Highly efficient architecture of elliptic curve scalar multiplication with fault tolerance over GF($2^m$)

Fang Zhou¹, a), Caixian Fei¹, Ning Wu¹, b) and Fen Ge¹

Abstract In this paper, an efficient hardware architecture of scalar multiplication is proposed for elliptic curve cryptography. To reduce circuit area, we propose an elliptic curve operation unit architecture for Montgomery Ladder Algorithm in projective coordinates. The basic modular arithmetic circuit in elliptic curve group operation module is reused to realize coordinates transformation and y-coordinate recovery operation. Considering concurrent error-detecting and fault-tolerant, we improve the existing error detection scheme by reusing intermediate results and predicting fault. The simulation and DC synthesis results show that the scalar multiplication circuit designed according to the proposed architecture reduces the time cost of fault detection in single iteration to 1 clock cycle at 100% fault detection rate, and the efficiency is improved 96% than the existing literature.

Keywords: elliptic curve cryptography (ECC), elliptic curve scalar multiplication (ECSM), error detection and recovery, Montgomery ladder

Classification: Integrated circuits (memory, logic, analog, RF, sensor)

1. Introduction

Elliptic curve cryptography (ECC) can provide privacy and data integrity protection with less computational overhead compared with other public key cryptography (PKC), while hardware-based ECC is more safe and fast compared to software approach. In ECC-based cryptosystems, the major computation is Elliptic Curve Scalar Multiplication (ECSM). The most attractive ECSM algorithm is Montgomery Ladder Algorithm [1, 2], which has the capability to resist some bypass attacks such as sign change attack [3] and simple power attack (SPA) [4].

Faults caused by natural or artificial reasons may result in data corruption and even security leakage of the cryptosystem. In the past years, various fault attacks to ECC-based cryptosystems have been presented [5, 6, 7]. For resisting fault attacks, Point Verification (PV) is an effective way [6, 8, 9]. However, PV cannot detect all faults. Attacks have been proposed to manipulate the faulty results so they can pass PV process [7]. A number of structures for the error detection in finite field multiplier have been proposed [10, 11, 12, 13, 14, 15], but the faults out of multiplier cannot be detected. Besides, fault detection is not enough for the cryptosystems that need transient faults tolerance. The scalar multiplication structures in [16] is based on re computation or parallel computation, where faults are detected or corrected by comparing or voting the results, but it requires more than 100% hardware overhead or time overhead. A low-cost error detection and recovery (LOEDAR) scheme for the Montgomery Ladder Algorithm is presented in [17]. It allows the trade-off between the time cost of error detection and recovery cost. However, the error detection time of this scheme even exceeds the time of one initial iteration. In response to this problem, we make a further optimization.

The organization of the paper is as follows. Section 2 make a brief overview of ECC and Montgomery Ladder Algorithm. Section 3 presents an elliptic curve operation unit architecture. Section 4 proposed a low-delay error detection and recovery (LOEDAR) scheme, which is an efficiency improved version of LDEDR. The experimental results are shown in Section 5. Section 6 makes a conclusion.

2. Scalar multiplication over GF($2^m$)

2.1 ECC overview

This paper implements scalar multiplication in the binary finite field $GF(2^m)$ due to its hardware-efficient field arithmetic operations [18]. An elliptic curve $E$ over $GF(2^m)$ is defined to be a set of points $(x, y)$ that satisfy the following equation, where $a, b \in GF(2^m)$.

$$y^2 + x y = x^3 + ax^2 + b(b \neq 0) \quad (1)$$

The scalar multiplication can be finished through a series of point addition and point doubling operations. Depending on how these two basic operations are organized, ECSM algorithms are divided into many types, among which Montgomery Ladder Algorithm can resist sign change attack and be implemented area-efficiently.

2.2 Montgomery ladder algorithm

Montgomery presented a method to compute multiples of points for elliptic curve, utilizing binary ladder [19]. His technique can be generalized to Algorithm 1.

The binary ladder makes this algorithm against timing analysis [20] and SPA. With the trait that the difference between $Q_1$ and $Q_2$ is always $P$, it is possible to have an error detection using coherency check [21] among involved variables. Additionally, considering the fact that the addition of two points in elliptic curve can be obtained without $y$-coordinates when knowing the difference between them, the $x$-coordinate of the sum $Q_1 + Q_2$ can be computed using

1 College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
2) zfnuaa@nuaa.edu.cn
b) wunee@nuaa.edu.cn

DOI: 10.1587/elex.18.20210160
Received April 6, 2021
Accepted April 21, 2021
Publicized May 11, 2021
Copyedited June 10, 2021

Copyright © 2021 The Institute of Electronics, Information and Communication Engineers
only the $x$-coordinates of $Q_1$, $Q_2$ and their difference $P$.
 Lopez and Dahab [2] used projective coordinates to reduce the number of division times for applications where the multiplicative inverse is relatively expensive. The affine coordinates of point $P = (x, y)$ is transformed to projective coordinates $P = (X, Y, Z)$ with $x = X/Z$ when only $x$-coordinate is used. Point addition, point doubling and $y$-coordinate recovery formulas are given in Eq. (2), Eq. (3) and Eq. (4), respectively.

$$X(Q_1 + Q_2) = xZ_1 + (X_1Z_2)(X_2Z_1)$$

$$Z(Q_1 + Q_2) = (X_1Z_2 + X_2Z_1)^2$$

$$y_Q = \left(\frac{xX_1Z_2}{X_2Z_2} + x\right)$$

$$X(2Q_1) = X_1^4 + bZ_1^4$$

$$Z(2Q_1) = X_1^2Z_1^2$$

3. Proposed architecture for elliptic curve operation

In Algorithm 1, one loop in the projective coordinates requires a total of six field multiplication according to Eq. (2), Eq. (3). The six multiplications can be computed in two steps using three multipliers or in three steps using two multipliers or in six steps by serial multiplications using one multiplier [22, 23]. To reduce delay, based on three multipliers, we designed separate Data Flow Graph (DFG) of point addition and point doubling shown in Fig. 1 so they can be called independently.

It comprises three $m$-bits multipliers, three squarers, and two adders. To show the data flow clearly, the intermediate variables are annotated in the figure. But actually, it’s no need to store them to registers, which will cause unnecessary delay [22]. The results of point addition and point doubling are stored in $(X_1, Z_1)$ and $(X_2, Z_2)$, respectively, and the store operation will cost one clock. So the delay of the longest point addition path is $2T_M + 1$, and that of point doubling path is $T_M + 1$, where $T_M$ is the delay of multiplication. The circuit is designed for Koblitz curves recommended by National Institute of Standard and Technology (NIST), where parameter $b$ in Eq. (3) is equal to 1, so the point doubling in Fig. 1 only contains one multiplication.

Coordinates conversion and $y$-coordinate recovery are only appear once respectively at the start and the end of Montgomery Ladder Algorithm, executed serially with point addition and point doubling. It is reasonable to reuse the modular operation units in point addition and point doubling circuits to complete the coordinates conversion, $y$-coordinate recovery, and the multiplication inversion in the recovery progress, so as to improve the hardware-efficiency. Therefore, we proposed an elliptic curve arithmetic and logic unit (ECALU) architecture, as shown in Fig. 2.

![Algorithm 1 Basic Montgomery’s ladder ECSM](image)

**Algorithm 1** Basic Montgomery’s ladder ECSM

**Input:** $k = (k_t-1k_t-2 \cdots k_1k_0)_2$ with $t \leq m$, $k_{t-1} = 1$, and $P \in E$;

**Output:** $Q = kP$;

1. $Q_1 \leftarrow P$, $Q_2 \leftarrow 2P$;
2. for $i = t-2$ downto 0 do do
3. if $k_i = 1$ then
4. $Q_1 \leftarrow Q_1 + Q_2$, $Q_2 \leftarrow 2Q_2$;
5. else
6. $Q_2 \leftarrow Q_1 + Q_2$, $Q_1 \leftarrow 2Q_1$;
7. end if
8. end for
9. Return $Q = Q_1$;

![Fig. 1 DFG of point addition and point doubling in ECSM](image)

![Fig. 2 Architecture of ECALU](image)

![Fig. 3 Multiplication inversion circuit](image)

The five functions of ECALU are controlled by five state machines, whose enable signals make up the operation code. The ECALU contains 8 $m$-bits registers, which are $\{X_1, Z_1, X_2, Z_2, X_3, Z_3, T_1, T_2\}$. The registers $T_1$ and $T_2$ is used in modular inversion. The design details of $y$-coordinate recovery is not presented, which is similiar to that of point operation. Multiplication inversion is calculated by DJ Algorithm [24], where partial squares can overlap with multiplication to decrease the delay in the same circuit as the TT algorithm [25, 26, 27, 28] hardware implementation. The circuit is shown in Fig. 3.
4. Fault-tolerant architecture for ECSM

The error detection and recovery (EDR) mechanism proposed in [17] uses coherency check to detect errors, and makes it possible to roll back after detecting errors by data backup. This scheme is called low-cost error detection and recovery (LOEDAR). In case of transient faults in ECSM, the probability of undetected error is no larger than $1/2m^3$. LOEDAR is described below and an improved scheme for time overhead is proposed.

4.1 Existing LOEDAR scheme

At the end of each iteration of the Montgomery ladder algorithm, equation $Q_2 = Q_1 + P$ holds, which can be used for coherency check. However, this equation cannot be verified directly to detect errors, for the reason that $Q_1 + P$ is hard to calculate without $y$-coordinate. To slove this problem, a new point called verification point, $Q_v$, is introduced. Fig. 4 shows accumulation details in one iteration.

![Accumulation of verification point $Q_v$](image)

The accumulation of $Q_v$ and point addition in basic Montgomery Ladder Algorithm are executed simultaneously. The two operations involved, $Q_v + Q_2$ and $Q_v + Q_1$ can both be calculated without $y$-coordinate. If $Q_v$ is updated as described above, then equation $Q_v + Q_2 = 2Q_1$ is true at the end of each loop if no fault occurs. Verification of this equation and recovery when errors are detected are referred to as EDR, which are executed between the two iterations and involves a point addition, a point doubling and two multiplications followed by a comparison to verify if $X_{Q_v}+Q_2 \cdot Z_{Q_v} = X_{Q_2} \cdot Z_{Q_v}+Q_2$. To reduce the time cost of EDR, run the EDR process much less frequently, i.e. one EDR is performed after every $l$ iterations of ECSM.

4.2 Proposed LDEDR scheme

Considering that the data path of the point addition is longer than that of the point doubling, and the point addition in EDR is the same as the calculation of $Q_v$, at $k_i = 1$, the EDR is scheduled after the iterations at $k_i = 1$ to reuse the result of $Q_v = Q_v + Q_2$. So, we can save the time of an point addition operation. Only when $k_i = 1$, the verification result is judged. If the equation holds, back up the middle point $Q_1, Q_2, Q_v$, key, and iteration position. If the check failed, roll back to the state when the previous check passed. The anti-SPA property of Montgomery Ladder Algorithm is not destroyed.

Based on the above analysis, the improved scalar multiplication algorithm is shown in Algorithm 2, where the random() function generates random $Z$ coordinate to resist differential power analysis (DPA) [29], and the reco() function recovers the affine coordinates of the $kP$.

**Algorithm 2** Montgomery’s ladder ECSM with error detection and recovery

**Input:** $k = (k_{t-1}k_{t-2} \cdots k_1k_0)_2$ with $t \leq m$, $k_{t-1} = 1$, and $P(x, y) \in E$;

**Output:** $Q = kP$;

1: $Z \leftarrow \text{random}(); \text{check}_\text{cnt} \leftarrow 0$;
2: if $P(x, y) \neq 1$ then Restart;
3: $Q_1 \leftarrow (x \cdot Z, Z), Q_2 \leftarrow 2Q_1, Q_v \leftarrow O$;
4: for $i = t - 2$ downto 0 do do
5:    $\text{check}_\text{cnt} \leftarrow \text{check}_\text{cnt} + 1$;
6: if $k_i = 1$ then
7:    $Q_1 \leftarrow Q_1 + Q_2, Q_2 \leftarrow 2Q_2, Q_v' \leftarrow Q_v - Q_2$;
8: else
9:    $Q_2 \leftarrow Q_1 + Q_2, Q_1 \leftarrow 2Q_1, Q_v' \leftarrow Q_v + Q_2$;
10: end if
11: EDR part;
12: if check$_\text{cnt}=l$ then
13:    $Q_v' = 2Q_1, \text{check}_\text{cnt} = 0$;
14: $T_1 = X_{Q_v}^{-1} \times Z_{Q_v}^{-1}, T_2 = X_{Q_v}^{-1} \times Z_{Q_v}^{-1}$;
15: end if
16: if check$_\text{cnt}=l$ and $k_i-1 = 1$ then
17: if $T_1 = T_2$ then
18:    Save $\{Q_1, Q_2, Q_v', key, i\}$;
19: else
20:    Recover $\{Q_1, Q_2, Q_v', key, i\}$;
21: end if
22: end if
23: end for
24: $(x_1, y_1) \leftarrow \text{reco}(X_1, Z_1, X_2, Z_2, x, y)$;
25: if $P(x_1, y_1) \neq 1$ then recover $\{Q_1, Q_2, Q_v, key, i\}$;
26: Return $Q = (x_1, y_1)$;

Because EDR is performed every $l$ iterations, so we set a counter check$_\text{cnt}$ to control the calculation in EDR. The $Q_v'$ means the value of $Q_v$ in the $i$-th iteration, and similarly, the $Q_v'^{-1}$ means the value of $Q_v$ in the $(i-1)$-th iteration.

The structure of ECSM circuit resisting fault attack is shown in Fig. 5. It is mainly composed of an ECALU, a Point Addition module for $Q_v$ accumulation, an Error Detection module for point verification and point equality check, and a register file. Both ECALU module and the extra module for error detection contain three multipliers, three squarers and two adders.

![Proposed ECSM circuit architecture](image)
In [17], the check of $Q_v = Q_d$ is done after $Q_v$ and $Q_d$ are figured out. To further reduce the time cost of error detection, we proposed a scheme to make this check and the calculation of other variables run at the same time, which is shown in Fig. 6.

Fig. 6 (a) shows the schedule of LDEDR, where the verification is performed serially. Fig. 6 (b) shows two iterations in our LDEDR scheme. The extra calculation for error detection is in blue block, and gray blocks is the calculation in our LDEDR scheme. The extra calculation for error detection is performed serially. Fig. 6 (b) shows two iterations.

If the data passes the error detection, contents in temporary data registers will be covered by that of security data registers.

5. Experiments

The scalar multiplication algorithm over $GF(2^{233})$ are modeled and synthesized in SMIC 0.18\(\mu m\) library by Synopsys Design Compiler, and the max frequency of the ECSR circuit is 134MHZ. We used the most significant digit (MSD) digit serial multipliers [30] with digit size $d = 4$. The squarer is similar to [31]. Divider used for coordinate transformation is the same as the one given in [31]. To evaluate the area saving of ECALU, we also modeled each function in it independently, and the area data are reported in Table I. It can be seen that the sum of the area of the Coordinate Transformation, y-coordinate Recovery, INV modules, Point Doubling, and Point Addition modules is 126.6K, and the ECALU reduced the area by 58%. The hardware overhead of LDEDR is about 53.5%.

Table I The area data of point operation modules

| Module                        | NAND2-equivalent area |
|-------------------------------|-----------------------|
| Scalar Multiplication         | 120.06K               |
| ECALU                         | 53.19K                |
| Coordinate Transformation     | 8.71K                 |
| y-coordinate Recovery         | 35.18K                |
| INV                           | 21.57K                |
| Point Doubling                | 13.09K                |
| Point Addition                | 28.04K                |
| Point Verification            | 18.57K                |
| Point Register                | 2.94K                 |

We use parameters recommended by NIST to test the function of the circuit. Eq. (5) shows the parameters.

\[
\begin{aligned}
a &= 0, b = 1 \\
x &= 0x172322ba853a7e731af129ff22ff4149 \\
563a419c26bf50a4e9d6eef6126 \\
y &= 0x1db537e8e81b7f70f555a67c427a \\
8c9d9bf18aebe9b56e0e11056fae33
\end{aligned}
\]

EDR is configured to performed every 5 iterations. Fault was injected at the end of 20-th iteration, and the functional simulation results near fault injection location were shown in Fig. 8.

The SUCCESS signal at the bottom of Fig. 8 is the check result given by the Error Detection module. It can be seen that the interval of two EDR is 7.96\(\mu s\). The SUCCESS signal should jump to high level at 25.545\(\mu s\) but that do not happen, because error are detected. Fig. 9 shows the time of fault detection in this fault injection.

![Fig. 7 The data transmission between three types registers](image-url)
The iterations where EDR are performed are marked in red. It takes 347.47 μs to finish the scalar multiplication under 100MHz. Further more, we select different keys and injection locations to run simulation and compare them to the results calculated by software program, and all the scalar multiplication results are correct.

To compare the efficiency of our LDEDR scheme and the LOEDAR scheme in [17] concisely, we consider the use of registers and finite field modular arithmetic module in the Montgomery ladder iteration, but not coordinates transformation, point verification after y-coordinate recovery and so on. Table II shows the theoretical area and delay comparison, where M, A, S and R represent the area of multiplier, adder, squarer and m-bits register respectively.

Table II Theoretical area and delay overhead

| Design | Area | Time |
|--------|------|------|
|        | ECSM | EDR  | ECSM | EDR  |
| [17]   | 2M+2A+2S+4R | (4M + 1) | (5M + 1) | (t − 1) |
|        | 3M+2A+3S+4R | (2M + 1) | (t − 1) |
| Ours   | 3M+2A+3S+12R | (2M + 1) | (t − 1) |

In SMIC 0.18 μm library, the NAND2-equivalent area of 163-bits register is 1051, and we approximate it as the data under TMSC 65 nm library. Combined with data of other arithmetic units in [17], the practical area and delay comparison are shown in Table III. Hardware efficiency here is defined as 1/(area × cycles). The effiency of LDEDR is improved 96% than LOEDAR. the time percent of error detection is 10%, which is 82% lower than the existing scheme.

Table III Practical area and delay overhead

| Design | NAND2 equivalent area | Time(clock cycles) | Efficiency |
|--------|-----------------------|--------------------|------------|
|        | ECSM | EDR  | ECSM | EDR  |
| [17]   | 61786 | 35097 | 2754 | 3402 | 0.51 |
| Ours   | 90169 | 97051 | 1458 | 162  | 1    |

6. Conclusion

In this paper, we have presented a highly efficient architecture for ECSM implementation based on Montgomery Ladder Algorithm, containing an elliptic curve arithmetic and logic unit (called ECALU) and a low-delay error detection and recovery (called LDEDR) module. Experimental results show that compared with scheme presented in [17], the time overhead in single iteration of LDEDR are about 10%, and the efficiency is improved 96%. And fault injection experiment illustrates LDEDR’s capability of error detection and recovery.

Acknowledgments

This work was supported by Aeronautical Science Foundation of China (No. 201943052001), the Fundamental Research Funds for Central Universities (No. NP2019102), Project of Science and Technology on Electronic Information Control Laboratory.

References

[1] P.L. Montgomery: “Speeding the pollard and elliptic curve methods of factorization,” Mathematics of Computation 48 (1987) 243 (DOI: 10.1109/12.869328).
[2] J. López and R. Dahab: “Fast multiplication on elliptic curves over GF(2^m) without precomputation,” Proceedings of the Workshop on CHES (1999) 316 (DOI: 10.1007/3-540-48059-5_27).
[3] J. Blömer, et al.: “Sign change fault attacks on elliptic curve cryptosystems,” Workshop on FDTC (2006) 36 (DOI: 10.1007/11889700_4).
[4] P.C. Kocher, et al.: “Differential power analysis,” CRYPTO (1999) 388 (DOI: 10.1007/3-540-48405-1_25).
[5] M. Ciet and M. Joye: “Elliptic curve cryptosystems in the presence of permanent and transient faults,” Designs Codes & Cryptography 36 (2005) 33 (DOI: 10.1007/s10623-003-1160-8).
[6] I. Biehl, et al.: “Differential fault attacks on elliptic curve cryptosystems,” CRYPTO (2000) 131 (DOI: 10.1007/3-540-44598-6_8).
[7] P.-A. Fouque, et al.: “Fault attack on elliptic curve Montgomery ladder implementation,” Workshop on FDTC (2008) 92 (DOI: 10.1109/FDTC.2008.15).
[8] A. Antipa, et al.: “Validation of elliptic curve public keys,” PKC (2003) 211 (DOI: 10.1007/3-540-36288-6_16).
[9] S.-M. Yen and M. Joye: “Checking before output may not be enough against fault-based cryptanalysis,” IEEE Trans. Comput. 49 (2000) 967 (DOI: 10.1109/12.869328).
[10] A. Reyhani-Masoleh and M.A. Hasan: “Fault detection architectures for field multiplication using polynomial bases,” IEEE Trans. Comput. 55 (2006) 1089 (DOI: 10.1109/12.869328).
[11] A. Hariri and A. Reyhani-Masoleh: “Fault detection structures for the montgomery multiplication over binary extension fields,” Workshop on FDTC (2007) 37 (DOI: 10.1109/FDTC.2007.19).
[12] S. Bayat-Sarmadi and M.A. Hasan: “Concurrent error detection in finite-field arithmetic operations using pipelined and systolic architectures,” IEEE Trans. Comput. 58 (2009) 1553 (DOI: 10.1109/TC.2009.62).
[13] A. Hariri and A. Reyhani-Masoleh: “Concurrent error detection in Montgomery multiplication over binary extension fields,” IEEE Trans. Comput. 60 (2011) 1341 (DOI: 10.1109/TC.2010.258).
[14] C. Lee and J. Xie: “High capability and low-complexity: novel fault detection scheme for finite field multipliers over GF(2^m) based on MSPA,” HOST (2019) 21 (DOI: 10.1109/HST.2019.8741034).
[15] C. Fei, et al.: “A scalable bit-parallel word-serial multiplier with fault detection on GF(2^m),” ICCT (2020) 1660 (DOI: 10.1109/ICCT50939.2020.9295940).
[16] A. Dominguez-Oviedo and M.A. Hasan: “Error detection and fault tolerance in ECSM using input randomization,” IEEE Trans. Dependable Secure Comput. 6 (2009) 175 (DOI: 10.1109/TDSC.2008.21).
[17] K. Ma and K. Wu: “Error detection and recovery for ECC: a new approach against side-channel attacks,” IEEE Trans. Comput.-Aided Design Integr. Circuits Syst. 33 (2014) 627 (DOI: 10.1109/TCAD.2013.2293058).
[18] E. Wenger and M. Hutter: “Exploring the design space of prime field vs. binary field ECC-hardware implementations,” Information Security Technology for Applications (2011) 256 (DOI: 10.1007/978-3-642-29615-4_18).
[19] R.E. Crandall: U.S. Patent 5159632 (1992).
[20] P.C. Kocher: “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems,” CRYPTO (1996) 104 (DOI: 10.1007/3-540-68697-5_9).
[21] A. Dominguez-Oviedo and M.A. Hasan: “Algorithm-level error detection for Montgomery ladder-based ECSM,” Journal of Cryptographic Engineering 1 (2011) 57 (DOI: 10.1007/s13389-011-0003-1).
[22] B. Ansari and M.A. Hasan: “High-performance architecture of elliptic curve scalar multiplication,” IEEE Trans. Comput. 57 (2008) 1443 (DOI: 10.1109/TC.2008.133).
[23] G.D. Sutter, et al.: “Efficient elliptic curve point multiplication using digit-serial binary field operations,” IEEE Trans. Ind. Electron. 60 (2013) 217 (DOI: 10.1109/TIE.2012.2186104).
[24] V. Dimitrov and K. Jarvinen: “Another look at inversions over binary fields,” ARITH (2013) 211 (DOI: 10.1109/ARITH.2013.25).
[25] T. Itoh and S. Tsujii: “A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases,” Information and Computation 78 (1988) 171 (DOI: 10.1016/0890-5401(88)90024-7).
[26] J. Hu, et al.: “Fast and generic inversion architectures over GF(2^m) using modified Itoh–Tsujii algorithms,” IEEE Trans. Circuits Syst. 62 (2016) 367 (DOI: 10.1109/TCSII.2014.2387612).
[27] C. Rebeiro, et al.: “Revisiting the Itoh-Tsujii inversion algorithm for FPGA platforms,” VLSI 19 (2011) 1508 (DOI: 10.1109/TVLSI.2010.2051343).
[28] S.S. Roy, et al.: “Theoretical modeling of the Itoh-Tsujii inversion algorithm for enhanced performance on k-LUT based FPGAs,” Design, Automation & Test (2011) 1 (DOI: 10.1109/DATE.2011.5763197).
[29] J.-S. Coron, et al.: “Resistance against differential power analysis for elliptic curve cryptosystems,” CHES (1999) 292 (DOI: 10.1007/3-540-48059-5_25).
[30] L. Song and K.K. Parhi: “Low-energy digit-serial/parallel finite field multipliers,” Journal of VLSI Signal Processing Systems for Signal, Image and Video Technology 19 (1998) 149 (DOI: 10.1023/A:1008013818413).
[31] S.M. Shohdy, et al.: “FPGA implementation of elliptic curve point multiplication over GF(2^{191}),” ISA (2009) 619 (DOI: 10.1007/978-3-642-02617-1_63).