AUTOMATIC ARCHITECTURE GENERATION FOR COARSE-GRAINED RECONFIGURABLE ARRAY

Tiago Teixeira, Brian Luppi, Ricardo Ferreira (Advisor)
Departamento de Informática Universidade Federal de Viçosa
Viçosa, 365700-000
{thiagus,bpimentel,cacau}@dpi.ufv.br

ABSTRACT

Coarse-grained reconfigurable computing architectures vary widely in the number and characteristics of the processing elements and routing topologies used. This paper proposes a design space exploration able to generate automatically different coarse-grained reconfigurable array architectures. The achieved experimental results show that our approach can be useful to evaluate a large set of interconnect topologies as far as coarse-grained reconfigurable computing architectures are concerned.

1. INTRODUCTION

Coarse-grained architectures appears as a more scalable solution than fine-grained architectures (e.g: FPGA) for reconfigurable hardware, and become increasingly important in recent years [1, 2, 3]. This approach reduces the configuration time and complexity, as well placement and routing problem size, and design and synthesis time. However, many array architectures seem to be designed without strong evidences for the architectural decisions taken. Remarkably the work presented in [4] has been one of the few exceptions which addressed the exploration of architectural features. Our approach propose an environment to generate and to evaluate a wide number of local interconnection patterns. These local patterns are scalable, which is fundamental to take advantages of new silicon technology advances.

Our approach is based on genetic algorithm (GA) optimization technique to explore the whole design space. A set of architecture is taken as initial population, and each GA step creates new architectures during the crossover operation and the worst architectures are removed from the population. The selective fitness function is calculated as to be proportional to the average routing cost of a set of DSP kernel benchmarks mapped on each architecture. Figure 1 shows a block diagram of our approach.

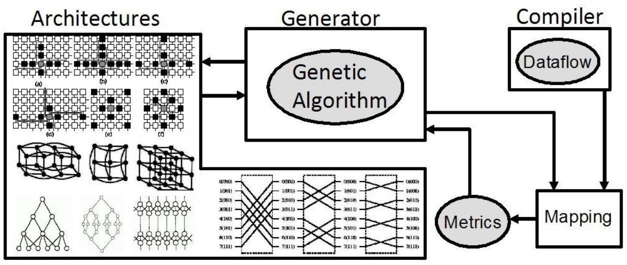

Figure 1 – Design Exploration Environment

The rest of this paper is structured as follows. Section 2 describes the initial population composition. Section 3 presents two approaches to represent an architecture as an individual and the crossover operation. Section 4 shows the experimental results. Finally, the conclusions and ongoing work are presented in Section 5.

2. INITIAL POPULATION

This work consider only 2D array architectures. However, our architecture model is a graph and can handle a n-dimensional architectures. Each architecture is composed by a set of Processor Elements (PE), for example, ALU, MUX, counters, memory cells, etc. and a interconnection network. The interconnection network is composed by a set direct connection between two PE's. Let us consider that each PE has an unique ID number. This number could be the array line and column address. For instance, a PE(i,j) has a direct connection to a PE(i+1,j) in a Grid Interconnection Network. The set of architecture used to build the initial population consist on a selection of most used architectures for coarse-grained and random architectures. In addition, we propose to taken into account a set of bit-operation based architectures. This set is similar to multi-stage interconnection of SIMD architectures.

The coarse-grained architecture set is composed by Grid, Hexagonal, Octal and N-Hop interconnection local patterns. The random architecture set is generated by using random local connection for each PE. The bit operation set is created by using local patterns mixed to bit based pattern like hypercube, shuffle-exchange, butterfly, baseline, etc. For instance, a cube pattern can be generated a local connection for the PE(i,j) by using inverting a bit on line address. Let us consider the PE(4,5) or PE(100,101) in binary. A cube based connection on bit 2 will interconnect PE(**1**00,101) to PE(000,101) or PE(0,5).

3. AUTOMATIC ARCHITECTURE GENERATION

We propose two approaches to represent an architecture as an individual in GA. The first approach is based on PE set or node. The second approach is based on a local PE interconnect set. Let us consider a NxM array architecture, where N and M are the line and column number, respectively.

3.1 PE SET APPROACH

A architecture can be modeled as a set of PE. Each PE is defined by its ID and a local connection set. Let us consider the two architectures in shown in Figure 2(a), where each PE has 2 output connections. In the first architecture, each PE has a line connection and a column connection. For the second architecture, each PE has a diagonal connection and either a line or a column connection. The architecture can be represented as one-dimensional vector, where the position i stores the set of local output connection from the PE i. Let us consider the vector A and B in Figure 2(b), which represent the two architectures in shown in Figure 2(a). The genetic algorithm will create new architectures by applying a crossover operation on Individual A and B. Suppose the vector cut-point at position 2. Two new architectures will be generated as shown in Figure 2(d). The first one has the PE 1 and 2 from A and PE 3 and 4 from B. Each PE brings its output connections to the new architecture.

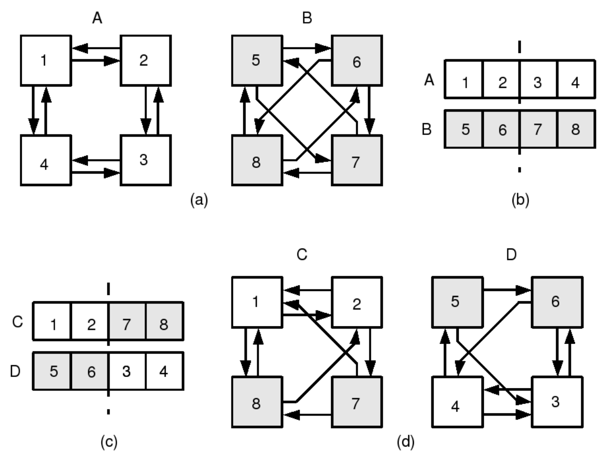

Figure 2 – PE Set Approach

3.2 LOCAL INTERCONNECT APPROACH

Let us consider homogeneous architectures, where a PE has the same interconnect set. Let us consider the architecture in shown in Figure 3(a), where each PE has 4 output connections in line and column directions. This architecture is modeled by the vector shown at the top of Figure 3(c). Each output has a pattern in function of its line and column address, where we use i as the line number and j as the column number. For instance, the output 1 is a function of i-1, that is, the output is connected to previous line and same column. Figure 3(b) shows another architecture which local set is shown in Figure 3(c). The crossover operator selects a cut-point in local set vector and two new local sets are generated. Figure 3(d-f) shows the result of crossover operation at cut-point 3 for the architectures displayed in Figure 3(a-c).

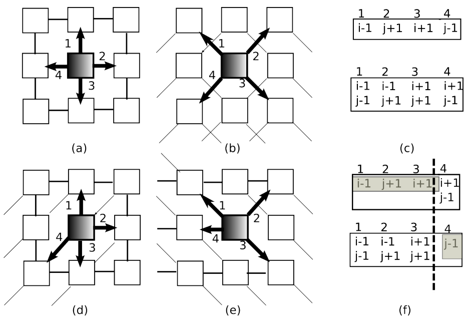

Figure 3 – LOCAL INTERCONNECT APPROACH

4. EXPERIMENTAL RESULTS

The current prototype environment has been used to evaluate the set of architectures by mapping a set of benchmarks on each array architectures. The mapping tool presented in [5] has been used. The set of benchmarks includes the following DSP algorithms: FIR, CPLX, FDCT, Paeth, FilterRGB and SNN. FIR is a 1-D finite-impulse response filter (examples with 16 and 64 of taps are used). CPLX is a FIR filter using complex arithmetic. FDCT is a fast discrete cosine transform implementation. Paeth is the PNG (Portable Network Graphics) Paeth encoding routine. The Filter RGB is an image filter to highlight an image by brightening or darkening the pixels in the images, SNN is a symmetric nearest neighbor image filter.

The initial architecture set has 100 individuals, where 60% was composed by regular architectures, 40% by random architectures. The genetic algorithm has been runner for 100 generations. The fitness cost function of each architecture has been calculated as the average routing connection cost for each mapped benchmark. The renovation rate at each generation was 10%.

Figure 5 shows the fitness cost for each 100 hundreds generations by using the crossover presented in Section 3.1, based on set of PE. Although the random characteristics, the best architectures have same common features. A more detailed statistics analyze is needed, however preliminary results point out architecture with local connections (eg.: i+1 or j-1) and hop connections (eg.: i+3, j-4). For all architectures, each PE has a local set of eight neighbors, and two directed connection for each neighbors.

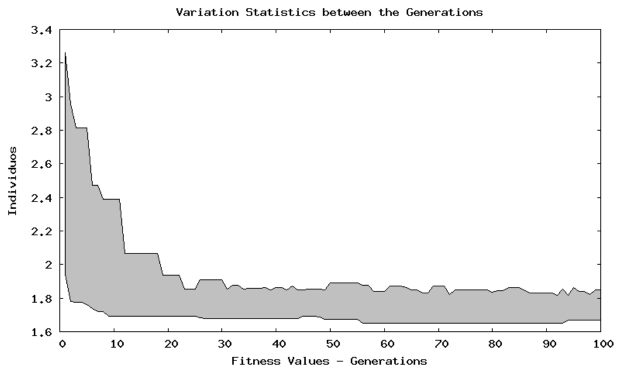

Figure 5 – Genetic Evolution for 100 generations.

5. CONCLUSIONS

This paper presents a design exploration environment for reconfigurable, coarse-grained array architectures. Particularly, results achieved for this preliminary evaluation of a number of architectures with different routing topologies have been shown. Those results clearly point out the capability of our approach to automatically explore different design characteristics, in few hours. Short term plans include more depth statistics analyzes, crossover and fitness improvements, Simulating Annealing and other optimization techniques to explore the best architecture generated by the GA approach.

6. REFERENCES

[1] Hartenstein, R. "A Decade of Reconfigurable Computing: a Visionary Retrospective", DATE, 2001. Munich, Germany, 2001.

[2] Bossuet, L.; Gogniat, G.; Philippe, J. "Fast Design Space Exploration Method for Reconfigurable Architectures", The International Conference on Engineering of Reconfigurable Systems and Algorithm, ERSA'03, Las Vegas, USA, 2003.

[3] Bansal, N.; Gupta, S.; Dutt, N.; Nicolau, A.; Gupta, R. "Network Topology Exploration of Mesh-Based Coarse-Grain Reconfigurable Architectures", DATE '04: Proceedings of the Conference on Design, Automation and Test in Europe,

[4] Hartenstein, R.; Herz, M.; Hoffmann, T.; Nageldinger, U. "KressArray Xplorer: a new cad environment to optimize reconfigurable datapath array", In: Proceeding of the 2000 conference on Asia South Pacific design automation 2000.

[5] Ferreira, R.; Garcia, A.; Teixeira, T.; Cardoso, J.. "A Polynomial Placement Algorithm for Data Driven Coarse-Grained Reconfigurable Architectures", In International Symposium on VLSI, pages 61-66, IEEE Computer Society, Los Alamitos, CA, USA, 2007