Design And Implementation Of A High Efficient Architecture Of FFT Processor

M.Sasikala 1 , D.Venkanna Babu2

1. PG Scholar, Department of ECE, Ramachandra College of Engineering and Technology, JNTUK, A.P...Mail id –sasi.munni@gmail.com

2.Associate Professor, Department of ECE, Ramachandra College of Engineering and Technology, , JNTUK, A.P...Mail id-dvenkannababu@gmail.com

ABSTRACT: *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. Soft errors pose a reliability threat to modern electronic circuits. This makes protection against soft errors a requirement for many applications. For some applications, an interesting option is to utilize algorithmic-based fault tolerance (ABFT) techniques that try to exploit the algorithmic properties to detect and correct errors. Signal processing and communication applications are well suited for ABFT. One example is fast Fourier transforms (FFTs) that are a key building block in many systems. Several protection schemes have been proposed to detect and correct errors in FFTs. In modern communication systems, it is increasingly common to find several blocks operating in parallel. Recently, a technique that exploits this fact to implement fault tolerance on parallel filters has been proposed. In this brief, this technique is first applied to protect FFTs. Then, two improved protection schemes that combine the use of error correction codes and Perceval checks are proposed and evaluated.*

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

I.INTRODUCTION

One of the main problem in real time communication is repetition of corrupt messages. Here the data should be delivered with low delay and the use of techniques will avoid the overloads by transmitting. During the digital information transmitting through a channel, practically inevitable errors are produced. To ensure reliable transmission, the data are further encoded Via Error Correcting Code (ECC).This could be used to recognize and correct errors. In this work the well-known binary linear block Hamming codes are used because they have been used in the optimization problems that we accelerate thanks to the circuit explained further on. A binary linear (N, k) code is a k-dimensional subspace of the space of N-bit code words, and therefore has 2^K code words. But we solve for blocks or subsets of M code words in the code, where $M \leq 2^K$, used to transmit a message. When a message is transmitted, its binary string can suffer modifications (changed bits), arriving an incorrect code word in the receiver side

The complexity of correspondences and flag preparing circuits builds each year. This is made conceivable by the CMOS innovation scaling that empowers the mix of an ever increasing number of transistors on a solitary gadget. This expanded unpredictability makes the circuits more defenceless against mistakes. In the meantime, the scaling implies that transistors work with lower voltages and are more powerless to blunders caused by commotion and assembling varieties. The significance of radiation-instigated delicate blunders likewise increments as innovation Scales. Soft errors can change the intelligent estimation of a circuit hub making a brief mistake that can influence the framework Operation. To guarantee that delicate blunders don't influence the task of a given circuit, a wide assortment of methods can be utilized. These incorporate the utilization of extraordinary assembling forms for the coordinated circuits like, for instance, the silicon on encasing. 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 (*2*m)* based on irreducible AOP.

III. EXISTED SYSTEM

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 (*2*m)* based on irreducible AOP.

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.

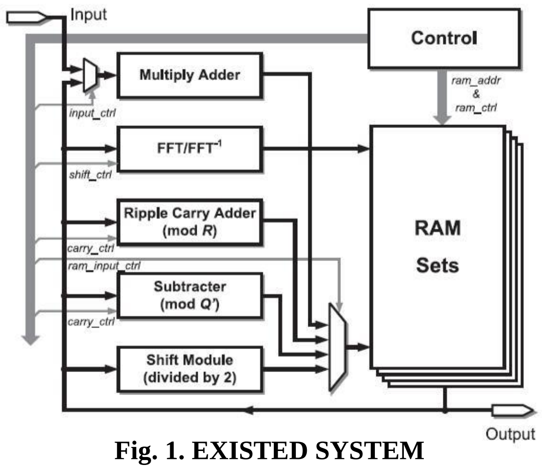


Fig. 1. EXISTED SYSTEM

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 first technique of Montgomery FFT multiplication. The beginning stage for our work is the security plot dependent on the utilization of ECCs that was exhibited for computerized channels. This plan is appeared in Fig. 1. In this system, a straightforward single revision Hamming code is utilized. The contributions to the three repetitive modules are direct blends of the information sources and they are utilized to check straight mixes of the yields. This will be indicated asc1 check. The same reasoning applies to the other two redundant modules that will provide checks c2andc3. Based on the distinctions saw on every one of the checks, the module on which the mistake has happened can be resolved. When the module in mistake is known, the error can be amended by remaking its yield utilizing the rest of the modules. Comparative conditions can be utilized to address errors on alternate modules. Further developed ECCs can be utilized to address error on different modules if that is required in a given application. This shows how the overhead decreases with the number of FFTs. So to overcome this a new system is proposed which is discussed in below section.

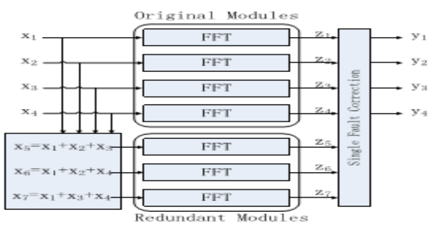


Fig. 2. MONTGOMERY FFT MULTIPLICATION SYSTEM

The below figure (3) shows the architecture of proposed system. In this system parallel FFT operation is performed. The operations involved in FMLM are computed sequentially. In this the pipelined architecture is designed for each unit. The principle advantage over the one is parity SOS is to diminish the quantity of SOS checks required. The mistake area process is equivalent to for the ECC and remedy is as in the equality SOS scheme. They are three main contributions of proposed system are

1) Error Correction Code is assessed to protect the parallel FFTs which show its effectiveness in terms of overhead and protection effectiveness.
2) A new technique is proposed based on the use of Parseval or sum of squares (SOSs) checks combined with parity FFT.
3) A new technique is proposed on which the ECC is used on the SOS checks instead of the FFTs.

The triplication of these blocks has a small impact on circuit complexity as they are much simpler than the FFT computations. A final observation is that the ECC scheme can detect all errors that exceed a given threshold (given by the quantization used to implement the FFTs). On the other hand, the SOS check detects most errors but does not guarantee the detection of all errors. Therefore, to compare the three techniques for a given implementation, fault injection experiments should be done to determine the percentage of errors that are actually corrected. This means that an evaluation has to be done both in terms of overhead and error coverage. . A four-point decimation-in-frequency FFT core is used to compute the FFT iteratively. This core has been developed to implement MIMO-OFDM for wireless systems. The implementation of the four-point FFT core is shown in Fig. (3).

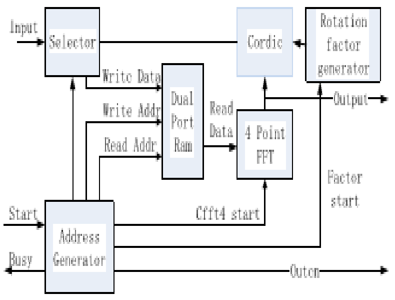


Fig 3. FFT IMPLEMENTATION

The two proposed schemes and the ECC scheme presented have been implemented on an FPGA and evaluated both in terms of overhead and error coverage. A four-point decimation-in-frequency FFT core is used to compute the FFT iteratively. This core has been developed to implement MIMO-OFDM for wireless systems. The below figure (4) shows the architecture of proposed system. The number of FFT points is programmable and the rotation coefficients are calculated on-line for each stage and stored in registers. For the evaluation, a 1024 points FFT is configured with five stages calculation (log41024=5), so in total 5 ∗1024=5120 cycles are needed to calculate the FFT for 1024 input samples. The inputs are 12-bit wide and the outputs are 14-bit wide. For the redundant FFT, the bit widths are extended to 14 and 16 bit, respectively, to cover the larger dynamic range (as the inputs are the sum of several signals)

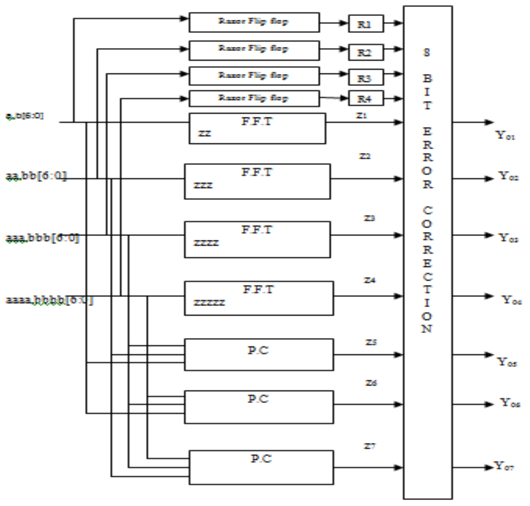


Fig. 4. PROPOSED SYSTEM

To minimize the impact of round offs on the fault coverage, the outputs of the accumulator are 39-bit wide. For the evaluation, several values of the number of parallel FFTs are considered. This is done to compare the different techniques as a function of the number of parallel FFTs in the original system. The error detection and correction blocks are implemented as multiplexers that select the correct output depending on the error pattern detected. In the Error Correction Codes (ECC) technique, each filter can be equivalent of a bit and by using addition parity check bits can be computed. The operation of this technique is the output of the sum of the several inputs is the sum of the individual outputs. So, this is valid for any linear operation. This scheme is evaluated by using FPGA implementations to assess the protection overhead. The protection overhead can be reduced by combining the use of ECCs and parseval checks. Hence this proposed system gives effective results compared to existed system.

V. RESULTS

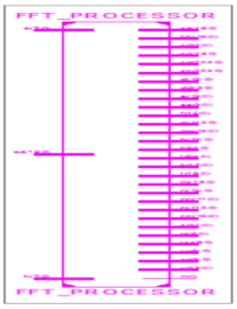


Fig. 5: RTL SCHEMATIC

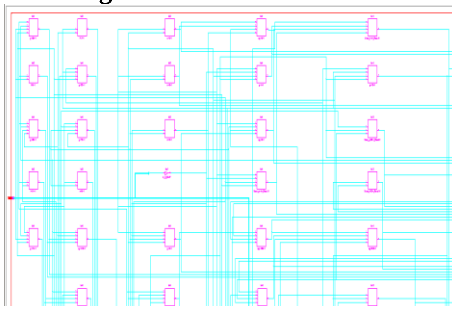
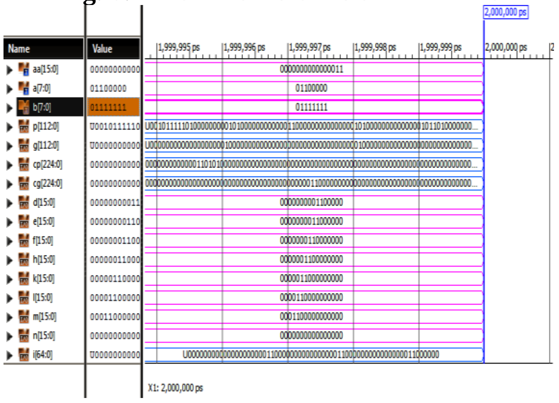
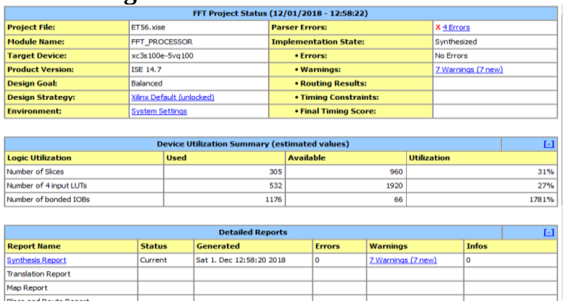


Fig. 6: TECHNOLOGY SCHEMATIC



Name	Value	
aa[15:0]	00000000000	0000000000000011
a[7:0]	01100000	01100000
b[7:0]	01111111	01111111
p[112:0]	U0010111110	U0010111110100000000101000000000001100000000000000101000000000000010110100000000...
g[112:0]	U0000000000	U000000000000000000100000000000000000000000000000001000000000000000000000000000...
cp[224:0]	00000000000	000000000000110101000000000000000000000000000000000000000000000000000000000000...
cg[224:0]	00000000000	000000000000000000000000000000000000001100000000000000000000000000000000000000...
d[15:0]	00000000011	0000000001100000
e[15:0]	00000000110	0000000011000000
f[15:0]	00000001100	0000000110000000
h[15:0]	00000011000	0000001100000000
k[15:0]	00000110000	0000011000000000
l[15:0]	00001100000	0000110000000000
m[15:0]	00011000000	0001100000000000
n[15:0]	00000000000	0000000000000000
i[64:0]	U0000000000	U0000000000000000000001100000000000000001100000000000000011000000

Fig. 7: OUTPUT WAVEFORM



FFT Project Status (12/01/2018 - 12:58:22)			
Project File:	ET56.xise	Parser Errors:	X 4 Errors
Module Name:	FFT_PROCESSOR	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 12:58:20 2018	0	7 Warnings (7 new)	0
Translation Report					
Map Report					

Fig. 8: REPORT

VI. CONCLUSION

In this work, we proposed a modified version of the FFT based Montgomery modular multiplication algorithm under McLaughlin's framework (FMLM3). By applying cyclic and nega-cyclic convolutions to compute the modular multiplication steps, the zero-padding operation is avoided and the transform length is reduced by half compared to the regular FFT-based multiplication. Furthermore, we explored for some special cases, the number of transforms can be further reduced from 7 to 5 without extra computational efforts, so that the FMLM3 can be further accelerated. A general method of efficient parameter set selection has been summarized for a given operand size. The estimation results indicate a feasible physical approach can be implemented which could trade area cost for faster speed by adding more butterfly structures. The Virtex-6 FPGA implementation results shows the proposed FMLM3 with both one and two butterfly structures have better area latency efficiency than the state-of-the-art FFT-based Montgomery modular multiplication.

VII. REFERENCES

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," Commun. ACM, vol. 21, no. 2, pp. 120–126, 1978.

[2] R. L. Rivest, "A description of a single-chip implementation of the RSA cipher," Lambda, vol. 1, no. Oct.–Dec., pp. 14–18, 1980.

[3] "Recommendation for key management," NIST, Tech. Rep. Special Publication 800-57, Part-1, Rev.-3, 2012.

[4] P. L. Montgomery, "Modular multiplication without trial division,"Mathematics Comput., vol. 44, no. 170, pp. 519–521, 1985.

[5] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," Soviet Physics Doklady, vol. 7, 1963, Art. no. 595.

[6] S. A. Cook and S. O. Aanderaa, "On the minimum computation time of functions," Trans.Amer.Math. Soc., vol. 142, pp. 291–314, 1969.

[7] A. Sch€onhageand V. Strassen, "Schnelle multiplikation Großer Zahlen," Computing, vol. 7, no. 3/4, pp. 281–292, 1971.

[8] M. F€urer, "Faster integer multiplication," SIAM J. Comput., vol. 39, no. 3, pp. 979–1005, 2009.

[9] D. Harvey, J. van der Hoeven, and G. Lecerf, "Even faster integer multiplication," CoRR, vol. abs/1407.3360, 2014. [Online]. Available: http://arxiv.org/abs/1407.3360

[10] S. Covanov and E. Thom_e, "Fast arithmetic for faster integer multiplication,"
CoRR, vol. abs/1502.02800, 2015. [Online]. Available: http://arxiv.org/abs/1502.02800