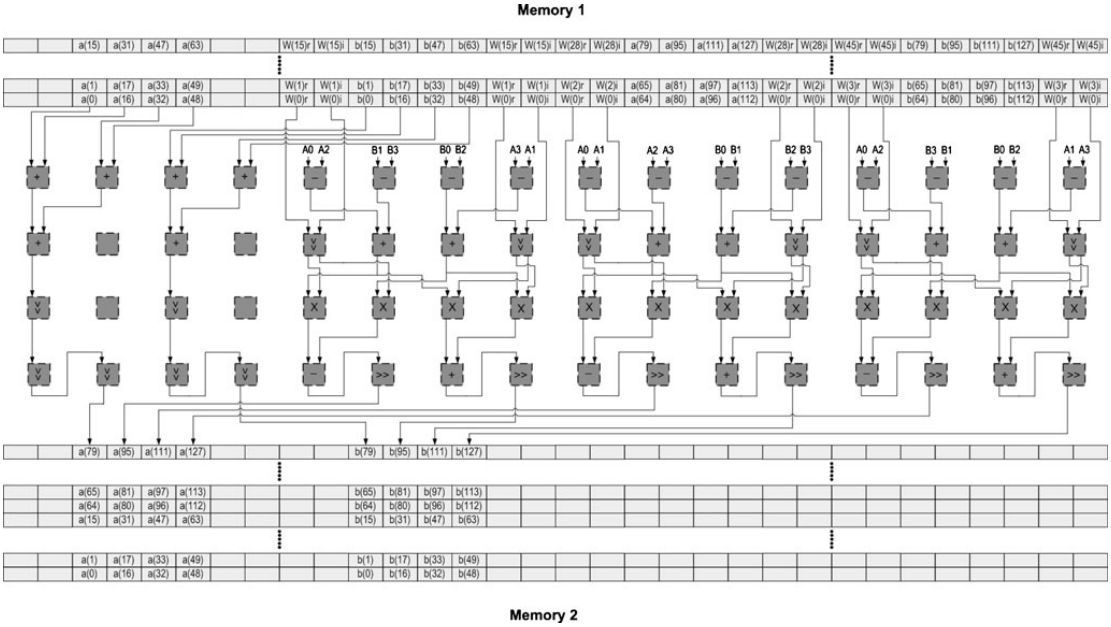


Figure 6 The second context consisting of a radix-4 butterfly.

Table 2 Synthesis frequencies (MHz) of COFFEE RISC and AVATAR generated accelerators in single and dual AVATAR system (D. AVATAR sys).

Timing model	COFFEE	CREMA	AVATAR
CREMA Sys (Slow)	125.96	165.02	×
CREMA Sys (Fast)	199.28	273.15	×
AVATAR Sys (Slow)	120.63	×	177.34
AVATAR Sys (Fast)	195.96	×	278.32
D. AVATAR Sys (Slow)	118.41	×	166.42
D. AVATAR Sys (Fast)	194.59	×	256.81

(radix-4) is enabled which processes the data generated by the first context from the left and right half of the second memory in two steps. In the first step, the I/O buffer selects the data from the left half of the second memory and then from the right half. The two-step execution by the second context require 2×32 cc with a latency of 5 cc, thus in total makes 42 cc. In the third and fourth stage, the processing of a stage requires 32 cc plus a latency of 5 cc making a total of 37 cc.

From Table 1, we can see that a radix-4 butterfly requires 12 multiplications and therefore 12 shift operations. Since we are processing 12-bit numbers on 32-bit register elements, there are not any overflows even if we do an addition or a subtraction after the multiplication and then a shift operation. In this way we have reduced the number of shift operations to only six instead of the required figure of twelve.

The radix-4 processing starts in the second stage and each stage from that on requires 32 cc with a latency of 5 cc. The context designed and implemented for the radix-4 butterfly is shown in Fig. 6. Between two consecutive stage processing, data re-ordering is required. The data re-ordering is performed by two additional contexts that are based on delay chains. To re-order the data, preprocessing contexts are enabled with certain timing parameters to distribute the data in a way as to get processed by the next context. The preprocessing stages require the number of clock cycles based on the length and the type of FFT structure.

5 Simulation and Synthesis Results

Different lengths and types of FFT kernels were simulated in MATLAB and then mapped on the radix-4 and mixed-radix (2, 4) accelerator generated by CREMA and AVATAR respectively. Simulations were performed using ModelSim-Altera software and number of system clock cycles were measured for each kernel. The primary measurements are performed in terms of COFFEE RISC clock cycles both in case of CREMA and AVATAR generated accelerators. The results of simulations by ModelSim were compared with MATLAB to verify the correctness. The radix-4 and mixed-radix FFT accelerators designed using CREMA and AVATAR were synthesized using a Altera's Stratix-IV FPGA device (EP4S100G5H40I1). Table 2 shows the synthesis frequencies for COFFEE RISC in the fast and slow timing models and also for CREMA and AVATAR generated accelerators. The table shows that the CREMA-generated radix-4 accelerator has a 1.37 X and AVATAR-generated mixed-radix accelerator has a 1.42 X higher operating frequency than COFFEE RISC. After setting these ratios, we performed the simulations for different length and types of FFT kernels. The number of COFFEE RISC clock cycles required for execution were measured and execution time considering the fast timing model was calculated shown in Table 3.

The processing of 4×64-points by radix-4 FFT accelerator generated using CREMA requires 802 COFFEE RISC clock cycles and the total execution times comes out to be 4.02 μs which does not satisfy the 3.6 μs constraint for 802.11n standard. AVATAR generated mixed-radix accelerator satisfies this constraint as shown in Table 3 but the system shown in Fig. 3 was unable to qualify the 802.11n timing constraints for 4×128-points FFT processing. To achieve this timing constraint, a dual system was assembled in a way that the two systems shown in Fig. 3 are connected with each other using a dual-port memory interfaced with both of the network of switched interconnections on each side. The first stage of 4×128-point FFT is processed by radix-2 butterflies in both of the systems and then each half gets processed by one of the mixed-radix accelerators in the dual system. From Table 2, we can observe that AVATAR works 1.3X faster than COFFEE RISC in the dual system. The area of the FPGA device consumed by the system shown in Fig. 3 and the dual system is shown in Table 4. The table also shows area consumed by a CREMA-based system

Table 3 Clock cycles required by different length and type of FFT accelerators.

FFT type	Length	Clock cycles	Exe time	CGRA template	IEEE standard
radix-4	64	320	1.6 μs	CREMA	802.11 a/g ✓
radix-4	1,024	12,952	64.9 μs	CREMA	3GPP-LTE ✓
radix-4	4×64	603	3.07 μs	AVATAR	802.11n ✓
mixed-radix-(2, 4)	4×128	759	3.9 μs	AVATAR	802.11n ✓

Table 4 Area utilization summary of EP4S100G5H40I1 device.

Systems	ALUT	Logic Regs	DSP	%
CREMA sys	19,463	13,613	64	7
AVATAR sys	40,227	19,313	128	15
D. AVATAR sys	80,263	37,609	256	31

which is exactly like the system shown in Fig. 3 but AVATAR replaced by CREMA. In sumamry, we can observe that the resources of the FPGA device double if Dual AVATAR system is compared with AVATAR system and AVATAR system with CREMA system.

The Single-path Delay Feedback (SDF) and Multi-path Delay Commutator (MDC) architectures discussed in [9] and [10] were specific for FFT processing and targeted for ASIC implementation. These SDF and MDC architectures are the most common to achieve timing constraints for IEEE-802.11n MIMO OFDM systems. The use of a template-based CGRA is not only limited to generate FFT accelerators to achieve OFDM performance but also suitable for many other applications as discussed about CREMA in Section 2. The suitability for multiple applications makes our architectures different from conventional SDF and MDC architectures that are specific for FFT processing. To the best of our literature study, mixed-radix FFT accelerator based on a CGRA has not been used so far to achieve IEEE-802.11n performance. In future, AVATAR will be used to generate accelerators for many other applications requiring high computational performance.

6 Conclusion

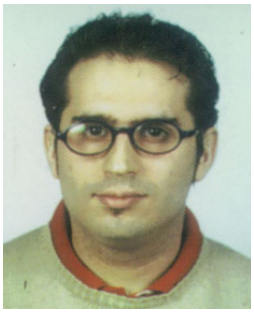
In this paper, we have presented the design of a large scale template based Coarse Grain Reconfigurable Array called AVATAR which generates special purpose accelerators on the specified mapping. A mixed-radix Fast Fourier Transform accelerator was designed using AVATAR and different lengths and types of Fast Fourier Transform kernels were mapped on it. The simulation and synthesis results show the performance of the Fast Fourier Transform accelerators satisfies different Single Input Single Output and Multiple Input and Multiple Output Orthogonal Frequency-Division Multiplexing timing constraints for Fast Fourier Transform processing.

References

1. Brunelli, C., Garzia, F., & Nurmi, J. (2008). A Coarse-Grain Reconfigurable Architecture for Multimedia Applications Featuring Subword Computation Capabilities. In *Journal of real-time image processing* (Vol. 3, Nos. 1–2, pp. 21–32). Springer-Verlag. doi:10.1007/s11554-008-0071-3.
2. Singh, H., Lee, M. H., Lu, G., Kurdahi, F. J., Bagherzadeh, N., & Filho, E. M. C. (2000). Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications. *IEEE Transactions on Computers, 49*(5), 465–481.
3. Mei, B., Vernalde, S., Verkest, D., Man, H. D., & Lauwereins, R. (2003). ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix. *Field-Programmable Logic and Applications, 2778*, 61–70, ISBN 978-3-540-40822-2.
4. Baumgarte, V., Ehlers, G., May, F., Nuckel, A., Vorbach, M., & Weinhardt, M. (2003). PACT XPP-A self-reconfigurable data processing architecture. *The Journal of Supercomputing, 26*(2), 167–184.
5. Garzia, F., Hussain, W., & Nurmi, J. (2009). CREMA, a coarse-grain re-configurable array with mapping adaptiveness. In *Proc. 19th international conference on field programmable logic and applications (FPL 2009)*. Prague, Czech Republic: IEEE.
6. Hussain, W., Garzia, F., & Nurmi, J. (2010). Evaluation of Radix-2 and Radix-4 FFT Processing on a Reconfigurable Platform. In *Proceedings of the 13th IEEE international symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'10)* (pp. 249–254). IEEE. ISBN 978-1-4244-6610-8.
7. Garzia, F., Hussain, W., Airoldi, R., & Nurmi, J. (2009). A reconfigurable soc tailored to software defined radio applications. In *Proc of 27th Norchip conference*. Trondheim (NO).
8. IEEE Standard for Information Technology–Telecommunications and Information Exchange Between Systems–Local and Metropolitan Area Networks–Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput. IEEE, 3 Park Avenue, NY 10016-5997, USA, Oct 2009, E-ISBN: 978-0-7381-6046-7, Print ISBN: 978-0-7381-6047-4.
9. Bo, F., & Ampadu, P. (2009). An area efficient FFT/IFFT processor for MIMO-OFDM WLAN 802.11n. *Journal of Signal Processing Systems, 56*, 59–68. ISSN: 1939-8018.
10. Lin, Y. W., & Lee, C. Y. (2007). Design of an FFT/IFFT processor for MIMO OFDM systems. *IEEE Transactions on Circuits and System I, Reg. Papers, 54*(4), 807–815.
11. Cooley, J. W., & Tukey, J. W. (1965). An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation, 19*, 297–301.
12. Hussain, W., Garzia, F., & Nurmi, J. (2010). Exploiting control management to accelerate radix-4 fft on a reconfigurable platform. In *Proc. international symposium on system-on-chip 2010* (pp. 154–157). Tampere: IEEE. ISBN: 978-1-4244-8276-4.
13. Kylliainen, J., Ahonen, T., & Nurmi, J. (2007). General-purpose embedded processor cores—the COFFEE RISC example. In J. Nurmi (Ed.), *Processor design: System-on-chip computing for ASICs and FPGAs* (Ch. 5, pp. 83–100). Kluwer Academic Publishers/Springer Publishers. ISBN-10: 1402055293, ISBN-13: 978-1-4020-5529-4.
14. Brunelli, C., Garzia, F., Giliberto, C., & Nurmi, J. (2008). A dedicated dma logic addressing a time multiplexed memory to reduce the effects of the system buss bottleneck. In *Proc. 18th International conference on field programmable logic and applications, (FPL 2008)* (pp. 487–490). Germany: Heidelberg, 8–10 September 2008.

15. Garzia, F., Brunelli, C., & Nurmi, J. (2008). A pipelined infrastructure for the distribution of the configuration bitstream in a coarse-grain reconfigurable array. In *Proceedings of the 4th international workshop on Reconfigurable Communication-Centric System-on-Chip (ReCoSoC'08)* (pp. 188–191). Univ Montpellier II, ISBN:978-84-691-3603-4.
16. Garzia, F. (2009). From run-time reconfigurable coarse-grain arrays to application-specific accelerators design, Ph.D. dissertation, Tampere University of Technology (TUT), Department of Computer Systems (DCS), Tampere, Finland, December 2009, 125 pages, TUT Publications 860, ISSN: 1459-2045, ISBN: 978-952-15-2280-2.

Waqar Hussain is working as a Research Scientist and studying as a Doctor of Technology student in Department of Computer Systems, Tampere University of Technology, Finland. His research work includes study and design of the systems that are specialized for signal processing in software defined radio applications. He is also lecturing at Department of Computer Systems, Tampere University of Technology, Finland for the last two years. Currently, he is editing Springer book on "Computation Platforms for SDR". He has a B.Sc degree in Computer Engineering from COMSATS Institute of Information Technology, Islamabad, Pakistan and an M.Sc degree in Digital and Computer Systems from Tampere University of Technology, Finland.



Fabio Garzia received his MSc degree in Electronics Engineer in March 2005 from the University of Bologna, Italy. After his graduation, he was involved in two projects carried out at ARCES Labs in Bologna. In 2006 he moved to Finland and he started his PhD studies at Tampere University of Technology. His PhD Thesis focused on the design of a coarse-grain reconfigurable accelerator at RTL level, the development of programming tools, design of SoC platforms based on this accelerator, implementation on FPGA and mapping of some applications on it. He graduated in December 2009. From January 2010 to April 2011, he received a research grant for the development of a high-level compiler for the reconfigurable accelerator.

Tapani Ahonen Ahonen is a Senior Research Fellow (Vanhempi tutkija) at Tampere University of Technology (TUT) in Tampere, Finland, where he has held various positions since 2000. Since 2004 he is co-managing a group of about 30 researchers. He is a part-time Lecturer (Nebenberuflicher Lektor) at Carinthia Tech Institute (CTI) - University of Applied Sciences in Villach, Austria since 2007. In 2009–2010 Ahonen was a Visiting Researcher (Chercheur Invité) at Université Libre de Bruxelles (ULB) in Bruxelles, Belgium. His work is focused on proof-of-concept driven computer systems design with emphasis on many-core processing environments. Ahonen has an MSc in Electrical Engineering and a PhD in Information Technology from TUT. Positions of trust recently held by Dr. Ahonen include technical board member of the EU co-funded project Cutting edge Reconfigurable ICs for Stream Processing (CRISP), Finance Chair of the 2009 IEEE Workshop on Signal Processing Systems (SiPS), editorial board member and Guest Editor of the International Journal of Embedded and Real-Time Communication Systems (IJERTCS), and Program Co-Chair of the 2010 Conference on Design and Architectures for Signal and Image Processing (DASIP), He performs reviews for various international journals and participates in program committees of many high-quality conferences on a regular basis. Ahonen has an extensive international publication record including edited books and journals, written book chapters and journal articles, invited talks in high-quality conferences, as well as full-length papers and paper abstracts in conference proceedings.

Jari Nurmi is the general chairman of the annual International Symposium on System-on-Chip (SoC) and its predecessor SoC Seminar in Tampere since 1999, and a board member of ICL-GNSS, SoC, FPL, DASIP and NORCHIP conference series. He was/is also the general chair of FPL 2005, SiPS 2009, ICL-GNSS 2011, DASIP 2011, ESWeek 2012 conferences. He was the head of the national TELESOC graduate school 2001–2005. He is the author or co-author of over 250 international papers, editor of Springer book "Processor Design: System-on-Chip Computing for ASICs and FPGAs," co-editor of Kluwer book "Interconnect-centric Design for Advanced SoC and NoC", associate editor of the "International Journal of Embedded and Real-Time Communication Systems," and is now editing Springer books on "GALILEO Positioning Technology" and "Computation Platforms for SDR." He has supervised 112 MSc theses and 13 Doctoral theses at TUT, and been the opponent or reviewer of 17 PhD theses in other universities. He is a senior member in IEEE Circuits and Systems Society, Computer Society, Signal Processing Society, Solid-State Circuits Society and Communications Society. He was a co-recipient of Nokia Educational Award 2004, the recipient of Tampere Congress Award 2005, and a co-recipient of IIDA Innovation Award 2011. He was awarded one of the Academy of Finland Research Fellow grants for 2007–2008. He is also reviewing projects for EU and project proposals for national funding agencies in Belgium, Canada, The Netherlands, Saudi-Arabia, Slovenia, Sweden, and Switzerland.

PUBLICATION 3

Energy and Power Estimation of Coarse-Grain Reconfigurable Array based Fast Fourier Transform Accelerators

Waqar Hussain, Tapani Ahonen, Roberto Airoldi, Jari Nurmi
Department of Computer Systems, Tampere University of Technology
P. O. Box 553, FIN-33101, Tampere, Finland
Email: firstname.lastname@tut.fi

***Abstract*— In recent past, we developed 4×8 and 4×16 processing element (PE) template-based Coarse-Grain Reconfigurable Arrays (CGRAs) and mapped different length and type of Fast Fourier Transform (FFT) algorithms on them. In this paper, we have considered radix-4 and radix-(2, 4) FFT accelerators which were generated from 4×8 and 4×16 PE CGRA templates respectively. We estimated their power and energy consumption while radix-4 accelerator was processing 64 and 1024 points and radix-(2, 4) accelerator was processing 64 and 128 points of FFT algorithms. The power consumption was estimated by timing simulation of postfit gate-level netlist of both of the accelerators for a Field Programmable Gate Array (FPGA) used as target platform. Based on the measurements, we have compared both of the accelerators for their power and energy consumption.**

I. Introduction

The demand by the consumer for running multiple applications on a platform is increasing every day building a greater challenge to energy resources. For example, the users want their cell phone to work on different communication standards with multiple office or home related applications showing real-time performance. On the other hand, they expect the batteries to survive longer. The limiting factor of energy resources drives us to seek other possible dimensions like run-time scalability and reconfigurability. As the area increases, the power consumption increases but if a device can be scaled at run-time then only those hardware resources can be employed that are necessary for an application under any execution time constraints. Run-time scaling of hardware is possible now as we have Run-Time Partial Reconfigurable (RTPR) Field Programmable Gate Array (FPGA) devices that also allows to replace the logic on a fraction as well as on the whole of FPGA fabric while the rest of the logic stays unaltered. In other words, there has to be application-driven scalability of hardware under power and execution time constraints. In recent past, we have been focusing on scaling the array-based accelerators to achieve execution time constraints and in this paper, we have presented the power and energy measurements of the scaled versions.

Coarse-Grain Reconfigurable Array (CGRA) is a powerful solution to industrial and academic research needs. Due to their array-based structure, they provide high throughput and parallelism. These features make them ideal for processing computationally intensive signal processing algorithms. Some examples of CGRAs are Morphosys[1], ADRES[2], PACT-XPP[3], BUTTER[4]. A general-purpose CGRA may not be a suitable candidate in many systems as it requires an area of a few million gates and their presence in the system can not be justified unless they are heavily utilized. In recent past, we developed two template-based CGRAs and published them as CREMA [5] and AVATAR [6]. CREMA and AVATAR generated Fast Fourier Transform (FFT) accelerators have successfully fulfilled the execution time constraints for IEEE-802.11a/g, IEEE-802.11n and 3GPP-LTE standards. Many other computationally intensive kernels including Wideband Code Division Multiple Access (WCDMA) cell search [7], Finite Impulse Response Filters [8], Viterbi decoders [9] and image and video processing [4], [10] were implemented on other state-of-the-art platforms.

CREMA based radix-4 FFT accelerator was able to achieve the timing constraints of IEEE-802.11a/g and 3GPP-LTE standard but due to its limited parallelism, it could not achieve the FFT timing requirements of IEEE-802.11n standard. To increase the parallelism in hardware, CREMA which was a 4×8 PE CGRA was scaled-up to 4×16 PE CGRA called AVATAR, that was able to satisfy the timing constraints of the IEEE-802.11n. The resource utilization increased as CREMA was scaled-up to AVATAR, however the power consumption was never measured. It is important to know the cost that has to be paid in terms of resource expense and power consumption if scaling-up is required to meet performance goals.

This paper is organized as follows. Section II discusses CREMA and AVATAR based processing model. Section III highlights the FFT accelerators generated and algorithms mapped on them. Section IV discusses the power and energy measurements in accelerating FFTs of different length and type. Section V presents the conclusions.

II. CGRA based Processing Model

CGRAs can be found as stand-alone systems and working as an accelerator with a general-purpose processor. CREMA works as a coprocessor with a Reduced Instruction Set Computing (RISC) core called COFFEE [11]. CREMA as a template device generates special-purpose accelerator which acts as a coprocessor based on the user input. In this system COFFEE can perform general-purpose processing while

CGRA Type	Size (PEs) (rows × cols)	I/O-buffer Size N × (Mux Size)	Local Memory X-bit, rows × cols
CREMA	4 × 8	16 × (16 × 1)	32-bit, 128 × 16
AVATAR	4 × 16	32 × (32 × 1)	32-bit, 128 × 32

TABLE I

ARCHITECTURAL COMPARISON BETWEEN CREMA AND AVATAR, (N SHOWS THE TOTAL NUMBER OF MULTIPLEXERS)

CREMA generated accelerators can perform the computationally intensive task. The basic unit of structure of CREMA is an Arithmetic and Logic Unit (ALU) type of Processing Element (PE). Each PE has two inputs and can perform both 32-bit integer and IEEE-754 format floating-point operations. The architecture and functionality of PE is explained in [5]. An operator or a set of operators can be instantiated by the user at compile time inside a PE. At run-time, based on the algorithm flow, the operation to be performed can be selected at any desired cycle of execution. Each PE connects with the neighbouring PE in a point-to-point fashion for the exchange of data. The interconnections of every PE can switch to any neighbouring PE at different clock cycles. The user can select point-to-point switching of interconnections at compile time to be made at run-time for in order processing of data and according to the flow of algorithm. The user selection of operators to be instantiated and the interconnection between PEs has to be based on the mapping of a particular algorithm. At any clock cycle, the operation to be performed by the PEs and the pattern of interconnections among all the PEs is called a *context*. The context selection is based on configuration words stored in configuration memories. The configuration words are generated by the software tool on compilation. At the system start-up time, the configuration words are loaded in the configuration memories with the help of a Direct Memory Access (DMA) [13] device and using a pipelined infrastructure [15] which injects the configuration words therefore reducing the overhead to distribute the words over the whole array. CREMA has two local memories and the data to be processed is loaded in the local memories with the help of the DMA. The control flow of the program is written in C where COFFEE controls the cycle accurate processing of CREMA by writing control words in the control registers of CREMA. The execution flow to be written for CREMA based system in explained in [14]. Between the local memories and the processing array, there are I/O-buffers that provide interleaving to the data stored in the local memories before and after it is passed to the processing array. The I/O-buffers contain multiplexers that allow each input in the first row of processing array to receive data from any of the memory banks in a local memory. The CREMA based processing model is shown in Fig. 1.

AVATAR based system is exactly the same as of CREMA shown in Fig. 1 with the only difference that it is a scaled-up version. It has the same structure of the PE as of CREMA and the differences can be observed from Table I.

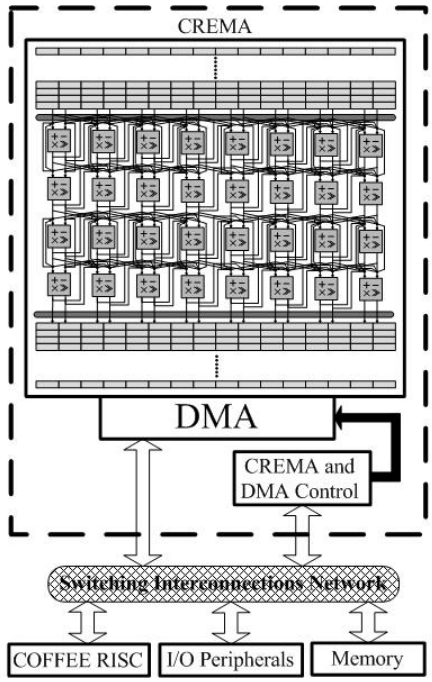


Fig. 1. System Architecture and Scaling in CREMA

III. CGRA BASED FFT ACCELERATORS

FFT is the most suitable technique to process discrete Fourier Transform in digital systems and mathematically can be defined as

$$X[k] = \sum_{n=0}^{N-1} x[k] W_N^{nk} \tag{1}$$

where n is any sample among N samples to be processed and $W_N^{nk} = exp(-j2\pi nk/N)$ is a twiddle factor. Radix-$2^m, m \in Z^+$ FFT algorithm were presented for the first time by Cooley and Tukey [16]. The basic unit of any FFT structure is called a 'butterfly'. The mathematical representation of the butterfly structure for $m = 2, 4$ is explained in [14].

Using CREMA, a radix-4 FFT accelerator was produced and 64-point FFT was processed over it [14]. Later radix-4 1024-point FFT algorithm was mapped on it [7]. The radix-4 FFT accelerator generated using CREMA required three different contexts to map a radix-4 butterfly mainly because of its limited dimension. To process any stage of radix-4 FFT algorithm, the three execution contexts containing a radix-4 butterfly were enabled in a sequence. The main idea to scale-up CREMA to AVATAR was to use only one context to map a radix-4 butterfly and therefore reduce the overall execution time. AVATAR required more hardware resources than CREMA but had an additional advantage of ease in application mapping [17]. Radix-(2, 4) FFT accelerator could

FFT Accelerator	FFT Length	Static Power	Dynamic Power	I/O Power	Total Power
CREMA	64	435.66 mW	259.88 mW	46.87 mW	742.41 mW
radix-4	1024	435.90 mW	264.15 mW	57.58 mW	757.63 mW
AVATAR	64	466.76 mW	472.73 mW	46.41 mW	985.90 mW
radix-(2, 4)	128	466.76 mW	472.73 mW	46.41 mW	985.90 mW

TABLE II

POWER CONSUMPTION BY CREMA BASED RADIX-4 AND AVATAR BASED RADIX-(2, 4) FFT ACCELERATORS WHILE EXECUTING DIFFERENT LENGTHS OF FFT ALGORITHMS

FFT Accelerator	FFT Length	Execution Time	Total Power	Energy (Joules)
CREMA	64	2,736 ns	742.41 mW	2.03 μJ
radix-4	1024	121,344 ns,	757.63 mW	91.933 μJ
AVATAR	64	2,130 ns	985.90 mW	2.09 μJ
radix-(2, 4)	128	3,420 ns	985.90 mW	3.37 μJ

TABLE III

ENERGY CONSUMPTION BY CREMA BASED RADIX-4 AND AVATAR BASED RADIX-(2, 4) FFT ACCELERATORS

process both radix-2 and 4 FFT algorithm. The main purpose of implementing a radix-(2, 4) FFT accelerator was to process 128 points in four stages which would otherwise will need seven stages by a radix-2 FFT accelerator.

IV. POWER AND ENERGY MEASUREMENTS

The static power consumption increases as the logic increases on the FPGA device which means that the AVATAR generated accelerators will need more static power than the ones generated by CREMA in they are deployed on a FPGA. The dynamic power consumption will also be more for AVATAR generated accelerators as there will be more hardware available to use for parallel data processing and therefore increasing the switching activity at any particular time instant. Other than the power consumption, we need to consider energy efficiency. A CGRA with a larger size can have higher power consumption but can be more energy efficient than a CGRA with a smaller size if a large data set is supposed to be processed. For CREMA or AVATAR generated accelerators, we can consider four types of power consumptions which are as follows.

1) Power-up
2) Configuration Power
3) Static Power
4) Dynamic Power

At this time we are not considering the power required for the system start-up as we assume that the system is already powered-up. Additionally configuration power consumption has not be been considered as we also assume that the system is ready to process data and all the configuration words are already stored in their respective configuration memories. Static power consumption is the minimum power to keep the accelerator in power-up state and the dynamic power power consumption is based on its switching activity.

We synthesized both of the accelerators using Quartus II for EP4SGX70HF35C2 device and obtained a working frequency of 169.26 MHz and 170.65 MHz for 4×8 PE radix-4 and 4×16 PE radix-(2, 4) FFT accelerators at 0°C and 900mV. We then generated the postfit gate-level netlist (PGN) for working condition temperature of 0°C of the system enclosed in dashed border shown in Fig. 1. The system outside the dashed border served as test-bench when PGN was simulated for timing in Modelsim. Using modelsim, the Value Change Dump (VCD) file was written that contains signal transition information during the execution of FFT algorithms. The VCD file was then supplied to power analyzer tool of Quartus II for power measurements. The power measurement using the timing simulation of PGN is the most accurate method to estimate power consumption. An operating frequency of 166.66 MHz was used for both of the accelerators for generating the VCD file. The power measurement tool showed a 'HIGH' confidence level in all the estimations performed. Table II shows the power measurement details for radix-4 and radix-(2, 4) FFT accelerators. From the table, we can observe that radix-4 FFT accelerator requires 4.27mW more dynamic power consumption while executing 1024-point FFT than processing 64-point FFT. The estimation about the dynamic power consumption of radix-4 FFT accelerator can be considered more accurate in case of processing 1024-point FFT as there are more points available to be processed than processing a 64-point FFT. While processing 64-point FFT, AVATAR based radix-(2, 4) accelerator required $1.8X$ dynamic power consumption compared to the CREMA based radix-4 FFT accelerator. Radix-(2, 4) FFT accelerator requires the same amount of static and dynamic power consumption for executing 64 and 128-point FFT algorithm. This is because radix-4 FFT processing requires all the 4×16 PEs at a time to be used to map a radix-4 FFT butterfly which is required to process both 64 and 128-point FFT.

Table III shows the energy consumption by CREMA and AVATAR based FFT accelerators. From the table, it can be observed that AVATAR consumes almost the same energy as CREMA while being $1.3X$ faster. The power consumption is the same by an FFT accelerator for executing different lengths of FFT while energy requirement increases as the length of FFT increases. For example, in Table III, AVATAR based

Resources	radix-(2, 4)	radix-4	Cost Ratio
Combinational ALUTs	33,765	12,996	2.6X
Dedicated logic registers	13,828	8,123	1.7X
DSP block 18-bit elements	112	48	2.3X

TABLE IV
RESOURCE UTILIZATION SUMMARY OF RADIX-(2, 4) AND RADIX-4 FFT ACCELERATORS ON EP4SGX70HF35C2 FPGA DEVICE

radix-(2, 4) FFT accelerator requires $1.6X$ energy for processing 128-point FFT than required for 64-point. Similarly, CREMA based radix-4 FFT accelerator requires $45X$ energy for processing 1024-point FFT compared to a 64-point one.

In comparison to power and energy consumption, it is interesting to know the resource utilization of both of the accelerators on the FPGA device used which is shown in Table IV. The DSP resources consume almost negligible power as compared to the static and dynamic power consumption of the reconfigurable fabric of the FPGA. If we make a rough estimate of the collective resource utilization by only considering the cost ratio related to combinational ALUTs and dedicated logic registers, we can assume that radix-(2, 4) FFT accelerator requires $2X$ resources and there has not been any significant increase in static power consumption in comparison to radix-4 FFT accelerator. If we also consider the dynamic power consumption then in total, radix-(2, 4) FFT accelerator needs only $1.3X$ more power compared to the radix-4 FFT accelerator.

V. CONCLUSION

Scaling-up a Coarse-Grain Reconfigurable Array (CGRA) reduces the execution time of a kernel on the cost of increased use of resources on an FPGA. As the number of resources increases, static and dynamic power consumption increases but a large-scale CGRA can be more energy efficient than a small-scale CGRA. We measured the energy consumption of AVATAR based radix-(2, 4) Fast Fourier Transform (FFT) accelerator which is a 4×16 processing element (PE) CGRA and CREMA based radix-4 FFT accelerator which is a 4×8 PE CGRA. It was found that for processing 64-point FFT, AVATAR based radix-(2, 4) FFT accelerator is quite exactly as energy efficient as CREMA based radix-4 FFT accelerator. Radix-(2, 4) FFT accelerator requires twice as many resources as radix-4 FFT accelerator but there is no significant difference in the static power consumption but considering the dynamic power consumption also then in total, radix-(2, 4) FFT accelerator needs only $1.3X$ more power than the radix-4 FFT accelerator.

REFERENCES

[1] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. C. Filho, "Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications". IEEE Trans. Computers, vol. 49, no. 5, pp. 465-481, 2000.

[2] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix", Field-Programmable Logic and Applications, vol. 2778, pp. 61-70, September 2003, ISBN 978-3-540-40822-2.

[3] V. Baumgarte, G. Ehlers, F. May, A. Nuckel, M. Vorbach, and M. Weinhardt, "PACT XPP-A Self-Reconfigurable Data Processing Architecture", The Journal of Supercomputing, vol. 26, no. 2, pp. 167-184, September 2003.

[4] C. Brunelli, F. Garzia, and J. Nurmi, "A Coarse-Grain Reconfigurable Architecture for Multimedia Applications Featuring Subword Computation Capabilities", in Journal of Real-Time Image Processing, Springer-Verlag, 2008, 3 (1-2): 21-32. doi:10.1007/s11554-008-0071-3.

[5] F. Garzia, W. Hussain and J. Nurmi, "CREMA, A Coarse-Grain Reconfigurable Array with Mapping Adaptiveness", in Proc. 19th International Conference on Field Programmable Logic and Applications (FPL 2009). Prague, Czech Republic: IEEE, September 2009.

[6] W. Hussain, F. Garzia, T. Ahonen and J. Nurmi, "Designing Fast Fourier Transform Accelerators for Orthogonal Frequency-Division Multiplexing Systems", in the Springer's Journal of Signal Processing Systems, December 2010.

[7] F. Garzia, W. Hussain, R. Airoldi, J. Nurmi, "A Reconfigurable SoC tailored to Software Defined Radio Applications", in Proc of 27th Norchip Conference, Trondheim (NO), 2009.

[8] P. Kunjan, C. Bleakley, "Systolic Algorithm Mapping for Coarse Grained Reconfigurable Array Architectures", Reconfigurable Computing: Architectures, Tools and Applications, Lecture Notes in Computer Science, 2010, Springer Berlin Heidelberg, pp 351-357, Vol. 5992, DOI: 10.1007/978-3-642-12133-3_33.

[9] Y. Kishimoto, S. Haruyama, H. Amano, "Design and Implementation of Adaptive Viterbi Decoder for Using A Dynamic Reconfigurable Processor" in Proc. Reconfigurable Computing and FPGAs, 2008. ReConFig '08, pp 247-252, doi=10.1109/ReConFig.2008.39, ISBN: 978-1-4244-3748-1.

[10] Chia-Cheng Lo, Shang-Ta Tsai, Ming-Der Shieh; , "A reconfigurable architecture for entropy decoding and IDCT in H.264," VLSI Design, Automation and Test, 2009. VLSI-DAT '09. International Symposium on , vol., no., pp.279-282, 28-30 April 2009, doi: 10.1109/VDAT.2009.5158149, ISBN: 978-1-4244-2781-9.

[11] J. Kylliainen, T. Ahonen, and J. Nurmi, "General-purpose embedded processor cores - the COFFEE RISC example", In Processor Design: System-on-Chip Computing for ASICs and FPGAs, J. Nurmi, Ed. Springer Publishers, June 2007, ch. 5, pp. 83-100, ISBN-10: 1402055293, ISBN-13: 978-1-4020-5529-4.

[12] W. Hussain, F. Garzia and J. Nurmi, "Exploiting Control Management to Accelerate Radix-4 FFT on a Reconfigurable Platform", in Proc. International Symposium on System-on-Chip 2010. Tampere, Finland: IEEE, pp. 154-157, September 2010, ISBN: 978-1-4244-8276-4.

[13] C. Brunelli, F. Garzia, C. Giliberto and J. Nurmi, "A Dedicated DMA Logic Addressing a Time Multiplexed Memory to Reduce the Effects of the System Buss Bottleneck", in Proc. 18th International Conference on Field Programmable Logic and Applications, (FPL 2008), Heidelberg, Germany, 8-10 September 2008, pp. 487-490.

[14] W. Hussain, F. Garzia, and J. Nurmi, "Evaluation of Radix-2 and Radix-4 FFT Processing on a Reconfigurable Platform," in Proceedings of the 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'10). IEEE, pp. 249-254, April 2010, ISBN 978-1-4244-6610-8.

[15] F. Garzia, C. Brunelli and J. Nurmi, "A pipelined infrastructure for the distribution of the configuration bitstream in a coarse-grain reconfigurable array", in Proceedings of the 4th International Workshop on Reconfigurable Communication-centric System-on-Chip (ReCoSoC'08). Univ Montpellier II, July 2008, pp. 188-191, ISBN:978-84-691-3603-4.

[16] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series", Math. Comp., vol. 19, pp, 297-301, April 1965.

[17] W. Hussain, T. Ahonen F. Garzia and J. Nurmi, "Application-Driven Dimensioning of a Coarse-Grain Reconfigurable Array", in Proc. NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2011), San Diego, California, USA.

PUBLICATION 4

Effects of Scaling a Coarse-Grain Reconfigurable Array on Power and Energy Consumption

Waqar Hussain, Tapani Ahonen, Jari Nurmi
Department of Computer Systems, Tampere University of Technology
P. O. Box 553, FIN-33101, Tampere, Finland
Email: firstname.lastname@tut.fi

***Abstract*— In recent past, we scaled a 4×8 processing element (PE) template-based Coarse-Grain Reconfigurable Array (CGRA) to a 4×4, 4×16 and 4×32 PE CGRA and generated matrix-vector multiplication (MVM) accelerators from each one of them. Furthermore, on each of the accelerators, MVM kernels of order $N = 4, 8, 16, 32$ were mapped. In this paper, we have estimated the power and energy consumption by generating the postfit gate-level netlist of each accelerator for a Field Programmable Gate Array as target platform. Based on our measurements, we have studied the effects of scalability of a CGRA on power and energy consumption.**

I. Introduction

Embedded systems are required in almost every field of science and engineering. The requirements of the users may demand different types of embedded systems ranging from single processor system to multiprocessors. They also may vary from single accelerator system to multiple accelerator systems. One of the important system types is processor/ coprocessor model in which the general purpose processing is performed by the processor and the coprocessor accelerates the computationally intensive tasks. One of the important class of accelerators is a Coarse-Grain Reconfigurable Array (CGRA) which by its structure offers high hardware level parallelism and throughput. A number of CGRAs have been developed so far, for example ADRES [2], Morphosys [1], PACT-XPP [3] and BUTTER [4]. The general-purpose CGRA required an area of few million gates and their presence in the systems becomes expensive unless they are extensively used. To avoid this problem, we developed template-based CGRAs like CREMA [5] and [6]. Using template-based CGRAs, the user was able to generate special purpose accelerators on a specified mapping which reduces resource utilization. Many computationally intensive kernels were mapped on CGRAs in past, for example Wideband Code Division Multiple Access (WCDMA) cell search [7], image and video processing [4], [9] and Viterbi decoders [8]. Execution time constraints for processing Fast Fourier Transform (FFT) were also achieved for many wireless standards like IEEE-802.11a/g, 3GPP-LTE and IEEE-802.11n [7], [6].

Scaling the hardware is important from execution and resource utilization point-of-view. Especially in mobile devices, we may not need a high processing bandwidth at all times. For example, our mobile device may not need resources to carry out processing for a stream generated in multiple-input multiple-output Orthogonal Frequency-Division Multiplexing (OFDM) environment if the user ports it to a single-input single-output OFDM environment. Run-Time Partial Reconfigurable (RTPR) Field Programmable Gate Arrays (FPGA) allow reconfiguration of a fraction of the fabric at run-time. Using the RTPR FPGA, we can increase or decrease the resources on FPGA at run-time and therefore save the resources and power consumption. In this context, we need to know how scaling will effect the execution time, resource utilization, power consumption and also the application developement time [15].

In the next section, we will discuss the processing model based on a scalable CGRA template. In Section III, we will discuss the matrix-vector multiplication (MVM) accelerators generated from the scalable CGRA templates. In Section IV, we will focus on power and energy estimation of the generated accelerators. In the next section, we will discuss the scalability analysis based on the findings in the previous sections. Finally, we will present conclusions.

II. SCREMA based Processing Model

Structure of SCREMA template is similar to CREMA template except that SCREMA can be scaled to different CGRA templates which can then generate accelerators of different sizes. SCREMA is written in VHDL and can be scaled to different sizes of templates by changing just a few parameters in the definition package VHDL file. After changing the definition package file, a parameter package VHDL file can be used which is generated on user specifications by a graphical tool. The parameter package file contains the information to craft the template to an accelerator on compilation.

The structural unit of SCREMA is a processing element (PE) like in CREMA. Each PE has two inputs (a, b) and two outputs (A, B). Output A carries the result based on the operand received by (a, b) and the output B receives the operand b so that it can transfered to other PE as required. Each PE can perform 32-bit integer and floating-point operations in IEEE-754 format. The internal structure of the PE is well defined in [5] and also the way it exchanges data with the neighbouring PEs in point-to-point fashion.

The generated accelerator is equiped with two local memories and the data to be processed is loaded in these local memories with the help of a Direct Memory Access (DMA) device. The DMA transfers the data from the main memory of the system to the local memories in a specific

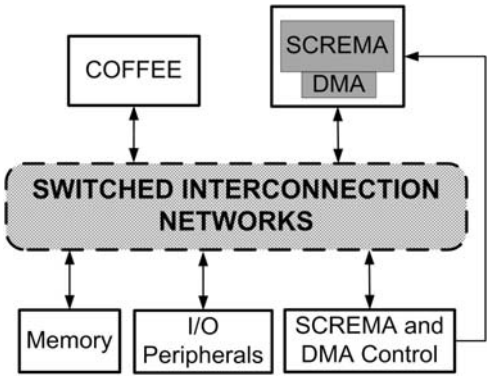


Fig. 1. SCREMA based Embedded Processing Model

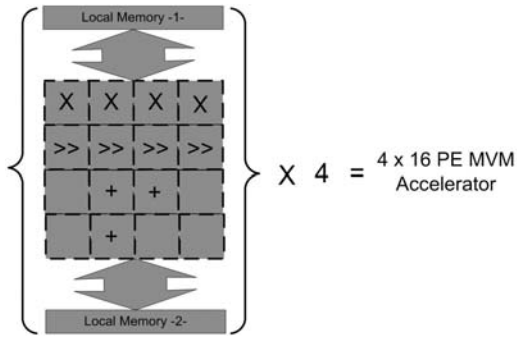


Fig. 2. 4×4 PE MVM accelerator generated by 4×4 PE SCREMA shown in braces. Four of such accelerators working in parallel will be equal to MVM accelerator generated by a 4×16 PE CGRA template.

pattern introduced by the user. Before the system start-up, the configuration data is loaded in the configuration memory of each PE. The configuration words are injected in the array using a pipelined infrastructure [14]. These words are used to select an operation to be performed by a PE and also the interconnections among PEs. The operations to be performed and the pattern of interconnection among all PEs is called a context. Different contexts can be designed by the user at compile-time and can be enabled at run-time based on the flow of an algorithm. The application mapping on SCREMA versions and the execution flow to be written in C is similar to the one explained in [13] for designing FFT accelerators.

SCREMA generated accelerators work as coprocessors of COFFEE RISC [10]. The program is written in C and compiled for COFFEE RISC which controls the processing of the accelerator in a polling mechanism. COFFEE writes the control words in the control registers of the generated accelerator which in return performs cycle-accurate processing. COFFEE and SCREMA generated accelerator interact with each other by a network of switched interconnections that provide dedicated connections for faster communication. Fig. 1 shows the overall system.

III. MVM Accelerators

In this paper, we concentrate on Matrix-Vector Multiplication (MVM) applications as they are largely used in different science and engineering applications. We can define MVM process mathematically by considering a matrix $a = [a_{i,j}]$ and vector $\overrightarrow{b} = [b_i]$ of N^{th}-order and they are supposed to be multiplied to produce a product vector $\overrightarrow{p} = [p_i]$. Then the multiplication process can be defined as

$$[p_i] = \sum_{j=1}^{N} [a_{j,i}] \times [b_j] \tag{1}$$

where i, j = 1, 2, 3,...,N.

MVM accelerators were generated from CGRA-templates of sizes 4×4, 4×8, 4×16 and 4×32 PEs which were developed in recent past. The basic MVM accelerator was generated from 4×4 PE CGRA-template and its structure is shown in Fig. 2. The first row consist of multipliers, the second performs the shift operation to avoid any possible overflows at the later stages. The third and fourth row perform the addition operations required for MVM process. The MVM accelerator generated by 4×8 PE CGRA template is similar to two 4×4 PE MVM accelerators working in parallel. Similarly 4×16 PE MVM accelerator contains four of such accelerators and 4×32 will have eight of those working in parallel. As each of the PEs has two inputs, the first input can receive a matrix element and the second input will receive the related vector element to be multiplied. The matrix and vector data to be multiplied is distributed over the MVM accelerator's local memory in a specific pattern. The data placement pattern in the local memories of the MVM accelerators is same for all the accelerators to carry out justified comparisons. The execution time shown in Table III does not include the time required by the DMA to load the data in the local memories. It is the time, when the data start processing over the array until it completes the processing and stores the data in the local memory as we are only interested in the performance measurement of the array.

The MVM accelerators were synthesized for Altera's Stratix-IV FPGA device (EP4SGX70HF35C2) and the resource utilization can be oberved from Table I for each of the accelerators. As the size of the MVM accelerator increases, the resource utilization increases almost proportionally. As all the accelerators have 32-bit processing so the multiplication of two 32-bit numbers will result in a 64-bit number. To fit a 64-bit number, four 18-bit DSP elements are required. As a 4×4 MVM accelerator shown in braces in Fig. 2 has four multipliers in the first row, it will need 16 of 18-bit DSP elements on the FPGA as shown in Table I.

IV. Power and Energy Estimation

To estimate the power consumption, we generated the postfit gate-level netlist for each of the MVM accelerators and performed the timing simulation which is the most accurate method for estimating the power conumption. Table II shows

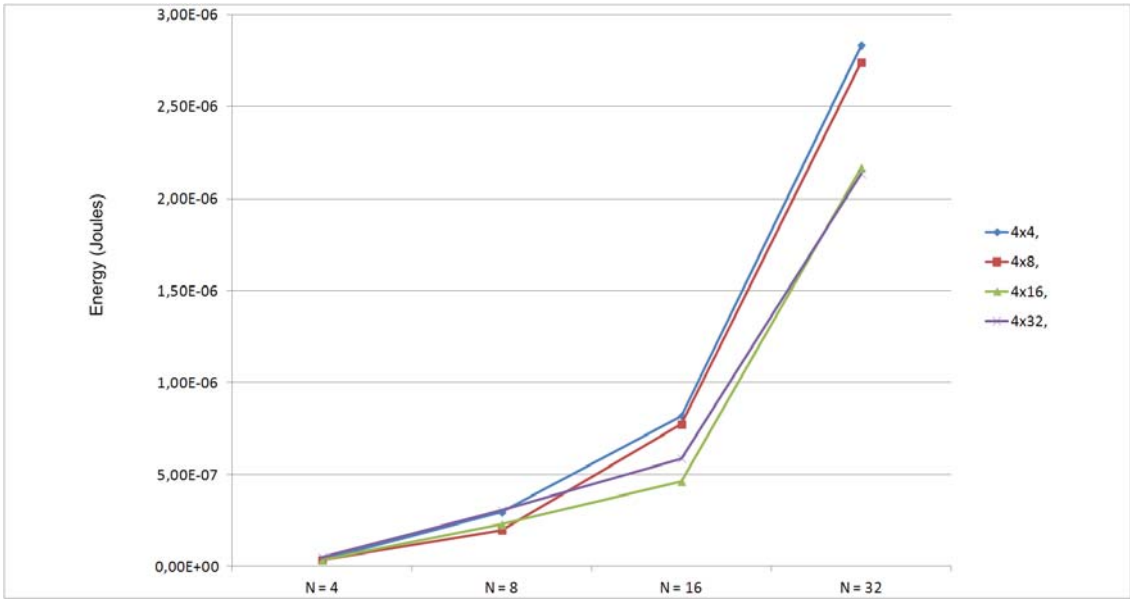


Fig. 3. Curves showing 4×16 PE MVM accelerator as the most energy efficient relatively. X-axis shows the order of MVM process and Y-axis shows the energy consumption in Joules.

MVM Accelerator Size	Comb ALUTs	Logic Registers	DSPs
4×4 PE	$2,566$	$2,766$	16
4×8 PE	$4,805$	$3,820$	32
4×16 PE	$8,259$	$6,784$	64
4×32 PE	$15,522$	$12,057$	128

TABLE I
RESOURCE UTILIZATION BY MVM ACCELERATORS ON STRATIX-IV (EP4SGX70HF35C2) DEVICE

the static and dynamic power consumption of the four MVM accelerators at 85°C and 900mV. At these conditions, the four accelerators achieved the operating frequency between 169-172 MHz. From the table, it can be observed that there is no significant change in the static power consumption. A large offset approximately equal to 427 mW is visible due to the unused portion of the FPGA chip. If this offset is subtracted, we can observe that the static power consumption increases almost by a factor of two as the size of the accelerator doubles. Significant difference in dynamic power consumption can be observed as it depends on the number of signals having switching activity at a particular time instant. As the size of the MVM accelerator increases, the dynamic power consumption increases but the processing time for MVM decreases. Depending on the requirements for resource utilization, execution time and power consumption, it may be desireable to make a choice among these four MVM accelerators. An optimal choice can be based on energy consumption which is the product of total power consumption and the execution time but in that case a comparison should also be made to the resouce utilization of the choosen MVM accelerator. The energy consumption is estimated only for the data processing by the accelerators and not for the process of transferring the data from the main memory of the system to the local memories of the accelerators.

V. SCALABILITY EFFECTS

The digital hardware due to its binary characteristics will tend to scale by a factor of 2^m where $m \in Z^+$. The application to be mapped should be built on the same scale to have best-fit compatibility. This is why we have chosen the CGRA sizes of 4×2^n and the matrix and vector of sizes $2^n \times 2^n$ and 2^n respectively, where $n = \{2, 3, 4, 5\}$.

From the curves shown in Fig. 3, we can easily observe that 4×8 and 4×16 PE MVM accelerators are relatively the most energy efficient for processing MVM of $N = 4, 8$ and $N = 16, 32$ relatively. The 4×4 PE MVM accelerator is the least energy efficient relatively otherwise while processing MVM of $N = 4$. Its relatively low dynamic power consumption was not enough to overcome the number of clock cycles required to process MVM kernels. The 4×32 PE MVM accelerator is not relatively as energy efficient as others than it has shown to be slightly more energy efficiency while processing MVM of order $N = 32$ then 4×16 PE MVM accelerator. From the curves in Fig. 3, it can be observed that an MVM accelerator is the most energy efficient when its order is matching with the order of the MVM process. As the experimental data set is kept fixed upto only MVM of order $N = 32$, we can speculate

MVM Accelerator	Static Power	Dynamic Power	I/O Power	Total Power
4×4 PE	428.48 mW	127.27 mW	58.07 mW	613.82 mW
4×8 PE	430.41 mW	208.63 mW	56.53 mW	695.57 mW
4×16 PE	435.11 mW	387.21 mW	56.95 mW	879.27 mW
4×32 PE	448.51 mW	728.25 mW	51.04 mW	1227.80 mW

TABLE II

POWER CONSUMPTION BY MVM ACCELERATORS OF DIFFERENT SIZES

MVM Accelerator	MVM Order	Execution Time	Total Power	Energy
4×4	4	57 ns	613.82mW	0.034 μJ
	8	479 ns		0.29 μJ
	16	1334 ns		0.818 μJ
	32	4617 ns		2.83 μJ
4×8	4	52 ns	695.57 mW	0.036 μJ
	8	285 ns		0.19 μJ
	16	1111 ns		0.773 μJ
	32	3940 ns		2.74 μJ
4×16	4	41 ns	879.27 mW	0.036 μJ
	8	259 ns		0.228 μJ
	16	525 ns		0.462 μJ
	32	2466 ns		2.17 μJ
4×32	4	41 ns	1227.80 mW	0.05 μJ
	8	248 ns		0.304 μJ
	16	478 ns		0.587 μJ
	32	1740 ns		2.14 μJ

TABLE III

ENERGY CONSUMPTION BY DIFFERENT SIZE OF ACCELERATORS WHILE EXECUTING DIFFERENT ORDERS OF MVM

that 4×32 PE MVM accelerator will be significantly more energy efficient than all others while processing $N \geq 32$.

An important dimension of comparison will be to evaluate energy consumption with the resource utilization of the MVM accelerators. A rough estimate about the resource utilization can be the average of the number of ALUTs and logic registers. We can ignore the DSP elements as their power consumption is almost negligible compared to the reconfigurable fabric of the FPGA. From the data in Table I, we calculated the averages and found that the cost in terms of resources is almost $1.7X$ which is required to scale-up an MVM accelerator to the next level. As the size of MVM accelerator is scaled-up, the dynamic power consumption increases by a factor of almost $1.8X$.

From the curves shown in Fig. 3, the investment of $1.7X$ resources was generally the most cost effective for scaling-up 4×8 PE MVM accelerator to 4×16 PE MVM accelerator while processing $N = 16, 32$ MVM and even for processing $N = 4, 8$, it is very close to 4×8 PE MVM accelerator curve.

VI. CONCLUSION

We estimated the power and energy consumption of 4×4, 4×8, 4×16 and 4×32 processing element (PE) matrix-vector multiplication (MVM) accelerators. We found that scaling-up MVM accelerators results in a constant increase of dynamic power consumption of almost $1.8X$ and requires around $1.7X$ of more resources when doubling the accelerator array width. The 4×8 and 4×16 PE MVM accelerators are relatively the most energy efficient while processing MVM of orders $N = 4, 8$ and $N = 16, 32$, respectively. The 4×32 PE MVM accelerator is not as energy efficient as any of the other MVM accelerators.

REFERENCES

[1] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. C. Filho, "Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications". IEEE Trans. Computers, vol. 49, no. 5, pp. 465-481, 2000.

[2] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix", Field-Programmable Logic and Applications, vol. 2778, pp. 61-70, September 2003, ISBN 978-3-540-40822-2.

[3] V. Baumgarte, G. Ehlers, F. May, A. Nuckel, M. Vorbach, and M. Weinhardt, "PACT XPP-A Self-Reconfigurable Data Processing Architecture", The Journal of Supercomputing, vol. 26, no. 2, pp. 167-184, September 2003.

[4] C. Brunelli, F. Garzia, and J. Nurmi, "A Coarse-Grain Reconfigurable Architecture for Multimedia Applications Featuring Subword Computation Capabilities", in Journal of Real-Time Image Processing, Springer-Verlag, 2008, 3 (1-2): 21-32. doi:10.1007/s11554-008-0071-3.

[5] F. Garzia, W. Hussain and J. Nurmi, "CREMA, A Coarse-Grain Reconfigurable Array with Mapping Adaptiveness", in Proc. 19th International Conference on Field Programmable Logic and Applications (FPL 2009). Prague, Czech Republic: IEEE, September 2009.

[6] W. Hussain, F. Garzia, T. Ahonen and J. Nurmi, "Designing Fast Fourier Transform Accelerators for Orthogonal Frequency-Division Multiplexing Systems", in the Springer's Journal of Signal Processing Systems, December 2010.

[7] F. Garzia, W. Hussain, R. Airoldi, J. Nurmi, "A Reconfigurable SoC tailored to Software Defined Radio Applications", in Proc of 27th Norchip Conference, Trondheim (NO), 2009.

[8] Y. Kishimoto, S. Haruyama, H. Amano, "Design and Implementation of Adaptive Viterbi Decoder for Using A Dynamic Reconfigurable Processor" in Proc. Reconfigurable Computing and FPGAs, 2008. ReConFig '08, pp 247-252, doi=10.1109/ReConFig.2008.39, ISBN: 978-1-4244-3748-1.

[9] Chia-Cheng Lo, Shang-Ta Tsai, Ming-Der Shieh; , "A reconfigurable architecture for entropy decoding and IDCT in H.264", VLSI Design, Automation and Test, 2009. VLSI-DAT '09. International

Symposium on , vol., no., pp.279-282, 28-30 April 2009, doi: 10.1109/VDAT.2009.5158149, ISBN: 978-1-4244-2781-9.

[10] J. Kylliainen, T. Ahonen, and J. Nurmi, "General-purpose embedded processor cores - the COFFEE RISC example", In Processor Design: System-on-Chip Computing for ASICs and FPGAs, J. Nurmi, Ed. Springer Publishers, June 2007, ch. 5, pp. 83-100, ISBN-10: 1402055293, ISBN-13: 978-1-4020-5529-4.

[11] W. Hussain, F. Garzia and J. Nurmi, "Exploiting Control Management to Accelerate Radix-4 FFT on a Reconfigurable Platform", in Proc. International Symposium on System-on-Chip 2010. Tampere, Finland: IEEE, pp. 154-157, September 2010, ISBN: 978-1-4244-8276-4.

[12] C. Brunelli, F. Garzia, C. Giliberto and J. Nurmi, "A Dedicated DMA Logic Addressing a Time Multiplexed Memory to Reduce the Effects of the System Buss Bottleneck", in Proc. 18th International Conference on Field Programmable Logic and Applications, (FPL 2008), Heidelberg, Germany, 8-10 September 2008, pp. 487-490.

[13] W. Hussain, F. Garzia, and J. Nurmi, "Evaluation of Radix-2 and Radix-4 FFT Processing on a Reconfigurable Platform," in Proceedings of the 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'10). IEEE, pp. 249-254, April 2010, ISBN 978-1-4244-6610-8.

[14] F. Garzia, C. Brunelli and J. Nurmi, "A pipelined infrastructure for the distribution of the configuration bitstream in a coarse-grain reconfigurable array", in Proceedings of the 4th International Workshop on Reconfigurable Communication-centric System-on-Chip (ReCoSoC'08). Univ Montpellier II, July 2008, pp. 188-191, ISBN:978-84-691-3603-4.

[15] W. Hussain, T. Ahonen F. Garzia and J. Nurmi, "Application-Driven Dimensioning of a Coarse-Grain Reconfigurable Array", in Proc. NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2011), San Diego, California, USA.

PUBLICATION 5

A Reconfigurable Application-specific Instruction-set Processor for Fast Fourier Transform Processing

Waqar Hussain†, Xiaolin Chen∗, Gerd Ascheid∗, Jari Nurmi†
† Department of Electronics and Communications Engineering, Tampere University of Technology,
FI-33101, Tampere, Finland Email: firstname.lastname@tut.fi
∗ Institute for Communication Technologies and Embedded Systems,
Rheinisch-Westfaelische Technische Hochschule, DE-52070,
Aachen, Germany, Email: firstname.lastname@ice.rwth-aachen.de

***Abstract*— In this paper, we have presented a Reconfigurable Application-specific Instruction-set Processor (rASIP) that processes mixed-radix(2, 4) 64 and 128-point Fast Fourier Transform (FFT) algorithms while satisfying the partial execution-time requirements of IEEE-802.11n standard. The rASIP was designed by integrating a template-based Coarse-Grain Reconfigurable Array (CGRA) in the datapath of a simple Reduced Instruction-Set Computing (RISC) Processor. The instruction set of the RISC processor was extended to add special instructions to enable cycle-accurate processing by the CGRA. The rASIP is synthesized for Field Programmable Gate Arrays for the measurement of resource utilization and execution time. The postfit gate-level netlist of rASIP was simulated to estimate the power and energy consumption. Based on our measurements and estimates, we have studied the advantages of using rASIP in comparison with other systems.**

I. Introduction

Reconfigurable Application-specific Instruction-set Processor (rASIP) is one of the most optimal solutions from cost and performance point of view as it combines the features from general-purpose processors and application-specific accelerators. The add-on of reconfigurability allows to target multiple computationally-intensive algorithms. In recent past many ASIPs were developed specialized for image and video processing [1], Viterbi or LDPC channel coding/decoding ([2], [3]) and Fast Fourier Transform processing ([4], [5]). Later, the idea of adding dynamically reconfigurable circuitry in the datapath of the processor got popular and rASIPs were made for Software-Defined Radio (SDR) applications [6] and also for basic Digital Signal Processing (DSP) applications [7]. To control the reconfigurable part of rASIPs, special instructions were added in the instruction-set of the processor and also necessary additional hardware for interfacing. rASIPs require less area/resource utilization than traditional processor/coprocessor based systems because the reconfigurable part is closely integrated with the processor and also the control logic is reduced as most of the control flow is written in the software. During the execution, while the reconfigurable part of rASIP stays busy in processing the large amount of data, the ASIP can handle the other specific applications of relatively less computation intensity.

An important class of reconfigurable devices is a Coarse-Grain Reconfigurable Array (CGRA). CGRAs have a proven track record of almost ten years and some of the most popular CGRAs are ADRES [9], Morphosys [8], PACT-XPP [10]. The only drawback is that they require an area of a few million gates and such a large area utilization is not justified unless heavily utilized. To avoid this problem, CGRAs were designed as a template which could generate special-purpose array-based accelerators tailored for a set of applications [12]. CGRAs are ideal for processing computationally-intensive signal processing algorithms as they offer a high throughput and parallelism. Some of the interesting examples are Wideband Code Division Multiple Access (WCDMA) cell search [14], Viterbi decoders [15] and image/video processing [11], [16].

In this paper, we have used a template-based CGRA called AVATAR [13] as the reconfigurable part in ASIP. In past, AVATAR-generated mixed-radix(2, 4) FFT accelerator could process 64 and 128-point of FFT algorithms while satisfying IEEE-802.11n execution time constraints. The supporting system for the AVATAR generated accelerator was a RISC processor called COFFEE [17], a Direct Memory Access (DMA) device [19] which was responsible to fetch a large amount of data from the main memory of the system and provide it to the internal memory of the accelerator. In between all these modules, there was a network of switched interconnections which used to provide dedicated connections for high-speed transfer of data. The rASIP presented in this paper has the AVATAR integrated in the datapath of a simple RISC processor and the control flow is operated by special instructions added in the instruction-set of the RISC processor. In this way, we have reduced the overhead caused by the DMA and network of switched interconnections in terms of resource utilization. Furthermore, the control logic has become simpler as now it only has to control the cycle accurate processing by the accelerator and not the functionality of the DMA. We synthesized the rASIP for Stratix-IV Field Programmable Gate Arrays (FPGAs), measured the execution time, resource utilization and estimated power and energy consumption for

comparison with COFFEE-based system and other state-of-the-art.

In the next section, we will explain the COFFEE/AVATAR (C/A)-based system. Then we will discuss, how AVATAR was separated from the previous system and then integrated into the datapath of a simple RISC processor to make an rASIP. Section IV discusses the synthesis results and comparisons based on execution time, resource utilization, power and energy consumption. In the last section, we will draw the conclusions.

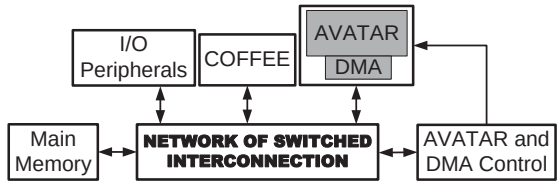


Fig. 1. COFFEE/AVATAR-based System

II. COFFEE/AVATAR-based Systems and the FFT Accelerator

This section is composed of two parts; the first one describes the architecture of C/A-based system which is considered as a reference for the design of rASIP. The second section describes the design and functionality of mixed-radix(2, 4) FFT accelerator that was generated using AVATAR.

A. COFFEE RISC Processor and AVATAR

COFFEE RISC processor and AVATAR work in a processor/coprocessor model where a C code can be compiled for COFFEE using a customized gcc compiler. COFFEE controls the cycle accurate processing of the AVATAR-generated accelerator by passing the control words to the control registers of the accelerator. AVATAR is written in VHDL and most of the information related to AVATAR can be found in [13] and [12] but the key points are highlighted here to build the discussion.

AVATAR is a 4×16 processing element (PE) template-based CGRA. Each PE can perform 32-bit integer arithmetic and logic operations plus 32-bit IEEE-754 floating-point operations. The PEs have two inputs and two outputs and they can connect with their neighboring PEs in point-to-point fashion. In between the local memories and the processing array of AVATAR, there are I/O-buffers that provide interleaving to the data from the local memories before and after it is processed through the array.

As AVATAR is a template-based device, the user can specify using a graphical platform different contexts that comprise the operation(s) to be performed by each PE and the pattern of interconnection among the PEs to be made at each clock cycle. The graphical platform in return generates a VHDL file containing the parameters that set the condition for the hardware components to be instantiated inside the accelerator. Another file generated is a C header file containing the configuration stream. At run-time, the configuration stream is fetched from the main memory of the system by the DMA and then distributed to the respective configuration memories of the PEs by a pipeline infrastructure designed to reduce the cost of distribution of the configuration bit-stream [18]. Then the DMA starts loading the data in one of the local memories in a time-multiplexed way as required by the accelerator. AVATAR is equipped with two local memories that contain the data to be processed. Once the data to be processed is loaded in the local memories, the COFFEE RISC processor can enable different contexts as required by the flow of algorithm. Once the data has been processed, it can be fetched from the local memories and stored in the main memory of the system. Fig. 1 shows the C/A-based system.

B. The FFT Accelerator

The main reason for the design of AVATAR was to accommodate a radix-4 butterfly in a single context as CREMA-generated accelerator needed three different contexts for the same purpose [20]. CREMA was a 4×8 PE template-based CGRA and was needed to be scaled-up to 4×16 PEs to satisfy the execution time constraints of IEEE-802.11n standard for FFT processing. The design of the FFT accelerator is described in detail in [13] but some of the important details are highlighted in this subsection. AVATAR-generated mixed-radix(2, 4) FFT accelerator could process both 64 and 128-point FFT algorithms. To process 64-point of FFT, only radix-4 scheme will be required that completes the processing in three stages. To process a 128-point FFT, the first stage is processed by radix-2 scheme and it needs three more stages by radix-4 scheme to complete the processing. The main structure of the accelerator consisted of two contexts; the first context contained four radix-2 butterflies and the second context contained one radix-4 butterfly. The other contexts were designed for preprocessing which is required between two processing steps. Preprocessing is required to reorder the data as we violate the inherent parallelism of the algorithm by employing less number of butterflies than the algorithm demands from its signal flow graph. Finally, the accelerator processed four different streams of 64 and 128-point FFT within 3.2μs+0.8μs (guard interval) = 4.0μs.

III. The Architecture of rASIP, Integration and Testing

We have used Synopsys Processor Designer, LISA (Language for Instruction Set Architectures) for the integration of AVATAR generated accelerator with a RISC processor core [21]. A template RISC processor model in LISA is used as the starting point in integration. The RISC processor has five pipeline stages that are instruction-fetch, instruction-decode, execute, memory-access and write-back plus all the respective pipeline registers. Special instructions are extended in the RISC processor to operate the AVATAR generated accelerator. The design of rASIP was complicated from integration point of view, requiring three different phases which are explained as follows.

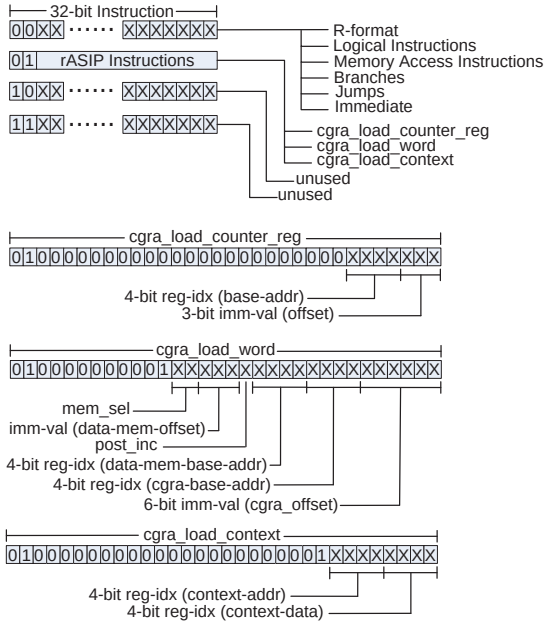


Fig. 2. The Extension in the Instruction Set

A. rASIP Architecture

The RISC-template in LISA tool is a 32-bit architecture but the instructions that directly belong to RISC are 30-bit wide, so the two most significant bits can be used for adding special instructions to control the operations on AVATAR. The register-file contains 16 32-bit general-purpose registers and the register-file indexing is of 4 bits. The immediate values are of 16-bits in regular RISC related instructions. Fig. 2 shows the 32-bit instruction-set and the extensions we have introduced.

As described in section II, there are three basic steps that need to be followed for operating an AVATAR-generated accelerator. They are

1) load the configuration stream
2) load the data to be processed
3) write the control registers

The configuration stream and the data to be processed reside in the memory of rASIP. Configuration stream may consist of many configuration arrays in the main memory. The total number of configuration arrays and their sizes depend on the design of the context(s) that are made to constitute the accelerator from the AVATAR template during the design time. All of these configuration arrays have to be loaded one by one into the configuration memories of the accelerator. To address each and every configuration array, we need the base address and the size of the array. To store the configuration array, we also need to know the base address plus the offset to be added to address the configuration memories. These are the same parameters that are required to store the configuration words in the configuration memory of I/O-buffers that are described in section II. To load the data to be processed from the main memory and store it in one of the local memories of the accelerator, we need the base address plus the offset of the respective memories. In this regard, we have employed two adders to calculate the address of the main memory and one of the local memories of the accelerator. The two adders calculate the addresses at the execute pipeline stage of the RISC processor. Each adder increments the address iteratively starting from the base address with an offset. The iteration stops when all the configuration and data words are loaded from an array stored in the main memory. To count the iterations, the counter in the memory-access stage counts down starting from the size (total number of words) of the configuration or data array. By using these two adders and a counter, we complete the load from the main memory and store in one of the local memories of the accelerator at the cost of only one clock cycle per word. The instruction *cgra_load_counter_reg* and its instruction coding for loading the counter register by the size of the configuration or data array is shown in Fig. 2. To load and store the configuration, data and context words, we added special instructions that are *cgra_load_word*, *cgra_load_context* and their coding is also mentioned in Fig. 2. The *mem_sel* is of two bits that either enables the configuration memory or one of the two local memories for write operations. If *mem_sel*=”00”, then the data can be read from second local memory in case that the accelerator has completed the data processing. The one-bit field *post_inc* is used as an enable signal for the register in the counter. The rest of fields are obvious considering the above discussion. The context words are written at specific addresses of the data memory, so we used RISC load-word instruction to load these context words from those specific addresses and write them in the register file of the RISC processor. To carry out this operation, the *cgra_load_context* instruction contain two 4-bit fields for register indexing. The first field contains the address of the register file where the address of the main memory will be loaded and the other for the context word to be loaded in the register file from that address in the main memory. The address of the main memory from where the context word is fetched has to be transfered to the accelerator as there are two control registers that have to be written one at a time. The address in the main memory corresponding to the context word is provided to the address decoder inside the accelerator which decides which of the control registers has to be written by a specific context word.

B. Integration and Testing

The integration between RISC and AVATAR-template is supported by an interface register as shown in Fig. 3. The upper part of the figure shows the RISC processor while the lower part is the AVATAR template. The interface register has the following fields which can be updated as required by the RISC processor.

- address[31:0]: 32-bit address field
- data[31:0]: 32-bit data field

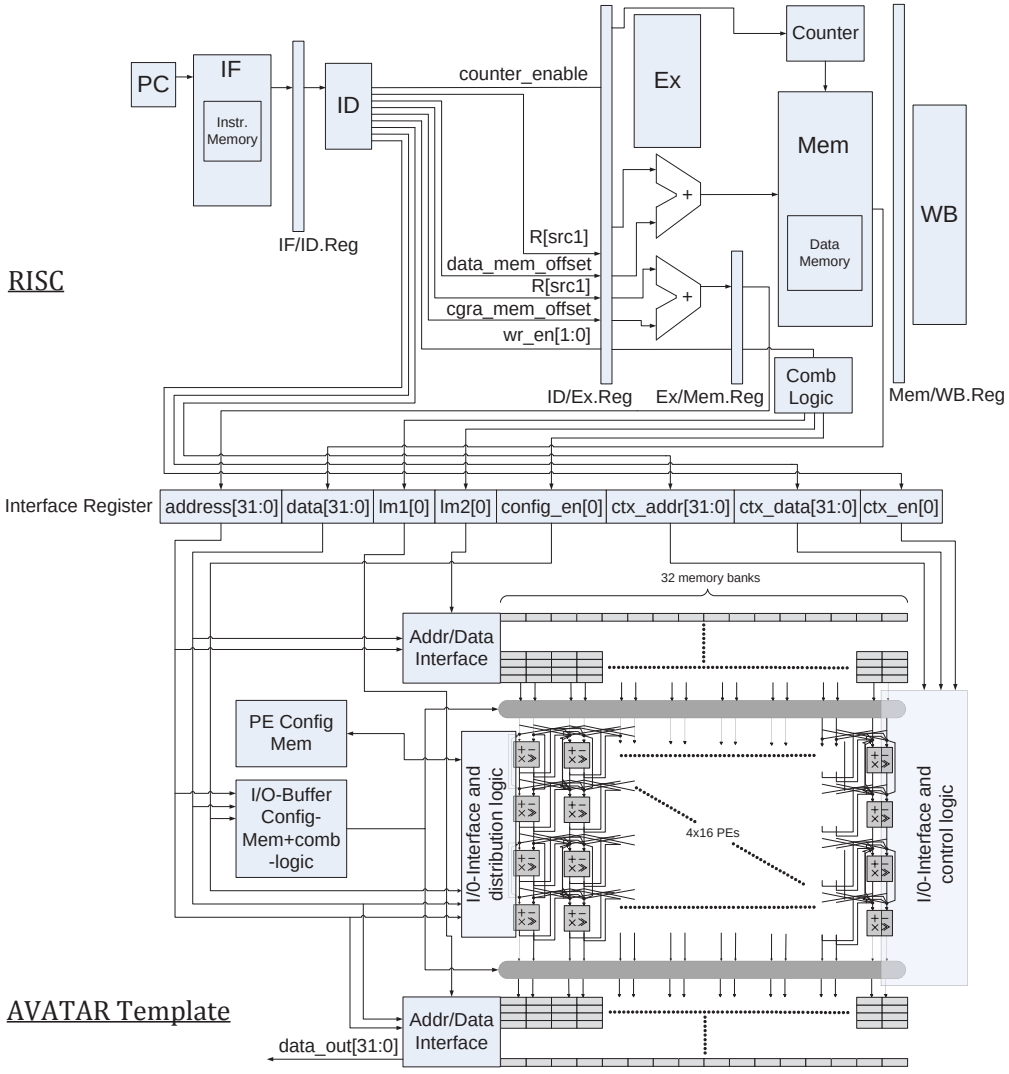


Fig. 3. Integration of AVATAR in the datapath of RISC processor

- lm1[0]: enable for first local memory
- lm2[0]: enable for second local memory
- config_en[0]: enable for configuration memories
- ctx_addr[31:0]: 32-bit context address
- ctx_data[31:0]: 32-bit context data
- ctx_en[0]: context enable

The 32-bit address and data bus is provided to multiple blocks of the AVATAR-generated accelerator and the data is loaded to the specific block when the relevant enable signal is high. The decision on whether the configuration word will be stored in PEs configuration memory or I/O-buffer configuration memory depends on the address field. The processed data can be read from the second local memory of AVATAR generated accelerator and in that case $lm1[0]$, $lm2[0]$ and $config\_en[0]$ signals will all be in active low state.

Integration of this rASIP was a complicated task as the system in which AVATAR was integrated before, shown in Fig. 1, was an effort of many engineers spanning over several years. To extract AVATAR from this system, we needed to be sure that the functional correctness remains unchanged. We wanted the AVATAR-generated mixed-radix(2, 4) FFT accelerator to be operated in a VHDL test-bench before its

System	Comb ALUTs	Logic Registers	DSPs
C/A [13]	40,227	19,313	128
rASIP	36,083	14,429	112
Savings [%]	10.3	25.2	12.5

TABLE I
RESOURCE UTILIZATION BY COFFEE/AVATAR (C/A) SYSTEM AND RASIP ON STRATIX-IV FPGA DEVICE (EP4S100G5H40I1)

FFT Algo	Freq. (MHz) Slow (100°C)	Freg. (MHz) Fast (0°C)	Cycles	Exe. Time
64-point	159.16	280.11	357	1.27μs
128-point			575	2.05μs
4×64			884	3.15μs

TABLE II
OPERATING FREQUENCY AND EXECUTION TIME BY RASIP FOR PROCESSING 64 AND 128-POINT FFT ALGORITHMS

functional requirements are analyzed to be integrated with the RISC processor. As the accelerator was controlled by the flow written in C compiled for COFFEE RISC core, we step-by-step commented the C code for radix-4 64-point FFT execution and replaced the commented C functions by VHDL test sequences while keeping the synchronization between C code execution and VHDL by introducing fixed-time delay intervals. Using this method, the whole C code for 64-point FFT was commented and the accelerator was completely stimulated by a VHDL test-bench. The compiler and VHDL code for the rASIP is generated by Processor Designer. Since AVATAR template is also written in VHDL, we port-mapped the accelerator with the interface register of the RISC processor. At the time of integrating the accelerator with the RISC processor customized using LISA tool, we step-by-step commented the VHDL code and wrote equivalent micro-coded functions that were called in the C flow to be compiled. After the completion of each step, we used to check the functional correctness for any errors and the synchronization between C flow and VHDL stimulator was established again using constant delay intervals. In this way, the whole VHDL test-bench was commented and the radix-4 64-point FFT execution control was completely transferred to the customized RISC core. We also mapped 128-point FFT on this accelerator and after comprehensive functional testing, we are confident that any other application-specific accelerator can be tailored using AVATAR template and can be operated just by writing the control flow in C for this customized RISC processor.

IV. SYNTHESIS RESULTS AND COMPARISONS

We have synthesized the rASIP on two different Stratix-IV FPGA devices to establish comparison based on resource utilization, execution time, power and energy consumption. As described in section I, the DMA along with the network of switched interconnections was removed as now the configuration and data words can be written using special instructions. Furthermore the control unit for AVATAR-generated accelerator is simpler as now it does not have to control the functionality of DMA but only the processing of the accelerator. The resource utilization by rASIP on Stratix-IV FPGA device (EP4S100G5H40I1) is shown in Table I in comparison with the system shown in Fig. 1. The system in Fig. 1 was also synthesized on EP4S100G5H40I1 device in [13]. From the table, we can observe that there is 25% decrease in the usage of logic registers and 10% decrease in ALUTs consumption when rASIP is compared with C/A-based system. This reduction in resource utilization was the target to be achieved besides studying the advantages of using rASIP. In rASIP and C/A-based system, most of the resources are consumed by AVATAR as it is a large 4×16 PE CGRA. If a smaller CGRA is used in both of the systems, for example a 4×4 PE CGRA, then there will be a larger difference in resource utilization between the two systems.

The operating frequencies achieved at the temperatures of 0 and 100°C on the same FPGA device are shown in Table II. To calculate the execution time, we have considered operating frequency at 0°C (fast timing model) as even in [13], the fast time model was used to calculate the execution time. However, we don't see much time difference between the rASIP and C/A-based system. For example rASIP needs 3.15μs to process 4×64-point FFT while C/A-based system needed 3.07μs while both of them are satisfying the execution-time constraints for IEEE-802.11n standard. On top of this, rASIP requires less area than the C/A-based system which shows the significance of using rASIP. To satisfy the timing constraints for 4×128-point FFT, there is a double C/A-based system which requires double the amount of resources than a single C/A-based system. The double C/A-based system contains two AVATAR and such a system cannot be compared with rASIP which contains only a single AVATAR. It can not be expected by a device like rASIP to meet constraints of IEEE-802.11n to process four streams of 128-point for FFT within 4.0μs unless two AVATARs employed in rASIP.

The power consumption of rASIP was estimated by generating the post-fit gate-level netlist of rASIP for Stratix-IV FPGA device (EP4SGX70HF35C2) device. The netlist was then simulated at 0°C and at an operating frequency of 160 MHz to create the value-change-dump file which is used to estimate the power consumption. This is the most accurate method for estimating the power consumption and we achieved a 'HIGH' estimation confidence by the Quartus II tool. The static and dynamic power consumption of rASIP and the C/A-based system are almost the same as shown in Table III. The power consumption mentioned in the table for the C/A-based system is related only to the mixed-radix(2, 4) FFT accelerator and its control unit plus the DMA. It does not include the power consumption by the COFFEE processor and the network of switched interconnections. The rASIP contains the mixed-radix(2, 4) FFT accelerator generated from AVATAR, a simplified control unit, a simple RISC processor and requires almost the same power consumption. The energy consumption by rASIP for processing 64 and 128-point FFT algorithms is shown in Table IV which is almost the same in

System	Static Power	Dynamic Power	I/O Power	Total Power
C/A [22]	466.76 mW	472.73 mW	46.41 mW	985.90 mW
rASIP	471.09 mW	492.02 mW	61.45 mW	1024.55 mW

TABLE III

COMPARISON OF RASIP POWER CONSUMPTION WITH THE C/A-BASED SYSTEM

System	FFT Algorithm	Frequency (MHz)	Clock Cycles	Execution-time	Power	Energy
C/A	64-point	166.6	355	2,130 ns	985.90 mW	2.09μJ
rASIP	64-point	160.0	357	2,231 ns	1024.55 mW	2.28μJ
C/A	128-point	166.6	570	3,420 ns	985.90 mW	3.37μJ
rASIP	128-point	160.0	575	3,593 ns	1024.55 mW	3.68μJ

TABLE IV

ENERGY CONSUMPTION BY C/A ACCELERATOR AND RASIP FOR PROCESSING 64 AND 128-POINT FFT ALGORITHMS

comparison with C/A's accelerator, control and DMA.

V. ACKNOWLEDGEMENT

This work is the result of joint collaboration between Department of Electronics and Communications Engineering, Tampere University of Technology, Finland and Chair for Integrated Signal Processing System (ISS), Institute for Communication Technologies and Embedded Systems (ICE), Rheinisch-Westfaelische Technische Hochschule (RWTH), Aachen, Germany. It was partially funded by the Academy of Finland under contract # 258506 (DEFT: Design of a Highly-parallel Heterogeneous MP-SoC Architecture for Future Wireless Technologies and ISS-ICE-RWTH provided the necessary resources for its implementation.

VI. CONCLUSION

In this paper, we have presented a Reconfigurable Application-specific Instruction-set Processor (rASIP) which is designed by integrating a template-based Coarse-Grain Reconfigurable Array (CGRA) in the datapath of a Reduced Instruction-set Computing (RISC) processor. The integration was carried out by extending the instruction set of RISC processor with special instructions to control the cycle-accurate processing by the CGRA generated accelerator. A mixed-radix(2, 4) Fast Fourier Transform (FFT) accelerator had been generated from the CGRA which is used in this work. The rASIP satisfies the partial execution-time constraints of IEEE-802.11n standard for FFT processing. We have observed substantial reduction in resource utilization by rASIP while the power and energy consumption are almost the same in comparison with a standard processor/coprocessor model.

REFERENCES

[1] S. Saponara, M. Casula, L. A. Fanucci, "ASIP-based reconfigurable architectures for power-efficient and real-time image/video processing", Journal of Real-Time Image Processing, Vol. 3, September 2008, http://dx.doi.org/10.1007/s11554-008-0084-y, Springer-Verlag.

[2] T. Vogt, N. Wehn, "A reconfigurable application specific instruction set processor for Viterbi and Log-MAP decoding". IEEE Workshop Signal Proc Syst Des Impl, pp. 142147 (2006).

[3] L. Dinoi, R. Martini, G. Masera, F. Quaglio, F. Vacca: ASIP design for partially structured LDPC codes. Electron Lett 42(18), 4950 (2006).

[4] J. Lee, J. Moon, K. Heo, M. Sunwoo, S. Oh, I. Kim, "Implementation of application-specific DSP for OFDM systems". In Proceedings of IEEE International Conference on Circuits and Systems (ISCAS), pp. 665668 (2007)

[5] H. Yue, M.-C. Lai, K. Dai, Z. Y. Wang, "Design of a configurable embedded processor architecture for DSP functions" in proceedings of IEEE ICPADS05, pp. 2731 (2005).

[6] M. Alles, T. Vogt, C. Brehm, N. Wehn, "FlexiChaP: A Dynamically Reconfigurable ASIP for Channel Decoding for Future Mobile Systems", in Dynamically Reconfigurable Systems, Chapter 14, Editors: M. Platzner, J. Teich, N. Wehn, http://dx.doi.org/10.1007/978-90-481-3485-414, Springer Netherlands, pp. 293-314, January 2010.

[7] H. Hassan, K. Mohammed, A. Shalash, "Implementation of a reconfigurable ASIP for high throughput low power DFT/DCT/FIR engine", in EURASIP Journal on Embedded Systems, April-2012, pp. 1-18, Springer International Publishing, http://dx.doi.org/10.1186/1687-3963-2012-3

[8] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. C. Filho, "Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications". IEEE Trans. Computers, vol. 49, no. 5, pp. 465-481, 2000.

[9] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix", Field-Programmable Logic and Applications, vol. 2778, pp. 61-70, September 2003, ISBN 978-3-540-40822-2.

[10] V. Baumgarte, G. Ehlers, F. May, A. Nuckel, M. Vorbach, and M. Weinhardt, "PACT XPP-A Self-Reconfigurable Data Processing Architecture", The Journal of Supercomputing, vol. 26, no. 2, pp. 167-184, September 2003.

[11] C. Brunelli, F. Garzia, and J. Nurmi, "A Coarse-Grain Reconfigurable Architecture for Multimedia Applications Featuring Subword Computation Capabilities", in Journal of Real-Time Image Processing, Springer-Verlag, 2008, 3 (1-2): 21-32. doi:10.1007/s11554-008-0071-3.

[12] F. Garzia, W. Hussain and J. Nurmi, "CREMA, A Coarse-Grain Reconfigurable Array with Mapping Adaptiveness", in Proc. 19th International Conference on Field Programmable Logic and Applications (FPL 2009). Prague, Czech Republic: IEEE, September 2009.

[13] W. Hussain, F. Garzia, T. Ahonen and J. Nurmi, "Designing Fast Fourier Transform Accelerators for Orthogonal Frequency-Division Multiplexing Systems", in the Springer's Journal of Signal Processing Systems, December 2010.

[14] F. Garzia, W. Hussain, R. Airoldi, J. Nurmi, "A Reconfigurable SoC tailored to Software Defined Radio Applications", in Proc of 27th Norchip Conference, Trondheim (NO), 2009.

[15] Y. Kishimoto, S. Haruyama, H. Amano, "Design and Implementation of Adaptive Viterbi Decoder for Using A Dynamic Reconfigurable Processor" in Proc. Reconfigurable Computing and FPGAs, 2008. ReConFig '08, pp 247-252, doi=10.1109/ReConFig.2008.39, ISBN: 978-1-4244-3748-1.

[16] Chia-Cheng Lo, Shang-Ta Tsai, Ming-Der Shieh; , "A reconfigurable architecture for entropy decoding and IDCT in H.264", VLSI Design, Automation and Test, 2009. VLSI-DAT '09. International Symposium on , vol., no., pp.279-282, 28-30 April 2009, doi: 10.1109/VDAT.2009.5158149, ISBN: 978-1-4244-2781-9.

[17] J. Kylliainen, T. Ahonen, and J. Nurmi, "General-purpose embedded pro-

cessor cores - the COFFEE RISC example", In Processor Design: System-on-Chip Computing for ASICs and FPGAs, J. Nurmi, Ed. Springer Publishers, June 2007, ch. 5, pp. 83-100, ISBN-10: 1402055293, ISBN-13: 978-1-4020-5529-4.

[18] F. Garzia, C. Brunelli and J. Nurmi, "A pipelined infrastructure for the distribution of the configuration bitstream in a coarse-grain reconfigurable array", in Proceedings of the 4th International Workshop on Reconfigurable Communication-centric System-on-Chip (ReCoSoC'08). Univ Montpellier II, July 2008, pp. 188-191, ISBN:978-84-691-3603-4.

[19] C. Brunelli, F. Garzia, C. Giliberto and J. Nurmi, "A Dedicated DMA Logic Addressing a Time Multiplexed Memory to Reduce the Effects of the System Buss Bottleneck", in Proc. 18th International Conference on Field Programmable Logic and Applications, (FPL 2008), Heidelberg, Germany, 8-10 September 2008, pp. 487-490.

[20] W. Hussain, F. Garzia, and J. Nurmi, "Evaluation of Radix-2 and Radix-4 FFT Processing on a Reconfigurable Platform," in Proceedings of the 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'10). IEEE, pp. 249-254, April 2010, ISBN 978-1-4244-6610-8.

[21] A. Hoffmann, O. Schliebusch, A. Nohl, G. Braun, O. Wahlen, H. Meyr, "A methodology for the design of application specific instruction set processors (ASIP) using the machine description language LISA", in Proc. IEEE/ACM International Conference on Computer Aided Design, 2001. ICCAD 2001, pp.625-630, 2001.

[22] W. Hussain, T. Ahonen F. Garzia and J. Nurmi, "Energy and power estimation of Coarse-Grain Reconfigurable Array based Fast Fourier Transform accelerators", in Proc. 7th International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), pp.1-4, 9-11 July 2012.

PUBLICATION 6

”W. Hussain, R. Airoldi, H. Hoffmann, T. Ahonen and J. Nurmi, *”Design of an Accelerator-Rich Architecture by Integrating Multiple Heterogeneous Coarse Grain Reconfigurable Arrays over a Network-on-Chip”*.

Design of an Accelerator-Rich Architecture by Integrating Multiple Heterogeneous Coarse Grain Reconfigurable Arrays over a Network-on-Chip

Waqar Hussain∗†, Roberto Airoldi∗, Henry Hoffmann†, Tapani Ahonen∗, Jari Nurmi∗
∗ Department of Electronics and Communications Engineering, Tampere University of Technology,
FI-33101, Tampere, Finland. Email: firstname.lastname@tut.fi
† Department of Computer Science, University of Chicago, IL-60637, USA.
Email: hankhoffmann@cs.uchicago.edu, waqar@cs.uchicago.edu

***Abstract*—This paper presents an accelerator-rich system-on-chip (SoC) architecture integrating many heterogeneous Coarse Grain Reconfigurable Arrays (CGRA) connected through a Network-on-Chip (NoC). The architecture is designed to maximize the reconfigurable processing capacity for the execution of massively parallel algorithms. The central node of the NoC contains a Reduced Instruction Set Computer (RISC) core that manages distribution of computing functions and data within the SoC while the other nodes contain CGRAs of application-specific sizes. Prior approaches coupled only a few accelerators with a RISC core using special instructions and/or a direct memory access device. In contrast, our design couples a RISC core to many CGRAs through the NoC. This approach provides for independent and simultaneous execution of multiple computing kernels. Furthermore, the proposed architecture mitigates power dissipation as CGRA sizes are tailored for the individual application kernels. We present a proof-of-concept design with a total of 408 reconfigurable processing elements. This instance and its subsystems are customized and tested for different computationally-intensive signal processing algorithms. The overall single-chip computing system is synthesized for a Field Programmable Gate Array device. We present comparison to and evaluation against some of the existing multicore systems in terms of multiple performance metrics.**

I. Introduction

This work is motivated to maximize the number of processing resources in the form of accelerators available on a System-on-Chip considering the fact that there is always a fraction of the chip that can not be used at full speed due to technology-imposed utilization wall. In [1], it is shown that only a 7% of 300mm^2 die can be used at full frequency within a power budget of 80W. The un-utilized part of the chip is now-a-days called *Dark Silicon*. The dark silicon can be replaced with accelerators as they are not used most of the time.

Another motivation to propose an accelerator-rich platform is to meet an increasing demand for throughput, for example the number of spatial streams in IEEE-802.11 family of wireless standard has increased from one to four [2]. The upcoming IEEE-802.11ac would require processing of 5-8 different streams simultaneously. A platform with many independent accelerators will be able to execute multiple applications with different throughput requirements.

In this regard, we present Heterogeneous Accelerator-Rich Platform (HARP) designed by integrating many Coarse Grain Reconfigurable Arrays (CGRAs) over a Network-on-Chip (NoC). The CGRAs connected to the NoC nodes are of application-specific sizes except the central node of the NoC which contains a Reduced Instruction-Set Computing (RISC) core. The RISC core is responsible for controlling the overall processing. In addition to control tasks for CGRAs, the RISC can perform general-purpose processing.

The general-purpose CGRA have a proven track-record of executing many computationally-intensive and data parallel applications. They consume an area of a few million gates, for example BUTTER [6], Morphosys [7], ADRES [8] and PACT-XPP [9]. In HARP design, we have used template-based CGRAs where each and every resource is instantiated tailored to the requirements of the target application. The overall design of HARP is also template-based, for example if a node on the network is not required then the user has an option not to instantiate it.

In recent past, processor architectures tightly coupled CGRAs with a RISC processor first by using a Direct Memory Access (DMA) device [10] and then by integrating the CGRA in the datapath of RISC core [12]. Special instructions were introduced by extending the instruction set of the RISC core to support CGRA operations. Tight coupling has an advantage as RISC and the CGRA has a close cooperation that allows faster communication protocols and data transfers. The disadvantage is, as the CGRA acts as a functional unit of the RISC core, therefore it can not be addressed by other cores in the system. The loosely coupled accelerators can be shared among many cores on the system, the drawback is the slow data transfer between the cores over the NoC but this overhead can be reduced if the data is transmitted to all nodes simultaneously.

This paper demonstrates the benefits of HARP's loosely coupled accelerator-rich design through a case study with seven accelerators placed around the central RISC core. The main contributions in HARP are as follows.

1) Integrate many CGRAs over an NoC so they are loosely coupled with each other and with the RISC core.
2) Each CGRA can address other CGRAs and the RISC

core.

3) The RISC processor can supervise the overall control and processing.
4) Simultaneous execution of multiple kernels independently on different CGRAs.

In HARP, a few nodes perform Fast Fourier Transform (FFT) while others perform complex and real Matrix-Vector Multiplication (MVM). We have used these algorithms for the proof-of-concept and for testing the architectural functionality of the design.

In the next section, we will discuss some of the existing platfoms. Section III explains the architectural and programming features of HARP. In Section IV, implementation of kernels on HARP is presented and in Section V, we will discuss the experimental results based on different performance metrics. In Section VI, we perform comparison with the other state-of-the-art platforms. Conclusions are presented in the last section.

II. Existing Platforms

The application-specific heterogeneous accelerator systems are designed to accelerate many specific algorithms. In HARP, for specific algorithms, we have used specifically tailored accelerators to application's specific needs. In addition to HARP, a system is presented in [3] which is composed of heterogeneous cores motivated to match the hardware granularity with application granularity.

NineSilica [4] is a general-purpose homogeneous multicore system, composed of the same processing cores connected with each other over a NoC. It is a network of nine RISC cores connected in 3×3 mesh topology. NineSilica requires 10.3 μs to compute a 64-point FFT.

Another MPSoC is presented in [13] composed of three application-specific Transport Triggered Architecture (TTA) based processors that are communicating with each other using a shared memory. The TTA-MPSoC platform performs LTE baseband processing kernels including carrier frequency synchronization, demodulation, channel estimation and symbol detection. The system has a total dynamic power consumption of 105.04 mW at 200 MHz using a 130 nm technology.

In [14], a Multiple Instruction Multiple Data (MIMD) platform is presented by integrating several general-purpose microprocessors (μPs). The μPs can be assisted by additional accelerators called Processing Units (PUs). The μPs and PUs exchange data and messages using two separate networks. The system is synthesized on an FPGA providing a 19.2 Giga Operations per Second (GOPS) performance.

A DSP processor called DREAM which is based on dynamically reconfigurable datapath in presented in [15]. It delivers a 0.2 GOPS/mW performance using a 90nm CMOS technology. DREAM is suitable for integration in complex heterogeneous multicore platforms.

In [16], a platform named P2012 is presented that consists of four 16-processor clusters connected via a NoC. The processors are locally synchronous and globally asynchronous and perform image processing related algorithms. The platform promises to deliver 80 GOPS/2W performance using a 28nm technology.

MORPHEUS [17] is the most recently published heterogeneous multiple accelerator platform consisting of a fine-grain device called FlexEOS, a mid-grain reconfigurable datapath DREAM and a CGRA called XPP-III. The MORPHEUS chip is implemented using 90nm CMOS technology, dissipates an average dynamic power of 700 mW and delivers a performance of 20 GOPS/W.

In comparison to all of these existing platforms, we have been able to integrate more accelerators and to maximize the number of processing elements on our platform, therefore delivering a high computational power.

III. HARP Architecture

The HARP is described textually as parameterized VHDL. It consists of nine nodes connected with each other in a 3×3 mesh topology using a NoC [18]. Each node except the central node contains a CGRA, a data memory, a DMA device with master and slave interface. The master interface of the DMA sends data/control information to the network. It can also request access to the node's slave device (data memory) for read/write operation. The slave interface of the DMA receives information from the network which in return activates the DMA master. The central node contains an instruction memory, a data memory and a RISC core called COFFEE [11]. The overall processing is supervised by the RISC processor acting as master to the network. The RISC core transfers the data from its data memory to the data memory of the slave nodes. It also establishes synchronization between the nodes by broadcasting control information and receiving acknowledgments. The slave side of the DMA receives control information and passes it to the CGRA control unit for cycle accurate processing. Once the CGRA completes the processing, the DMA sends the acknowledgment to the central node.

The top level block diagram for HARP is shown in Fig. 1 where each CGRA is of different size. The CGRAs in the nodes of HARP act as FFT, complex and real MVM accelerators. The details related to these accelerators are not the subject of this article, however Section III-A briefly discusses these accelerators. These accelerators were generated from AVATAR, CREMA and SCREMA CGRAs. The design, functionality and performance related to these accelerators can be found in [21], [10], [19], [20] and [5]. Table I shows the types of CGRA connected to each node of HARP, the size in terms of the number of rows and columns of PEs and the kernel accelerated using that CGRA.

As HARP is a template-based architecture, the user can connect CGRA of a specific dimension based on the application requirements to any of the nodes of HARP. The architecture allows a CGRA not to be instantiated if it is not required for a node, for example the CGRA for Node 7 in Fig. 1 is not instantiated to demonstrate this feature. Such specifications are parameterized by the user in the definition VHDL package file of the design.

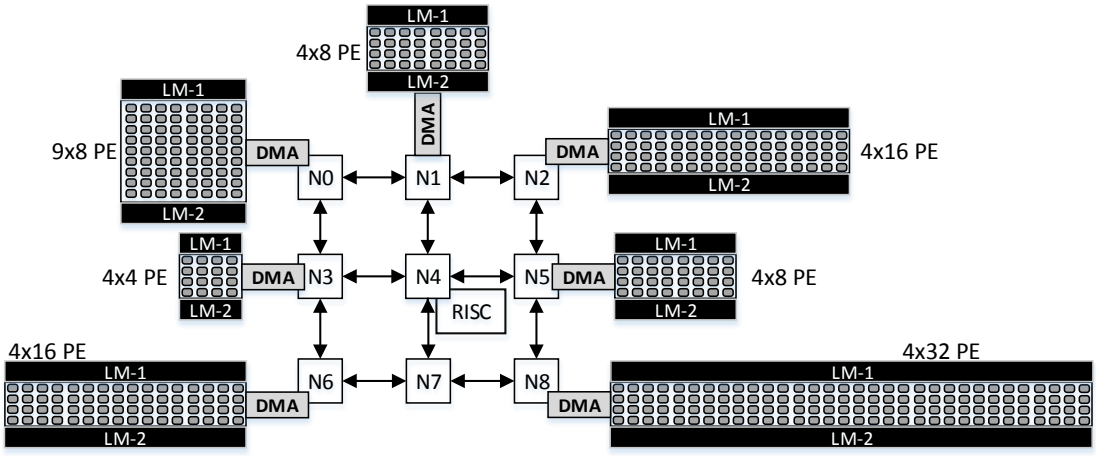


Fig. 1: Heterogeneous Accelerator-Rich Platform. LM and PE stand for Local Memory and Processing Element, respectively.

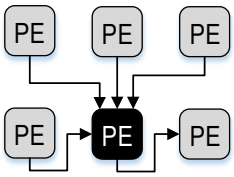
Node	PEs	CGRA Type	Kernel
N0	8×9	CREMA	radix-4 64-point FFT
N1	4×8	CREMA	complex MVM 64^{th}-order
N2	4×16	AVATAR	radix-(2, 4) 128-point FFT
N3	4×4	SCREMA	real MVM 64^{th}-order
N5	4×8		
N6	4×16		
N8	4×32		

TABLE I: Sizes in terms of the number of PEs in order of rows×columns for different CGRAs connected to HARP nodes, and the kernels accelerated on them, respectively.

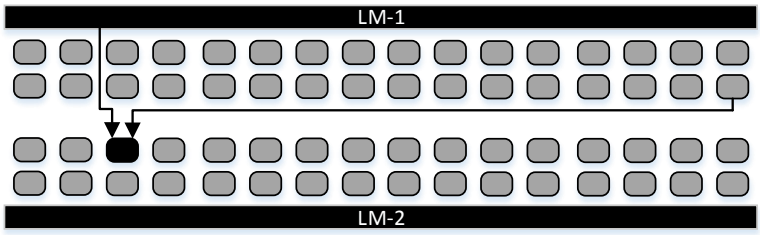
A. Coarse Grain Reconfigurable Arrays

The CGRAs used to build HARP have the same features except their sizes vary from one another due to scaling required for different applications. We use the common term 'CGRA' for all instances of CREMA, SCREMA and AVATAR. CREMA is a 4×8 Processing Element (PE) CGRA which is scalable only by increasing the number of rows of PEs. AVATAR is a scaled-up version of CREMA with a size of 4×16 PEs. SCREMA is a scalable version of CREMA which can be instantiated to 4, 8, 16 and 32 number of PE columns. The scaling and choice for each CGRA is made by the user based on the target application's computational requirements. The PE is the unit of computational structure for the CGRA which can do arithmetic and logic operations. The arithmetic operations can be performed both in 32-bit integer and IEEE-754 floating-point format. Each PE has two inputs (a, b) and two outputs (A, B). The operands to be processed arrive at inputs a and b. The input b is registered to output B for data reuse while the result from the arithmetic or logic operation inside each PE is registered to output A. The PEs can exchange data with the neighboring PEs in point-to-point fashion using local interconnections. The global interconnections are used to exchange data with PEs located at outermost PE column or directly from the local memory of the CGRA. The local and global interconnections among PEs are shown in Fig. 2. The CGRAs used in HARP are template-based devices as only those computational resources are generated in a PE which are required for a particular application. The connections between the PEs are dynamically configurable. This configurability is achieved with the help of multiplexers and the size of the multiplexer to be instantiated is also based on the application's data routing requirements. It also applies to the size of the configuration memories which are inside each PE, the address space, width of different address and data buses. The definition of a CGRA for a particular application is set using a Graphical User Interface (GUI) tool where each and every connection between different PEs and also the operation to be performed by each PE are set manually. A pattern of interconnections and operations to be performed by PEs at a particular clock cycle is called a context. The user designs different contexts for an application and the GUI tool generates the corresponding configuration stream. The configuration stream from the data memory is loaded by the DMA into the configuration memories of the CGRA during the system start-up time.

The CGRA is equipped with two local data memories of size based on the amount of data to be processed. In order to provide maximum bandwidth, the number of banks in the local data memory has to be equal to the total number of inputs of all PEs in the row of a CGRA. The data to be processed is loaded from the data memory of the respective node by



(a) Local Interconnections



(b) Global Interconnections

Fig. 2: Types of Interconnections between PEs. The target PE is shown in black as reference destination for connections.

the DMA device into one of these local memories. Once the DMA passes the control words from the RISC processor to the CGRA's control unit, the execution starts and data from one local memory flows through the PE array and results are stored in the other local memory. Based on the requirements of the application, the direction of data flow can be reversed and this process continues till the final results are achieved. Fig. 1 also shows the local memories associated to each CGRA.

B. Network, Nodes and Synchronization

The central node of the network containing the RISC processor acts as master and rest of the nodes act as slaves. Fig. 3 shows detailed view of both the master and the slave node. At the system start-up time, the RISC processor starts the data transfer from its data memory to the data memories of the other nodes. The data transfer includes the configuration streams and initial data which remains fixed throughout the processing. For example, in case of FFT and MVM, it sends twiddle factors and the matrix to the respective nodes as initial data. After the system start-up, it sends the vectors to all the nodes for processing.

The data transfer over the NoC is in form of packet transmission. The destination node receives the data packet when it arrives at its initiator. The initiator then sends it to the request switch. The request switch in return selects the target slave device. It can be data memory or the DMA slave. A node's master can also contact the request switch, which allows access to one of its local slave devices. This is required when the RISC needs to read or write data to the instruction

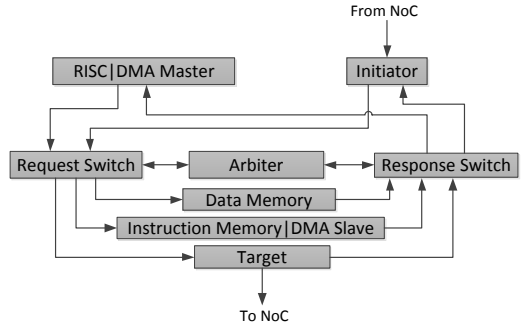


Fig. 3: A detailed view of master and slave node of HARP.

or data memory. In case of slave nodes, the DMA master may need to read data from data memory. If a node wants to transfer a data packet to the NoC, its master contacts the request switch which establishes connection to the target module of the node. The data packet is then delivered to the network. The routing information routes the packet to the destination node where it requests access to a slave device. In the master node, the master device is RISC core while in slave nodes, the master is DMA which is tightly integrated with the CGRA. Further details about NoC can be found in [18].

The RISC core has a shared space in its data memory which can only be read by the RISC core while the other nodes can

only write on it. This shared space is used for synchronization between two different nodes. The transfer between the data memories of nodes does not require any synchronization. However, the data transfer within a node is not packet-switched and needs synchronization as one data transfer has to complete before the next one starts. The synchronization is established such that at first the RISC sets its shared memory location corresponding to the destination node and sends a control packet for the DMA. Once the packet arrives at the destination node, it requests DMA slave and passes the control information. In return the DMA's master activates and starts the data transfer between the node's data memory to one of the local data memories of the CGRA. On the completion of the transfer, the DMA's master sends acknowledgment over the network, routed to the central node and targeted to its corresponding shared memory location which was set by the RISC core. As long as the acknowledgment is not written, the RISC does not send any other packet for the same Node's DMA.

The transfer of data can also be established between slave nodes. The results located in one of the local memories of CGRA can be transferred to the other node's data memory for further processing. This can be explained with the help of an example. Consider Node-0 (N0) needs to transfer data from one of its CGRA's local data memory to Node-1's (N1) data memory. Synchronization is established by the RISC core as it sets its own shared memory's location corresponding to N1 and then targets the DMA of N0. The Node-0's DMA starts transferring the data from one of the local memories of CGRA to N1's data memory and once the transfer is complete, it sends acknowledgment to the RISC shared memory's location corresponding to N1. The RISC does not target the DMA of N0 as long as it is busy and also does not write to the data memory of node N1.

IV. Application Mapping

The application mapping on HARP demonstrates and tests its design capabilities and functionality. The kernels mapped for each CGRA can be seen from Table I. A key design objective was to achieve loose coupling among the cores, so they can address each other. The mapping demonstrate that the slave nodes can exchange data with each other.

After the system start-up time, the Node-4 transports the vector to be processed to Node-0 where a 64-point, radix-4 FFT is computed. In the second step, the results of FFT computation are transported from the local data memory of the CGRA to the data memory of Node-1 by the DMA device. Node-1 then computes a 64^{th}-order complex MVM on the results from Node-0. Once the results are computed, the DMA of Node-1 transports the results from the local data memory of the CGRA to the data memory of Node-4. In this way, we are able to demonstrate that Node-4 has the overall supervisory control and the results of computation can be exchanged between the slave nodes before the final results are transported back to the master node. Node-2 is executing 128-point, radix-(2, 4) FFT independently and does not interact with other slave nodes. However, its execution is supervised by Node-4's RISC processor.

In order to demonstrate the simultaneous execution by the nodes of HARP, we employ integer MVM accelerators of different sizes for Node-3, Node-5, Node-6 and Node-8. Table I shows the sizes and types of the CGRAs used to generate accelerators for nodes. All of the CGRA's are performing 32^{nd}-order MVM with a relative performance increase as the size of the CGRA increases. The 32^{nd}-order matrix and vector to be processed is transported to all of the CGRAs simultaneously from the data memory of Node-4. Once the data is transported, Node-4 broadcasts the control words for the slave nodes. The slave nodes then start processing and send acknowledgments to Node-4 after the processing is complete. Table II shows clock cycles required for data transfers and processing by each CGRA. In the table, there are two categories of data transfers; the first one is about the data transfer from the data memory of Node-4 to the data memory of the slave node. The second is about the data transfer from the data memory of slave node to one of the local memories of the CGRA. This division is beneficial to enable the data transfers into pipeline steps to speed up the execution.

Node-to -Node	D. Mem to D. Mem	D. Mem to CGRA	Trans. Total	Exe. Total
N4-N0	1017	659	1676	420
N0-N1	1036	446*	1482	457
N1-N4	-	448*	-	-
N4-N2	2042	1033	3075	571
N4-N3	75622	-	75622	728
N4-N5				609
N4-N6				340
N4-N8				211

TABLE II: Clock cycles required for different stages of data transfer and processing. In the table, D. Mem, Trans. and Exe. stand for Data Memory, Transfer and Execution, respectively. Clock cycles with * sign represent data transfer from CGRA to Node's data memory.

ALMs	78,845 / 158,500	50%
Registers	64436	-
Memory Bits	21,000,928 / 38,912,000	54%
DSP	236 / 256	92%

TABLE III: Resource Utilization Summary for Stratix-V FPGA Device (5SGXEA4H1F35C1).

V. Implementation Results

HARP is synthesized on Stratix-V FPGA device for prototyping purposes. The overall design works on a single clock and after placement and routing, its frequency is found to be 133.08 MHz at 0°C and 117.94 MHz at 85°C for a slow timing model. The fast timing model shows 179.66 MHz at 0°C and 162.76 MHz at 85°C.

Considering an ambient temperature of 25°C and the Clock Cycles (CC) presented in Table II, we operate our design

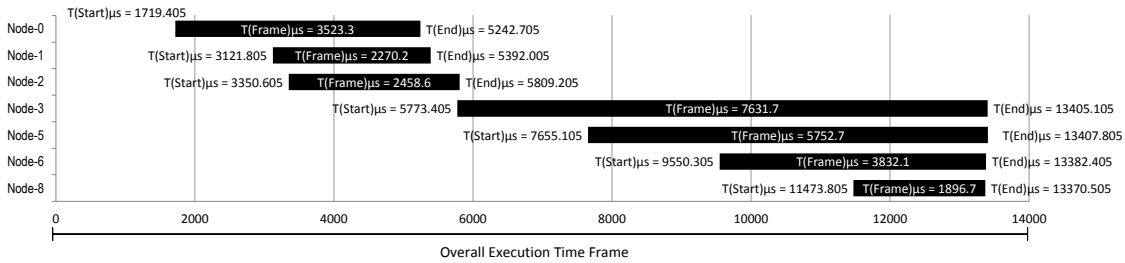


Fig. 4: The activation time for each kernel on a CGRA node with respect to the overall execution time frame.

Node	Accelerator Type	32-bit Multipliers	DSPs
Node-0	Radix-4 FFT	12	24
Node-1	Complex MVM	12	24
Node-2	Radix-(2, 4) FFT	28	56
Node-3	Real MVM	4	8
Node-4	RISC Core	6	12
Node-5	Real MVM	8	16
Node-6	Real MVM	16	32
Node-7	-	-	-
Node-8	Real MVM	32	64
Total	-	118	236

TABLE IV: Node-by-node break-down of DSP resources for the overall design.

Node CGRA Size Other HW	Algorithm (Type)	Dynamic Power (mW)	Active Time (μs)	Dynamic Energy (μJ)
N0 8×9 PE	Radix-4 FFT (64-point)	12.98	3523.3	45.7
N1 4×8 PE	Complex MVM (64-Order)	11.54	2270.2	26.2
N2 4×16 PE	Radix-(2, 4) FFT (128-point)	21.21	2458.6	52.1
N3 4×4 PE	Real MVM (32nd Order)	6.16	7631.7	47.0
N4 RISC	General Flow Control	5.72	13382.4	76.5
N5 4×8 PE	Real MVM (32nd Order)	9.96	5752.7	57.3
N6 4×16 PE	Real MVM (32nd Order)	17.57	3832.1	67.3
N7	-	0.03	-	-
N8 4×32 PE	Real MVM (32nd Order)	32.64	1896.7	61.9
NoC	-	0.67	13382.4	8.97
Integration Logic	-	31.54	-	-
Total	-	150.02	-	442.97

TABLE V: Dynamic Energy Estimation for each CGRA Node and the NoC.

at a frequency of 100 MHz. At these operating conditions, the execution time required by Node-0 for 64-point FFT processing is 420 CC×1/(100.0 MHz) = 4.2 μs. Similarly Node-2 processes 128-point FFT in 571 CC×1/(100.0 MHz) = 5.71 μs.

Table III shows the resource utilization summary for the same FPGA device. The design is consuming almost 50% of device resources. The total number of 18-bit DSPs resources utilized is 236 (92%). For each 32-bit multiplier instantiated in a PE, two 18-bit DSP elements are required on an FPGA. The break-down of the number of 32-bit multipliers instantiated in each node of design is given in Table IV. The reason for instantiating a specific number of multipliers for each CGRA generated accelerator can be found in [21], [19], [5] and [20].

The power dissipation estimates are also generated at an ambient temperature of 25°C and at an operating frequency of 100.0 MHz. The estimates were achieved by simulating the gate-level netlist of the design for an execution time frame of 14,000 μs. The estimation tool Quartus II PowerPlay Power Analyzer calculates the dynamic, static and I/O thermal power dissipation equal to 150.02 mW, 1088.72 mW and 18.41 mW respectively, a total of 1257.15 mW while observing a HIGH confidence metric in the user-provided signal toggle-rate data.

The bar chart in Fig. 4 shows the kernel activation time window in comparison to the overall simulated time frame. The chart shows three independent test schemes applied on HARP. The first one activates only Node-0 and Node-1. The Node-0 performs a 64-point FFT and transfers the results to Node-1, meanwhile Node-1 completes its configuration to perform complex MVM processing. As soon as Node-1 completes processing, it transfers the results back to the data memory of Node-4. The second test scheme is to demonstrate the independent execution feature of the platform. In this regard, Node-2 performs 128-point FFT independent of the computational activity of the other nodes. The third test case features the simultaneous execution of multiple kernels by HARP as Node-3, Node-5, Node-6 and Node-8 perform 32^{nd}-order real MVM and complete their jobs almost simultaneously. In the chart, the large inactive times contribute only to increase in static power estimate but are required for this proof-of-concept testing. However, using power management techniques, e.g., clock and power gating, the estimated 1088.72 mW static power dissipation can be brought down significantly.

Table V shows the dynamic power dissipation and energy consumption for each node of the system. It can be observed that the dynamic power dissipation increases as the size of the CGRA increases. However, CGRAs are scaled based on application's execution-time constraints, a large scale CGRA offers more parallelism and performs faster execution of ker-

Platform / Technology	Performance Metric	Platform's Value	HARP's Value	Gain
NineSilica [4] / FPGA 40 nm	FFT Exe. Time	10.3μs	4.2μ	2.5X
TTA-MPSoC / [13] CMOS 130 nm	dyn.pwr/freq (mW/MHz)	0.52	1.5	0.34X
[14] / FPGA 90 nm	GOPS	19.2	40.8	2.0X
DREAM / [15] CMOS 90 nm	GOPS/mW	0.2	0.032	0.16X
P2012 / [16] CMOS 90 nm	GOPS/mW	0.04	0.032	0.8X
MORPHEUS / [17] CMOS 90 nm	GOPS/mW	0.02	0.032	1.6X

TABLE VI: Comparisons based on different performance metrics with HARP implementation at 100 MHz on 28 nm FPGA. Exe, dyn.pwr, freq stand for Execution, Dynamic Power and Frequency, respectively.

nels and vice-versa. Table V also shows that the CGRAs in the system are custom tailored to the execution-time requirements of algorithms and therefore have a constraint-specific dynamic power dissipation. The dynamic energy consumption depends on the active time of the CGRA. The active time includes both the data transfer and processing time required by the CGRA node.

The current instance of the system offers 408 PEs. Considering the operating frequency of 100.0 MHz and total power dissipation of 1257.15 mW, HARP can deliver a performance of 0.032 GOPS/mW.

VI. Comparisons and Evaluation

The homogeneous MPSoC [4], requires 10.3 μs to compute a 64-point radix-4 FFT. In comparison, our design can compute the same task in 4.2 μs using only a single node. The homogeneous MPSoC requires 71,679 ALUTs on Stratix-IV device. The overall design of HARP requires 78,845 ALM on Stratix-V device, where an ALM on Stratix-V device is equal to two ALUTs on Stratix-IV device. It shows that HARP can provide a 2.5X speed-up at the cost 2X logic resources in comparison to homogeneous MPSoC.

The platforms in comparison with HARP were synthesized for both ASIC and FPGA as shown in Table VI. Based on the values of different performance metrics related to the platforms presented in Section II, we establish comparisons with HARP while considering the fact that cross-technology comparisons are not always accurate. Considering TTA-MPSoC with only three processors synthesized for ASIC and tailored for specific applications in comparison to our accelerator-rich architecture HARP, we expect to achieve better performance in terms of mW/MHz if HARP is also synthesized for ASIC. We achieved 2.0X more GOPS and nearly an equal performance in units of GOPS/mW while comparing to MIMD approach and P2012 platform, respectively. In comparison to MORPHEUS, a large heterogeneous platform with multiple accelerators, HARP performs 1.6X more in terms of GOPS/mW.

VII. Acknowledgment

This research work is jointly conducted by the Department of Electronics and Communications Engineering, Tampere University of Technology, Finland and the Department of Computer Science, University of Chicago, Illinois, USA. It was partially funded by the Academy of Finland under contract # 258506 (DEFT: Design of a Highly-parallel Heterogeneous MP-SoC Architecture for Future Wireless Technologies) and Tampere Doctoral Programme in Information Science and Engineering, Finland. The Department of Computer Science, University of Chicago, Illinois, USA also provided the financial and the on-site resources for its implementation.

VIII. Conclusions

In this paper, we present the design of a multiple Heterogeneous Accelerator-Rich Platform (HARP) and mapped different computationally-intensive kernels to verify its functionality and design features. HARP is designed by integrating many Coarse Grain Reconfigurable Arrays over a Network-on-Chip (NoC) while only the central node of the NoC contains a Reduced Instruction-Set Computing processor.

HARP is prototyped for a Field Programmable Gate Array device at an operating frequency of 100.0 MHz at 25°C. It showed a speed-up of 2.5X for 64-point FFT processing at the cost of 2X additional logic resources in comparison to a general-purpose homogeneous multi-processor system-on-chip.

HARP is also compared to heterogeneous systems, including a platform with several microprocessors (MIMD Approach), four 16-processor cluster connected via a NoC called P2012 and MORPHEUS which is a large heterogeneous platform with many accelerators. It is found that HARP performs 2.0X in Giga Operations per Second (GOPS) in comparison to MIMD Approach, 0.8X of P2012 and 1.6X of MORPHEUS in terms of GOPS/mW, respectively.

The implementation results and comparisons demonstrate the advantages of maximizing the number of processing elements on a platform while developing an accelerator-rich loosely coupled heterogeneous architecture.

References

[1] G. Venkatesh, J. Sampson, N. Goulding, S. Gracia, V. Bryksin, J. L. Martinez, S. Swanson, M. B. Taylor, "Conservation cores: reducing the energy of mature computations", ASPLOS 10, pp. 205218, 2010.

[2] "IEEE Standard for Information technology Telecommunications and information exchange between systems Local and metropolitan area networks Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 5: Enhancements for Higher Throughput", IEEE, 3 Park Avenue, NY 10016-5997, USA, Oct 2009, E-ISBN : 978-0-7381-6046-7, Print ISBN: 978-0-7381-6047-4.

[3] G. K. Rauwerda and P. M. Heysters and G. J. M. Smit, "Towards Software Defined Radios Using Coarse-Grained Reconfigurable Hardware," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 16, no. 1, pp. 313, Jan. 2008.

[4] R. Airoldi, F. Garzia, O. Anjum, J. Nurmi, "Homogeneous MPSoC as baseband signal processing engine for OFDM systems", International Symposium on System on Chip (SoC), 2010, pp. 26-30, Sept. 2010, doi: 10.1109/ISSOC.2010.5625562.

[5] W. Hussain, F. Garzia, T. Ahonen, J. Nurmi "Designing Fast Fourier Transform Accelerators for Orthogonal Frequency-Division Multiplexing Systems" Journal of Signal Processing Systems, Springer, ISSN 1939-8018, Vol 69, pp 161-171, December, 2012.

[6] C. Brunelli, F. Garzia, and J. Nurmi, "A Coarse-Grain Reconfigurable Architecture for Multimedia Applications Featuring Subword Computation Capabilities", in Journal of Real-Time Image Processing, Springer-Verlag, 2008, 3 (1-2): 21-32. doi:10.1007/s11554-008-0071-3.

[7] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. C. Filho, "Morphosys: An integrated reconfigurable system for data-parallel and computation-intensive applications". IEEE Trans. Computers, vol. 49, no. 5, pp. 465-481, 2000.

[8] B. Mei, S. Vernalde, D. Verkest, H. D. Man, and R. Lauwereins, "ADRES: An architecture with tightly coupled VLIW processor and coarse-grained reconfigurable matrix", Field-Programmable Logic and Applications, vol. 2778, pp. 61-70, September 2003, ISBN 978-3-540-40822-2.

[9] V. Baumgarte, G. Ehlers, F. May, A. Nuckel, M. Vorbach, and M. Weinhardt, "PACT XPP-A Self-Reconfigurable Data Processing Architecture", The Journal of Supercomputing, vol. 26, no. 2, pp. 167-184, September 2003.

[10] F. Garzia, W. Hussain and J. Nurmi, "CREMA, A Coarse-Grain Reconfigurable Array with Mapping Adaptiveness", in Proc. 19th International Conference on Field Programmable Logic and Applications (FPL 2009). Prague, Czech Republic: IEEE, September 2009.

[11] J. Kylliainen, T. Ahonen, and J. Nurmi, "General-purpose embedded processor cores - the COFFEE RISC example", In Processor Design: System-on-Chip Computing for ASICs and FPGAs, J. Nurmi, Ed. Kluwer Academic Publishers / Springer Publishers, June 2007, ch. 5, pp. 83-100, ISBN-10: 1402055293, ISBN-13: 978-1-4020-5529-4.

[12] W. Hussain, X. Chen, G. Ascheid, J. Nurmi, "A Reconfigurable Application-specific Instruction-set Processor for Fast Fourier Transform processing", 2013 IEEE 24th International Conference on Application-Specific Systems, Architectures and Processors (ASAP), pp. 339-345, 5-7 June 2013, Washington, D.C., USA.

[13] O. Anjum, T. Ahonen, J. Nurmi, "MPSoC based on Transport Triggered Architecture for baseband processing of an LTE receiver", Journal of Systems Architecture, Volume 60, Issue 1, January 2014, Pages 140-149, ISSN 1383-7621.

[14] P. Bonnot, F. Lemonnier, G. Edelin, G. Gaillat, O. Ruch, P. Gauget, "Definition and SIMD implementation of a multi-processing architecture approach on FPGA". In Proc. of Design, Automation and Test in Europe (DATE '08). ACM, New York, NY, USA, 610-615.

[15] F. Campi, A. Deledda, M. Pizzotti, L. Ciccarelli, P. Rolandi, C. Mucci, A. Lodi, A. Vitkovski, L. Vanzolini, "A dynamically adaptive DSP for heterogeneous reconfigurable platforms". In Proc. of Design Automation and Test in Europe (DATE '07). EDA Consortium, San Jose, CA, USA, 9-14.

[16] D. Melpignano, L. Benini, E. Flamand, B. Jego, T. Lepley, G. Haugou, F. Clermidy, D. Dutoit, "Platform 2012, a Many-Core Computing Accelerator for Embedded SoCs: Performance Evaluation of Visual Analytics Applications". In Proc. 49th Annual Design Automation Conference (DAC '12). ACM, New York, NY, USA, 1137-1142.

[17] N. S. Voros, M. Hubner, J. Becker, M. Khnle, F. Thomaitiv, A. Grasset, P. Brelet, P. Bonnot, F. Campi, E. Schler, H. Sahlbach, S. Whitty, R. Ernst, E. Billich, C. Tischendorf, U. Heinkel, F. Ieromnimon, D. Kritharidis, A. Schneider, J. Knaeblein, W. Putzke-Rming, "MORPHEUS: A Heterogeneous Dynamically Reconfigurable Platform for Designing Highly Complex Embedded Systems". ACM Trans. Embed. Comput. Syst. 12, 3, Article 70 (April 2013), 33 pages.

[18] T. Ahonen and J. Nurmi, "Hierarchically heterogeneous network-on-chip", in Proceedings of the 2007 International Conference on Computer as a Tool (EUROCON'07), pp. 25802586, IEEE, 9-12 September 2007. ISBN: 978-1-4244-0813-9, DOI:0.1109/EURCON.2007.4400469.

[19] W. Hussain, F. Garzia, and J. Nurmi, "Evaluation of Radix-2 and Radix-4 FFT Processing on a Reconfigurable Platform," in Proceedings of the 13th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'10). IEEE, pp. 249-254, April 2010, ISBN 978-1-4244-6610-8.

[20] W. Hussain, T. Ahonen and J. Nurmi, "Effects of Scaling a Coarse-Grain Reconfigurable Array on Power and Energy Consumption", in Proc. SoC 2012, Tampere, Finland, Oct 2012.

[21] W. Hussain, T. Ahonen F. Garzia and J. Nurmi, "Application-Driven Dimensioning of a Coarse-Grain Reconfigurable Array", in Proc. NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2011), pp. 234-239, 6-9 June 2011, San Diego, California, USA.

Tampereen teknillinen yliopisto
PL 527
33101 Tampere

Tampere University of Technology
P.O.B. 527
FI-33101 Tampere, Finland

ISBN 978-952-15-3406-5
ISSN 1459-2045