Optimum Digit Serial $GF(2^m)$ Multipliers for Curve Based Cryptography

Sandeep Kumar[1], Thomas Wollinger, and Christof Paar

Communication Security Group (COSY)

Ruhr-Universitaet Bochum, Germany

Abstract

Digit Serial Multipliers are used extensively in hardware implementations of elliptic and hyperelliptic curve cryptography. This contribution shows different architectural enhancements in Least Significant Digit (LSD) multiplier for binary fields $GF(2^m)$. We propose two different architectures, the Double Accumulator Multiplier (DAM) and N-Accumulator Multiplier (NAM) which are both faster compared to traditional LSD multipliers. Our evaluation of the multipliers for different digit sizes gives optimum choices and shows that presently used digit sizes are the worst possible choices. Hence, one of the most important results of this contribution, is that, digit sizes of the form $2^l - 1$, where l is an integer, are preferable for the digit multipliers. Furthermore, one should always use the NAM architecture to get the best timings. Considering the time area product DAM or NAM gives the best performance depending on the digit size.

Keywords: Bit Serial Multiplier, Digit Serial Multiplier, Least Significant Digit Multiplier, Elliptic/Hyperelliptic Curve Cryptography, Public Key Cryptography

I. Introduction

Curve based cryptography, especially elliptic curve cryptography (ECC) [Mil86], [Kob87] and more recently also hyperelliptic curve cryptosystems (HECC) [Kob88], has become increasingly popular in the last few years. These cryptographic primitives are used for exchanging keys over an insecure channel and for digital signatures. Furthermore, these schemes show good properties for software and hardware implementation, because of the relatively short operand length compared to other public-key schemes, like RSA [RSA78]. They are thus often the cryptosystem of choice for the important domain of embedded applications.

There are two types of Finite fields standardized as underlying structure for ECC and HECC: prime fields $GF(p)$ and characteristic two fields $GF(2^m)$. The latter ones are often chosen for hardware realizations due to the smaller hardware circuits required for the corresponding

[1]The author has been partially supported by Sun Microsystems Laboratories, California, USA.

arithmetic. In fact, the over-all time and area complexity of ECC and HECC implementations heavily depends on the finite field multiplier architecture used. Hence, optimizing the multiplier can have major benefits. The most commonly cited implementations of ECC over characteristic two fields in literature [GCE+01], [OP00] use digit multipliers with digit sizes of power of 2. Using a digit multiplier allows implementations to do a tradeoff between speed and area. This allows fast implementations tuned to the available resources of the hardware.

In this paper we present different architectures like Single Accumulator Multiplier (SAM), Double Accumulator Multiplier (DAM) and N-Accumulator Multiplier (NAM) for the implementation of the digit multiplier. The naming is based on the number of internal accumulators used to store the intermediate result. We use these extra accumulators to increase the maximum operating frequency by reducing the critical path delay of the multipliers. We also give the necessary conditions that need to be satisfied by the irreducible polynomial for such implementations. We find that all the standardized NIST polynomials satisfy the required conditions for implementing these techniques. Evaluating the multiplication speed and the area-time product for the different architectures leads to the optimum digit sizes for an implementation. These results show that the cryptosystems can be implemented more efficiently than had been done in the past. For example for NIST B-163, the most optimum architectures are SAM with digit-size=3 and DAM with digit-size=6, giving the developer a good choice between area and time.

The remaining of the paper is organized as follows. Section 2 gives an overview of Digit-Serial Multipliers and conditions for choosing efficient reduction polynomials. Section 3 introduces our different architectures for the Digit-Serial Multiplier. Section 4 summarizes the different architectures and Section 5 presents an evaluation of the different architectures to find optimum digit sizes. Finally, we end this contribution with a discussion of our results and some conclusions.

II. Digit-Serial Multipliers.

Finite field multiplication in $GF(2^m)$ of two elements A and B to obtain a result $C = A \cdot B \bmod p(\alpha)$ (where $p(\alpha)$ is the irreducible polynomial) can be done in various ways based on the available resources. Digit-serial multipliers, introduced in [SP98] for binary fields $GF(2^m)$, are a trade-off between speed, area, and power consumption. This is achieved by processing several of B's coefficients at the same time. The number of coefficients that are processed in parallel is defined to be the digit-size D.

The total number of digits in the polynomial of degree $m-1$ is given by $d = \lceil m/D \rceil$. Then, we can re-write the multiplier as $B = \sum_{i=0}^{d-1} B_i \alpha^{Di}$, where

$$B_i = \sum_{j=0}^{D-1} b_{Di+j} \alpha^j \quad 0 \leq i \leq d-1 \tag{1}$$

assuming B has been appropriately padded with zero coefficients for the most significant bits. Hence, the multiplication can be performed as shown in Algorithm 1.

Algorithm 1 Least Significant Digit (LSD) Multiplier [SP98]

Require: $A = \sum_{i=0}^{m-1} a_i \alpha^i$, where $a_i \in GF(2)$, $B = \sum_{i=0}^{\lceil \frac{m}{D} \rceil - 1} B_i \alpha^{Di}$, where B_i is as in (1)

Ensure: : $C \equiv A \cdot B \bmod p(\alpha) = \sum_{i=0}^{m-1} c_i \alpha^i$, where $c_i \in GF(2)$

1: $C \leftarrow 0$

2: **for** $i = 0$ to $\lceil \frac{m}{D} \rceil - 1$ **do**

3: $\quad C \leftarrow B_i A + C$

4: $\quad A \leftarrow A\alpha^D \bmod p(\alpha)$

5: **end for**

6: Return ($C \bmod p(\alpha)$)

1) Reduction $\bmod p(\alpha)$ *for Digit Multipliers:* In LSD, products of the form $W\alpha^D$ occurs (as seen in Step 4 of Algorithm 1) which have to be reduced $\bmod p(\alpha)$. One can derive equations for the modular reduction for *general* irreducible polynomials $p(\alpha)$. However, it is more interesting to search for polynomials that minimize the complexity of the reduction operation. For determining optimum irreducible polynomials we use two theorems from [SP98].

Theorem 1: Assume that the irreducible polynomial is of the form $p(\alpha) = \alpha^m + p_k \alpha^k + \sum_{j=0}^{k-1} p_j \alpha^j$, with $k < m$. For $t \leq m-1-k$, α^{m+t} can be reduced to a degree less than m in one step with the following equation:

$$\alpha^{m+t} \bmod p(\alpha) = p_k \alpha^{k+t} + (\sum_{j=0}^{k-1} p_j \alpha^{j+t}) \tag{2}$$

Theorem 2: For digit multipliers with digit-element size D, when $D \leq m-k$, the intermediate results in Algorithm 1 (Step 4 and Step 6) can be reduced to degree less than m in one step.

Theorems 1 and 2 implicitly say that for a given irreducible polynomial $p(\alpha) = \alpha^m + p_k \alpha^k + \sum_{j=0}^{k-1} p_j \alpha^j$, the digit-element size D has to be chosen based on the value of k, the degree of the second highest coefficient in the irreducible polynomial.

III. Architecture Options

In this section, we provide different architectural possibilities for the implementation of the LSD multiplier. The architectures are named based on the number of accumulators present in the multiplier.

A. Single Accumulator Multiplier (SAM)

The Single Accumulator Multiplier (SAM) is same as the Song/Parhi multiplier architecture as introduced in [SP98]. This kind of architecture is most commonly used in cryptographic hardware implementations [GCE+01], [OP00]. This architecture consists of three main components as shown in the Fig. 1.

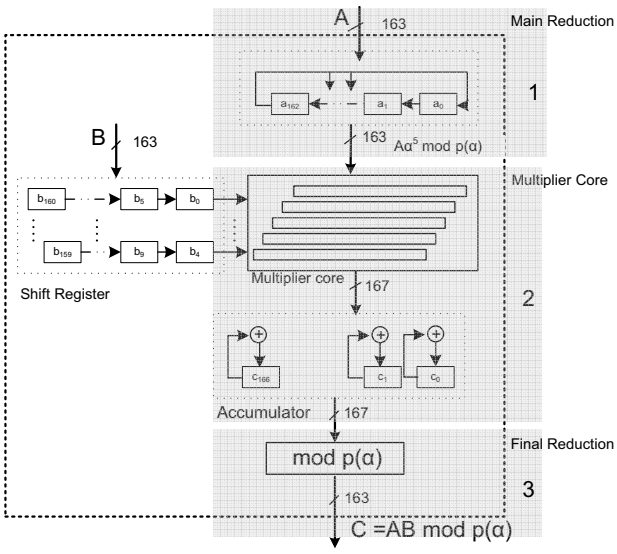

Fig. 1: LSD-Single Accumulator Multiplier Architecture ($D = 5$) for $GF(2^{163})$

- The *main reduction circuit* to shift A left by D positions and to reduce the result $\mod p(\alpha)$ (Step 4, Algorithm 1).
- The *multiplier core* which computes the intermediate C and stores it in the accumulator (Step 3, Algorithm 1).
- The *final reduction circuit* to reduce the contents in the accumulator to get the final result C (Step 6, Algorithm 1).

All the components run in parallel requiring one clock for each step and the critical path of the whole multiplier normally depends on the critical path of the multiplier core.

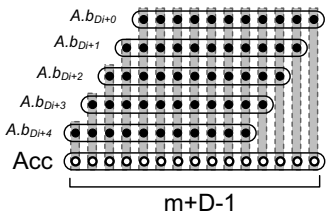

Fig. 2: SAM multiplier core for $D = 5$

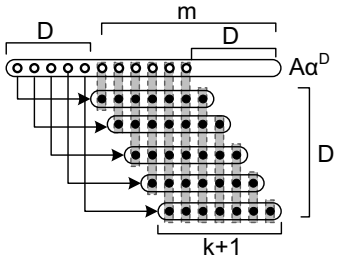

Fig. 3: SAM main reduction circuit for $D = 5$

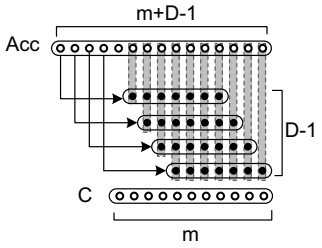

Fig. 4: SAM final reduction circuit for $D = 5$

We give here a further analysis of the area requirements and the critical path of the different components of the multiplier. In the figures, we will denote an AND gate with a filled dot and elements to be XORed by a vertical line over them. The number of the XOR gates and the critical path is based on the binary tree structure that has to be built to XOR the required elements. For n elements, the number of XOR gates required is $n-1$ and the critical path delay comes out to be the binary tree depth $\lceil \log_2 n \rceil$. We calculate the critical path as a function of the delay of one XOR gate (Δ_{XOR}) and for one AND gate (Δ_{AND}). This allows our analysis to be independent of the cell-technology used for the implementation.

1) Multiplier core: The multiplier core performs the operation $C \leftarrow B_i A + C$ (Step 3 Algorithm 1). The implementation of the multiplier core is as shown in Fig. 2 for a digit size $D = 5$. It consists of ANDing the multiplicand A with each element of the digit of the multiplier B and XORing the result with the accumulator Acc and storing it back in Acc. The multiplier core requires mD AND gates (denoted by the black dots), mD XOR gates (for XORing the columns denoted by the vertical line plus the XOR gates for the accumulator) and $m + D - 1$ Flip-Flops (FF) for accumulating the result C.

It can be seen that the multiplier core has a maximum critical path delay of one Δ_{AND} (since all the ANDings in one column are done in parallel) and a delay for XORing $D+1$ elements as shown in Fig. 2. Thus the total critical path delay of the multiplier core is $\Delta_{AND} + \lceil log_2(D+1) \rceil \Delta_{XOR}$.

2) Main reduction circuit: The main reduction circuit performs the operation $A \leftarrow A\alpha^D \bmod p(\alpha)$ (Step 4 Algorithm 1) and is implemented as shown in Fig. 3. Here the multiplicand

A is shifted left by the digit-size D which is equivalent to multiplying by α^D. The result is then reduced with the reduction polynomial by ANDing the higher D elements of the shifted multiplicand with the reduction polynomial $p(\alpha)$ (shown in the figure as pointed arrows) and XORing the result. We assume that the reduction polynomial is chosen according to Theorem 2 which allows reduction to be done in one single step. It can be shown that the critical path delay of the reduction circuit can be at most equal or less than that for the multiplier core.

The main reduction circuit requires $(k+1)$ ANDs and k XORs gates for each reduction element. The number of XOR gates is one less because the last element of the reduction are XORed to empty elements in the shifted A. Therefore a total of $(k+1)D$ AND and kD XOR are needed for D digits. Another m Flip-Flops(FF) are needed to store A and $k+1$ FFs to store the general reduction polynomial.

The critical path of the main reduction circuit (as shown in Fig. 3) is one AND (since the ANDings occur in parallel) and the critical path for summation of the D reduction components with the original shifted A. Thus the maximum possible critical path delay is $\Delta_{AND} + \lceil log_2(D+1) \rceil \Delta_{XOR}$, which is the same as the critical path delay of the multiplier core.

3) Final reduction circuit: The final reduction circuit performs the operation $C \bmod p(\alpha)$, where C is of size $m+D-1$. It is implemented as shown in Fig. 4 which is similar to the main reduction circuit without any shifting. Here the most significant $(D-1)$ elements are reduced using the reduction polynomial $p(\alpha)$ similarly shown with arrows. The area requirement for this circuit is $(k+1)(D-1)$ AND gates and $(k+1)(D-1)$ XOR gates. The critical path of the final reduction circuit is $\Delta_{AND} + \lceil log_2(D) \rceil \Delta_{XOR}$ which is less than that of the main reduction circuit since the degree of the polynomial reduced is one less (Fig. 4).

An r-nomial reduction polynomial satisfying Theorem 2, i.e., $\sum_{i=0}^{k} p_i = (r-1)$, is a special case and hence the critical path is upper-bounded by that obtained for the general case given here. For a fixed r-nomial reduction polynomial, the area for the main reduction circuit is $(r-1)D$ ANDs and $(r-2)D$ XORs. In addition, we require m flip flops to store intermediate result A. However, no flip flops are needed to store the reduction polynomial as it can be hardwired.

Thus the total area is given in Table III and our analysis of the optimum digit size for critical path and area can be found in Section 5.

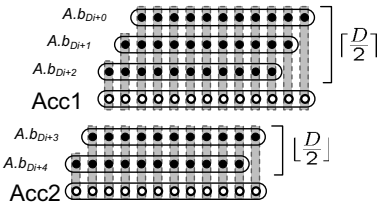

Fig. 5: DAM multiplier core for $D = 5$

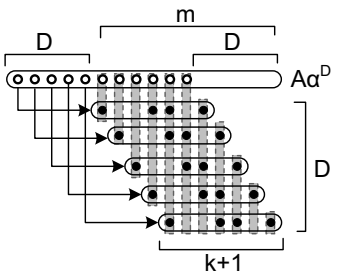

Fig. 6: DAM main reduction circuit for $D = 5$

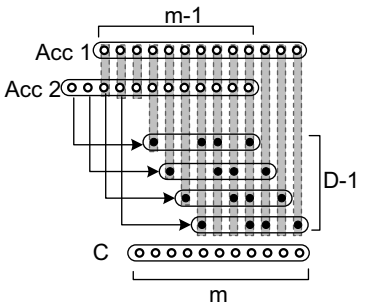

Fig. 7: DAM final reduction circuit for $D = 5$

B. Double Accumulator Multiplier (DAM)

The DAM multiplier is the new variant of the SAM multiplier that we propose. They differ in the multiplier core only. Here we use two accumulators to store the partial product C such that we can reduce the critical path of the multiplier core. The architecture is shown in the Fig. 5. The first accumulator Acc1 adds $\lceil D/2 \rceil$ of the elements and the other accumulator Acc2 adds the remaining $\lfloor D/2 \rfloor$ elements.

Therefore the longest critical path in the DAM core is caused by the part involving Acc1. The delay here is one AND gate (since ANDings occur in parallel) and the delay for accumulating $\lceil D/2 \rceil + 1$ elements. Thus the critical path delay of the multiplier core is $\Delta_{AND} + \lceil log_2(\lceil D/2 \rceil + 1) \rceil \Delta_{XOR}$. The lower delay has an advantage only if the critical path of the other components (reduction circuits) also have a smaller or equal delay. Therefore the conditions on the reduction polynomial are more stringent than in the SAM case. We provide here a theorem which shows how to choose such a reduction polynomial.

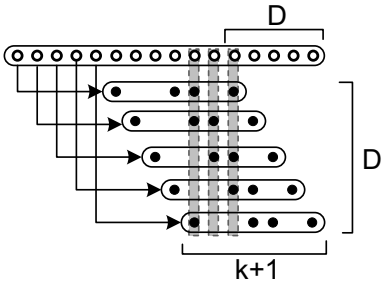

Fig. 8: Overlap case

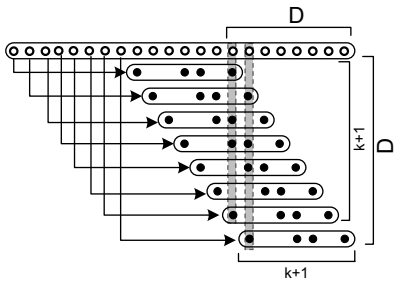

Fig. 9: Underlap case

Theorem 3: Assume an r-nomial irreducible reduction polynomial $p(\alpha) = \alpha^m + p_k\alpha^k + \sum_{i=0}^{k-1} p_i\alpha^i$, with $k \leq m - D$ and $\sum_{i=0}^{k} p_i = (r-1)$. For a digit multiplier implemented using

two accumulators for the multiplication core (DAM), the reduction polynomial $p(\alpha)$ satisfying the following condition can perform reduction with a smaller critical path than the multiplier core:

$$\begin{aligned} &D \leq (m+1)/2: \quad \textstyle\sum_{i=0+j}^{D+j} p_i \leq \lceil D/2 \rceil \quad \text{for } 0 \leq j < m-2D+2 \\ &D > (m+1)/2: \quad (r-1) \leq \lceil D/2 \rceil \end{aligned} \tag{3}$$

Proof: There are two different cases which affect how r can be chosen based on the number of reduction elements in the XOR tree. The first we call the *overlap* case when there are a maximum of D reducing elements in a column (Fig. 8) and the second the *underlap* case where the maximum number of reducing elements in a column is less than D (Fig. 9).

1) Case 1: Overlap: For the reduction elements to overlap as shown in the Fig. 8, the second highest degree of the reduction polynomial k, should satisfy the condition $(k+1) \geq D$. We denote the number of overlapping columns (shown by the shaded lines) as $q = k+1-(D-1) \leq m-2D+2$. If we now analyze each of these columns in the overlapping region, it consists of XORing D consecutive coefficients of $p(\alpha)$ with the shifted A. For example, in the rightmost column it is $(\sum_{i=0}^{D-1} p_i + 1)$ and in the next column it is $(\sum_{i=1}^{D} p_i + 1)$ and so on. Therefore the critical path of the circuit is the maximum critical path of any of these columns and should be less than equal to that of the multiplier core. This can be expressed as

$$\lceil log_2(\textstyle\sum_{i=0+j}^{D-1+j} p_i + 1) \rceil \Delta_{XOR} \leq \lceil log_2(\lceil D/2 \rceil + 1) \rceil \Delta_{XOR} \quad \text{for all } 0 \leq j < m-2D+2 \tag{4}$$

The result in Equation (3) can be easily obtained from this. The implication of the result is that the sum of any D consecutive coefficients in the reduction polynomial that lie in the overlap region should be less than equal to $\lceil D/2 \rceil$.

2) Case 2: Underlap: The reduction elements underlaps as shown in the Fig. 9 for the remaining possible values of k, i.e., $(k+1) < D$. The number of reduction elements being added in the underlap region is not more than k. The maximum number of reduction elements that can be present along any column can be $(r-1)$ (like the shaded columns in the figure) since the reduction polynomial is an r-nomial. The condition on the critical path delay is now

$$\lceil log_2(\textstyle\sum_{i=0}^{k} p_i + 1) \rceil \Delta_{XOR} \leq \lceil log_2(\lceil D/2 \rceil + 1) \rceil \Delta_{XOR} \tag{5}$$

This leads to the condition Equation (3) given in the theorem. This implies that the sum of all the non-zero coefficients (except the highest degree) in the reduction polynomial should be less than equal to $\lceil D/2 \rceil$. ■

In the Table I, we provide the possible digit sizes for NIST recommended ECC reduction polynomials. We see that the only difference in the condition between SAM and DAM is for 163-bit reduction polynomial where the digit-size $D = 2$ is not possible. This shows that NIST curves implemented in the SAM architecture can be easily converted to the DAM architecture and does not require any extra constraints.

TABLE I: NIST recommended reduction polynomial for ECC and digit sizes possible

	possible D	
p(t)	SAM	DAM
$x^{163} + x^7 + x^6 + x^3 + 1$	≤ 156	≤ 156 except $\{2\}$
$x^{233} + x^{74} + 1$	≤ 159	≤ 159
$x^{283} + x^{12} + x^7 + x^5 + 1$	≤ 271	≤ 271
$x^{409} + x^{87} + 1$	≤ 322	≤ 322
$x^{571} + x^{10} + x^5 + x^2 + 1$	≤ 561	≤ 561

The addition of an extra accumulator also increases the size of the multiplier. Therefore we give an exact count of gates for the new multiplier which will allow us to perform a realistic comparison in terms of area-time product with the other multiplier designs. Evaluating the size of the multiplier, the three different components of the multiplier require the following area:

- The *multiplier core* needs mD AND gates and for the XORing we have two accumulators, where accumulator Acc1 needs to XOR $\lceil D/2 \rceil + 1$ elements and accumulator Acc2 needs to XOR $\lfloor D/2 \rfloor + 1$ elements. Hence the total XOR-gates required are $\lceil D/2 \rceil * m + \lfloor D/2 \rfloor * m = mD$.

  We require $(m + \lceil D/2 \rceil - 1) + (m + \lfloor D/2 \rfloor - 1) = 2m + D - 2$ FFs for the two accumulators. Adding up the two accumulators (which is done with the final reduction) requires additional $m - 1$ XORs which is the overlapping region of the two accumulators.
- The *main reduction circuit* area is the same as discussed for SAM r-nomial irreducible polynomial: $(r - 1)D$ AND, $(r - 2)D$ XOR gates and m FF for A.
- *Final reduction* is done using $(r - 1)(D - 1)$ AND and $(r - 1)(D - 1)$ XOR gates.

Remark 1: The addition of the two accumulators is done as part of the final reduction circuit in the last cycle. Since the final reduction circuit has a critical path smaller that of the main reduction circuit, the overall critical path is not greater than that of the multiplier core.

C. N-Accumulator Multiplier (NAM)

The N-Accumulator Multiplier is a more generalized version of the DAM. Here we try to reduce the critical path further (assuming additional conditions on the reduction polynomial) by having multiple accumulators calculating the partial sum C.

Assuming n accumulators summing the partial sum, the largest critical path in the multiplier core is $\Delta_{AND}+\lceil log_2\lceil D/n\rceil+1\rceil\Delta_{XOR}$. The accumulators are themselves XORed using a different tree of critical path $\lceil log_2 n\rceil\Delta_{XOR}$ in an extra clock at the end (hence the overall latency of the multiplier will be increased by one clock cycle). Care has to be taken that the final accumulation critical path is not greater than that of the multiplier core, i.e., $\lceil log_2 n\rceil \leq \lceil log_2\lceil D/n\rceil+1\rceil$. This is true when the number of accumulators is less than equal to the maximum number of elements XORed in any of the accumulators (which is a tighter bound than in the equation).

The condition on the reduction polynomial such that the reduction circuit has lesser critical path delay than the multiplier core is an extension of Theorem 3 as given below.

Theorem 4: Assume that r-nomial reduction polynomial $p(\alpha) = \alpha^m + p_k\alpha^k + \sum_{i=0}^{k-1} p_i\alpha^i$, with $k \leq m-D$ and $\sum_{i=0}^{k} p_i = (r-1)$. For a digit multiplier implemented using n accumulators for the multiplication core (NAM), the reduction polynomial $p(\alpha)$ satisfying the following condition can perform reduction with a smaller critical path than the multiplier core:

$$\begin{aligned} &D \leq (m+1)/2: \quad \textstyle\sum_{i=0+j}^{D-1+j} p_i \leq \lceil D/n\rceil \quad \text{for } 0 \leq j < m-2D+2 \\ &D > (m+1)/2: \quad (r-1) \leq \lceil D/n\rceil \end{aligned} \tag{6}$$

Proof: Similar to the proof for Theorem 3. ■

For the calculation of the area requirement for NAM, we assume that each of the accumulator accumulates $q_i,\ 1 \leq i \leq n$ elements such that $\sum_{i=1}^{n} q_i = D$.

- The multiplier core needs mD AND gates and for the XORing we require $q_1 * m + q_2 * m + \ldots + q_n * m = mD$ XOR gates. The FFs required for the accumulators are $(m+q_1-1)+(m+q_2-1)+\ldots+(m+q_n-1) = nm+D-n$.
- The main reduction is same as before $(r-1)D$ AND, $(r-2)D$ XOR and m FF for A.
- The final reduction is done using $(r-1)(D-1)$ AND and $(r-1)(D-1)$ XOR gates.
- For the final accumulation, any two adjacent accumulators have only $(m-1)$ elements overlapping. Therefore the total number of XORs for the accumulation tree is $(m-1)(n-1)$. An additional $m+D-1$ FF are required to store the result as unlike the DAM, the addition is done in a separate clock cycle.

IV. Summary of the Multiplier Options

In this section we summarize the different architecture options. Table II shows the latency (in clocks) and the critical path of three different architectures assuming the digit-sizes satisfy the required conditions. The latency for NAM is greater due an extra last cycle to sum all the accumulators.

TABLE II: LSD Multiplier: Latency and critical path

	Latency	Critical Path
SAM ($D \geq 2$) r-nomial	$\lceil m/D \rceil + 1$	$1\Delta_{AND} + \lceil \log_2(D+1) \rceil \Delta_{XOR}$
DAM ($D \geq 2$) r-nomial	$\lceil m/D \rceil + 1$	$1\Delta_{AND} + \lceil \log_2(\lceil D/2 \rceil + 1) \rceil \Delta_{XOR}$
NAM ($n \geq 3$, $D > n$) r-nomial	$\lceil m/D \rceil + 2$	$1\Delta_{AND} + \lceil \log_2(\lceil D/n \rceil + 1) \rceil \Delta_{XOR}$

Table III shows the area requirement for the proposed architectures. As expected the area is larger for the new architectures, but the area-time product is better in the DAM and NAM case as will be shown in Section 5.

TABLE III: LSD Multiplier: Area

	# XOR	# AND	# FF
SAM ($D \geq 2$)			
general	$(m+k)D + (k+1)(D-1)$	$(m+k+1)D + (k+1)(D-1)$	$2m + D + k$
r-nomial	$(m+r-2)D + (r-1)(D-1)$	$(m+r-1)D + (r-1)(D-1)$	$2m + D - 1$
DAM ($D > 2$) r-nomial	$(m+r-2)D + (r-1)(D-1)$ $+(m-1)$	$(m+r-1)D + (r-1)(D-1)$	$3m + D - 2$
NAM ($n \geq 3$, $D > n$), r-nomial	$(m+r-2)D + (r-1)(D-1)$ $+(m-1)(n-1)$	$(m+r-1)D + (r-1)(D-1)$	$(n+2)m + 2D$ $-(n+1)$

V. Evaluation of the Implementation Options

We evaluated the NIST B-163 polynomial (which is in wide-spread use in real-world applications) on the multipliers for different digit sizes to find the optimum values. We use the

critical path estimation and latency to calculate the time required for one multiplication. The area is calculated using the estimation we made for each component of the multiplier (tabulated in Table III). Real world estimations are done using the standard cell library from [VLS03].

A. Evaluation of the SAM

Our single accumulator multiplier architecture is same as the digit multipliers used in the open literature. In order to allow an analysis of this multiplier we draw Fig. 10. This figure shows the time taken to complete one multiplication for different digit sizes for SAM.

Concluding from Fig. 10 one realizes, that the digit-size D equals a powers of 2 like $4, 8, 16, 32, 64$ are the local worse values. However, these kind of D values are normally used as digit-sizes for ECC implementations [GCE$^+$01], [OP01].

In addition one can see that values of D of the form $2^l - 1$ are optimum because of the optimum binary tree structure they generate. Since the multiplier is the most important component in these cryptosystems which also dictates the overall efficiency, changing the digit sizes to the more optimum values can give a much better performance.

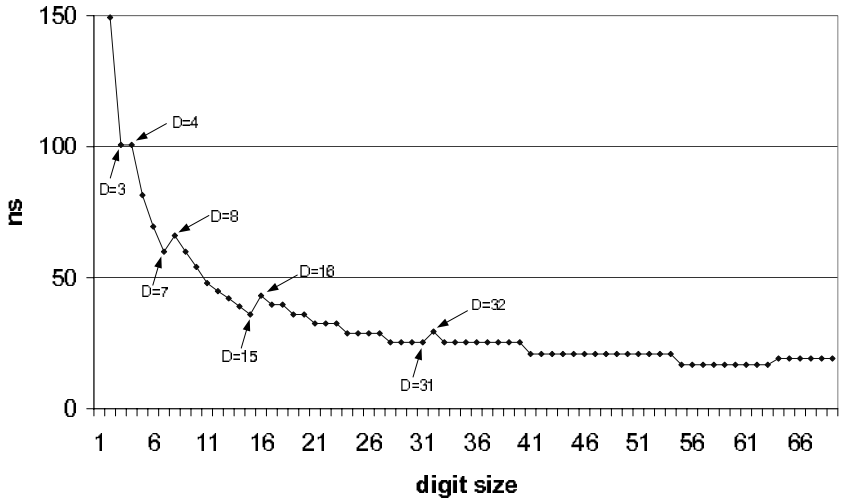

Fig. 10: Time to complete one multiplication of the single accumulator multiplier

We can generate a very fast design by using a lot of resources, hence a large digit-size leads to a faster multiplier. Thus, one has to consider speed and area in order to get the optimum multiplier, like using the area-time product. Fig. 11 we draw the area-time product over different

digit-sizes of the single accumulator multiplier. This plot clearly shows that most commonly used digit sizes are not only slower but also inefficient in terms of the area-time product used. Better and faster implementations of public key cryptography can be obtained using traditional LSD SAM multiplier by choosing $D = 2^l - 1$. For example a SAM multiplier with D=3 can compute a multiplication in the same time as D=4 but will require much smaller area. For D=7, the multiplier would compute the result faster than D=8 with 10% lesser area.

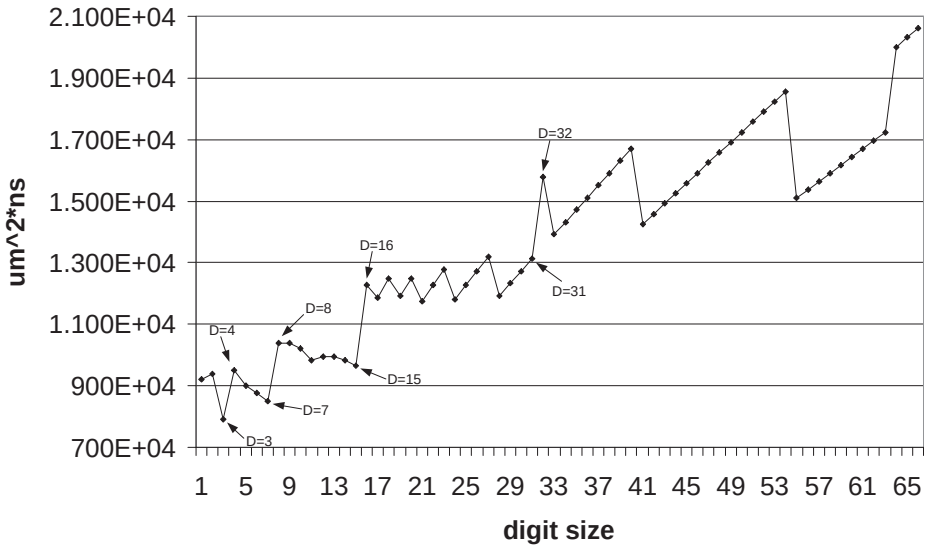

Fig. 11: Area-Time Product of the single accumulator multiplier

B. Evaluation of all multiplier options

In this subsection, we are going to compare all our introduced multiplier options. Fig. 12 shows the time requirements for the different multipliers. As expected, DAM and NAM (n=3) architectures are faster than SAM. Its important to note that the optimum digit sizes changes for different architectures. This is because of the optimum tree structures which are formed at different sizes of D within the DAM and NAM architecture. For example one should rather use D=4 for NAM, whereas this digit size will be not optimal for DAM and SAM. For DAM and SAM we rather would use D=3.

The area-time product of the multipliers is plotted in Fig. 13. This shows that DAM and NAM are also efficient architectures when we consider speed and resources of hardware used.

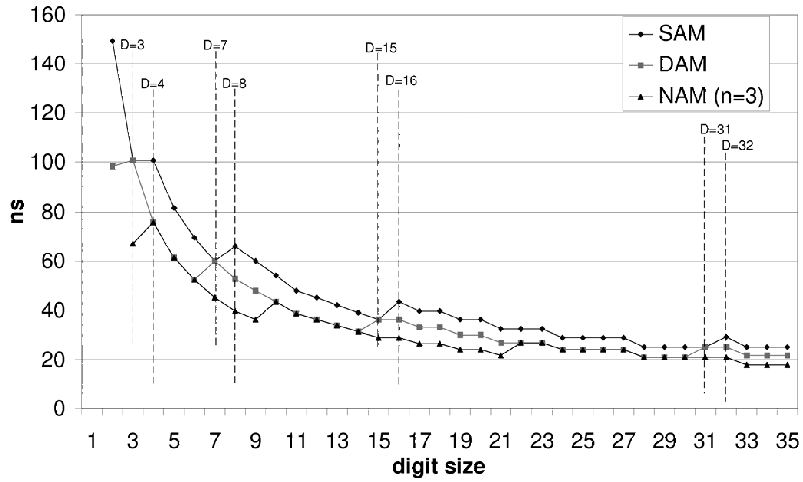

Fig. 12: Time to complete one multiplication of all the different multiplier implementations

NAM can be inefficient for small D sizes because of the extra overhead in area due to the registers and the extra clock cycle in the last step. The designer has to choose the right architecture based on various constraints like area and speed. For example, if the designer had a compromise of area and speed at D=8 for a SAM architecture, then he can either implement it with the same or smaller area with SAM (D=7) or DAM with D=6 or NAM (n=3) with D=3 with much better speed. This extra flexibility eventually allows the designer to built more optimum cryptosystems than presently available.

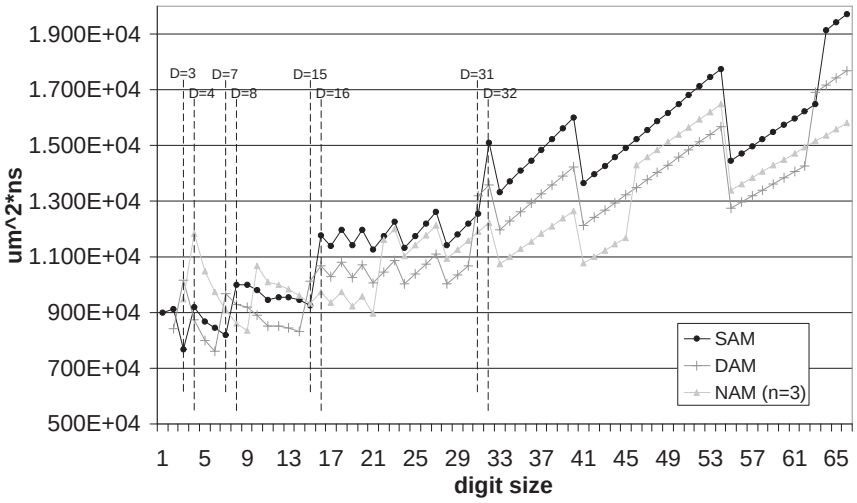

Fig. 13: Area-Time Product of the different multiplier implementations

VI. CONCLUSIONS

In this contribution, we showed new architectures for implementing LSD multipliers. The conditions that apply on the irreducible polynomial to successfully implement such architectures are given. It can be seen that all NIST recommended polynomials easily satisfy these conditions, which make these architectures very promising for implementing curve based public key cryptosystems. An evaluation of the multipliers for different digit sizes provide optimum values of D which give the best efficiency for a required speed. This has enabled us to show that present digit-sizes being used are the worst choices and much better implementations are possible. The different possible architectures also provide the designer with more flexibility in making the compromise between area and time which is inherent in all implementations.

REFERENCES

[GCE+01] N. Gura, S. Chang, H. Eberle, G. Sumit, V. Gupta, D. Finchelstein, E. Goupy, and D. Stebila. An End-to-End Systems Approach to Elliptic Curve Cryptography. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, volume LNCS 1965, pages 351–366. Springer-Verlag, 2001.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[Kob88] N. Koblitz. A Family of Jacobians Suitable for Discrete Log Cryptosystems. In Shafi Goldwasser, editor, *Advances in Cryptology - Crypto '88*, volume LNCS 403, pages 94 – 99, Berlin, 1988. Springer-Verlag.

[Mil86] V. Miller. Uses of elliptic curves in cryptography. In H. C. Williams, editor, *Advances in Cryptology — CRYPTO '85*, volume LNCS 218, pages 417–426, Berlin, Germany, 1986. Springer-Verlag.

[OP00] G. Orlando and C. Paar. A High-Performance Reconfigurable Elliptic Curve Processor for $GF(2^m)$. In Ç. K. Koç and C. Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2000*, volume LNCS 1965. Springer-Verlag, 2000.

[OP01] G. Orlando and C. Paar. A Scalable $GF(p)$ Elliptic Curve Processor Architecture for Programmable Hardware. In Ç. K. Koç, D. Naccache, and C. Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems — CHES 2001*, volume LNCS 2162, pages 348–363. Springer-Verlag, May 14-16, 2001.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[SP98] L. Song and K. K. Parhi. Low energy digit-serial/parallel finite field multipliers. *Journal of VLSI Signal Processing*, 19(2):149–166, June 1998.

[VLS03] VLSI Computer Architecture, Arithmetic, and CAD Research Group – Deparment of Electrical Engineering, IIT, Chicago, IL. IIT Standard Cells for AMI $0.5\mu m$ and TSMC $0.25\mu m/0.18\mu m$ (Version 1.6.0) , 2003. Library and documentation available from http://www.ece.iit.edu/ vlsi/scells/home.html.