Architecture Driven Partitioning

Joachim Küter
Infineon Technologies AG
MP TI CS ATS
D-81541 Munich, Germany

Erich Barke
Institute of Microelectronic Systems
University of Hannover
D-30167 Hannover, Germany

Abstract

In this paper, we present a new algorithm to partition netlists for logic emulation under consideration of the targeted emulator architecture. The proposed algorithm allows the flexible use for a wide variety of applications because the description of the architecture is part of the input data. It combines a new approach of finding and improving an initial solution with existing algorithms to cluster the netlist and optimize the number of cut nets between blocks. As a result, the algorithm ensures that the cut nets between the created blocks can be connected within the emulation system, even without a full interconnect structure. Experiments on a number of designs and architectures demonstrate that the algorithm is competitive for architectures with full interconnect and that it is unique for architectures with limited interconnect resources.

1 Introduction

The partitioning of netlists is a standard procedure in order to use logic emulation as a means for functional verification. After partitioning, the resulting blocks are mapped onto Field Programmable Gate Arrays (FPGAs). Plenty of partitioning algorithms have been presented. Most of them are focused on cost minimization (e.g. number of created blocks, number of cut nets). In general, they incorporate the assumption that the required resources to connect the cut nets will be provided by the emulation system. Therefore, the used model of the underlying architecture is that of completely connected FPGAs.

Architectures of existing emulation systems usually do not provide such a kind of complete interconnect structure. Often, groups of FPGAs (e.g. on a PCB) are wired by an approximation of complete interconnect. However, the entire system mostly comprises several of such groups which have only a limited number of connections between them. Those limitations in system level routing resources are usually not taken into account by partitioning algorithms.

Several published methods focus on limited routing resources. However, they are mostly fixed to a special type of architecture or lack the capability of handling real world constraints, like emulation systems being subdivided into several boards. It is not possible to apply those algorithms to freely defined emulator architectures.

This paper presents a partitioning algorithm which takes the architecture of the targeted emulation system into account. The formal description of the architecture is part of the input data and can be freely modeled to easily address different target systems. It takes care of limited routing resources and heterogeneous FPGA types.

The described algorithm combines existing methods with novel aspects and forms a new partitioning flow. Experimental results on a number of benchmarks and emulation system architectures demonstrate the benefits of the proposed algorithm for architecture driven partitioning.

2 Problem Description

In order to consider the architecture of an emulation system during the partitioning process, it is essential to use a formal description of the structure of the system, i.e. the basic building blocks, the routing resources and hierarchy of the system.

2.1 Structure of Emulation Systems

The basic building blocks of emulation systems fall into two categories. Programmable chips for logic elements, usually FPGAs, will each contain a portion of the netlist. In some cases they may also be used for routing purposes. The second category comprises chips which are solely used for routing purposes, e.g. FPICs (Field Programmable Interconnect Circuit) [1]. They mostly contain a switch matrix which can create connections between any of their pins. FPGAs which are dedicated to routing purposes belong to the same class as specific routing circuits.

The pins of the basic chips are connected at board and at system level via interconnect nets. The simplest ones are two point nets, which connect exactly two pins of two

different basic chips. Nets which connect more than two pins will be referred to as multi-pin-nets. Their main purpose is to form interconnect for the distribution of low skew signals like clocks, which will be distributed to a larger number of basic chips.

The structure of existing emulation systems comprises different basic chips and direct interconnect. FPGAs may be interconnected via a fixed topology or via programmable routing circuits. With the help of routing chips, it is possible to form a crossbar connection, or with a more limited hardware effort, a partial crossbar structure [2] between the FPGAs. In both cases it can be assumed that, while not exceeding the available pin numbers, there are unlimited interconnect resources. It is possible to create connections between any two pins.

If there are only direct connections between FPGAs available, some pins will be required for connections between FPGAs which are no direct neighbors. In these cases the already pin limited partitioning process requires even more of the scarce pin resources.

In many cases, the structure is subdivided into different hierarchical levels. Often, a board contains fully connected FPGAs and a couple of boards is combined in racks. These racks can again be combined to larger entities. Frequently, within one level there are almost complete interconnect structures, but between different levels there are strong limitations.

2.2 Formal Description of the Netlist

The netlist which shall be partitioned, can be described as a hypergraph, like in most partitioning algorithms. To facilitate the partitioning process, it is reasonable to keep the internal hierarchy of the netlist. In this case, the initial graph contains the top level cell of the netlist. During the partitioning process it is possible to expand the nodes, which means that they are replaced by nodes which represent the subcells of the original cells. The edges have to be adapted accordingly. One possible realization of such a graph has been described in [3].

2.3 Formal Description of the Emulator Architecture

The formal description of the architecture is also based on a hypergraph. The basic chips and the interconnect are represented as nodes which will be displayed as squares, marked with ‘F’ for FPGAs and ‘R’ for routing chips. Direct interconnect structures are represented as hyperedges.

In order to describe a hierarchical structure, there are special nodes which represent the interfaces and the internal nets of hierarchical structures, marked with ‘i’ and ‘n’ respectively.

The algorithm which will be described in the next section, will subdivide the complete partitioning problem into smaller problems, based on hierarchy. Each must contain all applicable information about routing resources between the contained entities. This requires the graph to be constructed in a special way.

The requirement shall be illustrated by the following example: A system containing a couple of boards, each with two FPGAs, connects all FPGAs via a backplane. Taking a board as a subproblem, it would comprise two FPGAs with no common interconnection. This would obviously result in a non solvable subproblem. This situation may be avoided by following rules for hypergraph construction.

First of all, the hierarchical structure is represented by a hypergraph which fulfills the condition of chordality.

> **Definition:** A graph is chordal, if it does not contain induced subgraphs in the form of circles with more than three nodes.

The resulting structure has similarities to a tree, which is a typical representation of hierarchy. In order to be able to identify distinct subproblems, a sufficient condition for the creation of the graph has to be applied:

> **Sufficient Condition:** Higher levels of hierarchy are connected to contained basic elements solely over not more than one crossbar node or one multi-pin-net which is not connected to all basic elements or one direct connection and any number of multi-pin-nets which are connected to all basic elements.

One typical result of a formal description is presented in Figure 1. The described emulation system consists of two racks with three boards each. On every board there are two FPGAs and the various entities are connected via crossbar chips.

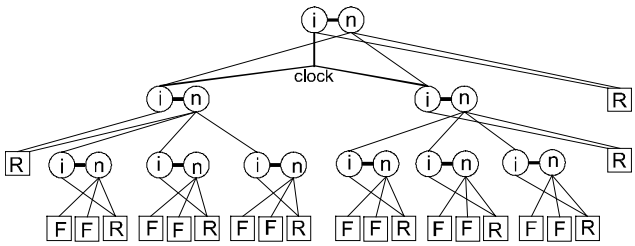

Figure 1: Chordal hypergraph

2.4 Partitioning Problem

Using the previous description of the emulator architecture and the netlist, the partitioning problem can be formulated as follows:

An assignment has to be found for every node of the netlist graph to those nodes of the architecture graph, which represent a basic cell of the type 'FPGA'.

This assignment has to fulfill three conditions:

i. The total size of cells assigned to an FPGA may not exceed the FPGA's gate capacity.
ii. The pins of the cells are connected via nets to other cells. The placement of cells in different FPGAs results in cut nets. The number of cut nets connected to an FPGA must not exceed the available number of pins of the FPGA.
iii. The cut nets have to be connected on system level via the routing resources of the emulation system. The required routing resources must not exceed the available routing resources.

While the first two conditions are common for most partitioning problems, the third one is additionally considered by the presented algorithm.

3 The Architecture Driven Partitioning Algorithm

3.1 Overview

The presented architecture driven algorithm delivers a partitioning result which will fit into the available resources. It is not the primary intention to minimize the number of used chips or cut nets, as long as the required numbers are below the available ones. It focuses instead on the routing resources to ensure that the cut nets may finally be connected. If such a valid solution has been found, the algorithm stops without further optimization effort because it means that the partitioned netlist may already be emulated by the hardware.

The algorithm subdivides the partitioning problem into several smaller problems, each comprising only a subset of the architecture graph and the netlist graph. As already described, it is important that the problems are separate and do not depend on each other. This will be ensured by the way the chordal hypergraph is created to describe the emulator architecture.

The first step is to identify subproblems. This is done following a top down approach. For an emulation system consisting of FPGAs which are distributed over several boards, the netlist is divided into parts which are assigned to boards. On every board a partitioning process takes place, which forms the blocks that are put into the FPGAs.

To formalize this approach, it is possible to identify an elimination order for the chordal hypergraph. This creates a sequence in which the nodes may be removed while the remaining graph will still be chordal. The inverse elimination order can then be used to determine the order in which the subproblems will be solved.

To every subproblem the same procedure is applied. Initially, a prepartitioning step is performed: The netlist graph is clustered and the resulting nodes are assigned to the nodes of the architecture graph after they have been placed geometrically. Then, the borders of the blocks are optimized until the limits are satisfied.

Finally, the separately solved subproblems will be recombined. At this time, the assignment to the FPGAs is completed.

The last step is the assignment of routing resources of the emulator in order to connect the cut nets. If this step can be completed successfully, the algorithm has created a valid solution.

The structure of the algorithm is shown in Figure 2. The following paragraphs expand on various steps.

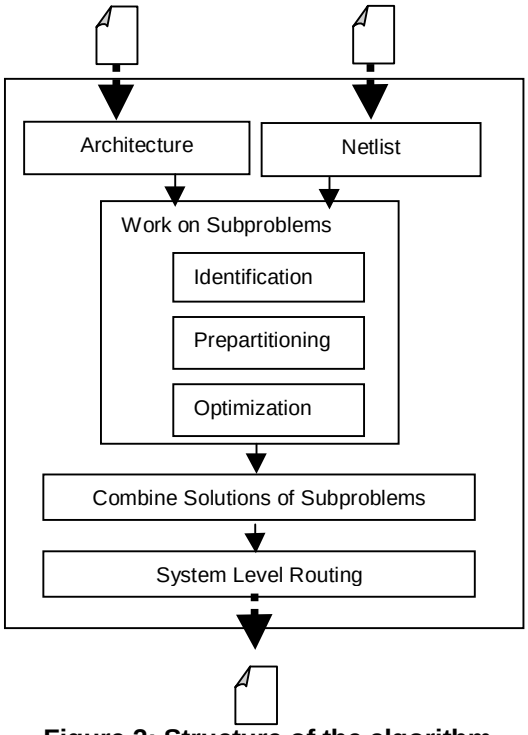

Figure 2: Structure of the algorithm

3.2 Prepartitioning Approach

The prepartitioning shall lead to a good basis for the further optimization steps. It is intended to group together parts of the netlist which are closely connected.

In order to reduce the complexity of the problem, the nodes of the netlist graph will represent clusters of cells. These clusters can be formed by keeping hierarchical elements of the original netlist as entities or by applying a standard partitioning algorithm. In this case the created blocks are later used as clusters. The Hierarchy Driven Partitioning (HDP) algorithm is well suited for this purpose [4]. It keeps the internal hierarchy structure of the created blocks, which allows an expansion of nodes to increase granularity. It will be shown that based on such

an initial solution, it is possible to find an improved solution with a superior quality. This is even valid for architectures with full interconnect.

The initial solution of the partitioning problem is created by a two-dimensional placement of the nodes of the architecture graph and the netlist graph. It commences with a ranking of the nodes. In a second step, the branches of a spanning tree, called sequences, are isolated. These sequences are then ordered, realizing a (local) minimum of the required interconnection length between the nodes.

In the next step, both representations are scaled to the same size values. For the nodes of the architecture graph, target areas are determined which cover the complete area between minimum and maximum coordinates in both directions. Then, the nodes of the netlist graph are assigned to the target areas, thus creating an assignment to the architecture nodes.

This results in closely connected nodes to be placed together. At the end of this step, the size limits of the target blocks are met, but the number of cut nets may still exceed the available number of pins.

The described procedure is illustrated in Figure 3 by an example, comprising a small netlist and an emulation system consisting of a 3x3 array of FPGAs. The two-dimensional placement is the actual result of the implemented version of the described algorithm.

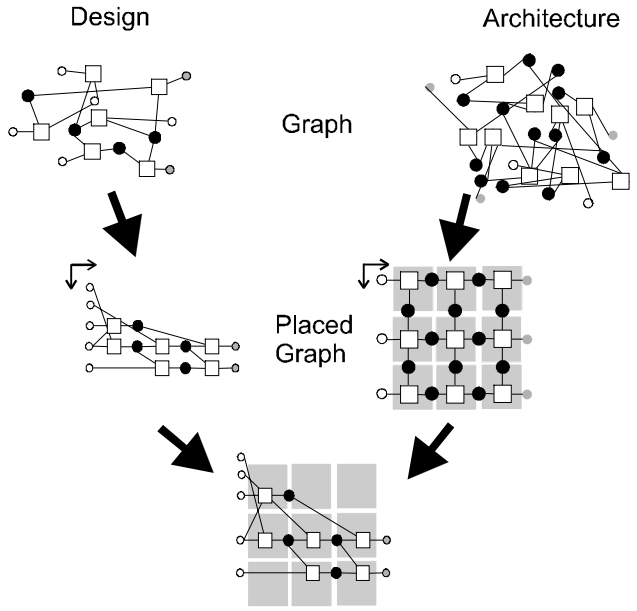

Figure 3: Prepartitioning process

3.3 Optimization Steps

In order not to exceed the maximum number of cut nets and available routing resources, the preliminary solution has to be optimized.

To determine the areas to optimize, two matrices are calculated. One describes the available connections between the involved target elements and a second describes the cut nets between the blocks.

To satisfy the conditions, a sequence of optimizations between two blocks is executed. With each optimization exactly two borders between blocks are involved. The Fiduccia Mattheyses (FM) algorithm is ideally suited for this purpose [5]. It requires a modification to take into account nodes that may have connections to other blocks. A move which might be good under the isolated view of the two blocks might not be ideal considering the additional nets.

The optimization is performed until the limits are not exceeded or until it can be stated that no solution can be found. Before every iteration, nodes of the hypergraph are expanded, i.e. they are replaced by nodes for cells of a lower hierarchy, to increase the granularity of the problem.

3.4 System Level Routing

When every basic cell of the design is assigned to an FPGA of the emulation system, the partitioning process is completed. To ensure that the blocks can be connected, an actual routing process has to take place. In this context, routing means the assignment of resources to the cut nets of the netlist. For this purpose the Lee-algorithm [6] has been chosen. It has originally been designed for two-pin-nets and it had to be modified to route nets with more than two pins. The algorithm is executed until all nets have been routed or until it is determined that at least one cut net cannot be routed. In this case, the partitioning process would have failed.

4 Experimental Results

The presented algorithm has been implemented in a program called ‘Adaptive Partitioning with Architecture Consideration’ (APA). The results of APA are compared to the partitioning results of two different algorithms who have been proven as strong in various publications with respect to fully interconnected FPGAs. The first one is the Hierarchy Driven Partitioning (HDP) algorithm [4]. It uses hierarchy information contained in a netlist and respects the pin and gate limits of the target FPGAs while it minimizes their required number. The second one is hMETIS [7]. It works on a flat netlist and finds a solution for a given number of blocks while it minimizes the sum of cut nets, not exceeding a balance value for the sizes of the blocks.

As these algorithms are not able to take limited architectures into account, the created blocks were randomly assigned to FPGAs. The same routing algorithm as in APA was used to interconnect the blocks on system level.

Table 1: Completely connected FPGAs

design	FPGA Type	APA			HDP			hMETIS		
		b	øp	t[min]	b	øp	t[min]	B	øp	t[min]
watch	220 pins	2	21	0:19	2	14	0:06	2	14	0:02
sab	2.5 kgates	7	203	23:21	5	212	15:48	7	154	0:14
pe		7	181	17:06	7	191	45:10	25	87	0:43
iir		7	101	1:25	7	101	0:18	7	112	0:27
fidct	240 pins	4	215	6:19	4	199	2:05	4	183	0:30
t17	20 kgates	3	145	0:24	3	145	0:29	3	141	0:46
sp7		(./.)	(./.)	(8h0)	(40)	(225)	(3h31)	(25)	(150)	(4:05)
parpe		4	94	0:21	4	94	0:33	4	118	1:43
t18		6	200	1:46	7	160	1:11	6	176	2:58
s4md		25	225	8:12	(26)	(113)	(3:47)	(25)	(181)	(13:00)
sp7	440 pins	11	391	1h12	11	395	1h02	(25)	(150)	(4:05)
t18	40 kgates	3	144	3:09	3	135	0:46	3	165	2:03
s4md		9	394	6:08	8	402	3:58	(25)	(181)	(13:00)

The comparative results of HDP and hMETIS will demonstrate, whether the high quality of the partitions, resulting in low pin counts, will be in itself sufficient for addressing limited architectures.

All measurements were done on a SUN SPARC with 167 MHz. The designs used as benchmarks are listed in Table 2. We used hierarchical netlists in the EDIF 2 0 0 format [8], many of them taken from industry projects.

Table 2: Benchmark designs

name	*size [gates]*	*primary ports*
watch	2752	16
sab	4858	149
se	9184	94
iir	12357	52
fidct	30445	170
t17	30960	136
sp7	51285	157
parpe	73472	94
t18	79914	61
s4md	180900	112

In the following sections, partitioning results are presented for a selection of different emulator architectures. They represent typical scenarios from real life applications. If an algorithm was unable to find a solution or if the final routing step failed, the results are shown in brackets. The value of b describes the number of blocks, øp the average number of pins per block and t gives the elapsed time.

4.1 Complete Interconnect

These structures connect FPGAs via crossbar chips or via a partial crossbar structure. This is the only architecture which is directly addressed by the comparative algorithms. The results presented in Table 1 indicate the competitiveness of APA even for this kind of interconnect structure. This is applicable to the number of created blocks as well as to the required runtime. One exception is the case where no solution may be found, in this case for the design ‘sp7’. APA tries to optimize the result until all steps in the algorithm are completed. This results in a significantly longer time.

4.2 Limited Interconnect Resources

One common example for limited interconnect resources is the division of emulation systems into several boards. For example, in order to increase the capacity of an APTIX emulation system [1], two of them may be combined as shown in Figure 4, where the squares represent FPGAs and the circles FPICS. The following results of HDP and hMETIS show, that neglecting the limitations, valid partitions are not guaranteed. In Table 3, the first two systems contain 10 FPGAs, the second two systems 16 FPGAs.

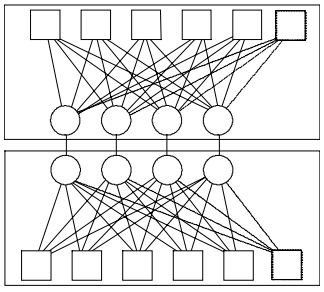

Figure 4: Combination of two APTIX systems

Table 3: Connected APTIX-Systems

des.	APA			HDP			hMETIS		
	b	øp	t[min]	b	øp	t[min]	b	øp	t[min]
parpe	8	106	1:17	8	94	0:32	(10)	(178)	(3:58)
t18	9	146	2:01	(8)	(148)	(1:10)	(10)	(180)	(3:43)
sp7	(./.)	(./.)	(2h23)	(11)	(395)	(1h02)	(16)	(283)	(3:22)
s4md	11	370	10:24	(11)	(356)	(3:57)	(16)	(250)	(11:22)

The netlist ‘sp7’ is too complex to be partitioned successfully by any of the three algorithms. The other netlists could all be successfully partitioned by APA, wheras out of the six runs of HDP and hMETIS only one resulted in a partition which could be mapped onto the emulation system.

4.3 Directly Connected Chips

The WEAVER board [9] is a system with an architecture consisting of FPGAs which are connected directly without programmable routing circuits. In the following example two boards have been combined as shown in Figure 5. The partitioning results are presented for the design ‘iir’. It is assumed that pin multiplexing will be applied to virtually increase the number of available pins while sacrificing emulation speed [10].

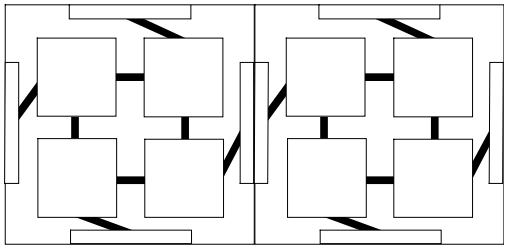

Figure 5: Combination of two WEAVER boards

Table 4: Directly connected chips

x	APA			HDP			hMETIS		
	b	øp	t[min]	b	Øp	t[min]	b	øp	t[min]
1	(./.)	(./.)	(2:53)	(7)	(103)	(0:42)	(8)	(108)	(0:26)
2	7	181	1:49	(7)	(101)	(0:19)	(8)	(108)	(0:26)
4	6	247	4:44	6	112	0:19	8	108	0:26

The results in Table 4 demonstrate that without the help of pin multiplexing (x=1), neither of the three algorithms is able to find a valid partition. With the use of multiplexing, for APA a factor x=2 is sufficient, while the other two algorithms require a factor x=4. The maximum achievable emulation frequency, under neglection of other effects, will be twice as high with the use of APA than with the use of HDP or hMETIS.

4.4 Heterogeneous Systems

In the following example, an APTIX system has been equipped with two different types of FPGAs. This is a realistic situation, as the system enables the user to reuse existing hardware in combination with more recent types of FPGAs to provide enough capacity for larger designs. In this case, each FPGA has 240 pins and the size as indicated in Figure 6.

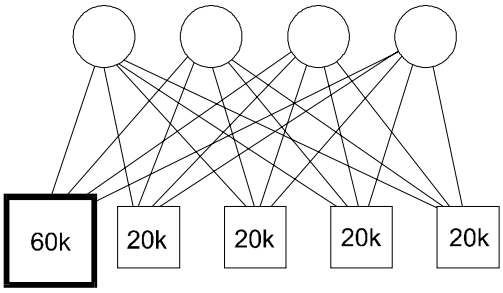

Figure 6: Heterogeneous system

Table 5: Partitioning for heterogeneus systems

FPGA	APA		HDP		hMETIS	
	Pins	Gates	Pins	Gates	Pins	Gates
1	139	33950	132	20802	157	23682
2	168	11067	132	20802	200	15765
3	217	19093	203	16804	156	9497
4	224	15804	117	10158	91	10081
5	0	0	232	11348	141	20889
Σ	748	79914	816	79914	745	79914
pro FPGA	187	19979	163	15983	149	15983
time	**1:42**		**1:11**		**2:45**	
result	**succesful**		**failed**		**failed**	

Neither HDP nor hMETIS are able to specifically address this case. As shown in Table 5, APA achieves a valid partition with only four FPGAs. In this case, the exceeding of the maximum values was comparably small. For heterogeneous structures with larger differences in FPGA sizes or more than two different types, the advantage for APA will be even bigger.

4.5 Example of Improvement by Combination

The following case gives an example for the initially stated hypothesis that overestimating given pin restrictions will yield better results than just using the exact limits.

In the described subproblem, there are two FPGAs present, with 120 pins each. The graph is clustered by applying HDP, using the given FPGA parameters as limits. As a result, the six cluster nodes of Table 6 have been created, which have to be distributed into the two FPGAs.

Table 6: Blocks created by HDP

Name	*Pins*	*Gates*
/IIR/MAIN_ADD_88_PLUS_1	114	2587
/IIR/MAIN_MUL_83_MULT	115	4618
/IIR/MAIN_MUL_71_MULT	104	4596
/IIR/INST1	89	251
/IIR/INST3	108	199
/IIR/INST4	70	106
Σ	600	12357

The demonstrated result of clustering the graph would be the final result when using HDP to partition the netlist. This means that six FPGAs would be required.

APA uses the created blocks as nodes and places them into the two target FPGAs. After the placement, the resulting blocks exceed the allowed pin limits (see Table 7).

Table 7: Result after prepartitioning

Block	*Pins*	*Gates*
1	168	7404
2	169	4953
Σ	337	12357

This is the basis for the optimization step. It finally leads to a number of cut nets, which is clearly beneath the allowed limit as shown in Table 8.

Table 8: Result after optimization

Block	*Pins*	*Gates*
1	75	7211
2	78	5146
Σ	153	12357

As a result, two FPGAs are sufficient for emulating the netlist instead of six FPGAs, required by HDP. This shows that it is advantageous to initially exceed the number of available pins and reduce them later in optimization steps. The demonstrated effect can regularly be observed with various designs and architectures.

5 Conclusions

We presented a novel approach for partitioning netlists for emulation. The algorithm is able to take the formal description of the target architecture into account and can adapt the partitioning result to this exact description. It makes use of existing algorithms to cluster the netlist and to optimize the result in combination with a new approach of finding a starting solution with the help of a two-dimensional placement.

The experimental results show that the algorithm is competitive for architectures with full interconnect with respect to the required runtime and FPGAs. The special strength of the algorithm lies in the application for architectures with limited interconnect resources, where the other algorithms are not able to consistently deliver partitions. Results for a variety of different architectures have demonstrated the advantages.

The algorithm facilitates the use of emulation and rapid prototyping, as it is not necessary to have a full interconnect structure between FPGAs. Instead, it is possible to reuse existing hardware or create architectures which require a reduced hardware effort.

References

[1] Aptix: “MP4 System Explorer User’s Manual”, 1998.

[2] Varghese, J.: “An Efficient Logic Emulation System”, in: IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 1, No. 2, June 1993, pp. 171-174.

[3] Harbich, K.; Hoffmann, H.; Barke, E.: “A New Hierarchical Graph Model for Multiple FPGA Partitioning”, Proceedings of WDTA, June 1998.

[4] Behrens, D.; Harbich, K.; Barke, E.: “Hierarchical Partitioning”, ICCAD ’96: International Conference on Computer Aided Design, San Jose, 1996, pp. 470-477.

[5] Fiduccia, C.; Mattheyses, R.: “A Linear-Time Heuristic for Improving Network Partitions”, Proc. Design Automation Conf., 1982, pp. 175-181.

[6] C. Lee, "An Algorithm for Path Connections and its Applications", IRE Transactions on Electronic Computers, Sept. 1961, pp. 346-365.

[7] Karypis, G.; Aggarval, R.; Kumar, V.; Shekhar, S.: “Multilevel Hypergraph Partitioning: Application in VLSI Domain”, 34th IEEE/ACM Design Automation Conference, 1997, pp. 526-529.

[8] EDIF Version 2 0 0. EIA Interim Standard No. 44, 1987.

[9] Koch, G.; Kebschull, U.; Rosenstiel, W.: “The Weaver Prototyping Environment for Hardware/Software Co-Design and Co-Debugging”, DATE 98, Designer Track, Paris, 1998.

[10] Babb, J. et al.: “Logic Emulation with Virtual Wires”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 16, No. 6, 1997, S. 609–626.