

## Quad Core Dual Field Cryptoprocessor on FPGA Platform

C. Veeraraghavan<sup>1</sup>, K. Rajendran<sup>2</sup>

<sup>1</sup>Department of Electronics and communication, Sri Krishna Arts and Science College, Coimbatore, Tamilnadu, India

<sup>2</sup> Department of Electronics, Government Arts College, Kulithalai, Karur District, Tamilnadu India

---

**Abstract:** This paper is devoted to the design of Quad core crypto processor for executing both Prime field and binary extension field instructions. The proposed design is specifically optimized for Field programmable gate array (FPGA) platform. Combination of two different field (prime field  $GF(P)$  and Binary extension field  $GF(2^m)$ ) instructions execution is analyzed. Quad core will execute four instruction at a same time. The design is implemented in Spartan 3E, virtex4 and virtex5. The performance results between them are compared. The implementation result shows the execution of parallelism using dual field instructions.

**Keywords:** Binary extension field, Cryptoprocessor, FPGA, Primefield

---

### I. Introduction

The term cryptography represents the encryption of data. For the secured data communication cryptography is essential one. Cryptography is used now a day in a variety of different applications. A secure crypto processor is a dedicated microprocessor chip for carrying out cryptographic operation. Every application has its own design criteria and raises special requirements for hardware designs. The most fundamental decision concerning future hardware designs is whether to use a binary-extension field or a prime field as basis of the used crypto processor. Only a few papers compared binary and prime fields in hardware. Dual-Field Arithmetic Unit for  $GF(p)$  and  $GF(2^m)$  is designed by Johannes Wolkerstorfer [1]. Jun-Hong Chen et al. designed the high performance unified field reconfigurable cryptoprocessor [2].

The cryptography operations involving the integer value is carry over by  $GF(p)$ . The binary extension fields  $GF(2^m)$ , where elements can be represented as polynomials instead of integers. Binary fields  $GF(2^m)$  are considered advantageous for hardware solutions because addition and modular reduction of polynomials are somewhat easier than those of integers. We will present a dual-field arithmetic unit that is capable to calculate the operations in both types of fields:  $GF(p)$  and  $GF(2^m)$

A reconfigurable multicore cryptoprocessor for multi channel communication systems [3] result shows the increase in speed of execution. For increasing the speed of execution this paper is designing the quad core crypto processor. The quad core will execute the instruction concurrently. The parallel execution will increase the speed of execution. The parallelization of the hardware design is a significant ongoing topic of research. At the same time for decreasing the power consumption this design is implemented in the FPGA.

Santosh Ghosh et al. design the secure dual core cryptoprocessor using the prime field instructions [4]. This paper is concentrate to design a quad core crypto processor used to execute both prime and binary extension field instructions.

### II. Implementing Prime Field And Binary Extension Field In FPGA

The Prime field and Binary extension field consist of several hardware blocks. In this section we discuss about the hardware design involvement of both prime and Binary extension field.

#### 2.1. Prime field unit architecture

Blakley introduced an algorithm to perform Modular multiplication of two integers A and B modulo an integer M [5]. It is an iterative binary double-and-add algorithm. The main idea of the algorithm is that it keeps the intermediate result after each iteration below the modulus value, which it avoids final division. In this paper, the modulus M corresponds to p and we say it  $F_p$  multiplication. All arithmetic in  $F_p$  are performed in two's complement number system, which avoids input and output conversions like existing implementations [6], [7].

The main difficulty of the Blakley algorithm is the computation of addition on large operands. The modified Blakley algorithm for large operands is shown in [8] and [9]. The use of carry save adder (CSA) helps to speed up the repeated additions on large operands. However these modified versions require at least one final addition on large carry chain. Some pre-computed values too are used by this technique which requires additional time and storage area.

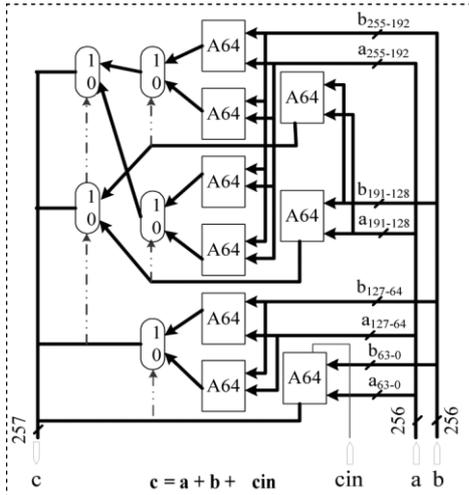


Fig. 1. Internal Structure of a 256 bit adder

At every level of hierarchy it adds one additional MUX in the critical path. Thus the latency of a 256-bit adder is  $1 FCC + 3$  MUX delay, which is 9.9 ns on a Virtex-4 FPGA, whereas the latency of a 256-bit carry look ahead adder on the same platform is 16.7 ns, which is 1.7 times slower than the above technique. we develop a programmable  $F_p$ -primitive based on above 256-bit high-speed adder circuits. Prime field operations carry over are addition, subtraction, and multiplication. Fig. 1 depicts the overall resulting architecture of the  $F_p$  -adder/subtractor/multiplier unit, where the internal dataflow of A256 blocks are shown in Fig. 1.

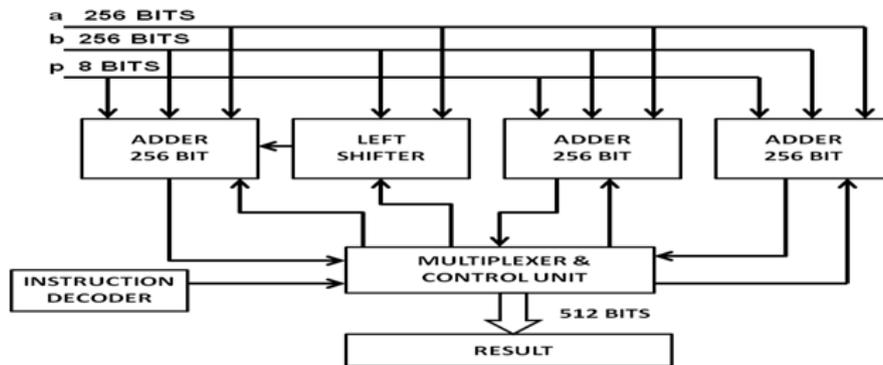


Fig2. Block diagram of  $F_p$  adder/subtractor/multiplier unit

The architecture for prime field consists of several independent blocks which operate in parallel for accelerating the execution of respective operations. The adder unit is used for performing the various addition and subtraction function involved in the algorithm. Multiplication is performed by the help of the left shifter. At first the instruction is decoded, depending upon the instruction algorithm is selected. Various control signals depending upon the algorithm is produced, that are used for executing the instructions.

### 2.2 Binary extension field unit block diagram

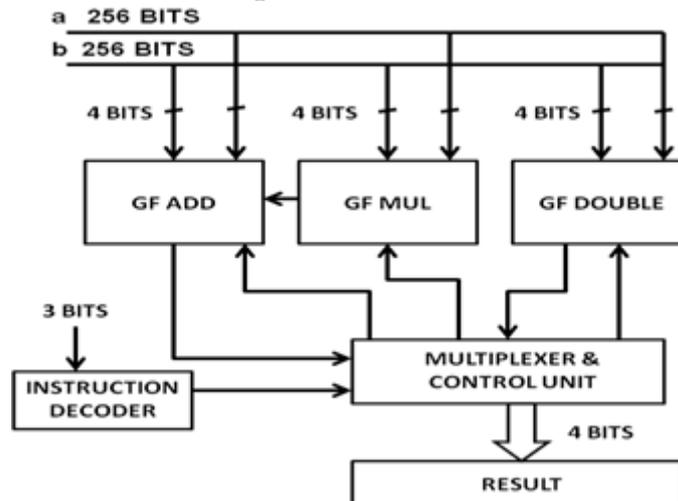


Fig.3. Binary extension Unit Block Diagram

The binary extension unit designed for the polynomial equation  $p(x)=x^4+x+1$ . Therefore out of 256 input bits 4 bits are taken for the each binary extension unit blocks. The binary extension field unit consists of GF addition, multiplication and double. The binary field addition is designed by using the XOR gate operation.

The multiplier unit is designed by the look up table procedure. So this block is consist of memory blocks, which are used for storing the all possible output. The GF double unit consists of a simple addition unit.

### III. Instructions Implementation For Prime Field And Binary Extension Field

#### 3.1. Instruction 1- Computation of $F_p$ Multiplication

The  $F_p$  Multiplication based on the following algorithm

**Algorithm 1:** The interleaved multiplication based on Montgomery ladder.

**Input:**  $p, a = \sum_{i=0}^{n-1} 2^i a_i$  and  $b = \sum_{i=0}^{n-1} 2^i b_i$ .

**Output:**  $a \cdot b \pmod p$

1.  $S_1^{(n)} \leftarrow 0; S_2^{(n)} \leftarrow a;$
2. **For**  $i = n - 1$  down to 0 **do**
3. **If**  $b_i = 1$  **then**  $u^{(i)} \leftarrow s_2^{(i+1)};$  **else**  $u^{(i)} \leftarrow s_1^{(i+1)}$
4.  $v_1^{(i)} \leftarrow 2u^{(i)};$
5.  $v_2^{(i)} \leftarrow s_1^{(i+1)} + s_2^{(i+1)};$
6.  $w_1^{(i)} \leftarrow v_1^{(i)} + (-p) + 1;$
7.  $w_2^{(i)} \leftarrow v_2^{(i)} + (-p) + 1;$
8.  $c_1^{(i)} \leftarrow (v_1^{(i)})_n \mid (w_1^{(i)})_n;$
9.  $c_2^{(i)} \leftarrow (v_2^{(i)})_n \mid (w_2^{(i)})_n;$
10. **If**  $c_1^{(i)} = 1$  **then**  $t_1^{(i)} \leftarrow w_1^{(i)};$  **else**  $t_1^{(i)} \leftarrow v_1^{(i)};$
11. **If**  $c_2^{(i)} = 1$  **then**  $t_2^{(i)} \leftarrow w_2^{(i)};$  **else**  $t_2^{(i)} \leftarrow v_2^{(i)};$
12. **If**  $b_i = 1$  **then**  $s_1^{(i)} \leftarrow t_1^{(i)}$  **else**  $s_1^{(i)} \leftarrow t_2^{(i)};$
13. **If**  $b_i = 1$  **then**  $s_2^{(i)} \leftarrow t_2^{(i)};$  **else**  $s_2^{(i)} \leftarrow t_1^{(i)};$
14. **end for**
15. **return**  $s_1^{(0)};$

[In this algorithm,  $x^{(i)}$  represents the value of  $x$  at  $i$ th iteration,  $(x)_n$  indicates the  $n$ th bit of  $x$ , and  $\mid$  indicates logical OR.]

#### 3.2. Instruction 2- Computation of $F_p$ Addition

The  $F_p$  Addition is based on the following algorithm

**Algorithm 2:** The addition in prime field.

**Input**  $p, a = \sum_{i=0}^{n-1} 2^i a_i$  and  $b = \sum_{i=0}^{n-1} 2^i b_i$ .

**Output**  $a + b \pmod p$ .

1.  $s_1 \leftarrow a; s_2 \leftarrow b;$
2.  $v_2 \leftarrow s_1 + s_2;$
3.  $w_2 \leftarrow v_2 + (-p) + 1;$
4.  $c_2 \leftarrow (v_2)_n \mid (w_2)_n;$
5. **If**  $c_2 = 1$  **then**  $t_2 \leftarrow w_2;$  **else**  $t_2 \leftarrow v_2;$
6. **return**  $t_2;$

#### 3.3. Instruction 3- Computation of $F_p$ Subtraction

The  $F_p$  Subtraction based on the following Algorithm

**Algorithm 3:** The subtraction in prime field.

**Input**  $p, a = \sum_{i=0}^{n-1} 2^i a_i$  and  $b = \sum_{i=0}^{n-1} 2^i b_i$ .

**Output**  $a - b \pmod p$ .

1.  $s_1 \leftarrow a; s_2 \leftarrow b;$
2.  $v_2 \leftarrow s_1 + (-s_2) + 1;$
3.  $w_2 \leftarrow v_2 + p;$
4.  $c_2 \leftarrow (v_2)_n$
5. **If**  $c_2 = 1$  **then**  $t_2 \leftarrow w_2;$  **else**  $t_2 \leftarrow v_2;$
6. **return**  $t_2;$

#### 3.4 Instruction 4 - Binary Extension field multiplication

Values in  $GF(2^4)$  are 4-bits each, spanning the decimal range [0..15]. Multiplication takes place on 4-bit binary values (with modulo 2 addition) and then the result is computed modulo  $P(x) = (10011) = 19$  (decimal).

Bit-parallel finite field multiplication using polynomial basis is analysed by Huapeng Wu [10]. Here we are generating the polynomial based multiplication for the equation  $p(x)=x^4+x+1$ . This operation is developed by the look up table procedure.

For example: Polynomials in Galois Field  $GF(2^4) = GF(16)$  based on  $P(x) = x^4 + x + 1$

The terms above represent the coefficients of the polynomials:

$$a(x) = 8$$

$$b(x) = 7$$

$$a(x) \times b(x) = 13$$

where each coefficient shown in decimal is a 4-bit value in the range (0 to 15).

Example analyzing by the MATLAB

Operations in  $GF(2^m)$  are handled directly by MATLAB functions found within the optional *Communications Toolbox*. For this  $GF(16)$ , it is necessary to specify the power  $2^m$  ( $m=4$ ) and to specify  $P(x) = x^4+x+1$  as given by binary (10011) = decimal 19. If only  $m$  is provided, a default primitive polynomial is supplied. If  $m$  is also omitted,  $GF(2)$  is assumed.

```
>> a = gf( [8], 4, 19 );
```

```
>> b = gf( [7], 4, 19 );
```

```
>> c = conv(a,b) % convolution is equivalent to multiplication
```

```
c = GF(2^4) array. Primitive polynomial = D^4+D+1 (19 decimal)
```

```
Array elements = 13
```

×	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	2	4	6	8	10	12	14	3	1	7	5	11	9	15	13
3	3	6	5	12	15	10	9	11	8	13	14	7	4	1	2
4	4	8	12	3	7	11	15	6	2	14	10	5	1	13	9
5	5	10	15	7	2	13	8	14	11	4	1	9	12	3	6
6	6	12	10	11	13	7	1	5	3	9	15	14	8	2	4
7	7	14	9	15	8	1	6	13	10	3	4	2	5	12	11
8	8	3	11	6	14	5	13	12	4	15	7	10	2	9	1
9	9	1	8	2	11	3	10	4	13	5	12	6	15	7	14
10	10	7	13	14	4	9	3	15	5	8	2	1	11	6	12
11	11	5	14	10	1	15	4	7	12	2	9	13	6	8	3
12	12	11	7	5	9	14	2	10	6	1	13	15	3	4	8
13	13	9	4	1	12	8	5	2	15	11	6	3	14	10	7
14	14	15	1	13	3	2	12	9	7	6	8	4	10	11	5
15	15	13	2	9	6	4	11	1	14	12	3	8	7	5	10

Fig 4: 4 Bit multiplication table for the polynomial equation  $p(x)=x^4+x+1$

**3.5. Instruction 5- Computation of Binary addition**

Addition of a and b in binary field is performed by the exclusive OR gate Result = a XOR b. Values in GF(2<sup>4</sup>) are 4-bits each, spanning the decimal range [0..15]. Addition takes place on these 4-bit binary values using bitwise XOR.

Result = a XOR b.

**3.6 Instruction 6 – Binary Extension field GF(Double)**

Same point value is added. Result will produce the double the value of present point in the graph .

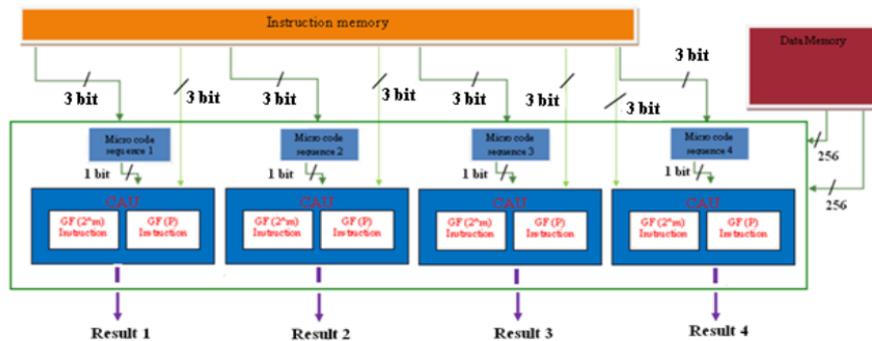
Result = a + a

**Table 1: Instruction Table:**

Sl.no	Instruction	Operation
1.	Interleaved multiplication based on Montgomery	Result = a.b mod p -- Galois field(P)
2.	Addition in prime field	Result = a + b mod p -- Galois field(P)
3.	Subtraction in prime field	Result = a – b mod p -- Galois field(P)
4.	GF_multiplication	Result = a * b -- Galois field(2 <sup>m</sup> )
5.	GF Addition	Result = a ^ b -- Galois field(2 <sup>m</sup> )
6.	GF_Double	Result = (r3,r4) + (r3,r4) -- Galois field(2 <sup>m</sup> )

**IV. Execution Of Parallel Instructions.**

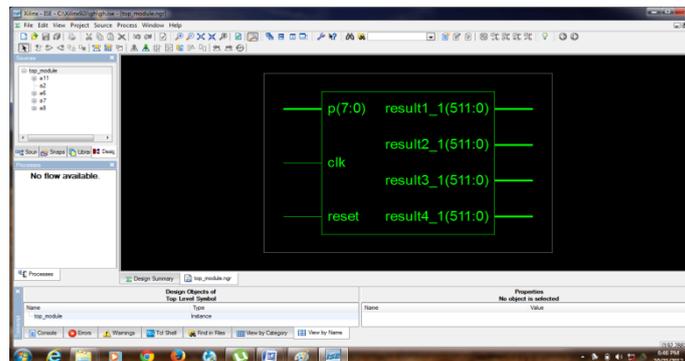
The Fig5 shows the quad core structure execution unit block diagram. The mechanism and regularity of data access for computing all instructions are fairly simple. The instructions are supplied by the instruction memory. Data are supplied by the data memory. The microcode sequence unit checks the incoming instruction is a valid one or not. If the instruction is a valid instruction then only microcode sequence unit enable the execution unit. The execution unit is named as configurable arithmetic unit (CAU). The CAU unit contains the both blocks of Prime field instruction execution unit and Binary extension field execution unit. The data accesses and instruction sequences are hard coded into the *sequence control* of the architecture which avoids the additional software development costs. The quad core contains the 4 blocks of CAU unit, each one will execute any one of prime field or binary extension field instructions and produce the result.



**Fig 5 Quad core structure block diagram.**

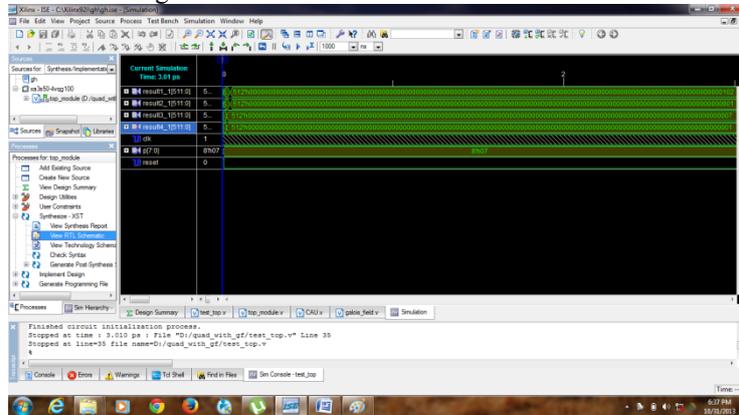
**V. Implementation Results**

The whole design has been done in verilog on Xilinx ISE design suit using a Spartran 3E, virtex4 and Virtex5. Performances are compared between the architecture. The synthesis output of the Xilinx ISE is shown in the Fig 6



**Fig 6: Synthesis output of the design**

Various combination of instruction execution in parallel is verified. The instructions selection and input a and b value loading is inbuilt in the coding. Example execution for Prime field addition, subtraction and binary extension field multiplication and double for the same input value of a=5h,b=4h, and p=7h is executed and the simulation output result is shown in the Fig 7. The output for the prime field addition is 102h, subtraction is 1h, Binary extension field multiplication is 7h, and binary extension field double is 1h. This simulation output is shown in the Fig 7.



**Fig 7 Simulation waveform for Parallel execution of four instructions.**

The complete design is implemented in Spartan 3E, Virtex 4 and Virtex 5. The comparison is shown in the table 2.

**Table 2: Comparison Table:**

Sl.no	Spartan 3E	Virtex 4	Virtex 5
Area	847%	135%	30%
Delay	32.503ns	17.063ns	12.418ns
Maximum frequency	30.766 MHz	58.608 MHz	80.527 MHz

### VI. Conclusion

The quad core structure executes the parallelism with both fields of instructions. At the same time 4 number of instructions are executing, this parallelism action will increase the speed. This architecture is an example design of crypto processor for executing prime field and binary field operations. This work has been further developed for the superscalar architecture with more number of instructions.

### References

- [1] Johannes Wolkerstorfer, "Dual-Field Arithmetic Unit for  $GF(p)$  and  $GF(2^m)$ " Institute for Applied Information Processing and Communications, Graz University of Technology, Inffeldgasse 16a, 8010 Graz, Austria. This work origin from the European Commission funded project *USB CRYPT* established under contract IST-2000-25169 in the Information Society Technologies (IST) Program.
- [2] Jun-Hong Chen, Ming-Der Shieh, Member, IEEE, and Wen-Ching Lin, "A High-Performance Unified-Field Reconfigurable Cryptographic Processor" IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 18, NO. 8, AUGUST 2010.
- [3] Michael Grand, Lilian Bossuet, Guy Gogniat, Bertrand Le Gal, Jean-Philippe Delahaye and Dominique Dallet, "A Reconfigurable Multi-core Cryptoprocessor for Multi-channel Communication Systems", published in "IPDPS - 25th IEEE International Parallel & Distributed Processing Symposium, Anchorage : United States (2011)"
- [4] Santosh Ghosh, Debdeep Mukhopadhyay, and Dipanwita Roychowdhury "Secure Dual-Core Cryptoprocessor for Pairings Over Barreto-Naehrig Curves on FPGA Platform" IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS,
- [5] P.C. Kocher, "Timing attacks on implementations of diffie-hellman, RSA, DSS and other systems," in *Adv. Cryptology-CRYPTO '96, LNCS 1109*, 1996, pp.104-113.
- [6] D.Kammler, D.Zhang, P.Schwabe, H. charwaechter, M. Langenberg, D. Auras, G. Ascheid, and R. Mathar, "Designing an ASIP for cryptographic pairings over Barreto-Naehrig Curves," *CHES '09, LNCS 5747*, pp. 254-271, 2009.
- [7] J. Fan, F. Vercauteren, and I. Verbauwhede, "Faster  $F_p$ -arithmetic for Cryptographic pairings on Barreto-Naehrig curves," *CHES '09, LNCS 5747*, pp.240-253, 2009.
- [8] D. N. Amanor, C. Paar, J. Pelzl, V. Bunimov, and M. Schimmler, "Efficient hardware architectures for modular multiplication on FPGAs," in *proc. Int. Conf. Field Program. Logic Appl.* 2005, pp.539-542.
- [9] V. Bunimov and M. Schimmler, "Area and time efficient modular multiplication of large integers," in *Proc. ASAP*, 2003, pp.400-409.
- [10] Huapeng Wu, Member IEEE, "Bit-Parallel Finite Field Multiplier and Squarer Using Polynomial Basis", IEEE TRANSACTIONS ON COMPUTERS, VOL. 51, NO. 7, JULY 2002