A Parallel Architecture for Radix-2 Fast Hartley Transform/Real-valued Fast Fourier Transform*

Ph. Philipov, N. Tasheva

Key Words: *Parallel FHT; parallel FFT; high-performance computer architectures.*

Abstract. *Fast Hartley transform (FHT) /Real-valued Fast Fourier transform (RFFT) algorithms are important Fourier-related transforms, because they lower twice the operational and memory requirements when input data is real-valued. However, these types of algorithms, have irregular computational structure, which makes their parallel implementation a difficult tas The aim of this paper is to show that these algorithms have the same potential for parallel implementation as the complex fast Fourier transform (FFT), as well as they share common natural architectures with FFT, based on the perfect shuffle permutation method. Presented is a processor array architecture based on the „indirect hypercube" concept which is suitable for realization of fast FHT/RFFT processors for real time applications.*

Introduction

Fast Fourier transform (FFT) is an efficient algorithm to compute the discrete Fourier transform (DFT) and its inverse. FFT plays significant role in several important scientific and technical areas as solving of partial differential equations, spectral analysis, digital signal processing, image processing etc.

In many applications, the input data for the DFT are real-valued and outputs exhibit conjugational symmetry. Specialized real-valued FFT algorithms (RFFT) have been designed for this situation [Sorensen-1] to remove the redundant operations, lowering twice memory and operational requirements.

The discrete Hartley transform (DHT) [Bracewell-1] is a Fourier-related transform of discrete, periodic data, similar to DFT with analogous applications in signal processing and related fields. Its main distinction from the DFT is that it transforms real data, thus being an alternative for the real-valued DFT algorithms.

The fast Hartley transform (FHT) was first described by Bracewell in 1984 [Bracewell-2]. It performs much faster because it requires only real arithmetic computations compared to the complex arithmetic computations required by the FFT thus in theory the FHT like RFFTneeds only half the computer memory and less than half of computation time in comparison with the complex FFT.

FHT/RFFT algorithms are important because with their help other *Fourier-related* transforms, e.g. discrete cosine transforms (DCT), discrete sine transforms (DST) etc. can be obtained.

In many cases, from performance viewpoint, a parallel implementation of these algorithms is necessary. Parallel implementations of FHT/RFFT have been made for systolic arrays [Marchesi-1] [Chang-1], hypercuboid multicomputers [Zapata-1], as well as VLSI implementations [Zapata-2] [Liu-1]. Nevertheless, the literature concerning the parallelization of FHT/RFFT is not abundant. These algorithms have irregular computational structure which makes their efficient parallelization a very difficult task. Thus, in practice, it is often easier to obtain high performance in such cases with standard FFT-based algorithms.

It is known that parallelism is an intrinsic property of FFT. Natural architectures for a given algorithm are such architectures, which are derived directly from the data flow of this algorithm. For this reason, they are considered to be the most appropriate architectures for parallel realization of this algorithm. For FFT, two types of natural architectures - the *direct* and *indirect* hypercube are known.

There are some publications which show that the *direct* hypercube can be considered to be a natural architecture for FHT/RFFT algorithms [Zapata-2], [Arguello-1].

The aim of this paper is to show that the *indirect* hypercube can also be considered to be a natural architecture for FHT/RFFT algorithms, thus showing that the parallelism is an intrinsic property of FHT/RFFT, as well as that FHT/RFFT share with FFT the same types of natural architectures. Presented is a processor array architecture based on the „indirect hypercube" concept, which is suitable for realization of fast FHT/RFFT processors for real time applications.

FHT Description

DHT is defined according to the following formula:

$$(1)\quad H_k = \sum_{n=0}^{N-1} X_n[cos(\frac{2\pi}{N}nk) + sin(\frac{2\pi}{N}nk)],\ k = 0,...,N-1.$$

Sometimes *cos(z) + sin(z)* is denoted as *cas(z)*.

Generally, FHT is based on the so called *Danielson-Lanczos lemma* [Danielson-1] - the base of the Cooley - Tukey radix-2 FFT algorithm.

FHT splits the N-point sequence X(n) into two smaller (N/2) - point sequences $X_0(n)$ (even) and $X_1(n)$ (odd). The two sequences are Hartley transformed into $Xh_0(k)$ and $Xh_1(k)$ and combined into $X_h(k)$ in accordance with the next relations:

$$(2)\ X_h(k)=\begin{cases} Xh_{0(k)}+\cos((2\pi/N)k)Xh_{1(k)}+\sin((2\pi/N)k)Xh_{1(N/2-k)}, \\ \text{for } 0\le k \le N/2-1 \\ Xh_{0(k-N/2))} - \cos((2\pi/N)(k-N/2)\ Xh_{1(k-N/2))} - \\ \sin((2\pi/N)(k-N/2)\ Xh_{1(N-k)}, \text{ for } N/2 \le k \le N-1 \end{cases}$$

Equations (2) result in the so called *double FHT butterfly* (DFHTB)[Ulman-1] which is of two types.

Type A DFHTB (k = 0)

$$(3a)\quad \begin{aligned} X_{h(0)} &= X_{h0(0)} + X_{h1(0)} \\ X_{h(N/4)} &= X_{h0(N/4)} + X_{h1(N/4)} \\ X_{h(N/2)} &= X_{h0(0)} - X_{h1(0)} \\ X_{h(3N/4)} &= X_{h0(N/4)} - X_{h1(N/4)} \end{aligned}$$

* *This work is supported by the project □□□□ 02/1.*

Type B DFHTB ($0 < k \le N/4-1$)

(3b) $X_{h(k)} = X_{h0(k)} + \cos((2\pi/N)k).X_{h1(k)} + \sin((2\pi/N)k).X_{h1((N/2)-k)}$

$X_{h((N/2)-k)} = X_{h0((N/2)-k)} - \cos((2p/N)k).X_{h1((N/2)-k)} + \sin((2p/N)k).X_{h1(k)}$

$X_{h(k+N/2)} = X_{h0(k)} - \cos((2\pi/N)k).X_{h1(k)} - \sin((2\pi/N)k).X_{h1((N/2)-k)}$

$X_{h(N-k)} = X_{h0((N/2)-k)} + \cos((2\pi/N)k).X_{h1((N/2)-k)} - \sin((2\pi/N)k).X_{h1(k)}$

Real-valued FFT

The Discrete Fourier Transform (DFT) (direct and inverse) is defined as:

(4a)

(4b) $X(n) = \frac{1}{N}\sum_{k=0}^{N-1} F(k)\, W^{k\,n}$, where

$W = e^{i2\pi/N} = \cos(2\pi/N) + i\sin(2\pi/N)$.

When the initial sequence $X(n)$ is real-valued, we have

a) For $k = 0$, $F(0)$ and $F(N/2)$ are real numbers,

b) For $0 < k < N/2$, $F_{(N-k)} = F_{(k)}^*$ (conjugated complex numbers), and

(5) $$F(k)imag = \sum_{n=0}^{N-1} X(n).Sin(\frac{2\pi}{N})kn$$

$$F(N-k)real = \sum_{n=0}^{N-1} X(n).Cos(\frac{2\pi}{N})kn \quad F(N-k)imag = -\sum_{n=0}^{N-1} X(n).Sin(\frac{2\pi}{N})kn$$

From (5) it is very well seen how the information in this case is doubled.

As it was mentioned, efficient FFT algorithms have been designed for this situation. One approach consists in taking an ordinary FFT algorithm (e.g. Cooley-Tukey) and removing the redundant parts of the computation [Bergland-1]. Alternatively, it is possible to express an *even*-length real-input DFT as a complex DFT of half the length, followed by O(*N*) post-processing operations.

We examine the first approach.

The *simplified* butterfly for radix-2 DIT FFT (Cooley-Tukey) operates as follows:

(6) $X_{f(k)} = X_{f0(k)} + e^{-i(2\pi/N)k}.X_{f1(k)}$

$X_{f(N/2+k)} = X_{f0(k)} - e^{-i(2\pi/N)k}.X_{f1(k)}$, $0 \le k \le N/2 - 1$.

For $k = 0$ and $N/4$, inputs are real numbers and we have the following real system:

$X_{f(0)real} = X_{f0(0)real} + X_{f1(0)real}$

$X_{f(0)imag} = 0$

(7a) $X_{f(N/2)real} = X_{f0(0)real} - X_{f1(0)real}$

$X_{f(N/2)imag} = 0$

$X_{f(N/4)real} = X_{f0(N/4)real}$

$X_{f(N/4)imag} = - X_{f1(N/4)real}$

$X_{f(3N/4)real} = X_{f0(N/4)real}$

$X_{f(3N/4)imag} = X_{f1(N/4)real}$

For $0 \le k \le N/4 - 1$, combining symmetrical butterflies, we have the following real system:

(7b) $X_{f(k)real} = X_{f0(k)real} + \cos((2\pi/N)k).X_{f1(k)real} + \sin((2\pi/N)k).X_{f1(k)imag}$

$X_{f(k)imag} = X_{f0(k)imag} + \cos((2\pi/N)k).X_{f1(k)imag} - \sin((2\pi/N)k).X_{f1(k)real}$

$X_{f(N/2+k)real} = X_{f0(k)real} - \cos((2\pi/N)k).X_{f1(k)real} - \sin((2\pi/N)k).X_{f1(k)imag}$

$X_{f(N/2+k)imag} = X_{f0(k)imag} - \cos((2\pi/N)k).X_{f1(k)imag} + \sin((2\pi/N)k).X_{f1(k)real}$

$X_{f(N/2-k)real} = X_{f0(N/2-k)real} - \cos((2\pi/N)k).X_{f1(N/2-k)real} + \sin((2\pi/N)k).X_{f1(N/2-k)imag}$

$X_{f(N/2-k)imag} = X_{f0(N/2-k)imag} - \cos((2\pi/N)k).X_{f1(N/2-k)imag} - \sin((2\pi/N)k).X_{f1(N/2-k)real}$

$X_{f(N-k)real} = X_{f0(N/2-k)real} + \cos((2p/N)k).X_{f1(N/2-k)real} - \sin((2\pi/N)k).X_{f1(N/2-k)imag}$

$X_{f(N-k)imag} = X_{f0(N/2-k)imag} + \cos((2\pi/N)k).X_{f1(N/2-k)imag} + \sin((2\pi/N)k).X_{f1(N/2-k)real}$

Removing redundances in (7a) and (7b), and making the following substitutions:

$X_{f(0)} = X_{f(0)real}$	$X_{f0(0)} = X_{f0(0)real}$
$X_{f(N/2)} = X_{f(N/2)real}$	$X_{f1(0)} = X_{f1(0)real}$
$X_{f(N/4)} = X_{f(N/4)real} = X_{f(3N/4)real}$	$X_{f0(N/4)} = X_{f0(N/4)real}$
$X_{f(3N/4)} = X_{f(N/4)imag} = - X_{f(3N/4)imag}$	$X_{f1(N/4)} = X_{f1(N/4)real}$
$X_{f(k)} = X_{f(k)real} = X_{f(N-k)real}$	$X_{f0(k)} = X_{f0(k)real} = X_{f0(N/2-k)real}$
$X_{f(N-k)} = X_{f(k)imag} = - X_{f(N-k)imag}$	$X_{f0(N/2-k)} = X_{f0(k)imag} = -X_{f0(N/2-k)imag}$
$X_{f(N/2+k)} = X_{f(N/2+k)real} = X_{f(N/2-k)real}$	$X_{f1(k)} = X_{f1(k)real} = X_{f1(N/2-k)real}$
$X_{f(N/2-k)} = X_{f(N/2+k)imag} = - X_{f(N/2-k)imag}$	$X_{f1(N/2-k)} = X_{f1(k)imag} = -X_{f1(N/2-k)imag}$

we get the final systems:

For $k = 0$ and $N/4$

(8a) $X_{f(0)} = X_{f0(0)} + X_{f1(0)}$

$X_{f(N/4)} = X_{f0(N/4)}$

$X_{f(N/2)} = X_{f0(0)} - X_{f1(0)}$

$X_{f(3N/4)} = - X_{f1(N/4)}$

For $0 < k \le N/4-1$

(8b) $X_{f(k)} = X_{f0(k)} + \cos((2\pi/2N)k).X_{f1(k)} + \sin((2\pi/2N)k).X_{f1(N/2-k)}$

$X_{f(N-k)} = X_{f0(N/2-k)} + \cos((2\pi/2N)k).X_{f1(N/2-k)} - \sin((2\pi/2N)k).X_{f1(k)}$

$X_{f(N/2+k)} = X_{f0(k)} - \cos((2\pi/2N)k).X_{f1(k)} - \sin((2\pi/2N)k).X_{f1(N/2-k)}$

$X_{f(N/2-k)} = X_{f0(N/2-k)} - \cos((2\pi/2N)k).X_{f1(N/2-k)} + \sin((2\pi/2N)k).X_{f1(k)}$

Comparison between systems (3.x) and (8.x) shows that (3b) and (8b) are equivalent (structurally and arithmetically). (8a) and (3a) are structurally equivalent and differ in arithmetics. Besides this, they are equivalent somehow to a standard simplified FFT butterfly operation. Despite differences, systems (3.x) and (8.x) presuppose the identical flow of data (whatever it be) of FHT and RFFT, and it is a reason FHT and RFFT to be considered as one algorithm with two versions.

It is not difficult to see from (3a) and (3b) that:

1. Inputs ($X_{h0(k)}$, $X_{h0((N/2)-k))}$ and ($X_{h1(k)}$, $X_{h1((N/2)-k)}$) form two pairs of signals that come from two butterflies from the previous stage;

2. Outputs ($X_{h(k)}$, $X_{h(N-k)}$) and ($X_{h((N/2)-k)}$, $X_{h(N/2+k)}$) form two pairs of signals that go to two butterflies from the next stage;

3. If we assume that inputs ($X_{h0(0)}$, $X_{h0(N/4)}$) and ($X_{h1(0)}$, $X_{h1(N/4)}$) form two pairs of signals that come from two butterflies from the previous stage, then outputs ($X_{h(0)}$, $X_{h(N/2)}$) and ($X_{h(N/4)}$, $X_{h(3N/4)}$) will form two pairs of signals that go to two butterflies from the next stage.

These ascertainments permit the double butterfly from (3.x) and (8.x) to be reguarded as a standard simplified FFT

butterfly of type (6), assuming that signals contain two components - P and N - which are composed in accordance with the following conventions:

1. For inputs, P components are the parts with $0 \le k < N/4$;
2. For outputs, P components are the parts with $0 \le k < N/2$;
3. For $k > 0$, the N complement is $N/2 - k$ for inputs and $N - k$ for outputs;
4. For $k = 0$, the N complement is $N/4$ for inputs and $N/2$ for outputs.

If we want to create with the help of the PN butterfly regular and easily parallelizable structures, it is necessary to answer the two following questions:

1. How the signals in the PN butterfly should be identified?
2. What are the properties, which signal identifiers should possess, that would guarantee intrinsic regularity and parallelism for the created structures?

Let us see what is the answer to these questions in the case of Cooley-Tukey FFT.

FFT Identifier Analysis

Radix-2 DIT (Cooley-Tukey) FFT, as FHT, is based on the already mentioned *Danielson-Lanczos lemma* - formaton of two N-point sequences ($X_{f(k)}$ and $X_{f(N/2+k)}$) on the base of two (even and odd) N/2-point sequences ($X_{f0(k)}$ and $X_{f1(k)}$).

Let $N = 2^n$.

The signal identifier is an n-bit binary number and contains two fields - *S* field and *K* field. *S* field is the sequence number, while *K* field identifies the parameter *k* (the transform parameter).

Sequence numbers are formed in the following manner. From an N-point sequence are formed two (N/2)- point sequences - even (0) and odd (1). From the even (N/2)- point sequence are formed two (N/4)-point sequences - even (00) and odd (10). From the odd (N/2)-point sequence are formed also two (N/4)-point sequences - even (01) and odd (11). This procedure continues up to the enumeration of all the points.

Signal identifiers are organized according to the following rules:

1. The *S* field is the most significant part of the signal identifier.
2. The *K* field is the least significant part of the signal identifier, and it is the bit-reversed value of the *k* parameter.

With every stage *K* field increases with one digit and the *S* field decreases with one digit. So, the initial identifiers contain only the *S* field (the point index), while the final identifiers contain only the *K* field (final results in bit-reversed form).

The following two properties of the signal identifiers are very important:

1. In a given butterfly enter two signals, whose identifiers are equal with the exception of their most significant digit;
2. The identifiers of the output signals of a given butterfly can be obtained by means of left cyclic rotation of the identifiers of the respective input signals.

These two properties identify the *perfect shuffle*. It is the *perfect shuffle* that ensures the perfect regularity and the intrinsic parallelism for the FFT flow of data [Stone-1].

Now, let us see one of the ways with the help of which the perfect shuffle ensures the intrinsic parallelism for FFT. We describe the concept of the *indirect hypercube*.

FFT Analysis

Usually, the flow of data of the radix-2 DIT FFT algorithm is presented with the help of the respective butterfly diagram. However, this method somehow hides the parallelism as an intrinsic feature of FFT. In [Phil-1] is described the method for dataflow presentation and analysis of radix-2 DIT FFT, based on Omega network. This method leads to parametrical synthesis of the *indirect* hypercube. Here we describe in brief this approach.

The Omega network is a multistage interconnection network [Bhuyan-1] which is used in multiprocessor computer architectures and whose basic property is the perfect shuffling of the input signals. Its main characteristics are:

- It has $2^k = N$ inputs and so many outputs;
- It is composed of $\log_2 N$ stages each stage including N/2 switching elements (crossbar switches);
- Every crossbar switch has two inputs (0 □ 1) and two outputs (0 □ 1);

Technically, the interconnection between the switching elements of successive stages is accomplished in correspondence with the following procedure:

- Every switch receives a number (binary) within a given stage;
- With every input/output of a given switch a personal identifier is associated;
- Input identifiers include the input number (0 or 1) as most significant digit, followed by the switch number;
- Output identifiers include the switch number as most significant part, followed by the output number (0 or 1).
- Output identifiers of a given switch can be obtained by means of a left cyclic rotation of the respective input identifiers.
- The interconnection between consecutive stages is accomplished on the basis of coincidence of input and output identifiers.

The Omega network is a perfect model for viewing the flow of data o the radix-2 DIT (or DIF) FFT. Each stage of this network is associated with the generation of a given group of sums. Every switch of a given stage receives two partial sums from the previous stage and sends two sums to the next stage. *Figure 1* shows well the FFT parallel intrinsic features - the parallel threads as well as the homogeneity of the flow of data from stage to stage. This homogeneity makes possible the operation of all the stages of the FFT algorithm to be performed by one physical stage. This is accomplished when the outputs of one stage of the Omega network are fed to the inputs of the same stage in accordance with the presented manner of interconnection (*figure 2*). The structure, presented on *figure 2*, is known as the *indirect* binary hypercube [Pease-1], and it can be regarded as the maximum parallel structure of 8-point FFT of this type (8-point FFT on 4 processors).

It is clear that the *maximum parallel structure* is not the best solution, and that it is preferable (recommendable) that

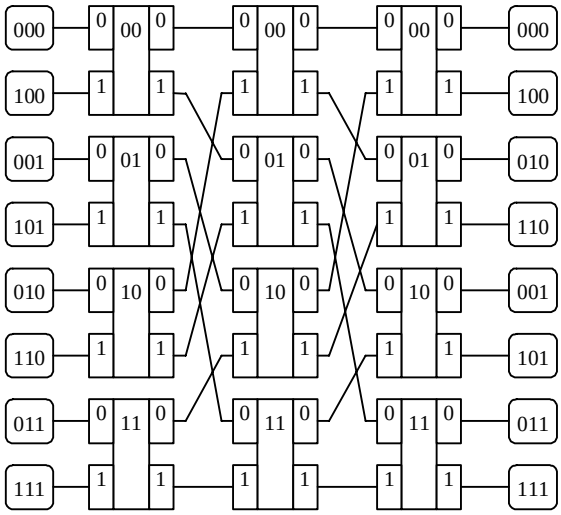

Figure 1. 8-point Omega network

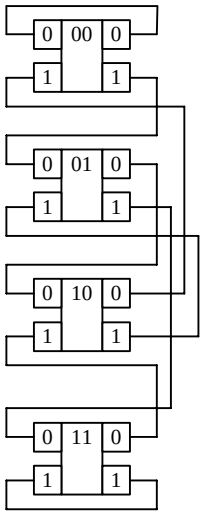

Figure 2. 8-point indirect hypercube

we could effectively use the structure from *figure 2* (including 4 PE and very well suited for 8-point FFT) for FFT of greater number of points - 16, 32, etc. In other words, it is necessary to have a method for parametrical synthesis of such structures, where the parameters are the number of points and the number of processors.

Parametrical Synthesis of the Indirect Hypercube

The above mentioned problem (the parametrical synthesis) is solved in [Phil-1]. Here we repeat in brief this solution.

Let $N = 2^n$, $n = n_1 + n_2$, $1 < n_1 < n - 1$. We intend to construct an analogous parallel structure for N-point FFT on the basis of 2^{n1} *generalized crossbar switches*.

The generalized crossbar switch (GCS) of 2x2 dimension (two inputs and two outputs) is an integrated object, including a set of ordinary 2x2 crossbar switches whose n_1 least significant binary digits of their numbers coincide (GCS number). GCS is considered as a complex structure which includes one simple 2x2 crossbar switch, performing the functions of a processor element (PE), and one dual-port memory (DM) witch serves for reading of input data for PE and storing of intermediate or final results.

From the global identifier two types of identifiers are extracted - the *Group identifier* and the *Local identifier*.

The Group Identifier (GI) includes the $n_1 + 1$ least significant digits of the global identifier.

- The GI identifies the groups of signals entering/exiting given GCS input/output and is of two types - input GI and output GI.
- The Input GI includes the GCS input number as most significant part (0 or 1), followed by the GCS number.
- The Output GI includes the GCS number as the most significant part (0 or 1), followed by the GCS output number (0 or 1).
- The Output GI of given output is obtained through a left cyclic rotation of the corresponding input GI.
- The GCS interconnection is based on the coincidence of input and output GI.

The Local identifier (LI) includes the n_2 most significant digits of the global identifier.

- The LI identifies the signals inside GCS.
- The most significant digit of LI indicates the PE input number.
- The least significant digit of LI indicates the GCS input number.
- The basic property of LI - left cyclic rotation at transitions from stage to stage.

Now the question is: Can we use effectively the concept of *indirect hypercube* for the purposes of parallel FHT/RFFT?

PN Butterfly Identifier Analysis

In the case of PN butterfly any signal contains two parts - P and N. So, its identifier should contain n-1 binary digits. We can assume the same structure of the signal identifier consisting of two parts - the *S* field and the *K* field. *S* field is again the most significant part of the identifier, and the *K* field is the least significant part. The *K* field is the bit-reversed code of the transform parameter *k*. The *S* field is the same as in the FFT case. Problems arise with the *K* field. In a PN pair, the two parts - P and N - are quite differently identified, so, beforehand it is not clear how the pair should be identified - on the base of P values, on the base of N values, or somehow in another way. The P-based (respectively N-based) identification does not fit the case - P values generate P and N values (respectively N values generate N and P values).

If we analyse k-values in PN pairs, we see that these k-values in a given pair always belong to two different classes. The first class, say the D (direct) class, is generated by a '0'- value of the parameter *k*. The second class, say the C (complementary) class, is generated by a '1'-value of the parameter *k*. In a given PN pair, when the P part is a D-value, the N part is a C-value, and vice versa - when the P part is a C-value, the N part is a D-value. The D-values generate always D-values and

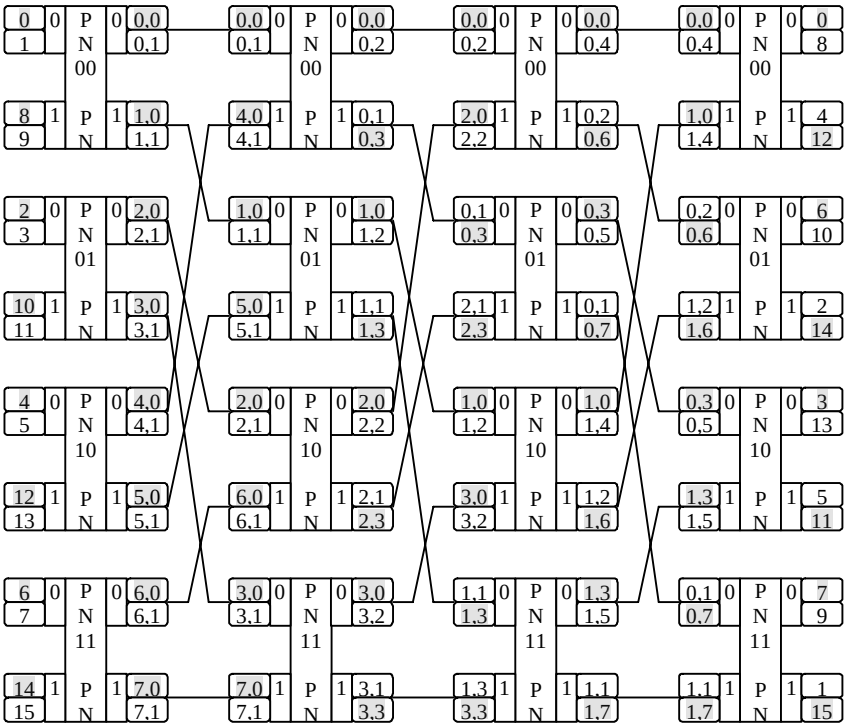

Figure 3. 16-point FHT - 8-point DC Omega network

the C-values generate always C-values - that's what is necessary - identification can be performed either on the base of D- or on the base of C- values.

It is not difficult to see that classes D and C can be characterized in the following manner:

(9) $k \in$ D : $\{k = 0$ or $k = (4p+3)2^q$, for some p,q $\geq 0\}$
$k \in$ C : $\{k \neq 0$ and $k = (4p+1)2^q$, for some p,q $\geq 0\}$

Now we apply D-value based identification of the K field. Since we want to identify pairs of signals, it is evident that the identifiers of a pair of signals should contain one digit less than the identifiers of single signals. The S field is OK and it should not be modified. This means that one digit should be removed from the K field. Analyzing (9) we see that we can remove the least significant '1' in the binary representation of a given D value, when it is not zero, or just remove one binary zero when it is zero.

Very important is the fact that this type of identification possesses the two mentioned properties which idenyify the *perfect shuffle*.

This type of identification leads to a variant of the described Omega network, let us say *DC Omega network*. The difference is that the *DC* Omega network (*figure 3*) has one stage more than the conventional Omega network. This is due to the presence of one preliminary stage in the FHT/RFFT algorithms. In any case, both networks result in one and the same type of *indirect* hypercube, including the parametrical synthesis of this *indirect* hypercube.

Processor Array Architecture „Indirect Hypercube"

The discussed model of the indirect hypercube can be used for different type of concrete implementations, where GCS play the role of a processor block. Analysis shows that this type of topology is very suitable for realization of fast FFT/FHT/RFFT processors for real-time applications. It enables the optimization of resource utilization, ensures low latency and high efficiency - the main requirements for such systems.

In [Phil-2] it is described a parallel SIMD processor array radix-2 FFT architecture based on the *indirect* hypercube interconnection pattern. This architecture with very slight modifications fits directly for the radix-2 FHT/RFFT parallel implementation.

The basic building block of this architecture is the processor block (PB). PB (*figure 4*) includes one processor element (PE) operating as a butterfly unit and two dual-port memories (DM). For a given stage of the transform one of these two DMs is used by PE as a source of data for the current butterfly operations and the other DM is used for storing the results of the current butterfly operations of other PBs (two), which are connected with the given PB. DMs alternatively change their role on stage basis.

PE is realized as a 7-stage linear pipeline. It computes radix-2 butterflies and yields a result every cycle. Input data, as well as intermediate results are presented in fixed point, 2's complement format. Twiddle factors (unique for different PEs) are held in CLUTs. From coefficients viewpoint, the stages are divided into two classes: low stages and high stages. Low stages are the first n_1 stages. High stages are the remaining n_2 stages. For the low stages every PE needs one coefficient pair (e.g. $sinx$ and $cosx$) per stage. For the high stages every PE needs totally 2^{n2-1} coefficient pairs. The set of coefficients for the high stages of given PE includes all the coefficients, whose arguments have as most significant digits of their binary presen-

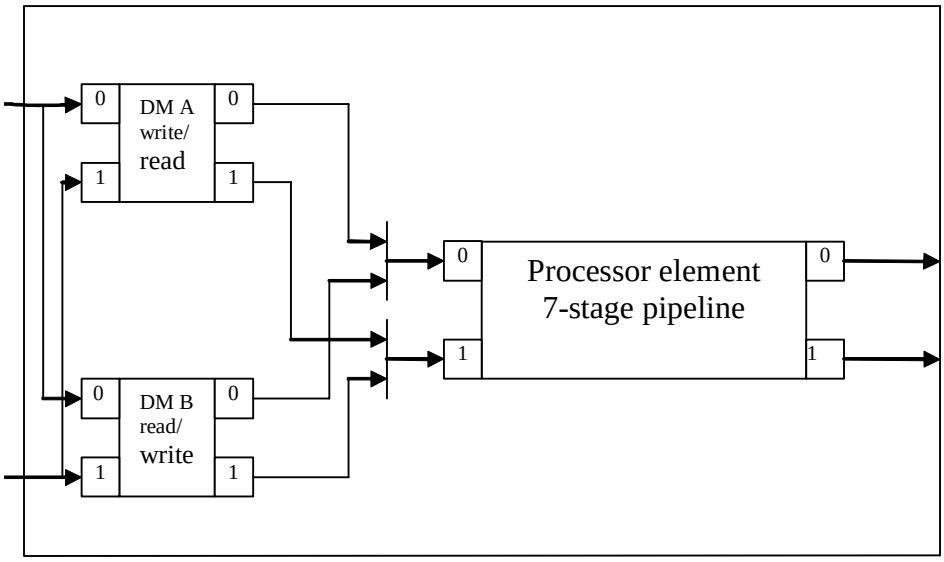

Figure 4. Processor block

tation the bit reversed number of PB.

The control unit provides the necessary control information for PBs - synchronization of the arithmetic pipelines, addresses for DMs and CLUTs, etc.

The DM addressing is based on the main property of the local identifier - a left cyclic rotation on transitions from stage to stage. Generated are four address sequences for DMs - two read address and two write address sequences. The algorithms for generation of these address sequences are similar to the respective algorithms in the scalar case (one PB) - a left cyclic rotation of store address sequences. The addresses of the low parts of CLUTs follow the stage number. The algorithm for generation of the address sequence for the high parts of CLUTs is similar to the respective algorithm in the scalar case.

The resource utilization is a linear function of all input parameters.

The system latency (L) and the throughput (T) are as follows:

$$L = n.(2^{n2-1} + P_l).t_{clk},$$

$$T = N.L^{-1},$$

where P_l is the pipeline length - 7 cycles, t_{clk} is the system clock.

Conclusions

The described systems provide the following advantages:

- parametric generation and utilization in a wide operational range of input parameters;
- the utilization of distributed shared dual-port memories as well as their connection with the processor elements according to the perfect shuffle rule solves efficiently the two basic problems - switching of intermediate data between processors, and memory conflicts;
- uniform control for the different PBs - realization of SIMD architecture;
- theoretically, the architecture allows unlimited scalability depending in practice only on the FPGA capabilities;
- system performance is a linear function (increasing) of the number of PB;
- optimal resource utilization which is a linear function of all input parameters;
- optimization of the parallel system, based on the optimization of the scalar system;
- implementation of block-floating point provides efficient strategy for treating the overflow problem;

The comparison between the two (complex FFT and FHT/RFFT) architectures (as for the performance and efficiency, so for the resource utilization) shows that an N-point FFT architecture in practice is equivalent to a 2N-point FHT/RFFT architecture.

References

1. [Hartley-1] Hartley, R. V. L. A More Symmetrical Fourier Analysis Applied to Transmission Problems. Proc. IRE 30, 1942, 144-150.

2. [Danielson-1] Danielson, G. C. and C. Lanczos. Some Improvements in Practical Fourier Analysis and Their Application to X-ray Scattering From Liquids. J. Franklin Inst., 233, April 1942, 365-380,435-452.

3. [Cooley-1] Cooley J. W., J. W. Tukey. An Algorithm for the Machine Calculation of Complexes Fourier Series. - *Math. Comput.,* 19, 1965, No. 90.

4. [Bergland-1] Bergland Glenn D. A Fast Fourier Transform Algorithm for Real-valued Series. - *Communications of the ACM*, 11, Oct. 1968, Issue 10, 703-710.

5. [Stone-1] Stone, H. S. Parallel Processing with the Perfect Shuffle. IEEE Trans. Computers, C-20, 1971,153-161.

6. [Pease-1] Pease, M. C. The Indirect Binary n-cube Microprocessor Array. IEEE Trans. on Computers, May 1977.

7. [Bracewell-1] Bracewell, R. N. Discrete Hartley transform. - *J. Opt. Soc. Am.*, 73 (12), 1983, 1832-1835.

8. [Bracewell-2] Bracewell, R. N. The Fast Hartley Transform. - *Proc. IEEE*, 72 (8), 1984, 1010-1018.

9. [Ulman-1] Ullmann, Ronald F. An Algorithm for Fast Hartley Transform. Technical Report, Stanford University, 1984.

10. [Bhuyan-1] Bhuyan L. Interconnection Networks for Parallel and Distributed Processing. - *Computer*, June 1987, 9-12.

11. [Sorensen-1] Sorensen, H. V., D. L. Jones, M. T. Heideman, C. S. Burrus. Real-valued Fast Fourier Transform Algorithms. IEEE Trans.

Acoust. Speech Sig. Processing 35, 1987, 849-863.

12. [Marchesi-1] Marchesi, M., G. Orlandi, F. Piazza. A Systolic Circuit for Fast Hartley Transform. IEEE International Symposium on Circuits and Systems, 1988, 2685-2688, 7-9 Jun 1988.

13. [Zapata-1] Zapata, E. L., F. Arguello, F. F. Rivera, J. D. Bruguera. Multidimensional fast Hartley Transform Onto SIMD Hypercubes. - *Microprocessing and Microprogramming*, 29, September 1990, 2, 121-134.

14. [Chang-1] Chang, L.-W., S.-W. Lee. Systolic Arrays for the Discrete Hartley Transform. - *IEEE Transactions on Signal processing*, 39, Nov. 1991,11, 2411-2418.

15. [Zapata-2] Zapata, E. L., F. Arguello. A VLSI Constant Geometry Architecture for the Fast Hartley and Fourier Transforms. - IEEE Transactions on Parallel and Distributed Systems, 3, Jan 1992, 1, 58-70.

16. [Liu-1] Liu, K. J. R., C.-T. Chiu. Unified Parallel Lattice Structures for Time-recursive Discrete Cosine/sine/Hartley Transforms. - *IEEE Transactions on Signal Processing*, Mar. 1993, 1357-1377.

17. [Arguello-1] Arguello F., J. D. Bruguera, R. Doallo, E. L. Zapata. Parallel Architecture for Fast Transforms with Trigonometric Kernel. - *IEEE Transactions on Parallel and Distributed Systems*, 5, Oct. 1994, 10, 1091-1099.

18. [Phil-1] Philipov, Ph., V. Lazarov, Z. Zlatev, M. Ivanova. A Parallel Architecture for Radix-2 Fast Fourier Transform. IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing, Sofia 2006, 229-234.

19. [Phil-2] Philipov, Philip, Ilian Costov, Vladimir Lazarov, Zlaty Zlatev, Milena Ivanova. Implementation of a Parallel Architecture for Radix-2 Fast Fourier Transform. - *Information Technologies and Control*, ISSN 1312-2622, 2008, No. 2, 12-16.

20. [Jones-1] Jones, K. J. Design and Parallel Computation of Regularised Fast Hartley Transform. - *Vision, Image and Signal Processing* - IEEE Proceedings, 153, 9 Feb. 2006, 1, 70-78.

Manuscript received on 30.06.2011

Philip Philipov *was born in 1952. He graduated the Technical University - Sofia as an engineer in electronics. Since 1993 he is with the Institute for Information and Communication Technologies, Bulgarian academy of Sciences, as a research associate. In 2010 he received PhD degree. His main interests are computer architectures, parallel processing and analytical modeling.*

Contacts:
e-mail: filip@bas.bg

Nina Tasheva *(born 1955) received the MA degree in Technical University - Sofia, Bulgaria in 1981. Since 1997 she is M.Sc. I degree in IICT- Bulgarian Academy of Sciences. She has been teaching, consulting and working on projects in the field of Computer Organization, Computer Architecture and e-learning.*

Contacts:
Bulgarian Academy of Sciences, Institute of Information and Communication Technologies
e-mail: tasheva@bas.bg