On the invertibility of the XOR rotations of a binary word

The MIT Faculty has made this article openly available. Please share how this access benefits you. Your story matters.

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>As Published</td>
<td><a href="http://dx.doi.org/10.1080/00207161003596708">http://dx.doi.org/10.1080/00207161003596708</a></td>
</tr>
<tr>
<td>Publisher</td>
<td>Taylor &amp; Francis Group</td>
</tr>
<tr>
<td>Version</td>
<td>Author’s final manuscript</td>
</tr>
<tr>
<td>Citable Link</td>
<td><a href="http://hdl.handle.net/1721.1/72052">http://hdl.handle.net/1721.1/72052</a></td>
</tr>
<tr>
<td>Terms of Use</td>
<td>Creative Commons Attribution-Noncommercial-Share Alike 3.0</td>
</tr>
<tr>
<td>Detailed Terms</td>
<td><a href="http://creativecommons.org/licenses/by-nc-sa/3.0/">http://creativecommons.org/licenses/by-nc-sa/3.0/</a></td>
</tr>
</tbody>
</table>
On the invertibility of the XOR of rotations of a binary word

Ronald L. Rivest*

November 10, 2009

Abstract

We prove the following result regarding operations on a binary word whose length is a power of two: computing the exclusive-or of a number of rotated versions of the word is an invertible (one-to-one) operation if and only if the number of versions combined is odd.

(This result is not new; there is at least one earlier proof, due to Thomsen in his PhD thesis [12]. Our proof may be new.)

Keywords: invertibility, exclusive-or, rotation, binary words, circulant matrix.

1 Introduction and proof of main result

This short note considers some simple operations on binary words.

We only consider binary words whose length is a power of two, as this is typically the case for actual computer operations (e.g., with 32-bit or 64-bit words).

We focus on operations based on rotations and exclusive-or's, as these are typically standard built-in operations.

Simple invertible operations such as these are used in many applications, such as pseudo-random number generation [7, 9], encryption [4], and cryptographic hash function design [10].

We state and prove the main result, and then provide some related discussion afterwards.

Theorem 1 If \( n \) is a power of two, \( v \) is an \( n \)-bit word, and \( r_1, r_2, \ldots, r_k \) are distinct fixed integers modulo \( n \), then the function

\[
R(v) = R(v; r_1, r_2, \ldots, r_k) = (v \ll r_1) \oplus (v \ll r_2) \oplus \cdots \oplus (v \ll r_k)
\]  

(1)

is invertible if and only if \( k \) is odd, where \((v \ll r)\) denotes the \( n \)-bit word \( v \) rotated left by \( r \) positions, and where \( \oplus \) denotes the bit-wise “exclusive-or” of \( n \)-bit words.

*Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 rivest@mit.edu
Proof: Let \( V = \{0, 1\} \), and let \( V^n \) denote the set of all \( n \)-bit words. We identify \( V^n \) with \( GF(2)^n \), the set of \( n \)-element vectors over the finite field \( GF(2) \).

With this identification, \( R \) is a linear operation over \( V^n \); \( R(v) \) may be obtained by multiplying \( v \) by an \( n \times n \) circulant matrix over \( GF(2) \) having \( k \) ones per row and per column. (An equivalent statement of our theorem is that when \( n \) is a power of two, an \( n \times n \) circulant matrix over \( GF(2) \) is invertible if and only if the number \( k \) of ones in each row is odd.)

We define the Hamming weight (or weight) of an \( n \)-bit word \( v \) to be the number of ones in \( v \).

Our proof identifies words in \( V^n \) with polynomials in \( GF(2)[x] \) of degree less than \( n \).

For each \( n \)-bit word \( v \) we define an associated polynomial \( v(x) \) in \( GF(2)[x] \) in the natural way: if

\[
v = (v_{n-1}, v_{n-2}, \ldots, v_1, v_0)
\]

then the associated polynomial \( v(x) \) is

\[
v(x) = \sum_{i=0}^{n-1} v_i x^i.
\]

For example, the unit-weight word \( u_i \) having a one in position \( i \) is associated with the polynomial \( u_i(x) = x^i \). This association between words and polynomials is one-to-one.

Let \( f_n(x) = x^n + 1 \), a polynomial in \( GF(2)[x] \). We now work with polynomials modulo \( f_n(x) \), so that rotation can be effected by polynomial multiplication modulo \( f_n(x) \), as is typically done when working with cyclic error-correcting codes (see [6, Section 9.2]) or circulant matrices (see [1]).

Now the word

\[(v \ll r)\]

is associated with the polynomial

\[v(x) * u_r(x) \pmod{f_n(x)} ;\]

reducing modulo \( f_n \) captures the effects of the rotation. In other words, multiplying by \( u_r(x) \) modulo \( f_n(x) \) represents a left-rotation by \( r \) positions.

Computing \( R(v) \) combines the effect of several rotations, so the word \( R(v) \) is associated with the polynomial

\[v(x) * r(x) \pmod{f_n(x)} \]

where

\[r(x) = x^{r_1} + x^{r_2} + \cdots + x^{r_k} .\]

Note that \( R \) is an invertible operation if and only if \( r(x) \) is relatively prime to \( f_n(x) \); (This result is due to Guan et al. [5, Theorem 2.4]; see also Bini et al. [1, Theorem 2.2].) If \( \gcd(r(x), f_n(x)) = 1 \), then an inverse to \( r(x) \) modulo \( f_n(x) \) can be found by the extended version of Euclid’s algorithm, otherwise no inverse exists. These propositions hold whether or not \( n \) is a power of two.
If $n$ is a power of two, then
\[ f_n(x) = x^n + 1 = (x + 1)^n, \]
since we are working in $GF(2)$ (see [6, Thm. 1.46]). In this case, $r(x)$ is relatively prime to $f_n(x)$ if and only if $r(x)$ is relatively prime to the polynomial $x + 1$.

Polynomials that are not relatively prime to $x + 1$ must be multiples of $x + 1$, since $x + 1$ is irreducible. A polynomial in $GF(2)[x]$ is a multiple of $x + 1$ if and only if its value at $x = 1$ is 0. But $r(1) = 0$ if and only if $r(x)$ has an even number of non-zero coefficients. Therefore $r(x)$ is relatively prime to $f_n(x)$ if and only if $k$ is odd.

Thus, when $n$ is a power of two, $R$ is an invertible operation on $GF(2)^n$ if and only if $k$ is odd.

2 Discussion

The inverse operation to $R$ can be found using Euclid’s extended algorithm on input polynomials $r(x)$ and $f_n(x)$, to find polynomials $s(x)$ and $t(x)$ such that
\[ s(x) \cdot r(x) + t(x) \cdot f_n(x) = 1. \]
The inverse operation $S$ to $R$ corresponds to the polynomial $s(x)$, representing another function of the same form as $R$ (that is, an xor of rotations). In matrix terms, the inverse of a circulant matrix is another circulant matrix.

In terms of computational complexity, $R(v)$ is easy to compute when $k$ is small, requiring not more than $k$ rotations and $k - 1$ xors. Although the inverse $S$ has the same form as $R$, it may require considerably more work to compute. For example, if $r(x)$ has degree $d$, then $s(x)$ must have degree at least $n/d$ and at least $n/d$ terms, so that evaluating $S(v)$ requires at least $\log_2(n/d)$ additions, since each addition in a computation chain can at most double the number of terms. Here multiplication by $x^r$ (rotations) are “free” and we are only counting exclusive-or. The exact complexity, in terms of rotations and xors, of evaluating $R(v)$ or $S(v)$ may be non-trivial to determine precisely, and we leave these questions as open problems. Thus, when $k$ and $d$ are small $R$ may be considered to be in some sense “very modestly one-way”—easier to compute in one direction than another. Stephen Boyack [3] has interesting related results on the complexity of matrix operations over $GF(2)$ and their inverses.

Efficient invertible operations are useful in many applications. A linear operation somewhat similar to the one studied here is the “xorshift” operation:

\[ v = v \oplus (v \ll r) \]
where “$\ll$” is the “left-shift” operator; xorshift has been used in pseudorandom number generation [7, 9] and hash-function design [10]. Schnorr and Vaudenay [11, Lemma 5] study the related operation

\[ (v \land d) \oplus (v \ll\ll r) \]
where “∧” denote bitwise “and” and where \( d \) is a constant \( n \)-bit word; they show that this operation is invertible if and only if the iterates \((d \ll\ll (r\cdot i))\) take for each bit position the value 0 for some \( i \).

The result of this paper may be useful to those working on similar applications. For example, we began our study of \( R \) when thinking about possible improvements to the MD6 hash function [10]. We also note that the \( k = 3 \) version of the operation discussed here is used in the C2 cipher [2] (although not in manner that required its invertibility (it is part of the feedback function in a Feistel block-cipher)), and in the SHA hash function standard message expansion computation [8] (as the \( \Sigma \) function; invertibility of \( \Sigma \) is not claimed or proven).

When \( n \) is not a power of 2, we don’t know of any comparably simple characterization of when \( R(v) \) is invertible, other than the requirement that \( \gcd(f_n(x), r(x)) = 1 \); perhaps simpler characterizations can be found for some cases, such as when \( n = 3 \cdot 2^k \).

3 Related Work

Lars Knudsen points out that a different proof for the same result is available in the Ph.D. thesis [12, Theorem 3.3, pages 86–87] of Søren Thomsen. Thomsen’s cute proof considers powers \( R^{2^i} \) of the original operation, notes that

\[
R^2(v; r_1, r_2, \ldots, r_k) = R(v; 2r_1, 2r_2, \ldots, 2r_k)
\]

from which it follows that \( R \) is invertible since \( R^n \) will be the identity function (if and only if \( k \) is odd).

4 Conclusions

This note provides an alternate proof of a characterization as to when an easily computed operation, based on the exclusive-or of rotated versions of a word, is invertible.

Acknowledgments

I’d like to thank Lars Knudsen, Niels Ferguson, Louay M. J. Bazzi, and Sanjoy K. Mitter for helpful comments and suggestions.

References


