COSET NETWORKS AS CONNECTORS IN PARALLEL PROCESSORS*

A. Yavuz Oruç
Electrical Engineering Department,
University of Maryland,
College Park, MD 20740

Seth Schneider
Electrical, Computer and Systems Engineering Department,
Rensselaer Polytechnic Institute,
Troy, NY 12180

ABSTRACT

An active area of research regarding parallel computer systems deals with the design of interconnection networks. Among all interconnection networks, permutation networks play a special role as all other networks can be constructed using such networks. It was recently shown that many permutation networks reported in the literature are manifestations of coset decompositions of symmetric groups. This result is generalized here to obtain several other previously unknown permutation networks which satisfy such decompositions. In addition, analyses of the edge-count, propagation delay, fan-out and setup time of such networks are provided. The results lead to some anamolous behavior as well as several tradeoffs among these parameters. For example, it is shown that a complete bipartite graph is the fastest and most economical direct realization of a permutation network. Furthermore, it is shown that the fan-out of a network is inversely proportional to the propagation delay whereas the setup time may or may not relate to the propagation delay at all depending on the network in question. Finally, two constant fan-out implementations of these networks using $O(n^{1.59})$ 2×1 multiplexers, and 2×2 switches are presented.

This work is supported in part by the National Science Foundation under Grant No: CCR-8708864.

1 Introduction

Most parallel computations need an infrastructure to rearrange their operands between succesive operations. These rearrangement patterns take many forms ranging from simple shifts to arbitrary permutations, and if computations need to be executed faster such transformations must be collectively coded into hardware by using what is commonly referred to as a *permutation network*. Simply stated, a permutation network is a device which can provide any one of $n!$ one-to-one and onto mappings between two sets of terminals, each having n elements. Networks presented in this paper are permutation networks which are synthesized in a particular way by decomposing symmetric groups into cosets. They will, therefore, be referred to as *coset networks* to emphasize this relationship.

Coset networks play a key role in array processor architectures [Feng 1981], [Lawrie 1975], [Parker 80], [Pease 1977], [Siegel 1985], [Siegel et al. 87]. Typically, an array processor uses a coset network to provide direct simultaneous communications among its processors, or between its processors and memory units. In this type of computing environment coset networks outperform unique-path connectors such as omega, shuffle-exchange and indirect binary cube networks [Lawrie 1975], [Parker 80], [Pease 1977] because the latter networks are all blocking, whereas coset networks are rearrangeably nonblocking. Coset networks can also be used to construct other types of connection networks such as partitioners [Gecsei 1977], and generalized connection networks [Thompson 1978], [Masson et al. 1979], [Thurber 1978] which are more suitable for multiprocessing systems.

A practical problem regarding coset networks is concerned with the customization of network parameters such as edge-count, fan-out and fan-in, propagation delay, and setup time. Each of these parameters measures the performance of a network in some fashion. Edge-count is a measure of cost, fan-in and fan-out are parameters which are often constrained by the available technology, propagation delay of a network is the number of edges along the longest path from any of its inputs to any of its outputs, and setup time is the amount of time it takes to decide the states of all of the switches in the network to realize a permutation. For most of the networks considered in this paper, fan-in and fan-out values will be identical, and henceforth we shall speak only in terms of fan-out.

The customization of network parameters was addressed earlier in the literature by

[Clos 1953] and [Oruc and Thirumalai 1987]. In general, the customization process is applied within a family of coset networks whose parameters can be controlled by varying certain design options. For example, a 3-stage Clos network [Clos 1953] can be constructed out of cells of many different shapes and sizes. Each cell size leads to a different fan-out and cost. Furthermore, by decomposing the cells into smaller Clos networks, one can also control the propagation delay and setup time.

The parameter customization problem was addressed in [Oruc and Thirumalai 1987] in the context of another family of coset networks which result from recursive decompositions of symmetric groups. Certain observations were made about the relations between edge-count, fan-out and propagation delay for several networks, but an indepth analysis of the available tradeoffs was not provided. Here the recursive decompositions of symmetric networks are generalized to probe further into the nature of such tradeoffs. First, it is shown that the coset networks which are obtained from such decompositions exhibit a wide spectrum of values for the four network parameters. For example, it is demonstrated that a complete bipartite graph is both the least expensive and fastest realization of a coset network which can be synthesized within this family of networks. It is also shown that fanout is inversely proportional to the propagation delay of a network whereas the setup time may or may not relate to the depth of decomposition depending upon the network in question. It is further shown that the edge-count increases as fan-out decreases. Finally, implementations of coset networks using multiplexers and 2×2 switches are presented. These constructions provide both a reduction in edge-count over the networks given in [Oruc and Thirumalai 1987] and maintain a uniform fan-out over the depth of decomposition.

The remainder of the paper is organized as follows. Section 2 provides the coset decomposition design approach, and derive expressions for edge-count, propagation delay, and fan-out for a representative coset network. In Section 3, these results are generalized to other coset networks, and in Section 4 network setup procedures are presented. The implementations of coset networks are given in Section 5, and the paper is concluded in Section 6.

2 Design Approach

It is well-known that the set of all permutations over n symbols forms the symmetric group of degree n, denoted Σ_n and has $n!$ elements. The complete bipartite graph is an explicit realization of Σ_n in that every permutation is coded in terms of direct edges between its inputs and outputs. An alternative is to decompose Σ_n into disjoint sets of permutations and code each set by a distinct permutation. This decomposition ties with the notion of *cosets* as follows.

Definition: Let Σ_n and Σ_m denote the symmetric groups defined over sets $\{1, 2, \ldots, n\}$ and $\{1, 2, \ldots, m\}$ inputs respectively, where $m \leq n$. Then a *right coset* of Σ_m in Σ_n is defined for a permutation $p \in \Sigma_n$ as the set of permutations: $\Sigma_m \cdot p = \{\sigma_m \cdot p : \sigma_m \in \Sigma_m\}$ ||.

In words, $\Sigma_m \cdot p$ is a translation of the permutations in Σ_m into another set of $m!$ permutations obtained by postmultiplying each of its permutations by p. A $left\ coset$ is defined similarly except that the translation is performed by premultiplication rather than postmultiplication by p.

Right, and left cosets are the basis of decomposing a coset network into a cascade of two networks of smaller size. Here, the underlying fact is that any two cosets are either disjoint or identical. Consequently, we can decompose Σ_n into its right or left cosets as

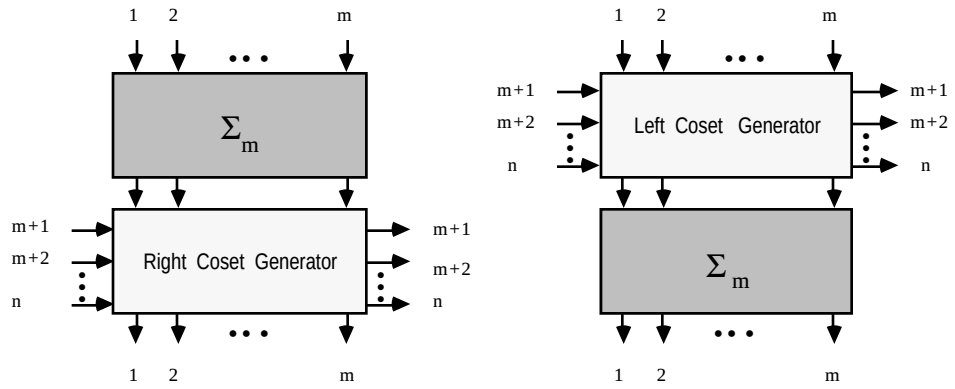
$$\Sigma_n = \Sigma_m \cdot p_1 \cup \Sigma_m \cdot p_2 \cup \cdots \cup \Sigma_m \cdot p_r$$

$$\Sigma_n = p'_1 \cdot \Sigma_m \cup p'_2 \cdot \Sigma_m \cup \cdots \cup p'_r \cdot \Sigma_m$$

where the cosets in each expansion are pairwise disjoint, and r is the number of right (left) cosets.

We depict the network realizations corresponding to these expressions in Figure 1. Each network consists of two subnetworks: a network, called Σ_m in each of the figures, that realizes the symmetric group over its set of inputs, $\{1, 2, \ldots, m\}$, and another network, called a *coset generator*, that generates Σ_n together with the Σ_m network. It should be noticed that the network in Figure 1(b) can be obtained by reversing the directions of the arrows in the network of FIgure 1(a). Consequently, we shall carry out our discussion only in terms of right cosets and Figure 1(a).

In order to complete the decomposition of an n-input permutation network into a 2-stage



(a) Right Coset Decomposition **(b) Left Coset Decomposition**

Figure 1: Networks based on coset generators.

network as depicted in Figure 1(a), we must specify how the inputs and outputs of the coset generator are connected. First note that we must place an edge between every horizontal input and every output of the coset generator, since lacking any such edge will prevent the realization of some $(n-1)!$ permutations. Since there are $n-m$ horizontal inputs and n outputs, this requires a total of $n(n-m)$ edges. These edges are shown in Figure 2(a) for one horizontal input.

As for the vertical inputs, we have the following theorem.

Theorem 1: If a 2-stage coset network can realize Σ_n then each vertical input of the coset generator is connected to at least $n-m+1$ outputs.

Proof: Suppose a vertical input is connected to only s outputs where $s < n-m+1$. We can then construct a permutation p which maps some set of $s \leq n-m$ horizontal inputs to those outputs and hence prevent that vertical input from being connected to any output. That is, the network cannot realize p and so the assertion. ||

Moreover, we have

Theorem 2: If a 2-stage coset network can realize Σ_n then each vertical input of the coset generator need not be connected to more than $n-m+1$ outputs.

Proof: Connect each vertical input to all the horizontal outputs. This requires $n-m$ edges per each vertical input. Using one more edge, connect each vertical input to the vertical output which is directly below it as shown in Figure 2(b) for one vertical input. In fact,

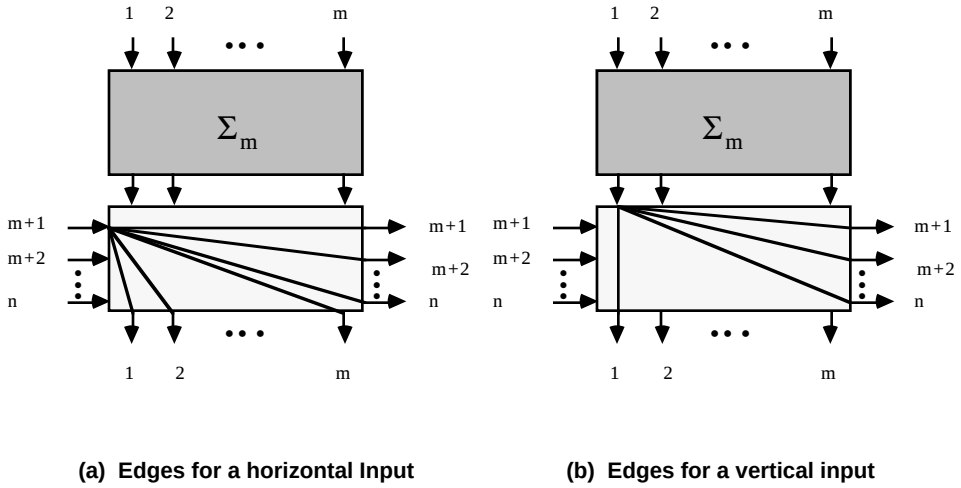


Figure 2: Edges in the coset generator for horizontal and vertical inputs.

the vertical edges can be connected quite arbitrarily so long as they induce a permutation between the vertical inputs and vertical outputs of the coset generator. To conclude that the edges specified so far suffice to generate all $n!$ permutations, let p be an arbitrary permutation. p can be viewed as a composition of three one-to-one maps p_h, p_{vv}, and p_{vh} where p_h is that portion of p which maps horizontal inputs to outputs (both horizontal and vertical), p_{vv} is that portion of p which maps vertical inputs to vertical outputs, and p_{vh} is that portion of p which maps vertical inputs to horizontal outputs. To realize p_h, activate the edges between the horizontal inputs and their images under p on the output side. This can always be done, regardless of how p is specified since each horizontal input is connected to all outputs and the mapping is one-to-one. Now, consider the vertical inputs. To realize p_{vv}, use the coset network at the top so that, if vertical input x is to be mapped to vertical output y, then the edge inside the Σ_m network which falls between input x, and that input of the coset generator, say x', which is directly above output y is activated. To complete the path, activate the edge in the coset generator that falls between x' and output y. This is always possible, regardless of what p is, since the top subnetwork can map any of its inputs onto any of its outputs, and furthermore, an edge is made available between each vertical input of the coset generator and the vertical output which is directly below it (see Figure 2(b).) Now, to complete the mapping, we must use the remaining, i.e., unassigned inputs of the coset generator to realize p_{vh}. This can be done, since the vertical inputs of the coset

generator can be reached from the inputs of Σ_m in any one-to-one fashion, and since there exists an edge between each vertical input and each horizontal output of the coset generator. That is, they are connected in a complete bipartite fashion, and therefore any subassignment that remains between vertical inputs and horizontal outputs is always realizable. ||

It follows from the preceding results that we need to have $(n-m)n + (n-m+1)m$ edges for the coset generator. Assuming that the top subnetwork is realized by a complete bipartite graph, the overall network then has

$$m^2 + (n-m)n + (n-m+1)m = n^2 + m \tag{1}$$

edges. Notice that this network uses m more edges than a complete bipartite graph. Moreover, its propagation delay is twice as long as that for a complete bipartite graph. However, unlike a complete bipartite graph, only $n-m$ of its inputs have fan-out $= n$. The other inputs, i.e., the vertical inputs of the Σ_m network have fan-out $= m$ and vertical inputs of the coset generator have fan-out $= n-m+1$.

The relation between edge-count and fan-out can be further analyzed by recursively decomposing the top subnetwork Σ_m network into two stages. Let C_n denote the number of edges in an n-input coset network obtained this way. For an example of such a decomposition, refer to Figure 3. It was shown in [Oruc and Thirumalai 1987] that

$$C_n = (2k+1)\left\{n\alpha - \frac{k\alpha(\alpha-1)}{2}\right\} - \alpha(k^2+k) + C_{n-\alpha k} \tag{2}$$

where $k = n-m$ is the number of horizontal inputs to each of the coset generators, α is the depth of recursion, and $C_{n-\alpha k}$ is the number of edges in the $\Sigma_{n-\alpha k}$ subnetwork which is left undecomposed. In the network of Figure 3, $n = 8$, $k = 2$, and $\alpha = 3$. The reader interested in the derivation of Equation 2 is referred to the appendix.

Assuming that k divides n and the recursion is terminated when $n - \alpha k = 0$, we have

$$n - \alpha k = 0, \tag{3}$$

or,

$$\alpha = \frac{n}{k} \tag{4}$$

and the propagation delay P_n of the resultant network becomes

$$P_n = \alpha = \frac{n}{k}. \tag{5}$$

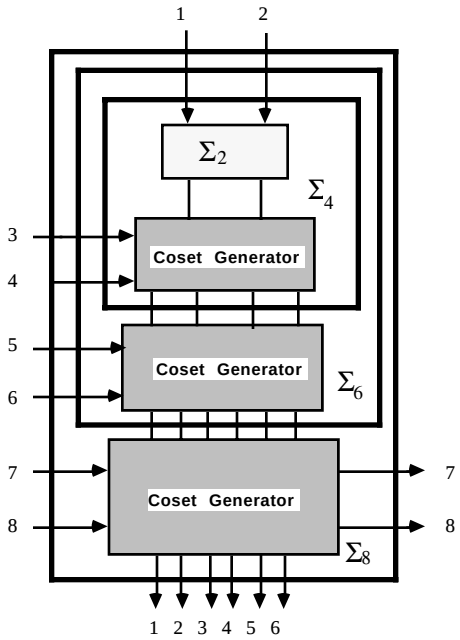


Figure 3: A recursive decomposition of a 2-stage network.

By substituting for α in (2) and letting $C_{n-\alpha k} = C_0 = 0$, we obtain

$$C_n = n^2 - \frac{n}{2} + \frac{n^2}{2k}. \tag{6}$$

Finally, it follows from the construction of the coset generator that

$$fan - out = k + 1 \tag{7}$$

for all vertical inputs and

$$fan - out = n - (i-1)k \tag{8}$$

for all horizontal inputs of the coset generator in decomposition level $i; 1 \leq i \leq \frac{n}{k}$.

We can summarize the results of this section as follows.

Theorem 3: In a coset network which is recursively decomposed through a single subnetwork into a cascade of coset generators with k horizontal inputs, the larger the value of k, the smaller C_n, P_n and fan-out for horizontal inputs become, and the larger the fan-out for vertical inputs gets. ||

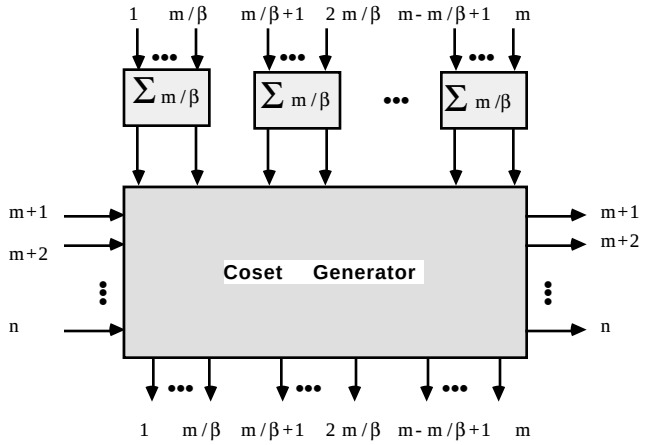


Figure 4: 2-stage realization of Σ_n with β subgroups.

3 Extension to Other 2-Stage Coset Networks

The coset networks described in the previous section are a special case of a more general coset decomposition of Σ_n. In this section we consider the network realization of this more general coset decomposition and provide formulas for its edge complexity, propagation delay and fan-out.

The basis for the generalization is to divide the m vertical inputs into β sets of $\frac{m}{\beta}$ inputs each for some β which divides m, and replace the Σ_m subnetwork by β subnetworks where the ith subnetwork realizes the symmetric group over the ith set of $\frac{m}{\beta}$ inputs. The 2-stage network corresponding to this decomposition is depicted in Figure 4.

The edges inside the coset generator are specified as follows. As in the construction of the coset generator in the earlier section, the horizontal inputs on the left are connected to all outputs. This is so, since these inputs are not permuted onto the outputs in any way prior to arriving at the coset generator. This accounts for $(n-m)n$ edges.

As for the vertical inputs of the coset generator, they are divided into β groups and the inputs in each group are connected to the outputs directly below them in a one-to-one and onto fashion. We then establish a complete bipartite graph between the inputs in each group and all horizontal outputs and all vertical outputs except those which are directly

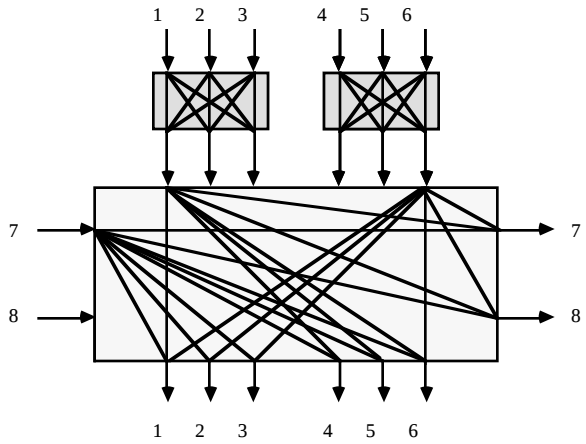


Figure 5: A partial description of coset generator for $n = 8$, $m = 6$, and $\beta = 2$.

below them. All of these edges are illustrated in Figure 5 for one horizontal and one vertical input from each subnetwork where $n = 8$, $m = 6$, and $\beta = 2$. It is easily seen that each vertical input in this connection scheme is connected to $n - m + \frac{m(\beta-1)}{\beta} + 1$ edges. The necessity and sufficiency of these edges to realize Σ_n are stated in the following result.

Theorem 4: If a 2-stage coset network with β subnetworks in its top stage realizes Σ_n then it is both sufficient and necessary that the fan-out of each vertical input of its coset generator be $n - m + \frac{m(\beta-1)}{\beta} + 1$.

Proof: To prove the necessity case, suppose that $fan - out < n - m + \frac{m(\beta-1)}{\beta} + 1$ for some input within a group of $\frac{m}{\beta}$ vertical inputs. We can then construct a permutation, say p, that maps the $n - m$ horizontal inputs, and $\frac{m(\beta-1)}{\beta}$ vertical inputs in all of the remaining groups of vertical inputs to the $n - m + \frac{m(\beta-1)}{\beta}$ outputs to some of which that vertical input is connected. This then blocks that input, and hence p cannot be realized. The proof of sufficiency follows along a similar argument given in Theorem 2. ||

Using Theorem 4, the number of edges required for a 2-stage coset network with β subnetworks in its first stage can be determined as

$$C_n = (n - m)n + (n - m + \frac{m(\beta - 1)}{\beta} + 1)m + \beta C_{\frac{n-k}{\beta}}. \tag{9}$$

The first term accounts for the edges connected to the horizontal inputs of the coset generator,

the second term is the number of edges connected to all of the vertical inputs, and the last term accounts for the number of edges in all of the β subnetworks in the first stage.

If the top subnetworks are recursively decomposed as before, it can be shown that (see the appendix)

$$C_n = \alpha(n + \frac{k}{\beta - 1}) - \frac{(\beta^\alpha - 1)}{(\beta - 1)^2}(k^2 + \beta k) + (n + \frac{k}{\beta - 1})^2(1 - \frac{1}{\beta^\alpha}) + \beta^\alpha C_{\frac{(\beta-1)n-(\beta^\alpha-1)k}{\beta^\alpha(\beta-1)}}. \quad (10)$$

Furthermore, this expression does simplify to Equation (2) after a few straightforward algebraic manipulations. Finally, by letting $\frac{n-(\beta^\alpha-1)k}{\beta^\alpha} = 0$ and $C_{\frac{(\beta-1)n-(\beta^\alpha-1)k}{\beta^\alpha(\beta-1)}} = 0$, we obtain

$$P_n = \alpha = \lfloor \log_\beta \left(\frac{(\beta - 1)n + k}{k} \right) \rfloor, \quad (11)$$

$$C_n = n^2 - \frac{\beta n}{\beta - 1} + \frac{(\beta - 1)n + k}{\beta - 1} \lfloor \log_\beta \left(\frac{(\beta - 1)n + k}{k} \right) \rfloor, \quad (12)$$

$$fan - out = \begin{cases} \frac{1}{\beta^{i-1}}(n + \frac{k}{\beta-1}) - \frac{k}{\beta-1} & \text{for a horizontal input at level } i; 1 \le i \le \alpha \\ 1 + \frac{n(\beta-1)+k}{\beta^i} & \text{for a vertical input at level } i; 1 \le i \le \alpha, \end{cases} \quad (13)$$

where $k = n - m$ as before. It is noted that the expressions for P_n and C_n do not simplify to their counterparts given in Equations 5 and 6 by simply letting $\beta = 1$. This is due to the fact that $\frac{n-(\beta^\alpha-1)k}{\beta^\alpha} = 0$ has no real solution for α when $\beta = 1$. Note, however, the expressions for the fan-out do reduce to those given in Equations 7 and 8 when $\beta = 1$.

To provide a further insight to these formulas, Figure 6 shows a coset network which is obtained by recursively decomposing a 2-stage network with $n = 14, k = 2$, and $\beta = 2$. For clarity, only the edges for three inputs are shown in the bottom coset generator. These values result in $P_n = 3$, $C_n = 216$. Fan-out for the vertical inputs varies between 2 and 9, that for the horizontal inputs varies between 2 and 14. It should be noticed that both the edge-count and propagation delay of this network are larger than those of the 14-input complete bipartite graph. On the other hand, the fan-out of this network is smaller than that of the 14-input complete bipartite graph except for two inputs. In fact, by direct examination of the edge-count expressions derived above and in the earlier section, we conclude the following main results of this section.

Theorem 5: Among all the coset networks which can be derived from the constructions of Theorems 1 2 and 4, the complete bipartite graph has the minimal edge-count, shortest propagation delay, and maximum fan-out. ||

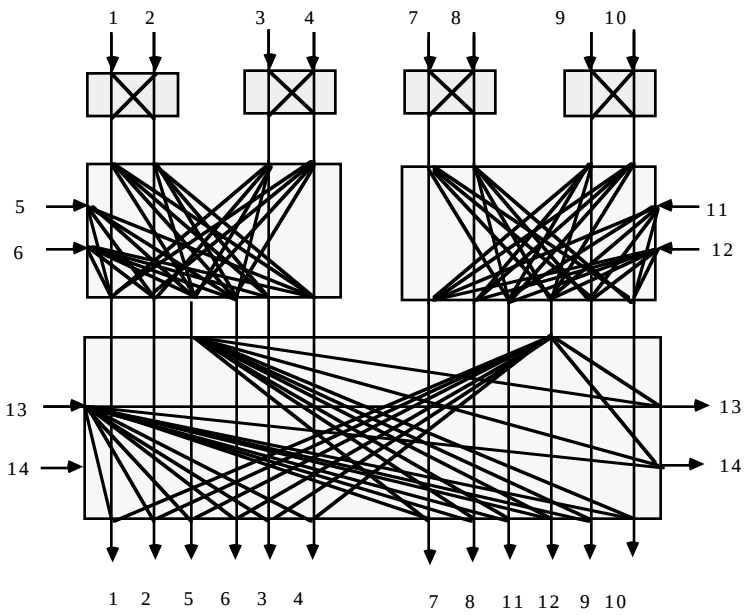


Figure 6: A partially specified 14-input coset network.

More generally, we have

Theorem 6: In a coset network which is recursively decomposed through β subnetworks into a cascade of coset generators, the larger the value of β, the smaller C_n, P_n, and fan-out for horizontal and vertical inputs become. ||

4 Setup Procedures

Setting up a coset network is the process of specifying which of its edges should be activated to realize a given permutation. The time which is needed to determine the edges to be activated along with the propagation delay characterizes the speed of the network in question. In this section we show that coset networks and their recursive decompositions can be set up in $O(n)$ serial time when the decomposition uses a single subnetwork. This time is the best possible since it takes at least n steps to read in an n size permutation from a single port random access memory. When the number of subnetworks is larger than one, then the setup time is shown to be $O(nP_n)$ where P_n is the propagation delay of the network. This time is also shown to be optimal for any serial setup algorithm.

First we consider the case of the top stage being composed of a single subnetwork Σ_m. To illustrate the setup procedure, refer to Figure 7 where $n = 10$, $m = 6$ and let the permutation

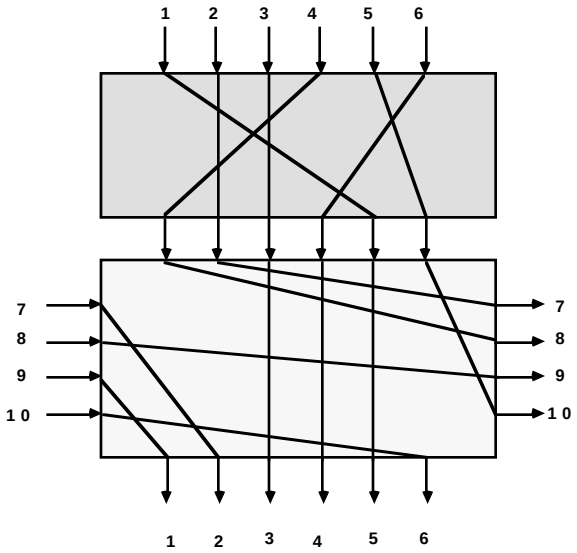


Figure 7: Illustration of the setup procedure for a single subnetwork.

map p be

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 5 & 7 & 3 & 8 & 10 & 4 & 2 & 9 & 1 & 6 \end{pmatrix}.$$

Read the outputs to which the four horizontal inputs $7, 8, 9$, and 10 are to be connected, and mark the appropriate edges in the coset generator as indicated in Figure 7. In addition, record, in a set called N_1, those vertical outputs to which $7, 8, 9$, and 10 are connected. In this example, $N_1 = \{2, 1, 6\}$. Next read the inputs to which the four horizontal outputs 7, 8, 9, and 10, are to be connected and record those horizontal outputs whose inputs come from the set of vertical inputs, in a set called N_2. Again, in this example, $N_2 = \{7, 8, 10\}$. Select a permutation between the inputs in N_1 and the outputs in N_2 and mark the appropriate edges in the coset generator to affect this permutation. In this example, these edges are also indicated in Figure 7. Assuming that the coset generator is initialized to the identity permutation by default, this completes the setup procedure for the coset generator. Once the assignment for the coset generator is completed, the submap for the top subnetwork can be obtained directly from p by replacing the outputs in N_2 with the symbols in N_1. For the

current example, this submap is

$$p' = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 2 & 3 & 1 & 6 & 4 \end{pmatrix}.$$

The above example can be formalized into the following setup procedure for a recursively decomposed coset network which uses a single subnetwork in each level of the decomposition.

Procedure Setup(p, n, k)
If $n = k$
then// Set up last subnetwork
  for $i = 1$ **to** k **do**
  Mark edge $(i, p(i))$ in Σ_k
  endfor
else// Set up the coset generator
  Begin
  for $i = n - k + 1$ **to** n **do**
  Mark edge $(i, p(i))$ in the coset generator
  endfor
  $j := 1$
  for $i = n - k + 1$ **to** n **do**
    if $p(i) \in \{1, 2, \ldots, n - k\}$
    then $N_1(j) := p(i)$ $j := j + 1$
    else {skip}
  endfor
  $j := 1$
  for $i = n - k + 1$ **to** n **do**
    if $p^{-1}(i) \in \{1, 2, \ldots, n - k\}$
    then $N_2(j) := p(i)$ $j := j + 1$
    else {skip}
  endfor
  for $i = 1$ **to** j **do**
  Mark edge $(N_1(i), N_2(i))$ in the coset generator
  endfor
  // Find the map for the next subnetwork.
  for $i = 1$ **to** j **do**
  $p(p^{-1}(N_2(i))) := N_1(i)$
  endfor
  // Set up the next subnetwork.
  Setup$(p, n - k, k)$
  End
Endprocedure

It can immediately be verified that each of the **for** loops between the **Begin** and **End** statements takes $O(k)$ steps. Thus the setup time T_n of a recursively decomposed coset network which uses a single subnetwork in each decomposition is given by the recurrent relation

$$T_n = O(k) + T_{n-k} \tag{14}$$

subject to the boundary condition $T_k = k$. The solution of this recursive formula gives $T_n = O(n)$.

The above setup procedure can be extended to coset networks with β subnetworks. In this case, it takes $O(k)$ time to mark the edges between the horizontal inputs and their corresponding images. In addition, we can mark the remaining edges, by reading the images of all the vertical inputs in $n - k$ time. Thus a total of n steps suffice to set up the coset generator. The maps for the subnetworks can also be determined in $O(n)$ steps once the edges for the coset generator are specified. Notice that in this case, even if the coset generator is assumed to be initialized to the identity permutation, the inputs from a given subnetwork which are not mapped to the horizontal outputs of the coset generator cannot just be tied to the vertical outputs which are directly below them since other subnetworks can also contribute to these vertical outputs. This makes it necessary to make at least n memory accesses to set up the coset generator. Furthermore, if the subnetworks are recursively decomposed, then the setup time T_n is given by

$$T_n = O(n) + \beta T_{\frac{n-k}{\beta}} \tag{15}$$

The solution of this equation yields $T_n = O(P_n(n + \frac{k}{\beta-1}) - \frac{\beta^{P_n}-1}{(\beta-1)^2})$ where P_n is the propagation delay of the decomposed network. Notice that if we let $\beta = 2$, $k = O(1)$ then by Equation (11), $P_n = \log_2 n$, and hence upon simplfying Equation (15), we obtain $T_n = O(n \log_2 n)$.

5 Network Implementation

The derivations of the preceding sections suggest that it is in principle, possible to construct coset networks by directly coding the described edge schemes into hardware. Nonetheless, the heterogeneity of the fan-out values over the range of decomposition as exhibited in the network of Figure 6 can make such a construction undesirable. This problem was addressed

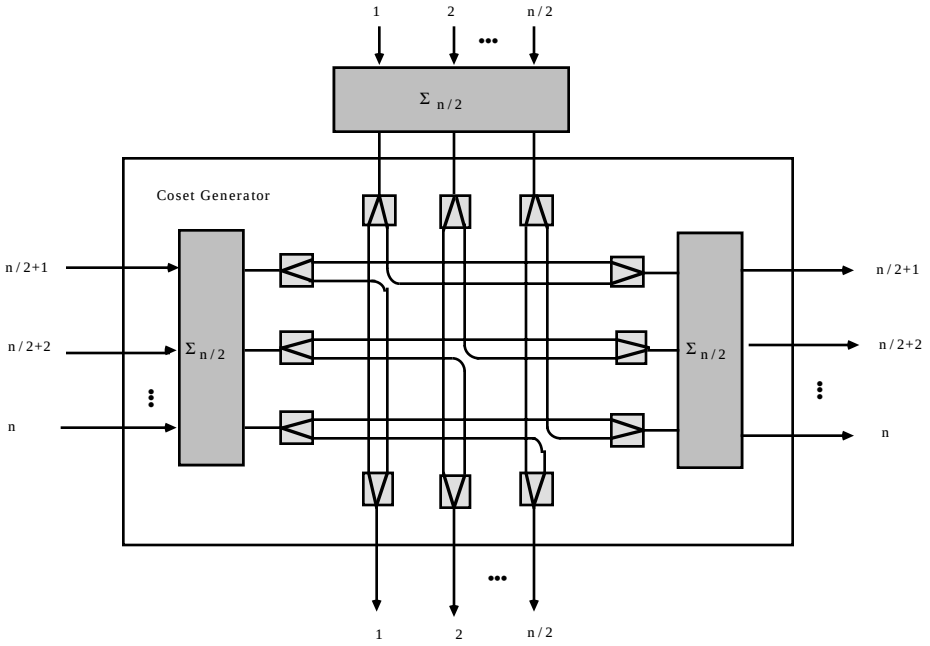


Figure 8: Multiplexer/demultiplexer implementation of a coset network.

in [Oruc and Thirumalai 1987] and as a solution, coset generators were implemented by hexagonal cells with three inputs and three outputs, leading to cellular permutation arrays. One drawback of these arrays is that they use $O(n^2)$ hexagonal switches. In this section, we present two alternative implementations of coset networks which provide uniform fan-out as well as further reduction in edge-counts.

The first implementation uses multiplexers, demultiplexers and two coset networks of size $n-m$ to replace the coset generator in a 2-stage decomposition of a Σ_n network as depicted in Figure 8 for $m = \frac{n}{2}$. It should be easy to see that all of the permutations which are realizable by the original coset generator can also be realized by this new connection scheme. Just notice that the two $\Sigma_{n/2}$ networks along with the multiplexers and demultiplexers inside the coset generator make it possible to permute the horizontal inputs onto any $n/2$ of the outputs in any one of $(n/2)!$ ways. Likewise, the vertical inputs can be permuted to any $n/2$ of the outputs in any one of $n/2!$ ways without disturbing the horizontal input assignments.

In addition, this scheme limits the fan-outs of all the terminals which are tied to the multiplexers and demultiplexers to two. What is more is that each of the three $\Sigma_{n/2}$ subnetworks

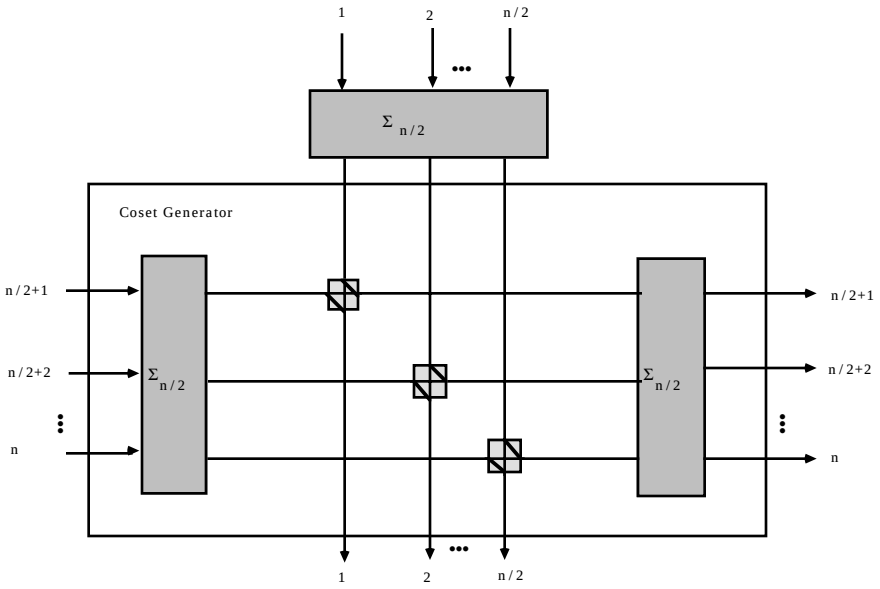


Figure 9: 2×2 switch implementation of the center of a coset network.

can be further decomposed in the same fashion. This provides a uniform decomposition comprising only 2×1 multiplexers, and 1×2 demultiplexers. Furthermore, by direct observation, the number M_n of such multiplexers/demultiplexers satisfies the recurrence

$$M_n = 2n + 3M_{\frac{n}{2}} \tag{16}$$

subject to the boundary condition $M_1 = 0$. Solution of this recurrence yields

$$M_n = 4n^{\log_2 3} = O(n^{1.59}). \tag{17}$$

Since each multiplexer and demultiplexer encompasses two edges, the edge-count C_n of this network is $8n^{\log_2 3}$ which is also $O(n^{1.59})$.

The second implementation of a coset generator directly follows from the first one. Rather than use multiplexers and demultiplexers, we combine them into 2×2 switches as shown in Figure 9. Each switch in the center is obtained by merging together the two multiplexers and two demultiplexers which were originally connected to "would be" the inputs and outputs of that switch in Figure 8. As a result, a total of $n/2$ binary switches are generated in the center, and the Σ_{n-m} subnetworks remain intact.

It is clear that the two implementations are functionally equivalent, and both generate Σ_n. Furthermore, it immediately follows from (17) that the number of 2×2 switches in the recursive version of the latter implementation is $n^{\log_2 3} = O(n^{1.59})$, and its fan-out is two. The only difference between the two networks is in packaging the edges: in the first one, they are contained in multiplexers/demultiplexers while in the latter they are contained in 2×2 switches.

The propagation delay of both networks can be shown to be $O(n)$ by noticing that the propagation delay between horizontal inputs and horizontal outputs, or vertical inputs and horizontal outputs satisfy the recurrence

$$P_n = 1 + 2P_{\frac{n}{2}}; \;\; P_1 = 0 \tag{18}$$

which yields $P_n = n - 1$. Notice that this is the longest propagation delay. The propagation delay betweeen horizontal (or vertical) inputs and vertical outputs in the decomposed network is only $O(\log_2 n)$.

Finally, the setup procedure given in the earlier section can also be used with these two networks after minor modifications. In this case, the setup time obeys the recurrence

$$T_n = \frac{n}{2} + 3T_{\frac{n}{2}} \tag{19}$$

with the boundary condition $T_1 = 0$. The solution of this recurrence gives $T_n = O(n^{1.59})$. It is also possible to implement other 2-stage coset networks using multiplexers and demultiplexers, or elementary switches. Detailing these implementations here is not warrented since the procedure is the same as above except for the size of multiplexers or switches used.

6 Concluding Remarks

The main contribution of this paper has been to point out that connection networks for parallel computers can be designed systematically by a coset decomposition technique outlined in the paper. The major advantage of this technique is that it uncovers various tradeoffs between key network parameters such as edge-count, propagation delay, setup time and fan-out. These tradeoffs are summarized in Theorems 3, 5 and 6. These theorems and the accompanying expressions for the four network parameters provide exact relations by which

the values of the parameters interact. This information is valueable in choosing a network and meeting technological and other implementation constraints in a parallel computing environment.

For example, a complete bipartite graph can be unacceptable because of its large fan-out and high cost when compared to a network with asymptotically optimal edge-count such as those reported in [Benes 1965,Joel 1968,Waksman 1968]. But it is the fastest network and can be set up trivially in $O(n)$ serial or $O(1)$ parallel steps. If the speed is important, but the fan-out of a complete bipartite graph is unacceptable, then one can use a coset network with a single subnetwork decomposition to partially control the fan-out without sacrifying speed. On the other hand, if more restriction is to be exercised on the fan-out, then decompositions with multiple subnetworks can be used at the expense of increasing the setup time to $O(n \log n)$, or $O(n^{1.59})$. Both asymptotically optimal coset networks and the network implementations described in Section 5 achieve constant fan-out with these setup times. Another approach to obtain constant fan-out is to use cellular permutation networks described in [Kautz et al. 1968,Bandyopadhyay et al. 1972,Oruc 1987]. The major disadvantage of this approach is, of course, that the number of edges in such networks grows quadratically with the number of inputs.

An interesting extension of the results given here will be to see if the four network parameters satisfy other relations when a different network construction is used. For example, a network with constant propagation delay and $O(n)$ serial setup time exists (it is the complete bipartite graph). It is also possible to construct a network with constant fan-out and $O(n)$ serial setup time; cellular permutation arrays described in [Kautz et al. 1968,Oruc 1987] satisfy these constraints. But how about a network with constant fan-out, $O(n^{1.59})$ or $O(n \log_2 n)$ edge-count and $O(n)$ serial setup time ?

A more general question is whether it is possible to construct a coset network with constant fan-out, $O(n)$ serial setup time, and edge-count which grows slower than $O(n^2)$. None of the networks described in this paper and those reported in the literature have this feature. If they can be constructed, these networks will combine the attractive fan-out and edge-count features of optimal or near optimal coset networks with the simplicity of the setup procedures of cellular permutation arrays. It is therefore worthwhile to further investigate the existence of such networks to obtain better connection networks for parallel computers.

Acknowledgements

The authors thank the reviewers for their constructive comments and criticisms.

References

[Bandyopadhyay et al. 1972] Bandyopadhyay S., Basu S., and Choudhury K. 1972. A Cellular permuter array. *IEEE Trans. Comp.*, C-21, 10 (Oct.) 1116- 1119.

[Benes 1965] Benes V. E. 1965. *Mathematical Theory of Connecting Networks and Telephone Traffic,* Academic Press, New York, NY.

[Clos 1953] Clos C. 1953. A study of non-blocking swithcing networks. *BSTJ*, 32., 1 (Feb.), 406-425.

[Feng 1981] Feng T. A survey of interconnection networks. *Computer*, 14, 12 (Dec.), 12-27.

[Gecsei 1977] Gecsei J. 1977. Interconnection networks from three-state cells. *IEEE Trans. Comp.*, C-26, 8 (Aug.), 705-711.

[Joel 1968] Joel A. 1968. On permutation switching networks. *BSTJ*, 47, 2 (May-June), 813-822.

[Kautz et al. 1968] Kautz W. H., Levitt K. N., and Waksman A. 1968. Cellular interconnection arrays. *IEEE Trans. Comp.*, C-17, 5 (May), 443-451.

[Lawrie 1975] Lawrie D. K. 1975. Access and alignment of data in an array processor. *IEEE Trans. Comp.*, C-24, 12 (Dec.), 1145-1155.

[Masson et al. 1979] Masson G. M., Gingher G. C., and Nakamura S. 1979. A sampler of circuit switching networks. *Computer*, 12, 6 (June), 32-48.

[Oruc 1987] Oruç A. Y. 1987. Designing cellular permutation networks through coset decomposition of symmetric groups *Journal of Parallel and Distributed Computing*, 3, 3 (Aug.), 402-422.

[Oruc and Oruc 1987] Oruc A. Y., and Oruc M. Y.1987. Programming cellular permutation networks through coset decomposition of symmetric groups. *IEEE Trans. Comp.*, C-36, 7 (July), 802-809.

[Oruc and Thirumalai 1987] Oruç A. Y., and Thirumalai A. 1987. New cellular permutation networks for parallel communications. In *Conference Proceedings– The 21st Conference on Information and Systems Sciences* (Johns Hopkins University, Baltimore, Mar. 25-27), 465-471.

[Parker 80] Parker D. S. 1980. Notes on shuffle-exchange type switching networks. *IEEE Trans. Comp.*, C-29, 3 (Mar.), 213-22.

[Pease 1977] Pease M. C. 1977. The indirect binary n-cube. *IEEE Trans. Comp.*,C-26, 5 (May), 443-473.

[Siegel 1985] Siegel H. J. 1985. *Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies*, Lexington Books, D. C. Heath and Company, Lexington, MA.

[Siegel et al. 87] Siegel H. J., Tsun-Yuk Hsu W., Jeng M. 1987. An introduction to the multistage cube family of interconnection networks *Journal of Supercomputing*, 1, 1 (Jan.), 13-42.

[Thompson 1978] Thompson C. D. 1978. Generalized connection networks for parallel processor intercommunication. *IEEE Trans. Comp.*, C-27, 12 (Dec.), 1119-1125.

[Thurber 1978] Thurber K. J. 1978. Circuit switching technology: A state of the art survey. *In Conference Proceedings– The Compcon 1978 Fall Conference.*, Sept., 1978, 116-124.

[Waksman 1968] Waksman A. 1968. A permutation network. *JACM*, 15, 1 (Jan.), 159-163.

Appendix

Here, we derive some of the expressions given earlier in the paper.

A1. Equation (2) is derived as follows. Let C_n denote the number of edges of a 2-stage coset network constructed as in Figure 3, and let $k = n - m$. We can write

$$C_n = (2k+1)n - (k^2+k) + C_{n-k}$$

where the first two terms account for the number of edges in the coset generator, and the recurring term, i.e., C_{n-k} is the number of edges in the top subnetwork. Now, upon decomposing this subnetwork into a 2-stage coset network with k horizontal inputs, and $n-2k$ vertical inputs, we obtain

$$\begin{aligned} C_n &= (2k+1)n - (k^2+k) + (2k+1)(n-k) - (k^2+k) + C_{n-2k} \\ &= (2k+1)(2n-k) - 2(k^2+k) + C_{n-2k}. \end{aligned}$$

Repeating this process $\alpha - 1$ times, we obtain

$$\begin{aligned} C_n &= (2k+1)n - (k^2+k) + (n-k)(2k+1) - (k^2+k) + (n-2k)(2k+1) - (k^2+k) \\ &+ \cdots + (n-(\alpha-1)k)(2k+1) - (k^2+k) + C_{n-\alpha k}, \\ &= (2k+1)\{\alpha n - (1+2+\ldots+\alpha-1)k\} - \alpha(k^2+k) + C_{n-\alpha k} \\ &= (2k+1)\{\alpha n - \frac{k(\alpha-1)\alpha}{2}\} - \alpha(k^2+k) + C_{n-\alpha k}.|| \end{aligned}$$

A2. Equation (10) is derived as follows. Let C_n be the number of edges of a 2-stage coset network constructed as in Figure 4, and let $k = n - m$ as before. We can write

$$C_n = nk + (n-k)(k+1+\frac{n-k}{\beta}(\beta-1)) + \beta C_{\frac{n-k}{\beta}}$$

where the first two terms account for the number of edges in the coset generator and the recurring term $C_{\frac{n-k}{\beta}}$ is the number of edges in each of the $\Sigma_{m/\beta}$ subnetworks at the top.

Now if each subnetwork is further decomposed into a 2-stage network with k horizontal inputs and $(\frac{n-k}{\beta} - k)/\beta$ vertical inputs, we then have

$$\begin{aligned} C_n &= nk + (n-k)(k+1+\frac{n-k}{\beta}(\beta-1)) + \beta\{(\frac{n-k}{\beta})k \\ &+ (\frac{n-k}{\beta} - k)(k+1+(\frac{n-k}{\beta} - k)\frac{\beta}{(\beta-1)} + \beta C_{n-k(\beta+1))/\beta}\} \end{aligned}$$

Repeating this process $\alpha - 1$ times, we obtain

$$C_n = X_n + Y_n + Z_n + \beta^\alpha C_{\frac{(\beta-1)n-k(\beta^\alpha-1)}{(\beta-1)\beta^\alpha}} \tag{20}$$

where

$$\begin{aligned}
X_n &= k\sum_{i=1}^{\alpha-1} n - k(1+\beta+\beta^2+\ldots+\beta^{i-1}) = kn\alpha - \frac{k^2}{(\beta-1)^2}(\beta^{\alpha_1} - \alpha(\beta-1)) \\
Y_n &= (k+1)\sum_{i=1}^{\alpha-1} n - k(1+\beta+\beta^2+\ldots+\beta^{i-1}) \\
&= (k+1)n\alpha - \frac{k(k+1)}{(\beta-1)^2}(\beta^{\alpha+1} - 1 - (\alpha+1)(\beta-1)) \\
Z_n &= \sum_{i=1}^{\alpha-1}(n - k(1+\beta+\beta^2+\ldots+\beta^{i-1}))^2/\beta^i \\
&= n^2(1-\frac{1}{\beta^\alpha}) - \frac{2nk}{\beta-1}(\alpha - \frac{\beta^\alpha-1}{\beta^\alpha(\beta-1)}) + \frac{k^2}{(\beta-1)^2}(\beta^\alpha-1)(\beta + \frac{1}{\beta^\alpha} - 2\alpha(\beta-1))
\end{aligned}$$

Substituting X_n, Y_n, and Z_n in (20), we obtain Equation (10), i.e.,

$$C_n = (n + \frac{k}{\beta-1})^2(1-\frac{1}{\beta^\alpha}) - \frac{\beta^\alpha-1)}{(\beta-1)^2}(k^2+\beta k) + \alpha(n+\frac{k}{\beta-1}) + \beta^\alpha C_{\frac{(\beta-1)n-(\beta^\alpha-1)k}{(\beta-1)\beta^\alpha}}.||$$