TEMPORAL GRAPH PLACEMENT ON MESH-BASED COARSE GRAIN RECONFIGURABLE SYSTEMS USING THE SPECTRAL METHOD

Florian Dittmann
Heinz Nixdorf Institute, University Paderborn
Fuerstenallee 11, 33102 Paderborn, Germany
roichen@upb.de

Christophe Bobda
University of Erlangen-Nuremberg
Am Weichselgarten 3, 91058 Erlangen, Germany
christophe.bobda@informatik.uni-erlangen.de

Abstract: Coarse grain architectures need domain specific place and route methods. The existence of such methods provided, algorithms can be executed in time and space on the processing elements of coarse grain reconfigurable systems. Additionally, the short reconfiguration time and reduced power requirements comparing to FPGA makes coarse grain systems worth to be scrutinized as alternative execution platforms. By help of spectral methods, which target quadratic distance optimization of graphs, we can achieve short communication distances. This paper shows how to use the spectral method for efficiently scheduling processing elements of coarse grain reconfigurable devices by combining the spectral method and ASAP scheduling.

Keywords: Coarse Grain Reconfigurable Computing, Temporal Placement

1. INTRODUCTION

The advent of multimedia into our life seems just to have started. We are surrounded by multiple electronic devices in every situation and independent of our location (sometimes referred to as nomadic computing). Presumably, the amount of services will even increase further. Yet, more services also means an increase in the amount of requirements, i. e., different/new protocols, decoding and encoding methods, compression

techniques, etc. Due to often used advanced concepts, these requirements demand for special purpose hardware (e. g. ASICs), as general purpose processors often cannot meet the speed requirements. Reconfigurable systems, which facilitate the combination of the performance of ASICs and the flexibility of general purpose processors, seem to be worth to be scrutinized as an alternative.

Thereby, the realm of mobile devices often comprises power constraints. Power-intensive FPGAs as reconfigurable devices are only of limited value as processing units for mobile devices. Coarse grain reconfigurable architectures seem to be an alternative. They comprise path-widths greater than 1 bit, which enables more area-efficient operation. Additionally, they do not have such a huge routing area overhead and poor routability compared to FPGAs (Hartenstein, 1997). Further, coarse grained architectures provide operator level configurable function blocks, word level datapaths, and powerful and very area-efficient datapath routing switches. They are often used to speed up computational intensive loops (Hartenstein, 2001).

The limited routing capabilities of coarse grained devices demands for advanced compilation and mapping methods. Connections between random processing elements often is impossible. However, local connection facilities are often numerously available. In order to avoid additional computation delays due to poor routing, mapping algorithms should particularly target at short wire lengths. In this paper, we introduce an approach how to find a placement for mesh-like coarse grain reconfigurable architectures, which explicitly aims at nearby placement when the communication is intensive. Therefore, we use the spectral method as introduced by Hall, 1970. The quadratic objection of the spectral method is capable to reduces long wires.

The rest of this paper is organized as follows. We review related work, before we introduce our topology abstraction, including the modeling of the problem. In Section 4, we introduce the spectral method for data flow graphs. Section 5 shows how we combine the spectral method and ASAP scheduling to temporally place data flow graphs on the processing elements of coarse grain reconfigurable devices. Finally, we conclude.

2. RELATED WORK

Coarse grain reconfigurable devices are not widespread. Consequently, their compilation, mapping and placement concepts are sparsely present in the literature.

Recent work in the area was done by Bansal et al., 2004, performing network topology exploration. The authors propose a consecutive place-

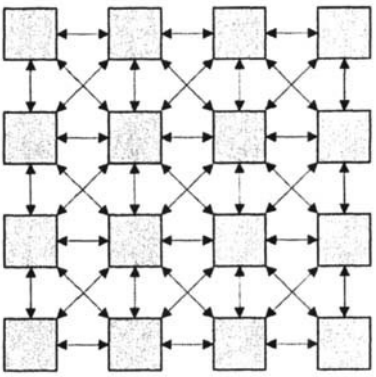

Figure 1. Topology abstraction of mesh-based coarse grain reconfigurable devices.

ment (topology traversal). The concept abstracts from heterogenous architectures and produces reasonable results. They do not focus on the possibility to reuse processing elements in directly succeeding cycles.

Lee et al., 2003 present a generic reconfigurable architecture targeting coarse grain architectures. They describe a compilation approach to manage the memory bottleneck that typically limits the performance of many applications. Their mapping flow comprises three steps, the first targets the processing elements mapping, the second operation grouping on processing element lines, and the third the arrangement of lines including stacking and time multiplexing.

Nageldinger, 2001 differentiates the necessary design steps for coarse grain reconfigurable architectures into *technology mapping*, *placement algorithm*, and *routing algorithm*. He summarizes known approaches and shows that some designers of coarse grain devices use manual placement (e. g. Singh et al., 2000; Miyamori and Olukotun, 1998), while others use heuristic methods like simulated annealing (e. g. Hartenstein and Kress, 1995; Ebeling et al., 1996). Most often, the place and route methods are target specific and require a lot of work put into to adapt to different reconfigurable devices. Seldom, topology abstraction is present.

3. TOPOLOGY ABSTRACTION

We consider coarse-grain reconfigurable architectures that consist of processing elements (PEs) connected in a mesh-like network topology (see Fig. 1). The PEs, which are represented as square boxes, are arranged in an equally spaced 2-D style. The double headed arrows denote data communication links between the PEs. We call this style a *grid* following Bansal et al., 2004.

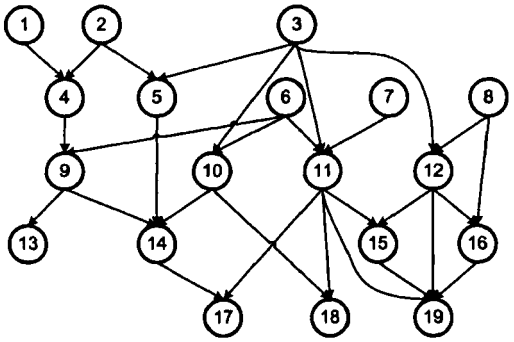

Figure 2. Data flow graph

In coarse grain reconfigurable systems, often the reconfiguration can be done individually on every node, i. e., reconfiguration can take place partially and during run-time. The temporal assignment of tasks to nodes must be planned with respect to the whole period of this task including the communicating processing elements. Concerning internode communication, direct neighboring communication comprises the lowest costs and thus is most desirable.

We assume that every PE can hold its result as long as there arrive no new results. Thus, the storing of the results is independent of possible reconfigurations of the node. Additional memory for intermediate results is available externally. As the access of this memory is costly, we prefer the local storage. In particular, results stored externally must be saved to the memory bank as well as restored from this bank. We can avoid external storing, if succeeding nodes execute in the neighboring regions. The algorithm introduced below achieves this goal by using the spectral method.

4. SPECTRAL METHOD

The spectral method as proposed by Hall, 1970 is originally suited for the layout of VLSI chips. Its inherent quadratic objective functions reduces wire lengths significantly. The result of placing elements on a 2-D area using the spectral method can be parameterized by weighting the connections. It is possible to extend the initial 2-D result to a 3-D version in order to introduce the temporal domain. Yet, the adding of a third axis increases the complexity and may lead into results that are difficult to handle. For example, we could derive a placement that is communication optimized and place the nodes referring to these results, while ignoring precedence constraints that would enforce a differ-

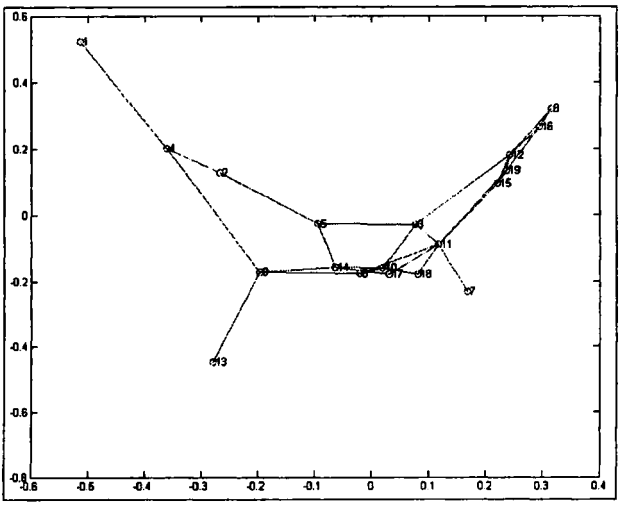

Figure 3. Spectral placement of the data flow graph of Fig. 2

ent placement. Thus, we realize our placement strategy based on the 2-D version of the spectral method. The temporal domain is introduced by ASAP scheduling (see below). There already exists investigations of the spectral method for fine grained reconfigurable systems targeting temporal placement (Bobda, 2003).

As input for our approach we need a graphical representation G of the algorithm, as displayed in Fig. 2. The vertices V of this graph G represent tasks that can be mapped on one processing element. The edges E denote the connections, i. e., internode dependencies like precedence constraints. The pure communication optimized spectral placement is achieved without referring to additional parameters of the input graphs, i. e., different execution times or precedence constraints can be considered later. Yet, there exist variants to include execution time similarities into the spectral placement (Bobda, 2003). In this work, we use the basic spectral method only.

Fig. 3 shows the results of the spectral placement of the data flow graph G of Fig. 2. If our reconfigurable device is large enough, we can place the whole graph by assigning the top left node to the top left PE and so on. Yet, in most cases, the graph does not fit completely on the device. Thus, the temporal functionality (dynamic reconfiguration) must be used to execute the graph over time. We use this placement of the input graph and derive an appropriate temporal placement method

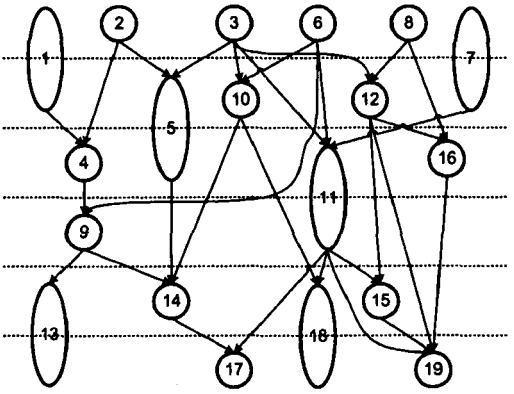

Figure 4. ASAP schedule of the graph.

in the next section. Thereby, the results of the spectral method are valuable information to keep wire lengths short.

The mathematical background comprises the following: We represent the input graph as the Laplacian matrix B, derived from the connection matrix C and the degree matrix D ($B = D - C$). The spectral method minimizes the sum of squared distances between the nodes $R = X_1^T B X_1 + X_2^T B X_2 + \ldots + X_k^T B X_k$ subject to $X_1^T X_1 = X_2^T X_2 = \ldots = X_k^T X_k = 1$. To solve these equations, we apply the Lagrange multiplier method with the k Lagrange multipliers $\lambda_1, \lambda_2, \ldots, \lambda_k$. The solution are the Eigenvectors associated to the k smallest non zero Eigenvalues. The Eigenvectors place the vertices in space.

5. TEMPORAL PLACEMENT

We show the concept by referring to the data flow graph of Fig. 2. As processing platform, we assume a coarse grain reconfigurable device consisting of homogenous processing elements. The network topology is as displayed in Fig. 1. Without loss of generality, we reduce the amount of PE to nine in this example.

Fig. 4 shows the data flow graph in some more detail. We have arranged the vertices in ASAP (as soon as possible) style, respecting different execution times, as we want to schedule and place such vertices comprising different execution times. The ASAP scheduling introduces the temporal dimension to the problem by defining execution levels for each node. Due to the precedence constraints and different execution times, we derive six levels in this example.

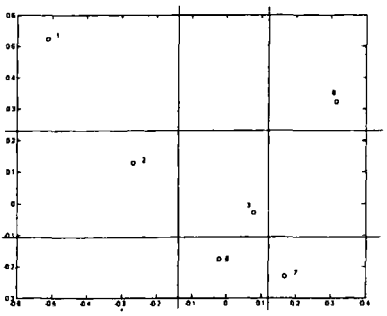

Figure 5. Spectral placement and processing element assignment of level 1.

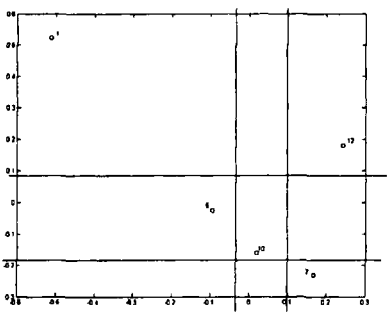

Figure 6. Spectral placement and processing element assignment of level 2.

In the following, we combine the results of the ASAP scheduling and the spectral placement to efficiently place the nodes on the PE of the reconfigurable device.

We select the vertices and their coordinates of level 1 in the spectral placement of the data flow graph. Their distances respect their closeness in terms of input for later nodes of the graph. Thus, we use this information to assign the vertices to the PEs. The placement is realized by a consecutively assignment. We start with the node comprising the smallest x and largest y values (top-left corner). This node is placed in the top-left corner of the reconfigurable device. We continue with this procedure until the node comprising the largest x and smallest y values is reached. Fig. 5 shows how the vertices of level 1 are assigned to the processing elements.

When scheduling the next level, we have to take care of vertices that have already started with their execution in the previous level. These nodes must not be placed to different processing elements. Additionally, their location acts as an indicator for placing close nodes of the ASAP scheduling in proximity, as far distributed results would foil the intended reduction of communication achieved by the spectral method.

Thus, the procedure is as follows: We extract the corresponding vertices of the level 2 from the communication optimized spectral placement of the data flow graph (refer to Fig. 6). We first lock the already placed vertices to their location (node 1 and 7). Then, we fill the gaps (speaking in terms of free PEs between locked PEs) by assigning the remaining vertices of level 2. The selection of the PE is done as above, i. e., evaluating the x and y location of the vertices. Finally, we achieve a PE assignment that respects the assignment of the previous level and serves

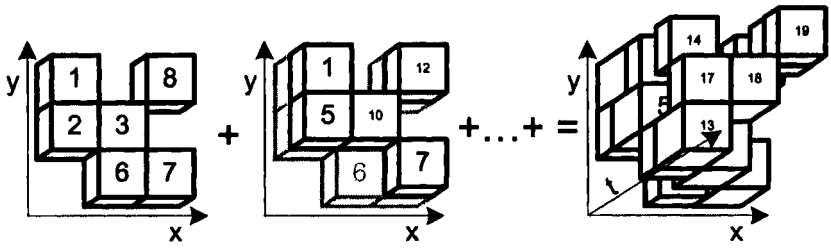

Figure 7. Combination of micro level placement.

as solid basis for the next level due to the PE assignment referring to the spectral placement of the whole graph.

If this PE assignment is continued, we achieve an amount of so called micro levels, which comprise a valid placement each. The dependencies due to the spectral method guarantees short communication distance from one to the next level. When combining the placed levels, we derive a complete schedule as displayed in Fig. 7. Thereby, different execution times of the nodes result in different heights of the boxes (nodes) in the final schedule. The results gained are valuable for a configuration scheduler for the coarse grain device.

Due to using the spectral quadratic wire length optimized placement, we achieve direct neighbor connections or connections to the same processing element in the next micro level in majority. For the exemplary data flow graph, Fig. 8 shows the connections. We have mapped all levels into the figure, i. e., all vertices mapped to the same PE are displayed in the corresponding square box. As an additional result, we derive PEs that will not be used during execution of the data flow graph.

In this exemplary description, the amount of PEs is in the range of the amount of vertices each level. In detail, there are always enough PE to hold all possible nodes of each level and directly derive a valid schedule. In order to overcome this limitation, we extend our placement as follows:

If there are always twice as much PEs than vertices per level, we combine two levels to be placed together. This aggregation can be continued with all integer multiples of the amount of PEs. Thereby, we can often increase the proximity of vertices comprising internode communication resulting in more efficient execution.

In order to overcome the opposite limitation, where the amount of nodes each level is larger than the amount of available PEs, we can split each level into sub-levels, which each encloses the maximum amount of nodes. Depending on the individual cases, the intermediate results

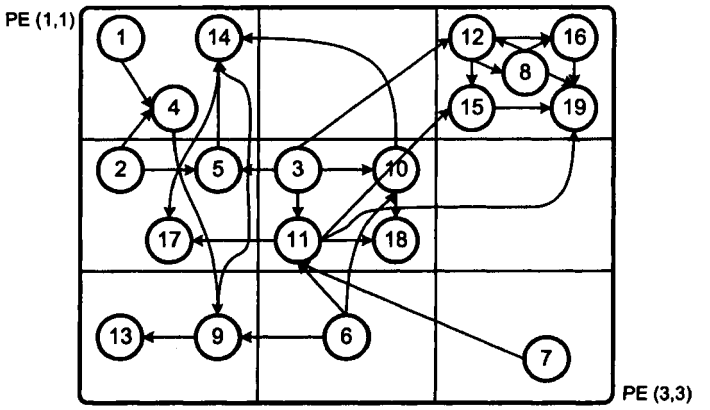

Figure 8. Internode connection.

might have to be stored externally, increasing the costs. We prefer another option, where we form subproblems, which comprise each a cluster of nodes that are strongly connected. The clusters can be formed independent of the levels by referring to the spectral placement and applying a cluster growth method (Alpert and Kahng, 1994). Then, we start the temporal PE assignment on the nodes of these clusters separately using the method described above.

6. CONCLUSION

In this paper, we have presented how to use the spectral method for placement on coarse grain reconfigurable systems. The method suits the limited routing capabilities of the devices very well. Using as soon as possible scheduling allows us to effectively introduce the temporal domain into spectral method based placement. The ASAP scheduling thereby enables us to select a sub-amount of the spectral placed vertices. The PE assignment takes place with respect to previously scheduled nodes. Thus, the flexibility, i.e., temporal adaptation of the processing elements of coarse grain reconfigurable devices is achieved with optimized wire length. Finally, we yield a schedule that can reduce the execution time, which otherwise often is negatively impacted by poor routing, i.e., routing that must traverse several processing elements before reaching the data sink.

In the future, we plan to extend the placement in order to reduce waiting cycles due to complicated memory access of coarse grain reconfigurable systems.

Acknowledgements

This work was partly funded by the *Deutsche Forschungsgemeinschaft (SPP 1148).*

REFERENCES

Alpert, C. J. and Kahng, A. B. (1994). Multi-way partitioning via spacefilling curves and dynamic programming. In *DAC '94: Proceedings of the 31st annual conference on Design automation*, pages 652–657. ACM Press.

Bansal, Nikhil, Gupta, Sumit, Dutt, Nikil, Nicolau, Alex, and Gupta, Rajesh (2004). Network Topology Exploration of Mesh-Based Coarse-Grain Reconfigurable Architectures. In *Proceedings of the DATE*, Paris, France.

Bobda, Christophe (2003). *Synthesis of Dataflow Graphs for Reconfigurable Systems using Temporal Partitioning and Temporal Placement.* PhD thesis, University Paderborn, Heinz Nixdorf Institute.

Ebeling, Carl, Cronquist, Darren C., and Franklin, Paul (1996). RaPiD - Reconfigurable Pipelined Datapath. In *FPL '96: Proceedings of the 6th International Workshop on Field-Programmable Logic, Smart Applications, New Paradigms and Compilers*, pages 126–135. Springer-Verlag.

Hall, Kenneth M. (1970). An r-dimensional Quadratic Placement Algorithm. *Management Science*, 17(3):219–229.

Hartenstein, Reiner (2001). A Decade of Reconfigurable Computing: a Visionary Retrospective. In *Proceedings of the Conference on Design, Automation and Test in Europe (DATE 01).*

Hartenstein, Reiner W. (1997). The Microprocessor is no more General Purpose: why Future Reconfigurable Platforms will win. In *Proceedings of the International Conference on Innovative Systems in Silicon, ISIS'97*, Austin, Texas, USA. IEEE Computer Society. Invited Paper.

Hartenstein, Reiner W. and Kress, Rainer (1995). A datapath synthesis system for the reconfigurable datapath architecture. In *Proceedings of the 1995 conference on Asia Pacific design automation (ASP-DAC'95)*, page 77, Makuhari, Chiba, Japan. ACM Press.

Lee, Jong-eun, Choi, Kiyoung, and Dutt, Nikil D. (2003). Compilation Approach for Coarse-Grained Reconfigurable Architectures. *IEEE Design & Test of Computers*, 20(1):26–33.

Miyamori, Takashi and Olukotun, Kunle (1998). REMARC: Reconfigurable Multimedia Array Coprocessor (Abstract). In *FPGA*, page 261.

Nageldinger, Ulrich (2001). *Coarse-Grained Reconfigurable Architecture Design Space Exploration.* PhD thesis, University of Kaiserslautern, CS department (Informatik).

Singh, Hartej, Lee, Ming-Hau, Lu, Guangming, Bagherzadeh, Nader, Kurdahi, Fadi J., and Filho, Eliseu M. Chaves (2000). MorphoSys: An Integrated Reconfigurable System for Data-Parallel and Computation-Intensive Applications. *IEEE Trans. Comput.*, 49(5):465–481.