Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2016, Article ID 3561317, 7 pages
http://dx.doi.org/10.1155/2016/3561317

Research Article

Modules for Pipelined Mixed Radix FFT Processors

Anatolij Sergiyenko and Anastasia Serhienko

Computer Science Department, National Technical University of Ukraine, Peremogy Avenue 37, Kiev 03056, Ukraine

Correspondence should be addressed to Anatolij Sergiyenko; asergy@bigmir.net

Received 26 October 2015; Revised 2 January 2016; Accepted 5 January 2016

Academic Editor: Michael Hübner

A set of soft IP cores for the Winograd r-point fast Fourier transform (FFT) is considered. The cores are designed by the method of spatial SDF mapping into the hardware, which provides the minimized hardware volume at the cost of slowdown of the algorithm by r times. Their clock frequency is equal to the data sampling frequency. The cores are intended for the high-speed pipelined FFT processors, which are implemented in FPGA.

1. Introduction

Fast Fourier transform (FFT) algorithm is widely used in many signal processing and communication systems. Due to its intensive computational requirements, it occupies large area and consumes high power if implemented in hardware.

FFT uses divide and conquer approach to reduce the computations of the discrete Fourier transform (DFT). In Cooley-Tukey radix-2 algorithm, the N-point DFT is subdivided into two $(N/2)$-point DFTs and then $(N/2)$-point DFT is recursively divided into smaller DFTs until a two-point DFT. The last procedure, named as radix-2 butterfly, is just an addition and a subtraction of complex numbers.

Higher radix algorithms, such as radix-4 and radix-8, can be employed to reduce the complex multiplications, but the butterfly structure becomes complex. So, a split radix algorithm [1] is adopted to get the benefits of both radix-2 and radix-4 algorithms.

Prime factor algorithms use Good-Thomas mapping and Chinese Remainder Theorem for decomposing the N-point DFT into smaller DFTs, which are the factors of N and are mutually prime [2]. With this mapping the twiddle factor multiplications are avoided at a cost of increased number of additions and irregular structure. A modification of the prime factor algorithm is Winograd fast Fourier transform algorithm. It is capable of achieving minimum complex multiplications but the number of additions is increased.

Pipelined FFT architectures are fast and high throughput architectures with parallelism and pipelining [3]. Among them the single-path delay feedback architecture and multipath delay commutator architecture are the most popular.

In the first kind of architectures, each of the pipeline stages contains twiddle factor multiplier, r-point DFT unit ($r = 2, 4$), and data buffer, which stays in the feedback of the stage. This buffer is filled before the DFT computations. Therefore, such structure could not be fully loaded [4, 5].

In the second kind of architectures, the pipeline stages contain data buffer, r-point DFT unit, and twiddle factor multiplier, which are connected in sequence. The data buffer is based on multipath delay commutator and provides sets of r complex data which feed the DFT unit in parallel. Such architecture provides the maximum throughput at the cost of the high hardware volume [3, 6]. Besides, it must implement the uniform radix r FFT, because for the mixed radix FFT the data buffers become too complex.

Systolic array scheme has also been proposed for FFT computations [7, 8]. The r-point DFT in it is calculated as r separate sums of r weighted samples. It is attractive because of its regularity, scalability, locality of interconnections, and suitability for non-power-of-two transforms. However, such processor requires substantial hardware volume. For example, the 16×16-point processor for 16-bit data contains 3982 adaptive logic modules (ALMs) and 33 multipliers of the Altera Stratix III FPGA, comparing to the usual 20-bit width

pipelined FFT processor, which contains 4261 ALMs, and only 24 multipliers [8].

The implementation of the pipelined FFT architecture in modern field programmable gate arrays (FPGAs) provides the high-speed hardware solution with small energy consumption. One FFT of 256 16-bit complex points dissipates approximately one microjoule in FPGA [6]. The FFT processor for $N = 4096$, which occupies 36.7 k ALMs, and 60 multipliers provides the speed up to 90 GFLOPS, and the efficiency near 10 GFOPS per watt, which is in many times higher than in CPU, or GPU implementation [9]. Besides, this architecture can be accommodated in FPGA to the solved problem conditions by exchanging the throughput, transform length N, or computation precision.

The papers [10–12] describe the design and implementation of radix-2^2 single-path delay feedback pipelined FFT.

In most cases the power-of-two FFT processors are implemented in FPGA. In the paper [13] it was proven that Radix-2 FFT method provides the least number of FPGA slices, the Good-Thomas method is faster than Cooley-Tukey, and the Rader method had the lowest operating frequency of the pipelined processor in FPGA.

The non-power-of-two transforms are widely used in the OFDM modems. In such transform the Winograd algorithm minimizes the number of multiplications in the DFT modules but also adds a degree of complexity and significantly increases the total number of utilized adders in FPGA [14, 15].

In [16] the parallel architecture of the DFT module has been proposed for the computation of this algorithm. This architecture is able to deal with a large amount of FFT sizes, decomposable in product factors that are 2, 3, 4, 5, 7, or 8.

In [17] the pipelined processor design is proposed, which uses the Cooley-Tukey FFT algorithm for FFT computation only in those cases where the factors of the number N are not relatively prime.

The DFT modules, which are used in the examples of the pipelined FFT processors, are designed by the one-to-one mapping of the respective small point FFT algorithms. As a result, they need the data feeding through r input ports. Consider the two stage pipeline; the number N is factored to factors p, r, $p \neq r$. Then the buffer has r FIFOs of the length of more than p, which are fed from p inputs in the nonuniform order. Therefore, to provide the proper input data order for these stages, the complex data buffers must be attached to their ports.

Consider the DFT modules, which accept the r input data sequentially for r steps. Then both data buffers and twiddle factor multipliers are simplified substantially. These modules have the slowdown operation in r times. Besides, the hardware volume of the DFT modules can be decreased up to r times. The disadvantage of this architecture is the decrease of the FFT processor throughput up to r times. But in this situation we can provide the proper system throughput by the increase of the FFT processors number, which are configured in FPGA.

In this paper we propose the design of a set of r-point DFT units, which help to implement the pipelined FFT processors, when the data flow is a single sample per a clock cycle.

2. A Method of Pipelined Datapath Synthesis

By the high level synthesis the DSP algorithm is usually described by a signal flow graph or a synchronous data flow (SDF). In SDF the nodes-actors and edges represent the operators and data transfers between them, respectively. Each actor consumes and generates the same amount of data in each SDF cycle [21, 22].

Uniform SDF has the property that its graph is equal to the graph of the pipelined datapath, which implements the algorithm with the period of $L = 1$ clock cycle. Then the SDF nodes are mapped into the operational resources like adders, multipliers, the edges are mapped into the connections, and the delays in edges represent the pipelined registers. This property is widely used by the synthesis of DSP modules for FPGA in many CAD tools like Matlab-Simulink System Generator [23].

The synthesis of the pipelined datapath with the period of $L > 1$ cycles is usually performed by the steps of resource selection, actor scheduling, and resource assignment. Then the datapath structure is found, and the control unit is synthesized.

It is worth mentioning that most of DFT algorithms are acyclic ones. The most popular scheduling methods for limited resources and execution time consider the acyclic SDF. These methods are list scheduling and force directed scheduling [24]. The register allocation is effectively implemented by the Tseng heuristic and by the left edge scheduling. The use of the cyclic interval graph takes into account the cyclic nature of the SDF algorithm [25]. The retiming methods and the graph folding methods simplify the SDF mapping [26, 27].

In [28, 29] the method of the datapath synthesis is proposed, which is based on SDF. This method, adapted to the acyclic SDF, is described below. In this method, SDF is represented in the three-dimensional space in the form of a triple $K_G = (K, D, A)$, where K is the matrix of vectors-nodes $\mathbf{K}_i$, which mean the operators or actors, D is matrix of vectors-edges $\mathbf{D}_j$, performing the links between operators, and A is the incidence matrix of SDF.

In the vector $\mathbf{K}_i = (k_i, s_i, t_i)^T$ the coordinates k_i, s_i, and t_i correspond to the type of operator, the processor unit (PU) number, and the clock cycle. The SDF graph in such representation is called spatial SDF.

Spatial SDF is split into the spatial configuration $K_{GS} = (K_S, D_S, A)$ and event configuration $K_{GT} = (K_T, D_T, A)$, which correspond to the datapath structure, and its schedule. By the splitting process the vectors $\mathbf{K}_i = (k_i, s_i, t_i)^T$ are decomposed into vectors $\mathbf{K}_{Si} = (k_i, s_i)^T$, corresponding to the PU coordinates, and vectors $\mathbf{K}_{Ti} = t_i$, which mean the execution times of the relevant operators in PU $\mathbf{K}_{Si}$. Then the temporal component $\mathbf{D}_{Ti} = t_i$ of the vector $\mathbf{D}_i$ is equal to the delay of transfer or processing of the respective variable.

We can assume that the matrix K encodes some acceptable structural solution, since the matrix D is calculated by

$$D = KA. \tag{1}$$

The structural optimization consists in finding such matrix K, which minimizes a given quality criterion. It is

possible to specify a matrix D_O which provides the minimum value of T_C. Then the vectors $\mathbf{K}_i$ are found from a relationship

$$K = D_O A_O^{-1}, \tag{2}$$

where D_O is the matrix of vectors-nodes and A_O is the incidence matrix of the maximum spanning tree for SDF. When looking for the effective structural solution, the following relations have to be considered. Spatial SDF is valid, if the matrix K has no two identical vectors; that is,

$$\forall \mathbf{K}_i, \mathbf{K}_j \quad \left(\mathbf{K}_i \neq \mathbf{K}_j,\ i \neq j\right). \tag{3}$$

The schedule with the period of L clock cycles is correct if the operators, which are mapped into the same PU, are performed in different cycles; that is,

$$\forall \mathbf{K}_i, \mathbf{K}_j \quad \left(k_i = k_j,\ s_i = s_j\right) \Longrightarrow t_i \not\equiv t_j \bmod L. \tag{4}$$

This inequality provides the correct circular schedule. Moreover, the next operator has to be executed no earlier than the previous one; that is,

$$\forall \mathbf{D}_j \quad \left(t_i \geq 0\right). \tag{5}$$

The operators of the same type should be mapped into PU of the same type; that is,

$$\mathbf{K}_i, \mathbf{K}_j \in K_{p,q} \quad \left(k_i = k_j = p,\ s_i = s_j = q\right),\ \left|K_p, q\right| \leq L, \tag{6}$$

where $K_{p,q}$ is a set of p-type vectors-operators, which are mapped in the qth PU of pth type ($q = 1, 2, \ldots, q^p{}_{\max}$).

Then the search for the effective schedule consists in the following. The vectors $\mathbf{D}_i \in D_O$ are assigned the coordinate $t_i = 1$; that is, the respective operators have the delays of a single clock cycle. The matrix K_T is found from (2). The remaining elements of the matrix D_T are found from (1). If inequality (5) is not satisfied for some of vectors, then the coordinate t_i is increased for certain vectors $\mathbf{D}_i \in D_O$, and the schedule search is repeated. The rest of $\mathbf{K}_i$ coordinates are found from conditions (3)–(6). In such wise the fastest schedule is built, as each statement is executed in a single clock cycle without unnecessary delays.

The resulting spatial SDF can be described by the VHDL language, so the pipelined datapath description can be translated into the gate level description of the FPGA configuration by the proper compiler-synthesizer [30].

During the structure synthesis, the nodes are placed in the space according to a set of rules, providing the minimum hardware volume for the given number of clock cycles in the algorithm period. The resulting spatial SDF is described by VHDL language and is modelled and compiled using proper CAD tools.

The method is similar to the known method of the SDF folding in L times [31]. However, it is distinguished from the intuitive folding procedure in the formal approach and directed optimization process. In this method, the steps of resource selection, operator scheduling, and resource allocation are implemented in a single step, providing more effective optimization.

The method was built in the framework which is intended for the SDF graph input and its graphical editing. Both algorithm and resulting structure are stored in XML files. The framework can translate the XML description into the VHDL synthesizable model, which can be modelled and synthesized by usual CAD tools provided by different companies. The present limitation consists in that the SDF graph is optimized only by hand using the relations, definitions, and theorems, mentioned above [32]. The shown below examples were synthesized by Xilinx ISE, Ver. 13.3.

The method is successfully proven by the synthesis of a set of pipelined FFT processors, IIR filters, and other pipelined datapaths for FPGA [33]. A set of DFT modules was designed using it as well.

3. Example of the DFT Module Synthesis

Consider a design example of a DFT module of $r = 3$ points. It performs the Winograd DFT algorithm, which is described in [5]:

$$\begin{aligned} t &= x_1 + x_2, \\ p &= x_2 - x_1, \\ m_0 &= x_0 + t, \\ m_1 &= \left(\cos\frac{2\pi}{3} - 1\right) \cdot t, \\ m_2 &= j \cdot \sin\frac{2\pi}{3} \cdot p, \\ s &= m_0 + m_1, \\ X_0 &= m_0, \\ X_1 &= s + m_2, \\ X_2 &= s - m_2, \end{aligned} \tag{7}$$

where $\{x_0, x_1, x_2\}$ is the input complex data set, $\{X_0, X_1, X_2\}$ is the complex result set, and $j = \sqrt{-1}$. In this algorithm $\cos 2\pi/3 = -0.5$; therefore, $m_1 = -1.5 \cdot t$. The algorithm has twelve real additions and four real multiplications.

To minimize the number of multiply units (MPUs) and to increase the clock frequency, it is worth to use the application specific MPUs [34]. Then the coefficient $k = \sin 2\pi/3 = 0.866025 = 0.1101110110110100_2$. To minimize the addition operations it is represented by digits 1, 0, and −1; that is, $k = 1.00^-; 10\,00^-; 10\,0^-; 10^-; 1\,01$.

Then the multiplication $k \cdot p = \sin 2\pi/3 \cdot p$ is implemented as

$$k \cdot p = p - \left(p + p2^{-4}\right)2^{-3} + \left(\left(p2^{-2} - p\right)2^{-2} - p\right)2^{-10}. \tag{8}$$

SDF of this algorithm is shown in Figure 1. The black circles in it represent the input-output nodes, circle with a

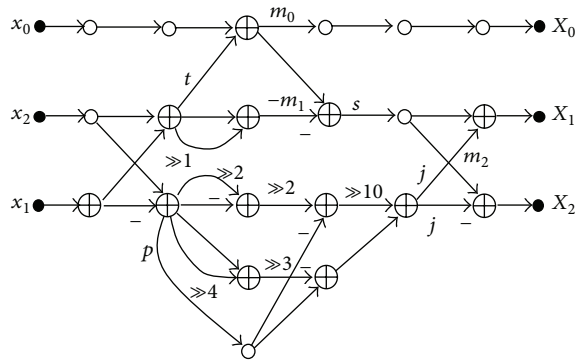

Figure 1: SDF of the 3-point DFT.

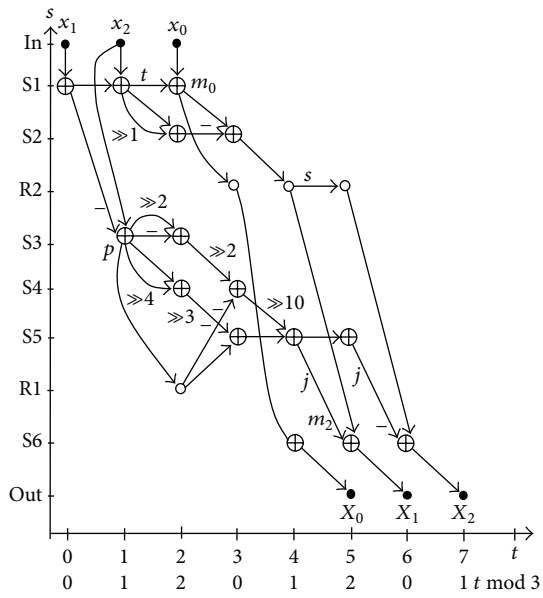

Figure 2: Spatial SDF of the 3-point DFT.

cross does complex addition, and symbols "$\gg l$" mean the shift right operation to l bits. The edge, which is loaded by j, means the multiplication to j, which is performed as inversion of the image part of data and swapping the real and image parts of data. Each node performs a delay to a single clock cycle. Therefore, this SDF makes the structure of a module, which computes DFT with the period of $L = 1$ cycle.

Due to the method described above, SDF is represented in the three-dimensional space for $L = 3$ as a spatial SDF, which is illustrated by Figure 2. Comparing to SDF in Figure 1, the spatial SDF is placed in the space with the coordinates of resources s and time t. The coordinates of a node in it mean the number of PUs, where it is triggered, and the number of the clock cycles. The operator type is coded by the character R as a register and character S as an adder. Below the axis ot the axis with figures (t mod 3) is placed to simplify the check of the SDF correctness due to formula (4).

We can see that the spatial SDF codes both the algorithm schedule and the module structure where it is performed.

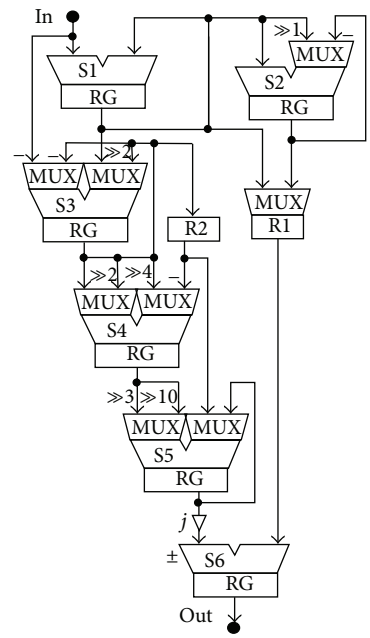

Figure 3: Structure of the DFT module.

This SDF is formally translated into VHDL description in the synthesis style as shown in Algorithm 1.

Here the signals and ports are used, which represent the outputs of the respective operator nodes of SDF in Figure 2. Note that the complex variables are substituted by a couple of signed-type signals with the suffixes r, i in their names, respectively. The input impulse START synchronizes the phase generator, which is described in the process operator CNTRL. It generates the phase number signal CYC with the period of $L = 3$ clocks.

The DFT calculations are performed in the process operator CALC. The CASE operator in it consists of three alternatives depending on the signal CYC. In the ith alternative the operators are placed, which are performed in the clock cycle $i = t$ mod 3 due to the SDF in Figure 2.

Each signal assignment in this process is mapped into the respective pipeline register because the activation of it is implemented in the rising edge of the clock signal. The CASE operator alternatives are mapped into respective PUs of adders using the known resource sharing technique [35]. The resulting DFT module structure is shown in Figure 3. It is shown only as an illustration, because its forming is not necessary for the module design.

As we can see in Figure 3, the adders with two input multiplexers are placed between two neighboring registers in this module. A single digit of such unit is mapped into a single 6-input CLB, which is used in modern FPGAs. Therefore, this module has the shortest critical path and maximized clock frequency.

4. Results of the DFT Module Synthesis

A set of DFT modules was designed using the method of spaced SDF, described above. Each of them inputs a single

```
library IEEE;
use IEEE.STD_LOGIC_1164.all, IEEE.STD_logic_arith.all;
entity DFT3 is
      port(
      CLK: in STD_LOGIC;
      START: in STD_LOGIC;
      DRI: in std_logic_vector(15 downto 0);
      DII: in std_logic_vector(15 downto 0);
      DRO: out std_logic_vector(17 downto 0);
      DIO: out std_logic_vector(17 downto 0));
end DFT3;
architecture synt of DFT3 is
   signal S1r,S2r,S3r,S5r,S6r,R1r,R2r: signed(17 downto 0);
   signal S1i,S2i,S3i,S5i,S6i,R1i,R2i: signed(17 downto 0);
   signal S4r,S4i: signed(17 downto 0);
   signal CYC: natural range 0 to 3;
begin
  CNTRL:process(CLK) begin
      if rising_edge(CLK) then
         if START ='1' then
            CYC<=0;
         else
            if CYC =2 then
               CYC <=0;
            else
               CYC <=CYC +1;
            end if;
         end if;
      end if;
   end process;
  CALC: process(CLK) begin
      if rising_edge(CLK) then
   case CYC is
     when 0 =>
        S1r<= signed(SXT(DRI, S1r'length));
        S1i<= signed(SXT(DII, S1I'length));
        S2r<=S1r − S2r;
        S2i<=S1i − S2i;
        R2r<= S1r;
        R2i<= S1i;
        S4r<= SHR(S3r, "010") − R1r;
        S4i<= SHR(S3i, "010") − R1i;
        S5r<=R1r − SHR(S4r, "011");
        S5i<=R1i − SHR(S4i, "011");
        S6r<= R2r − S5i;
        S6i<= R2i + S5r;
     when 1 =>
        S1r<= S1r + signed(DRI);
        S1i<= S1i + signed(DII);
        R2r<= S2r;
        R2i<= S2i;
        S3r<= S1r − signed(DRI);
        S3i<= S1i − signed(DII);
        S5r<=S5r + SHR(S4r, "1010");
        S5i<=S5i + SHR(S4i, "1010");
        S6r<=R2r;
        S6i<=R2i;
     when others=>
        S1r<= S1r + signed(DRI);
        S1i<= S1i + signed(DII);
        S2r<= S1r + SHR(S1r, "001");
        S2i<= S1i + SHR(S1i, "001");
        S3r<= SHR(S3r, "010") − S3r;
        S3i<= SHR(S3i, "010") − S3i;
        S4r<= S3r + SHR(S3r, "0100");
        S4i<= S3i + SHR(S3i, "0100");
        R1r<=S3r;
        R1i<=S3i;
        S6r<= R2r + S5i;
        S6i<= R2i − S5r;
     end case;
   end if;
  end process;
  DRO<= std_logic_vector(S6r);
  DIO<= std_logic_vector(S6i);
end synt;
```

Algorithm 1: Continued.

Algorithm 1

Table 1: Results of configuration of the DFT modules.

DFT length	Hardware volume, LUTs + DSP48	Maximum clock frequency, MHz
3	245	640
3	201 + 2	433
4	215	548
5	945	435
8	1187	424
$15 = 3 \cdot 5$	2131	346
16	3616	368
$64 = 8 \cdot 8$	1985 + 4	338
$128 = 8 \cdot 16$	5277 + 4	324

sample per clock cycle, which provides the simple manner of connecting them in a system. Besides, the respective reorder buffers, based on the Xilinx SRL16 serial shift registers, were synthesized as well. This helps to design the DFT modules of the higher order on the base of the Good-Thomas algorithm, like $r = 15 = 3 \cdot 5$, which have the minimized hardware volume.

The results of configuring the modules in Xilinx Kintex-7 FPGAs for the 16-bit input data are shown in Table 1. To compare the effect of the use of the application specific multipliers, the example of mapping the radix-3 DFT module with two DSP48 multipliers is shown in the table as well. The comparison of both DFT modules shows that the clock frequency of the multiplier-free module can be increased up to 1.5 times.

The analysis of Table 1 shows also that the clock frequency of the module decreases with the increase of the transform length. This is explained by the fact that the ratio of delays in the routes to the critical path delay in FPGA achieves 80% and higher. Therefore, the place and route tool could not optimize effectively the large projects with a lot of interconnections.

Table 2: 64-point FFT processors configured in Xilinx FPGAs.

FPGA series	Hardware volume, CLBs + DSP48	Maximum clock frequency, MHz	Reference
Spartan-3E	758 + 8	170	[18]
	1063 + 12	116	[19]
	1984 + 4	127	Proposed
Virtex-5	695 + 24	384	[20]
	628 + 4	325	Proposed

The example of 64-point FFT processor is compared with similar processors in Table 2. Its advantages are small hardware volume in the number of DSP48 units and high clock frequency by the nonrestrictive constraints.

5. Conclusions

The implementation of the r-point DFT modules in FPGA provides the design of the high-speed pipelined FFT processors with optimized hardware volume. It is proven that the DFT module with the slowdown operation in r times has the high clock frequency and small hardware volume due to the pipelined calculations, properties of the 6-input LUTs, and application specific multipliers. The designed DFT modules were used to build the pipelined FFT processors with $N = 64$, 128, and 256, which are deposited in the free IP core site [36], and can be downloaded for investigation and use.

Our future work aims at design of the framework which provides automatic synthesis of pipelined FFT processors based on the DFT modules.

Competing Interests

The authors declare that there are no competing interests regarding the publication of this paper.

References

[1] P. Duhamel and H. Hollmann, "'Split radix' FFT algorithm," *Electronics Letters*, vol. 20, no. 1, pp. 14–16, 1984.

[2] H. J. Nussbaumer, *Fast Fourier Transform and Convolution Algorithms*, Springer, New York, NY, USA, 1982.

[3] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*, Prentice-Hall, Upper Saddle River, NJ, USA, 1975.

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

[5] S. He and M. Torkelson, "New approach to pipeline FFT processor," in *Proceedings of the 10th International Parallel Processing Symposium (IPPS '96)*, pp. 766–770, Honolulu, April 1996.

[6] G. Bi and E. V. Jones, "Pipelined FFT processor for word-sequential data," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 12, pp. 1982–1985, 1989.

[7] P. K. Meher, J. C. Patra, and A. P. Vinod, "Efficient systolic designs for 1- and 2-dimensional DFT of general transform-lengths for high-speed wireless communication applications," *Journal of Signal Processing Systems*, vol. 60, no. 1, pp. 1–14, 2010.

[8] J. G. Nash, "High-throughput programmable systolic array FFT architecture and FPGA implementations," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC '14)*, pp. 878–884, IEEE, Honolulu, Hawaii, USA, February 2014.

[9] M. Parker, *Optimizing Complex Floating Point Calculations on FPGAs*, Altera, 2014, http://www.embedded.com/.

[10] A. Chahardahcherik, Y. S. Kavian, O. Strobel, and R. Rejeb, "Implementing FFT algorithms on FPGA," *International Journal of Computer Science and Network Security*, vol. 11, no. 11, pp. 148–156, 2011.

[11] A. Saeed, M. Elbably, G. Abdeel, and M. I. Eladawy, "FPGA implementation of Radix-2^2pipelined FFT processor," in *Proceedings of the 3rd WSEAS International Symposium on Wavelets Theory and Applications in Applied Mathematics, Signal Processing & Modern Science (WAV '09)*, pp. 109–114, World Scientific and Engineering Academy and Society (WSEAS), January 2009.

[12] K. Harikrishna, T. R. Rao, and V. A. Labay, "FPGA implementation of FFT algorithm for IEEE 802.16e (Mobile WiMAX)," *International Journal of Computer Theory and Engineering*, vol. 3, no. 2, pp. 197–203, 2011.

[13] B. Zhou, Y. Peng, and D. Hwang, "Pipeline FFT architectures optimized for FPGAs," *International Journal of Reconfigurable Computing*, vol. 2009, Article ID 219140, 9 pages, 2009.

[14] V. N. Sudheer and V. B. Gopal, "FPGA implementation of 64 point FFT processor," *International Journal of Innovative Technology and Exploring Engineering*, vol. 1, no. 4, pp. 63–66, 2012.

[15] V. Gautam, K. C. Ray, and P. Haddow, "Hardware efficient design of variable length FFT processor," in *Proceedings of the 14th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS '11)*, pp. 309–312, IEEE, Cottbus, Germany, April 2011.

[16] F. Camarda, J.-C. Prevolet, and F. Nouvel, "Towards a reconfigurable FFT: application to digital communication systems," in *Fourier Transforms—New Analytical Approaches and FTIR Strategies*, G. S. Nikolić, Ed., chapter 10, pp. 185–202, InTech, Rijeka, Croatia, 2011.

[17] R. Bhakthavatchalu, M. Viswanath, P. M. Venugopal, and R. Anju, "Programmable N-point FFT design," *Lecture Notes on Software Engineering*, vol. 2, no. 3, pp. 247–250, 2014.

[18] Y. Ouerhani, M. Jridi, and A. Alfalou, "Implementation techniques of high-order FFT into low-cost FPGA," in *Proceedings of the 54th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS '11)*, pp. 1–4, IEEE, Seoul, Republic of Korea, August 2011.

[19] *Xilinx Product Specification. High Perfomance 64-Point Complex FFT/IFFT*, V.7.0, Xilinx, San Jose, Calif, USA, 2009, http://www.xilinx.com/ipcenter.

[20] J. Grajal, M. A. Sanchez, M. Garrido, and O. Gustafsson, "Pipelined radix-2^k feedforward FFT architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 23–32, 2013.

[21] E. Lee and D. Messerschmitt, "Synchronous data flow," *Proceedings of the IEEE*, vol. 75, no. 9, pp. 1235–1245, 1987.

[22] S. Edwards, L. Lavagno, E. A. Lee, and A. Sangiovanni-Vincentelli, "Design of embedded systems: formal models, validation, and synthesis," *Proceedings of the IEEE*, vol. 85, no. 3, pp. 366–389, 1997.

[23] *System Generator for DSP. User Guide*, UG640 (V 14.3), Xilinx, San Jose, Calif, USA, 2012, http://www.xilinx.com/.

[24] P. G. Paulin and J. P. Knight, "Force—directed sheduling for the behavioral synthesis of ASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 3, pp. 356–370, 1988.

[25] P. Micheli, U. Lauther, and P. Duzy, Eds., *The Systhesis Approach to Digital System Design*, Kluwer Academic Publishers, 1992.

[26] K. Keshab, K. K. Parhi, and Y. Chen, "Signal flow graphs and data flow graphs," in *Handbook of Signal Processing Systems*, S. S. Bhattacharyya, E. F. Deprettere, and R. Leupers, Eds., pp. 791–816, Springer, New York, NY, USA, 2010.

[27] Y. Robert and F. Vivien, Eds., *Introduction to Scheduling*, CRC Press, Taylor &Francis Group, 2010.

[28] O. Maslennikow and A. Sergiyenko, "Mapping DSP algorithms into FPGA," in *Proceedings of the International Symposium on Parallel Computing in Electrical Engineering (PARELEC '06)*, pp. 208–213, IEEE, Bialystok, Poland, September 2006.

[29] A. M. Sergiyenko, O. Maslennikow, and Y. Vinogradow, "Tensor approach to the application specific processor design," in *Proceedings of the 10th International Conference—The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM '09)*, pp. 146–149, IEEE Library, Lviv, Ukraine, February 2009.

[30] A. M. Sergiyenko, *VHDL for Design of Computers*, DiaSoft, Kiev, Ukraine, 2003 (Russian).

[31] R. Hartley and K. K. Parhi, *Digit-Serial Computation*, vol. 306, Springer, New York, NY, USA, 1995.

[32] A. Sergiyenko, V. Simonenko, Y. Vinogradov, and K. Gluchenko, "IP core synthesis in a cloud," in *Proceedings of the 3rd International Conference "High Performance Computing" (HPC-UA '13)*, pp. 350–353, Kiev, Ukraine, October 2013, http://hpc-ua.org/hpc-ua-13/files/proceedings/66.pdf.

[33] A. Sergiyenko, T. Lesyk, and O. Maslennikow, "Mapping DSP algorithms into FPGA," in *Proceedings of IEEE East-West Design & Test Symposium (EWDTS '08)*, pp. 343–348, KNURE, Lviv, Ukraine, October 2008.

[34] S. A. Khan, *Digital Design of Signal Processing Systems: A Practical Approach*, John Wiley & Sons, New York, NY, USA, 2011.

[35] *Synopsys Synthesis and Simulation Design Guide*, Ver. 2.1i, Xilinx, San Jose, Calif, USA, 1999.

[36] A. Sergiyenko and O. Usenkov, "Pipelined FFT/IFFT 128 points processor," 2010, http://www.opencores.org/project, pipelined_fft_128.