Architecture design of reconfigurable accelerators for demanding applications.

Citation for published version (APA):

Document status and date:
Published: 01/01/2010

Document Version:
Publisher’s PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher’s website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the “Taverne” license above, please follow below link for the End User Agreement:
www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:
openaccess@tue.nl
providing details and we will investigate your claim.
Abstract—This paper focuses on mastering the architecture development of reconfigurable hardware accelerators for highly demanding applications. It presents the results of our analysis of the main issues that have to be addressed when designing accelerators for demanding applications, when using as an example the accelerator design for LDPC decoding for the newest communication system standards. Based on the results of our analysis, we formulate the main requirements that have to be satisfied by an adequate methodology of reconfigurable accelerator design for highly demanding applications, and propose an architecture design methodology which satisfies these requirements.

Keywords— reconfigurable accelerators; advanced applications; architecture design; design-space exploration;

I. ISSUES AND REQUIREMENTS OF ACCELERATOR DESIGN FOR DEMANDING APPLICATIONS

Hardware acceleration of critical computations has been intensively researched during the last decade, mainly for signal, video and image processing applications, for efficiently implementing transforms, filters and similar complex operations [1][3]-[9]. Common features of these operations include functional parallelism and relatively simple and regular, limited in space and time local memory accesses between which relatively large portions of computations are performed. In consequence, the main problems of their acceleration are related to effective and efficient processing unit synthesis through an adequate functional parallelism exploitation of the register transfer level (RTL) operations needed for implementation of the required computations. The micro-architecture design for such accelerators can reasonably well be supported by the methods of high-level synthesis [1][3]-[9] and emerging commercial high-level synthesis tools [10].

However, many modern applications (e.g. various decoders in (wireless) communication and multimedia, network access nodes, encryption applications, etc.) require hardware acceleration of algorithms that involve complex interrelationships between the data and computing operations. For applications of this kind, the main design problems are related to an adequate resolution of memory and communication bottlenecks and to decreasing of the memory and communication hardware complexity, what has to be achieved through an adequate memory and communication structure design. Moreover, the memory and communication structure design, and micro-architecture design for computing units cannot be performed independently, because they substantially influence each other. For example, exploitation of more data parallelism in a computing unit micro-architecture usually requires getting the data in parallel for processing, i.e. having simultaneous access to memories in which the data reside and simultaneous transmission of the data, or pre-fetching the data in parallel to performing other computations. For applications of this kind complex interrelationships exist between the computing unit design and corresponding memory and communication structure design, and complex tradeoffs have to be resolved between the accelerator effectiveness (e.g. computation speed or throughput) and efficiency (e.g. hardware complexity, power and energy consumption etc.).

One of important application classes of this kind is the low-density parity-check code (LDPC) decoding of the newest communication system standards like IEEE 802.11n, 802.16e, 802.15.3c, 802.3an, etc., for digital TV broadcasting, mm-wave WPAN, etc. This application class will be used in this paper to illustrate the issues and requirements of reconfigurable accelerator design for demanding applications, and to introduce and illustrate our accelerator design methodology.

A systematic LDPC encoder encodes a message of \( k \) information bits into a codeword of length \( n \) with the \( k \) message bits followed by \( m \) parity checks. Each parity check is computed based on a sub-set of message bits. The codeword is transmitted through a communication channel to a decoder. The decoder checks the validity of the received codeword by re-computing the parity checks, using a parity check matrix (PCM) \( H \) of size \( m \times n \). To be valid, a codeword must satisfy the set of all \( m \) parity checks. In Figure 1 an example PCM for a \((7, 4)\) LDPC code is given. \( 1 \) in a position \( H_{i,j} \) of this matrix means that a particular bit participates in a parity check equation. Each parity check matrix can be represented by its corresponding bipartite Tanner graph [11]. The Tanner graph corresponding to an \((n, k)\) LDPC code consists of \( n \) variable (bit) nodes (VN) and \( m = n - k \) check nodes (CN), connected with each other through edges, as shown in Figure 1. Each row in the parity check matrix represents a parity check equation \( c_i, 0 \leq i \leq m-1 \), and each column represents a code bit \( \tilde{h}_j, 0 \leq j \leq n-1 \). An edge exists between a CN \( i \) and VN \( j \), if the corresponding value \( H_{i,j} \) is non-zero in the PCM.
Usually, iterative Message Passing Algorithms (MPA) [12] are used for decoding the LDPC codes. During decoding specific messages are exchanged among the nodes through the edges. The messages represent the log-likelihood ratios (LLRs) of the codeword bits based on the channel observations [12]. The algorithm starts with the so-called intrinsic LLRs of the received symbols based on the channel observations. Starting with the intrinsic LLR values, the algorithm iteratively updates the extrinsic LLR messages from the check nodes to variable nodes and from the variable nodes to check nodes and sends them among the VNs and CNs along the corresponding Tanner graph edges. If after several iterations the parity check equation is satisfied, the decoding stops, and the decoded codeword is created and considered to be a valid codeword. Otherwise, the algorithm further iterates until a given maximum number of iterations is reached. Since Tanner graphs corresponding to practical LDPC codes of the newest communication system standards involve hundreds of variable and check nodes, and even more edges, LDPC decoding represents a massive computation and communication task. Moreover, the modern communication system standards require very high throughput in the range of Gbps and above, for applications like digital TV broadcasting, mmWave WPAN, etc. For the realization of the so high throughput complex highly parallel hardware accelerators are necessary.

Furthermore, many modern applications involve algorithms with massive data parallelism at the macro-level or task-level functional parallelism. To adequately serve these applications, hardware accelerators with parallel multi-processor macro-architectures have to be considered, involving several identical or different concurrently working hardware processors, each operating on a different data subset. Each of these processors can also be more or less parallel. Moreover, there is a trade-off between the amount of parallelism and resources at the macro-architecture and micro-architecture level (e.g. similar performance can be achieved with less processors each being more parallel or better targeted to particular part of application, as with more processors each being less parallel or less application-specific). The two architecture levels are strongly interrelated and interwoven, also through their relationships with the memory and interconnection structures. In consequence, optimization of the performance/resources trade-off required by a particular application can only be achieved through a careful construction of an adequate application-specific macro-/micro-architecture combination.

For instance, in LDPC decoding each Tanner graph node has several inputs and the basic node operations are multi-input. In the corresponding accelerator, the spectrum of possible implementations of each of these multi-input operations spans between the two extremes of a fully serial to a fully parallel. When the variable nodes perform their computations the check nodes are waiting on the computation results and vice versa, but all nodes of a given kind, i.e. all the variable nodes or all the check nodes, may perform their computations in parallel. If all the nodes of a given kind would actually perform their computations simultaneously, this would require a complex parallel access to the memories of all nodes of the opposite kind, and could only be realized with a very distributed memory structure and very complex and expensive interconnection structure. In contrary, performing the computations corresponding to different nodes fully serially can require just one memory access at a time and result in reasonably simple corresponding memory and interconnection structures. Summing up, in the hardware accelerators for LDPC decoding, the possible micro-architectures span the full spectrum from a fully serial to a fully parallel, and the possible macro-architectures of the multi-accelerator structures span the full spectrum from a fully serial [13] to a fully parallel [14], with large variety of partially parallel architectures between them (e.g. [15]-[17]). Also, complex tradeoffs are possible between the parallelism and resources at the micro-architecture level, and parallelism and resources at the macro-architecture level. Moreover, changing the parallelism for computations in the micro- or macro-architecture of the LDPC accelerator requires a corresponding change of the memory and communication structure. The data processed in parallel must also be accessible in parallel. Thus, the computation, memory and communication architectures are strictly interrelated and cannot be designed in separation. Moreover, a large number of possible macro-architecture/micro-architecture combinations and related node mappings possible leads to a large number of various tradeoff points in the LDPC accelerator design space representing various accelerator architectures with different characteristics.

Finally, different applications adopt different algorithms and have different requirements in relation to throughput, area and other parameters. For instance, the above mentioned new communication system standards adopt different LDPC codes classes and decoding

![Figure 1. An example PCM for a (7,4) LDPC code and its corresponding Tanner graph](image)

\{b_0...b_6\} represents variable (bit) nodes, \{c_0...c_2\} represents check nodes, and \{l_0...l_6\} represents the input intrinsic channel information.
algorithms and have different requirements regarding code rate, code length, throughput etc. As it is not known in advance which of the proposed standards will actually be accepted, and due to profitability it is extremely important for the industry to have their equipment for a new standard ready before the standard is actually accepted, accelerators for applications involving new standards have to be multi-standard, i.e. adaptable for a particular standard after being designed and/or fabricated. Their adaptability can be achieved through design-time adaptation and/or post-production (field-use) adaptation. In particular, the field-use adaptation can have a form of a run-time reconfiguration.

An additional advantage of a run-time reconfigurable accelerator is its ability to dynamically adapt to changing operating conditions, e.g. it can adapt its structure and operation to different quality-of-service, energy usage and transmission speed requirements, noise levels and other environmental conditions.

From the above it should be clear, that the existing high-level synthesis, specifically developed and limited to RTL-level micro-architecture synthesis, is only able to partly support the internal architecture design for particular computation units, and is not sufficient to adequately support the total complex architecture design process of reconfigurable accelerators for the modern demanding applications. A new more complex and sophisticated design methodology is needed. In parallel to accounting for the computation unit micro-architecture synthesis, this new accelerator design methodology should to adequately address many more issues, including:

- memory and communication structure synthesis,
- macro-architecture synthesis of the multi-accelerator structures,
- strong interrelationships between the computation unit, memory and communication organization, and between the micro-and macro-architecture,
- tradeoff exploitation between the micro and macro-architecture, and between the various aspects of the accelerator's effectiveness and efficiency, and
- adaptable accelerator design accounting for the design-time and field-use adaptation.

To implement the field-use (run-time) adaptation the popular reconfigurable FPGA technology could potentially be used. In FPGAs the reconfiguration resources and mechanisms are pre-implemented, and they guarantee a complete total reconfiguration ability. In consequence, when developing a reconfigurable accelerator for an FPGA implementation, a trivial approach can be used of separately developing particular accelerators for different acceleration cases (e.g. for different LDPC codes and corresponding requirements) and totally (or partially when applicable) reconfiguring the FPGA to implement each of the accelerators. However, due to the extremely high throughput and/or low energy consumption requirements of many modern demanding applications, reconfigurable accelerators for such applications cannot usually be implemented using FPGAs. Due to the high overhead of their general reconfiguration resources, FPGAs are unable to deliver so high throughput and are not energy efficient. Therefore, reconfigurable accelerators for highly demanding applications require a high-performance energy-efficient application-specific integrated circuit (ASIC) implementation. However, in the case of an ASIC implementation, the complexity and other features of the total computation, memory, communication and reconfiguration resources supporting acceleration for all the required acceleration cases are of crucial importance, and therefore, the trivial development approach cannot be used. For example, an optimal architecture for a set of requirements related to an individual code, being the best choice for an individual accelerator construction for this particular code, is usually not the best choice if we have to construct a reconfigurable accelerator architecture supporting several different codes and their related requirements. For instance, the code class 7/8 of the IEEE 802.15.3c standard with the highest check node degree (number of inputs) of 32 can be efficiently realized using serial micro-architecture, while the code class 1/2 could be more efficiently realized in a parallel micro-architecture. Since the two architectures have not much in common, their joint reconfigurable implementation as an ASIC would involve extensive computation, communication and reconfiguration resources, and would not be efficient. The goal of the reconfigurable accelerator design space exploration should be therefore to decide one globally optimal adaptable accelerator architecture for all the acceleration cases (e.g. LDPC decoders) required, and not a set of the best individual accelerators for each particular acceleration case and their combined reconfigurable implementation. Moreover, during the design-space exploration reconfiguration resources have to be accounted for. However, not the general total reconfiguration resources as in FPGAs should be aimed at, but limited efficient design-case-specific reconfiguration resources.

II. ACCELERATOR DESIGN METHODOLOGY

In this section we propose a design methodology for reconfigurable accelerators which addresses the issues and satisfies the requirements of accelerator design for demanding applications, as considered in Section I. From the discussion in Section 1 it follows that a sophisticated design space exploration of accelerator architectures is necessary to arrive at high-quality accelerator designs, in which some of the most promising architectures have to be efficiently constructed and analyzed, and the best of these architectures have to be selected for further analysis, refinement and actual implementation.

In our recent paper [18] an accelerator architecture exploration method was proposed for non-reconfigurable accelerators. This paper extends the method for reconfigurable accelerators. Our accelerator design method in based on the quality-driven model-based design paradigm proposed by the first author of this paper [2]. According to this paradigm, system design is actually about a definition of
the required quality, in the sense of a satisfactory answer to the questions: what quality is required and how can it be achieved? To bring the quality-driven design into effect, quality has to be modeled, measured and compared. In our approach, the quality of the accelerator required is modeled in the form of the demanded accelerator behavior, and structural and parametric constraints and objectives to be satisfied by its design, as described in [2][18]. Our approach exploits the concept of a pre-designed generic architecture platform, which is modeled as abstract generic architecture template (e.g. Fig. 2). Based on the analysis results of the so modeled required quality, the generic architecture template is adequately instantiated and used to design space exploration that aims at analysis of various architectural choices and macro-/micro-architecture tradeoffs, and finally, at the construction of one or several most promising accelerator architectures supporting the required behavior and satisfying the demanded constraints and objectives. Our approach considers the macro-architecture and micro-architecture synthesis and optimization, as well as, the computing, memory and communication structures’ synthesis as one coherent complex accelerator architecture synthesis and optimization task, and not as several separate tasks, as in the state-of-the-art methods. This allows for an adequate resolution of the strong interrelationships between the micro- and macro-architecture, and computation unit, memory and communication organization, as well as, for an effective tradeoff exploitation between the micro- and macro-architecture, and between the various aspects of accelerator’s effectiveness and efficiency. Moreover, our design-space exploration of reconfigurable accelerator architectures aims at finding one globally optimal adaptable accelerator architecture that adequately satisfies the requirements for all the particular accelerator instances required for a given application. According to our knowledge, the so formulated accelerator design problem is not yet explored in any of the previous works related to hardware accelerator design.

The exploration of promising architecture designs is performed as follows (see Fig. 3). For a given class of applications, a pool of generic architecture templates, including their corresponding processing units, memory units and other architectural resources, is prepared in advance by analyzing various applications of this class, and particularly, analyzing the applications’ required behavior, and ranges of their structural and parametric demands. Each generic architecture template specifies several general aspects of the modeled architecture set, such as presence of certain module types and the possibilities of the modules’ structural composition, and leaves other aspects (e.g. the number of modules of each type or their specific structural composition) to be derived through the design space exploration in which a template is adapted for a particular application. In fact, the generic templates represent generic conceptual architecture designs which become actual designs after adequate further template instantiation, refinement and optimization. The adaptation of a generic architecture template to a particular application with its particular set of behavioral and other requirements consists of the design space exploration through performing the most promising instantiations of the most promising generic templates and their resources to implement the required behavior, when satisfying the remaining application requirements. In result, several most promising architectures are designed and selected that match the requirements of application under consideration to a satisfactory degree.

During the design space exploration two major aspects of the accelerator design, being its macro-architecture and micro-architecture, are considered and decided, as well as, the tradeoffs between these two aspects in relation to the design quality metrics (such as throughput, area, energy consumed, cost etc.). It is important to stress that these macro- and micro-architecture decisions are taken in combination, because both the macro- and micro-architecture decisions influence the throughput, area, and other important parameters, but they do it in different ways and to different degrees. For instance, by a limited area, one can use more elementary accelerators, but with less parallel processing and related hardware in each of them, or vice versa, and this can result in a different throughput and different values of other parameters for each of the alternatives. To decide the most suitable architecture, the promising architectures constructed during the design space exploration are analyzed in relation to the quality metrics of

![Fig. 2: Example of a generic architecture template for reconfigurable LDPC decoding accelerators](image-url)
interest and basic controllable system attributes affecting them (e.g. number of accelerator modules of each kind, clock frequency of each module, communication structures between modules, schedule and binding of the required behavior to the modules etc.), and the results of this analysis are compared to the design constraints and optimization objectives. This way the designer receives feedback, composed of a set of instantiated architectures and important characteristics of each of the architectures, showing to what degrees the particular design objectives and constraints are satisfied by each of them. If some of the constraints cannot be satisfied for a particular application through instantiation of given templates and their modules, new more effective modules or templates can be designed to satisfy the stringent requirements, or the requirements can be reconsidered and possibly lowered. Subsequently, the next iteration of the design space exploration can be started. If all the constraints and objectives are met to a satisfactory degree, the corresponding final application specific architecture template is instantiated, further analyzed and refined to represent the actual detailed design of the required accelerator.

Our reconfigurable accelerator design space exploration aims at deciding one globally optimal adaptable accelerator architecture that adequately satisfies the requirements for all the particular accelerator instances required for a given application. In case of the LDPC decoding, this corresponds to finding one most promising reconfigurable accelerator architecture to adequately satisfy the combined requirements of all the codes for a particular application, and not a set of the best individual accelerators for each code separately and their combined reconfigurable implementation.

Using the exploration approach described above, it first determines suitable accelerator architectures for the particular required acceleration cases that are most demanding in terms of throughput, area etc. (i.e. suitable decoding accelerators for the particular most demanding code classes in case of LDPC). This results in finding a subset of suitable architectures for each of the most demanding acceleration cases that satisfy hard constraints. The architectures determine alternative sets of the necessary resources and configurations of the future reconfigurable accelerator that are able to satisfy the requirements of the most demanding acceleration cases – i.e. the resources and their configuration that must be present in the reconfigurable acceleration system under construction. Subsequently, some sub-sets of promising architectures for the less demanding acceleration instances (codes) are explored and analyzed to determine similarity in their structure to the most demanding accelerator architectures. Exploitation of the architecture similarity for different

![Diagram](image-url)

**Fig. 3:** Architecture exploration framework for reconfigurable accelerator design
acceleration cases required for a given application is of crucial importance for an adequate reconfigurable accelerator construction. It is necessary to efficiently re-use the processing, communication and memory resources for different required acceleration cases, and to reduce the re-configuration resources. Finally, knowing the necessary resources and the architecture similarity for different required acceleration cases, our reconfigurable accelerator architecture construction adequately adapts the accelerator architecture of the most demanding case(s), or adapts and combines the accelerator architecture of the most demanding case(s) and several other individual accelerator architectures in such a way, as to realize all the required acceleration cases (decoders for all the required codes) and their particular requirements, but at the same time to minimize the total processing, memory, communication and reconfiguration resources, and the related power consumption, costs etc. As explained in the previous section, for ASIC-implemented reconfigurable accelerators the total reconfiguration when using very general reconfiguration resources as in FPGAs is impractical. Therefore, to instantiate a particular accelerator we aim to (almost) only reconfigure the interconnections between the basic processing units to form larger processing units, and between the processing units and memories, and to avoid the low-level reconfiguration inside of basic processing units (e.g. to form the basic processing units from gates). For the LDPC codes this is possible due to the strong similarities in the code structures, decoding algorithms and operations required for different code rates, code lengths and other requirements. Due to the reconfiguration limited to the interconnect configuration at the level of basic functional units, the reconfiguration resources remain limited, and the reconfiguration overhead in the form of the configuration controller and configuration information remains low.

Based on the information from the design-time decisions, during the run-time reconfiguration the configuration controller (see Fig.2):
- re-connects the basic processing units to form larger processing units;
- re-organizes the memory through forming the required number of memory modules with the required number of ports from the elementary memory modules;
- assigns particular parts of the required computations to particular processing units and memories (e.g. in the LDPC case assigns particular parts of the PCM to particular processing units (CNP and VNP) and memories), and
- configures the corresponding interconnections between the processing units and memories.

III. CONCLUSION

This paper presented the results of our analysis of the main problems that have to be solved in design of reconfigurable accelerators for modern demanding applications, formulated the main requirements that have to be satisfied by an adequate methodology of reconfigurable accelerator design for such applications, and proposed a quality-driven model-based design methodology for reconfigurable accelerators which satisfies the requirements. We are currently applying the methodology to the design of reconfigurable hardware accelerators for LDPC decoding for some of the newest demanding communication system standards.

REFERENCES