T.R.
GEBZE TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

VERSATILE DIGIT SERIAL MULTIPLIERS FOR BINARY EXTENSION FIELDS

BİLAL USLU
A THESIS SUBMITTED FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
DEPARTMENT OF ELECTRONICS ENGINEERING

GEBZE
2016

T.R.

GEBZE TECHNICAL UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

VERSATILE DIGIT SERIAL MULTIPLIERS FOR BINARY EXTENSION FIELDS

BİLAL USLU

A THESIS SUBMITTED FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRONICS ENGINEERING

THESIS SUPERVISOR

ASSIST. PROF. DR. SERDAR SÜER ERDEM

GEBZE

2016

T.C.
GEBZE TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İKİLİ SONLU GENİŞLEME CİSMİ İÇİN ESNEK VE ÖLÇEKLENEBİLİR DİGİT SERİ ÇARPICILAR

BİLAL USLU

DOKTORA TEZİ
ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

DANIŞMANI
YRD. DOÇ. DR. SERDAR SÜER ERDEM

GEBZE
2016

GEBZE TEKNİK ÜNİVERSİTESİ	**DOKTORA JÜRİ ONAY FORMU**

GTÜ Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 14/12/2015 tarih ve 2015/74 sayılı kararıyla oluşturulan jüri tarafından 25/12/2015 tarihinde tez savunma sınavı yapılan Bilal Uslu' nun tez çalışması Elektronik Mühendisliği Anabilim Dalında DOKTORA tezi olarak kabul edilmiştir.

JÜRİ

ÜYE (TEZ DANIŞMANI)	: Yrd. Doç. Dr. Serdar Süer Erdem	
ÜYE	: Yrd. Doç. Dr. Köksal Hocaoğlu	
ÜYE	: Doç. Dr. Ali Tangel	
ÜYE	: Yrd. Doç. Dr. Önder Şuvak	
ÜYE	: Yrd. Doç. Dr. Suhap Şahin	

ONAY

Gebze Teknik Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun

...../...../..... tarih ve ...../..... sayılı kararı.

İMZA/MÜHÜR

SUMMARY

This thesis investigates the digit serial polynomial basis multipliers performing multiplication in multiple binary extension fields $\mathbb{F}_{2^{m_1}}, \mathbb{F}_{2^{m_2}}, \ldots, \mathbb{F}_{2^{m_\lambda}}$. Designing such versatile multipliers encounters a number of difficulties. First of all, the element sizes of the supported fields are different from each other, and thus the elements are represented with different number of bits for each field. To deal with different sized elements, designs with left or right justified operands are investigated in this study. Secondly, each field multiplication involves modular reduction with a different irreducible polynomial, and thus the complexity can increase rapidly with the number of supported fields λ. To prevent this, two methods are studied: Using sparse irreducible polynomials and unifying the modular reduction computation of the fields by choosing the irreducible polynomials suitably. The thesis shows that multiple fields can be supported at the cost of an $\mathcal{O}(\lambda)$ increase in area and an $\mathcal{O}(\sqrt{\lambda})$ increase in time.

Key Words: Binary extension fields, digit serial multiplier, polynomial basis, elliptic curve cryptography.

ÖZET

Bu tez çoklu $\mathbb{F}_{2^{m_1}}, \mathbb{F}_{2^{m_2}}, \ldots, \mathbb{F}_{2^{m_\lambda}}$ ikili alan uzantılarında çalışan polinom bazlı dijit seri çarpıcıları incelemektedir. Bu tür esnek ve ölçeklenebilir çarpıcıların tasarımında çeşitli sıkıntılarla karşılaşılmaktadır. Desteklenen alanların eleman uzunlukları birbirinden farklı olduğundan, bu elemanlar birbirinden farklı bit sayıları ile temsil edilmektedir. Bu tez çalışmasında ilk olarak, birbirinden farklı uzunluktaki elemanlar ile çalışabilmek için sola ve sağa dayalı terimlere sahip tasarımlar araştırıldı. Ayrıca her bir çarpma alanı, birbirinden farklı indirgenemez polinomlar kullanan modüler indirgeme işlemleri içermektedir. Bu yüzden desteklenen alan sayısı λ ile karmaşıklık seviyesi çok fazla miktarda artabilmektedir. Daha sonra, bu durumu engellemek için iki yöntem üzerinde çalışıldı: Az terimli indirgenemez polinomların kullanımının yanısıra her bir alanın modüler indirgeme hesap işlemlerinin birleştirilmesi için en uygun indirgenemez polinomlar seçildi. Bu tez çoklu alanların kullanımının donanım maliyeti $\mathcal{O}(\lambda)$ ile ve hesaplama zaman maliyeti olarak $\mathcal{O}(\sqrt{\lambda})$ ile arttığını göstermektedir.

Anahtar Kelimeler: İkili alan uzantıları, dijit seri çarpıcılar, polinomsal baz, eliptik eğri şifrelemesi.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Asst. Prof. Dr. Serdar Süer Erdem for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank my family: my parents and to my wife and kids for supporting me spiritually throughout writing this thesis and my my life in general.

TABLE of CONTENTS

LIST of ABBREVIATIONS and ACRONYMS

Abbreviations and Acronyms		**Explanations**
ASIC	:	Application Specific Integrated Circuit
FPGA	:	Field Programmable Gate Array
LSD	:	Least Significant Digit
MSD	:	Most Significant Digit
NIST	:	National Institute of Standards and Technology

LIST of FIGURES

LIST of TABLES

1. INTRODUCTION

The arithmetic of the binary extension field $\mathbb{F}_{2^m}$ is commonly used in cryptography and coding theory applications [1], [2]. Important cryptographic applications such as elliptic curve cryptography need large number of field multiplications in $\mathbb{F}_{2^m}$ to perform cryptographic transformations [3], [4]. Also, these applications use quite large fields. For example, the field size m is typically selected from the range $160 \leq m \leq 1000$ in elliptic curve cryptography. Thus, efficient hardware and software implementations of the field multiplication is crucial to reduce the cost of cryptographic systems.

In this thesis, digit serial $\mathbb{F}_{2^m}$ multiplier architectures supporting multiple field sizes m are presented. The proposed architectures support a set of field sizes $m_1, m_2, \ldots, m_\lambda$, i.e. work in one of the fields $\mathbb{F}_{2^{m_1}}, \mathbb{F}_{2^{m_2}}, \ldots, \mathbb{F}_{2^{m_\lambda}}$ according to the user selection. The increase in the cost due to supporting multiple fields are also anlayzed. Moreover, as a case study, The multipliers supporting the five NIST fields $\mathbb{F}_{2^{163}}$, $\mathbb{F}_{2^{233}}$, $\mathbb{F}_{2^{283}}$, $\mathbb{F}_{2^{409}}$, and $\mathbb{F}_{2^{571}}$, recommended for elliptic curve cryptography are investigated [5], [6].

The proposed digit serial multipliers use polynomial basis. That is, $\mathbb{F}_{2^m}$ elements are represented by the binary polynomials of degree less than $m \left\{ a(x) = \sum_{i=0}^{m-1} a_i x^i \,\middle|\, a_i \in \mathbb{F}_2 \right\}$ where x represents the root of some degree m irreducible polynomial $x^m + g(x)$ called generator. The product $f(x)$ of any two elements $a(x)$ and $b(x)$ is computed by the polynomial multiplication $a(x)b(x)$ modulo the generator polynomial $x^m + g(x)$. The digit serial multipliers keep the coefficients of each polynomial operand in an array of digits. One of the operands is multiplied by the digits of the other operand, one digit at a time. Each of these partial product computations is interleaved with a modular reduction step and an accumulation step.

There are several works studying polynomial basis digit serial multipliers in the literature [7] - [13] but these multipliers have been developed to support a particular field size m. The work in [7] introduces efficient digit serial multipliers using polynomial basis. The work in [11] investigates optimum digit sizes and the effects of using multiple accumulators. The work in [12] proposes using T flip flops in the accumulators instead of D flip flops to reduce the area complexity.

The versatility is an important feature for hardware designs since the ASIC circuits cannot be altered after fabrication. Flexible implementations are possible with reconfigurable FPGA devices but FPGAs cannot compete with the ASIC in terms of performance, cost and power consumption. Naturally, customizable elliptic curve systems [14] and versatile multipliers [15] - [19] have been proposed in the literature but the proposed multipliers are all bit serial and there is not much work about versatile digit serial multipliers.

Let the supported field sizes be $m_1 < m_2 < \cdots < m_\lambda$. Then, the number of the bits used in the representation must be large enough to represent m_λ coefficients. When a field with smaller size $m_k < m_\lambda$ is selected, the elements

$$a(x) = \sum_{i=0}^{m_k-1} a_i x^i \in \mathbb{F}_{2^{m_k}} \tag{1.1}$$

must be justified either to the right or to the left. When $a(x)$ is right justified, the leading $m_\lambda - m_k$ coefficients are set to zero as follows:

$$a(x) = 0x^{m_\lambda-1} + \cdots + 0x^{m_k} + a(x)\ . \tag{1.2}$$

In left justified representation, $a(x)$ is shifted, then the trailing $m_\lambda - m_k$ coefficients are set to zero as follows:

$$\hat{a}(x) = x^{m_\lambda-m_k}a(x) + 0x^{m_\lambda-m_k-1} + \cdots + 0x + 0\ . \tag{1.3}$$

Also, the fields $\mathbb{F}_{2^{m_1}}, \mathbb{F}_{2^{m_2}}, \ldots, \mathbb{F}_{2^{m_\lambda}}$ supported by the multipliers use different generator polynomials $x^{m_k} + g^{(k)}(x)$ for $k = 1,2,\ldots,\lambda$. Naturally, the modular reduction by each $x^{m_k} + g^{(k)}(x)$ can be performed with a separate circuit. Nevertheless, the reduction circuits can be unified when the generator polynomials $x^{m_k} + g^{(k)}(x)$ are appropriately chosen.

The thesis is organized as follows: Section 2 introduces the basic polynomial basis digit serial multipliers supporting a single field size and presents a detailed analysis of their complexities. Section 3 proposes digit serial multiplier architectures supporting multiple field sizes. Some of them work with right justified operands and

some of them work with left justified operands. Also, some of these employ separate reduction circuits and some of them employ unified reduction circuits. The complexities of the proposed multipliers are studied in Section 4. Discussion of our results are presented in Section 5.

2. DIGIT SERIAL MULTIPLIERS USING POLYNOMIAL BASIS

Naturally, m bits are sufficient to represent an element

$$a(x) = \sum_{i=0}^{m-1} a_i x^i \tag{2.1}$$

in the field $\mathbb{F}_{2^m}$. This is because each binary valued coefficient of $a(x)$ can be stored in a bit. Let the hardware digit size be w bits. A digit serial multiplier divides one of the operands, say $b(x)$, into $\lceil m/w \rceil$ digits as follows:

$$b(x) = \sum_{i=0}^{m-1} b_i x^i = \sum_{j=0}^{\lceil m/w \rceil - 1} B_j x^{wj} \tag{2.2}$$

where

$$B_j = \sum_{i=0}^{w-1} b_{wj+i} x^i \tag{2.3}$$

is the jth digit of $b(x)$ and holds its consecutive w coefficients. Then,

$$a(x)b(x) \bmod \left(x^m + g(x)\right) = \sum_{j=0}^{\lceil m/w \rceil - 1} a(x)B_j x^{wj} \bmod \left(x^m + g(x)\right) \tag{2.4}$$

gives the field product. The multiplier computes this product, accumulating the partial products of the digits B_j in $\lceil m/w \rceil$ iterations. Thus, the multiplier gets faster as the digit size w increases. However, its area also increases because of the digits B_j, and thus their partial products get larger. In each iteration, one of the partial products $a(x)B_j$ is computed and added to an intermediate result $I(x)$ where $deg\big(I(x)\big) < m + w$. The intermediate result $I(x)$ can be split into degree $w - 1$ and $m - 1$ polynomials as follows:

$$I(x) = \sum_{i=0}^{m+w-1} I_i x^i = q(x)x^m + r(x) \tag{2.5}$$

where $q(x) = \sum_{i=0}^{w-1} I_{m+i} x^i$ and $r(x) = \sum_{i=0}^{m-1} I_i x^i$. The generator $x^m + g(x)$ is usually chosen such that

$$deg\big(g(x)\big) = \mu \leq m - w \tag{2.6}$$

for fast modular reduction. Then, the intermediate result can be reduced without division as follows:

$$\begin{aligned} I(x)\ mod\ \big(x^m + g(x)\big) &= q(x)x^m + r(x)\ mod\ \big(x^m + g(x)\big) \\ &= g(x)q(x) + r(x)\ . \end{aligned} \tag{2.7}$$

This can be done because $deg\big(g(x)q(x)\big) < m$ for any degree $w-1$ polynomial $q(x)$ when Equation (2.6) holds.

2.1. Most Significant Digit First Multipliers

The most significant digit first multipliers compute the field product in (2.4), starting from the most significant digit B_{M-1} as illustrated in the Table 2.1.

Table 2.1: MSD first field multiplication.

Inputs:	$a(x) = \sum_{i=0}^{m-1} a_i x^i$, $g(x) = \sum_{i=0}^{\mu} g_i x^i$
	$b(x) = \sum_{i=0}^{m-1} b_i x^i = \sum_{j=0}^{\lceil m/w \rceil - 1} B_j x^{wj}$ where $B_j = \sum_{i=0}^{w-1} b_{wj+i} x^i$
Output:	$f(x) = a(x)b(x)\ mod\ \big(x^m + g(x)\big)$
1:	$I(x) = 0$
2:	**for** $j = \lceil m/w \rceil - 1$ to 0
3:	$f(x) = I(x)\ mod\ \big(x^m + g(x)\big)$, $I(x) = a(x)B_j + x^w f(x)$
4:	$f(x) = I(x)\ mod\ \big(x^m + g(x)\big)$

In the Table 2.1, $I(x)$ is the accumulated sum of the partial products. $I(x)$ is initialized to zero in the beginning and reduced to modulo $x^m + g(x)$ in each iteration. The reduced result

$$f(x) \equiv \sum_{i=j+1}^{\lceil m/w \rceil - 1} a(x) B_i x^{w(i-j-1)} mod\big(x^m + g(x)\big) \tag{2.8}$$

$x^w f(x)$ is added with the partial product $a(x)B_j$ to update $I(x)$ in the jth iteration. After all partial products are added, $I(x)$ is reduced one last time. Figure 2.1 and Figure 2.2 illustrate MSD first multiplication. Here, it is chosen that

$$deg\big(g(x)\big) = \mu \leq m - w\ . \tag{2.9}$$

Thus, the reduction $I(x)\ mod\ \big(x^m + g(x)\big) = g(x)q(x) + r(x)$ as shown in (2.7) where $q(x)$ and $r(x)$ are higher and lower terms of $I(x)$.

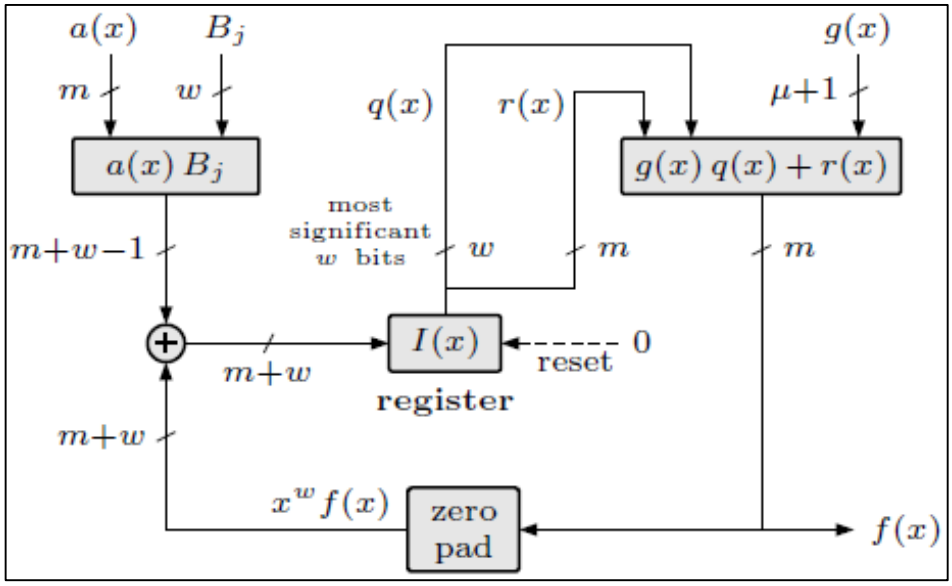

Figure 2.1: The block diagram of the MSD first multiplier where $M = \lceil m/w \rceil$ and $deg\big(g(x)\big) = \mu \leq m - w$.

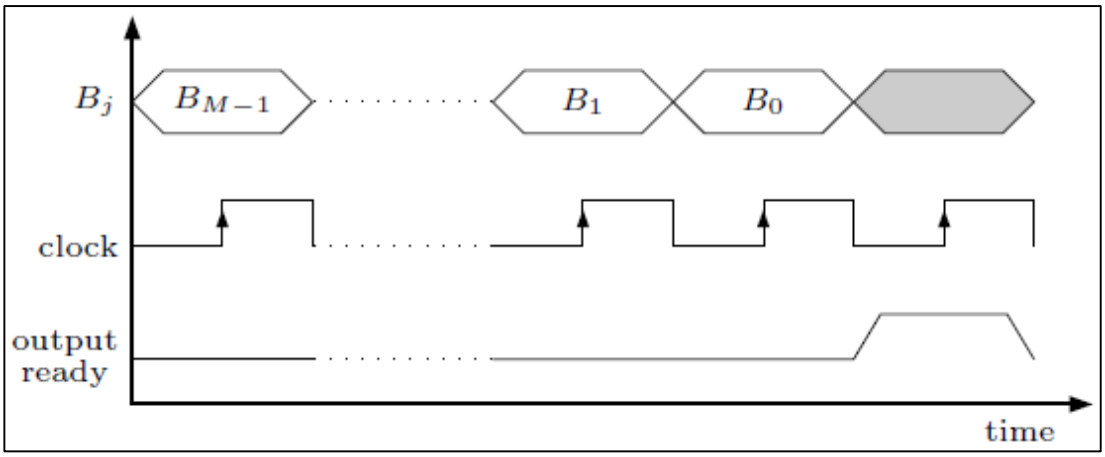

Figure 2.2: The timing diagram of the MSD first multiplier where $M = \lceil m/w \rceil$ and $deg\big(g(x)\big) = \mu \leq m - w$.

2.2. Least Significant Digit First Multipliers

These multipliers compute the field product in (2.4), starting from the least significant digit B_0 as given in Table 2.2 :

Table 2.2: LSD first field multiplication.

Inputs: $a(x) = \sum_{i=0}^{m-1} a_i x^i, \quad g(x) = \sum_{i=0}^{\mu} g_i x^i$

$b(x) = \sum_{i=0}^{m-1} b_i x^i = \sum_{j=0}^{\lceil m/w \rceil - 1} B_j x^{wj}$ where $B_j = \sum_{i=0}^{w-1} b_{wj+i} x^i$

Output: $f(x) = a(x)b(x) \; mod \; \big(x^m + g(x)\big)$

1: $J(x) = 0$

2: **for** $j = 0$ to $\lceil m/w \rceil - 1$

3: **if** $j = 0$ **then** $I(x) = a(x)$ **else** $I(x) = A(x)$

4: $A(x) = I(x)x^w \; mod \; \big(x^m + g(x)\big), \quad J(x) = I(x)B_j + J(x)$

5: $f(x) = J(x) \; mod \; \big(x^m + g(x)\big)$

In the jth iteration of the LSD first multiplication,

$$I(x) \equiv x^{wj} a(x) \; mod \; \big(x^m + g(x)\big) \tag{2.10}$$

$$J(x) \equiv \sum_{i=0}^{j} a(x)B_i x^{wi} \; mod \left(x^m + g(x)\right). \tag{2.11}$$

Figure 2.3 and Figure 2.4 illustrates this computation. Here, it is chosen that

$$deg\big(g(x)\big) = \mu \leq m - w. \tag{2.12}$$

Thus, $x^w I(x)$ and $J(x)$ can be reduced without division as shown in (2.7). For this, $I(x)$ and $J(x)$ are split as follows:

$$I(x) = \sum_{i=0}^{m-1} I_i x^i = q(x)x^{m-w} + r(x) \tag{2.13}$$

$$J(x) = \sum_{i=0}^{m+w-1} J_i x^i = q'(x)x^m + r'(x) \tag{2.14}$$

to compute $x^w I(x) \; mod \; \left(x^m + g(x)\right) = g(x)q(x) + x^w r(x)$ and $J(x) \; mod \; \left(x^m + g(x)\right) = g(x)q'(x) + r'(x)$.

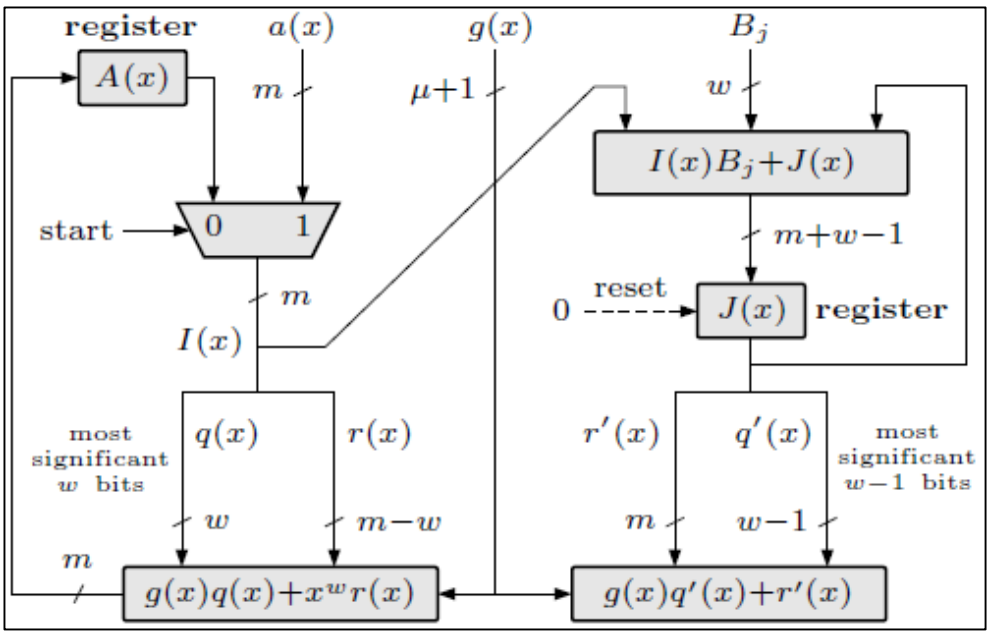

Figure 2.3: The block diagram of the LSD first multiplier where $M = \lceil m/w \rceil$ and $deg\big(g(x)\big) = \mu \leq m - w$.

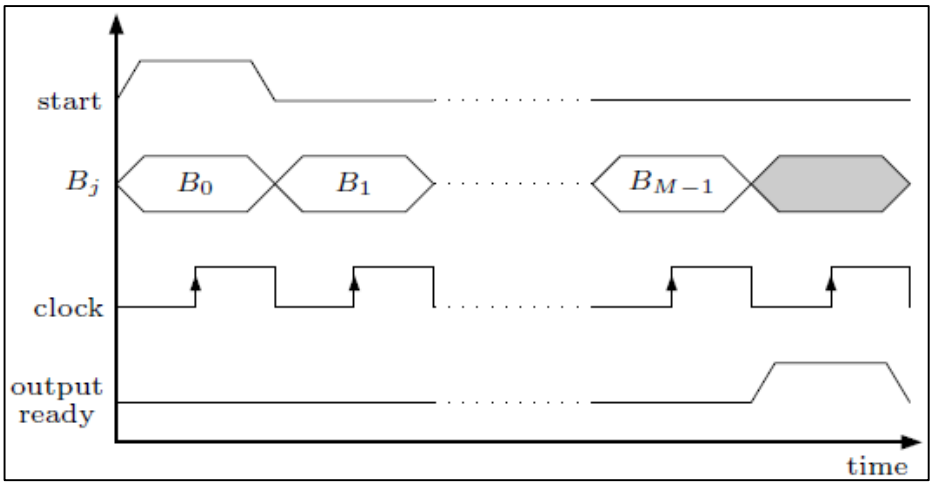

Figure 2.4: The timing diagram of the LSD first multiplier where $M = \lceil m/w \rceil$ and $deg\big(g(x)\big) = \mu \leq m - w$.

2.3. Complexities of Multipliers

Now, the complexities of the multipliers are studied, using the results of the following theorem.

Theorem 2.1: Let $\Delta(x)$, $u(x)$, *and* $v(x)$ *be three binary polynomials such that*

$$\Delta(x) = \sum_{i=0}^{\omega-1} \Delta_i x^i \ , \quad u(x) = \sum_{i=0}^{\ell-1} u_i x^i \ , \quad v(x) = \sum_{i=0}^{\grave{\ell}-1} v_i x^i \ , \tag{2.15}$$

where $deg\big(v(x)\big) \geq deg\big(\Delta(x)u(x)\big)$. *Let* T_{AND} and T_{XOR} *denote an AND gate delay and an XOR gate delay, respectively. The multiply operation* $\Delta(x)u(x)$ *has the following area requirement and critical path delay :*

- $\omega\ell$ AND gates, $(\omega - 1)(\ell - 1)$ XOR gates, $T_{AND} + T_{XOR}\lceil \log_2\big(min(\omega, \ell)\big)\rceil$.

The multiply-add operation $\Delta(x)u(x) + v(x)$ *has the following area requirement and critical path delay.*

- $\omega\ell$ AND gates, $\omega\ell$ XOR gates, $T_{AND} + T_{XOR}\lceil \log_2\big(min(\omega, \ell)\big) + 1\rceil$.

Corollary 2.1: Let $u(x)$ *have at most* τ *nonzero terms, i.e.*

$$u(x) = \sum_{i=0}^{\ell-1} u_i x^i = \sum_{i=1}^{\tau} u_{\mu_i} x^{\mu_i} \tag{2.16}$$

for some distinct integers μ_i *such that* $0 \le \mu_i \le \ell$. *Then, the multiply-add operation* $\Delta(x)u(x) + v(x)$ *has the following area requirement and critical path delay :*

- $\omega\tau$ AND gates, $\omega\tau$ XOR gates, at most $T_{AND} + T_{XOR}\lceil \log_2(min(\omega, \ell)) + 1 \rceil$.

Proof 2.1: The case $\omega = min(\omega, \ell, \tau)$ *is analyzed in this proof. At the end, the results are extended to the general case. Note that the product* $\Delta(x)u(x)$ *is a polynomial of degree* $\ell + \omega - 2$ *and its terms are as follows:*

$$\Delta(x)u(x) = \begin{cases} \left(\sum_{j=0}^{i} \Delta_j u_{i-j}\right) x^i, & 0 \le i < \omega - 1 \\ \left(\sum_{j=0}^{\omega-1} \Delta_j u_{i-j}\right) x^i, & \omega - 1 \le i < \ell \\ \left(\sum_{j=i+1-\ell}^{i} \Delta_j u_{i-j}\right) x^i, & \ell \le i < \ell + \omega - 1 \end{cases} \tag{2.17}$$

$$\Delta(x)u(x) = \begin{cases} \left(\sum_{j=0}^{i} \Delta_j u_{i-j}\right) x^i, & 0 \le i < \omega - 1 \\ \left(\sum_{j=0}^{\omega-1} \Delta_j u_{i-j}\right) x^i, & \omega - 1 \le i < \ell \\ \left(\sum_{j=i+1}^{\omega-1} \Delta_j u_{\ell+i-j}\right) x^{\ell+i}, & 0 \le i < \omega - 1 \,. \end{cases} \tag{2.18}$$

As seen, $\Delta(x)u(x)$ *has the terms* $\left(\sum_{j=0}^{\omega-1} \Delta_j u_{i-j}\right)x^i$ *for* $\omega - 1 \le i < \ell$. *Computing the coefficient of each* x^i *requires* ω AND *gates and* $(\omega - 1)$ XOR *gates. Thus, computing all these terms requires* $\omega(\ell - \omega + 1)$ AND *gates and* $(\omega - 1)(\ell - \omega + 1)$ XOR *gates. Also,* $\Delta(x)u(x)$ *has the following terms.*

$$\left(\sum_{j=0}^{i} \Delta_j u_{i-j}\right) x^i, \quad \left(\sum_{j=i+1}^{\omega-1} \Delta_j u_{\ell+i-j}\right) x^{\ell+i} \quad \text{for} \;\; 0 \le i < \omega - 1 \qquad (2.19)$$

Computing the coefficients of each x^i *and* $x^{\ell+i}$ *pair requires* ω AND *gates and* $(\omega - 2)$ XOR *gates. Thus, computing all these terms requires* $\omega(\omega - 1)$ AND *gates and* $(\omega - 2)(\omega - 1)$ XOR *gates.*

As a result, all the coefficients of $\Delta(x)u(x)$ *are obtained by* $\omega(\ell - \omega + 1) + \omega(\omega - 1) = \omega\ell$ AND *gates and* $(\omega - 1)(\ell - \omega + 1) + (\omega - 2)(\omega - 1) = (\omega - 1)(\ell - 1)$ XOR *gates. The coefficient computation* $\sum_{j=0}^{\omega-1} \Delta_j u_{i-j}$ *causes the largest delay. Thus, the critical path delay is* $T_{\text{AND}} + T_{\text{XOR}}\lceil \log_2 \omega \rceil$. *When* $\Delta(x)u(x) + v(x)$ *is computed instead of* $\Delta(x)u(x)$, XOR *gate count increases from* $(\omega - 1)(\ell - 1)$ *to*

$$(\omega - 1)(\ell - 1) + deg(\Delta(x)u(x)) + 1 = \omega\ell \qquad (2.20)$$

and the critical path delay becomes $T_{\text{AND}} + T_{\text{XOR}}\lceil \log_2(\omega + 1) \rceil$.

In the corollary, the case $u(x)$ *has at most* τ *nonzero coefficients are considered, i.e.* $u(x) = \sum_{i=0}^{\ell-1} u_i x^i = \sum_{i=1}^{\tau} u_{\mu_i} x^{\mu_i}$. *Then,* $\omega(\ell - \tau)$ AND *gates and* $\omega(\ell - \tau)$ XOR *gates are redundant. Thus,* AND *and* XOR *gate requirements each reduces from* $\omega\ell$ *to* $\omega\tau$ *. Also, the critical path delay remains the same or reduced.*

The complexities of $\Delta(x)u(x)$ *and* $\Delta(x)u(x) + v(x)$ *have already been analyzed for the case that* $\Delta(x)$ *has less terms than* $u(x)$, *i.e.* $\omega = min(\omega, \ell, \tau)$. *Since* $\Delta(x)u(x) = u(x)\Delta(x)$, *the same analysis can be repeated just by swapping* $\Delta(x)$ *with* $u(x)$ *for the case that* $\Delta(x)$ *has more terms than* $u(x)$. *Then, the complexities can be found by swapping* ω *with* ℓ, *and* ω *with* τ. *Naturally, the area complexities* $(\omega - 1)(\ell - 1)$, $\omega\ell$, *and* $\omega\tau$ *remain the same. However, the time complexities change and the general case time complexities are obtained by replacing* ω *with* $min(\omega, \ell)$ *and* ω *with* $min(\omega, \tau)$.

2.3.1. MSD First Multiplier

The area and time complexities of the MSD first multiplier can be found by analyzing Figure 2.1.

Computing the product $a(x)B_j$ corresponds to the multiplication in Theorem 2.1 for the case $\omega = w$ and $\ell = m$. Thus, it requires wm AND plus $(w-1)(m-1)$ XOR gates, and has the delay

$$T_{a(x)B_j} = T_{AND} + T_{XOR}\lceil \log_2 w \rceil. \tag{2.21}$$

Computing the reduction $f(x) = g(x)q(x) + r(x)$ corresponds to the multiply-add operation in Theorem 2.1 for the case $\omega = w$, $\ell = \mu + 1$, and $\ell' = m$. Thus, it requires $w(\mu + 1)$ AND plus $w(\mu + 1)$ XOR gates and has the delay

$$T_{f(x)} = T_{AND} + T_{XOR}\lceil \log_2(min(w, \mu + 1) + 1) \rceil. \tag{2.22}$$

$a(x)B_j$ is $m + w - 1$ bits and $f(x)$ is m bits. Thus, computing and storing $I(x) = a(x)B_j + x^w f(x)$ require $m - 1$ XOR gates plus $m + w$ flip flops and has the delay T_{XOR}. Then, the critical path delay

$$\begin{aligned} T_{delay} &= T_{XOR} + max\left(T_{a(x)B_j}, T_{f(x)}\right) \\ &= T_{AND} + T_{XOR} + T_{XOR}\lceil \log_2(w+1) \rceil. \end{aligned} \tag{2.23}$$

Actually, $T_{delay} = T_{AND} + T_{XOR} + T_{XOR}\lceil \log_2 w \rceil$ when $w > \mu + 1$ but this minor improvement is ignored.

2.3.2. LSD First Multiplier

The area and time complexities of the LSD first multiplier can be found by analyzing Figure 2.3 and Figure 2.4.

Computing $I(x)B_j + J(x)$ corresponds to the multiply-add operation in Theorem 2.1 for the case $\omega = w$, $\ell = m$, and $\ell' = w + m - 1$. Thus, it requires wm AND plus wm XOR gates and has the delay

$$T_{I(x)B_j + J(x)} = T_{AND} + T_{XOR}\lceil \log_2(w+1) \rceil. \tag{2.24}$$

Computing the reduction $g(x)q(x) + x^w r(x)$ corresponds to the multiply-add operation in Theorem 2.1 for the case $\omega = w$, $\ell = \mu + 1$, and $\ell' = m$. Thus, it requires $w(\mu + 1)$ AND plus $w\mu$ XOR gates. Actually, not only $w\mu$ XOR gates but also $w(\mu + 1)$ XOR gates are needed according to the theorem. Nevertheless, w XOR gates can be saved since the least significant w bits of $x^w r(x)$ are zero.

Computing the final reduction $g(x)q'(x) + r'(x)$ corresponds to the multiply-add operation for the case $\omega = w - 1$, $\ell = \mu + 1$, and $\ell' = m$. Thus, it requires $(w - 1)(\mu + 1)$ AND plus $(w - 1)(\mu + 1)$ XOR gates.

Storing $A(x)$ and $J(x)$ requires $2m + w - 1$ flip flops. Multiplexing $A(x)$ and $a(x)$ requires m-bit multiplexer. Also, the critical path delay

$$T_{delay} = T_{MUX} + T_{I(x)B_j + J(x)} = T_{AND} + T_{MUX} + T_{XOR}\lceil \log_2(w+1) \rceil. \qquad (2.25)$$

2.3.3. Comparison of MSD First and LSD First Multipliers

Table 2.3 and Table 2.4 summarize the complexity analyses of the MSD first and LSD first multipliers. Table 2.3 gives the area complexities for both general and *t*-nomial generator polynomials. The general and *t*-nomial generator polynomials are respectively given by :

Table 2.3: The area complexities of the multipliers for both general and *t*-nomial generator polynomials.

Multiplier	#AND	#XOR	#LATCH	#MUX
MSD 1st (general)	$wm + w(\mu + 1)$	$wm + w\mu$	$m + w + \mu + 1$	-
MSD 1st (*t*-nomial)	wm	$wm + w(t - 2)$	$m + w$	-
LSD 1st (general)	$wm + w(\mu + 1)$ $+(w - 1)(\mu + 1)$	$wm + w\mu$ $+(w - 1)(\mu + 1)$	$2m + w + \mu$	m
LSD 1st (*t*-nomial)	wm	$wm + w(t - 2)$ $+(w - 1)(t - 1)$	$2m + w - 1$	m

$$x^m + g(x) = x^m + \sum_{i=0}^{\mu} g_i x^i \,, \qquad x^m + g(x) = x^m + \sum_{i=1}^{t-1} x^{\mu_i} \tag{2.26}$$

where μ_i , $\mu \leq m - w$ to satisfy the restriction (2.6). The complexity results for general generator polynomials are obtained from the previous analyses in the thesis. Also, $\mu + 1$ additional flip flops are needed to store the generator coefficients g_i for $i = 0, 1, \ldots, \mu$ in general case as seen from Table 2.1. On the other hand, a *t*-nomial is a constant coefficient polynomial with t nonzero terms. Since the coefficients are constant, they do not need to be stored. Also, reduction with a *t*-nomial requires only hardwiring and XORing. This computation is very efficient when t is small. Note that the fast modular reduction in (2.7) becomes

$$\begin{aligned} q(x)x^m + r(x) \; mod \left(x^m + g(x)\right) &= r(x) + q(x)g(x) \\ &= r(x) + \sum_{i=1}^{t-1} q(x)x^{\mu_i} \end{aligned} \tag{2.27}$$

when the generator polynomial $x^m + g(x)$ is a *t*-nomial. As seen, reduction with a *t*-nomial modulo requires 0 AND gates and $w(t - 1)$ XOR gates where $deg\big(q(x)\big) \leq w$. Our previous analyses in the thesis show that modular reduction requires $w(\mu + 1)$ AND gates and $w(\mu + 1)$ XOR gates when a general generator polynomial with $deg\big(g(x)\big) = \mu$ is used. Therefore, the complexities in Table 2.1 for the *t*-nomial case are obtained from the complexities for the general case in two steps.

- First, the AND gates used in the reduction are excluded from the total AND count because reduction with constant coefficient *t*-nomials does not need any AND gates.
- Secondly, μ in the XOR count is substituted with $t - 2$ because XOR complexity is $w(t - 1)$ for the t-nomial case while XOR complexity is $w(\mu + 1)$ for the general case.

Table 2.4: The time complexities of the multipliers.

Multiplier	Critical Path Delay	Latency
MSD 1st	$T_{AND} + T_{XOR} + T_{XOR}\lceil \log_2(w+1) \rceil$	$\lceil m/w \rceil + 1$
LSD 1st	$T_{AND} + T_{MUX} + T_{XOR}\lceil \log_2(w+1) \rceil$	$\lceil m/w \rceil + 1$

Table 2.4 gives the time complexities of the multipliers. The delay of the partial product computation dominates the delay of the modular reduction as seen from (2.23) and (2.25). Thus, using sparse t-nomial generator polynomials cannot decrease the critical path delay here.

3. MSD FIRST DIGIT SERIAL MULTIPLIERS SUPPORTING MULTIPLE FIELDS

In this section, MSD first multipliers supporting multiple fields are proposed. MSD first multiplication is preferred to LSD first multiplication in the proposed design because the former requires less area than the latter as seen from Table 2.1.

3.1. Bit Level Representation

Let the supported field sizes be $m_1 < m_2 < \cdots < m_\lambda$. Then, at least m_λ bits are required to represent the field elements. When the multiplier works in the field $\mathbb{F}_{2^{m_k}}$, its inputs and output are m_k bits and they must be stored in m_λ bits with a proper alignment. Let $u(x)$ be a multiplier input or output. As seen from Figure 3.1 and Figure 3.2, $u(x)$ can be either right justified by zero extending or left justified by zero padding as follows:

$$\hat{u}(x) = x^{m_\lambda - m_k} u(x) = x^{m_\lambda - m_k} \sum_{i=0}^{m_k - 1} u_i x^i \ . \tag{3.1}$$

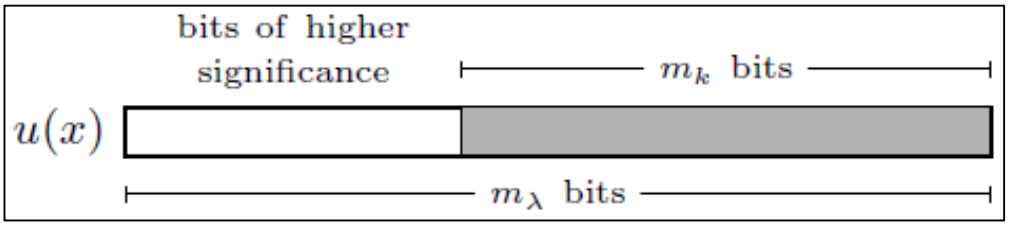

Figure 3.1: Bit level representation of $u(x) = \sum_{i=0}^{m_k-1} u_i x^i$. Here, the bits in the unshaded areas are all zero.

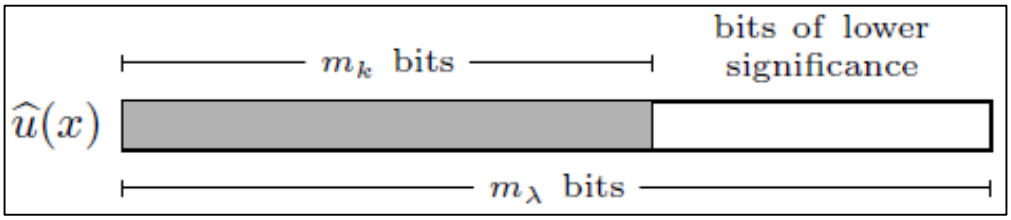

Figure 3.2: Bit level representation of left justified version $\hat{u}(x) = x^{m_\lambda - m_k} u(x)$. Here, the bits in the unshaded areas are all zero.

Note that the digit serial multiplication in (2.7) can be written for any $\Delta \geq 0$ as follows:

$$x^{\Delta} f(x) = \sum_{j=0}^{\lceil m_k/w \rceil - 1} x^{\Delta} a(x) B_j x^{wj} \; mod \; \left(x^{\Delta + m_k} + x^{\Delta} g(x) \right). \tag{3.2}$$

Then, when $\Delta = 0$ and when $\Delta = m_\lambda - m_k$, the digit serial multiplication becomes

$$f(x) = \sum_{j=0}^{\lceil m_k/w \rceil - 1} a(x) B_j x^{wj} \; mod \; \left(x^{m_k} + g(x) \right), \tag{3.3}$$

$$\hat{f}(x) = \sum_{j=0}^{\lceil m_k/w \rceil - 1} \hat{a}(x) B_j x^{wj} \; mod \; \left(x^{m_k} + \hat{g}(x) \right), \tag{3.4}$$

respectively. Here, $a(x), b(x), g(x), f(x)$ are the right justified and $\hat{a}(x), \hat{b}(x), \hat{g}(x), f(x)$ are the left justified operands. Figure 3.3 and Figure 3.4 illustrates the multipliers working with both the right and the left justified operands. As seen, the left justified alignment is advantageous since $I(x)$ is reduced to modulo degree m_k polynomial,

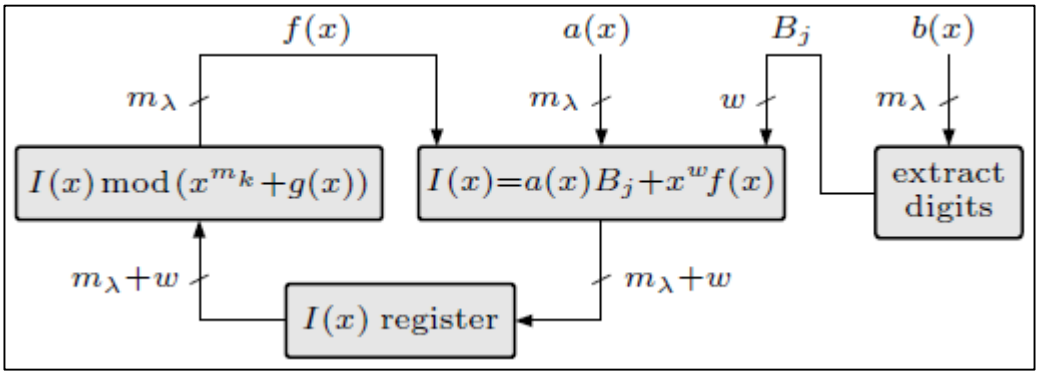

Figure 3.3: Multiplier working with the right justified input and output.

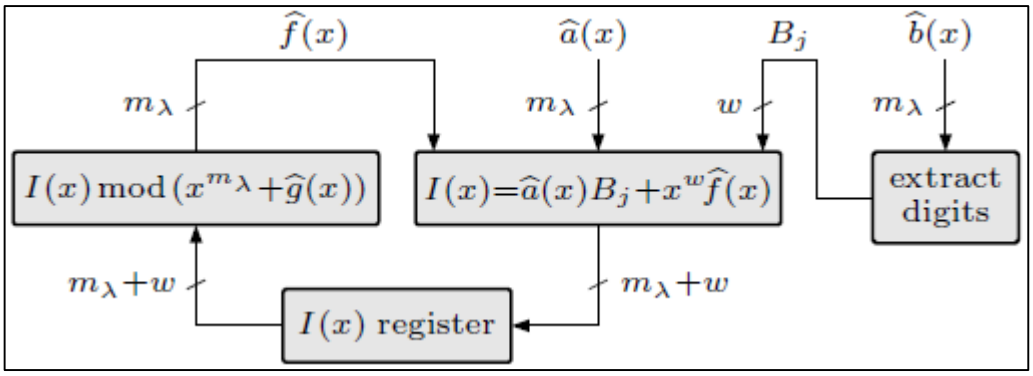

Figure 3.4: Multiplier working with the left justified input and output.

regardless of the selected field size m_k. However, when the operands are right justified, the reduction of $I(x)$ is performed modulo a degree m_k polynomial, and thus dependent on the selected field size.

3.2. Multipliers with Right Justified Operands

In this section, the MSD first multiplier in Figure 2.1 is modified to support multiple field sizes. The modified multiplier has right justified inputs and output.

3.2.1 Obtaining the Digits B_j

The MSD first multiplier in Figure 2.1 must extract the digits B_j from $b(x)$. For this, $b(x)$ is put in a $w\lceil m/w \rceil$ bit shift register. As the register shifts left by w bits at each clock, the most significant w bits are extracted to obtain the digits B_j one by one. When the multiple fields are supported, the digits of $b(x)$ are extracted as shown in Figure 3.5. The circuit in Figure 3.5 extracts the following $w\lambda$ bits of $b(x)$ in each cycle.

$$b_{w\lceil m_k/w \rceil - i}\,, \quad 0 < i \leq w\,, \;\; 1 \leq k \leq \lambda\,. \tag{3.5}$$

Let the selected field size be m_{k_0}. Then, only the following bits are needed among the extracted bits.

$$B_j = \{\, b_{w\lceil m_k/w\rceil - i} \;\mid\; 0 < i \le w\,,\;\; k = k_0 \,\}\,. \tag{3.6}$$

Thus, the extracted bits are ANDed with the selection bits $s_k = \delta[k - k_0]$ and the results are XORed to produce the digit B_j as shown in Figure 3.5.

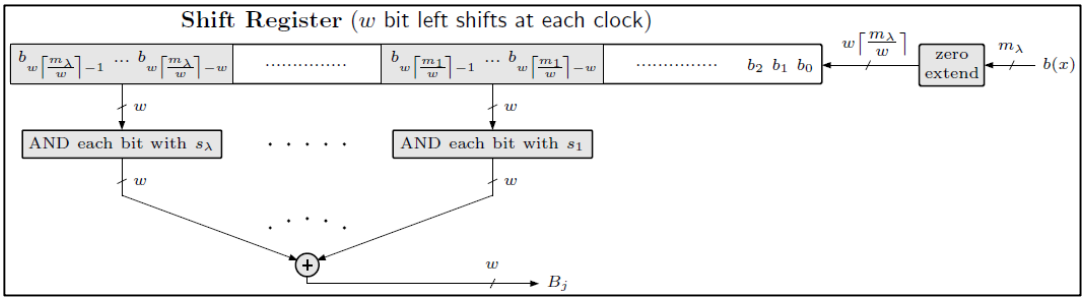

Figure 3.5: Obtaining the w bit digits B_j of the operand $b(x)$ where $m_1 < m_2 < \cdots < m_\lambda$ are the sizes of the supported fields and $s_1, s_2, \ldots, s_\lambda$ are the selection bits.

3.2.2 Reducing $I(x)$ modulo $x^{m_k} + g^{(k)}(x)$

The MSD first multiplier in Figure 2.1 splits the accumulated sum $I(x)$ into the polynomials $q(x) = \sum_{i=0}^{w-1} I_{m+i}x^i$ and $r(x) = \sum_{i=0}^{m-1} I_i x^i$ and reduces it to $f(x) = I(x)\ mod\ \left(x^m + g(x)\right) = q(x)g(x) + r(x)$.

Figure 3.6 illustrates the reduction of $I(x)$, when the multiple field sizes are supported and the operands are right justified. In the figure, $I(x)$ is split into

$$q(x) = \sum_{i=0}^{w-1} I_{m_{k_0}+i}x^i \quad , \quad r(x) = \sum_{i=0}^{m_{k_0}-1} I_i x^i \tag{3.7}$$

where m_{k_0} is the selected field size. The selection bits $s_k = \delta[k - k_0]$ are used to select the correct $q^{(k)}(x) = \sum_{i=0}^{w-1} I_{m_k+i}x^i$. Thus, $q(x)$ in (3.7) is computed in Figure 3.6 as follows:

$$q(x) = \sum_{k=1}^{\lambda} s_k q^{(k)}(x) = q^{(k_0)}(x) = \sum_{i=0}^{w-1} I_{m_{k_0}+i}x^i. \tag{3.8}$$

Also, $r(x)$ in (3.7) is computed in Figure 3.6 as follows:

$$r(x) = \sum_{\substack{i \notin [m_k\,,\, m_k+w) \\ for\ 1 \le k \le \lambda}} I_i x^i + \sum_{\substack{i \in [m_k\,,\, m_k+w) \\ for\ 1 \le k \le \lambda-1}} I_i \bar{s_k} x^i. \tag{3.9}$$

$\bar{s_k} = 1 - s_k$ is the negation of the selection bit. Thus, $\bar{s_k} = 0$ when $k = k_0$ and $\bar{s_k} = 1$ when $k \neq k_0$. Then,

$$r(x) = \sum_{i \notin [m_{k_0}\,,\, m_{k_0}+w)} I_i x^i = \sum_{i=0}^{m_{k_0}-1} I_i x^i \tag{3.10}$$

as given by (3.7). Note that $I_i = 0$ for $m_{k_0} + w \le i < m_\lambda$ above, because the operands are right justified.

The polynomials $x^{m_k} + g^{(k)}(x)$ are the generators of the fields $\mathbb{F}_{2^{m_k}}$ for $1 \le k \le \lambda$. Let the field size m_{k_0} be selected. Then, the polynomial used in the reduction

$$g(x) = \sum_{k=1}^{\lambda} s_k g^{(k)}(x) = \sum_{k=1}^{\lambda} \delta[k - k_0] g^{(k)}(x) = g^{(k_0)}(x) \tag{3.11}$$

and $f(x) = I(x)\ mod\ \left(x^{m_{k_0}} + g(x)\right) = q(x)\ g(x) + r(x)$ can be computed as shown in Figure 3.6. For example,

$$\begin{aligned}
x^{m_1} + g^{(1)}(x) &= x^{163} + x^{38} + x^{12} + x^2 + 1\,, \\
x^{m_2} + g^{(2)}(x) &= x^{233} + x^{38} + x^{12} + x^2 + 1\,, \\
x^{m_3} + g^{(3)}(x) &= x^{283} + x^{38} + x^{12} + x^2 + 1\,, \\
x^{m_4} + g^{(4)}(x) &= x^{409} + x^{38} + x^9 + x^2 + 1\,, \\
x^{m_5} + g^{(5)}(x) &= x^{571} + x^{22} + x^{12} + x^2 + 1
\end{aligned} \tag{3.12}$$

are irreducible polynomials and can be used as generator polynomials for the NIST fields. Then,

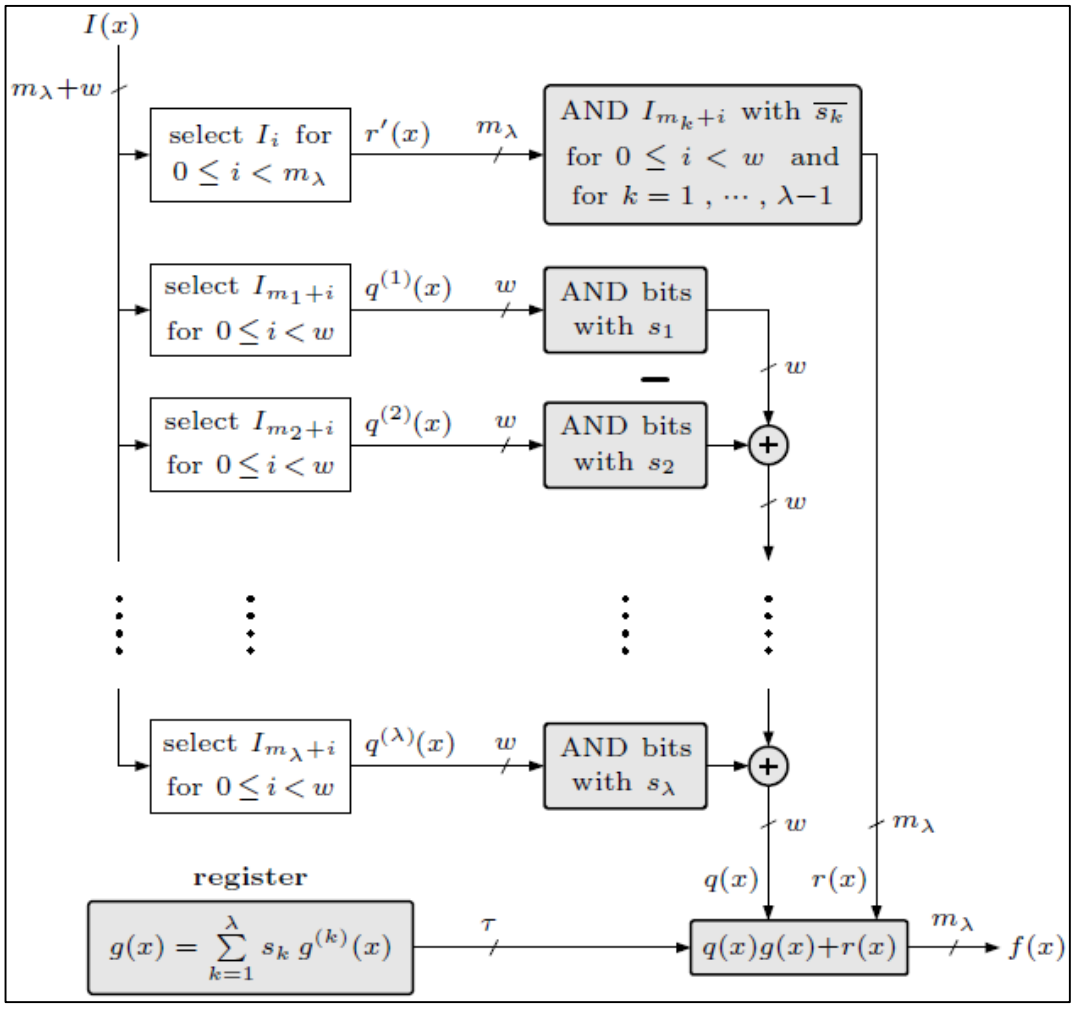

Figure 3.6: The circuit computing the unified reduction $f(x) = I(x)\ mod\ x^{m_k} + g^{(k)}(x)$ where $m_1 < m_2 < \ldots < m_\lambda$ are the supported field sizes and $s_1, s_2, \ldots, s_\lambda$ are the selection bits.

$$g(x) = g^{k_0}(x) = \bar{s_5}x^{22} + s_5x^{38} + \bar{s_4}x^{12} + s_4x^9 + x^2 + 1\,. \qquad (3.13)$$

Here, $g(x)$ is a polynomial with $\tau = 6$ terms and $s_k = \delta[k - k_0]$ are the bits selecting the field.

The appropriate reduction polynomial $g(x)$ is stored in a register and used in the reduction $q(x)g(x) + r(x)$ as shown in Figure 3.6. However, the reduction can also be carried out separately for each field by using its generator polynomial $x^{m_k} + g^{(k)}(x)$ as shown in the same Figure.

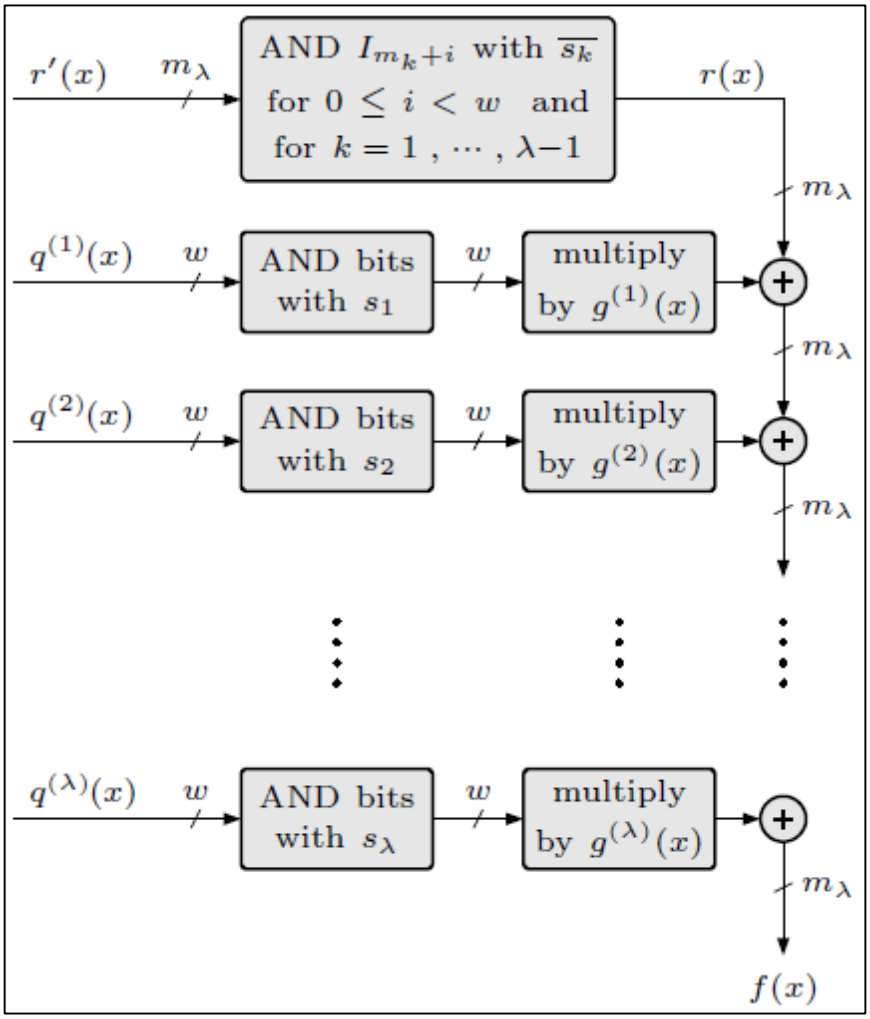

Figure 3.7: The circuit computing the separate reduction $f(x) = I(x)\ mod\ x^{m_k} + g^{(k)}(x)$ where $m_1 < m_2 < \ldots < m_\lambda$ are the supported field sizes and $s_1, s_2, \ldots, s_\lambda$ are the selection bits.

3.2.3 Performing Modular Reductions Separately

When the reduction is carried out separately for each field, the irreducible polynomials $x^{m_k} + g^{(k)}(x)$ must be chosen as sparse as possible to reduce the complexity. An irreducible pentanomial exists $x^m + g(x) = x^m + x^{\mu_1} + x^{\mu_2} + x^{\mu_3} + 1$ for each field size $m \geq 4$ [20]. When pentanomials are used as generators, the area requirement and the worst case critical path delay of the reductions are as follows:

$$w(2\lambda - 1)\ \text{ANDs}\ ,\quad 4w\lambda\ \text{XORs}\ ,\ T_{\text{AND}} + T_{\text{XOR}}\lceil \log_2(4\lambda + 1) \rceil\ . \qquad (3.14)$$

These complexities can be found by analyzing Figure 3.7. When the generator polynomials are pentanomials,

$$g^{(k)}(x) = x^{\mu_1^{(k)}} + x^{\mu_2^{(k)}} + x^{\mu_3^{(k)}} + 1\,. \tag{3.15}$$

Thus, $\sum_{k=1}^{\lambda} q^{(k)}(x)g^{(k)}(x) + r(x)$ is equal to the following, which needs 4 $deg\left(q^{(k)}(x)\right)\lambda \;=\; 4w\lambda$ XOR gates.

$$r(x) \mathrel{+}= \sum_{k=1}^{\lambda} q^{(k)}(x)x^{\mu_1^{(k)}} + q^{(k)}(x)x^{\mu_2^{(k)}} + q^{(k)}(x)x^{\mu_3^{(k)}} + q^{(k)}(x). \tag{3.16}$$

Also, the worst case critical path delay is $T_{\text{XOR}}(\lceil \log_2(4\lambda + 1)\rceil)$. Note that, when w and λ are small, the delay can be decreased by choosing $\mu_i^{(k)}$ suitably. Also, as seen in Figure 3.7, obtaining $r(x)$ from $r'(x)$ and selecting the correct $q^{(k)}$ require $w(\lambda - 1)$ plus $w\lambda$ AND gates and has a delay of T_{AND}.

3.2.4 Unified Modular Reduction Case

In this case, the irreducible polynomials $x^{m_k} + g^{(k)}(x)$ generating the supported fields $\mathbb{F}_{2^{m_k}}$ are chosen such that their trailing coefficients $g^{(k)}(x) = \sum_{i=1}^{\tau} g_{\mu_i}^{(k)} x^{\mu_i}$ can be nonzero only for the same τ terms.

Let the selected field size be m_{k_0}. Then, $g(x) = g^{(k_0)}(x)$ is stored into a τ bit register and used in the reduction computation $q(x)g(x) + r(x)$ as seen in Figure 3.7. The area and delay are as follows:

$$\begin{gathered} w(\tau + 2\lambda - 2)\text{ ANDs},\;\; w(\tau + \lambda - 1)\text{ XORs},\;\; \tau\; flip\; flops\,, \\ 2T_{\text{AND}} + T_{\text{XOR}}(\lceil \log_2 \lambda\rceil + \lceil \log_2(min(w,\tau) + 1)\rceil) \end{gathered} \tag{3.17}$$

where τ is the nonzero term count of $g(x)$. These are the complexities of storing $g(x)$, obtaining $q(x)$ and $r(x)$, and computing $g(x)q(x) + r(x)$. As seen from Figure 3.7,

obtaining $q(x)$ and $r(x)$ needs $w(2\lambda-1)$ AND plus $w(\lambda-1)$ XOR gates and has a delay of $T_{\text{AND}}+T_{\text{XOR}}(\lceil\log_2\lambda\rceil)$. Also, the area and delay

$$w\tau \text{ AND}, \qquad w\tau \text{ XOR}, \qquad T_{\text{AND}}+T_{\text{XOR}}\lceil\log_2(min(w,\tau)+1)\rceil \tag{3.18}$$

are needed to compute $g(x)q(x)+r(x)$ due to Corollary 2.1. But, w AND gates can be saved above. This is because $g(x)=g^{(k_0)}(x)$ where $x^{m_{k_0}}+g^{(k_0)}(x)$ is an irreducible and thus, its zeroth term $g_0=1$ always.

The nonzero term count τ of $g(x)$ must be as small as possible to decrease the complexities in (3.17).

$$\tau \le \log_2 m_\lambda + 3 \tag{3.19}$$

is guaranteed actually where the largest field size $m_\lambda \le 1000$. This is because it is an experimental fact that there always exists an irreducible $x^m+\sum_{i=1}^{\tau} g_{\mu_i}x^{\mu_i}$ in the following forms :

$$\begin{aligned}
&x^m+\sum\nolimits_{\mu\in\{0,1,\ldots,9\}\setminus\{4,7\}} g_\mu x^\mu \quad \text{for} \quad m\le 133,\\
&x^m+\sum\nolimits_{\mu\in\{0,1,\ldots,12\}\setminus\{3,8\}} g_\mu x^\mu \quad \text{for} \quad m\le 658,\\
&x^m+\sum\nolimits_{\mu\in\{0,1,\ldots,11\}\setminus\{8,9\}} g_\mu x^\mu \quad \text{for} \quad m\le 372,\\
&x^m+\sum\nolimits_{\mu\in\{0,1,\ldots,12\}\setminus\{8\}} g_\mu x^\mu \quad \text{for} \quad m\le 1000,
\end{aligned} \tag{3.20}$$

As seen above, τ can be as small as 8, 10, 11, and 12 for $m \le 133$, $m \le 372$, $m \le 658$, and $m \le 1000$, respectively. As a result, $min(\tau) \le \log_2 m+3$ for the polynomials in the cryptographic range of interest. Also, always an irreducible polynomial $x^m+g(x)$ exists for $m \le 2000$ such that $deg\big(g(x)\big) \le \log_2 m+3$ [21].

These practical facts are natural consequences of the following heuristic argument: Let m be an integer large enough so that $(1+1/m)^m \approx e$ (say $m>50$). Let $\mu_1,\mu_2,\ldots,\mu_\tau$ be some distinct integers such that $\mu_1=0<\mu_i<\mu_{i+1}<m$ for $2\le i\le\tau$. The probability of any of the polynomials

$$\left\{x^m + g(x) \middle| g(x) = \sum_{i=1}^{\tau} g_{\mu_i} x^{\mu_i} = g_0 + \sum_{i=2}^{\tau} g_{\mu_i} x^{\mu_i}\right\} \tag{3.21}$$

being irreducible is approximately equal to $(1/e)^{2^{\varepsilon}}$ when $\tau = \log_2 m + \varepsilon$ for some $\varepsilon \geq 0$. This is because a binary polynomial of degree m is irreducible with $1/m$ probability [1]. Then, the probability of any of the polynomials $x^m + \sum_{i=1}^{\tau} g_{\mu_i} x^{\mu_i}$ being irreducible is

$$\left(1 - \frac{1}{m}\right)^{2^{\tau}} = \left(1 - \frac{1}{m}\right)^{2^{\log_2 m + \varepsilon}} = \left(1 - \frac{1}{m}\right)^{m2^{\varepsilon}} \approx \left(\frac{1}{e}\right)^{2^{\varepsilon}} . \tag{3.22}$$

3.3. Multipliers with Left Justified Operands

In this section, the MSD first multiplier in Figure 2.1 is modified to support multiple field sizes. The modified multiplier has left justified inputs and output as illustrated in Figure 3.2.

3.3.1 Obtaining the Digits B_j

When the left justified operands are used instead of the right justified ones, the circuit in Figure 3.5 must be replaced with the one in Figure 3.8. In Figure 3.8, the following $w\lambda$ bits of $b(x)$ are extracted in each cycle.

$$b_{w\lceil m_k/w\rceil - i + \Delta} \ , \quad 0 < i \leq w, \ \ 1 \leq k \leq \lambda \tag{3.23}$$

where $\Delta = m_\lambda - m_k$. Note that, let $\phi_k = (-m_k \ mod \ w) + m_\lambda$, then these bits can also be given as follows:

$$b_{w\lceil m_k/w\rceil - i + \Delta} = b_{\phi_k - i} \ , \quad 0 < i \leq w, \ \ 1 \leq k \leq \lambda \ . \tag{3.24}$$

Let the selected field size be m_{k_0} . Then, only the following bits are needed among the extracted bits :

$$B_j = \{b_{w\lceil m_k/w\rceil - i + \Delta} = b_{\phi_k - i} | 0 < i \leq w,\ \ k = k_0\}. \tag{3.25}$$

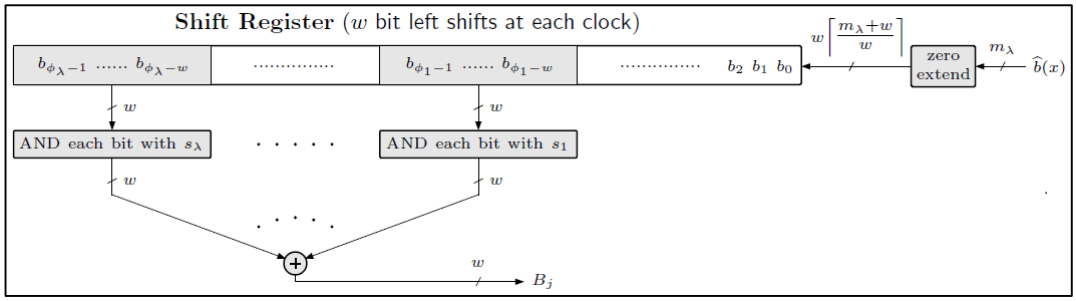

Figure 3.8: Obtaining the digits B_j from the left justified operand $\hat{b}(x)$. Here, $s_1, s_2, \ldots, s_\lambda$ are the selection bits and $\phi_k = (-m_k \ mod \ w) + m_\lambda$ for the supported field sizes $m_1 < m_2 < \cdots < m_\lambda$.

Thus, the extracted bits are ANDed with the selection bits $s_k = \delta[k - k_0]$ and the results are XORed to produce the digit B_j as shown in Figure 3.8.

3.3.2 Reducing $I(x)$ modulo $x^{m_\lambda} + \hat{g}^{(k)}(x)$

Figure 3.9 and Figure 3.10 illustrate the reduction of $I(x)$, when the multiple field sizes are supported and the operands are left justified.

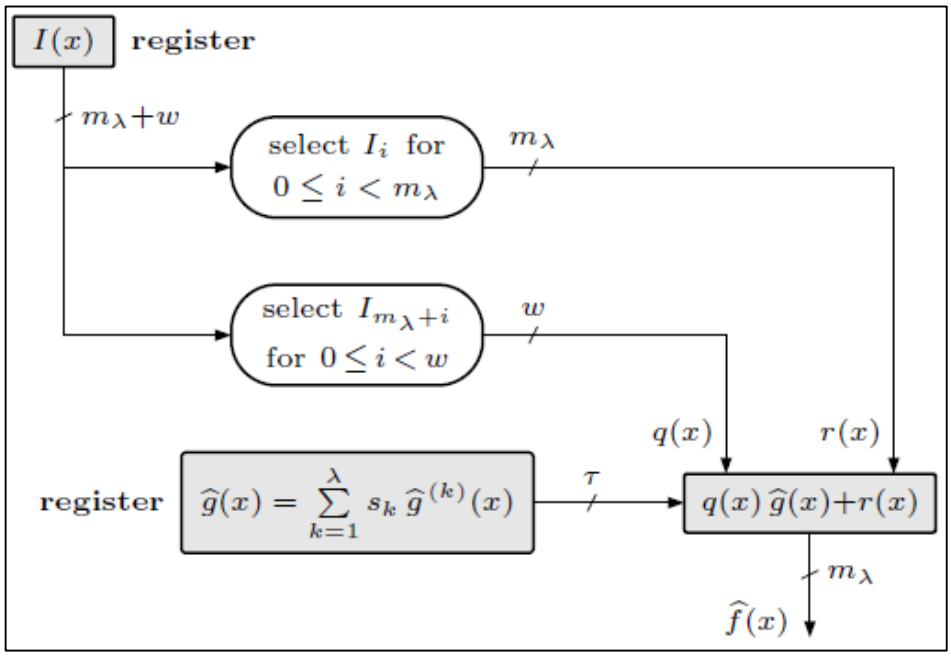

Figure 3.9: The circuit computing the unified reduction $\hat{f}(x) = I(x) \ mod \ x^{m_\lambda} + \hat{g}^{(k)}(x)$ where $m_1 < m_2 < \ldots < m_\lambda$ are the supported field sizes and $s_1, s_2, \ldots, s_\lambda$ are the selection bits.

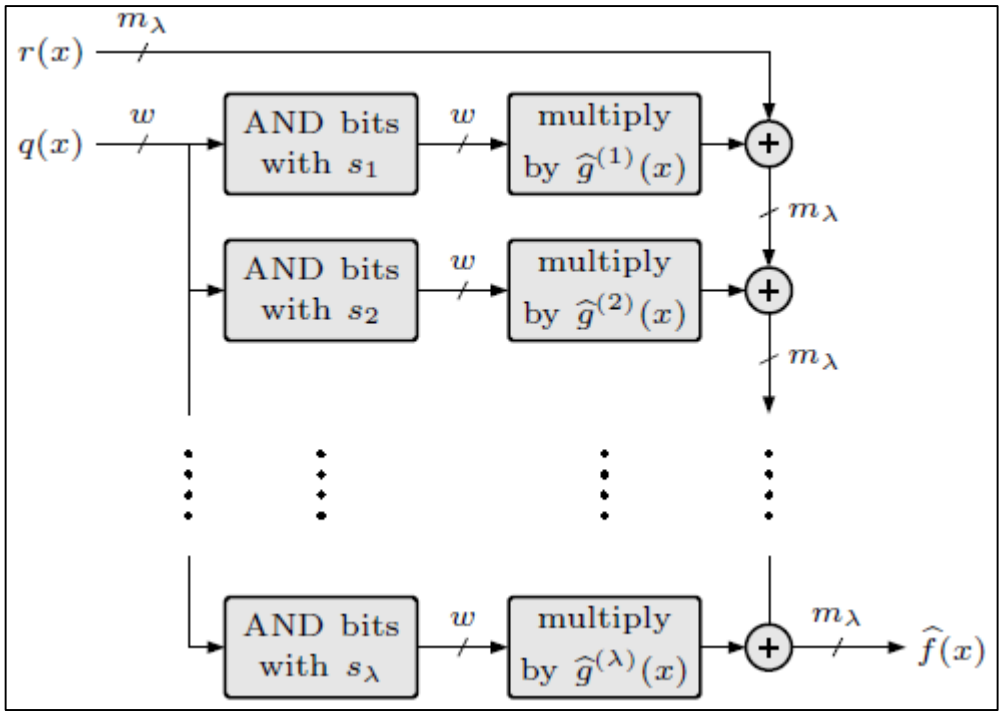

Figure 3.10: The circuit computing the separate reduction $\hat{f}(x) = I(x)\ mod\ x^{m_\lambda} + \hat{g}^{(k)}(x)$ where $m_1 < m_2 < \ldots < m_\lambda$ are the supported field sizes and $s_1, s_2, \ldots, s_\lambda$ are the selection bits.

The modular reduction circuit in Figure 3.9 and Figure 3.10 is much simpler than the one in Figure 3.6 and Figure 3.7 for the right justified operands since the left justified operands have the factor $x^{m_\lambda - m_k}$ where m_λ is the largest field size. Thus, regardless of the field size m_k, the most significant ith bit of an operand is stored in the bit $m_\lambda - i$ of its representation and $q(x)$ is stored in the bits $m_\lambda + i$ of $I(x)$ for $0 \leq i < w$. Thus,

$$q(x) = \sum_{i=0}^{w-1} I_{m_\lambda + i} x^i \quad , \quad r(x) = \sum_{i=0}^{m_\lambda - 1} I_i x^i \tag{3.26}$$

as shown in Figure 3.9 and Figure 3.10. Also, if the generators of the fields $\mathbb{F}_{2^{m_k}}$ are $x^{m_k} + g^{(k)}(x)$ for $1 \leq k \leq \lambda$,

$$\hat{g}^{(k)}(x) = x^{m_\lambda - m_k} g^{(k)}(x) \quad , \quad x^{m_\lambda} + \hat{g}^{(k)}(x) = x^{m_\lambda - m_k}\left(x^{m_k} + g^{(k)}(x)\right) . \tag{3.27}$$

Let the field size m_{k_0} be selected. Then,

$$\hat{g}(x) = \sum_{k=1}^{\lambda} s_k \hat{g}^{(k)}(x) = \sum_{k=1}^{\lambda} \delta[k - k_0]\hat{g}^{(k)}(x) = \hat{g}^{(k_0)}(x) \tag{3.28}$$

and $I(x)$ can be reduced into $\hat{f}(x) = I(x)\ mod\ \left(x^{m_\lambda} + \hat{g}(x)\right) = q(x)\hat{g}(x) + r(x)$ as shown in Figure 3.9 and Figure 3.10. Here, the output $\hat{f}(x)$ is left justified since the modulus is a left justified generator polynomial.

For example, the irreducible polynomials:

$$\begin{aligned}
x^{m_1} + g^{(1)}(x) &= x^{163} + x^{140} + x^{27} + x^{21} + 1, \\
x^{m_2} + g^{(2)}(x) &= x^{233} + x^{110} + x^{91} + x^{70} + 1, \\
x^{m_3} + g^{(3)}(x) &= x^{283} + x^{160} + x^{141} + x^{120} + 1, \\
x^{m_4} + g^{(4)}(x) &= x^{409} + x^{286} + x^{176} + x^{126} + 1, \\
x^{m_5} + g^{(5)}(x) &= x^{571} + x^{448} + x^{429} + x^{408} + 1
\end{aligned} \tag{3.29}$$

can be used as generators for the NIST recommended fields. Then, the left justified generator polynomials $x^{m_\lambda} + \hat{g}(x) = \left(x^{m_k} + g(x)\right)x^{m_\lambda - m_k}$ and the reduction polynomial $\hat{g}(x)$ are as follows:

$$\begin{aligned}
x^{m_\lambda} + \hat{g}^{(1)}(x) &= x^{571} + x^{448} + x^{435} + x^{429} + x^{408}, \\
x^{m_\lambda} + \hat{g}^{(2)}(x) &= x^{571} + x^{448} + x^{429} + x^{408} + x^{338}, \\
x^{m_\lambda} + \hat{g}^{(3)}(x) &= x^{571} + x^{448} + x^{429} + x^{408} + x^{288}, \\
x^{m_\lambda} + \hat{g}^{(4)}(x) &= x^{571} + x^{448} + x^{338} + x^{288} + x^{162}, \\
x^{m_\lambda} + \hat{g}^{(5)}(x) &= x^{571} + x^{448} + x^{429} + x^{408} + 1
\end{aligned} \tag{3.30}$$

$\hat{g}(x) = \hat{g}^{(k_0)}(x) = x^{448} + s_1 x^{435} + \bar{s_4} x^{429} + \bar{s_4} x^{408} + (s_2 + s_4)x^{338} + (s_3 + s_4)x^{288} + s_4 x^{162} + s_5$. Here, $\hat{g}(x)$ is a polynomial with $\tau = 8$ terms and $s_k = \delta[k - k_0]$ are the bits selecting the field.

The appropriate reduction polynomial $\hat{g}(x)$ is stored in a register and used in the reduction $q(x)\hat{g}(x) + r(x)$ as seen in Figure 3.9 and Figure 3.10 but the reduction

can also be carried out separately for each field by using its left justified generator polynomial $x^{m_\lambda} + \hat{g}^{(k)}(x)$ as seen in the same figure.

3.3.3 Performing Modular Reductions Separately

When the reduction is carried out separately for each field as seen in Figure 3.9, the generator polynomials must be chosen as sparse as possible to reduce the complexity. When the generators are pentanomials, the area requirement and worst case critical path delay can be found as follows:

$$w\lambda \text{ AND}, \quad 4w\lambda \text{ XOR}, \quad T_{\text{AND}} + T_{XOR}\lceil \log_2(4\lambda + 1) \rceil . \tag{3.31}$$

$\hat{g}^{(k)}(x) = x^{m_\lambda - m_k} g^{(k)}(x) = x^{m_\lambda - m_k}\left(x^{\mu_1^{(k)}} + x^{\mu_2^{(k)}} + x^{\mu_3^{(k)}} + 1\right)$ when the generators $x^{m_k} + g^{(k)}(x)$ are pentanomials, Thus, the reduction $\hat{f}(x) = \sum_{k=1}^{\lambda} s_k q(x)\, \hat{g}^{(k)}(x)$ is equal to the following :

$$r(x) + \sum_{k=1}^{\lambda} s_k q(x) x^{m_\lambda - m_k + \mu_1^{(k)}} + s_k q(x) x^{m_\lambda - m_k + \mu_2^{(k)}} + s_k q(x) x^{m_\lambda - m_k + \mu_3^{(k)}} + s_k q(x) x^{m_\lambda - m_k}. \tag{3.32}$$

Note that $deg(q(x)) < w$. Therefore, computing $s_k q(x)$ for $k = 1, 2, \ldots, \lambda$ requires $w\lambda$ AND gates and computing the additions above requires $4w\lambda$ XOR gates. All these computations have a worst case critical path delay of $T_{\text{AND}} + T_{XOR}(\lceil \log_2(4\lambda + 1) \rceil)$. Also, when w and λ are small, the delay can be reduced by choosing $\mu_i^{(k)}$ suitably.

3.3.4 Unified Modular Reduction Case

In this case, the irreducible polynomials $x^{m_k} + g^{(k)}(x)$ generating the supported fields $\mathbb{F}_{2^{m_k}}$ are chosen such that the coefficients of the polynomials $\hat{g}^{(k)}(x)$ can be nonzero only for the same τ terms as follows:

$$\hat{g}^{(k)}(x) = g^{(k)}(x)x^{m_\lambda - m_k} = \sum_{i=1}^{\tau} g_{\mu_i}^{(k)} x^{\mu_i + m_\lambda - m_k}. \quad (3.33)$$

Let the selected field size be m_{k_0} . Then, $\hat{g}^{(k)}(x) = \hat{g}^{(k_0)}(x)$ is stored into a τ bit register and used in the reduction computation $q(x)\hat{g}(x) + r(x)$ as seen in Figure 3.10. The area requirement and the critical path delay are as follows:

$$w\tau \text{ AND}, w\tau \text{ XOR}, \tau \text{ flip flops}, T_{\text{AND}} + T_{XOR}\lceil \log_2(\min(w, \tau) + 1)\rceil \quad (3.34)$$

where $\hat{g}(x)$ has τ nonzero terms. These are the complexities of storing $\hat{g}(x)$ and computing $q(x)\hat{g}(x) + r(x)$. As seen from Figure 3.9, $q(x)$ and $r(x)$ are obtained by just wiring without any cost since the operands are left justified. Due to Corollary 2.1, the cost of computing $q(x)\hat{g}(x) + r(x)$ is

$$w\tau \text{ AND}, \quad w\tau \text{ XOR}, \quad T_{\text{AND}} + T_{XOR}\lceil \log_2(\min(w, \tau) + 1)\rceil. \quad (3.35)$$

Note that the nonzero term count τ of $\hat{g}(x)$ must be as small as possible to decrease the complexities in (3.34). Actually, the probability of finding a value of

$$\tau \leq \log_2 m_1 + \varepsilon + \lambda - 1 \quad (3.36)$$

is very high for a small positive number ε where the supported field sizes are $m_1 < m_2 < \ldots < m_\lambda$.

The section with a heuristic proof of this claim has finished. The irreducible generator polynomials for the supported fields is $x^{m_k} + g^{(k)}(x)$ for $1 \leq k \leq \lambda$. The terms of $\hat{g}(x)$ with degree larger than $m_k - m_1$ form the polynomial $\alpha^{(k)}(x)x^{m_k - m_1}$. Then, the polynomials

$$g^{(k)}(x) = \alpha^{(k)}(x)x^{m_k - m_1} + \sum_{i=0}^{m_k - m_1} g_i^{(k)} x^i, \quad (3.37)$$

$$\hat{g}^{(k)}(x) = \alpha^{(k)}(x)x^{m_k - m_1} + \sum_{i=0}^{m_k - m_1} g_i^{(k)} x^{i + m_\lambda - m_k}. \tag{3.38}$$

The modular reduction by setting some polynomial terms to zero can be eased as follows:

$$g^{(k)}(x) = \sum_{j=1}^{\rho} \alpha_{v_j}^{(k)} x^{v_j + m_k - m_1} + \sum_{j=1}^{k} g_{m_k - m_j}^{(k)} x^{m_k - m_j} \ , \tag{3.39}$$

$$\hat{g}^{(k)}(x) = \sum_{j=1}^{\rho} \alpha_{v_j}^{(k)} x^{v_j + m_\lambda - m_1} + \sum_{j=1}^{k} g_{m_k - m_j}^{(k)} x^{m_\lambda - m_j}. \tag{3.40}$$

Now, the polynomials $\hat{g}^{(k)}(x)$ can be nonzero only for the following $\rho + k$ terms where $1 \leq k \leq \lambda$:

$$\begin{aligned} x^{\mu}, \quad \mu \in \{v_j + m_\lambda - m_1 | j = 1,2,\dots,\rho\} \\ x^{\mu}, \quad \mu \in \{m_\lambda - m_j | j = 1,2,\dots,k\}. \end{aligned} \tag{3.41}$$

Also, remember that $m_1 < m_2 < \ldots < m_\lambda$. Thus, $m_\lambda - m_1 < m_\lambda - m_2 < \ldots < m_\lambda - m_\lambda = 0$. The terms of the left justified polynomials $\hat{g}^{(k)}(x)$ can be classified into the two following groups :

- High order terms $\hat{g}_i^{(k)} x^i$ with degree $i > m_\lambda - m_1$.
- Low order terms $\hat{g}_i^{(k)} x^i$ with degree $i \leq m_\lambda - m_1$.

As seen from (3.41), the ρ high order terms of the polynomials $\hat{g}^{(k)}(x)$ are allowed to be nonzero. Also, among the low order terms of $\hat{g}^{(k)}(x)$, only the terms $x^{m_\lambda - m_1}$, $x^{m_\lambda - m_2}, \dots$, $x^{m_\lambda - m_\lambda} = 1$ are allowed to be nonzero. These low order terms cannot be eliminated since $x^m + g^{(k)}(x)$ for $1 \leq k \leq \lambda$ are all binary irreducible polynomials. Thus, $g_0^{(k)} = 1$ and the terms $g_{m_\lambda - m_k}^{(k)} x^{m_\lambda - m_k} = g_0^{(k)} x^{m_\lambda - m_k} = x^{m_\lambda - m_k}$ cannot be set to zero for all $k = 1, 2, \dots, \lambda$, but allowing the other low order

terms of $\hat{g}^{(k)}(x)$ to be nonzero is not useful since $\hat{g}^{(k)}(x) = g^{(k)}(x)x^{m_\lambda - m_k}$ is left justified. Thus, their many low order terms are already zero.

Then, the coefficients of the left justified polynomials can be nonzero only for the same $\rho + k$ terms as seen from (3.41).

$$\hat{g}^{(k)}(x) = g^{(k)}(x)x^{m_\lambda - m_k} = \sum_{i=1}^{\rho+k} \hat{g}_{\mu_i}^{(k)} x^{\mu_i} \tag{3.42}$$

Then, the probability of none of the polynomials

$$\left\{ x^{m_k} + g^{(k)}(x) \middle| g^{(k)}(x)x^{m_\lambda - m_k} = \textstyle\sum_{i=1}^{\rho+k} \hat{g}_{\mu_i}^{(k)} x^{\mu_i} \right\} \tag{3.43}$$

being irreducible is $(1 - 1/m_k)^{2^{\rho+k}}$ because binary polynomial of degree m_k is irreducible with $1/m_k$ probability [1]. When $\rho + k = \log_2 m_k + \varepsilon$, this probability is

$$\left(1 - \frac{1}{m_k}\right)^{2^{\log_2 m_k + \varepsilon}} = \left(1 - \frac{1}{m_k}\right)^{m_k 2^\varepsilon} \approx \left(\frac{1}{e}\right)^{2^\varepsilon} . \tag{3.44}$$

As a result, finding an irreducible $x^{m_k} + g^{(k)}(x)$ for each k is very highly possible when choosen

$$\rho = \max_{1 \le k \le \lambda} (\log_2 m_k - k) + \varepsilon . \tag{3.45}$$

The supported fields $m_1 < m_2 < \ldots < m_\lambda$ satisfy $m_k < 2^{k-1} m_1$ in practical applications. Then,

$$\rho = \max_{1 \le k \le \lambda} (\log_2 m_k - k) + \varepsilon \le \log_2 m_1 - 1 + \varepsilon \tag{3.46}$$

and $\tau \le max(\rho) + max(k) = \log_2 m_1 - 1 + \varepsilon + \lambda$ as in (3.36).

4. COMPLEXITY ANALYSIS

In this section, the complexities of the proposed multipliers are analyzed.

4.1. Area Requirement and Delay

Table 4.1 gives the area requirements of the MSD first multipliers supporting λ different fields. Table 4.1 also gives the area requirement of a usual MSD first multiplier supporting the single field size $m = m_\lambda$ and using a generator polynomial with $t = 5$ terms (pentanomial). The complexity of the multiplier supporting single field is obtained from Table 2.1. The complexities of the multipliers supporting multiple fields are obtained by adding the complexities of the three main computations shown in Figure 3.3 and Figure 3.4. These three computations and their area requirements are as follows:

Table 4.1: The area requirements of the proposed multipliers where the supported field sizes are $m_1 < m_2 < \ldots < m_\lambda$.

	#AND	#XOR	#LATCH
MSD 1st multiplier for the field $\mathbb{F}_{2^{m_\lambda}}$	wm_λ	$wm_\lambda + 3w$	$m_\lambda + w$
Right justified, separate reduction	$wm_\lambda + w(3\lambda - 1)$	$wm_\lambda + w(5\lambda - 2)$	$m_\lambda + w$
Right justified, unified reduction	$wm_\lambda + w(3\lambda - 2 + \tau)$	$wm_\lambda + w(2\lambda - 3 + \tau)$	$m_\lambda + w + \tau$
Left justified, separate reduction	$wm_\lambda + 2w\lambda$	$wm_\lambda + w(5\lambda - 2)$	$m_\lambda + w$
Left justified, unified reduction	$wm_\lambda + w(\lambda + \tau)$	$wm_\lambda + w(\lambda - 2 + \tau)$	$m_\lambda + w + \tau$

- Obtaining the digits B_j from $b(x)$ (or from $\hat{b}(x)$ for the left justified operands) is the first computation. It requires $w\lambda$ ANDs and $w(\lambda - 1)$ XORs regardless

of whether the operands are right or left justified. This can be seen from Figure 3.5 and Figure 3.8 easily.

- Computing and accumulating the partial products $x^w f(x) + a(x)B_j$ (or $x^w \hat{f}(x) + \hat{a}(x)B_j$ for the left justified operands) is the second computation. It requires wm_λ ANDs and $w(m_\lambda - 1)$ XORs regardless of whether the operands are right or left justified. This follows from Theorem 2.1. According to this theorem, accumulating the products of ω bit and ℓ bit operands requires $\omega\ell$ AND plus $\omega\ell$ XOR gates. The maximum supported field size is m_λ. Thus, $\ell = m_\lambda$ bits are used to represent the field element $a(x)$ and $\omega = w$ bits are used to represent the digits B_j . Then, wm_λ AND and wm_λ XOR gates are needed to accumulate the products $a(x)B_j$ (or $\hat{a}(x)B_j$ for the left justified operands). However, w XOR gates can be saved since the terms $x^0, x^1, \dots, x^{w-1}$ are not need to be add , in the sum $x^w f(x) + a(x)B_j$ (or $x^w \hat{f}(x) + \hat{a}(x)B_j$ for the left justified operands).
- The modular reduction $f(x) = I(x)\ mod\ (x^{m_k} + g(x))$ (or $\hat{f}(x) = I(x)\ mod\ (x^{m_k} + \hat{g}(x))$ for the left justified operands) is the third computation. The area requirements of two different modular reduction schemes are given by (3.14) and (3.17) (or (3.31) and (3.34) for the left justified operands).

The critical path delays of the proposed multipliers can be given as follows:

$$2T_{\text{AND}} + (\lceil \log_2 \lambda \rceil + \lceil \log_2 w \rceil + 1)T_{XOR} \tag{4.1}$$

where λ is the number of the supported fields and w is the digit size in bits. This fact can be explained as follows: As seen from Figure 3.3, the critical path delay of the proposed design can not be smaller than the sum of the delays of the following three computations.

- The first one is the extraction of the digit B_j from $b(x)$ (or from $\hat{b}(x)$ for the left justified operands). This has the delay $T_{\text{AND}} + T_{XOR}\lceil \log_2 \lambda \rceil$ as seen from Figure 3.5 and Figure 3.8.

- The second one is the $m_\lambda \times w$ bit partial product $a(x)B_j$ (or $\hat{a}(x)B_j$ for the left justified operands). This has the delay $T_{\text{AND}} + T_{XOR}\lceil \log_2 min(w, m_\lambda)\rceil = T_{\text{AND}} + T_{XOR}\lceil \log_2 w\rceil$ according to Theorem 2.1.
- Third one is the accumulation of the partial products, which has one T_{XOR} delay.

The total delay due to these computations is $T = 2T_{\text{AND}} + (\lceil \log_2 \lambda\rceil + \lceil \log_2 w\rceil)T_{XOR} + T_{XOR}$ and equal to (4.1). As seen from Figure 3.3 , the critical path delay is actually $max(T\ , T_{\text{reduction}} + T_{\text{XOR}})$ where $T_{\text{reduction}}$ is the delay of the modular reduction $f(x) = I(x)\ mod\ (x^{m_k} + g(x))$ (or $\hat{f}(x) = I(x)\ mod\ (x^{m_k} + \hat{g}(x))$ for the left justified operands). $T_{\text{reduction}}$ is given by (3.14), (3.17), (3.31), and (3.34) for four different cases. A simple investigation shows that the critical path delay is $T = max(T\ , T_{\text{reduction}} + T_{\text{XOR}})$ and equal to (4.1) .

4.2. NIST Recommended Binary Fields

Table 4.2 gives the area complexities of the MSD first multipliers when they support the five binary fields recommended by NIST. The number of the NIST fields $\lambda = 5$ and the largest NIST field size $m_\lambda = 571$.

Table 4.2: The area requirements of the proposed multipliers supporting the NIST fields $\mathbb{F}_{2^{163}}, \mathbb{F}_{2^{233}}, \mathbb{F}_{2^{283}}, \mathbb{F}_{2^{409}}, \mathbb{F}_{2^{571}}$.

	#AND	#XOR	#LATCH
MSD 1st multiplier for the field $\mathbb{F}_{2^{571}}$	$571w$	$574w$	$571 + w$
Right justified field elements, separate reduction circuits	$585w$	$594w$	$571 + w$
Right justified field elements, unified reduction circuits ($\tau = 6$)	$590w$	$584w$	$577 + w$
Left justified field elements, separate reduction circuits	$581w$	$594w$	$571 + w$
Left justified field elements, unified reduction circuits ($\tau = 8$)	$584w$	$582w$	$579 + w$

The complexities in the table are obtained from Table 4.1 for these NIST parameters.

The modular reduction can be unified for the NIST fields by using a common reduction polynomial. When the field elements are right justified, the polynomial with $\tau = 6$ terms

$$g(x) = g_{38}x^{38} + g_{22}x^{22} + g_{12}x^{12} + g_9x^9 + g_2x^2 + g_0 \tag{4.2}$$

given by (3.13) can be used as reduction polynomial. Also, when the field elements are left justified, the polynomial with $\tau = 8$ terms

$$\hat{g}(x) = \hat{g}_{448}x^{448} + \hat{g}_{435}x^{435} + \hat{g}_{429}x^{429} + \hat{g}_{408}x^{408} + \hat{g}_{338}x^{338} + \hat{g}_{288}x^{288} + \hat{g}_{162}x^{162} + \hat{g}_0 \tag{4.3}$$

given by (3.30) can be used as reduction polynomial. The coefficients of these polynomials are determined by the selected field as seen from (3.13) and (3.30).

Table 4.2 shows that supporting multiple fields instead of a single field does not increase the cost much and working with the left justified operands is slightly advantageous. Also, because the number of the fields $\lambda = 5$ is small for the NIST case, using a separate reduction circuit for each field is affordable. The delay of the multipliers supporting the NIST fields can be obtained from (4.1) by substituting $\lambda = 5$.

4.3. Comparison with Other Multipliers

Table 4.3 and Table 4.4 give the space and time complexities of several digit serial multipliers. The space complexities of the proposed architectures in the Table are obtained from Table 4.1 by substituting τ with the upper bounds in (3.19) and (3.36). As seen, the complexities of the multipliers supporting a single field are very similar and the usual MSD first multiplier shows one of the best performances. Note that the MSD first multiplier needs where the size of the working field $m_k = m_\lambda$. If this area

and the areas of the proposed multipliers supporting λ fields are analyzed, the following conclusions are reached :

$$wm_\lambda \text{ ANDs}, \quad wm_\lambda + 3w \text{ XORs}, \quad m_\lambda + w \text{ latches} \tag{4.4}$$

- The area increases linearly with λ
- $w\lambda$ AND gates can be saved when the multiplication is performed with left justified operands.
- Unifying the modular reductions of the supported λ fields causes the terms $\log_2 m_1$ or $\log_2 m_\lambda$ to appear in the area complexities additionally but also, decreases 5λ term in the XOR complexity to 2λ. Thus, the multipliers with unified reduction circuit can be advantageous for large λ.

Table 4.3: Comparison of area complexities of digit serial multipliers working in the field $\mathbb{F}_{2^{m_k}}$.

Multiplier	#AND	#XOR	#MUX	#LATCH
† right justified operands separate reduction for field sizes $m_1 \leq m_k \leq m_\lambda$	$w(m_\lambda + 3\lambda - 1)$	$w(m_\lambda + 5\lambda - 2)$		$m_\lambda + w$
† left justified operands separate reduction for field sizes $m_1 \leq m_k \leq m_\lambda$	$w(m_\lambda + 2\lambda)$	$w(m_\lambda + 5\lambda - 2)$		$m_\lambda + w$
right justified operands unified reduction for field sizes $m_1 \leq m_k \leq m_\lambda$	$w(m_\lambda + 3\lambda - 2) + w(\log_2 m_\lambda + \varepsilon)$	$w(m_\lambda + 2\lambda - 3) + w(\log_2 m_\lambda + \varepsilon)$		$m_\lambda + w + \log_2 m_\lambda + \varepsilon$

Table 4.3: Continue.

Multiplier	#AND	#XOR	#MUX	#LATCH
left justified operands unified reduction for field sizes $m_1 \leq m_k \leq m_\lambda$	$w(m_\lambda + 2\lambda - 1) + w(\log_2 m + \varepsilon)$	$w(m_\lambda + 2\lambda - 3) + w(\log_2 m_1 + \varepsilon)$		$m_\lambda + w + \lambda + \log_2 m_1 + \varepsilon - 1$
† MSD 1st multiplier	wm_k	$wm_k + 3w$		$m_k + w$
† LSD 1st multiplier [11]	wm_k	$wm_k + 7w - 4$	m_k	$2m_k + w - 1$
Kim et al. [8], [9]	$(2w^2 + w)\left\lceil \frac{m_k}{w} \right\rceil$	$2w^2\left\lceil \frac{m_k}{w} \right\rceil$	$2w\left\lceil \frac{m_k}{w} \right\rceil$	$(10w + 1)\left\lceil \frac{m_k}{w} \right\rceil$
‡ Meher [12]	wm_k	$(w-1)m_k + \frac{(w^2 + w)}{2}$		$2m_k + w$
Talapatra et al. [22]	wm_k	$wm_k + 2w$	$2m_k$	$4m_k + 3w + 1$
† pentanomial and ‡ trinomial generator polynomials are used, w is the digit size, λ is the number of the supported fields. ε is a small positive number, TA, TX, and TMUX are respectively the delays of AND gate, XOR gate, and multiplexer.				

Table 4.4: Comparison of time complexities of digit serial multipliers working in the field $\mathbb{F}_{2^{m_k}}$.

Multiplier	Critical Path	Latency
† right justified operands separate reduction for field sizes $m_1 \leq m_k \leq m_\lambda$	$2T_A + T_X + (\lceil \log_2 \lambda \rceil + \lceil \log_2 w \rceil)T_X$	$\left\lceil \frac{m_k}{w} \right\rceil + 1$

Table 4.4: Continue.

Multiplier	Critical Path	Latency
† left justified operands separate reduction for field sizes $m_1 \leq m_k \leq m_\lambda$	$2T_A + T_X + (\lceil \log_2 \lambda \rceil + \lceil \log_2 w \rceil)T_X$	$\left\lceil \frac{m_k}{w} \right\rceil + 1$
right justified operands unified reduction for field sizes $m_1 \leq m_k \leq m_\lambda$	$2T_A + T_X + (\lceil \log_2 \lambda \rceil + \lceil \log_2 w \rceil)T_X$	$\left\lceil \frac{m_k}{w} \right\rceil + 1$
left justified operands unified reduction for field sizes $m_1 \leq m_k \leq m_\lambda$	$2T_A + T_X + (\lceil \log_2 \lambda \rceil + \lceil \log_2 w \rceil)T_X$	$\left\lceil \frac{m_k}{w} \right\rceil + 1$
† MSD 1st multiplier	$T_A + T_X + (\lceil \log_2 w \rceil + 1)T_X$	$\left\lceil \frac{m_k}{w} \right\rceil + 1$
† LSD 1st multiplier [11]	$T_A + T_{MUX} + (\lceil \log_2 w \rceil + 1)T_X$	$\left\lceil \frac{m_k}{w} \right\rceil + 1$
Kim et al. [8], [9]	$w(T_A + T_X) + (w-1)T_{MUX}$	$3\left\lceil \frac{m_k}{w} \right\rceil$
‡ Meher [12]	$T_A + (\lceil \log_2 w \rceil + 1)T_X$	$\left\lceil \frac{m_k}{w} \right\rceil$
Talapatra et al. [22]	$T_A + T_{MUX} + \lceil \log_2 w \rceil T_X$	$2\left\lceil \frac{m_k}{w} \right\rceil$
† pentanomial and ‡ trinomial generator polynomials are used, w is the digit size, λ is the number of the supported fields. ε is a small positive number, TA, TX, and TMUX are respectively the delays of AND gate, XOR gate, and multiplexer.		

Table 4.5: Virtex 5 implementations of digit serial multipliers supporting NIST field sizes 163, 233, 283, 409, 571.

Multipliers with w = 8 digit size	FF	increase	LUT	increase	Period (ns)	increase
MSD 1st multiplier for the field $F_{2^{571}}$	1728	0%	3186	0%	3.224	0%
Right justified operands, separate reduction	1744	1%	3239	2%	3.987	24%
Left justified operands, separate reduction	1756	2%	3248	2%	3.688	14%
Right justified operands, unified reduction	1744	1%	3487	9%	3.896	21%
Left justified operands, unified reduction	1756	2%	3477	9%	3.891	21%

Compared to the digit serial multipliers, the versatile bit serial multipliers [15] - [19] require less area and their critical path delays are also smaller. However, they need at least m_k cycles to finish the multiplication while many digit serial multipliers need approximately $\lceil m_k/w \rceil$ cycles.

Table 4.5 gives the FPGA implementation results of the usual MSD first multiplier and proposed multipliers. These results were synthesized using Xilinx ISE Webpack version 14.7 for the Xilinx Virtex xc5vlx50-11ff1153. As seen, the areas of the proposed multipliers are not significantly larger than the area of the usual MSD first multiplier, even though they support five NIST fields. However, there is a considerable increase in the minimum clock period. Also, the area needed for the multipliers using unified reduction strategy is larger than the area needed for the ones using separate reduction strategy. This is because the number of the supported fields $\lambda = 5$ is small compared to $\log_2 m_\lambda$. The unified reduction strategy becomes more advantageous as λ increases.

5. CONCLUSION

Digit serial multipliers performing multiplication in a collection of binary extension fields $\mathbb{F}_{2^{m_1}}, \mathbb{F}_{2^{m_2}}, \ldots, \mathbb{F}_{2^{m_\lambda}}$ have been proposed. Their complexities where the digit size is w bits and the field sizes satisfy $m_1 < m_2 < \ldots < m_\lambda$ have been analyzed. The results are presented in Table 4.3 and Equation (4.1). The area requirements of the multipliers increase linearly with m_λ, λ, and w . On the other hand, the critical path delay grows logarithmically with λ and w .

When the multiplication is performed with left justified operands, $w\lambda$ AND gates can be saved. However, this is a minor improvement, compared to the overall area requirements. The area improvement is obtained because the left justified alignment eases the modular reduction.

Also, unifying the modular reductions of the supported fields decreases the area requirements for large λ . To unify the modular reductions, the modulus polynomials are chosen so that their trailing coefficients can be nonzero only for the same τ terms. Then, the modular reduction involves only these τ terms, and thus the reduction process gets easier. However, τ must be large enough so that the irreducible modulus polynomials can be found for the chosen τ terms easily. For this purpose, τ must be at least as large as the logarithm of the supported field sizes as shown in this thesis.

REFERENCES

[1] Lidl R., Niederreiter H., (1994), "Introduction to Finite Fields and Their Applications", 2nd Edition, Cambridge University Press.

[2] Vanstone S. A., Oorschot P., (1989), "An Introduction to Error Correcting Codes with Applications", 1st Edition, Kluwer.

[3] Hankerson D., Menezes A. J., Vanstone S., (2004), "Guide to Elliptic Curve Cryptography", Springer.

[4] Miller V. S., (1986), "Use of Elliptic Curves in Cryptography", Springer, 218, 417–426.

[5] NIST, (1999), "Recommended Elliptic Curves for Federal Government Use", National Institute of Standards and Technology.

[6] NIST, (2009), "Digital Signature Standard", National Institute of Standards and Technology.

[7] Song L., Parhi K. K., (1998), "Low-energy digit-serial/parallel finite field multipliers", Journal of VLSI Signal Processing, 19 (2), 149–166.

[8] Kim C. H., Han S. D., Hong C. P., (2001), "An efficient digit-serial systolic multiplier for finite fields GF(2m)", in: Proc. 14th Ann. IEEE Int. ASIC/SOC Conf., IEEE, 361–365, Kyungbuk, Korea, 12-15 September.

[9] Kim C. H., Hong C. P., Kwon S., (2005), "A digit-serial multiplier for finite field GF(2m)", IEEE Transactions on Very Large Scale Integration Systems, 13 (4), 476–483.

[10] Kim C. H., Kwon S., Hong C. P., (2005), "A fast digit-serial systolic multiplier for finite fields GF(2m)", in: Proc. Asia South Pacific Des. Autom. Conf. (ASP-DAC), Vol. 2, IEEE, 1268–1271, Jinryang, Kyungsan, Korea, 18-21 January.

[11] Kumar S. S., Wollinger T., Paar C., (2006), "Optimum digit serial GF(2m) multipliers for curve-based cryptography", IEEE Transactions on Computers, 55 (10), 1306–1311.

[12] Meher P. K., (2009), "On efficient implementation of accumulation in finite field over GF(2m) and its applications", IEEE Transactions on Very Large Scale Integration Systems, 17 (4), 541–550.

[13] Jeng-Shyang P., Chiou-Yng L., Meher P. K., (2013), "Low-latency digit-Serial and digit-parallel systolic multipliers for large binary extension fields", IEEE Transactions on Circuits and Systems, 60 (12), 3195–3204.

[14] Cheung R. C. C., Telle N. J., Luk W., Cheung P. Y. K., (2005), "Customizable elliptic curve cryptosystems", IEEE Transactions on Very Large Scale Integration Systems 13 (9), 1048–1059.

[15] Kitsos P., Theodoridis G., Koufopavlou O. G., (2003), "An efficient reconfigurable multiplier architecture for Galois field GF(2m)", Microelectronics Journal, 34, 975–980.

[16] Fournaris A. P., Koufopavlou O. G., (2008), "Versatile multiplier architectures in GF(2m) fields using the montgomery multiplication algorithm", Integration, 41 (3), 371–384.

[17] Selimis G. N., Fournaris A. P., Michail H. E., Koufopavlou O. G., (2009), "Improved throughput bit-serial multiplier for GF(2m) fields", Integration, 42 (2), 217–226.

[18] Nikooghadam M., Malekian E., Zakerolhosseini A., (2009), "A versatile reconfigurable bit-serial multiplier architecture in finite fields GF(2m)", Advances in Computer Science and Engineering Communications in Computer and Information Science, 6 (2), 227–234.

[19] Zakerolhosseini A., Nikooghadam M., (2013), "Low-power and high-speed design of a versatile bit-serial multiplier in finite fields GF(2m)", Integration, 46 (2), 211–217.

[20] Seroussi G., (1998), "Table of Low-Weight Binary Irreducible Polynomials", Technical Reportno : HPL-98-135, Hewlett-Packard Laboratories, Palo Alto, CA, USA.

[21] Gao S., Howell J., Panario D., (1999), "Irreducible polynomials of given forms", Contemporary Mathematics, 225, 43–54.

[22] Talapatra S., Rahaman H., Saha S. K., (2010), "Unified digit serial systolic montgomery multiplication architecture for special classes of polynomials over GF(2m)", in: 2013 Euromicro Conference on Digital System Design, IEEE, 427–432, Shibpur, India, 1-3 September.

BIOGRAPHY

Bilal Uslu was born December 26, 1979 in İzmir. He received the Bachelor of Science in Electronics Engineering from Kocaeli University in 2002 and the Master of Science in Electronics Engineering from the Graduate School of Engineering and Sciences of Gebze Institute of Technology in 2006. He received Ph.D. degree in Electronics Engineering from the Graduate School of Natural and Applied Sciences of Gebze Technical University in 2016. His research interests include cryptography, computer arithmetic, finite fields, and software design.

APPENDICES

Appendix A: Publications of the Thesis

Uslu B., Erdem S.S., (2015), “Versatile digit serial multipliers for binary extension fields”, Elsevier Computers & Electrical Engineering, 46, 29-45