Design And Implementation Of Efficient FFT Processor Using Modular Multiplier

[1]S.MOUNIKA, [2]SURESH.V [3]S.SREECHANDRA

[1]*M.Tech student, Dept of ECE, Sri Mittapalli Institute of technology for women, Guntur, A.P*

[2]*Assistant Professor, Dept of ECE, Sri Mittapalli Institute of technology for women, Guntur, A.P*

[3] *HOD,Dept of ECE, Sri Mittapalli Institute of technology for women, Guntur, A.P*

ABSTRACT: *Algorithmic Based Fault Tolerance (ABFT) technique utilizes the algorithmic properties for detecting and correcting the errors. FFTs are the key building blocks in many communication and signal processing systems. An efficient conflict-free address scheme for arbitrary point memory-based fast Fourier transform (FFT) processor was exhibited in this paper. In the proposed scheme, a high radix decomposition method was utilized to reduce the computation levels and small radix connected multipath-delay-commutator butterfly units were adopted to eliminate the complexity of the computation engine as well. Several important functions of memory-based FFT processor were combined together, including the continuous-flow mode, variable computation size and conflict-free address scheme. Moreover, a prime factor algorithm was employed to decrease the multiplications and the twiddle factor storage when there subsist prime factors in the decomposition*

KEY WORDS: Error Correction Codes (ECC), Fast Fourier Transforms (FFTs), Soft Errors

I.INTRODUCTION

The complexity of communications and signal processing circuits increases every year. This is made possible by the CMOS technology scaling that enables the integration of more and more transistors on a single device. This increased complexity makes the circuits more vulnerable to errors. At the same time, the scaling means that transistors operate with lower voltages and are more susceptible to errors caused by noise and manufacturing variations. The importance of radiation-induced soft errors also increases as technology scales. Soft errors can change the logical value of a circuit node creating a temporary error that can affect the system Operation. To ensure That soft errors do not affect the operation of a given circuit, a wide variety of techniques can be used. These include the use of special manufacturing processes for the integrated circuits like, for example, the silicon on insulator. Another option is to design basic circuit blocks or complete design libraries to minimize the probability of soft errors.

The subject of the paper is hardware implementations of the RSA algorithm with larger than 1,024-bit modulus length. In particular, our objective is to create implementations that achieve high area-time efficiency, rather than creating very low area or ultra high speed implementations at the high cost of the other. The RSA algorithm, being the very first public-key encryption and digital signature algorithm since 1978, is ubiquitously deployed and used, from smart cards to cell phones and SSL boxes. Its security depends on the difficulty of factoring a modulus n to find its two prime factors p and q. The very first implementations of the RSA algorithm in Early 1980s assumed 512-bit modulus (and thus, two 256-bit primes) would be sufficient, but within a decade, advances in Factorization methods increased the modulus length to 1024 bits.

This has been the case for almost 2 decades, but now, as recently as 2010s, the security of 1,024-bitwas questioned. Many implementations now use 2,048-bitmodulus, while the National Institute of Standard and Technology (NIST) recommended 3,072-bit or 4,096-bit modulus size for the near future in order to maintain RSA secure. Needless to say, larger key sizes lead to longer processing time and more hardware resource when computing, due to the fact the RSA computation requires the modular exponentiation (xm modN), which is computed by repeated modular multiplications. Therefore, the performance of modular multiplication has a direct impact on the efficiency of RSA computation, and therefore, high performance modular multipliers supporting 3,072-bit or higher operand size are required.

Montgomery modular multiplication (MMM) is an efficient method to compute modular multiplication. Due to this fact, integer multiplication has been extensively studied in order to improve MMM. Existing multiplication methods can be classified into two groups. Methods of the first group are performed only in time domain, including the school book method, the Karatsuba method and the Toom-Cook method. Methods of the second group are performed in both time and spectral domains. Since the fast Fourier transform (FFT) based algorithm is applied to the second group, a lower asymptotic complexity can be achieved compared to the methods in the first ones. So the proposed system gives better results.

II. RELATED WORK

A number of architectures have been proposed for efficient implementation of multiplication over *GF* (2*m*). Multipliers with different basis of representation, e.g., dual basis, normal basis and polynomial basis have been realized to be used for various applications. The polynomial basis multipliers are therefore more widely used compared with the multipliers based on the other two basis. Several algorithms and architectures are suggested in the literature for polynomial basis multiplication for the fields generated by trinomials and pentanomials, primarily due to their computational simplicity.

All-one polynomials (AOP) or 1-equally spaced polynomials form a special class which can be used for simpler and more efficient implementation compared to trinomials and penanomial-based multipliers. The AOP-based representation of elements, thus, expected to have potential application in efficient hardware implementation of elliptic curve cryptosystems and error control coding. Irreducible AOPs are not as abundant as irreducible trinomials or pentanomials, but it is also not difficult to find the AOP bases for generating the finite fields. It is known that for *m* < 2000, there exists 108 possible AOP bases, and infinitely many more form > 2000. Efficient architectures for the field multiplication and the computation of power-sum of the form *(A*+*B*2*)* for field generated by AOPs have also been proposed in the literature. ltoh and Tsujii had proposed the first multiplier for *GF (*2*m)* generated by AOP which was followed by some bit-parallel architectures.

The bit-parallel designs are useful for low-latency realization, but due to their large critical path, they cannot provide high throughput rate and involve high average computation time which increases rapidly with the field order *m*. Systolic designs represent an attractive architectural paradigm for efficient VLSI and FPGA implementation due to their simplicity, regularity, and modularity of structure along with significant potential to yield a high-throughput rate using pipelining, parallel processing, or both. There are several bit-serial or digit-serial systolic structures and a few bit-parallel systolic structures for canonical basis multiplier over *GF (*2*m)* based on irreducible AOP.

In, the authors presented a transformation method to implement low complexity

Montgomery multipliers for all-one polynomials and trinomials. Recently, a novel cut-set retiming is proposed for area-time-efficient systolic multiplier structure based on irreducible AOP. Chen et al have suggested a scheme for AOP-based field multiplication through an extended polynomial basis representation. But there is no scalable design or digit-serial multipliers for irreducible AOP are suggested yet. In this paper, we present a new algorithm for modular reduction and an efficient recursive formulation for canonical basis multiplication over *GF (2m)* based on irreducible AOP.

III. EXISTED SYSTEM

The below figure (1) shows the architecture of existed system. Operations of the FMLM, are computed sequentially, while pipelined architectures are designed inside each unit. The below architecture consists of multiply adder, Fast Fourier Transform, Ripple Carry Adder, Subtracter, Shift Module, RAM Sets. Let us discuss each component in detail manner.

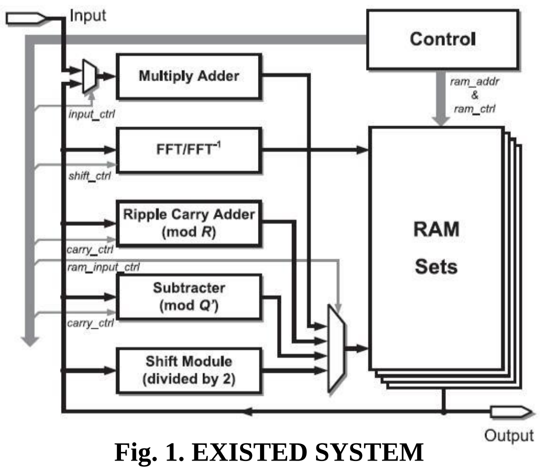


Fig. 1. EXISTED SYSTEM

The first and main important component in the architecture is multiply and adder unit. This unit implements the component wise multiplication and addition of FFT-RAM. To realize the component wise multiplication when operand size is not larger than few hundred bits then karatsuba method is used. The multiplier and adder units works with pipeline of 3 bit inputs and one bit output. At last to enhance the performance of multiplication, karatsuba method is applied recursively. This is about multiply and adder unit and let us discuss about FFT unit. is applied to FFT computation. The main comparison of in-place and constant geometry FFT is it has same connection network between every adjacent stages. The FFT is designed with six inputs. In this the four inputs forward the digits into BFSs for FFT computation and the other two inputs forward the pre-computed upper bound constraints into FSO.

Next one is RAM unit. RAM unit consists of several RAM sets which stores the pre-computed data, the intermediate results, and the final modular product. In RAM the data storage requirement during FFT-RAM computation is not trivial. Now to well manage the input and output of data and to reduce the wiring workload, RAM unit is built. The remaining are the Ripple Carry Adder (RCA), the Subtracter and the Shift Module units are responsible for the time domain operations, such as modulo R and Q0 reductions, conditional selections. Now to generate all control signals of entire system, control unit is designed. But this system does not gives better results, so a new system is proposed which is discussed in below section. Forward and reverse networks are formed in FFT unit. Basically, this unit is targeted on high clock frequency and small resource cost. Here constant geometry FFT.

IV. PROPOSED SYSTEM

The below figure (2) shows the architecture of proposed system. It consists of the following main parts: an input and an output index vector generator, a computation address generator, three different memory bank groups, a PE unit applied with the HRSB scheme, some commutators between the memory and the PE, some pre stored twiddle factors, and FFT size parameters, and an exponent scaling unit. The dashed lines represent the Control signals while the real lines denote the flowing data. Let us discuss each component in detail manner. The input index vector generator distributes the input data to different memory banks without data conflicts, and the output one reorders the output data to a natural sequence. The computation address generator obtains all the concurrent data of each cycle and stores back the intermediate results. Memory groups 1 and 2 are in a ping pong mode to hold two continuous data symbols in input sampling, and memory group 3 is used to output the computed data in right order

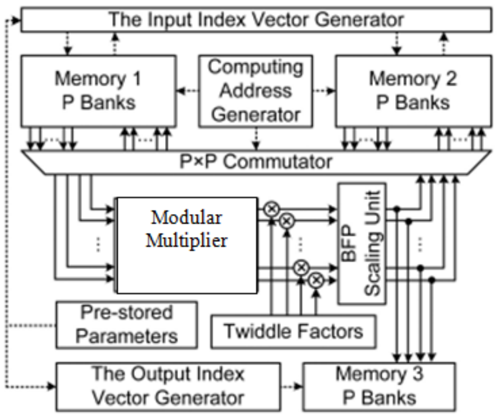


Fig. 2. PROPOSED SYSTEM

Three memory groups only exist in NSPP FFTs. For SPP FFTs, as in-place strategy is available, only two memory groups are enough. The HRSB unit is the kernel processing engine. The commutators located between the memories and HRSB unit provide efficient data routing mechanism which is controlled by the computation address generator. The twiddle factors and the FFT size parameters are all pre stored in register files, which are used to configure the FFT working modes. The exponent scaling unit controls scaling operations for block floating-point, which can increase the signal-to-quantization noise ratio and reduce the memory storage.

The computation can be completed within 128 clock cycles. The input and output index vector generators in the 2*n*-point FFT are merged to one. The only difference is that the binary representation of the index for the input is in a forward manner, while it is in a reversed manner for the output. In this paper, we present a new factorization method that will make the continuous-flow mode completely available and simultaneously ensure the effectiveness of the conflict free address scheme. So it gives efficient results compared to existed system.

V. RESULTS

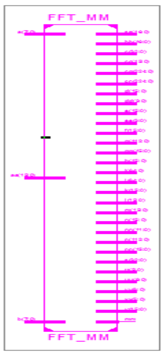


Fig. 3: RTL SCHEMATIC

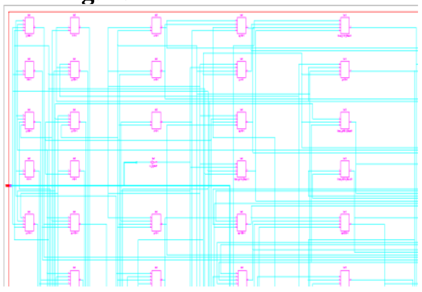


Fig. 4: TECHNOLOGY SCHEMATIC

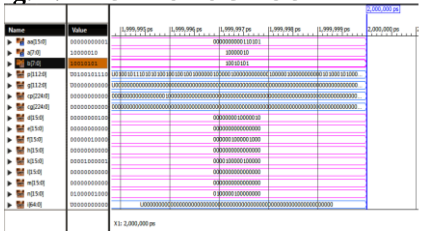
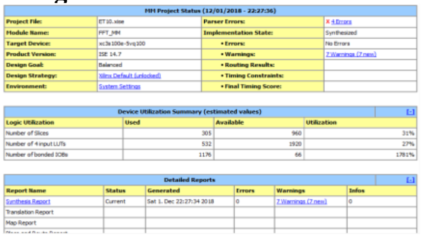


Fig. 5: OUTPUT WAVEFORM



MM Project Status (12/01/2018 - 22:27:36)			
Project File:	ET10.xise	**Parser Errors:**	X 4 Errors
Module Name:	FFT_MM	**Implementation State:**	Synthesized
Target Device:	xc3s100e-5vq100	• **Errors:**	No Errors
Product Version:	ISE 14.7	• **Warnings:**	7 Warnings (7 new)
Design Goal:	Balanced	• **Routing Results:**	
Design Strategy:	Xilinx Default (unlocked)	• **Timing Constraints:**	
Environment:	System Settings	• **Final Timing Score:**	

Device Utilization Summary (estimated values) [-]

Logic Utilization	Used	Available	Utilization
Number of Slices	305	960	31%
Number of 4 input LUTs	532	1920	27%
Number of bonded IOBs	1176	66	1781%

Detailed Reports [-]

Report Name	Status	Generated	Errors	Warnings	Infos
Synthesis Report	Current	Sat 1. Dec 22:27:34 2018	0	7 Warnings (7 new)	0
Translation Report					
Map Report					

Fig. 6: REPORT

VI. CONCLUSION

We implemented memory-based FFT implementations with generalized efficient conflict-free address schemes. Address schemes for different FFT lengths are integrated in this paper to endorse FFT processing for various systems. The memory bank and address can be obtained by modulo and multiplication operations of the decomposition digits. For both SPP and NSPP FFTs, high-radix algorithm and parallel-processing technique can be used to increase the throughput. And the address scheme for FFTs applied with PFA is explored. Moreover, a decomposition method, named HRSB, is designed to attire the high-radix algorithm. Full hardware architectures for the FFTs in LTE systems are illustrated, including the index vector generator, the butterfly engine.

VII. REFERENCES

[1] N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, Dependability in Electronic Systems: Mitigation of Hardware Failures, Soft Errors, and Electro-Magnetic Disturbances. New York, NY, USA: Springer-Verlag,2010.

[2] R. Baumann, "Soft errors in advanced computer systems," IEEE Des. Test Comput., vol. 22, no. 3, pp. 258–266, May/Jun. 2005.

[3] M. Nicolaidis, "Design for soft error mitigation," IEEE Trans. Device Mater. Rel., vol. 5, no. 3, pp. 405–418, Sep. 2005.

[4] A. L. N. Reddy and P. Banerjee, "Algorithm-based fault detection forsignal processing applications," IEEE Trans. Comput., vol. 39, no. 10, pp. 1304–1308, Oct. 1990.

[5] T. Hitana and A. K. Deb, "Bridging concurrent and non-concurrent error detection in FIR filters," in Proc. Norchip Conf., Nov. 2004, pp. 75–78.

[6] S. Pontarelli, G. C. Cardarilli, M. Re, and A. Salsano, "Totally fault tolerant RNS based FIR filters," in Proc. 14th IEEE Int. On-Line Test Symp. (IOLTS), Jul. 2008, pp. 192–194.

[7] B. Shim and N. R. Shanbhag, "Energy-efficient soft error-tolerant digital signal processing," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 14, no. 4, pp. 336–348, Apr. 2006.

[8] E. P. Kim and N. R. Shanbhag, "Soft N-modular redundancy," IEEETrans. Comput., vol. 61, no. 3, pp. 323–336, Mar. 2012.

[9] J. Y. Jou and J. A. Abraham, "Fault-tolerant FFT networks,"IEEE Trans. Comput., vol. 37, no. 5, pp. 548–561, May 1988.

[10] S.-J. Wang and N. K. Jha, "Algorithm-based fault tolerance for FFT networks,"IEEE Trans. Comput., vol. 43, no. 7, pp. 849–854, Jul. 1994