ISSN No. 0976-5697

Volume 4, No. 4, March-April 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Low Power Compressed Context Architecture Based Processor Design

P.Sathiya*, Ms.M.Jasmin
PG Scholar*, Assistant Professor
Electronics& Communication Engineering Bharath University
sathiya_ravi2007@yahoo.co.in*, rifriz@gmail.com

Abstract: ACGR A that is focused on data path computations for a particular application domain is a balancing act akin to designing an ASIC and a FPGA simultaneously. Narrowing the application domain significantly makes the design of the CGRA very much like that of a programmable ASIC. Widening the application domain requires a more flexible data path that requires more configurable over head and has less overall efficiency compared to an FPGA. Are configurable architecture issued with compressed context architecture in ALU arrays. There by reducing power in cache (context memory). Architecture presented is replaceable to all processors including DSP processors and power can be reduced.

Keyword: Coarse-grained reconfigurable architecture (CGRA), configuration cache, context architecture, low power

I. INTRODUCTION

An application specific architecture solution is too rigid, and a general purpose processor solution is too inefficient. Neither general purpose processors nor application specific architectures are capable of satisfying the power and flexibility requirements of future mobile devices. Instead, we want to make the machine fit the algorithm, as opposed to making the algorithm fit the machine. This is the area of reconfigurable computing systems.

This paper proposes a scheme of dynamic context Management aiming to minimize the reconfiguration overhead by reusing and switching contexts [2]. The technique permits background loading of configuration data without interrupting the regular execution. It prevents read/write operation for redundant part of context words dynamically and overlaps computation with reconfiguration. And stored configurations can be switched dramatically reducing reconfiguration overhead if the next configuration is present in one of the alternate contexts. In this paper, we address the power reduction issues in CGRA and provide a framework to achieve this[4],[5]. A new design flow and a new configuration cache structure are presented to reduce power consumption in configuration cache.

The power saving is achieved by dynamic context compression in the configuration cache—only required bits of the context words are set to enable and the redundant bits are set to disable. Therefore, the new design flow for CGRA has been proposed to generate architecture specifications that are required for supporting dynamically compressible context architecture without performance degradation. The validation of the proposed approaches is demonstrated through the use of real application benchmarks and gate level simulations[2]. The proposed CGRA architecture has shown to reduce power consumption by up to 39.72% in configuration cache compared to conventional context architecture with negligible area overhead of only 2.16%.

II. FINE GRAIN VS. COARSE GRAIN

Reconfigurable processors have been widely associated with Field Programmable Gate Array (FPGA)-based system designs[1]. An FPGA consists of a matrix of programmable logic cells with a grid of interconnecting lines running between them. In addition, there are I/O pins on the perimeter that provide an interface between the FPGA, the interconnecting lines and the chip's external pins. However, FPGAs tend to be somewhat fine-grained in order to achieve a high degree of flexibility. This flexibility has its place for situations where the computational requirements are either not known in advance or vary considerably among the needed applications. However, in many cases this extreme level of flexibility is unnecessary and would result in significant overheads of area, delay and power consumption.

Contrasted with FPGAs, the data-path width of coarse grained reconfigurable architectures is more than one bit. Over the last 15 years, many projects have investigated and successfully built systems where the reconfiguration is coarse-grained and is performed within a processor or amongst processors. In such systems the reconfigurable unit is a Specialized hardware architecture that supports logic reconfiguration. The reconfiguration procedure is much faster than that found in FPGAs[8]. Because the application domain is known, full custom data paths could be designed, which are drastically more area-efficient. Typically, a CGRA consists of a main processor, reconfigurable array architecture (RAA)[4][5], and their interface, as shown in Fig. 1.1. The RAA has identical PEs containing functional units and a few storage units such as ALU, multiplier, shifter, and register file. The data buffer provides operand data to PE array through a high-bandwidth data bus. The configuration cache (or context memory) stores the context words used for configuring the PE array elements. The context register between a PE and a cache element (CE) in configuration cache is used to keep the cache access path from being the critical path of the CGRA[7].

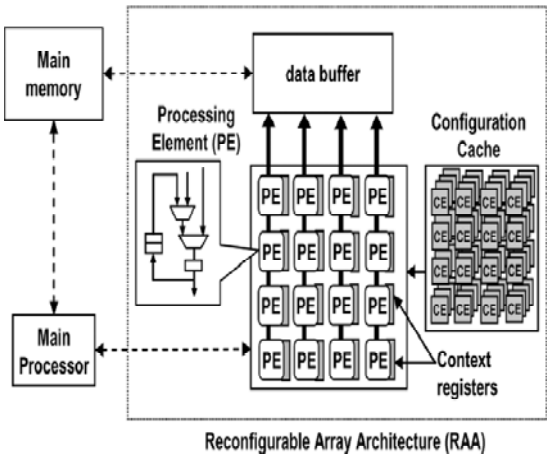

Figure 2.1 General block diagram of CGRA

A. Power Advantages In CGRA:

Studies of the power consumption within an FPGA, as shown by Poor identify that 50% to 60% of the energy of an FPGA is dissipated in the routing fabric, 20% to 40% in the logic blocks and 5% to 40% in the clock network. Using these number as a guideline, we see that the interconnect offers the greatest chance for energy optimization, followed by the logic blocks. One key aspect in the transition from fine-grained to coarse-grained configurable architectures is the overhead associated with configuration control logic.

Therefore, improvements in the CGRA will be measured indirectly at the system level. By abstracting the effects of a given silicon manufacturing technology, custom layout, or even the specifics of implementing a power efficient arithmetic unit we can focus on architectural features that provide a significant power advantage. The primary source of architectural power efficiency is the transition from a bitwise to word-wide datapath, which can be further refined into categories for interconnect, arithmetic and logic, and configuration overhead.

Some options for improving the efficiency of the interconnect within the data plane are:

a) Reducing the number of pair-wise routing options, i.e. connectivity.
b) Reducing interconnect switching complexity by bundling wires into buses.
c) Reducing the average interconnect length by inserting pipelining registers.
d) Using a more power efficient numerical representation, such as signed magnitude or Gray code.

Some options for improving the efficiency of the arithmetic and logic resources within the data plane are:

a) Using dedicated, atomic, resources for computation.
b) Pipelining arithmetic units.
c) Reduced resources for intra-word communication in arithmetic units, e.g. carry-chains or conditional flags.

Reduction in the overhead of the configuration logic for the data plane can be achieved by:

a) Configuring buses rather than individual wires.
b) Sharing configuration bits across larger compute elements.
c) An overall reduction in total number of possible configurations and thus a reduction in the configuration state size.

Making an architecture more coarse-grained means that computation is done via dedicated adders, multipliers, etc. rather than constructing such units from more primitive functional units such as a 4-LUT. While a dedicated arithmetic unit is guaranteed to be more efficient than one composed of 4-LUTs, finding a balance of dedicated resources that match the flexibility of the 4-LUTs is difficult, and the overhead of multiple dedicated resources could easily outweigh the individual advantages of each resource. One common use for CGRAs is to accelerate the computationally intensive kernel(s) of a larger application.

These kernels are typically inner loops of algorithms or a set of nested loops. Frequently, it is possible to pipeline these kernels, thus exploiting the application's existing parallelism and increasing its performance. If the application domain of a CGRA is rich with pipelinable kernels then it is advantageous to have a general bias towards a specific flow of computation and data in the datapath and to include dedicated resources for pipelining in the interconnect. The control plane of a CGRA plays a similar role to the general purpose spatial computing fabric of an FPGA. It is composed of bitwise logic and communication resources, is flexible and highly connected. Given these requirements, it is likely that the control plane's architecture will be similar to a standard FPGA's architecture. From a power efficiency standpoint, the control plane will perform similarly, and thus provide no advantage over a standard FPGA. However, the control plane will be only one portion of a CGRA making its contribution to the energy overhead smaller

III. CONTEXT ARCHITECTURE

The configuration cache provides context words to the context register of each PE on a cycle-by-cycle basis. From the context register, these context words configure the PEs. Fig. 2 shows an example of PE structure and context architecture for MorphoSys [2]. Thirty-two bit context word specifies the function for the ALU-multiplier, the inputs to be selected from MUX_A and MUX_B, the amount and direction of shift of the ALU output, and the register for storing the result as Fig. 2(a). Context architecture means organization of context word with several fields to control resources in a PE, as shown in Fig. 2(b). The context architectures of other CGRAs such as are similar to the case of MorphoSys although there is a wide variance in context width and kind of fields used by different functionality. For particular coarse-grained reconfigurable array, the performance improvement depends on the inherent parallelism of target applications. In many cases, the parallelism of an application is smaller than the number of processing element (PE) arrays. Accordingly, PE array shows lots of redundant PEs not used at runtime.

If the configuration cache can provide only required bits (valid bits) of the context words to PE array at runtime, it is possible to reduce power consumption in configuration cache. The redundant bits of the context words can be set to disable and make these invalid at runtime. That way, one can achieve low-power implementation of CGRA without performance degradation while context architecture dynamically supports both the cases at runtime: one case is uncompressed context word with full bit width and another case is compressed context word with setting unused part of configuration cache disabled. In order to support such a

dynamic context compression, we propose a new context architecture and configuration cache structure in this paper.

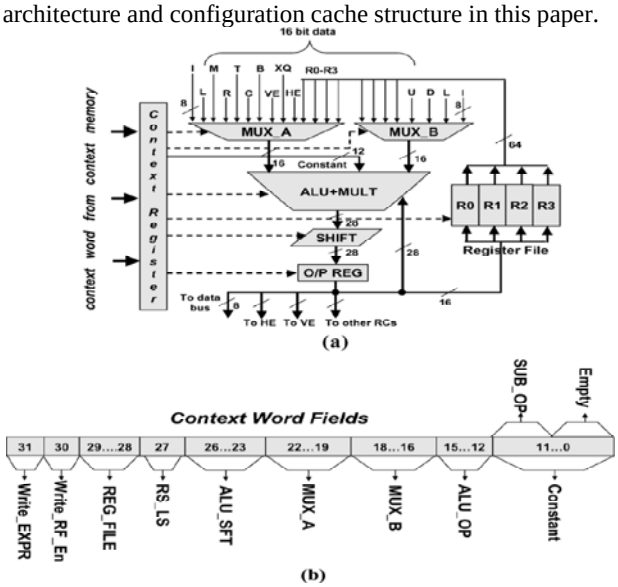

Figure. 3.1. PE structure and context architecture of MorphoSys. (a) PE structure. (b) Context architecture.

IV. PROPOSED SYSTEM

A. Compressed ALU Architecture:

In order to design and evaluate compressible context architecture, we propose a new context architecture design flow. This design starts from context architecture initialization, which is similar to the architecture specification stage of general CGRA design flow.

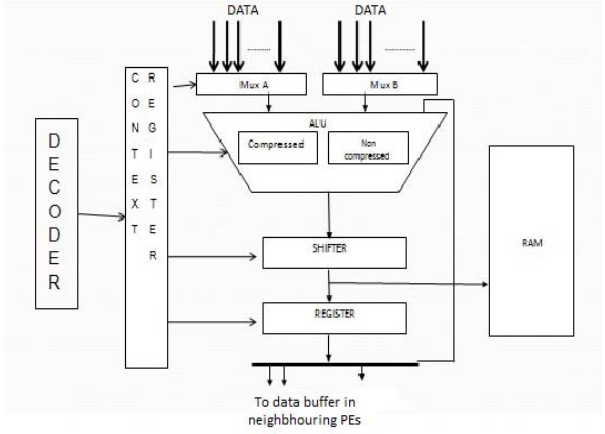

Figure 4.1. Compressed ALU Architecture

If the configuration cache can provide only required bits (valid bits) of the context words to PE array at runtime, it is possible to reduce power consumption in configuration cache [3]. The redundant bits of the context words can be set to disable and make these invalid at runtime. That way, one can achieve low-power implementation of CGRA without performance degradation. One more method to reduce power is by dividing the ALU into two parts that's in compressed and non-compressed method. The normally repeating instructions are executed in compressed mode and instructions that are frequently not used such as branching instructions are executed in non-compressed mode. The normal instructions are add, sub etc are executed in compressed mode.

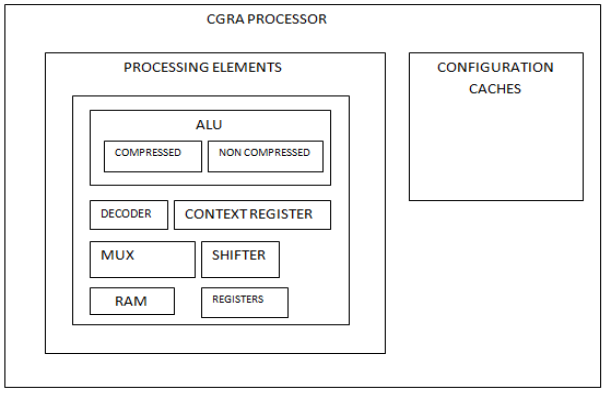

Figure 4.2. Overall Block Diagram

In compressed mode we use a carry look ahead adder for execution the normal instructions. The use of carry look ahead adder is helpful in increasing the speed and saving power than that of ripple carry adder. Also it reduces area than the ripple carry adder.

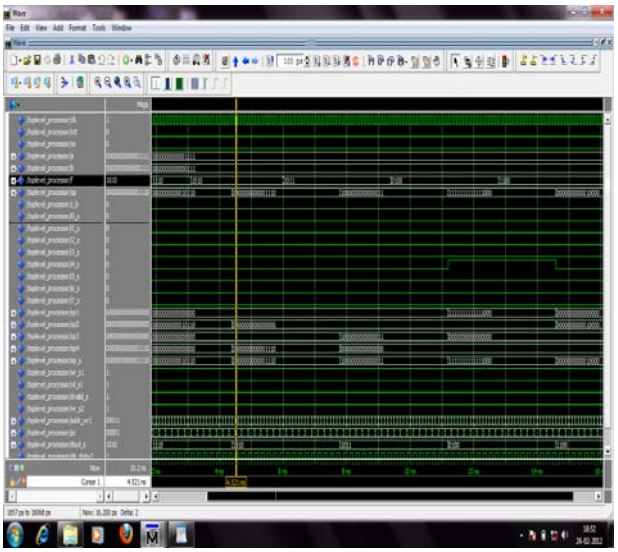

Figure 4.3 and4.4 Shows the output obtained using the logic based on the compressed ALU idea explained in this paper.

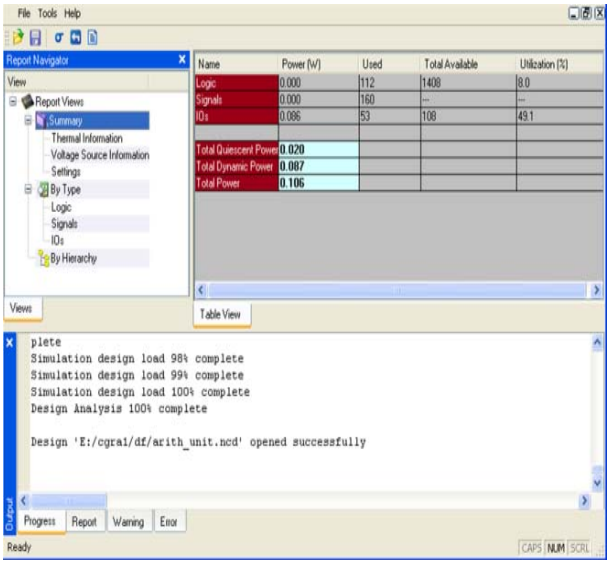

Figure 4.4.Output

V. CONCLUSION

Power consumption is very crucial for the CGRA for embedded systems and all reconfigurable architectures have a configuration cache for dynamic reconfiguration, which consumes significant amount of power. In this paper, we introduced new context architecture with its design flow. We are compressing the architecture in each instruction cycle from the decoding section of the processing of a instructions. The exploration flow efficiently rearranges PEs with reducing array size, and changes interconnection scheme to save area and power. In addition, we suggest the design scheme which splits the computational resources into two groups (primitive resources and critical resources). Primitive resources are replicated for each processing element of the reconfigurable array, whereas area-critical resources are shared among multiple basic PEs.

VI. REFERENCES

[1]. H. Reiner, “A decade of reconfigurable computing: A visionaryretrospective,” in Proc. Des. Autom. Test Eur. Conf., Mar. 2001, pp. 642–649.

[2]. H. Reiner, M. Herz, T. Hoffmann, and U. Nageldinger, “KressArrayXplorer: A new CAD environment to optimize reconfigurable datapath array architectures,” in Proc. Asia South Pacific Des. Autom. Conf., Jan. 2000, pp. 163–168.

[3]. B. Mei, S. Vernalde, D. Verkest, and R. Lauwereins, “Design methodology for a tightly coupled VLIW/reconfigurable matrix architecture: A case study,” in Proc. Des. Autom. Test Eur. Conf., Mar. 2004, pp. 1224–1229.

[4]. N. Bansal, S. Gupta, N. Dutt, and A. Nicolau, “Analysis of the performance of coarse-grain reconfigurable architectures with different processing element configurations,” presented at theWorkshop Appl. Specific Processors, San Diego, CA, Dec. 2003.

[5]. A. Lambrechts, P. Raghavan, and M. Jayapala, “Energy-aware interconnect- exploration of coarse-grained reconfigurable processors,” presented at the Workshop Appl. Specific Processors, New York, Sep.2005.

[6]. R.Hartenstein. A decade of reconfigurable computing: A visionary retrospective. In Proc. DATE, pages 642-649, 2001.

[7]. Hideharu Amano, Yohei Hasegawa, Satoshi Tsutsumi "MuCCRA chips: Configurable Dynamically-Reconfigurable Processors" IEEE Asian Solid-State Circuits Conference November 12-14,2007 / Jeju, Korea [3] Sudang Yu, LeibaLiu,"Automatic Contexts Switch in Loop Pipeline for Embedded Coarse-grained Reconfigurable Processor" Communications, Circuits and Systems, 2008, ICCCAS 2008, International Conference.

[8]. F.Veredas,M.Scheppler,W.Moffat, and B.Mei, "Custom Implementation of the Coarse-Grained Reconfigurable ADRES Architecture for Multimedia Purposes," in Proc. Of FPL, Aug. 2005, pp. 106-111.