Parallel FFT Implementation on FPGA

Plamenka I. Borovska and Petyr G. Manoilov

Abstract - **The paper examines the parallel execution of the Fast Fourier Transformation (FFT) computations in a multiprocessor system with shared data memory. The multiprocessor system is designed and implemented on a FPGA – chip. Some experimental results – the speed-up of parallel FFT execution and the amount of occupied FPGA-resources for different number of processors in the system are exposed.**

Keywords **– Fast Fourier Transformation, Butterfly, Multiprocessor System, Field Programmable Gate Array.**

I. INTRODUCTION. THE FFT COMPUTATION.

The Fast Fourier Transform (FFT) [1] is essentially a computationally efficient algorithm for extracting spectral information from signal waveforms, which may be in real time or recorded form (i.e. a transformation from the time domain to the frequency domain). It is often used to dramatic effect in a growing range of applications including radar and sonar processing, speech recognition and image processing.

According to the FFT transform, the underlying computation for the spectral representation of a signal with N samples(points) can be expressed as:

$$\dot{R}_k = \dot{X}_k + \dot{Y}_k e^{-j\frac{2\pi k}{N}},\ 0 \le k \le \frac{N}{2} - 1 \qquad (1)$$

and

$$\dot{R}_k = \dot{X}_{k-N/2} - \dot{Y}_{k-N/2} e^{-j\frac{2\pi}{N}(k-N/2)},\quad \frac{N}{2} \le k \le N-1. \qquad (2)$$

Equations (1) and (2) represent the basic computational requirements for the FFT algorithm and are commonly known as the "butterfly" macro-operation due to the sample of the flow graph often used to represent the this operation, as illustrated in Figure 1. The arguments and the results in these equations are complex numbers. The circle in the butterfly graph notes a complex adding operation, when the result line is upwards and notes a complex subtracting operation, when the result line is downwards.

Plamenka I. Borovska, PhD, is with the Technical University of Sofia, Faculty of Computer Science and Control, 8 boul. "Kl.Ohridski", Sofia, 1756, Bulgaria, E-mail: pborovska@tu-sofia.bg

Petyr G. Manoilov, PhD, is with the Technical University of Sofia, Faculty of Computer Science and Control, 8 boul. "Kl.Ohridski", Sofia, 1756, Bulgaria, E-mail: p.manoilov@mail.bg.

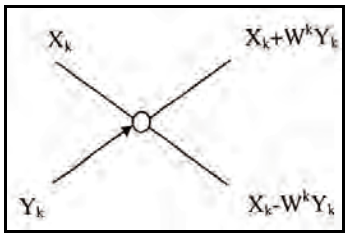

Fig.1. Flow graph, representing a "butterfly" FFT macro-operation

The arrow line notes a complex multiplying operation with multiplier (factor) $W^k = e^{-j\frac{2\pi}{N}k}$ .A single butterfly macro-operation is a FFT for a signal with 2 samples (points). We find, that it requires one complex multiplication, one complex addition and one complex subtraction. Since the multiplications and additions of the FFT algorithm are complex, the algorithm is ideally suited to handle complex input sequences. In transforming real data sequences an additional computational saving can be realized by combining two real N-point sequences into a single N-point complex sequence.

The operations on real and imaginary parts of numbers according to the butterfly of Figure 1 are shown in flow graph of Figure 2. In Figure 2 the symbols R_1, R_2, X_1, X_2, Y_1, Y_2, W_1, W_2 mean the real and imaginary parts of complex numbers $\dot{R}, \dot{X}, \dot{Y}, \dot{W}$. Using the butterflies with the appropriate factors W^k, any N- point FFT can be designed.

Figure 3 shows the flow graph of the 16- point FFT. This computational task requires execution of 32 butterfly macro-operations, arranged in four passes and each butterfly includes the operations, presented in flow graph of Figure 2 – four multiplications, three additions and three subtractions.

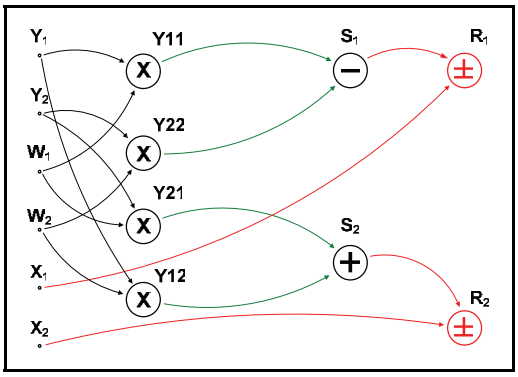

Fig.2. Flow graph, representing the "butterfly" FFT operations on real and imaginary parts of numbers

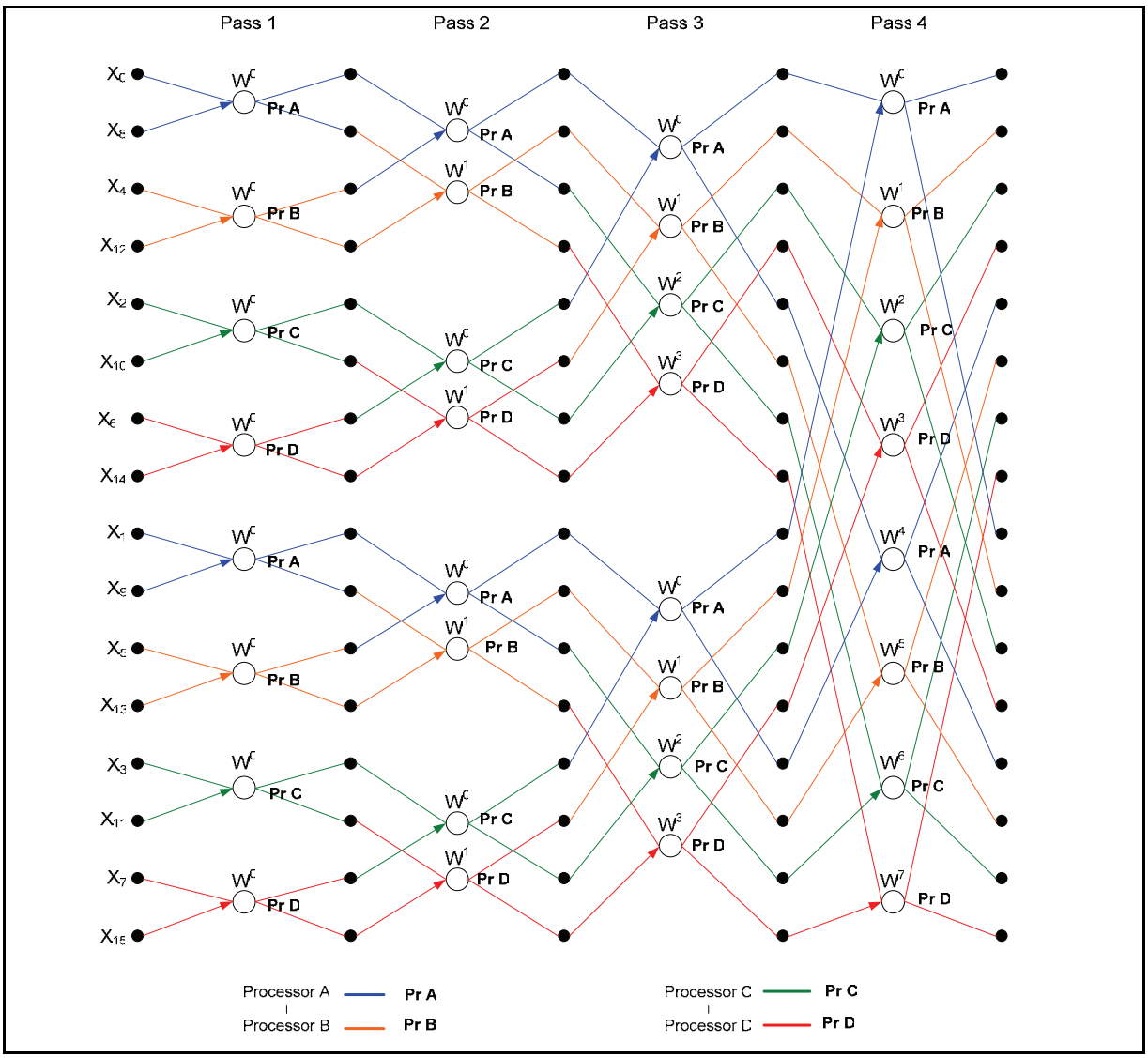

Fig.3. Flow graph of the 16-point FFT

II. Multiprocessor System, Used for Parallel FFT Implementation

The high – level block diagram of a multiprocessor system, designed and implemented on a Field Programmable Gate Array (FPGA) chip is shown on the Figure 4. The system consists of several processor modules (controller KCPSM3 + FFT accelerator) and common shared data memory, used for read and write processor operations by means of memory arbiter.

In accordance with Flinn's taxonomy this parallel computer architecture is classified in MIMD category - each processor module executes its own instructions and operates on its own data [2]. The system presents a symmetric multiprocessor (SMP), because all the data memory is in one block and has the same access time from every processor module.

The memory arbiter block contains a control logic that allows access to shared memory and solves collisions in case of simultaneous access. The arbiter accepts requests from each processor module and arbitrates which one is granted access to the shared memory at any time. Each processor module initiates a memory request signal by its I/O logic device when it wants access to the shared memory and deactivates it when finished. If more than one processor requests the memory at the same time, access is granted by using "round robin" priority access algorithm.

The block diagram of a system processor module is shown on the Figure 5. We use the KCPSM3 controller and the PicoBlaze processor (processor core) in it, designed by Xilinx. The processor core is delivered as synthesizable VHDL source code. PicoBlaze executes code (program for FFT implementation in this instance) from it's own program memory.

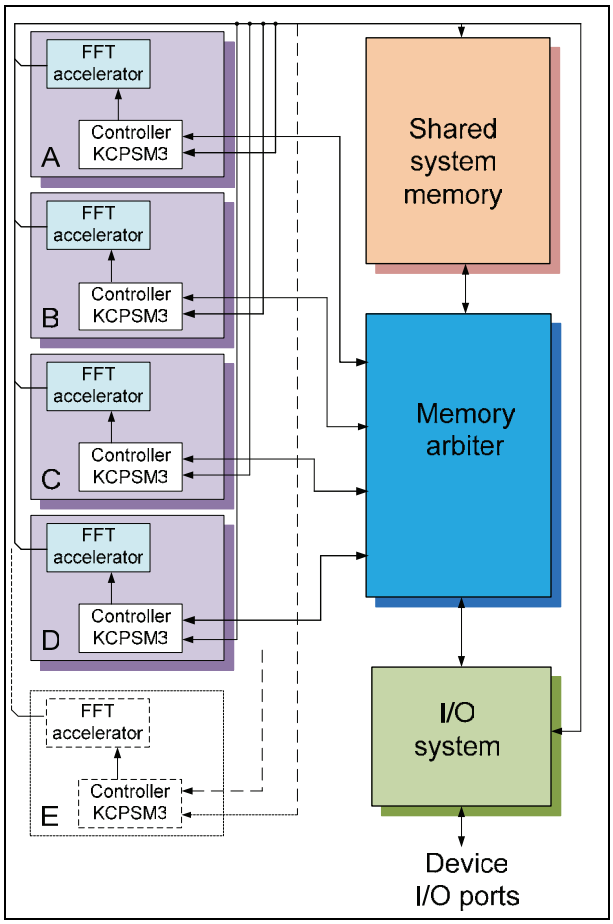

Fig.4. Block diagram of the multiprocessor system

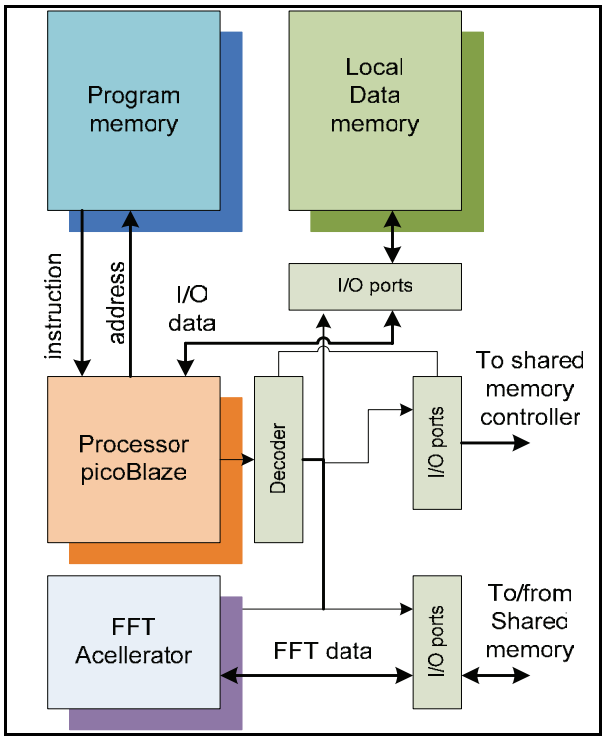

Fig.5.Block diagram of the processor module

The FFT accelerator block is a specialized coprocessor for FFT butterfly implementation. It executes the butterfly operations (shown on Figure 2) in pipeline mode with three pipeline stages [3].

Under instruction of PicoBlaze processor, the accelerator reads the butterfly operands and writes the butterfly results in the shared memory, avoiding the processor. Thus, the FFT computation speed grows. The PicoBlaze processors generate the operand and result addresses in the shared memory, only.

The organization of the parallel FFT execution is shown by the flow graph on the Figure 3. The processor modules (marked by **Pr#** on the graph) execute in parallel the butterflies of every task pass. The **barrier synchronization** is used by the multiprocessor system for this parallel execution [4]. That sort of synchronization guarantees, that no butterfly execution will proceed beyond any pass end, called the barrier, until every other butterfly execution in this pass has reached the barrier.

The multiprocessor system is designed, simulated and implemented on Xilinx FPGA – chips of Spartan3E family by means of WebPACK design environment, simulator ModelSym and VHDL source code. Figure 6 shows the chip area utilization (FPGA-chip xc3s500e) for a four-processor system.

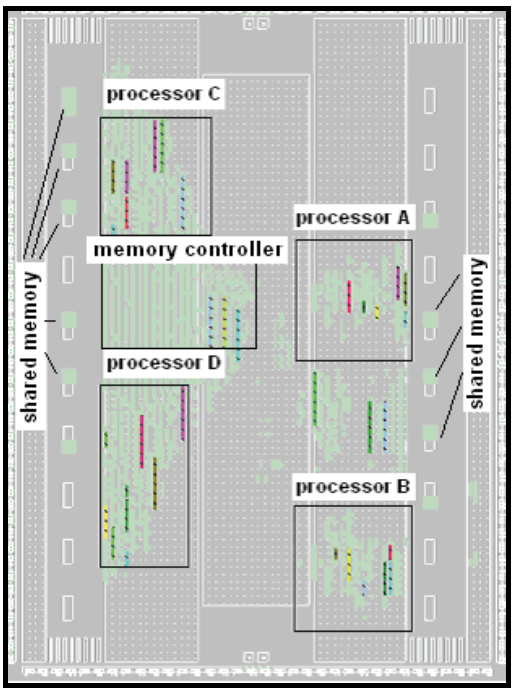

Fig.6. Four-processor system on FPGA-chip

III. Experimental results.

We have examined the parallel FFT implementation in multiprocessor systems on FPGA-chips with different number of processors. It is well-known, that the common shared memory is a “bottleneck” in shared memory systems. For that

reason, we have examined systems with different number of ports (one, two, four). Some of the important results of our investigations are shown on Figures 7-9.

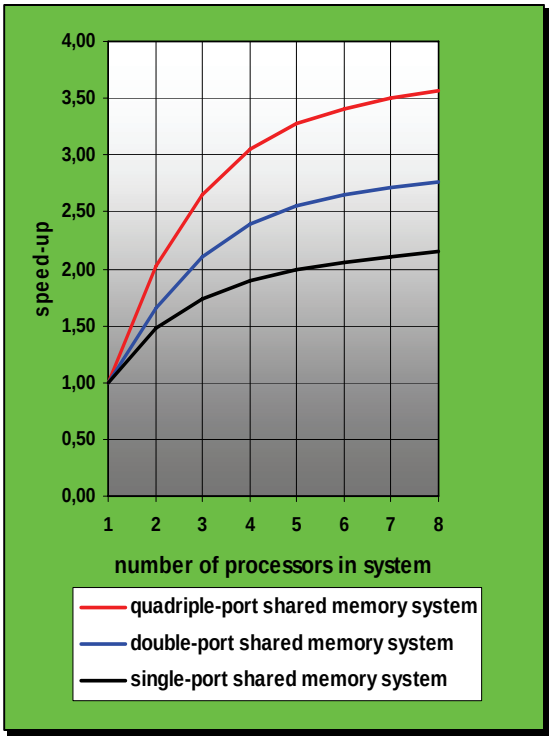

Fig.7. Multiprocessor speed-up as function of processor number and memory port number

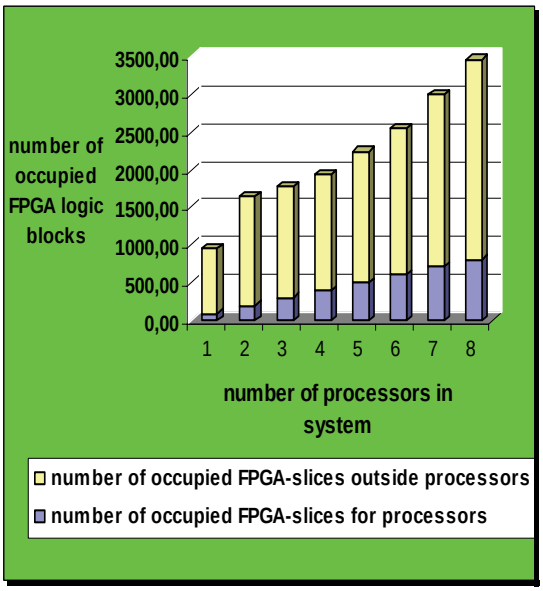

Fig.8. Number of occupied FPGA logic blocks as function of processor number

The results of our investigations are obtained by program and hardware simulation, parameter calculations, graphical and text documents of design environment working.

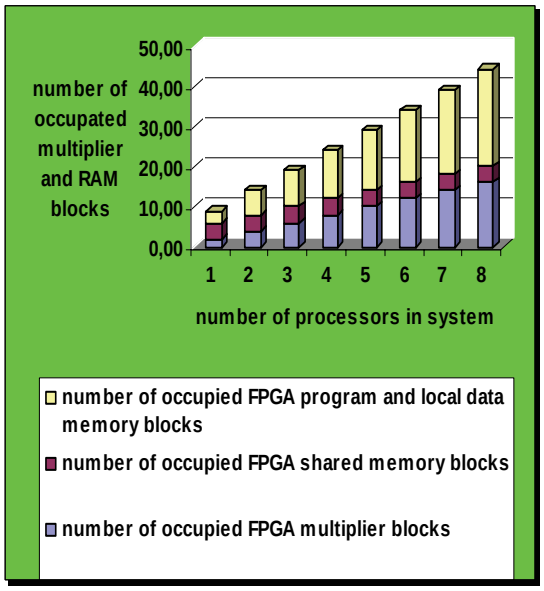

Fig.9. Number of occupied FPGA memory and multiplier blocks as function of processor number

IV. Analysis of results. Conclusion

Speed-up (the ratio between sequential FFT execution time of single processor and parallel execution time of multiprocessor) is probably the most important parameter of the multiprocessor system effectiveness. The results of Figure 7 suggest, that the processor number growth is low effective for more than 4-6 processors in the system (saturation threshold). Figure 8 and Figure 9 show, that the amount of occupied FPGA resources continue to grows, though.

In conclusion, we consider, that the results, exposed in this paper, will be useful to designers of Systems-On-Chip and Digital Signal Processing.

References

[1] Bowen B., W. Brown, “VLSI Systems Design for Digital Signal Processing”, Prentice-Hall, 1982.

[2] El-Rewini H., M. Abd-El-Barr, “Advanced Computer Architecture and Parallel Processing” John Wiley & Sons, 2005.

[3] Fleury M., A. Downton , “Pipelined Processor Farms – Structured design for embedded parallel systems”, John Wiley & Sons,2001.

[4] Kamburov G., P.Manoilov, P. Zaykov, P. Borovska, “Parallel Architecture Implemented in FPGA Based on Shared Memory System”, Third International Scientific Conference COMPUTER SCIENCE, Conference Proceedings, pp.96-101, Instanbul, Turkey, 2006.