A new FFT Architecture for 4×4 MIMO-OFDMA Systems with Variable Symbol Lengths

A. Karachalios, K. Nakos, D. Reisis
National and Kapodistrian Univ. of Athens
Department of Physics
Electronics Laboratory
email: dreisis@phys.uoa.gr

H. Alnuweiri
Texas A&M University at Qatar
Electrical & Computer Engineering
Doha, Qatar
email: hussein.alnuweiri@qatar.tamu.edu

Abstract

We present a new FFT architecture for multi-input multi-output (MIMO) OFDMA wireless systems that require processing variable symbol lengths, ranging from 128 to 2048 complex points. The organization is based on 16 concurrent butterfly processing elements with each element computing a 128-point FFT by implementing an in-place technique. A novel processor-memory interconnection scheme allows the processing elements to operate in sets of k, $1 \leq k \leq 16$, for completing FFT computations of size $128 \times k$, up to 2048 points. The architecture scales to support 4×4 MIMO-OFDMA operation. An FPGA implementation shows that the proposed organization requires 9995 slices on Xilinx Virtex-4 compared to 21624 slices of four parallel FFT architectures accomplishing the same task.

1 Introduction

The performance of the Fast Fourier Transform (FFT) and the Inverse FFT (IFFT) algorithms plays a crucial role in emerging wireless technology standards that are based on Orthogonal Frequency Division Multiple Access (OFDMA). The two key standards that use OFDMA as a core function in their baseband processing are IEEE 802.16 (WiMAX), and the 3G Long Term Evolution (LTE) standard which has emerged as a comprehensive evolution of the Universal Mobile Telecommunications System (UMTS). These standards support high-data rates that require executing FFT/IFFT at relatively high speeds, while cost constraints imply the use of minimal resources. The problem of achieving high throughput rates leads to a pipelined execution by using a processor for each FFT stage. Single datapath Delay Feedback (SDF) architectures [3] and variations proposed for OFDM systems [6], [2], [1] have been considered as the most appropriate solution because, apart their pipelined structure, they need minimal memory volume for data storing. Researchers also focused in systems supporting multiple antennas [4], [5], [7], [9], [10]. In such systems though, and especially as in the baseline MIMO-OFDMA, adopting the SDF solution leads to a large number of multiplication processing units.

Aiming at providing a solution to computing FFT in MIMO-OFDMA systems and save on resources, this paper presents a flexible Macro-Pipelined FFT architecture for concurrent multi-symbol processing. The proposed organization supports the FFT computations of up to four different symbols. Each symbol can be of variable length with minimum length 128 and maximum 2048. The architecture's processing core consists of a parallel structure involving 16 butterfly processors with each processor being able to compute a 128 complex point FFT algorithm. An interconnection network allows each butterfly processor to load (store) data from (to) the memory banks of other four processors. Our approach is based on optimizing interprocessor communications to realize FFT computations of input size ranging from 256 to 2048, with reduced hardware resources. To validate the design we have developed a complete implementation of our design on a Xilinx Virtex-4 FPGA and compared it to an architecture with four distinct SDF FFT units such that both systems achieve the same throughput.

The paper is organized as follows. The second section provides an overview of the proposed architecture and the data flow. Section 3 explains the 128-point butterfly processor. Section 4 presents the new interconnection scheme. Finally, section 5 concludes the paper.

2 The FFT Organization

The basis of our design is a parallel and fully pipelined architecture capable of processing multiple variable-length OFDMA symbols concurrently. In a 4x4 MIMO system, upto 4 different streams can be transmitted concurrently,

over multiple antennas, using spatial multiplexing techniques. Therefore, our organization supports the FFT computations of four distinct symbols with length ranging from 128 to 2048 data. There are 16 butterfly processors with 32 banks and an interconnection network that is used to group the processors in sets of $2,\ 4, \ldots, 16$. The purpose is to let each processor execute all calculations regarding a 128 point FFT with all the relevant input data stored in its own memory banks. This solves the problem of computing the bins for symbols with length 128. We use the interconnection so that a set of k ($k = 2, \ldots, 16$) processors computes the $128 \times k$ point FFT and we let the organization to operate at a frequency 33.33 MHz, which is 37.5% higher than the input's 24.24 MHz, so that the 16 processors complete the FFT calculations on 2048 points in 512 input clock cycles. Hence, we process in parallel symbols whose lengths add up to 2048 else we pipeline the processing of these. In the worst case the four symbols will have length 2048 each and all four FFTs will be completed in 2048 cycles. A second memory with also 32 banks is used to input and output the data. The organization performs Decimation in Time (DIT) and the input is written in the memories by following bit reversed order. The overview of the organization is shown in Fig. 1

To highlight the functionality, each processor P_i, $0 \leq i \leq 15$, performs radix-2 butterfly computations and has a memory of size 128 words divided into 2 banks ($B_{i,0},\ B_{i,1}$). The architecture is designed to let each processor access two data words in parallel by realizing an *in-place* technique similar to [8] and complete each stage in 64 cycles. The interconnection network allows each P_i to access data of other four processors P_x, where x is the bitwise XOR of i ($= i_3 i_2 i_1 i_0$) with $0001,\ 001\overline{i_1},\ 01\overline{i_2 i_2},\ 1\overline{i_3 i_3 i_3}$. P_i is able to use the banks $B_{x,1}$. For example, consider the 256 point FFT executed in $P_{2k},\ P_{2k+1},\ 0 \leq k \leq 7$. The outcome of the 7th stage is 128 data with lower indices (the elements with initial addresses $d_0, \ldots, d_{127}$) stored in the two memory banks of P_{2k} with each bank storing a block of 64 elements. Further, the 128 data with higher indices ($d_{128}, \ldots, d_{255}$) are stored in the banks of P_{2k+1} divided also into blocks of 64. The organization will have P_{2k} storing $d_0, \ldots, d_{63}$ in $B_{2k,0}$ and $d_{64}, \ldots, d_{127}$ in $B_{2k,1}$, while P_{2k+1} will store $d_{128}, \ldots, d_{191}$ in $B_{2k+1,1}$ and $d_{192}, \ldots, d_{255}$ in $B_{2k+1,0}$. This is accomplished by following permutations on the FFT data during the first 7 stages. The interconnection allows P_{2k} to complete the upper half of the 8th stage by accessing the elements in $B_{2k,0}$ and $B_{2k+1,1}$, while P_{2k+1} completes the lower by accessing the elements in $B_{2k+1,0}$ and $B_{2k,1}$. The remainder of this section shows the data flow, while the description of the butterfly processor and the interconnection are given in the following sections.

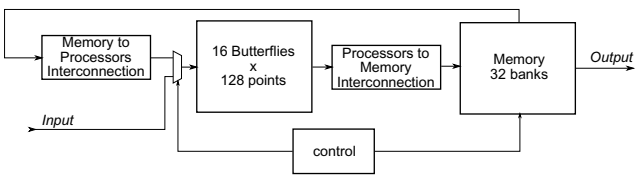

Figure 1. The overall FFT organization

The data flow is similar to that in [8]. The in-place technique of [8] is modified to produce a sorted FFT output by using as a key the indices of the elements (the initial address of the elements). Initially all input elements are stored in banks $B_{i,0},\ B_{i,1}$ so that the LSB of each element's index specifies its storing bank. In the presented organization, the completion of each butterfly computation is followed by a permutation performed at the butterfly's output. To describe this permutation we consider the elements $x_s,\ x_r$ forming a transformation couple at stage (pass) j. The indices $x_s,\ x_r$ differ only at the jth bit and the results will be exchanging memory locations if the bit $(j+1)$ of the indices $x_s,\ x_r$ is 1. Also, an input permutation is performed: we will exchange $x_s,\ x_r$ at the butterfly input if x_s is stored in $B_{i,1}$. Proof of the technique can be found in [8] and the data flow is depicted in Fig. 2. Fig. 2 shows the input and output permutations: the dotted lines denote that a transformation pair $x_s,\ x_r$ is loaded and stored without exchanging positions. Heavy lines denote that $x_s,\ x_r$ are exchanging position. The figure also shows that the output elements are sorted with indices $0, \ldots, N/2-1$ stored in banks $B_{i,0}$ in increasing order and indices $N/2, \ldots, N-1$ stored in $B_{i,1}$ in decreasing order.

The proposed FFT organization requires reduced resources compared to four independent FFT architectures. This is because of two reasons. First, the design includes only 16 butterfly processors. Second, the data permutations applied at each stage allow the reduction of the interconnection since each processor P_i needs to access only four other banks $B_{x,1}$ besides its own $B_{i,0},\ B_{i,1}$.

3 Butterfly Processor Architecture

The radix-2 butterfly processor P_i has two inputs (I_R, I_S) and two outputs (O_R, O_S), the two dual-port memory banks $B_{i,0}$, $B_{i,1}$, the FFT control, the interconnection between the processor and the banks, the data address generation circuit and the twiddle address generation. Fig. 3 depicts the processor architecture. The FFT control includes a 10 bits *up* counter and a *down* counter with 4 bits for the stages (passes). The interconnection involves two multiplexers connecting the banks to each processor input and two multiplexers connecting the processor's outputs to each bank.

The radix-2 processor is organized to compute the FFT

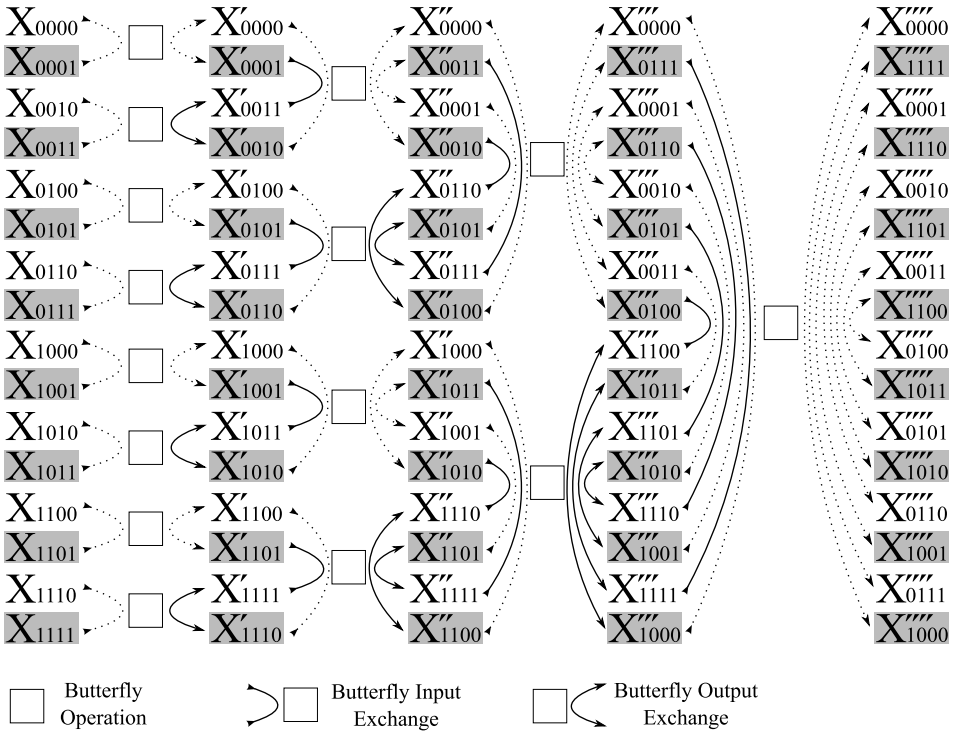

Figure 2. Data Flow of The FFT on 16 points. Unshaded elements are in $B_{i,0}$, shaded in $B_{i,1}$

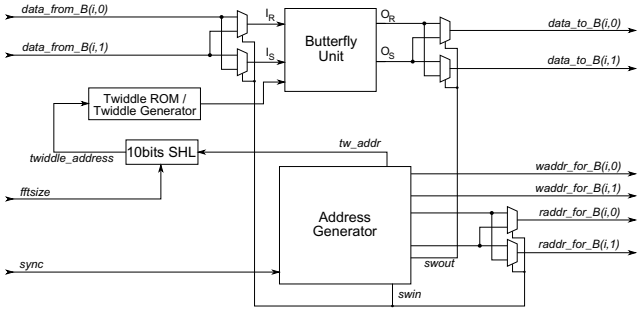

Figure 3. The radix-2 Processor

of 128 points by performing DIT and produce the 128 results sorted: after the FFT completion the elements with indices $d_0, \ldots, d_{63}$ will be in the addresses $0, \ldots, 63$ of $B_{i,0}$ and the elements $d_{64}, \ldots, d_{127}$ will be in the addresses $63, \ldots, 0$ of $B_{i,1}$ respectively.

The processor uses the up counter to handle 1024 pairs of data. The first 64 pairs are in $B_{i,0}$, $B_{i,1}$. In this section we describe the computations on the first 64 pairs stored in $B_{i,0}$, $B_{i,1}$. In the cases of processing more than 64 pairs (2^j pairs, $7 \leq j \leq 10$) the processor performs the same operations on data stored in $B_{i,0}$, $B_{x,1}$ with x defined by the interconnection. The address generation circuit shown in figure 4 uses the two control counters to generate the data addresses at each stage and controls the 4 multiplexers. During the jth stage the circuit will address $N/2$ pairs belonging to $N/2^{j+1}$ FFT sub-blocks. The circuit generates the addresses of the pairs by forming a word consisting of the 6 least significant bits of the up counter. The addresses for $B_{i,0}$ are generated by replacing bit $j-1$ of the word format with a 0. The addresses for $B_{i,1}$ are generated by inverting the $j-1$ least significant bits of the $B_{i,0}$ address.

The multiplexers at the butterfly outputs (O_R, O_S) realize the permutation and at each pass j, $0 \leq j \leq 6$, are controlled by the jth bit of the up counter: if the jth bit is 1 then the multiplexers exchange the outputs of the butterfly (*swapout* signal of Fig. 4). The multiplexers at the I_R, I_S inputs of the butterfly are controlled by the $(j-1)$th bit of the up counter (*swapin* signal of Fig. 4). The multiplexers at the output of the address generator (read addresses) are also controlled by the *swapin* signal.

The 1024 twiddles are stored in a ROM and we use the up counter to generate their address. Before executing the last stage of a sub-FFT of size 2^{j+1} on the two sub-FFTs of size 2^j, the two sub-FFTs have their results sorted (Fig. 2) according to their indices. The results in the upper sub-FFT are sorted such that the 2^{j-1} lower indices are in increasing order in $B_{i,0}$ and the 2^{j-1} higher indices are in decreasing order in $B_{i,1}$. The lower sub-FFT are sorted such that the 2^{j-1} lower indices are in decreasing order in $B_{i,1}$ and the 2^{j-1} higher indices are in increasing order in $B_{i,0}$. There-

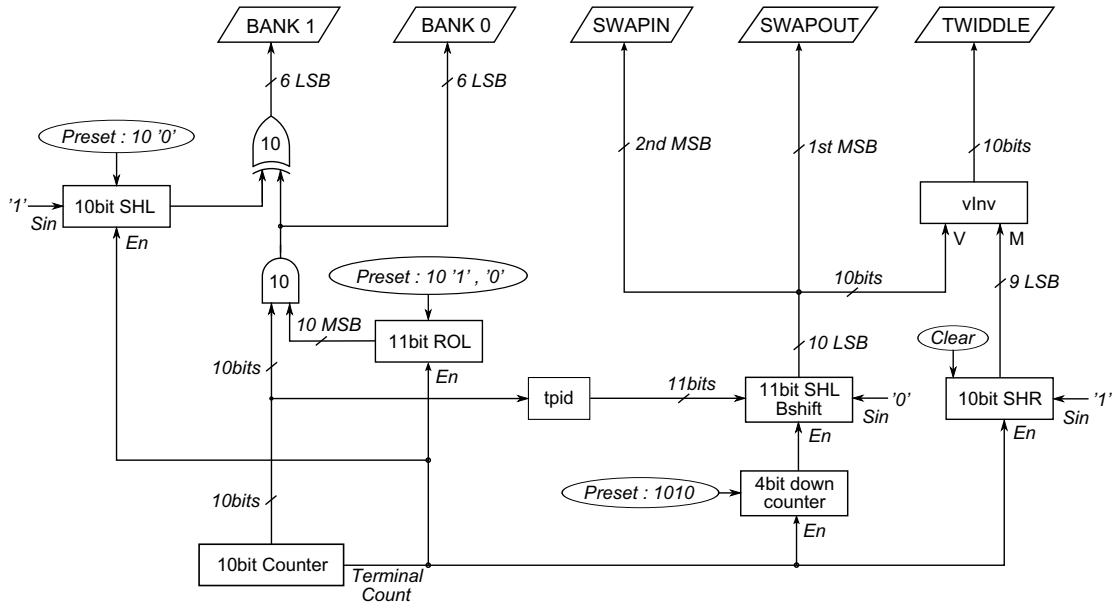

Figure 4. The Processor's Address Generator

fore, we read the twiddles of the first 2^{j-1} pairs by increasing a counter and the twiddles of the remaining 2^{j-1} pairs by decreasing a counter. This operation is simplified and accomplished as follows: At pass j we use the $j+1$ least significant bits to create a 10 bit word: these $j+1$ bits are used as MSBs followed by 0's (input V of the *vInv* of Fig. 4). In case of the MSB of the above word equal to 0 then this word will be used as a twiddle address. In case of the MSB of the word equal to 1 then the remaining j MSBs (all the above $j+1$ MSBs apart the MSB) are inverted to create the twiddle address.

4 Interconnection

The 2048-point transform is the largest size supported by the proposed organization. As mentioned above, the first 7 stages are completed by using only local memory addresses at processor P_i. In the worst case there will be four more stages (stages 7 to 10) to be computed and processor P_i will have to access the memories of at most 4 other processors. To simplify the analysis and the implementation, we fix P_i's upper input/output to always connect to $B_{i,0}$. Hence, the problem is reduced to assign to each P_i's lower input/output a set of banks, drawn from the superset of $B_{x,1}$ for all x. Furthermore, the set always contains the bank $B_{i,1}$ –which is used during the first 7 stages– and also contains four other banks, which are utilized by (at most) four remaining stages.

The set of banks is determined by the flowgraph of the FFT algorithm as this has been mapped using the data flow of the in-place technique. Assuming a worst case size of $N = 2048$ points, the data are initially divided into groups of 128 and assigned to processors in normal, increasing order (i.e. the first group to P_0, the second to P_1 etc.). To come up with the set of banks we restrict the analysis to stages j after stage 6, and let $j = 6+j', 1 \leq j' \leq 4$. We will use the binary representation of the index of the ith processor P_i as: $i = [i_3 i_2 i_1 i_0]$. It is straightforward to show that by following the data flow of section 2, each processor P_i at the jth stage will access data belonging to processor P_k: the index k (in binary $k = [k_3 k_2 k_1 k_0]$) is obtained by using i in the following two steps. First, we consider the effect of the output permutation which is a bitwise exclusive-or (XOR) operation on the index i with a 4 bit number containing j' ones in the j' LSBs and zeros otherwise. Second, data exchanges at the input occur at processors whose index has the $(j'-1)$th bit set. For these processors, the index k is computed by the first step calculation and corrected by performing another bitwise exclusive-or operation with a number containing $j'-1$ ones in the $j'-1$ LSBs (and zeros otherwise). Therefore, k is produced by super-imposing the two permutations (input and output) during stages $j \geq 8$ (stage 7 does not require correction). More specifically:

$$\begin{aligned} k &= [k_3 k_2 k_1 k_0] \\ &= [i_3 i_2 i_1 i_0] \oplus [0 \ldots \underbrace{1 \ldots 1}_{j'}] \oplus [0 \ldots \underbrace{i_{j'-1} \ldots i_{j'-1}}_{j'-1}] \\ &= [i_3 i_2 i_1 i_0] \oplus [\, 0 \ldots 1 \underbrace{\overline{i_{j'-1}} \ldots \overline{i_{j'-1}}}_{j'-1} \,] \end{aligned}$$

Therefore, the interconnection network for each processor consists of a 5-to-1 multiplexer at the processor's lower

input I_s and a 1-to-5 demultiplexer at its lower output O_s. The connections to each (de)multiplexer can be computed from the equation above. Note that the depth of the address calculation circuit by using the proposed technique is constant, irrespective of the size N of the FFT transform.

The advantage of using the proposed design is shown by a FPGA implementation. The FFT architecture has been compared to a solution including four SDF architectures (R-2) [3], which also support variable symbol length (128 to 2048 point FFT). The FFT organization requires 9995 slices on Xilinx Virtex-4 SX35 operating at 33.33 MHz, while the solution with four distinct SDF FFT architectures and achieving the same results utilizes 21624 slices (each SDF occupies 5406 slices) operating at 24.24 MHz.

Moreover, we have compared the proposed architecture with improved FFT SDF architectures, namely the R-2^2 [1] and the R-2^3 [2]. The results are shown in table 1. compares the features of each architecture implemented on the Xilinx Virtex-4 SX35. These results show that for the execution of 4 independent FFTs of variable length, the proposed architecture is superior compared to the organizations, which include 4 inpependent SDF architectures.

Table 1. FPGA Implementation Comparison

Architecture	Xilinx Slices	Operating Frequency	DSP blocks	RAM blocks
Proposed	9995	33 MHz	128	240
R-2	21624	24 MHz	360	147
R-2^2	18300	24 MHz	192	168
R-2^3	18060	24 MHz	176	168

5 Conclusion

This paper has presented a new FFT architecture for MIMO-OFDMA systems in which multiple variable length symbols from different space-multiplexed streams need be processed concurrently. The work builds on a butterfly processor realizing an in-place technique and able to perform a 128 point FFT. An interconnection network allows 16 butterfly processors to communicate data and process four symbols with length ranging from 128 to 2048 words. FPGA implementation showed that the proposed FFT organization performs at a frequency 37.5% higher than a four SDF FFT solution, while the structure with the four SDF FFT paths requires an extra 116% of hardware resources.

References

[1] S. He and M. Torkelson *"A New Approach to Pipeline FFT Processor,"* Proceedings of the IPPS, 1996.

[2] S. He and M. Torkelson *"Designing Pipeline FFT Processor for OFDM (de)Modulation,"* in the Proc of URSI Intl Symposium on Signals, Systems and Electronics 1998.

[3] Clark D. Thompson *"Fourier Transform in VLSI,"* IEEE Transactions on Computers, 1983.

[4] P. Coulton and D. Carline, *"An SDR inspired design for the FPGA implementation of 802.11a baseband system,"* in proc. of IEEE International Symposium on Consumer Electronics, pp. 470-475, Sep 2004.

[5] K. Masselos, A. Pelkonen, M. Cupak, and S. Blionas, *"Realization of wireless multimedia communication systems on reconfigurable platforms,"* Journal of Systems Architecture, vol. 49, no. 4-6, pp. 155-175, 2003.

[6] R. Storn, *"Radix-2 FFT- Pipeline Architecture with Reduced Noise to Signal Ratio,"* IEE Proceedings-Vision, Image, and Signal Processing, vol. 141, no. 2, pp. 81-86, Apr 1994.

[7] UCLA MIMO wireless communication research group, http://www.ee.ucla.edu/∼mimo.

[8] K. Nakos, D. Reisis, N.Vlassopoulos, *"Addressing Technique for Parallel Memory Accessing in Radix-2 FFT Processors"* ICECS, pp.52-56, September 2008.

[9] Jeoong S. Park, Hong-Jip Jung and Viktor K. Prasanna, *"Efficient FPGA-based Implementations of the MIMO-OFDM Physical Layer,"* World Congress in Computer Science, Computer Engineering and Applied Computing, Las Vegas, Nevada USA 2006 (WORLDCOMP'06) Proceedings of ERSA'06, pp. 153-163

[10] Rice University TAPs Research Group, http://taps.rice.edu/.