A SPLIT MEMORY SCHEME FOR EFFICIENT IMPLEMENTATION OF FFT ALGORITHM

Christos Meletis Giannis Sifnaios Paul Bougas Kostas Marinis Kostas Asfis Kiamal Pekmestzi
e-mail: {chris, john, paul, kmarinis, kasfis, pekmes} @microlab.ntua.gr
Microprocessors and Digital Systems Laboratory, Department of Electrical and Computer Engineering,
National Technical University of Athens, 15773 Zografou, Athens, Greece
Phone +30 1 7722500

Key words: Digital signal processing, Fast Fourier Transform (FFT), Field Programmable Gate Arrays (FPGAs)

ABSTRACT

In this paper, we propose a new architecture for the implementation of the N-point Fast Fourier Transform (FFT), based on the Radix-2 Decimation in frequency FFT algorithm. This architecture is based on a split memory scheme and can compute the FFT in N/2·(1+log$_2$N) clock cycles.

I. INTRODUCTION

Modern telecommunication systems are based more than ever before on digital signal processing for the implementation of complicated protocols. High-speed telecommunication systems such as OFDM (orthogonal frequency division multiplexing) and DSL (digital subscriber lines) need real-time computation of the N-point DFT transform for large number of points (512 or more).

There are many architectures proposed for the implementation of the FFT algorithm that can compute one N-point FFT transform in N clock cycles [2]-[3]-[5]-[6]. However, these architectures demand a very large circuit, since $\log_2 N$ complex multipliers are needed [5]-[6]. This circuit is very complicated for on chip implementation of telecommunication systems. For these applications, architectures with only one butterfly are more suitable. These architectures require two memories (each with size of N complex words), one for the input and one for the output data of the butterfly. These schemes compute the FFT in $N\cdot\log_2 N$ clock cycles. In this paper, we propose a new architecture with the same memory size, which can compute the FFT in $N/2\cdot\log_2 N$ clock cycles. This is achieved by splitting each memory in two separate blocks with independent memory interfaces.

II. FFT ALGORITHM

The N-point Discrete Fourier Transform (DFT) is defined as

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

where

$$W_N = e^{-j\frac{2\pi}{N}} \tag{2}$$

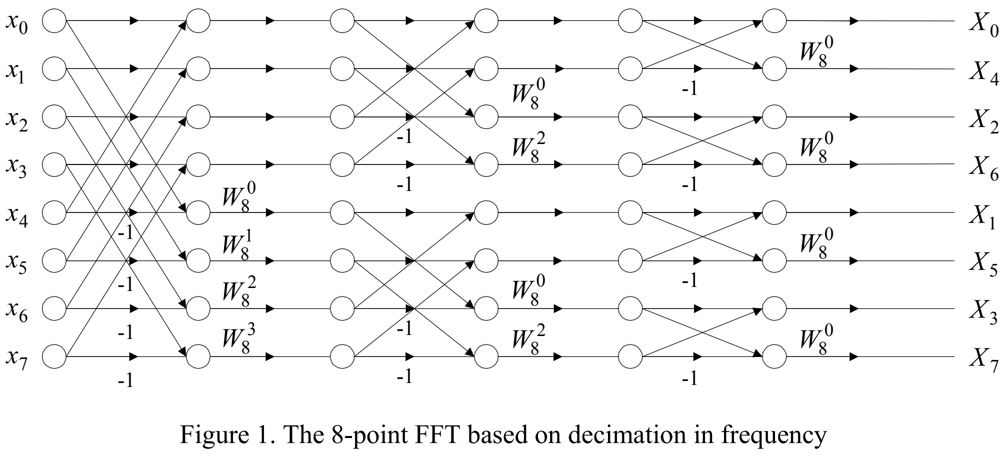

Figure 1. The 8-point FFT based on decimation in frequency

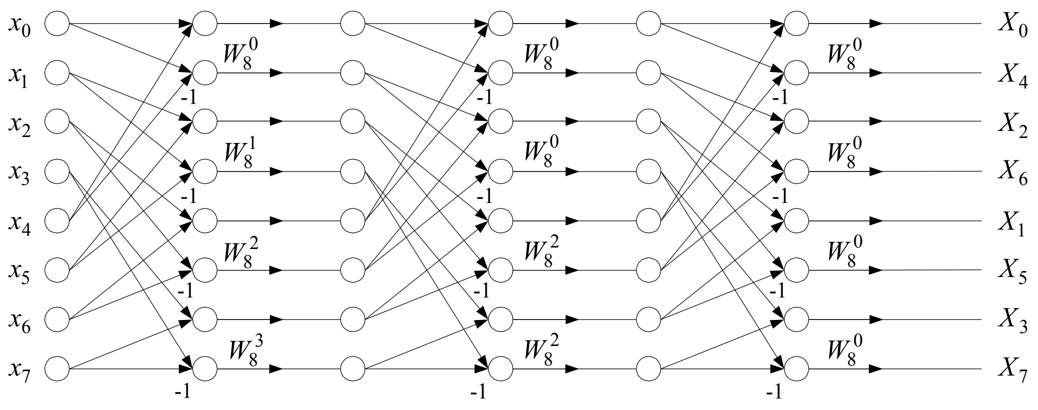

Figure 2. The constant geometry 8-point FFT

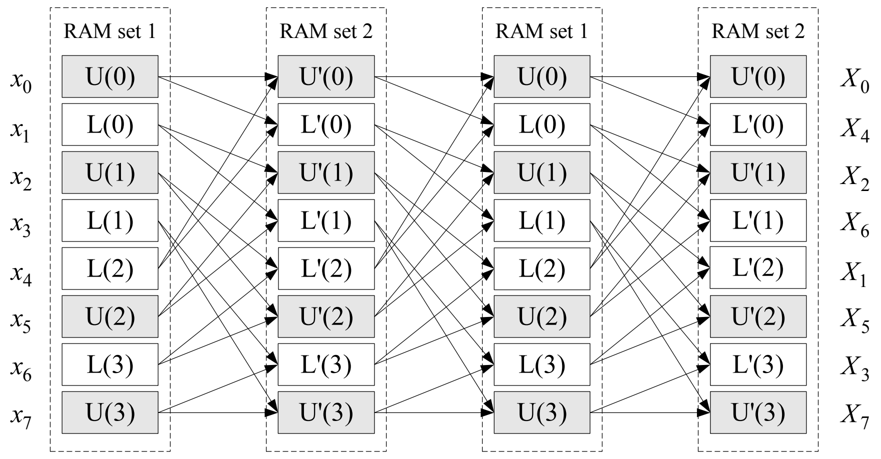

Figure 3. The organization of the data in upper (U) and lower (L) RAM

FFT algorithms permit an efficient implementation of the DFT. Such an algorithm is the Cooley-Tukey Radix-2 Decimation in frequency FFT. This algorithm divides the output sequence into even and odd-numbered samples [1].

$$X_{2k} = \sum_{n=0}^{\frac{N}{2}-1} \left(x_n + x_{n+\frac{N}{2}} \right) W_{N/2}^{nk} \tag{3}$$

$$X_{2k+1} = \sum_{n=0}^{\frac{N}{2}-1} \left(x_n + x_{n+\frac{N}{2}} \right) W_N^n W_{N/2}^{nk} \tag{4}$$

where k=0, 1, …, N/2-1

If we continue the decimation for every one of the above, then this algorithm can be represented with a flow graph like the one in Figure 1, where the 8-point FFT is shown. The output of this algorithm is in bit-reverse order.
All the calculations are performed by 2-input butterfly units like the one shown in Figure 4 [1].

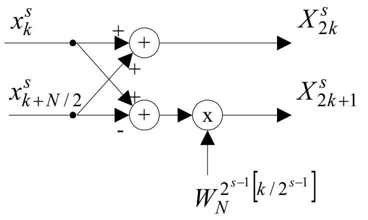

Figure 4. 2-input butterfly unit

If index s indicates the stage of butterflies (s=1, 2, …, log$_2N$) then the output of the butterfly units is given by (5) where k=0, 1,… , N/2-1 and [x] is the integer part of x.

$$\begin{aligned} X_{2k}^s &= x_k^s + x_{k+N/2}^s \\ X_{2k+1}^s &= \left(x_k^s - x_{k+N/2}^s \right) W_N^{2^{s-1}\left[k/2^{s-1}\right]} \end{aligned} \tag{5}$$

For an N-point FFT we need n=log$_2N$ stages of butterflies. Each butterfly unit consists of two complex adders, that is four adders, and one complex multiplier, which, actually, consists of four multipliers and two adders. Therefore, for the butterfly unit we need a total of four multipliers and six adders.

By rearranging the nodes of Figure 1 we come up with the geometry of Figure 2. In this figure, we have the same geometry for each stage. This constant geometry is suitable for implementation, since the same approach can be used for the operations of all stages. Therefore, the *N*-point FFT algorithm can be implemented using two sets of RAM and a butterfly unit like the one shown in Figure 4, which uses the RAMs for reading the inputs and storing the outputs as shown in Figure 3, where the 8-point FFT algorithm is presented.

III. THE ARCHITECTURE

The implementation proposed can execute one butterfly operation every clock cycle. For this to be possible, the circuit must have the ability to read both inputs and store both outputs of the butterfly in one cycle. This is achieved by partitioning each RAM set into two RAMs: RAM U (upper) and L (lower) for RAM set 1 and U' and L' for RAM set 2, as shown in Figure 3. By storing the data in the appropriate memory addresses, shown in the parenthesis, we ensure that both the inputs and the outputs are stored in different RAMs and, therefore, can be accesses in the same clock cycle. Therefore, the implementation of each stage is achieved in $N/2$ and the full FFT in $N/2 \cdot \log_2 N$ clock cycles.

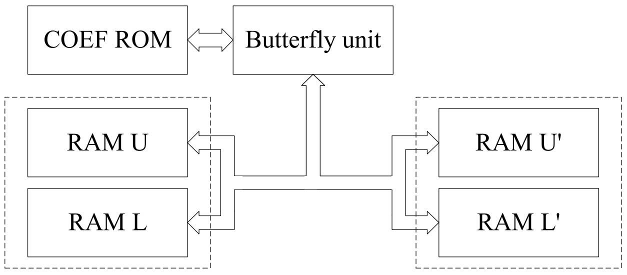

Figure 5. Block diagram of the architecture proposed

The architecture proposed for this algorithm is shown in Figure 5. The circuit, besides the two sets of RAM (RAM U-L and RAM U'-L'), consists of a ROM and a butterfly unit that performs the calculations. Each one of the four RAMs can store $N/2$ complex numbers, so we need a total of $2N$ words storage capacity. The ROM contains the $N/2$ complex FFT transform coefficients, where transform coefficient W_N^k is stored at address k (k=0, 1, …, N/2-1). The input data are initially stored in RAM U and L. The butterfly unit performs all the operations of the first stage by reading in every clock cycle two complex numbers from RAM set U-L and one transform coefficient from the ROM and storing the two outputs to RAM set U'-L'. When after $N/2$ clock cycles all the operations of the first stage have been performed, then the butterfly unit proceeds to the next stage where the inputs are read from RAM set U'-L' and the outputs are stored to the RAM set U-L. This procedure is repeated $\log_2 N$ times in order to implement the FFT algorithm. Next, the output of the results of the FFT from the one RAM set requires N/2 clock cycles while at the same time the new input data are stored in the other RAM set. Consequently the total FFT computation time is $N/2 \cdot (1+\log_2 N)$.

This architecture allows the spiting of each RAM set in such a way that the input and the output of the butterflies are always from different RAMs. The organization of the data in the four RAMs for the 8-point FFT transform is shown in Figure 3. This organization allows the simultaneous reading of the two inputs (one from U and one from L) and storing of the two outputs (one to U' and one to L' or vice versa) of the butterfly.

The simplified circuit of the architecture proposed is shown in Figure 6. A control unit produces the source addresses SA1 and SA2 for reading the inputs of the butterfly unit, the destination address DA for storing the outputs of the butterfly unit, the coefficient address CA for reading the transform coefficients from the ROM and the control signals C_1 and C_2 for the routing of the data between the butterfly unit and the RAMs. Most of the signals produced by the control unit are the same for all stages of butterflies, because the implementation is based on the constant geometry FFT algorithm. The control unit, swaps the functionality of the RAM sets U-L and U'-L', using the appropriate signals, in order to proceed from one stage to the next. In addition, the control unit produces the coefficient addresses CA for each stage of butterfly operations. For the 8-point FFT transform, the signals produced by the control unit are shown in Table I. All of these signals can be produced using a counter that counts the butterfly operations of each stage. For the N-point FFT, if $k=k_{n-2}k_{n-3}\ldots k_1k_0$ is the counter value, where $n=\log_2 N$, then the signals mentioned above are given in (6).

$$\begin{gathered} C_1=k_0 \\ C_2=k_{n-2} \\ SA1=k_0k_{n-2}k_{n-3}\ldots a_1 \\ SA2=\bar{k}_0k_{n-2}k_{n-3}\ldots a_1 \\ DA=k_{n-2}k_{n-3}\ldots k_0\ (=\text{counter}) \end{gathered} \tag{6}$$

The same counter can produce the coefficient address CA that depends on the stage of the butterflies. For the N-point FFT transform the coefficient address is given by (7).

$$CA = k_{n-2}k_{n-3}\ldots k_{s-1}\ \underbrace{00\ldots0}_{s-1\text{ zeros}} \tag{7}$$

where s is the current stage (s=1, 2, …, n).

Table II shows the coefficient addresses for each stage for the 1024-point FFT.

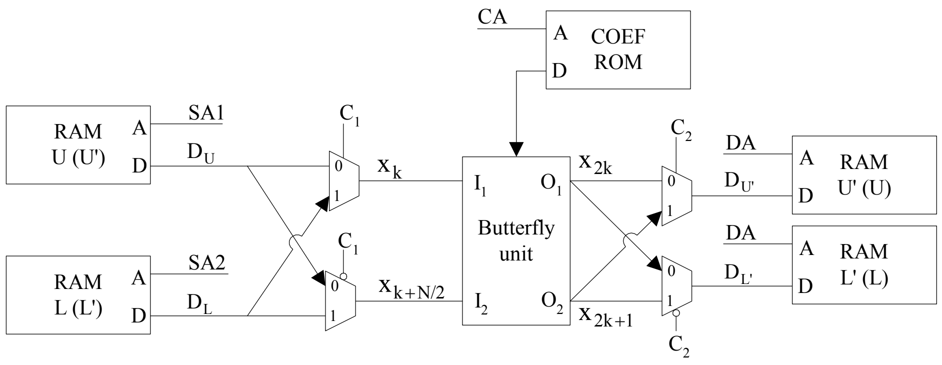

Figure 6. The simplified circuit of the architecture proposed

Table I
Signals produced by the control unit and data routing for the 8-point FFT.

Counter (k)	C_1	C_2	SA1	SA2	DA	Butterfly $I_1=x_k$	$I_2=x_{k+N/2}$	$D_{U'}$	$D_{L'}$
00 (0)	0	0	00 (0)	10 (2)	00 (0)	x0	x4	X0	X1
01 (1)	1	0	10 (2)	00 (0)	01 (1)	x1	x5	X2	X3
10 (2)	0	1	01 (1)	11 (3)	10 (2)	x2	x6	X5	X4
11 (3)	1	1	11 (3)	01 (1)	11 (3)	x3	x7	X7	X6

Table II
Coefficient addresses for the 1024-point FFT

Stage	Coefficient ROM Address (CA)
1	$k_8k_7k_6k_5k_4k_3k_2k_1k_0$
2	$k_8k_7k_6k_5k_4k_3k_2k_1$ 0
3	$k_8k_7k_6k_5k_4k_3k_2$ 0 0
4	$k_8k_7k_6k_5k_4k_3$ 0 0 0
5	$k_8k_7k_6k_5k_4$ 0 0 0 0
6	$k_8k_7k_6k_5$ 0 0 0 0 0
7	$k_8k_7k_6$ 0 0 0 0 0 0
8	k_8k_7 0 0 0 0 0 0 0
9	k_8 0 0 0 0 0 0 0 0
10	0 0 0 0 0 0 0 0 0

IV. IMPLEMENTATION

The most time critical part of the design was the butterfly, hence we concentrated our efforts in implementing it in an optimal way. More specifically, we tried both combinational and pipelined versions for the multipliers used in the butterfly. In the combinational version, a real multiplication is computed in a single cycle. The computation latency is minimized, at the expense of a longer critical path, resulting in a lower operating frequency. In the pipelined version, the multiplier has four levels of internal pipelining, and registers at the output stage, resulting in a latency of 5 clock cycles. The partitioning of the critical path of the multiplier by the incorporation of the pipeline registers yielded a smaller clock cycle, hence a higher operating frequency was achieved.

The implementations of the proposed architecture were targeted at the Xilinx Virtex XCV1000E-6. The Virtex-E FPGAs comprise of two major configurable elements: Configurable Logic Blocks (CLBs) and Input/Output blocks (IOBs). The CLBs are divided into two slices each and provide the functional elements for constructing logic. The IOBs provide the interface between the package pins and the CLBs. The CLBs interconnect through a General Routing Matrix (GRM), an array of routing switches located at the intersections of horizontal and vertical routing channels. The main reasons for the selection of Virtex-E series as target devices are that they incorporate fast and efficient routing resources (GRM), and a large number of internal memory blocks, named Block SelectRAM+. The Block SelectRAM+ are embedded memory blocks, organized in columns starting at the left and right outside edges of the device and inserted every 12 CLB columns [7]. This organization allowed a more efficient placement and routing of the design.

The designs were implemented in RTL VHDL. The Exemplar Leonardo Spectrum 2000.1a and Xilinx Foundation 3.1i tools were used for synthesis and implementation, respectively. All implementation runs were targeted at maximum speed with normal optimization effort, and constraints applied at the critical paths to reduce routing delays. We let the implementation tools place and route the designs automatically, without any manual floorplanning or routing. It is expected that even higher performance can be achieved by manual floorplanning and routing. All designs were fully tested, simulated and verified using the Model Technology's ModelSim v.5.2e simulator.

Table III
FFT Implementation results for the XCV1000E-6

N	Multiplier Type	Slices	LUTs	FFs	Block RAMs	f_{max} (MHz)	Throughput (TP) (Msamples/s)	TP/Slices (Ksamples/s/ Slices)
256	Pipelined	1891	3263	2642	18	68.790	14.9	7.9
	Comb.	1680	3198	880	18	50.098	11.2	6.6
512	Pipelined	1905	3274	2656	28	66.872	13.2	6.9
	Comb.	1687	3203	894	28	48.578	9.8	5.8
1024	Pipelined	1920	3298	2670	56	64.416	11.7	6.0
	Comb.	1692	3217	908	56	44.514	8.1	4.8

The multipliers used were fully customizable, relationally placed IP cores provided by Xilinx. The relational placement option allows the multiplier cores to be used as pre-placed hard macros, easing the placement and routing effort, and providing more predictable results between the different implementation runs. The datapaths were 48-bits wide for data (24-bits real and imaginary parts), and 32-bits for the coefficients, in order to maintain a reasonable amount of precision.

The results from the various implementations are illustrated in Table III. As was expected, the pipelined implementations yield higher operating frequencies, at the expense of hardware area. In the same table, a hardware performance metric is also given, defined as the ratio of the throughput vs. hardware complexity. This metric can be a useful means of comparing the various implementations. As can be seen from Table III, the pipelined implementations yield higher hardware performance compared to the combinational ones.

V. CONCLUSIONS

The architecture proposed computes the *N*-point FFT algorithm, where *N* is a power of two, in $N/2 \cdot (1+\log_2 N)$. The circuit consists of one complex multiplier, two complex adders, one ROM for the *N*/2 complex transform coefficients and a RAM with capacity of 2N complex words partitioned into four independent memories of N/2 complex words.

Using the proposed architecture, we achieved the doubling of the processing speed of the FFT by partitioning the RAM sets for the inputs and the outputs of the butterfly into upper and lower part, so that we can read and store the two inputs and the two outputs of the butterfly in one clock cycle. As a result, one full butterfly computation is performed in every clock cycle. The complexity of the control unit has not been increased, since all the control signals can be directly (without extra logic) produced by a simple counter. This is possible due to the partitioning method that has been applied on the constant geometry FFT algorithm.

Acknowledgement:
The work presented in this paper was partially funded by the Greek General Secretariat for Research and Technology (GSRT) project 99ED481.

REFERENCES

1. A.V. Oppenheim and R.W. Schafer, "Digital Signal Processing", Prentice Hall, 1975
2. L.-W Chang and M.-Y. Wu, "A new systolic array for discrete Fourier transform", IEEE Trans. Acoust., Speech, Signal Processing, vol.36, pp.1165-1167, Oct. 1988.
3. N. R. Murphy and M. N. S. Swamy, “On the real-time computation of DFT and DCT through systolic architecture,” IEEE Trans. Signal Processing, vol. 42, pp. 988–991, Apr. 1993.
4. J. Choi and V. Boriakoff, “A new linear systolic array for FFT computation,” IEEE Trans. Circuits Syst. II, vol. 39, pp. 236–239, Apr. 1992.
5. V. Boriakoff, “FFT computation with systolic arrays, a new architecture,” IEEE Trans. Circuits Syst. II, vol. 41, pp. 278–284, Apr. 1994.
6. C.-H. Chang, C.-L. Wang, Y.-T. Chang, "Efficient VLSI Architectures for Fast Computation of the Discrete Fourier Transform and Its Inverse", IEEE trans. on signal processing, vol. 48, pp. 3206-3216, Nov. 2000.
7. Xilinx Inc.: “Virtex-E 1.8V Field Programmable Gate Arrays”, http://www.xilinx.com/partinfo/ds022-2.pdf