Optimal Streamed Linear Permutations

(Invited Paper)

François Serre
Department of Computer Science
ETH Zurich
serref@inf.ethz.ch

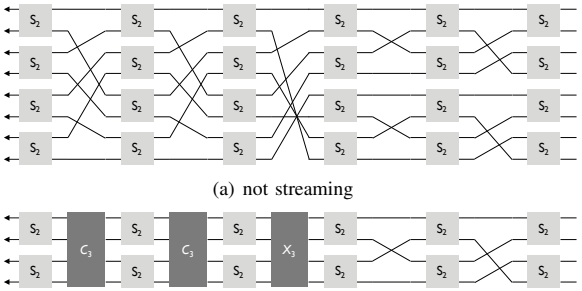
Markus Püschel
Department of Computer Science
ETH Zurich
pueschel@inf.ethz.ch

I. Introduction

A fully parallel hardware implementation of algorithms on large data sets is usually impossible due to the resources it requires. Therefore, the corresponding datapaths need to be *folded* into a streaming architecture, which accepts the input over several cycles. Particularly suited for folding are regular algorithms such as the fast Fourier transform [1], Viterbi decoding, or sorting networks [2]. An example of the latter for $2^3 = 8$ elements is shown in Fig. 1(a) and an associated folded version with streaming width $2^k = 4$ in Fig. 1(b). The folded version halves the number of two input sorters S_2 needed for its implementation [3].

Some permutations are trivial to fold due to their spatial periodicity (e.g., the two rightmost permutations in Fig.1(a)). However, in general, implementing a *streaming permutation* is challenging, as it requires both routing between ports and delays across cycles. In this short paper we overview our work on deriving optimal streaming circuits for the subclass of *linear permutations* (that we define), which include those needed in the applications mentioned above. We assume only 2-by-2 switches and dual-ported RAMs as building blocks, which is well-suited for FPGAs.

Methods to design streaming circuits for arbitrary permutations exist [5], [6], but are thus more costly for linear permutations than our method. Optimal register-based implementations for the specific class of stride permutations and bit reversal (which are linear) have been designed in [7].



(a) not streaming

(b) streaming with streaming width $2^k = 4$

Fig. 1. Dataflow (right to left) of a sorting network for 2^3 elements (from [4]).

II. Linear permutations

We consider permutations π on 2^n elements with indices $\{0, 1, \ldots, 2^n - 1\}$. A permutation π is *linear* [8], if it can be expressed as a linear mapping of the bit representation of the element indices. In other words, there exists an invertible $n \times n$ bit matrix P (or, mathematically, $P \in \mathrm{GL}_n(\mathbb{F}_2)$), such that for $0 \le i < 2^n$,

$$\pi(i) = j \Leftrightarrow j_b = P \cdot i_b, \tag{1}$$

where i_b, j_b are the bit representations of i, j, respectively (as column vectors). The most significant bit is at the top. In the case of (1) we write $\pi = \pi(P)$.

As an example, writing in binary the indices for the central permutation in Fig. 1(a) yields

$$\begin{pmatrix}0\\0\\0\end{pmatrix} \mapsto \begin{pmatrix}0\\0\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix} \mapsto \begin{pmatrix}0\\1\\0\end{pmatrix} \mapsto \begin{pmatrix}1\\0\\0\end{pmatrix} \mapsto \begin{pmatrix}1\\1\\1\end{pmatrix} \mapsto \begin{pmatrix}0\\0\\1\end{pmatrix},$$
$$\begin{pmatrix}0\\1\\1\end{pmatrix} \mapsto \begin{pmatrix}1\\1\\0\end{pmatrix} \mapsto \begin{pmatrix}0\\1\\1\end{pmatrix}, \text{ and } \begin{pmatrix}1\\0\\1\end{pmatrix} \mapsto \begin{pmatrix}1\\0\\1\end{pmatrix}.$$

This permutation is linear with associated matrix

$$P = \begin{pmatrix}1 & 1 & \\ 1 & & 1 \\ 1 & & \end{pmatrix}.$$

A special class of linear permutations are bit-index permutations, meaning that P itself is again a permutation. These include the stride permutations and the bit reversal.

Our goal is to implement a given $\pi = \pi(P)$ on 2^n data with streaming width 2^k, $k \le n$, using only 2-by-2 switches and dual-ported RAMs. We solve the problem by considering only P. To do so, we first block P according to k:

$$P = \begin{pmatrix}P_4 & P_3 \\ P_2 & P_1\end{pmatrix}, \text{ with } P_1 \text{ of size } k \times k. \tag{2}$$

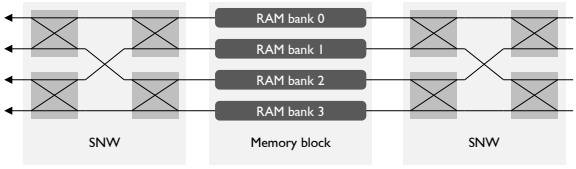
Under these assumptions, the following lower bound holds [4]:

Theorem 1. *A full-throughput streaming implementation for a linear permutation $\pi(P)$ with 2^k ports that only uses 2×2-switches for routing requires at least $\mathrm{rk}\, P_2 \cdot 2^{k-1}$ many switches ($\mathrm{rk}\, P_2$ = rank of P_2).*

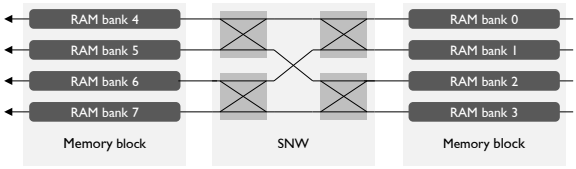
If P is of the special form

$$P = \begin{pmatrix}I_t & \\ P_2 & P_1\end{pmatrix}, \tag{3}$$

1063-6889/17 $31.00 © 2017 IEEE
DOI 10.1109/ARITH.2017.13



(a) SNW/RAM/SNW



(b) RAM/SNW/RAM

Fig. 2. Two possible architectures for a streaming permutation (from [4]).

then $\pi(P)$ is *spatial* [9]: it permutes only across ports and no memory is required. [10] gives a method to implement it using a switching network (SNW) with the minimal number of $\operatorname{rk} P_2 \cdot 2^{k-1}$ switches.

If P is of the special form

$$P = \begin{pmatrix} P_4 & P_3 \\ & I_k \end{pmatrix}, \tag{4}$$

then $\pi(P)$ is *temporal* [9]: it permutes only across cycles, and can be implemented using an array of 2^k banks of RAM with a capacity of at most 2^{n-k} elements per bank.

III. Implementing a general linear permutation

We build optimal circuits for linear permutation by factorizing P into spatial (3) and temporal (4) matrices using the property $\pi(P) \circ \pi(Q) = \pi(PQ)$ [9], [10], [4]. Three such matrices always suffice; thus there are two choices depicted in Fig. 2: SNW/RAM/SNW and RAM/SNW/RAM.

The first choice corresponds to the factorization

$$P = \begin{pmatrix} I_t & \\ L_2 & L_1 \end{pmatrix} \begin{pmatrix} M_4 & M_3 \\ & I_k \end{pmatrix} \begin{pmatrix} I_t & \\ R_2 & R_1 \end{pmatrix}, \tag{5}$$

which requires $(\operatorname{rk} L_2 + \operatorname{rk} R_2) \cdot 2^{k-1}$ switches (Theorem 1) and has to be minimized. [11] provides the following lower bound and an algorithm to construct the associated factorization:

Theorem 2. *Given P as in* (2)*, then any factorization* (5) *satisfies*

$$\operatorname{rk} L_2 + \operatorname{rk} R_2 \geq \max(\operatorname{rk} P_2, n - \operatorname{rk} P_4 - \operatorname{rk} P_1).$$

Thus, if $\operatorname{rk} P_2 \geq n - \operatorname{rk} P_4 - \operatorname{rk} P_1$ the algorithm [11] achieves the lower bound in Theorem 1. In the opposite case the second choice of factorization (RAM/SNW/RAM)

$$P = \begin{pmatrix} L_4 & L_3 \\ & I_k \end{pmatrix} \begin{pmatrix} I_t & \\ M_2 & M_1 \end{pmatrix} \begin{pmatrix} R_4 & R_3 \\ & I_k \end{pmatrix}. \tag{6}$$

achieves the minimum number of switches, but at the price of twice the RAM. In this case the following theorem yields optimality. The factorization is again obtained with [11]:

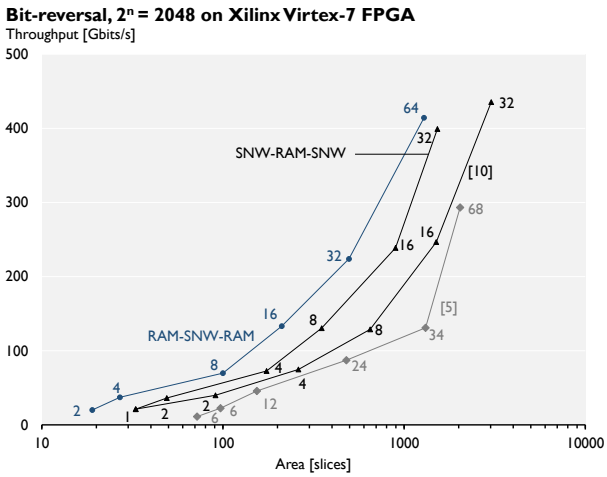


Fig. 3. Comparison of our two structures for a bit-reversal permutation on 2048 16-bit elements for different multiplexer sizes vs. [5] and [10]. Labels: number of BRAM tiles.

Theorem 3. *Given P as in* (2)*, then any factorization* (6) *satisfies*

$$\operatorname{rk} M_2 = \operatorname{rk} P_2.$$

IV. Results

As a proof of concept, we compare in Fig. 3 our two structures on an FPGA for a bit-reversal against the general method proposed in [5]. For the same throughput, we observe a significantly reduced area consumption, even more so if additional RAM is invested.

References

[1] M. C. Pease, "An adaptation of the fast fourier transform for parallel processing," *Journal of the ACM*, vol. 15, no. 2, pp. 252–264, Apr. 1968.

[2] D. E. Knuth, *The Art of Computer Programming, 2Nd Ed. (Addison-Wesley Series in Computer Science and Information*, 2nd ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1978.

[3] M. Zuluaga, P. A. Milder, and M. Püschel, "Streaming sorting networks," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 21, no. 4, 2016.

[4] F. Serre, T. Holenstein, and M. Püschel, "Optimal circuits for streamed linear permutations using RAM," in *International Symposium on Field-Programmable Gate Arrays (FPGA)*. ACM, 2016, pp. 215–223.

[5] P. A. Milder, J. C. Hoe, and M. Püschel, "Automatic generation of streaming datapaths for arbitrary fixed permutations," in *Design, Automation and Test in Europe (DATE)*, 2009, pp. 1118–1123.

[6] R. Chen, S. Siriyal, and V. Prasanna, "Energy and memory efficient mapping of bitonic sorting on FPGA," in *International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2015, pp. 240–249.

[7] T. Järvinen, P. Salmela, H. Sorokin, and J. Takala, "Stride permutation networks for array processors," in *Application-Specific Systems, Architectures and Processors Proceedings (ASAP)*, 2004, pp. 376–386.

[8] J. Lenfant and S. Tahé, "Permuting data with the Omega network," *Acta Informatica*, vol. 21, no. 6, pp. 629–641, 1985.

[9] K. J. Page and P. M. Chau, "Folding large regular computational graphs onto smaller processor arrays," in *Proc. SPIE*, vol. 2846, 1996, pp. 383–394.

[10] M. Püschel, P. A. Milder, and J. C. Hoe, "Permuting streaming data using RAMs," *Journal of the ACM*, vol. 56, no. 2, pp. 10:1–10:34, 2009.

[11] F. Serre and M. Püschel, "Generalizing block LU factorization: A lower-upper-lower block triangular decomposition with minimal off-diagonal ranks," *Linear Algebra and its Applications*, vol. 509, pp. 114–142, 2016.