

CUSTOMIZED RECONFIGURABLE INTERCONNECTION NETWORKS FOR MULTIPLE APPLICATION SOCS

*Hongbing Fan**

Wilfrid Laurier University, Waterloo ON Canada
email: hfan@wlu.ca

Jason Ernst

University of Guelph, Guelph ON Canada

Yu-Liang Wu †

The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
email: ylw@cse.cuhk.edu.hk

ABSTRACT

A Customized Reconfigurable Interconnection Network (CRIN) refers to a minimal switching network, yielding routing solutions for any element in a pre-given set of routing requirements. The CRIN design problem looks for the best performance and resource-flexibility trade-off between two extreme design contexts ASIC and FPGA. In this paper we give the modeling of this problem for both directed and undirected interconnections. A heuristic algorithm for automatic generation of CRIN is proposed along with its experimental justifications. This study was motivated from the design of reconfigurable systems-on-a-chip for multiple applications.

1. INTRODUCTION

The interconnections of input/output ports of functional modules in a traditional ASIC design usually have a good performance in time, area and power consumption, but are not subject to any changes once the design is fixed and fabricated. On the other hand, the interconnections of logic blocks and other functional components in FPGAs are totally programmable, i.e., different interconnections can be made for different applications by users at any time. The routing networks in FPGAs offers quite a high interconnection flexibility for interconnection programming, however at a high expense on chip area, power consumption and signal delays. To study a new design style with a potentially better performance and resource-flexibility trade-off between ASIC and FPGA, we consider the design of a Customized Reconfigurable Interconnection Network (CRIN), which is a minimal switching network consisting of terminals, wires and switches, and is routable for a pre-given set of routing/connection requirements.

Our study on CRINs was motivated by the design of reconfigurable systems-on-a-chip (SoC) for multiple applications. The reconfigurable SoCs we considered have a fixed or predictable set of applications. Each application consists

of a specific set of functional modules. Different applications may call for different modules and are realized through interconnection reconfigurations. The system can be reconfigured to run one application at one time, and another application at another time. The configuration can be done either statically or dynamically. Fig. 1 shows a conceptual diagram where CRIN is used in a reconfigurable SoC. The interconnection done by CRIN is at the component level, making direct interconnections of functional modules to form a large functional module or an application. Such interconnections are dedicated and good for parallel data or stream data transactions. CRIN may contain bus type interconnections, in which data links are shared by functional modules. Some advanced interconnection component products such as WishBone contain both bus type and crossbar for high-performance point-to-point interconnections.

CRINs can be used in the design of Built-In-Self-Testing (BIST), in which a CRIN is placed between the outputs of a pseudo-random pattern generator and the scan inputs of the circuit under test. Likewise, CRINs can be used for fault-tolerant designs. Redundant modules can be added and the interconnection of modules can be altered if a module becomes dysfunctional. CRINs can be further used to design reconfigurable on-chip power and clock grids, by providing different voltage power/clock supplies to functional modules. In such CRIN applications, a set of functional modules and a set of routing requirements are given, a specific CRIN needs to be designed for the given set of routing requirements.

In this study, we focus on the design methodology for CRINs and aim at developing a systematic scheme and automatic design tools for CRIN design. FPGA routing networks [1, 5] or customized FPGA routing networks can be adapted for CRIN designs. But FPGA routing networks tend to use more routing resources than necessary, and it may not be routable for some of the given routing requirements. We try to find a design scheme which can derive a customized CRIN of high-performance starting from the ground. Specifically, the objective of the scheme is able to drive a CRIN, which is guaranteed to have routing solutions

*Research partially supported by the NSERC, Canada.

†Research partially supported by RGC Earmarked Grant 2150500 and ITSP Grant 6902308, Hong Kong.

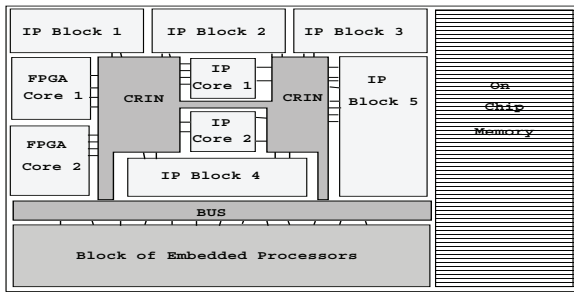


Fig. 1. CRIN in reconfigurable SoC.

for a pre-given set of routing requirements, and to have a minimum number of switches for area efficiency, a small number of stages for short signal delay, an efficient routing algorithm for reconfiguration computation and a simple structure for fabrication regularity.

This article presents the formulation of the CRIN design problem, a CRIN design scheme and an algorithm for generating CRIN topology. Note that HDL code for a CRIN design can be generated based on the interconnection topology and switch technology used. Section 2 gives the CRIN problem modeling for both directed and undirected interconnections. Section 3 gives our CRIN design scheme with experimental justifications.

2. PROBLEM MODELLING

A CRIN is used to make interconnections of ports of functional modules according to application specifications. There are three types of ports: input, output and inout. A connection (or a net) is a physical connection (a connection request) of some ports. A net is said to be directed if it has exact one output port (as source) connecting to some other input/inout ports (as sinks); otherwise it is called undirected. An undirected net can contain more than one inout ports and some other input ports; an inout port can act as a source at one time, and as a sink at another time. For example, a data connection in a bus can be viewed as an undirected net. An application consists of a group of functional modules connected together by a group of disjoint nets. Such a group of disjoint nets is referred to as a routing requirement.

A directed net can be realized by unidirectional switches such as tri-state buffers and multiplexers. A directed CRIN is a CRIN consisting of unidirectional switches for directed net routing requirements. We define switch delay of a directed net realization to be the maximum number of switches from the source to a sink. An undirected net can be realized by bidirectional switches, therefore all ports can be treated the same and switch delay is defined to be the maximum number of switches on a path from one port to another port. An undirected CRIN is a CRIN consisting of bidirectional switches. We note that undirected CRINs can be used for directed net routing requirements. However, when using undirected CRIN for directed nets we may lose delay per-

formance due to the use of bidirectional switches and delay constraints.

Given a set of ports, and a set of routing requirements over the ports, the CRIN design problem is to find a minimal CRIN (with the minimum number of switches) satisfying the routing specifications. That is, for each of the given routing requirements, there exists a valid configuration, also called a realization, of the routing requirement, which is an ON/OFF assignment to all switches, such that only the ports in the same net of the routing requirement are connected by wires and ON switches.

2.1 Directed CRIN

Let $P = \{p_1, \dots, p_m\}$ and $Q = \{q_1, \dots, q_n\}$ be the sets of output and input ports, respectively. We represent a directed net connecting output port $p_i \in P$ to a subset of input ports $I(p_i) \subset Q$ as $(p_i, I(p_i))$. If p_i is not required to be connected to any input port, then $I(p_i) = \emptyset$. Thus a routing requirement can be represented as $R = \{(p_i, I(p_i)) : i = 1, \dots, m\}$, where $I(p_i) \cap I(p_j) = \emptyset$ if $p_i \neq p_j$, and a set of routing requirements as $\mathcal{R} = \{R_1, \dots, R_t\}$, $R_j = \{(p_i, I_j(p_i)) : i = 1, \dots, m, j = 1, \dots, t\}$. A directed CRIN G routable for \mathcal{R} is a programmable system with input ports: $P = \{p_1, \dots, p_m\}$ and configuration bits $C = \{x_1, \dots, x_h\}$, and output ports: $Q = \{q_1, \dots, q_n\}$, satisfying that, for each routing requirement $R_j \in \mathcal{R}$, there is an assignment A_j for the configuration bits C such that every net $(p_i, I_j(p_i)) \in R_j$ is realized, i.e., the signal to p_i will show up in the ports of $I_j(p_i)$ satisfying the delay specification.

Since the number of different routing requirements is t , the minimum number of configuration bits is at least $\lceil \log t \rceil$. It is known that a CRIN with $\lceil \log t \rceil$ configuration bits can be designed by the traditional combinatorial logic approach, i.e., expressing each q_i as the sum of product of the related code term of routing requirements. However, a CRIN designed in this way will have a switch delay of $O(\log t)$ if bounded fan-in switches are used. For example, if we want to design a directed CRIN capable of routing all possible routing requirements from m output ports to n input ports, the number of routing requirements is equal to $\sum_{k=0}^n S(n, k) m(m-1) \dots (m-k+1) = m^n$, where $S(n, k)$ is the Stirling number of the second kind, counting the number of ways to partition a set of n elements into k nonempty subsets. Thus the minimum number of configuration bits is $\lceil m \log n \rceil$, and switch delay is $O(m \log n)$.

Switch delay can be reduced by introducing more configuration bits and switches as a tradeoff with area. We thus model a directed CRIN as a directed bipartite graph G with vertex set pair (P, Q) , and there is a switch (edge) from $p_i \in P$ to $q_k \in Q$ if and only if $q_k \in I_j(p_i)$ and $(p_i, I_j(p_i)) \in R_j$ for some $R_j \in \mathcal{R}$. With this modelling, a directed CRIN can be implemented as a partial crossbar [5], and the switch delay with this implementation is one for

every net. There is also a linear time algorithm to compute a configuration for a routing requirement. However, the number of switches is equal to $\sum_{i=1}^m d_i$, which is usually more than necessary, where d_i denotes the total number of output ports that p_i is connected to in all routing requirements. When $m = n$ and G is a complete bipartite graph and switch delay specification is flexible (bigger than one), the number of switches can be reduced by employing the existing interconnection network topologies such as crossbar networks, Clos networks and Beneš networks [5, 3].

2.2 Undirected CRIN

For undirected CRINs design, since no source/sink ports are specified for nets, we view all ports the same and use the term terminal instead of input/output ports. Then a net consists of a subset of terminals and a routing requirement consists of a group of mutually disjoint nets. An undirected CRIN consists of terminals, wires, bidirectional switches and control circuits. Let $V = \{v_1, \dots, v_m\}$ denote the set of terminals. A t -pin net connecting requirement t terminals v_{i_1}, \dots, v_{i_t} is represented as a set $N = \{v_{i_1}, \dots, v_{i_t}\}$. A routing requirement is a set of disjoint subsets $\{N_1, \dots, N_s\}$, where each $N_i \subset V, N_i \cap N_j = \emptyset, 1 \leq i < j \leq s$. We model the interconnection topology of an undirected CRIN as a simple graph $G = (V \cup V_I, S)$, where V_I is the set of internal nodes corresponding to internal wires, and S is the set of edges representing switches. With this modeling, G is routable for routing requirement $\{N_1, \dots, N_s\}$ if there is an ON/OFF assignment to all edges of G , such that two nodes $u, v \in V$ are in the same connected component of $(V \cup V_I, S_{ON})$ if and only if u and v are in the same net N_i , where S_{ON} denotes the set of ON-switches (edges). Given a set of routing requirements on $V, \mathcal{R} = \{R_i : i = 1, \dots, h\}$, where $R_i = \{N_{i,j} : j = 1, \dots, s_i\}$ is a routing requirement on $V, i = 1, \dots, h$. Then G is a valid CRIN for \mathcal{R} if G is routable for every routing requirement R_j in \mathcal{R} . The undirected CRIN design problem is to find a minimal CRIN, i.e., a valid CRIN with the minimum number of switches.

3. CRIN DESIGN SCHEME

For convenience, we use term CRIN instead of undirected CRIN, and denote by (V, \mathcal{R}) a given CRIN design instance. We present a three-phase CRIN design scheme. Phase one is reduction, which simplifies the design instance to its kernel. Phase two is to find a minimal interconnection graph design, which is a valid CRIN without internal nodes. Phase three improves the CRIN design using clustering technique and results of switch block designs developed in FPGA architecture study.

3.1 Reduction

The reduction is to remove the fixed part of interconnections from the pre-given set of the routing requirements. The fixed part of an interconnection is the interconnection

where no switch is needed. A reduction reduces one instance (V, \mathcal{R}) to another instance (V', \mathcal{R}') such that an equivalent CRIN for (V, \mathcal{R}) can be constructed from a valid CRIN for (V', \mathcal{R}') without adding extra switches. The following simple rules satisfy the above reduction requirement.

Rule 1. If a terminal $v \in V$ is used by at most one net, remove v from V .

Rule 2. If a net $N_{i,j} \in R_i$ contains only one terminal, remove $N_{i,j}$ from routing requirement R_i .

Rule 3. If two nodes u, v are always in the same net, remove one of them from V .

Repeatedly apply the above reduction rules until no further reduction can be done. The resulting (V', \mathcal{R}') is called the *kernel* of the original instance. It can be proven that the kernel is uniquely determined and every node is in at least two non-inclusive nets, and there must be a switch connecting it. We then only need to design a CRIN for a kernel instance.

3.2 Interconnection graph

Given a kernel instance of CRIN design (V, \mathcal{R}) . Let R denote the set of nets in all routing requirements of \mathcal{R} , then $H = (V, R)$ forms a hypergraph. A graph $G = (V, E)$ is said to be an *interconnection graph* of H if $G[N]$ is connected for every $N \in R$, where $G[N]$ denotes the induced subgraph of G by the vertices in N .

The interconnection graph of H is a valid CRIN for (V, \mathcal{R}) since, for any $R_i \in \mathcal{R}$, set ON to switches in a spanning tree of $G[N_{i,j}]$ for every $N_{i,j}$, we obtain a routing for R_i . Such a routing can be computed in time linear. The interconnection graph of H is also a valid CRIN without containing internal nodes, so it is optimal in terms of the number of wires. The existence of an interconnection graph is obvious since the click graph of $H, C_H = (V, \{uv : u, v \in N, N \in R\})$, is an interconnection of graph H . We want to find a minimal interconnection graph, i.e., an interconnection graph of H with the minimum number of switches. We refer to this problem as interconnection graph problem.

It is proven that the interconnection graph problem is NP-hard. We propose the so-called reduction based greedy algorithm as shown in Fig. 2. The algorithm starts with an empty graph on V and constructs an interconnection graph by adding edges and removing hyperedges as long as they induce a connected subgraph in the currently constructed graph, until there is no hyperedge left.

We tested three classes of hypergraphs. The first class is hypergraph $H_1(m)$ consisting of vertices $1, \dots, 2m$ with hyperedges $\{i, i+1\}, \{i+m, i+m+1\}, \{i, i+1, i+m\}, \{i+m, i+m+1, i+1\}, i = 1, \dots, m-1, \{1, m\}, \{2m, m+1\}, \{m, 1, 2m\}, \{2m, m+1, 1\}$. The second class is hypergraph $H_2(m)$ consisting of vertices $1, \dots, 2m+2$, and hyperedges $\{0, 1, i, i+m\}, i = 2, \dots, m+1, \{i, i+$

Input: hypergraph $H = (V, R)$
 $R' = R, E' = \emptyset, \bar{E}' = \{uv : u, v \in N, N \in R\}$.
 $H' = (V, R'), G' = (V, E')$
reduction = true
While $R' \neq \emptyset$
 If reduction
 $Reduction(R', E', \bar{E}')$
 reduction = false
 Else
 choose $e^* \in \bar{E}'$ with $d_{H', G'}(e^*) = \max\{d_{H', G'}(e) : e \in \bar{E}'\}$
 $E' = E' \cup \{e^*\}, \bar{E}' = \bar{E}' \setminus \{e^*\}$
 reduction = true
 End If
End While
Output: $G = (V, E')$

$Reduction(R', E', \bar{E}')\{$
 $H' = (V, R'), G' = (V, E')$
repeat = true
r1 = r2 = r3 = r4 = true;
While repeat
 If there is an $N \in R'$ such that $G'[N]$ is connected
 remove N from R'
 Else r1 = false
 If $u, v \in V$ satisfy that $\{u, v\} \cap N \neq \emptyset$ indicates $\{u, v\} \subseteq N$
 and exist $N \in R'$ such that u, v are components of $G'[N]$
 add uv to E'
 remove v from hyperedges that contain v
 remove edges from \bar{E}' that contain v
 r1 = true
 Else r2 = false
 If there is an $N = \{u, v\} \in R'$
 add uv to E' and remove uv from \bar{E}' .
 remove N from R'
 r1 = true, r2 = true
 Else r3 = false
 If exist $N = \{u, v, w\} \in R'$ with $uv \in \bar{E}$ and $d_{H', G'}(uw) = 1$
 add $\{uw, vw\}$ to E' and remove $\{uv, uw, vw\}$ from \bar{E}' .
 remove N from R' .
 r1 = true, r2 = true, r3 = true
 Else r4 = false
 repeat = r1 OR r2 OR r3 OR r4
End While
Return (R', E', \bar{E}') . $\}$

$ V(H) $	$ E(H) $	$ E(CH) $	$ E(G) $	Improvement %	Time (s)
Results of test case one					
200	400	400	300	25	0.41
400	800	800	600	25	2.36
600	1200	1200	900	25	8.27
800	1600	1600	1200	25	20.6
1000	2000	2000	1500	25	42.28
Results of test case two					
202	200	901	301	66.6	0.34
402	400	1801	601	66.6	1.06
602	600	2701	901	66.6	3.34
802	800	3601	1201	66.6	7.78
1002	1000	4501	1501	66.6	14.79
Results of random hypergraphs with 5n hyperedges					
20	100	190	78	59	0.16
40	200	774	238	69	4.17
60	300	1689	456	73	23.31
80	400	2866	703	75	81.99
100	500	4155	989	76	244.39
120	600	5613	1300	77	541.75

Fig. 2. Reduction based greedy algorithm and test results.

$1, i+m, i+m+1\}, i = 2, \dots, m, \{m+1, 2, 2m+1, m+2\}$. The third class is random hypergraph H consisting of vertices $1, \dots, n$ and a set of random hyperedges. The program was able to find the minimal interconnection graphs for both $H_1(m)$ and $H_2(m)$. For the random hypergraphs, the program found an interconnection graph with average improvement of 70% over the corresponding clique graphs. The testing results are shown in the table of Figure 2.

3.3 Improvement by clustering and switch box

Our method has two steps. Step one employs a clustering technique to find dense clusters ([6]) in the interconnection graph. Step two substitutes the dense clusters with switch modules. A switch module consists of a switch block at the center surrounded by several crossbars. Depending on the nets over the cluster, we choose USB ([2]) or HUSB ([4]) as the center switch block. For example, when a cluster has n nodes and $O(n^2)$ switches, then it can be substituted by a polygonal switching network ([7]) with $O(n^{3/2})$ switches. The improvement phase results in a structured CRIN containing switch boxes and connection boxes plus some additional switches. In addition, there is an efficient routing algorithm with the improved CRIN.

4. CONCLUSIONS

We proposed the new problem of designing customized reconfigurable interconnection networks for reconfigurable SoCs. A three-phase CRIN design scheme was presented, which includes reduction, interconnection graph design and cluster switch box based improvement. Existing techniques were employed in phase one and three. For phase two, a reduction based greedy algorithm for generating interconnection graph was proposed and implemented. Initial experimental results showed that the resource usage of CRIN can be greatly reduced through the optimization process.

5. REFERENCES

- [1] S. Brown, R. Francis, J. Rose, and Z. Vranesic. *Field Programmable Gate Arrays*. Kluwer-Academic Publisher, 1992.
- [2] Y. W. Chang, D. F. Wong, and C. K. Wong. "Universal Switch Modules for FPGA Design". *ACM Transactions on Design Automation of Electronic Systems.*, 1(1):80–101, Jan. 1996.
- [3] W. J. Dally. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2004.
- [4] H. Fan, J. Liu, and Y. L. Wu. "General Models and a Reduction Design Technique for FPGA Switch Box Designs". *IEEE Transactions on Computers*, 52(1):21–30, Jan. 2003.
- [5] G. Lemieux and D. Lewis. *Design of Interconnection Networks for Programmable Logic*. Kluwer-Academic Publisher, 2003.
- [6] M. E. J. Newman. Modularity and community structure in networks. *PROC.NATL.ACAD.SCI.USA*, 103:8577, 2006.
- [7] M. Yen, S. Chen, and S. Lan. "A Three-Stage One-Sided Rearrangeable Polygonal Switching Network". *IEEE Trans. on Computers*, 50(11):1291–1294, Nov. 2001.