Indian Journal of Science and Technology, Vol 10(39), DOI: 10.17485/ijst/2017/v10i39/119863, October 2017
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

An Improved Signed Digit Representation of Integers

Arash Eghdamian and Azman Samsudin

School of Computer Sciences, Universiti Sains Malaysia, 11800, Penang, Malaysia; ae11_com109@student.usm.my, azman@cs.usm.my

Abstract

This research modifies the GNAF (Generalized NAF), which is one of the signed-digit representations that have been used to speed up point multiplication in pairing-based cryptosystems. The hamming weight of MGNAF (Modified Generalized Non-Adjacent Form), which is proposed in this paper, is less than GNAF for radixes higher than 2 (in cost of a bigger digit set). Moreover, in radix 2 the hamming weight of MGNAF is as low as GNAF and the well-known NAF (Non-Adjacent Form) with the same digit set.

Keywords: Cryptography, Digit Set, Generalized NAF, Hamming Weight, Non-Adjacent Form, Radix-*r* Representation

1. Introduction

Modular exponentiation is one of the most expensive (time-consuming) operations in most of cryptosystems. For that reason, efficient algorithms to perform this operation are important in the performance of the resulting cryptographic protocols. Generally there are two types of exponentiations for computing g^m.

In the first type of exponentiation, the base g is fixed and just the exponent m varies, as it is in the ElGamal cryptosystem[1,2]. In that case, good performances are obtained by the basic square-and-multiply technique.

The other type involves exponentiations with a fixed exponent *m*, such as in the RSA cryptosystem[3,4]. The goal is then to quickly compute g^m for randomly chosen *g*.

In this paper, we are mainly concerned with the first type of exponentiation. Another use of this type is when inverses can be virtually computed for free, like in elliptic curves[5]. The basic idea is to improve the efficiency of multiplication in this technique.

In order to achieve faster scalar multiplication, we have to exploit an efficient class of the radix-*r* representation, for example, by reducing the number of non-zero digits. The Generalized Non-Adjacent Form (GNAF) is known as an efficient class of radix-*r* representation[6,7]. The average density of non-zero digits (non-zero density) of the GNAF is asymptotically *(r - 1) / (r +1)* with *(r - 1)* pre-computed points. For example, $r = 3$ attains 0.5 non-zero density with 2 pre-computed point. On the other hand, the non-zero density of the standard radix-*r* representation is *(r–1) / r* with the same pre-computed point (0.67 for $r = 3$). Therefore, the GNAF is able to improve the efficiency of computing the paring based cryptosystem, especially scalar multiplication. In addition, another way that may help to achieve lower non-zero density could be choosing a larger digit set.

2. Generalized Non-Adjacent Form (GNAF)

In this section some properties related to the GNAF representation would be discussed.

An integer *M* is represented using the radix-*r* representation, namely:

$$M = \sum_{j=0}^{n-1} M_j r^j, \quad M_j \in \{0,1,\dots,r-1\} \qquad (1)$$

This research denotes the radix-*r* representation of *M* by $M = (M_{n-1}, \dots, M_1, M_0)$ and in that M_j is called the *j*-th digit and *n* is the digit length of the radix-*r* representation for *M*. Hamming Weight of the radix-*r* representation of *M* is the number of non-zero digits of it. The average hamming weight of the radix-*r* representation is $(r - 1) / r$.

*Author for correspondence

It is called signed radix-*r* representation, if digit M_j be allowed to take a negative value (for example: $M_j \in \{0, \pm 1, ..., \pm(r-1)\}$). In a binary signed-digit representation if no two adjacent digits are nonzero, it will be called canonical or for that reason, it is also called the nonadjacent form (NAF) of a number by some authors[8].

Reitwiesner studied the canonical recoding[9]. He proved that this representation (NAF) is unique. But in general, the signed radix-*r* representation is not unique and an integer can have several signed representations. Ebeid and Hasan[10] by using the facts that different representations of *M* can be obtained by replacing 01 with $1\bar{1}$, $0\bar{1}$ with $\bar{1}1$ and vice versa[9], proposed an algorithm to generate all possible Binary Signed Digit representations of *M* in 2007. For instance, Figure 1 shows all binary signed representations of 13 with length of 5 bits.

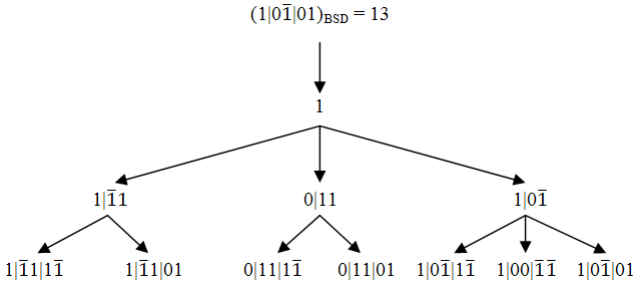

Figure 1. All 5-bit signed digit representations of 13[11].

The general case was later addressed by Clark and Liang[6]. They presented a minimal representation for any signed-radix r. In that case, Amo and Wheeler[12] proved that the average proportion of nonzero digits is equal to (r - 1)/(r + 1). This should be compared with the average of the number of non-zero digits in the standard radix-r representation which is (r - 1)/r.

The generalized non-adjacent form (GNAF) can represent each integer uniquely. GNAF is the signed radix-*r* representation which satisfied the following two conditions.

$$1\text{-}\ |M_i + M_{i+1}| < r \text{ for all } i,$$

$$2\text{-}\ |M_i| < |M_{i+1}| \text{ if } M_i M_{i+1} < 0.$$

In case of $r = 2$, the definition is equal to the classical NAF for binary representation. GNAF has the minimal Hamming weight among all signed radix-*r* representation with digit set $\{0, \pm 1, ..., \pm(r-1)\}$[6].

GNAF, for radix-*r* representation of integer *M*, is generated by computing (r + 1)M – M, and in that the minus "–" is a digit-wise subtraction of (r + 1)M by M. This construction is a generalization of Reitwiesner algorithm for generating nonadjacent form (NAF)[13]. Because there is a carry for computing the radix-*r* representation of (r + 1)M, this algorithm is not computed in the left-to-right approach. Joye and Yen in[7] proposed a left-to-right based algorithm for generating a signed radix-*r* representation which has same hamming weight and digit set as GNAF.

3. Proposed Work

The modified version of GNAF in this research is called MGNAF. The difference between GNAF and MGNAF lies on how they treat some consecutive same nonzero elements, which may appear in a big number. Although GNAF is only able to reduce the Hamming Weight of consecutive *r-1*s in a number, MGNAF is able to decrease the HW of any sequence of same nonzero elements.

In GNAF recoding:
When a sequence of n *r-1*s appears in a number of radix *r*, like:

$$\underbrace{r-1, r-1, \ldots, r-1}_{n \text{ times}} \tag{2}$$

It will be changed to:

$$\underbrace{1, 0, 0, \ldots, 0}_{n-1 \text{ times}}, \bar{1} \tag{3}$$

The *1* at the beginning would be added to the next digit to the left of the sequence and $\bar{1}$ represents *-1* in this number. The other sequences of any other nonzero element would remain untouched. For example a sequence of five 3s in a number of radix 4 would be changed to $(10000\bar{1})_4$ but a sequence of five 2s in a number of radix 4 would remain untouched. The reason is if 1 be added to $(33333)_4$ the result would be $(100000)_4$, now that added 1 should be reduced. So the result will become $(10000\bar{1})_4$ which is the same number as $(33333)_4$ but with different formation and less Hamming Weight.

On the other hand, MGNAF would reduce the hamming weight of any sequence which is composed of one nonzero element.

It will change

$$(\underbrace{x, x, \ldots, x}_{n}) \tag{4}$$

to

$$(\frac{x}{r-1}, \underbrace{0,0,\ldots,0}_{n-1\ times}, \frac{\bar{x}}{r-1}) \quad (5)$$

In which the $\frac{x}{r-1}$ would be added to the next digit to the left of the sequence. The mathematical explanation for this recoding, for an example of five *x*-es in radix *r*, is as below ($x \neq 0$ and $r \neq 1$):

$$(x,x,x,x,x)r = \frac{x}{r-1}(r\text{-}1,r\text{-}1,r\text{-}1,r\text{-}1,r\text{-}1)r \quad (6)$$

$$= \frac{x}{r-1}(1,0,0,0,0,\bar{1})r$$

$$= (\frac{x}{r-1}, 0,0,0,0, \frac{\bar{x}}{r-1})r$$

The following three-step algorithm applies on a standard radix-*r* integer and recodes it to a MGNAF representation. This algorithm uses the right-to-left approach so finding the left-to-right approach for this representation can be a topic for further studies.

Algorithm: Given integer $M = \sum_{i=0}^{n} a_i r^i$, $|a_i| < r$, $i = 0,1,\ldots,n$, this algorithm computes MGNAF for *M*.

***Step 1*:** Set $a_{n+1} = 0$.

***Step 2*:** Do *step 3* for $i = 0,1,\ldots,n$; then the algorithm terminates with $M = \sum_{i=0}^{n+1} a_i r^i$ as the MGNAF for *M*.

***Step 3*:** If $a_i \neq 0$, then consider the following cases:

1) $a_i = a_{i+1}$. Set $a_i = \frac{-a_i}{Radix-1}$ and set $Digit = a_{i+1}$. While $a_{i+1} = Digit$, set $a_{i+1} = 0$ and increase *i* by 1. After while loop set $a_{i+1} = a_{i+1} + \frac{a_i}{Radix-1}$.

2) $a_i > 0$ and $a_{i+1} \geq 0$. If $a_i + a_{i+1} \geq r$, then set $a_i = a_i - r$ and set $a_{i+1} = a_{i+1} + 1$, if $a_{i+1} < r - 1$, $a_{i+1} = 0$, and $a_{i+2} = a_{i+2} + 1$, if $a_{i+1} = r - 1$.

3) $a_i < 0$ and $a_{i+1} \leq 0$. If $a_i + a_{i+1} \leq -r$, then set $a_i = +r$ and set $a_{i+1} = a_{i+1} - 1$, if $a_{i+1} > -(r-1)$, $a_{i+1} = 0$, and $a_{i+2} = a_{i+2} - 1$, if $a_{i+1} = -(r-1)$.

4) $a_i > 0$ and $a_{i+1} < 0$. If $a_i \geq -a_{i+1}$, then set $a_i = a_i - r$ and set $a_{i+1} = a_{i+1} + 1$.

5) $a_i < 0$ and $a_{i+1} > 0$. If $-a_i \geq a_{i+1}$, then set $a_i = a_i + r$ and set $a_{i+1} = a_{i+1} - 1$.

4. Comparison

The GNAF representation, in order to improve the performance of computing scalar multiplication, reduces the arithmetic weight. The GNAF achieves $\frac{r-1}{r+1}$ non-zero density recoding while this number for MGNAF is lower, but the size of the digit set for MGNAF is larger.

Numerical details of these comparisons are provided in Table 1. The result is that the proposed MGNAF has better performance in implementing scalar multiplication with the cost of additional digits in its digit set. For practical applications, a noticeable computational speedup with only three additional pre-computed value is possible by selecting $r = 3$. In this case, a 0.4 non-zero density recoding is achieved for MGNAF. By using GNAF, a 0.5 non-zero density recoding is obtained.

Table 1. Comparison of GNAF and the proposed MGNAF

Radix	Normal	GNAF HW[1]	MGNAF HW[1]	Improvement[2] (%)
2	50	33.33	33.33	0
3	66.67	50	44.87	30.78
4	75	60	53.25	45
5	80	66.67	59.55	53.37
6	83.33	71.43	64.7	56.52
7	85.71	75	68.5	60.67
8	87.5	77.78	71.7	62.51
9	88.89	80	74.15	65.81

[1]Hamming Weight; [2]MGNAF improvement compare to GNAF

According to the figures in Table 1, for radix 2 the hamming weight of GNAF and MGNAF are both 33.33% (Same as NAF) where this for normal form is 50%. But with increasing the radix, the hamming weight of the numbers in MGNAF representation does not increase as it does in GNAF form. So the nonzero density of the numbers in MGNAF form is lower than same number in GNAF form.

Moreover, another advantage of MGNAF which is important is the impact of this representation on higher radixes in comparison with GNAF. As the radix goes higher MGNAF performs much better than GNAF and the impact of MGNAF form on the hamming weight increases in comparison with the GNAF improvement. For example, in radix 3, MGNAF can reduce the hamming weight about 22% which is more than 30% better

than GNAF does (16.67% for GNAF) but this improvement over GNAF increases to more than 65% for radix (Figure 2).

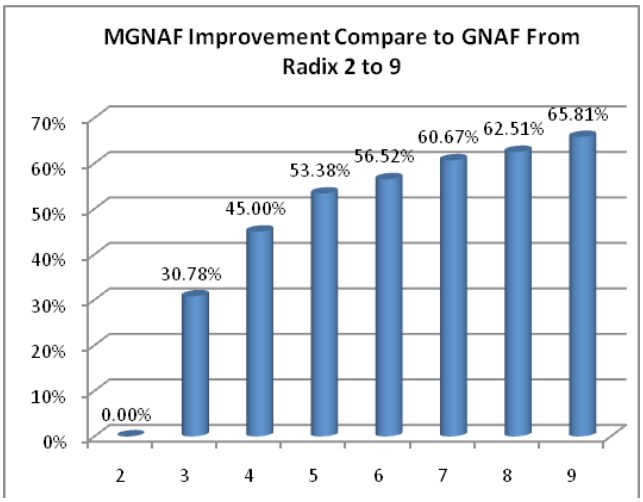

Figure 2. MGNAF over GNAF Improvement.

The additional digits in the MGNAF digit set are the result of the first part of the third step in its creation algorithm. Therefore, we can calculate the number of these extra digits based on this step. As it has been mentioned before, a sequence of same non-zero numbers (here for example $\underbrace{x,x,\ldots,x}_{n}$) would change to $\frac{x}{r-1}$ followed by n-1 zeros and $\frac{x}{r-1}$ at the end, in which the $\frac{x}{r-1}$ would be added to the digit before these consecutive x-es. As $x \in \{1, 2,..., r-1\}$ so $\frac{x}{r-1}$ Would have r-1 different results, but if we choose x = r-1, $\frac{x}{r-1}$ would become 1 which is same as GNAF. So $\frac{x}{r-1}$ will have r-2 new results and these results would be added to a digit from {0,1,...,r-1} which are r different numbers. In conclusion $r(r$-2) new digits would be created for MGNAF algorithm for radix r in comparison with GNAF(Table 2).

Table 2. Comparison of GNAF and the proposed MGNAF

Radix	2	3	4	5	6	7	8	9
Extra Digits*	0	3	8	15	24	35	48	63

*Note that the number of digits ignoring their sign was counted

5. Acknowledgment

The authors would like to thank the Universiti Sains Malaysia (USM) for supporting this research through Project Grant (1001/PKOMP/817059).

6. References

1. Elgamal T. A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Trans. Inf. Theory. Jul. 1985; 31(4):469–72. Crossref.
2. Menezes A, Van Oorschot P, Vanstone S. Handbook of applied cryptography. CRC Press, 1997.
3. Rivest R, Shamir A, Adleman L. A method for obtaining digital signatures and public-key cryptosystems, Commun. ACM. 1978; 21(2):120–26. Crossref.
4. Koc C. High-speed RSA implementation, Technical Report TR 201, RSA Laboratories, 1994.
5. Cois Morain F, Olivos J. Speeding up the computations on an elliptic curve using addition-subtraction chains, Theor. Informatics Appl. 1990; 24:531–43. Crossref.
6. Clark W, Liang J. On arithmetic weight for a general radix representation of integers, Inf. Theory, IEEE Trans. 1973; 19:823–26.
7. Joye M, Yen S. New minimal modified radix-r representation with applications to smart cards. In: Public Key Cryptography, 2002, p. 375–83.
8. Gordon D. A survey of fast exponentiation methods, J. Algorithms. 1998; 27:129–46. Crossref.
9. Reitwiesner G. Binary arithmetic, Adv. Comput. 1960; 1:231–308. Crossref.
10. Ebeid N, Hasan M. On binary signed digit representations of integers, Des. Codes Cryptogr. 2007; 42:43–65. Crossref.
11. Wu T, Zhang M, Du H, Wang R. On optimal binary signed digit representations of integers, Appl. Math. J. Chinese Univ. Aug. 2010; 25(3):331–40. Crossref.
12. Arno S, Wheeler F. Signed digit representations of minimal Hamming weight, Comput. IEEE Trans. 1993; 42(8):1007–10. Crossref.
13. Group IW. IEEE 1363-2000: Standard specifications for public key cryptography. IEEE Stand. IEEE, New York: NY, 2000.