

Efficient Randomized Routing on Clos Networks

Abdou Youssef

Department of EECS

The George Washington University

Washington, DC 20052

Abstract: Clos networks are well-known universal multistage networks that realize all permutations. However, the routing algorithms to realize permutations on these networks are so slow that their usefulness is severely limited. This paper will give a simple, efficient randomized algorithm that routes any given permutation on Clos networks. Randomization takes place in the first column of the network, while self-routing is used in the remaining columns. Probabilistic analysis of the algorithm will be conducted and the queuing delay of any permutation will be shown to be bounded by a small constant (≤ 19) with a probability greater than 99.31%. The simplicity of the algorithm and the almost certainly constant delay will make Clos networks very practical.

§1. Introduction

Clos networks [2] [3] were among the first proposed multistage networks that are universal, that is, that realize all permutations. However, the routing algorithms to set the switches of Clos networks to realize permutations are so slow that the usefulness of these networks is severely limited [4]. One notable exception is when the number of switches in the first and last column of the Clos network is two in which case the network is self-routed and hence very useful [5]. But this special case limits the number of input/output terminals and is suitable only for very small parallel systems. Therefore, for large Clos networks, alternative solutions to Clos routing have to be sought.

An approach to fast control of Benes-Clos networks was developed in [7]. It is based on the observation that if the leftmost column of switches is set to some fixed configuration, the resulting network becomes a delta network, that is,

self-routed using destination addresses. Accordingly, the approach seeks, for every given family of permutations, a configuration to which to set the first column so that the resulting delta network realizes all the permutations of the family. If such a configuration exists, the family is said to be compatible and the left column configuration is called the compatibility factor. Compatibility factors were found in [8] for several important families of permutations that have proved useful in many parallel algorithms and applications. However, no fast algorithm to decide compatibility and compute compatibility factors has been found. Therefore, this approach does not apply to all situations, leaving the need for a more general solution to efficient Clos routing.

This paper proposes a randomized routing algorithm for general Clos networks with queueing switches. Briefly, when routing an arbitrary permutation, each source first sends its message to a uniformly randomly selected output port in its switch in the first column. Afterwards, the messages are self-routed to their destinations using the destinations addresses as routing tags. This is possible and guarantees correct delivery of messages because the network to the right of the first column of any Clos network is a self-routed delta network. Due to randomization, conflict over output ports in the first column may occur and messages may have to be queued. In the remaining columns conflict may still occur because delta networks do not realize all permutations.

The focus of the paper is on the probabilistic analysis of the queueing delay in Clos networks under this routing algorithms. It will be shown that with overwhelming probability the queueing delay is bounded by a constant. Specifically, the probability that a message will wait for a total of

at most 19 messages (ahead of it) in the queues till it reaches its destination is greater than 99.31%. Therefore, with overwhelming probability the algorithm takes a small constant time to route any permutation, assuming the communication time in a crossbar switch is constant. Thus the algorithm is extremely efficient. This efficiency makes the Clos networks compare very favorably in speed with the self-routed Batcher network [1] in addition to being cheaper in hardware.

The paper is organized as follows. Next section will review Clos networks. Section 3 will present the details of the randomized routing algorithm. The probabilistic analysis of the algorithm is carried out in section 4. Section 5 will give some concluding remarks and future directions.

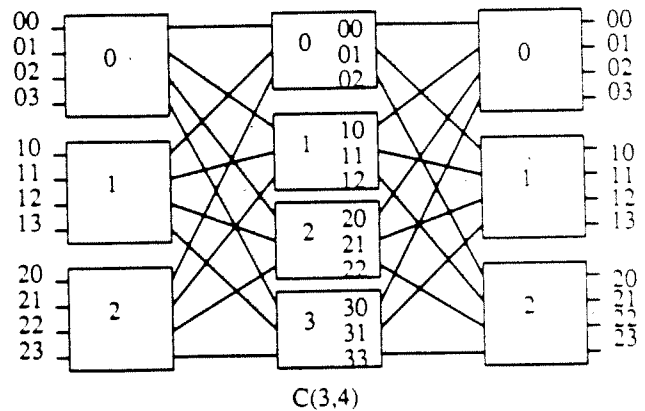
§2. Overview of Clos Networks

In this section, Clos networks will be reviewed and related concepts discussed.

Let p and q be two positive integers. A Clos network $C(p, q)$ has pq input terminals, pq output terminals and three columns of switches (see Figure 1). The left (i.e., input) column and the right (i.e., output) column have $p \times q$ crossbar switches each. The middle column has $q \times p$ crossbar switches. Assume that the switches in the leftmost and rightmost column are labeled $0, 1, \dots, p-1$, and the switches in the middle column are labeled $0, 1, \dots, q-1$. Assume also that the input ports and the output ports of every switch in the end columns (resp., middle column) are labeled $0, 1, \dots, q-1$ (resp., $0, 1, \dots, p-1$) from top to bottom. It can be seen that the input (or output) y of a switch x in any column can be uniquely identified by xy . The interconnection between the left column and the middle column is such that output port xy in the left column is linked to input port yx in the middle column. Similarly, the interconnection between the middle column and the right column is such that output port yx in the middle column is linked to input port xy in the right column. These interconnections will be referred to as the first and second interconnection, respectively.

It will prove convenient to express every integer label a of input/output terminals in two digits

$a_1 a_0$, where $0 \leq a_1 \leq p-1$ and $0 \leq a_0 \leq q-1$. a_1 represents the switch to which a belongs, and a_0 represents its position in the switch.



A Clos Network
Figure 1

It is known that if the first column of $C(p, q)$ is dropped, the remaining network is a delta network, that is, the messages can be self-routed using the destination addresses. To see this, suppose that a message is to be sent from output port xy of the left column of $C(p, q)$ to the output terminal $x'y'$ (that is, the output port $x'y'$ of the right column). Through the first interconnection the message enters the input port yx of the middle column. This input port is the input port x of switch y in the middle column. Using the first component x' of the destination address $x'y'$, the message exits the middle switch y through its output port x' , that is, through output port yx' of the middle column. Through the second interconnection the message enters the input port $x'y$ of the right column, which is the input port y of switch x' in that column. Using now the second component y' of the destination address $x'y'$, the message exits the right switch x' through its output port y' , that is, through the output port $x'y'$ of the right column, which is the desired destination. In summary, to go to destination $x'y'$, x' is

used to control the middle column, and y' is used to control the right column.

To accommodate and resolve conflicts in $C(p, q)$ when routing randomly, queues have to be used inside every switch. In every $n \times n$ switch, where $n = p$ or $n = q$, there are n queues, one queue per output port. Whenever a message enters a switch and is to exit through output port i of that switch, it goes into queue i associated with output port i . The switches in the right column need not have queues if only permutations are routed. The reason for this is that only one message will be received at every output port in the right column when a permutation is routed. In the switches of the left and middle columns, each queue needs to be of length q to avoid overflows when routing a permutation.

§3. The Randomized Routing Algorithm for Clos networks

Every input terminal processor is assumed to have an independent uniform random number generator that generates integers in the range from 0 to $q - 1$. The algorithm has two phases:

Phase 1:

Every input terminal $s = s_1 s_0$ randomly selects a random number s'_0 . Then, it sends its message with a header containing the destination address $f(s) = d_1 d_0$ to output port $s_1 s'_0$ (that is, output port s'_0 of switch s_1). The messages received at each output port are queued there. This is done by all the input terminals simultaneously.

Phase 2:

```

for  $i = 0$  to  $pq - 1$  do in parallel
  while (queue  $i$  of the left column  $\neq \emptyset$ ) do
    self-route the message in the queue front
    to its designated destination using
    the destination address;
  endwhile
endfor

```

§4. Analysis of the Algorithm

In this section, the probabilistic analysis of the algorithm will be carried out. Specifically, the probability that the queueing delay to entirely

route an arbitrary permutation is less than a given value will be derived. Based on the derived lower bound of this probability, it will be shown that with extremely high probability (approaching 1) the queueing delay is less than 19. To help with the queueing analysis, the following two lemmas about binomial distributions are needed.

4.1 Lemma. If X is a random variable of binomial distribution $B(N, p)$, and if $m \geq Np$, then $Pr[X \geq m] \leq \left(\frac{Np}{m}\right)^m e^{m-Np}$.

Proof. The proof can be found in [6]. ■

4.2 Lemma. Under the assumptions of Lemma 4.1 the following holds:

$$Pr[X \geq m] \leq \begin{cases} \left(\frac{e}{m}\right)^m, & \text{if } Np \leq 1; \\ \left(\frac{2e}{m}\right)^m, & \text{if } Np \leq 2. \end{cases}$$

Proof. It follows from Lemma 4.1. ■

Let f be a given (arbitrary) permutation of $\{0, 1, \dots, pq - 1\}$. To determine the queueing delay of f when routed by the randomized algorithm, one should consider the link conflict between every source-destination path R and all the other paths. Note that two source-destination paths can conflict in $C(p, q)$ over at most two links. Every link conflict with R causes some messages one additional queueing delay unit. Thus, the path R remains busy for a period equal to the number of link conflicts with R . Consequently, the queueing delay of f is bounded from above by the maximum time every path remains busy.

It follows from the preceding discussions that to analyze the queueing delay of f , it is sufficient to evaluate the number of link conflicts between any given arbitrary path R on the one hand and all the other source-destination paths $s \rightarrow f(s)$ of f on the other hand. Denote by $R \cap (s \rightarrow f(s))$ the set of links over which the paths R and $s \rightarrow f(s)$ conflict, where s is an arbitrary source terminal and $f(s)$ is its corresponding destination (i.e., output) terminal. We are then interested in the quantity $\sum_s |R \cap (s \rightarrow f(s))|$ which represents the total number of link conflicts between R and all the other source-destination paths of f .

Since f is routed randomly, each source-destination path $s \rightarrow f(s)$ is a random path in the sense that the output port of the left column along that path is chosen randomly and the rest of path is uniquely determined afterwards according

to that random choice. We will assume throughout that R is an arbitrary, fixed (i.e., non-random) path, and $s \rightarrow f(s)$ is a random path from source s to destination $f(s)$. Therefore, $R \cap (s \rightarrow f(s))$ is a random set, and $|R \cap (s \rightarrow f(s))|$ is an integer random variable and so is $\sum_s |R \cap (s \rightarrow f(s))|$.

It should be evident at this point that the probability distributions of the two random variables $|R \cap (s \rightarrow f(s))|$ and $\sum_s |R \cap (s \rightarrow f(s))|$ form a focal point of the probabilistic analysis of the queuing delay.

4.3 Lemma. Let R be a given path $a \rightarrow b$ from a source $a = a_1 a_0$ to a destination $b = b_1 b_0$ in $C(p, q)$, $s = s_1 s_0$ be a given source, $d = f(s) = d_1 d_0$ the corresponding destination, and $s \rightarrow f(s)$ a random path constructed by the algorithm. Let P and Q be the following two logical expressions: $P = (s_1 = a_1 \wedge d_1 \neq b_1) \vee (s_1 \neq a_1 \wedge d_1 = b_1)$ and $Q = (s_1 = a_1 \wedge d_1 = b_1)$. Then we have the following:

- a) $Pr[|R \cap (s \rightarrow f(s))| = 1] = \begin{cases} \frac{1}{q}, & \text{if } P \text{ holds;} \\ 0, & \text{otherwise;} \end{cases}$
b) $Pr[|R \cap (s \rightarrow f(s))| = 2] = \begin{cases} \frac{1}{q}, & \text{if } Q \text{ holds;} \\ 0, & \text{otherwise.} \end{cases}$

Proof. a) Assume $s_1 = a_1$ and $d_1 \neq b_1$. The fact that $d_1 \neq b_1$ implies that the destinations $f(s)$ and b are output ports of two distinct switches in the third column, and consequently, the two paths R and $s \rightarrow f(s)$ cannot conflict over a link in the second interconnection. Thus, if these two paths conflict over exactly one link, that link must be in the first interconnection. That is, the two sources a and s select the same output port in the left column. This happens with probability $\frac{1}{q}$. The case where $s_1 \neq a_1$ and $d_1 = b_1$ can be treated similarly. In the case where $s_1 \neq a_1$ and $d_1 \neq b_1$, the two paths cannot conflict over a link in the first interconnection because the two sources a and s are input ports of two different switches, neither can they conflict over a link in the second interconnection because the destinations b and $f(s)$ are output ports of two different switches. Thus, the probability of conflicting over one link is 0. In the final case where $s_1 = a_1$ and $d_1 = b_1$, the two paths can be easily seen to conflict over two links or not conflict at all. Hence, the probability of conflicting over exactly one link is 0.

b) If the two paths conflict over two links, then the sources must be input ports of one same switch and the destinations must be output ports of one same switch. Thus, we must have $s_1 = a_1$ and $d_1 = b_1$. Otherwise, the two paths cannot conflict, or equivalently, the probability of their conflict is 0. Now if $s_1 = a_1$ and $d_1 = b_1$, and if there is a conflict over a link in the first interconnection (which happens with probability $\frac{1}{q}$), then there must be a conflict in the second interconnection because $d_1 = b_1$. Thus, under this case, the probability of conflicting over two links is $\frac{1}{q}$.

■

To continue the analysis, the following definitions are needed. Let

$I_i = \{s \mid |R \cap (s \rightarrow f(s))| = i\}$, for $i = 0, 1, 2$

$\Sigma_f^R = \{s \mid s_1 = a_1 \wedge d_1 = b_1\}$, and

$\Delta_f^R = \{s \mid (s_1 = a_1 \wedge d_1 \neq b_1) \vee (s_1 \neq a_1 \wedge d_1 = b_1)\}$, where $f(s) = d = d_1 d_0$.

Let also $\sigma = |\Sigma_f^R|$ and $\delta = |\Delta_f^R|$.

The probability distribution of $\sum_s |R \cap (s \rightarrow f(s))|$ can now be addressed. Let l be a non-negative integer and $l' = \lfloor \frac{l}{2} \rfloor$.

$Pr[\sum_s |R \cap (s \rightarrow f(s))| \geq l] = Pr[(\sum_{s \in I_0} |R \cap (s \rightarrow f(s))| + \sum_{s \in I_1} |R \cap (s \rightarrow f(s))| + \sum_{s \in I_2} |R \cap (s \rightarrow f(s))|) \geq l]$ which is equal to

$$Pr[(|I_1| + 2|I_2|) \geq l]$$

because $\sum_{s \in I_i} |R \cap (s \rightarrow f(s))| = i \times |I_i|$.

We should then consider the distribution of $|I_1|$ and $|I_2|$. Note that a given source s is in I_1 if and only if s is in I_1 and Δ_f^R , after part (a) of the previous lemma. Therefore, we have the following equalities between random events:

$$[|I_1| = k] = [\text{exactly } k \text{ sources are in } I_1] = [\text{exactly } k \text{ sources in } \Delta_f^R \text{ are in } I_1].$$

As the probability that a given source s in Δ_f^R is in I_1 is $\frac{1}{q}$ (after part (a) of the previous lemma), it follows that $Pr[\text{exactly } k \text{ sources in } \Delta_f^R \text{ are in } I_1] = \binom{\delta}{k} (\frac{1}{q})^k (1 - \frac{1}{q})^{\delta - k}$. Therefore, the random variable $|I_1|$ follows a binomial distribution $B(\delta, \frac{1}{q})$. The distribution of the random variable $|I_2|$ can be similarly shown to be the binomial distribution $B(\sigma, \frac{1}{q})$.

Since $\sigma = |\{s \mid s_1 = a_1 \wedge d_1 = b_1\}| \leq |\{s \mid s_1 = a_1\}| = q$, we have $\sigma \leq q$, that is,

$\sigma \times \frac{1}{q} \leq 1$. Making use of the fact that $|I_2|$ follows the distribution $B(\sigma, \frac{1}{q})$ and applying Lemma 4.2, we conclude that $Pr[|I_2| \geq m] \leq (\frac{\epsilon}{m})^m$ for all $m \geq 1$. Similarly, $\delta = |\{s \mid (s_1 = a_1 \wedge d_1 \neq b_1) \vee (s_1 \neq a_1 \wedge d_1 = b_1)\}| = |\{s \mid s_1 = a_1 \wedge d_1 \neq b_1\}| + |\{s \mid s_1 \neq a_1 \wedge d_1 = b_1\}| \leq |\{s \mid s_1 = a_1\}| + |\{s \mid d_1 = b_1\}| \leq 2q$, yielding that $\delta \times \frac{1}{q} \leq 2$. Applying Lemma 4.2 to $|I_1|$ which follows the binomial distribution $B(\delta, \frac{1}{q})$, we conclude that $Pr[|I_1| \geq m] \leq (\frac{2\epsilon}{m})^m$ for all $m \geq 2$.

Before we proceed to bound $Pr[\sum_s |R \cap (s \rightarrow f(s))| \geq l]$, the independence between $|I_1|$ and $|I_2|$ will be shown. To see this, observe that Δ_f^R and Σ_f^R are fixed (i.e., non-random) since they depend on the given f and R , and that $\Delta_f^R \cap \Sigma_f^R = \emptyset$. As $I_1 \subseteq \Delta_f^R$ and $I_2 \subseteq \Sigma_f^R$, and as every two sources independently choose the output ports in the left column, it follows that any source s in Δ_f^R and source s' in Σ_f^R independently choose whether or not to belong to I_1 and I_2 , respectively. Consequently, the random sets I_1 and I_2 are independent, and hence their cardinalities are independent.

It can be concluded from the preceding discussions that for $l \geq 2$

$$\begin{aligned} Pr[|I_1| + 2|I_2| \geq l] &= \\ \sum_{i=0}^{l'} Pr[|I_2| = i \wedge |I_1| \geq l - 2i] &= \\ \sum_{i=0}^{l'} Pr[|I_2| = i] \times Pr[|I_1| \geq l - 2i] &= \\ Pr[|I_2| = 0] \times Pr[|I_1| \geq l] + \\ Pr[|I_2| = l'] \times Pr[|I_1| \geq l - l'] + \\ \sum_{i=1}^{l'-1} Pr[|I_2| = i] \times Pr[|I_1| \geq l - 2i] & \\ \leq Pr[|I_1| \geq l] + Pr[|I_2| = l'] + \\ \sum_{i=1}^{l'-1} Pr[|I_2| = i] \times Pr[|I_1| \geq l - 2i] & \\ \leq (\frac{2\epsilon}{l})^l + (\frac{\epsilon}{l'})^{l'} + \sum_{i=1}^{l'-1} (\frac{\epsilon}{i})^i (\frac{2\epsilon}{l-2i})^{l-2i}. \end{aligned}$$

The summand $(\frac{\epsilon}{i})^i (\frac{2\epsilon}{l-2i})^{l-2i}$ of the right-most sum above will be bounded from above next. To do so, the summand will be viewed as a continuous (as opposed to discrete) function $g(x) = (\frac{\epsilon}{x})^x (\frac{2\epsilon}{l-2x})^{l-2x}$, where x is a variable in the interval $[1, l' - 1]$. Let $h(x) = \ln(g(x)) = x - x \ln(x) + (l - 2x) \ln(2\epsilon) - (l - 2x) \ln(l - 2x)$. The derivative $h'(x) = \ln(\frac{x - \frac{1}{2}}{x})$. Thus, $h'(x) \leq 0$ if and only if $\frac{(x - \frac{1}{2})^2}{x} \leq 1$, that is, iff $x^2 - (l + 1)x + \frac{l^2}{4} \leq 0$,

which holds iff $\frac{l+1-\sqrt{2l+1}}{2} \leq x \leq \frac{l+1+\sqrt{2l+1}}{2}$. Since $x \leq l' \leq \frac{l}{2}$ as initially supposed, and since $\frac{l}{2} \leq \frac{l+1+\sqrt{2l+1}}{2}$, it follows that $h'(x) \leq 0$ if and only if $x \geq \frac{l+1-\sqrt{2l+1}}{2}$. Thus, h reaches a peak at $x = \frac{l+1-\sqrt{2l+1}}{2}$ in the interval $[1, l' - 1]$. Since $h = \ln(g)$, g and h behave in the same way, and hence, g reaches a peak at $x = \frac{l+1-\sqrt{2l+1}}{2}$ in the interval $[1, l' - 1]$. Therefore, for every $i = 1, 2, \dots, l' - 1$, we have

$$\left(\frac{\epsilon}{i}\right)^i \left(\frac{2\epsilon}{l-2i}\right)^{l-2i} \leq g\left(\frac{l+1-\sqrt{2l+1}}{2}\right) \text{ which is equal to } \left(\frac{e}{\frac{l+1-\sqrt{2l+1}}{2}}\right)^{\frac{l+1-\sqrt{2l+1}}{2}} \left(\frac{2e}{\sqrt{2l+1}-1}\right)^{\sqrt{2l+1}-1}.$$

We have thus proved the following theorem.

4.4 Theorem. For every given permutation f and every given source-destination path R in $C(p, q)$, and for every integer $l \geq 2$ the following holds

$$\begin{aligned} Pr[\sum_s |R \cap (s \rightarrow f(s))| \geq l] &\leq \left(\frac{2\epsilon}{l}\right)^l + \left(\frac{\epsilon}{\lfloor \frac{l}{2} \rfloor}\right)^{\lfloor \frac{l}{2} \rfloor} \\ &+ (\lfloor \frac{l}{2} \rfloor - 1) \left(\frac{e}{\frac{l+1-\sqrt{2l+1}}{2}}\right)^{\frac{l+1-\sqrt{2l+1}}{2}} \left(\frac{2e}{\sqrt{2l+1}-1}\right)^{\sqrt{2l+1}-1} \end{aligned} \quad (1)$$

where, for every source s , the path $s \rightarrow f(s)$ is a random source-destination path in $C(p, q)$ selected by the randomized routing algorithm.

Every term in the right hand side of (1) can be shown to converge to 0 exponentially as l increases. Therefore, for every constant ϵ , $0 < \epsilon < 1$, there exists some integer $l_0 \geq 2$ such that $Pr[\sum_s |R \cap (s \rightarrow f(s))| \geq l_0] < \epsilon$, or equivalently, $Pr[\sum_s |R \cap (s \rightarrow f(s))| \leq l_0 - 1] \geq 1 - \epsilon$. As the convergence is exponential, it follows that l_0 remains reasonably small for arbitrarily small ϵ . In other terms, with an overwhelming probability approaching 1, the queuing delay is smaller than a small value $l_0 - 1$.

More specifically, the right hand side of (1)

$$\text{is equal to } \begin{cases} 0.222541, & \text{if } l = 16; \\ 0.0421121, & \text{if } l = 18; \\ 0.0069, & \text{if } l = 20. \end{cases}$$

Consequently,

$$Pr[\sum_s |R \cap (s \rightarrow f(s))| \leq 15] \geq 0.77$$

$$Pr[\sum_s |R \cap (s \rightarrow f(s))| \leq 17] \geq 0.95$$

$$Pr[\sum_s |R \cap (s \rightarrow f(s))| \leq 19] \geq 0.9931$$

thus concluding that with a probability greater

than 99.31%, the queueing delay to route any permutation in $C(p, q)$ is at most 19 for all p and q . This almost certain queueing delay is very small, especially for large networks, and is by far faster than any existing algorithm that routes arbitrary permutations on Clos networks.

§5. Conclusions

This paper presented a simple, efficient randomized routing algorithm for Clos networks. Probabilistic analysis of the algorithm was carried out and the queueing delay when routing any permutation was shown to be a small constant (≤ 19) with a probability greater than 99.31%. The simplicity of the algorithm and the almost certainly constant delay make Clos networks very practical.

Future work will extend the algorithm and its probabilistic analysis to Clos networks of more than 3 stages (defined recursively). The algorithm will also be applied to mesh and torus networks as there is a direct correspondence between Clos networks and cartesian product graphs.

§6. References

- [1] K. E. Batcher, "Sorting networks and their applications," *1968 Spring Joint Comput. Conf., AFIPS Conf.* Vol. 32, Washington, D.C.: Thompson, 1968, pp. 307-314.
- [2] V. E. Benes, *Mathematical theory on connecting networks and telephone traffic*, Academic Press, New York, 1965.
- [3] C. Clos, "A study of non-blocking switching networks," *Bell System Tech. J.* Vol. 32, pp. 406-424, 1953.
- [4] D. C. Opferman and N. T. Tsao-Wu "On a class of rearrangeable switching networks, Part I: Control algorithms," *Bell System Tech. J.* Vol. 50, No. 5, pp. 1579-1600, 1971.
- [5] B. G. Douglas and A. Y. Oruc, "On self-routability of three-stage switching networks," Proc. of the 27-th Annual Allerton Conf. on Comm., Control & Computing, pp. 640-648, Sept. 1989.
- [6] L. G. Valiant, "A scheme for fast parallel communication," *SIAM J. Comput.*, Vol. 11, NO. 2, pp. 350-361, May 1982.
- [7] A. Youssef and B. Arden, "A new approach to fast control of $r^2 \times r^2$ 3-stage Benes networks of $r \times r$ crossbar switches," *Proc. 17th Int'l Symp. Comp. Arch.*, Seattle, pp. 50-59, May 1990.