

Concentrator Access Networks for Programmable Logic Cores on SoCs

Bradley R. Quinton
Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada
bradq@ece.ubc.ca

Steven J. E. Wilton
Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada
steve@ece.ubc.ca

Abstract - The inclusion of programmable logic cores in modern SoCs motivates the need for an access network to make full use of this resource. The programmable nature of these cores removes the requirement of input/output ordering on this access network. Theoretical work on a class of unordered networks called concentrators has shown that as these networks become large, they have a lower cost than ordered or permutation networks. However, currently known constructions of concentrator networks are not lower cost than permutation networks for the entire range of networks of the size required for SoCs. This paper demonstrates the differences in the cost and depth of concentrator and permutation networks. It will also present a new construction of a concentrator network that has lower cost and depth than a permutation network for all configurations.

I. INTRODUCTION

Advances in integrated circuit (IC) technology have made possible the integration of a large number of functional blocks on a single chip. This process is commonly referred to as system-on-a-chip (SoC) design. As the complexity of SoCs has increased, there has been a desire to increase the flexibility of these designs to allow a single SoC to perform multiple functions. One interesting trend in SoC design is the inclusion of programmable logic cores [1,2]. These programmable logic cores are a collection of uncommitted logic and routing resources that can be configured *after fabrication*. In order to maximize flexibility, it is desirable to also provide a *configurable network* that connects fixed cores in the SoC to the programmable logic core(s). Different configurations of the programmable logic core(s) may require different input and output signals; the network provides a means to programmably select some number of signals from across the chip, and connect these signals to the programmable logic core(s).

The idea of using a network for SoC communication is not new [3]. These networks have mainly targeted communication between a CPU and a number of peripherals, using packet-based networks. These proposed networks are not well suited for the task of connecting to a programmable logic core. Embedded programmable cores provide a synchronous, high-bandwidth, fixed-latency interface. Connecting a packet-based network to this interface is difficult and undesirable, as it would limit the potential applications of the programmable logic. Indirect, non-

blocking networks, such as those defined for traditional telephone networks are better suited to the interface and applications of embedded programmable logic. These networks provide a direct, fixed-bandwidth, fixed-latency connection that can be configured on a per pin basis for each pin on the programmable core. A great deal of research has been done on non-blocking multistage networks that allow arbitrary connections between any input and any output of the network [4,5]. These networks can be viewed as permutation networks, since they provide all possible permutations of the inputs on the output side of the network. These permutation networks can be used to programmably connect signals from across the chip to the programmable logic core(s).

These permutation networks may provide *more* flexibility than is required. Before configuration, all pins of a programmable logic core are equivalent. The circuit implemented in the core can be adapted to any arbitrary signal-to-pin assignment. Thus, it is not necessary to be able to connect every network input to every network output. Instead, it is only necessary to be able to connect every set of network inputs to the set of network outputs. Exactly which pin is assigned to each signal does not matter. In order to lower the cost and improve the performance of the access network we would like to take advantage of this flexibility.

Research has been done on the problem of unordered networks [6,7]. A specific class of unordered network called *concentrators* has been defined which allow a number of inputs to connect to a smaller number of outputs. As defined concentrator networks are well suited to the problem of providing access to a programmable core. However, while some previously described concentrator constructions are lower depth than permutation networks in all cases, they are not lower cost (area) than permutation networks for all possible configurations. This makes the choice of network architectures difficult because the network size (the number of inputs and outputs), and relative importance of cost and depth must be considered before choosing an architecture. This paper will demonstrate the difference in cost and depth of permutation networks and concentrator networks for a variety of network sizes. Following this, a new concentrator network construction that is lower cost and lower depth than a permutation network for all network sizes will be presented.

II. NETWORK REQUIREMENTS

The access network required for programmable logic in a SoC can be viewed as two separate networks. One network will connect a number of potential sources to a smaller number of programmable logic inputs. The other will connect a number of programmable logic outputs to a larger number of potential sinks. Essentially these networks are mirrors of each other. Therefore a network solution that is appropriate for the input-side network will satisfy the output-side requirements when configured in reverse. Therefore in this paper we will only consider the network connecting a number of sources to a smaller number of inputs of the programmable logic.

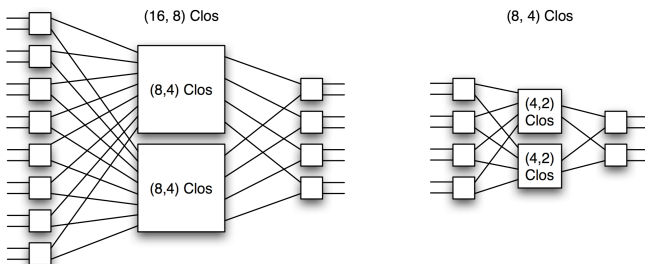
The network required for access to a programmable core does not have to be configured in real time. It would instead be programmed once for each application. For this reason the network does not need to handle incremental connection requests. Therefore the network is only required to be rearrangeably non-blocking [5].

III. PERMUTATION AND CONCENTRATION NETWORKS

A. Permutation Networks

A permutation network is a network with n inputs and m outputs for which any number of inputs $q \leq m$ can each connect to any one specified output. Since permutation networks allow the arbitrary connection of any input to any output, it is possible to use a permutation network to access a programmable core. The Clos network has been shown to be an effective means to construct a permutation network that is low cost and rearrangeably non-blocking [5]. The Benes network can be formed by recursively replacing the middle stage of a Clos network with another Clos network to reduce the network cost [4]. Most simple descriptions of the Benes networks assume that the number of inputs is equal to the number of outputs, and that the number of inputs is a power of 2. However, when viewed as a recursive construction of Clos networks, it is possible to build a rearrangeably non-blocking network with an arbitrary number of inputs and outputs (Figure 1).

Figure 1 Asymmetric Benes



This construction requires that the number of inputs and outputs be divisible by two. When this is not the case, the size of the inputs and/or outputs can be increased by one and the extra input and/or output can be left unconnected. During the construction, once the number of outputs required for the Clos network is less than or equal to two, a binary tree structures can be used in place of a Clos network.

B. Concentration Networks

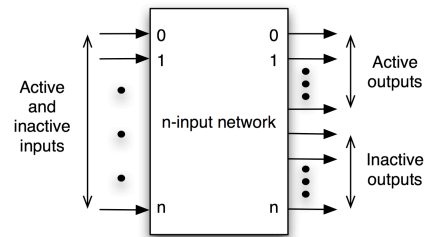
A (n, m) -concentrator is a network with n inputs and m outputs, with $m \leq n$, for which every set $k \leq m$ of the inputs can be mapped to some k outputs, but without the ability to distinguish between those outputs. An important implication of this definition is that a completely new set of paths may be determined for every new set k , therefore a concentrator is rearrangeably non-blocking.

Theoretical research has shown that it is possible to implement an n -input concentrator with $O(n)$ crosspoints for sufficiently large n [6]. In contrast, it has been shown that a rearrangeably non-blocking n -input permutation network must have at least $O(n \lg n)$ crosspoints [8]. This result highlights a fundamental difference between the two networks.

Finding explicit constructions of concentrators with a linear cost has proved difficult. Margulis was able to provide an explicit construction, but could not determine the resulting cost [9]. Gabber and Galil were able to determine the cost of Margulis's construction and were able to construct a concentrator with a cost of approximately $273n$ [10]. This value was later improved to approximately $123n$ [11]. These linear cost constructions prove to be impractical for networks of the size required in SoC design. The constant multiple in the cost equation is large and the construction of the lowest cost concentrator is not valid for values of n less than 6048.

Work on practical concentrator constructions has been done. Nakamura and Masson were able to show that the lower bound on the crosspoint complexity of a sparse crossbar (or single stage network) implementation of a (n, m) -concentrator was $(n-m+1)m$ [12]. Later an explicit construction of sparse crossbars was determined for all n and m which meets this lower bound [13]. The cost of a sparse crossbar concentrator is large for all but very small values of n and m , therefore multistage constructions are required. Jan and Oruc demonstrated a multistage concentrator network called a Mux-Merger network with $O(n \lg n)$ cost [14]. However, the constant multiple in the cost equation is high and the network depth is not balanced, making the worst-case depth large.

Figure 2 Narasimha Network Definition

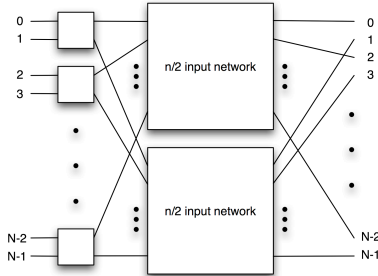


Narasimha demonstrated the construction of a multistage network that performs a superset of the functionality of a (n, m) -concentrator. This network has n inputs and n outputs for which any active set of inputs $k \leq n$ can be routed to a continuous set of output starting from the first output (Figure 2) [15]. This network can be used as a (n, m) -concentrator by ignoring the unneeded $(n-m)$ outputs. Used in this way the network provides a lower cost and depth concentrator than

any other previously described concentrator for network sizes of interest to SoC design.

Narasimha's network is constructed recursively. An n sized network is constructed from two $n/2$ -input networks (Figure 3). A stage of 2×2 crossbar switches is used to route the inputs of each of the two $n/2$ -input networks. The outputs of these networks are then interleaved. The recursive construction proceeds until the required network is 2×2 , at which point a simple crossbar is used.

Figure 3 Narasimha Network Construction



The routing on the input side proceeds in 2 possible ways for the 2×2 crossbar stage: a) if both inputs are active or inactive, the routing is set arbitrarily, and b) if only one input is active then it is routed to the lower concentrator if the previous active input was directed to the upper concentrator and vice versa. Narasimha shows that for all possible input sets this network will produce the desired output.

Narasimha only considered the case when the number of inputs is a power of 2. However, the construction of each stage of this network only requires that the number of inputs be divisible by two. When this is not the case, the size of the inputs can be increased by one and the extra input can be left unconnected.

IV. COMPARISON OF CONCENTRATOR AND PERMUTATION NETWORKS

In this section, we compare the implementation cost and depth of concentrator and permutation networks. In order to accurately compare the constructions of permutation networks and concentrators, it was necessary to determine their construction using SoC techniques. This provided a more accurate cost and depth value than comparing generic crosspoints or graph edges. Since most SoCs are designed with standard cells, the implementation of each network has been accomplished using $2:1$ standard cell multiplexers for all switching.

Since both the number of inputs and number of outputs can vary for a (n, m) -concentrator, it is difficult to present the construction cost for all configurations. However a representative graph will illustrate the difference in cost and depth between the two networks. Permutation networks and concentrators were constructed such that the number of outputs was varied while the number of inputs is held constant at 5000. The cost and depth in terms of $2:1$ multiplexers was extracted from these constructions while any gates required for signal buffering were ignored (Figure 4 and 5). The value of 5000 was selected arbitrarily based on the expected size of current large SoCs. However the

relationship between the costs of the networks remains the same for any number of inputs.

Figure 4 Permutation vs. Concentrator Network Cost

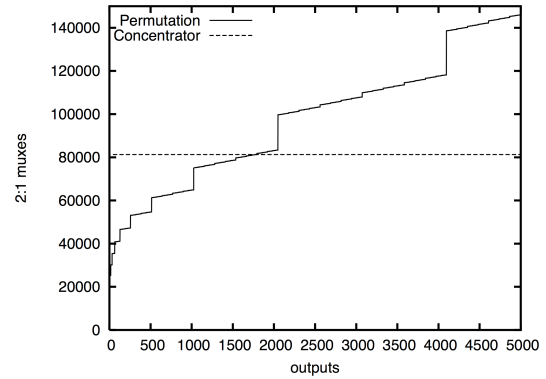
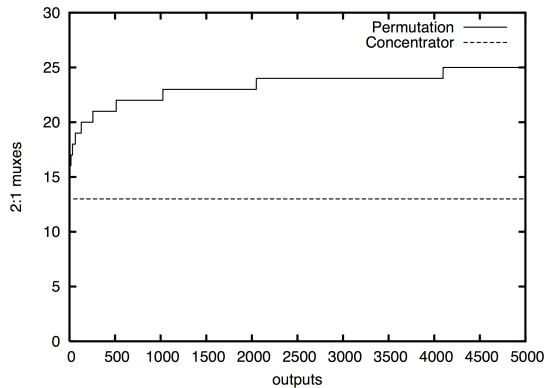


Figure 5 Permutation vs. Concentrator Depth



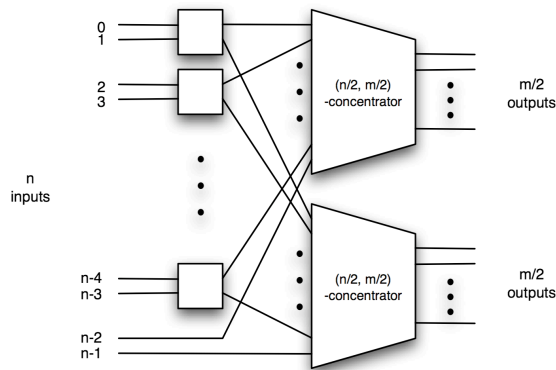
A comparison of the two networks clearly shows that when the number of outputs is small with respect to the number of inputs, the cost of the permutation network is lower than Narasimha's concentrator. This case is important for the proposed access network since there may be a large number of sources that potentially require access to the programmable logic.

V. NEW CONCENTRATOR CONSTRUCTION

A. Network Description

It is possible to create a new (n, m) -concentrator construction that is lower cost and lower depth than a permutation network for all n and m . This concentrator is built recursively from two, $(n/2, m/2)$ -concentrators (Figure 6). The first $n-2$ inputs are connected to $(n/2)-1$, 2×2 crossbars. Each of these is then connected to each of the concentrators. If the number of inputs/outputs is not divisible by 2, the next larger value is used and the unneeded inputs/outputs are ignored. The final 2 inputs are each directly connected to one of the concentrators. Each of these two smaller concentrators can be built using the same construction. Once the concentrator reaches a size where the number of outputs is ≤ 2 , a sparse crossbar construction can be used for the concentrator.

Figure 6 New Concentrator Construction



To see that this construction is non-blocking, observe that as long as each of the $(n/2, m/2)$ -concentrators has no more than $m/2$ inputs requiring a connection to its outputs, the (n, m) -concentrator will be non-blocking. This is true since by definition a $(n/2, m/2)$ -concentrator is able to route $m/2$ inputs to its outputs. Since there can be no more than m active inputs, all that is required is to show that the initial stage has enough flexibility to balance the inputs. To balance the inputs first consider the 2×2 switches that have two occupied or two unoccupied inputs, in this case the balance is maintained. Next consider the directly connected inputs, if both of these inputs are used then the balance is maintained. However, if only one is used then the first 2×2 switch with only one active input is routed to the opposite concentrator. Then for each 2×2 with only one input, the inputs are routed to each concentrator in an alternating fashion. This procedure will ensure that no more than $m/2$ inputs are routed to a single $(n/2, m/2)$ -concentrator.

B. Cost and Depth Comparison

The cost of the new concentrator construction is less than or equal to the cost of both the permutation network and Narasimha's concentrator construction for all network configurations (Figure 7). The new concentrator construction maintains the same worst-case depth as Narasimha's construction for all network configurations (Figure 8).

Figure 7 New Concentrator Cost

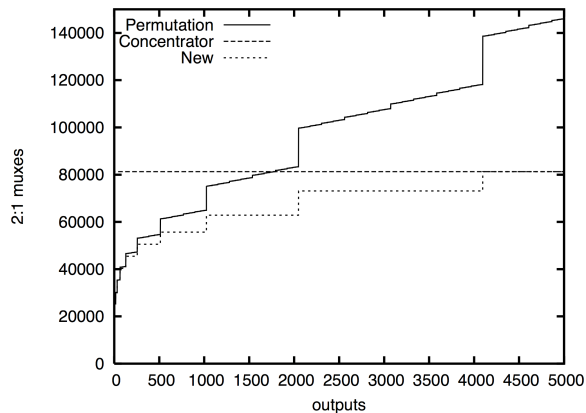
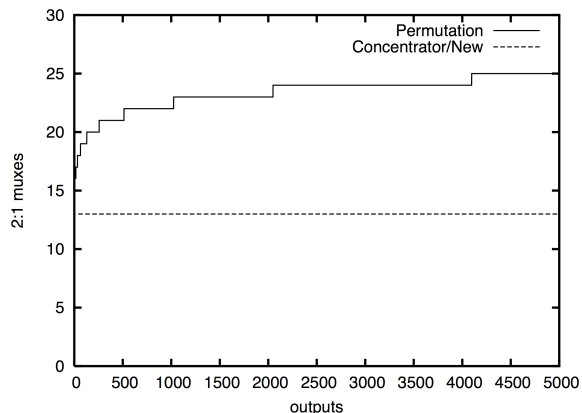


Figure 8 New Concentrator Depth



VI. CONCLUSIONS

Based on the cost and depth of the newly constructed concentrator network it is clear that a concentrator provides significantly better performance than a permutation network, while also providing equal or lower cost. Based on this access networks for programmable logic cores should take advantage of their built-in flexibility by using a concentrator construction.

REFERENCES

- [1] S.J.E. Wilton and R. Saleh, "Programmable Logic IP Cores in SoC Design: Opportunities and Challenges", *Proc. IEEE Custom IC Conf.*, San Diego, CA, pp. 63-66, May 2001.
- [2] P. Zuchowski, et al., "A Hybrid ASIC and FPGA Architecture", *Proc. IEEE/ACM Inter. Conf. on Computer-Aided Design*, pp. 187-194, 2002.
- [3] V. Raghunathan, et al., "A Survey of Techniques for Energy Efficient On-Chip Communication", *Proc. of Design Automation Conference*, pp. 900-905, June 2003.
- [4] V. E. Benes, *Mathematical Theory of Connecting Networks and Telephone Traffic*, Academic, New York, 1965.
- [5] F. K. Hwang, *The Mathematical Theory of Nonblocking Switching Networks*, World Scientific, New Jersey, 1998.
- [6] M. S. Pinsker, "On the complexity of a concentrator," in *Proc. 7th Int. Teletraffic Congr.* Stockholm, Sweden, pp. 318/1-318/4, 1973.
- [7] F. R. K. Chung, "On Concentrators, Superconcentrators, Generalizers, and Nonblocking Networks," *Bell Sys. Tech. J.*, Vol. 58, No. 8, pp. 1765-1777, 1978.
- [8] C. E. Shannon, "Memory requirements in a telephone exchange," *Bell Sys. Tech J.*, pp. 343-349, 1950.
- [9] G. A. Margulis, "Explicit constructions of concentrators", *Probl. Inform. Transm.*, vol. 9, no. 4, pp. 325-332, 1973.
- [10] O. Gabber and Z. Galil, "Explicit constructions of linear sized super-concentrators", *J. Comput. Syst. Sci.*, pp. 407-420, 1981.
- [11] N. Alon, Z. Galil, and V. D. Milman, "Better Expanders and Superconcentrators", *J. Algorith.*, vol. 8, pp 337-347, 1987.
- [12] S. Nakamura and G. M. Masson, "Lower bounds on crosspoints in concentrators", *IEEE Trans. Comput.*, vol. C-31, pp. 1173-1178, 1982.
- [13] A. Y. Oruc, and H. M. Huang, "Crosspoint Complexity of Sparse Crossbar Concentrators", *IEEE Trans. on Information Theory*, vol. 42, no. 5, pp. 1466-1471, Sept. 1996.
- [14] M. V. Chien and A. Y. Oruc, "High performance concentrators and superconcentrators using multiplexing schemes," *IEEE Trans. Communication*, vol. 42, no. 11, pp. 3045-3051, Nov.1994.
- [15] M. J. Narasimha, "A Recursive Concentrator Structure with Applications to Self-Routing Switching Networks", *IEEE Trans. on Communication*, vol. 42, no. 2/3/4, pp. 896-897, April 1994.