Duality between Multiplication and Modular Reduction

Wieland Fischer[1] and Jean-Pierre Seifert[2]

[1] Infineon Technologies AG
Secure Mobile Solutions
D-81609 Munich, Germany
Wieland.Fischer@infineon.com
[2] Intel Corporation
Systems Technology Laboratory
Corporate Technology Group
Hillsboro, OR 97124, USA
Jean-Pierre.Seifert@intel.com

Abstract. This paper presents a duality between the classical optimally speeded up multiplication algorithm and some “fast” reduction algorithm. For this, the multiplier is represented by the unique signed digit representation with minimal Hamming weight using Reitwiesner's multiplier recoding algorithm. In fact, the present paper proves that this optimal multiplier recoding technique naturally translates into a canonical modular reduction technique. Thus, the resulting reduction algorithm is optimal with respect to its average-time complexity as well. Besides these two new results, our proof of the transfer-theorem serves another interesting purpose: The reason that the considered reduction algorithm from [Sed] is so unknown might lie in the fact that it is rather un-intuitive and no proper understanding was available so far. Therefore, our proper mathematical derivation/explanation solves this lack of understanding.

Keywords: Computer arithmetic, Booth recoding, Canonical signed-digit representation, Modular reduction, Multiplication, Minimum Hamming weight, Optimal algorithm, Signed digit representation, Reitwiesner recoding.

1 Introduction

Without doubt it is clear that the modular multiplication is the basic arithmetical operation of today's asymmetric cryptography, cf. [MvOV]. Therefore, over the past three decades (i.e., since the publication of [RSA]), a substantial amount of research has established new algorithms for the problem of modular multiplication, cf. [Bar,Bri,Mon,Omu,WQ,Sed]. Many of such algorithms have been proven to be very successful in practical implementations and especially for time/area efficient hardware implementations of the modular multiplication, cf. [NMR]. For a thorough overview of this area we refer to the many papers contained in [KP99,KP00,KNP01,KKP02]. However, until now no theoretical investigation of the basic main problem itself has been done so far. Namely, dividing the modular multiplication into its comprising subproblems leaves the two more fundamental problems of *multiplication* and *modular reduction*, which is the basic main problem.

Indeed, starting with the papers by [Boo,Mac,Rei] the multiplication itself is now being very well understood, cf. [Hwa,Knu,Kor,Par,Spa]. Namely, to speed up the classical shift and add method several variants of the so called original Booth algorithm [Boo] have been proposed by [Mac]. These may be divided into two categories: variable shift methods and uniform shift methods. In this paper we will be only concerned with the variable shift method as it intrinsically leads in a natural sense to a uniquely determined optimum method. Seen from an abstract point of view this method aims to recode a multiplier β into an equivalent binary *signed-digit* (SD_2) representation $\bar{\beta}$, such that the number of nonzero digits of $\bar{\beta}$ is minimized. Here, the set of possible digits is $\{-1, 0, +1\}$. Reitwiesner [Rei] proved that the minimal Hamming weight $\omega(\bar{\beta})$ of the SD_2 representation for an n-bit number β is given by $(n+1)/3$, on average. Here, $\omega(\bar{\beta})$ denotes the *Hamming weight* of $\bar{\beta}$, i.e., the number of non-zero digits in the SD_2 representation of β. Later, a simplified proof of this result was given by [AW], and also a closed-form expression for the average weight of signed-digit representations has been developed by [WH]. Reitwiesner also investigated the uniqueness of SD_2 representations. To formalize this, Reitwiesner defined an SD_2 representation $(\bar{\beta}_n, \ldots, \bar{\beta}_0)_{\mathrm{SD}_2}$ as *canonical* or in *nonadjacent* form, if no two adjacent digits are nonzero. He proved that this representation is unique, if the binary representation is viewed as padded with an initial 0. In addition, he presented an algorithm to find this uniquely determined SD_2 representation. Moreover, Reitwiesner proved that among all SD_2 representations, the canonical form has a minimal Hammning weight. I.e., it is optimal in terms of the resulting multiplication speedup.

On the other side, despite the development of many new reduction algorithms, implicitly contained within the many new modular multiplication algorithms, cf. [Bar,Bri,Mon,Omu,WQ,Sed], no theoretical analysis of the modular reduction process itself has been done so far. This is the goal of the present paper. Namely, we prove that the uniquely determined "canonical "counterpart to Booth's algorithm (in Reitwiesner's form) is the reduction algorithm due to Sedlak [Sed]. This is proved by establishing an isomorphism between multiplication algorithms and reduction algorithms relying on the "shift and add/subtract" paradigm. Hence, this reduction algorithm is also optimal in terms of necessary "shift and add/subtract" operations, on average. While the connecting isomorphism might look trivial after seeing it, its proper mathematical proof nevertheless gives a substantial insight into the problem of reduction itself. Namely, the proof reveals a rather intuitive and proper mathematical derivation of Sedlak's reduction algorithm which is new and one of our contributions. Thus, the present paper proves that Sedlak's reduction algorithm is optimal in terms of performance, on average. Besides, it gives a nice mathematical explanation of Sedlak's reduction algorithm, which was not available so far. Moreover, it solves the lack of understanding the reduction process itself. Namely, it shows that the reduction process is nothing else than the dual of the multiplication process. So far, we have not seen in the literature anywhere this interesting connection between multiplication algorithms and reduction algorithms. Therefore, we have the feeling that our results are an interesting contribution to the foundations of computer science.

The paper is organized as follows. Section 2 explains our notations and definitions. Section 3 and Section 4 defines how we are going to see multiplication and reduction algorithms in the rest of the paper. Also, they give important examples of our algorithm notion which are necessary for the later understanding our paper. Section 5 presents the connecting morphism, and Section 6 shows an explicit construction of our morphism relating to "real" algorithms. Finally, Section 7 presents the formal Duality Theorem whose technical proof is rather educational in understanding the reduction process itself.

2 Notation

Throughout the paper we use the following notations: We set $\mathbb{N} = \{1, 2, 3, \ldots\}$ and $\mathbb{N}_0 = \{0, 1, 2, \ldots\}$. For intuition, we will also use sometimes the definitions $\mathcal{N} := \mathbb{N}$, $\mathcal{A} := \mathbb{N}$, $\mathcal{Z} := \mathbb{Z}$ and $\mathcal{B} := \mathbb{Z}$.

The set of all maps $f\colon \mathbb{N} \to \mathbb{N}_0 \times \{-1, 0, 1\}$ will be abbreviated by $\mathrm{Map}(\mathbb{N},\ \mathbb{N}_0 \times \{-1, 0, 1\})$ and $\mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1, 0, 1\}) = \{f \in \mathrm{Map}(\mathbb{N},\ \mathbb{N}_0 \times \{-1, 0, 1\}) : f(x) \neq (0, 0)$ for finitely many $x\}$. For $f \in \mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1, 0, 1\})$ we set $\|f\| := \max\{i \in \mathbb{N} : f(i) \neq (0, 0)\} \cup \{0\}$.

For any $(\beta_{n-1}, \ldots, \beta_0) \in \{0, 1\}^n$ the symbol $(\beta_{n-1}, \ldots, \beta_0)_2$ denotes the integer $\beta := \sum_0^{n-1} \beta_i \cdot 2^i$. For $(\bar{\beta}_n, \ldots, \bar{\beta}_0) \in \{-1, 0, 1\}^{n+1}$ the symbol $(\bar{\beta}_n, \ldots, \bar{\beta}_0)_{\mathrm{SD}_2}$ denotes the integer $\beta := \sum_0^n \bar{\beta}_i \cdot 2^i$. We strictly distinguish between the tuples $(\ldots)$ and their value $(\ldots)_2$ or $(\ldots)_{\mathrm{SD}_2}$.

Moreover, $\ell(n)$ denotes the bitlength of the integer $|n|$, i.e., $2^{\ell(n)-1} \leq |n| < 2^{\ell(n)}$.

3 Formal Multiplication Algorithms

A multiplication algorithm often has the following form:

input: α, β
output: $\alpha \cdot \beta$
$Z := 0$
loop
  $Z := Z \cdot 2^s + v \cdot \alpha$ /* for some $s \in \mathbb{N}_0, v \in \{-1, 0, 1\}$, depending on β */
endloop
return Z

We want to formalize this notion in the following way:

Definition 1. *A formal multiplication algorithm for* $(\alpha, \beta) \in \mathcal{A} \times \mathcal{B}$ *denoted by* $\mathrm{mul} \equiv ((s_1, v_1), \ldots, (s_n, v_n))$ *is a collection of integers* $s_1, \ldots, s_n \in \mathbb{N}_0$ *and signs* $v_1, \ldots, v_n \in \{-1, 0, 1\}$ *(for some* n*) so that the following assignments:*

1. $Z_0 := 0$
2. $Z_{i+1} := Z_i \cdot 2^{s_{i+1}} + v_{i+1} \cdot \alpha, \quad$ *for* $i = 0, \dots, n-1$,

result in $Z_n = \alpha \cdot \beta$. *The set of all formal multiplication algorithms for* (α, β) *is denoted by* $\mathcal{FMA}_{(\alpha,\beta)}$.

Remark 1. A formal multiplication algorithm $\mathrm{mul} \in \mathcal{FMA}_{(\alpha,\beta)}$ can be seen as an element of $\mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1, 0, 1\})$, where

$$\mathrm{mul}(i) = \begin{cases} (s_i, v_i) & \text{if } i \leq n \\ (0, 0) & \text{if } i > n \end{cases}$$

and this is the way we are going to see these objects.

Note 1. Clearly, every map $f \in \mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1, 0, 1\})$ can be interpreted as a formal multiplication algorithm for some (α, β).

Definition 2. *For any* $\mathrm{mul} \in \mathcal{FMA}_{(\alpha,\beta)}$ *we define the norm of* mul *to be* $\|\mathrm{mul}\|$ *interpreted as element of* $\mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1, 0, 1\})$.

Example 1. We define the simplest formal multiplication algorithm $\mathrm{mul}_{\mathrm{simple}}(\alpha, \beta) \in \mathcal{FMA}_{(\alpha,\beta)}$. $\mathrm{mul}_{\mathrm{simple}}(\alpha, \beta) = ((s_1, v_1), \dots, (s_n, v_n))$ with $n := |\beta|$ is given by $s_1 = s_2 = \cdots = s_n = 0$ and $v_1 = v_2 = \cdots = v_n = \mathrm{sign}(\beta)$. This is just "adding α exactly β times".

Example 2. The classical "double-and-add"-method is transformed into $\mathrm{mul}_{\mathrm{class}}(\alpha, \beta) \in \mathcal{FMA}_{(\alpha,\beta)}$ as follows. Let $(\beta_{n-1}, \dots, \beta_0)_2$ be the binary representation of $|\beta|$, then $\mathrm{mul}_{\mathrm{class}}(\alpha, \beta) = ((s_1, v_1), \dots, (s_n, v_n))$ is given by $s_1 = s_2 = \cdots = s_n = 1$ and $v_i = \mathrm{sign}(\beta) \cdot \beta_i$ for $i = 1, \dots, n$.

Example 3. The "famous" Booth algorithm also can be transformed into some formal multiplication algorithm $\mathrm{mul}_{\mathrm{Booth}}(\alpha, \beta) \in \mathcal{FMA}_{(\alpha,\beta)}$. This uses the representation of a number in the signed digit notation SD_2. Let $\beta = (\beta_{n-1}, \dots, \beta_0)_2$ be positive, then there exist representations $\bar{\beta} = (\bar{\beta}_n, \dots, \bar{\beta}_0)$ of β in the form $\beta = \sum_{i=0}^{n} \bar{\beta}_i \cdot 2^i$, where $\bar{\beta}_i \in \{-1, 0, 1\}$. Among these there is (at least) one with minimal Hamming weight $\omega(\bar{\beta})$. With algorithms described in [Hwa,Kor,Par,Spa] such representations can be efficiently obtained. We fix one of these algorithms and denote it by $\mathrm{Booth}\colon (\beta_{n-1}, \dots, \beta_0)_2 \longmapsto (\bar{\beta}_n, \dots, \bar{\beta}_0)$. If we define

$$\begin{aligned} \mathcal{SD}_2 :&= \{0\} \cup \Big(\bigcup_{i=0}^{\infty} \big(\{-1, 1\} \times \{-1, 0, 1\}^i\big)\Big) \\ &= \{(\bar{\beta}_n, \dots, \bar{\beta}_0) \in \{-1, 0, 1\}^n : n \in \mathbb{N}, \bar{\beta}_n \neq 0\} \cup \{0\}, \end{aligned}$$

then Booth can be interpreted as a map $\mathrm{Booth}\colon \mathbb{N} \longrightarrow \mathcal{SD}_2$ or even $\mathrm{Booth}\colon \mathbb{Z} \longrightarrow \mathcal{SD}_2$, with $\mathrm{Booth}(\beta) = -\mathrm{Booth}(-\beta)$ for negative β. In order to define a formal multiplication algorithm for (α, β), we need another map

$$\mathrm{step}\colon \mathcal{SD}_2 \longrightarrow \mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1, 0, 1\});$$

this is given as follows. Let $\bar{\beta} = (\bar{\beta}_n, \ldots, \bar{\beta}_0) \in \mathcal{SD}_2$ with $\bar{\beta}_n \neq 0$, and let $n =: m_1 > m_2 > \cdots > m_x$ be such indices i for which $\bar{\beta}_i \neq 0$. Then, with $m_0 := n$

$$(s_i, v_i) := (m_{i-1} - m_i,\ \bar{\beta}_{m_i}) \quad \text{for } i = 1, \ldots, x.$$

If $m_x > 0$, i.e., if $\bar{\beta}_0 = 0$, then additionally we set $(s_{x+1}, v_{x+1}) := (m_x, \bar{\beta}_0)$. This defines the step-map. Now,

$$\mathrm{mul}_{\mathrm{Booth}}(\alpha, \beta) := \mathrm{step}(\mathrm{Booth}(\beta)).$$

Remark 2. Note that for $\bar{\beta} = (\bar{\beta}_n, \ldots, \bar{\beta}_0) \in \mathcal{SD}_2$ and $((s_1, v_1), \ldots, (s_x, v_x)) := \mathrm{step}(\bar{\beta})$ the equality $n = s_1 + \cdots + s_x$ holds. Or, in other words $(\bar{\beta}_n, \ldots, \bar{\beta}_0) = (v_1, \ldots, v_2, \ldots, v_{x-1}, \ldots, v_x)$ where v_i on the position of $\bar{\beta}_{n-s_1-\cdots-s_i}$ is different from zero and all other positions are zero.

4 Formal Reduction Algorithms

We also want to formalize reduction algorithms which have a similar structure of "shifting and adding/subtracting".

Definition 3. *A formal reduction algorithm for* $(\nu, \zeta) \in \mathcal{N} \times \mathcal{Z}$ *denoted by* $\mathrm{red} \equiv ((s_1, v_1), \ldots, (s_n, v_n))$ *is a collection of integers* $s_1, \ldots, s_n \in \mathbb{N}_0$ *and signs* $v_1, \ldots, v_n \in \{-1, 0, 1\}$ *(for some* n*) so that the following assignments:*

1. $c := s_1 + \cdots + s_n$
2. $Z_0 := \zeta \cdot 2^{-c}$ *(computation in* $\mathbb{Q}$*)*
3. $Z_{i+1} := Z_i \cdot 2^{s_{i+1}} + v_{i+1} \cdot \nu$*, for* $i = 0, \ldots, n-1$ *(computation in* $\mathbb{Q}$*)*

result in $Z_n = \zeta \bmod \nu$*. The set of all formal reduction algorithms for* (ν, ζ) *is denoted by* $\mathcal{FRA}_{(\nu,\zeta)}$*.*

Remark 3. A formal reduction algorithm $\mathrm{red} \in \mathcal{FRA}_{(\nu,\zeta)}$ can be seen as an element of $\mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1, 0, 1\})$, where

$$\mathrm{red}(i) = \begin{cases} (s_i, v_i) & \text{if } i \leq n \\ (0, 0) & \text{if } i > n\ . \end{cases}$$

Note 2. Clearly, every map $f \in \mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1, 0, 1\})$ can be interpreted as a formal reduction algorithm for some (ν, ζ).

Definition 4. *For any* $\mathrm{red} \in \mathcal{FRA}_{(\nu,\zeta)}$ *we define the norm of* red *to be* $\|\mathrm{red}\|$*, interpreted as element of* $\mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1, 0, 1\})$*.*

Example 4. The "simple"method of reduction "subtract ν from ζ for $q := \lfloor \frac{\zeta}{\nu} \rfloor$ times" gives the following formal reduction algorithm $((s_1, v_1), \ldots, (s_n, v_n)) = \mathrm{red}_{\mathrm{simple}}(\nu, \zeta) \in \mathcal{FRA}_{(\nu,\zeta)}$. Let $n := |q|$, $s_1 = s_2 = \cdots = s_n = 0$ and $v_1 = v_2 = \cdots = v_n = -\mathrm{sign}(\zeta)$.

Example 5. The "classical" equation $\zeta = q \cdot \nu + r$ for $q := \lfloor \frac{\zeta}{\nu} \rfloor$ and $0 \leq r < \nu$, or equivalently $\zeta \bmod \nu = \zeta - q \cdot \nu = \zeta - \sum_{i=0}^{n-1} q_i 2^i \cdot \nu$ where $q = (q_{n-1}, \ldots, q_0)_2$ gives the somewhat classical reduction $\text{red}_{\text{class}}(\nu, \zeta) = ((s_1, v_1), \ldots, (s_n, v_n)) \in \mathcal{FRA}_{(\nu,\zeta)}$ via $s_1 = s_2 = \cdots = s_n = 1$ and $v_i = -\text{sign}(\zeta) \cdot q_{n-i}$, for $i = 1, \ldots, n$.

Example 6. We now want to present another important example, namely Sedlak's [Sed] reduction algorithm. Due to its principle it is also known as the so called ZDN-algorithm ($2/3N$ = "**Z**wei **D**rittel **N**" in German). The ZDN-reduction is based on the following easy mathematical observation.

Lemma 1. *Let $\nu \in \mathbb{N}$ and $0 \neq \zeta \in [-\frac{\nu}{3}, \frac{\nu}{3}[$. Then, there exists exactly one pair $(s_\zeta, v_\zeta) \in \mathbb{N}_0 \times \{-1, 1\}$, so that again*

$$\zeta \cdot 2^{s_\zeta} + v_\zeta \cdot \nu \in [-\tfrac{\nu}{3}, \tfrac{\nu}{3}[.$$

Here, s_ζ is uniquely given by $\zeta \cdot 2^{s_\zeta} \in [-\frac{4}{3}\nu, -\frac{2}{3}\nu[\cup [\frac{2}{3}\nu, \frac{4}{3}\nu[$ and $v_\zeta = -\text{sign}(\zeta)$.

The proof is trivial. We additionally set $(s_\zeta, v_\zeta) := (\infty, 0)$ for $\zeta = 0$.

Remark 4. Note that for all $\zeta \in [-\frac{\nu}{3}, \frac{\nu}{3}[$ we have $s_\zeta \geq 2$.

The former lemma immediately leads to the ZDN-reduction due to Sedlak [Sed].

input: ν, ζ with $0 \leq |\zeta| < \frac{\nu}{3} \cdot 2^c$ for some c
output: $\zeta \bmod \nu$
$Z := \zeta \cdot 2^{-c}; i := 1$
while $c > 0$ **do**
  get (s_Z, v_Z) according to the lemma
  if $s_Z > c$ **then**
    $(\widetilde{s}_i, \widetilde{v}_i) := (c, 0)$
  else
    $(\widetilde{s}_i, \widetilde{v}_i) := (s_Z, v_Z)$
  endif
  $Z_i := Z_{i-1} \cdot 2^{\widetilde{s}_i} + \widetilde{v}_i \cdot \nu$
  $c := c - \widetilde{s}_i$
  i ++
endwhile
if $Z < 0$ **then**
  $Z := Z + \nu$
  $(\widetilde{s}_i, \widetilde{v}_i) := (0, 1)$ /* end-reduction */
else
  $(\widetilde{s}_i, \widetilde{v}_i) := (0, 0)$ /* no end-reduction */
endif return Z

This reduction algorithm generates the pairs $(\widetilde{s}_1, \widetilde{v}_1), \ldots (\widetilde{s}_n, \widetilde{v}_n) \in \mathbb{N}_0 \times \{-1, 0, 1\}$ for some n. Obviously, they have the following properties:

1. $\widetilde{s}_1 \geq 0$, $\widetilde{s}_2, \ldots, \widetilde{s}_{n-2} \geq 2$, $\widetilde{s}_{n-1} \geq 1$ and $\widetilde{s}_n = 0$.
2. $\widetilde{v}_1, \ldots, \widetilde{v}_{n-2} \neq 0$, $\widetilde{v}_{n-1} \in \{-1, 0, 1\}$ and $\widetilde{v}_n \in \{0, 1\}$.

3. for $i = 1, \dots, n-1$:
   $\widetilde{v}_i \neq 0 \quad \Rightarrow \quad Z_i \in [-\frac{\nu}{3}, \frac{\nu}{3}[$,
   $\widetilde{v}_i = 0 \quad \Rightarrow \quad Z_i \in [-\frac{2}{3}\nu, \frac{2}{3}\nu[$, which can happen only for $i = n-1$.

Now, the formal reduction algorithm $\text{red}_{\text{ZDN}}(\nu, \zeta) = ((s_1, v_1), \dots, (s_n, v_n))$ is defined by

$$\begin{aligned} (s_1, v_1) &:= (0, \widetilde{v}_1) \\ (s_i, v_i) &:= (\widetilde{s}_i, \widetilde{v}_i), \quad \text{for } i = 2, \dots, n. \end{aligned}$$

Remark 5. Note that s_1 never has any effect on the reduction algorithm.

Remark 6. Assume $v_1 \neq 0$, which always holds if $n > 2$. Then, for $c := s_1 + \cdots + s_n$, we automatically get $\zeta \cdot 2^{-c} \in [-\frac{4}{3}\nu, -\frac{2}{3}\nu[\cup [\frac{2}{3}\nu, \frac{4}{3}\nu[$, by virtue of the property 3 from above.

The ZDN-reduction can be succinctly characterized by the following lemma.

Lemma 2. *Let $\zeta \neq 0$ and* red $= ((s_1, v_1), \dots, (s_n, v_n)) \in \mathcal{FRA}_{(\nu,\zeta)}$*, so that the following properties are fulfilled.*

1. *$s_1 = 0$, $s_2, \dots, s_{n-2} \neq 0$, $s_{n-1} \neq 0$ and $s_n = 0$.*
2. *$v_1, \dots, v_{n-2} \neq 0$, $v_{n-1} \in \{-1, 0, 1\}$ and $v_n \in \{0, 1\}$.*
3. *For $i = 1, \dots, n-1$: $v_i \neq 0$ implies $Z_i \in [-\frac{\nu}{3}, \frac{\nu}{3}[$, and for $i = n-1$: $v_i = 0$ implies $Z_i \in [-\frac{2}{3}\nu, \frac{2}{3}\nu[$.*

Then red $= \text{red}_{\text{ZDN}}(\nu, \zeta)$.

Proof. We assume $n \geq 3$. Since $\text{red}_{\text{ZDN}}(\nu, \zeta)$ has the properties 1 to 3 we only have to prove the uniqueness of such a reduction. By the uniqueness properties of Lemma 1, the values $(s_2, v_2), \dots, (s_{n-2}, v_{n-2})$ are unique. Clearly, s_1 is unique. Obviously $v_1 = -\text{sign}(\zeta)$. Furthermore, $c = s_1 + \cdots + s_n$ is unique since $\zeta \cdot 2^{-c} \in [-\frac{4}{3}\nu, -\frac{2}{3}\nu[\cup [\frac{2}{3}\nu, \frac{4}{3}\nu[$ has to be fulfilled. Therefore, s_{n-1} is given by $c - s_1 - \cdots - s_{n-2}$ and $s_n = 0$ anyway. Then, the only remaining ambivalence could be a) $(v_{n-1}, v_n) = (-1, 1)$ versus $= (0, 0)$ and b) $(v_{n-1}, v_n) = (1, 0)$ versus $= (0, 1)$. Since $Z_{n-2} \cdot 2^{s_{n-1}}$ is unique already, we have the cases:

1. If $Z_{n-2} \cdot 2^{s_{n-1}} \in [-\frac{\nu}{3}, \frac{\nu}{3}[$ then $v_{n-1} = 0$; otherwise $Z_{n-1} \notin [-\frac{2}{3}\nu, \frac{2}{3}\nu[$.
2. If $Z_{n-2} \cdot 2^{s_{n-1}} \in [-\frac{2}{3}\nu, \frac{2}{3}\nu[\setminus [-\frac{\nu}{3}, \frac{\nu}{3}[$ then $v_{n-1} = 0$; otherwise by (3) it has to be $Z_{n-1} \in [-\frac{\nu}{3}, \frac{\nu}{3}[$ which is not possible.
3. If $Z_{n-2} \cdot 2^{s_{n-1}} \in [-\frac{4}{3}\nu, \frac{4}{3}\nu[\setminus [-\frac{2}{3}\nu, \frac{2}{3}\nu[$ then $v_{n-1} \neq 0$, i.e., $v_{n-1} = -\text{sign}(Z_{n-2})$; otherwise $Z_{n-1} \notin [-\frac{2}{3}\nu, \frac{2}{3}\nu[$.
4. $Z_{n-2} \cdot 2^{s_{n-1}} \notin [-\frac{4}{3}\nu, \frac{4}{3}\nu[$ is not possible.

Thus, we are done. □

The next proposition shows an easy and straightforward connection between $\mathcal{FRA}$ and $\mathcal{FMA}$.

Proposition 1. *Let* $\mathrm{red} = ((s_1, v_1), \ldots, (s_n, v_n)) \in \mathcal{FRA}_{(\nu,\zeta)}$*, then the assignments*

1. $Z_0 := 0$
2. $Z_{i+1} := Z_i \cdot 2^{s_{i+1}} + v_{i+1} \cdot \nu$*, for* $i = 0, \ldots, n-1$*,*

result in $Z_n = -\nu \cdot q (= (\zeta \bmod \nu) - \zeta)$ *where* $q = \lfloor \frac{\zeta}{\nu} \rfloor$*.*
In other words: $((s_1, v_1), \ldots, (s_n, v_n))$ *can be seen as an element of* $\mathcal{FMA}_{(\nu,-q)}$*.*

Proof. Plugging in the equations consecutively for $c = s_1 + \cdots + s_n$ yields

$$\begin{aligned} Z_0 &:= Z \cdot 2^{-c} \\ Z_1 &:= Z_0 \cdot 2^{s_1} + v_1 \cdot \nu \\ &\vdots \\ Z_n &:= Z_{n-1} \cdot 2^{s_n} + v_n \cdot \nu, \text{ for } i = 0, \ldots, n-1, \end{aligned}$$

meaning that $Z_n = x \cdot Z + y$ for some $x, y \in \mathbb{Z}$. Obviously, by definition of c we have $x = 1$. If $Z := \zeta$ then by definition of the formal reduction algorithm for (ν, ζ) we have $1 \cdot \zeta + y = \zeta \bmod \nu$, hence $y = (\zeta \bmod \nu) - \zeta$. For $Z := 0$ we get the desired result. □

Finally, for this section we want to state a simple technical lemma about the step-function and formal reduction algorithms.

Lemma 3. *Let* $\bar{q} = (\bar{q}_m, \ldots, \bar{q}_0) \in \mathcal{SD}_2$ *be an* $\mathcal{SD}_2$ *representation of* $q = (\bar{q}_m, \ldots, \bar{q}_0)_{\mathrm{SD}_2} \in \mathbb{Z}$ *and let* $((s_1, v_1), \ldots, (s_n, v_n)) := \mathrm{step}(\bar{q})$*. By definition of* step *it holds that* $v_1 \neq 0$ *and* $s_n \neq 0$*. For* $m = s_1 + \cdots + s_n$ *set*

1. $Z_0 := Z \cdot s^{-m}$*, and*
2. $Z_{i+1} := Z_i \cdot 2^{s_{i+1}} + v_{i+1} \cdot \nu$*, for* $i = 0, \ldots, n-1$*.*

Then, for $i = 1, \ldots, n$ *it holds that* $Z_i = Z \cdot 2^{-j} + (\bar{q}_m, \ldots, \bar{q}_j)_{\mathrm{SD}_2} \cdot \nu$ *where* $j = m - s_1 - \cdots - s_i = s_{i+1} + \cdots + s_n$*.*

Proof. The obvious proof is done by induction on i.

5 The connecting morphism

The above proposition gives a morphism $\mathcal{FRA}_{(\nu,\zeta)} \longrightarrow \mathcal{FMA}_{(\nu,-\lfloor \zeta/\nu \rfloor)}$. This is indeed even a bijection! It shows that multiplication algorithms and reduction algorithms are related in some way. Mathematically, this is clear by the formula $\zeta = q \cdot \nu + r$ with $q := \lfloor \frac{\zeta}{\nu} \rfloor$ and $r := \zeta \bmod \nu$. For computing the residue r we may compute the product $q \cdot \nu$ and then subtract it from ζ.

Theorem 1. *For any* $(\nu, \zeta) \in \mathcal{N} \times \mathcal{Z}$ *there is a bijective morphism*

$$\begin{aligned} \widehat{\ } : \mathcal{FMA}_{(\nu,\lfloor \frac{\zeta}{\nu} \rfloor)} &\longrightarrow \mathcal{FRA}_{(\nu,\zeta)} \\ ((s_1, v_1), \ldots, (s_n, v_n)) &\longmapsto ((s_1, -v_1), \ldots, (s_n, -v_n)). \end{aligned}$$

The straightforward proof is omitted.

Remark 7. If $inv\colon \{-1,0,1\} \to \{-1,0,1\}$ is given by $inv(x) = -x$, then the above morphism transforms into

$$\begin{aligned} \mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1,0,1\}) &\longrightarrow \mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1,0,1\}) \\ f &\longmapsto (\mathrm{id}_{\mathbb{N}_0}, inv) \circ f \end{aligned}$$

Remark 8. $\widehat{\ }$ respects the norms $\|\cdot\|$ on $\mathcal{FRA}$ and $\mathcal{FMA}$.

6 A certain construction for formal algorithms

Definition 5. *We set* $\mathcal{SD}_2^+ := \mathcal{SD}_2 \times \{-1,0,1\}$.

Remark 9. Naturally $\mathcal{SD}_2 \subset \mathcal{SD}_2^+$ via $\bar{\beta} \mapsto (\bar{\beta}, 0)$.

There is an evaluation function

$$\begin{aligned} ev\colon \mathcal{SD}_2^+ &\longrightarrow \mathbb{Z} \\ (\bar{\beta}, x) = ((\bar{\beta}_n, \dots, \bar{\beta}_0), x) &\longmapsto ev(\bar{\beta}, x) = (\bar{\beta}_n, \dots, \bar{\beta}_0)_{\mathrm{SD}_2} + x, \end{aligned}$$

which is surjective and certainly not injective. There exist several sections of ev.

Example 7. The binary representation $\mathrm{bin}\colon \mathbb{Z} \to \mathcal{SD}_2 \subset \mathcal{SD}_2^+$, which assigns to every positive β its binary representation $(\beta_{n-1}, \dots, \beta_0) \in \{0,1\}^n$, where $\beta = (\beta_{n-1}, \dots, \beta_0)_2$. For negative integers we define $\mathrm{bin}(-\beta) := (-\beta_{n-1}, \dots, -\beta_0)$.

Example 8. Look at Reitwiesner's [Rei] signed digit representation of an integer β. An element $(\bar{\beta}_n, \dots, \bar{\beta}_0) \in \mathcal{SD}_2$ with $\bar{\beta} = (\bar{\beta}_n, \dots, \bar{\beta}_0)_{\mathrm{SD}_2}$ is unique with respect to the following properties:

1. It is non adjacent, i.e., $\bar{\beta}_i \cdot \bar{\beta}_{i+1} = 0$, for all i.
2. The Hamming weight of $(\bar{\beta}_n, \dots, \bar{\beta}_0)$ is minimal in the set $ev^{-1}(\beta)$.

This representation is denoted by $\mathrm{RW}\colon \mathbb{Z} \longrightarrow \mathcal{SD}_2 \subset \mathcal{SD}_2^+$.

Now, we want to assign to every section $\mathrm{sec}\colon \mathbb{Z} \longrightarrow \mathcal{SD}_2^+$ of ev a formal multiplication algorithm. For this, we first have to continue the map $\mathrm{step}\colon \mathcal{SD}_2 \longrightarrow \mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1,0,1\})$ to $\mathrm{step}\colon \mathcal{SD}_2^+ \longrightarrow \mathrm{Map}'(\mathbb{N},\ \mathbb{N}_0 \times \{-1,0,1\})$. This is done trivially. If $(\bar{\beta}, x) \in \mathcal{SD}_2^+$ and $\mathrm{step}(\bar{\beta}) = ((s_1, v_1), \dots, (s_n, v_n))$, then $\mathrm{step}(\bar{\beta}, x) := ((s_1, v_1), \dots, (s_n, v_n), (0, x))$.

Definition 6. *For a section* $\mathrm{sec}\colon \mathbb{Z} \longrightarrow \mathcal{SD}_2^+$ *the formal multiplication algorithm* $\mathrm{mul}_{\mathrm{sec}}(\alpha, \beta) \in \mathcal{FMA}_{(\alpha,\beta)}$ *is defined as*

$$\mathrm{mul}_{\mathrm{sec}}(\alpha, \beta) = \mathrm{step}(\mathrm{sec}(\beta)).$$

The formal reduction algorithm $\mathrm{red}_{\mathrm{sec}}(\nu, \zeta) \in \mathcal{FMA}_{(\nu,\zeta)}$ *associated to* sec *is given by*

$$\mathrm{red}_{\mathrm{sec}}(\nu, \zeta) = \widehat{\Big(\mathrm{step}\Big(\,\mathrm{sec}\,\Big(\Big\lfloor \frac{\zeta}{\nu} \Big\rfloor\Big)\Big)\Big)},$$

i.e., $\mathrm{red}_{\mathrm{sec}}(\nu, \zeta) = \mathrm{mul}_{\mathrm{sec}}(\nu, \lfloor \frac{\zeta}{\nu} \rfloor)\widehat{\ } \left(=: \widehat{\mathrm{mul}}_{\mathrm{sec}}(\nu, \lfloor \frac{\zeta}{\nu} \rfloor)\right)$.

Example 9. Obviously we have $\mathrm{mul}_{\mathrm{bin}}(\alpha,\beta) = \mathrm{mul}_{\mathrm{class}}(\alpha,\beta)$ and $\mathrm{red}_{\mathrm{bin}}(\nu,\zeta) = \mathrm{red}_{\mathrm{class}}(\nu,\zeta)$.

Example 10. The simple algorithms are not representable in this form, because here we have $s_i = 0$ for all i.

Example 11. If we take Reitwiesner's algorithm for the Booth construction from example 3 then

$$\mathrm{mul}_{\mathrm{Booth}}(\alpha,\beta) = \mathrm{mul}_{\mathrm{RW}}(\alpha,\beta).$$

7 The Duality

Roughly speaking, the Duality Theorem states that the ZDN-reduction is the dual to the Booth-multiplication (in Reitwiesner's form), i.e., "$\mathrm{red}_{\mathrm{ZDN}}(\nu,\zeta) = \widehat{\mathrm{mul}}_{\mathrm{Booth}}(\nu,\lfloor\zeta/\nu\rfloor)$". This is seen to be true "cum grano salis". Namely, for technical reasons we have to adapt the Booth-multiplication at the final loop rounds. For the pair (ν,ζ) we are constructing a derivation $\mathrm{mul}_{\widetilde{\mathrm{Booth}}}(\nu,\zeta/\nu)$ of the Booth multiplication $\mathrm{mul}_{\mathrm{Booth}}(\nu,\lfloor\zeta/\nu\rfloor)$. Write $\zeta = q\cdot\nu + r$ in the following form:

- If $\zeta \bmod \nu \in [0,\frac{1}{3}\nu[$, then set $q := \lfloor\frac{\zeta}{\nu}\rfloor$, $r := \zeta \bmod \nu$ and $x = 0$.
- If $\zeta \bmod \nu \in [\frac{1}{3}\nu,\frac{2}{3}\nu[$ and $\lfloor\frac{\zeta}{\nu}\rfloor$ is even, then set $q := \lfloor\frac{\zeta}{\nu}\rfloor$, $r := \zeta \bmod \nu$ and $x = 0$.
- If $\zeta \bmod \nu \in [\frac{1}{3}\nu,\frac{2}{3}\nu[$ and $\lfloor\frac{\zeta}{\nu}\rfloor$ is odd, then set $q := \lfloor\frac{\zeta}{\nu}\rfloor+1$, $r := (\zeta \bmod \nu)-\nu$ and $x = -1$.
- If $\zeta \bmod \nu \in [\frac{2}{3}\nu,\nu[$, then set $q := \lfloor\frac{\zeta}{\nu}\rfloor+1$, $r := (\zeta \bmod \nu)-\nu$ and $x = -1$.

Now, either $r \in [-\frac{1}{3}\nu,\frac{1}{3}\nu[$ or $r \in [-\frac{2}{3}\nu,\frac{2}{3}\nu[\setminus[-\frac{1}{3}\nu,\frac{1}{3}\nu[$ but q is even. Thus, we define

$$\mathrm{mul}_{\widetilde{\mathrm{Booth}}}\left(\nu,\frac{\zeta}{\nu}\right) := \mathrm{step}(\mathrm{RW}(q),x).$$

Note that $(\mathrm{RW}(q),x)$ is also a section of *ev*.

Theorem 2. *For any $(\nu,\zeta) \in \mathcal{N}\times\mathcal{Z}$ we have the equality of the two formal reduction algorithms for (ν,ζ), i.e.,*

$$\mathrm{red}_{\mathrm{ZDN}}(\nu,\zeta) = \widehat{\mathrm{mul}}_{\widetilde{\mathrm{Booth}}}\left(\nu,\frac{\zeta}{\nu}\right).$$

Proof. Let $((s_1,v_1),\dots,(s_n,v_n)) := \mathrm{mul}_{\widetilde{\mathrm{Booth}}}(\nu,\frac{\zeta}{\nu})$, then we have to show

$$\mathrm{red}_{\mathrm{ZDN}}(\nu,\zeta) = ((s_1,-v_1),\dots,(s_n,-v_n)).$$

We assume again $n \geq 3$. In order to prove this equality, we may apply lemma 2 and verify the described points 1 to 3. The points 1 and 2 are clear by the construction of step. Note that $((s_1,v_1),\dots,(s_{n-1},v_{n-1})) = \mathrm{step}(q)$ and $(s_n,-v_n) = (0,0)$ or $(0,1)$. So, we have to verify the third point of lemma 2:

a) $Z_1,\dots,Z_{n-2} \in [-\frac{1}{3}\nu,\frac{1}{3}\nu[$,

b) $Z_{n-1} \in [-\frac{1}{3}\nu, \frac{1}{3}\nu[$ if $v_{n-1} \neq 0$, otherwise $Z_{n-1} \in [-\frac{2}{3}\nu, \frac{2}{3}\nu[$.

With $(\bar{q}_m, \ldots, \bar{q}_0) = \mathrm{RW}(q)$ we have by virtue of lemma 3 for $i = 0, \ldots, n-1$ that

$$Z_i = \zeta \cdot 2^{-j} - (\bar{q}_m, \ldots, \bar{q}_j)_{\mathrm{SD}_2} \cdot \nu,$$

for some j, where $\bar{q}_j \neq 0$. Since $\zeta = (\bar{q}_m, \ldots, \bar{q}_0)_{\mathrm{SD}_2} \cdot \nu + r$ it holds that

$$Z_i \cdot 2^j = (\bar{q}_{j-1}, \ldots, \bar{q}_0)_{\mathrm{SD}_2} \cdot \nu + r.$$

For a) we have to prove

$$-\frac{1}{3}\nu \cdot 2^j \leq r + (\bar{q}_{j-1}, \ldots, \bar{q}_0)_{\mathrm{SD}_2} \cdot \nu < \frac{1}{3}\nu \cdot 2^j$$

or better

$$-2^j \leq 3\Big(\frac{r}{\nu}\Big) + 3 \cdot (\bar{q}_{j-1}, \ldots, \bar{q}_0)_{\mathrm{SD}_2} < 2^j.$$

Since $3(\frac{r}{\nu}) \in [-1, 1[$ or $3(\frac{r}{\nu}) \in [-2, 2[$ with q even, i.e., $\bar{q}_0 = 0$, property a) is implied by the next lemma 4. For b) it follows immediately, since $\bar{q}_0 = v_{n-1}$. If $\bar{q}_0 = 0$ then $3(\frac{r}{\nu}) \in [-2, 2[$, i.e.,

$$-2 \cdot 2^0 \leq 3\Big(\frac{r}{\nu}\Big) + 3 \cdot 0 < 2 \cdot 2^0$$

and if $\bar{q}_0 \neq 0$, then by definition $3(\frac{r}{\nu}) \in [-1, 1[$ and

$$-2^0 \leq 3\Big(\frac{r}{\nu}\Big) + 3 \cdot 0 < 2^0,$$

which concludes the proof □

Lemma 4. *Let $(\bar{q}_n, \ldots, \bar{q}_0) \in \mathcal{SD}_2$ be nonadjacent and $j \in 1, \ldots, n$ so that $\bar{q}_j \neq 0$. Then,*

$$-|\bar{q}_0| - 2^j + 2 \leq 3 \cdot (\bar{q}_{j-1}, \ldots, \bar{q}_0)_{\mathrm{SD}_2} \leq 2^j - 2 + |\bar{q}_0|.$$

Proof. The largest value for $(\bar{q}_{j-1}, \ldots, \bar{q}_0)_{\mathrm{SD}_2}$ if $\bar{q}_{j-1} = 0$, since $\bar{q}_j \neq 0$, is $(0101\ldots b)_{\mathrm{SD2}}$ of the same length, where $b = |\bar{q}_0|$. Therefore, we have

$$3 \cdot (\bar{q}_{j-1}, \ldots, \bar{q}_0)_{\mathrm{SD}_2} \leq (111 \cdots 1b)_{\mathrm{SD}_2} = (2^j - 1) - (1 - b).$$

The left part is shown analogously. □

8 Conclusion

From this paper we have learned two things. First, we have seen that every reduction algorithm corresponds to a multiplication algorithm with the same norm, i.e., performance. Clearly, the other direction holds as well. For multiplication algorithms one knows that Booth's algorithm is the best in performance. Therefore, the best reduction algorithm is the one which is dual to the Booth multiplication. On the other hand, we have presented the so called ZDN-reduction by a proper mathematical derivation solving the lack of its intuitive/mathematical understanding. It relies on a simple mathematical observation with respect to reduction with – at the first moment – strange bounds. However, we proved that this reduction is the realization of Booth's dual. Hence, it also has an optimal performance.

References

[AW] S. Arno, F. S. Wheeler, "Signed digit representation of minimum Hamming weight", *IEEE Trans. on Computers*, **42**:1007-1010, 1993.

[Bar] P. Barret, "Implementing the Rivest, Shamir and Adleman public-key encryption algorithm on a standard digital signal processor", *Proc. of CRYPTO '86*, Springer LNCS, vol. 263, pp. 311-323, 1987.

[Bri] E. F. Brickell, "A fast modular multiplication algorithm with application to two key cryptography', *Proc. of CRYPTO '86* Springer LNCS vol. 263, pp. 311-323, 1987.

[Boo] A. D. Booth, "A signed binary multiplication technique", *Q. J. Mech. Appl. Math*, **4**(2):236-240, 1951.

[CL] W. E. Clark, J. J. Liang, "On arithmetic weight for a general radix representation of integers", *IEEE Trans. on Inform. Theory*, **19**:823-826, 1973.

[DJQ] J.-F. Dhem, M. Joye, J.-J. Quisquater, "Normalisation in diminished-radix modulus transformation", *Electronics Letters*, **33**(23):1931, 1997.

[DQ] J.-F. Dhem, J.-J. Quisquater, "Recent results on modular multiplication for smart cards", *Proc. of CARDIS '98* Springer LNCS vol. 1820, pp. 336-352, 1998.

[HP] H. Handschuh, P. Pailler, "Smart Card Crypto-Coprocessors for Public-Key Cryptography", *Proc. of CARDIS '98* Springer LNCS vol. 1820, pp. 372-379, 1998.

[Hwa] K. Hwang, *Computer Arithmetic, Principles, Architecture and Design*, John Wiley & Sons, New York, 1979.

[JY] M. Joye, S.-M. Yen, "Optimal left-to-right binary signed-digit exponent recoding", *IEEE Transactions on Computers*, **49**(7):740-748, 2000.

[Kor] I. Koren, *Computer Arithmetic Algorithms*, Brookside Court Publishers, Amherst MA, 1998.

[Knu] D. E. Knuth, *The Art of Computer Programming, Vol.2: Seminumerical Algorithms*, 3rd ed., Addison-Wesley, Reading MA, 1999.

[KP99] C. Koc, C. Paar, *Cryptographic Hardware and Embedded Systems*, First International Workshop, CHES'99 Worcester, MA, USA, August 12-13, 1999 Proceedings, Springer LNCS, vol. 1717, 1999.

[KP00] C. Koc, C. Paar, *Cryptographic Hardware and Embedded Systems - CHES 2000*, Second International Workshop Worcester, MA, USA, August 17-18, 2000 Proceedings, Springer LNCS, vol. 1965, 2000.

[KNP01] C. Koc, D. Naccache, C. Paar, *Cryptographic Hardware and Embedded Systems - CHES 2001*, Third International Workshop, Paris, France, May 14-16, 2001 Proceedings, Springer LNCS, vol. 2162, 2001.

[KKP02] B. Kaliski, C. Koc, C. Paar, *Cryptographic Hardware and Embedded Systems - CHES 2002*, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002 Revised Papers, Springer LNCS, vol. 2523, 2002.

[Mac] O. L. MacSorley, "High-speed arithmetic in binary computers", *Proc. IRE*, **49**:67-91, 1961.

[MvOV] A. J. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, New York, 1997.

[Mon] P. L. Montgomery, "Modular multiplication without trial division", *Math. of Computation*, **44**:519-521, 1985.

[NMR] D. Naccache, D. M'Raihi, "Arithmetic co-processors for public-key cryptography: The state of the art", *IEEE Micro*, pp. 14-24, 1996.

[Omu] J. Omura, "A public key cell design for smart card chips", *Proc. of International Symposium on Information Theory and its Applications*, pp. 983-985, 1990.

[Par] B. Parhami, *Computer Arithmetic*, Oxford University Press, New York, 2000.

[Rei] G. W. Reitwiesner, "Binary Arithmetic", *Advances in Computers*, **1**:231-308, Academic Press, 1960.

[RSA] R. Rivest, A. Shamir, L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Comm. of the ACM*, **21**:120-126, 1978.

[Sed] H. Sedlak, "The RSA cryptographic Processor: The first High Speed One-Chip Solution", *Proc. of EUROCRYPT '87*, Springer LNCS, vol. 293, pp. 95-105, 198.

[Spa] O. Spaniol, *Arithmetik in Rechenanlagen*, B. G. Teubner, Stuttgart, 1976.

[WQ] D. de Waleffe, J.-J. Quisquater, "CORSAIR, a smart card for public-key cryptosystems", *Proc. of CRYPTO '90*, Springer LNCS, vol. 537, pp. 503-513, 1990.

[WH] H. Wu, M. A. Hasan, "Closed-form expressions for the average weight of signed-digit representations", *IEEE Trans. on Computers*, **46**(2):162-172, 1997.