PAPER Special Section on Multiple-Valued Logic and VLSI Computing

An Algebraic Approach to Verifying Galois-Field Arithmetic Circuits with Multiple-Valued Characteristics*

Akira ITO[†a)], ***Nonmember***, **Rei UENO**[†], ***and*** **Naofumi HOMMA**[†], ***Members***

SUMMARY This study presents a formal verification method for Galois-field (GF) arithmetic circuits with the characteristics of more than two values. The proposed method formally verifies the correctness of circuit functionality (i.e., the input-output relations given as GF-polynomials) by checking the equivalence between a specification and a gate-level netlist. We represent a netlist using simultaneous algebraic equations and solve them based on a novel polynomial reduction method that can be efficiently applied to arithmetic over extension fields $\mathbb{F}_{p^m}$, where the characteristic p is larger than two. By using the reverse topological term order to derive the Gröbner basis, our method can complete the verification, even when a target circuit includes bugs. In addition, we introduce an extension of the Galois-Field binary moment diagrams to perform the polynomial reductions faster. Our experimental results show that the proposed method can efficiently verify practical $\mathbb{F}_{p^m}$ arithmetic circuits, including those used in modern cryptography. Moreover, we demonstrate that the extended polynomial reduction technique can enable verification that is up to approximately five times faster than the original one.

key words: *decision diagrams, formal verification, Galois-field arithmetic circuits, multiple-valued logic*

1. Introduction

Cryptography based on Galois-field (GF) arithmetic has been widely utilized in many secure information systems that require secret communication, authentication, and digital signatures. Cryptographic algorithms are frequently implemented in hardware to achieve lower latency and power/energy consumption, particularly in the case of resource-constrained embedded devices, such as smartcards.

Some cryptographic algorithms can be more efficiently implemented with multiple-valued logic than with binary logic. For example, some practical elliptic curve cryptographies (ECCs) and pairing-based cryptographies (PBCs) are defined over the GFs with characteristics greater than two (i.e., $\mathbb{F}_{p^m}$, where p is a characteristic and m is the extension degree). In [1]–[4], PBC over a GF with $p = 3$ exhibits a high level of security with a shorter key and less computational complexity than those defined over binary fields. In addition, it was reported that a hyperelliptic curve cryptography over a GF with $p = 5$ or 7 was useful for the efficient implementation of PBC [5], [6]. Thus, arithmetic circuits over GFs with multiple-valued characteristics play a key role in the efficient implementation of ECCs and PBCs.

However, existing tools and methods encounter practical difficulties in the design and functional verification of such GF arithmetic circuits. For example, most standard cell libraries do not have a good design for GF arithmetic circuits. The difficulties in verifying GF arithmetic circuits are more serious and critical than designing them. GF multipliers that are used in ECCs and PBCs typically have input word lengths greater than 64-bits; therefore, the generation of exhaustive test patterns and complete simulation-based verification are practically impossible. Conventional formal verification methods based on satisfiability solvers, satisfiability modulo theories solvers, and binary decision diagrams (BDDs) cannot be efficiently applied to GF arithmetic circuits. In [7], it was shown that these conventional methods could only handle up to 16-bit GF arithmetic circuits. There are extended BDDs, such as binary moment diagrams (BMDs), index BDDs (IBDDs), and multiple-output decision diagrams (MODD), that are specifically for verifying arithmetic circuits. However, BMDs are suitable for integer arithmetic circuits [8] and cannot be directly applied to represent the polynomials that appear during the verification of GF arithmetic circuits. IBDD is an extension of BDD which has a layer structure to represent integer multipliers compactly [9]. Although IBDDs were firstly introduced for equivalence checking [10], [11] showed that the equivalence test in IBDDs is coNP-complete due to the lack of canonical representation. MODD is an extension of BDD which represents multiple-valued functions using the multiple-valued Shannon expansion. However, the result of [12] implies that the sizes of BDD and MODD would explode if they represent circuits that include many XORs, such as GF arithmetic circuits. This indicates that it is difficult to apply BDDs and MODDs to verification of GF arithmetic circuits.

Recently, some formal verification methods based on computer algebra have been reported for GF arithmetic circuits [7], [13]–[15]. Assuming that the specification (i.e., functionality) of a GF arithmetic circuit can be given as GF polynomials, these methods examine whether the GF polynomials of the specification can be derived by simultaneous algebraic equations, which are derived from a target netlist. Computer algebra techniques make it possible to completely and soundly verify practical GF multipliers, such as 571-bit Mastrovito multipliers. However, the applicability of

Manuscript received August 26, 2020.
Manuscript revised February 18, 2021.
Manuscript publicized April 28, 2021.

[†]The authors are with Tohoku University, Sendai-shi, 980–8579 Japan.

*This work was supported in part by the JSPS Research Felow under Grant 20J12887, in part by the JSPS KAKENHI under Grant 20K19765, and in part by the Secom Science and Technology Foundation.

a) E-mail: ito@riec.tohoku.ac.jp

DOI: 10.1587/transinf.2020LOP0004

such methods to GF arithmetic circuits with multiple-valued characteristics is currently unknown. In fact, many conventional methods were optimized for extension fields of $\mathbb{F}_2$, and would not be efficiently applied to circuits over $\mathbb{F}_{p^m}(p \geq 3)$.

To address the aforementioned problem, we present a formal verification method for GF arithmetic circuits with multiple-valued characteristics. The proposed method verifies the equivalence between a specification and a target netlist. In the proposed method, we represent the netlist as simultaneous algebraic equations and solve them based on a new polynomial reduction method that can be efficiently applied to arithmetic over extension fields $\mathbb{F}_{p^m}$, where the characteristic p is larger than two. The use of the reverse topological term order (RTTO) to derive the Gröbner basis makes the verification feasible, even when a target netlist includes bugs.

Compared with the preliminary version [16], we present a faster and scalable polynomial reduction method for the above verification that is based on an extension of zero-suppressed binary decision diagrams (ZDDs). These are well known for their effectiveness in the verification of GF arithmetic circuits over the $\mathbb{F}_2$ extension field. In this study, we validate the extended method through a series of experimental verifications for GF multipliers with some multiple-valued characteristics. The results show that the proposed method can perform a complete verification of the GF multiplier with 256-digit inputs within approximately two minutes.

2. Related Work

2.1 Gröbner Basis-Based Equivalence Checking

An algebraic method based on the Gröbner basis and polynomial reduction is considered as one of the most promising methods to verify GF arithmetic circuits. The method examines the equivalence between the input–output relation of a target circuit (i.e., specification) and a given netlist, both of which are described by a set of GF polynomials. The verification consists of the following two steps: (i) the Gröbner basis is derived corresponding to the simultaneous equations that represent the netlist; and (ii) a polynomial reduction is performed by the Gröbner basis. Here, the computation of the Gröbner basis in step (i) is typically time-consuming. The reduction in step (ii) also requires considerable time and memory for the verification of practical circuits. Thus far, we have two approaches to compute the time-consuming steps efficiently: hierarchical and non-hierarchical approaches. In this section, we briefly introduce these two approaches.

The major hierarchical method is based on the GF arithmetic circuit graph (GF-ACG), which is a hierarchal and mathematical graph that represents a GF arithmetic circuit with a functional assertion given by GF equations and an internal structure. Given an arithmetic circuit description in the GF-ACG, we can verify its function using equivalence checking between the functional assertion and its internal structure. In [17], [18], polynomial reduction using the Gröbner basis is adopted for a complete and efficient verification. Here, if a target circuit is described in a good hierarchical manner, the computational cost of the Gröbner basis and polynomial reduction becomes trivial because the number of variables and polynomials per Gröbner basis computation is reduced. It was shown that the GF-ACG-based methods can verify the correctness of practical GF arithmetic circuits, such as 128-bit GF multipliers and advanced encryption standard round data paths [18]. In addition, GF-ACGs were extended to GF arithmetic circuits with multiple-valued characteristics [19]. However, one major limitation of GF-ACGs is that it cannot be efficiently applied to flattened descriptions. Although a target circuit is frequently given by a (flattened) gate-level netlist, the effectiveness and practicality of GF-ACGs are unclear in such cases. In addition, GF-ACG-based methods experience difficulties in handling circuits with bugs, owing to the Büchberger algorithm used to derive the Gröbner basis.

In contrast, the non-hierarchical approach can verify a wider range of descriptions, including the flattened gate-level netlist. However, the computational costs of steps (i) and (ii) would be significantly more critical if a straightforward computation using the Büchberger's or F4 algorithms is employed.

In 2013, Lv et al. presented the RTTO to mitigate the computational cost of the Gröbner basis in step (i). The RTTO is the term order that is determined in accordance with the circuit topology, where a variable (i.e., wires) closer to the primary outputs (POs) of a target circuit always has a higher term order. Formally, if a wire w_k is closer to the POs than another wire w_i, the order of w_k and w_i is given by $w_k > w_i$ in the RTTO. It was proven that a set of $\mathbb{F}_2$ polynomials that represent the specification and logic gates (i.e., combinational circuit) should always be a Gröbner basis if the RTTO is applied to the expressions of polynomials. As a result, we can skip the derivation of the Gröbner basis and reduce most of the computational cost in step (i).

Accordingly, the reduction of PO variables in step (ii) becomes the most time-consuming part of the verification. Several studies that aim to decrease the computational cost of step (ii) have also been reported. For example, Gupta et al. used the ZDD to represent polynomials over a Boolean ring to perform the reduction of PO variables in a shorter time and with a smaller memory size compared with the corresponding AND-XOR representation [13]. In [20], the reduction of PO variables was performed in parallel to reduce the total computational time at the expense of computational resources.

Owing to the RTTO and the above-mentioned technique of step (ii), the recent non-hierarchical method succeeded in verifying the correctness of netlists for large GF multipliers, such as 571-bit Mastrovito multipliers. However, it is unknown whether such conventional methods can be applied to the netlists of GF arithmetic circuits with multiple-valued characteristics. The representation of poly-

nomials in the above-mentioned methods is highly optimized for arithmetic over $\mathbb{F}_2$, which implies that it would not be efficiently applicable to GF arithmetic circuits with characteristics greater than two.

2.2 Decision Diagrams

As mentioned above, Gupta et al. reported a ZDD-based polynomial reduction for efficient verification. The reason why ZDDs are effective for verifying GF arithmetic circuits is that ZDDs can compactly represent Boolean polynomials using the following positive Davio expansion[†]

$$f(x) = f_1 + xf_2, \tag{1}$$

where $f(x)$ is a logical function (Boolean polynomial) of a Boolean variable x, and f_1 and f_2 denote Boolean polynomials independent of x. ZDDs are graph representations based on recursive applications of the positive Davio expansion with respect to all variables. That is, a ZDD indicates whether the corresponding polynomial contains a particular product term. BDDs are different from ZDDs in that they represent satisfaction conditions. The signal value of a GF arithmetic circuit frequently changes with its input signal because additions over the prime field $\mathbb{F}_2$ correspond to XOR gates. Therefore, the satisfaction conditions of a GF arithmetic circuit become a complicated polynomial, and ZDDs are more suitable for the verification than BDDs.

For the extension of DDs to multiple-valued functions, there are some multiple-valued DDs, such as multi-valued DDs (MDDs), multiple-output DDs (MODDs), and Galois-field functional DDs (GFDDs) [23]–[25]. However, it is difficult to apply such conventional multiple-valued DDs to the verification of GF arithmetic circuits with multiple characteristics. MDDs and MODDs are multiple-valued extensions of BDDs; thus, they represent the satisfaction conditions. That is, it seems unlikely that they are able to compactly represent the polynomials that appear during the proposed verification process for the same reason as BDDs. GFDD is a multiple-valued extension of a ZDD that can compactly represent polynomials over $\mathbb{F}_p$ [25]. However, its construction method, which is shown in [25], is very complicated, and the APPLY algorithms of GFDDs are unknown. Thus, it would also be difficult to use GFDDs for the polynomial reductions mentioned in this study. Therefore, in Sect. 3, we introduce an extension of ZDDs to verify GF arithmetic circuits with multiple-valued characteristics.

3. Proposed Method

The proposed verification procedure follows an algebraic non-hierarchical approach, which is shown in [13]. More precisely, the proposed method can be considered as an extension for $p > 2$ of [13], which uses RTTO and ZDDs. First, we describe the basic idea and procedure of our method. Then, we introduce a diagram to represent polynomials over $\mathbb{F}_p$ to perform verification in an efficient and scalable manner.

3.1 Equivalence Checking

In this subsection, we describe our equivalence checking (i.e., verification) method for arithmetic circuits over an extension field of $\mathbb{F}_p$. In contrast to the conventional methods, the proposed method examines the equivalence checking over $\mathbb{F}_p$ by using a new reduction algorithm. To perform equivalence checking with a shorter time and a smaller memory size, we extract $\mathbb{F}_p$ polynomials from a target circuit. The proposed method can be considered to be a generalization of the non-hierarchical verification of multiple-valued characteristics because the conventional methods assume that $\mathbb{F}_2$ polynomials are extracted from a target netlist. Note that the target circuit is described as a netlist of $\mathbb{F}_p$ adders and multipliers (which correspond to combinational circuits consisting of XOR and AND over $\mathbb{F}_2$, respectively), which are used to verify arithmetic circuits over $\mathbb{F}_{p^m}$. Thus, the lowest-level component of the netlist should be given by p-valued logics in the verification (and if necessary, the correctness and validity of the lowest-level components, including the conversion between binary and p-valued signals, should be separately verified). Note that we assume that the building blocks for addition and multiplication are the smallest units of design in HDL and that they are implemented hypothetically with (non-binary) technology.

The proposed method consists of the following four steps:

Step 1: The GF equations of a specification are converted to a polynomial over $\mathbb{F}_p$,

Step 2: A set of polynomials over $\mathbb{F}_p$ representing the given netlist is extracted and its Gröbner basis is derived,

Step 3: The PO variables are reduced using the Gröbner basis derived from step 2,

Step 4: The equivalence between the polynomials of step 1 and step 3 are checked through a comparison.

These steps generally follow the conventional method in [13], and the proposed method can be seen as an extension of the method for $p > 2$.

Step 1 obtains the GF equations of the specification over $\mathbb{F}_p$. For a concrete explanation of step 1, we describe the case in which the specification of a GF multiplier is given as $Z = A \times B$, where $A, B \in \mathbb{F}_{p^m}$ are the inputs, and Z is the output. Let $a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_m, z_1, z_2, \ldots, z_m \in \mathbb{F}_p$ be the variables representing the coefficients of A, B, and Z, respectively. The inputs A, and B and the output Z are given by

$$A = a_1 + a_2\alpha + \cdots + a_m\alpha^{m-1} = \sum_{i=1}^{m} a_i\alpha^{i-1}, \tag{2}$$

$$B = b_1 + b_2\alpha + \cdots + b_m\alpha^{m-1} = \sum_{i=1}^{m} b_i\alpha^{i-1}, \tag{3}$$

[†]ZDDs are typically explained from the viewpoint of combinational sets [21]. However, in this study, we explain them using the positive Davio expansion to clarify the difference between ZDDs and BDDs [22]

$$Z = z_1 + z_2\alpha + \cdots + z_m\alpha^{m-1} = \sum_{i=1}^{m} z_i\alpha^{i-1}, \tag{4}$$

respectively. Here, α is an indeterminate of an irreducible polynomial $P(x)$ (i.e., $P(\alpha) = 0$). Substituting (2)–(4) into $Z = AB$, we derive

$$\begin{aligned}\sum_{i=1}^{m} z_i\alpha^{i-1} &= \left(\sum_{i=1}^{m} a_i\alpha^{i-1}\right)\left(\sum_{i=1}^{m} b_i\alpha^{i-1}\right)\\ &= \sum_{i=1}^{m}\sum_{j=1}^{m} a_i b_j \alpha^{i+j-2}\\ &= \sum_{i=1}^{m} f_i(a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_m)\alpha^{i-1},\end{aligned} \tag{5}$$

where $f_i(a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_m)$ is an element of $\mathbb{F}_p$, i.e., a polynomial of the input variables $a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_m$. We can obtain the specification of a GF multiplier by explicitly solving $f_i(a_1, a_2, \ldots, a_m, b_1, b_2, \ldots, b_m)$.

Step 2 extracts simultaneous algebraic equations over the prime field $\mathbb{F}_p$ from the given netlist. As mentioned above, the smallest component of the netlist for the proposed method is an arithmetic module (i.e., adder and multiplier) over $\mathbb{F}_p$. For the case in which the characteristic p is 2, all logical operations, such as AND, XOR, and OR, have a one-to-one correspondence with the operations over $\mathbb{F}_2$.

To formally explain step 2, we define the notation. First, the smallest component that appears in a given circuit description (i.e., an operation over a prime field $\mathbb{F}_p$) is called a node, and the connection between them is called an edge. Note that the PIs and POs of the circuit can also be handled as nodes. If the characteristic is 2, the edge is the same as a wire. Let l be the total number of edges in the circuit, and let $w_1, w_2, \ldots, w_i, \ldots, w_j, \ldots, w_l$ be the variables of all edges, where $j > i$ holds if an edge w_j is closer to the POs than an edge w_i. In addition, let u and v be the numbers of edges connected to the PIs and the POs, respectively. Therefore, $w_1, w_2, \ldots, w_u$ and $w_{l-v+1}, w_{l-v+2}, \ldots, w_l$ are connected to the PIs and POs, respectively. All edges, except those connected to the PIs, must be connected to the output of an intermediate node. That is, there exists a polynomial p_i that represents the relation between the output w_i and its input variables. Formally, the relation $p_i = w_i - \mathrm{tail}(p_i)$ holds, where $\mathrm{tail}(p_i)$ is the remaining part of $p_i - w_i$. Note that we consider a polynomial set $G = J \cup J_0$, where $J = \{p_i \mid u+1 \le i \le l\}$ is the set of all input–output relations, and $J_0 = \{w_i^q - w_i \mid 1 \le i \le l\}$ is the set of vanishing polynomials. For the case of $p \ge 2$, we can extend [7, Theorem 6.1], such that the polynomial set G can be regarded as a Gröbner basis with RTTO (i.e., the leading term of each polynomial $\mathrm{lt}(p_i)$ is the output edge w_i). Therefore, in step 2, the simultaneous algebraic equations over the prime field $\mathbb{F}_p$ that are extracted from the netlist in the HDL format become the Gröbner basis G.

Step 3 reduces the variables that represent POs $w_{l-v+1}, \ldots, w_l$ by the Gröbner basis, which are denoted by G. Algorithm 1 shows the algorithmic description of the proposed reduction, which is considered to be a generalization of the conventional method [26] for the polynomial ring over $\mathbb{F}_2$. First, one of the edges connected to the POs is assigned to the variable z in line 2, and it is then reduced by G in lines 3–6.

Algorithm 1 Algorithm of the polynomial reduction

Require: $w_1, \ldots, w_l$: All edges, $p_{u+1}, \ldots, p_l$: All the polynomials of input–output relations
Ensure: $p'_{l-v+1}, \ldots, p'_l$: Canonical representations of all edges connected to the POs

1: **for** $k \leftarrow 1$ to l **do**
2: $\quad z \leftarrow w_{l-v+k}$
3: $\quad$ **while** z is not a canonical representation **do**
4: $\quad\quad p_i \leftarrow \mathrm{GetPoly}(z)$ ▷ Get $p_i \in G$ such that $\mathrm{lt}(p_i) \mid \mathrm{lt}(z)$.
5: $\quad\quad z \leftarrow \mathrm{Reduce}(z, p_i)$ ▷ Reduce z by p_i.
6: $\quad$ **end while**
7: $\quad p'_{l-v+k} \leftarrow z$
8: **end for**

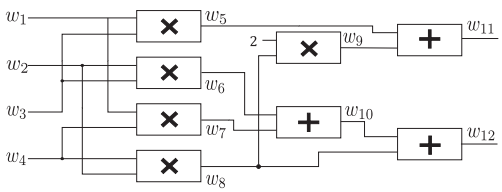

Fig. 1 GF multiplier over $\mathbb{F}_{3^2}$.

More precisely, in line 4, we obtain the polynomial p_i using "GetPoly," such that the leading term $\mathrm{lt}(z)$ is divisible by $\mathrm{lt}(p_i)$, where i is an integer among $\{u+1, u+2, \ldots, l\}$. In line 5, we perform a reduction of z by p_i, where the leading term $\mathrm{lt}(p_i) = w_i$ is replaced with $\mathrm{tail}(p_i) = -(p_i - w_i)$. We repeat the procedure of lines 4–5 until z cannot be reduced. Finally, we assign z to p'_{l-v+k} in line 7. We can easily confirm that Algorithm 1 eventually halts, owing to the definition of the Gröbner basis. In addition, the computational cost of Algorithm 1 increases in proportion to the number of nodes because it depends on the number of polynomial reductions.

Step 4 checks the equivalence between the polynomials obtained in step 3 $p'_{l-v+1}, \ldots, p'_l$ with those given in step 1. Finally, we verify the correctness of the circuit functionality by checking whether they are equal.

Example 3.1. *Figure 1 shows an example of a multiplier over $\mathbb{F}_{3^2}$ (i.e., $p = 3$ and $m = 2$.) In Fig. 1, blocks with + and $\times$ indicate the modules for addition and multiplication over $\mathbb{F}_3$, respectively. Note that the following procedure is almost the same as the conventional ($p = 2$) one.*

Step 1: *We first convert the specification $Z = AB$ to the polynomials over $\mathbb{F}_3$, where A and B are the input variables, and Z is the output variable defined in $\mathbb{F}_{3^2}$. According to the PI and PO edges in Fig. 1, A, B, and Z are given as follows*

$$A = \alpha w_2 + w_1, \tag{6}$$

$$B = \alpha w_4 + w_3, \tag{7}$$

$$Z = \alpha w_{12} + w_{11}, \tag{8}$$

where α is an indeterminate that is determined by an irreducible polynomial $P(x) = x^2 + 2x + 1$ (i.e., $\alpha^2 + 2\alpha + 1 = 0$).

From (6)–(8), the specification $Z = AB$ is converted to

$$\alpha w_{12} + w_{11} = \alpha(w_4w_2 + w_4w_1 + w_3w_2) + 2w_4w_2 + w_3w_1. \tag{9}$$

Because α is the indeterminate, we have

$$w_{12} = w_4w_2 + w_4w_1 + w_3w_2, \tag{10}$$
$$w_{11} = 2w_4w_2 + w_3w_1. \tag{11}$$

Step 2: *According to Fig. 1, we obtain the polynomials over $\mathbb{F}_3$ as follows:*

$$p_5 = w_5 - w_3w_1, p_6 = w_6 - w_3w_2, p_7 = w_7 - w_4w_1,$$
$$p_8 = w_8 - w_4w_2, p_9 = w_9 - 2w_8,$$
$$p_{10} = w_{10} - (w_7 + w_6), p_{11} = w_{11} - (w_9 + w_5),$$
$$p_{12} = w_{12} - (w_{10} + w_8).$$

Step 3: *Using Algorithm 1, we reduce the output variables w_{11} and w_{12} by G. The output variable w_{11} is reduced to*

$$w_{11} \xrightarrow{p_{11}} w_9 + w_5 \xrightarrow{p_9} 2w_8 + w_5$$
$$\xrightarrow{p_8} w_5 + 2w_4w_2 \xrightarrow{p_5} 2w_4w_2 + w_3w_1. \tag{12}$$

Similarly, the other variable w_{12} is reduced to

$$w_{12} \xrightarrow{p_{12}} w_{10} + w_8 \xrightarrow{p_{10}} w_8 + w_6 + w_7, \tag{13}$$
$$\xrightarrow{p_8} w_7 + w_6 + w_4w_2 \xrightarrow{p_7} w_6 + w_4w_2 + w_4w_1$$
$$\xrightarrow{p_6} w_4w_2 + w_4w_1 + w_3w_2. \tag{14}$$

Step 4: *Finally, we compare the canonical forms derived in step 3 with the specification polynomials obtained in step 1. In this case, we finally confirm the correctness of the circuit shown in Fig. 1.*

3.2 Extension of ZDD to GF Arithmetic Circuits with Multiple-Valued Characteristics

In this subsection, we introduce an extension of ZDD to the GF arithmetic circuits with multiple-valued characteristics called Galois-field binary moment diagrams (GFBMDs) to make the abovementioned method (particularly step 3) faster.

The generation of GFBMDs is similar to that of binary moment diagrams (*BMDs), which are used to verify integer arithmetic circuits. *BMDs are given by the following expansion:

$$f(x) = c_1f_1 + xc_2f_2, \tag{15}$$

where c_1 and c_2 are integer coefficients, and the other variables are the same as those in Eq. (1). Using the expansion recursively, we can represent an arbitrary integer coefficient polynomial, where the order of each variable is one or lower. In contrast, the introduced GFBMD is given by Eq. (15), where the coefficients c_1 and c_2 are defined as the elements over the prime field $\mathbb{F}_p$. Note that a GFBMD can be considered as an extension of a ZDD because Eq. (1) is a special case of Eq. (15). GFBMDs have various applications; however, they cannot represent all polynomials with a degree of more than one.

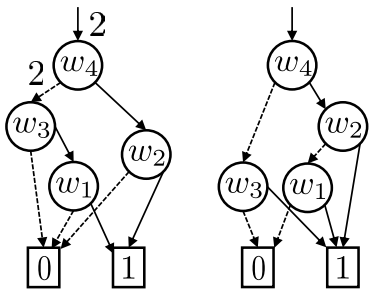

Fig. 2 GFBMDs of $2w_4w_2 + w_3w_1$ (left) and $w_4w_2 + w_4w_1 + w_3w_2$ (right) shown in Example 3.1.

Like *BMDs, GFBMDs are not unique in the case of representing polynomials that consist of two or more variables. In general, for the uniqueness of *BMDs, we normalize the coefficients in Eq. (15) by factoring out their greatest common divisor. However, this does not guarantee the uniqueness of GFBMDs because their coefficients are the elements of the prime field $\mathbb{F}_p$, not those of the ring of integers. Therefore, we employ the coefficient of the expanded variable (i.e., c_2) as the normalization factor[†]. To represent the polynomials over $\mathbb{F}_p$ using GFBMDs, the addition and multiplication of polynomials must be carried out on GFBMDs. This can be implemented using the APPLY algorithm corresponding to the *BMDs.

In conclusion, the difference between *BMDs and GFBMDs originates from the normalization factor for the weights; thus, the construction algorithm of GFBMDs is similar to that of *BMDs. More precisely, GFBMDs can be implemented by changing only the "NormWeight" function, as shown in [27]

Figure 2 shows examples of GFBMDs that correspond to the outputs of the circuit shown in Example 3.1. The GFBMDs consist of some nodes that represent the variables, two types of edges, which are denoted as solid and dotted lines, and 0 and 1-terminal nodes. The solid and dotted lines from a node w_i indicate that the variable w_i and "1" are multiplied respectively. In addition, the numbers next to the edges indicate the multiplier values. These edges and numbers correspond to the decomposition denoted in Eq. (15). For example, in the left of Fig. 2, the path through the root node, the dotted line of a node w_4, the solid line of a node w_3, the solid line of a node w_1, and the 1-terminal node represent a polynomial $2\times2\times w_3\times w_1 = w_3w_1$. In this way, each path from the root node to a terminal node corresponds to a polynomial that is defined as the product term of all variables/labels denoted by the nodes, edges, and the terminal

[†]GFBMD normalized by c_2 is well-defined because all elements over the prime field have their inverse elements, and the uniqueness of GFBMDs defined in the above manner can easily be confirmed by mathematical induction on the depth of GFBMDs.

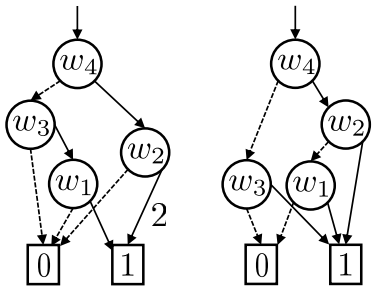

Fig. 3 *BMDs of $2w_4w_2 + w_3w_1$ (left) and $w_4w_2 + w_4w_1 + w_3w_2$ (right) shown in Example 3.1.

node on the path. The polynomial denoted by GFBMD is defined as the sum of the product terms that correspond to all the paths.

For comparison, Fig. 3 shows two examples of *BMDs that correspond to the same equations as the GFBMDs. The *BMD of $2w_4w_2 + w_3w_1$ is different from the corresponding GFBMDs, owing to the difference in the normalization method. In the examples, we can observe that the multiplier value of the solid line of a node w_2 on the left of Fig. 3 is two; however, this weight is unacceptable in GFBMDs because the normalization factor of this node is given by the weight of the solid line (i.e., two). However, the GFBMD and *BMD of $w_4w_2 + w_4w_1 + w_3w_2$ are the same because their normalization factors are equal in this case.

4. Experimental Evaluation

In this section, we demonstrate the effectiveness of the proposed method using verification experiments of 2-input GF multipliers over $\mathbb{F}_{p^m}$. We first show the experimental results using the list representation of the polynomials (i.e., without GFBMDs)[†]. Then, we compare the performance of the proposed method with the lists (i.e., without GFBMDs) to that with GFBMDs.

4.1 Experimental Verification without GFBMDs

In this subsection, we present the results of the verification experiments using straightforward list representations. We evaluate the verification times for GF multipliers with a characteristic p of 2, 3, 5, and 7 for various extension degrees m, from 8–256. Table 1 shows the verification time of the GF multipliers by using the proposed method and the number of addition and multiplication modules. We confirm here that the verification time depends on m, but not on the characteristic. In GF multipliers, the structure/size of arithmetic modules over the prime field is determined by the characteristic, and the number of arithmetic modules over $\mathbb{F}_{p^m}$ is primarily determined by the degree of extension. According to Table 1, an increase in the characteristic has minimal impact on the computational cost. This is because the number of reductions (i.e., the number of adders and multipliers over $\mathbb{F}_p$ in the circuit) does not depend on the characteristics. Note that the verification time strongly depends on the number of reductions in Algorithm 1. Thus, the verification time basically depends on the degree of extension; however, the verification time differs slightly in p, owing to the difference in irreducible polynomials.

Table 1 also shows the number of terms in the irreducible polynomials for each multiplier and the verification time divided (i.e., normalized) by it. In this evaluation, the number of terms in the irreducible polynomials are smallest and largest when the characteristics are 5 and 2, respectively. The verification times for a characteristic of 2 are larger than those for 5; however, the difference between the normalized verification times shown in the third row is trivial, which indicates that the difference in the verification time primarily arises from the number of terms in irreducible polynomials.

In addition, we evaluate the performance of the proposed method when a target GF multiplier includes a bug. Here, we insert a bug by connecting an input edge of a node in a GF multiplier to an incorrect node output. Note that we do not deal with logical bugs and/or bugs that cannot be represented by addition and multiplication modules because we assume that the verified circuit is given as a combination of addition and multiplication modules. There is a possibility that other types of bugs can be inserted in logic optimization, synthesis, etc., and further evaluation with such bugs should be considered in future work. Note that the conventional hierarchical methods, such as GF-ACG, require an unrealistic time to verify a circuit even with such a simple bug because they use the Büchberger algorithm to derive a Gröbner basis. Table 2 shows the verification time of the GF multipliers with the abovementioned bug and the ratio of the verification times of the multipliers with and without the bug. In addition, Fig. 4 shows the verification time of the GF multipliers with and without the bug. The table shows that the verification times of buggy multipliers are smaller than those of bug-free ones. This is because our method compares the polynomials of the specification and canonical form every time each PO variable is reduced, and it immediately ends when the difference is found. From the table, we can infer that the proposed method efficiently verifies buggy multipliers.

From the above results, we also confirm that the formal verification can be performed in approximately eight minutes using the proposed method, even for a practical multiplier with a degree of extension of 256. Although the GF-ACG-based method can also verify GF multipliers with multiple characteristics, it is necessary to describe the circuit in a finely hierarchical manner. Note again the limitation that verification cannot be performed unless the circuit is given in GF-ACG. However, the proposed method can accept any circuit description for which the smallest components are given by arithmetic modules over the prime field. Thus, the proposed method can be applied to a wider range of circuit descriptions.

[†]List representation means that the polynomials are represented by a list that contains each term of the polynomial as an element.

Table 1 Verification time of GF multiplier

	$m = 8$				$m = 16$				$m = 32$			
	$p = 2$	$p = 3$	$p = 5$	$p = 7$	$p = 2$	$p = 3$	$p = 5$	$p = 7$	$p = 2$	$p = 3$	$p = 5$	$p = 7$
Num. of additions	143	85	56	84	627	487	240	360	2,545	2,488	992	1,488
Num. of multiplications	64	92	92	120	256	376	376	496	1,024	1,520	1,520	2,016
Num. of terms of IP	5	3	2	3	5	4	2	3	5	5	2	3
Verification time [ms]	6.39	4.83	4.23	5.46	18.9	18.3	9.89	15.2	88.5	140	39.3	69.8
Time / Num. of terms	1.28	1.61	2.12	1.82	3.78	4.58	4.95	5.07	17.7	28.0	19.7	23.3

	m=64				m=128				m=256			
	$p = 2$	$p = 3$	$p = 5$	$p = 7$	$p = 2$	$p = 3$	$p = 5$	$p = 7$	$p = 2$	$p = 3$	$p = 5$	$p = 7$
Num. of additions	10,095	6,051	4,032	6,048	40,704	40,673	16,256	24,385	163,324	195,875	65,280	163,209
Num. of multiplications	4,096	6,112	6,112	8,128	16,394	24,512	24,512	32,640	65,536	98,176	98,176	130,819
Num. of terms of IP	5	3	2	3	5	5	2	3	5	6	2	5
Verification time [ms]	1030	599	424	706	21,100	26,700	6,290	15,300	315,000	554,000	105,000	480,000
Time / Num. of terms	206	200	212	235	4,220	5,340	3,150	5,100	63,000	92,300	52,500	96,000

Table 2 Verification time of GF multiplier with and without bugs.

	$m = 8$				$m = 16$				$m = 32$			
	$p = 2$	$p = 3$	$p = 5$	$p = 7$	$p = 2$	$p = 3$	$p = 5$	$p = 7$	$p = 2$	$p = 3$	$p = 5$	$p = 7$
Verification time of bug-free one [ms]	6.39	4.83	4.23	5.46	18.9	18.3	9.89	15.2	88.5	140	39.3	69.8
Verification time of buggy one [ms]	4.54	5.17	3.03	4.91	15.9	16.2	12.2	15.5	70.5	111	42.3	70.5
Ratio	0.710	1.07	0.716	0.899	0.841	0.885	1.23	1.02	0.797	0.793	1.08	1.01

	$m = 64$				$m = 128$				$m = 256$			
	$p = 2$	$p = 3$	$p = 5$	$p = 7$	$p = 2$	$p = 3$	$p = 5$	$p = 7$	$p = 2$	$p = 3$	$p = 5$	$p = 7$
Verification time of bug-free one [ms]	1,030	599	424	706	21,100	26,700	6,290	15,300	315,000	554,000	105,000	480,000
Verification time of buggy one [ms]	776	492	357	677	18,500	25,200	5,870	14,300	239,000	506,000	90,800	449,000
Ratio	0.753	0.821	0.842	0.959	0.877	0.944	0.933	0.935	0.759	0.913	0.865	0.935

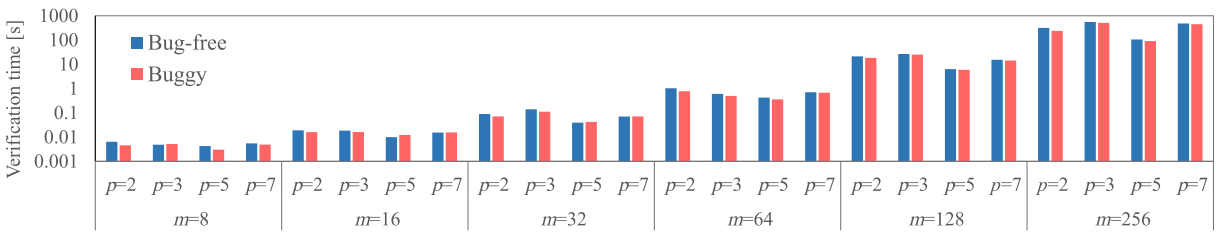

Fig. 4 Verification time of GF multipliers with and without bugs.

4.2 Experimental Verification Using GFBMDs

In this subsection, we demonstrate the effectiveness of GFBMD through verification experiments that are compatible with those above. We evaluate the verification times for GF multipliers with a characteristic p of 2, 3, 5, and 7 for extension degrees m from 64–256 because such large multipliers cannot be verified completely using computer simu-

Table 3 Verification time of GF multiplier with and without GFBMDs

	$m = 64$				$m = 128$				$m = 256$			
	$p = 2$	$p = 3$	$p = 5$	$p = 7$	$p = 2$	$p = 3$	$p = 5$	$p = 7$	$p = 2$	$p = 3$	$p = 5$	$p = 7$
W/ GFBMDs [s]	2.05	9.77×10^{-1}	6.16×10^{-1}	1.01	1.54×10	1.28×10	4.53	6.70	1.20×10^{2}	1.35×10^{2}	3.42×10	9.58×10
W/o GFBMDs [s]	1.03	5.99×10^{-1}	4.24×10^{-1}	7.06×10^{-1}	2.11×10^{1}	2.67×10^{1}	6.29	1.53×10^{1}	3.15×10^{2}	5.54×10^{2}	1.05×10^{2}	4.80×10^{2}
Ratio	5.03×10^{-1}	6.13×10^{-1}	6.88×10^{-1}	6.99×10^{-1}	1.37	2.09	1.40	2.28	2.63	4.10	3.07	5.01

Table 4 Number of terms in the largest polynomial over $\mathbb{F}_2$ in each multiplier

Characteristic [p]	$m = 2$	$m = 3$	$m = 4$
2	3	5	7
3	56	1290	17848
5	1130	94662	2028364

lations with exhaustive test patterns. Note that we can use GFBMDs to verify such GF multipliers because polynomials with degrees of two or higher do not appear during the verification process. Table 3 shows the verification times with and without GFBMDs and their ratios. From the table, we can confirm that the verification time with GFBMDs becomes relatively shorter than that without GFBMDs as the degree of extension increases. In particular, the verification with GFBMD is approximately 5 times faster than that without GFBMD in the case of $p = 7$ and $m = 256$. This is related to the complexity of additions and multiplications of polynomials on GFBMDs. Generally, the addition and multiplication of polynomials represented by the list have complexity $O(NM)$, where N and M are the numbers of terms in the given two polynomials, respectively. However, the conventional *BMDs have worst-case complexities of addition and multiplication, which increase linearly and exponentially with the number of variables, respectively, and GFBMDs would have the same complexities. However, as demonstrated by Bryant [8], these exponential cases do not always appear in practical applications. Furthermore, most polynomials that may appear during verification have regularities; thus, they can be factorized according to the decomposition shown in Eq. (15). As a result, the complexity of addition and multiplication using GFBMDs is smaller than that of the straightforward list representation. In fact, Table 3 also supports the above considerations.

We also describe the results for the case where the characteristic p is two. At $p = 2$, the proposed method is the same as the conventional method for GF arithmetic circuits [13]. Therefore, for the characteristic of 2, the performance of our method should be comparable to that of the conventional method. However, our method is 10 times slower than the conventional best method [13]. One of the reasons is that a very fast open-source library, the Colorado University Decision Diagram (CUDD), cannot be used to implement GFBMDs because the data structure provided by CUDD to represent DDs cannot handle the weighted edges of GFBMDs. Therefore, in these experiments, we implemented GFBMDs with C++ from scratch without optimizing our library to the same level as CUDD. Therefore, the GFBMD-based verification can potentially be 10 times faster than the results shown in Table 3, if it receives sufficient optimization.

Finally, we discuss the applicability of the conventional non-hierarchical method, such as in [13], to the verification problem addressed in this study. The conventional method uses a polynomial ring over $\mathbb{F}_2$ to represent the netlist functionality. Thus, if the polynomial that represents a circuit specification over $\mathbb{F}_2$ has a considerable number of terms, it is not applicable to the circuit. As an example, Table 4 shows the maximum number of terms of the specification of GF multipliers with multiple-valued characteristics. From the table, we can confirm that the number of terms increases exponentially when the characteristic is larger than two. This indicates that the application of the conventional non-hierarchical method to GF arithmetic circuits with multiple-valued characteristics is difficult in practice, even with optimization techniques such as ZDD manipulation, as shown in [13].

5. Conclusions

In this study, we presented a novel formal method to verify GF arithmetic circuits with multiple-valued characteristics on the basis of computer algebra. In the proposed method, the polynomial ring over $\mathbb{F}_p$ is used instead of $\mathbb{F}_2$, which was used in previous methods, to perform equivalence checking with less computational time and a smaller memory size. The new algorithm proposed in this study efficiently performed the polynomial reduction using the Gröbner basis, even for GFs with multiple-valued characteristics. In addition, we introduce an extension of ZDDs to efficiently perform polynomial reductions. We experimentally demonstrated that our method can verify large GF arithmetic circuits, such as a GF multiplier with a characteristic of seven and extension degree of 256.

The investigation of a new type of decision diagram that can efficiently represent a polynomial with a high degree should be considered in future work because some practical circuits may have redundant representations, which may generate high-degree polynomials during verification.

References

[1] E. Savas and C.K. Koc, "Finite field arithmetic for cryptography," IEEE Circuits and Systems Magazine, vol.10, no.2, pp.40–56, Secondquarter 2010.
[2] I. Duursma and H.S. Lee, "Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$," Advances in Cryptology - ASIACRYPT 2003, ed. C.S. Laih, Lect. Notes Comput. Sci., vol.2894, pp.111–123, Springer, Berlin, Heidelberg, 2003.
[3] D.J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.Y. Yang, "High-speed high-security signatures," J. Cryptographic Engineering, vol.2, no.2, pp.77–89, Sept. 2012.
[4] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," J. Cryptology, vol.17, no.4, pp.297–319, Sept. 2004.
[5] I. Duursma and K. Sakurai, "Efficient algorithms for the jacobian

variety of hyperelliptic curves $y^2 = x^p - x + 1$ over a finite field of odd characteristic p," Coding Theory, Cryptography and Related Areas, ed. J. Buchmann, T. Høholdt, H. Stichtenoth, and H. Tapia-Recillas, Berlin, Heidelberg, pp.73–89, Springer, 2000.
[6] E. Lee, H.S. Lee, and Y. Lee, "Eta pairing computation on general divisors over hyperelliptic curves $y^2 = x^p - x + d$," J. Symbolic Computation, vol.43, no.6, pp.452–474, June 2008.
[7] J. Lv, P. Kalla, and F. Enescu, "Efficient Gröbner Basis Reductions for Formal Verification of Galois Field Arithmetic Circuits," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.32, no.9, pp.1409–1420, Sept. 2013.
[8] R.E. Bryant and Y.A. Chen, "Verification of Arithmetic Circuits with Binary Moment Diagrams," Design Automation Conference, pp.535–541, Jan. 1995.
[9] R. Drechsler and D. Sieling, "Binary decision diagrams in theory and practice," International J. Software Tools for Technology Transfer, vol.3, no.2, pp.112–136, May 2001.
[10] J. Jain, J. Bitner, M.S. Abadir, J.A. Abraham, and D.S. Fussell, "Indexed BDDs: Algorithmic advances in techniques to represent and verify Boolean functions," IEEE Trans. Comput., vol.46, no.11, pp.1230–1245, Nov. 1997.
[11] B. Bollig, M. Sauerhoff, D. Sieling, and I. Wegener, "Hierarchy theorems for kOBDDs and kIBDDs," Theoretical Computer Science, vol.205, no.1, pp.45–60, Sept. 1998.
[12] B. Becker, R. Drechsler, and R. Werchner, "On the relation between BDDs and FDDs," LATIN '95: Theoretical Informatics, ed. R. Baeza-Yates, E. Goles, and P.V. Poblete, Lect. Notes Comput. Sci., vol.911, pp.72–83, Springer, Berlin, Heidelberg, 1995.
[13] U. Gupta, P. Kalla, and V. Rao, "Boolean Gröbner Basis Reductions on Finite Field Datapath Circuits Using the Unate Cube Set Algebra," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.38, no.3, pp.576–588, March 2019.
[14] A. Ito, R. Ueno, and N. Homma, "Efficient Formal Verification of Galois-Field Arithmetic Circuits Using ZDD Representation of Boolean Polynomials," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., in press.
[15] N. Homma, K. Saito, and T. Aoki, "A formal approach to designing cryptographic processors based on $gf(2^m)$ arithmetic circuits," IEEE Trans. Inf. Forensics Security, vol.7, no.1, pp.3–13, Feb. 2012.
[16] A. Ito, R. Ueno, and N. Homma, "Effective Formal Verification for Galois-field Arithmetic Circuits with Multiple-Valued Characteristics," 2020 IEEE 50th Int. Symp. Multiple-Valued Logic (ISMVL), pp.46–51, Nov. 2020.
[17] N. Homma, K. Saito, and T. Aoki, "Toward Formal Design of Practical Cryptographic Hardware Based on Galois Field Arithmetic," IEEE Trans. Comput., vol.63, no.10, pp.2604–2613, Oct. 2014.
[18] R. Ueno, N. Homma, Y. Sugawara, and T. Aoki, "Formal Approach for Verifying Galois Field Arithmetic Circuits of Higher Degrees," IEEE Trans. Comput., vol.66, no.3, pp.431–442, March 2017.
[19] R. Ueno, N. Homma, and T. Aoki, "Automatic Generation System for Multiple-Valued Galois-Field Parallel Multipliers," IEICE Trans. Inf. & Syst., vol.E100-D, no.8, pp.1603–1610, Aug. 2017.
[20] C. Yu and M. Ciesielski, "Formal Analysis of Galois Field Arithmetic Circuits-Parallel Verification and Reverse Engineering," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.38, no.2, pp.354–365, Feb. 2019.
[21] S.i. Minato, "Zero-suppressed BDDs for set manipulation in combinatorial problems," Proc. 30th International Design Automation Conference, pp.272–277, July 1993.
[22] S.i. Minato, "Zero-suppressed BDDs and their applications," International J. Software Tools for Technology Transfer, vol.3, no.2, pp.156–170, May 2001.
[23] D. Miller and R. Drechsler, "On the construction of multiple-valued decision diagrams," Proceedings 32nd IEEE Int. Symp. Multiple-Valued Logic, pp.245–253, May 2002.
[24] A. Jabir and D. Pradhan, "MODD: A new decision diagram and representation for multiple output binary functions," Automation and Test in Europe Conference and Exhibition Proceedings Design, pp.1388–1389 vol.2, Feb. 2004.
[25] R. Stankovic, "Functional decision diagrams for multiple-valued functions," Proceedings 25th Int. Symp. Multiple-Valued Logic, pp.284–289, May 1995.
[26] T. Pruss, P. Kalla, and F. Enescu, "Efficient Symbolic Computation for Word-Level Abstraction From Combinational Circuits for Verification Over Finite Fields," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.35, no.7, pp.1206–1218, July 2016.
[27] R. Bryant and Y.a. Chen, "Verification of Arithmetic Functions with Binary Moment Diagrams," Technical Report CMUCS, May 1994.

Akira Ito received the B.E. degree in information engineering and the M.S. degree in information sciences from Tohoku University, Japan, in 2017, and 2019, respectively. He is currently enrolled in a doctoral course at Tohoku University. His research interests include arithmetic circuits, formal verification, and hardware security.

Rei Ueno received the B.E. degree in information engineering and the M.S. and Ph.D. degrees in information sciences from Tohoku University, Japan, in 2013, 2015, and 2018, respectively. He is an Assistant Professor at the Research Institute of Electrical Communication, Tohoku University, and is also joining the JST as a researcher for a PRESTO project. His research interests include arithmetic circuits, cryptographic implementations, formal verification, and hardware security. Dr. Ueno received the Kenneth C. Smith Early Career Award in Microelectronics at ISMVL 2017.

Naofumi Homma received the B.E. degree in information engineering, and the M.S. and Ph.D. degrees in information sciences from Tohoku University, Sendai, Japan, in 1997, 1999, and 2001, respectively. From 2001 to 2009, he was an Assistant Professor at the Graduate School of Information Sciences at Tohoku University, and he became an Associate Professor at the same department in 2009. Since 2016, he has been a Professor at the Research Institute of Electrical Communication, Tohoku University. During 2009–2010 and 2016–2017, he was a Visiting Professor at Telecom ParisTech in Paris, France. His research interests include computer arithmetic, electronic design automation methodology, and hardware security. He received an IP Award at the LSI IP Design Award in 2005, the Best Paper Award at the Workshop on Synthesis and System Integration of Mixed Information Technologies in 2007, the RIEC Award in 2012, the Best Symposium Paper Award at the 2013 IEEE International Symposium on Electromagnetic Compatibility, the Best Paper Award at the 2014 IACR Conference on Cryptographic Hardware and Embedded Systems, the Japan Society for the Promotion of Society Prize in 2018, and the German Innovation Award in 2018.