Board-Level Multiterminal Net Assignment

Xiaoyu Song[1], William N. N. Hung[2], Alan Mishchenko[1], Malgorzata Chrzanowska-Jeske[1], Alan Coppola[3] and Andrew Kennings[4]

[1]Department of ECE, Portland State University, Portland, Oregon, USA
[2]Intel Corporation, Hillsboro, Oregon, USA
[3]Cypress Semiconductor, Beaverton, Oregon, USA
[4]Department of ECE, University of Waterloo, Waterloo, Ontario, Canada

ABSTRACT

The paper presents a satisfiability-based method for solving the board-level multiterminal net routing problem in Clos-Folded FPGA based logic emulation systems. The approach transforms the FPGA board-level routing task into a single, large Boolean equation with the property that any assignment of input variables that satisfies the equation specifies a valid routing. The approach considers all nets simultaneously and the absence of a satisfying assignment implies that the layout is unroutable. We use two of the fastest SAT solvers: *Chaff* and *DLM* to perform our experiments. Empirical results show that the method is time-efficient and applicable to large layout problem instances.

Categories and Subject Descriptors

B.7.2 [**Integrated Circuits**]: Design Aids – placement and routing

General Terms

Algorithms, Design, Experimentation, Measurement, Verification

1. INTRODUCTION

There has been ever-increasing interest in computing engines based on field programmable gate array (FPGA) [2]. These engines make possible high-speed reconfigurable prototyping [6] and emulation systems [15]. An FPGA-based computing system consists of multiple FPGAs which require interconnections among them. To implement circuits that cannot fit onto a single FPGA chip, the field programmable interconnect chip (FPIC) was introduced [1]. Large circuits are divided into several parts, and each part is implemented in a separate FPGA chip. An FPIC is used to interconnect these FPGAs on a printed circuit board. Each external pin (I/O pin) of the FPGA is connected to an FPIC pin through a trace on the board. An FPIC does not implement any logic function, rather, it provides interconnection paths between pins. The FPIC offers a quick and reprogrammable means for inter-FPGA connections and permits fast modification of large circuits [5].

Successful design methodologies require design verification to be an integral part of the design process. As much as 80% of time spent on designing a system is devoted to the design verification. One of the important verification methods is logic *simulation*. Logic simulation software reports on how the circuit under design will respond to a sequence of input vectors, so the designer can judge whether the circuit behaves as expected over the input sequence. As the circuit complexity increases and the time to market shortens, inadequate simulation speed becomes a major bottleneck in the design process. Currently, the fastest simulation-based verification is done with a *hardware emulator* such as *QuickTurn* [3]. A hardware emulator consists of a large number of FPGAs interconnected either directly or indirectly through FPICs [4]. A hardware emulator can be configured as the circuit under design. Even running at a few hundred kilohertz, a hardware emulator still evaluates input vectors much faster than a simulation software does (as much as 10^5 times faster). As a result, FPGA-based logic emulators can verify large designs that otherwise are not verifiable by software simulators. Therefore, FPGA-based hardware emulation is becoming indispensable in many state-of-the-art VLSI design projects [3, 5, 7].

There are two major steps in logic emulation: (1) a large design is partitioned into parts which can fit inside a single FPGA on the logic emulator [12]; (2) board-level routing is performed to connect the signals between the FPGA chips [8, 9]. In logic emulators such as the Realizer system [3] and the Enterprise Emulation system [5], the set of FPGAs implementing the logic are interconnected by a set of small full crossbars. The interconnection crossbars only connect to the FPGAs but not to each other. The I/O-pins of each FPGA are evenly divided into proper subsets. The pins of a crossbar can be connected only to the same subset of pins on each FPGA.

In this paper, we present a new satisfiability-based method for solving the board-level multiterminal net routing problem in Clos-Folded FPGA based logic emulation systems. The problem was also referred to as the "partial crossbar interconnection structure" in [3]. The problem was studied using different approaches [8, 9, 10, 13, 23], where no experimental results for large problems were reported. Our satisfiability-based approach transforms the FPGA routing task into a single, large Boolean equation with the property that any assignment of input variables that satisfies the equation specifies a valid routing. The fixed routing resources in an FPGA and their rigid structure make it particularly easy to model the problem of net assignment in the Boolean domain. The method considers all nets simultaneously and the absence of a satisfying assignment implies that the layout is unroutable.

The remainder of this paper is organized as follows. In Section 2, we review related work and formally define the problem. In Section 3, a novel approach based on satisfiability is presented. Experimental results are presented in Section 4. Section 5 concludes the paper.

GLSVLSI'02, April 18-19, 2002, New York, New York, USA.

2. PRELIMINARIES AND PRIOR WORK

2.1. Previous Work

Some heuristics were proposed for solving the BLRP in [3, 4]. Optimal algorithms for board-level routing, when all nets are two-terminal nets and the I/O-pin subset size is even, were proposed independently by Chan and Schlag [13] and Mak and Wong [9]. An $O(N^2)$-time algorithm for solving any two-terminal BLRP was presented in [9], where N is the number of nets. An I/O pin capacity constraint was proposed to assure the existence of a solution. The algorithm was based on the iterative computation of Euler circuits in graphs of BLRPs. They also proved that the multiterminal routing problem is NP-complete.

Mak and Wong [9] showed how multiterminal nets can be handled by decomposition into two-terminal nets, and how the decomposition problem can be modeled as a bounded-degree hypergraph-to-graph transformation problem where hyperedges are transformed to spanning trees. A network flow-based algorithm that solves both problems was proposed. It determines if there is a feasible decomposition and gives one whenever such a decomposition exists.

The assignment problem for multiterminal nets can be solved in two ways. One is to tackle the multiterminal nets directly. The other is to transform all multiterminal nets into two-terminal nets and solve it using a polynomial algorithm. Although the latter approach has better scalability since two-terminal nets can be routed in polynomial time, it is just a heuristic because the original problem is transformed into a different problem in which more pins are required. If the only solution to the original problem uses up all pins of all FPGAs, then a solution to the transformed problem will not be feasible to the original problem. Moreover, in practice, the utilization of logic FPGAs is usually very low, and the utilization of pins is very high. The increase in pin count can make it impossible to configure an emulation system. Therefore, it is preferable to take the former approach. In [23], the multiterminal routing problem was formulated as an integer-linear programming problem where the number of variables and runtime are both exponential.

To handle routing constraints that may arise from certain timing requirement, Mak and Wong [11] proposed a performance-driven routing algorithm for the board-level routing problem that can handle additional routing constraints and reduce the delay of the routing solutions. It is an optimization algorithm based on minimum cost flow computation.

Devadas [14] was the first to use satisfiability techniques to address the physical design problems. Satisfiability-based approaches in [18, 19] have been proposed for FPGA detailed and global routing.

2.2. Problem Formulation

The FPGAs are referred to as *chips*. Let us assume that all chips are identical and the interconnection crossbars are used only to connect to the chips but not to each other. Let F be a set of p identical FPGA chips, numbered by 1, 2, ..., p. A chip has a set of I/O ports. I/O ports of each chip are *evenly* divided into k groups of size m: $S_1, S_2, ..., S_k$. We assign a distinct type for each group, S_i, $i = 1,..., k$. We use labels: A, B, C, ..., to represent their types. An I/O port in a group S_i possesses the type of S_i. In other words, we say I/O ports of each chip are evenly divided into k groups of size m such that (i) type(S_1), type(S_2), ..., type(S_k) are pairwise distinct; (ii) size(S_1) = size(S_2) =...= size(S_k) = m.

In terms of the number of terminals per net, two kinds of board-level routing problems (BLRPs) are identified: (i) two-terminal BLRP where all nets are two-terminal nets; and (ii) multiterminal BLRP where at least one net has more than two terminals.

A *two-terminal board level routing problem* (*2BLRP*) is defined as follows. Given a set of two-terminal interchip nets $M=\{n_1, n_2, ..., n_N\}$, where $n_t = \langle i, j \rangle$, $i \neq j$, $i, j \in \{1, ..., p\}$, and $t=1, 2, ..., N$, we want to find an assignment of M to I/O ports of F such that, for each net $n_t = \langle i, j \rangle$, type($i$) = type($j$), $i \neq j$, $i, j \in \{1, ..., p\}$, and $t = 1, 2, ... , N$.

Note that no net can have its two terminals in the same chip. Any two I/O ports of the same type on different chips can be connected by a FPGA crossbar switch. Since we have a one-to-one correspondence between the terminal (the ending point of a net) and its chip, we use the chip number to identify the ends of a net.

In the following, m is the number of the pins having the same type in each chip, k is the number of types in each chip which represents the number of crossbars, and p is the number of FPGA chips. Pins of each chip are evenly routed to k crossbar switches using N nets. A 2BLRP with parameters m, k, p and N is denoted by 2BLRP(m, k, p, N). An instance of the 2BLRP(2, 2, 4, 8) is shown in Figure 1, where m = 2, k = 2, p = 4 and N = 8.

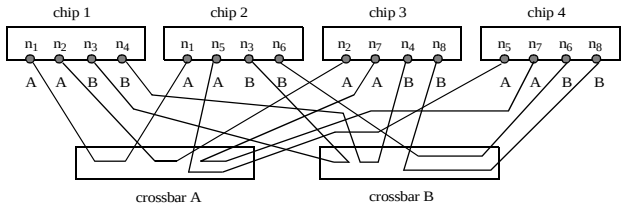

Figure 1. An instance of 2BLRP(2, 2, 4, 8).

A *multiterminal board level routing problem* (*BLRP*) is defined as follows. Given a set of multiterminal interchip nets $M=\{n_1, n_2, ..., n_N\}$, where $n_t = \{i_1, ..., i_s\}$, $i_g \neq i_h$, $i_g, i_h \in \{1, ..., p\}$, $g, h \in \{1, ..., s\}$, $t = 1, 2, ..., N$, find an assignment of M to I/O ports of F such that, for each net $n_t = \{i_1, ..., i_s\}$, type(i_g) = type(i_h), $i_g \neq i_h$, $i_g, i_h \in \{1, ..., p\}$, $g, h \in \{1, ..., s\}$, and $t = 1, 2, ..., N$.

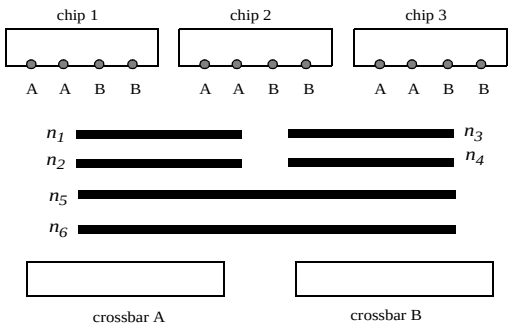

Figure 2. An instance of 2BLRP(2, 2, 3, 6).

For routability, consider 2*BLRP*(2, 2, 3, 6) in Figure 2 where the nets are divided into three groups, connecting each pair of chips, each group consisting of two nets. Without losing generality, assume that nets n_1 and n_2 connect chips 1 and 2, nets n_3 and n_4 connect chips 2 and 3, while nets n_5 and n_6 connect chips 1 and 3.

If we start routing the nets as shown in Figure 3, by connecting chip 1 and chip 2 with two nets of the same type, and next connecting chip 2 and chip 3 with two nets of another type, the problem has no solution. The remaining two nets connecting chips 1 and 3 cannot be routed, because the unused pins on these chips have incompatible type. Figure 4, on the other hand, represents a feasible solution.

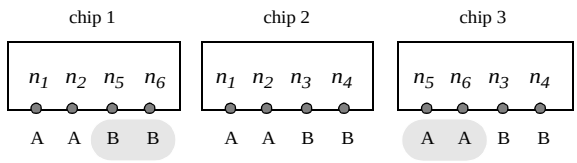

Figure 3. A unfeasible net assignment for the instance in Fig. 2.

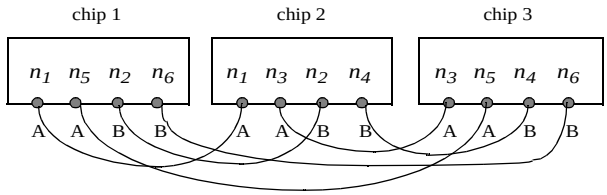

Figure 4. A feasible routing for the instance in Fig. 2.

3. BLRP(*M, K, P, N*) VIA SATISFIABILITY

3.1. Introduction

A *satisfiability problem* (*SAT*) is defined as follows. Given a set of *n clauses* $\{C_1, C_2, ..., C_n\}$ on *m* variables $x=\{x_1, x_2, ..., x_m\}$, $x_i \in \{0, 1\}$, where each clause C_i consists of only logical *OR* connected variables (*positive literals*) or *negations* of variables (*negative literals*), and a Boolean formula in conjunctive normal form (*CNF*), which is logical *AND* of clauses:

$$C_1 \wedge C_2 \wedge ... \wedge C_n \qquad (1)$$

The problem is to find an assignment of values to the variables so that (1) evaluates to be true or prove its infeasibility if (1) has no satisfying assignments. *A clause is satisfied* if at least one of its literals is true. The entire Boolean expression is satisfied if all its clauses are simultaneously satisfied, given a particular assignment to the variables. *SAT* lies at the core of many practical application domains including EDA automatic test generation, formal verification, and logic synthesis. *SAT* was one of the central NP-complete problems [17]. As the best NP-complete algorithms have exponential time complexity, NP-complete is generally considered to mark the boundary between tractability and intractability. Nevertheless, a large amount of research in recent years has shown that satisfiability problems can often be solved efficiently in practice [20, 21], and numerous solver algorithms have been proposed and implemented. A SAT solver is *complete* if it can guarantee to find a satisfying assignment if one exists or prove unsatisfiability. Otherwise it is *incomplete*.

3.2. SAT Formulation

In order to reduce BLRP to the satisfiability formula, it is necessary to encode the problem by introducing *Boolean variables* and formulating Boolean constraints in terms of *SAT clauses*. Given the problem with *N* multiterminal nets and *k* types of I/O ports, it is reasonable to introduce $N \times k$ Boolean variables to encode the problem. Each variable represents the possibility to route the given net using one of *k* possible pin types. If the variable is 1, the routing includes the given possibility; if the variable is 0, this possibility is not used.

To express the constraints, consider the set of requirements for a feasible solution. The requirements are of two types: (1) the covering constraints, and (2) the closure constraints. The first group of constraints ensures that each net is routed at least once. The second group ensures that (2a) no net is routed more than once, and that (2b) for each chip and each pin type, the number of associated nets does not exceed the number of available pins.

SAT-based formulation of the FPGA board-level routing problem consists of finding and solving a single, large Boolean equation with the property that any assignment of input variables that "satisfies" the equation specifies a valid routing. The formulation considers all nets simultaneously and the absence of a satisfying assignment implies that the layout is unroutable.

3.2.1 Covering constraints

For each net *n*, there is only one covering constraint. This constraint relates all the variables associated with this net in the following way:

$$x_n^1 \vee x_n^2 \vee \ldots \vee x_n^k = 1$$

As discussed above, this formula means that the net can be routed using any pin type.

3.2.2 Closure constraints

The first type of closure constraints requires that each net was routed no more than once. In other words, among each pair of variables (x_n^i, x_n^j), at least one is assigned to zero:

$$\overline{x_n^i} \vee \overline{x_n^j} = 1, \ \forall i, j \in \{1, 2, \ldots, k\}, i \neq j.$$

Let $x_{ab-n_i}^k$ be a Boolean variable representing a net n_i connecting chips *a* and *b* using pin type *k*. We need to introduce the set of variables X_c^k corresponding to nets connecting chip c with other chips using the pin type *k*. More formally, we have:

$$X_c^k = \left\{ x_{ab-n_i}^k \middle| (a = c) \vee (b = c) \right\}.$$

The second type of closure constraints combines all variables X_c^k that correspond to all nets connecting chip *c* with other chips using the pin type *k*. Among variables belonging to X_c^k there should be no more than *m* variables equal to 1. In other words, in every group of $m+1$ variables belonging to X_c^k, there should be at least one variable equal to 0.

To formulate these constraints, it is necessary to create the set X_c^k for each chip and each pin type, and next take all possible combinations of m+1 variables in the negative polarity. The illustration of this type of constraints is given below.

3.2.3 An Example

We use the example in Figure 2 to show our SAT formulation. The variables are introduced as follows. There are 12 variables totally. The solution shown in Figure 4 corresponds to the following assignment:

Chips 1 and 2: $x^{A}_{12-n_1} = 1, x^{B}_{12-n_1} = 0, x^{A}_{12-n_2} = 0, x^{B}_{12-n_2} = 1$.

Chips 2 and 3: $x^{A}_{23-n_3} = 1, x^{B}_{23-n_3} = 0, x^{A}_{23-n_4} = 0, x^{B}_{23-n_4} = 1$.

Chips 1 and 3: $x^{A}_{13-n_5} = 1, x^{B}_{13-n_5} = 0, x^{A}_{13-n_6} = 0, x^{B}_{13-n_6} = 1$.

The constraints can be expressed as follows.

The covering constraints for all nets:

$$x^{A}_{12-n_1} \vee x^{B}_{12-n_1} = 1;$$

$$x^{A}_{12-n_2} \vee x^{B}_{12-n_2} = 1;$$

$$x^{A}_{23-n_3} \vee x^{B}_{23-n_3} = 1;$$

$$x^{A}_{23-n_4} \vee x^{B}_{23-n_4} = 1;$$

$$x^{A}_{13-n_5} \vee x^{B}_{13-n_5} = 1;$$

$$x^{A}_{13-n_6} \vee x^{B}_{13-n_6} = 1.$$

The first set of closure constraints for all nets:

$$\overline{x^{A}_{12-n_1}} \vee \overline{x^{B}_{12-n_1}} = 1;$$

$$\overline{x^{A}_{12-n_2}} \vee \overline{x^{B}_{12-n_2}} = 1;$$

$$\overline{x^{A}_{23-n_3}} \vee \overline{x^{B}_{23-n_3}} = 1;$$

$$\overline{x^{A}_{23-n_4}} \vee \overline{x^{B}_{23-n_4}} = 1;$$

$$\overline{x^{A}_{13-n_5}} \vee \overline{x^{B}_{13-n_5}} = 1;$$

$$\overline{x^{A}_{13-n_6}} \vee \overline{x^{B}_{13-n_6}} = 1.$$

The second set of closure constraints is:

Chip 1 with pin type *A*, $X_c^k = \left\{ x^{A}_{12-n_1}, x^{A}_{12-n_2}, x^{A}_{13-n_5}, x^{A}_{13-n_6} \right\}$,

$$\overline{x^{A}_{12-n_1}} \vee \overline{x^{A}_{12-n_2}} \vee \overline{x^{A}_{13-n_5}} = 1,$$

$$\overline{x^{A}_{12-n_1}} \vee \overline{x^{A}_{12-n_2}} \vee \overline{x^{A}_{13-n_6}} = 1,$$

$$\overline{x^{A}_{12-n_1}} \vee \overline{x^{A}_{13-n_5}} \vee \overline{x^{A}_{13-n_6}} = 1,$$

$$\overline{x^{A}_{12-n_2}} \vee \overline{x^{A}_{13-n_5}} \vee \overline{x^{A}_{13-n_6}} = 1.$$

Chip 1 with pin type *B*, $X_c^k = \left\{ x^{B}_{12-n_1}, x^{B}_{12-n_2}, x^{B}_{13-n_5}, x^{B}_{13-n_6} \right\}$,

$$\overline{x^{B}_{12-n_1}} \vee \overline{x^{B}_{12-n_2}} \vee \overline{x^{B}_{13-n_5}} = 1,$$

$$\overline{x^{B}_{12-n_1}} \vee \overline{x^{B}_{12-n_2}} \vee \overline{x^{B}_{13-n_6}} = 1,$$

$$\overline{x^{B}_{12-n_1}} \vee \overline{x^{B}_{13-n_5}} \vee \overline{x^{B}_{13-n_6}} = 1,$$

$$\overline{x^{B}_{12-n_2}} \vee \overline{x^{B}_{13-n_5}} \vee \overline{x^{B}_{13-n_6}} = 1.$$

Similarly, the other four sets of closure constraints can be obtained for chips 2 and 3 with A and B, respectively, and each of them contains 4 constraints. It is easy to verify that the assignments of variables corresponding to the solution in Figure 4 satisfies the given set of constraints.

3.2.4 Complexity

The bottleneck of the proposed approach is in the number of the closure constraints of the second type. It grows exponentially with the number of nets involving the given chip. However, it is polynomial for small *m*. Assumming that all pins of the chips are used, there are $z = k*m$ nets connecting each chip. Then, the total number of clauses can be approximated as follows:

$$N \times 1 + N \times \frac{k \times (k-1)}{2} + p \times k \times \binom{k \times m}{m+1}$$

For example, if $p = 50$, $k = 4$, $m = 4$, altogether there are 800 pins. Assuming the 4-terminal nets and that all the available pins are used, there are 200 nets. The number of variables is 200 *4 = 800, the number of clauses is 200 + 1200 + 873,600 = 875,000.

4. EXPERIMENTAL RESULTS

We initially attempted to use BDD-based method to solve the Boolean equations representing BLRP. However, the BDD-based implementation failed due to the excessive BDD size for problems involving more than 60 Boolean variables. Thus, we could not solve even the smallest examples out of those that are listed in the experimental results section below.

We subsequently formulated the routing constraints as *CNF* formulas that were checked by SAT solvers. We used two of the fastest solvers [21] from the numerous available SAT solvers: *Chaff* [22] and *DLM* [20], which are complete and incomplete solvers, respectively. Chaff employs a conflict resolution scheme that is philosophically very similar to *GRASP* [21], using the same type of conflict analysis, conflict clause addition. *DLM* is a discrete Lagrange-multiplier-based global-search method for solving satisfiability problems. In contrast to clause weight schemes that rely only on the weights of violated constraints to escape from local minima, *DLM* uses the value of an objective function to provide further guidance. The dynamic shift in emphasis between the objective and the constraints is the key of Lagrangian methods. One of the major advantages of *DLM* method is that it has very few algorithmic parameters to be tuned by users and the search procedure can be made deterministic. *DLM* often performs as one

of the best existing methods and can achieve an order-of-magnitude speedup for some problems.

Our test results presented in Table 1 are of great importance since we have the first experimental results for the problem studied in [8]. *Benchmark* is the name of a generated benchmark. *P* is the number of chips. *K* is the number of pin types. *M* is the number of pins of each type. *Max* is the largest number of terminals (size) of a net. *Ave* is the average size of all nets. *Vars* is the number of variables needed to encode the problem for the *SAT* solver. *Clauses* is the number of clauses given to the *SAT* solver. *Literals* is the number of positive and negative polarity literals in all clauses. *Prep* is the time needed to transform the problem into a *SAT* instance. *DLM* is the time needed to solve the problem by *DLM* SAT solver. *Chaff* is the time needed to solve the problem by the *zChaff* SAT solver. The runtime is in seconds on 850MHz Pentium® III with 1GB RDRAM (only small part of memory has been used). The run time can be improved by fine-tuning the *SAT* solver parameters. All solutions have been automatically verified using a built-in verifier.

For the same parameters of *P*, *K* and *M*, we increase *N* (number of nets) gradually. It is pretty clear that the number of variables, clauses and literals generated in our satisfiability formulation also increases with *N*. However, the time it takes for the *DLM* SAT solver does not necessarily increase with the rising *N*. The same phenomenon can be observed for the number of chips (*P*). Increasing *P* (and *N*) results in more variables, clauses and literals. But the time it takes to solve the problem does not necessarily increase with *P*. Since *DLM* transforms the routing problem into discrete Lagrangean domain, the time it takes to solve the problem is not directly proportional to the number of constraint clauses. We have tested two sets of problems: $M = 2$ and $M = 3$. It takes longer time to solve the problem for $M = 3$ than for $M = 2$. This is understandable as more pins implies more complexity for the problem.

For the majority of cases, *Chaff* takes longer to compute than *DLM*. This does not necessarily mean that *Chaff* is in general slower than *DLM*. There are a lot of other parameters that could be tuned by these SAT checker implementations to speed up the runtime for particular types of problems. In addition, *Chaff* is a complete *SAT* checker that is able to confirm unroutability. This is very important as heuristic methods and local searches (like *DLM*) runs forever under those circumstances.

5. CONCLUDING REMARKS

We have studied a satisfiability-based method for solving the board-level multiterminal net routing problem in Clos-Folded FPGA based logic emulation systems. The approach transformed the FPGA routing task into a single, large Boolean equation with the property that any assignment of input variables that satisfies the equation specifies a valid routing. Experimental results demonstrate its time-efficiency and applicability to large layout problem instances.

Incremental redesign is an increasingly essential step in any complex design. Our formulation allows for an incremental approach to BLRP. If the problem is not satisfiable, a feasible routing does not exist. In this case, it is possible to relax some of the constraints, and remove them from the constraint database, while leaving most of the constraints intact. Since the crossbars are distributed over different locations on a board, the distances a net signal needs to travel when a net is routed via different crossbars can be largely different. For some highly critical nets, there should be some restrictions as which crossbars each can be routed through to satisfy certain timing constraints. If we are given some nets with stringent timing requirement, and thus can only be routed via certain crossbars, it is possible to tighten the constraints by assigning nets in a different way. If only a small number of nets are changed, there is no need to regenerate all the constraints in the *SAT* constraint data base.

6. ACKNOWLEDGMENTS

The first author would like to thank Dr. W. Mak for helpful discussions on the problem.

7. REFERENCES

[1] M. Khalid and J. Rose, "A Hybrid Complete-Graph Partial-Crossbar Routing Architecture for Multi-FPGA Systems," *Proc. ACM Symposium on FPGAs*, Feb 1998, pp. 45-54.

[2] S. D. Brown, R.J. Francis, J.Rose, and Z. G. Vranesic, *Field Programmable Gate Arrays*, Norwell, MA: Kluwer, 1992.

[3] J. Varghese, M. Butts, and J. Baatcheller, “An efficient logic emulation system,” *IEEE Trans. VLSI Systems*, Vol. 1(2), pp. 171-174, 1993.

[4] M. Slimane-Kadi, D. Brasen, and G. Saucier, “A fast FPGA prototyping system that uses inexpensive high-performance FPIC,” *Proc. ACM/SIGDA Int. Workshop FPGAs*, 1994.

[5] L. Maliniak, “Multiplexing enhances hardware emulation,” *Electronic Design*, pp. 76-78, 1992.

[6] S. Walters, “Computer-aided prototyping for ASIC-based systems,” *IEEE Design Test*, pp. 4-10, 1991.

[7] K. Yamada, H. Nakada, A. Tsutsui, and N. Ohta, “High-speed emulation of communication circuits on a multiple-FPGA system,” *Proc. ACM/SIGDA Int. Workshop FPGAs*, 1994.

[8] W.K. Mak, and D.F. Wong, "Board-Level Multi-Terminal Net Routing for FPGA-based Logic Emulation," *Proc. International Conference on Computer Aided Design*, pp. 339-344,1995.

[9] W.K. Mak and D.F. Wong, “On optimal board-level routing for FPGA-based logic emulation,” *IEEE Trans. CAD*, Vol. 16(3), 1997.

[10] S. Lin, Y. Lin, and T. Hwang, “Net Assignment for the FPGA-Based Logic Emulation System in the Folded-Clos Network Structure,” *IEEE Trans. CAD*, Vol. 16(3), 1997.

[11] W.K. Mak, and D.F. Wong, "Performance-Driven Board Level Routing for FPGA-based LOgic Emulation," *Proc. International Conference on Computer Design*, pp. 199-201,1998.

[12] N.C. Chou, L.T. Liu, C. K. Cheng, W.J. Dai, and R. Lindelof, "Circuit Partitioning for Huge Logic Emulation Systems," *Proc. ACM/IEEE Design Automation Conference*, pp. 244-249,1994.

[13] P.K. Chan, and M.D.F. Schlag, “Architectural Trade-offs in Field-Programmable-Device Based Computing Systems,” *Proc. IEEE Workshop on FPGAs for Custom Computing Machines*, pp. 138-141, April 1993.

[14] S. Devadas, “Optimal layout via Boolean Satisfiability,” *Proc. ACM/IEEE ICCAD*, 1989, pp. 294-297.

[15] M. Gokhale, W. Holmes, A. Kopser, S. Lucas, R. Minnich, and D. Sweely, "Building and using a highly parallel programmable logic arrays," *Computer*, Vol. 24., pp. 81-89, Jan. 1991.

[16] P. Berlin, D. Roncin, and J. Vuillemin, "Programmable active memories: A performance assessment," *Proc. 1st International Workshop on FPGAs*, Feb. 1992, pp. 57-59.

[17] M.R. Garey, and D.S. Johnson, *Computers and Intractability, A guide to the Theory of NP-completeness*, W.H. Freeman, 1979.

[18] Gi-Joon Nam, Karem A. Sakallah and Rob. A. Rutenbar, "Satisfiability-Based Layout Revisited: Detailed Routing of Complex FPGAs Via Search-Based Boolean SAT," *Proc. ACM International Symposium on FPGAs*, February 1999, Monterey.

[19] Gi-Joon Nam, Fadi Aloul, Karem A. Sakallah and Rob A. Rutenbar, "A Comparative Study of Two Boolean Formulations of FPGA Detailed Routing Constraints", *Proc. International Symposium on Physical Design (ISPD)*, April 2001, Sonoma.

[20] Y. Shang and B. W. Wah, "A Discrete Lagrangian-Based Global-Search Method for Solving Satisfiability Problems," *Journal of Global Optimization*, Kluwer, 12(1), 1998, pp. 61-99.

[21] M.N. Velev, "Effective Use of Boolean Satisfiability Procedures in the Formal Verification of Superscalar and VLIW Microprocessors," *Proc. ACM/IEEE Design Automation Conference*, June 18-22, 2002, Las Vegas, Nevada, pp. 226-231.

[22] L. Zhang, C. Madigan, M. Moskewicz, and S. Malik, "Efficient Conflict Driven Learning in a Boolean Satisfiability Solver," *Proc. ACM/IEEE International Conference on Computer-Aided Design* (ICCAD), San Jose, CA, November 2001.

[23] A. Ejnioui and N. Ranganathan, "Multi-terminal net routing for partial crossbar-based multi-FPGA systems," *Proc. ACM International Symposium on FPGA*, 1999, Monterey, CA.

Table 1. Experimental Results

Benchmark	P	K	M	N	Max	Ave	Vars	Clauses	Literals	Prep	Routability	DLM	Chaff
p020_k5_m2_n07	20	5	2	49	7	4.0	245	12039	35725	0	yes	0.03	0.68
p020_k5_m2_n08	20	5	2	52	8	3.8	260	12147	36025	0	yes	0.04	1.31
p020_k5_m2_n06	20	5	2	59	6	3.3	295	12149	35975	0	yes	0.03	2.01
p020_k5_m2_n05	20	5	2	64	5	3.1	320	12279	36325	0	yes	0.02	0.22
p020_k5_m2_n04	20	5	2	76	4	2.6	380	12411	36625	0	yes	0.02	0.16
p020_k5_m3_n14	20	5	3	55	13	5.4	275	133855	534375	0.09	yes	2.37	85.88
p020_k5_m3_n11	20	5	3	60	11	5.0	300	135340	540220	0.1	yes	0.43	100.71
p020_k5_m3_n09	20	5	3	66	9	4.5	330	132051	526950	0.09	yes	0.39	2.28
p020_k5_m3_n08	20	5	3	68	8	4.4	340	132898	530300	0.09	yes	0.32	0.33
p020_k5_m3_n07	20	5	3	77	7	3.9	385	132997	530525	0.09	yes	0.50	10.29
p020_k5_m3_n06	20	5	3	88	6	3.4	440	134218	535200	0.09	yes	0.35	83.91
p020_k5_m3_n05	20	5	3	99	5	3.0	495	134339	535475	0.1	yes	0.28	69.36
p020_k5_m3_n04	20	5	3	114	4	2.6	570	134504	535850	0.09	yes	0.30	0.99
p020_k3_m3_n08	20	3	3	36	8	5.0	108	7536	29892	0	yes	0.04	0.01
p020_k4_m3_n10	20	4	3	30	10	6.0	120	21110	84080	0	no	N/A	0.08
p020_k4_m3_n08	20	4	3	67	8	3.6	268	39409	156832	0.02	yes	0.11	0.19
p020_k5_m3_n15	20	5	3	30	15	8.0	150	91620	365910	0	no	N/A	0.01
p020_k7_m3_n08	20	7	3	105	8	4.0	735	811055	3240125	0.62	yes	3.59	207.79
p050_k5_m3_n07	50	5	3	150	7	4.0	750	244720	975030	0	no	N/A	0.01
p050_k5_m3_n08	50	5	3	171	8	4.4	855	337956	1348575	0.25	yes	2.85	109.56
p050_k6_m3_n08	50	6	3	212	8	4.2	1272	911222	3638952	0.65	yes	9.54	109.20
p050_k7_m3_n08	50	7	3	241	8	4.3	1687	2070897	8274189	1.57	yes	23.97	1453.00
p100_k5_m3_n08	100	5	3	344	8	4.4	1720	684464	2731320	0.48	yes	10.54	112.62
p150_k5_m3_n08	150	5	3	552	8	4.1	2760	1024647	4088100	0.72	yes	7.31	69.64
p200_k3_m3_n45	200	3	3	45	45	24.0	135	15843	63057	0	no	N/A	0.01
p200_k4_m3_n55	200	4	3	50	55	28.0	200	70646	281984	0	no	N/A	0.01
p200_k5_m3_n55	200	5	3	90	55	28.0	450	862730	3449210	0	no	N/A	0.01
p200_k5_m3_n08	200	5	3	717	8	4.2	3585	1368537	5460525	1.1	yes	14.6	91.11