JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

DESIGN OF EFFICIENT 4096-POINT RADIX 4 MEMORY BASED FFT ARCHITECTURE USING DSP SLICES

M.ANUSHA[1], Dr.B.NAGESHWAR RAO[2] ,Dr.T.VAMSHI[3]

[1]M.Tech Student, Talla Padmavathi College of Engineering, Somidi, Kazipet, Telangana, 506003
[2]Assoc Professor, Talla Padmavathi College of Engineering Student, Somidi, Kazipet, Telangana, 506003
[3]Assoc Professor, Talla Padmavathi College of Engineering, Somidi, Kazipet, Telangana, 506003
[1]mahankali.anusha94@gmail.com,[2]nagesh.south@gmail.com, [3]vamshi22g@gmail.com

Abstract

When it comes to computing the Discrete Fourier Transform, the Fast Fourier (FFT) is a highly efficient approach (DFT). Field Programmable Gate Arrays (FPGAs) are used to build the fast Fourier transform (FFT) architecture with radix-8 addition and multiplication in TWD of memory-based FFT. The butterflies components BTF0 and BTF1, TWD, and the I/O unit have been studied and optimised. A Spartan3E Gate array FPGA array has been used to implement the FFT in this architecture (FPGA). It was able to pass the simulation and test. In contrast to standard FFT, the proposed design's results are noticeably superior to the latter.

Key words: radix-8; FFT; FPGA; twiddle factor

1. Introduction

A wide variety of modern applications make use of FFT processors. In addition to broadband networks, digital television, and other areas such as radar, medical electronics, imaging, and the SETI project, this is a crucial component (Search for Extraterrestrial Intelligence). Several of these technologies are real-time devices, which necessitates that the systems generate an output within a predetermined time frame.In order to meet the demands at a reasonable cost, the FFT computations require a more advanced method than a special purpose processor can provide. Reducing hardware complexity, particularly in terms of size and power consumption, and enhancing CPU processing speed are among the top priorities for academics. It's called the DFT algorithm. What is the definition of the DFT transform

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn} \ , k = 0,1,2,\dots,N-1 \quad \text{(Eq.1)}$$

$$W_N^{kn} = e^{-j2\pi nk/N} \quad \text{(Eq.2)}$$

Calculating all N values using DFT necessitates doing N2complex multiplications and N(N-1)complex additions, as shown by these formula. It will take a long time to compute big values of N because the quantity of calculation and hence the computation time is roughly proportional to N2. Because of this, it is critical to decrease the number of addition and multiplication and additions. The Fast Fourier Transform (FFT) technique is an efficient method for calculating the DFT. The DFT algorithm's regularities are exploited by the FFT algorithm to address these issues.

1.1 FFT Processor.

There is a butterfly functional unit, memory and storage units for data, an address generation and sequential control unit in the FFT processor's construction. The butterfly baseband processor and the address generation unit are the two most important parts of the FFT processor. Input data, intermediate results, and output are all stored in the dual port RAM. Memory is used to store the twiddle factors. For both receiving information for butterfly actions and storing output results in RAM, the address creation unit generates addresses. Each module's control signals are generated by the sequential control unit.

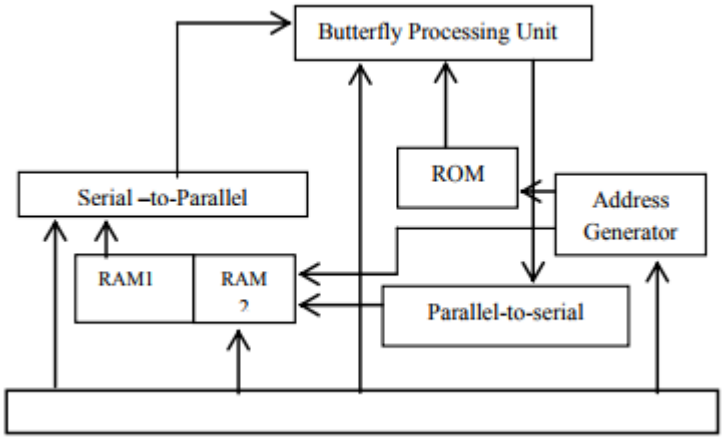


Fig.1 Block Diagram of FFT processor

1.2 Memory Based Architecture.

Memory-based architectures rely heavily on the available storage space in the computer's hard drive. Depending on the computation, memory blocks, and control unit, these architectures can have one or multiple processing elements (PEs) or butterflies. A single butterfly processor is typically used in memory-based designs, unlike pipelined architectures. As a result, they are able to save money on hardware. This means that a single butterfly kernel can only do one butterfly function per clock cycle. The high radix approach can be used to make up for this shortcoming. The system's clock rate can also be increased in order to meet the desired speed requirements.

Additionally, a memory-based architecture as depicted in Figure 2 includes a butterfly processor, a main memory and an address generator. Attached to main memory are the butterfly's input and output. The address generator controls the memory access read/write addresses. In order to perform the FFT computation, the commutator arranges data after reading and writing to and from the memory.

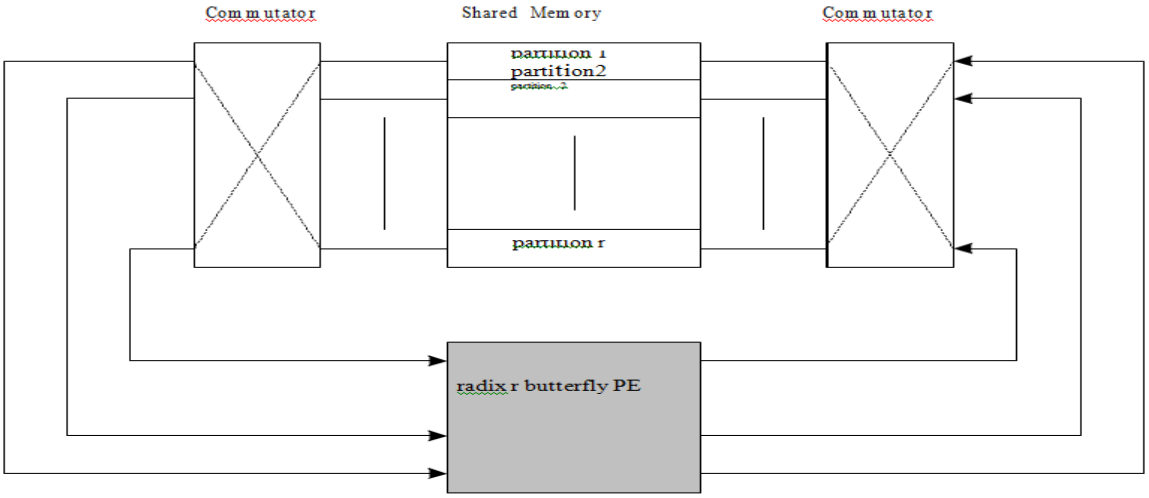


Figure 2 Schematic diagram of memory architecture

1.3 Memory–Based Processing Method.

(1.3.1) A solitary memory-based design.

The cheapest memory-based architectural is the sole memory-based architecture. Figure 3 shows that it only has one memory. Hence it's used in place of such a memory addressing scheme that saves the new FFT company has developed in much the same position used in last FFT operation data.

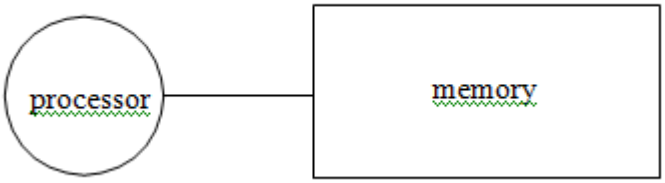


Figure 3 Single memory-based architecture block diagram

(1.3.2) Dual memory-based architecture.

To perform read-and-then-write operations, dual memory-based architecture uses a "ping-pong-like" strategy. Butterfly inputs and outputs alternate between all of the two memory. This suggests that it can simultaneously read and write data from the memory. Figure 4 depicts a dual memory-based architecture schematic diagram.

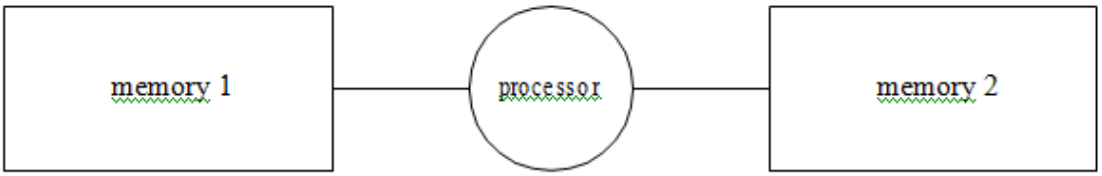


Figure 4 Dual memory-based architecture block diagram

(1.3.3) Architecture based on cache memory

Fig. 1.5 illustrates a cache memory-based architecture that consists of a single CPU, a cache memory, and a main memory. With a cache, it's like the single memory design in Figure 5. Even if the Dft length is enormous,

this architecture is designed to limit the amount of memory accesses and hence reduce power consumption.

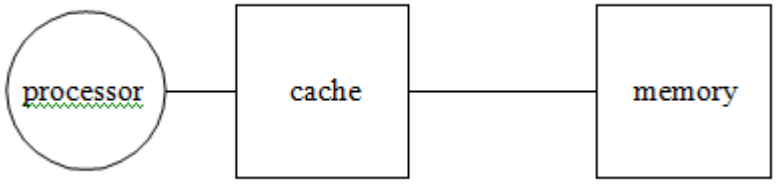


Figure 5 Cache memory-based architecture block diagram Radix-2 Memory-Based

2. Literature survey

This discrete Fourier transform can be effectively calculated by using fast Fourier transform (FFT), one of the most essential algorithms in digital signal processing. The Fast Fourier Transform (FFT) is used in a wide range of systems and applications. The system may request an extremely high rate of FFT computation from time to time. Pipelined FFTs are primarily employed for this reason. Performance requirements are less stringent in other systems. Instead, the architecture places demands on the available space and/or hardware resources. Memory-based FFTs, also known as in-place or iterative FFTs, are commonly used in these cases.The data is stored in a memory or a bank of memories in a memory-based FFT. Butterfly and rotator processors read the data from the memory and process it, before storing it back in the same place. This iterative process is repeated until all steps of the Merge sort have been computed. iterations. Memory-based FFTs have the advantage of decrease the quantity of butterfly and rotators because they can be reused for multiple phases of the FFT. Several memory-based FFT architectures have been proposed in the literature.. The processed element (PE) size is the most significant difference between the two (butterflies and rotators). Radix-2 butterflies are the most common, but other radices such as radix-4 and others are also used. Additionally, memory-based FFTs have a conflict-free access mechanism to the memory. N-point memory-based FFTs typically use N or 2N bytes of memory. In addition to the memory, there is also the issue of the access strategy. The FFT presented here is based on radix-8 memory and is new to the literature. There are a number of advantages to the design as proposed. In comparison to prior radix-4 techniques, it requires the bare minimum of N samples of memory plus a few extra multiplexers. DSP48E slices have been used to implement the proposed method on a programmable logic array (FPGA) (FPGA). Reduced hardware and distributed logic can be achieved by integrating aspects of the architecture into DSP48E. By using DSP48E slices, the suggested approach is a small FPGA solution that frees up space in the FPGA for more complicated and area-demanding components.

2.1 Design and implementation of a 1024-point pipeline FFT processor.

The formulation and simulation of a 1024-point pipelines FFT processor is shown. As an alternative to the standard FFT algorithm, this one uses the radix-2/sup 2 algorithm. The geographic regular of the new technique allowed for a modest VLSI implementation: only four complex multiplies and 1024 complex

information memory again for parallelizing 1K FFT processor. Because of its size, the device was designed to be implemented on just 40 mm/sup 2 of silicon. An external power supply can be used to perform sophisticated point forward as well as inverse FFTs in real time having sampling frequencies of up to 30 MHz on the 3.3 V model. For white noise input, the SQNR exceeds 50 dB.

Authors: S. He and M. Torkelson.

2.2 Pipelined radix-2k feed forward FFT architectures.

For pipelined FFT hardware architectures, radix-22 marked a turning point. It was later extended to radix-2k. It was only for mono delay feedback (SDF) designs that radix-2k was proposed, not for feed-forward architectures, also known as multi-path delay commutators. radix-2k (MDC). The radix-2k feed forwards (MDC) FFT designs are discussed in this study. As long as the number of concurrent samples is greater than two, radix-2k can be employed. The decompositions DIF and DIT, which both involve decimation in frequency, can also be utilised. In addition, the designs are capable of achieving extremely high throughputs, making them ideal for the most rigorous applications.. When several samples must be processed in parallel, the suggested radix-2k feed forward designs need less hardware than parallel feedback ones, often known as multi-path delay feedback (MDF). Since feedforward structures can be applied to both present and future systems, radix-2k feedforward architectures are a viable option for both current and future applications.

Authors: M. Garrido, J. Grajal, M. A. Sánchez, and O. Gustafsson.

2.3 Simplified control of FFT hardware.

A novel method for controlling fourier Transformation (FFT) circuitry has been described.. The approach generates indices for both inputs and outputs of each butterfly operation. It's also possible to divide each butterfly's inputs and outputs into separate memory units so that they can all be accessed simultaneously.

Authors: D. Cohen

3. Proposed system

Pipelined (distributive) and memory-based designs are the two main types of FFT processor architectures to choose from. No matter how long it takes to compute an FFT, memory-based FFTs recursively feed data via a single processor (PE) or numerous processors, each having a separate memory bank to store intermediate results. Memory-based FFTs outperform pipeline ones in terms of hardware efficiency. However, the butterfly radix & parallel data access contentions frequently limit the performance of memory-based FFTs.

3.1.Basic Architecture

Six rounds of the FFT algorithm are used in the architecture, which is based on the fact that in memory-based FFTs, there are six iterations in radix-4.

$$It = \frac{\log_2 N}{\log_2 r} = \frac{n}{\log_2 r}$$

Rather than a single N-sample memory, this architecture utilises four parallel N/4-sample memories. Reading and writing data in all of the circuit's memories at the same time minimises latency and enhances throughput. Thus, at each clock pulse, the PE collects and transmits four samples simultaneously, one from or to each of the four memory modules.

3.2 Memory banks:

Concurrent data accessing from various memory banks necessitates conflict-free address methods. For the first time, a method for checking the parity of one or even more radix-2 Units is presented. The entire amount of memory storage can be reduced to as little as N using an in-place method. There are conflict-free algorithms for mixed radix-4/2 in-place schemes that are then extended to a mixed-radix method.

It is possible to implement logic in Block RAM (functions as ROM memory) Determine how many inputs and outputs your logic function has. The address is formed via inputs (like the LUT structure) INIT parameters (or "initial" blocks) can be used to implement truth tables if they are smaller than the block RAM size. As an example, an 18Kb Block RAM may hold up to 18 10-input functions.

Memory can be blocked using HDL code

```
always @(posedge clock) begin
    if (en) begin
      if (we)
        ram[addr] <= wdata;
      rdata <= ram[addr];
    end
end
```

It takes the address from the address generator to store the input sequence samples. Write operation complete if rw==1 and WE write enable=1. We can retrieve from BRAM if rw==0.

3.3 Generation of the address

Initially, memory store samples in a logical order. MEM0 stores sample 0 to N/4 1, MEM1 samples N/4 to N/2 1, and etc. For each sample I, the memory location in memory is determined by its index number, which ranges from zero to N - 1.

$$P_1 \equiv \underbrace{b_{n-3}, b_{n-4} \ldots, b_0}_{\text{serial (address)}} \mid \underbrace{b_{n-1}, b_{n-2}}_{\text{parallel (memory)}}$$

Bn-1 and Bn-2 indicate which one of the four memory the sample is kept in, whereas Bn-3, Bn-4, B0 are used to determine the address of the memory in which the sample resides. As a result, a sample address must include the BRAM location's row and column positions in order to be valid.Samples with differing indices are handled by the butterfly at phase s in an FFT that uses radix-2 FFT In radix-4, the butterflies at phase s works on specimens that vary in bits bn2s1bn2s.. These samples must come simultaneously with the PE at each FFT iteration. Bits bn-1 and bn-2 are still in different memories during the first stage/iteration, according to the equation. Samples with a bit difference of bn3bn4 must arrive in parallel with the PE for the second iteration. The permutation is used to do this.

$$\sigma(u_{n-1}, \ldots, u_2|u_1, u_0) = u_{n-3}, u_{n-4}, \ldots, u_0|u_{n-1}, u_{n-2}$$

Only the substance of the memories is affected. The read/write memory address is obtained as follows:

$$W_{i+1} = R_i = c_{2i-1} \oplus m_1, c_{2i-2} \oplus m_0, \ldots c_1 \oplus m_1, c_0 \oplus m_0$$
$$c_{n-3}, \ldots, c_{2i} \qquad ($$

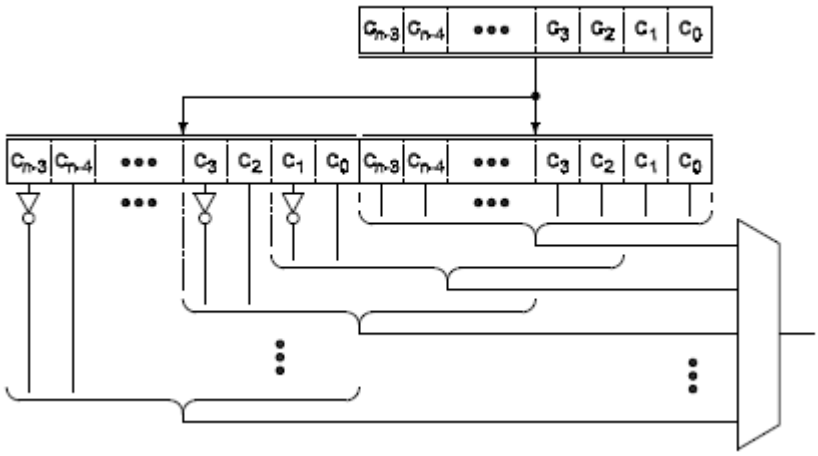


Fig. 6: Generation of the memory address for MEM2

Figure 6 shows how to generate the MEM2's memory address. The multiplexer selects the bits associated with each of the two copies of the counter. The sole components in the circuit are the NOT gating and the multiplexing.

4. Discussion on results:

Fig. 7.Entity diagram:

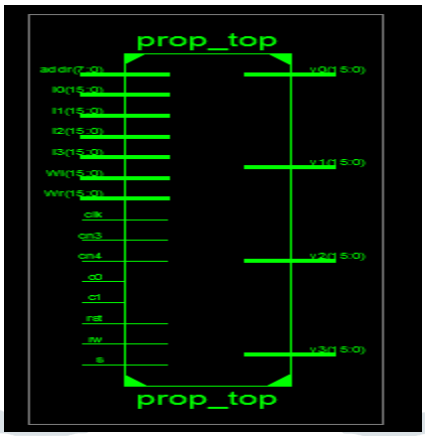


Fig. 8.RTL schematic:

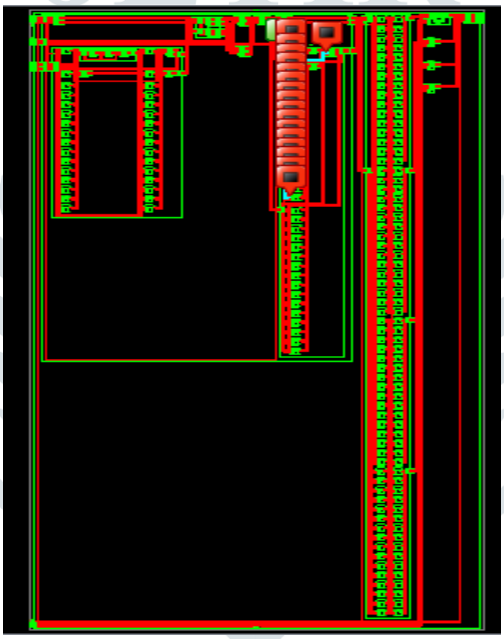
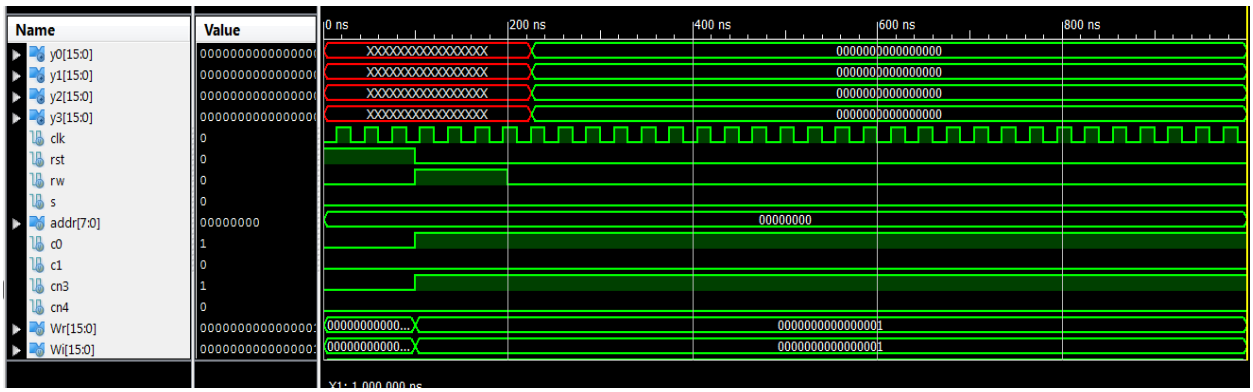


Fig. 9.Simulation results:



5. Conclusion

The FPGA implementation of a 4096-point radix-8 FFT is described in this work. It minimises the utilisation of the FPGA's internal memory. It was successfully emulated and verified. Using DSP slices, the suggested FFT has been efficiently implemented on an FPGA. With the same number of DSP layers and BRAM, the suggested solution uses less dispersed logic than earlier FPGA findings.

6. Future scope:

The methodologies provided in this thesis have been successful in their endeavour to develop arithmetic algorithms and architecture layer improvement initiatives for low power and high multiplier design. However, this study has its limits, and numerous future research avenues are available as follows: Higher order compressors such as 7:2, 9:2 can be utilised to collect the partial products in order to improve performance. Improved performance can be achieved by implementing a deep level pipeline architecture. Having the capacity to build very compact high-performance multipliers opens up a number of exciting new avenues of exploration. DSP and graphics applications, both of which require a lot of multiplication, might greatly benefit from the inclusion of multiple high-performance multiplier on the same chip. With a single or more very high-throughput multipliers working together on a single device we may be able to create chipset video signal processors.

7. References

[1] S. He and M. Torkelson, “Design and implementation of a 1024-point pipeline FFT processor,” in Proc. IEEE Custom Integr. Circuits Conf., May 1998, pp. 131–134.

[2] M. Garrido, J. Grajal, M. A. Sánchez, and O. Gustafsson, “Pipelined radix-2k feedforward FFT architectures,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 21, no. 1, pp. 23–32, Jan. 2013.

[3] D. Cohen, “Simplified control of FFT hardware,” IEEE Trans. Acoust., Speech, Signal Process., vol. 24, no. 6, pp. 577–579, Dec. 1976.

[4] Y. Ma and L. Wanhammar, “A hardware efficient control of memory addressing for high-performance FFT processors,” IEEE Trans. Signal Process., vol. 48, no. 3, pp. 917–921, Mar. 2000.

[5] Z.-G. Ma, X.-B. Yin, and F. Yu, “A novel memory-based FFT architecture for real-valued signals based on a radix-2 decimation-in-frequency algorithm,” IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 62, no. 9, pp. 876–880, Sep. 2015.

[6] B. G. Jo and M. H. Sunwoo, “New continuous-flow mixedradix (CFMR) FFT processor using novel in-

place strategy," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 52, no. 5, pp. 911–919, May 2005.

[7] X. Xiao, E. Oruklu, and J. Saniie, "Fast memory addressing scheme for radix-4 FFT implementation," in Proc. IEEE Int. Conf. Electro/Inf. Technol., Jun. 2009, pp. 437–440.

[8] P.-Y. Tsai and C.-Y. Lin, "A generalized conflict-free memory addressing scheme for continuous-flow parallel-processing FFT processors with rescheduling," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 19, no. 12, pp. 2290–2302, Dec. 2011.

[9] S.-J. Huang and S.-G. Chen, "A high-throughput radix-16 FFT processor with parallel and normal input/output ordering for IEEE 802.15.3c systems," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 59, no. 8, pp. 1752–1765, Aug. 2012.

[10] D. Reisis and N. Vlassopoulos, "Conflict-free parallel memory accessing techniques for FFT architectures," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 55, no. 11, pp. 3438–3447, Dec. 2008.

[11] C.-F. Hsiao, Y. Chen, and C.-Y. Lee, "A generalized mixedradix algorithm for memory-based FFT processors," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 57, no. 1, pp. 26–30, Jan. 2010.

[12] M. Garrido, "Efficient hardware architectures for the computation of the FFT and other related signal processing algorithms in real time," Ph.D. dissertation, Dept. Signals, Syst. Radiocommun., Tech. Univ. Madrid, Madrid, Spain, 2009.

[13] M. Garrido, J. Grajal, and O. Gustafsson, "Optimum circuits for bit reversal," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 58, no. 10, pp. 657–661, Oct. 2011.

[14] M. Garrido and J. Grajal, "Efficient memoryless CORDIC for FFT computation," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process., vol. 2. Apr. 2007, pp. II-113–II-116.