

# On The Efficiency Analysis of wNAF and wMOF

Diploma thesis  
Supervised by Professor Dr. Tsuyoshi Takagi



Rong Fan

Technische Universität Darmstadt, Fachbereich Informatik  
fanrong@rbg.inforamtik.tu-darmstadt.de

7. September 2005



## Acknowledgment

Here at the beginning I want to thank Junior-Professor Dr. Tagaki, who has given me this very interesting topic and helped me to find out a way to begin and finish this work. All the suggestions and his graceful patients on answering my questions are really appreciated. He spent always his precious time talking about the way how to write a report and how to work effective. It is very important to me, and it is also my precious experience.

I want to also thank my friends, Xiaoyue Gong, Katja Schmidt-Samoa, Jun Li etc, who have given me a lot of help. On pointed out some mistakes in this paper, so I could modify them in time; and they also helped me to compute some necessary data by using their private time.

At last I must thank to my parents, they always support and encourage me by my side on my study in Germany.



## Abstract

Elliptic Curve Cryptosystems (ECC)[1] are much more suitable for small memory device such as smart cards rather than Rivest Shamir Adleman (RSA) cryptosystems due to their small key size. In fact, an EC cryptosystem with a 160-bit-key-size and a RSA cryptosystem with a 1024-bit-key-size both have the same security level. However, when put the ECC into practice, the major problem is how to enhance the speed of the scalar multiplication, which the most important operation in ECC with limited memory resources. The most common scalar multiplication algorithm based on the binary method with pre-computation, for example sliding window on NAF, wNAF, MOF and wMOF.

In this paper, we will analyze some details of wNAF (w-width non-adjacent-form) and wMOF (w-width mutual-opposite-form), from which we know how they raise the speed of scalar multiplication in ECC, how much efficiency they have raised and some other details. Our basic aim is to analyze some attributes of wNAF and wMOF, to compare them with theories and to test them by practice.

For the exhaustive analysis of ECC scalar multiplication with a particular exponent recoding method, it is very important to know the average amount of 0s and non-0s in the achieved form, the average length of consecutive 0s and the average bit-length of the output in the achieved form. If we knew them in advance, we could optimize the code and save more memory resources.

In this paper we will discuss these three attributes of wNAF form and wMOF form.

- (1) What is the average amount of 0s and non-0s?
- (2) What is the average length of consecutive 0s?
- (3) What is the average bit-length?



# Inhaltsverzeichnis

<b>1</b>	<b>Introduction</b>	<b>9</b>
<b>2</b>	<b>Preliminaries</b>	<b>11</b>
2.1	Elliptic Curves . . . . .	11
2.2	Definitions . . . . .	12
2.3	Probability Theory . . . . .	13
2.4	Markov Chain . . . . .	14
2.5	Scalar Multiplication . . . . .	16
<b>3</b>	<b>wNAF and wMOF</b>	<b>18</b>
3.1	wNAF . . . . .	18
3.1.1	NAF . . . . .	18
3.1.2	wNAF . . . . .	19
3.1.3	Fractional wNAF . . . . .	20
3.2	wMOF . . . . .	22
3.2.1	MOF . . . . .	22
3.2.2	wMOF . . . . .	23
3.2.3	Fractional wMOF . . . . .	25
3.3	Optimal Window Size . . . . .	27
3.4	Conclusion . . . . .	28
<b>4</b>	<b>Attributes of wNAF and wMOF</b>	<b>29</b>
4.1	Amount of 0s and non-0s in wNAF and wMOF . . . . .	29
4.1.1	Amount of 0s and non-0s in wNAF . . . . .	29
4.1.2	Amount of 0s and non-0s in wMOF . . . . .	31
4.1.3	Amount of 0s and non-0s in fractional wNAF . . . . .	32
4.1.4	Amount of 0s and non-0s in fractional wMOF . . . . .	35
4.1.5	Conclusion and Test . . . . .	36
4.2	Average length of consecutive 0s in wNAF and wMOF . . . . .	37
4.2.1	Average length of consecutive 0s in wNAF . . . . .	37
4.2.2	Average length k of consecutive 0s in wMOF . . . . .	39
4.2.3	Average length of consecutive 0s in fractional wNAF . . . . .	40
4.2.4	Average length k of consecutive 0s in fractional wMOF . . . . .	41
4.2.5	Conclusion and Test . . . . .	42
4.3	Average bit-length of wNAF or wMOF . . . . .	43
4.3.1	Average bit-length of wNAF . . . . .	43
4.3.2	Average bit-length of wMOF . . . . .	47
4.3.3	Average bit-length of fractional wNAF . . . . .	48
4.3.4	Average bit-length of fractional wMOF . . . . .	51

---

4.3.5	Conclusion and Test . . . . .	52
<b>5</b>	<b>Conclusion</b>	<b>55</b>
<b>6</b>	<b>Data</b>	<b>56</b>
6.1	Data of wNAF . . . . .	56
6.2	Data of wMOF . . . . .	57
6.3	Data of fractional wNAF . . . . .	58
6.4	Data of fractional wMOF . . . . .	59
<b>7</b>	<b>Appendix</b>	<b>62</b>

# 1 Introduction

The most commonly used public key cryptosystem today is RSA. It was developed by Ron Rivest, Adi Shamir and Leonard Adleman in 1977 [2] [3]. It is an algorithm that depends on a one-way function involving the factorization of a large integer  $n$ . Here  $n$  is a product of two large prime numbers,  $p$  and  $q$ . The one-way function RSA is defined as exponentiation modulo  $n$ , which can be computed efficiently. However, inverting the RSA function is believed to be intractable if  $p$  and  $q$  are unknown.

Elliptic curve cryptography (ECC) provides a relatively novel alternative to RSA. Both RSA and ECC can establish secure communications. RSA is based on the integer factorization problem whilst ECC is based on the discrete logarithm problem. Elliptic Curve Cryptosystems (ECC) became a very important technology for cryptography because of their high security level with shorter key-size and faster computation than any other existing cryptographic schemes. We compared ECC with RSA and got the result of Table 1. It tells us that with the same key-size, ECC is much more efficient than RSA and DSA. The superiority will be more and more obvious when the key-size grows. Figure 1 illustrates the proportion of key-length between ECC and RSA when the bit-length grows. We conclude from these two Figures that ECC are suitable, especially for small-memory device. They improve the speed of encryption and decryption as well as the signature generation and verification. They also save much memory resources. The dominant com-

Time of crack (ns)	RSA bit-length	ECC bit-length	RSA/ECC
$10^4$	512	106	5:1
$10^8$	768	132	6:1
$10^{11}$	1024	160	7:1
$10^{20}$	2048	210	10:1
$10^{78}$	21000	600	35:1

Tabelle 1: comparison of RSA & ECC

putation of all ECC algorithms is scalar multiplication, i.e. the computation of  $d * P$  with an integer  $d$  and a point  $P \in E(K)$ . Elliptic curve addition (ECADD) and Elliptic curve doubling (ECDBL) are the basic operations for the scalar multiplication. The well-known binary method needs one ECDBL for each bit '0' of  $d$  and one ECDBL plus one ECADD for every bit '1' of  $d$ . So the major job of the optimization is to reduce the amount of non-zero-bits. wNAF is a good choice because it minimizes the pre-computation

effort whilst the non-zero density is nearly optimal. For example, we will transfer a  $w$ -width binary string into a  $w$ -width NAF whilst only the last bit is non-zero, it means in this  $w$ -width string has only one non-zero bit. Unfortunately, wNAF can only be computed from the least significant bit, i.e. Right-to-Left. So the first step (recoding) in scalar multiplication with wNAF is done Right-to-Left and the second step (evaluation) in scalar multiplication is done Left-to-Right (See Appendix: Scalar Multiplication with wNAF). It is impossible to merge recoding and evaluation together, so it means we need extra  $O(n)$  ( $n$  is the bit-length of integer  $d$ ) memory storage for the recoding. The new scheme wMOF can be computed from the least significant bit (Left-to-Right). The first step (recoding) and the second step (evaluation) can be merged when it is done Left-to-Right (see Appendix: Scalar Multiplication with wMOF) and it means when we compute the scalar multiplication with wMOF, it saves extra  $O(n)$  memory storage.

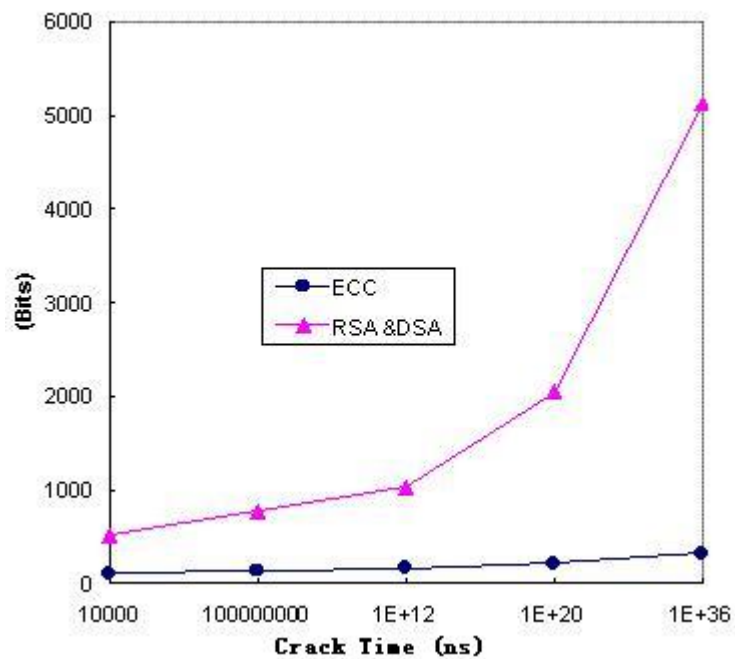


Abbildung 1: compare RSA & ECC

## 2 Preliminaries

This chapter introduces some preliminaries, including some basic knowledge about elliptic curves and probability theory, which will be used in this paper to derive some interesting properties of wNAF and wMOF. We will also give some useful definitions in this chapter.

### 2.1 Elliptic Curves

In 1985, Neal Koblitz and Victor Miller independently introduced the Elliptic Curve Cryptography as another alternative public key cryptography. Its security relies on the difficulty of the elliptic curve discrete logarithm problem that allows the delivery of the highest strength per bit of all previously known public key cryptosystems. The security level is determined by the size and structure of the group of rational points lying on an elliptic curve defined over a finite field.

Let  $K$  be a field, an elliptic curve over  $K$  can be defined as the following set:

$$E(K) : \{(x, y) \in K \times K \mid y^2 = x^3 + ax + b\} \cup O$$

$$\text{where } (a, b \in K, b \neq 0, 4a^3 + 27b^2 \neq 0) \quad (1)$$

Here,  $O$  is called the point at infinity and for every  $P$  on the curve, we have  $P = P + O$  [4] [5].

If  $P$  is different from the point  $O$ , then  $-P$  is the point with the same  $x$ -coordinate as  $P$  but with negative  $y$ -coordinate, that is  $-P = -(x, y) = (x, -y)$ . It is clear that, when  $(x, y)$  is on the curve, then  $(x, -y)$  is also on this curve. For the point  $Q = -P$  holds  $P + Q \rightarrow O$ .

This shows that we can define a rule of adding points on the curve such that  $E(K)$  achieves an additive group structure with identity element  $O$ . For two points  $P_1$  and  $P_2$  on the curve  $E(K)$ ,  $P_1 + P_2 (P_1 \neq P_2)$  is called ECADD and  $P_1 + P_2 = 2P_1 (P_1 = P_2)$  is called ECDBL.

The idea, which we usually compute  $P_1 + P_2 (P_1 \neq P_2)$  is, find the line  $l = P_1P_2$ , and it will intersect the curve at exactly one more point  $P_3$ , define  $P_1 + P_2 (P_1 \neq P_2)$  to be  $P_3$ , which is the mirror image of the third point of intersection with respect to the  $x$ -axis.

For the case where  $P_1 + P_2 = 2P_1$  ( $P_1 = P_2$ ), we let  $l$  be the tangent line to the curve at  $P_1$  ( $P_2$ ) and let  $P_3$  be the only other point of intersection of  $l$  with the curve. Define  $2P_1 = P_3$ .

The attribute of elliptic curve is when we have an integer  $n$  and a point  $P$ , it is easy to compute the  $Q = nP$ , but when we have only the point  $P$  and the product  $Q$ , it is hard to recover  $n$ . The latter is called the discrete logarithm assumption, and it is the basis for the security of ECC.

Following is an example of using the elliptic curve encryption schema to send a message. We suppose that B wants to send a message to A.

- 1 A must send the public key to B. A must choose an elliptic curve and a point  $P = (xp, yp)$  on the curve, then choose an integer  $n$  where  $2 \leq n \leq (\#E - 1)$  and compute  $Q = nP$ , after which A can send the key  $(P, Q, p, \#E)$  to B and keep  $n$  as a secret key.
- 2 Now B can send a message to A, We suppose the message is  $(x_1, y_1)$ , so B must firstly choose a random number  $k$ , where  $2 \leq k \leq (\#E - 1)$ , then compute  $R = kP$ ,  $(x_2, y_2) = kQ$  and  $(x_3, y_3) = (x_2x_1, y_2y_1)(mod p)$ , B can send the triple  $(R, x_3, y_3)$  to A now.
- 3 Now A has received the cipher text, Because  $(x_2, y_2) = nR$ ,  $(x_1, y_1) = (x_3/x_2, y_3/y_2)(mod p)$ .

In this process, A and B didn't sent the secret key  $n$ , we suppose there is a attacker, who can receive all the information, which transferred from A to B or B to A, so he can get  $P, Q, p, \#E, R, e, f$ , but it is hard for him to find out which  $n$  was used by A and B. We have sent out the message successfully and safely.

## 2.2 Definitions

In this paper we need some definitions to explain theorems, following is a list of them.

- (1)  $Z(w, k)$  : The amount of 0s, where  $w$  is the window size and  $k$  is the length of input.
- (2)  $NZ(w, k)$  : The amount of non-0s, where  $w$  is the window size and  $k$  is the length of input.

- (3)  $CZ(w, k)$  : Average length of consecutive 0s, where  $w$  is the window size and  $k$  length of input.
- (4)  $GC(w, k)$  : The probability that the output will generate a extra carry bit, where  $w$  is the window size and  $k$  length of input.
- (5)  $L(w, k)$  : Average length of the output, where  $w$  is the window size and  $k$  length of input.
- (6)  $EL(w, k)$  : Average efficacious length of the output, where  $w$  is the window size and  $k$  length of input. It means the length from first non-0 bit to end.
- (7)  $Pr(x)$  : The probability, that  $x$  is satisfied.

## 2.3 Probability Theory

In order to analyze some attributes of wNAF and wMOF, we need to use a few probability theories, so we introduce the probability theories [6] in advance. The following three axioms are known as the Kolmogorov axioms:

**Axiom 1** For any event  $E$ ,  $0 \leq P(E) \leq 1$ .

That is, the probability of an event is represented by a real number between 0 and 1.

**Axiom 2**  $P(\Omega) = 1$ .

That is, the probability that some elementary event occurs in the entire sample set is 1. More specifically, there are no elementary events outside the sample set. This is often overlooked in some mistaken probability calculations. If you can't precisely define the whole sample set, the probability of any subset can't be defined either.

**Axiom 3** Any countable sequence of mutually disjoint event  $E_1, E_2, \dots, E_n$  satisfies  $P(E_1 \cup E_2 \cup \dots \cup E_n) = \sum_{i=1}^n P(E_i)$ .

That is, the probability of an event set which is the union of other disjoint subsets is the sum of the probabilities of those subsets.

From the Kolmogorov axioms we can deduce other useful rules to calculating probabilities:

**Lemma 1**  $P(A \cup B) = P(A) + P(B) - P(A \cap B)$ .

That is, the probability that A or B will happen is the sum of the probabilities that A will happen and that B will happen, minus the probability that A and B will happen.

**Lemma 2**  $P(\Omega - E) = 1 - P(E)$ .

That is, the probability that E will not happen is 1 minus the probability that E will happen.

**Lemma 3**  $P(A \cap B) = P(A)P(B | A)$ .

That is, the probability that A and B will happen is the probability that A will happen, times the probability that B will happen given that A happened.

**Lemma 4** *If A and B are independent, we have  $P(A \cap B) = P(A)P(B)$ .*

That is, the probability that A and B will happen is the probability that A will happen times the probability that B will happen, when A and B are independent.

## 2.4 Markov Chain

In this paper we need use some Markov theory to compute some probabilities, so we introduce Markov Chain in advance [8]. A Markov chain is a sequence of random values whose probabilities at a time interval depends upon the value of the number at the previous time. A simple example is the non-returning random walk, where the walkers are restricted to not go back to the location just previously visited.

The controlling factor in a Markov chain is the transition probability, it

is a conditional probability for the system to go to a particular new state, given the current state of the system. For many problems, such as simulated annealing, the Markov chain obtains the much desired importance sampling. This means that we get fairly efficient estimates if we can determine the proper transition probabilities.

The construction of a Markov chain requires two basic ingredients, namely a transition matrix and distribution. We start with the definition of the transition matrix. Assume a finite set  $S = \{1, \dots, m\}$  of states. Assign to each pair  $(i, j) \in S^2$  of states a real number  $P_{ij}$  such that the properties

- (1)  $P_{ij} \geq 0 \forall (i, j) \in S^2$ ,  $P_{ij}$  means the transition probability from state  $i$  to state  $j$ .
- (2)  $\sum_{j \in S} p_{ij} = 1 \forall i \in S$

are satisfied and define the transition matrix  $P$  by

$$P = \begin{pmatrix} p_{11} & p_{12} & \cdot & \cdot & \cdot & p_{1m} \\ p_{21} & p_{22} & \cdot & \cdot & \cdot & p_{2m} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ p_{m1} & p_{m2} & \cdot & \cdot & \cdot & p_{mm} \end{pmatrix}$$

The stationary distribution is  $\pi = (\pi_1, \pi_2, \dots, \pi_m)$ , and satisfied with the condition

$$(\pi_1, \pi_2, \dots, \pi_m) * \begin{pmatrix} p_{11} & p_{12} & \cdot & \cdot & \cdot & p_{1m} \\ p_{21} & p_{22} & \cdot & \cdot & \cdot & p_{2m} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ p_{m1} & p_{m2} & \cdot & \cdot & \cdot & p_{mm} \end{pmatrix}^n = (\pi_1, \pi_2, \dots, \pi_m)$$

**Theorem 1** *Let  $P$  be the transition matrix of a Markov chain. The  $ij$ -th entry  $p_{ij}^{(n)}$  of the matrix  $P^n$  gives the probability that the Markov chain, starting in state  $s_i$ , will be in state  $s_j$  after  $n$  steps.*

Here is an example to explain how we get the stationary distribution of a Markov Chain matrix.

$$\begin{array}{ccc} P & Q_1 & Q_2 \\ Q_1 & a & b \\ Q_2 & c & d \end{array}$$

In this matrix, it must fulfil the condition:

$$a + b = c + d = 1$$

We suppose the stationary distribution of this Markov Chain matrix is:  $(\pi(Q_1), \pi(Q_2))$ . It means

$$(\pi(Q_1), \pi(Q_2)) * \begin{pmatrix} a & b \\ c & d \end{pmatrix} = (\pi(Q_1), \pi(Q_2))$$

It equals to:

1.  $a * \pi(Q_1) + c * \pi(Q_2) = \pi(Q_1)$
2.  $b * \pi(Q_1) + d * \pi(Q_2) = \pi(Q_2)$

Finally we can get the stationary distribution.

$$\pi = \pi(Q_1), \pi(Q_2) = \left( \frac{b}{b - a + 1}, \frac{1 - a}{b - a + 1} \right)$$

## 2.5 Scalar Multiplication

This thesis analyzes how to improve the speed of scalar multiplication with wNAF and wMOF. So what is scalar multiplication? Here is an algorithm [7], in this algorithm we suppose  $d$  is an integer and  $d$  is given in a specially representation involving the digits  $\{0, 1, 3, 5, \dots, 2^w - 1\}$  with a specially form  $d = [d_{n-1}, \dots, d_0]_w$ , which the value of  $d$  is  $d = \sum_{i=0}^{n-1} 2^i d_i$  and  $w$  is window size.  $P$  is a point, now we use scalar multiplication to compute the value of  $dP$ . The aim of the algorithms wMOF and wNAF is to construct such special representations. This is also called recoding".

---

**Algorithm 1** Scalar Multiplication

---

*Input:* integer  $d$  and specially form  $d = [d_{n-1}, \dots, d_0]_w$ , where  $d_i = 0$  or  $d_i \in T$ ,  $T = \{1, 3, 5, \dots, 2^w - 1\}$ , a point  $P$

*Output:*  $dP$

Let  $Q = P$

**for**  $i = n - 2$  to  $0$  **do**

$Q = DBL(Q)$ ;

**if**  $d_i \neq 0$  **then** **then**

$Q = ADD(Q, d_i * P)$

**end if**

**end for**

**Return**  $Q$

---

### 3 wNAF and wMOF

Here we will introduce wNAF and wMOF step by step. We start with NAF and MOF, then wNAF and wMOF, at the end we will introduce fractional wNAF and fractional wMOF. The latter are specially forms of wNAF and wMOF. The window size in fractional wNAF and fractional wMOF [9] was made up of an integer and a fraction like form  $w_0 + w_1$ , where  $w_0$  is an integer and  $w_1$  is an appropriate fraction. It means that the window size can be either  $w_0$  or  $w_0 + 1$ . Fractional wNAF and fractional wMOF are a good way to use the memory resources fully. More details will be explained in sub-section 3.1.3 and sub-section 3.2.3.

When  $w = 2$ , We name them 2NAF and 2MOF. 2NAF is same as NAF, but 2MOF isn't MOF.

#### 3.1 wNAF

##### 3.1.1 NAF

When we compute the  $nP$  with ECC scalar multiplication [10] [11], we must first transform the integer  $n$  into binary form. Then multiply the binary form with  $P$ , it is a typical elliptic curve scalar multiplication. In this multiplication, the probabilities of every bit with the value of '0' or '1' are  $1/2$ . And when the length of the input is  $n$ , so we need about  $n$  times ECDBL and  $n/2$  times ECADD ( for every bit '1' needs one time ECDBL and one time ECADD; and for every bit '0' needs only one time ECDBL ). But if we compute the scalar multiplication with NAF form instead of the binary form, it will enhance the speed of multiplication. NAF is a special wNAF with window size 2. The binary input was converted Right-to-Left with such a rule as: When the current bit is '0', it will be kept, and when the current bit is '1', the next two bits should be regarded as one block. It means '01' remains '01' while '11' will be replaced with '10 $\bar{1}$ '. In the second case, the block generates a carry. It changes the value of next bit, but this is no problem for analyzing asymptotic properties ( because the next bit can be either '0' or '1', and when it gets a carry from the preceding bit, it has same distribute of

'0' and '1' ). It saves one time ECADD with using '0' instead of '1'. So when we compute the elliptic curve scalar multiplication with NAF form instead of binary form, it only needs about  $n$  times ECDBL and  $n/3$  times ECADD. Following is an example of this transform:

$$1011011001 = 10\bar{1}00\bar{1}0\bar{1}001$$

### 3.1.2 wNAF

wNAF has the window size  $w$  and it is converted Right-to-Left from binary form. Following is the definition of wNAF [12] [13]:

**Definition 1** *A sequence of signed digits is called wNAF iff the following three properties hold:*

- 1 *The most significant non-zero bit is positive.*
- 2 *Among any  $w$  consecutive digits, at most one is non-zero.*
- 3 *Each non-zero digit is odd and less than  $2^{w-1}$  in absolute value.*

There are several algorithms, which can be used to convert binary form into wNAF. Algorithm 2 is proposed by Solinas. In this algorithm, "a mod b"

---

#### Algorithm 2 Generation of wNAF

---

*Input:* width  $w$ , an  $n$ -bits integer  $d$

*Output:* width  $w$ ,  $\{\sigma_n, \sigma_{n-1}, \dots, \sigma_0\}_w$  is wNAF of  $d$

$i \leftarrow 0$

**while**  $d \geq 1$  **do**

**if**  $d$  is even **then**

$\sigma_i \leftarrow 0$ ;

**else**

$\sigma_i \leftarrow d \bmod 2^w$ ;      $d \leftarrow d - \sigma_i$ ;

**end if**

$d \leftarrow d/2$ ;      $i \leftarrow i + 1$ ;

**end while**

**Return**  $\{\sigma_n, \sigma_{n-1}, \dots, \sigma_0\}_w$

---

means the signed modulo, it is defined as  $a \bmod b$  and  $-\frac{b}{2} \leq a \leq \frac{b}{2}$ , and it means that the "mods" generates a carry sometimes. Finally we list all of the possible values of non-zero bits used for wNAF,

$$T = \{\pm 1, \pm 3, \pm 5, \dots, \pm(2^{w-1} - 1)\}$$

Following is an example of this transform with the window size 5:

$$1011011001 = 10000\bar{9}0000\bar{7}$$

### 3.1.3 Fractional wNAF

Fractional wNAF is a little different from wNAF. From sub-section 3.1.2 we know when we compute a scalar multiplication with wNAF, firstly we must pre-compute the values of all  $P_{2j+1}$ , where  $0 \leq j \leq 2^{w-1} - 1$  and  $P_{2j+1} = (2j+1)P$ , that is we used to store  $2^{w-2}$  points. So when the window size is incremented, it needs nearly double memory size to store the values of  $P_{2j+1}$ . It is not quite suitable for small memory storage device. For example, we have a smart card which can only store 100 pre-computed points. When  $w = 8$ , the amount of the pre-computed values are  $2^6 = 64$ , the rest 36 memory place are wasted. We can't choose  $w = 9$ , because it needs 128 memory place and it is bigger than the memory size 100.

Fractional wNAF can solve the problem successfully. Now we suppose the  $q$  is the maximum memory size, and  $w_0$  is the maximum value which we can choose with  $2^{w_0-2} \leq q \leq 2^{w_0-1}$ , so the remaining place in the memory is  $r = q - 2^{w_0-2}$ , Now we introduce a new notation  $w_1$ , where  $w_1 = r/2^{w_0-2}$ . We notice that  $0 \leq r < 2^{w_0-2}$  and  $0 \leq w_1 < 1$ . Compare with the wNAF, we need pre-compute the table from  $P_1$  to  $P_{2j+1}$  in wNAF, where  $0 \leq j \leq 2^{w_0-2} - 1$ , all the values build a set  $\{1P, 3P, \dots, (2^{w_0-1} - 1)P\}$ . But in fractional wNAF, the memory place will be completely used. The set is  $\{1P, 3P, \dots, (2q+1)P\}$ , we use  $w_0$  and  $w_1$  instead of  $q$ , so we can get the set  $\{1P, 3P, \dots, (2^{w_0-1} - 1)P, (2^{w_0-1} + 1)P, \dots, (2^{w_0-1} - 1 + w_1 2^{w_0-1})P\}$ . This table has  $q = (1 + w_1)2^{w_0-2}$  elements. Here is  $w = w_0 + w_1$ . In the above example, we can select the window size  $8\frac{36}{64}$ , the amount of the pre-computed value are  $(1 + \frac{36}{64}) * 2^6 = 100$ .

The window size  $w$  will be sometimes  $w_0$  and sometimes  $w_0 + 1$  in this algorithm. We must put in an additional step in the algorithm to confirm, whether the values with window size  $w_0 + 1$  is stored in the table. When it was not pre-computed and stored in the table, we must change the current window size  $w_0 + 1$  into  $w_0$  and recompute the value.

The conversion from a binary form into fractional wNAF is also done Right-to-Left, Algorithm 3 is the algorithm, which converts the binary form into fractional wNAF. In this algorithm, "mods" has the same meaning as in Algorithm 2, it was defined as  $a \bmod b$  and  $-b/2 \leq a \leq b/2$ , so Algorithm 3 can also generate a carry at the most bit. All of the possible values of non-zero bits build a set  $T$ , where

$$T = \{\pm 1, \pm 3, \dots, \pm(2^{w_0-1} - 1), \pm(2^{w_0-1} + 1), \dots, \pm(2^{w_0-1} - 1 + w_1 2^{w_0-1})\}$$

---

**Algorithm 3** Generation of fractional wNAF
 

---

*Input:* integer  $d$ , a fractional width  $w = w_0 + w_1$

*Output:*  $\{\sigma_n, \sigma_{n-1}, \dots, \sigma_0\}_w$

$i \leftarrow 0$ ;

$k \leftarrow d$ ;

**while**  $k > 0$  **do**

**if**  $k$  is odd **then**

$\sigma_{w_0+1}[i] \leftarrow k \bmod 2^{w_0+1}$ ;

**if**  $|\sigma_{w_0+1}[i]| > ((1 + w_1)2^{w_0+1} - 1)$  **then**

$\sigma_w[i] \leftarrow k \bmod 2^{w_0}$ ;

**else**

$\sigma_w[i] \leftarrow \sigma_{w_0+1}[i]$ ;

**end if**

$k \leftarrow k - \sigma_w[i]$ ;

**else**

$\sigma_w[i] \leftarrow 0$ ;

**end if**

$k \leftarrow k/2$ ;      $i \leftarrow i + 1$ ;

**end while**

**Return**  $\{\sigma_n, \sigma_{n-1}, \dots, \sigma_1, \sigma_0\}_w$

---

But we must focus on the case, when  $w = 2.*$ . It is a special case for fractional wNAF. And we can prove that the result, which was converted into fractional wNAF with the window size  $w = 2.*$ , will be same as the result, which was

converted into wNAF with the window size  $w = 2$ . Because for  $w = 2$ , we need  $2^{w-2} = 1$  pre-computed point, and for  $w = 3$ , we need  $2^{w-2} = 2$  pre-computed points. Hence there is no gap between the table sizes for  $w=2$  and  $w=3$ , consequently fractional wNAF is only reasonable for  $w > 2$ . To be more concrete,  $w$  is a valid fractional window size, if  $w = w_0 + w_1$  where  $w_0 = \lfloor w \rfloor$  and  $w_1 = r/2^{w_0-2}$  for an  $r$  between 0 and  $2^{w_0-2} - 1$ .

Because of this reason, all of the window size in the next section, which we used in the tests and proofs for fractional wNAF, must satisfied with the condition  $w_0 > 2$ .

## 3.2 wMOF

### 3.2.1 MOF

We have introduced NAF, which must be converted from Right-to-Left in sub-section 3.1.1. The Right-to-Left generation is a great disadvantage, because the evaluation stage in elliptic curve multiplication must be done from Left-to-Right, which means we need extra  $O(n)$  memory usage to store the NAF, where  $n$  is the bit length of input. In order to improve the efficiency of the ECC scalar multiplication, we need to find out a new form. With the new form, the recoding stage can be done Left-to-Right. At CRYPTO-2004 a new signed binary representation was introduced, which is MOF [14]. With this form we can convert the binary form either from Left-to-Right or from Right-to-Left. Conversion from a binary form into MOF is very simple, it is  $2d \ominus d$ , where  $d$  is the binary form  $d_{n-1}d_{n-2} \dots d_0$  and ' $\ominus$ ' denotes bitwise subtraction. So the  $i$ -th bit in MOF is computed with  $md_i = d_{i-1} - d_i$ , before the computation we must suppose that  $d_{-1} = 0$  and  $d_n = 0$ . Following is an example:

Input: 110110000

Output: (1-0)(1-1)(0-1)(1-0)(1-1)(0-1)(0-0)(0-0)(0-0)(0-0)

Result: 10 $\bar{1}$ 10 $\bar{1}$ 0000

**Lemma 5** *The mutual opposite form (MOF) is an signed binary string that satisfies the following properties:*

- 1 *The signs of adjacent non-zero bits (without considering 0 bits) are opposite.*

- 2 The most non-zero bit and the least non-zero bit are 1 and -1, respectively.

### 3.2.2 wMOF

The non-zero density of MOF is the same as for binary, namely  $\frac{1}{2}$ . To decrease the amount of non-zero digits, we have to apply window method on MOF. The result is denoted wMOF, if a window size  $w$  is used. Following is the definition of wMOF.

**Definition 2** A sequence of signed digits is called wMOF iff the following three properties hold:

- 1 The most significant non-zero bit is positive.
- 2 All but the least significant non-zero digit  $x$  are adjoint by  $w-1$  zeros as follows:
  - in case of  $2^{k-1} < |x| < 2^k$  for an integer  $2 \leq k \leq w-1$  the pattern equals  $\underbrace{0 \dots 0}_k x \underbrace{0 \dots 0}_{w-k-1}$
  - in case of  $|x| = 1$  either the pattern equals  $x \underbrace{0 \dots 0}_{w-1}$  and the next lower non-zero digit has opposite sign from  $x$  or the pattern equals  $0x \underbrace{0 \dots 0}_{w-2}$  and the next lower non-zero digit has the same sign as  $x$ . If  $x$  is the least significant non-zero digit, it is possible that the number of right-hand adjacent zeros is smaller than stated above. In addition it is not possible that the last non-zero digit is a 1 following any non-zero digit.
- 3 Each non-zero digit is odd and less than  $2^{w-1}$  in absolute value.

wMOF is based upon MOF, it means when we begin the conversion Right-to-Left, if the current bit is same as the next bit, the bit will be stored as '0'. If the current bit is differ from the next bit, we regard the next  $w$  bits as one block, convert the block using MOF conversion and store the value in wMOF.

We must take care that the length of the last block may be shorter than  $w$ . Algorithm 4 tells us how to convert a binary form into wMOF.

Before the conversion with algorithm 4, we must create a matching table

---

**Algorithm 4** Generation of wMOF
 

---

*Input:* width  $w$ , a  $n$ -bits binary with form  $d = \{d_{n-1}, d_{n-2}, \dots, d_0\}_2$ .  
*Output:* wMOF  $\{\sigma_n, \sigma_{n-1}, \dots, \sigma_0\}_w$  of  $d$   
 $d_{-1} \leftarrow 0; \quad d_n \leftarrow 0; \quad i \leftarrow n;$   
**while**  $i \geq w - 1$  **do**  
  **if**  $d_i = d_{i-1}$  **then**  
     $\sigma_i \leftarrow 0; i \leftarrow i - 1;$   
  **else**  
     $(\sigma_i, \sigma_{i-1}, \dots, \sigma_{i-w+1}) \leftarrow Table_{wSW} \overrightarrow{(d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{i-w} - d_{i-w+1})};$   
     $i \leftarrow i - w;$   
  **end if**  
**end while**  
**if**  $i \geq 0$  **then**  
   $(\sigma_i, \sigma_{i-1}, \dots, \sigma_0) \leftarrow Table_{i+1SW} \overrightarrow{(d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_0 - d_1, -d_0)};$   
**end if**  
**Return**  $\{\sigma_n, \sigma_{n-1}, \dots, \sigma_1, \sigma_0\}_w;$

---

first, which involve all the possible values with the length  $w$ . For example, in the case when  $w = 3$ , we get the following matching table  $Table_{3SW} \overrightarrow{\phantom{}}$ :

$$\begin{array}{cccc} 100 \rightarrow 100 & \bar{1}00 \rightarrow \bar{1}00 & 1\bar{1}0 \rightarrow 010 & \bar{1}10 \rightarrow 0\bar{1}0 \\ & & & \\ & 1\bar{1}1 \rightarrow 003 & \bar{1}\bar{1}\bar{1} \rightarrow 00\bar{3} & \\ & 10\bar{1} & \bar{1}01 & \end{array}$$

When  $w = 4$ , we get the following matching table  $Table_{4SW} \overrightarrow{\phantom{}}$ :

$$\begin{array}{cccc} 1000 \rightarrow 1000 & \bar{1}000 \rightarrow \bar{1}000 & 1\bar{1}00 \rightarrow 0100 & \bar{1}100 \rightarrow 0\bar{1}00 \\ & & & \\ & 1\bar{1}10 \rightarrow 0030 & \bar{1}\bar{1}\bar{1}0 \rightarrow 00\bar{3}0 & \\ & 10\bar{1}0 & \bar{1}010 & \\ & & & \\ & 1\bar{1}01 \rightarrow 0005 & \bar{1}\bar{1}\bar{1}1 \rightarrow 000\bar{5} & \\ & 1\bar{1}\bar{1}\bar{1} & \bar{1}10\bar{1} & \end{array}$$

$$\begin{array}{cc} 100\bar{1} & \rightarrow 0007 \\ 10\bar{1}1 & \end{array} \quad \begin{array}{cc} \bar{1}001 & \rightarrow 000\bar{7} \\ \bar{1}0\bar{1}\bar{1} & \end{array}$$

All the non-zero-bits build such a set  $T = \{\pm 1, \pm 3, \pm 5, \dots, \pm(2^{w-1} - 1)P\}$ , it means wMOF requires  $2^{w-2}$  pre-computed elements.

### 3.2.3 Fractional wMOF

In sub-section 3.1.3 we have explained what is fractional wNAF and how to use fractional wNAF, In this chapter we will introduce its Left-to-Right analogue.

Fractional wMOF has the same attributes as fractional wNAF. We suppose  $q$  is the maximum memory size, and  $w_0$  is the maximum value which we can choose with  $2^{w_0-2} \leq q \leq 2^{w_0-1}$ , so we get also  $r = q - 2^{w_0-2}$  and  $w_1 = r/2^{w_0-2}$ , where  $0 \leq r < 2^{w_0-2}$  and  $0 \leq w_1 < 1$ .

The pre-computation table build the set is also like what we have got in fractional wMOF, it is  $\{1P, 3P, \dots, (2^{w_0-1} - 1)P, (2^{w_0-1} + 1)P, \dots, (2^{w_0-1} - 1 + w_1 2^{w_0-1})P\}$ , and it has  $q = (1 + w_1)2^{w_0-2}$  elements.

Here is  $w = w_0 + w_1$ , actual window size is  $w_0 + 1$  if possible and  $w_0$  else. We must also insert an additional step in the algorithm to confirm, whether the values with window size  $w_0 + 1$  is stored in the table. When it was not pre-computed and stored in the table, we must change  $w_0 + 1$  into  $w_0$  and recompute the value.

Fractional wMOF can be converted from a binary form either Right-to-Left or Left-to-Right. The following algorithm 5 [15] [16] tells us how to do ECC scalar multiplication with fractional wMOF, in this algorithm, recoding stage and evaluation stage are merged together.

We must create a mapping table for fractional wMOF, and confirm the value whether it is in the table. All of the possible non0s values build a set  $T$ , where

$$T = \{\pm 1, \pm 3, \pm 5, \dots, \pm(2^{w_0-1} - 1), \pm(2^{w_0-1} + 1), \dots, \pm(2^{w_0-1} - 1 + w_1 2^{w_0-1})\}$$

Like what we have proved in fractional wNAF, here  $w = 2.*$  is also meaningless. It has the same result as  $w = 2$ . So all of the window size in the

**Algorithm 5** Scalar Multiplication with fractional wMOF

---

*Input:* a Point  $P$ , a scalar  $d = \{d_{n-1}, \dots, d_0\}_2$ , a fractional width  $w = w_0 + w_1$ , where  $w_1 = r/2^{w_0-2}$  for a  $r$  between 0 and  $2^{w_0-2} - 1$

*Output:* the point  $dP$

Pre-computation:  
 $P_1 \leftarrow P; P_2 \leftarrow ECDBL(P_1);$   
**for**  $j = 1$  to  $(1 + w_1)2^{w_0-1} - 1$  **do**  
     $P_{2j-1} \leftarrow ECADD(P_{2j-1}, P_2);$   
**end for**

Recoding and Evaluation:  
 $A \leftarrow O; i \leftarrow n;$   
**while**  $i \geq 0$  **do**  
    **if**  $d_{i-1} = d_i$  **then**  
         $A \leftarrow ECDBL(A); i \leftarrow i - 1;$   
    **else**  
         $s \leftarrow \max(i - w_0, 0);$   
        **if**  $(d_{s-1} \neq d_s$  and  
             $|\{d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{s-1} - d_s\}_2| \geq 2^{w_0-1(1+w_1)})$  **then**  
             $s = s + 1;$   
        **end if**  
        Let  $t$  be the smallest integer such that  $t \geq s$  and  $d_{t-1} \neq d_t;$   
        **for**  $k = 1$  to  $i - t + 1$  **do**  
             $A \leftarrow ECDBL(A);$   
        **end for**  
        **if**  $[d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{t-1} - d_t]_2 > 0$  **then**  
             $A \leftarrow ECADD(A, P_{\{d_{i-1}-d_i, d_{i-2}-d_{i-1}, \dots, d_{t-1}-d_t\}_2});$   
        **else**  
             $A \leftarrow ECADD(A, -P_{\{d_{i-1}-d_i, d_{i-2}-d_{i-1}, \dots, d_{t-1}-d_t\}_2});$   
        **end if**  
        **for**  $k = 1$  to  $t - s$  **do**  
             $A \leftarrow ECDBL(A);$   
        **end for**  
         $s = s - 1;$   
    **end if**  
**end while**  
**Return**  $A;$

---

next section, which we used in the tests and proofs for fractional wMOF, must also be satisfied with the condition  $w_0 > 2$ .

### 3.3 Optimal Window Size

We have introduced wNAF and wMOF in 3.1 and 3.2. When the window size increases, time of the evaluation will decrease, while time of the pre-computation will increase [17]. So we must find out the most efficiency proper window size, where the length of input binary form is 1024-bits or 160-bits. Figure 2 and Figure 3 illustrate the relation among the window size  $w$ , the speed of the evaluation and pre-computed processes.

The blue curve in Figure 2 and Figure 3 illustrate when the window size  $w$  grows, the total compute time will firstly decrease, then increase again. It means a too big  $w$  is useless. In Figure 2 the optimal  $w$  is circa 6.2 when the input is 1024-bits. And in Figure 3, the optimal  $w$  is circa 4.5 when the inputs is 160-bits. So all the tests in this paper will be processed from  $w = 2$  to  $w = 8$ .

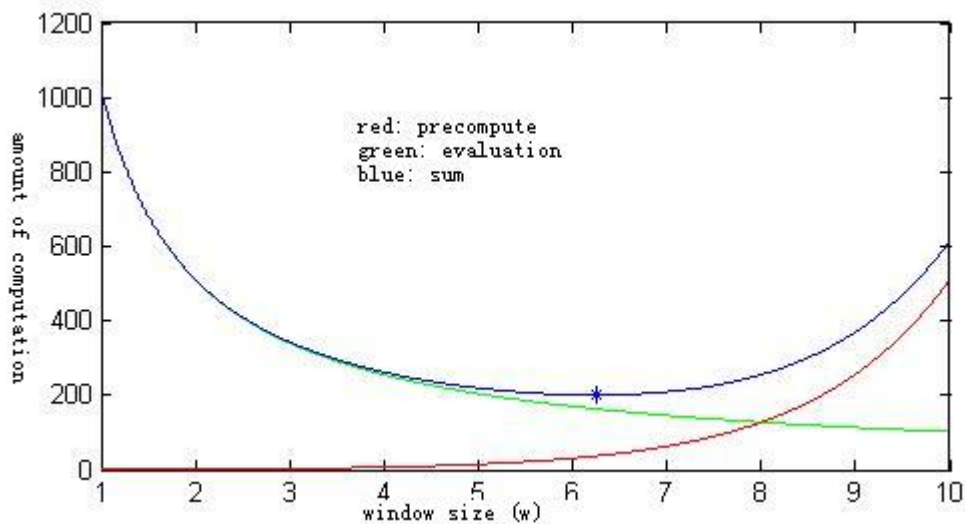


Abbildung 2: Pre-compute and evaluation with 1024-bits input

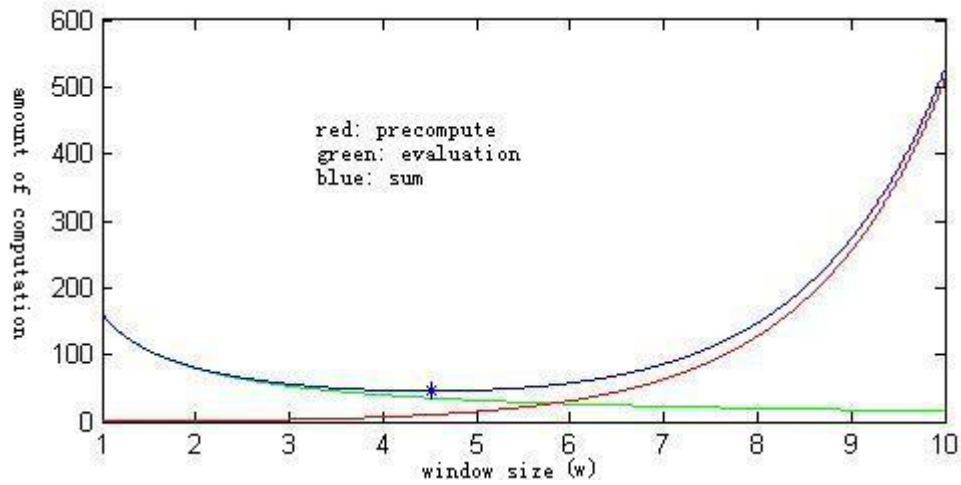


Abbildung 3: Pre-compute and evaluation with 160-bits input

### 3.4 Conclusion

We have introduced NAF, MOF, wNAF, wMOF, fractional wNAF, fractional wMOF and window method in this chapter. Now we finish this chapter with an example. It is clear that 2-NAF is NAF, but 2-MOF isn't MOF.

<i>Binary</i>	10001110100101010111000101011010
<i>MOF</i>	1 $\bar{1}$ 00100 $\bar{1}$ 1 $\bar{1}$ 01 $\bar{1}$ 1 $\bar{1}$ 1 $\bar{1}$ 100 $\bar{1}$ 001 $\bar{1}$ 1 $\bar{1}$ 10 $\bar{1}$ 1 $\bar{1}$ 0
<i>2MOF</i>	01001000 $\bar{1}$ 10010101100 $\bar{1}$ 000101100 $\bar{1}$ 10
<i>3MOF</i>	010010000 $\bar{3}$ 000300 $\bar{3}$ 100 $\bar{1}$ 000030 $\bar{1}$ 000 $\bar{3}$ 0
<i>4MOF</i>	010000070005000 $\bar{5}$ 0 $\bar{1}$ 000007000 $\bar{5}$ 000 $\bar{3}$ 0
<i>NAF</i>	0100100 $\bar{1}$ 01010 $\bar{1}$ 0 $\bar{1}$ 0 $\bar{1}$ 00 $\bar{1}$ 0010 $\bar{1}$ 0 $\bar{1}$ 0 $\bar{1}$ 010
<i>3NAF</i>	01001000 $\bar{1}$ 00 $\bar{3}$ 00 $\bar{3}$ 00300 $\bar{1}$ 0001003000 $\bar{3}$ 0
<i>4NAF</i>	100070000 $\bar{3}$ 0003000 $\bar{5}$ 000007000 $\bar{5}$ 000 $\bar{3}$ 0

## 4 Attributes of wNAF and wMOF

As the most important part, here in this chapter we will analyze the three attributes of wNAF and wMOF that were listed in chapter 1. We will use some mathematics theories to compute the amount of 0s and non-0s, the average length of consecutive 0s and the average bit-length of wNAF and wMOF. Moreover, we will also test and verify all the results, which we have got from the theoretical asymptotic proof. All the test data can be found in chapter 6.

### 4.1 Amount of 0s and non-0s in wNAF and wMOF

#### 4.1.1 Amount of 0s and non-0s in wNAF

**Theorem 2** *The amount of non-0s and 0s in wNAF are  $NZ_{wNAF}(w, k) = \frac{k}{1+w}$  and  $Z_{wNAF}(w, k) = \frac{k w}{1+w}$ , where  $k$  is the bit-length of the binary form and  $w$  is the window size of wNAF.*

**Proof:**

By getting the non-zero density of the wNAF or wMOF, we must use some facts of Markov Chain theory (see Appendix: Markov Chain). Firstly we suppose the length of input binary is  $k$  and analyze the output of wNAF, while the input is an infinitely long sequence of binary form, the output is also an infinitely long sequence and constituted from two types of blocks:

1.  $b_1 = (0)$ , length of this block is 1;
2.  $b_2 = (0 \dots 0^*)$ , length of this block is  $w$ ;

Now we try to determine the conditional probabilities  $P(X_{n+1} = b_1 | X_n)$ :

1. If the current state is  $X_n = b_1$ , there are two sub-cases:
  - The current bit is 0 and it generated no carry at the preceding step. We suppose the probability of this case is  $P$ . The next block

will be 0 iff the next bit is 0, which happens with the probability  $1/2$ .

- The current bit is 1 and there was a carry at the preceding step. So the probability of this case is  $(1 - P)$ . The next block will be  $b_1$  iff the next bit is 1, which happens also with the probability  $1/2$ .

So the probability of these two sub-cases is  $P * 1/2 + (1 - P) * 1/2 = 1/2$ .

2. If the current state is  $X_n = b_2$ , there are also two sub-cases:

- The current  $b_2$  generates no carry. We suppose the probability of this case is  $P$ . The next block will be  $b_1$  if and only if the next bit is 0, which happens with the probability  $1/2$ .
- The current  $b_2$  generates a carry. So the probability of this case is  $(1 - P)$ . The next block will be  $b_1$  if and only if the next bit is 1, which happens also with the probability  $1/2$ .

So the probability of the two sub-cases is also  $P * 1/2 + (1 - P) * 1/2 = 1/2$ .

Now we can determine the probability of form  $P(X_{n+1} = b_2 | X_n)$ :

1. If the current state is  $X_n = b_1$ , the probability of  $P(X_{n+1} = b_2 | b_1)$  is  $P(X_{n+1} \neq b_1 | b_1)$ , it is  $1 - P(X_{n+1} = b_1 | b_1) = 1/2$ .
2. If the current state is  $X_n = b_2$ , the probability of  $P(X_{n+1} = b_2 | b_2)$  is  $P(X_{n+1} \neq b_1 | b_2)$ , it is  $1 - P(X_{n+1} = b_1 | b_2) = 1/2$ .

And now we can get the transition matrix of Markov Chain with random process, it is a  $2 \times 2$  matrix in table 2, The Markov Chain is irreducible and

$P$	$b_1$	$b_2$
$b_1$	$\frac{1}{2}$	$\frac{1}{2}$
$b_2$	$\frac{1}{2}$	$\frac{1}{2}$

Tabelle 2: Markov Chain transition matrix of wNAF

aperiodic, and we can compute its stationary distribution [18]. And it is

$$\pi = (\pi(b_1), \pi(b_2)) = \left(\frac{1}{2}, \frac{1}{2}\right)$$

Because every block  $b_2$  has a non-zero bit, so the non-zero density of wNAF is  $\frac{\pi(b_2)}{\pi b_1 l(b_1) + \pi b_2 l(b_2)}$ , where  $l(x)$  means the length of the block  $x$ . Finally we get the non-zero density of wNAF is  $\frac{1/2}{1/2 * 1 + 1/2 * w} = \frac{1}{1+w}$ .

**q.e.d.**

#### 4.1.2 Amount of 0s and non-0s in wMOF

We can proof the following theorem:

**Theorem 3** *The amount of non-0s and 0s in wMOF are  $NZ_{wMOF}(w, k) = \frac{k}{1+w}$  and  $Z_{wMOF}(w, k) = \frac{k w}{1+w}$ , where  $k$  is the bit-length of the binary form and  $w$  is the window size of wMOF.*

**Proof:**

The input is an infinitely long sequence binary form and the output is also an infinitely long sequence constituted from two types of blocks:

1.  $b_1 = (0)$ , length of this block is 1;
2.  $b_2 = (0^i * 0^{w-i-1})$ , length of this block is  $w$  and  $0 \leq i \leq w - 1$ ;

We also try to determine the conditional probabilities of  $P(X_{n+1} = b_1 | X_n)$

1. If the current state is  $X_n = b_1$ , we suppose the position of current bit is  $i$ , so it means the current bit must be same with the next bit, and the next bit must be same with the next following bit. It means  $P(X_{n+1} = b_1 | X_n = b_1) = P(d_{i-2} = d_{i-1} | d_{i-1} = d_i) = P(d_{i-2} = d_{i-1}) = 1/2$ .
2. If the current state is  $X_n = b_2$ , we also suppose the position of current bit is  $i$ , so it means the bit  $d_{i-w-1} = d_{i-w-2}$ . Hence,  $P(X_{n+1} = b_1 | X_n = b_2) = P(d_{i-w-2} = d_{i-w-1} | d_{i-1} \neq d_i) = P(d_{i-w-2} = d_{i-w-1}) = 1/2$

So we can also determine the conditional probabilities of  $P(X_{n+1} = b_2 | X_n)$

1. The probability of  $P(X_{n+1} = b_2 | b_1)$  is  $P(X_{n+1} \neq b_1 | b_1)$ , it is  $1/2$ .

2. The probability of  $P(X_{n+1} = b_2 | b_2)$  is  $P(X_{n+1} \neq b_1 | b_2)$ , it is  $1/2$ .

The transition matrix of Markov Chain for wMOF is also  $2 \times 2$  matrix in table 3, We can get the same stationary distribution as wNAF, it is

$P$	$b_1$	$b_2$
$b_1$	$\frac{1}{2}$	$\frac{1}{2}$
$b_2$	$\frac{1}{2}$	$\frac{1}{2}$

Tabelle 3: Markov Chain transition matrix of wMOF

$$\pi = (\pi(b_1), \pi(b_2)) = \left( \frac{1}{2}, \frac{1}{2} \right)$$

And the non-zero density of wMOF is same as wNAF. It is  $\frac{\pi(b_2)}{\pi(b_1)l(b_1) + \pi(b_2)l(b_2)}$ , where  $l(x)$  means the length of the block  $x$ . Finally we get the non-zero density of wMOF  $\frac{1}{1+w}$ .

**q.e.d.**

### 4.1.3 Amount of 0s and non-0s in fractional wNAF

Now we analyze fractional wNAF interestingly, the following theorem is perfectly analog to the integer windth case.

**Theorem 4** *The amount of non-0s and 0s in factional wNAF are  $NZ_{fwNAF}(w, k) = \frac{k}{1+w}$  and  $Z_{fwNAF}(w, k) = \frac{k w}{1+w}$ , where  $k$  is the bit-length of binary form and  $w$  is the window size of fractional wNAF.*

**Proof:**

Different from wNAF, fractional wNAF has three types of blocks. They are:

1.  $b_1 = (0)$ , length of this block is 1;
2.  $b_2 = (0 \dots 0^*)$ , length of this block is  $w_0$ ;
3.  $b_3 = (0 \dots 0^*)$ , length of this block is  $w_0 + 1$ ;

In order to get the transition Matrix of fraction wNAF, we try to determine the conditional probabilities  $P(X_{n+1} = b_1 | X_n)$

1. If the current state is  $X_n = b_1$ , there are two sub-cases:
  - The current bit is 0 and there was not any carry at the preceding step. We suppose the probability of this case is  $P$ . The next block will be  $b_1$  iff the next bit is 0, which happens with the probability  $1/2$ .
  - The current bit is 1 and there was not any carry at the preceding step. We suppose the probability of this case is  $1 - P$ . The next block will be  $b_1$  iff the next bit is 1, which happens with the probability  $1/2$ .

The probability is the sum of these two sub-cases. It is  $P * 1/2 + (1 - P) * 1/2 = 1/2$ .

2. If the current state is  $X_n = b_2$ , therefore we are in the case when  $|\sigma_{w_0+1}[i]| > ((1 + w_1)2^{w_0-1} - 1)$ . It means the current window size is  $w_0$  and there are also two sub-cases:

- The current step doesn't generate any carry, and the next bit is 0. So in this case means we increase the window size from  $w_0$  to  $w_0 + 1$  with a extra bit '0'. And the new block with window size  $w_0 + 1$  satisfies the condition, which is  $|\sigma_{w_0+1}[i]| < ((1 + w_1)2^{w_0-1} - 1)$ , but we have already supposed that  $|\sigma_{w_0+1}[i]| > ((1 + w_1)2^{w_0-1} - 1)$ , They are self-contradict, and the probability of this sub-case is 0.
- The current step generates a carry and the next bit is 1. we have this form  $\underbrace{11 * \dots * 1}_{w_0+1}$ . We know that the block  $b_2$  generates a carry,

so in Algorithm 4,  $k$  must match the condition  $k \bmod 2^{w_0} = (k \bmod 2^{w_0}) - 2^{w_0}$ , the most bit is also 1, so it also match the condition  $k \bmod 2^{w_0+1} = (k \bmod 2^{w_0+1}) - 2^{w_0+1}$  and the condition  $k \bmod 2^{w_0+1} = (k \bmod 2^{w_0+1}) + 2^{w_0}$ . We reduce these conditions, and we can get the equation:

$$\begin{aligned}
 & k \bmod 2^{w_0+1} \\
 &= (k \bmod 2^{w_0+1}) - 2^{w_0+1} \\
 &= (k \bmod 2^{w_0}) + 2^{w_0} - 2^{w_0+1} \\
 &= (k \bmod 2^{w_0}) + 2^{w_0} \\
 &= k \bmod 2^{w_0}
 \end{aligned}$$

Because  $k \bmod 2^{w_0} \leq 2^{w_0-1}$ , so  $k \bmod 2^{w_0+1} \leq 2^{w_0-1}$ , it conflicts with the precondition  $|\sigma_{w_0+1}[i]| > ((1 + w_1)2^{w_0-1} - 1)$ . And the probability of this sub-case is also 0.

So the probability of  $P(X_{n+1} = b_1 | X_n = b_2)$  is 0.

3. If the current state is  $X_n = b_3$ , there are also two sub-cases:

- The current  $b_3$  generates no carry. We suppose the probability of this case is  $P$ . The next block will be  $b_1$  iff the next bit is 0, which happens with the probability  $1/2$ .
- The current  $b_3$  generates a carry. So the probability of this case is  $1 - P$ . The next block will be  $b_1$  if and only if the next bit is 1, which happens also with the probability  $1/2$ .

So the probability of these two sub-cases is also  $P * 1/2 + (1 - P) * 1/2 = 1/2$ .

Now we try to determine the probabilities of the other cases, they are  $P(X_{n+1} = b_2 | X_n)$  and  $P(X_{n+1} = b_3 | X_n)$ . When  $X_{n+1} \neq b_1$ , it means the probability of  $1 - P(X_{n+1} = b_1 | X_n)$  is the sum of the probabilities of  $P(X_{n+1} = b_2 | X_n)$  and  $P(X_{n+1} = b_3 | X_n)$ . Because the current bit must be 1, so we must check whether  $|\sigma_{w_0+1}[i]| > ((1 + w_1)2^{w_0-1} - 1)$ , so the  $b_3$  corresponds to  $\frac{(1+w_1)2^{w_0-1}}{2^{w_0}} = \frac{1+w_1}{2}$  of the cases and  $b_2$  corresponds to the remaining cases  $\frac{1-w_1}{2}$ . finally we get  $(X_{n+1} = b_2 | X_n) = (1 - P(X_{n+1} = b_1 | X_n)) \frac{1+w_1}{2}$  and  $(X_{n+1} = b_3 | X_n) = (1 - P(X_{n+1} = b_1 | X_n)) \frac{1-w_1}{2}$ . The Markov chain with the transition matrix is in table 4, The Markov Chain is also

$P$	$b_1$	$b_2$	$b_3$
$b_1$	$\frac{1}{2}$	$\frac{1-w_1}{4}$	$\frac{1+w_1}{4}$
$b_2$	0	$\frac{1-w_1}{2}$	$\frac{1+w_1}{2}$
$b_3$	$\frac{1}{2}$	$\frac{1-w_1}{4}$	$\frac{1+w_1}{4}$

Tabelle 4: Markov Chain transition matrix of fractional wNAF

irreducible and aperiodic, and we can compute its stationary distribution  $\pi = (\pi(b_1), \pi(b_2), \pi(b_3)) = (\frac{1+w_1}{3+w_1}, \frac{1-w_1}{3+w_1}, \frac{1+w_1}{3+w_1})$ .

Each block  $b_2$  and  $b_3$  has a non-0 bit, so the non-zero density of fractional wNAF is  $\frac{\pi(b_2) + \pi(b_3)}{\pi(b_1)l(b_1) + \pi(b_2)l(b_2) + \pi(b_3)l(b_3)}$ , and the value is:

$$\frac{\frac{1-w_1}{3+w_1} + \frac{1+w_1}{3+w_1}}{\frac{1+w_1}{3+w_1} * 1 + \frac{1-w_1}{3+w_1} * w_0 + \frac{1+w_1}{3+w_1} * (w_0 + 1)} = \frac{1}{1 + w}$$

where  $l(x)$  means the length of block  $x$ .

**q.e.d.**

## 4.1.4 Amount of 0s and non-0s in fractional wMOF

Again, the following theorem is perfectly analog to the integer width case.

**Theorem 5** *The amount of non-0s and 0s in fractional wMOF are  $NZ_{fwMOF}(w, k) = \frac{k}{1+w}$  and  $Z_{fwMOF}(w, k) = \frac{k w}{1+w}$ , where  $k$  is the bit-length of binary form and  $w$  is the window size of fractional wMOF.*

**Proof:**

The input is also an infinitely long sequence of binary form and the output is an infinitely long sequence constituted from three types of blocks.

1.  $b_1 = (0)$ , length of this block is 1;
2.  $b_2 = (0^i * 0^{w_0-i-1})$ , length of this block is  $w_0$ ;
3.  $b_3 = (0^i * 0^{w_0-i})$ , length of this block is  $w_0 + 1$ ;

The conditional probabilities of form  $P(X_{n+1} = b_1 | X_n)$

1. If the current state is  $X_n = b_1$ , we suppose the position of current bit is  $i$ , so it means the current bit must be same as the next bit, and the next bit must be same with the next following bit. Hence,  $P(X_{n+1} = b_1 | X_n = b_1) = P(d_{i-2} = d_{i-1} | d_{i-1} = d_i) = P(d_{i-2} = d_{i-1}) = 1/2$ .
2. If the current state is  $X_n = b_2$ , we also suppose the position of current bit is  $i$  and  $s = i - w_0$ , from Algorithm 5 we know  $d_{s-1} \neq d_s$ . But for  $X_{n+1} = b_1$  we have  $d_{s-1} = d_s$ , so the probability of this case is 0.
3. If the current state is  $X_n = b_3$ , we also suppose the position of current bit is  $i$  and  $s = i - w_0$ , from algorithm 5 we know  $d_{s-1} = d_s$ . Hence,  $P(X_{n+1} = b_1 | X_n = b_3) = P(d_{s-2} = d_{s-1}) = 1/2$ .

We try to determine the probabilities  $P(X_{n+1} = b_2 | X_n)$  and  $P(X_{n+1} = b_3 | X_n)$ . If the new state is not  $b_1$ ,  $w_0 + 2$  input bits are processed, and the first two of these are differ. Therefore, there are  $2^{w_0+1}$  possibilities for these input bit. Because of the properties of MOF, for every odd integer  $1 \leq x \leq 2^{w_0+1}$  there are 4 choices of  $d_i \neq d_{i-1}, d_{i-2}, \dots, d_s, d_{s-1} \in \{0, 1\}$  that  $|[d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{s-1} - d_s]_2| = x$ . Then there are still  $2^{w_0-1}$  possibilities. It is simply to compute the probability when  $X_n = b_3$ , it is  $\frac{2^{w_0-2}(1+w_1)}{2^{w_0-1}} = \frac{1+w_1}{2}$  and when  $X_{n+1} = b_2$ , the probability is  $\frac{1-w_1}{2}$ .

So we can get the Markov Chain transition matrix of fractional wMOF in

$P$	$b_1$	$b_2$	$b_3$
$b_1$	$\frac{1}{2}$	$\frac{1-w_1}{4}$	$\frac{1+w_1}{4}$
$b_2$	0	$\frac{1-w_1}{2}$	$\frac{1+w_1}{2}$
$b_3$	$\frac{1}{2}$	$\frac{1-w_1}{4}$	$\frac{1+w_1}{4}$

Tabelle 5: Markov Chain transition matrix of fractional wMOF

table 5, and we can also get the same result as 4.1.3. The Markov Chain stationary distribution is  $\pi = (\pi(b_1), \pi(b_2), \pi(b_3)) = (\frac{1+w_1}{3+w_1}, \frac{1-w_1}{3+w_1}, \frac{1+w_1}{3+w_1})$ .

The non-zero density of fractional wMOF is  $\frac{\pi(b_2)+\pi(b_3)}{\pi(b_1)l(b_1)+\pi(b_2)l(b_2)+\pi(b_3)l(b_3)} = \frac{1}{1+w}$ , where  $l(x)$  means the length of block  $x$ .

**q.e.d.**

#### 4.1.5 Conclusion and Test

Now we get the result that wNAF, wMOF, fractional wNAF and fractional wMOF have the same asymptotic non-zero density and zero density. When the input is an infinitely long binary form sequence, after the conversion, the output will be also an infinitely long sequence, in which the probabilities of non-zero density and zero density are  $\frac{1}{1+w}$  and  $\frac{w}{1+w}$ , the amount of non-0s and 0s are almost  $\frac{k}{1+w}$  and  $\frac{k w}{1+w}$ , where  $k$  is the length of input.

We gather the test result together in Table 6. In this table 'A' means academic value, 'P' means practical value, '160' and '16000' mean the length of input, 'b' means binary form input, 'wN' means wNAF, 'wM' means wMOF, 'fwN' means fractional wNAF and 'fwM' means fractional wMOF. All the following tables have the same meanings like table 6.

We conclude from the results in Table 6, that the heuristic data approximates the expected academic values quite accurate.

$w$	2(.5)	3(.5)	4(.5)	5(.5)	6(.5)	7(.5)	8(.5)
$A160b(wN)$	53.333	40.000	32.000	26.667	22.857	20.000	17.778
$P160b(wN)$	53.768	40.467	32.426	27.114	23.331	20.514	18.275
$A16000b(wN)$	5333.333	4000.000	3200.000	2666.667	2666.667	2000.000	1777.778
$P16000b(wN)$	5333.529	3999.783	3200.236	2667.045	2285.884	2000.546	1778.090
$A160b(wM)$	53.333	40.000	32.000	26.667	22.857	20.000	17.778
$P160b(wM)$	53.758	40.499	32.428	27.117	23.331	20.513	18.276
$A16000b(wM)$	5333.333	4000.000	3200.000	2666.667	2666.667	2000.000	1777.778
$P16000b(wM)$	5333.769	3999.186	3200.880	2667.198	2286.216	2000.726	1778.230
$A160b(fwN)$	X	35.556	29.091	24.615	21.333	18.824	X
$P160b(fwN)$	X	36.053	29.462	25.081	21.824	19.279	X
$A16000b(fwN)$	X	3555.556	2909.091	2461.538	2133.333	1882.353	X
$P16000b(fwN)$	X	3555.702	2909.586	2462.454	2133.664	1883.028	X
$A160b(fwM)$	X	35.556	29.091	24.615	21.333	18.824	X
$P160b(fwM)$	X	36.054	29.461	25.079	21.824	19.276	X
$A16000b(fwM)$	X	3555.556	2909.091	2461.538	2133.333	1882.353	X
$P16000b(fwM)$	X	3555.775	2909.441	2462.102	2133.711	1883.928	X

Tabelle 6: Test Data: Amount of 0s and non-0s

## 4.2 Average length of consecutive 0s in wNAF and wMOF

### 4.2.1 Average length of consecutive 0s in wNAF

**Theorem 6** *The average length of consecutive 0s in wNAF is  $CZ_{wNAF}(w, k) = w$ , where  $k$  is the bit-length of binary form and  $w$  is the window size of wNAF.*

**Proof:**

In sub-section 4.1.1 we have explained the output of wNAF. There are two types of blocks:

1.  $b_1 = (0)$ , length of this block is 1;
2.  $b_2 = (0 \dots 0^*)$ , length of this block is  $w$ ;

A carry occurs if and only if the preceding output was the block  $b_2$  and the corresponding value  $*$  is negative. We suppose the probability of the

preceding step which generates a carry is  $P(c)$ , then the probability of the preceding step which generates no carry is  $1 - P(c)$ . The probability of the current bit 0 or 1 are  $1/2$ , so the probability of  $b_1$  is  $P(0) = \frac{1}{2}(P(c)) + \frac{1}{2}(1 - P(c)) = \frac{1}{2}$ .

We have also proved in sub-section 4.1.1 that the probabilities of  $P(b_1 | b_1), P(b_2 | b_1), P(b_1 | b_2)$  and  $P(b_2 | b_2)$  are  $1/2$ . In order to compute the average of the consecutive 0s in wNAF, we can analyze the probability of several forms and here we suppose  $k$  is the length of consecutive 0s.

1. When  $k < w - 1$ , it is impossible, because we have only two types of blocks. It means the minimal length of the consecutive 0s is  $w - 1$ .
2. When  $k = w - 1$ , it match the block  $b_2$ , the probability of this case is when  $b_2$  is behind also such a block  $b_2$ . It means the probability is  $P(b_2 | b_2)$ . It is  $1/2$ .
3. When  $k = w$ , it means we have such a form  $(b_2 \circ b_1 \circ b_2)$ , and  $P(b_2 \circ b_1 \circ b_2) = P(b_2 | b_1) * P(b_1 | b_2)$ , it is  $1/2 * 1/2 = 1/4$ , where 'o' means concatenation of block.
- .....
- j. When  $k = w + i$ , we can compute the probability with  $P(b_2 \circ b_1 \circ \dots \circ b_2) = P(b_2 | b_1) * P(b_1 | b_1) * \dots * P(b_1 | b_2) = \frac{1}{2^{i+2}}$ .
- .....

Now we can compute the sum of these probabilities and get the average length of consecutive 0s:

$$L = \sum_{k=w-1}^K L(k)P(k) = \sum_{k=w-1}^K k * \frac{1}{2^{k-w+2}}$$

where  $k$  is the length of the input. When the input is an infinitely long sequence, we can suppose  $K \rightarrow \infty$  and  $K \gg w$ , it can be reduced to

$$L = \lim_{K \rightarrow \infty} \left( \sum_{k=w-1}^K k * \frac{1}{2^{k-w+2}} \right) = w$$

**q.e.d.**

4.2.2 Average length  $k$  of consecutive 0s in wMOF

**Theorem 7** *The average length of consecutive 0s in wMOF is  $CZ_{wMOF}(w, k) = w$ , where  $k$  is the bit-length of binary form and  $w$  is the window size of wMOF, and we regard two consecutive non-zero bits ( occurring in the pattern  $0..0 * | * 0..0$  ) as a special block of 0s of block-length 0.*

**Proof:**

In sub-section 4.1.2 we have introduced the two types of blocks in wMOF:

1.  $b_1 = (0)$ , length of this block is 1;
2.  $b_2 = (0^i * 0^{w-i-1})$ , length of this block is  $w$  and  $0 \leq i \leq w - 1$ ;

Because we can't ensure the position of the non-0 bit in  $b_2$ , so we define two cases in advance:

1. The non-zero bit is at the last position of  $b_2$
2. The non-zero bit is at a random position in  $b_2$

And we must prove that this two cases have the same result when we use them to compute the average length of consecutive 0s in wMOF.

Because the non-zero bit split the 0s, so the average length of consecutive 0s can be express in the way that:

$$l = \frac{\text{amount of 0s}}{\text{amount of non - 0s}}$$

When we move the non-zero bit in  $b_2$  from one position to another, the amount of 0s will not change, and the amount of the non-0s will also not change. So the average length of consecutive 0s in this two case are equivalent. Here we don't analyze the special case of the bit-string  $(0 \dots 0X | X0 \dots 0)$ , it reduces one block of consecutive 0s, but in fact we can also regard it as a special block of consecutive 0s with the length 0. In the following proof, we use the form of the first case.

As the transition matrices of wMOF and wNAF are the same, and we have seen that the position of the non-zero bit is not important, the rest of the proof can be done exactly as the proof of theorem 6. therefore we conclude that the average length of consecutive 0s in wMOF is  $L = w$ .

**q.e.d.**

4.2.3 Average length of consecutive 0s in fractional wNAF

**Theorem 8** *The average length of consecutive 0s in fractional wNAF is  $CZ_{fwNAF}(w_0 + w_1, k) = w_0 + w_1 = w$ , where  $k$  is the bit-length of binary form and  $w$  is the window size of fractional wNAF.*

**Proof:**

In sub-section 4.1.3 we have introduced the three types of blocks in fractional wNAF.

1.  $b_1 = (0)$ , length of this block is 1;
2.  $b_2 = (0 \dots 0^*)$ , length of this block is  $w_0$ ;
3.  $b_3 = (0 \dots 0^*)$ , length of this block is  $w_0 + 1$ ;

It is a little different from wNAF, because in wNAF it is only required to analyze the block  $b_2$  and compute the consecutive 0s, but in fractional wNAF, we must first confirm whether the block is  $b_2$  or  $b_3$  when we get non- $b_1$  block, then we can use this block to compute the consecutive 0s. We have also deduced the distribution of these two cases,  $\frac{1-w_1}{2}$  for  $b_2$  and  $\frac{1+w_1}{2}$  for  $b_3$ . In addition, we have got the probabilities between every two blocks and built a Markov Chain matrix in subsection 4.1.3, Now we can analyze the length of consecutive 0s in fractional wNAF with these forms and here we suppose  $k$  is the length of consecutive 0s.

1. When  $k < w_0 - 1$ , it will never appear.
2. When  $k = w_0 - 1$ , this case occurs only when the block isn't behind  $b_1$  and itself is a block  $b_2$ . So the probability is  $(1 - 0) * \frac{1-w_1}{2}$ .
3. When  $k = w_0$ , the probability is only when the block isn't behind  $b_1$  and itself is a block of  $b_3$ . The probability is  $(1 - \frac{1}{2}) * \frac{1+w_1}{2} = \frac{1+w_1}{4}$ .
4. When  $k = w_0 + 1$ , when the block is behind  $b_1$  and itself is a block of  $b_3$  and the block  $b_1$  isn't behind another  $b_1$ . The probability is  $\frac{1}{2} * (1 - \frac{1}{2}) * \frac{1+w_1}{2} = \frac{1+w_1}{8}$ .
- .....
- $j$ . When  $k = w_0 + i$ , The probability is  $\frac{1+w_1}{2^{i+2}}$ .
- .....

Now we can compute the sum of these probabilities

$$L = \sum_{k=w_0-1}^K L(k)P(k) = \frac{1-w_1}{2} * (w_0 - 1) + \sum_{k=w_0}^K k * \frac{1+w_1}{2} * \frac{1}{2^{k-w_0+2}}$$

where  $k$  is the length of the input and when the input is an infinitely long sequence, we can suppose  $K \rightarrow \infty$  and  $K \gg w$ , it can be reduced to

$$L = w_0 + w_1 = w$$

**q.e.d.**

#### 4.2.4 Average length k of consecutive 0s in fractional wMOF

**Theorem 9** *The average length of consecutive 0s in fractional wMOF is  $CZ_{fwMOF}(w_0 + w_1, k) = w_0 + w_1 = w$ . where  $k$  is the bit-length of binary form and  $w$  is the window size of fractional wMOF, and we regard two consecutive non-zero bits ( occurring in the pattern  $0..0 * | * 0..0$  ) as a special block of 0s of block-length 0.*

**Proof:**

In sub-section 4.1.4 we have defined the three types of blocks in fractional wMOF:

1.  $b_1 = (0)$ , length of this block is 1;
2.  $b_2 = (0^i * 0^{w_0-i-1})$ , length of this block is  $w_0$ ;
3.  $b_3 = (0^i * 0^{w_0-i})$ , length of this block is  $w_0 + 1$ ;

In sub-section 4.2.2 we have proved, the average length of consecutive 0s doesn't depend on the position of the non-zero bit. So the proof is same as the proof of theorem 8, because fractional wNAF and fractional wMOF have the same Markov Chain transition matrix, so when  $K$  is the length of the input and when the input is an infinitely long sequence, it means  $K \rightarrow \infty$  and  $k \gg w$ , finally, we can get the average length of consecutive 0s  $L = w_0 + w_1 = w$ .

**q.e.d.**

$w$	2(.5)	3(.5)	4(.5)	5(.5)	6(.5)	7(.5)	8(.5)
$A160b(wN)$	2.000	3.000	4.000	5.000	6.000	7.000	8.000
$P160b(wN)$	1.988	2.963	3.947	4.913	5.873	6.812	7.770
$A16000b(wN)$	2.000	3.000	4.000	5.000	6.000	7.000	8.000
$P16000b(wN)$	2.000	3.000	4.000	5.000	6.000	6.998	7.999
$A160b(wM)$	2.000	3.000	4.000	5.000	6.000	7.000	8.000
$P160b(wM)$	1.988	2.963	3.946	4.913	5.874	6.812	7.769
$A16000b(wM)$	2.000	3.000	4.000	5.000	6.000	7.000	8.000
$P16000b(wM)$	2.000	3.000	3.999	4.999	5.999	6.998	7.998
$A160b(fwN)$	X	3.500	4.500	5.500	6.500	7.500	X
$P160b(fwN)$	X	3.451	4.444	5.394	6.347	7.313	X
$A16000b(fwN)$	X	3.500	4.500	5.500	6.500	7.500	X
$P16000b(fwN)$	X	3.500	4.499	5.498	6.499	7.497	X
$A160b(fwM)$	X	3.500	4.500	5.500	6.500	7.500	X
$P160b(fwM)$	X	3.451	4.444	5.395	6.347	7.315	X
$A16000b(fwM)$	X	3.500	4.500	5.500	6.500	7.500	X
$P16000b(fwM)$	X	3.500	4.497	5.499	6.500	7.498	X

Tabelle 7: Test Data: Consecutive 0s

#### 4.2.5 Conclusion and Test

This chapter concludes that the average length of consecutive 0s in wNAF, wMOF, fractional wNAF and fractional wMOF are all  $w$ . But we must ensure that  $w \ll k$ . For a small  $w$ , we also have short consecutive 0s, thus the evaluation step of the scalar multiplication becomes more involved. But on the other side, we need less memory resources to save the results of pre-computation. When we choose a big  $w$ , we can get a long consecutive 0s, which speeds up the evaluation step but more memory resources to save the results of pre-computation.

We gather the test data together in Table 7. The amount of consecutive 0s are nearly  $w$ . We can also observe the result that when  $k \gg w$  and when the  $w$  increases, then the data becomes more and more precise.

### 4.3 Average bit-length of wNAF or wMOF

#### 4.3.1 Average bit-length of wNAF

**Theorem 10** *The average bit-length of wNAF is  $L_{wNAF}(w, k) = k + \frac{1}{1+w}$ , where  $k$  is the length of binary sequence and  $w$  is the window size.*

**Proof:**

In order to speed up the scalar multiplication and estimate the time of computation, we can first estimate the average bit-length of wNAF and wMOF, which was transferred from input. Specially to the scalar multiplication with wNAF, the new form will be first stored, then do the multiplication, so the first ECADD will begin at the first non-0 bit in wNAF. We can computed the non-zero density in the above section, but we don't know which bit is the first non-zero bit in wNAF. When we know where is the first non-zero bit in wNAF, it saves much time. So in this subsection, we try to determine the average bit-length of wNAF and wMOF, and the average bit-length of wNAF and wMOF from first non-0 bit. We denote the latter efficous length. First we give an example, when  $w = 2$ . In order to determine the average length of wNAF, we first suppose there is a input sequence with the length  $n$ . After the transferring with wNAF, we analyze the result at the beginning of the output. We list all the possibilities which the output wNAF looks like:

1.  $\left| \underbrace{0}_{1} \underbrace{*\dots*}_{n-1} \right.$
2.  $1 \left| \underbrace{0}_{1} \underbrace{*\dots*}_{n-1} \right.$
3.  $\left| \underbrace{0*}_{2} \underbrace{*\dots*}_{n-2} \right.$
4.  $1 \left| \underbrace{0*}_{2} \underbrace{*\dots*}_{n-2} \right.$
5.  $\left| \underbrace{*}_{1} \underbrace{*\dots*}_{n-1} \right.$

The first case means the output begins with  $b_1 = (0)$ , the second case means the output begins with  $b_1 = (0)$  and the length of the input increases. The third case means the output begins with a block  $b_2 = (0*)$ , the forth case begins with a block  $b_2 = (0*)$  and the length of the input increases, and the fifth case means the output begins with a part of block  $b_2 = (*)$ . Now we add

a extra bit before the input sequence, and the length of the output turns to  $n + 1$ . Analyze these five cases, when we add a bit '0' before first case, it will be still the type of first case, and when we add a bit '1' before the first case, it will turn to the type of the fifth case. So we suppose the states from case.1 to case.5 are  $Q_1, Q_2, Q_3, Q_4, Q_5$ . When the length of input sequence increases one bit, the Markov chain transition matrix is:

$$\begin{array}{c|cccccc}
 P & Q_1 & Q_2 & Q_3 & Q_4 & Q_5 \\
 \hline
 Q_1 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\
 Q_2 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\
 Q_3 & \frac{1}{2} & 0 & 0 & 0 & \frac{1}{2} \\
 Q_4 & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{2} \\
 Q_5 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0
 \end{array}$$

Because it is irreducible and aperiodic, so we can compute the stationary distribution of this Markov chain, it is:

$$\pi = (\pi(Q_1), \pi(Q_2), \pi(Q_3), \pi(Q_4), \pi(Q_5)) = \left( \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{3} \right)$$

And the probability of the length of the input increases is  $\pi(Q_2) + \pi(Q_4) = \frac{1}{3}$ . Now we use this method to solve the problem, when  $w = W$ .

First we list all the possibility of the output.

1.  $\left| \underbrace{0}_{1} \underbrace{* \dots *}_{n-1} \right.$
2.  $1 \left| \underbrace{0}_{1} \underbrace{* \dots *}_{n-1} \right.$
3.  $\left| \underbrace{0 \dots *}_{W} \underbrace{* \dots *}_{n-W} \right.$
4.  $1 \left| \underbrace{0 \dots *}_{W} \underbrace{* \dots *}_{n-W} \right.$
5.  $\left| \underbrace{0 \dots *}_{W-1} \underbrace{* \dots *}_{n-W+1} \right.$
- .....
- j-1.  $\left| \underbrace{0*}_{2} \underbrace{* \dots *}_{n-2} \right.$

$$j. \quad | \underbrace{*}_{1} \underbrace{* \dots *}_{n-1}$$

From case.1 to case.4 are normal states, and from case.5 to case. $j$  mean the wNAF begins with a breaking block  $b_2$ . When the window size is  $W$ , the amount of all of these possibilities is  $W + 3$ , we suppose the states from case.1 to case. $W + 3$  are denoted by  $Q_1, Q_2, Q_3, Q_4, Q_5, \dots, Q_{W+3}$ . When the length of input sequence increases 1, analyze like the previous Example, we get the following Markov chain transition matrix:

$$\begin{array}{c|cccccccc}
 P & Q_1 & Q_2 & Q_3 & Q_4 & Q_5 & \cdot & \cdot & \cdot & Q_{W+3} \\
 \hline
 Q_1 & \frac{1}{2} & 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & \frac{1}{2} \\
 Q_2 & 0 & \frac{1}{2} & 0 & 0 & 0 & \cdot & \cdot & \cdot & \frac{1}{2} \\
 Q_3 & \frac{1}{2} & 0 & 0 & 0 & 0 & \cdot & \cdot & \cdot & \frac{1}{2} \\
 Q_4 & 0 & \frac{1}{2} & 0 & 0 & 0 & \cdot & \cdot & \cdot & \frac{1}{2} \\
 Q_5 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & \cdot & \cdot & \cdot & 0 \\
 \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot \\
 \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\
 \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & \cdot & \cdot \\
 Q_{W+3} & 0 & 0 & 0 & 0 & 0 & \cdot & \cdot & 1 & 0
 \end{array}$$

And the stationary distribution of this Markov chain is:  $(\pi(Q_1), \dots, \pi(Q_{W+3})) =$

$$\left( \frac{1}{2(W+1)}, \frac{1}{2(W+1)}, \frac{1}{2(W+1)}, \frac{1}{2(W+1)}, \underbrace{\frac{1}{W+1}, \dots, \frac{1}{W+1}}_{W-1} \right)$$

Now we can get the

probability of the length of input increases, it is  $\pi(Q_2) + \pi(Q_4) = \frac{1}{W+1}$ . It means  $GC_{wNAF}(w, k) = \frac{1}{W+1}$ .

**q.e.d.**

**Theorem 11** *The efficacious length of wNAF is  $EL_{wNAF}(w, k) = k - \frac{w(w-1)}{2(w+1)}$ , where  $k$  is the length of binary sequence and  $w$  is the window size.*

**Proof:**

The second step we can compute the average efficacious length of the binary input. When the efficacious length is  $n$ , it has the probability  $\frac{1}{2}$ . When the efficacious length is  $n - 1$ , it has the probability  $\frac{1}{2} * \frac{1}{2} = \frac{1}{4}$  and when the

efficacious length is  $n - k$ , it has the probability  $\frac{1}{2^{k+1}}$ , so we can get the the average efficacious length of the binary input, it is:

$$\frac{1}{2} * n + \frac{1}{2^2} * (n - 1) + \dots + \frac{1}{2^w} * (n - w + 1) + \dots + \frac{1}{2^n} * 1$$

It can be reduced to

$$EL_{binary}(1, n) = \sum_{i=0}^n \frac{1}{2^{i+1}} (n - i) = n - 1$$

When the input is an infinity sequence, it means  $n \rightarrow \infty$ , so the average efficacious length of a binary input is nearly  $n - 1$ , where  $n$  is the length of this input.

Now we can compute the average efficacious length of wNAF form.

1. efficacious length is  $n + 1$ :  $\pi(Q_2) + \pi(Q_4) = \frac{1}{W+1}$
2. efficacious length is  $n$ :  $\pi(Q_{W+3}) = \frac{1}{W+1}$
3. efficacious length is  $n - 1$ :  $\pi(Q_{W+2}) = \frac{1}{W+1}$
- .....
- j. efficacious length is  $n - W + 1$ :  $\pi(Q_3) = \frac{1}{2^{*(W+1)}}$
- k. efficacious length is  $n - W$ :  $\pi(Q_1) * P(b_1 | b_2) = \frac{1}{2^{2*(W+1)}}$
- .....
- l. efficacious length is  $n - W - k$ :  $\pi(Q_1) * P(b_1 | b_1) * \dots * P(b_1 | b_2) = \frac{1}{2^{(k+2)*(W+1)}}$
- .....

Now we compute the sum of these possibilities, it is

$$EL_{wNAF}(w, n) = \sum_0^{n+1} EL * Pr(EL) = n - \frac{w(w - 1)}{2(w + 1)}$$

**q.e.d.**

When  $w = 2, 3, 4, \dots, 8$ , the  $EL_{wNAF}(2, n), EL_{wNAF}(3, n), \dots, EL_{wNAF}(8, n)$  are  $n - \frac{1}{3}, n - \frac{3}{4}, n - \frac{6}{5}, n - \frac{5}{3}, n - \frac{15}{7}, n - \frac{21}{8}, n - \frac{28}{9}$ . It means when we choose a  $w$ , which is bigger than 4, the efficacious length of wNAF is smaller than the length of the binary form.

When the input isn't an infinity sequence, but it is big enough, so that stationary distribution approaches to the ideal value, we can use this stationary distribution to complete the result of 4.1.1. Now we suppose the input length is  $k$ , because we have got the in 4.1.1 that  $NZ_{wNAF}(w, k) = \frac{1}{1+w}$ , So

1. when the form is  $|\underbrace{0}_1 \underbrace{X \dots X}_{k-1}$ , non0s bits are  $(k-1)\frac{1}{1+w}$
2. when the form is  $1|\underbrace{0}_1 \underbrace{X \dots X}_{k-1}$ , non0s bits are  $(k-1)\frac{1}{1+w} + 1$
3. when the form is  $|\underbrace{0 \dots X}_w \underbrace{X \dots X}_{k-w}$ , non0s bits are  $(k-w)\frac{1}{1+w} + 1$
4. when the form is  $1|\underbrace{0 \dots X}_w \underbrace{X \dots X}_{k-w}$ , non0s bits are  $(k-w)\frac{1}{1+w} + 2$
5. when the form is  $|\underbrace{0 \dots X}_{w-1} \underbrace{X \dots X}_{k-w+1}$ , non0s bits are  $(k-w+1)\frac{1}{1+w} + 1$
- .....
- j. when the form is  $|\underbrace{1}_1 \underbrace{X \dots X}_{k-1}$ , non0s bits are  $(k-1)\frac{1}{1+w} + 1$

We have also got the probabilities for every case, so the sum of these possibilities is  $NZ_{wNAF}(w, k)$ , but here  $k \rightarrow \infty$ , it equal:

$$\frac{1}{2(1+w)} \left( \frac{1}{1+w} (4k - 2w - 2) + 4 \right) + \frac{1}{1+w} \left( \frac{1}{1+w} \sum_{i=1}^{w-1} (k-i) + w-1 \right)$$

It can be reduced to

$$NZ_{wNAF}(w, k) = \frac{w^2 + 3w + 2wk + 2k}{2(1+w)^2}$$

So when the input length is 160 and  $w = 8$ , the amount of non0s bits is about 18.321, we can compare it with the data in table 6. When the input length is 160 and  $w = 8$ , the amount of non0s bits in wNAF is about 18.275.

### 4.3.2 Average bit-length of wMOF

**Theorem 12** *The average bit-length of wMOF is  $L_{wMOF}(w, k) = k + \frac{1}{2w}$ , where  $k$  is the length of binary sequence and  $w$  is the window size.*

**Proof:**

From Algorithm 3 we know, that  $\sigma_n$  will be 0 if  $d_{n-1} = 0$ , so we only discuss

the state, when  $d_{n-1} \neq 0$ . So we get the first block from the binary input and compute the signed bit pattern  $(1, d_{n-2} - 1, \dots, d_{n-w} - d_{n-w+1})$ . Then we need to find out the corresponding wMOF in the table, which is matched with this pattern. From the definition we know that the only possibility where a length increment occurs

$$(1, d_{n-2} - 1, \dots, d_{n-w} - d_{n-w+1}) = 1 \underbrace{0 \dots 0}_{w-1}$$

It means all the bits from position  $n - 1$  to  $n - w$  must equal '1', so the probability is  $\frac{1}{2^w}$ . When  $w = 2$ , it is about 0.25 . When  $w = 3$ , it is about 0.125.

**q.e.d.**

### 4.3.3 Average bit-length of fractional wNAF

**Theorem 13** *The average bit-length of fractional wNAF is  $L_{fwNAF}(w, k) = k + \frac{1}{1+w}$ , where  $k$  is the length of binary sequence and  $w = w_0 + w_1$  is the window size.*

**Proof:**

Like what we have done in 4.3.1, we list all the possibilities for fractional wNAF, which the output looks like:

1.  $| \underbrace{0}_1 \underbrace{X \dots X}_{n-1}$
2.  $1 | \underbrace{0}_1 \underbrace{X \dots X}_{n-1}$
3.  $| \underbrace{0 \dots X}_{w_0} \underbrace{X \dots X}_{n-w_0}$
4.  $1 | \underbrace{0 \dots X}_{w_0} \underbrace{X \dots X}_{n-w_0}$
5.  $| \underbrace{0 \dots X}_{w_0+1} \underbrace{X \dots X}_{n-w_0-1}$
6.  $1 | \underbrace{0 \dots X}_{w_0+1} \underbrace{X \dots X}_{n-w_0-1}$

$$\begin{array}{l}
7. \quad | \underbrace{0 \dots X}_{w_0} \underbrace{X \dots X}_{n-w_0} \\
\quad \dots \dots \dots \\
j. \quad | \underbrace{1}_{1} \underbrace{X \dots X}_{n-1}
\end{array}$$

When the window size is  $w = w_0 + w_1$ , the amount of all of these possibilities is  $w_0 + 6$ , we suppose the stats from case.1 to case. $w_0 + 6$  are  $Q_1, Q_2, \dots, Q_{w_0 + 6}$ . When the length of input sequence increases one bit, analyze like subsection 4.3.1, so we can get the Markov chain transition matrix of next state is:

$P$	$Q_1$	$Q_2$	$Q_3$	$Q_4$	$Q_5$	$Q_6$	$Q_7$	$Q_8$	$Q_9$	$\dots$	$\dots$	$\dots$	$Q_{w_0 + 6}$
$Q_1$	$\frac{1}{2}$	0	0	0	0	0	0	0	0	$\dots$	$\dots$	$\dots$	$\frac{1}{2}$
$Q_2$	0	$\frac{1}{2}$	0	0	0	0	0	0	0	$\dots$	$\dots$	$\dots$	$\frac{1}{2}$
$Q_3$	$\frac{1}{2}$	0	0	0	0	0	0	0	0	$\dots$	$\dots$	$\dots$	$\frac{1}{2}$
$Q_4$	0	0	0	0	0	$\frac{1}{2}$	0	0	0	$\dots$	$\dots$	$\dots$	$\frac{1}{2}$
$Q_5$	$\frac{1}{2}$	0	0	0	0	0	0	0	0	$\dots$	$\dots$	$\dots$	$\frac{1}{2}$
$Q_6$	0	$\frac{1}{2}$	0	0	0	0	0	0	0	$\dots$	$\dots$	$\dots$	$\frac{1}{2}$
$Q_7$	0	0	0	0	$\frac{1}{2}$	$\frac{w_1}{1+w_1}$	0	0	0	$\dots$	$\dots$	$\dots$	$\frac{1-w_1}{2(1+w_1)}$
$Q_8$	0	0	0	$\frac{1-w_1}{2}$	0	0	$\frac{1+w_1}{2}$	0	0	$\dots$	$\dots$	$\dots$	0
$Q_9$	0	0	0	0	0	0	0	1	0	$\dots$	$\dots$	$\dots$	0
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	1	$\dots$	$\dots$	$\dots$	$\dots$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	1	$\dots$	$\dots$	$\dots$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	1	$\dots$	$\dots$
$Q_{w_0 + 6}$	0	0	0	0	0	0	0	0	0	$\dots$	$\dots$	1	0

And the stationary distribution of this Markov Chain is  $(\pi(Q_1), \dots, \pi(Q_{w_0 + 6})) = (\frac{1+w_1}{4(1+w)}, \frac{1+w_1}{4(1+w)}, 0, \frac{1-w_1}{2(1+w)}, \frac{1+w_1}{4(1+w)}, \frac{1+w_1}{4(1+w)}, \frac{1+w_1}{2(1+w)}, \underbrace{\frac{1}{1+w} \dots \frac{1}{1+w}}_{w_0 - 1}, 0)$ , So when the

$w = w_0 + w_1$ , the probability of wNAF form with a carry is  $\pi(Q_2) + \pi(Q_4) + \pi(Q_6) = \frac{1}{1+w_0+w_1} = \frac{1}{1+w}$ . It means  $L_{fwNAF}(w, k) = k + \frac{1}{1+w}$

**q.e.d.**

**Theorem 14** *The efficacious length of fractional wNAF is  $EL_{fwNAF}(w_0 + w_1, k) = k - \frac{w_0^2 + 2w_0w_1 - w_0}{2(1+w)}$ , where  $k$  is the length of binary sequence and  $w = w_0 + w_1$  is the window size.*

**Proof:**

Now we compute the average efficacious length of fractional wNAF form. We list all the probabilities:

1. efficacious length is  $n + 1$ :  $\pi(Q_2) + \pi(Q_4) + \pi(Q_6) = \frac{1}{1+w}$
2. efficacious length is  $n$ :  $\pi(Q_{w_0+6}) = \frac{1}{1+w}$
3. efficacious length is  $n - 1$ :  $\pi(Q_{w_0+6}) = \frac{1}{1+w}$
- .....
- j. efficacious length is  $n - w_0 + 1$ :  $\pi(Q_7) = \frac{1+w_1}{2(1+w)}$
- k. efficacious length is  $n - w_0$ :  $\pi(Q_5) = \frac{1+w_1}{4(1+w)}$
- l. efficacious length is  $n - w_0 - 1$ :  $\pi(Q_1) * P(b_1 | b_3) = \frac{1}{2} * \frac{1+w_1}{4(1+w)}$
- .....
- m. efficacious length is  $n - w_0 - k$ :  $\frac{1}{2^k} * \frac{1+w_1}{4(1+w)}$

Now we compute the average efficacious length of fractional wNAF form is

$$EL_{wNAF}(w_0 + w_1, n) = \sum_0^{n+1} EL * Pr(EL) = n - \frac{w_0^2 + 2w_0w_1 - w_0}{2(1+w)}$$

**q.e.d.**

When the input isn't an infinity sequence, but it is big enough, so that stationary distribution approaches to the ideal value, we can use this stationary distribution to complete the result of 4.1.3 . We suppose the input length is  $k$  and  $k \rightarrow \infty$  and amount of non0s bits is  $x$ , because we have got the in 4.1.1 that  $NZ_{fwNAF}(w, k) = \frac{1}{1+w}$ , so

1. when the form is  $|\underbrace{0}_1 \underbrace{X \dots X}_{n-1}$ , non0s bits are  $(k - 1) \frac{1}{1+w}$
2. when the form is  $1 |\underbrace{0}_1 \underbrace{X \dots X}_{n-1}$ , non0s bits are  $(k - 1) \frac{1}{1+w} + 1$
3. when the form is  $|\underbrace{0 \dots X}_{w_0} \underbrace{X \dots X}_{n-w_0}$ , non0s bits are  $(k - w_0) \frac{1}{1+w} + 1$
4. when the form is  $1 |\underbrace{0 \dots X}_{w_0} \underbrace{X \dots X}_{n-w_0}$ , non0s bits are  $(k - w_0) \frac{1}{1+w} + 2$
5. when the form is  $|\underbrace{0 \dots X}_{w_0+1} \underbrace{X \dots X}_{n-w_0-1}$ , non0s bits are  $(k - w_0 - 1) \frac{1}{1+w} + 1$
6. when the form is  $1 |\underbrace{0 \dots X}_{w_0+1} \underbrace{X \dots X}_{n-w_0-1}$ , non0s bits are  $(k - w_0 - 1) \frac{1}{1+w} + 2$
7. when the form is  $|\underbrace{0 \dots X}_{w_0} \underbrace{X \dots X}_{n-w_0}$ , non0s bits are  $(k - w_0) \frac{1}{1+w} + 1$

.....  
 j. when the form is  $\left| \underbrace{1}_1 \underbrace{X \dots X}_{n-1} \right|$ , non0s bits are  $(k-1) \frac{1}{1+w} + 1$

Now we try to product the non0s bits and probabilities of every form

1. form is  $\left| \underbrace{0}_1 \underbrace{X \dots X}_{n-1} \right|$ , non0s bits are  $((k-1) \frac{1}{1+w}) * \frac{1+w_1}{4(1+w)}$
2. form is  $1 \left| \underbrace{0}_1 \underbrace{X \dots X}_{n-1} \right|$ , non0s bits are  $((k-1) \frac{1}{1+w} + 1) * \frac{1+w_1}{4(1+w)}$
3. form is  $\left| \underbrace{0 \dots 0}_{w_0} \underbrace{X \dots X}_{n-w_0} \right|$ , non0s bits are  $((k-w_0) \frac{1}{1+w} + 1) * 0$
4. form is  $1 \left| \underbrace{0 \dots 0}_{w_0} \underbrace{X \dots X}_{n-w_0} \right|$ , non0s bits are  $((k-w_0) \frac{1}{1+w} + 2) * \frac{1-w_1}{2(1+w)}$
5. form is  $\left| \underbrace{0 \dots 0}_{w_0+1} \underbrace{X \dots X}_{n-w_0-1} \right|$ , non0s bits are  $((k-w_0-1) \frac{1}{1+w} + 1) * \frac{1+w_1}{4(1+w)}$
6. form is  $1 \left| \underbrace{0 \dots 0}_{w_0+1} \underbrace{X \dots X}_{n-w_0-1} \right|$ , non0s bits are  $((k-w_0-1) \frac{1}{1+w} + 2) * \frac{1+w_1}{4(1+w)}$
7. form is  $\left| \underbrace{0 \dots 0}_{w_0} \underbrace{X \dots X}_{n-w_0} \right|$ , non0s bits are  $((k-w_0) \frac{1}{1+w} + 1) * \frac{1+w_1}{2(1+w)}$
- .....
- j. form is  $\left| \underbrace{1}_1 \underbrace{X \dots X}_{n-1} \right|$ , non0s bits are  $((k-1) \frac{1}{1+w} + 1) * \frac{1}{1+w}$

So the sum of these possibilities is  $NZ_{fwNAF}(w, k)$ , but here  $k \rightarrow \infty$ , it equal:

$$NZ_{fwNAF}(w_0+w_1, k) = \frac{2kw + 2k + 3w + 1 + w^2 - w_1}{2(1+w)^2} = \frac{k}{1+w} + \frac{1}{2} + \frac{w_0}{2(1+w)^2}$$

Let us see the table 6, when  $w = 7.5$ , we get the academic value is 19.275, and the practical value is 19.279.

### 4.3.4 Average bit-length of fractional wMOF

**Theorem 15** *The average bit-length of fractional wMOF is  $L_{fwMOF}(w_0 + w_1, k) = k + \frac{1}{2^{w_0}}$ , where  $k$  is the length of binary sequence and  $w = w_0 + w_1$  is the window size.*

**Proof:**

We have computed the average bit-length of wMOF in 4.3.2, and in fractional wMOF we need also only the first block to analyze. The first block of the binary input is  $(1, d_{n-2} - 1, \dots, d_{n-w_0-1} - d_{n-w_0})$ . Then we need to test whether the value satisfies with the condition  $| [d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{s-1} - d_s]_2 | \leq 2^{w_0-1}(1 + w_1)$ , and if it is satisfied, then find out the matched fractional wMOF value in the smaller matching table and if it isn't satisfied with the condition, the window size will decrease and find out the fractional wMOF value in the matching table. It remains to compute the value of  $(1, d_{n-2} - 1, \dots, d_{n-w_0} - d_{n-w_0+1})$ , There are two cases that lead to a length increment.

1. When it full-fil the condition, it must be  $\underbrace{10 \dots 0}_{w_0+1}$
2. When it doesn't full-fil the condition, so it must be  $\underbrace{10 \dots 0\bar{1}}_{w_0+1}$

It means the binary input must be begin with  $\underbrace{1 \dots 11}_{w_0+1}$  or  $\underbrace{1 \dots 10}_{w_0+1}$ . The probability that one of these cases happen are  $\frac{1}{2^{w_0}}$ .

**q.e.d.**

### 4.3.5 Conclusion and Test

We have computed the average length and the probabilities of occur a carry for wNAF and wMOF, fractional wNAF and fractional wMOF. Figure 4 illustrates the change when the  $w$  increases. This chapter tells us that the average length of wNAF is  $k + \frac{1}{1+w}$ , of wMOF is  $1 + \frac{1}{2^w}$  of fractional wNAF is  $1 + \frac{1}{2^w}$  and of fractional wMOF is  $1 + \frac{1}{2^{w_0}}$ . But we must ensure that  $w \ll k$ . We gather the test data together in following Table 8.

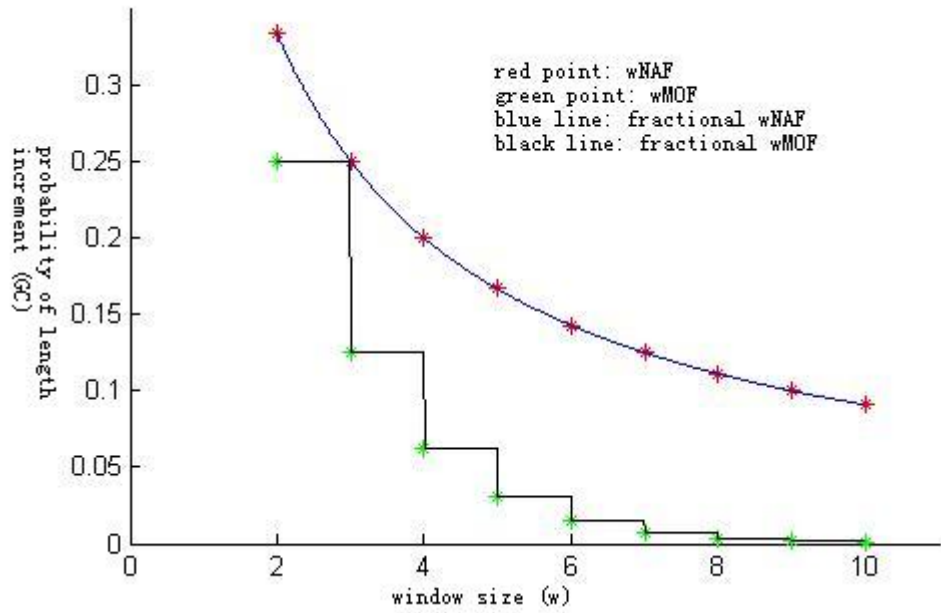


Abbildung 4: average length of wNAF and wMOF

$w$	2(.5)	3(.5)	4(.5)	5(.5)	6(.5)	7(.5)	8(.5)
$A160b(wN)$	0.333	0.250	0.200	0.167	0.143	0.125	0.111
$P160b(wN)$	0.328	0.247	0.198	0.170	0.150	0.127	0.113
$A16000b(wN)$	0.333	0.250	0.200	0.167	0.143	0.125	0.111
$P16000b(wN)$	0.330	0.249	0.200	0.170	0.150	0.127	0.112
$A160b(wM)$	0.250	0.125	0.063	0.031	0.016	0.008	0.004
$P160b(wM)$	0.247	0.122	0.062	0.031	0.016	0.009	0.004
$A16000b(wM)$	0.250	0.125	0.063	0.031	0.016	0.008	0.004
$P16000b(wM)$	0.249	0.125	0.062	0.031	0.016	0.009	0.004
$A160b(fwN)$	X	0.222	0.182	0.154	0.133	0.118	X
$P160b(fwN)$	X	0.219	0.173	0.144	0.129	0.121	X
$A16000b(fwN)$	X	0.222	0.182	0.154	0.133	0.118	X
$P16000b(fwN)$	X	0.223	0.182	0.154	0.133	0.117	X
$A160b(fwM)$	X	0.125	0.063	0.031	0.016	0.008	X
$P160b(fwM)$	X	0.123	0.062	0.031	0.016	0.009	X
$A16000b(fwM)$	X	0.125	0.063	0.031	0.016	0.008	X
$P16000b(fwM)$	X	0.125	0.063	0.031	0.016	0.008	X

Tabelle 8: Test Data: average bit-length

## 5 Conclusion

In this thesis we have analyzed some attributes of wNAF, wMOF, fractional wNAF and fractional wMOF. We also provided test data with the length of the input binary form is 160 bits and 16000 bits. We have got the result that the amount of non0s is about  $\frac{k}{1+w}$ , and the consecutive 0s is about  $w$ . And the average length of wNAF and fractional wNAF is  $k + \frac{1}{1+w}$ . The average length of wMOF and fractional wMOF is  $k + \frac{1}{2^w}$  and  $k + \frac{1}{2^w}$ . From our experiments we conclude that the larger the input length is, the more will the test data approximate the academic data expected from the theorems. However, our results also show that even for parameters size of cryptographic relevant (i.e.160 bits), the academic and practical data are indeed very close. Our results are useful for practical applications, for example, we have got the average of the consecutive 0s, so we can use k-iterated ECDBL instead of repeatedly applying ECDBL  $k$  times when using wNAF and fractional wNAF. It improves the efficiency of computation.

We believe that wNAF and wMOF will be widely used in small memory OS in the future. Especially fractional wNAF and fractional wMOF are promising, because they can fully use the memory resources of OS.

	wNAF	wMOF	fwNAF	fwMOF
NZ	$\frac{1}{1+w}$	$\frac{1}{1+w}$	$\frac{1}{1+w}$	$\frac{1}{1+w}$
CZ	$w$	$w$	$w$	$w$
L	$k + \frac{1}{1+w}$	$k + \frac{1}{2^w}$	$k + \frac{1}{1+w}$	$k + \frac{1}{2^w}$

Tabelle 9: Conclusion Table

## 6 Data

### 6.1 Data of wNAF

$w$	2	3	4	5	6	7	8
<i>Non0s</i>	53.769	40.495	32.426	27.116	23.331	20.514	18.272
	53.769	40.499	32.427	27.114	23.331	20.515	18.276
	53.765	40.498	32.425	27.114	23.332	20.514	18.276
<i>Cons0s</i>	1.988	2.963	3.947	4.913	5.873	6.812	7.770
	1.988	2.963	3.947	4.913	5.873	6.812	7.770
	1.988	2.963	3.947	4.914	5.873	6.812	7.770
<i>Carry</i>	0.328	0.247	0.198	0.170	0.150	0.127	0.113
	0.328	0.247	0.198	0.171	0.151	0.127	0.113
	0.328	0.247	0.198	0.170	0.150	0.127	0.112

Tabelle 10: 160-bits input & 1-million times

$w$	2	3	4	5	6	7	8
<i>Non0s</i>	5333.863	3999.806	3200.253	2667.015	2286.329	2000.755	1778.524
	5333.036	3999.690	3200.342	2666.869	2285.583	2000.521	1778.080
	5333.689	3999.852	3200.114	2667.250	2285.741	2000.362	1777.675
<i>Cons0s</i>	2.000	3.000	4.000	5.000	5.998	6.997	7.996
	2.000	3.000	3.999	5.000	6.000	6.998	7.999
	2.000	3.001	4.000	4.999	6.000	6.998	7.999
<i>Carry</i>	0.330	0.249	0.201	0.170	0.150	0.127	0.113
	0.321	0.248	0.200	0.171	0.150	0.127	0.112
	0.330	0.250	0.200	0.170	0.149	0.127	0.112

Tabelle 11: 16000-bits input & 1-million times

## 6.2 Data of wMOF

$w$	2	3	4	5	6	7	8
<i>Non0s</i>	53.764	40.506	32.430	27.109	23.330	20.513	18.273
	53.760	40.500	32.434	27.123	23.329	20.512	18.278
	53.751	40.492	32.421	27.119	23.334	20.513	18.278
<i>Cons0s</i>	1.988	2.962	3.950	4.915	5.874	6.812	7.770
	1.988	2.964	3.946	4.912	5.874	6.813	7.769
	1.989	2.963	3.948	4.912	5.872	6.812	7.769
<i>Carry</i>	0.248	0.123	0.062	0.031	0.016	0.009	0.004
	0.247	0.122	0.062	0.031	0.016	0.009	0.004
	0.246	0.122	0.062	0.031	0.016	0.009	0.004

Tabelle 12: 160-bits input &amp; 1-million times

$w$	2	3	4	5	6	7	8
<i>Non0s</i>	5333.845	4000.268	3200.466	2666.828	2286.239	2000.987	1778.011
	5333.482	4000.290	3201.197	2667.472	2286.387	2000.614	1778.357
	5333.980	4000.006	3200.978	2667.293	2286.022	2000.577	1778.321
<i>Cons0s</i>	2.000	3.000	3.999	5.000	5.999	6.996	7.999
	2.000	3.000	3.999	4.998	5.998	6.998	7.997
	2.000	3.000	3.998	4.999	5.999	6.998	7.997
<i>Carry</i>	0.250	0.125	0.062	0.031	0.017	0.009	0.004
	0.249	0.125	0.063	0.032	0.016	0.009	0.004
	0.249	0.124	0.062	0.031	0.016	0.009	0.005

Tabelle 13: 16000-bits input &amp; 1-million times

## 6.3 Data of fractional wNAF

$w$	3.5	4.5	5.5	6.5	7.5
<i>Non0s</i>	36.061	29.464	25.077	21.824	19.278
	36.049	29.463	25.084	21.830	19.281
	36.050	29.241	25.080	21.819	19.279
<i>Cons0s</i>	3.450	4.444	5.396	6.347	7.314
	3.453	4.444	5.394	6.345	7.312
	3.452	4.444	5.394	6.348	7.314
<i>Carry</i>	0.221	0.174	0.142	0.127	0.122
	0.2210	0.172	0.145	0.130	0.123
	0.219	0.174	0.144	0.129	0.122

Tabelle 14: 160-bits input &amp; 1-million times

$w$	3.5	4.5	5.5	6.5	7.5
<i>Non0s</i>	3555.623	2909.809	2462.571	2133.700	1882.679
	3555.549	2909.259	2462.110	2133.535	1883.279
	3555.934	2909.691	2462.682	2133.756	1883.125
<i>Cons0s</i>	3.500	4.499	5.497	6.499	7.499
	3.500	4.500	5.499	6.499	7.499
	3.500	4.499	5.497	6.499	7.499
<i>Carry</i>	0.222	0.183	0.153	0.133	0.118
	0.223	0.182	0.154	0.133	0.117
	0.223	0.182	0.154	0.132	0.117

Tabelle 15: 16000-bits input &amp; 1-million times

## 6.4 Data of fractional wMOF

$w$	3.5	4.5	5.5	6.5	7.5
<i>Non0s</i>	36.058	29.462	25.080	21.818	19.278
	36.043	29.460	25.079	21.826	19.275
	36.062	29.460	25.078	21.830	19.275
<i>Cons0s</i>	3.451	4.444	5.395	6.349	7.314
	3.453	4.445	5.395	6.346	7.315
	3.450	4.445	5.395	6.345	7.315
<i>Carry</i>	0.123	0.062	0.030	0.016	0.009
	0.122	0.062	0.032	0.016	0.009
	0.124	0.063	0.031	0.016	0.009

Tabelle 16: 160-bits input &amp; 1-million times

$w$	3.5	4.5	5.5	6.5	7.5
<i>Non0s</i>	3555.314	2909.867	2461.958	2133.625	1882.861
	3556.310	2909.440	2462.315	2134.062	1882.981
	3555.702	2909.615	2462.037	2133.446	1882.944
<i>Cons0s</i>	3.500	4.499	5.499	6.500	7.498
	3.499	4.499	5.498	6.498	7.497
	3.500	4.499	5.498	6.498	7.497
<i>Carry</i>	0.124	0.062	0.031	0.015	0.009
	0.125	0.064	0.032	0.016	0.008
	0.125	0.063	0.031	0.016	0.008

Tabelle 17: 16000-bits input &amp; 1-million times

## Literatur

- [1] I.F.Blake, G.Seroussi, and N.P.Smart. Elliptic Curves in Cryptography. Cambridge University Press, 1999. ISBN 0-512-65374-6.
- [2] K.Itoh, M.Takenaka, N. Torii, S.Temma, and Y.Kurihara. Fast implementation of public-key cryptography on dsp tms320c6210. CHES'99 pages 61-72,1999
- [3] IEEE P1363. Standard specifications for public-key cryptography. [Http://groupe.ieee.org/groups/1363/](http://groupe.ieee.org/groups/1363/).
- [4] Jerome A.Solinas Efficient Arithmetic on Koblitz Curves. Designs, Codes and Cryptography, 19, 195-249(2000).
- [5] J.A.Solinas. Efficient arithmetic on Koblitz curves. Designs, Codes and Cryptography.
- [6] E. T. Jaynes, PROBABILITY THEORY:THE LOGIC OF SCIENCE.
- [7] B. Moeller. Improved techniques for fast exponentiation. In Information Security and Cryptology C ICISC 2002, volume 2587 of Lecture Notes in Computer Science, pages 298-312, Berlin, 2003. Springer-Verlag.
- [8] O.Haeggstroem. Finite Markov Chains and Algorithmic Applications. Cambridge University Press, 2002.
- [9] Bodo Moeller. Fractional Windows Revisited: Improved Signed-Digit Representations for Efficient Exponentiation.
- [10] B. Moeller. Improved techniques for fast exponentiation. In Information Security and Cryptology ICISC 2002, volume 2587 of Lecture Notes in Computer Science, pages 298-312, Berlin, 2003. Springer-Verlag.
- [11] D.M.Gordon. A survey of fast exponentiation methods. Journal of Algorithms.
- [12] Katsuyuki Okeya, Katja Schmidt-Samoa, Christian Spahn, and Tsuyoshi Takagi. Signed Binary Representations Revisited.
- [13] K.Okeya, K.Schmidt-Samoa, C.Spahn, and T.Takagi. Signed binary representations revisited. In Advances in Cryptology C CRYPTO 2004, volume 3152 of Lecture Notes in Computer Science. Springer-Verlag, 2004.

- [14] MOF, <http://www.sdl.hitachi.co.jp/crypto/mof/index-g.html>.
- [15] Cohen, H. Miyaji, A. And Ono, T. Efficient Elliptic Curve Exponentiation Using Mixed Coordinates, Advances in Cryptology C ASIACRYPT'98.
- [16] Joye, M and Yen, S.M. Optimal Left-to-right Binary Signed-digit Exponent Recoding, IEEE Transactions on Computers.
- [17] Katja Schmidt-Samoa, Olivier Semay, and Tsuyoshi Takagi. Analysis of Some Efficient Window Methods and their Application to Elliptic Curve Cryptosystems. 2004.
- [18] Olivier Semay and Tsuyoshi Takagi. Efficiency analysis of window methods using Markov Chains.

## 7 Appendix

---

### Algorithm 6 Scalar Multiplication with wMOF

---

*Input:* a point  $P$ , a scalar  $d = [d_{n-1}, \dots, d_0]_2$ , a width  $w$   
*Output:* the point  $dP$   
 $P_1 \leftarrow P; P_2 \leftarrow ECDBL(P_1);$   
**for**  $j = 1$  to  $2^{w-2} - 1$  **do**  
     $P_{2j+1} = ECADD(P_{2j-1}, P_2)$   
**end for**  
 $A \leftarrow O; i \leftarrow n; d_{-1} \leftarrow 0; d_n \leftarrow 0;$   
**while**  $i \geq 0$  **do**  
    **if**  $d_{i-1} = d_i$  **then**  
         $A \leftarrow ECDBL(A); i \leftarrow i - 1;$   
    **else**  
         $s \leftarrow \max(i - w + 1, 0);$   
        Let  $t$  be the smallest integer such that  $t \geq s$  and  $d_{t-1} \neq d_t;$   
        **for**  $k = 1$  to  $i - t + 1$  **do**  
             $A \leftarrow ECDBL(A);$   
        **end for**  
        **if**  $[d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{t-1} - d_t]_2 > 0$  **then**  
             $A \leftarrow ECADD(A, P_{[d_{i-1}-d_i, d_{i-2}-d_{i-1}, \dots, d_{t-1}-d_t]_2});$   
        **end if**  
        **if**  $[d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{t-1} - d_t]_2 < 0$  **then**  
             $A \leftarrow ECADD(A, -P_{|[d_{i-1}-d_i, d_{i-2}-d_{i-1}, \dots, d_{t-1}-d_t]_2|});$   
        **end if**  
        **for**  $k = 1$  to  $t - s$  **do**  
             $A \leftarrow ECDBL(A);$   
        **end for**  
         $i \leftarrow s - 1;$   
    **end if**  
**end while**  
**Return**  $A$

---

---

**Algorithm 7** Sliding window applied on NAF
 

---

*Input:* a point  $P$ , a scalar  $d = [d_{n-1}, \dots, d_0]_2$ , a width  $w$

*Output:* the point  $dP$

Pre-computation:

$P_1 \leftarrow P; P_2 \leftarrow ECDBL(P_1);$

**for**  $j = 1$  to  $\frac{2^w + (-1)^{w+1}}{3} - 1$  **do**

$P_{2j+1} = ECADD(P_{2j-1}, P_2)$

**end for**

Recoding, first part:

$i \leftarrow 0; k \leftarrow d$

**while**  $k > 0$  **do**

**if**  $k$  is odd **then**

$v[i] \bmod 4;$

$k \leftarrow k - v[i];$

**else**

$v[i] \leftarrow 0;$

**end if**

$k \leftarrow \frac{k}{2}; i \leftarrow i + 1;$

**end while**

Recoding (second part) and Evaluation:

$A \leftarrow O; i \leftarrow n - 1;$

**while**  $i \geq 0$  **do**

**if**  $V_i = 0$  **then**

$A \leftarrow ECDBL(A); i \leftarrow i - 1;$

**else**

$s \leftarrow \max(i - w + 1, 0);$

**for**  $k = 1$  to  $i - t + 1$  **do**

$A \leftarrow ECDBL(A);$

**end for**

**if**  $[V_i, V_{i-1}, \dots, V_t]_2 > 0$  **then**

$A \leftarrow ECADD(A, P_{[V_i, V_{i-1}, \dots, V_t]_2});$

**else**

$A \leftarrow ECADD(A, -P_{[V_i, V_{i-1}, \dots, V_t]_2})$

**end if**

**for**  $k = 1$  to  $t - s$  **do**

$A \leftarrow ECDBL(A);$

**end for**

$i \leftarrow s - 1$

**end if**

**end while**

**Return**  $A$

---

---

**Algorithm 8** Scalar Multiplication with wNAF
 

---

*Input:* a point  $P$ , a scalar  $d = [d_{n-1}, \dots, d_0]_2$ , a width  $w$

*Output:* the point  $dP$

Pre-computation:

$P_1 \leftarrow P; P_2 \leftarrow ECDBL(P_1);$

**for**  $j = 1$  to  $2^{w-2} - 1$  **do**

$P_{2j+1} = ECADD(P_{2j-1}, P_2)$

**end for**

Recoding:

$i \leftarrow 0; k \leftarrow d$

**while**  $k > 0$  **do**

**if**  $k$  is odd **then**

$V_w[i] \leftarrow k \bmod 2^w;$

$k \leftarrow k - V_w[i];$

**else**

$V_w[i] \leftarrow 0;$

**end if**

$k \leftarrow \frac{k}{2}; i \leftarrow i + 1;$

**end while**

Evaluation:

find the largest  $c$  with  $V_w[c] \neq 0$

**if**  $V_w[c] > 0$  **then**

$A \leftarrow P_{V_w[c]}$

**end if**

**if**  $V_w[c] < 0$  **then**

$A \leftarrow -P_{V_w[c]}$

**end if**

**for**  $i = c - 1$  to  $0$  **do**

$A \leftarrow ECDBL(A);$

**if**  $V_w[i] > 0$  **then**

$ECADD(A, P_{V_w[i]});$

**end if**

**if**  $V_w[i] < 0$  **then**

$ECADD(A, -P_{V_w[i]});$

**end if**

**end for**

**Return**  $A;$

---

**Algorithm 9** Scalar Multiplication with fractional wMOF

*Input:* a point  $P$ , a scalar  $d = [d_{n-1}, \dots, d_0]_2$ , a width  $w = w_0 + w_1$ , where  $w_1 = \frac{r}{2^{w_0-2}}$  and  $w_0 = w - w_1$  for an  $r$  between 0 and  $2^{w_0-2}$

*Output:* the point  $dP$

Pre-computation:

$P_1 \leftarrow P; P_2 \leftarrow ECDBL(P_1);$

**for**  $j = 1$  to  $2^{w_0-2}(1 + w_1) - 1$  **do**

$P_{2j+1} = ECADD(P_{2j-1}, P_2)$

**end for**

Recoding and Evaluation:

$A \leftarrow O; i \leftarrow n; d_{-1} \leftarrow 0; d_n \leftarrow 0;$

**while**  $i \geq 0$  **do**

**if**  $d_{i-1} = d_i$  **then**

$A \leftarrow ECDBL(A);$

$i \leftarrow i - 1;$

**else**

$s \leftarrow \max(i - w_0, 0);$

**if**  $d_{s-1} \neq d_s$  and  $|[d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{s-1} - d_s]_2| \geq 2^{w_0-1}(1 + w_1)$

**then**

$s = s + 1;$

**end if**

        Let  $t$  be the smallest integer such that  $t \geq s$  and  $d_{t-1} \neq d_t;$

**for**  $k = 1$  to  $i - t + 1$  **do**

$A \leftarrow ECDBL(A);$

**end for**

**if**  $[d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{t-1} - d_t]_2 > 0$  **then**

$A \leftarrow ECADD(A, P_{[d_{i-1}-d_i, d_{i-2}-d_{i-1}, \dots, d_{t-1}-d_t]_2});$

**end if**

**if**  $[d_{i-1} - d_i, d_{i-2} - d_{i-1}, \dots, d_{t-1} - d_t]_2 < 0$  **then**

$A \leftarrow ECADD(A, -P_{|[d_{i-1}-d_i, d_{i-2}-d_{i-1}, \dots, d_{t-1}-d_t]_2|});$

**end if**

**for**  $k = 1$  to  $t - s$  **do**

$A \leftarrow ECDBL(A);$

**end for**

$i \leftarrow s - 1;$

**end if**

**end while**

**Return**  $A$