ESKİŞEHİR TECHNICAL UNIVERSITY JOURNAL OF SCIENCE AND TECHNOLOGY
A- APPLIED SCIENCES AND ENGINEERING

2018, 19(4), pp. 796 - 804, DOI: 10.18038/aubtda.423280

SINGLE BANK DUAL-PORT MEMORY-BASED FFT FOR FIELD PROGRAMMABLE GATE ARRAYS

Erol SEKE [1,*], Zeynep KAYA[2]

[1] Department of Electrical-Electronics Engineering, Faculty of Engineering & Arch., Eskişehir Osmangazi University, Eskişehir, Turkey
[2] Department of Electrical-Electronics Engineering, Faculty of Engineering, Bilecik Şeyh Edebali University, Bilecik, Turkey

ABSTRACT

A new efficient memory-based FFT calculation method is presented using dual-port memories. Algorithm mainly targets Field Programmable Gate Arrays (FPGA). A semi-in-place calculation of FFT stages is presented to have both reads and writes in a single clock while keeping the memory size equal to the FFT size. The "semi-" word implies that the writes are not to the original/expected position. At each intermediate calculation, the outputs are written to the position where the reads are done so that the unused data is not overwritten. Compared to multi-bank memory FFT approaches, proposed memory addressing schema is both simpler to logically establish and requires lower count of logical elements. It is shown that the proposed approach accomplishes FFT task in lowest count of clock cycles among the single bank memory-based FFT algorithms. However, two-port single-bank size-N memory requirement limits the proposed design to radix-2.

Keywords: FFT, Single bank, Memory-based, Dual-port, FPGA

1. INTRODUCTION

Discrete Fourier Transform (DFT) is one of the main and most used operations in signal processing. However, it is computationally intensive in its basic form given as

$$X_k = \sum_{n=0}^{N-1} x_n W_N^{nk}\ ,\ k = 0,\ldots N-1 \tag{1}$$

where $W_N^{nk} = \exp(j2\pi nk/N)$. Since Cooley and Tukey [1] introduced a method that reduced computational complexity from N^2 to $N/2\log_2(N)$, a variety of computation algorithms were proposed and implemented ranging from memory-based to pipelined architectures for both software and hardware implementations. Requirements of the application mostly dictates how it is implemented since all methods have advantages and disadvantages in terms of speed, latency, design complexity, power dissipation, chip-surface area and number of logic elements when implemented in hardware. Similar arguments exist for software implementations too.

Radix-2 algorithms split *N*-point DFT into two *N*/2-point DFTs as

$$X_k = \sum_{m=0}^{N/2-1} x_{2m} W_{N/2}^{mk} + W_N^k \sum_{m=0}^{N/2-1} x_{2m+1} W_{N/2}^{mk} \tag{2}$$

and continue splitting until it is calculated using 2-point DFTs. Since the number of operations (adds and multiplies) are reduced, it is calculated faster, hence the name Fast Fourier Transform (FFT) is started being used, albeit FFT is no different than DFT other than the calculation order. For the splitting

*Corresponding Author: eseke@ogu.edu.tr
Received: 14.05.2018 Accepted:02.10.2018

can be done, the number of points at each sub-parts must be a multiple of *r*, called the radix. If $N = r^m$, where *m* is an integer, entire FFT can be implemented by $N \log_r(N)/r$ *r*-point FFTs.

An example radix-2 N=16 input points FFT calculation structure is given in Figure 1. Entire FFT is calculated by the same 32 2-point FFT architecture. These 2-point FFT structures are called processing elements (PE) and all PEs are made of a butterfly like sum-difference connections and a multiplier. All PEs are same except the constant multiplier at the lower-leg of the butterfly. This structure lends itself to recursive/repetitive algorithms that use a single PE shown in Figure 2.

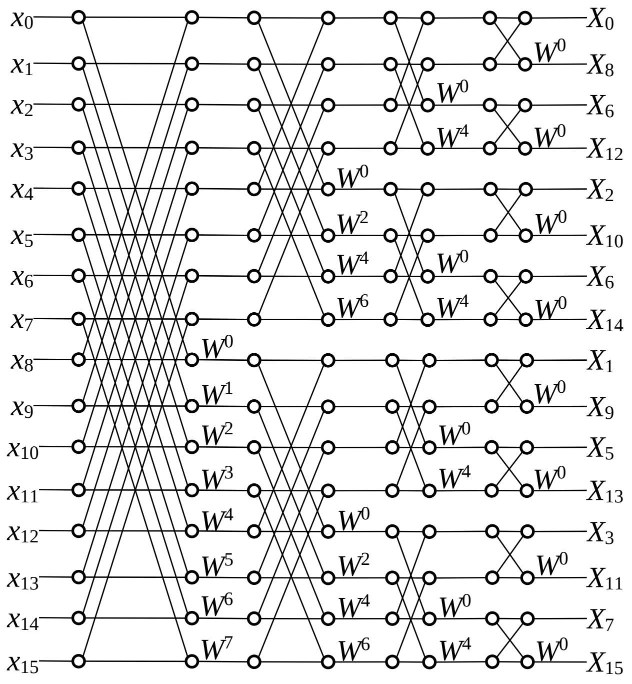

Figure 1. N=16 point decimation in frequency (DIF) FFT

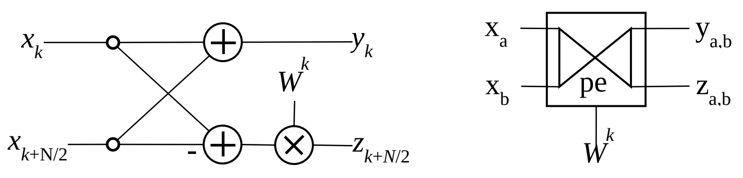

Figure 2. A single PE (first stage signals are shown)

When *r* gets bigger, algorithmic complexity is further reduced but data transfer paths get more complex. In this paper, a radix-2 memory-based FFT method is proposed. The term "memory-based" usually mean that the data (and intermediate data) are held in conventional memory blocks and they are retrieved when needed for calculations. Conventional memory usage has implications/limitations on the number of read/write data paths. However, many researchers [2-8] proposed methods to overcome this limitation by dividing memory into blocks that have separate addressing circuitry (address decoders).

In general, storage area requirements are much larger than chip areas required by processing, address generation and twiddle factor generation/storage. Therefore, it is logical to attempt to reduce main and intermediate storage requirements by in-place FFT processing. In [2] a fixed radix algorithm on multibank memory proposed. Multibank memory structure is required when single port memories considered, in order apply a ping-pong style processing approach. Similarly, a constant geometry FFT implementation [3] also required multibank memory where they claim 50% area reduction. [7] uses coprime access technology to avoid memory access conflicts which might be faced in multibank designs. It should be noted that these approaches require at least 2N memory while improving throughput.

Multi-bank approaches [2-10], while allowing multiple reads/writes simultaneously and increasing throughput by implementing higher radix algorithms, also introduce additional circuit complexity for handling multiple memory banks. Most of the proposed algorithms [2-10] use larger memory than FFT-size for performance improvement which may hit resource limitations for larger FFT-sizes. In this paper, we aim to design an FFT processor that is simplest to understand, lowest in resources and lowest in operation clock counts. At the same time, we wished the processor to be expandable and easily modifiable when designed using HDL. Therefore, we opt to use a single block of dual-port memory that is available in many FPGAs. This, however, limits us to r=2 when r-point FFT completion is targeted in one single clock cycle.

2. PROPOSED METHOD

2.1. Use of Dual Port Memories

Dual port memory blocks are in wide-spread use in FPGAs. Although the second port introduces additional address decoder logic, many digital designers prefer them in the designs where two writes and reads are needed at the same time, simplifying the design and eliminating the need for additional clock cycle. These are especially useful in the projects with stringent real-time requirements. A dual port memory block can be modeled as illustrated in Figure 3.
Dual port memory blocks usually have different operating modes for different requirements of the application. Read-Before-Write mode can be used to output data from a memory location while writing new data into the same location at the same time. In fact, this is the mode that is used in designing the FFT processor presented in this paper.

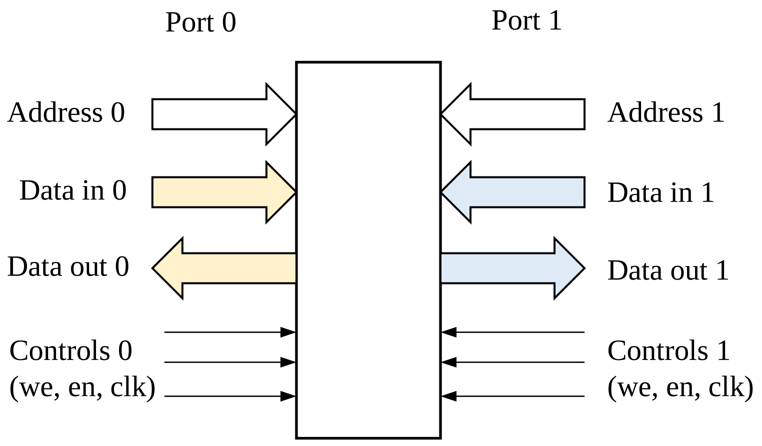

Figure 3. Dual port memory block diagram illustrates that the same memory block can be addressed by two ports independently. Conflicts occur when one port try to write at the same address the other port is writing/reading.

Since memory output ports are buffered until the next clock cycle, there is no need for additional buffer. However, depending on the non-existence of an output buffer, one may need to add a buffer with the width of the stored data at a small surface area cost.

The key to calculate entire single-block-memory-based FFT in minimum number of clock cycles is to read new data for processing while writing previously calculated values into memory simultaneously. With single block memory, this can only be done when read and write operations are on the same memory address. Since addresses must be updated at the same clock pulses that reads/writes occur, values calculated for the memory positions (i,j), for example, cannot be written back to (i, j) as implied in Figure 1. However, calculated values can be written to next read positions and values at these positions can be read at the same time.

The situation is illustrated in Figure 4. In a clock cycle, 2 positions are read through ports P0 and P1 (shown on the left) and two positions are filled with current PE outputs (shown on the right). Memory positions for reading and writing must be the same since there are only two ports. However, written data is not related to data currently being read but belongs to the previously read ones.

Memory read/write operations for the first FFT stage is illustrated in Figure 5 where input data $(x_0,\cdots,x_{N-1})$ are shown as prewritten into memory positions $(0,\cdots,N-1)$ and memory buffers have unrelated initial contents. Address lines carry the addresses of very first data to be operated on initially. At the first clock, values calculated from these unrelated buffer contents are written into first memory positions, at the same time these positions are read into buffers and addresses are updated to point at the next read positions. Before the next clock pulse, combinatorial PE outputs are stabilized and becomes ready to be written back, but into the next memory positions which are pointed at by new address values. These reads and writes continue until all processing is completed for the first FFT stage. However, results are not actually in their original place but in the next memory positions. That is, when the first stage is completed, memory addresses $(0,\cdots,N-1)$ contain intermediate data $(z_{N/2-1,N-1}, y_{0,N/2}, y_{1,N/2+1},\cdots,z_{N/2-2,N-2})$. It would have contain $(y_{0,N/2}, y_{1,N/2+1},\cdots,z_{N/2-2,N-2},z_{N/2-1,N-1})$ if the operation were really in-place. This rotation allows us to complete a single PE calculation in a single clock period.

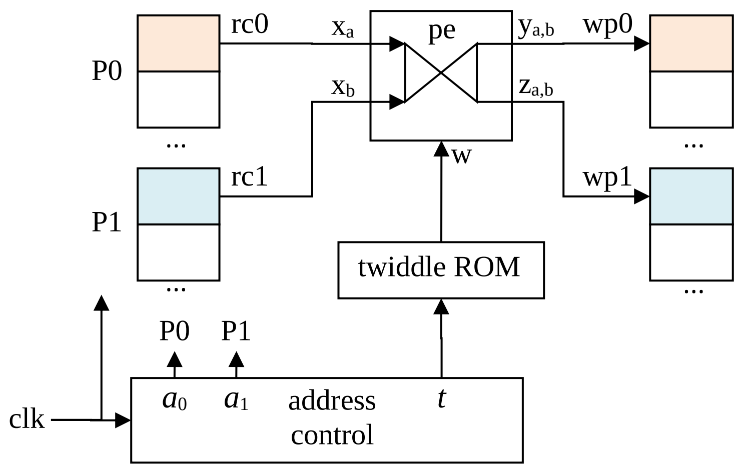

Figure 4. Data transmisson to and from the processing element. Two inputs of PE receives data from two read ports of memory and two outputs of PE are written through two write ports. a_0 and a_1 are the address lines of the first and second ports respectively.

After completion of first stage in (N/2+1) clock cycles, second stage starts by reading pairs with N/4 positions apart using the same strategy. This stage starts with positions addressed as 1 and N/2+1, not like the first stage which started from 0 and N/2. Therefore, there must be a single clock cycle spent for address adjustment and reading first pairs as done at the startup of the first stage. This extra clock cycle is needed not only when moved to the next stage but also whenever write positions are not to be read. At each stage, there are 2^s butterfly clusters each consisting of $N/2^{s+1}$ butterflies where s is the stage number starting from 0. For example, in Figure 1, there is 1 cluster of 8 butterflies in stage 0, there are 2 clusters of 4 butterflies in stage 1, and so on. Since the last write addresses in a cluster are not the same with the first read addresses in the next cluster, there must be an address adjustment requiring an additional clock cycle. One exception for this is the last stage where each cluster has only one butterfly and allows writing calculated values to the 2-ahead memory positions instead of 1. That is, each PE output can be written into the next butterfly input data positions, eliminating the need for extra clock cycles between clusters.

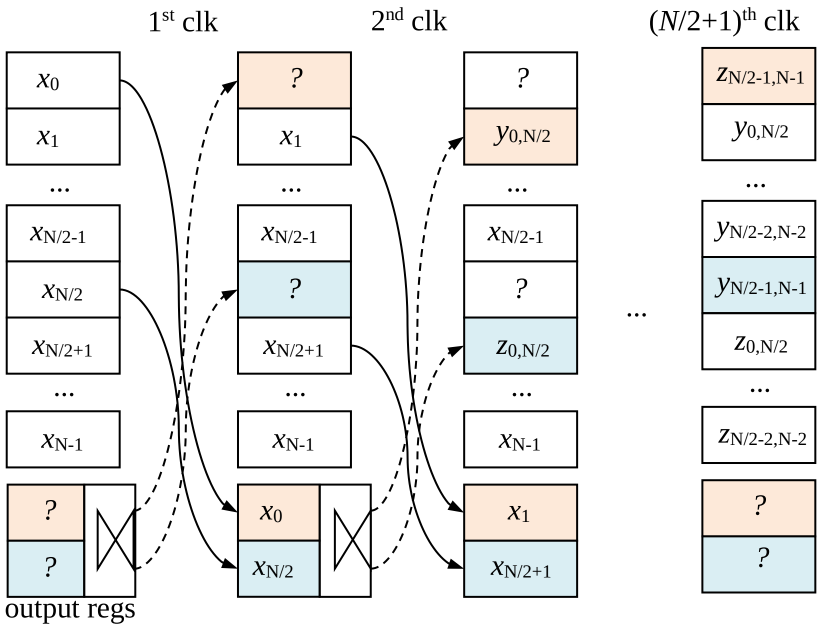

Figure 5. Data transactions at the first stage. Since the memory output buffers initially contain unrelated data, first operational clock just fills the buffers with the first pair of data to be sent to PE and writes initial content of the buffers into the positions of the data just read. At the same time, addresses are changed to the next read/write position so that the PE outputs are written to the next position at the next clock cycle. Therefore, no data is lost but results are always written into the next positions.

With such an approach described here and used in this paper, all stages (complete FFT) are completed in $\frac{N}{2}\log_2(2N)=\frac{N}{2}(1+\log_2 N)$ clock cycles. This is the lowest count of clock cycles among the algorithms that use single PE and single block memory in the literature and this one is not in the literature yet. In fact, single memory block algorithms are extremely rare. PE utilization is close to 100% for large FFTs. Divergence from 100% is caused by the extra clock cycles spent between clusters.

2.2. Memory Address Generation

Address control circuitry shown as a block in Figure 4 is binary counter based. Current stage number (ranging from 0 to $\log_2(N)-1$), current cluster number (ranging from 0 to 2^s-1) and process number in a cluster (ranging from 0 to $N/2^{s+1}-1$), are kept in three counters named as s, c and a respectively. For simplicity, all values are kept in registers/counters with $ceil(\log_2 N)$ bits. Figure 6 shows the values of these counters at each calculation step of 16-point FFT. a_0 and a_1 are address lines applied to memory ports connected to upper and lower legs of PE respectively. t is the address for the twiddle rom in which each position holds twiddle factor W^t. In figure, the row values show the values of counters s, a, a_0, a_1 and t, as appropriately indicated by column headers, at the rising edge of the clock pulse. t=x marks the points where the results of the previously calculated values (currently at the output) of PE are being written but the values read at the same time are to be ignored. Since these values are not used, the value of t is also unimportant, hence it is marked as x, meaning "don't care". At the rows just below the xs, new values are read from the new cluster but the values written at the same time are meaningless (PE outputs for the irrelevant values previously read). One can avoid writing of these dummy values by replacing them with 0s but it is not necessary as they will be replaced by actual values in the subsequent operations. Nevertheless, at these points, value written at address a_1 actually overwrites the value required in the following read. Therefore, the value read at the same time must be kept in a register to be used afterwards. For example, the values read from a_1=9 in the stage s=1 (Figure 6) must be stored for the subsequent operation that uses this value. This is the only place where a data commutator (mux) is needed in the entire proposed FFT architecture (not counting address decoders of memories).

The net result at the end of every stage is that stage output values are placed at their succeeding positions and the last value will be written at memory position 0 (all positions are rotated down). This poses no problem as long as an appropriate offset value is added to the addresses in the subsequent stages. Figure 6 shows a_0 and a_1 with added offset value. Offset value is actually the value of the s register. Calculating a_0 and a_1 addresses using arithmetic operations (addition) instead of keeping individual counters is trivial since their widths are small compared to actual data widths, and synthesizers use simple combinatorial logic blocks for these additions.

s=0					s=1					s=2					s=3				
c	a	a_0	a_1	t	c	a	a_0	a_1	t	c	a	a_0	a_1	t	c	a	a_0	a_1	t
0	0	0	8	0	0	0	1	5	0	0	0	2	4	0	0	0	3	4	0
0	1	1	9	1	0	1	2	6	2	0	1	3	5	4	1	0	5	6	0
0	2	2	10	2	0	2	3	7	4	0	2	4	6	x	2	0	7	8	0
0	3	3	11	3	0	3	4	8	6	1	0	6	8	0	3	0	9	10	0
0	4	4	12	4	0	3	5	9	x	1	1	7	9	4	4	0	11	12	0
0	5	5	13	5	1	0	9	13	0	1	2	8	10	x	5	0	13	14	0
0	6	6	14	6	1	1	10	14	2	2	0	10	12	0	6	0	15	0	0
0	7	7	15	7	1	2	11	15	4	2	1	11	13	4	7	0	1	2	0
0	7	8	0	x	1	3	12	0	6	2	2	12	14	x	0	0	3	4	0
					1	3	13	1	x	3	0	14	0	0					
										3	1	15	1	4					
										3	2	0	2	x					

Figure 6. Generated data memory and twiddle rom addresses for N=16.

Obviously, implied addressing circuitry can also be realized in several other ways. Registers come quite cheap in FPGAs. Therefore, instead of having a single counter and generating all other signals from it, assigning their own counters and related logic to all multibit logic signals is preferable for reducing combinatorial circuitry and consequently improving operation frequency. Whatever the design criteria are, it can be pre-said that PE combinatorial logic is probably the critical path for the complete FFT design as it will contain complex adders followed by a complex multiplier with desired precision. Therefore, it can be concluded that, unless PE is improved for higher clock frequencies, the rest of the design should have no impact on operating frequency. Number of clock pulses from the beginning to the completion of last butterfly of the last stage is actually what counts and comparable to other similar FFT processor architectures. Similarly, as most of the used area on the chip is occupied by the memory as expected for large N, efficient memory usage (excluding discrete registers) is also comparable. The proposed design is compared to other memory-based FFT architectures in conclusion section.

2.3. Twiddle ROM

Twiddle ROM supplies complex multiplier constants to PEs. A multitude of researchers worked on twiddle factors to reduce the memory size required to hold them. The work is actually in parallel with the sinusoidal signal generation using direct digital synthesis (DDS) methods since storage area (for storing *sine* signal samples) reduction is desired in both [11, 12]. For an N-point FFT there are N/2 complex multipliers that need either to be stored or calculated in run-time. Reduction is generally based on the symmetry of twiddle factors on unit-circle (
Figure 7b). In this work, basic half symmetry is implemented. That is, no compression is implemented and for an *N*-point FFT, size N/2 memory (ROM) is needed.

Figure 1 and Figure 6 together show that the constants stored in a memory can easily be addressed by the address counter *a* when the twiddle ROM size is *N*/2 and *s*=00. Content of this memory is illustrated in
Figure 7a, along with the corresponding points on unit circle on *r*-*i* complex plane (
Figure 7b).

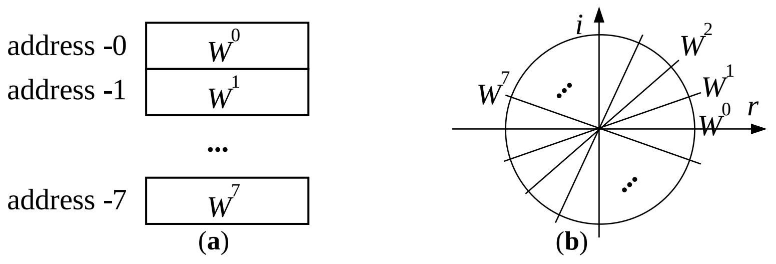

Figure 7. (a)Twiddle ROM contents for *N*=16. (b) twiddle constants on complex plane.

For s=00, t (twiddle ROM address signal) is equal to a. In general, t can be calculated by left-shifting a by s times and having only least significant $\log(N)-1$ bits. That is, when *s*=0, *t*=*a*, when *s*=1, t=2*a*, when *s*=2, t=4*a*, etc. It is obvious that, for the last stage, all bits of *t* will be 0 since $length(a) = range(s)$. It can also be noticed that $\mathrm{Re}\{W^t\} = -\mathrm{Re}\{W^{N/2-t}\}$ and $\mathrm{Im}\{W^t\} = \mathrm{Im}\{W^{N/2-t}\}$, meaning that half of these values can be easily calculated from the other half. That is, when twiddle ROM size is *N*/4, *t*=*N*/2-*t'* address conversion is applied when the calculated address *t'* is greater than *N*/4 in order to find the actual driving address. And when this is done, sign of the real part read from memory is changed to obtain the actual value. For small FFTs, this is not feasible, considering the added circuitry for comparison, subtraction, sign change etc. On the other hand, for large FFTs, such an implementation can save considerable ROM area. For larger FFTs, one might even consider using 4-fold symmetry which involves switching real and imaginary parts too, requiring data-wide multiplexers. Even for much larger FFTs, twiddle factors can be and are calculated using an iterative method, requiring the storage of only a single complex constant. Since this paper is on FFT calculation part, we stored all required multipliers in an *N*/2 sized memory in the experimental implementation.

3. CONCLUSION

A simple radix-2 FFT method that uses a single memory block is presented. The proposed approach is implemented using VHDL and tested on Xilinx FPGAs with dual-port memory blocks. Since we aimed for minimum memory size and simplest architecture, we limited ourselves to radix-2 and single memory block. Consequently, clock counts are higher than other comparable algorithms that implement higher radices and multiple memory blocks. Higher radices and multiple memory blocks mean higher circuit complexity and complex algorithms for obtaining lower latency. The biggest advantage of our approach is, therefore, required memory size and simplicity of the design. There is no single block dual-port memory-based FFT architecture that uses memory size equal to FFT size in the literature. Nevertheless, we compared our approach against some popular and new algorithms in Tables 1 and 2 for their clock cycles, memory size and number of memory blocks.

We conclude that the proposed approach can find usage in applications where hardware resources are scarce and latency is not the main concern but where soft approaches cannot just fulfil the latency requirements.

Table 1. Comparison of clock counts of various approaches

	Proposed		[6]		[2], [3]*		[7]	
Size	cycle	*r*	cycle	*r*	cycle	*r*	cycle	*r*
512	2560	2	-	-	-	-	640	mix
1024	5632	2	1024	2/8	2560	4	2560	4
2048	12288	2	2048	4/8	22528	2	6144	2/4
4096	26624	2	4096	8	4096	8	12288	4
8192	57344	2	10240	2/8	106496	2	28672	2/4

- : not given
* : obtained from [6]

Table 2. Comparison of memory size and number of banks

	Proposed	[6]	[2],[3],[7]	[4]
Memory	N	2N	2N	2N
#Banks	1	4/8	2+	5/7

ACKNOWLEDGEMENTS

This research has been funded by Eskisehir Osmangazi University Scientific Research Commission with project no 2015-728.

REFERENCES

[1] Cooley JW, Tukey JW. An algorithm for the machine calculation of complex Fourier series. Math Comput 1965; 19: 297-301.

[2] Johnson LG. Conflict free memory addressing for dedicated FFT hardware. IEEE Trans Circuits Syst II, Analog Digit Signal Process 1992; 39, 5: 312–316.

[3] Hidalgo JA, Lopez J, Arguello F, Zapata EL. Area-efficient architecture for fast Fourier transform. IEEE Trans Circuits Syst II, Analog Digit. Signal Process 1999; 46, 2: 187–193.

[4] Chen J, Hu J, Lee S, Sobelman GE. Hardware efficient mixed radix-25/16/9 FFT for LTE systems. IEEE Trans Very Large Scale Integr (VLSI) Syst 2015; 23, 2: 221–229.

[5] Xia K, Wu B, Xiong T, Ye T. A memory-based FFT processor design with generalized efficient conflict-free address schemes. IEEE Trans Very Large Scale Integr (VLSI) Syst 2017; 25, 6: 1919-1929.

[6] Hsiao CF, Chen Y, Lee CY. A generalized mixed-radix algorithm for memory-based FFT processors. IEEE Trans Circuits Syst II, Express Briefs 2010; 57, 1: 26–30.

[7] Jo BG, Sunwoo MH. New continuous-flow mixed-radix (CFMR) FFT processor using novel in-place strategy. IEEE Trans Circuits Syst I, Reg Papers 2005; 52, 5: 911–919.

[8] Baek J, Choi K. New address generation scheme for memory based FFT processor using multiple radix-2 butterflies. In: Proc. Int. SoC Design Conf; Nov 2008; 1:I-273–I-276.

[9] Altera. DFT/IDFT Reference Design 2007; [Online]. Available 2018: http://www.altera.com.

[10] Xilinx. LogiCORE IP Discrete Fourier Transform V9.0 2017; [Online]. Available 2018: http://www.xilinx.com.

[11] Sansaloni T, Pérez-Pascual A, Torres V, Valls J. Scheme for reducing the storage requirements of FFT twiddle factors on FPGAs. J VLSI Signal Proc 2007; 47: 183-187.

[12] De Caro D, Strollo AGM. High-Performance Direct Digital Frequency Synthesizers Using Piecewise-Polynomial Approximation. IEEE Trans Circuits Syst I:Regular Papers 2005; 52, 2: 324-337.