Volume 4, Issue 2, February 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

FPGA-based Hardware Architecture of Elgamal Encryption using Carry Save Adder

Vivek Kumar, Ashish Joshi, Banit Negi
CSE-Department, THDC-IHET
India

Abstract— In this paper an efficient implementation of the Elgamal Encryption algorithm on an FPGA is presented. The main purpose of the implementation is the time-cost reduction which is achieved by using efficient modular multiplication algorithm of Montgomery in conjunction with Carry Save Adder. The paper describes the underlying architecture of the implemented design and shows the results obtained.

Keywords— FPGA, Montgomery multiplication, Carry Save Adder, Crypto-accelerator, Digital Signature

I. INTRODUCTION

Public key or asymmetric key security algorithms are widely used methods for digital signatures and forms an integral part of the authentication scheme. Various software approaches have been applied to implement the algorithms but, because the algorithms are computationally intensive (where the order of key may be upto 1024 bits), it demands the use of a dedicated hardware to perform the computation efficiently. A crypto-accelerator is a dedicated hardware which performs the function of encryption as well as decryption, works independently and acts as a co-processor. Computation are now performed without the intervention of the primary CPU. FPGA are useful for implementing the prototype because they offer high performance hardware at a lower cost in comparison with Application specific Integrated Circuits (ASIC). In this paper Elgamal algorithm, which is a public key algorithm and is used for digital signature is implemented on an FPGA and is used for digital signatures. The algorithm was developed by Taher Elgamal in 1984. In the next chapter there is a brief theoretical introduction that shows the working of Elgamal algorithm [1], followed by a section describing the hardware implementation of the encryption algorithm. In the end result of the implementation on a re-programmable structure is presented.

II. THEORETICAL OVERVIEW

Elgamal is a public key cryptosystem based on the discrete logarithm [2] problem in contrast to RSA or Rabin which use prime factorization problem as there trapdoor functions. If p is a very large prime and e_1 is a primitive root in the group $G=< Z*_p,\times>$ with r as an integer, then $e_2=e^r{}_1 \bmod p$ is easy to compute using the fast exponential algorithm, but given e_2,e_1 and p, it is infeasible to calculate $r = \log e_1 e_2 \bmod p$, this is called discrete logarithm problem. Figure 2.1 shows key generation, encryption and decryption in Elgamal.
Elgamal cryptosystem uses two ciphertext C1 and C2, the Plain-text P is encrypted using equation (1) and (2)

$$C1 = e^r{}_1 \bmod p \tag{1}$$

$$C2 = (e^r{}_2 \times P) \bmod p \tag{2}$$

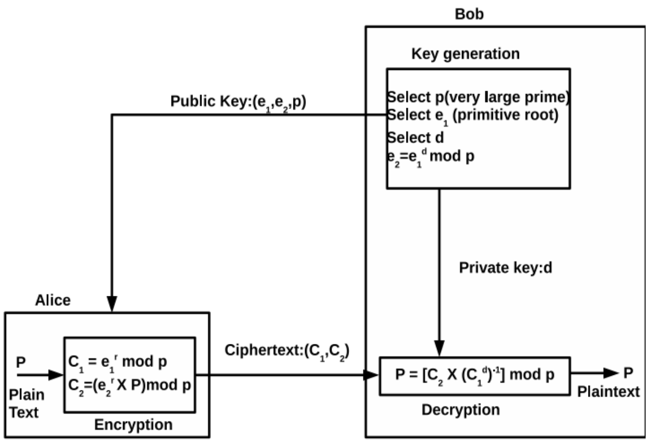

FIG 1. KEY GENERATION, ENCRYPTION, AND DECRYPTION IN ELGAMAL ALGORITHM

where e_1,e_2 and p are public key while r is a random integer in group $G = <Z^*_p, \times >$. The plain-text is obtained using equation (3).

$$P = [C_2 \times (C^d{}_1)^{-1}] \bmod p \tag{3}$$

The algorithm used for exponentiation is the right-to-left algorithm 1 where we can employ two parallel modular multipliers to achieve it. The parallelism is achieved as the modification of S and T variables in a single iteration is independent of each other. Unlike left-to-right modular exponentiation where only one variable S is modified in each iteration, here instead two variables are used.

```
INPUT: A,M and K are all n bit binary numbers.
OUTPUT S = $A^K$ mod M.
1: S←1 and T←A
2: for i=0 to n−1 do
3:         if $K_i$ = 1 then
4:             S←S.T mod M                  // multiplication
5:         end if
6:     T←$T^2$ mod M                     // squaring
7: end for
8: return S
```

Algorithm 1 Right-to-left modular exponentiation

For multiplication purpose Montgomery modular multiplication [4] algorithm 2 is used. The algorithm produces output, with an extra factor of 2^{-n}.

```
INPUT: X,Y< M< $2^n$, with $2^{n-1}$ < M< $2^n$ and M=2t+1,
with t ∈ N.
OUTPUT P = X.Y.$2^{-n}$ mod M.
n: number of bits in X.
$X_i$: ith bit in X
1: P←0
2: for i=0 to n−1 do
3:         P←P + $X_i$.Y
4:         if $P_i$=1 then
5:             P ← P + M
6:     end if
7: P←P/2
8: end for
9: if P≥M then
10: P ← P − M
11: end if
12: return P
```

Algorithm 2 Montgomery Modular Multiplication

III. HARDWARE IMPLEMENTATION

The Elgamal module implementation is based on the two major operation which are exponentiation and multiplication. Exponentiation module uses the multiplication module iteratively. Designing has been done using the top down approach which first describes the exponentiation and then the multiplication module.

The module which performs the Elgamal encryption is shown in Fig 2. e1, e2, p and P are 32 bit operands where e1, e2 and p are the public key and P is the plain-text to be encrypted. The module is synchronized with the use of clk while clear is use to reset the circuit. C1 and C2 are the output ports which hold the encrypted data which is available in the ports when the done signal is high. Figure 3 shows the exponentiation module which carries out the exponentiation using the multiplication sub modules. The algorithm used for exponentiation is the Right-to-left modular exponentiation (Algorithm 2). The Modular exponentiation module uses 2 montgomery multiplication modules. Input ports are p, m, r, e1, clk, clear and p1 where p holds the multiplication output from first montgomery module, p1 holds the multiplication output from the second Montgomery module, both of the module work in parallel, r is a random number and m is the modular no. In algorithm 2 in each iteration equation 4 is used.

$$T \leftarrow T.T(\text{mod } M) \tag{4}$$

The first montgomery module is employed to carry out this equation while equation 5 is used whenever k_i equals 1

$$S \leftarrow S.T (\text{mod } M) \tag{5}$$

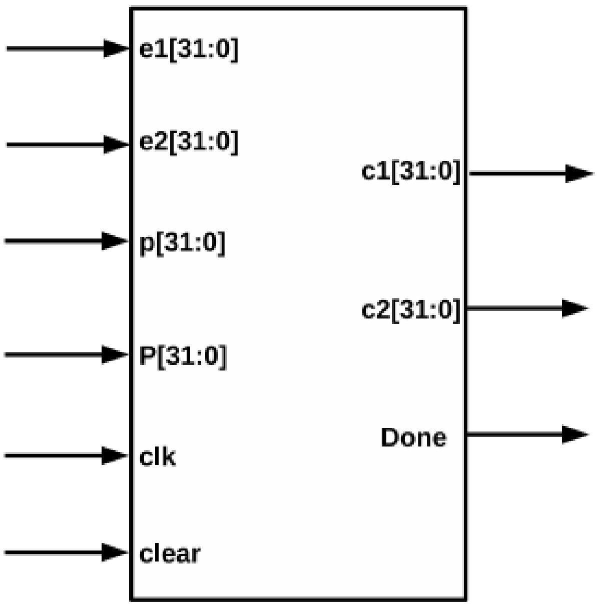

Fig. 1 Elgamal Module Entity

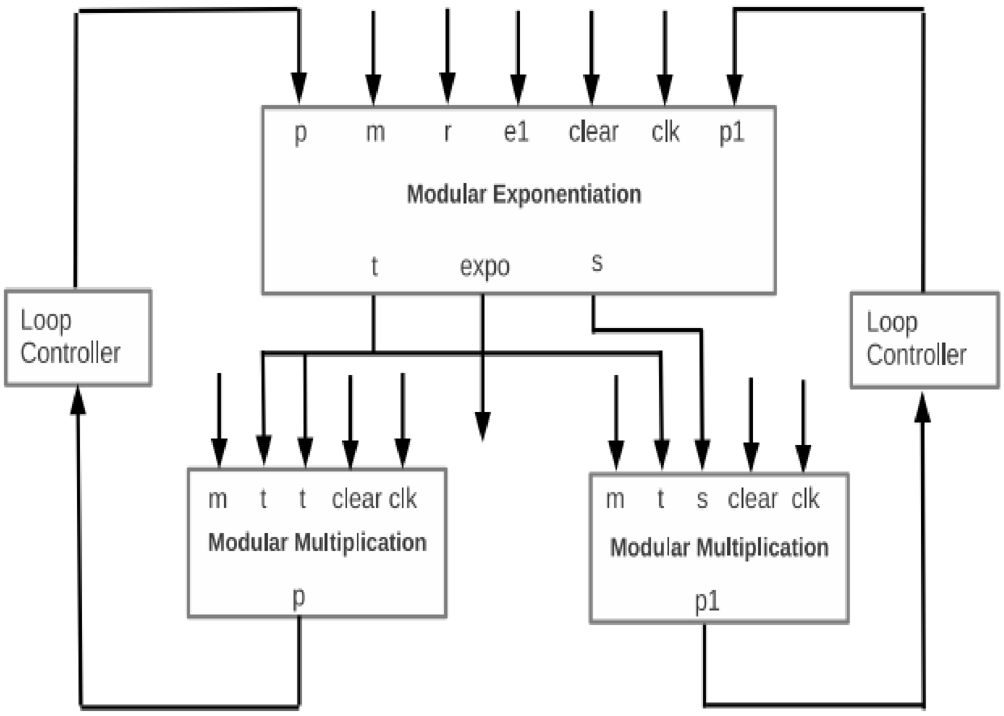

Fig. 3 Modular Exponentiation

This modular multiplication is carried out by the second montgomery module. The respective output p and p1 are feed back into the exponential module. Exponential output is obtained at the expo output port. This modular exponentiation module is used by the elgamal module which calculates equation 1 and equation 2.The module will be used twice for one encryption once for calculating e^r_1 mod p (C1) and once for calculating e^r_2 mod p, finally modular multiplication will be used for calculating multiplication of plaintext P and and output of e^r_2 mod p (C2). Figure 4 shows the architecture of montgomery modular multiplication. The algorithm is implemented by using one carry save adder(CSA)[5]. On the basis of x[i] either y or 0 is selected using mux 1, selected value is stored in ytemp. Ptemp is a temporary register which is initialize to 0. X-Ored value of ptemp[0] and ytemp[0] is fed as a selection line for mux 2 which selects either m or 0. The output of mux2 is stored in temporary register mtemp. ptemp, ytemp and mtemp is fed as input in a CSA.The output of the carry save adder is stored again in ptemp variable. The value of ptemp is divided by 2 which can be performed by shifting ptemp one position to right.

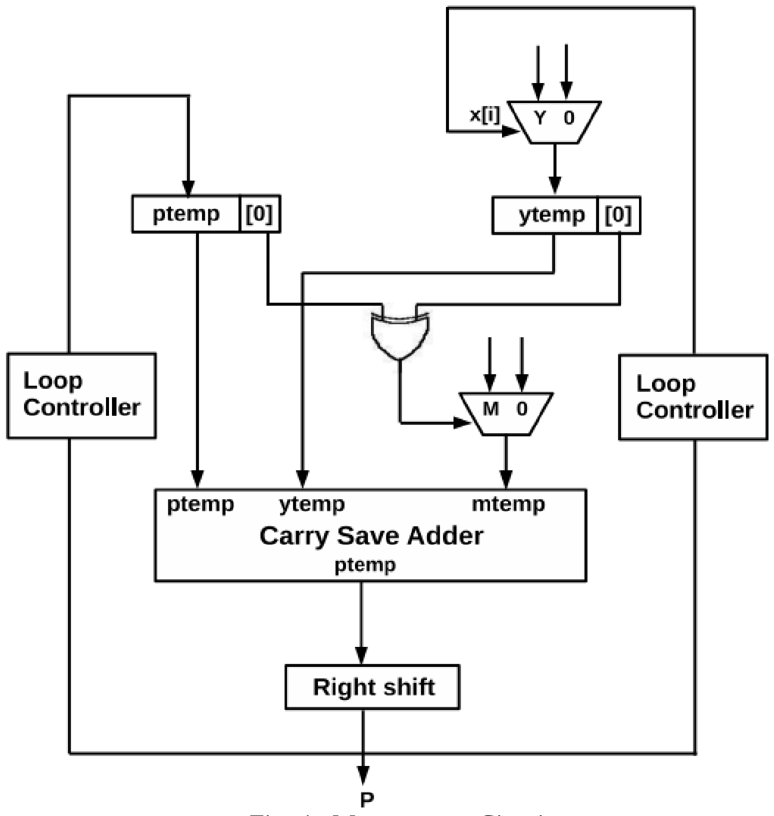

Fig. 4 : Montgomery Circuit

CSA is a type of digital adder which is used to add three or more n-bit number. CSA is better than Carry Propagation Adder(CPA) and Carry Look-ahead Adder(CLA). CSA generates sum and carry in a single clock pulse while in CPA, carry has to propagate through the entire addition, which increases delay. CLA reduces this delay but it is not too much helpful for very large integers. When dealing with 512-bit to 2048-bit number that are required in public- key cryptography, CLA is not of much help.

IV. RESULTS

After the implementation of Elgamal Encryption Algorithm on 3S400FG456 FPGA the following (table 1) device utilization summary was obtained for encryption of 32-bit message.

Table 1 : Device Utilization Summary of 32-bit for Elgamal encryption

Devices	Used	Available	Area(%)
Number of Slices	294	3584	8
Number of Slice Flip Flops	177	7168	2
Number of 4 Input LUTs	557	7168	7
Number of IOs	195		
Number of bounded IOBs	98	264	37
Number of GCLKs	1	8	12

Table 2 shows the timing summary for encrypting different message size, ranging from 32 to 256 bit of data.

Table 2: Device utilization Summary of 32-bit for Elgamal encryption

Bits	Min. Period	Min. Input arrival time	Max. output required time
32	16.801ns	14.421ns	6.216ns
64	20.567ns	18.187ns	6.216ns
128	27.841ns	25.461ns	6.216ns
256	42.978	40.598ns	6.216ns

V. CONCLUSION

In this paper, the Elgamal encryption technique is implemented on FPGA of type 3S400FG456.The design successful implemented the encryption using the carry save adder for montgomery multiplication. As is presented in previous section, physical resources of FPGA used to implement Elgamal encryption system are very small, compared with the FPGA capacity, allowing further development of algorithms with higher encryption capacity.

REFERENCES

[1] T. Elgamal,"A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, " Lecture Notes in Computer Science, vol.196, pp. 10-18 , 1985.

[2] K.S. McCurley,"The Discrete Logarithm Problem, " Proceedings of Symposia in Applied Mathematics, vol. 42, 1990.

[3] J.A.Menezes, C.P. Oorschot, and A.S Vanstone, Hand- book of Applied Cryptography,CRC Press, 1997.

[4] P. Montgomery,"Modular Multiplication without Trial Division, " Mathematics of Computation, vol. 44,pp. 519-521, 1985.

[5] D. Narh Amanor,C. Paar, J. Pelzl, V. Bunimov and M. Schimmler ,"Efficient hardware architectures for modular multiplication on FPGAs",Field Programmable Logic and Applications,pp. 539 - 542, 2005.