Board-Level Multi-Terminal Net Routing for FPGA-based Logic Emulation*

Wai-Kei Mak and D. F. Wong
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712

Abstract— We consider a board-level routing problem applicable to FPGA-based logic emulation systems such as the Realizer System [3] and the Enterprise Emulation System [5] manufactured by Quickturn Systems. Optimal algorithms have been proposed for the case where all nets are two-terminal nets [10,11]. In this paper, we show how multi-terminal nets can be handled by decomposition into two-terminal nets. We show that the multi-terminal net decomposition problem can be modelled 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 is proposed. It determines if there is a feasible decomposition and gives one whenever such a decomposition exists.

1 Introduction

Logic simulation is indispensable for the verification of digital system designs. Recently severval logic emulation systems [3-7] that consist of a set of interconnected *Field-Programmable Gate Arrays* (FPGAs) [1,2] to prototype large digital logic designs have been developed. These systems can emulate complex digital system designs several orders of magnitude faster than software simulators. As a result, FPGA-based logic emulators can verify large designs that otherwise are not possible by software simulators.

For logic emulation, we first partition a large design into parts each of which can fit inside a single FPGA on the logic emulator [8,9]. And then board-level routing is performed to connect signals between the FPGA chips. We call this the *board-level routing problem* (BLRP).

In logic emulators such as the Realizer system [3] and the Enterprise Emulation system [5], the set of FPGAs for implementing the logics are interconnected by a set of FPICs where each FPIC is a small full crossbar. In this paper, we address the problem of board-level routing applicable to the logic emulation systems that use small full crossbars for interconnection.

*This work was partially supported by the Texas Advanced Research Program under Grant No. 003658459, by a DAC Design Automation Scholarship, and by a grant from AT&T Bell Laboratories.

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, using the same division on each one. The pins of each crossbar are connected to the same subset of pins from each FPGA. Thus crossbar x is connected to subset x of each FPGA's pins (Figure 1). As many crossbars are used as subsets, and each crossbar has as many pins as the number of pins in the subset times the number of FPGA chips. An inter-chip net can be connected via crossbar x if its net-pins in different FPGA chips are all assigned to I/O-pins of subset x. And the FPGAs allow the use of any of their I/O-pins for any given net. In the rest of the paper, all "net(s)" mentioned should be understood as "inter-chip net(s)" since only inter-chip nets have to be routed.

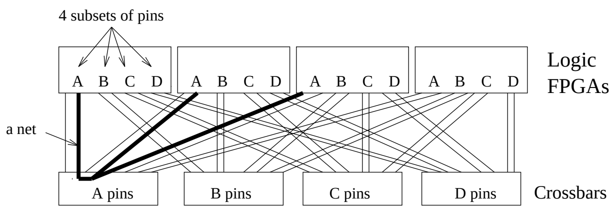

Figure 1: Interconnect Architecture.

Optimal algorithms for board-level routing when all nets are two-terminal nets and the I/O-pin subset size is even were proposed in [10,11]. The algorithms assign both net-pins of each two-terminal net to I/O-pins of the same subset type to connect each net via a crossbar, and guarantees 100% routing completion. However, we also showed in [11] that the problem of assigning net-pins to I/O-pins such that all net-pins of the same net have to be assigned to I/O-pins of the same subset type is NP-complete in the presence of multi-terminal nets. So in this paper, we propose a way to relax the constraint so that net-pins of the same multi-terminal net can be assigned to I/O-pins of different subset types but will still allow us to connect the net by connecting some net-pins to more than one I/O-pin on a FPGA.

In section 3, we present a network flow-based algorithm to decompose multi-terminal nets into sets of two-terminal nets when there are some *extra* I/O-pins so that the two-terminal net BLRP algorithm can be applied. (When the number of inter-chip nets in a FPGA chip is less than the number of I/O-pins on the chip, there are some extra I/O-pins.)

First, in section 2 we introduce a hypergraph-to-graph transformation problem closely related to our multi-terminal net decomposition problem.

2 Bounded-Degree Hypergraph-to-Graph Transformation

We are interested in the problem of transforming a hypergraph to a graph by modelling each hyperedge as a spanning tree so that the degree of each vertex v in the resultant graph does not exceed some given bound σ_v. We call this the *bounded-degree hypergraph-to-graph transformation problem.* Figure 2 shows a transformation of a hypergraph to a graph where the degrees of all vertices are bounded by 3. Each hyperedge is transformed to a spanning tree that connects all the vertices in the hyperedge. In general, the degree bound σ_v can be different for different vertex v.

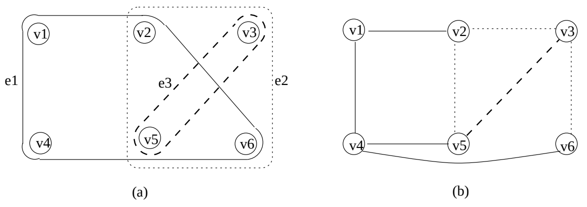

Figure 2: hypergraph-to-graph transformation. (a) A hypergraph with three hyperedges: $e1$, $e2$, and $e3$. (b) A graph formed by combining three spanning trees corresponding to the three hyperedges in (a).

To model a hyperedge of $p(\geq 2)$ vertices as a spanning tree that connects the p vertices, clearly the sum of the degrees of the vertices in the spanning tree must be $2(p-1)$ and the degree of each vertex must be at least one. On the other hand, we will prove that given any vector $d = (d_1, \ldots, d_p) \in \mathbf{N}^p$ such that $\sum_{i=1}^{p} d_i = 2(p-1)$ and $d_1, \ldots, d_p \geq 1$, we always can construct a spanning tree of p vertices whose degrees are equal to the p elements of d.

Definition 1 *A vector $d = (d_1, d_2, \ldots, d_p) \in \mathbf{N}^p$ is said to be a valid degree specification vector if there exists a spanning tree of p vertices whose degrees are equal to the p elements of d.*

Lemma 1 *A necessary and sufficient condition for $d = (d_1, d_2, \ldots, d_p) \in \mathbf{N}^p$ to be a valid degree specification vector when $p \geq 2$ is*

(i) $\sum_{i=1}^{p} d_i = 2(p-1)$, *and*
(ii) $d_i \geq 1$ *for* $i = 1, 2, \ldots, p$.

Proof: It is obvious that the condition is a necessary one. We will prove that it is sufficient by induction on the vector size p.
Base case: When $p = 2$, by conditions (i) and (ii) both d_1 and d_2 must be equal to 1. Clearly, a spanning tree with a single edge can be constructed which satisfies the degree specification vector $d = (1, 1)$.
Induction step: Assume the lemma holds for $p = r$ where $r \geq 2$. Let $d = (d_1, d_2, \ldots, d_{r+1}) \in \mathbf{N}^{r+1}$ be a vector that satisfies conditions (i) and (ii). We want to prove that d is a valid degree specification vector. Without loss of generality, we may assume $d_i \leq d_{i+1}$ for $i = 1, \ldots, r$. Since $r+1 \geq 3$, conditions (i) and (ii) imply that there exist d_j,d_k $(1 \leq j < k \leq r+1)$ such that $d_j = 1$ and $d_k \geq 2$. Define a degree specification vector $d' = (d'_1, d'_2, \ldots, d'_r) \in \mathbf{N}^r$ by

$$d'_i = \begin{cases} d_i, & \text{for } 1 \leq i \leq j-1 \\ d_{i+1}, & \text{for } j \leq i \leq r \text{ and } i \neq k-1 \\ d_k - 1, & \text{for } i = k-1 \end{cases}$$

Then $\sum_{i=1}^{r} d'_i = \sum_{i=1}^{r+1} d_i - d_j - 1 = \sum_{i=1}^{r+1} d_i - 2$ which is equal to $2(r-1)$ since d satisfies (i). As d satisfies (ii) and $d_k \geq 2$, we have $d'_i \geq 1$ for $i = 1, \ldots, r$. Hence $d' \in \mathbf{N}^r$ satisfies (i) and (ii), and is a valid degree specification vector by the assumption. So there exists a spanning tree T' of r vertices whose degrees are specified by d'. Let u be the vertex in T' whose degree is d'_{k-1} $(= d_k - 1)$. If we add a new vertex to T' and connect it to u, we will get a spanning tree of $r+1$ vertices whose degrees are exactly the $r+1$ elements of d. Hence d is a valid degree specification vector. □

We can easily turn the sufficiency proof into an efficient algorithm for constructing a spanning tree given any valid degree specification vector.

Now we describe the algorithm for the bounded-degree hypergraph-to-graph transformation problem.

Suppose we are to transform a hypergraph $H = (V, E)$ to a graph G given the degree bound σ_v of each vertex v in V. We construct a flow network $W = (\mathcal{N}, \mathcal{A})$ as follows. The node set $\mathcal{N}$ is $\{e_1, e_2, \ldots, e_{|E|}, v_1, v_2, \ldots, v_{|V|}, s, t\}$ where node e_i corresponds to hyperedge e_i in E $(i = 1, \ldots, |E|)$, node v_j corresponds to vertex v_j in V $(j = 1, \ldots, |V|)$, node s is the *source* and node t is the *sink*. For every hyperedge e_i, if it connects p vertices, then there is an arc from node s to node e_i with capacity $c(s, e_i) = p - 2$, and for every vertex v_j connected by hyperedge e_i, there is an arc from node e_i to node v_j with capacity $c(e_i, v_j) = p - 2$. For every vertex v_j in V, there is an arc from node v_j to node t with capacity $c(v_j, t) = \sigma_{v_j} - deg_H(v_j)$ where $deg_H(v_j)$ is the degree of vertex v_j in H. For example, to transform the hypergraph in Figure 2(a) to a graph where the degree of each vertex is bounded by 3, we construct the network shown in Figure 3.

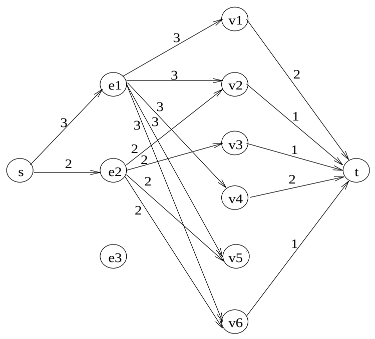

Figure 3: Flow network W.

To model each hyperedge as a spanning tree so that the total degree of each vertex v in the resultant graph is bounded by σ_v, we have to find an integral maximum flow from s to t in the constructed network [12]. It is well-known that if the capacities of all arcs in a network are integers, then there exists an integral maximum flow (i.e., the flow in each arc is an integer). And in this case, maximum flow algorithms such as the Ford-Fulkerson method [13] always produce an integral maximum flow. In the following theorem, we show how a feasible transformation can be derived from an integral maximum flow solution.

Theorem 1 *A bounded-degree hypergraph-to-graph transformation problem is feasible iff in a maximum flow of the constructed network W, the flow in arcs (s, e_1), $(s, e_2), \ldots, (s, e_{|E|})$ are all at their capacities.* [1]

Proof: Due to space limitation, we only prove the "if" part. Assume we have an integral maximum flow f such that the flow in arcs (s, e_i) are all at their capacities for $i = 1, \ldots, |E|$, we show how to find a feasible transformation of the hypergraph H.

Let $f(u, v)$ denote the flow from node u to node v. For every hyperedge e_i, if e_i connects p vertices: v_{i_1}, v_{i_2}, …, v_{i_p}, then $(f(e_i, v_{i_1}) + 1$, $f(e_i, v_{i_2}) + 1$, …, $f(e_i, v_{i_p}) + 1)$ is a valid degree specification vector by Lemma 1. Since $\sum_{k=1}^{p}(f(e_i, v_{i_k}) + 1) = f(s, e_i) + p = 2(p - 1)$ (as $f(s, e_i) = p - 2$, the capacity of arc (s, e_i) by assumption) and $f(e_i, v_{i_k}) + 1 \geq 1$ for all k. Hence a spanning tree can be constructed for hyperedge e_i where the degrees of v_{i_1}, $v_{i_2}, \ldots, v_{i_p}$ in the spanning tree are $f(e_i, v_{i_1}) + 1$, $f(e_i, v_{i_2}) + 1, \ldots, f(e_i, v_{i_p}) + 1$, respectively.

This can be done for all hyperedge e_i and the total degree of any vertex v_j in the resultant graph is $\sum_{e:(e,v_j)\in\mathcal{A}}(f(e, v_j) + 1) = f(v_j, t) + deg_H(v_j)$. But $f(v_j, t)$ is limited by the capacity of arc (v_j, t) which is $\sigma_{v_j} - deg_H(v_j)$. It follows that the degree of v_j in the resultant graph is bounded by σ_{v_j}. $\square$

3 Multi-Terminal Net Decomposition

We introduce a decomposition scheme for multi-terminal nets where a multi-terminal net is decomposed into a set of two-terminal nets based on the bounded-degree hypergraph-to-graph transformation.

We want to decompose each p-terminal net n (a hyperedge) into a set of $p - 1$ two-terminal nets (a spanning tree), which we call the *subnets* of n. We use a simple example to illustrate the idea. Consider the BLRP shown in Figure 4(a) where net 1 and net 2 are four-terminal nets, and there are extra I/O-pins on FPGA chips 2, 3, and 4 (one on each of chip 2 and chip 3, and two on chip 4). We can transform this multi-terminal net BLRP to a two-terminal net BLRP as shown in Figure 4(b). Net 1 is decomposed into three two-terminal nets, namely, subnets 1', 1", and 1"'. Similarly, net 2 is decomposed into three two-terminal nets, namely, subnets 2', 2", and 2"'. The underlying spanning trees of this decomposition of net 1 and net 2 are shown in Figure 8.

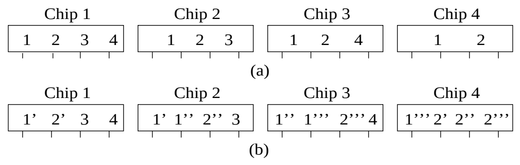

Figure 4: Decomposition of multi-terminal nets into two-terminal subnets.

To see why this decomposition is useful, we first apply an optimal algorithm for two-terminal net BLRP [10,11] to determine a feasible assignment of the subnets and the original two-terminal nets to the I/O-pins on the FPGA chips. A solution is shown in Figure 5. Note that it is not necessary to have all subnets of the same multi-terminal net assigned to the same pin

[1] By the construction of W, if one maximum flow saturates arcs $(s, e_1), \ldots, (s, e_{|E|})$, then any other maximum flow will also saturate arcs $(s, e_1), \ldots, (s, e_{|E|})$.

subset type. In Figure 5, subnets 1' and 1"' are assigned to pin subset A while subnet 1" is assigned to pin subset B. To connect net 1, we connect its net-pin in chip 1 to the I/O pin assigned to subnet 1', connect its net-pin in chip 2 to the I/O-pins assigned to subnets 1' and 1", connect its net-pin in chip 3 to the I/O-pins assigned to subnets 1" and 1"', and connect its net-pin in chip 4 to the I/O-pin assigned to subnet 1"'. In a similar way, we can connect net 2. Thus when connecting a multi-terminal net, we take advantage of the fact that a net-pin inside a chip can be connected to more than one I/O-pin on the chip.

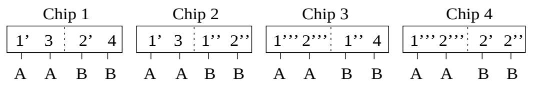

Figure 5: A feasible assignment of two-terminal nets to I/O-pins.

If we view the set of FPGA chips in a BLRP as a set of vertices and each net as a hyperedge connecting the vertices (see Figure 6), then the decomposition of multi-terminal nets into two-terminal subnets is equivalent to a bounded-degree hypergraph-to-graph transformation. Since subnets of the same multi-terminal net in the same chip (vertex) are assigned to distinct I/O-pins, the degree bound of each vertex in the resultant graph should be set to the number of I/O pins on the chip.

Multi-Terminal Net Decomposition Algorithm

1. Model the BLRP instance as a hypergraph H by representing the FPGA chips as vertices and the nets as hyperedges.
2. Transform hypergraph H to a graph where the degree of each vertex does not exceed the number of I/O-pins per chip:
   (a) Construct network W.
   (b) Find a maximum flow in W.
   (c) Get a valid degree specification vector for each hyperedge from the maximum flow solution.
   (d) Transform each hyperedge into a spanning tree according to its degree specification vector.
3. Decompose each multi-terminal net into subnets according to the corresponding hyperedge to spanning tree transformation.

For example, to decompose the multi-terminal nets in the BLRP shown in Figure 4(a). We first model it as the hypergraph in Figure 6 where hyperedge e_i represents net i and vertex v_j represents chip j. Then we construct the network in Figure 7(a). A maximum flow f of the network is found in Figure 7(b). Using flow f, the degree specification vector for hyperedge e_1

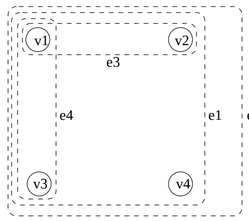

Figure 6: Viewing a BLRP instance as a hypergraph.

is $(f(e_1,v_1)+1, f(e_1,v_2)+1, f(e_1,v_3)+1, f(e_1,v_4)+1) = (1,2,2,1)$ and the degree specification vector for hyperedge e_2 is $(f(e_2,v_1)+1,\ f(e_2,v_2)+1, f(e_2,v_3)+1, f(e_2,v_4)+1) = (1,1,1,3)$. So we model hyperedge e_1 as a spanning tree T_1 where the degrees of vertices v_1, v_2, v_3, and v_4 in T_1 are 1, 2, 2, and 1, respectively. And model hyperedge e_2 as a spanning tree T_2 where the degrees of vertices v_1, v_2, v_3, and v_4 in T_2 are 1, 1, 1, and 3, respectively. See Figure 8. According to the way T_1 is connected, net 1 is decomposed into subnets 1', 1", and 1"' where subnet 1' is shared by chip 1 and chip 2, subnet 1" is shared by chip 2 and chip 3, subnet 1"' is shared by chip 3 and chip 4. And net 2 is decomposed into subnets 2', 2", and 2"' according to T_2. Finally, we get the BLRP in Figure 4(b).

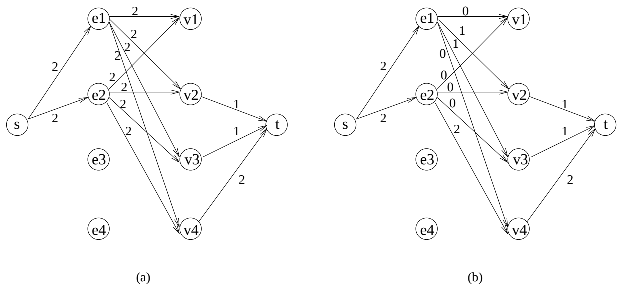

Figure 7: (a) Flow network. The number besides each arc is the capacity of the arc. (b) An integral maximum flow. The number besides each arc is the flow in the arc.

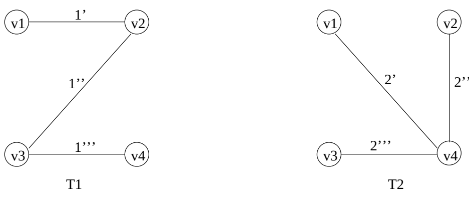

Figure 8: Spanning trees to model hyperedges e_1 and e_2.

The counterpart of Theorem 1 is given below.

Theorem 2 *Given any BLRP instance, all multi-terminal nets can be successfully decomposed iff a maximum flow of the constructed network W saturates arc (s, e_i) for all i.*

Corollary 1 *The network flow-based algorithm can successfully decompose all the nets if the number of I/O-pins on each chip is no less than* $\sum_{p \geq 2} \left\lceil \frac{2(p-1)\Delta_p}{p} \right\rceil$ *where* Δ_p *is the maximum number of* p*-terminal nets in a chip.*

The details of the proof are omitted. It involves the construction of a new network W' from W, which has a maximum flow value no larger than that of W. And proving that the capacity of any cut of the network W' is at least equal to the sum of the capacities of all outgoing arcs from s in W. Then the result follows from the max-flow min-cut theorem and Theorem 2.

For example, if no more than 15% and 5% of the inter-chip nets in each chip are three-terminal and four-terminal nets, and the rest are two-terminal nets, then the network flow-based algorithm is guaranteed to find a feasible decomposition whenever all chips are less than 93.02% full (i.e., the number of inter-chip nets in a chip is less than 93.02% of the number of I/O-pins on the chip).

We note that our network flow-based algorithm can be used for any combination of multi-terminal nets and it allows some chips to be full or almost full. Corollary 1 suggests that the algorithm should also be very useful in general.

4 Postprocessing

If in the maximum flow solution, the flow in some arc (s, e_i) is less than the arc capacity, it means there are not enough extra I/O-pins left on the chips sharing net n_i to be used for the decomposition of net i as a spanning tree of subnets. But it is possible that there are still extra I/O-pins left on other chips not used for the decomposition of any net.

For example, in the BLRP shown in Figure 9(a), both net 1 and net 2 are shared by chips 1, 2, and 3. Among chips 1, 2, and 3, there is only one extra I/O-pin which is in chip 3. So we can only use it for the decomposition of either net 1 or net 2. And we get the BLRP shown in Figure 9(b) where net 2 remains not decomposed after applying the network flow-based decomposition algorithm.

However, we may take advantage of the extra I/O-pins on chip 4. In Figure 9(c), we show a way to use the extra I/O-pins on chip 4 to decompose net 2 into a set of two-terminal nets. Then any two-terminal net BLRP algorithm can be applied to determine a feasible assignment of two-terminal nets 1', 1", 2', 2", 2"', 3, 4, and 5 to the I/O-pins on the chips. After that, to connect all the signals of net 2, we simply connect its net-pin in chip 1 to the I/O-pin assigned

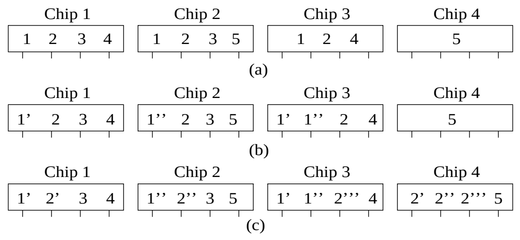

Figure 9: Decomposition of multi-terminal nets into two-terminal subnets.

to subnet 2', connect its net-pin in chip 2 to the I/O-pin assigned to subnet 2", connect its net-pin in chip 3 to the I/O-pin assigned to subnet 2"', and finally interconnect the I/O-pins assigned to subnets 2', 2", and 2"' in chip 4 internally.

This gives us the motivation to propose a postprocessing step to handle the multi-terminal nets left over after the application of the network flow-based decomposition algorithm (i.e., those net i for which the flow in arc (s, e_i) is below capacity in the maximum flow solution) using any extra I/O-pins left.

In the postprocessing step, a p-terminal net is decomposed into p or more two-terminal subnets making use of extra I/O-pins from any chips. However, a p-terminal net successfully decomposed by the network flow-based algorithm is decomposed into exactly $p-1$ two-terminal subnets only making use of extra I/O-pins from the chips sharing the net.

Let $C = \{c_1, c_2, \ldots, c_k\}$ be the set of chips that have at least three extra I/O-pins left such that the number of extra I/O-pins left on c_j is no less than that on c_{j+1} $(j = 1, \ldots, k-1)$. (Since the postprocessing step involves a maximum overhead of two extra I/O-pins per chip, we only consider those chips that have at least three extra I/O-pins left.)

For each p-terminal net, we "reserve" p extra I/O-pins from one or more chips in C in order to decompose the net into a set of p two-terminal subnets (sometimes more than p). We will use up all the extra I/O pins on chip c_j before starting to reserve those on chip c_{j+1} $(j = 1, \ldots, k-1)$ except for the following situation. If after processing a net, there is only one extra I/O-pin left on chip c_j, then we will start reserving extra I/O-pins on chip c_{j+1} for the next net.

Suppose we are to process a p-terminal net n and are to reserve extra I/O-pins for n from chip c_j. If net n has terminals on chips $c_{n_1}, c_{n_2}, \ldots, c_{n_p}$, then we will regard its terminal in chip c_{n_i} as one of the two terminals of subnet n^i $(i = 1, \ldots, p)$. See Figure 10 and Figure 11. Assume there are e $(e > 1)$ extra I/O-pins left on chip c_j. We have two cases.

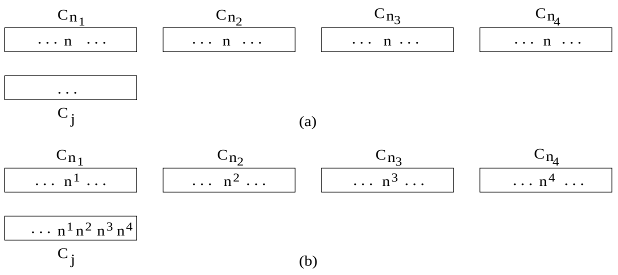

Figure 10: Net n has 4 ($= p$) terminals and chip c_j has no less than 4 ($= p$) extra I/O-pins left.

Case 1: If $e \geq p$, we simply reserve p extra I/O-pins on chip c_j and make a second terminal in chip c_j for each subnet n^i for $i = 1, \ldots, p$. See Figure 10.
Case 2: If $e < p$, we reserve all e extra I/O-pins on chip c_j and make a second terminal in chip c_j for each subnet n^i for $i = 1, \ldots, e-1$; in addition we make a terminal in chip c_j for yet another subnet n^{p+1}. We also reserve one extra I/O-pin on chip c_{j+1} and make a second terminal in chip c_{j+1} for subnet n^{p+1}. (The purpose of subnet n^{p+1} is to "connect" the part of net n modelled in chip c_j with the part of net n modelled in chip c_{j+1}. Hence there is a maximum overhead of using two more extra I/O-pins per chip.) Then we reserve $p - e + 1$ more extra I/O-pins on chip c_{j+1} and make a second terminal in chip c_{j+1} for each of the subnets n^i for $i = e, e+1, \ldots, p$. (If there are not enough extra I/O pins on chip c_{j+1}, we can repeat the same technique we just did.) See Figure 11.

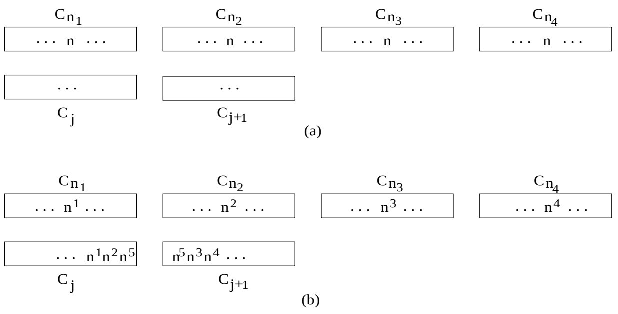

Figure 11: Net n has 4 ($= p$) terminals but chip c_j has only 3 ($< p$) extra I/O-pins left.

For example, the result of applying the decomposition algorithm with postprocessing on the BLRP in Figure 9(a) is shown in Figure 9(c).

Since a multi-terminal net is always transformed to a set of two-terminal subnets, we can connect a multi-terminal net by interconnecting its corresponding set of subnets. And this can be done by first applying the optimal two-terminal net BLRP algorithm to the transformed problem to determine a feasible assignment of the subnets to the I/O-pins on the chips to connect each individual subnet. Then for each multi-terminal net n, we can interconnect the corresponding set of subnets as follows. For each chip that contains one or more subnets of n, we interconnect all those I/O-pins on the chip which are used by the subnets of n, and if the chip contains a net-pin of n we also connect the net-pin with these I/O-pins. Hence a multi-terminal net is connected not only through connections in the crossbars but also through some internal connections of I/O-pins on the same FPGA chips.

5 Conclusions

We showed that by using our multi-terminal net decomposition algorithm to decompose all multi-terminal nets into two-terminal nets, any optimal two-terminal net BLRP algorithm can be applied to complete the routing.

References

[1] S.D. Brown, R.J. Francis, J. Rose, and Z.G. Vranesic, *Field-Programmable Gate Arrays*, Kluwer Academic Publishers, 1992.

[2] S. Trimberger (edited), *Field-Programmable Gate Array Technology*, Kluwer Academic Publishers, 1994.

[3] J. Varghese, M. Butts, and J. Baatcheller, "An Efficient Logic Emulation System", *IEEE Transactions on VLSI, Vol. 1, No. 2*, June 1993, 171-174.

[4] M. Slimane-Kadi, D. Brasen, and G. Saucier, "A Fast-FPGA Prototyping System that Uses Inexpensive High-Performance FPIC," *ACM/SIGDA International Workshop on Field-Programmable Gate Arrays*, Feb 1994.

[5] L. Maliniak, "Multiplexing Enhances Hardware Emulation," *Electronic Design*, Nov. 1992, 76-78,

[6] S. Walters, "Computer-Aided Prototyping for ASIC-based Systems," *IEEE Design and Test*, June 1991, 4-10.

[7] K. Yamada, H. Nakada, A. Tsutsui, and N. Ohta, "High-Speed Emulation of Communication Circuits on a Multiple-FPGA System," *ACM/SIGDA International Workshop on Field-Programmable Gate Arrays*, Feb 1994.

[8] N.C. Chou, L.T. Liu, C. K. Cheng, W.J. Dai, and R. Lindelof, "Circuit Partitioning for Huge Logic Emulation Systems," *ACM/IEEE Proc. 31st Design Automation Conference*, 1994.

[9] H.H. Yang, and D. F. Wong, "Circuit Clustering for Delay Minimization under Area and Pin Constraints," *The European Design and Test Conference*, 1995, 65-70.

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

[11] W.K. Mak, and D.F. Wong, "On Optimal Board-Level Routing for FPGA-based Logic Emulation," *ACM/IEEE Proc. 32nd Design Automation Conference*, 1995, 552-556.

[12] M.S. Bazaraa, J.J. John, and H.D. Sherali, *Linear Programming and Network Flows*, John Wiley & Sons, 1990.

[13] L.R. Ford, Jr., and D.R. Fulkerson, *Flows in Networks*, Princeton University Press, 1962.