RADIX 2 AND SPLIT RADIX 2-4 ALGORITHMS IN FORMAL SYNTHESIS OF PARALLEL-PIPELINE FFT PROCESSORS

Alexander A. Petrovsky, Sergei L. Shkredov*

Real-Time Systems Department, Bialystok Technical University
Wiejska 45A Street, 15-351 Białystok, Poland
phone: +48 85 7443073, 742 20 41 ext. 110, email: palex@it.org.by,
web: www.pb.bialystok.pl
Computer Engineering Department, Belarusian State University of Informatics and Radioelectronics
P.Brovki 6; 220027 Minsk, Belarus
phone/fax: +375 172 398420, email: shkredov@mipp.msk.ru
web: www.bsuir.unibel.by

ABSTRACT

The article is devoted to creating a complete methodology for automatic synthesis of real-time FFT-processors at structural level under the given restrictions: speed of input data receipt, structure of the computing element, and the time of the butterfly operation execution. The suggested approach involves creating parallel-pipeline structures for fixed radices FFT and for modified split radix FFT algorithms. The structures employed in the design show good possibilities for scaling the degree of parallelization, thus changing the overall throughput of the system. They are particularly suited for implementing in programmable logic basis (FPGA).

1. INTRODUCTION

There have been a lot of interesting developments in the field of digital signal processing lately. Such multimedia applications as speech encoding, voice recognition systems, hands-free telephones and other tend to employ real-time data processing approach. Current data acquisition technologies can easily provide multidimensional data flow with sampling rates varying in very wide range. Main part of various signal processing algorithms employs Discrete Fourier Transform (DFT) calculation. Moreover, the quality of the signal processing greatly depends on the throughput of the system. In this respect, calculating DFT of a vector signal becomes a critical point in many real-time solutions.

Traditional approaches employ either specially designed signal processors optimized for high-speed arithmetic (DSPs), or application-specific integrated circuits (ASICs). DSP processors implement a limited set of arithmetic and control instructions that can be arranged for making any computation. However, this flexibility can not provide good performance in cases where massive parallelism is necessary. ASICs contain dedicated circuitry optimized for a specific set of tasks. They have the advantages of low power consumption and high sampling rates, but can not solve any task except for the one they were designed for. Programmable logic (PLD, FPGA) represents a new trend in hardware design, capable of creating systems with flexibility never reached before. Static and dynamic reconfiguration abilities of this hardware basis allow creating a new design methodology that is based on formal description of the devices being synthesized.

* - Corresponding author.

2. FFT PROCESSOR FORMAL DESCRIPTION

In the most general case, DFT of any vector process can be described as an independent, joint transformation of D components of D-dimensional process $z=[z_1(n)\ z_2(n)\ldots z_D(n)]^T$ and represented by a set of DFTs of size N. Usually, this task is implemented through the use of various FFT algorithms.

Flexibility of DFT calculations by FFT algorithms rests on the following theorem [1]: let the elements of $z_d(n)$, $d = \overline{1, D}$; $n = \overline{0, N\text{-}1}$ of D complex sequences $\{z_d(n)\}$ of length $N=2^l$ be structured as an one-dimensional array $z(n^*)$ of size DN as following: $z(n^*)=z_d(n)$, $n^* = nD + d$, $n^* = \overline{0, N\text{-}1}$, then, applying the direct FFT algorithm to the base 2 to the array $z(n^*)$ and stopping the algorithm after l iterations yields the D Fourier transforms of size N. A similar theorem can be proved as well for the inverse FFT algorithm only the D-dimensional vector process z is mapped into the one-dimensional array $z(n^*)$ of size DN by means of index transform: $z(n^*) = z_d(n)$, $n^* = dN + n$, $n^* = \overline{0, N\text{-}1}$

FFT algorithms provide recursive and regular processor structures and seem to be ideal candidates for implementing calculations of DFT of vector processes. Parallel-pipeline structures enable wide range of possibilities for arranging parallel processing and flexible changing of DFT format by creating a formal description of the parallel-pipeline FFT processor. Such model is based on a two-dimensional representation of the data-flow. Let s and t be the bit capacity of the array indices. Than the position of any element is defined by its two indices $[x, y] = [\{a_{s-1}\ldots a_1a_0\}\{b_{t-1}\ldots b_1b_0\}]$, where $x=a_s2^{s-1} + a_{s-1}2^{s-2} +\ldots+ a_1$ is a column number ($a_i = \{0,1\}$),

and $y = b_t 2^{t-1} + b_{t-1}2^{t-2} + \ldots + b_1$ is a row number ($b_i = \{0,1\}$).

The operators describing a parallel-pipeline structure are defined by their effect on the indices of data elements and give certain permutations of these elements in the array [2]. When describing the parallel-pipeline structure the agreement taken in [3] is executed, that is the operators are written from left to right $\pi_1\pi_2\,[x, y] = \pi_2(\pi_1\,[x, y])$. This agreement is associated with accepted direction of data movement in the processor structure and is made for precise correspondence of operators' sequences to the structure images.

The first group of operators determines the functionality of the core stages of parallel-pipeline FFT-processor:

- Operator B describes the butterfly operation of the chosen FFT algorithm, and is considered purely arithmetic because it does not affect the positions of elements within the dataflow.
- Operator $M_{(i)}$ implements the following permutation of elements in two-dimensional array: $M_{(i)}[x, y]=[\{a_{s-1}...a_i b_0 a_{i-2}...a_0\}, \{b_{t-1}...b_1 a_{i-1}\}]$, $i < s$.
- Operator $\beta_{(i)}$ permutes the rows of original data array after the following rule: $\beta_{(i)}[x, y] = [x, \{b_{t-1}\ldots b_{i+1}\ b_0\ b_{i-1}\ldots b_1\ b_i\}]$, $i < (t\text{-}1)$.

The second group of operators deals with rearranging the elements in input and output memory of parallel-pipeline FFT processor:

- Operator $\delta_{(i)}$ divides every row of the initial data array into 2^i rows: $\delta_{(i)}[x,y] = [\{a_{s-1}\ldots a_i\},\{b_{t-1}\ldots b_0\ a_{i-1}\ldots a_0\}]$, $i \le s$.
- Introduction of operator $\sigma_{(i)}$ allows forming inter reverse permutations when describing parallel-pipeline FFT-processors' input and output [4]: $\sigma_{(i)}[x, y] = [x, \{b_{t-1}\ldots b_{i+1}\ b_{i-1}\ldots b_0\ b_i\}]$; $\sigma_{(i)}^{-1}[x, y] = [x, \{b_{t-1}\ldots b_{i+1}\ b_0\ b_i\ldots b_1\}]$, $i < (t\text{-}1)$.

3. RADIX 2 FFT PARALLEL-PIPELINE STRUCTURE

By means of the language of formal description introduced in the previous section, it is possible to submit the description of forward and inverse parallel-pipeline structures computing DFT of vector process consisting of $D=2^q$ components of length $N=2^l$, where $m=2^r$ computing elements at a stage simultaneously implement m consecutive butterfly operations. Formula (1) describes forward FFT-processor, and formula (2) - the inverse one:

$$FFT^{F}_{(q,l,r)} = \begin{cases} \delta_{(r)}\mu_{(q+l-r)}BM_{(q+l-r-1)}B\ldots \\ \ldots BM_{(q-r+1)}B\mu^{-1}_{(q-r+1)}\mu^{-1}_{(r,r)}, r \le q; \\ \delta_{(r)}\mu_{(q+l-r)}BM_{(q+l-r-1)}B\ldots BM_{(1)}B\beta_{(r)}B\ldots \\ \ldots B\beta_{(q+1)}B\sigma^{-1}_{(q)}\sigma_{(r)}\mu^{-1}_{(1)}\mu^{-1}_{(r,r)}, r > q; \end{cases} \quad (1),$$

$$FFT^{I}_{(q,l,r)} = \begin{cases} \delta_{(r)}\mu_{(1)}\sigma^{-1}_{(r)}B\beta_{(1)}B\ldots B\beta_{(r)}BM_{(1)}B\ldots \\ \ldots BM_{(l-r-1)}B\mu^{-1}_{(l-r)}\mu^{-1}_{(r,r)}, r < l-1; \\ \delta_{(r)}\mu_{(1)}\sigma^{-1}_{(r)}B\beta_{(1)}B\ldots \\ \ldots B\beta_{(l-1)}B\sigma^{-1}_{(l-1)}\sigma_{(r)}\mu^{-1}_{(1)}\mu^{-1}_{(r,r)}, r \ge l-1; \end{cases} \quad (2)$$

Operator B represents simultaneous execution of butterfly operations in all pipelines of the processor. Operator $M_{(i)}$ describes the functioning of intermediate memory of $2x2^{i-1}$ words, operator $\beta_{(i)}$ shows direct (without intermediate memory) connection between two stages of the parallel-pipeline FFT processor. This feature, as well as direct correspondence between formulas (1, 2) and processor structures is shown on Figure 1. The example

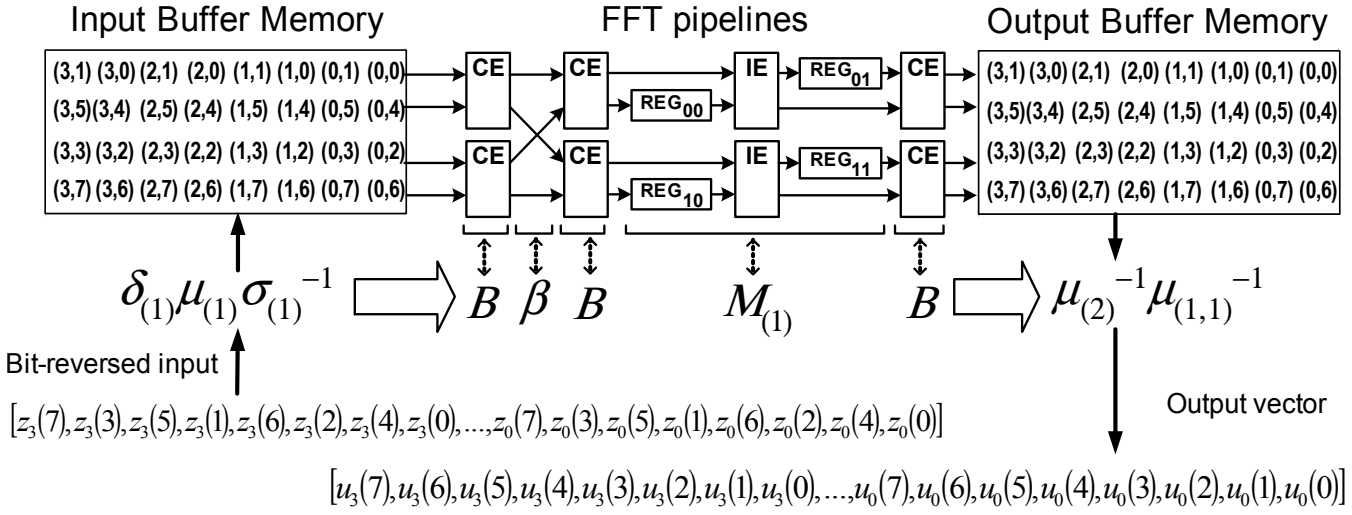

Figure 1. Radix-2 FFT pipelines for two Computing Elements (CEs) at a stage (m=2) and its correspondence to formal description of the processor.

demonstrates structure of parallel-pipeline FFT processor for computation of inverse vector DFT (see formula 2) for D=4 vectors (q=2) of size N=8 (l=3), and m=2 (r=1). In parallel-pipeline processing of vector FFT with m Computing Elements (CEs) on a stage, CEs calculate the butterfly operations which have the numbers with $DN/2m$ difference. The number of parallel working CEs on the stage can be various. The lower bound of m is *1* that corresponds to one-pipeline FFT processor. The upper bound of m is $DN/2$ that corresponds to a matrix processor. However, it is more convenient to limit m by values equal to powers of 2 because in this case every CE calculates the same number of butterfly operations and the performance is uniformly distributed between the CEs.

4. GENERAL DESCRIPTION OF SPLIT RADIX 2-4 FFT PARALLEL-PIPELINE STRUCTURE

There are some algorithms of calculating DFT providing better efficiency in comparison with fixed radix ones [5]. Thus, algorithm suggested by Duhamel and Hallmann uses decomposition of the basic formula for DFT calculation and provides substantial reduction of computational complexity [6]. Split radix 2-4 FFT

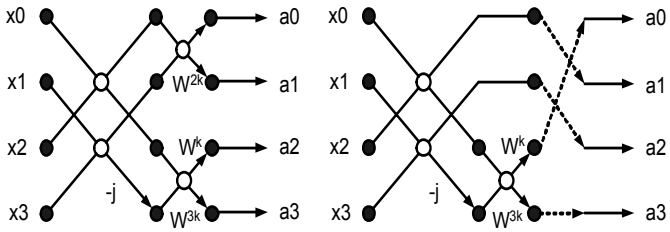

Figure. 2. Radix-4 butterfly (to the left) versus modified butterfly operation of the split radix FFT (to the right).

algorithm is an in-place algorithm employing the butterfly operation analogous to the one used in radix-4 FFT (see Figure 2). Along with calculating DFT of the sequences of size 2^n split radix 2-4 FFT algorithm shows regularity of the radix 4 FFT one. However, split radix FFT stages are irregular that makes its control a more difficult task.

It was shown in [7], that simple permutation of outputs in split radix FFT butterfly operation can recoup to some extent this drawback of the split radix FFT algorithm. The modified butterfly operation is shown on Figure 2. The resulting structure shows better regularity and allows creating a unified computing element for odd and even stages of the split radix 2-4 FFT algorithm implementation. It, in turn, enables the use the operators introduced above in formal description of parallel-pipeline split radix 2-4 FFT structures.

Developing the pipeline structure based on the modified butterfly operation [7], it is possible to create a parallel-pipeline structure, that is described in the same manner as it was presented for radix 2 FFT algorithm in the previous section. Computing Elements (CEs) are cascaded with Interconnection Elements (IEs), which are routing the data between the CEs. The structure of Computing Element presumes that the butterfly multiplications are done in parallel with the add-subtract operations of the following stage of computation. At the same time, the last stage of the algorithm involves radix 2 butterflies only. These features necessitate input and output stages, calculations at which are different from the rest of the computational stages.

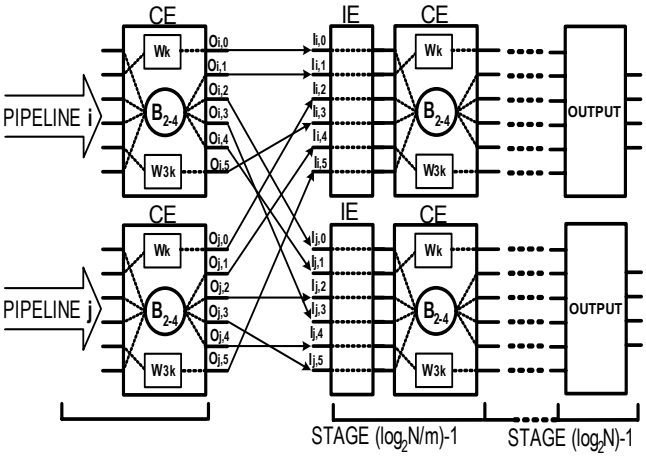

Figure 3. Split radix 2-4 FFT pipelines and general structure of interstage communications.

Never the less split radix FFT pipeline offers decent possibilities for vector data flow processing. The number of parallel working CEs on the stage can be various. Increasing m correspondingly increases the throughput of the system. Let us suppose that number of split radix FFT pipelines m is a power of 2. In this case, until certain point, the modified split radix FFT algorithm processes m data flows independently. Starting from stage log_2N/m-1 and up to the stage log_2N-2 (the stage preceding the last one), outputs of every previous stage should be permutated. In other words, log_2m stages (from stage (log_2N/m)-1 till the stage with number (log_2N)-2) start their work from making a certain switching procedure between pairs of pipelines, thus implementing all permutations, required by the graph of split radix FFT algorithm.

Considering the most simple case of two pipelines (m=2), it is obvious, that the only additional permutation is required before stage $log_2N/2$-1, which determines the stage preceding the output stage of the algorithm. Numbers of pipelines are i=0, j=1. Switching procedure is determined as mapping the vector $O=[O_{i,0},O_{i,1},O_{i,2},O_{i,3},O_{i,4},O_{i,5},O_{j,0},O_{j,1},O_{j,2},O_{j,3},O_{j,4},O_{j,5}]$ representing the outputs of the previous stage into the vector of inputs of the following stage $I=[I_{i,0}, I_{i,1}, I_{i,2}, I_{i,3}, I_{i,4}, I_{i,5}, I_{j,0}, I_{j,1}, I_{j,2}, I_{j,3}, I_{j,4}, I_{j,5}]$. The corresponding mapping is demonstrated on Figure 3.

Specific features of modified split radix 2-4 FFT graph enable formulating the general procedure of permutations at every stage of the algorithm: Let s be number of the stage requiring permutations. Then the pairs of pipelines with numbers i, j for stage s permutation procedure are selected

according to the simple equation:

$j = i + N/2^{s+1}$, where $i = [0,1,2,...(N/2^{s+1}-1)]$ and
$\log_2 N/m - 1 \le s < (\log_2 N) - 1$ (3)

5. REAL-TIME FFT PROCESSORS SYNTHESIS

The basic criteria while creating a real-time vector FFT processor structure is maintenance of computation of vector DFT in real time with the minimum structural complexity. Development of the FFT-processor on the basis of parallel-pipeline structure requires solving the following two basic problems of structural synthesis:

1) Defining the optimum number of computing elements at a stage ensuring, that under given time of base operation execution, the computation of the given vector DFT in real time is possible and is reached with minimal structural complexity of the processor.

2) Defining topology of structure of the FFT-processor for the chosen number of CEs at a stage.

The problem of optimum number of CEs in the parallel-pipeline FFT processor is normally reduced to finding the minimal number of CEs at a stage that at the given time of butterfly operation execution can provide calculation of vector DFT in real time. This approach is well justified from the point of view of reducing the hardware complexity and power consumption. Condition for calculation of vector DFT in real time for any FFT structure will be $(DN/k)T_B \le N\Delta t$, where T_B is a time of butterfly operation execution for the chosen algorithm and hardware; Δt is a sampling period; k is number of data samples processed by each pipeline; D is number of components of the vector process; N is size of a component. Then $(DT_B/\Delta t) \le k$ ($k=2m$ in case of radix-2 FFT algorithm). In case of split radix 2-4 FFT algorithm $k=4m$ and the butterfly execution time T_B should be determined as max [$2Ta$, Tm], where Ta is addition-subtraction time and Tm - time of twiddle factor multiplication. As the number of CEs at a stage in the parallel pipeline FFT-processor is limited by the value that is a power of 2, the minimum number of CEs at a stage m_{min} for radix-2 FFT is defined as follows:

$$m_{\min} = \begin{cases} 1, r = 0, if\ DT_B < 2\Delta t, \\ 2^r, r = \left]\log(DT_B / 2\Delta t\right[, if\ DT_B \ge 2\Delta t, \end{cases}$$

where $]a[$ is nearest integer, greater than a or equal to it; T_B is a time of butterfly operation execution of algorithm FFT (split radix or fixed radix one); Δt is a sampling period; m is number of CEs at a stage; D is number of components of vector process; N is size of a component. The analogous equation can be derived for split-radix FFT.

Having found the number of pipelines for the corresponding parallel-pipeline structure, the second problem of structural synthesis is easily solved according to the methodology suggested in previous sections. Depending on the algorithm, the resulting structure will be described either by formulas (1, 2), or be determined by equation (3) applied for pattern shown on Figure 3.

6. CONCLUSION

The suggested methodology involves synthesis of real-time parallel-pipeline FFT-processors at structural level. The design algorithm is based on a limited set of input data and creates an optimal parallel-pipeline structure under the given restrictions: speed of input data receipt, structure of the computing element, and the time of the butterfly operation execution. It also provides employing parallel-pipeline structures either for fixed radices FFT, or for modified split radix FFT algorithms.

The additional flexibility of the hardware implementations can be gained by choosing proper hardware basis. The suggested implementations are characterized by little involvement of logical operations, but modest memory requirements. Due to these features, the most efficient realization of the presented parallel-pipeline FFT processor is achieved by means of FPGA technology.

Considering the unified design algorithm and the general description of hardware structures received, mapping into FPGA basis brings ability of integrating the corresponding processor structures into a wide range of devices requiring real-time calculation of vector DFT.

REFERENCES

[1] D.G.Korn, J.J.Lambiotte, “Computing the fast Fourier transform on a vector computer” / *Mathematics of computation, vol. 33, no. 147*, *1979* - pp.977-992

[2] E. H. Wold, A. M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations", *IEEE The Transactions on Computers, vol.C-33, no.5, May. 1984* - pp. 414-426.

[3] D.S. Parker, "Notes on shuffle/exchange-type switching networks", *IEEE The Trans .Comput., vol. C-29, no. 3*, *Mar. 1980,* pp. 213-222.

[4] A. A. Petrovsky, M. V. Kachinsky, "Design Method of Real-Time Parallel-Pipeline Processors Computing the Vector DFT", *International conference on Parallel Computing in Electrical Engineering, September 2-5, 1998, Bialystok, Poland* - pp.242-247.

[5] R. D. Preuss, "Very Fast Computation of the Radix-2 Discrete Fourier Transform", *IEEE Trans., 1982, ASSP-30*, pp.595-607

[6] Duhamel, H. Hollmann, "Split Radix FFT Algorithm", *Electronics Letters, Vol.20 No.1, 5th January, 1984* - pp.14-16.

[7] J. Garcia, J. A. Michell, A. M. Buron., "VLSI Configurable Delay Commutator for a Pipeline Split Radix FFT Architecture", *IEEE Transactions on Signal Processing, Vol.47, No.11, November, 1999* - pp.3098-3107.