A Note on the Signed Sliding Window Integer Recoding and a Left-to-Right Analogue

Roberto Maria Avanzi*

Institute for Experimental Mathematics (IEM) − Universität Duisburg–Essen
Ellernstraße 29, D-45326 Essen, Germany
mocenigo@exp-math.uni-essen.de
Communication Security (COSY) − Electrical Engineering and Information Technology
Ruhr-Universität Bochum, Universitätsstraße 150, D-44780 Bochum, Germany

Abstract. Addition-subtraction-chains obtained from signed digit recodings of integers are a common tool for computing multiples of random elements of a group where the computation of inverses is a fast operation. Cohen and Solinas independently described one such recoding, the w-NAF. For scalars of the size commonly used in cryptographic applications, it leads to the current scalar multiplication algorithm of choice. However, we could find no formal proof of its optimality in the literature. This recoding is computed right-to-left.

We solve two open questions regarding the w-NAF. We first prove that the w-NAF is a redundant radix-2 recoding of smallest weight among all those with integral coefficients smaller in absolute value than 2^{w-1}. Secondly, we introduce a left-to-right recoding with the same digit set as the w-NAF, generalizing previous results. We also prove that the two recodings have the same (optimal) weight. Finally, we sketch how to prove similar results for other recodings.

Keywords: Computer arithmetic, Integer Recoding, Non-adjacent form, Width-w non-adjacent form (w-NAF), Signed-digit representation, Redundant number representation, Left-to-right recoding.

1 Introduction

This paper deals with *signed sliding window integer recodings*. They are used to speed up computations in Abelian groups where inversion has negligible computational cost. Notable examples of such groups are: the rational point group of an elliptic curve, independently suggested for cryptographic applications by Koblitz [14] and Miller [16]; the group of rational divisor classes of a hyperelliptic curve [15]; and the XTR subgroup in its natural representation [24]. In the cryptosystems designed around such groups the fundamental computation is the *scalar multiplication*, *i.e.* the computation of the n-fold ng of a group element g for an arbitrary scalar $n \in \mathbb{Z}$.

An addition (resp. addition-subtraction) chain for an integer n is a sequence of integers beginning with 1 and ending with n such that each element is the sum (resp.

* The work described in this paper has been supported in part by the European Commission through the IST Programme under Contracts IST-2001-32613 (AREHCC).

H. Handschuh and A. Hasan (Eds.): SAC 2004, LNCS 3357, pp. 130–143, 2005.

sum or difference) of two previous elements of the sequence. For example $\{1,2,3,6,9\}$ and $\{1,2,4,8,9\}$ are addition chains for 9. The computation of ng for any element g of a group G can be done using an addition(-subtraction) chain for n. For example, $9g$ can be obtained by computing $g, 2g, 3g, 6g$ and finally $9g$. Methods for finding short addition(-subtraction) chains are very important: The shorter is a chain, the faster will be the corresponding scalar multiplication. An addition-subtraction can be shorter than the shortest addition chain for the same integer. A shortest addition chain for 31 is $\{1,2,3,6,7,14,28,31\}$, but a shortest addition-subtraction chain is $\{1,2,4,8,16,32,31\}$.

Downey, Leong, and Sethi [6] proved that the decision problem whose instances are the tuples $(k, n_1, n_2, \ldots, n_k, \ell)$ such that there is an addition chain of length ℓ containing $n_1, n_2, \ldots, n_k$ is NP-complete. It is conjectured that the subset of instances with fixed k, such as $k = 1$, is also NP-complete. Even if it were so, and thus the problem of finding shortest chains were NP-hard, this would not necessarily rule out the existence of efficient algorithms to find near optimal chains. Erdős [7] proved that for almost all integers n the shortest addition chain for n has length $\lambda(n) + \lambda(n)/\lambda(\lambda(n))$ where $\lambda(n) = \lfloor \log_2(n) \rfloor$. The upper bound $\lambda(n) + (1 + o(1))\lambda(n)/\lambda(\lambda(n))$ is attained by means of Thurber's [25] *sliding windows* variation of Brauer's method [3] - which is therefore a concrete example of a method giving almost always near optimal addition chains. Similar considerations hold for addition-subtraction chains.

Let $n = \sum_{i=0}^{\ell} n_i 2^i$ be a *recoding* of the integer n, where the integers n_i belong to a digit set $\mathcal{S}$ with $\{0,1\} \subseteq \mathcal{S} \subset \mathbb{Z}$, and $n_\ell \neq 0$. If digits other than zero and one are allowed, we have a *redundant representation*.

It is easy to build an addition-subtraction chain from such a recoding by a technique called *double-and-add*. The chain starts with $\{1,2,\ldots,\max\{|n_i|\}\}$. For $i = \ell - 1, ..., 2, 1, 0$, we do the following: if $n_i = 0$ we append $a_i := 2a_{i+1}$ to the chain; if $n_i \neq 0$ we append both $2a_{i+1}$ and $a_i := 2a_{i+1} + n_i$. At the end, $a_0 = n$. If the non-zero coefficients are odd, to save some operations the chain can begin after 2 begin with $\{1,2,3,5,7,\ldots,\max\{|n_i|\}\}$ (the even numbers other than 2 are omitted). Apart from the cost of computing $2g, 3g$,...,$\max\{|n_i|\}g$, the amount of doublings, resp. generic additions in G equals the length, resp. the weight of the recoding: We recall that the *length* of any expression $n = \sum_{i=0}^{\infty} n_i 2^i$ with only a finite number of non-vanishing terms, is $\ell + 1$, where ℓ is the highest index for which $n_\ell \neq 0$; its *weight* is the number of non-zero coefficients. It is then interesting to find recodings which minimize those quantities.

Cohen et al. [5] and Solinas [22] independently introduced the *width-w non-adjacent form*, or *w-NAF* of the scalar. This signed digit representation is defined in Section 2. For scalars of the size commonly used in cryptographic applications, it leads to the current scalar multiplication algorithm of choice. Cohen [4] analyzes weight and length of the w-NAF as a function of the input length.

A potential drawback of the w-NAF is that it scans the bits from the least significant to the most significant ones, *i.e.* from right to left. This means that the recoding must be completely known and stored in memory before the scalar multiplication in the group G is performed with a classical double-and-add scalar multiplication algorithm. This can be disadvantageous on a device with limited resources like a smart card. On the other hand, if the bits could be scanned and the digits of the recoding generated from

left to right, one could *interleave* recoding and scalar multiplication: this is sometimes called *online recoding*. Of course one can interleave a right-to-left recoding with Yao's [26] scalar multiplication algorithm, but this method requires a few more operations than double-and-add even with the same recoding. (Surveys of scalar multiplication algorithms can be found in [8] (slightly outdated) and [2].)

Not only for its intrinsic practical interest, but also because the subject itself is challenging, the development of algorithms for left-to-right optimal recodings has recently attracted a lot of attention in the mathematical community [9, 12, 13]. Rizzo [21] has analyzed the left-to-right unsigned sliding window method.

The *non adjacent form* (NAF) [20] is the special case of the w-NAF with $w = 2$. It is known that the NAF is a recoding of minimal weight among all those having coefficients in $\{0, \pm 1\}$. As we started our research, there was no formal proof in the literature that the general w-NAF is a recoding of minimal weight among those having the same digit set. We provide this proof in Section 2, Theorem 2.3.

Joye and Yen [12] devised a left-to-right recoding using the digit set $\{0, \pm 1\}$, having the same weight as the NAF, and which can be used online. Their method does not apply to the w-NAF. We introduce a left-to-right analogue of the w-NAF in Section 3, and in Section 4 we prove that it has the same weight as the w-NAF. While proving our results, we in fact show that the two recodings parse their inputs essentially in the same way, and use the same function of $w + 1$ consecutive bits to generate the non-zero digits. Some remarks, including generalizations of our results to unsigned recodings and joint recodings of more integers, as well as references to related similar work, conclude the paper.

2 Optimality

For each integer n and for any value of the positive integral parameter $w \geq 2$, the w-NAF of n is a representation $n = \sum_{j=0}^{\ell} n_j 2^j$ where the integer coefficients n_j satisfy the following two conditions:

(w-NAF-1) Either $n_j = 0$ or n_j is odd and $|n_j| < 2^{w-1}$.
(w-NAF-2) If $n_j \neq 0$, then $n_{j+1} = \cdots = n_{j+w-1} = 0$.

Using the fact that the set of odd integers of absolute value smaller than 2^{w-1} is a complete residue set modulo 2^w for the odd integers, it is easy to write a simple algorithm to generate a w-NAF for each integer n and to prove that it is uniquely determined.

The algorithm works as follows: A temporary variable x is put equal to the input. If x is even, it is halved and a zero is output; otherwise the smallest residue r of x modulo 2^w is subtracted from x; x is then halved w times and the digit r is output followed by $w - 1$ zeros. This process is repeated until x reaches 0. Correctness and termination are obvious. For the applications we use a version that works directly on the binary representation of the input n and does not modify it, whence it needs (at least in theory) to keep a carry.

Algorithm 2.1 w-NAF Recoding

INPUT: An integer $n = \sum_{j=0}^{\ell-1} e_j\, 2^j$, and a parameter $w \geq 2$.

OUTPUT: The w-NAF $\sum_{j=0}^{\ell} n_j\, 2^j$ of n.

1. Initialize $n_j \leftarrow 0$ for $0 \leq j \leq \ell$. Assume $e_\ell = 0$.
2. $i \leftarrow 0, c \leftarrow 0$.
3. **while** $(i \leq \ell)$ **do** $\{$
4. **if** $e_i = c$ **then** $\{$
5. $i \leftarrow i+1$.
6. $\}$ **else** $\{$
7. $w' \leftarrow \min\{w, \ell - i + 1\}$.
8. $v \leftarrow c + \sum_{j=0}^{w'-1} e_{i+j} 2^j$.
9. **if** $v \geq 2^{w-1}$ **then** $c \leftarrow 1, v \leftarrow v - 2^w$ **else** $c \leftarrow 0$.
10. $n_i \leftarrow v$.
11. $i \leftarrow i + w'$. $\}$ $\}$
12. **return**

Note that, at Step 8 it is in fact $v = 1 + \sum_{j=1}^{w'-1} e_{i+j} 2^j$ because $c \neq e_j$ and the variables c, e_j can only take the values 0 and 1. This also guarantees that v, being odd, is non-zero.

Definition 2.2 *Let n, w be integers. A w-*signed digit recoding*, or w-SDR for short, is an expression of the type $n = \sum_{i=0}^{\ell} n_i 2^i$ where the digits n_i are integers with $|n_i| < 2^{w-1}$. (There is no restriction that the digits of a w-SDR either vanish or are odd.)*

We use the notation $\mathcal{H}(\cdot)$ to denote the Hamming weight of any recoding, where by Hamming weight of $\sum_{i=0}^{\ell} n_i 2^i$ we understand the number of non vanishing digits n_i.

The w-NAF of an integer n is a particular w-SDR.

Theorem 2.3 *Let n, w be integers. The w-NAF of n is a recoding of minimal weight among all the w-SDR's of n.*

Proof. We define *two* transformations of recodings. The first one, called *coefficient formation* (**CF**), takes as input any w-SDR of an integer n, and outputs another recoding of n. If n is odd, the least significant digit of the new recoding is non-zero, the next $w-1$ digits are zero and the w-th digit (the coefficient of 2^w) may have absolute value possibly $\geq 2^{w-1}$ – in which case we say that an *overflow* has been generated – but still smaller than 2^w; all the other digits are equal to the corresponding ones of the input. If n is even, the least significant digit of the new recoding will be zero, and the second least significant digit may be an overflow. The Hamming weight of the output of **CF** is never greater than that of the input, and it is strictly smaller than that if an overflow has been generated. The second transformation, called *overflow propagation* (**OP**) is applied if

only if **CF** has generated an overflow. Its output may have Hamming weight equal to that of the input, plus one.

We now describe in detail the two transformations, and prove some of their properties. Later we shall use them to prove the theorem.

Coefficient formation: If $n = \sum_{i=0}^{\ell} n_i 2^i$ is even, then n_0 is even. Put $n'_0 = 0$ and $n'_1 = n_1 + n_0/2$. If $n_1 = 0$, then $|n'_1| = |n_1| < 2^{w-1}$ and the Hamming weight does not change if we replace n_0, n_1 with n'_0, n'_1. If $|n'_1| \geq 2^{w-1}$ then n_0 and n_1 were both non-zero and the Hamming weight decreases by one. Plainly $|n'_1| < \frac{3}{2}2^{w-1} < 2^w$ and $n_0 + 2n_1 = n'_0 + 2n'_1$, hence if we replace n_0, n_1 with n'_0, n'_1 we obtain a new recoding for n, which is also a w-SDR except for at most one overflow.

Let now n be odd. Define $M = \sum_{i=0}^{w-1} n_i 2^i$ (if the input recoding has length than w, we suitably pad it with zeros). Now,

$$|M| \leq (2^{w-1} - 1) \cdot \sum_{i=0}^{w-1} 2^i = (2^{w-1} - 1)(2^w - 1) = 2^{2w-1} - 3 \cdot 2^{w-1} + 1 \ ,$$

hence there exist integers c, r such that $M = c\,2^w + r$ with $|c| < 2^{w-1}$ and $|r| \leq 2^{w-1}$. Put now $n'_w = n_w + c$, $n'_0 = r$ and $n'_j = 0$ for all j with $0 < j < w$.

Let $h = \mathcal{H}\big(\sum_{i=0}^{w} n_i 2^i\big)$ and $h' = \mathcal{H}\big(\sum_{i=0}^{w} n'_i 2^i\big)$. We want to show that $h \geq h'$.

First note that only one coefficient among the w least significant ones in $\sum_{i=0}^{w} n'_i 2^i$ is non-zero, namely n'_0, whereas at least one of the integers $n_0, n_1, \ldots, n_{w-1}$ is non-zero.

It can be $c \neq 0$ only if at least one of the digits $n_1, \ldots, n_{w-1}$ is non-zero, because if these digits all vanish then $|M| < 2^{w-1}$ and $c = 0$. In the case where $c \neq 0$, it is therefore $h \geq 2$ and $h' \leq 2$.

If an overflow occurs, it must be $c \neq 0$ and $n_w \neq 0$, hence $h \geq 3$ but $h' \leq 2$ by construction.

If we replace n_j with n'_j for $0 \leq j \leq w$, we get a recoding for n whose w least significant digits satisfy the properties of the w-NAF.

Overflow propagation: Let t be the only index in a recoding (which is the output of **CF**) such that $|n_t| \geq 2^{w-1}$, but $|n_t| < 2^w$. Define $n'_t = \mathrm{sign}(n_t)(n_t \bmod 2)$ and $n'_{t+1} = n_{t+1} + \frac{n_t - n'_t}{2}$. Replace then n_t and n_{t+1} by n'_t and n'_{t+1}. Clearly, $|n_{t+1}| < 2^w$. If $|n_{t+1}| \geq 2^{w-1}$, we repeat the procedure just described with $t+1$ in place of t, otherwise we stop. This process must terminate by the finiteness of the recoding, and it can increase the Hamming weight at most by one.

OP is performed after **CF** if and only if the latter generates an overflow and decreases the Hamming weight at least by one. Therefore the application of the two transformations will not increase the Hamming weight of the input.

We can now prove the theorem by induction.

For integers of absolute value at most 2^{w-1} it is obvious that the w-NAF has minimal weight. Let then $|n| > 2^{w-1}$ in the sequel.

We consider first the case where n is odd. Let $\sum_{i=0}^{\ell} n_i 2^i$ be original recoding, of weight h, and $\sum_{i=0}^{\ell'} n'_i 2^i$ be the output of **CF** and **OP**, which has weight not greater than h. Recall that $n'_0 \neq 0$ and $n'_1 = \ldots = n'_{w-1} = 0$ by construction.

Consider $m = \frac{n-n'_0}{2^w}$. Its w-NAF $\sum_{i=0}^{\lambda} m_i 2^i$ (for some upper bound λ on the indices) has by the inductive assumption weight not greater than the weight of $\sum_{i=0}^{\ell'-w} n'_{i+w} 2^i$. Since $n'_0 + \sum_{i=0}^{\lambda} m_i 2^{i+w}$ satisfies the defining properties of the w-NAF, it is the w-NAF of n and its weight satisfies

$$\mathcal{H}\left(n'_0 + \sum_{i=0}^{\lambda} m_i 2^{i+w}\right) = 1 + \mathcal{H}\left(\sum_{i=0}^{\lambda} m_i 2^{i+w}\right) \leq$$
$$\leq 1 + \mathcal{H}\left(\sum_{i=0}^{\ell'-w} n'_{i+w} 2^i\right) = \mathcal{H}\left(\sum_{i=0}^{\ell'} n'_i 2^i\right) \leq h \ .$$

If n is even, proceed in a similar (but simpler) way, using the w-NAF of $n/2$ to complete the proof. □

Remark 2.4 It is immediate to see that, upon repeated application of **CF** and **OP**, one can obtain the w-NAF of any integer from a w-SDR. Making some obvious simplifications and the overflow propagation implicit, we obtain the following algorithm:

Algorithm 2.5 w-SDR to w-NAF recoding

INPUT: An integer n given by a w-SDR $n = \sum_{j=0}^{\ell-1} e_j 2^j$.

OUTPUT: The w-NAF $\sum_{j=0}^{\ell+w-1} n_j 2^j$ of n.

1. Initialize $n_j \leftarrow 0$ for $0 \leq j < \ell + w$.
2. $i \leftarrow 0$, $c \leftarrow 0$.
3. **while** $(i < \ell)$ **do** {
4. **if** $e_i + c$ is even **then** {
5. $i \leftarrow i+1$, $c \leftarrow \frac{e_i + c}{2}$.
6. } **else** {
7. $M \leftarrow c + \sum_{j=0}^{w-1} e_{i+j} 2^j$. $[|M| \leq 2^{2w-1} - 2^{w-1} + 1]$
8. Write $M = c2^w + r$ with r odd, $|r| < 2^{w-1}$, and $|c| < 2^{w-1}$.
9. Put $n_i \leftarrow r$.
10. $i \leftarrow i + w$. } }
11. **if** $c \neq 0$ **then** {
12. **while** c is even **do** {
13. $i \leftarrow i+1$, $c \leftarrow c/2$. }
14. $n_i \leftarrow c$. }
15. **return**

3 A Left-to-Right Recoding

We introduce a left-to-right recoding using the same digit set as the w-NAF. For $w = 2$, it is different from Joye and Yen's analogue of the NAF, but it produces the same output. (On the other hand, Joye and Yen's algorithm has no conditional tests and branches and is thus resistent against simple power analysis on the recoding of the scalar.) We shall then prove the correctness of our algorithm.

In the next Section we shall see that it scans its input essentially in the same way as Algorithm 2.1, but in reversed order and that its output has the same Hamming weight as the w-NAF.

Algorithm 3.1 w-LtoR: a left-to-right recoding

INPUT: An integer $n = \sum_{j=0}^{\ell-1} e_j 2^j$, with $e_{\ell-1} = 1$, and a parameter $w \geq 2$.

OUTPUT: A signed digit recoding of n as $\sum_{j=0}^{\ell} n_j 2^j$ where each digit n_j are either 0 or odd and $|n_j| < 2^{w-1}$.

1. Put $n_j \leftarrow 0$ for $0 \leq j < \ell + w$. Assume $e_\ell = e_{-1} = 0$.
2. $i \leftarrow \ell$.
3. **while** $(i \geq 0)$ **do** {
4. **if** $(e_i = e_{i-1})$ **then** {
5. $i \leftarrow i - 1$.
6. } **else** {
7. $w' \leftarrow \min\{w, i+1\}$.
8. $v \leftarrow -e_i 2^{w'-1} + \sum_{j=0}^{w'-2} e_{i-(w'-1)+j} 2^j + e_{i-w'}$.
9. $n_{i-(w'-1)+s} \leftarrow v/2^s$, where $2^s \| v$.
10. $i \leftarrow i - w'$. } }
11. **return**

Remark 3.2 We show that v as computed in Step 8 is non-zero. We are in the second branch of the if-then-else construct, in other words $e_i \neq e_{i-1}$. If $i = 0$ then $w' = 1$, but since the second branch has been taken, this means that $e_0 \neq 0$, and clearly $v = -1$. Let now be $w' > 1$. If $e_i = 1$ then $e_{i-1} = 0$ and the positive contributions to v are $\leq 2^{w'-2}$, therefore $v < 0$. (Of course, if in a summation the upper value of the index is smaller than the initial value, the sum is intended empty, hence 0.) If $e_i = 0$ then $e_{i-1} = 1$ and $v \geq 2^{w'-2} > 0$.

An useful way of interpreting the expression in Step 8 is the following one. The value of v is obtained considering the number represented by the string of w' consecutive bits whose most significant bit is the i-th bit, with the sign of the i-th bit itself changed, and adding the following bit to this number. For example, if $w = 3$ and the 3 bits starting

from the i-th one are 1011, the value of v is $\bar{1}01 + 1 = -2$. If these bits had been 0111, v would have been $\bar{0}11 + 1 = 4$.

Remark 3.3 Algorithms 2.1 and 3.1 look very similar. In fact, they have been written intentionally that way, in order to make the proof that the recodings they generate have the same Hamming weight (Theorem 4.2) as straightforward as possible. Nevertheless, they can produce quite different outputs.

Theorem 3.4 *Algorithm 3.1 is correct, i.e. its output is an expression which evaluates to the integer represented by its input.*

Proof. Consider the expression

$$\sum_{j=0}^{i-1} e_j 2^j - e_i 2^i + \sum_{j=i+1}^{\ell} n_j 2^j \quad . \tag{1}$$

At the beginning of the algorithm, before entering the loop with $i = \ell$, (1) equals n. Upon exiting the loop, (1) is the output of the algorithm. To prove the statement, we just need to show that (1) is an invariant of the algorithm.

Suppose first that the condition at Step 4 is evaluated to true, *i.e.* $e_i = e_{i-1}$. The only action is to decrement i by 1, and $n_i = 0$ is left as initialised, whence

$$\begin{aligned}\sum_{j=0}^{i-1} e_j 2^j - e_i 2^i + \sum_{j=i+1}^{\ell} n_j 2^j &= \sum_{j=0}^{i-2} e_j 2^j + e_{i-1} 2^{i-1} - e_i 2^i + \sum_{j=i}^{\ell} n_j 2^j \\ &= \sum_{j=0}^{i-2} e_j 2^j - e_{i-1} 2^{i-1} + \sum_{j=i}^{\ell} n_j 2^j \quad .\end{aligned}$$

Let now $i' = i - 1$, *i.e.* i' is the value that i will take after Step 5, we see that the last expression is in fact $\sum_{j=0}^{i'-1} e_j 2^j - e_{i'} 2^{i'} + \sum_{j=i'+1}^{\ell} n_j 2^j$, which proves that (1) is invariant in this case.

Suppose, on the other hand, that the second branch is taken. Before Step 9 we have $n_{i-(w'-1)+s} = 0$ for all s with $0 \le s \le w' - 1$ and

$$\begin{aligned}&\sum_{j=0}^{i-1} e_j 2^j - e_i 2^i + \sum_{j=i+1}^{\ell} n_j 2^j = \\ &= \Big(\sum_{j=0}^{i-w'-1} e_j 2^j + e_{i-w'} 2^{i-w'} + \sum_{j=0}^{w'-2} e_{i-(w'-1)+j} 2^{i-(w'-1)+j} \Big) - e_i 2^i + \sum_{j=i+1}^{\ell} n_j 2^j \\ &= \sum_{j=0}^{i-w'-1} e_j 2^j - e_{i-w'} 2^{i-w'} + \Big(- e_i 2^{w'-1} + \sum_{j=0}^{w'-2} e_{i-(w'-1)+j} 2^j + e_{i-w'} \Big) 2^{i-(w'-1)} + \sum_{j=i+1}^{\ell} n_j 2^j \\ &= \sum_{j=0}^{i-w'-1} e_j 2^j - e_{i-w'} 2^{i-w'} + \Big(\sum_{j=i+1}^{\ell} n_j 2^j + v 2^{i-(w'-1)} \Big) \quad .\end{aligned} \tag{2}$$

Putting $n_{i-(w'-1)+s} = v/2^s$ (note that $0 \leq s \leq w'-1$) in Step 9 – and leaving $n_{i-(w'-1)+t} = 0$ for all $t \neq s$ with $0 \leq t \leq w'-1$, as initialised at the beginning of the algorithm – we obtain that (2) equals

$$\sum_{j=0}^{i-w'-1} e_j 2^j - e_{i-w'} 2^{i-w'} + \sum_{j=i-w'+1}^{\ell} n_j 2^j \ .$$

Similarly to what we have done before, let now $i' = i - w'$, i.e. i' is the value that i will take after Step 10. The last expression becomes $\sum_{j=0}^{i'-1} e_j 2^j - e_{i'} 2^{i'} + \sum_{j=i'+1}^{\ell} n_j 2^j$, proving (1) invariant also in this case. □

4 Equivalence

Definition 4.1 *For any integer n, denote by w-NAF(n) its w-NAF, i.e. the output of Algorithm 2.1, and by w-LtoR(n) the output of Algorithm 3.1.*

If $n = \sum_{i=0}^{\ell} n_i 2^i$ with $n_i \in \{0,1\}$ and $n_\ell = 1$, let rev(n) *be the integer obtained by reversing the binary representation of n, i.e.* $\mathrm{rev}(n) = \sum_{i=0}^{\ell} n_{\ell-i} 2^i$. *If n is odd,* $\mathrm{rev}(\mathrm{rev}(n)) = n$.

In this section we shall prove the following result.

Theorem 4.2 *For all integers n, and for every value of the parameter $w \geq 2$, the w-NAF of n and the output of Algorithm 3.1 always have the same Hamming weight. In other words $\mathcal{H}(w\text{-NAF}(n)) = \mathcal{H}(w\text{-LtoR}(n))$ for all n and w.*

In particular, the w-LtoR is a recoding of minimal weight among all the w-SDR's, besides the w-NAF.

Proof. The proof is based on a comparison of Algorithms 2.1 and 3.1, showing that they both scan their input in the same way.

At Step 8 of Algorithm 2.1 it is $v \geq 2^w$ (in which case, in fact, $v > 2^w$) if and only if $w = w'$ and $e_{i+w} = 1$. This implies that c in Step 9 is equal to e_{i+w}: In other words c contains the "last bit read" (in the previously formed window). Assuming $e_k = 0$ for $k < 0$, Step 4 can be written as "**if** $(e_i = e_{i-1})$ **then** …", and in Step 9

$$v = -e_{i+w'-1} 2^{w'-1} + \sum_{j=0}^{w'-2} e_{i+j} 2^j + e_{i-1} \ .$$

We see that there is no need to store a carry bit to compute the w-NAF, exactly as for the w-LtoR.

In Algorithm 3.1 the test in Step 4 is also a comparison of the bit at the current position with the "last bit read" (becaue the bits between two windows are equal to each other).

We can rewrite Algorithm 3.1 by modifying a few steps as follows:

7. $w' \leftarrow \min\{w, i+1\},\ i \leftarrow i-(w'-1)$
8. $v \leftarrow -e_{i+w'-1}2^{w'-1} + \sum_{j=0}^{w'-2} e_{i+j}2^j + e_{i-1}.$
9. $n_{i+s} \leftarrow v/2^s$, where $2^s \| v$.
10. $i \leftarrow i-1.$ } }

We immediately see that *the expression for the value v is the same for both algorithms*. The expression is deceivingly simple, too: *consider a window of $w'+1$ consecutive bits; add the least significant one to the number represented by the w' most significant ones but with the highest bit negated*. In fact, the algorithms also scan the bit pattern of the input using the same set of rules!

The algorithms' behaviour can be described by a simple finite state machine that reads the input (in the form of a string of zeros and ones) from a *finite* read-only tape. At the beginning a reading head is placed on one end of the tape and it can only advance towards the other end. The machine stops when the tape has been read completely. We shall also assume that the input string is padded with zeros at both ends in order to properly handle the termination and the cases when $w' < w$, yet using only the parameter w and ignoring the additional variable w'. (Putting $e_{-1} = e_\ell = 0$ in Algorithms 2.1 and 3.1 served the same purpose, yet only for the case $w' = 1$.) The machine has also two registers: c, holding the "last bit read"; and e, containing the bit at the current position, which decides whether a new window has to be formed from the current position. The machine might write something with a second head on a second tape (in fact the algorithms write on an *output* tape), but the details of this operation are not important here.

1. $c \leftarrow 0$
2. **while** (tape not finished) **do** {
3. $e \leftarrow$ bit at current position
4. **if** $(e = c)$ **then** {
5. Advance head by one position.
6. } **else** {
7. Advance head by $w-1$ positions.
8. $c \leftarrow$ bit at current position.
9. Advance head by one position.
10. *(Output a non-zero digit.)*
11. } }

The only difference which concerns us here is the following: Algorithm 2.1 scans the bit string corresponding to the input from right to left, whereas Algorithm 3.1 works left-to-right.

Let us now feed a bit sequence to the first algorithm, and then the reversing of the same bit sequence to the second algorithm. As far as only the number of non-zero digits in the output is concerned, the operations of the two algorithms can be described by two different runs of the machine, where the tape *and* the direction of the head have been reversed between the two runs. Clearly, the machine will perfom in the two runs exactly the same steps and in the same order.

The two algorithms will therefore output the same number of non-zero digits even though the values of the digits may differ. In other words: $\mathcal{H}(w\text{-NAF}(n)) = \mathcal{H}(w\text{-LtoR}(\text{rev}(n)))$ and $\mathcal{H}(w\text{-LtoR}(n)) = \mathcal{H}(w\text{-NAF}(\text{rev}(n)))$.

Suppose now that the Theorem is false. Then, being the w-NAF a w-SDR of minimal weight (Theorem 2.3), there exists a (without loss of generality) odd positive integer n for which $\mathcal{H}(w\text{-NAF}(n)) < \mathcal{H}(w\text{-LtoR}(n))$. This in turn implies that $\mathcal{H}(w\text{-LtoR}(\text{rev}(n))) < \mathcal{H}(w\text{-NAF}(\text{rev}(n)))$, a contradiction. □

Note that not only the individual digits, but also the *lengths* of the w-NAF and of the w-LtoR of the same integer may differ. The 4-NAF and the 4-LtoR of 1971 (the author's year of birth) coincide and are equal to $(1000000\bar{5}0003)$. On the other hand, the 4-NAF of 2004 is $(10000\bar{1}000500)$ whereas its 4-LtoR is $(1000000\bar{5}\bar{1}00)$ – here the two recodings differ but have the same length. Let us consider now 2359, the dreaded CET last minute for submitting papers to SAC (without the seconds): the 4-NAF is $(1000\bar{7}00030007)$ but the 4-LtoR is $(500\bar{3}00\bar{1}00\bar{1})$, which is shorter. Of course, in all examples shown, the w-NAF and the w-LtoR have the same Hamming weight. The recodings of 2004 and 2359 are examples of the fact that the w-LtoR does not necessarily satisfy the generalized non-adjacency property ***w*-NAF-2** of the w-NAF.

5 Additional Remarks

The approach presented in this paper can be used to show the corresponding results for the unsigned sliding window recodings. The only (marginal) difficulty is adapting the proof of the optimality to the right-to-left unsigned sliding window recoding. Then, the result can be interpreted at a purely combinatorial level: the problem is here *grouping and counting substrings of bounded length in strings of two symbols, say 0 and 1, where the zeros do not need to belong to a substring, but all ones must belong to some substring*. Then, it is obvious that the right-to-left and the left-to-right algorithms create the same number of windows on a bit sequence and on its reversing, respectively. From this, as in the concluding arguments of the proof of Theorem 4.2, it follows that they must form the same number of windows also on the same input. Therefore, it does not come as a surprise that Cohen's [4] analysis of the expected Hamming weight for the right-to-left unsigned sliding window method and Rizzo's [21] for the left-to-right method lead to the same result. Here, too, the *lengths* of the expansions can be different.

As another application of these ideas, let us consider the *Joint Sparse Form* (JSF), introduced by Solinas [23] to make *Shamir's trick* more effective for elliptic curves. It is a simultaneous recoding of two integers $n_i = \sum_{j=0}^{\ell} e_{i,j} 2^j$ for $i = 0, 1$ with digit set $\{0, \pm 1\}$, uniquely determined by the following properties:

(JSF-1) Of any three consecutive columns $\binom{e_{0,j}}{e_{1,j}}$, at least one is zero.
(JSF-2) Adjacent non-zero bits have the same sign, *i.e.* $e_{i,j+1}e_{i,j} = 0$ or 1.
(JSF-3) If $e_{i,j+1}e_{i,j} \neq 0$ then $e_{1-i,j+1} \neq 0$ and $e_{1-i,j} = 0$.

Here the weight is the number of non-zero columns. The JSF has optimal weight and expected density $1/2$. Note that joining the NAFs of the two given integers produces a representation with expected density $5/9$. The JSF is computed right-to-left, but a left-to-right variant of the same weight exists [9]. Avanzi in [1] lets windows slide over a JSF to further speed-up the computation of linear combinations of elements of a group. The windows are formed distinguishing only between zero and non-zero columns, the actual content of the non-zero columns playing no role. The complexity of this method has been carefully analyzed in [10]. Constructing the windows from, say, right-to-left, produces a minimal number of windows: To prove this claim we reuse the result for the unsigned sliding windows recoding of one integer, in its combinatorial interpretation - where we have two symbols, say 0 and $\star$ in place of every zero, resp. non-zero column. By our arguments the number of windows obtained forming from right-to-left and from left-to-right must be the same. (The same clearly applies also to windows sliding over a joint recoding of more than two integers.)

During the preparation of the final version of the paper the author became aware of the fact that Muir and Stinson [17, 18] independently obtained similar results. In particular, they also proved the minimality of the w-NAF and found an optimal left-to-right recoding with the same digit set. Our Theorem 2.3 is slightly more general and our proofs are shorter. Their left-to-right algorithm is optimal, is different from ours and can output up to two different recodings of the same integer, one of which is equal to that of our algorithm, whereas the other one differs on some of the least significant digits. Okeya et al. [19] also have a left-to-right algorithm, do not prove equivalence to the w-NAF but only give asymptotic density estimates using Markov chains.

The w-NAF and the w-LtoR of an integer n can be computed also letting windows of length w slide (from right to left and from left to right respectively) on the *alternating greedy expansion* [10] of n. This is a representation $n = \sum_{i=0}^{\ell} \varepsilon_i 2^i$ with digits 0 and ± 1 satisfying the following two properties

(AGE-1) If $\varepsilon_j = \varepsilon_i \neq 0$ for some $j < i$, then there is an index k with $j < k < i$ such that $\varepsilon_j = -\varepsilon_k = \varepsilon_i$.
(AGE-2) For $j_* := \min\{j : \varepsilon_j \neq 0\}$ and $j^* := \max\{j : \varepsilon_j \neq 0\}$, we have $\mathrm{sign}(n) = \varepsilon_{j^*} = -\varepsilon_{j_*}$.

The alternating greedy expansion is computed from the binary expansion $n = \sum_{i=0}^{\ell-1} e_i 2^i$ simply putting $\varepsilon_i = e_{i-1} - e_i$, where it is understood $e_{-1} = e_\ell = 0$.

To obtain the w-NAF and the w-LtoR, the windows slide on the alternate greedy expansion distinguishing only between zero and non-zero digits. This interpretation can be found in [11]. Using it and the results on the minimality of sliding window methods

on string with two symbols, the fact that these representations have the same Hamming weight follows at once.

Acknowledgment. Many results in this paper have been discovered by the author in November 2002 traveling in a train to Oberwolfach with Tanja Lange and Preda Mihăilescu. The author acknowledges useful discussions with Henri Cohen, Matthijs Coster, Clemens Heuberger, Tanja Lange, Bodo Möller, James Muir, Helmut Prodinger and Ottavio Rizzo, and the support of Simonetta. Gratitude goes also to the anonymous reviewers for their comments.

References

1. R. M. Avanzi. *On the complexity of certain multi-exponentiation techniques in cryptography*. To appear in: J. of Cryptology.
2. D. J. Bernstein. *Pippenger's exponentiation algorithm*. Preprint. Available from `http://cr.yp.to`
3. A. Brauer. *On addition chains*. Bull. AMS. **45**, pages 736–739 (1939).
4. H. Cohen. *Analysis of the flexible window powering algorithm*. To appear in: J. of Cryptology.
5. H. Cohen, A. Miyaji, and T. Ono. *Efficient elliptic curve exponentiation*. In *Proceedings ICICS'97*, LNCS 1334, 282–290. Springer, 1997.
6. P. Downey, B. Leong, and R. Sethi. *Computing sequences with addition chains*. SIAM J. Computing **10**, 638–646 (1981). MR 82h:68064.
7. P. Erdφs. *Remarks on number theory III. On addition chains.* Acta Arith., **6**, 77–81 (1960).
8. D. Gordon. *A Survey of Fast Exponentiation Methods*. J. of Algorithms, **27**, 129–146 (1998).
9. P. J. Grabner, C. Heuberger, and H. Prodinger. *Distribution results for low-weight binary representations for pairs of integers*. Theoretical Computer Science, to appear.
10. P. J. Grabner, C. Heuberger, H. Prodinger, and J. Thuswaldner. *Analysis of linear combination algorithms in cryptography.* Preprint. Available from: `http://www.opt.math.tu-graz.ac.at/~cheub/publications/Windows.pdf`
11. C. Heuberger, R. Katti, H. Prodinger, and X. Ruan. *The Alternating Greedy Expansion and Applications to Left-To-Right Algorithms in Cryptography*. Preprint.
12. M. Joye, and S.-M. Yen. *Optimal left-to-right binary signed-digit recoding*. IEEE Trans. on Comp. **49** (7), 740–748 (2000).
13. M. Joye, and S.-M. Yen. *New Minimal Modified Radix-r Representation*. In *Proceedings of PKC 2002*. LNCS 2274, 375–384. Springer, 2003.
14. N. Koblitz. *Elliptic curve cryptosystems*. Math. Comp. **48** (177), 203–209 (1987).
15. N. Koblitz. *Hyperelliptic cryptosystems*. J. of Cryptology **1**, 139–150 (1989).
16. V. S. Miller. *Use of elliptic curves in cryptography*. In: Proceedings of Crypto '85, LNCS 218, 417–426. Springer, 1986.
17. J. A. Muir, and D. R. Stinson. *Minimality and Other Properties of the Width-w Nonadjacent Form.* Technical Report CORR 2004-08, Centre for Applied Cryptographic Research. Available from `http://www.cacr.math.uwaterloo.ca/techreports/2004/`
18. J. A. Muir, and D. R. Stinson. *New Minimal Weight Representations for Left-to-Right Window Methods.* Technical Report CACR 2004-03, Centre for Applied Cryptographic Research. Available from `http://www.cacr.math.uwaterloo.ca/techreports/2004/`
19. K. Okeya, K. Schmidt-Samoa, C. Spahn, and T. Takagi. *Signed Binary Representations Revisited*. Proceedings of Crypto 2004.
20. G. W. Reitwiesner. *Binary arithmetic.* Advances in Computers **1**, 231–308 (1960).

21. O. Rizzo. *On the complexity of the 2^k-ary and of the sliding window algorithms for fast exponentiation*. To appear in: Rivista di Matematica dell'Universitá di Parma.
22. J. A. Solinas. *An improved algorithm for arithmetic on a family of elliptic curves.* In *Proceedings of CRYPTO '97*, LNCS 1294, 357–371. Springer, 1997.
23. J. A. Solinas. *Low-Weight Binary Representations for Pairs of Integers*. Centre for Applied Cryptographic Research, University of Waterloo, Combinatorics and Optimization Research Report **CORR 2001-41**, 2001. Available from:
`http://www.cacr.math.uwaterloo.ca/techreports/2001/corr2001-41.ps`
24. M. Stam and A. K. Lenstra. *Efficient subgroup exponentiation in quadratic and sixth degree extensions*. In *Proceedings of CHES 2002*. LNCS 2523, 318–332. Springer, 2003.
25. E. G. Thurber. *On addition chains $l(mn) \leq l(n)b$ and lower bounds for $c(r)$*. Duke Math. J. **40** 907–913 (1973).
26. A. C. Yao. *On the evaluation of powers.* SIAM J. Computing **5** 100–103 (1976).