Multiway Netlist Partitioning onto FPGA-based Board Architectures

U. Ober, M. Glesner
Institute of Microelectronic Systems, Darmstadt University of Technology,
Karlstr. 15, 64283 Darmstadt, Germany

Abstract

FPGAs are well accepted as an alternative to ASICs and for rapid prototyping purposes. Netlists of designs which are too large to be implemented on a single FPGA, have to be mapped onto a set of FPGAs, which could be organized on an FPGA board containing various FPGAs connected by interconnection networks. This paper presents an efficient approach to the problem of multiway partitioning of large FPGA netlists onto heterogeneous FPGA boards. To optimize the resulting partitioning with respect to the target architecture, our algorithm is able to consider the board architecture.

1. INTRODUCTION

Field programmable gate arrays (FPGAs) combine the flexibility of mask programmable gate arrays (MPGAs) and "time to market" advantages of programmable logic devices (PLDs). For this reason, they are often used for rapid prototyping purposes.

Upon specification, a design is transformed into a netlist for FPGA implementation. If the logic capacity of a single FPGA is insufficient for the entire design, its netlist has to be partitioned into smaller pieces. FPGA-based rapid prototyping boards have been developed to implement large designs. One example is the Quickturn emulation system containing several FPGA based boards. These are made up of Xilinx LCAs and routing devices. The system allows validation of the design and reprogrammability ensures a quick reimplementation of the corrected design in case of malfunction. Partitioning on a rapid prototyping board can be performed by hand, automatically, or interactively. Manual partitioning is work and time consuming, thus, with increasing design complexity, there is a growing need for automatic partitioning tools.

Partitioning can be performed before or after technology mapping. During technology mapping, logic gates are clustered to logic modules to be mapped onto an FPGA's configurable logic blocks (CLBs). If partitioning is executed before technology mapping, the number of logic modules used for this design has to be estimated. This number depends on many variable factors, e.g. used technology mapper, optimization target, logic complexity of every subcircuit, thus, partitioning after technology mapping is more efficient.

Finding a circuit partitioning with a minimum number of interconnecting nets between the subsets is called mincut problem. The mincut partitioning problem with constraints concerning the size of subcircuits is proven to be NP-hard [1]. To alleviate the complexity problem, a number of mincut partitioning heuristics have been developed.

Group migration methodes include the Kernighan Lin and the Fiduccia Mattheyses bipartitioning heuristics [2, 3]. These heuristics have been improved with cell replication by Kring and Newton [4]. Metrical allocation methods do not cut the circuit but cluster nodes together. The metrics used for this method usually use eigenvalues and eigenvectors of matrices obtained from the graph representing the netlist to be partitioned [5, 6, 7]. Other methods rely on simulated annealing, such as the one developed by Green and Supowit [8].

A mincut partitioning heuristic with cost minimization was developed by Kuznar, Brglez, and Kozminski [9, 10] realizing multiway partitioning by recursively executing a bipartitioning heuristic. This approach works under the condition that a heterogeneous board with various FPGA types will be assembled specifically for this application, whereby the costs of the devices should be minimized.

In our approach a rapid prototyping system with an FPGA based ASIC emulation board already exists. Thus, the number and types of FPGAs and the interconnection network are predefined. To partition the netlist so as to fit onto this board we consider all devices at once, unlike recursive bipartitioning which considers only one device in every iteration. The processed device will be considered no more at a later time in the partitioning process. If no interconnection network has to be taken to account, the recursive scheme yields good results. However, due to the local nature of its transformations, it insufficiently covers the search space. Thus, introducing additional constraints on the size and topology of the interconnecting network usually yields suboptimum results. By considering all devices during the complete partitioning process, we expand the search space sufficient to find a suitable partition matching the interconnection network of the board.

We propose different FPGA board configurations. They contain various FPGAs of the same device family

and eventually a routing device. These boards are well suited for ASIC emulation purposes. Some examples for applications of these boards include rapid prototyping [11], parallel hardware and software development [12], algorithm accelleration [13], and programmable coprocessors [14]. Our algorithm allows the specification of different board architectures matching one of the basic models described in section 2 and is able to partition an FPGA netlist onto the specified board.

This paper is organized as follows: In section 2 the main board architectures are presented. A formalised problem definition is given in section 3. Section 4 describes the partitioning process and section 5 presents experimental results. A short summary concludes the paper.

2. BOARD-ARCHITECTURES

We consider two basic models of board architectures: pure FPGA architectures, boards containing a routing device in addition to the FPGAs. They can differ in the number and type of FPGAs and the interconnection network. These architectures were extracted for typical applications in the fields of ASIC emulation, rapid prototyping, etc.

A. Pure FPGA-Architecture

Pure FPGA architecture, as shown in figure 1, contain a certain number of FPGAs. The devices can be of different parttypes, but of the same device family. Adjacent devices are connected via local wires. Some or all devices can have external wires passing the board's boundaries to connect the board to other hardware. In addition, there exists a bus to create multi-point connections or to supply connections between devices which are not directly linked via local wires. The interconnection network can differ in the number of local, outside, and bus connections. But the sum of all connections of an FPGA must not exceed the number of its IOBs. The number of buslines is also variable.

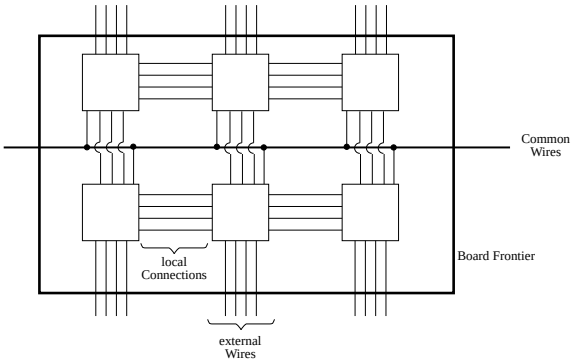

Figure 1: Pure FPGA Architecture

Due to the short distance local connections, boards with pure FPGA architecture are well suited for applications with strong timing constraints.

B. Board Architecture with Routing Device

Figure 2 shows a board architecture with the interconnect topology is created by a special routing device. All FPGAs are only connected with this device, because no interconnections between the FPGAs exist. The routing device also supplies all outside connections. A common routing device is the Aptix Field Programmable Interconnect Component. The interconnection network can differ in the number of outside and FPGA-routing device connections.

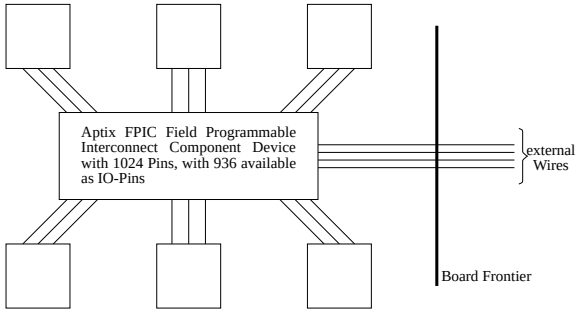

Figure 2: Board with a Routing Device

Typical applications for those boards are netlists with high complexity of interconnection structure.

3. PROBLEM DEFINITION

Given is an FPGA netlist $N=(V,E)$, where V is a set of vertices and E a set of hyperedges, and a board $B \in \{B_{FPGA}, B_{Rout}\}$. The set of the netlist's vertices $V=(X,Y)$ is composed of a set X of internal nodes representing the logic modules to be mapped onto the CLBs and set Y of terminal nodes, which reflect the IO ports to be mapped onto the IOBs. The set of hyperedges is defined as $E=\{(v_1,\ldots,v_n) \mid v_1,\ldots,v_n \in V\}$.

A board B can contain only FPGAs (B_{FPGA}) or a routing device in addition (B_{Rout}). A board is described by its devices and the interconnection network. The devices of a pure FPGA board $B_{FPGA}=(F,W_F)$ are only FPGAs $F=\{(C,I)\}$ which are described by a set C of CLBs and a set I of IOBs. The interconnection network $W_F=(L,S)$ comprises two kinds of nets, a set L of local connections defined as $L=\{(i_j,i_k) \mid f_j=(i_j,c_j), f_k=(i_k,c_k); f_j,f_k \in F\}$ of two-point connections between the IOBs of two FPGAs and a set S of shared multi-point connections via bus defined as $S=\{(i_k,\ldots,i_n) \mid f_j=(i_j,c_j), j=k,\ldots,n, f_k,\ldots, f_n \in F\}$ and $|S| \le S_{max}$, where S_{max} is the number of buslines on the board which can be used.

The second architecture $B_{Rout}=(D,W_R)$ consists of two kinds of devices $D=(R,F)$, where the routing device $R=\{p\}$ is described by its available routing pins, and only one kind of interconnect $W_R=\{(i,p) \mid i \in F, p \in R\}$.

Let $f_j,\ f_k \in F, j \neq k,\ L_C(f_j)$ denote the lower bound of available CLBs of FPGA f_j and $U_C(f_j)$ and $U_I(f_j)$ denote the upper bound of available CLBs and IOBs. The upper

bound of the available routing device pins is specified by $U(R), p \in R$. $C(L_{jk})$ denotes the number of local connections between the IOBs of f_j and those of f_k and $C(S_j)$ specifies the number of IOBs of f_j which can be connected to the bus. $C(O_j)$ indicates the number of outside connections of f_j. The number of connections between the routing device R and f_j is declared by $C(R_j)$. $L_{jk} \subset L$ denotes the local connections between f_j and f_k and $W_j \subset W_R$ the interconnections between f_j and the routing device.

A mapping $M_V: V \rightarrow D$ of the vertices V onto the devices $D=(F,R)$ is defined as $M_V(v)=d$, where d is the device v is mapped onto, and the mapping $M_E: E \rightarrow W$ of the hyperedges E to the interconnection $W \in \{W_F, W_R\}$ as $M_E(e)=\{d \in D \mid e=(v_1, \ldots, v_n) \wedge \sum_{m=1}^{n} map(v_m, d) > 0\}$, where the boolean function map is $map(v,d) = \begin{cases} 1 & \text{if } v \text{ is mapped onto } d \\ 0 & \text{else} \end{cases}$, $v \in V, d \in D$. The cut set $CUT(d_j, d_k)$ between the devices d_j and $d_k \in D$ is specified as $CUT(d_j, d_k) = \{e \in E \mid e = (v_1, \ldots, v_n) \wedge \sum_{m=1}^{n} map(v_m, d_j) > 0 \wedge \sum_{m=1}^{n} map(v_m, d_k) > 0\}$.

A partitioning is a projection $P: N \rightarrow B$ of the vertices V of the netlist N onto the devices F and R of the board B meeting the following constraints:

- The number of internal nodes $x \in X$ mapped onto the CLBs $c \in C$ of an FPGA is within the interval from the lower LC(f) to the upper bound $U_C(f)$ of CLB usage:
  $L_C(f) \leq |\{x \in X \mid P(x)=c, f=(c,i)\}| \leq U_C(f), \forall f \in F$
- The number of internal nodes $y \in Y$ mapped onto the IOBs $i \in I$ does not exceed the upper bound of IOBs:
  $|\{y \in Y \mid P(y)=i, f=(c,i)\}| \leq U_I(f), \forall f \in F$
- The number of internal nodes $y \in Y$ mapped onto the IO pins $p \in R$ does not exceed the upper bound of pin usage of the routing device R:
  $|\{y \in Y \mid P(y)=R\}| \leq U(R)$
- The size of the cut set $CUT(j,k)$ between two devices d_j and d_k does not exceed the number of local connections L_{jk} between these devices or the allowed number of interconnections W_j between the f_j and R, if $d_k=R$:
  $|CUT(j,k)| \leq \begin{cases} |L_{jk}| & \text{if } B_{FPGA} \\ |W_j| & \text{if } B_{Rout} \quad (d_k = R) \end{cases}$
- The maximum number of buslines S_{max} is not exceeded.

4. PARTITIONING

We will now look at the algorithms to partition a netlist onto a board. The netlist $N=(V,E)$ of a complex application is given and we assume that technology mapping has already been applied. The circuit is too large to be mapped onto one single FPGA. Further a prototyping board $B \in \{B_{FPGA}, B_{Rout}\}$ is available containing several FPGAs and possibly a routing device. The components are arranged in a predefined way and connected with given wires $W \in \{W_F, W_R\}$.

The problem is to partition the netlist that way, that every subcircuit can be mapped onto one of the board's FPGAs and the connections between the subcircuits match the interconnection network on the board. The partitioning $P: N \rightarrow B$ must comply to the following constraints:

- The FPGAs on the boards should be efficiently exploited to minimize the interconnecting nets and the delays they cause. A lower bound of logic block usage $L(c_j)$ will ensure this.
- Every FPGA used and the routing device have to be routable. A usage of 100% of logic blocks and of routing pins, respectively, generally leads to unroutability. Therefore, an upper bound for logic block usage $U(c_j)$ and one for routing pin usage $U(p)$ are given.
- An FPGA board B has a specific interconnection network $W \in \{W_F, W_R\}$. The nets in the cut sets of a partition $CUT(j,k)$ have to be mappable onto this architecture. Not every efficient partition P can be mapped onto the board, because the board's routing architecture W not necessarily contains all connections needed by the partition. That means that the interconnection network has to be considered during the partitioning process.

The algorithms of the system have to partition the netlist suitable for the given prototyping board architecture. The particular board architecture is read in from an architecture file. The partitioning process itself is a multiway partitioning heuristic using basic elements of the Fiduccia/ Mattheyses bipartitioning heuristic [3] improved by the cell replication by Kring and Newton [4].

A. The Basic Algorithm

First, the board architecture specification is read in from the architecture file to invoke the appropriate partitioning algorithm. An estimation is performed to check, whether the logic capacity of the board's devices is sufficient to hold the entire netlist.

To yield an initial partition, the devices are considered successively. The actual device f_j is filled up with internal nodes until the upper bound of CLB usage $U(c_j)$ is reached $\sum_{x \in X} map(x, f_j) = U(c_j)$. The nodes to be moved are selected using the gain principle of the Fiduccia/ Mattheyses heuristic. The improvement of the solution considering the requirements of the different board architectures is described in the following subsections.

B. Pure FPGA Architecture

The cost of a hyperedge $e \in E$ under a mapping M is defined as $cost(e,M) = \begin{cases} |M(e)| & \text{if } |M(e)|>1 \\ 0 & \text{else} \end{cases}$. and the cost of a M is $cost(M) = \sum_{e \in E} cost(e, M)$. The set $V \times F = \{(v,f) \mid v \in V, f \in F\}$ is build. An improving action $A_I = (M_I, R_I, D_I)$ can be:

- a move $M_I: (V,F) \rightarrow (V,F)$, where $v \in V$ is moved from device f_i to f_j; $f_i, f_j \in F$, $(v,f_i) \mapsto (v,f_j)$, if (v,f_j) is not locked,

- a replication $R_I:(V,F)\rightarrow(V,F)x(V,F)$, where $v\in V$ is replicated from device f_i to f_i and f_j; $f_i,f_j \in F$, $(v,f_i)\mapsto((v,f_i),(v,f_j))$, if (v,f_j) is not locked,
- a dereplication $D_I:(V,F)x(V,F)\rightarrow(V,F)$, where $v\in V$ is dereplicated from device f_i and f_j to f_i; f_i, $f_j\in F$, $((v,f_i),(v,f_j))\mapsto(v,f_j)$.

After performing the action M_I or D_I, the left pair (v,f_i) is locked, so vertice v cannot be moved back or replicated again to device f_i. The gain of action A_I $M\mapsto M'$ is $gain(v,f_i,f_j)=cost(M)-cost(M')$, $gain(e,v,f_i,f_j)=cost(e,M)-cost(e,M')$ denotes the part of e to the gain of the action $M\mapsto M'$. The gain of an action $(v,f_i)\mapsto(v,f_j)$ is $gain(v,f_i,f_j)=\sum_{e\in E_v} gain(e,v,f_i,f_j)$. The action with the highest $gain(v,f_i,f_j)$ is executed.

If all pairs (v,f) are locked, the best configuration which occured during the current iteration is stored and used as initial partition of the next improvement iteration. This configuration is checked, if it is suitable. In the case that it is not, it will be abandonned. Otherwise, it will be compared with the previous best suitable configuration . The best of both is stored as the new best configuration and the other one is abandonned.

The iteration stops, if no further improvement is obtained. To escape from local minima, the whole procedure is iterated five times which has proved to been a good compromise for the solution quality/run time trade-off.

C. Prototying Boards with a Routing Device

To partition a netlist suitable for prototyping boards containing one routing device, the interconnection network need not to be considered since the routing device creates all interconnection of the FPGAs. However, the number of available IO pins of the routing device is limited. To keep the device routable, not all available IO pins can be used. The sum of all interconnections between the FPGAs and the routing device is bound by the upper bound of IO pin usage $U(p)$ of the routing device. The number of interconnections to a single FPGA f_j is further bound by the number of IO pins $U(i_j)$ of f_j. So the number of allowed nets in the cut set $CUT(j, R)$ is restricted to

$$CUT(j,R) = \frac{U(i_j)}{U(I)} * U(p),$$

where $U(I)$ is the sum of all IOBs on the board.

No terminal nodes must be moved to a subcircuit to be mapped onto an FPGA and no internal node must be moved to the subcircuit being mapped onto the routing device. The only terminals being mapped onto the IOBs of the FPGAs are those terminals needed as IO ports for the interconnections between the FPGAs and the routing device. The original terminal nodes Y remain in the rest, i.e. are mapped onto the routing device $\sum_{y\in Y} map(y,R) = |Y|$ and all internal nodes X are distributed to the FPGAs $\sum_{f\in F}\sum_{x\in X} map(x,f) = |X|$.

With the initial partition as the starting point we apply an improvement procedure on the subnetlists. We build every combination of two devices. For every pair of devices we perform the improvement strategy of the Fiduccia/ Mattheyses heuristic, improved by cell replication according to Kring and Newton, considering the upper and lower bounds of CLB usage $L(c)$ and $U(c)$. If the overall solution has improved, this strategy is applied anew. As shown in algorithm 1, the whole procedure is iterated five times to prevent local minima.

```
1  procedure improve
2   variables
3    Sol_old,Sol_new=array[1.n]of|CUT(j,R)|,j=1,...,n
4    i, j = loop variables
5   begin
6    Sol_new = <Cut sets of initial partition>
7    for 5 times do
8     Sol_old =(∞,...,∞);
9     while Sol_old > Sol_new do
10     Sol_old = Sol_new
11     for i=1 to n-1 do
12      for j=i+1 to n do
13       Sol_new=<Improvement of CUT(i,R), CUT(j,R) by
                  moving, replicating, and dereplicating
                  internal nodes between f_i, f_j according to
                  gain principle of Fiduccia/Mattheyses and
                  Kring/Newton heuristic>
14      end for
15     end for
16    end while
17    Sol_new = Sol_old
18   end for
19  end improve
```

Algorithm1: Improvement Strategy for Boards with a Routing Device

5. RESULTS

We specified four board configurations composed of Xilinx LCAs of the XC3000 family. The first two boards contain the Aptix AX1024D routing device with 936 available IO-pins in addition. The number of CLBs and IOBs of the Xilinx XC3000 LCAs are summarized in table 1.

LCA	#CLB	#IOB
XC3020	64	64
XC3030	100	80
XC3042	144	96
XC3064	224	120
XC3090	320	144

Table 1: Xilinx LCAs of the XC3000 Family

We partitioned combinational and sequencial circuits from the MCNC benchmark Partitioning93 set. The chosen benchmarks with their number of CLBs, IOBs and nets are listed in table 2.

Benchmark	#CLB	#IOB	#Nets
c3540	283	72	489
c5315	377	301	699
c6288	833	64	1472
c7552	489	313	921
s5378	381	86	628
s9234	454	43	716
s15850	842	102	1265

Table 2: Benchmark Circuits

Board 1 (results shown in table 3) contains four LCAs of different sizes. It has a logic capacity of 612 CLBs, thus, it is insufficient for the Benchmarks c6288, c7552, and s15850. Table 3 shows the CLB utilization of the used LCAs. The utilization is between 30 and 90% for single FPGAs and between 62 and 83% for all used board resources. The high utilization of the routing device IO-pins with the combinational circuit c5315 shows that this circuit has a very complex interconnection structure.

Board 1	c3540	c5315	s5378	s9234
XC3064	89%	65%	88%	82%
XC3042	57%	50%	69%	81%
XC3042		90%	87%	83%
XC3030		30%		65%
Logic Capacity	368	612	512	612
Used Board LCAs	77%	62%	83%	79%
AX1024D	21%	72%	37%	29%

Table 3: Utilization Rates of Board 1

The logic capacity of the second board, composed of six LCAs, is 1376 CLBs, as shown in table 4. The smaller circuits c3540, s5378, and s9234 have not been investigated for board 2, as these would fit onto one or two FPGAs. The resource utilization numbers are listed in table 4.

Board 2	c5315	c6288	c7552	s15850
XC3090	16%	80%	32%	36%
XC3090	72%	80%	34%	52%
XC3064	46%	80%	75%	47%
XC3064		63%	49%	90%
XC3042				90%
XC3042				88%
Logic Capacity	864	1088	1088	1376
Used Board LCAs	44%	77%	45%	61%
AX1024D	73%	42%	83%	81%

Table 4: Utilization Rates of Board 2

The CLB utilization of a single FPGA is between 16 and 90%. The circuits with complex interconnection network c5315 and c7552 yield low utilization (between 16 and 75%) and the utilization of used resources here is also low (44 and 45%). Complex interconnection structure needs a high number of IOBs. Thus, the usage of CLBs for circuits with complex interconnection structure cannot be as high as for those circuits with average interconnection complexity. The CLB utilization of the other circuits are between 36 and 90% of a single FPGA and between 61 and 77% of the used board resources.

Due to the strong constraints given by the predefined interconnection structure, the utilization rate of the used board LCAs for pure FPGA architecture is not as high as for those boards with a routing device. The average of CLB utilization of board 1 and 2 (66%) is about 20 percent higer than the utilization of pure FPGA board's CLBs (45,5 %).

Board 3 contains four XC3090 LCAs and board 4 six different LCAs of the XC3000 family. The interconnection network is given in upper triangular matrices, as shown in tables 5 and 6. The number of buslines is 64.

Board 3	D0	D1	D2	D3	BUS	OUT
XC3090=D0	-	36	-	36	36	36
XC3090=D1		-	36	-	36	36
XC3090=D2			-	36	36	36
XC3090=D3				-	36	36

Table 5: Interconnection Structure of Board 3

The first four or six resp. columns denote the number of local connections between the FPGAs. The connections of the FPGAs to the bus and their outside connections are listed in the last two columns.

Board 4	0	1	2	3	4	5	BUS	OUT
XC3042=0	-	10	-	-	-	20	40	26
XC3064=1		-	20	-	40	-	35	15
XC3042=2			-	20	-	-	35	21
XC3042=3				-	15	-	40	21
XC3064=4					-	10	40	15
XC3042=5						-	40	26

Table 6: Interconnection Structure of Board 4

Board 3 has a logic capacity of 1280 CLBs and the interconnection network is composed by equal distribution of all possible connections (local, bus, and outside). The outside connection capacity is 144 IOBs.

The partitioned benchmarks yield utilization rates of 4 to 90% for a single FPGA and 23 to 66% for all used board FPGAs, as listed in table 7. The large span of CLBs utilization for single LCAs shows that in four FPGA constellation normally the first FPGA can contain a high number of CLBs, but due to the interconnection constraints the further FPGAs get lower rates. The utilization rate of the interconnection network is between 50 and 62%.

Board 3	c3540	c6288	s5378	s9234
XC3090	71%	89%	90%	85%
XC3090	8%	82%	17%	48%
XC3090	6%	75%	13%	5%
XC3090	5%	13%		4%
Logic Capacity	1280	1280	960	1280
Used Board LCAs	23%	66%	40%	36%
Connections	61%	62%	58%	50%

Table 7: Utilization Rates of Board 3

Although board 4 contains more FPGAs than board 3, its logic capacity (1024 CLBs) is lower, as well as its number of outside connections (124 IOBs). The CLB utilization rate is 28 and 82% of used board LCAs. The distribution, as to see in table 8, shows a high CLB utilization rate for the large devices in the middle and low utilization of the smaller devices on the edges of the board. The utilization of the interconnection network between 57 and 69% is comparable with that of board 3.

Board 4	c3540	c6288	s5378	s9234
XC3042	3%	89%		37%
XC3064	77%	89%	71%	84%
XC3042	11%	89%	22%	13%
XC3042	6%	33%	8%	3%
XC3064	35%	89%	79%	82%
XC3042	1%	89%	1%	4%
Logic Capacity	1024	1024	880	1024
Used Board LCAs	28%	82%	44%	45%
Connections	58%	69%	60%	57%

Table 8: Utilization Rates of Board 4

6. CONCLUSION

We presented a partitioning algorithm which partitons a netlist suitable for the special requirements of a FPGA based rapid prototyping board. The board architectures we consider are derived for typical applications, like rapid prototyping, ASIC emulation, etc.

We partitioned seven benchmark circuits from the MCNC benchmark Partitioning93 set onto four board configurations. As shown in the experimental results, our algorithm is able to efficiently partition a netlist suitable for the specified board architecture whenever the logic and outside connection capacity of the board is sufficient and yield good utilization of the used FPGAs.

We are currently working on a critical path handling to address timing issues crucial for finding partitions that allow maximum system performance. In the future we want to propose a third board architecture that contains more than one routing device. Then, we will be able to implement on such a board designs with a interconnection structure too complex to be mapped onto the single routing device architecture as presented in section 2.B.

7. REFERENCES

[1] M. Garey, D. Johnson: Computers and Intractibility; A Guide to the Theory of NP-Completeness, W.H. Freeman & Company, 1979.

[2] B. W. Kernighan, S. Lin: An Efficient Heuristic Procedure for Partitioning Graphs; Bell System Technical Journal, 49:291-307, 1970.

[3] C.M. Fiduccia, R. Mattheyses: A Linear Time Heuristic for Improving Network Partitions; Proc. of the 19th Annual Design Automation Conference, pp. 241-247 (175-181), July 1982.

[4] C. Kring, A. R. Newton: A Cell Replicating Approach to Mincut-Based Circuit Partitioning; Proc. IEEE International Conference on Computer-Aided Design, Santa Clara, California, November 1991.

[5] S. W. Hadley, B. L. Mark, A. Vannelli: An Efficient Eigenvector Approach for Finding Netlist Partitions; IEEE Transactions on Computer-Aided Design, 11(7):885-892, July 1992.

[6] L. Hagen, A. Kahng: Fast Spectral Methods for Ratio Cut Partitioning and Clustering; Proc. IEEE International Conference on Computer-Aided Design, Santa Clara, California, November 1991.

[7] J. Cong, L. Hagen, A. Kahng: Net Partitioning Yield Better Module Partitions; Proc. of the 29th ACM/IEEE Design Automation Conference, pp 47-52, June 1992.

[8] M. R. Green, J. Supowit: Simulated Annealing without Rejected Moves; Digest International Conference on Computer Design, pp 658-663, October 1984.

[9] R. Kuznar, F. Brglez, K. Kozminski: Partitioning Digital Circuits for Implementation in Multiple FPGA ICs; Technical Report, MCNC Center for Mikroelectronic Systems Technologies, March 8, 1993.

[10] R. Kuznar, F. Brglez, K. Kozminski: Cost Minimisation of Partitions into Multiple Devices; IEEE/ACM Proc. 30th Design Automation Conference, Dallas, Texas, June 14-18, 1993.

[11] H. J. Herpel, N. Wehn, M. Gasteier, M. Glesner: A Reconfigurable Computer for Embedded Control Applications; IEEE Workshop on FPGAs for Custom Computing Machines, Napa, California, April 5-7, 1993.

[12] J. Babb, R. Tessier, A. Agarwal: Virtual Wires: Overcoming Pin Limitations in FPGA-based Logic Emulators; IEEE Workshop on FPGAs for Custom Computing Machines, Napa, California, April 5-7, 1993.

[13] C. Iseli, E. Sanchez: Spyder: A Reconfigurable VLIW Processor using FPGAs; IEEE Workshop on FPGAs for Custom Computing Machines, Napa, California, April 5-7, 1993.

[14] P. W. Foulk: Data-folding in SRAM configurable FPGAs; IEEE Workshop on FPGAs for Custom Computing Machines, Napa, California, 1993.