Corinthios *SpringerPlus (2016)5:1511*
DOI 10.1186/s40064-016-3162-9

SpringerPlus

RESEARCH Open Access

Chrestenson transform FPGA embedded factorizations

Michael J. Corinthios*

*Correspondence: michael.corinthios@polymtl.ca
Ecole Polytechnique de Montréal, Campus Université de Montréal, 2500 Chemin de Polytechnique, Montréal, QC H3T 1J4, Canada

Abstract

Chrestenson generalized Walsh transform factorizations for parallel processing imbedded implementations on field programmable gate arrays are presented. This general base transform, sometimes referred to as the Discrete Chrestenson transform, has received special attention in recent years. In fact, the Discrete Fourier transform and Walsh–Hadamard transform are but special cases of the Chrestenson generalized Walsh transform. Rotations of a base-p hypercube, where p is an arbitrary integer, are shown to produce dynamic contention-free memory allocation, in processor architecture. The approach is illustrated by factorizations involving the processing of matrices of the transform which are function of four variables. Parallel operations are implemented matrix multiplications. Each matrix, of dimension $N \times N$, where $N = p^n$, n integer, has a structure that depends on a variable parameter k that denotes the iteration number in the factorization process. The level of parallelism, in the form of $M = p^m$ processors can be chosen arbitrarily by varying m between zero to its maximum value of $n - 1$. The result is an equation describing the generalised parallelism factorization as a function of the four variables n, p, k and m. Applications of the approach are shown in relation to configuring field programmable gate arrays for digital signal processing applications.

Keywords: Spectral analysis, Generalised spectral analysis, Generalised Walsh transform, Discrete Chrestenson transform, Discrete Fourier transform, Parallel processing, Hypercube transformations, General-radix matrix factorization

Background

Applications of the Discrete Fourier, Walsh–Hadamard and Chrestenson generalized Walsh CGW transforms in spectral analysis and digital signal processing (Corinthios 1985, 2009; Bespalov 2010) have received particular attention in recent years thanks to rapid advances of microelectronics in general and field programmable gate arrays FPGAs in particular. The search for higher processing speeds through increasing levels of parallelism motivate the search for optimal transform factorizations.

In this paper a formalism and an algorithm for configuring and sequencing parallel processors implementing factorizations of the ('Discrete') Chrestenson generalized Walsh CGW transform are presented. This general base transform has received special attention in recent years. In fact, Discrete Fourier transform and Walsh–Hadamard transform are but special cases of the CGW transform. The architecture of a digital signal processor is defined as optimal if it leads to a minimization of addressing

requirements, of shuffle operations and of the number of required memory partitions (Corinthios 1994). The factorizations are developed with a view to implementation as embedded architectures of presently available FPGAs (Harmut et al. 2010; Huda et al. 2014).

The algorithms and corresponding architectures relate to general base matrix factorizations (Corinthios 2009). Rotations of a base-p hypercube, where p is an arbitrary integer, produce dynamic memory allocation, in processor architecture. The approach produces factorizations involving the processing of matrices of the transform which are function of four variables. Parallel operations are implemented matrix multiplications. Each matrix, of dimension $N \times N$, where $N = p^n$, n integer, has a structure that depends on a variable parameter k. The level of parallelism, in the form of $M = p^m$ processors can be chosen arbitrarily by varying m between zero to its maximum value of $n - 1$. The result is an equation describing the generalised parallelism factorization as a function of the four variables n, p, k and m. Applications of the approach are shown in relation to configuring field programmable gate arrays for digital signal processing applications.

Hypercube transformations have been applied to diversified problems of information processing. The present paper describes an approach for FPGA parallel processor configuration using an *arbitrary number M* of general-base processing elements, where $M = p^m$, p being the general radix (base) of factorization. The input data vector dimension N, or input data matrix dimension $N \times N$, where $N = p^n$, the radix, or base, p of factorization of the transformation matrix, the number of processors M, and the span of the matrix, that is, the spacing between data simultaneously accessed are all variable. A unique optimal solution yielding a progressive degree parallel to massively parallel optimal architectures is presented.

Matrix structures

In what follows some definitions relating to the special structure of sparse, permutation and transformation matrices (Corinthios 1994) are employed. In particular matrix span is taken to mean the distance between two successive nonzero elements along a row or a column. A fixed topology processor is one that accesses data in a fixed geometry pattern where data points are equidistant throughout the different iterations, thus requiring no addressing. A shuffle-free algorithm is one that necessitates no data shuffling between iterations. A p^k-optimal algorithm is one that requires access of matrix elements which are spaced by a minimum distance of N/p^k elements. In addition we adopt the following definitions.

General base processing element

In what follows a general-base processing element PE with a base, or radix, p is a processor that receives simultaneously p input operands and produces simultaneously p output operands. The PE in general applies arithmetic or weighting operations on the input vector to produce the output vector. In matrix multiplication operations for example the PE applies a $p \times p$ matrix to the p-element input vector to produce the p-element output vector. The matrix elements may be real or complex.

Due to the diversified general applicability of such a processing element a universal processing element (UPE), which can be constructed in a 3D-type architecture has been proposed (Corinthios 1985). In the present context a UPE may be seen simply as a general base-p processing element PE as defined above, accepting p inputs, weighting them by the appropriate $p \times p$ matrix and producing p output operands.

Pilot elements, pilots matrix

Similarly to signals and images an $N \times N$ matrix may be sampled and the result is "impulses", that is, isolated elements in the resulting $N \times N$ samples matrix. We shall assume uniform sampling of rows and columns yielding p uniformly spaced samples from each of p rows and element alignment along columns, that is, p uniformly spaced samples along columns as well as rows. The samples matrix which we may refer to as a "frame" thus contains p rows of p equally spaced elements each, a rectangular grid of p^2 impulses, which we may refer to as "poles", which we shall call a "dispatch". With $N = p^n$ the N^2 elements of the "main" (or "parent") matrix, that is, the original matrix before sampling, may be thus decomposed into $N^2/p^2 = p^{n-2}$ such dispatches.

By fixing the row sampling period as well as the column sampling period, the row and column spans of the resulting matrix are known. It therefore suffices to know the coordinates (indices) of the top left element, that is, the element with the smallest of indices, of a dispatch to directly deduce the positions of all its other poles. The top left element acts thus as a reference point, and we shall call it the "pilot element". The other $p^2 - 1$ elements associated with it may be called its "satellites".

In other words if the element a_{ij} of A is a pilot element, the dispatch consists of the elements

$$a_{i+kc,j+lr}; \quad k = 0, 1, \ldots, p-1, \quad l = 0, 1, \ldots, p-1$$

c and r being the column and row element spacing (spans), respectively.

A processing element assigned to a pilot element can thus access all p^2 operands of the dispatch, having deduced their positions knowing the given row and column spans.

Since each pilot element of a frame originated from the same position in the parent matrix we can construct a "pilots matrix" by keeping only the pilot elements and forcing to zero all other elements of the parent matrix. The problem then is one of assignment, simultaneous and/or sequential, of the $M = p^m$ processors to the different elements of the pilots matrix.

Hypercube dimension reduction

The extraction of a pilots matrix from its parent matrix leads to a dimension reduction of the hypercube representing its elements. The dimension reduction is in the form of a suppression, that is, a forcing to zero, of one of the hypercube digits. Let $C = (j_{n-1}, \ldots, j_1 j_0)$, $j_k \in \{0, 1, 2, \ldots, p-1\}$ be an n-digit base-p hypercube. We will write $C_{\bar{k}}$ to designate the hypercube C with the digit k suppressed, that is, forced to zero. Several digits can be similarly suppressed. For example, $C_{\overline{2,4}} = \left(j_{n-1} \ldots j_5 0 j_3 0 j_1 j_0\right)$, and $C_{\overline{n-1}} = \left(0 j_{n-2} \ldots j_1 j_0\right)$.

Parallel configuration algorithm

A sequence of perfect shuffle operations effected through simple hypercube transformations can be made to broadcast the parallel configuration and access assignments to the different processors. The overall approach is described by the following algorithm.

Algorithm 1: Parallel Dispatch, State Assignment and Sequencing Algorithm
Read base p
$n = \log_p N$
$m = \log_p M$

Read Input matrix A
For $k = 0$ to $n-1$ do
For $r = 0$ to $n-2$ do
begin
Assign variables $i_0, i_1, ..., i_{m-1}$ to $M = p^m$ processors
Evaluate row span σ_R
Evaluate column span σ_c
Test optimality
Select scan type
Evaluate pitch
Dispatch M parallel processors
Assign variables $j_m, j_{m+1}, ..., j_{n-1}$ to the access sequencing order of each processor.
Effect hypercube transformations,

$$\left(j_{n-1}...j_{m+1}j_m i_{m-1}...i_1 i_0\right) \rightarrow \left(j_{n-1}...j_{m+1}j_m i_{m-1}...i_1 i_0\right)'$$

(primes denote cube transformation)
for $k = 0$ to p^{n-m-1} do
begin
Fork NEXT
Dispatch l processor, $l = 0, 1, \ldots, m-1$, in parallel, to Pilot address (row and column coordinates)

$$\left\{w\left(j_{n-1}...j_{m+1}j_m i_{m-1}...i_1 i_0\right)',\ z\left(j_{n-1}...j_{m+1}j_m i_{m-1}...i_1 i_0\right)''\right\}$$

NEXT
for $s = 0, 1, ..., p-1$
$w_R(s) \leftarrow w + s\sigma_R$
$z_c(s) \leftarrow z + s\sigma_c$
end
end
Increment j for sequential cycles
end
end

The parallel dispatch, state assignment and sequencing Algorithm 1 dispatches the $M = p^m$ processors for each stage of the matrix factorization. The base-p m-tuple $(i_{m-1}i_{m-2}...i_1i_0)$ is assigned to the parallel processors. The $(n - m)$ tuple $(j_{n-1}j_{n-2}...j_m)$ is assigned to the sequencing cycles of each processor. The algorithm subsequently applies hypercube transformations as dictated by the type of matrix, the stage of matrix factorization and the number of dispatched processors. It tests optimality to determine the type of scan of matrix elements to be applied and evaluates parameters such as pitch and memory optimal queue length, to be defined subsequently, it accesses the pilot elements and their satellites, proceeding to the parallel dispatch and sequencing of the processing elements.

Each processing element at each step of the algorithm thus accesses from memory its p input operands and writes into memory those of its output operands. The algorithm, while providing an arbitrary, generalised, level of parallelism up to the ultimate massive parallelism, produces optimal multiprocessing machine architecture minimizing

addressing, the number of memory partitions as well as the number of required shuffles. Meanwhile it produces virtually wired-in pipelined architecture and properly ordered output.

General matrix decomposition

In developing techniques for the general-base factorization of transformation matrix multiplications it is convenient to effect a decomposition of a matrix into the sum of matrices. To this end let us define an "impulse matrix" as the matrix $\delta(i, j)$ of which all the elements are zero except for the element at position (i, j), that is,

$$\left|\delta(i,j)\right|_{uv} = \begin{cases} 1, & u = i,\ v = j \\ 0, & otherwise \end{cases}. \tag{1}$$

An $N \times N$ matrix A having elements $[A]_{i,j} = a_{ij}$ can be written as the sum

$$\begin{aligned} A = {} & a_{0,0}\delta(0,0) + a_{0,1}\delta(0,1) + a_{0,2}\delta(0,2) + \cdots + a_{1,0}\delta(1,0) \\ & + a_{1,1}\delta(1,1) + \cdots + a_{N-1,N-1}\delta(N-1,N-1) \end{aligned} \tag{2}$$

where the $\delta(i, j)$ matrices are of dimension $N \times N$ each. The matrix A can thus be written in the form

$$A = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} a_{i,j}\delta(i,j). \tag{3}$$

Furthermore, in the parallel processing of matrix multiplication to a general base p it is convenient to decompose an $N \times N$ matrix with $N = p^n$ as the sum of dispatches, a dispatch being, as mentioned earlier, a matrix of p^2 elements arranged in a generally rectangular $p \times p$ pattern of p columns and p rows. Denoting by σ_R and σ_C the row and columns spans of a dispatch we can decompose a matrix A into the form

$$A = \sum_{i=0}^{N/p-1}\sum_{j=0}^{N/p-1}\sum_{k=0}^{p-1}\sum_{l=0}^{p-1} a_{i+k\sigma_C,j+l\sigma_R}\delta(i+k\sigma_C, j+l\sigma_R). \tag{4}$$

More generally we may wish to decompose A in an order different from the uniform row and column scanning as in this last equation. In other words we may wish to pick the dispatches at an arbitrary order rather than in sequence. As mentioned above, we shall call the top left element the pilot element and its $p^2 - 1$ companions its satellites. In this last equation the pilot elements are those where $k = 1 = 0$.

To effect a parallel matrix decomposition to a general base-p we use hypercubes described by base-p digits. The order of accessing the different dispatches is made in relation to a main clock. The clock K is represented by the hypercube to base p as

$$K \simeq (k_{n-1} \ldots k_1 k_0)_p; \quad k_i \in \{0, 1, \ldots, p-1\} \tag{5}$$

Its value at any time is given by

$$K = \sum_{t=0}^{n-1} p^t k_t. \tag{6}$$

At each clock value K a set of M UPE's (PE's) is assigned a set of M dispatches simultaneously. We will reserve the symbols w and z to designate the row and column indices of a pilot element at clock K. In other words, at clock K each selected pilot element shall be designated $a_{w,z}$, that is, $[A]_{w,z}$ where w and z are functions of K to be defined. They will be determined in a way that optimizes the parallel and sequential operations for the given matrix structure and the number $M = p^m$ of available UPE's.

With $M = p^m$ base-p processing elements the hypercube representing K shall be rewritten in the form

$$K \simeq \left(j_{n-1} \dots j_{m+1} j_m i_{m-1} \dots i_1 i_0\right)_p \tag{7}$$

where we have written

$$k_t = \begin{cases} i_t, & t = 0, 1, \dots, m-1 \\ j_t, & t = m, m+1, \dots, n-1. \end{cases} \tag{8}$$

The m-sub-cube (i_{m-1}, ..., i_1, i_0) designates operations performed in parallel. The remaining ($n - m$)-sub-cube (j_{n-1}, ..., j_{m+1}, j_m) designates operations performed sequentially by each of the M dispatched parallel processors. With $M = p^m$ processors dispatched in parallel at clock $K \simeq (j_{n-1} \dots j_{m+1} j_m i_{m-1} \dots i_1 i_0)_p$ the matrix A can be decomposed in the form

$$\begin{aligned} A = & \sum_{k_{n-2}=0}^{p-1} \dots \sum_{k_{m+1}=0}^{p-1} \sum_{k_m=0}^{p-1} \\ & \left\langle \sum_{k_{m-1}=0}^{p-1} \dots \sum_{k_1=0}^{p-1} \sum_{k_0=0}^{p-1} \sum_{l=0}^{p-1} \sum_{k=0}^{p-1} a_{w(k_0,k_1,\dots,k_{n-1})+k\sigma_C, z(k_0,k_1,\dots,k_{n-1})+l\sigma_R} \right. \\ & \left. \delta\left[\left\{w(k_0, k_1, \dots, k_{n-2}) + k\sigma_C\right\}, \left\{z(k_0, k_1, \dots, k_{n-2}) + l\sigma_R\right\}\right] \right\rangle \end{aligned} \tag{9}$$

where the "parentheses" $\langle$ and $\rangle$ enclose the elements accessed in parallel. In what follows we write $P_{\nu,\mu}$ to designate the pilot element of processor no. ν at real time clock μ.

Application to the CGW transforms

The lowest order base-p Chrestenson generalised Walsh CGW "core matrix" is the p-point the Discrete Fourier matrix

$$W_p = \frac{1}{\sqrt{p}} \begin{bmatrix} w^0 & w^0 & \dots & w^0 \\ w^0 & w^1 & \dots & w^{p-1} \\ \vdots & & & \\ w^0 & w^{p-1} & \dots & w^{(p-1)^2} \end{bmatrix} \tag{10}$$

where

$$w = \exp\left(-j2\pi/p\right), \quad j = \sqrt{-1}. \tag{11}$$

In the following, for simplicity, the scaling factor $1/\sqrt{p}$ will be dropped. We start by deriving three basic forms of the Chrestenson (generalised Walsh GW) transform in its

three different orderings: in natural order CGWN, in Walsh–Paley order CGWP and in Walsh–Kaczmarz order CGWK.

The CGWN transformation matrix

The CGWN transformation matrix W_N for $N = p^n$ data points is obtained from the generalised-Walsh core matrix W_p by the Kroneker multiplication of W_p by itself n times.

$$W_{N,nat} = W_p \times W_p \times \cdots \times W_p (n \text{ times}) = W_p^{[n]}. \tag{12}$$

CGWP transformation matrix

The generalised Walsh transform in the CGWP order is related to the transform in natural order by a digit-reverse ordering. The general-base digit reverse ordering matrix $K_N^{(p)}$ can be factored using the general-base perfect shuffle permutation matrix $P^{(p)}$, also denoted simply P, and Kroneker products

$$K_N^{(p)} = \prod_{i=0}^{n-1} \left(P^{(p)}_{p^{(n-i)}} \times I_{p^i} \right) \tag{13}$$

where I_K is the identity matrix of dimension K.

Operating on a column vector x of dimension K the base-p perfect shuffle permutation matrix of dimension $K \times K$ produces the vector

$$P_K x = \left[x_0, x_{K/p}, x_{2K/p}, \ldots, x_{(p-1)K/p}, x_1, x_{K/p+1}, \ldots, x_2, x_{K/p+2}, \ldots, x_{K-1} \right] \tag{14}$$

The CGWP matrix $W_{N,WP}$ can thus be written in the form

$$W_{N,WP} = K_N^{(p)} W_{N,nat} = \prod_{i=0}^{n-1} \left(P^{(p)}_{p^{(n-1)}} \times I_{p^i} \right) W_p^{[n]}. \tag{15}$$

CGWK transformation matrix

The CGWK transformation matrix is related to the CGWP matrix through a p-ary to Gray transformation matrix $G_N^{(p)}$.

$$W_{N,WK} = G_N^{(p)} W_{N,WP}. \tag{16}$$

The following factorizations lead to shuffle-free optimal parallel-pipelined processors.

CGWN optimal factorization

A fixed topology factorization of the CGWN transformation matrix has the form

$$W_{N,nat} = \prod_{i=0}^{n-1} P_N C_N = \prod_{i=0}^{n-1} P_N \left(I_{N/p} \times W_p \right) \tag{17}$$

which can be re-written in the form

$$W_{N,nat} = P \left\{ \prod_{n=0}^{n-1} CP \right\} P^{-1} = P \left\{ \prod_{n=0}^{n-1} F \right\} P^{-1} \tag{18}$$

$$C = C_N = I_{p^{n-1}} x W_p \tag{19}$$

and $F = \mathrm{CP}$, noting that the matrix F is p^2-optimal.

CGWP optimal factorization

We fixed topology factorization of the CGWP matrix has the form

$$W_{N,WP} = \prod_{i=0}^{n-1} J_i C_N \tag{20}$$

$$J_i = \left(I_{P^{n-i-1}} \times P_{p^{i+1}} \right) = H_{n-i-1} \tag{21}$$

Letting

$$Q_i = C_N J_{i+1} = C_N H_{n-i-2}, \quad i = 0, 1, \ldots, n-2$$

$$Q_{n-1} = C_N \tag{22}$$

we obtain

$$W_{N,WP} = \prod_{i=0}^{n-1} Q_i \tag{23}$$

where each matrix Q_i, $i = 0, 1, \ldots, n-2$, is p^2-optimal, while Q_{n-1} is p-optimal.

CGWK optimal factorization

The fixed topology CGWK factorization has the form

$$W_{N,WK} = P \left\{ \prod_{i=0}^{n-1} P^{-1} H_i C_N E_i \right\} P^{-1} \tag{24}$$

Letting

$$H_i = I_{p^i} \times P_{p^{n-i}}, \quad E_i = I_{p^i} \times D'_{p^{n-i}} \tag{25}$$

$$D'_{p^n} = \mathrm{quasidiag}\left(I_{p^{n-1}}, D_{p^{n-1}}, D^2_{p^{n-1}}, \ldots, D^{(p-1)}_{p^{n-1}} \right) \tag{26}$$

A quasidiagonal matrix is a matrix containing matrices along its diagonal and null matrices elsewhere.

$$D^i_{p^{n-1}} = D^i_p \times I_{p^{n-2}}$$

$$D_p = \mathrm{diag}\left(w^0, w^{-1}, w^{-2}, \ldots, w^{-(p-1)} \right) \tag{27}$$

$$W_{N,WK} = P\left\{\prod_{i=0}^{n-1} P^{-1} H_i G_i\right\} P^{-1}, \tag{28}$$

where

$$G_i = C_N E_i \tag{29}$$

Letting

$$S_i = P^{-1} H_i P = \left(I_{p^{i-1}} \times P_{p^{n-i}} \times I_p\right) \tag{30}$$

we have

$$W_{N,WK} = P^2\left\{\prod_{i=0}^{n-1} P^{-1} G_i S_{i+1}\right\} P^{-1} \tag{31}$$

with

$$S_{n-1} = S_n = I_N \tag{32}$$

The factorization can also be re-written in the form

$$W_{N,WK} = P\left\{\prod_{i=0}^{n-1} \Gamma_i\right\} P^{-1}, \tag{33}$$

where

$$\begin{aligned} \Gamma_i &= P^{-1} G_i S_{i+1} \\ &= P^{-1} G_i \left(I_{p^i} \times P_{p^{n-i-1}} \times I_p\right) i = 1, 2, \ldots, n-1; \\ \Gamma_0 &= G_0 S_1. \end{aligned} \tag{34}$$

The matrices Γ_i are p^2-optimal, except for Γ_0 which is maximal span. These are therefore optimal algorithms which can be implemented by an optimal parallel processor, recirculant or pipelined, with no shuffling cycle called for during any of the n iterations.

Application to image processing

The potential in enhanced speed through high-level parallelism of the optimal algorithms is all the more evident within the context of real-time image processing applications. For 2D signals, algorithms of generalised spectral analysis can be applied on sub-images or on successive column-row vectors of the input image. Factorizations of the algorithms of the Chrestenson transform applied on an $N \times N$ points matrix X representing an image, with $N = p^n$ can be written for the different transform matrices. The CGWN 2D transformation for optimal pipelined architecture can be written in the form

$$\begin{aligned} Y_{nat} &= P\left\{\prod_{i=0}^{n-1} F\right\} P^{-1} \times \left[P\left\{\prod_{i=0}^{n-1} F\right\} P^{-1}\right]^T \\ &= P\left\{\prod_{i=0}^{n-1} F\right\} P^{-1} \times P\left\{\prod_{i=0}^{n-1} F\right\} P^{-1}, \end{aligned} \tag{35}$$

where T stands for transpose. The CGWP factorization can be written in the form

$$\begin{aligned} Y_{WP} &= \prod_{i=0}^{n-1} Q_i \times \left(\prod_{i=0}^{n-1} Q_i \right)^T \\ &= \prod_{i=0}^{n-1} Q_i \times \prod_{i=0}^{n-1} Q_{n-i-1}^T, \end{aligned} \tag{36}$$

$$Q_i^T = C_N \left(I_{p^{n-i-1}} \times P_{p^{i+1}}^{-1} \right) \tag{37}$$

The CGWK factorization for optimal pipelined architecture can be written in the form

$$\begin{aligned} Y_{WK} &= P^2 \left\{ \prod_{i=0}^{n-1} \Gamma_i \right\} P \times \left[P^2 \left\{ \prod_{i=0}^{n-1} \Gamma_i \right\} P \right]^T \\ &= P^2 \left\{ \prod_{i=0}^{n-1} \Gamma_i \right\} P \times P^{-1} \left\{ \prod_{i=0}^{n-1} \Gamma_{n-i-1}^T \right\} P^{-2}, \end{aligned} \tag{38}$$

$$\Gamma_i^T = \left(I_{p^i} \times P_{p^{n-i-1}}^{-1} \times I_p \right) G_i^{-1} P. \tag{39}$$

These fast algorithms are all p^2-optimal requiring no shuffling between iterations of a pipelined processor. In applying these factorizations the successive iterations are effected on successive sub-images such that after $\log_p N$ stages the transform image Y is pipelined at the processor output. Applications include real-time processing of video signals.

The Discrete Fourier transform matrix for N points is the matrix F_N defined above in (10) with p replaced by N and the factor $1/\sqrt{p}$ optional:

$$F_N = \begin{bmatrix} w^0 & w^0 & \cdots & w^0 \\ w^0 & w^1 & \cdots & w^{N-1} \\ w^0 & w^2 & \cdots & w^{2(N-1)} \\ w^0 & w^{N-1} & \cdots & w^{(N-1)^2} \end{bmatrix} \tag{40}$$

For images the factorization leads to the optimal form

$$Y_F = \left\{ \prod_{i=0}^{n-1} F_i \right\} \times \left\{ \prod_{k=0}^{n-1} F_{n-k-1} \right\} \tag{41}$$

and for unidimensional signals the corresponding form for the Discrete Fourier matrix is

$$F_N = \prod_{i=0}^{n-1} (F_i) \tag{42}$$

$$\begin{aligned} F_i &= U_i C_i \\ C_i &= C J_{i+1}, \quad i = 0, 1, \ldots, n-1 \\ C_{n-1} &= C \end{aligned} \tag{43}$$

$$\begin{aligned} U_1 &= I_N \\ U_i &= I_{p^{n-i-1}} \times D_{p^{i+1}} = I_{p^{n-i-1}} \times D_{N/p^{n-i-1}} \\ D_{N/m} &= diag\left(I_{N/(pm)}, K_m, K_m^2, \ldots, K_m^{p-1}\right) \\ K_t &= diag\left(w^0, w^t, \ldots, w^{[N/(mp)-1]t}\right). \end{aligned} \tag{44}$$

Perfect shuffle hypercube transformations

The hypercube transformations approach is illustrated using the important matrices of the Chrestenson generalised Walsh–Paley (CGWP), generalised Walsh–Kaczmarz (CGWK) and the Discrete Fourier transforms.

We note that the matrices C_k in the Discrete Fourier transform expansion are closely related to the matrices J_i and H_i in the Chrestenson generalised Walsh Paley factorization. In fact the following relations are readily established:

$$\begin{aligned} C_N &\triangleq C \\ C_i &= C\,J_{i+1} = C\,H_{n-i-2} = Q_i \end{aligned} \tag{45}$$

where the equality $\triangleq$ sign means equal by definition.

$$Q_{n-1} = C_{n-1} = C. \tag{46}$$

Therefore, the CGWP matrices Q_i are the same as the C_i matrices defined above and have the same structure as the F_i matrices in the Fourier matrix factorization. Writing

$$B_k = CH_k \tag{47}$$

$$H_k = I_{p^k} \times P_{p^{n-k}} \tag{48}$$

the post-multiplication by H_k has the effect of permuting the columns of C so that at row w,

$$w \simeq \left(0\, j_{n-2} \ldots j_1\, j_0\right) \tag{49}$$

the pilot element is at column z as determined by the permutation H_k, that is,

$$z \simeq \left(j_k\, 0\, j_{n-2} \ldots j_{k+1}\, j_{k-1} \ldots j_1\, j_0\right) \tag{50}$$

with the special case $k = n-2$ producing

$$z \simeq \left(j_{n-2}\, 0\, j_{n-3} \ldots j_1\, j_0\right) \tag{51}$$

and that of $k = n-1$ yielding

$$z \simeq \left(0\, j_{n-2} \ldots j_1\, j_0\right). \tag{52}$$

Alternatively, we can write z directly as a function of w by using previously developed expressions of permutation matrices. For example,

$$B_0 = CH_0 = CP \tag{53}$$

and using the expression defining P, namely,

$$\left[P_{p^n}^k\right]_{uv} = \begin{cases} 1, & u = 0, 1, \ldots, p^n - 1, \\ & v = \left[u + \left(u \mod p^k\right)(p^n - 1)\right]/p^k \\ 0, & \text{otherwise} \end{cases} \qquad k = 0, 1, \ldots, N-1 \tag{54}$$

with $k = 1$, we can write

$$z = \left[w + (w \mod p)\left(p^n - 1\right)\right]/p \tag{55}$$

a relation that defines the pilot elements matrix.

Similarly,

$$B_1 = C\,H_1 = C\left(I_p \times P_{p^{n-1}}\right) \tag{56}$$

and from the definition given in Corinthios (1994):

$$\left[P_i^t\right]_{uv} = \begin{cases} 1, & u = 0, 1, \ldots, p^n - 1 \\ & v = p^{i-t \bmod (n-i)}[p^{-i}\left(u - u \mod p^i\right) + \{[p^{-i}(u - u \mod p^i)] \\ & \mod p^{t \bmod (n-i)}\}\left(p^{n-i} - 1\right)] + u \mod p^i \\ 0, & \text{otherwise} \end{cases} \tag{57}$$

with $i = 1$ and $t = 1$ we have

$$z = \left[p^{-1}(w - w \mod p) + \left\{\left[p^{-1}(w - w \mod p)\right] \mod p\right\}\left(p^{n-1} - 1\right)\right] + w \mod p. \tag{58}$$

Consider the permutation matrix

$$R_N = R_{p^n} = I_{p^m} \times P_{p^j} \times I_{p^k}. \tag{59}$$

Let the base-p hypercube describing the order in a vector x of $N = p^n$ elements be represented as the n-tuple.

$$x \simeq \left(j_{n-1} \ldots j_1 j_0\right)_p \quad j_i \in \{0, 1, \ldots, p-1\}. \tag{60}$$

The application of the matrix R_{p^n} on the n-tuple vector x, results in the n-tuple:

$$v = \left(j_{n-1} \ldots j_{n-k+1}\, j_{n-k}\, j_m\, j_{n-k-1} \ldots j_{m+2}\, j_{m+1}\, j_{m-1} \ldots j_1\, j_0\right). \tag{61}$$

We note that with respect to x the left k digits and the right m digits are left unchanged while the remaining digits are rotated using a circular shift of one digit to the right.

The pilot-elements matrix β_k corresponding to the matrix B_k is obtained by restricting the values of w (and hence the corresponding z values) to $w = 0, 1, \ldots, p^{n-1} - 1$.

Moreover, we note that if we write

$$L_i = P^{-1}\,G_i = P^{n-1}\,G_i \tag{62}$$

and note that G_i is similar in structure to C_N, we have

$$z = \left[w + \left(w \mod p^k\right)\left(p^n - 1\right)\right]/p^k \tag{63}$$

with $k = n - 1$.

To obtain the pilot elements matrix λ_i corresponding to L_i we write

$$z' = z \mod p^{n-1} \tag{64}$$

in order to reveal all satellite elements accompanying each pilot element. We then eliminate all the repeated entries in z' and the corresponding w values, retaining only pilot elements positions. Alternatively we simply force to zero the digit of weight $n - 2$ in w and that of weight $n - 1$ in z.

The CGWP factorization

We presently focus our attention on the matrices

$$B_k = C\,H_k; \quad k = 0, 1, \ldots, n-1. \tag{65}$$

In evaluating the pilot elements coordinates we begin by setting the number of processors $M = 1$. The corresponding $w - z$ relation of the pilot elements are thus evaluated with $m = 0$. Once this relation has been established it is subsequently used as the reference "$w - z$ conversion template" to produce the pilot element positions for a general number of $M = p^m$ processors. A "right" scan is applied to the matrix in order to produce the $w - z$ template with an ascending order of w. In this scanning type the algorithm advances the first index w from zero selecting pilot elements by evaluating their displacement to the right as the second index z. Once the template has been evaluated the value m corresponding to the number of processors to be dispatched is used to perform successive p-ary divisions in proportion to m to assign the M processors with maximum spacing, leading to maximum possible lengths of memory queues. A "down" scan is subsequently applied, where p-ary divisions are applied successively while proceeding downward along the matrix columns, followed by a selection of the desired optimal scan.

The template evaluation and subsequent p-ary divisions for the assignment of the M processors through a right type scan produce the following hypercube assignments. The assignments are as expected functions of the four variables n, p, k and m. The conditions of validity of the different assignments are denoted by numbers and letters for subsequent referencing. With K denoting the main clock, the following hypercube transformations are obtained

$$\begin{aligned} K &\simeq \left(j_{n-1} \ldots j_{m+1} j_m i_{m-1} \ldots i_1 i_0\right)_p \\ K_{\overline{n-1}} &\simeq \left(0 j_{n-2} \ldots j_{m+1} j_m i_{m-1} \ldots i_1 i_0\right)_p \\ K_{\overline{n-2}} &\simeq \left(j_{n-1} 0 j_{n-3} \ldots j_{m+1} j_m i_{m-1} \ldots i_1 i_0\right)_p \end{aligned} \tag{66}$$

1. $k < n - 2$

   (a) x: $m = 0$

   $$w \simeq K_{\overline{n-1}} \tag{67}$$

   $$z \simeq \left[\left(I_{p^k} \times P_{p^{n-k}}\right) K\right]_{\overline{n-2}} \tag{68}$$

   (b) y: $1 \le m \le n - k - 2$

$$w \simeq \left[\left(P_{p^{k+1}} \times I_{p^{n-k-1}}\right) \prod_{t=1}^{m-1} \left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right) K\right]_{\overline{n-1}} \tag{69}$$

$$z \simeq \left[P_{p^n} \prod_{t=1}^{m-1} \left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right) K\right]_{\overline{n-2}} \tag{70}$$

(c) $z: n-k-1 \le m \le n-1$

$$w \simeq \left[\left(P_{p^{k+1}} \times I_{p^{n-k-1}}\right) \prod_{t=1}^{m-1} \left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right) K\right]_{\overline{n-1}} \tag{71}$$

$$z \simeq \left[P_{p^n} \prod_{t=1}^{m-1} \left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right) K\right]_{\overline{n-2}} \tag{72}$$

2. $k = n-2$

(a) $u: m = 0$

$$\begin{aligned} w &\simeq K_{\overline{n-1}} \\ z &\simeq \left[\left(I_{p^{n-2}} \times P_{p^2}\right) K\right]_{\overline{n-2}} \end{aligned} \tag{73}$$

(b) $v: m \ge 1$

$$w \simeq \left[\prod_{t=0}^{m-1} \left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right) K\right]_{\overline{n-1}} \tag{74}$$

$$z \simeq \left[P_{p^n} \prod_{t=1}^{m-1} \left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right) K\right]_{\overline{n-2}} \tag{75}$$

$t: k = n-1$

$$w = z \simeq \left[\prod_{t=0}^{m-1} \left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right) K\right]_{\overline{n-1}} \tag{76}$$

Evaluated, these hypercubes yield the following pilot elements assignments:

$x: (k < n-2, \quad m = 0)$

$$w = \sum_{j=0}^{n-2} p^t j_t \tag{77}$$

$$z = \sum_{j=0}^{k-1} p^t j_t + p^{n-1} j_k + \sum_{t=k+1}^{n-2} p^{t-1} j_t \tag{78}$$

$y: k < n-2, \quad 1 \le m \le n-k-2$

$$w = p^k i_0 + \sum_{s=1}^{m-1} p^{n-1-s} i_s + \sum_{t=m}^{m+k-1} p^{t-m} j_t + \sum_{t=m+k}^{n-2} p^{t-m+1} j_t \tag{79}$$

$$z = p^{n-1} i_0 + \sum_{s=1}^{m-1} p^{n-2-s} i_s + \sum_{t=m}^{n-2} p^{t-m} j_t \tag{80}$$

$z: k < n-2, \quad n-k-1 \le m \le n-1$

$$w = p^k i_0 + \sum_{s=1}^{n-k-2} p^{n-1-s} i_s + \sum_{s=n-k-1}^{m-1} p^{n-2-s} i_s + \sum_{s=m}^{n-2} p^{t-m} j_t \tag{81}$$

$$z = p^{n-1} i_0 + \sum_{s=1}^{m-1} p^{n-2-s} i_s + \sum_{t=m}^{n-2} p^{t-m} j_t \tag{82}$$

$u: k = n-2, \quad m = 0$

$$w = \sum_{t=0}^{n-2} p^t j_t \tag{83}$$

$$z = \sum_{j=0}^{n-3} p^t j_t + p^{n-1} j_{n-2} \tag{84}$$

$v: k = n-2, \quad m \ge 1$

$$w = \sum_{s=0}^{m-1} p^{k-s} i_s + \sum_{t=m}^{n-2} p^{t-m} j_t \tag{85}$$

$$z = p^{n-1} i_0 + \sum_{s=1}^{m-1} p^{k-s} i_s + \sum_{t=m}^{n-2} p^{t-m} j_t \tag{86}$$

$t: k = n-1$

$$w = z = \sum_{s=0}^{m-1} p^{n-2-s} i_s + \sum_{t=m}^{n-2} p^{t-m} j_t. \tag{87}$$

Row and column scans for optimal assignment

A processor is considered optimal if it requires a minimum of memory partitions, is shuffle free, meaning the absence of clock times used uniquely for shuffling, and produces an ordered output given an ordered input. It is shown in Corinthios (1994) that p^2-optimal algorithms and processors lead to a minimum number of p^2 partitions of

N/p^2 queue length each. With $M = p^m$ base-p processors operating in parallel the number of partitions increases to p^{m+2} and the queue length of each partition reduces to N/p^{m+2}.

An optimal multiprocessing algorithm should satisfy such optimality constraints. The horizontal spacing between simultaneously accessed pilot elements defines the input memory queue length. The vertical spacing defines the output memory queue length. With M processors applied in parallel the horizontal spacing between the accessed elements will be referred to as the "input pitch", while the vertical spacing as the "output pitch".

By choosing the pilot elements leading to the maximum possible pitch, which is the highest of the two values: the minimum input pitch and minimum output pitch, optimality in the form of N/p^{m+2} queue length is achieved.

We note that optimal minimum memory queue length MMQL satisfies

$$MMQL = \begin{cases} p^{n-m-2}, & m \le n-2 \\ 1, & m = n-1 \end{cases}$$

The following algorithm, Algorithm 2, describes this approach to state assignment

Algorithm 2 : Optimality search
begin
Extract pilots matrix
Apply right scan
Evaluate input pitch
Evaluate output pitch
$p_{i,\min} = \min[\text{input pitch}]$
$p_{o,\min} = \min[\text{output pitch}]$
$p_{r,\min} = \min[p_{i,\min}, p_{o,\min}]$
Apply down scan
Evaluate output pitch
$p_{i,\min} = \min[\text{input pitch}]$
$p_{o,\min} = \min[\text{output pitch}]$
$p_{d,\min} = \min[p_{i,\min}, p_{o,\min}]$
Optimal pitch $= \max[p_{d,\min}, p_{r,\min}]$
If $p_{r,\min} \ge p_{d,\min}$ then optimal = right scan Else optimal = down scan
Apply hypercube transformations
Dispatch and sequence M processors
end

optimality.

In following the algorithm we note that in the validity condition y of the B_k matrix $y : 1 \le m \le n-k-2$ the results obtained are such that the digit i_0 of w is of a weight p^k. Hence the input pitch is p^k while the output pitch which can be deduced from the position of i_0 in z is p^{n-1}, that is, maximal possible. The input pitch is thus function of k and can be low if k is small. By performing a down scan of B_k we obtain the following solution:

$k < n-2$

$y: 1 \le m \le n-k-2$

$w: 0 \quad i_0 \quad i_1 \quad \ldots \quad i_{m-1} \quad j_{n-2} \quad \ldots \quad j_{m+1} \quad j_m$

$z: j_{m+k} \quad 0 \quad i_0 \quad i_1 \quad \ldots \quad i_{m-1} \quad j_{n-2} \quad \ldots \quad j_{m+k+1} \quad j_{m+k-1} \quad \ldots \quad j_{m+1} \quad j_m$

where now it is i_{m-1} that leads to a minimum pitch and it has a weight of p^{n-m-1} in w and p^{n-m-2} in z. We deduce that the minimum pitch in this solution is p^{n-m-2}, which is the optimal sought. The same reasoning leads to the optimal assignment for the case

$$k < n-2$$

$z: n-k-1 \le m \le n-1$

$w: 0 \quad i_0 \quad i_1 \quad \ldots \quad \ldots \quad i_{m-1} \quad j_{n-2} \quad \ldots \quad j_{m+1} \quad j_m$

$z: i_{n-2-k} \quad 0 \quad i_0 \quad i_1 \quad \ldots \quad i_{n-3-k} \quad i_{n-1-k} \quad i_{n-k} \quad \ldots \quad i_{m-1} \quad j_{n-2} \quad \ldots \quad j_{m+1} \quad j_m$

These are the only two cases of the matrix that need be thus modified for optimality. All results obtained above for the other validity conditions can be verified to be optimal.

Matrix span

In the above from one iteration to the next the value of k is incremented. In each iteration once the pilot element matrix coordinates (w, z) are determined as shown above each processor accesses p elements spaced by the row span starting with the pilot element and writes its p outputs at addresses spaced by the column span. The row and column spans of a matrix are evaluated as is shown in Corinthios (1994). In particular we note that the matrix

$$B_k = CH_k \tag{88}$$

has the same column span as that of C, namely $\sigma_c(B_k) = \sigma_c(C) = p^{n-1}$. The row span of B_k is evaluated by noticing that B_k has the same structure as C with its columns permuted in accordance with the order implied by

$$H_k^{-1} = I_{p^k} \times P_{p^{n-k}}^{-1} \tag{89}$$

The transformation of the hypercube $(i_{n-1}...i_1 i_0)$ corresponding to H_k^{-1} is one leading to a most significant digit equal to i_{n-2}. Since this digit changes value from 0 to 1 in a cycle length of p^{n-2} we deduce that the row span of all the B_k matrices is simply

$$\sigma_R(B_k) = p^{n-2}. \tag{90}$$

Each processing element thus accesses p operands spaced p^{n-2} points apart and writes their p outputs at points which are p^{n-1} points apart.

The CGWK factorization

The sampling matrices of the CGWK factorization are more complex in structure than the other generalised spectral analysis matrices. They are defined by

$$\Gamma_i = P^{-1} G_i S_{i+1} \tag{91}$$

Let

$$L_i \triangleq P^{-1} G_i \tag{92}$$

we have

$$\Gamma_i = L_i\, S_{i+1}. \tag{93}$$

We note that the sampling matrix G_i has the same structure in poles and zeros, that is, in the positions of non-zero and zero elements respectively, as that of the matrix C_N. We can write for the matrix G_i

$$\begin{aligned} w_{G_i} &\simeq \left(j_{n-2}\ \ldots j_1\, j_0\right) \\ z_{G_i} &\simeq \left(j_{n-2} \ldots j_1\, j_0\right) \end{aligned} \tag{94}$$

as the pilot elements positions.

Given the definition of the matrix L_i a hypercube rotation corresponding to the matrix P^{-1} would yield the w and z values of L_i as:

$$\begin{aligned} w_{L_i} &\simeq \left(j_{n-2}\, 0\, j_{n-3}\ \ldots j_1\, j_0\right) \\ z_{L_i} &= P^{-1}\, w_{L_i} \simeq \left(0\, j_{n-3} \ldots j_1\, j_0\, j_{n-2}\right) \end{aligned} \tag{95}$$

Alternatively, a z-ordered counterpart can be written as:

$$\begin{aligned} z_{L_i} &\simeq \left(0\, j_{n-2} \ldots j_i\, j_0\right) \\ w_{L_i} &\simeq \left(j_0\, 0\, j_{n-2} \ldots j_2\, j_1\right) \end{aligned} \tag{96}$$

Similarly, the matrix $\Gamma_0 = G_0 S_1$ which is obtained from G_0 by permuting its columns according to the order dictated by

$$S_1^{-1} = P_{p^{n-1}}^{-1} \times I_p \tag{97}$$

leads to the $m = 0$ template assignment

$$w_{\Gamma_0} \simeq \left(0\, j_{n-2} \ldots j_1\, j_0\right) \tag{98}$$

$$z_{\Gamma_0}\ = S_1\, w_{\Gamma_0} \simeq \left(0\, j_0\, j_{n-2} \ldots j_2\, j_1\right) \tag{99}$$

and a similar z-ordered state assignment counter part.

For

$$\Gamma_k = G_0 S_k\,, \quad k > 0 \tag{100}$$

we have

$$S_k^{-1} = I_{p^{k-1}} \times P_{p^{n-k}}^{-1} \times I_p \tag{101}$$

which leads to the state template assignment

$$\begin{aligned} w_{\Gamma_k} &\simeq w_{L_i} \simeq \left(j_{n-2}\, 0\, j_{n-3} \ldots j_1\, j_0\right), \\ z_{\Gamma_k} &= S_{k+1}\, z_{L_i} \simeq \left(0\, j_{k-1}\, j_{n-3}\ \ldots j_{k+1}\, j_k\, j_{k-2} \ldots j_1\, j_0\, j_{n-2}\right); \quad k > 0 \end{aligned} \tag{102}$$

With m made variable a right scan yields the following expressions for the different validity conditions.

The Γ_k transformations

1. $k = 0$

$a: k = 0, m = 0$

$$w \simeq K_{\overline{n-1}}$$

$$z \simeq P_{p^n} K_{\overline{n-1}} \equiv \left[\left(P_{p^{n-1}} \times I_p\right)K\right]_{\overline{n-1}} \tag{103}$$

$b: k = 0, m \geq 2$

$$w \simeq \left[\prod_{t=1}^{m-1}\left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right)K\right]_{\overline{n-1}} \tag{104}$$

$$z \simeq \left[\prod_{t=0}^{m-1}\left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right)K\right]_{\overline{n-1}} \tag{105}$$

2. $1 \leq k \leq n-3$

$c: m = 0$

$$w \simeq \left[\left(I_{p^{n-2}} \times P_{p^2}\right)K\right]_{\overline{n-2}} \tag{106}$$

$$z \simeq \left[\left(I_{p^k} \times P_{p^{n-k-1}} \times I_p\right)\left(P^{-1}_{p^{n-1}} \times I_p\right)K\right]_{\overline{n-1}} \tag{107}$$

$d: m = 1$

$$w \simeq \left[\left(I_{p^{n-2}} \times P_{p^2}\right)\left(P_{p^k} \times I_{p^{n-k}}\right)K\right]_{\overline{n-2}} \tag{108}$$

$$z \simeq \left[\left(I_p \times P_{p^{n-2}} \times I_p\right)\left(P^{-1}_{p^{n-1}} \times I_p\right)K\right]_{\overline{n-1}} \tag{109}$$

$e: m \geq 2$

$$z \simeq \left[\left(P_{p^{n-1}} \times I_p\right)\prod_{t=2}^{m-1}\left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right)K\right]_{\overline{n-1}} \tag{110}$$

(a) $m \geq n-k$

$$w \simeq \left[\left(P_{p^k} \times I_{p^{n-k}}\right)\prod_{t=1}^{m-1}\left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right)K\right]_{\overline{n-2}} \tag{111}$$

(b) $2 \leq m \leq n-k$

$$w \simeq \left[\left(P_{p^k} \times I_{p^{n-k}}\right)\prod_{t=1}^{m-1}\left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right)K\right]_{\overline{n-2}} \tag{112}$$

3. $k \geq n-2$

$$w \simeq \left[\left(I_{p^{n-2}} \times P_{p^2}\right)K\right]_{\overline{n-2}} \tag{113}$$

$$z \simeq \left[\left(P^{-1}_{p^{n-1}} \times I_p\right)K\right]_{\overline{n-1}} \tag{114}$$

$g: m = 1$

$$w \simeq \left[\left(I_{p^2} \times P_{p^{n-2}}\right)\left(P_{p^{n-2}} \times I_{p^2}\right)K\right]_{\overline{n-2}} \tag{115}$$

$$z \simeq \left[\left(P^{-1}_{p^{n-2}} \times I_{p^2}\right)\left(P_{p^{n-1}} \times I_p\right)K\right]_{\overline{n-1}} \tag{116}$$

$h: m \geq 2$

$$w \simeq \left[\left(P_{p^{n-2}} \times I_{p^2}\right)\prod_{t=1}^{m-1}\left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right)K\right]_{\overline{n-2}} \tag{117}$$

$i: 2 \leq m \leq n-2$

$$z \simeq \left[\left(P_{p^{n-1}} \times I_p\right)\prod_{t=2}^{m-1}\left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right)K\right]_{\overline{n-1}} \tag{118}$$

$j: m = n-1$

$$z \simeq \left[\left(P_{p^{n-1}} \times I_p\right)\prod_{t=2}^{m-1}\left(I_{p^t} \times P_{p^{n-t-1}} \times I_p\right)K\right]_{\overline{n-1}} \tag{119}$$

CGWK optimal assignments

A "down" scan of the Γ_k matrix yields optimal assignments for two validity conditions:

1. $k = 0$

$a: k = 0, \quad m = 1$

$$\begin{array}{llllllll} w: 0 & i_0 & j_{n-2} & \ldots & j_2 & j_1 \\ z: 0 & j_1 & i_0 & j_{n-2} & \ldots & j_3 & j_2 \end{array}$$

$b: k = 0, \quad m \geq 2$

$$\begin{array}{llllllllll} w: 0 & i_0 & i_1 & \ldots & i_{m-1} & j_{n-2} & \ldots & j_{m+1} & j_m \\ z: 0 & j_m & i_0 & i_1 & \ldots & i_{m-2} & j_{n-2} & \ldots & j_{m+1} \end{array}$$

All other assignments generated by the "right" scan are optimal and need not be replaced.

The CGWK matrix spans

Using the same approach we deduce the spans of the different CGWK factorization matrices. We have

$$\sigma_R(L_i) = \sigma_R(G_i) = p^{n-1} \tag{120}$$

$$\sigma_c(L_i) = p^{n-2} \tag{121}$$

$$\sigma_R(\Gamma_0) = p^{n-1} \tag{122}$$

$$\sigma_c(\Gamma_0) = \sigma_c(G_0) = p^{n-1} \tag{123}$$

and

$$\sigma_R(\Gamma_i) = p^{n-1} \tag{124}$$

$$\sigma_c(\Gamma_i) = \sigma_c\left(P^{-1}G_i\right) = \sigma_c(L_i) = p^{n-2}. \tag{125}$$

FPGA configuration

Configuring FPGAs to execute digital signal processing algorithms in real time has been rendered readily accessible through model simulation using Matalb© and Simulink. In what follows we summarize results obtained in configuring Xilinx FPGA boards and particular the Artix-7 Nexys 4 DDR platform. In these applications the basic Discrete Chrestenson transform matrices with $M = 1$, $p = 2$ and $n = 5$ defining 32-point transforms both as the Discrete Fourier transforms and Walsh–Hadamard transforms are presented. In both cases the transform of ramp is evaluated.

Figure 1 shows the waveforms which appear in the successive iterations and the final result in the case of the evaluation of the CGW Discrete Fourier transform.

The corresponding waveforms in the case of the CGWP Walsh–Hadamard transform successive iterations and the final result are shown in Fig. 2.

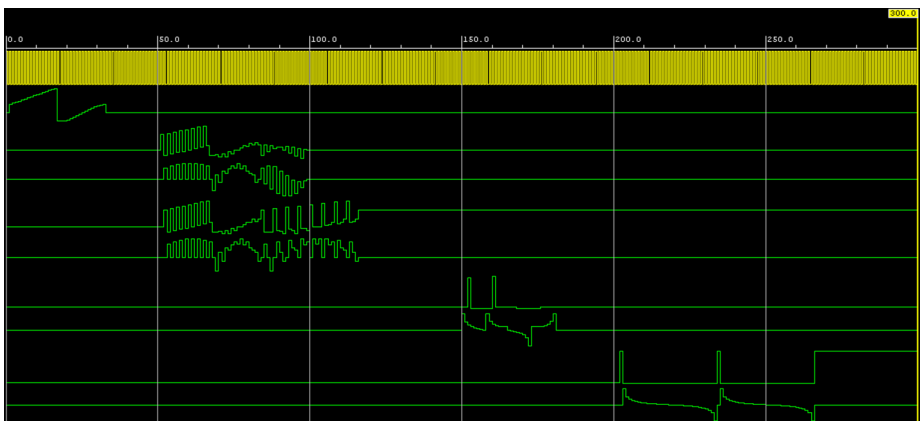

Fig. 1 Waveforms of successive iterations and the final result in the case of the evaluation of the CGW discrete Fourier transform

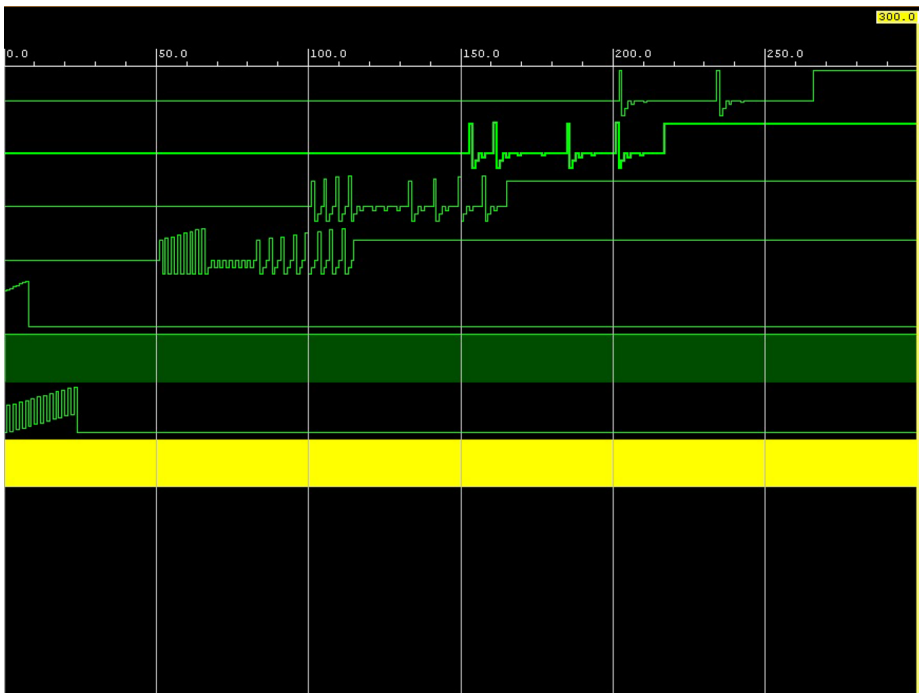

Fig. 2 Waveforms of successive iterations and the final result in the case of the evaluation of the CGWP Walsh–Hadamard transform

Conclusion

A formalism and an algorithm for the parallel implementation of the Chrestenson transform employing rotations of a general-base hypercube and their embedding into FPGA architectures has been presented. Closed-form general-radix factorizations of the transformation matrices, showing processor architecture and sequencing of an arbitrary number $M = p^{n-1}$ of general-base processors have been obtained. Pilot elements addresses and matrix spans to locate their satellites are automatically generated for dispatching and sequencing the parallel processors.

Acknowledgements
The information technology support of Saad Chidami in the process of transferring simulation models to FPGA platform is greatly appreciated. The author wishes to acknowledge the research grant received from the National Science and Engineering Council of Canada NSERC.

Competing interests
The author declare that he has no competing interests.

Received: 26 August 2015 Accepted: 25 August 2016
Published online: 08 September 2016

References

Bespalov MS (2010) Discrete Chrestenson transform. Prob Inf Transm 46(4):353–375
Corinthios MJ (1985) 3-D cellular arrays for parallel/cascade image/signal processing. In: Karpovsky M (ed) Spectral techniques and fault detection. Academic Press, New York

Corinthios M (1994) Optimal parallel and pipelined processing through a new class of matrices with application to generalized spectral analysis. IEEE Trans Comput 43(4):443–459
Corinthios MJ (2009) Signals, systems, transforms and digital signal processing with Matlab©. Taylor and Francis/CRC Press, London
Harmut F, Sadrozinski W, Wu J (2010) Applications of field programmable gate arrays in scientific research. Taylor and Francis, London
Huda S, Anderson JH, Tamura H (2014) Optimizing effective interconnect capacitance for FPGA power reduction. In: 22nd ACM/SIGDA international symposium on field programmable gate arrays, Monterey, CA, 26–28 Feb 2014