MULTI-PORT INTERCONNECTION NETWORKS FOR RADIX-R ALGORITHMS

Jarmo H. Takala[1], *Tuomas S. Järvinen*[1], *Perttu V. Salmela*[1], *and David A. Akopian*[2]

[1]Digital and Computer Systems Laboratory
Tampere University of Technology
P.O.B. 553, 33101 Tampere, Finland

[2]Nokia Mobile Phones
P.O.B. 429,
33101 Tampere, Finland

ABSTRACT

In array processors, complex data reordering is often needed to realize the interconnection topologies between the computational nodes in algorithms. Several important algorithms, e.g., discrete trigonometric transforms and Viterbi decoding, can be represented in a radix-R form where the principal topology is stride by R permutation. In this paper, a general factorialization of stride permutations is derived, which can be mapped onto register-based structures for constructing area-efficient multi-port interconnection networks. The networks can be modified to support several stride permutations and sequence sizes.

1. INTRODUCTION

Several high-speed architectures for digital signal processing algorithms are based on array processor concept and, in particular, pipelined architectures have gained popularity. E.g., Cooley-Tukey radix-2^r fast Fourier transform (FFT) algorithms have been realized with cascaded processing elements computing radix-2^r butterfly operation. In such a parallel architecture, reordering of data sequences between the processing elements, due to the data-dependencies in the algorithm, is essential. In the derivation of Cooley-Tukey 2^k-point radix-2 decimation-in-frequency FFT algorithms, the principal method is to interleave results of two 2^{k-1}-point FFTs, i.e., the elements of the output vector $(a_0, a_1, \ldots, a_{2^{k-1}}, b_0, b_1, \ldots, b_{2^{k-1}})^T$ are reordered to obtain the input vector $(a_0, b_0, a_1, b_1, \ldots, a_{2^{k-1}}, b_{2^{k-1}})^T$. This permutation is known as perfect shuffle permutation. In general, interconnection topologies in Cooley-Tukey radix-2^r FFT algorithms are based on stride by 2^r permutations [1]. Stride by R permutation of a K-point sequence reorders the elements of a vector $X = (x_0, x_1, \ldots, x_{K-1})$ as $Y = (x_0, x_R, x_{2R}, \ldots, x_{K-R+1}, x_1, x_{R+1} \ldots, x_{K-1})^T$. Therefore, perfect shuffle is a special case of stride permutations, i.e., stride by $K/2$ of a K-point sequence. Such interconnection topologies can also be found in some fast algorithms for other discrete trigonometric transforms, e.g., for discrete cosine, sine, and Hartley transforms [2, 3, 4]. The same topology is also present in trellis diagrams in k/n-code rate convolutional encoders [5]. Furthermore, the stride permutation according to the definition in [1] can be used in matrix transposition, i.e., transposition of a $K \times K$-matrix is stride by K permutation of a sequence where the columns of the matrix are arranged in a vector form.

In radix-2^r algorithms, the computation of basic operational node requires 2^r operands implying need for a multi-port reordering unit. Such an unit can be realized with multi-port memories where the results of the previous node are written through the first port at the same time as the operands for the next node are read through the second port. However, the parallel access introduces resource conflicts and, therefore, memory-based reordering units are realized with parallel memory modules. Such an realization is proposed in [6] for radix-2 FFT allowing conflict-free access of two input and output operands. In this scheme, a specialized address generator provides separate addresses for read and write operations for two dual-port memory modules. The principal problem of memory-based reordering units is the need for expensive multi-port memories for supporting simultaneous read and write operations. The additional complexity due to specialized address generation and multiple ports is a disadvantage especially when the sequence sizes are small.

Area-efficient reordering units can be realized with register-based structures, especially for smaller sequence sizes. The traditional approach used in cascaded FFT architectures is a multi-port commutator, which consists of skewed delay lines and a multi-port switch performing complex periodic switching patterns. Such an approach is used, e.g., in the FFT architecture proposed in [7]. The drawback of this approach is the complexity of the switching element. In [8], a Viterbi decoder architecture is described where stride permutations are realized with the aid of parallel tapped first-in, first-out (FIFO) buffers; several data elements are written into consecutive FIFO locations, shift of several elements is performed, and data elements are read from tapped outputs. Such an approach requires complex write, shift, and read schemes and the FIFOs need to operate at higher frequency than the sample clock.

In this paper, a factorialization of stride permutations, based on the results described earlier in [9], is derived. The factorializations can be efficiently mapped onto register-based structures, which can be used to construct interconnection networks for array processors supporting different strides and sequence sizes.

2. PRELIMINARIES

In this paper, left evaluation is used for ordinary products, i.e.,

$$\prod_{i=0}^{n} a_i = a_0 \cdot a_1 \cdot a_2 \cdot \ldots \cdot a_n \tag{1}$$

The formulation used here is based on tensor products; tensor product (or Kronecker product) and direct sum are denoted by $\otimes$ and $\oplus$, respectively. The stride by R permutation matrix of order K is a square matrix $P_{K,R}$ where the elements of the matrix are defined as

$$P_{K,R}(m,n) = \begin{cases} 1, \text{iff } n = (mR \bmod K) + \lfloor mR/K \rfloor \\ 0, \text{otherwise} \end{cases} \tag{2}$$

where $\bmod$ is the modulus operation and $\lfloor \cdot \rfloor$ is the floor function.

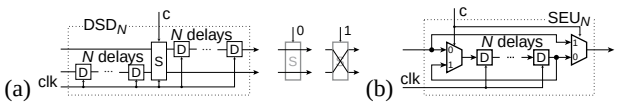

Fig. 1. Basic switching units: (a) delay-switch-delay unit and (b) shift-exchange unit. D: Delay register. clk: Sample clock signal. c: Control signal. S: 2-to-2 switch.

In the following, some theorems on stride permutations are given. The proofs for the theorems can be found, e.g., from [1, 10].

Theorem 1 *Factorializations of stride permutations:*

$$P_{a,bc} = P_{a,b}P_{a,c} \quad (3)$$

$$P_{abc,c} = (P_{ac,c} \otimes I_b)(I_a \otimes P_{bc,c}) \quad (4)$$

where I_K denotes the identity matrix of order K.

Theorem 2 *Relationship between tensor product and stride permutation. If A_a and B_b are matrices of order a and b, respectively, then,*

$$A_a \otimes B_b = P_{ab,a}(B_b \otimes A_a)P_{ab,b} \quad (5)$$

Corollary 1 *Stride by two permutation of a 2^k-point sequence can be decomposed as consecutive $P_{4,2}$ permutations as*

$$P_{2^k,2} = \prod_{i=0}^{k-2} I_{2^i} \otimes P_{4,2} \otimes I_{2^{k-i-2}} \quad (6)$$

Proof. According to (4) stride by 2 permutation can be decomposed as

$$\begin{aligned} P_{K,2} &= (P_{K/2,2} \otimes I_2)(I_{K/4} \otimes P_{4,2}) \\ &= (P_{K/4,2} \otimes I_4)(I_{K/8} \otimes P_{4,2} \otimes I_2)(I_{K/4} \otimes P_{4,2}) \\ &= (P_{4,2} \otimes I_{K/4}) \ldots (I_{K/16} \otimes P_{4,2} \otimes I_4) \cdot \\ &\quad (I_{K/8} \otimes P_{4,2} \otimes I_2)(I_{K/4} \otimes P_{4,2}) \end{aligned}$$

Finally, a special permutation matrix J_K of order K is defined:

$$J_K = \left(I_2 \otimes P_{K/2,K/4}\right) P_{K,2} \quad (7)$$

This permutation exchanges the odd elements in the first half of a vector with the even elements of the last half of the vector.

3. BASIC SWITCHING UNITS

The basic switching units used in this paper, the 2-port delay-switch-delay unit of size N (DSD$_N$) and 1-port shift-exchange unit [11] of size N (SEU$_N$), are illustrated in Fig. 1. In [9], we showed that a Q-port network consisting of $Q/2$ DSD$_N$ units in parallel performs the reordering defined by J_{QN}, $Q = 2^q, N = 2^n$. In addition, all the reorderings of type $(I_M \otimes J_{QN})$ can be mapped onto the same Q-port structure independent on M, $M = 2^m$. As an example, a 2-port realization of J_{16} permutation with the corresponding timing diagram is illustrated in Fig. 2(a).

In principle, SEU$_N$ can exchange data elements N apart in a sequential data stream. This unit can be used to construct a Q-port unit to perform reorderings of type $(I_M \otimes P_{4,2} \otimes I_{QN})$. E.g., a Q-port network constructed with Q parallel SEU$_1$ units can perform $(P_{4,2} \otimes I_Q)$ since the $4Q$-point input sequence is divided into four Q-point subsequences entering the reordering unit at consecutive cycles and the permutation is performed by exchanging the middle subsequences. In the same manner, $(P_{4,2} \otimes I_{QN})$ over Q ports can be performed with a network where Q SEU$_N$ units operate in parallel, i.e., Q-point subsequences to be exchanged are N cycles apart. Permutations of type $(I_M \otimes P_{4,2} \otimes I_{QN})$ can also be mapped onto the same Q-port structure. As an example, a 2-port realization of $(P_{4,2} \otimes I_4)$ with corresponding timing diagram is illustrated in Fig. 2(b). The latency of the basic switching units is the same depending on the number of delay registers connected to a port; the latency of DSD$_K$ and SEU$_K$ is K cycles.

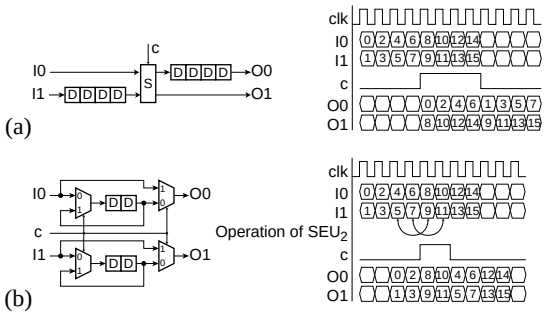

Fig. 2. Block and timing diagrams of 2-port structures performing: (a) J_{16} and (b) $(P_{4,2} \otimes I_4)$. Clock signals are omitted for clarity.

4. FACTORIALIZATION OF STRIDE PERMUTATIONS

In general, the reordering units for radix-2^r algorithms based on stride by 2^r permutation contain 2^r ports. This reflects the natural number of operations of the basic operation node, e.g., radix-2^r butterfly in Cooley-Tukey FFT. For such cases, in [9] we have shown a simple factorialization for the stride permutations:

$$P_{K,R} = (I_2 \otimes P_{K/2,R})J_K(I_{K/R} \otimes P_{R,R/2}) \quad (8)$$

where $R = 2^r$, $K = 2^k$, and J_K is defined in (7). By recursively applying this factorialization, the stride by 2^r permutation of order 2^k as a function of the number of ports 2^q can be defined as ($r \leq q$)

$$\begin{aligned} P_{2^k,2^r}(2^q) &= I_{2^{k-q}} \otimes P_{2^q,2^r} \cdot \\ &\left[\prod_{i=0}^{k-q-1} (I_{2^{k-q-i-1}} \otimes J_{2^{q+i+1}})(I_{2^{k-r}} \otimes P_{2^r,2^{r-1}})\right] \end{aligned} \quad (9)$$

This factorialization can be used to design simple register-based interconnection networks or reordering units for the the previous case; radix-2^r operation nodes connected to 2^r-port reordering units. However, the operation node can be mapped onto reduced number of computational resources, thus there is need to support stride by 2^r permutation over less number of ports, $2^q, q < r$.

In order to efficiently realize permutations with higher stride, we need to derive a factorialization where the order of the stride is decreased. According to (3) and (4), the stride permutations can

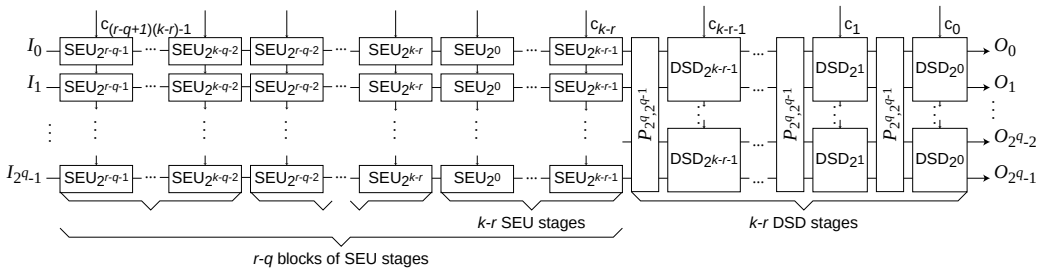

Fig. 3. General block diagram of 2^q-port stride by 2^r permutation network for 2^k-point sequences. Blocks labeled by $P_{2^q,2^{q-1}}$ are hard-wired perfect shuffle permutations.

be factorialized as, $K = 2^k$, $R = 2^r$,

$$P_{K,R} = P_{K,2} P_{K,R/2} \quad (10)$$

$$P_{K,R/2} = (P_{K/2,R/2} \otimes I_2)(I_{K/R} \otimes P_{R,R/2}) \quad (11)$$

By combining the previous equations and utilizing the fact that $P_{K,K/2} P_{K,2} = I_K$, we may formulate factorialization of stride by R permutation of order K as

$$P_{K,R} = (I_2 \otimes P_{K/2,R/2}) P_{K,2} (I_{K/R} \otimes P_{R,R/2}) \quad (12)$$

By applying the relation in (4) to the term $P_{K,2}$, we may rewrite the permutation as

$$P_{K,R} = \left(I_2 \otimes P_{K/2,R/2}\right) \left(P_{2K/R,2} \otimes I_{R/2}\right) \cdot \left(I_{K/R} \otimes P_{R,2}\right) \left(I_{K/R} \otimes P_{R,R/2}\right) \quad (13)$$

By noting that $\left(I_{K/R} \otimes P_{R,2}\right) \left(I_{K/R} \otimes P_{R,R/2}\right) = I_K$, we obtain a new factorialization as

$$P_{K,R} = \left(I_2 \otimes P_{K/2,R/2}\right) \left(P_{2K/R,2} \otimes I_{R/2}\right) \quad (14)$$

This factorialization can be recursively applied to the leftmost term in (14), which results in a structure with reduced order of stride. This implies that the recursion can be continued until the stride of the permutation matches the required number of ports. Therefore, the stride by 2^r permutation of order 2^k as a function of the number of ports 2^q can be defined as ($r > q$)

$$P_{2^k,2^r}(2^q) = I_{2^{r-q}} \otimes P_{2^{k+q-r},2^q} \cdot \prod_{i=0}^{r-q-1} I_{2^{r-q-1-i}} \otimes P_{2^{k-r+1},2} \otimes I_{2^{i+q}} \quad (15)$$

By applying the corollary I, (15) can be rewritten as

$$P_{2^k,2^r}(2^q) = I_{2^{r-q}} \otimes P_{2^{k+q-r},2^q} \cdot \prod_{i=0}^{r-q-1} I_{2^{r-q-i-1}} \otimes \left[\prod_{j=0}^{k-r-1} I_{2^j} \otimes P_{4,2} \otimes I_{2^{k-r+q+i-j-1}} \right] \quad (16)$$

5. REALIZATION

Based on the previous discussion, we may derive interconnection network structures in two different cases depending on the relation of the stride R and the required number of ports of the network Q.

Case I: $R \leq Q$. In this case, the factorialization in (9) can efficiently be mapped onto a network structure containing only DSD units as described in [9]. Latency of a 2^q-port network performing stride permutation of a 2^k-point sequence is $2^{k-q} - 1$, while the number of subsequences entering the network over 2^q ports is 2^{k-q}. Therefore, the latency is less than the total number of subsequences implying high utilization of the resources and efficiency, i.e., permutation is performed with less number of delay registers than actual data elements in the sequence.

As an example, we may consider a 16-point stride by 2 permutation over two ports, $q = 1, r = 1, k = 4$. By noting that $P_{2,2} = P_{2,1} = I_2$, the following decomposition is obtained

$$P_{16,2} = (I_4 \otimes J_4)\,(I_2 \otimes J_8)\,J_{16} \quad (17)$$

This can be realized with a cascade of DSD_4 for J_{16}, DSD_2 for J_8, and DSD_1 for J_4.

Case II: $R > Q$. In this case, the factorialization in (16) is needed to decrease the stride. For the first term in (16), $(I_{2^{r-q}} \otimes P_{2^{k+q-r},2^q})$, we can apply the factorialization in (9) since the stride is the same as the number of ports, and obtain a network consisting of DSD units. The remaining terms describe parallel $P_{4,2}$ permutations of data in blocks, which have larger size than the number of ports. Such permutations can be realized with cascades of parallel SEU units as discussed earlier. The general block diagram of the resulting interconnection network is depicted in Fig. 3. It should be noted that the last $k-r$ DSD stages are due to the factorialization in (9). The latency of a 2^q-port network performing stride by 2^r permutation on a 2^k-point sequence is $(2^{k-q} - 2^{r-q})$, which is again less than the number of subsequences entering the network, 2^{k-q}.

As an example, we may consider the factorialization of $P_{32,4}$ over two ports, which according to (16) is

$$P_{32,4} = (I_2 \otimes P_{16,2})(P_{4,2} \otimes I_8)(I_2 \otimes P_{4,2} \otimes I_4)(I_4 \otimes P_{4,2} \otimes I_2)$$

where the first term can be factorialized as shown in (17). The resulting signal flow graph and corresponding implementation is illustrated in Fig. 4(a). The same permutation over four ports is depicted in Fig. 4(b).

In some cases, there may be a need to support permutations with different strides, e.g., computation of a 2-D transform with the row-column decomposition requiring matrix transposition. In such a case, the reordering unit needs to support stride by 2^r permutation for interconnections in 1-D transform and stride by 2^k permutation for performing the $2^k \times 2^k$ matrix transposition. Such networks can be designed by developing network structures for

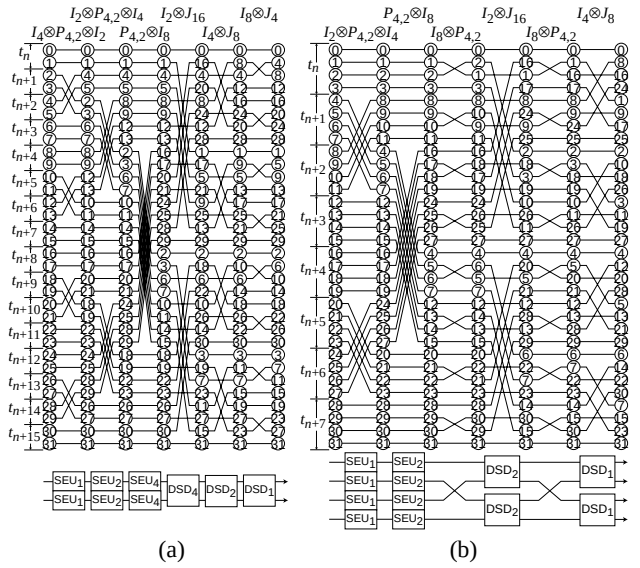

Fig. 4. Factorialization of 32-point stride permutations with corresponding implementations: (a) stride by 4 over two ports and (b) stride by 8 over four ports. t_i: ith time instant.

each stride and then mapping these onto a unified network, where additional paths are arranged with the aid of multiplexers.

As an example, let us consider a unified 4-port network for stride by 4 and 8 permutations for 32-point sequences. $P_{32,4}$ can be realized directly by utilizing the factorialization in (9) since the stride equals to the number of ports. The resulting network is illustrated in Fig. 5(a). For $P_{32,8}$ we need to apply (16), which results in the network in Fig. 5(b). By comparing the networks, we find that they contain several similarities, only the first switching stages are different. These can be unified by connecting the SEU stages as skewed delay registers in the DSD unit. However, the SEU units in a single port contain only three registers, thus an additional delay register for two ports is required to realize the DSD_4 unit. In addition, a single 2-to-2 switch is needed to realize the $P_{4,2}$ permutation needed in front of DSD_4 units in Fig. 5(a). The unified network is illustrated in Fig. 5(c).

6. SUMMARY

The general factorializations of stride by 2^r permutations, shown in this paper, allow mapping of stride permutations onto simple register-based structures. The resulting multi-port networks contain less registers than data elements in the sequence implying area-efficiency. The described networks can be used as reordering units in array processor architectures, e.g., pipeline, cascade, or partial column architectures, for realizing radix-2^r algorithms where the interconnection topologies are based on stride by 2^r permutations. Such algorithms exist, e.g., for discrete Fourier, sine, cosine, and Hartley transforms. The same topologies are found in Viterbi algorithm and matrix transposition. The proposed networks can be modified to support several stride permutations. In addition, support for several sequence sizes is possible with the aid of additional routing multiplexers as described in [9].

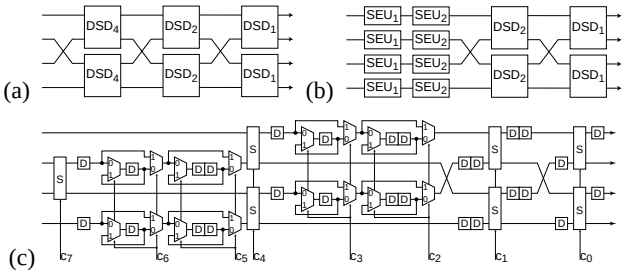

Fig. 5. 4-port interconnection networks for 32-point sequences: (a) stride by 4 permutation, (b) stride by 8 permutation, and (c) unified network supporting both the previous permutations.

7. REFERENCES

[1] J. Granata, M. Conner, and R. Tolimieri, "Recursive fast algorithms and the role of the tensor product," *IEEE Trans. Signal Processing*, vol. 40, no. 12, pp. 2921–2930, Dec. 1992.

[2] J. Astola and D. Akopian, "Architecture-oriented regular algorithms for discrete sine and cosine transforms," *IEEE Trans. Signal Processing*, vol. 47, no. 4, pp. 1109–1124, Apr. 1999.

[3] J. Kwak and J. You, "One- and two-dimensional constant geometry fast cosine transform algorithms and architectures," *IEEE Trans. Signal Processing*, vol. 47, no. 7, pp. 2023–2034, July 1999.

[4] J. Takala, D. Akopian, J. Astola, and J. Saarinen, "Constant geometry algorithm for discrete cosine transform," *IEEE Trans. Signal Processing*, vol. 48, no. 6, pp. 1840–1843, June 2000.

[5] D. Akopian, J. Takala, J. Astola, and J. Saarinen, "Multistage interconnection networks for k/n rate Viterbi decoders," in *Proc. IEEE Global Telecommun. Conf.*, Sydney, Australia, Nov. 8–12 1998, pp. 845–850.

[6] Y. Ma, "An effective memory addressing scheme for FFT processors," *IEEE Trans. Signal Processing*, vol. 47, no. 3, pp. 907–911, Mar. 1999.

[7] E. Bidet, D. Castelain, C. Joanblanq, and P. Senn, "A fast single-chip implementation of 8192 complex point FFT," *IEEE J. Solid-State Circuits*, vol. 30, no. 3, pp. 300–305, Mar. 1995.

[8] M. Bóo, F. Argüello, J. Bruguera, R. Doallo, and E. Zapata, "Mapping of trellises associated with general encoders onto high-performance VLSI architectures," *J. VLSI Signal Processing*, vol. 17, no. 1, pp. 57–73, Sept. 1997.

[9] J. Takala, D. Akopian, J. Astola, and J. Saarinen, "Scalable interconnection networks for partial column array processor architectures," in *Proc. IEEE ISCAS*, Geneva, Switzerland, May 28–31 2000, vol. IV, pp. 513–516.

[10] M. Davio, "Kronecker products and shuffle algebra," *IEEE Trans. Comput.*, vol. 30, no. 2, pp. 116–125, Feb. 1981.

[11] C. B. Shung, H.-D. Lin, R. Cypher, P. H. Siegel, and H. K. Thapar, "Area-efficient architectures for the Viterbi algorithm I. Theory," *IEEE Trans. Commun.*, vol. 41, no. 4, pp. 636–644, Apr. 1993.