# Efficient Exponentiation for a Class of Finite Fields $GF(2^n)$ Determined by Gauss Periods

Soonhak Kwon[1], Chang Hoon Kim[2], and Chun Pyo Hong[2]

[1] Inst. of Basic Science and Dept. of Mathematics, Sungkyunkwan University, Suwon 440-746, Korea
shkwon@math.skku.ac.kr

[2] Dept. of Computer and Information Engineering, Daegu University, Kyungsan 712-714, Korea
chkim@dsp.taegu.ac.kr,cphong@daegu.ac.kr

**Abstract.** We present a fast and compact hardware architecture of exponentiation in a finite field $GF(2^n)$ determined by a Gauss period of type $(n, k)$ with $k \geq 2$. Our construction is based on the ideas of Gao et al. and on the computational evidence that a Gauss period of type $(n, k)$ over $GF(2)$ is very often primitive when $k \geq 2$. Also in the case of a Gauss period of type $(n, 1)$, i.e. a type I optimal normal element, we find a primitive element in $GF(2^n)$ which is a sparse polynomial of a type I optimal normal element and we propose a fast exponentiation algorithm which is applicable for both software and hardware purposes. We give an explicit hardware design using the algorithm.

**Keywords:** Finite field, Gauss period, primitive element, exponentiation, optimal normal basis

## 1 Introduction

Arithmetic of finite fields finds various applications in many cryptographic areas these days. Especially, fast exponentiation is very important in such applications as Diffie-Hellman key exchange and pseudo random bit generators. Though exponentiation is the most time consuming and complex arithmetic operation, in some situations such as Diffie-Hellman key exchange, one can devise an efficient exponentiation algorithm since a fixed (primitive) element is raised to many different powers. Let $GF(q^n)$ be a finite field with $q^n$ element where $q$ is a power of a prime and let $g \in GF(q^n)$ be a primitive element (or an element of high multiplicative order). Roughly speaking, the computation of $g^s$ for arbitrary values of $s$ is studied from two different directions. One is the use of precomputation with vector addition chains such as BGMW method [1] and its improvements by Lim and Lee [6] and also by Rooij [7]. The other approach is suggested by Gao et al. [4,5] and it uses a special primitive element called a Gauss period which generates a normal basis for $GF(q^n)$ over $GF(q)$. The BGMW method and its improvements are applicable to arbitrary finite field $GF(q^n)$ and very flexible. On the other hand, an ideal version of BGMW method requires a memory of order $O(n \log q / \log(n \log q))$ values in $GF(q^n)$ and multiplications of order

C.D. Walter et al. (Eds.): CHES 2003, LNCS 2779, pp. 228–242, 2003.

$O(\log(n \log q))$ which amounts to an order of $O(n^2 \log^2 q \log(n \log q))$ bit additions. An algorithm proposed by Gao et al. is not applicable to all finite fields. However, it does not need a precomputation and the complexity of the algorithm is $O(kqn^2)$ additions. Therefore if $q$ is small and if there is a Gauss period of high order of type $(n, k)$ for a small value of $k$, then the method of Gao et al. outperforms the precomputation methods. In this paper, we will discuss an improved algorithm of Gao et al. for a hardware arrangement. We will present a compact and fast hardware architecture for exponentiation using Gauss periods of type $(n, k)$ in $GF(2^n)$ where $k \geq 2$, and detailed explanations will be given for $k = 2, 3$. Also we will give an algorithm for efficient exponentiation in the field determined by an irreducible all one polynomial (AOP). This is possible since we may successfully find a primitive element which is a trinomial of a root of an AOP for most of the cases. Since none of the papers in [1,4,5,6,7] mentions an explicit hardware architecture for exponentiation and since our construction of the circuit has the features of regularity and modularity for VLSI implementation, our result may have possible applications such as smart card purposes.

## 2 Gauss Periods of Type $(n, k)$ in $GF(q^n)$

We will briefly review the theory of Gauss periods and the method of Gao et al.. Let $n, k$ be positive integers such that $p = nk + 1$ is a prime not dividing $q$. Let $K = \langle \tau \rangle$ be a unique subgroup of order $k$ in $GF(p)^\times$. Let $ord_p q$ be the order of $q$ modulo $p$ and assume $gcd(nk/ord_p q, n) = 1$. Let $\beta$ be a primitive $p$th root of unity in $GF(q^{nk})$. Then the the following element

$$\alpha = \sum_{j=0}^{k-1} \beta^{\tau^j} \tag{1}$$

is called a Gauss period of type $(n, k)$ over $GF(q)$. It is well known that $\alpha$ is a normal element in $GF(q^n)$. That is, letting $\alpha_i = \alpha^{q^i}$ for $0 \leq i \leq n-1$, $\{\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_{n-1}\}$ is a basis for $GF(q^n)$ over $GF(q)$. Since $K = \langle \tau \rangle$ is a subgroup of order $k$ in $GF(p)^\times$, a cyclic group of order $p - 1 = nk$, the quotient group $GF(p)^\times / K$ is also a cyclic group of order $n$ and the generator of the group is $qK$. Therefore we have a coset decomposition of $GF(p)^\times$ as a disjoint union,

$$GF(p)^\times = K_0 \cup K_1 \cup K_2 \cdots \cup K_{n-1}, \tag{2}$$

where $K_i = q^i K, 0 \leq i \leq n-1$. Note that any element in $GF(p)^\times$ is uniquely written as $\tau^s q^t$ for some $0 \leq s \leq k-1$ and $0 \leq t \leq n-1$. Now for each $0 \leq i \leq n-1$, we have

$$\begin{aligned} \alpha \alpha_i &= \sum_{s=0}^{k-1} \beta^{\tau^s} \sum_{t=0}^{k-1} \beta^{\tau^t q^i} \\ &= \sum_{s=0}^{k-1} \sum_{t=0}^{k-1} \beta^{\tau^s (1 + \tau^{t-s} q^i)} = \sum_{s=0}^{k-1} \sum_{t=0}^{k-1} \beta^{\tau^s (1 + \tau^t q^i)}. \end{aligned} \tag{3}$$

Notice that there is unique $0 \le u \le k-1$ and $0 \le v \le n-1$ such that $1+\tau^u q^v = 0 \in GF(p)$. If $t \neq u$ or $i \neq v$, then we have $1+\tau^t q^i \in K_{\sigma(t,i)}$ for some $0 \le \sigma(t,i) \le n-1$ depending on $t$ and $i$. Thus we may write $1+\tau^t q^i = \tau^{t'} q^{\sigma(t,i)}$ for some $t'$. Now when $i \neq v$,

$$\begin{aligned} \alpha\alpha_i &= \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(1+\tau^t q^i)} = \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(\tau^{t'} q^{\sigma(t,i)})} \\ &= \sum_{t=0}^{k-1}\sum_{s=0}^{k-1} \beta^{\tau^{s+t'} q^{\sigma(t,i)}} = \sum_{t=0}^{k-1} \alpha^{q^{\sigma(t,i)}} = \sum_{t=0}^{k-1} \alpha_{\sigma(t,i)}. \end{aligned} \tag{4}$$

Also when $i = v$,

$$\begin{aligned} \alpha\alpha_v &= \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(1+\tau^t q^v)} = \sum_{t\neq u}\sum_{s=0}^{k-1} \beta^{\tau^s(\tau^{t'} q^{\sigma(t,v)})} + \sum_{s=0}^{k-1} \beta^{\tau^s(1+\tau^u q^v)} \\ &= \sum_{t\neq u}\sum_{s=0}^{k-1} \beta^{\tau^{s+t'} q^{\sigma(t,v)}} + \sum_{s=0}^{k-1} 1 = \sum_{t\neq u} \alpha^{q^{\sigma(t,v)}} + k = \sum_{t\neq u} \alpha_{\sigma(t,v)} + k. \end{aligned} \tag{5}$$

Therefore $\alpha\alpha_i$ is computed by the sum of at most $k$ basis elements in $\{\alpha_0, \alpha_1, \cdots, \alpha_{n-1}\}$ for $i \neq v$ and $\alpha\alpha_v$ is computed by the sum of at most $k-1$ basis elements and the constant term $k \in GF(q)$. Using these ideas, Gao et al. [4] showed the following.

**Theorem 1.** *Let $\alpha$ be a Gauss period of type $(n,k)$ over $GF(q)$, with $k$ and $q$ bounded. For any $0 \le r < q^n$, $\alpha^r$ can be computed in $O(n^2)$ additions in $GF(q)$.*

*Sketch of Proof.* Write $r = \sum_{j=0}^{n-1} r_j q^j$ with $0 \le r_j < q$. Then the following algorithm gives an output $\alpha^r$.

**Table 1.** An exponentiation algorithm in [4]

```
Input: r = Σ_{j=0}^{n-1} r_j q^j
Output: α^r = Π_{0≤i≤n-1} α_i^{r_i}
A ← 1
for (i = 0 to n − 1 ; i++)
  if r_i ≠ 0
    for (j = 1 to r_i ; j++)
        A ← Aα_i
    end for
  end if
end for
```

Assuming that $q$th Frobenius map $\alpha \to \alpha^q$ is almost free, $A\alpha_i$ is computed by $O(nk)$ additions in $GF(q)$ in a redundant basis $\{\alpha_0, \alpha_1, \cdots, \alpha_{n-1}, 1\}$. For

each $i$, the inner loop $A \leftarrow A\alpha_i$ runs $r_i$ times. Therefore the total number of multiplications $A \leftarrow A\alpha_i$ is $\sum_{i=0}^{n-1} r_i \leq (q-1)n$. Since $A\alpha_i$ is computed by $O(nk)$ additions, one can compute $\alpha^r$ by $O(kqn^2)$ additions in $GF(q)$. □

If above theorem should have any application, it must be guaranteed that the Gauss period $\alpha$ is a primitive element in $GF(q^n)$, or at least is of high order. This is not always satisfied. For example, a Gauss period $\alpha$ of type $(n, 1)$ is never a primitive element since $\alpha^{n+1} = 1$ and $n + 1 << q^n$. However, various computational results imply that the Gauss period $\alpha$ of type $(n, k)$, $k \geq 2$, over $GF(2)$ is very often primitive, and even in the cases that $\alpha$ is not primitive, it usually has a very high multiplicative order. For example, it is known [4] that, among the 177 values of $n \leq 1000$ for which a Gauss period $\alpha$ of type $(n, 2)$ over $GF(2)$ exists, $\alpha$ is a primitive element for 146 values of $n$. Moreover, when $\alpha$ is not primitive, it is usually of very high order. The same table in [4] implies that a Gauss period of type $(n, k)$ over $GF(2)$ is also very often primitive for $k \geq 3$. In the table, it is shown that for approximately 1050 values of $2 \leq n \leq 1200$, there is a primitive Gauss period of type $(n, k)$ for some $k$, and in many cases, one can choose $k < 20$. A theorem supporting this experimental evidence is obtained by Gathen and Shparlinski [17], where it is shown that a Gauss period of type $(n, 2)$ in $GF(q^n)$ has order at least $2^{\sqrt{2n}-2}$ for infinitely many $n$.

## 3 Hardware Arrangements for Exponentiation Using Gauss Periods of Type $(n, k)$ in $GF(2^n)$ for $k \geq 2$

Throughout this section, let us assume that $q = 2$. The algorithm in section 2 is not suitable for a hardware arrangement since one has to multiply different $\alpha_j$ for each step and since the exact number of additions in the coefficients of the expression $A\alpha_j$ is unclear. From now on, instead of using a redundant basis as in section 2, we will always use a normal basis $\{\alpha_0, \alpha_1, \cdots, \alpha_{n-1}\}$ because our approach is more suitable for a unified and simple hardware architecture. Let $A = \sum_{i=0}^{n-1} a_i\alpha_i$ with $a_i \in GF(2)$. Notice that there exist unique $0 \leq u \leq k-1$ and $0 \leq v \leq n-1$ such that $1 + \tau^u 2^v \equiv 0 \pmod{p}$. In this case there is no $0 \leq \sigma(u, v) \leq n-1$ satisfying $0 = 1 + \tau^u 2^v \in K_{\sigma(u,v)}$. Therefore from the equations (4) and (5),

$$A\alpha = \sum_{i=0}^{n-1} a_i\alpha_i\alpha = a_v k + \sum_{\substack{i=0 \\ (t,i)\neq(u,v)}}^{n-1} \sum_{t=0}^{k-1} a_i\alpha_{\sigma(t,i)}. \tag{6}$$

For each $0 \leq i \leq n-1$, letting $t_{ij} = |\{0 \leq t \leq k-1 | 1 + \tau^t q^i \in K_j\}| = |\{0 \leq t \leq k-1 | \sigma(t, i) = j\}|$, we have

$$A\alpha = a_v k + \sum_{i=0}^{n-1} a_i \sum_{j=0}^{n-1} t_{ij}\alpha_j = a_v k + \sum_{j=0}^{n-1} \left( \sum_{i=0}^{n-1} a_i t_{ij} \right) \alpha_j. \tag{7}$$

**Lemma 1.** *If $k$ is even, each coefficient of $A\alpha$ is computed by the sum of at most $k$ $a_i$s, $0 \le i \le n-1$, and if $k$ is odd, it is computed by the sum of at most $k+1$ $a_i$s.*

*Proof.* For each $j$, it is almost clear that the number of $0 \le i \le n-1$ such that $t_{ij} \neq 0$ is at most $k$. If not, there are $k+1$ different $i_s$ with $s = 1, 2, \cdots, k+1$ such that $1 + \tau^{t_s} 2^{i_s} \in K_j$ for some $t_s$, $s = 1, 2, \cdots, k+1$. Since the coset $K_j = 2^j K$ is a set with $k$ elements, there exist $1 \le s \neq l \le k+1$ such that $1 + \tau^{t_s} 2^{i_s} = 1 + \tau^{t_l} 2^{i_l}$, which implies $i_s = i_l$. Therefore $\sum_{i=0}^{n-1} a_i t_{ij}$ is the sum of at most $k$ $a_i$s. Because we have the field $GF(2^n)$ of characteristic two, $a_v k = 0$ if $k$ is even and $a_v k = a_v$ if $k$ is odd. Since the coefficient of $\alpha_j$ in $A\alpha$ is $\sum_{i=0}^{n-1} a_i t_{ij}$ if $k$ is even and $a_v + \sum_{i=0}^{n-1} a_i t_{ij}$ if $k$ is odd, our assertion is verified. □

Now we are ready to give a modified algorithm which is easily applicable to a hardware arrangement.

**Table 2.** A modified exponentiation algorithm for a hardware purpose

---

Input: $r = \sum_{j=0}^{n-1} r_j 2^j$
Output: $\alpha^r$
$A \leftarrow 1$
for $(i = n-1$ to $0$ ; $i--)$
    $A \leftarrow A^2 \alpha^{r_i}$
end for

---

Above algorithm is just a simple form of binary window method which computes

$$\alpha^r = \alpha^{\sum_{i=0}^{n-1} r_i 2^i} = (\cdots(((\alpha^{r_{n-1}})^2 \alpha^{r_{n-2}})^2 \alpha^{r_{n-3}})^2 \cdots)^2 \alpha^{r_0}. \tag{8}$$

Notice that by lemma 1, the operation $A \leftarrow A^2\alpha$ in our algorithm needs at most $k-1$ or $k$ additions under the normal basis expression. One may realize above exponentiation in a linear array circuit consisting of $n$ flip-flops, $n$ 2-1 MUXs and at most $n(k-1)$ or $nk$ (depending on the parity of $k$) XOR gates. The initial value $A = 1$ is loaded in $n$ flip-flops, i.e. we have $a_0 = a_1 = \cdots = a_{n-1} = 1$ initially. The signal of $r = \sum_{j=0}^{n-1} r_j 2^j$ is loaded serially in descending order. That is, $r_0, \cdots, r_{n-2}, r_{n-1} \longrightarrow$. Since $A \leftarrow A^2$ is free in a hardware arrangement (just a rewiring), $A \leftarrow A^2\alpha^{r_i}$ is computed at most $k-1$ or $k$ additions for each coefficient. This operation can be done in one clock cycle. Namely, at $i$th clock cycle, all the coefficients of $A^2$ and $A^2\alpha$ are loaded as input values of the MUXs where the control signal is $r_{n-i}$. Therefore if $r_{n-i} = 0$, then $A^2$ is selected, and if $r_{n-i} = 1$, then $A^2\alpha$ is selected. Let us remind that XOR is a 2-input XOR gate and MUX is a 2-1 multiplexer. Also $D_X$ is the delay time of a XOR and $D_M$ is the delay time of a MUX.

**Proposition 1.** *Let $\alpha$ be a Gauss period of type $(n,k)$, $k \geq 2$, in $GF(2^n)$. Let $r = \sum_{i=0}^{n-1} r_i 2^i$ with $r_i = 0, 1$. Then,*
*(a) we can construct a linear array which computes $\alpha^r$ using $n$ flip-flops, $n$ 2-1 MUXs, and at most $n(k-1)$ XOR gates if $k$ is even, at most $nk$ XOR gates if $k$ is odd.*
*(b) Each coefficient of $\alpha_j$ consists of an XOR tree with at most $k-1$ or $k$ XOR gates. Thus the depth of each XOR tree is at most $\lceil \log_2 k \rceil$ and the critical path delay of our architecture is $\lceil \log_2 k \rceil D_X + D_M$.*

We present the design of the circuit in Fig. 1. Note that we get the result $\alpha^r$ after $n$ clock cycles and at each $i$th clock cycle, $A^2$ and $A^2\alpha$ are simultaneously computed and pass through MUX to get the correct value $A \leftarrow A^2\alpha^{n-i}$.

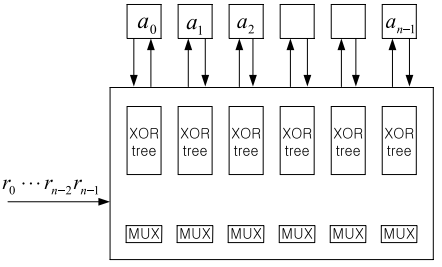


**Fig. 1.** A circuit for exponentiation $\alpha^r$ using a type $(n,k)$ Gauss period in $GF(2^n)$

To show the power of our architecture, which is a linear array but involves many parallel computations, let us think of a finite field $GF(2^{1188})$. It is known [4] that the lowest complexity primitive Gauss period in $GF(2^{1188})$ is of type $(1188, 19)$, i.e. $k = 19$. In this case, our architecture needs 1188 flip-flops and MUXs, and at most 21384 XOR gates. But the critical path delay is only $\lceil \log_2 k \rceil D_X + D_M = 5D_X + D_M$. When $n = 1194$, we have a primitive Gauss period of type $(1194, 2)$ in $GF(2^{1194})$. Thus we need only 1194 XOR gates and the critical path delay is $D_X + D_M$. It should be mentioned that a linear array for exponentiation is proposed by Wu and Hasan [15] using a polynomial basis. Though their method is quite efficient, the complexity and the structure of the design heavily depends on the choice of primitive irreducible polynomial. However our array provides high flexibility and modularity with respect to field size $n$. In the following subsections, we will discuss the circuits of Gauss periods of type $(n,2)$ and $(n,3)$ which have low computational complexity. In these cases, the exact number of necessary gates will be determined rather easily.

### 3.1 Optimal Normal Basis of Type II over $GF(2)$

Let $\alpha = \beta + \beta^{-1}$ be a Gauss period of type $(n,2)$ in $GF(2^n)$, where $2n+1 = p$ is a prime and $\beta$ is a primitive $p$th root of unity in $GF(2^{2n})$. It is also called an optimal normal element of type II and $\{\alpha_0, \alpha_1, \cdots, \alpha_{n-1}\}$ is called an optimal normal basis of type II over $GF(2)$. It has the lowest complexity in the sense that

the sum of the number of nonzero terms in the expression of $\alpha\alpha_i$ for $0 \le i \le n-1$ is minimal, which is $2n-1$. Since $\{1, 2, \cdots, 2n\}$ and $\{\pm 1, \pm 2, \cdots, \pm n\}$ are the same reduced residue system $\pmod p$, we easily find $\{\alpha_0, \alpha_1, \cdots, \alpha_{n-1}\}$ and $\{\beta+\beta^{-1}, \beta^2+\beta^{-2}, \cdots, \beta^n+\beta^{-n}\}$ are same sets. Letting $\alpha'_s = \beta^s + \beta^{-s}, 1 \le s \le n$, it is clear that $\alpha\alpha'_i = (\beta+\beta^{-1})(\beta^i+\beta^{-i}) = \alpha'_{i-1} + \alpha'_{i+1}$. A multiplication table can be constructed easily using above property. Or we may use the self dual property of a Gauss period of type $(n, k)$ for even $k$. We say that two bases $\{\beta_1, \beta_2, \cdots, \beta_n\}$ and $\{\gamma_1, \gamma_2, \cdots, \gamma_n\}$ of $GF(2^n)$ are dual if the trace map, $Tr : GF(2^n) \to GF(2)$, with $Tr(\beta) = \beta + \beta^2 + \cdots + \beta^{2^{n-1}}$, satisfies $Tr(\beta_i\gamma_j) = \delta_{ij}$ for all $1 \le i, j \le n$, where $\delta_{ij} = 1$ if $i = j$, zero if $i \ne j$. A basis $\{\beta_1, \beta_2, \cdots, \beta_n\}$ is said to be self dual if $Tr(\beta_i\beta_j) = \delta_{ij}$. One can directly prove that the Gauss period of type $(n, 2)$ in $GF(2^n)$ generates a self dual normal basis or more generally, one may refer the result in [3] which says that a normal basis of Gauss period of type $(n, k)$ in $GF(2^n)$ is self dual if and only if $k$ is even. Using this self duality, or by a straightforward computation, one can show [20] that

**Lemma 2.** *Let $\beta$ is a primitive pth root of unity in $GF(2^{2n})$ where $p = 2n+1$ is a prime, and let $\alpha = \beta + \beta^{-1}$ be an optimal normal element of type II in $GF(2^n)$. Let $\alpha'_i = \beta^i + \beta^{-i}$ for all $1 \le i \le n$. Let $A = \sum_{i=1}^{n} a_i\alpha'_i$ and $B = \sum_{i=1}^{n} b_i\alpha'_i$ be elements in $GF(2^n)$. Then we have $AB = \sum_{j=1}^{n}(AB)_j\alpha'_j$, where the jth coefficient $(AB)_j$ satisfies*

$$(AB)_j = \sum_{i=1}^{n} b_i(a_{j-i} + a_{j+i}),$$

*where it is defined that $a_0 = 0$, $a_s = a_{-s}$ if $s$ is negative, and $a_s = a_{2n+1-s}$ if $s > n$.*

For our purpose, we only need to know the formula of $A\alpha$ with respect to the basis $\{\alpha'_1, \alpha'_2, \cdots, \alpha'_n\}$. Letting $B = \alpha = \alpha'_1$ in above lemma, we get $b_1 = 1$ and $b_i$ is zero if $i \ne 1$. Thus $(A\alpha)_j = a_{j-1} + a_{j+1}$ for all $1 \le j \le n$. That is,

$$A\alpha = a_2\alpha'_1 + (a_1+a_3)\alpha'_2 + (a_2+a_4)\alpha'_3 + \cdots + (a_{n-2}+a_n)\alpha'_{n-1} + (a_{n-1}+a_n)\alpha'_n. \tag{9}$$

Using this formula and since $\{\alpha_0, \alpha_1, \cdots, \alpha_{n-1}\}$ and $\{\alpha'_1, \alpha'_2, \cdots, \alpha'_n\}$ are same sets where $\alpha_i = \alpha^{2^{i-1}}$ and $\alpha'_i = \beta^i + \beta^{-i}$, we find that the circuit for exponentiation needs exactly $n$ flip-flops, $n$ 2-1 MUXs, and $n-1$ XOR gates.

**Proposition 2.** *Let $\alpha$ be a type II optimal normal element in $GF(2^n)$. Then we can construct a linear array which computes $\alpha^r$ for any $r = \sum_{i=0}^{n-1} r_i 2^i$ with $r_i = 0, 1$ using $n$ flip-flops, $n$ 2-1 MUXs and $n-1$ XOR gates. The critical path delay of our architecture is $D_X + D_M$ and the latency is $n$.*

*Example 1.* Let $p = 11$ and $n = 5$ where the existence of a type II optimal normal element $\alpha = \beta + \beta^{-1}$ is well known. Notice that $\alpha$ is a primitive element. Also note the following correspondence,

$$\alpha_0 = \alpha'_1, \alpha_1 = \alpha'_2, \alpha_2 = \alpha'_4, \alpha_3 = \beta^8 + \beta^{-8} = \alpha'_3, \alpha_4 = \beta^{16} + \beta^{-16} = \alpha'_5, \tag{10}$$

where $\beta^{11} = 1$ is used. Now let $A = a_0\alpha_0 + a_1\alpha_1 + a_2\alpha_2 + a_3\alpha_3 + a_4\alpha_4$. Then $A^2 = a_4\alpha_0 + a_0\alpha_1 + a_1\alpha_2 + a_2\alpha_3 + a_3\alpha_4$. From the correspondence (10), we get $A^2 = a_4\alpha'_1 + a_0\alpha'_2 + a_2\alpha'_3 + a_1\alpha'_4 + a_3\alpha'_5$. Thus from the formula (9),

$$\begin{aligned} A^2\alpha &= A^2\alpha'_1 \\ &= a_0\alpha'_1 + (a_2 + a_4)\alpha'_2 + (a_0 + a_1)\alpha'_3 + (a_2 + a_3)\alpha'_4 + (a_1 + a_3)\alpha'_5 \\ &= a_0\alpha_0 + (a_2 + a_4)\alpha_1 + (a_2 + a_3)\alpha_2 + (a_0 + a_1)\alpha_3 + (a_1 + a_3)\alpha_4. \end{aligned} \tag{11}$$

The basis of our circuit is $\{\alpha_0, \alpha_1, \cdots, \alpha_{n-1}\}$, and at each $i$th clock cycle, the serial input $r_{n-i}$ selects via MUX one of the two values, $A^2$ or $A^2\alpha$. This is realized in the following circuit shown in Fig. 2.

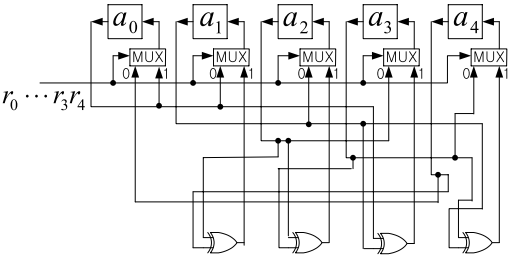


**Fig. 2.** A circuit for exponentiation $\alpha^r$ using a type II optimal normal element in $GF(2^n)$ for $n = 5$

### 3.2 Gauss Period of Type $(n, 3)$ over $GF(2)$

Let $3n + 1 = p$ is a prime and $\beta$ is a primitive $p$th root of unity in $GF(2^{3n})$. Let $\alpha = \beta + \beta^{\tau} + \beta^{\tau^2} \in GF(2^n)$ be a Gauss period of type $(n, 3)$ where $\tau$ is a generator of the unique cyclic subgroup $K$ of order 3 in $GF(p)^{\times}$. Note that there is unique $u$ and $v$ such that

$$1 + \tau^u 2^v = 0 \in GF(p). \tag{12}$$

Also notice $v \neq 0$ because $-1 \notin K = \langle\tau\rangle$. We claim that $v$ is a unique integer satisfying

$$1 + \tau, 1 + \tau^2 \in K_v = 2^v K. \tag{13}$$

Since $\tau$ is an element of order 3 in $GF(p)^{\times}$, we get

$$\tau^2 + \tau + 1 = 0. \tag{14}$$

Thus by (12) and (14),

$$\tau + \tau^2 = -1 = \tau^u 2^v, \tag{15}$$

which implies

$$1 + \tau = \tau^{u-1}2^v, \text{ and } 1 + \tau^2 = \tau^2(1 + \tau) = \tau^{u+1}2^v. \tag{16}$$

Therefore the equation (13) is verified. Now from the equation (5),

$$\alpha\alpha_v = \sum_{t\neq u} \alpha^{2^{\sigma(t,v)}} + 3 = \alpha_{\sigma(t_1,v)} + \alpha_{\sigma(t_2,v)} + 1, \tag{17}$$

where $t_1, t_2 \neq u$. We claim that $\sigma(t_1, v) \neq \sigma(t_2, v)$. In fact, more generally we have

**Lemma 3.** *If $i \neq 0, v$, then $\sigma(0,i), \sigma(1,i), \sigma(2,i)$ are all different. If $i = 0$, then $\sigma(1,0) = \sigma(2,0) = v$ and $\sigma(0,0) = 1$. If $i = v$, then $\sigma(t_1, v) \neq \sigma(t_2, v)$.*

*Proof.* The second statement is already proved in view of the equation (13). Now suppose $i \neq 0, v$. To prove the first statement, we have to show that $1 + 2^i, 1 + \tau 2^i, 1 + \tau^2 2^i$ are in all different cosets of $K$ in $GF(p)^\times$. Suppose on the contrary that there exist $s \neq t$ such that $1 + \tau^s 2^i$ and $1 + \tau^t 2^i$ belong to the same coset. Then we have

$$\frac{1 + \tau^t 2^i}{1 + \tau^s 2^i} \in K = \{1, \tau, \tau^2\}. \tag{18}$$

Since $s \neq t$, $\frac{1+\tau^t 2^i}{1+\tau^s 2^i} = 1$ is impossible. Suppose $\frac{1+\tau^t 2^i}{1+\tau^s 2^i} = \tau$. Then $1 + \tau^t 2^i = \tau + \tau^{s+1} 2^i$. If $t \equiv s + 1 \pmod 3$, we get $\tau = 1$ which is absurd. Therefore $t \equiv s + 2 \pmod 3$ and we get $1 + \tau^{s+2} 2^i = \tau + \tau^{s+1} 2^i$. Thus we get $2^i = \frac{\tau - 1}{\tau^{s+2} - \tau^{s+1}} = \tau^{-s-1} \in K$, which is a contradiction since $0 < i \leq n - 1$ and $n$ is the least positive integer satisfying $2^n \in K$. Now suppose $\frac{1+\tau^t 2^i}{1+\tau^s 2^i} = \tau^2$. Then $\frac{1+\tau^s 2^i}{1+\tau^t 2^i} = \tau^{-2} = \tau$ and the same technique can be applied. The proof of the last statement is also same. □

From lemma 3, the multiplication structure of $\alpha\alpha_i$ is completely determined. That is, when $i = 0$, $\alpha\alpha_0 = \alpha_1$ consists of one basis element. For $i = v$, we have $\alpha\alpha_v = \alpha_{\sigma(t_1,v)} + \alpha_{\sigma(t_2,v)} + 1$ with $\sigma(t_1, v) \neq \sigma(t_2, v)$. And for $i \neq 0, v$, we get $\alpha\alpha_i = \alpha_{\sigma(0,i)} + \alpha_{\sigma(1,i)} + \alpha_{\sigma(2,i)}$ where all the summands are different. Therefore, except for the constant term 1 in the expression $\alpha\alpha_v$, the number of elements which are in the summands of $\alpha\alpha_i$, $0 \leq i \leq n - 1$ is exactly $3(n - 1)$. On the other hand, $\alpha_v$ appears only once as a summand of $\alpha\alpha_i$ for some $i$ since two $\alpha_v$ in the expression of $\alpha\alpha_0$ are cancelled each other. Moreover $\alpha_0$ appears twice as a summand of $\alpha\alpha_i$ for two different values of $i$. This is because we have only two different pairs of $(t, i)$ satisfying $1 + \tau^t 2^i \in K$, i.e. $1 + \tau^t 2^i = 1$ is never satisfied. Since the proof of lemma 1 says that $\alpha_j$ appears at most 3 times as a summand of $\alpha\alpha_i$, $0 \leq i \leq n - 1$, we conclude that $\alpha_j$ $(j \neq 0, v)$ appears exactly 3 times as a summand of $\alpha\alpha_i$. Letting $A = \sum_{i=0}^{n-1} a_i \alpha_i$, the multiplication structure of $A\alpha$ in the equation (7) says,

$$A\alpha = \sum_{j=0}^{n-1} \left( a_v + \sum_{i=0}^{n-1} a_i t_{ij} \right) \alpha_j. \tag{19}$$

From the observations on the number of basis element as a summand of $\alpha\alpha_i$, we conclude that $a_v + \sum_{i=0}^{n-1} a_i t_{ij}$ needs 2 additions if $j = 0$, one addition if $j = v, \sigma(t_1, v), \sigma(t_2, v)$ and 3 additions if $j \neq 0, v, \sigma(t_1, v), \sigma(t_2, v)$.

**Proposition 3.** *Let $\alpha$ be a Gauss period of type $(n, 3)$ in $GF(2^n)$. Then we can construct a linear array which computes $\alpha^r$ for any $r = \sum_{i=0}^{n-1} r_i 2^i$ with $r_i = 0, 1$ using $n$ flip-flops, $n$ 2-1 MUXs and $3n - 7$ XOR gates. Each coefficient of $\alpha_j$ $(j \neq 0, v, \sigma(t_1, v), \sigma(t_2, v))$ consists of 3 XOR gates, For $j = v, \sigma(t_1, v), \sigma(t_2, v)$, each coefficient consists of one XOR gate, and the coefficient of $\alpha_0$ needs 2 XOR gates. Thus the critical path delay of our architecture is $2D_X + D_M$ and the latency is $n$.*

*Example 2.* Let $p = 19$ and $n = 6$ where a Gauss period $\alpha$ of type $(6, 3)$ in $GF(2^6)$ exists and is primitive. In this case, the unique cyclic subgroup of order 3 in $GF(19)^\times$ is $K = \{1, 7, 11\}$. Let $\beta$ be a primitive 19th root of unity in $GF(2^{18})$. Thus letting $\tau = 7$, $\alpha$ is written as $\alpha = \beta + \beta^7 + \beta^{11}$. The computations of $\alpha\alpha_i$, $0 \leq i \leq 5$ is easily done from the following table. For each block regarding $K$ and $K'$, $(s, t)$ entry with $0 \leq s \leq 2$ and $0 \leq t \leq 5$ denotes $\tau^s 2^t$ and $1 + \tau^s 2^t$ respectively.

**Table 3.** Computation of $K_i$ and $K'_i$

| $K_0$ | $K_1$ | $K_2$ | $K_3$ | $K_4$ | $K_5$ | $K'_0$ | $K'_1$ | $K'_2$ | $K'_3$ | $K'_4$ | $K'_5$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 4 | 8 | 16 | 13 | 2 | 3 | 5 | 9 | 17 | 14 |
| 7 | 14 | 9 | 18 | 17 | 15 | 8 | 15 | 10 | 0 | 18 | 16 |
| 11 | 3 | 6 | 12 | 5 | 10 | 12 | 4 | 7 | 13 | 6 | 11 |

From above table, we easily deduce

$$\alpha\alpha = \alpha_1, \qquad \alpha\alpha_1 = \alpha_1 + \alpha_2 + \alpha_5, \quad \alpha\alpha_2 = \alpha_0 + \alpha_4 + \alpha_5, \tag{20}$$

$$\alpha\alpha_3 = \alpha_2 + \alpha_5 + 1, \quad \alpha\alpha_4 = \alpha_2 + \alpha_3 + \alpha_4, \quad \alpha\alpha_5 = \alpha_0 + \alpha_1 + \alpha_4. \tag{21}$$

For example, see the block $K'_2$ for the expression of $\alpha\alpha_2$. The entries of $K'_2$ are $5, 10, 7$. Now see the blocks of $K_i s$ and find $5 \in K_4, 10 \in K_5, 7 \in K_0$. Thus we get $\alpha\alpha_2 = \alpha_4 + \alpha_5 + \alpha_0$. Note that $v = 3$ and $\sigma(t_1, v), \sigma(t_2, v) = 2, 5$ in our example. Let $A = \sum_{i=0}^{5} a_i \alpha_i$ be an element in $GF(2^6)$. Then $A^2 = \sum_{i=0}^{5} a_i \alpha_{i+1}$ where $\alpha_6$ is understood as $\alpha_0$. Thus

$$\begin{aligned} A^2\alpha &= \sum_{i=0}^{5} a_i \alpha_{i+1} \alpha \\ &= a_0(\alpha_1 + \alpha_2 + \alpha_5) + a_1(\alpha_0 + \alpha_4 + \alpha_5) + a_2(\alpha_2 + \alpha_5 + 1) \\ &\quad + a_3(\alpha_2 + \alpha_3 + \alpha_4) + a_4(\alpha_0 + \alpha_1 + \alpha_4) + a_5\alpha_1 \\ &= a_2 + (a_1 + a_4)\alpha_0 + (a_0 + a_4 + a_5)\alpha_1 + (a_0 + a_2 + a_3)\alpha_2 \\ &\quad + a_3\alpha_3 + (a_1 + a_3 + a_4)\alpha_4 + (a_0 + a_1 + a_2)\alpha_5 \\ &= (a_1 + a_2 + a_4)\alpha_0 + (a_0 + a_2 + a_4 + a_5)\alpha_1 + (a_0 + a_3)\alpha_2 \\ &\quad + (a_2 + a_3)\alpha_3 + (a_1 + a_2 + a_3 + a_4)\alpha_4 + (a_0 + a_1)\alpha_5. \end{aligned} \tag{22}$$

Thus the exponentiation algorithm is realized in the following circuit.

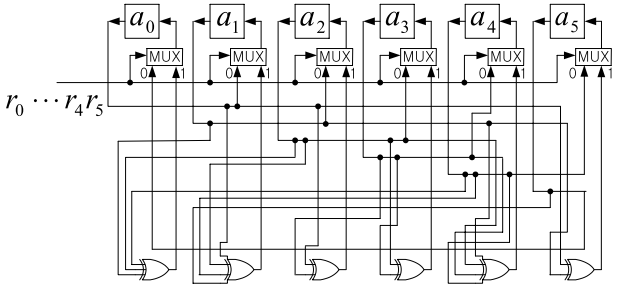


**Fig. 3.** A circuit for exponentiation $\alpha^r$ using Gauss period of type $(n, 3)$ in $GF(2^n)$ for $n = 6$

## 4 Primitive Elements Spanned by an Optimal Normal Basis of Type I over $GF(2)$

Let $\alpha$ be a Gauss period of type $(n, 1)$ over $GF(2)$, where $n + 1 = p$ is a prime. $\alpha$ is also called a type I optimal normal element and the corresponding normal basis is called a type I optimal normal basis. Note that $\alpha$ is never primitive and has a very low order, i.e. $\alpha^{n+1} = 1$ where $n+1 << 2^n - 1$. Therefore one cannot use the algorithms in Table 1, 2 for exponentiation for a practical purpose. On the other hand, it should be noticed that there are not so few primitive elements in arbitrary finite field $GF(2^n)$. That is, the number of primitive elements in $GF(2^n)$ is $\phi(2^n - 1)$, where $\phi(x)$ is Euler's *phi*-function. Thus, the probability for a randomly chosen element $\alpha \in GF(2^n)^\times$ to be a primitive element is

$$\phi(2^n - 1)/(2^n - 1) = \prod_{q|2^n-1} \left(1 - \frac{1}{q}\right), \tag{23}$$

where the product runs through all primes $q$ dividing $2^n - 1$. As long as $2^n - 1$ is not a product of many small prime factors, which is a necessary condition to avoid the Pohlig-Hellman attack for discrete logarithm problem, the probability is not so small. In fact, the following formula for average value of the probability is well known [25],

$$\sum_{n=1}^{N} \phi(n)/n = \frac{6}{\pi^2} N + O(\log N). \tag{24}$$

Of course, our choice of $\alpha$ is not a randomly chosen element. Though $\alpha$ is not a primitive element, we may ask a natural question whether there exists a primitive element which is a sparse polynomial of $\alpha$, for example, a binomial of the form $\alpha^s + \alpha^t$. However it turns out that they are never primitive if $n > 4$. To show this, note that $\alpha^s + \alpha^t = \alpha^s(1 + \alpha^{t-s}) = \alpha^s(1 + \alpha)^{2^j}$ for some $j$. Also from the observation, $(1 + \alpha)^{2^{n/2}} = 1 + \alpha^{2^{n/2}} = 1 + \alpha^{-1} = (1 + \alpha)/\alpha$, we get $\alpha = (1 + \alpha)^{-2^{n/2}+1}$. Therefore, neither $1 + \alpha$ nor $\alpha^s + \alpha^t$ is a primitive element if

$n+1 < 2^{n/2}+1$, i.e. if $n > 4$. The next possible choice is the elements of the form $\alpha^s + \alpha^t + \alpha^l$, or more simply, $1 + \alpha^s + \alpha^t$ because the multiplication by $\alpha$ contributes a negligible order $n+1$. For this type of elements, we could not use the same technique as proving $1+\alpha$ is of low order. In fact we found, by a computation, that a trinomial $1 + \alpha + \alpha^s$ of a type I optimal normal element $\alpha$ in $GF(2^n)$ is always a primitive element for some $s$ with only one exception among all $n \leq 550$.

**Table 4.** List of $n \leq 550$ for which a type I optimal normal element $\alpha$ exists and its corresponding primitive element

| $n$ | primitive element | $n$ | primitive element | $n$ | primitive element |
|---|---|---|---|---|---|
| 4 | $1+\alpha+\alpha^3$ | 138 | $1+\alpha+\alpha^4$ | 372 | $1+\alpha+\alpha^6$ |
| 10 | $1+\alpha+\alpha^3$ | 148 | $1+\alpha+\alpha^5$ | 378 | $1+\alpha+\alpha^3$ |
| 12 | $1+\alpha+\alpha^3$ | 162 | $1+\alpha+\alpha^3$ | 388 | $1+\alpha+\alpha^7$ |
| 18 | $1+\alpha+\alpha^4$ | 172 | $1+\alpha+\alpha^3$ | 418 | $1+\alpha+\alpha^5$ |
| 28 | $1+\alpha+\alpha^2+\alpha^6$ | 178 | $1+\alpha+\alpha^3$ | 420 | $1+\alpha+\alpha^8$ |
| 36 | $1+\alpha+\alpha^6$ | 180 | $1+\alpha+\alpha^5$ | 442 | $1+\alpha+\alpha^3$ |
| 52 | $1+\alpha+\alpha^4$ | 196 | $1+\alpha+\alpha^9$ | 460 | $1+\alpha+\alpha^7$ |
| 58 | $1+\alpha+\alpha^3$ | 210 | $1+\alpha+\alpha^3$ | 466 | $1+\alpha+\alpha^3$ |
| 60 | $1+\alpha+\alpha^3$ | 226 | $1+\alpha+\alpha^3$ | 490 | $1+\alpha+\alpha^7$ |
| 66 | $1+\alpha+\alpha^7$ | 268 | $1+\alpha+\alpha^8$ | 508 | $1+\alpha+\alpha^3$ |
| 82 | $1+\alpha+\alpha^3$ | 292 | $1+\alpha+\alpha^3$ | 522 | $1+\alpha+\alpha^3$ |
| 100 | $1+\alpha+\alpha^7$ | 316 | $1+\alpha+\alpha^7$ | 540 | $1+\alpha+\alpha^7$ |
| 106 | $1+\alpha+\alpha^3$ | 346 | $1+\alpha+\alpha^7$ | 546 | $1+\alpha+\alpha^3$ |
| 130 | $1+\alpha+\alpha^3$ | 348 | $1+\alpha+\alpha^6$ | | |

We used MAPLE for above computation. In the case of $n = 28$, there was no primitive element which is a trinomial of $\alpha$, so we chose the next simple expression. Let $\gamma = 1 + \alpha^s + \alpha^t$ be a fixed primitive element in $GF(2^n)$ where $\alpha$ is a type I optimal normal element. Let $\{1, \alpha, \alpha^2, \cdots, \alpha^n\}$ be an extended AOP (all one polynomial) basis. Then we have the following algorithm which computes $\gamma^r$ using the basis $\{1, \alpha, \alpha^2, \cdots, \alpha^n\}$.

**Table 5.** Exponentiation using $\gamma = 1 + \alpha^s + \alpha^t$ under the extended AOP basis

---

Input: $r = \sum_{j=0}^{n-1} r_j 2^j$
Output: $\gamma^r$
$A \leftarrow 1$
for $(i = n-1$ to $0$ ; $i--)$
    $A \leftarrow A^2 \gamma^{r_i}$
end for

---

Note that above algorithm is applicable for both software and hardware purposes. Though the case $q = 2$ is dealt in above algorithm, one may also use other small

primes $q = 3, 5, \cdots$ for efficient exponentiation. The operation $A \leftarrow A^2$ is free in our basis because $\{1, \alpha, \alpha^2, \cdots, \alpha^n\} = \{1, \alpha, \alpha^{2^1}, \alpha^{2^2}, \cdots, \alpha^{2^{n-1}}\}$. Now letting $A = \sum_{i=0}^{n} a_i \alpha^i$, we get

$$\begin{aligned} A\gamma &= A(1 + \alpha^s + \alpha^t) = A + A\alpha^s + A\alpha^t \\ &= \sum_{i=0}^{n} a_i \alpha^i + \sum_{i=0}^{n} a_{i-s} \alpha^i + \sum_{i=0}^{n} a_{i-t} \alpha^i = \sum_{i=0}^{n} (a_i + a_{i-s} + a_{i-t}) \alpha^i, \end{aligned} \tag{25}$$

where the coefficients $a_i, a_j$ are understood as $a_i = a_j$ if $i \equiv j \pmod{n+1}$ since $\alpha^{n+1} = 1$. Therefore the computation $A \leftarrow A\gamma$ needs 2 additions for each coefficient of $\alpha^i$ $(0 \le i \le n)$ and the total number of bit additions needed to compute $\gamma^r$ is $2n(n+1)$ which is of $O(n^2)$. By following the same ideas of previous section, we find that

**Proposition 4.** *Let $\alpha$ be a type I optimal normal element in $GF(2^n)$ and assume that $\gamma = 1 + \alpha^s + \alpha^t$ is a primitive element for some $s$ and $t$. Then we can construct a linear array which computes $\gamma^r$ for any $r = \sum_{i=0}^{n-1} r_i 2^i$ with $r_i = 0, 1$ using $n+1$ flip-flops, $n+1$ 2-1 MUXs and $2n+2$ XOR gates. The critical path delay of our architecture is $2D_X + D_M$ and the latency is $n$.*

*Example 3.* Let $n = 4$ and let $\alpha$ be a type one optimal normal element, i.e. $\alpha$ is a 5th root of unity over $GF(2)$. It is trivial to show that $\gamma = 1 + \alpha + \alpha^3$ is a primitive element in $GF(2^4)$. Let $A = a_0 + a_1\alpha + a_2\alpha^2 + a_3\alpha^3 + a_4\alpha^4$ be an element in $GF(2^4)$ with respect to the extended AOP basis. From

$$\begin{aligned} A^2 &= a_0 + a_3\alpha + a_1\alpha^2 + a_4\alpha^3 + a_2\alpha^4, \\ A^2\alpha &= a_2 + a_0\alpha + a_3\alpha^2 + a_1\alpha^3 + a_4\alpha^4, \\ A^2\alpha^3 &= a_1 + a_4\alpha + a_2\alpha^2 + a_0\alpha^3 + a_3\alpha^4, \end{aligned} \tag{26}$$

we find

$$\begin{aligned} A^2\gamma &= A^2(1 + \alpha + \alpha^3) \\ &= (a_0 + a_1 + a_2) + (a_0 + a_3 + a_4)\alpha \\ &\quad + (a_1 + a_2 + a_3)\alpha^2 + (a_0 + a_1 + a_4)\alpha^3 + (a_2 + a_3 + a_4)\alpha^4. \end{aligned} \tag{27}$$

From this information, the computation $\gamma^r$, $r = \sum_{i=0}^{n-1} r_i 2^i$ is easily realized in the following circuit.

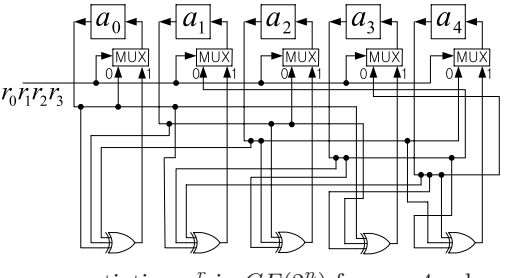


**Fig. 4.** A circuit for exponentiation $\gamma^r$ in $GF(2^n)$ for $n = 4$, where $\gamma$ is a trinomial of a type I optimal normal element $\alpha$

**Table 6.** Comparison with previously proposed exponentiation architectures

| | [10] | [12] | [13] | Fig. 1 |
|---|---|---|---|---|
| latency | $2n^2 + 2n$ | $n(n-1) + \lfloor \frac{n}{2} \rfloor + 1$ | $n(n-1)$ | $n$ |
| critical path delay | $2D_{AND} + 2D_{XOR}$ | $D_{AND} + 2D_{XOR}$ | $2D_{AND} + 2D_{XOR}$ | $\lceil \log_2 k \rceil D_{XOR} + D_{MUX}$ |
| complexity | | | | |
| AND | $4n^2(n-1)$ | $3n^2(n-1)$ | $3n(n-1)$ | 0 |
| XOR | $4n^2(n-1)$ | $3n^2(n-1)$ | $3n(n-1)$ | $kn$ |
| MUX | 0 | 0 | 0 | $n$ |
| flip-flop | $14n^2(n-1)$ | $\frac{9}{2}n^2(n-1)$ | $4n(n-1)$ | $n$ |

## 5 Conclusions

We proposed a compact and fast exponentiation architecture using a Gauss period of type $(n, k)$ in $GF(2^n)$ for all $k$. Using the computational evidence that a Gauss period of type $(n, k)$, $k \geq 2$, is very often primitive and by modifying the multiplication algorithm given by Gao et al., we successfully constructed low complexity arithmetic circuits which have possible applications such as smart cards. Also for the case of a type I optimal normal element, i.e. a Gauss period of type $(n, 1)$, we found primitive elements which yield low complexity multiplication structure and we gave an exponentiation algorithm which is applicable for both software and hardware purposes. We presented explicit designs of the circuits for Gauss periods of type $(n, k)$ when $k = 1, 2, 3$. Table 6 implies that our linear array has many superior properties in terms of latency and gate complexity compared with other existing exponentiation architectures, though our circuit works only for a fixed primitive element. The critical path delay $\lceil \log_2 k \rceil D_{XOR} + D_{MUX}$ in our architecture is not so long since in most of the cases of $n \leq 1200$, we could choose a Gauss period of type $(n, k)$ where $k \leq 32$. That is $\lceil \log_2 k \rceil \leq 5$.

**Acknowledgements.** This work was supported by grant No. R05-2003-000-11325-0 from the Basic Research Program of the Korea Science & Engineering Foundation.

## References

1. E.F. Brickell, D.M. Gordon, K.S. McCurley, and D.B. Wilson, "Fast exponentiation with precomputation," *Eurocrypt 92, Lecture Notes in Computer Science*, vol. 658, pp. 200–207, 1992.
2. T. Beth, B.M. Cook, and D. Gollman, "Architectures for exponentiation in $GF(2^n)$," *Crypto 86, Lecture Notes in Computer Science*, vol. 263, pp. 302–310, 1986.
3. S. Gao, J. von zur Gathen, and D. Panario, "Gauss periods and fast exponentiation in finite fields," *Latin 95, Lecture Notes in Computer Science*, vol. 911, pp. 311–322, 1995.

4. S. Gao, J. von zur Gathen, and D. Panario, "Orders and cryptographical applications," *Math. Comp.*, vol. 67, pp. 343–352, 1998.
5. S. Gao and S. Vanstone, "On orders of optimal normal basis generators," *Math. Comp.*, vol. 64, pp. 1227–1233, 1995.
6. C.H. Lim and P.J. Lee, "More flexible exponentiation with precomputation," *Crypto 94, Lecture Notes in Computer Science*, vol. 839, pp. 95–107, 1994.
7. P. de Rooij, "Efficient exponentiation using precomputation and vector addition chains," *Eurocrypt 94, Lecture Notes in Computer Science*, vol. 950, pp. 389–399, 1994.
8. B. Sunar and Ç.K. Koç, "An efficient optimal normal basis type II multiplier," *IEEE Trans. Computers*, vol 50, pp. 83–87, 2001.
9. A.J. Menezes, I.F. Blake, S. Gao, R.C. Mullin, S.A. Vanstone, and T. Yaghoobian, *Applications of finite fields*, Kluwer Academic Publisher, 1993.
10. C.L. Wang, "Bit level systolic array for fast exponenetiation in $GF(2^m)$," *IEEE Trans. Computers*, vol. 43, pp. 838–841, 1994.
11. P.A. Scott, S.J. Simmons, S.E. Tavares, and L.E. Peppard, "Architectures for exponentiation in $GF(2^m)$," *IEEE J. on Selected Areas in Communications*, vol. 6, pp. 578–586, 1988.
12. S.K. Jain, L. Song, and K.K. Parhi, "Efficient semisystolic architectures for finite field arithmetic," *IEEE Trans. VLSI Syst.*, vol. 6, pp. 101–113, 1998.
13. S.W. Wei, "VLSI architectures for computing exponentiations, multiplicative inverses, and divisions in $GF(2^m)$," *IEEE Trans. Circuits Syst. II*, vol. 44, pp. 847–855, 1997.
14. H. Wu, M.A. Hasan, I.F. Blake, and S. Gao, "Finite field multiplier using redundant representation," *IEEE Trans. Computers*, vol. 51, pp. 1306–1316, 2002.
15. H. Wu and M.A. Hasan, "Efficient exponentiation of a primitive root in $GF(2^m)$," *IEEE Trans. Computers*, vol. 46, pp. 162–172, 1997.
16. J. von zur Gathen and M.J. Nöcker, "Exponentiation in finite fields: Theory and Practice," *AAECC 97, Lecture Notes in Computer Science*, vol. 1255, pp. 88–133, 1997.
17. J. von zur Gathen and I. Shparlinski, "Orders of Gauss periods in finite fields," *ISAAC 95, Lecture Notes in Computer Science*, vol. 1004, pp. 208–215, 1995.
18. G.B. Agnew, R.C. Mullin, I. Onyszchuk, and S.A. Vanstone, "An implementation for a fast public key cryptosystem," *J. Cryptology*, vol. 3, pp. 63–79, 1991.
19. G.B. Agnew, R.C. Mullin, and S.A. Vanstone, "Fast exponentiation in $GF(2^n)$," *Eurocrypt 88, Lecture Notes in Computer Science*, vol. 330, pp. 251–255, 1988.
20. S. Kwon and H. Ryu, "Efficient bit serial multiplication using optimal normal bases of type II in $GF(2^m)$," *ISC 02, Lecture Notes in Computer Science*, vol. 2433, pp. 300–308, 2002.
21. D.M. Gordon, "A survey of fast exponentiation methods," *J. of Algorithms*, vol. 27, pp. 129–146, 1998.
22. W. Geiselmann and D. Gollmann, "VLSI design for exponentiation in $GF(2^n)$," *Auscrypt 90, Lecture Notes in Computer Science*, vol. 453, pp. 398–405, 1990.
23. Ç.K. Koç and B. Sunar, "Low complexity bit parallel canonical and normal basis multipliers for a class of finite fields," *IEEE Trans. Computers*, vol. 47, pp. 353–356, 1998.
24. C. Paar, P. Fleischmann, and P. Roelse, "Efficient multiplier archtectures for Galois fields $GF(2^{4n})$," *IEEE Trans. Computers*, vol. 47, pp. 162–170, 1998.
25. G. Tenenbaum, "*Introduction to analytic and probabilistic number theory*," Cambridge Univ. Press, 1995.