Journal of Systems Architecture 000 (2016) 1–11

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Parallel custom instruction identification for extensible processors

Chenglong Xiao[a,*], Shanshan Wang[a], Wanjun Liu[a], Emmanuel Casseau[b]

[a] Liaoning Technical University, China
[b] University of Rennes I, Irisa, Inria, France

ARTICLE INFO

Article history:
Received 27 January 2016
Revised 15 August 2016
Accepted 18 November 2016
Available online xxx

Keywords:
Custom instruction
Subgraph enumeration algorithm
Subgraph selection algorithm
Parallel algorithms
Extensible processors

ABSTRACT

With the ability of customization for an application domain, extensible processors have been used more and more in embedded systems in recent years. Extensible processors customize an application domain by executing parts of application code in hardware instead of software. Determining parts of application code as custom instruction generally requires subgraph enumeration and subgraph selection. Both subgraph enumeration problem and subgraph selection problem are computationally difficult problems. Most of previous works focus on sequential algorithms for these two problems. In this paper, we present a parallel implementation of a latest subgraph enumeration algorithm based on a computer cluster. A standard ant colony optimization algorithm (ACO), a modified version of ACO with local optimum search and a parallel ACO algorithm are also proposed to solve the subgraph selection problem in this work. Experimental results show that the parallel algorithms outperform the sequential algorithms in terms of runtime or (and) quality of results. In addition, we have formally proved the upper bound on the number of feasible solutions in subgraph selection problem with or without the overlapping constraint.

1. Introduction

In last decade, extensible processors have been used more and more in embedded systems. In extensible processors, the flexibility is guaranteed by the general purpose processor while the runtime performance is achieved by executing segments of code in custom functional units (CFUs) that impose high data-level parallelism [1]. The cluster of primitive instructions executing in CFU is formally called as custom instruction. Nowadays, it is very common to find commercial reconfigurable processors in embedded systems, e.g. Altera NIOS processors, Xilinx MicroBlaze and ARC processors that can implement such behavior.

The key problems involved in deciding the segments of code to be executed in custom functional units are: subgraph enumeration problem and subgraph selection problem. Both subgraph enumeration problem and subgraph selection problem are computationally difficult problems [2–5]. Parallel algorithms can speed up the search by taking benefit of using several computing elements. In this work, we propose parallel implementations for subgraph enumeration and subgraph selection. Fig. 1 shows the design flow of parallel identification of custom instructions. The data-flow graphs (DFGs) of basic blocks in a given application are firstly generated by a front-end compiler. Based on the DFGs, the parallel subgraph enumeration step tries to concurrently enumerate all possible subgraphs (graphic representation of custom instructions) that satisfy the micro-architecture constraints and user-defined constraints using a cluster of computers. The parallel subgraph selection step aims at concurrently finding an optimal subset of enumerated subgraphs as custom instructions such that the execution time of the given application is minimized (or the number of distinct patterns is minimized, etc.). Finally, the selected custom instructions are coded in the new functionally equivalent code. In this paper, we focus on the two crucial problems involved in the design flow: subgraph enumeration problem and subgraph selection problem.

This paper makes the following three contributions:

- a parallel implementation of a latest sequential algorithm for subgraph enumeration problem is proposed [5];
- a standard sequential ant colony optimization (ACO) algorithm and a modified version of standard ACO with local optimum search are adapted to the subgraph selection problem, a parallel ACO algorithm that can achieve higher quality of results in shorter time is also presented;
- the upper bounds on the number of feasible solutions for the subgraph selection problem with or without overlapping constraint are formally proved.

The rest of the paper is organized as follows. In Section 2, the state of the art is introduced. Section 3 formulates the subgraph

* Corresponding author.
E-mail addresses: chenglong.xiao@gmail.com (C. Xiao), celine.shanshan.wang@gmail.com (S. Wang), liuwanjun39@163.com (W. Liu), emmanuel.casseau@irisa.fr (E. Casseau).

http://dx.doi.org/10.1016/j.sysarc.2016.11.011
1383-7621/

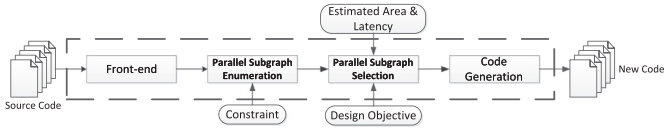

Fig. 1. Parallel custom instruction identification flow.

Table 1
The complexity of subgraph enumeration problem.

Type of subgraphs	Upper bound on the number of subgraphs
MIMO	$\|G\|^{IN_{max}}\|G\|^{OUT_{max}}$
MaxMIMO	$2^{\|V_{in}\|}$
CC-Subgraphs	$2^n + n + 1 - d_n$

enumeration problem and introduces a parallel algorithm implemented in this work. In Section 4, the subgraph selection problem is depicted, the upper bound on the number of feasible solutions for the subgraph selection problem is proved and a sequential ACO algorithm and a parallel ACO algorithm are proposed. Section 5 evaluates and compares the runtime and quality of results of the proposed algorithms. Finally, conclusions are presented in Section 6.

2. Related work

The key issues involved in the design flow of custom instruction identification are subgraph enumeration and subgraph selection. In recent years, plenty of works have been done for subgraph enumeration problem and subgraph selection problem. In the survey of the two problems, we organize them in terms of constraints.

2.1. Subgraph enumeration

Subgraph enumeration is the process to enumerate all the subgraphs that satisfy certain design constraints from a given application graph. It is a computationally difficult problem. The number of subgraphs in the DFG of a given application could be 2^n, where n is the number of nodes in the DFG [6]. Since the number of subgraphs is exponential with respect to the size of the DFG, efficient approaches to the subgraph enumeration problem are necessary. In order to reduce the complexity of the enumeration, several constraints are added as pruning criteria. Here, we survey the subgraph enumeration problem by classifying the previous works according to the specified constraints on the enumerated subgraphs.

Tree Shaped Subgraphs As the type of subgraphs enumerated directly relates to the enumeration problem complexity, some previous researches only focus on enumerating tree-shaped subgraphs [7]. Considering only tree-shaped subgraphs can radically reduce the complexity. In [8], a polynomial time dynamic programming method is applied to cover the DFG with the minimal number of tree-shaped subgraphs. However, enumerating only tree-shaped subgraphs may lead to limited improvements on performance or other aspects.

Multiple Inputs Single Outputs (MISO) In the context of extensible processors, the number of inputs and the number of outputs (I/O) are constrained due to the number of ports to the register files of the base processor. The tighter the I/O constraints are, the less number of subgraphs considered. In other words, the enumeration can be done in a shorter time if restricting the I/O constraints to lower values. Early works that try to enumerate single output subgraphs can be found in [9,10]. A subgraph with multiple inputs and only one output subgraphs is called MISO subgraph. The approach in [9] enumerates all the *K*-MISO subgraphs with dynamic programming, where *K* is the input constraint. Another approach in [10] iteratively considers the MISO subgraphs from the MISO subgraphs of maximal size under a specified input constraint. These approaches are efficient only when the input constraint is low. Unfortunately, relaxing the input constraint can lead to exponential computation.

Maximal Multiple Inputs Multiple Outputs (MaxMIMO) In recent years, enumerating the maximal MIMO subgraphs has drawn a wide interest from researchers who work on application-specific instruction-set extension processors (ASIPs) [11–15]. Pothineni et al. [11] were the first ones to propose an algorithm for MaxMIMO enumeration. The proposed algorithm is based on an incompatibility graph. However, the algorithm only generates connected MaxMIMOs. In [12], the MaxMIMOs enumeration problem is reformulated as a maximal clique enumeration problem after grouping equivalence nodes and building cluster graph. Atasu et al. [13] proved that the number of MaxMIMOs is bounded by $2^{|V_{in}|}$, where V_{in} is the set of invalid nodes in the DFG. A top-down manner algorithm proposed in [14] solves the MaxMIMOs enumeration problem efficiently by a division operation on the DFG.

Connected or Disjoint Subgraphs A computation subgraph can be a connected subgraph or a disjoint subgraph. To reduce the high computational complexity, some authors look only for connected subgraphs [16,17]. Those approaches for enumerating only connected subgraphs achieve lower complexity by sacrificing the optimality. As we known, imposing the parallelism is one of the major benefits by using custom operators. Compared with connected subgraph, the disjoint subgraph can exploit more parallelism when implemented as hardware function unit. Therefore, most of recent researches mainly focus on enumerating all subgraphs including connected subgraphs and disjoint subgraphs [2–5].

Multiple Inputs Multiple Outputs (MIMO) Compared with tree-shaped custom instructions and MISO custom instructions, MIMO custom instructions can provide significant performance improvements without sacrificing too much area [1]. Most of recent studies try to enumerate connected subgraphs and (or) disjoint subgraphs with multiple inputs and multiple outputs [2–5]. Generally, a subgraph could be iteratively formed by absorbing a node or nodes to a previously identified smaller subgraph. Enumerating all possible subgraphs under input and output (I/O) constraints is a computationally difficult problem, because the number of possible subgraphs grows exponentially in the number of inputs and outputs. The upper bound on the number of subgraphs is formally proved to be $|G|^{IN_{max}}|G|^{OUT_{max}}$ in [18], where IN_{max} is the input constraint and OUT_{max} is the output constraint. The work in [3] reports that the performance improvement increases with the relax of input constraint and output constraint.

Table 1 shows the upper bound on the number of enumerated subgraphs under different constraints. In the first column of Table 1, *MIMO* and *MaxMIMO* represent the convex subgraphs satisfying I/O constraints and the maximal convex subgraphs respectively. The connected convex subgraphs are represented as $CC-Subgraphs$.

As shown in Table 1, the subgraph enumeration problem is indeed a computationally difficult problem. However, all of previous works focus on proposing sequential algorithms for this problem. In this paper, we propose a parallel implementation of a latest sequential algorithm [5].

2.2. Subgraph selection

Subgraph selection is the process that selects the most profitable subset of subgraphs from the set of subgraphs enumerated in the subgraph enumeration step. When hardware design is targeted, the subgraph candidates are selected either due to the high frequency of occurrences (to make use of resource sharing) in the application or due to their high performance compared to

Please cite this article as: C. Xiao et al., Parallel custom instruction identification for extensible processors, Journal of Systems Architecture (2016), http://dx.doi.org/10.1016/j.sysarc.2016.11.011