Enhanced RAM-less Modular 2-Dimensional Pipelined FFT

Maan Musleh Mokhtar Aboelaze
Department of Computer Science and Engineering
York University
Toronto ON CANADA
email: {maan, aboelaze}@cse.yorku.ca

***Abstract*—Discrete Fourier Transform is one of the most important operations in digital signal processing. DFT is used in many applications in communication systems and digital signal processing. DFT is also used in UWB OFDM and in IEEE802.11n. Energy consumption in portable and wireless devices is a crucial factor in the design of such systems. In this paper, we propose an FFT processor based on the famous pipelined FFT architecture. Our design does not require any RAM and replaces it with buffers. Our design is based on 2-D FFT and does not require the power hungry RAM, instead the RAM is replaced by delay lines and a simple circuit to configure the delay lines. Our design consumes less energy compared with processors with RAM, and can handle any size FFT (not only squared size).**

***Index Terms*—Fast Fourier Transform, Modular FFT architecture, Pipelined FFT, Low power**

I. Introduction

The Discrete Fourier Transform (DFT) remains a critical computation in several digital communication as well as signal processing applications. The DFT is widely employed in analyzing frequencies contained in discrete signals, solving partial differential equations, and performing other operations such as convolution. The multitude of research devoted to enhance the DFT algorithm and hardware in terms of speed, time, and power underlines its huge importance. Several DFT implementations were proposed for all hardware systems from general-purpose computers to systolic arrays and special purpose hardware.

The most common algorithm for performing the DFT is the fast Fourier Transform (FFT) proposed in [5]. The algorithm is efficient enough to perform the same traditional DFT, $O(N^2)$ complexity, in $O(N \log N)$ time. A pipelined implementation of the algorithm [8] was later developed. The implementation consists of several modules sharing a common memory storage - of coefficient factors. Each module consists of a specific number of delay elements, multipliers and adders. In addition, each module contains two switches controlled by an external control circuit. A simple address generator proposed in [4] is commonly used to supply the ROM with the address of the specific coefficient to be used by one or more modules at a specific clock cycle. The proposed generator simply depends on a single counter to obtain memory addresses and switching logic.

– – – –
This research is funded by a grant from the National Science and Engineering Research Council (NSERC) of Canada

In the remaining parts of this section, we will briefly mention some recent attempts for designing low energy fast FFT processors. In [3] The authors designed a low power FFT processor using many low power design techniques such as clock gating, low power library cells, and operand isolation. In [2] the authors proposed a multi-path delay commutator structure. Their proposal doubled the throughput of the previous radix-2 and radix-4 FFT processors, which makes it attractive for high throughput long-length FFT processors. However, the authors did not consider the energy as an issue in their design.

[11] presented a design for 128/64 point FFT/IFFT processor. The authors used a multipath delay feedback architecture. Their design objective was a processor for multiple-input multiple-output (MIMO) orthogonal frequency- division multiplexing (OFDM) based IEEE 802.11n wireless local area network baseband processor. Their design can calculate 128-point FFT with four independent data sequences within 3.2 μsec which makes it able to meet the IEEE 802.11n standard requirement

Jinag in [9] proposed a high performance architecture for long sequence FFT (8k sequence) for OFDM digital video broadcasting. He used distributed arithmetic in order to reduce the cost of multipliers and adders. He also used CORDIC multipliers for the twiddle factor multiplication. His main objective tis to reduce the area cost of the design without sacrificing the speed. The authors in [12] proposed an FFT processor for a Digital Audio Broadcasting (DAB) receiver. They used in their design the principle of circuit-sharing pipeline design, where circuit redundancies between modules or blocks of different implementations of different standards are utilized in order to decrease the chip are required for the overall implementation of the system. They implemented their design using TSMC CMOS with 0.35 μm technology. Their implementation showed a reduction in the hardware overhead.

In [13] the authors presented a trade off analysis for fixed point memory based FFT processors. They also implemented their design on a Xilinx Spartans FPGA. Chen et al. in [1] proposed an architecture for long variable size FFT processor. They implemented their design on FPGA for variable size $N = 4^n, n = 1 \rightarrow 10$ points. Their design can work on speeds up to 150MHz. In [10] The authors proposed an FFT/IFFT processor for mobile in-car entertainment system using wireless broadband. Using partially oversampling architecture they showed that their design reduces the complexity by 37% and requires small aera for implementation.

There was no other implementation that was able to compete with the pipeline FFT architecture [8] in terms of latency. However, several attempts were made to improve power consumption and gate count. A modular pipelined FFT [6] implementation, using two pipelined FFTs and specialized central elements, introduces substantial savings in the amount of delay elements and distributed coefficient storage needed. The specialized central elements for a modular radix-r FFT uses r RAMs, r ROMs, r complex multiplier, and an address generator. The modular implementation assumes that the pipelined FFT produces r outputs at a time. Assuming the pipeline FFT produces a single output at a time, then the central elements will reduce to one RAM, one ROM, and one complex multiplier. The ROM(s) and the RAM(s) need to be accessed simultaneously, and thus requiring a more complex address generator that can provide addresses to ROM(s) and RAM(s) at the same time.

In addition, the modular pipelined FFT algorithm works only for squared sizes of input such that $M = N^2$. That means if an application needs to compute the DFT of an input of size 512, it has to append 512 zero values to the input and release 512 more output data having the value of 0. This case is major disadvantage in terms of efficiency and utilization.

The proposed RAM-less modular FFT replaces the RAM with specialized delay elements requiring simple control logic. This design will eliminate the extra power consumed by the RAM and its control logic. It will also eliminate the ROM and the complex multiplier by tweaking the second pipelined FFT as explained in Section II. The proposed scheme joins two pipelined FFT modules of sizes R and C, $RC = N$, with specialized central elements consisting solely of specialized delay elements and a simple control logic.

II. 2-D-like Fast Fourier Transform

The basic idea behind the 2-D-like FFT is that the input data of size N can be rearranged in a 2-Dimensional rectangle of size RC. Figure 1 shows that the ith input data in a 1-D form can be represented using two iterators r and c in a 2-D form input data using the following transformation $i = rC + c$, where $0 \leq r < R$ and $0 \leq c < C$. The one dimensional DFT is given by Equation 1

$$X(n) = \sum_{k=0}^{N-1} x(k) e^{-j\frac{2\pi nk}{N}} \qquad (1)$$

A commonly used notation replaces $e^{-j\frac{2\pi nk}{N}}$ with W_N^{nk}. This expression is commonly called the *twiddle factor*.

In order to transform Equation 1 to the 2-D like form of DFT, the summation has to be replaced by double summation using two iterators r and c replacing the iterator

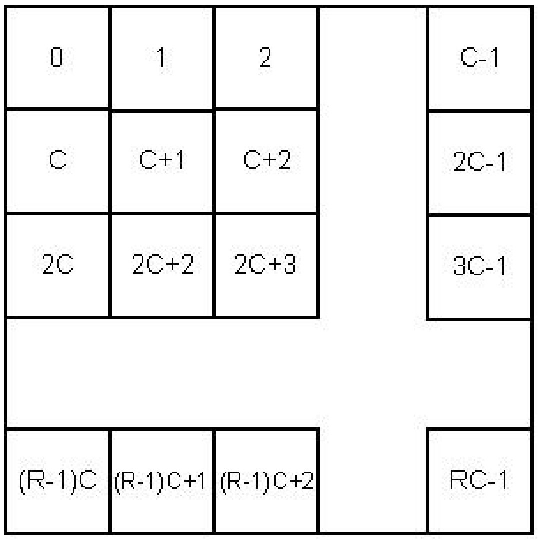

Fig. 1. data input rearranged in 2-D form

k.

$$X(n) = \sum_{r=0}^{R-1} \sum_{c=0}^{C-1} x(rC + c) W_N^{n(rC+c)} \qquad (2)$$

Equation 2 can be simplified more by realizing that $x^{(a+b)} = x^a x^b$. Using this property will lead to having two twiddle factors in the 2-D-like equation(W_N^{nrC} and W_N^{nc}). An extra simplification will help understand the mechanism of our design. The twiddle factor W_N^{nrC} equals to W_R^{nr} since $N = RC$. The equation will then yield to:

$$X(n) = \sum_{r=0}^{R-1} W_R^{nr} \sum_{c=0}^{C-1} x(rC + c) W_N^{nc} \qquad (3)$$

Equation 3 helps understand the logic behind our proposed hardware design. The inner sum of (3) represents an N-point FFT for each group of C items. There will be exactly R outputs. Those R outputs will be the input for an R-point FFT (outer sum). Let us represent an N-point DFT function that takes an input of size C by $F_N^C(x)$. That means, for $N = RC$, an N-point DFT function, $F_N^N(x)$, can be done by a function of a function of DFT, $F_R^R(F_N^C(x))$

III. Modular Pipelined FFT

The modular pipelined FFT [6] consists of two $\sqrt{N}$-point standard pipelined FFT coupled with central elements and address generation logic. The address generation logic is based on the work done by Cohen [4].

In their paper [7], Swartzlander et al. derived a 2-D like FFT equation. The derivation presumes a squared-size input ($N = M^2$) of data. The input is arranged in a square of size (M X M), while the output generated by the FFT is in the form of a transposed square of the same size.

$$X(Mk_1 + k_0) = \sum_{m_0=0}^{M-1} W_N^{m_0 k_0} \sum_{m_1=0}^{M-1} x(Mm_1 + m_0) W_M^{m_1 k_0} W_M^{m_0 k_1} \qquad (4)$$

The derived equation (4) tells more about the architecture used. It is apparent from the equation that two M-point FFTs are performed. The output of one of the FFTs is multiplied by a complex multiplier - $W_M^{m_0 k_1}$ - and becomes an input for the second FFT. The block diagram in Figure 2 shows the different components of the Modular Pipelined FFT [7].

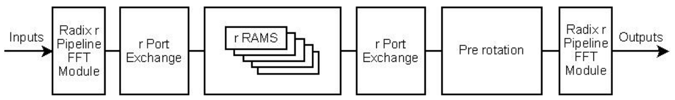

Fig. 2. Block Diagram of Radix-r Modular Pipeline FFT

The following sections explain the components of the Modular Pipeline FFT: the pipelined FFT module (A), the centralized elements (B) and the address generator (C).

A. Pipelined FFT

Several attempts were made to achieve a highly efficient FFT processor, nevertheless, there was no implementation that was able to improve in terms of speed on the Pipeline FFT processor [8]. The pipeline FFT is organized into different modules operating simultaneously. Relating this to the basic theory of FFT, each module represents a specific stage in the FFT computation The block diagram in Figure 3 shows how the different modules co-operate together to run the FFT computation. Each module in the processor accepts three inputs and produces one output. The first input is supplied by the previous module; the input of the first module comes from the main input stream. The second input comes from the ROM that stores rotation vectors. The third input is a control signal coming from the binary counter to switch the mode of operation of the module.

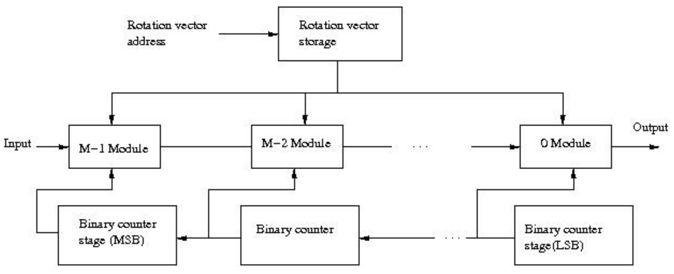

Fig. 3. Block Diagram for a Pipeline FFT Processor

Each operating module consists of delay elements, coefficient storage, commutators, multipliers and adders. All modules share a common memory where the coefficients (twiddle factors) are stored. The number of processing modules required to perform the Fast Fourier Transform depends on the size of input and the radix. To perform a radix r N-point FFT , $\log_r N$ modules are required. In theis discussion, an integer prefix is added to the name of the module to distinguish the different modules. This prefix is equal to the number of delay elements in the module.

In a radix-2 16 points FFT, 4 modules are required to perform FFT. The modules are arranged from input to output as follows: 2^3-module, 2^2-module, 2^1-module, and 2^0-module. As was mentioned before, the prefix integer indicates the number of delay elements in each module. That means the number of delay elements for each of the modules are 8, 4, 2, and 1, respectively.

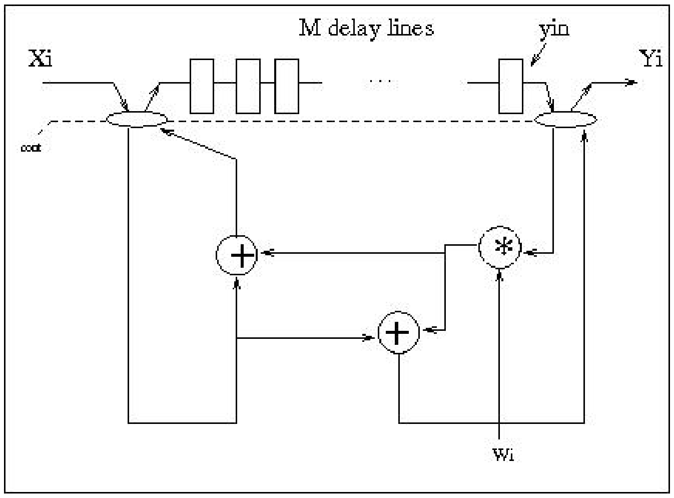

Fig. 4. M-Module of Pipelined FFT

Each module takes a complex input, twiddle factor (complex coefficient), and a control signal and produces one complex output. The complex output of one module is the complex input of the next module, with the exception of the last module in which the complex output is a single output of the system.

The signal *cont* controls the flow of data inside and between modules. The signal switches the module between two modes of operations. This paper refers to these modes as *Passive Mode* and *Active Mode.*

In the *Passive Mode*, the module accepts an input and passes it to the delay elements which in turn shift one position to the right. The right most element in the delay line becomes the output of the module. This output is in fact either the input for the next module or the output of the *Pipeline FFT.*

In the *Active Mode*, the module accepts an input $x_i n$ and passes it to the arithmetic unit. The rightmost element in the delay line, referred to as $y_i n$ in Figure 4, becomes the second input to the arithmetic unit. The arithmetic unit will need also a third input, W_i, to perform the required operations (5, 6.

$$x_o ut = x_i n + y_i n W_i \tag{5}$$

$$y_o ut = x_i n - y_i n W_i \tag{6}$$

The W_i is the twiddle factor taken from the *Shared Memory.* A very crucial step inside the pipeline FFT is to supply the modules with the correct twiddle factor at the correct time. A key factor that helps in simplifying this problem is the symmetrical nature of the different stages (resembled by the modules) in the pipelined FFT. In general, the M-module requires a different twiddle factor every

M cycles. Two important facts play a role in solving this problem. The first one is that M is always a multiple of 2. The other one is the fact that at any given clock cycle, all modules requiring a new twiddle factor ask for the same complex value. Thus the shared memory has to supply one twiddle factor per cycle and satisfy the demands, of all modules.

It is evident that the module accepts the input W_i only when it is in the *active mode*. In addition, the module accepts the input only when it is changing from passive mode to active mode (when the input data just fills the delay lines) and keeps using the same value of W_i throughout the active mode. However a new W_i will be accepted again when the module changes again from passive mode to active mode.

The switch between the active mode and the passive mode is controlled by the *Control Circuit* which in turn supplies the *cont* signal to the module. The advantage of the algorithm used in this design is that it simplifies the generation of the control signals for all modules to a simple counter of size $\lg N$ bits. Each bit of the counter supplies the signal for one module, where the most significant bit to the least significant bit supplies the signals for the first to the last modules respectively.

The *Control Circuit* in the pipelined FFT provides addresses to the *Shared Memory* and control signals to the individual modules. The *Control Circuit* consists of simply a counter of size $\log N$ [8]. The individual bits of the counter supply the control signals to the modules as shown in Figure A.

In sum, the N-point pipeline FFT processor contains N-bit counter, 32 X N bits memory storage, and $\log N$ M-modules. Each module contains two adders, one multiplier and M delay lines - summing up to N delay lines for all modules.

B. Centralized Elements

The modular pipeline FFT joins two standard pipeline FFT processors with specialized central elements. The central elements consists of RAM(s), ROM(s), multiplier(s), two switching circuits and an address generator. The RAM is used to store processed data generated by the first pipeline FFT. The switching circuit decides which RAM the data should be saved in. Once all data are generated and saved in the RAM(s), the address generator will generate addresses to read data from both the RAM(s) and the ROM(s). The ROM(s) contains rotation factors that should be multiplied by the data before they are streamed as input to the second pipeline FFT processor. Depending on the design the central elements can contain one or r of each of the RAM(s), ROM(s) and multiplier(s). The RAM(s) in total need to be large enough to save N complex data (this also applies to the ROM(s))

C. Address Generator

The last component in the modular design is the address generation logic. This is the crucial component that coordinates and connects all the distinct units together.

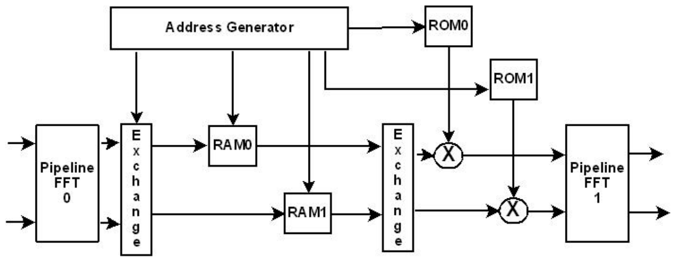

Fig. 5. Radix-2 Modular Pipelined FFT

As shown in Figure 6, the address generator is composed

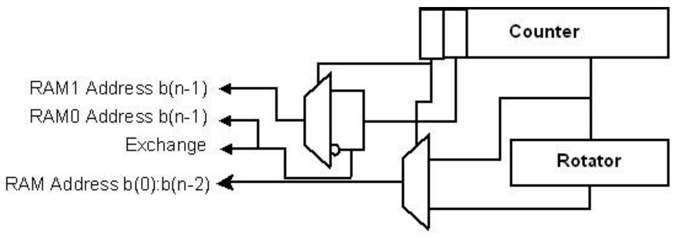

Fig. 6. Address Generator

of counters, multiplexers, and combinational logic [6]. The address generator creates addresses for each RAM unit and supplies signals to control the data exchangers [7]. Coefficient ROMs do not need special address generation logic; the coefficients can be preloaded to the ROM at the proper places for the counter to access them properly [6]. The size of the counter can be as large as $\log_2 N$. The highest bit of the counter determines the addressing mode. Depending on the the architecture of the central elements the rest of the bits will be used to supply signals to the exchanger (high bits) and generate addresses (lower bits). To give an example of this, we will assume a 16-bit radix-2 modular pipeline FFT with 2 RAMs and 2 ROMs in the central elements. The counter of the address generator needs 1 bit to change the addressing mode, 1 bit to specify which RAM/ROM to be used, and $\log_2 N/2$ bits to generate the address.

This architecture in Figure 5 is designed to handle a real-time stream of input data. When all outputs from the first pipeline FFT are written into the RAMs, the first pipeline FFT can accept input for another FFT operation. At some point, the address generator will generate 3 types of addresses simultaneously, an address to read from the ROMs, an address to read from the RAM, and an address to write to the RAM. That also implies that the RAM has to have simultaneous read-write access capability. This design, however, can be adapted to different implementations [6].

IV. Enhanced Ram-less Modular Pipelined FFT

In the proposed design, the central elements consists of solely delay elements controlled by a control circuit. Using the delay elements, the RAM can be eliminated. The ROM in the modular design [6] was used to supply the central multiplier with rotation factors before the data can be fed into the second stage pipelined FFT. The enhanced modular design modifies the second stage multiplier to eliminate the ROM(s) and the multiplier(s) from the central elements. The modification applied is in the twiddle factors saved in the internal ROM of the pipelined FFT. The twiddle factors saved in the ROM of the modified C-point pipelined FFT are of the N-th root of unity instead of the C-th root of unity. That will compensate for the lack of the rotation cofficients in our design.

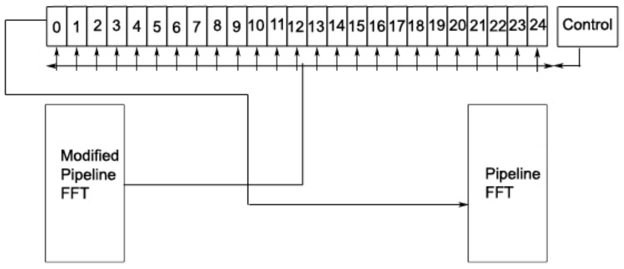

Fig. 7. Enhanced Ram-less Modular Pipelined FFT

Looking back at the derived function for the modular FFT, $F_N^N(x) = F_R^R(F_N^C(x))$, the function shows the one FFT operation will be the input of the other FFT operation. The twist is that the first FFT operation is of size C applying and N point FFT. This FFT is represented by the modified pipeline FFT in Figure 7.

The output of the modified pipeline FFT goes into the delay lines at specific positions defined by the control unit. This output will later be taken from delay0 register as an input for the second pipeline FFT. The position that the output has to be inserted in depends on the number of outputs left plus its order of input to the second pipeline FFT.

For any output i in an R x C located at row r and column c, the number of outputs left is the addition of the number of elements left in the same column and the number of rows left multiplied by the number of columns C, see A_i in Equation 7.

$$A_i = C - c - 1 + C * (R - r - 1) = N - Cr - c - 1 \quad (7)$$

For the same output i, the order of input to the second pipeline FFT is the addition of the number of elements above it in the same column and the number of columns preceding c multiplied by number of rows$R$, see B_i in Equation 8.

$$B_i = r + R * (c) = Rc + r \quad (8)$$

The control unit calculates $A_i + B_i + 1 = D_i(9)$ at every cycle i and inserts the corresponding output to the desired register. The logic which enforces one input at a time to the second pipeline FFT and one output at a time from first modified pipeline FFT ensures that no output will overwrite another in the delay registers.

$$D_i = N - Cr + r + Rc - c = N - r(C - 1) + c(R - 1) \quad (9)$$

It can be noted from Equation 9 that the maximum number of delay registers is $2N$. Table I shows a sample input orders for a 32-point FFT where $R = 8$ and $C = 4$.

TABLE I
Table of input orders for a 32-point FFT $R = 8$ and $C = 4$

i	r	c	A_i	B_i	D_i
0	0	0	31	0	32
1	0	1	30	8	39
2	0	2	29	16	46
3	0	3	28	24	53
4	1	0	27	1	28
.	.	.	.	.	.
.	.	.	.	.	.
15	3	3	16	27	44
16	4	0	15	4	20
17	4	1	14	12	27
18	4	2	13	20	34
19	4	3	12	28	41
.	.	.	.	.	.
.	.	.	.	.	.
28	7	0	3	7	11
29	7	1	2	15	18
30	7	2	1	23	25
31	7	3	0	31	32

The second stage FFT will give the output of the overall N-point FFT operation with the same order of the output. There will be no need to transpose the grid as in the modular pipeline FFT [6], [7].

V. Cost and Comparison

Comparing the proposed enhanced design with the modular design [6], [7], the main factors of difference recognized are in the number of delay elements and size of storage elements. The modular FFT uses less delay elements than the enhanced design while the enhanced design improves vastly in terms of storage units. In fact the N-point modular FFT uses $2\sqrt{N} - 2$ elements versus $2N - 1$ elements used by an N-point enhanced design. However, the N-point modular FFT uses $4N$ bytes of RAM and $4N$ bytes of ROM in the centralized elements and $4\sqrt{N}$ bytes of ROM inside the pipeline FFT(s), while the enhanced FFT design uses absolutely no storage units in the central elements and a sum of $2(R + C)$ bytes in the pipeline FFT(s).

The enhanced design also has a faster clock cycle than the modular design. The fact that no storage units are used in the central elements of the enhanced design gives room to improve the clock cycle since we eliminate the simultaneous read/write operations to the RAM(s) existing in the modular design. Table II compares bewteen the modular FFT [6], [7] and our proposed enhanced modular design with respect to the cost of the central unit. Note that the cost of the two pipelined FFT is the same for the modular and enhanced FFT.

Also our design does not use any multipliers in the central unit. For non-squared matrices, our esign require $2N$

buffers in the central element. Since we can write to any of them, that requires a decoding circuit equivalent to the same size memory. However for squared matrices, we write only to $2\sqrt{N}$ buffers; thus requiring a much simpler decoding circuit compared with the same size memroy.

TABLE II
Comparison of the cost of centrl unit

	Modular FFT	Enhanced FFT
Delay elements	$2\sqrt{N}-2$	$2N-1+2\sqrt{N}-2$
RAM	$4N$	–
ROM	$4N$	–
Multipliers	2	–

Another improvement over the modular design exists in the fact that the modular design does not accommodate non-squared sizes of N. For instance, to calculate a $8k$-point pipeline FFT using the modular design, one has to use $16k$-point modular pipeline FFT. That means we are using nearly 50% of the resources without any purpose contributing to the modular pipeline FFT.

Lastly, the elimination of the storage units in the central elements simplifies greatly the centralized control circuit and enables for better usage of the circuit area and power consumption. This design can be easily tweaked to meet any requirement for any given signal processing application.

VI. COnclusion

In this paper we proposed an enhanced RAM-less modular our design eliminates the RAM in the central element and replace it with buffers. Our design uses less buffer/RAM than the previous design, and uses also less multipliers. Our proposed design can be used with non-squared FFT length and require a shorter clock cycle since we eliminated the read/write to the RAM in the central element.

References

[1] H. Chen; W. Qiang; G. Zhenbin; and W. Hongxing "'A Pipelined Memory-efficient Architecture for Ultra-long Variable-size FFT Processors"' *Proceedings of the International Conference on Computer Science and Information technology* pp:357-361 Aug. 2008.

[2] C. C. Cheng and K. K. Parhi "'High-throughput VLSI architecture for FFT computation"' *IEEE Transactions on Circuits ans Systems* Issue 10, pp:863-867 Oct. 2007.

[3] K.-S. Chong; B.-H. Hwee; and J.S. Chang "'A low energy FFT/IFFT processor for hearing aid"' *Proceedings of the International Symposium on Circuits and Systems* pp 1169-1172 May 1007.

[4] D. Cohen, "Simplified control of FFT hardware," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. ASSP-24, pp. 577–579, 1976.

[5] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297–301, 1965.

[6] A. M. El Khashab and E. E. Swatzlander Jr., "A modular pipelined implementation of large fast Fourier," *he thirty-Sixth Asilomar Conference on Signals, Systems and Computers* vol. 2, pp. 995–999. Nov. 2002

[7] A.M. El-Khashab and E. E. Swartzlander, Jr., "An architecture for a radix-4 modular pipeline fast Fourier transform," *Application-Specific Systems, Architectures, and Processors, 2003. Proceedings. IEEE International Conference on*, pp. 378–388, 24-26 June 2003.

[8] H. L. Groginsky and G. A. Works, "A pipeline fast Fourier transform," *IEEE Transactions on Computers*, vol. C-19, no. 11, pp. 1015–1019, 11 1970.

[9] R. M. Jiang "'An area-efficient FFT architecture for OFDM digital video broadcasting"' *IEEE Transactions on Consumer Electronics* Issue 4, pp:1322:1326. Nov. 2007.

[10] D.-S. Kim; S.-Y. lee; and D.-J. Chung "'A partially operated FFT/IFFT processor for low complexity OFDM modulation and demodulation of WiBro in-car entertainment system"' *IEEE Transactions on Consumer Electronics* Issue 2, pp:431-436 May 2008.

[11] Y.-W. Lin; and C.-Y. Lee, "'Design of an FFT/IFFT processor for MIMO OFDM systems. *IEEE Transactions on Circuits and Systems I* Issue 4, pp:807-815. April 2007.

[12] C.-C. Wang; and Y.-C. Lin "'An Efficient FFT Processor for DAB Receiver Using Circuit-Sharing Pipeline Design"' *IEEE Transactions on Broadcasting* Issue 3, pp:670-677 Sept. 2007.

[13] H.G. Yeh; and G. Truong "'Speed and area analysis of memory based FFT processors in a FPGA"' *Wireless telecommunication Symposium WTS2007.* pp:1-6. April 2007.