IN SEARCH OF THE OPTIMAL WALSH-HADAMARD TRANSFORM FOR STREAMED PARALLEL PROCESSING

François Serre and Markus Püschel

Department of Computer Science
ETH Zurich
{serref, pueschel}@inf.ethz.ch

ABSTRACT

The Walsh-Hadamard transform (WHT) is computed using a network of butterflies, similar to the fast Fourier transform. The network is not unique but can be modified in exponentially many ways by properly changing the permutations between butterfly stages. Our first contribution is the exact characterization of all possible WHT networks. Then we aim to find the optimal networks for streaming implementations. In such an implementation the input is fed in chunks over several cycles and the hardware cost is thus reduced in proportion. To find the optimal network we smartly search through all possibilities for small sizes and discover novel networks that are thus proven optimal. The results can be used to extrapolate the optimal hardware cost for all sizes but the associated algorithms still remain elusive.

***Index Terms*—** Hadamard matrix, linear permutation, FPGA, streaming architecture, throughput

1. INTRODUCTION

The *Walsh-Hadamard transform* (WHT) is an important function in signal processing [1, 2] and coding theory [3, 4]. Similar to the fast Fourier transform (FFT), it is computed using a network of $n2^{n-1}$ butterflies but without the twiddle factors in between (see Fig 1(a) for $n = 4$). The network can be modified in exponentially many ways by properly changing the permutations between stages (e.g., [5]). For example, Fig 1(b) shows a Pease-like WHT [6] that consists of equal stages suitable for iterative implementation in hardware.

Knowing the exact space of valid WHT networks makes it possible to search for the optimal one for a given implementation task. In this paper we consider streaming implementations of the WHT. In these the input of size 2^n is fed in chunks of size 2^k over 2^{n-k} cycles and the hardware cost is reduced to $O(n2^k)$ [7, 8]. Fig. 2 shows an example for $2^k = 4$. Their implementation requires streaming permutation for which optimal solutions using RAM banks and switches are known [9, 10] under certain assumptions that hold here.

Contribution. In this paper we ask the following question: For given n and k, which is the optimal WHT network for a streaming implementation? Optimal means that the needed streaming permutations between stages require the minimal number of RAMs and switches. Towards answering this question we offer the following contributions:

- We exactly characterize the (exponentially large) set of all valid WHT networks such that the occurring permutations are linear. Linearity makes the efficient in implementation and is explained later.
- We present an algorithm to smartly search the large space of WHT networks at least for small sizes n.
- Using the search we find, for given n and k, novel and non-obvious WHT networks that have proven optimal hardware cost. An example is shown in Fig. 1(c).
- Our results show the trend in hardware cost and give evidence that there are for all sizes n yet undiscovered WHT networks that are optimal for streaming.

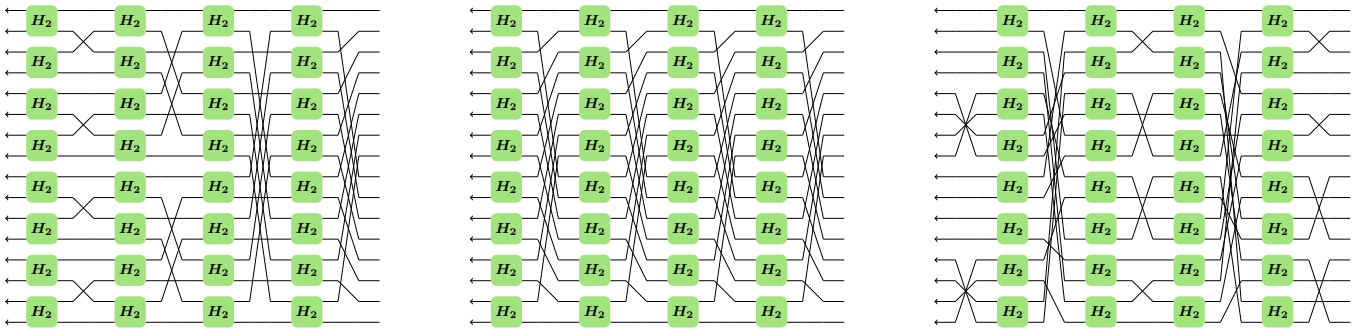
Related work. We use prior work on optimal streaming implementations with RAM banks of so-called linear permutations [9, 10]. Other types of memory in their implementations were studied in [11, 12, 13] and could be combined with our methods here. Streaming solutions for general permutations have been proposed in [14, 15].

Signal processing hardware with different resource size trade-offs is a classical research topic [16, 17, 18, 19]. Most closely related to our work are streaming implementations of FFTs [7, 8] that allow the choice of a desired trade-off between high performance and low resource consumption.

Finally, [5, 20] is similar in concept for WHT software implementations. It explores a large set of WHT algorithms to obtain efficient software library implementations, or as a test case for a model predicting the performances of libraries. However, the goal is to find the recursion best matched to the memory hierarchy; streaming in hardware poses a very different structural requirement.

2. BACKGROUND AND NOTATIONS

We provide background on Walsh-Hadamard transform (WHT) algorithms and their implementation in streaming



(a) Radix-$2^r = 2$ Iterative WHT (1)

(b) Radix-$2^r = 2$ Pease WHT (2)

(c) Algorithm that, once streamed with $2^k = 4$, yields an implementation that minimizes both the number of RAM stages, and the number of switch stages.

Fig. 1. Dataflows computing a WHT on $2^n = 8$ elements. The H_2 blocks represent butterflies.

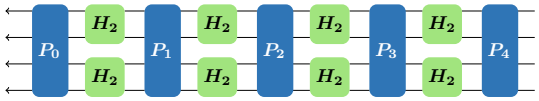


Fig. 2. Algorithms from Fig. 1 folded with a streaming width $2^k = 4$. This architecture uses a fourth of the number of butterflies, but processes the inputs over $2^{n-k} = 4$ cycles.

hardware. Particularly important are the permutations between butterfly stages for which there is a large degree of freedom that we use as a search space to find the optimum.

(Bit-)linear and bit-permutations. For an integer $0 \leq i < 2^n$, we denote its bit representation as i_b, viewed as column vector. For example, for $n = 3, 6_b = (1\,1\,0)^T$. Formally, $i_b \in \mathbb{F}_2^n$, where $\mathbb{F}_2$ is the Galois field with two elements.

Every $n \times n$ invertible bit-matrix P (i.e, $P \in \mathrm{GL}_n(\mathbb{F}_2)$) induces a permutation $\pi(P)$ on $\{0, 1, \ldots, 2^n - 1\}$. Namely, $\pi(P)(i) = j$ if $j_b = P \cdot i_b$. We call such permutation an LP (linear permutation) [21]. Note that not all permutations on 2^n points are linear: e.g., every $\pi(P)$ maps 0 to 0.

If P is even a permutation matrix, we call $\pi(P)$ a BP (bit-permutation). A well-known example is the bit-reversal permutation, where P has ones only on the diagonal from top-right to bottom-left. Another BP is the perfect shuffle, which on the bits is a cyclic right-shift C.

WHT algorithms. The WHT is a linear transform that computes $y = H_{2^n} x$ where $x, y \in \mathbb{C}^{2^n}$. It is defined by the *Hadamard matrix*:

$$H_2 = \left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right), \text{ and } H_{2^n} = H_2 \otimes H_{2^{n-1}}, \text{ for } n > 1.$$

Here, $\otimes$ denotes the *Kronecker product* defined as $A \otimes B = [a_{ij}B]_{i,j}$ for $A = [a_{ij}]$. H_2 is called butterfly.

The definition, recursively applied, directly yields the algorithm, i.e., butterfly network, shown in Fig. 1(a). It consists of n stages of 2^{n-1} butterflies $I_{2^{n-1}} \otimes H_2$, where I denotes the identity matrix. It is, in essence, a Cooley-Tukey FFT [22] without twiddle factors, requiring only $n2^n$ operations. As for the FFT, the occuring permutations are all BPs; thus, the algorithm can be formally written using the Kronecker formalism [23, 24] as

$$H_{2^n} = \pi(P_0) \cdot \prod_{\ell=1}^{n} \left((I_{2^{n-1}} \otimes H_2) \cdot \pi(P_\ell) \right), \qquad (1)$$

where $P_0, \ldots, P_n$ are $n \times n$ bit-permutation matrices. A given butterfly network, i.e., algorithm, can be manipulated in a myriad of ways to obtain variants, for example, by permuting the butterflies within stages. A popular example is the constant-geometry Pease-like [6] algorithm that has the same perfect shuffle $\pi(C)$ after every stage (see Fig. 1(b)). It is formally written as

$$H_{2^n} = \pi(I_n) \cdot \prod_{\ell=1}^{n} \left((I_{2^{n-1}} \otimes H_2) \cdot \pi(C) \right). \qquad (2)$$

We will later determine the exact set of possible permutations between stages to exhaustively search for the optimal WHT algorithm when used for the streaming implementations explained next.

Streaming WHT. A given WHT algorithm can be directly mapped to hardware but incurs $O(n2^n)$ area cost. To reduce this cost to $O(n2^k)$, while maintaining high throughput, the network can be folded as shown in Fig. 2 [7]. To do so, the input is split into chunks of 2^k elements (the *streaming width*) over 2^{n-k} cycles. This way, each stage only needs 2^{k-1} butterflies, which are reused 2^{n-k} times for the same dataset. The challenge is in implementing the needed streaming permutations, which now permute in space *and* time (across cycles).

Optimal solutions for BPs and LPs were developed in [9] and [10], respectively, assuming RAM banks and 2×2-switches as building blocks. The RAM optimality was shown

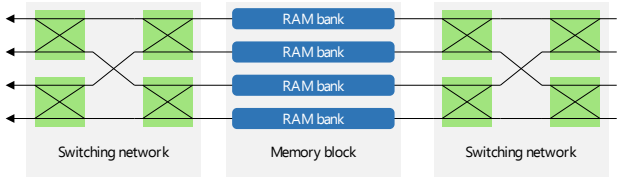


Fig. 3. A streaming LP implemented using the method in [10].

in [15]. An example for $2^k = 4$ is sketched in Fig. 3 with one stage of RAM banks and two blocks of 2 stages of switches each.

Important for this paper is only the minimal number of RAM and switch stages needed for a given P and k. First, P is blocked as

$$P = \begin{pmatrix} P_a & P_b \\ P_c & P_d \end{pmatrix}, \text{ with } P_d \text{ of size } k \times k.$$

If $P_a = I_{n-k}$ and $P_b = 0$ no RAM stage is needed, otherwise 1 stage (of 2^k RAMs). Further, the optimal number of switches needed in this case is

$$\max(\operatorname{rank} P_c, n - \operatorname{rank} P_a - \operatorname{rank} P_d)$$

stages of 2^{k-1} 2×2-switches each. Note that in some cases, the number of switches can be reduced at the price of twice the RAM [10]. We will not consider this option as we consider RAMs more costly.

3. ENUMERATION OF WHT ALGORITHMS

Besides the iterative and Pease WHT, many other variants can be derived, corresponding to different butterfly networks. For example, the butterflies in one stage can be permuted with a permutation σ. Formally, this means replacing

$$I_{2^{n-1}} \otimes H_2 = (\sigma \otimes I_2) \cdot (I_{2^{n-1}} \otimes H_2) \cdot (\sigma^{-1} \otimes I_2). \quad (3)$$

We are only interested in the cases where $\sigma \otimes I_2$ is linear, which is true if and only if σ is linear, i.e., $\sigma = \pi(P)$, $P \in \mathrm{GL}_{n-1}(\mathbb{F}_2)$. Using $\pi(P) \otimes I_2 = \pi(P \oplus I_1)$ ($\oplus$ is the block-diagonal composition) [9], (3) becomes

$$I_{2^{n-1}} \otimes H_2 = \pi(P \oplus I_1) \cdot (I_{2^{n-1}} \otimes H_2) \cdot \pi(P^{-1} \oplus I_1). \quad (4)$$

Using these degrees of freedom, (2) was derived from (1) and one could wonder whether there are more valid transformation of the WHT algorithms.

The following theorem is a main contribution of this paper and precisely characterizes all permutations between stages that produce a valid WHT:

Theorem 1. *The Hadamard matrix H_n satisfies*

$$H_{2^n} = \pi(P_0) \cdot \prod_{\ell=1}^{n} ((I_{2^{n-1}} \otimes H_2) \cdot \pi(P_\ell))$$

if and only if there exist $B \in \mathrm{GL}_n(\mathbb{F}_2)$ and $(Q_1, \cdots, Q_n) \in (\mathrm{GL}_{n-1}(\mathbb{F}_2))^n$ such that

$$P_0 = B \cdot \begin{pmatrix} Q_1 & \\ & 1 \end{pmatrix}, P_n = \begin{pmatrix} 1 & \\ & Q_n^{-1} \end{pmatrix} \cdot B^T, \text{ and}$$

$$P_\ell = \begin{pmatrix} 1 & \\ & Q_\ell^{-1} \end{pmatrix} \cdot \begin{pmatrix} Q_{\ell+1} & \\ & 1 \end{pmatrix}, \text{ for } 0 < \ell < n.$$

In particular, there are $g_{n-1}^n g_n$ possibilities, where $g_n = |\mathrm{GL}_n(\mathbb{F}_2)| = \prod_{i=0}^{n-1}(2^n - 2^i)$.

As an example, (2) corresponds to $Q_i = I_{n-1}$ for $1 \le i \le n$, and $B = I_n$.

Proof. We only provide a sketch of the proof due to space limitations; a complete one is available in [25]. First, we derive a necessary condition for the matrices $P_0, \ldots, P_n$ such that the matrix $\pi(P_0) \cdot \prod_{\ell=1}^{n} ((I_{2^{n-1}} \otimes H_2) \cdot \pi(P_\ell))$ has no zero elements. Then, assuming that this condition holds, we derive a general expression of this matrix, and match it with H_{2^n}. This yields the shown set of necessary conditions that turn out to be also sufficient. □

Note that the theorem shows that in addition to (4), there is another, non-obvious degree of freedom: for any LP $\pi(B)$, the WHT satisfies

$$H_{2^n} = \pi(B) \cdot H_{2^n} \cdot \pi(B^T). \quad (5)$$

This degree of freedom will later produce novel optimal WHT algorithms for streaming implementations. Note that, in general, $\pi(B^T) \neq \pi(B)^T$.

4. SEARCH OF OPTIMAL ALGORITHMS

The number of WHT algorithms $g_{n-1}^n g_n$ grows exponentially (e.g., it is about 10^{51} for $n = 5$), which makes enumeration infeasible, except for very small sizes. Here we propose a search algorithms that pushes feasibility into the region of $n = 5$–8 to find evidence for the existence of better, unknown algorithms for streaming.

We search for WHT algorithms that minimize a given implementation cost that can be evaluated on each matrix P as $\mathrm{Cost}(P)$. We assume a function G_ℓ that can enumerate all invertible $\ell \times \ell$ bit matrices; $G_\ell(i)$ is the i^{th} such matrix.

Our approach consists of two main steps. We first compute a matrix $C^{\text{int}} = (c^{\text{int}}_{i,j})_{0 \le i,j < g_{n-1}}$, containing the minimal cost of the internal permutations $\mathrm{Cost}(P_1) + \cdots + \mathrm{Cost}(P_{n-1})$ for each possible pair of matrices $Q_1 = G_{n-1}(i)$ and $Q_n = G_{n-1}(j)$. Then, we compute similarly a matrix C^{ext} containing the minimal cost of the external permutations $\mathrm{Cost}(P_0) + \mathrm{Cost}(P_n)$. The optimal cost is then the smallest element of $C^{\text{int}} + C^{\text{ext}}$.

Log of size (n)	2	3		4			5				6					7					
Log of str. width (k)	1	1	2	1	2	3	1	2	3	4	1	2	3	4	5	1	2	3	4	5	6
Radix-2 Pease	4	6	6	8	8	8	10	10	10	10	12	12	12	12	12	14	14	14	14	14	14
Radix-2 Iterative	4	6	4	8	6	4	10	8	6	4	12	10	8	6	4	14	12	10	8	6	4
Best with BP	4	6	4	8	8	4	10	10	8	4	12	12	12	8	4	14	14	12	12	8	4
Best with LP	4	6	3	8	5	3	10	6	5	3	12	?	?	?	3	?	?	?	?	?	?

Table 1. Number of stages of 2^{k-1} switches in WHTs of size 2^n implemented with a streaming width of 2^k.

Internal cost matrix C^{int}. We first store in a matrix $C = (c_{i,j})_{0 \le i,j < g_{n-1}}$ the cost that a single internal permutation $\pi(P_\ell)$ would have, given $Q_\ell = G_{n-1}(i)$ and $Q_{\ell+1} = G_{n-1}(j)$:

$$c_{i,j} = \text{Cost}\left(\begin{pmatrix} & G_{n-1}^{-1}(i) \\ 1 & \end{pmatrix} \cdot \begin{pmatrix} G_{n-1}(j) & \\ & 1 \end{pmatrix}\right).$$

The minimal cost that two consecutive internal permutations $\pi(P_\ell)$ and $\pi(P_{\ell+1})$ would have, given $Q_\ell = G_{n-1}(i)$ and $Q_{\ell+2} = G_{n-1}(j)$ is

$$\min_{Q_{\ell+1}} \text{Cost}\,(P_\ell) + \text{Cost}\,(P_{\ell+1}) = \min_k c_{i,k} + c_{k,j}.$$

This computation corresponds to the distance product [26] of C with itself. Getting C^{int} consists of performing this task $n-1$ times: $C^{\text{int}} = C^{n-1}$. We use a fast exponentiation algorithm, leading to an arithmetic complexity in $O(g_{n-1}^3 \log(n))$, and a memory footprint in $O(g_{n-1}^2)$ for this step.

External cost matrix C^{ext}. The matrix C^{ext} of the cost of the external permutations can be obtained by trying all the possible matrices for B, for each pair Q_1, Q_n. This step has an arithmetic complexity in $O(g_n g_{n-1}^2)$.

This search can be simplified for a given cost. For instance, for the cost shown in the result section, a close formulation for an optimal B was possible, making the second step negligible compared to the first. Further, our algorithm can be restricted to a subset of all linear permutations as shown in the results.

5. RESULTS

We implemented the search algorithm using as building block a specially designed linear algebra library for the efficient computation with 8×8 bit-matrices. Our cost function first minimizes the number of stages of RAM banks, and then the number of switching stages. However, the arithmetic complexity of the algorithm prevented us from completing the search for $n \ge 6$ (which would have taken years). Therefore, we have also run the search restricted to permutations that are BPs (thus reducing the complexity to $O((n-1)!^3 \log(n))$). This choice is also theoretically important as all-known WHT networks (just as power-of-two FFT networks) are built using BPs. We will see that BPs alone will not yield the optimum.

Number of RAM stages. The Pease algorithm, when streamed, requires n stages of RAM banks, the iterative WHT only $n-k+1$ (using [9]). The minimal number of RAM stages found by our search for both LPs and restricted to BPs is $\lceil n/k \rceil$ and thus better. Note that if k divides n this cost can be achieved using an iterative radix-2^k algorithm. For k not dividing n our search finds novel solutions.

Number of switching stages. The minimal number of switching stages over all RAM-optimal algorithms found by our search is shown in Table 1. Note that in some cases, the best algorithm that uses BPs has more switches than the iterative algorithm. In these cases, the latter is not RAM-optimal.

Most interestingly, for streaming widths $2^k \neq 2$ our search with LPs discovers novel WHT networks that improve prior ones in both RAM usage and required switches. Considering BPs alone (as in all known network variants including the large space in [5]) is not sufficient. Further, all optimal networks found have a non-trivial B in (5).

Unfortunately, we did not manage to extrapolate from the WHT networks found to optimal solutions for all sizes n and k. To illustrate the difficulty consider the optimal network found for $n = 4$ and $k = 2$ in Fig. 1(c).

6. CONCLUSION

We introduced an idea that is of both theoretical and practical interest: namely, for an algorithm with regular structure and a given implementation task one can enumerate all possible variants to find the optimal solution. We considered the WHT but the idea is in principle applicable to fast Fourier transforms, sorting networks, Viterbi decoders, and other regular algorithms. The first challenge is in characterizing the space of algorithmic variants, which we did for the WHT and was one main contribution. The other challenge is in extrapolating the optimal solutions found for small sizes to all sizes. This remains an open question but our results for small sizes strongly indicate that there is a class of yet undiscovered, and non-obvious WHT algorithms with reduced RAM and logic requirements in streaming implementation.

7. REFERENCES

[1] Jacques Hadamard, "Résolution d'une question relative aux déterminants," *Bulletin des sciences mathématiques*, vol. 17, pp. 240–246, 1893.

[2] K. G. Beauchamp, *Applications of Walsh and related functions with an introduction to sequency theory*, Academic Press, 1985.

[3] Florence Jessie MacWilliams and Neil James Alexander Sloane, *The theory of error-correcting codes*, Elsevier, 1977.

[4] Rao K. Yarlagadda and John E. Hershey, *Hadamard Matrix Analysis and Synthesis*, Springer Us, 2012.

[5] Jeremy Johnson and Markus Püschel, "In search of the optimal Walsh-Hadamard transform," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2000, vol. 6, pp. 3347–3350.

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

[7] Grace Nordin, Peter A. Milder, James C. Hoe, and Markus Püschel, "Automatic generation of customized discrete Fourier transform IPs," in *Proc. Design Automation Conference (DAC)*, 2005, pp. 471–474.

[8] Peter A. Milder, Franz Franchetti, James C. Hoe, and Markus Püschel, "Computer generation of hardware for linear digital signal processing transforms," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 17, no. 2, pp. 15:1–15:33, 2012.

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

[10] François Serre, Thomas Holenstein, and Markus Püschel, "Optimal circuits for streamed linear permutations using RAM," in *Proc. International Symposium on Field-Programmable Gate Arrays (FPGA)*, 2016, pp. 215–223.

[11] Keshab K. Parhi, "Systematic synthesis of DSP data format converters using life-time analysis and forward-backward register allocation," *IEEE Transactions on Circuits and Systems II (TCAS-II)*, vol. 39, no. 7, pp. 423–440, 1992.

[12] Kevin J. Page and Paul M. Chau, "Folding large regular computational graphs onto smaller processor arrays," in *Advanced Signal Processing Algorithms, Architectures and Implementations*, Proc. SPIE, Ed., 1996, vol. 2846, pp. 383–394.

[13] Tuomas Järvinen, Perttu Salmela, Harri Sorokin, and Jarmo Takala, "Stride permutation networks for array processors," in *Proc. International Conference on Application-Specific Systems, Architectures and Processors Proceedings (ASAP)*, 2004, pp. 376–386.

[14] Peter A. Milder, James C. Hoe, and Markus Püschel, "Automatic generation of streaming datapaths for arbitrary fixed permutations," in *Proc. Design, Automation and Test in Europe (DATE)*, 2009, pp. 1118–1123.

[15] Thaddeus Koehn and Peter Athanas, "Arbitrary streaming permutations with minimum memory and latency," in *Proc. International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1–6.

[16] Danny Cohen, "Simplified control of FFT hardware," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 24, no. 6, pp. 577–579, 1976.

[17] Pinit Kumhom, Jeremy R. Johnson, and Prawat Nagvajara, "Design, optimization, and implementation of a universal FFT processor," in *Proc. International ASIC/SOC Conference (ASIC)*, 2000, pp. 182–186.

[18] Ainhoa Cortés, Igone Vélez, and Juan F. Sevillano, "Radix r^k FFTs: Matricial representation and SDC/SDF pipeline implementation," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2824–2839, 2009.

[19] Berkin Akin, Franz Franchetti, and James C. Hoe, "FFTs with near-optimal memory access through block data layouts: Algorithm, architecture and design automation," *Journal of Signal Processing Systems*, vol. 85, no. 1, pp. 67–82, 2015.

[20] Jeremy Johnson and Michael Andrews, "Statistical evaluation of a self-tuning vectorized library for the Walsh-Hadamard transform," in *International Workshop on State-of-the-Art in Scientific and Parallel Computing (PARA)*, 2008.

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

[22] James W. Cooley and John W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[23] J. R. Johnson, R. W. Johnson, D. Rodriguez, and R. Tolimieri, "A methodology for designing, modifying, and implementing Fourier transform algorithms on various architectures," *Circuits, Systems and Signal Processing*, vol. 9, no. 4, pp. 449–500, 1990.

[24] C. Van Loan, *Computational Framework of the Fast Fourier Transform*, Siam, 1992.

[25] François Serre and Markus Püschel, "Characterizing and enumerating Walsh-Hadamard transform algorithms," *CoRR*, vol. abs/1710.08029, 2017.

[26] Robert W. Floyd, "Algorithm 97: Shortest path," *Commun. ACM*, vol. 5, no. 6, pp. 345, 1962.