Routing Clos-based Interconnection Networks for Post-Silicon Debug

Fredy A. M. Alves
Instituto de Ciências Exatas e Tecnológicas,Campus UFV-Florestal
Florestal, Brasil
fredy.alves@ufv.br

Fernando A. D. Teixeira
Instituto de Ciências Exatas e Tecnológicas,Campus UFV-Florestal
Florestal, Brasil
fernando.teixeira@ufv.br

André B. M. Gomes
Departamento de Informática, UFV
Instituto de Ciências Exatas e Tecnológicas,Campus UFV-Florestal
Florestal, Brasil
andre.maciel@ufv.br

Ricardo S. Ferreira
Departamento de Informática, UFV
Viçosa, Brasil
ricardo@ufv.br

José Augusto M. Nacif
Departamento de Informática, UFV
Instituto de Ciências Exatas e Tecnológicas,Campus UFV-Florestal
Florestal, Brasil
jnacif@ufv.br

ABSTRACT

In complex integrated circuit designs, pre-silicon verification can not ensure an error-free environment. A post-silicon verification is used to monitor the internal signal behavior after fabrication, allowing to capture some errors at full clock speed. With the trace buffer technique, it is possible to observe signals overtime, storing trace data in a buffer. The size of the buffer, however, is limited due to its cost, which also limits the amount of traced signals. An interconnection network is used to tap a larger set of signals, selecting a subset to be stored in the buffer. In this paper, we present a 3-stage Clos routing algorithm that connects a set of inputs to the outputs of the interconnection network. Our algorithm uses a graph edge coloring strategy and routes a 10,000x256 Clos network in less than 3 milliseconds.

Keywords

Verification, Trace-Based Debug, Interconnection Network

1. INTRODUCTION

Verification is a very challenging and the most time consuming phase in the integrated circuit development cycle. In complex integrated circuit designs, pre-silicon verification can not ensure an error-free environment. The techniques used on this strategy are based on simulation and formal verification. Simulation allows a wide circuit observability, but it is many times slower than running the circuit at real clock frequency. Post-silicon verification is a set of tools and techniques used to monitor and debug manufactured circuits internal behavior. This process captures errors after millions of clock cycles. The post-silicon debug consumes more than 35% of the chip development cycle [1].

Unlike pre-silicon verification, post-silicon debug observability is limited. Using the trace buffer method, a set of signals is captured and stored during execution [9]. The trace buffer size is defined by area cost constraints, limiting the number of traced signals [10]. An interconnection network allows the designer to choose a signal subset to store in the buffer. The Clos network can be used to select those signals. For a large number of signals, the Clos network introduces less area overhead than a crossbar network.

The Clos network is a multistage network proposed in [3]. The network is characterized by a triple (m, n, r) where m is the number of middle stage switches, n is the number of ports in the input and output switches, and r is the number of input and output switches. This interconnection network class can be designed with two types of topologies: strictly non-blocking and rearrangeable non-blocking. The first allows routing without collision treatment and the second performs a rearrangement when there is a path collision.

In this work, we present a 3-stage Clos routing algorithm that connects a set of inputs to the outputs of the interconnection network. State-of-the-art routing algorithms can be divided into two different categories: blocking avoidance routing, and routing irrespective of blocking [6]. The first tries to route a network avoiding to block paths in each connection performed. Routing irrespective of blocking routes requested connections treating path collisions. This algorithm rearranges some connections in a way that all connections are routed. Both categories, for multistage interconnection networks, can use 2 different types of strategies: Graph Coloring or Matrix Decomposition [4]. Graph Coloring algorithm usually presents better performance than matrix decomposition [6].

This work is outlined as follows. Sections 2 and 3 present a comprehensive explanation about the post-silicon trace buffer strategy, asymmetric 3-stages Clos interconnection networks and related work. Section 4 describes a routing algorithm using a graph edge coloring strategy. Section 5 discusses our results and, finally, we conclude our remarks and present future work in Section 6.

2. BACKGROUND

This Section presents background concepts about post-silicon debug trace buffer and Clos networks.

2.1 Trace Buffer

Trace buffer is a post-silicon debug technique that allows the designer to monitor internal signals over time at full clock speed. A buffer is used to store the traced signals. The buffer size is defined by two parameters: depth and width. The number of traced signals determines the width and how many times a signal value is stored determines the depth. The trigger logic monitors the circuit under debug behavior and starts tracing the signals when pre-defined conditions are met. The trace buffer strategy is illustrated in Figure 1. The interconnection network is located between the tapped signals and the buffer. It allows the designer to choose a large signal number to tap, and then a small number of signals is selected to be stored in the buffer.

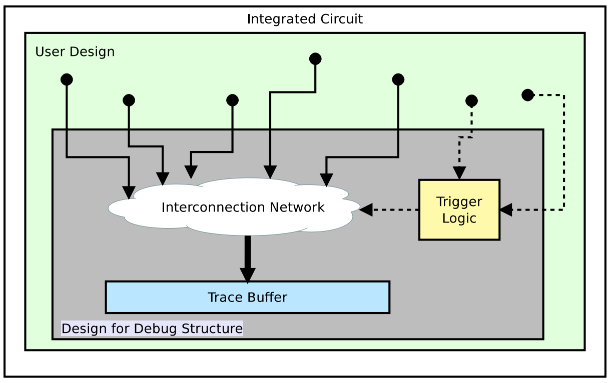

Figure 1: Trace buffer block diagram.

2.2 Clos Networks

The Clos interconnection network was designed as a multistage non-blocking network, so any input can be routed to an output. This topology was presented in [3] as a replacement to the crossbar in the context of telephone systems. A 3-stage Clos network is built by input, middle, and output stages. The Clos network is characterized by a triple $C(m, n, k)$, where m is the number of middle stage switches; n is the number of ports in input and output switches; k is the number of input and output switches. The 3-stage Clos topology is shown on Figure 2.

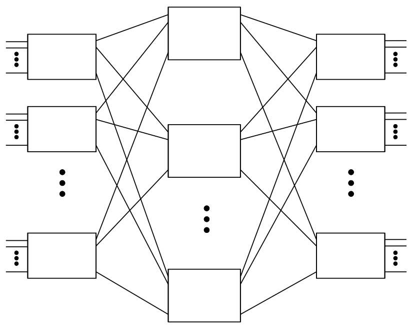

Figure 2: 3-stage Clos network.

Let N be the number of network inputs and outputs. If we assume that $n = N^{1/2}$, then $m = 2n - 1$. When $N \equiv r(mod\, n)$, an extra switch is used to drive the remainder r from the input stage to the output stage. So, under this condition, $N = k * n + r$. Considering an asymmetric 3-stage Clos network with N_1 inputs and N_2 outputs, the minimum number of switches in the intermediate stage is $m = (n_1 - 1) + (n_2 - 1) + 1$.

3. RELATED WORK

Minimum Distribution Algorithm is a heuristic proposed in [5]. In [8], an asynchronous heuristic algorithm was proposed. In [7] a study about multicast routing in Clos networks is presented. A study about randomized Clos network routing is realized in [2]. The results are given in terms of fault tolerance and hardware delay. In this work, we propose to present the results in terms of execution time in order to evaluate the feasibility to use our algorithm to route asymmetric Clos networks to be used in post-silicon debug.

4. BIPARTITE GRAPH ROUTING

The algorithm used to route the asymmetric rearrangeable Clos network is based on an unicast routing strategy where one input is routed to only one output. In order to illustrate this concept, we represent the network as a bipartite graph. In Figure 3, the left set of vertices is the input switches $(I1, I2, I3, In)$. The right set is the output switches $(O1, O2, O3, On)$. Each requisition to connect an input switch Ia to an output switch Ob through a middle switch Mc is represented by an edge from Ia to Ob. We treat this process as an edge coloring problem, where a color is assigned to each middle switch. The problem consists in assigning a color to each edge in a way that no vertex is incident to two edges of the same color [4].

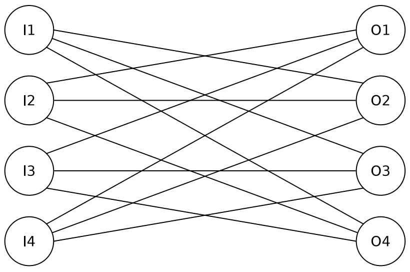

Figure 3: Clos R3 12x12 graph representation.

In algorithm 1 we show the pseudocode of our implementation. To route a connection from input Ia to output Ob, first we search for a middle switch that is free for both Ia and Ob. If such switch exists, the connection (Ia, Ob) is established and the routing process for that requisition stops. If there is no such switch available, the process of rearranging connections starts. We look for a free switch on Ia and on Ob, storing their positions on the variables $FreeMidIa$ and $FreeMidOb$, respectively. The connection (Ic, Ob) is disconnected on $FreeMidIa$ in order to enable the connection (Ia, Ob) which is then realized. The next step is to try to connect (Ic, Ob) on $FreeMidOb$. If such connection is free, it is established and the routing process stops. Otherwise, the connection (Ic, Od) is disconnected from $FreeMidOb$, (Ic, Ob) is established and the algorithm runs another iteration with Ic as the new Ia and Od as the new Ob.

To illustrate our algorithm, consider the $(3, 3, 4)$ Clos network in Figure 4. In order to connect the input port $IP0$ to the output port $OP0$, the algorithm looks for a free middle switch for both $I0$ and $O0$. The $M0$ switch is found, establishing the connection. The second connection is from $IP4$ to $OP1$. The algorithm finds that $M1$ switch is free on $I1$ and $O0$, and establishes the connection. The connections $(IP8, OP2)$, $(IP1, OP4)$, and $(IP5, OP5)$ are performed through middle switches $M2$, $M1$, and $M0$, respectively, without any conflict. When the connection $(IP6, OP3)$ is requested, the algorithm can not find a free middle

Algorithm 1 Routing algorithm

```
 1: procedure CONNECTIONREQUEST
 2:     FreeMidIaOb ← Search Mid(freemida & freemidb)
 3:     if FreeMidIaOb $>$ 0 then
 4:         Connect(Ia, Ob, FreeMidIaOb)
 5:         Stop_Routing
 6:     end if
 7:     FreeMidIa ← Search_Mid(FreeIa)
 8:     FreeMidOb ← Search_Mid(FreeOb)
 9:     Disconnect_IcOb_On_Mid(Ob, FreeMidIa)
10:     Connect_IaOb_On_Mid(Ia, Ob, FreeMidIa)
11:     Busy_IcOd ← Disconnect_On_Mid(Ic,FreeMidOb)
12:     if Busy_IcOd $=$ 0 then
13:         Connect_IcOb_On_Mid(Ic, Ob, FreeMidOb)
14:         Stop_Routing
15:     end if
16:     Ia ← Ic
17:     Ob ← Od
18: end procedure
```

switch, starting the rearrangement. The switch $M0$ is free on $I2$ and $FreeMidIa = M0$. $M2$ is free on $O1$, so $FreeMidOb = M2$. The connection $(IP5, OP5)$ is disconnected to free $(I1, O1)$ on $M0$, the connection $(IP6, OP3)$ is realized and $(IP5, OP5)$ finds a free path through $FreeMidOb$, which is accomplished connecting $(I1, O1)$.

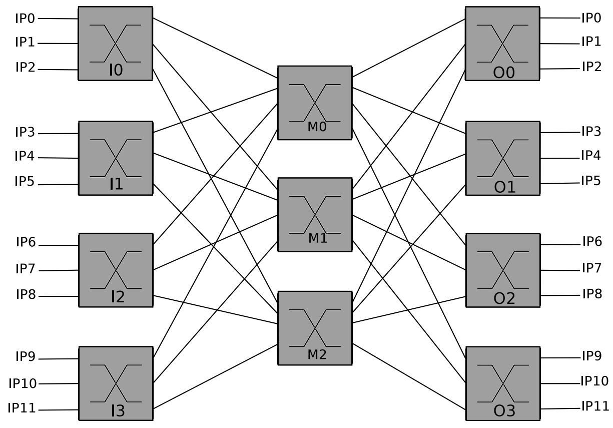

Figure 4: Clos R3 12x12 topology.

5. RESULTS

Our experiments have been performed using asymmetric rearrangeable 3-stage Clos networks. We have used the fixed numbers of inputs 100, 1,000, and 10,000. We have varied the number of outputs to 32, 64, 128, and 256, resulting in 10 different Clos configurations. These configurations have been routed 100 times each with a random set of inputs. The average of the running times is presented is Table 1 and Figure 5.

Execution times grow with the number of input ports because the algorithm takes longer analyzing the number of switches. The time for a 10,000x64 configuration is more than twice compared to the 1,000x64 which is composed by 10 times more input signals. This does not happen between 100x64 and 1,000x64. This difference can be explained because as 64 represents 64% of the inputs number on the 100x64 configuration, the probability that path collisions will occur during a routing process is higher than on a 1,000x64

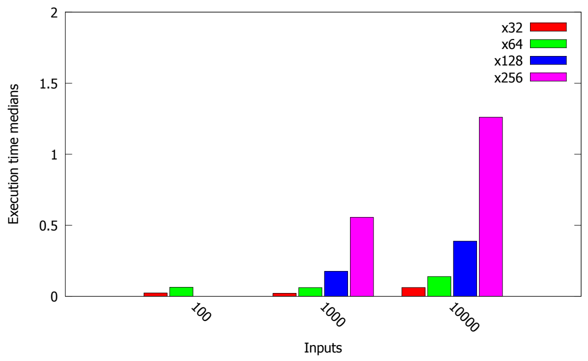

Figure 5: Configurations x average execution time.

Table 1: Algorithm average execution times.

Configuration	Time(milliseconds)
100x32	**0.023339**
100x64	**0.063789**
1,000x32	**0.020697**
1,000x64	**0.061192**
1,000x128	**0.175781**
1,000x256	**0.555299**
10,000x32	**0.061820**
10,000x64	**0.139006**
10,000x128	**0.388367**
10,000x256	**1.260512**

configuration where 64 only represent 6.40% of the input ports, therefore, the rearrangement process occur more on a 100x64 than on a 1000x64 configuration, taking almost the same routing time for both.

6. CONCLUSIONS AND FUTURE WORK

The post-silicon debug is one of the most important phases in an integrated circuit verification. When using a trace-buffer strategy, it is crucial to have an interconnection network in order to provide a better observability of the circuit under debug. We have implemented a robust routing algorithm for the asymmetric rearrangeable Clos networks using a Graph Coloring based heuristic.

Our results show that, for an asymmetric Clos network designed for a trace-buffer post silicon debug, our algorithm is able to route all sets of inputs with the same amount as the outputs in less than 2 milliseconds, with almost no blocking.

For future work we intent to integrate the routing tool with our post-silicon infrastructure. After that we will be able to dynamically change the Clos network configuration on our prototype FPGAs.

Acknowledgments

We would like to thank CNPq, CAPES, UFV, FAPEMIG, and FUNARBE for the financial support.

7. REFERENCES

[1] M. Abramovici, P. Bradley, K. Dwarakanath, P. Levin, G. Memmi, and D. Miller. A reconfigurable design-for-debug infrastructure for socs. In *Design Automation Conference, 2006 43rd ACM/IEEE*, pages 7–12, 2006.

[2] M. Bhatia and A. Youssef. Performance analysis and fault tolerance of randomized routing on clos networks. In *Frontiers of Massively Parallel Computing, 1996. Proceedings Frontiers '96., Sixth Symposium on the*, pages 272–281, Oct 1996.

[3] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32(2):406–424, 1953.

[4] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[5] X. Duan and S. Liu. A heuristic routing algorithm for clos network. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 5892–5895, June 2008.

[6] Z. S. Ghandriz, K. Zeinali, and P. Esmaeil. A new routing algorithm for a three-stage clos interconnection networks. *International Journal of Computer Science Issues (IJCSI)*, 8(5), 2011.

[7] J.-M. Ho, D.-R. Liang, and K.-H. Tsai. On multicast routing in clos networks. In *Parallel Architectures, Algorithms, and Networks, 1996. Proceedings., Second International Symposium on*, pages 394–400, Jun 1996.

[8] W. Song, D. Edwards, Z. Liu, and S. Dasgupta. Routing of asynchronous clos networks. *Computers Digital Techniques, IET*, 5(6):452–467, November 2011.

[9] J.-S. Yang and N. Touba. Improved trace buffer observation via selective data capture using 2-d compaction for post-silicon debug. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(2):320–328, Feb 2013.

[10] J.-S. Yang and N. A. Touba. Expanding trace buffer observation window for in-system silicon debug through selective capture. In *Proceedings of the 26th IEEE VLSI Test Symposium*, VTS '08, pages 345–351, Washington, DC, USA, 2008. IEEE Computer Society.