Generalized Shuffle–Exchange Networks
A Brief Summary

Hans Munthe-Kaas

Dept. of Informatics, University of Bergen, N-5020 Bergen Norway

Abstract. A class of generalized Shuffle-Exchange (SE) graphs is introduced. As permutation networks these have the same functionality as the classical SE net (and contains it as a special case), but some of them possess recursive structures lacking in the classical SE net. This allow them to be constructed from identical or a small number of different building blocks and make them very attractive from a hardware–design point of view. For an extensive account of these graphs and their mathematical properties the reader is referred to [11].

Index Terms interconnection networks, permutation networks, perfect shuffle, shuffle–exchange networks, parallel computers, SIMD computers

1 Introduction

In massively parallel SIMD computers, or data parallel computers, the most common form of interprocessor communication is permutations of the data set, i.e. each (virtual) processor sends out one data item and receive one. A natural efficiency measure for SIMD interconnection networks is therefore the number of *routing steps*, i.e. the number of parallel steps needed to do an arbitrary permutation[1]. The optimal routing problem is discused in a series of papers [6, 9, 10, 12, 13, 21].

We define the *cost* of a network as the product of the routing steps and the number of wires in the network, thus the cost is a price/performance measure. It is relatively easy to show that the cost of any interconnection network must satisfy $c(n) = \Omega(n \cdot \log(n))$, where n is the number of nodes in the network. Figures for some networks are summarized in Table (1). The *Shuffle Exchange* (SE) network [19] is among the few networks with optimal cost complexity. Despite of this, the SE network has not been extensively used in hardware design. The reason for this is probably that the SE network is rather complicated to draw in a nice regular fashion; the graph lacks the recursive structure found in e.g. the Boolean cube and other popular networks. Layouts for the SE network has been studied for VLSI design [7], but due to the lack of recursiveness, the layouts are very complicated.

[1] We consider here only static permutations, i.e. permutations that are known in advance, or that are to be performed many times, so that the cost of computing the optimal routing can be neglected. This is relevant for many algorithms and for e.g. measuring the ability of a given network to emulate arbitrary networks.

Network	No. of Routing Steps	No. of Wires	Cost
Complete graph	1	$\Theta(n^2)$	$\Theta(n^2)$
Boolean Cube	$\Theta(\log(n))$	$\Theta(n\log(n))$	$\Theta(n\log^2(n))$
Butterfly	$\Theta(\log(n))$	$\Theta(n\log(n))$	$\Theta(n\log^2(n)$
2–D Mesh	$\Theta(\sqrt{n}))$	$\Theta(n)$	$\Theta(n\sqrt{n})$
Ring	$\Theta(n)$	$\Theta(n)$	$\Theta(n^2)$
Cube Connected Cycles	$\Theta(\log(n))$	$\Theta(n)$	$\Theta(n\log(n))$
(Generalized) SE	$\Theta(\log(n))$	$\Theta(n)$	$\Theta(n\log(n))$

Table 1. Cost complexity for various interconnection networks

2 Definition of Generalized SE Networks

Our basis for searching for generalized versions of the SE net is the following question: *What is the most general form of a network with the same 'permuting functionality' as the (classical) SE net?* In [11] this question is stated in a suitable formal form and it is shown that it is fulfilled if and only if the graph consists of 2^n nodes identified with n-bit binary numbers $g = (g_{n-1}, g_{n-2}, \cdots, g_1, g_0)$; $g_i \in \{0,1\}$, and there exists two permutations of the nodes; $E(g)$ and $S(g)$, such that

$$E(g) = (g_{n-1}, g_{n-2}, \cdots, g_1, \overline{g_0}) \tag{1}$$

and

$$S(g) = (g_{n-2}, g_{n-3}, \cdots, g_0, f(g)) \tag{2}$$

where f is a boolean function satisfying

$$f(g) = f((g_{n-1}, g_{n-2}, \cdots, g_0)) = h((g_{n-2}, g_{n-3}, \cdots, g_0)) \oplus g_{n-1} \ , \tag{3}$$

for some boolean function h.

Note: A map of the form given in (2) is called a *shift register*, and with the additional property in (3) it is called a *non-singular* (or invertible) shift register. A thorough account for the mathematical properties of shift registers is found in [4].

Definition 1. A *Generalized Shuffle Exchange network*, GSE(n, f), is a graph consisting of $N = 2^n$ nodes $\mathcal{G}$, with edges (g, Eg) and (g, Sg) for all $g \in \mathcal{G}$, where E is defined in (1) and S is a non-singular shift register defined in (2)–(3).

If $f(g) = g_{n-1}$ we get the classical SE net. Given a routing algorithm for the classical SE net, it is easy to derive a routing for any GSE networks, thus all algorithms suited for the classical SE net are trivially moved to any GSE nets. This will in practice also include all algorithms suited for e.g. Boolean Cubes and Butterfly networks (see e.g. [8] for details).

In [11] the connection between GSE graphs and the socalled *de Bruijn graph* [2, 14] is elaborated upon. Via this connection it is prooved that for each n there are exactly $\frac{1}{2}(2^{2^{n-1}} + 2^{2^{n-2}})$ different (i.e. non-isomorphic) GSE(n, f) graphs!

Of special importance for computer hardware dezign is the study of *homomorphisms* of graphs. Homomorphisms of a graph $\mathcal{G}$ onto a graph $\mathcal{H}$ are surjective mappings $\phi : \mathcal{G} \rightarrow \mathcal{H}$ such that s.t. $(\phi(g), \phi(g'))$ is an edge of $\mathcal{H}$ if and only if (g, g') is an edge of $\mathcal{G}$. Homomorphisms allow the graph $\mathcal{G}$ to be folded to a simpler form, by collecting nodes into 'supernodes' identified with the nodes of $\mathcal{H}$. In this representation the graph $\mathcal{G}$ has a global structure resembling the graph $\mathcal{H}$, see Figs.(2) and (3). It is shown in [11] that for any graph GSE(m, h) and any $n > m$ there exist a $GSE(n, f)$ and a homomorphism $\phi : GSE(n, f) \rightarrow$ GSE(m, h). GSE(n, f) is called a *lifting* of GSE(m, h), and resembles a process where the number of nodes in the network is increased without changing the global structure. It is furthermore shown that for each n there exist two unique graphs GSE(n, f_n) that can be recursively folded n times, each time halving the size. These graphs, shown in Fig.(1) are called the *maximally foldable GSE graphs* and represents the most regular GSE graphs.

3 Linear GSE networks

Whereas the previous section showed that there are a tremendous number of different GSE graphs for each n, we will in this section restrict our attention to a much smaller class of networks, which can be studied in terms of linear recursion theory.

A *linear homogenous* boolean function is a function

$$f((g_{n-1}, \ldots, g_0)) = \sum_{i=0}^{n-1} c_i \cdot g_{n-1-i} ,$$

where additions and multiplications are modulo 2. A linear inhomogenous boolean function is defined as

$$\bar{f}(g) = f(g) \oplus 1 ,$$

where f is homogenous. A *linear* shift register is a shift register where f is linear (homogenous or inhomogenous), and we define a *linear GSE network* similarly. The mathematical theory for linear shift registers is rich, and a lot is known about their structure (see [4, 17]). The dynamics of a linear shift register is most easily studied by introducing the *characteristic polynomial*[2] defined for a homogenous f as:

$$r(x) = \sum_{i=0}^{n} c_i \cdot x^i , \qquad (4)$$

where $c_n = 1$. For the results in this paper, we do not need to define the characteristic polynomial for the inhomogenous case, and we will use a bar over the characteristic polynomial to indicate that it corresponds to an inhomogenous recursion. The networks are henceforth written in terms of the characteristic

[2] All polynomials are over the binary Galois field $GF[2]$, which means that e.g. $(1 + x) \cdot (1 + x) = 1 + x^2$, since the coefficients are reduced modulo 2.

polynomial as GSE($r(x)$) or GSE($\overline{r(x)}$). The maximally foldable GSE graphs, Fig.(1), are given as GSE($(1+x)^n$) and GSE($\overline{(1+x)^n}$). Of these the latter family is the nicest; it can be shown that its shuffle orbits are all of equal length $p = 2^{\lfloor \log_2(n) \rfloor + 1}$.

The most important result about linear GSE networks is that the study of homomorphisms can be translated into the study of factorizations of polynomials over $GF[2]$. *There exists a homomorphism ϕ : GSE($r(x)$) $\rightarrow$ GSE($s(x)$) if and only if $r(x) = s(x) \cdot t(x)$ for some polynomial $t(x)$.* Thus the graph GSE($s(x) \cdot t(x)$) can be folded into a graph with global structure GSE($s(x)$).

The local structure within each supernode depends on $t(x)$. In general it is possible to arrange the nodes locally such that the local structure in a supernode resembles either the shuffles in GSE($t(x)$) or in GSE($\overline{t(x)}$), thus the network can be buildt by two different building blocks. In the case where $t(x)$ and $s(x)$ are relative prime polynomials it is even possible to arrange the nodes such that *every* supernode has the internal structure of GSE($t(x)$), i.e. the network is buildt from a single building block. These results are summarized in Theorems (2) and (3).

Theorem 2. *Let $r(x) = s(x) \cdot t(x)$ where* $\gcd(s(x), t(x)) = 1$. *Then every node $g_r \in$ GSE($r(x)$) can be uniquely identified with a pair of nodes*

$$g_r = (g_s, g_t) \textit{ where } g_s \in GSE(s(x)) \textit{ and } g_t \in GSE(t(x)),$$

such that shuffle and exchange act according to:

$$E_r(g_r) = (E_s(g_s), E_t(g_t)) \textit{ and } S_r(g_r) = (S_s(g_s), S_t(g_t)) \ .$$

In other words: GSE($r(x)$) is the direct product of GSE($s(x)$) and GSE($t(x)$).

Theorem 3. *Let $r(x) = s(x) \cdot t(x)$. Then every node $g_r \in$ GSE($r(x)$) can be uniquely identified with a pair of nodes $g_r = (g_s, g_t)$ where $g_s \in$ GSE($s(x)$) and $g_t \in$ GSE($t(x)$), such that shuffle and exchange act according to:*

$$E_r(g_r) = (g_s, E_t(g_t)) \tag{5}$$

and

$$S_r(g_r) = \begin{cases} (S_s(g_s), S_t(g_t)) & \textit{if leftmost bit in } g_t = 0 \\ (S_{\bar{s}}(g_s), S_t(g_t)) & \textit{else} \end{cases} \tag{6}$$

4 Concluding Remarks

A major purpose of this work has been to show that SE-type networks are very attractive from a hardware design point of view, and to tie the bonds between parallel computing and the shift register art.

The theory summarized in this paper has several interesting applications that has not been addressed. One is in derivation of routing and mapping algorithms for SE networks. An other is in construction of optimal layouts (in terms of area usage) for VLSI design of SE graphs. Still another is in partitioning SE networks. It is known that the SE graph cannot be partitioned in smaller SE graphs [18]. Theorems (2) and (3) do, however, show that the generalized graphs can be made partitionalble by adding a small number of additional edges. These issues will be addressed in forthcomming papers.

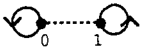
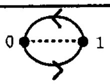
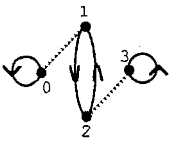
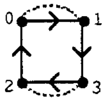
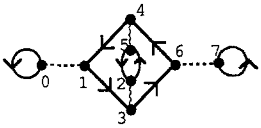
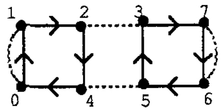
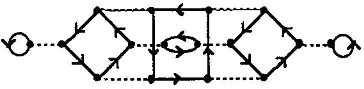
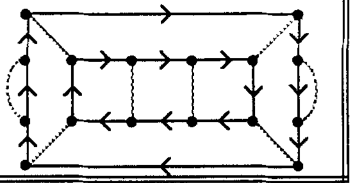
Character. pol.	Homogenous graph	Inhomogenous graph
1+x		
$(1+x)^2 = 1+x^2$		
$(1+x)^3 = 1+x+x^2+x^3$		
$(1+x)^4 = 1+x^4$		

Fig. 1. The two families of maximally fodable GSE graphs.

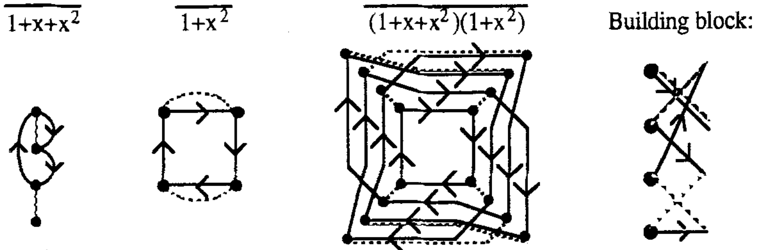


Fig. 2. Direct product of relative prime polynomials, ref. Thm.(3).

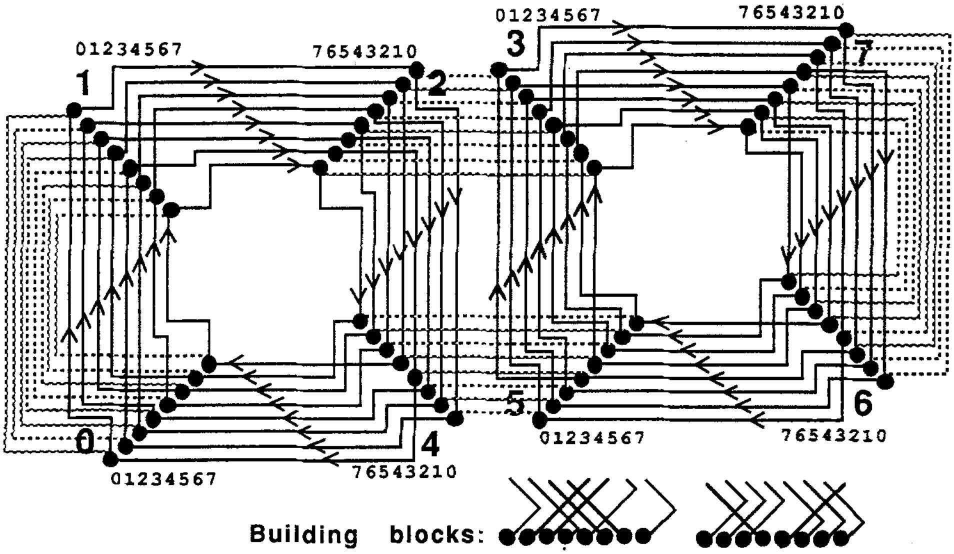


Fig. 3. GSE($\overline{(1+x)^6}$) , where $\overline{(1+x)^6} = \overline{(1+x)^3}\cdot\overline{(1+x)^3}$, ref. Thm.(4).

References

1. Beneš V.E, *Mathematical Theory of Connecting Networks and Telephone Traffic*, New York: Academic Press 1965.
2. Bermond J-C., Peyrat C.: *The deBruijn and Kautz Networks: A Competiton for the Hypercube?*, in Proc. of Hypercube and Distributed Computers, 1989, pp. 279-291.
3. Fredricksen H.: *Lechtures on Recursion*, unpublished lechture notes, University of Bergen, 1970.
4. Golomb S.W.: *Shift Register Sequences*, Holden-Day, San Francisco 1967 (224 pp.).
5. Guckenheimer J.: *A Brief Introduction to Dynamical Systems*, Lectures in Appl. Math. Vol. 17, pp.187-253, 1979.
6. Huang S-T, Tripathi S.K.: *Self-Routing Technique in Perfect-Shuffle Networks Using Control Tags*, IEEE Trans. Comput., vol. 37, pp. 251-256, no. 2 Feb. 1988.
7. Kleitman A., Leighton F.T., Lepley M., Miller G.L.: *An Asymptotically Optimal Layout for the Shuffle-Exchange Graph*, J. of Computer and Systems Sciences 26, 339-361 (1983).
8. Leighton F.T.: *Introduction to Parallel Algorithms and Architectures*, Morgan Kaufman Publ., San Mateo, Calif. 1992. (831 pp).
9. Lenfant J.: *Parallel Permutations of Data: A Benes Network Control Algorithm for Frequently Used Permutations*, IEEE Trans. Comput., vol. C-27,pp. 637-647, no. 7 July 1978.
10. Munthe-Kaas H.: *Practical Parallel Permutation Procedures*, to appear.
11. Munthe-Kaas H.: *Generalized Shuffle Exchange Networks*, Dept. of Informatics, Univ. of Bergen, Rep. no. 61, June 1992, ISSN 0333-3590.
12. Nassimi D., Sahni S.: *A Self-Routing Benes Network and Parallel Permutation Algorithms*, IEEE Trans. Comput., vol. C-30,pp. 332-340, no. 5 May 1981.
13. Parker S.D.: *Notes on Shuffle/Exchange-Type Switching Networks*, IEEE Trans. Comput., vol. C-29,pp. 213-222, no. 3 March 1980.
14. Pradhan D.K., Samtham M.R.: The deBruijn multiprocessor network: A versatile parallel processing and sorting network for VLSI, IEEE trans. on comput. 38 (1989).
15. Pradhan D.K., Kodandapani K.L: *A uniform Representation of Single and Multistage Interconnection Networks Used in SIMD Machines*, IEEE trans. on Comp., vol. C-29, no. 9, 1980.
16. Preparata F.P., Vuillemin J.: *The Cube-Connected Cycles: A versatile network for parallel computation*, Comm. ACM vol. 24 no. 5, 1981.
17. Selmer E.S.:*Linear Rercurrence Relations over Finite Fields*, (mimeograph) Dept. of Mathematics University of Bergen, Norway, 1966, (212 pp.).
18. Siegel H.J.: *The Theory Underlying the Partitioning of Permutation Networks*, IEEE Trans. on Computers, vol. C-29, No. 9, 1980.
19. Stone H.S.: *Parallel Processing with the Perfect Shuffle*, IEEE trans. on comput. C-20, 2, (Feb 1971).
20. Ward M.: *The Arithmetical Theory of Linear Recurring Series* Trans. Amer. Math. Soc. 33 (1931), pp. 153-165.
21. Wu C., Feng T.: *The Universality of the Shuffle-Exchange Network*, IEEE Trans. Comput., vol. C-30, pp. 324-331, May 1981.

This article was processed using the LATEX macro package with LLNCS style