FPGA accelerated multipliers over binary composite fields constructed via low hamming weight irreducible polynomials

C. Shu, S. Kwon and K. Gaj

Abstract: The efficient design of digit-serial multipliers for special binary composite fields, $\mathbb{F}_{2^{nm}}$ where $\gcd(n, m) = 1$, is presented. These composite fields can be constructed via an irreducible pentanomial of degree nm but not an irreducible trinomial of degree nm. The conventional construction method for such digit-serial multipliers is to exploit the simplicity of pentanomials to obtain efficent linear feedback shift registers together with AND–XOR arrays. In this approach, these binary fields are constructed via irreducible trinomials of degree m with respect to $\mathbb{F}_{2^n}$ which in turn are also constructed via an irreducible trinomial (Hybrid I) or pentanomial (Hybrid II) over $\mathbb{F}_2$. The bit-serial structure to the tower field and applying the bit-parallel structure to the ground field are applied to obtain the hybrid architecture. Three kinds of multipliers (conventional, Hybrid I and Hybrid II) are implemented using the same FPGA device. Since at least one level is constructed via a trinomial instead of a pentanomial, the hybrid multipliers are 10–33% more efficient than the conventional ones according to the post-place-and-route-timing analysis via Xilinx-ISE 7.1.

1 Introduction

In recent years, finite-field arithmetic has been attracting an increased attention of researchers because of its extensive applications in error correction or cryptographic algorithms adopted in the Internet and wireless communication systems. Especially, finite-field multiplication always plays a central role in directly determining the efficiency of public key schemes, such as elliptic curve cryptosystems (ECC). Therefore it is imperative to design the multipliers with high efficiency.

Our approach focuses on the digit-serial multipliers over the binary composite fields $\mathbb{F}_{2^{nm}}$ with polynomial basis representation. It is well-known that composite fields, namely $\mathbb{F}_{2^{nm}}$, yield efficient implementations if the basis is chosen wisely. They can be constructed via low Hamming weight irreducible polynomials of degree nm, such as trinomials or pentanomials, so that the field elements can be represented via the corresponding polynomial basis over $\mathbb{F}_2$. Alternatively, it can be constructed via a trinomial or pentanomial of degree m, and the field elements can be represented via its corresponding polynomial basis over the ground field $\mathbb{F}_{2^n}$, which in turn can also be constructed by a simple irreducible polynomial of degree n. One can derive either the traditional digit-serial multiplier with digit size n according to the first method or the composite field multiplier according to the second one [1].

FPGA technology provides the designers with more flexibility and manoeuvrability at the bit level, particularly suitable for the applications with wide operand sizes. For efficient FPGA implementation of the binary field multiplier, the designer not only needs to consider the circuit complexity in terms of gate count but also needs to consider other issues such as wire density and regularity of circuit structure. There have been several published literature sources discussing the circuit complexity and latency of both multipliers from the point of view of ASICs [1–5], whereas we are the first to compare these two architectures in terms of FPGA implementations. We derive tables for selected composite field $\mathbb{F}_{2^{nm}}$ which can be constructed via a pentanomial of degree nm but not a trinomial. However, for these composite degrees, there exist at least one irreducible trinomial for either $\mathbb{F}_{2^m}$ or $\mathbb{F}_{2^n}$. Accordingly, we implement both conventional multipliers and hybrid multipliers for these composite fields listed in the tables. Both designs are ported to the FPGA device, Xilinx xc2v1000-5-ff896 for comparisons in terms of timing and area.

Section 2 briefly reviews the structure of the binary composite fields and the construction method. Section 3 introduces both the traditional digit-serial multiplier and the composite field multiplier. Section 4 focuses on the FPGA implementations for both designs. Comparisons of performance and cost are demonstrated. Finally, conclusions are given in Section 5.

2 Mathematical background

We first introduce some notations in Fig. 1. Let α denote the generator of the polynomial basis of $\mathbb{F}_{2^{nm}}$ with respect to $\mathbb{F}_2$. Let β denote the generator of the polynomial basis of $\mathbb{F}_{2^n}$ with respect to $\mathbb{F}_2$. And let γ denote the generator of the polynomial basis of $\mathbb{F}_{2^m}$ with respect to $\mathbb{F}_2$.

There always exist irreducible trinomials or pentanomials for the field size $n \leq 10\ 000$ which can be exploited to

doi:10.1049/iet-cdt:20060168

Paper first received 30th September 2006 and in revised form 3rd June 2007

C. Shu and K. Gaj are with the Department of Electrical & Computer Engineering, George Mason University, 4400 University Drive, Fairfax, VA 22030-4444, USA

S. Kwon is with the Department of Mathematics, Sungkyunkwan University, Suwon 440-746, Korea

E-mail: cshu@gmu.edu

$$\begin{tikzcd}
 & \boxed{F_{2^{nm}}} & \\
\boxed{F_{2^n}} \arrow[ur, "\gamma"] & & \boxed{F_{2^m}} \arrow[ul, dash, dashed] \\
 & \boxed{F_2} \arrow[uu, "\alpha"'] \arrow[ul, "\beta"] \arrow[ur, dash, dashed, "\gamma"'] &
\end{tikzcd}$$

Fig. 1 *Composite field structure*

construct the binary fields with respect to $\mathbb{F}_2$ [6]. In our approach, we assume that α is the root of an irreducible pentanomial of degree nm. The conventional representation is to use the standard basis constructed by α, namely $B_s = \{1, \alpha, \alpha^2, \ldots, \alpha^{nm-1}\}$.

For the composite field $\mathbb{F}_{2^{nm}}$ where $\gcd(n, m) = 1$, the field elements can be represented in a different way. Let β and γ are the roots of irreducible trinomials or pentanomials of degree of n and m separately, then γ can be raised up to generate the polynomial basis, namely $B_t = \{1, \gamma, \gamma^2, \ldots, \gamma^{m-1}\}$, for the tower field $\mathbb{F}_{(2^n)^m}$ with respect to $\mathbb{F}_{2^n}$. β can be used to construct the polynomial basis, namely $B_g = \{1, \beta, \beta^2, \ldots, \beta^{n-1}\}$ for the ground field $\mathbb{F}_{2^n}$ with respect to $\mathbb{F}_2$. The proof of this theorem can be found in [7]. For instance, the elements of $\mathbb{F}_{2^{2\times 5}}$ can be represented via B_s generated by α, where α is the root of $f_s(x) = x^{10} + x^4 + x^3 + x + 1$. Alternatively, the field elements can be represented via B_t generated by γ and B_g generated by β, where γ is the root of $f_t(x) = x^2 + x + 1$ and β is the root of $f_g(x) = x^5 + x^2 + 1$. Then γ and β can be representedvia α (up to conjugates) as $\gamma = \alpha^9 + \alpha^6 + \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2$ and $\beta = \alpha^6 + \alpha^4 + \alpha^3 + \alpha^2$.

For some degrees nm, pentanomials are the lowest Hamming weight irreducible polynomials to construct $\mathbb{F}_{2^{nm}}$. However, it is still possible to obtain a more efficient digit-serial multiplier by adopting the second representation because there may exist at least one irreducible trinomial of degree n or m. In Table 1, we list generating pentanomials $f_s(x)$ for large fields and generating trinomials $f_t(x)$ and $f_g(x)$ for both tower fields and ground fields. In Table 2, the generating polynomial for $\mathbb{F}_{2^n}$ is a pentanomial. The composite exponents are bounded by $80 \le nm \le 300$ because of the requirements of operand sizes for cryptographic applications. These tables are organised as follows: a trinomial $x^n + x^k + 1$, with $n > k > 0$ is represented by the pair n, k. A pentanomial $x^n + x^{k_3} + x^{k_2} + x^{k_1} + 1$, with $n > k_3 > k_2 > k_1 > 0$, is represented by the quadruple n, k_3, k_2, k_1 [6]. For example, the field $\mathbb{F}_{2^{205}}$ has no trinomial basis and a pentanomial basis can be used in this case. The simplest pentanomial basis is generated by the roots of $x^{205} + x^9 + x^5 + x^2 + 1$. However, since $205 = 5 \cdot 41$ and trinomial basis exists for both fields, $\mathbb{F}_{2^5}$ and $\mathbb{F}_{2^{41}}$, we may realise efficient field arithmetic using the trinomial bases. In case $80 \le nm \le 300$, there exists approximately 30 composite fields which can be constructed in the first way and approximately 34 ones which can be constructed in the second way.

Table 1: Selected pentanomials for large fields and trinomials for both tower fields and ground fields

$f_s(x)$ of degree nm	$f_t(x)$ of degree m	$f_g(x)$ of degree n
82, 8, 3, 1	2, 1	41, 3
164, 10, 8, 7	4, 1	
205, 9, 5, 2	5, 2	
246, 11, 2, 1	6, 1	

Table 2: Selected pentanomials for large fields, trinomials for tower fields and pentanomials for ground fields

$f_s(x)$ of degree nm	$f_t(x)$ of degree m	$f_g(x)$ of degree n
96, 10, 9, 6	3, 1	32, 7, 3, 2
160, 5, 3, 2	5, 2	
224, 9, 8, 3	7, 1	
288, 11, 10, 1	9, 1	

3 Conventional and hybrid multipliers

Bit-parallel multiplier which can complete one multiplication in one clock cycle can achieve high operation speed. However, it is not suitable to adopt it directly in the public key cryptosystem such as ECC because of its large circuit complexity in case of large operand sizes. Bit-serial multiplier with an iterative structure sacrifices the operation speed to gain efficiency in terms of area. By combining both structures, we can derive the digit-serial hybrid multiplier for some composite fields as claimed previously [1], that is the bit-serial structure can be applied to the tower field and the bit-parallel structure can be applied to the ground field. On the other hand, we can also derive the digit-serial multiplier using the conventional polynomial basis representation [8–12]. We analyse these two different architectures using concrete examples.

3.1 Conventional digit-serial multiplier

The standard polynomial basis generated by α, B_s (Fig. 1), can be used to represent the elements belonging to $\mathbb{F}_{2^{nm}}$. α can be usually chosen as the root of an irreducible trinomial or pentanomial of degree nm. In our approach (Tables 1 and 2), α is the root of an irreducible pentanomial, that is

$$f_s(\alpha) = \alpha^{nm} + \alpha^{k3} + \alpha^{k2} + \alpha^{k1} + 1 = 0 \qquad (1)$$

Therefore the standard technique for reduction, replacing α^{nm} with $\alpha^{k3} + \alpha^{k2} + \alpha^{k1} + 1$, can be exploited to decrease the circuit complexity.

We use the left-to-right multiplication algorithm to develop the most significant digit-serial multiplier where the digit size is n. In Fig. 2, the two operands a and b and the product c can be represented as follows

$$a = \sum_{i=0}^{nm-1} a_i\alpha^i, \quad b = \sum_{i=0}^{nm-1} b_i\alpha^i, \quad c = \sum_{i=0}^{nm-1} c_i\alpha^i \qquad (2)$$

where a_i, b_i and $c_i \in F_2$.

AND–XOR arrays are adopted to perform the computations of Steps 3–7 in Fig. 2 in parallel. There are two ways to develop such AND–XOR arrays. The first one is to skip the modular operation for each d_j and keep the

```
Require: $a, b \in \mathbb{F}_{2^{nm}}$, and $f_s(x)$
Ensure: $c = a \cdot b \bmod f_s(x)$
 1: $c \leftarrow 0$
 2: for $i = m - 1$ downto 0 do
 3:     $d_{n-1} \leftarrow a_{nm-1}\alpha^{n-1} \cdot b \bmod f_s(x)$
 4:     $d_{n-2} \leftarrow a_{nm-2}\alpha^{n-2} \cdot b \bmod f_s(x)$
 5:     $\cdots$ {Computations of $d_j$ can be done in parallel}
 6:     $d_1 \leftarrow a_{nm-n+1}\alpha \cdot b \bmod f_s(x)$
 7:     $d_0 \leftarrow a_{nm-n} \cdot b \bmod f_s(x)$
 8:     $c \leftarrow (\alpha^n \cdot c \ mod\ f_s(x)) + \sum_{j=0}^{n-1} d_j$
 9:     $a \leftarrow a \ll n$
10: end for
```

Fig. 2 *Digit-serial left-to-right multiplication*

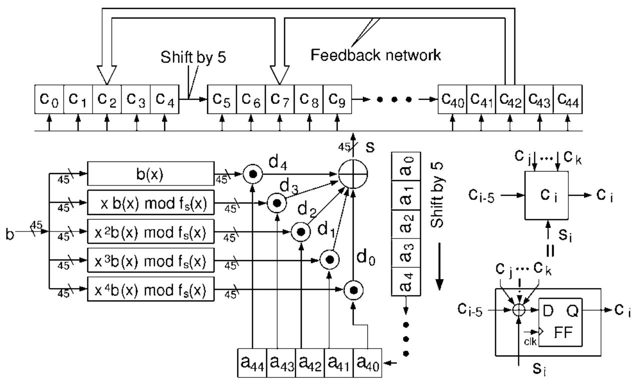

Fig. 3 *Conventional digit-serial multiplier over* $\mathbb{F}_{2^{45}}$, $n = 5$

partial sum as $n + m$ bits and perform the modular operation only once finally. The second one is to perform modular operations for each d_j so that the partial sum are kept as m bits instead of $n + m$. Even though more XOR gates are used, the wire density is decreased which is better for those applications with large operand size. We adopt the second structure. The linear feedback shift register (LFSR) structure is derived from Step 8.

For convenience of understanding, we provide a concrete example which will be also used in the derivation of the hybrid digit-serial multiplier. We choose $n = 5$, $m = 9$ and the generating polynomial of $\mathbb{F}_{2^{45}}$, $f_s(x) = x^{45} + x^4 + x^3 + x + 1$, by which the element, $a \in \mathbb{F}_{2^{45}}$, can be represented as $a = \sum_{i=0}^{44} a_i\alpha^i$, where $a_i \in \mathbb{F}_2$ and α is the root of the irreducible polynomial $f_s(x)$. Then we have (3), which can be used to derive the AND–XOR arrays as well as the LFSR structure for reduction.

$$\begin{aligned} &\alpha^{45} = \alpha^4 + \alpha^3 + \alpha + 1 && \alpha^{46} = \alpha^5 + \alpha^4 + \alpha^2 + \alpha \\ &\alpha^{47} = \alpha^6 + \alpha^5 + \alpha^3 + \alpha^2 && \alpha^{48} = \alpha^7 + \alpha^6 + \alpha^4 + \alpha^3 \\ &\alpha^{49} = \alpha^8 + \alpha^7 + \alpha^5 + \alpha^4 \end{aligned} \tag{3}$$

The architecture is described in Fig. 3 where $\oplus$ denotes XOR gate arrays, and $\odot$ denotes AND gate arrays.

3.2 Hybrid digit-serial multiplier

According to the theorem claimed in Section 2, we can develop a hybrid digit-serial multiplier by applying the bit-serial structure to the tower field and applying the bit-parallel structure to the ground field. In this paper, the tower field is constructed via an irreducible trinomial and the ground field is constructed via an irreducible trinomial or pentnomial. We will illustrate the derivation via the same example used in Section 3.1.

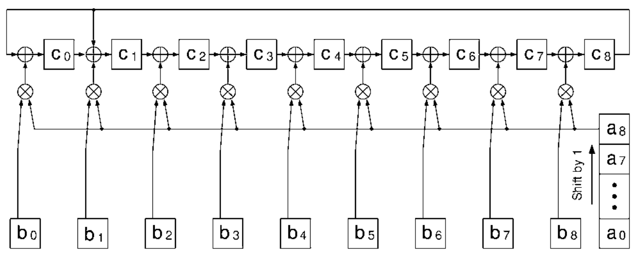

Fig. 4 *Bit-serial structure over* $\mathbb{F}_{(2^5)^9}$ *constructed via* $f_t(x)$

The two operands a and b and the product c in $\mathbb{F}_{2^{nm}}$ can be represented as

$$a = \sum_{i=0}^{m-1} a_i\gamma^i, \quad b = \sum_{i=0}^{m-1} b_i\gamma^i, \quad c = \sum_{i=0}^{m-1} c_i\gamma^i \tag{4}$$

where a_i, b_i and c_i $\mathbb{F}_2^n$. And these coefficients can be represented as

$$a_i = \sum_{j=0}^{n-1} a_{ij}\beta^j, \quad b_i = \sum_{j=0}^{n-1} b_{ij}\beta^j, \quad c_i = \sum_{j=0}^{n-1} c_{ij}\beta^j \tag{5}$$

where a_{ij}, b_{ij} and $c_{ij} \in \mathbb{F}_2$. In our example, $m = 9$ and $n = 5$.

3.2.1 Derivation of the bit-serial structure:

The generating polynomial for the tower field is chosen as

$$f_t(x) = x^9 + x + 1, \qquad \gamma^9 + \gamma + 1 = 0 \tag{6}$$

According to this trinomial together with Fig. 2, we can derive the bit-serial structure for the tower field shown in Fig. 4 where $\oplus$ denotes the adder over $\mathbb{F}_{2^5}$, that is five bitwise XORs, and $\otimes$ denotes the bit-parallel multiplier over $\mathbb{F}_{2^5}$. The modifications of Fig. 2 we need to consider are as follows. The coefficients of a, b and c locate in $\mathbb{F}_{2^n}$ instead of $\mathbb{F}_2$. The generating polynomial is $f_t(x)$ instead of $f_s(x)$, and the digit size is 1 instead of n.

3.2.2 Derivation of the bit-parallel structure:

The generating polynomial for the ground field are chosen as

$$f_g(x) = x^5 + x^2 + 1, \qquad \beta^5 + \beta^2 + 1 = 0 \tag{7}$$

Let a, b and $c = a \cdot b$ denote the elements in $\mathbb{F}_{2^n}$ with the polynomial basis generated by β.

Let a_s, b_t and d_j denote the coefficients in $\mathbb{F}_2$ accordingly where $0 \le s, t \le 4$ and $0 \le j \le 8$. By Mastrovito's method [13, 14]

$$c = \sum_{j=0}^{8} d_j\beta^j \qquad d_j = \sum_{0 \le s,t \le 4}^{s+t=j} a_s b_t \tag{8}$$

where $d_j \in \mathbb{F}_2$. By (7)

$$\begin{aligned} &\beta^5 = \beta^2 + 1 && \beta^6 = \beta^3 + \beta \\ &\beta^7 = \beta^4 + \beta^2 && \beta^8 = \beta^3 + \beta^2 + 1 \end{aligned} \tag{9}$$

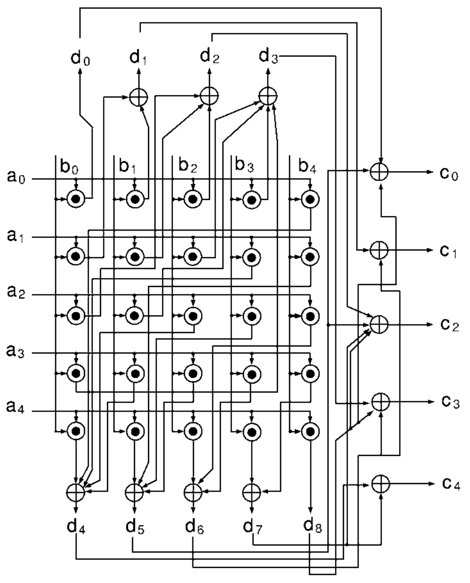

Fig. 5 *Bit-parallel structure over* $\mathbb{F}_{2^5}$ *constructed via* $f_g(x)$

Table 3: Summary of complexity results

Architecture	No. of AND	No. of XOR	Critical path length
Conventional	mn^2	$mn^2 + 2n^2 + 2n$	$T_A + (1 + \lceil \log_2(2n+1) \rceil) T_X$
Hybrid I	mn^2	$mn^2 + mn + n - m$	$T_A + (\lceil \log_2(n+2) \rceil) T_X$
Hybrid II	mn^2	$mn^2 + 3mn + n - 3m$	$T_A + (4 + \lceil \log_2 n \rceil) T_X$

Then we can get the coefficients c_j as follows.

$$\begin{aligned} c_4 &= d_7 + d_4 & c_3 &= d_8 + d_6 + d_3 \\ c_2 &= d_8 + d_7 + d_5 + d_2 & c_1 &= d_6 + d_1 \\ c_0 &= d_8 + d_5 + d_0 \end{aligned} \tag{10}$$

The bit-parallel multiplier over $\mathbb{F}_{2^5}$ is shown in Fig. 5, where $\odot$ denotes AND gate and $\oplus$ denotes XOR gate. More details can also be found in [2, 4].

Next we provide the derivations of the general pentanomial bit-parallel multiplier considering that it is the most complicated case for our hybrid multipliers. Suppose that the partial product $d = \sum_{i=0}^{2n-1} d_j \beta^i$ has been computed via Mastrovito's method. Then we need to perform reductions for those degrees $i \geq n$. If $i - n + k_3 < n$ then we have

$$\begin{aligned} \beta^i &= \beta^{i-n}\beta^n = \beta^{i-n}(\beta^{k_3} + \beta^{k_2} + \beta^{k_1} + 1) \\ &= \beta^{i-n+k_3} + \beta^{i-n+k_2} + \beta^{i-n+k_1} + \beta^{i-n} \end{aligned} \tag{11}$$

If $i - n + k_2 < n \leq i - n + k_3$ then two reductions are necessary.

$$\begin{aligned} \beta^i = {} & \beta^{i-2n+2k_3} + \beta^{i-2n+k_3+k_2} + \beta^{i-2n+k_3+k_1} \\ & + \beta^{i-2n+k_3} + \beta^{i-n+k_2} + \beta^{i-n+k_1} + \beta^{i-n} \end{aligned} \tag{12}$$

If $i - n + k_1 < n \leq i - n + k_2$ then one more reduction is needed.

$$\begin{aligned} \beta^i = {} & \beta^{i-2n+2k_3} + \beta^{i-2n+k_3+k_1} + \beta^{i-2n+k_3} + \beta^{i-2n+2k_2} \\ & + \beta^{i-2n+k_2+k_1} + \beta^{i-2n+k_2} + \beta^{i-n+k_1} + \beta^{i-n} \end{aligned} \tag{13}$$

Similarly, if $n \leq i - n + k_1$ then totally four reductions are necessary.

$$\begin{aligned} \beta^i = {} & \beta^{i-2n+2k_3} + \beta^{i-2n+k_3} + \beta^{i-2n+2k_2} + \beta^{i-2n+k_2} \\ & + \beta^{i-2n+2k_1} + \beta^{i-2n+k_1} + \beta^{i-n} \end{aligned} \tag{14}$$

Since $k_3 < n/2$, at most four reductions are needed. On the basis of the above equations, we can derive the bit-parallel pentanomial multiplier. The same method can be also applied to the bit-parallel trinomial multiplier.

3.3 Complexity analysis for both architectures

We declare some notations used in the following analysis. T_A denotes the delay of a two-input AND gate and T_X denotes the delay of a two-input XOR gate. The complexity comparisons in terms of ASIC estimation are summarised in Table 3, where Hybrid I denotes the hybrid multiplier in which the ground field is constructed via a trinomial and Hybrid II denotes the one in which the ground field is constructed via a pentanomial.

Table 4: Performance and cost comparisons between conventional and hybrid I multipliers

Architecture	*nm*	No. of FF	No. of LUT	No. of CLB slices	Clock period, ns	Latency, ns
Conventional	82	831	3311	1992	7.162	14.32
	164	2065	6307	3630	7.666	30.66
	205	2605	7987	4630	8.275	41.38
	246	3227	9594	5162	9.357	56.14
Hybrid I	82	594	2963	1885	6.837	13.67
	164	1438	5681	3339	6.998	27.99
	205	1686	7101	3916	7.329	36.65
	246	2424	8669	4434	8.695	52.17

Table 5: Performance and cost comparisons between conventional and hybrid II multipliers

Architecture	*nm*	No. of FF	No. of LUT	No. of CLB slices	Clock period, ns	Latency, ns
Conventional	96	1026	2994	1776	7.354	22.06
	160	1701	4833	2895	7.425	37.13
	224	2382	7155	4155	7.631	53.42
	288	3058	8686	5001	10.787	97.08
Hybrid II	96	880	2757	1649	7.218	21.65
	160	1289	4541	2470	6.997	34.99
	224	1879	6335	3430	7.182	50.27
	288	2384	8016	4276	9.992	89.93

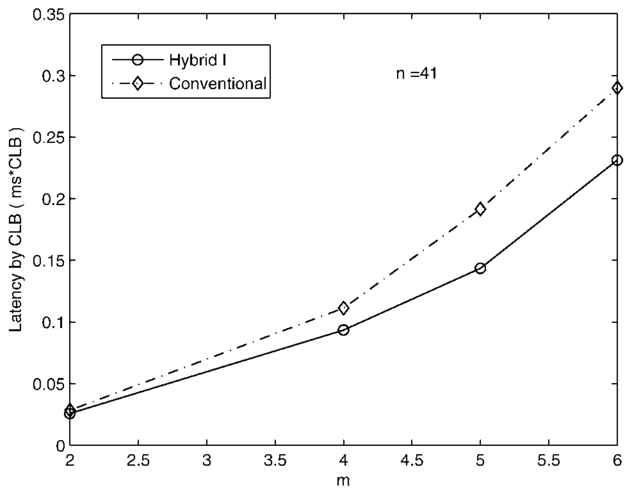

Fig. 6 *Conventional against Hybrid I, latency by area*

For the conventional architecture, the total number of two-input AND gates is mn^2 (n is the digit size) and the number of two-input XOR gates is determined by three parts, feedback network, modular components for $\alpha^i b$ and the partial sum in Step 8 of Fig. 2. For Hybrid I, a bit-parallel multiplier in $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ can be built with at most n^2 AND gates and $n^2 - 1$ XOR gates [2]. For Hybrid II, a bit-parallel multiplier can be built with at most n^2 AND gates and $n^2 + 2n - 3$ XOR gates [5]. The hybrid multiplier contains m bit-parallel multipliers and together with $(m + 1)n$ XOR gates considering that the tower field is constructed via an irreducible trinomial in our approach.

The difference in terms of latency between the conventional and hybrid architectures is not huge. If $m \ll n$, the hybrid architecture is more efficient in terms of area. The wire density due to the feedback network in the hybrid multipliers is decreased. Additionally, more regularity can be gained in the hybrid multipliers since the bit-parallel component has the same structure.

4 FPGA implementations

We choose parameters according to Tables 1 and 2 for our experiments. All multipliers are implemented using a Xilinx xc2v1000-5-ff896 which contains 15 360 slice flip flops, 15 360 4-input look-up tables (LUTs) and 7680 configurable logic blocks (CLBs). Both synthesis, and place and route are completed via Xilinx-ISE 7.1.

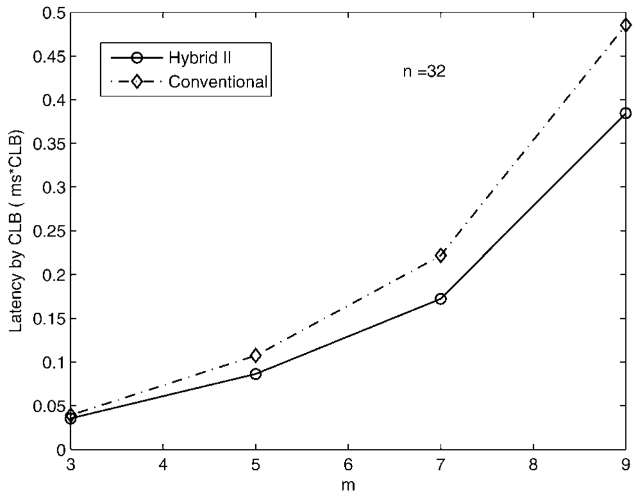

Fig. 7 *Conventional against Hybrid II, latency by area*

The performance comparisons of the FPGA implementation results after placing and routing are summarised in Tables 4 and 5. Comparisons of latency by area are shown in Figs. 6 and 7. The optimisation goal for synthesis is speed instead of area so that the registers are replicated by three times on average to shorten the critical path for both architectures. However, we find that fewer registers are used in the hybrid multipliers than in the conventional ones. This is because that the hybrid one is more regular and the wire density in feedback networks is decreased. In Figs. 6 and 7, we can see that the hybrid multipliers are 10–33% more efficient than the conventional ones in terms of the product of latency by area.

5 Conclusions

Efficient realisation of digit-serial multipliers for binary fields is important to applications such as cryptography and coding theory. It has been claimed that the property of composite fields $\mathbb{F}_{2^{nm}}$ can be used to derive more efficient multipliers; however, to date there has been very little empirical evidence, in particular for FPGA implementations to support this view. In our approach, we concentrate on those composite fields which can be constructed via a pentanomial of degree nm but not a trinomial of degree nm, and in which the tower field is constructed via a trinomial of degree m and the ground field is constructed via either a trinomial or a pentanomial of degree n. We investigate both conventional and hybrid digit-serial multipliers for these special binary composite fields. In case that $m \ll n$, the hybrid multiplier is more efficient than the conventional one in term of gate count. Furthermore, both architectures are implemented using the same FPGA devices. Fewer registers are replicated for shortening the critical path in the hybrid multiplier than in the conventional one because the hybrid architecture is more regular and its wire density in the feedback network is decreased considerably. The hybrid multipliers are generally 10–33% more efficient in terms of the product of latency by area than the conventional ones. Therefore we can conclude that for such special binary composite fields, the hybrid architecture is more efficient in terms of area, regularity and wire density, which make it more suitable for FPGA realisations compared with the conventional architecture.

5 Acknowledgment

The authors would like to thank the anonymous reviewers for their valuable comments.

6 References

1 Paar, C., Fleischmann, P., and Soria-Rodriguez, P.: ‘Fast Arithmetic for public-key algorithms in Galois fields with composite exponents’, *IEEE Trans. Comput.*, 1999, **48**, pp. 1025–1034

2 Wu, H.: ‘Bit-parallel finite field multiplier and squarer using polynomial basis’, *IEEE Trans. Comput.*, 2002, **42**, pp. 750–758

3 Wu, H.: ‘Low complexity bit-parallel finite field arithmetic using polynomial basis’. Int. Workshop on Cryptographic Hardware and Embedded Systems – CHES’99, LNCS 1717, 1999, pp. 280–291

4 Fan, H., and Dai, Y.: ‘Fast bit-parallel $GF(2^n)$ multiplier for all trinomials’, *IEEE Trans. Comput.*, 2005, **54**, pp. 485–490

5 Henriquez, F.R., and Koç, Ç.K.: ‘Parallel multipliers based on special irreducible pentanomials’, *IEEE Trans. Comput.*, 2002, **52**, pp. 1535–1542

6 Seroussi, G.: ‘Table of low-weight binary irreducible polynomials’ (Hewlett Packard, 1998)

7 Menezes, A.J., Blake, I.F., Gao, X., Mullin, R.C., Vanstone, S.A., and Yaghoobian, T.: ‘Application of finite fields’ (Kluwer Academic Publishers, 1993), ISBN 0-7923-928-5

8 Guo, J.-H., and Wang, C.-L.: ‘Digit-serial systolic multiplier for finite fields $GF(2^m)$’, *IEE Proc., Comput. Digital Tech.*, 1998, **145**, (2), pp. 143–148

9 Gustafsson, O.: ‘A digit-serial polynomial basis $GF(2^m)$ multiplier’. Proc. 3rd IEEE Nordic Signal Processing Symp., NORSIG-98, Aalborg, Denmark, June 1998, pp. 173–176

10 Hütter, M., Großschädl, J., and Kamendje, G.A.: ‘A Versatile and scalable digit-serial/parallel multiplier architecture for finite field $GF(2^m)$’. Proc. Int. Conf. on Information Technology: Computers and Communications - ITCC-’03, 2003, vol. 28–30, pp. 692–700

11 Song, L., and Parhi, K.K.: ‘Efficient finite field serial/parallel multiplication’. Proc. 10th IEEE Int. Conf. on Application-Specific Systems, Architectures, and Processors ASAP-96, August 1996 (IEEE Computer Society Press), pp. 72–82

12 Song, L., and Parhi, K.K.: ‘Low-energy digit-serial/parallel finite field multipliers’, *J. VLSI Signal Process.*, 1998, **19**, (2), pp. 149–166

13 Sunar, B., and Koç, Ç.K.: ‘Mastrovito multiplier for all trinomials’, *IEEE Trans. Comput.*, 1999, **48**, pp. 522–527

14 Mastrovito, E.: ‘VLSI design for multipliers for a class of fields $GF(2^k)$’, ‘Lecture Notes in Computer Sciences 357’, (Springer-Verlag, Berlin 1989), pp. 297–309