Efficient Word-Level Sequential Normal Basis Multiplier over $GF(2^m)$

Yong Suk Cho[1], Jae Yeon Choi[2,*]

[1] Department of Information & Communication Security, Young Dong University, Chung Buk, 370-701, Korea
yscho@yd.ac.kr
[2] Department of Information & Communication, Nam Seoul University, Chung Nam, 331-707, Korea
cjy@nsu.ac.kr

Abstract. In this paper, efficientword-levelsequential finite field multiplier using normal basis over $GF(2^m)$is presented. The proposed architecture takes $w$clock cycles to compute theproduct bits, where the value for$w$,$1 \leq w \leq m$, can be arbitrarily selected by thedesigner to set the trade-off between area and speed.The proposed architecture has significantlylower complexity and critical path delay in comparison to the previously proposedarchitectures.

Keywords: Finite fieldmultiplier, Normal basis, Cryptography

1Introduction

Finite fields are the most commonly used arithmetical structures in cryptographyand coding. Many algorithms in cryptographic and coding applicationsare defined in terms of finite field arithmetic operations [1], [2]. The elliptic curvecryptosystemsand the Diffie-Hellman key exchange algorithm are importantexamples of such cryptographic applications [3]. Also, common error controlcodes such as Reed-Solomon and BCH codes are based on finite field theory [4].

Addition and multiplication are two basic operations in thefinite field $GF(2^m)$. Addition in $GF(2^m)$is easily realizedusing m two-input XOR gates while multiplication is costlyin terms of gate count and time delay. The other operationsof finite fields, such as exponentiation, division, andinversion can be performed by repeated multiplications. As a result, there is a need to have fastmultiplication architecture with low complexity.

One important factor that could greatly affect computationperformance is the basis in which finite field elements arerepresented. The most commonly used bases include polynomialbasis (PB)or standard basisand normal basis (NB). PBprobably is the most popular basis and is efficient in both softwareand hardware implementations. On the other hand, NB has adistinct feature that its squaring operation in a binary field is free.Therefore, NB is especially attractive when performingexponentiation,where squaring operations are extensively required.

A word levelmultipliertakes w clock cycles, $1 \leq w \leq m$, to finish onemultiplication operation in a binary field of size m. The valueof w can be selected by designer to set the trade off betweenarea and speed. Decreasing the value

*Corresponding Author

of w will result in fasterand larger multipliers while increasing w will make smallerand slower multipliers.

In this paper,efficient word-levelsequentialnormal basis multiplier over *GF*(2^m) is presented. The proposed multiplier divides the data into the same length of word, uses bit-serial multiplier inside the word, and processes the multiplications using word-parallel method.The proposed architecture has significantlylower complexity and critical path delay in comparison to the previously proposedarchitectures [5], [6].

The organization of this paper is as follows: Normal basis and bit-serial multiplication using this basis [7] are brieflyreviewed in Section 2. In Section 3, efficient word-levelsequential multiplierusing normal basis is proposed.Finally some concluding remarks aregiven in Section 4.

2 The Bit-Serial Normal Basis Multiplier over *GF*(2^m)

Let field elements $A, B \in GF(2^m)$ be represented w.r.t. the normal basis N as

$$A = a_{m-1}\beta^{2^{m-1}} + a_{m-2}\beta^{2^{m-2}} + \cdots\cdots + a_0\beta,\ \ a_i \in GF(2), \tag{1}$$

$$B = b_{m-1}\beta^{2^{m-1}} + b_{m-2}\beta^{2^{m-2}} + \cdots\cdots + b_0\beta,\ \ b_i \in GF(2), \tag{2}$$

respectively. Then the product A and B can be given by

$$\begin{aligned} C = A \cdot B &= A \cdot \left(b_{m-1}\beta^{2^{m-1}} + b_{m-2}\beta^{2^{m-2}} + \cdots\cdots + b_0\beta\right) \\ &= b_{m-1}\left(A\beta^{2^{m-1}}\right) + b_{m-2}\left(A\beta^{2^{m-2}}\right) + \cdots\cdots + b_0(A\beta) \\ &= b_{m-1}\left(A^{1/2^{m-1}}\beta\right)^{2^{m-1}} + b_{m-2}\left(A^{1/2^{m-2}}\beta\right)^{2^{m-2}} + \cdots\cdots + b_0(A\beta). \end{aligned} \tag{3}$$

Also, (3) can be rewritten like the following

$$C = \left[\cdots\left[\left[b_{m-1}A^{1/2^{m-1}}\beta\right]^2 + b_{m-2}A^{1/2^{m-2}}\beta\right]^2 \cdots\cdots\right]^2 + b_0A\beta. \tag{4}$$

In order to realize (4), the structure of Fig. 1 is proposed. In the Fig. 1, the thick line is the$m$-bit bus, ▣is m-bit register,is⊕2-input XOR gates,is$m$ 2-i⊙ut AND gates,is a constant m$\textcircled{\scriptsize β}$tiplier that multiplies the element β of *GF*(2^m). If the register $U$and$V$are initialized with two input elements respectively and the register $Z$is cleared, then we can verify that after the$m$-thclock cycle the register$Z$ contains the product $C = AB$.

3 The word-level sequential normal basis multiplier

The bit-serial normal basis multiplier like the Fig. 1 requires m clock cycles to perform one operation. To speed up the process, one element in *GF*(2^m), B is divided into$d$ $(= \lceil m/w \rceil)$ words of w bits as follows:

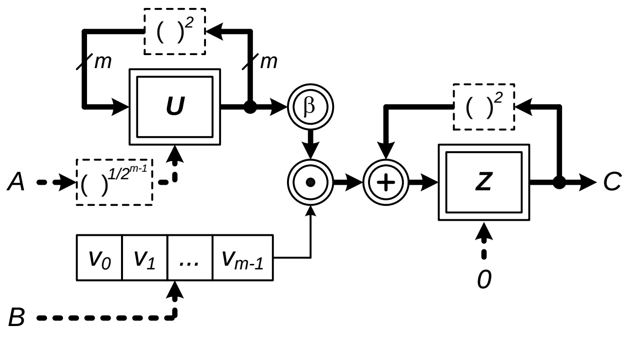

Fig. 1. $GF(2^m)$ bit-serial normal basis multiplier

$$\begin{aligned} C = {} & A\cdot\left(b_{m-1}\beta^{2^{m-1}} + b_{m-2}\beta^{2^{m-2}} + \cdots\cdots + b_{m-w}\beta^{2^{m-w}}\right) \\ & + A\cdot\left(b_{m-w-1}\beta^{2^{m-w-1}} + b_{m-w-2}\beta^{2^{m-w-2}} + \cdots\cdots + b_{m-2w}\beta^{2^{m-2w}}\right) \\ & + \cdots\cdots\cdots \\ & + A\cdot\left(b_{m-(d-1)w-1}\beta^{2^{m-(d-1)w-1}} + \cdots\cdots + b_1\beta^2 + b_0\beta\right) \end{aligned} \tag{5}$$

Then, (5) can be written as

$$\begin{aligned} C = {} & \left[\cdots\left[\left[b_{m-1}A^{1/2^{w-1}}\beta^{2^{m-w}}\right]^2 + b_{m-2}A^{1/2^{w-2}}\beta^{2^{m-w}}\right]^2\cdots\right]^2 + b_{m-w}A\beta^{2^{m-w}} \\ & + \left[\cdots\left[\left[b_{m-w-1}A^{1/2^{w-1}}\beta^{2^{m-2w}}\right]^2 + \cdots\right]^2\cdots\cdots\right]^2 + b_{m-2w}A\beta^{2^{m-2w}} \\ & + \cdots\cdots\cdots \\ & + \left[\cdots\left[\left[b_{m-(d-1)w-1}A^{1/2^{w-1}}\beta^{2^{m-dw}}\right]^2 + \cdots\right]^2\cdots\cdots\right]^2 + b_0A\beta \end{aligned} \tag{6}$$

Based on (6), we now present aword-levelsequentialnormal basis multiplier structure. The architecture is shown in Fig. 2.

4 Conclusion

In this paper, efficient word-levelsequentialnormal basis multiplier is proposed. Unlike the bit-serial multipliers which require$m$clock cycles for the latency, the proposed multiplier has the latency of $w\,(1 \le w \le m)$ clock cycle. The word-levelsequential multiplier is faster than bit-serial multipliers and has lower hardware area complexity than bit-parallel multipliers. Therefore, the most significant feature of the proposed multiplier is a proper trade-off between hardware complexity and delay time.

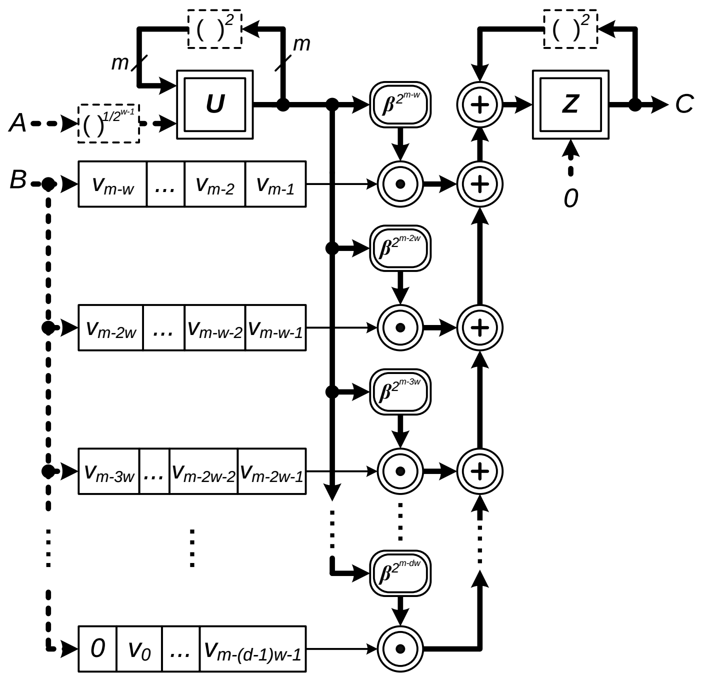
( )²
m
m
U
Z
C
( )²
A
$(\ )^{1/2^{w-1}}$
B
$\beta^{2^{m-w}}$
v_{m-w} … v_{m-2} v_{m-1}
0
$\beta^{2^{m-2w}}$
v_{m-2w} … v_{m-w-2} v_{m-w-1}
$\beta^{2^{m-3w}}$
v_{m-3w} … v_{m-2w-2} v_{m-2w-1}
$\beta^{2^{m-dw}}$
0 v_0 … $v_{m-(d-1)w-1}$

Fig. 2. The proposed word-levelsequentialnormal basis multiplier over*GF*(2^m)

References

1. Man Young Rhee,: Error-Correcting Coding Theory, McGraw-Hill, (1989)
2. R. Lidl and H. Niederreiter,: Introduction to Finite Fields and Their Applications, Cambridge Univ. Press, (1994)
3. D. Hankerson, A. Menezes, and S. Vanstone,: Guide to Elliptic Curve Cryptography, Springer-Verlag, (2004)
4. S. B. Wicker and V. K. Bhargava, editors,: Reed-Solomon Codes and TheirApplications, IEEE Press, New York, NY, (1994)
5. A. Reyhani-Masoleh and M.A. Hasan,:Low Complexity Word-Level Sequential Normal Basis Multipliers, IEEE Transactions on Computers, Vol.54, No.2, pp.98-110, February(2005)
6. A.H. Namin, H. Wu, and M. Ahmadi,: Comb Architectures forFinite Field Multiplication in F_{2^m}, IEEE Transactions on Computers, Vol.56, No.7, pp.909-916, July (2007)
7. T. Beth and D. Gollmann,: Algorithm Engineering for Public KeyAlgorithms, IEEE Journal on Selected Areas in Communications, Vol.7, No.4, pp.458-466, May (1989)