# Reed-Muller Codes

Sebastian Raaphorst
Carleton University

May 9, 2003

**Abstract**

This paper examines the family of codes known as Reed-Muller codes. We begin by briefly introducing the codes and their history before delving in and examining how they are used in practice. We investigate one particular representation of generator matrices for these codes, and then discuss the encoding and the decoding process.

After the basics have been covered, we examine the recursive nature of Reed-Muller codes, and touch on their links to design theory and affine geometries.

# Contents

# 1 Introduction and history

Reed-Muller codes are amongst the oldest and most well-known of codes. They were discovered and proposed by D. E. Muller and I. S. Reed in 1954.

Reed-Muller codes have many interesting properties that are worth examination; they form an infinite family of codes, and larger Reed-Muller codes can be constructed from smaller ones. This particular observation leads us to show that Reed-Muller codes can be defined recursively.

Unfortunately, Reed-Muller codes become weaker as their length increases. However, they are often used as building blocks in other codes.

One of the major advantages of Reed-Muller codes is their relative simplicity to encode messages and decode received transmissions. We examine encoding using generator matrices and decoding using one form of a process known as *majority logic*.

Reed-Muller codes, like many other codes, have tight links to design theory; we briefly investigate this link between Reed-Muller codes and the designs resulting from affine geometries.

Finally, we present the reader with an implementation of the Reed-Muller encoding and decoding process, written in ANSI C.

# 2 Monomials and vectors over $\mathbb{F}_2$

We begin by introducing the concept of vector spaces over $\mathbb{F}_2$ and their correspondance to certain rings of polynomials.

*Notation:* We use the notation $\mathbb{Z}_n$ to refer to the set $\{0, 1, \ldots, n-1\}$.

*Notation:* We use a string of length $n$ with elements in $\mathbb{F}_2$ to write a vector in the vector space $\mathbb{F}_2^n$ for brevity when this is unambiguous. For example, if we have the vector $\sigma = (1, 0, 1, 1, 0, 1, 0) \in \mathbb{F}_2^7$, we simply write $\sigma$ as 1011010.

*Notation:* Assume that $x$ and $y$ are vectors over our vector space $\mathbb{F}_2^n$ (with $x = (x_1, x_2, \ldots, x_n)$, and $y = (y_1, y_2, \ldots, y_n)$). Let $a$ be an element of $\mathbb{F}_2$, where we call $a$ a *scalar*. Then, we have the standard operations of addition and scalar multiplication for vectors (these are not described here, and we assume that the reader is familiar with them). However, we also extend the set of operations in the following way:

- *Scalar addition*
$$a + x = (a + x_1, a + x_2, \ldots, a + x_n)$$
- *Vector complement*
$$\overline{x} = 1 + x = (1 + x_1, 1 + x_2, \ldots, 1 + x_n)$$
- *Vector multiplication*
$$x * y = (x_1 * y_1, x_2 * y_2, \ldots, x_n * y_n)$$

Note that $(\mathbb{F}_2^n, +, *)$ forms a commutative ring.

Assume that we are given a vector space $\mathbb{F}_2^{2^m}$. Then we consider the ring $\mathcal{R}_m = \mathbb{F}_2[x_0, x_1, \ldots, x_{m-1}]$. We will see shortly that there exists a bijection between elements of $\mathcal{R}_m$ and $\mathbb{F}_2^{2^m}$ (in fact, an isomorphism of rings between $(\mathcal{R}_m, +, *)$ and $(\mathbb{F}_2^{2^m}, +, *)$).

*Definitions:* A *Boolean monomial* is an element $p \in \mathcal{R}_m$ of the form:

$$p = x_0^{r_0} x_1^{r_1} \ldots x_{m-1}^{r_{m-1}}$$

where $r_i \in \mathbb{N}$ and $i \in \mathbb{Z}_m$. A *Boolean polynomial* is simply, as expected, a linear combination (with coefficients in $\mathbb{F}_2$) of boolean monomials; any element of $\mathcal{R}_m$ may be thought of as a Boolean polynomial.

*Definition:* Given a Boolean monomial $p \in \mathcal{R}_m$, we say that $p$ is in reduced form if it is squarefree. For any Boolean monomial $q \in \mathcal{R}_m$, it is trivial to find the reduced form of $q$ by applying:

$$\begin{aligned} x_i x_j &= x_j x_i \quad \text{as } \mathcal{R}_m \text{ is a commutative ring} \\ x_i^2 &= x_i \quad \text{as } 0 * 0 = 0 \text{ and } 1 * 1 = 1 \end{aligned}$$

A Boolean polynomial in reduced form is simply a linear combination of reduced-form Boolean monomials (with coefficients in $\mathbb{F}_2$).

*Example:* Say we have the Boolean polynomial $p = 1 + x_1 + x_0^5 x_2^2 + x_0 x_1^4 x_2^{101} \in \mathcal{R}_3$. Then, by applying the above rules, we can get its reduced form, $p'$:

$$p' = 1 + x_1 + x_0 x_2 + x_0 x_1 x_2$$

Consider the mapping $\psi : \mathcal{R}_m \to \mathbb{F}_2^{2^m}$, defined as follows:

$$\begin{aligned}
\psi(0) &= \underbrace{00\ldots0}_{2^m} \\
\psi(1) &= \underbrace{11\ldots1}_{2^m} \\
\psi(x_0) &= \underbrace{11\ldots1}_{2^{m-1}}\underbrace{00\ldots0}_{2^{m-1}} \\
\psi(x_1) &= \underbrace{11\ldots1}_{2^{m-2}}\underbrace{00\ldots0}_{2^{m-2}}\underbrace{11\ldots1}_{2^{m-2}}\underbrace{00\ldots0}_{2^{m-2}} \\
\psi(x_2) &= \underbrace{11\ldots1}_{2^{m-3}}\underbrace{00\ldots0}_{2^{m-3}}\underbrace{11\ldots1}_{2^{m-3}}\underbrace{00\ldots0}_{2^{m-3}}\underbrace{11\ldots1}_{2^{m-3}}\underbrace{00\ldots0}_{2^{m-3}}\underbrace{11\ldots1}_{2^{m-3}}\underbrace{00\ldots0}_{2^{m-3}} \\
&\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
\psi(x_i) &= \underbrace{11\ldots1}_{2^{m-i}}\underbrace{00\ldots0}_{2^{m-i}}\ldots \\
&\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots
\end{aligned}$$

For any monomial $p \in \mathcal{R}_m$, to calculate $\psi(p)$, we find the reduced form:

$$p' = x_{i_1} x_{i_2} \ldots x_{i_r}$$

(where $i_j \in \mathbb{Z}_m$, $i_j = i_k \rightarrow j = k$, and $0 \leq r \leq m$). Then,

$$\psi(p) = \psi(x_{i_1}) * \psi(x_{i_2}) * \cdots * \psi(x_{i_r})$$

For any polynomial $q \in \mathcal{R}_m$, we can write $q$ as:

$$q = m_1 + m_2 + \ldots + m_r$$

(where $m_i$ is a monomial of $\mathcal{R}_m$, $m_i = m_j \rightarrow i = j$, and $0 \leq r \leq 2^m$). Then,

$$\psi(q) = \psi(m_1) + \psi(m_2) + \ldots + \psi(m_r)$$

*Example:* Let $p = 1 + x_1 + x_0^5 x_2^2 + x_0 x_1^4 x_2^{101} \in \mathcal{R}_3$, and we have seen from above that $p$ has reduced form $p' = 1 + x_1 + x_0 x_2 + x_0 x_1 x_2$. Then:

$$\begin{aligned}
\psi(p) &= \psi(p') \\
&= \psi(1) + \psi(x_1) + \psi(x_0 x_2) + \psi(x_0 x_1 x_2) \\
&= \psi(1) + \psi(x_1) + \psi(x_0) * \psi(x_2) + \psi(x_0) * \psi(x_1) * \psi(x_2) \\
&= 11111111 + 11001100 + 11110000 * 10101010 + 11110000 * 11001100 * 10101010 \\
&= 00010011
\end{aligned}$$

*Proposition:* $\psi$ is a bijection (not shown), and indeed, $\psi$ is a homomorphism of rings (proof not shown, but obvious by construction of $\psi$). Thus, $\mathcal{R}_m$ and $\mathbb{F}_2^{2^m}$ are isomorphic, and from this point on, we will interchangeably refer to vectors and their associated polynomials in reduced form.

# 3 Simple view of Reed-Muller Codes

We now investigate the basics of Reed-Muller codes, including what they are, and a technique for encoding and decoding. The design theory background of Reed-Muller codes is not discussed here, but interested readers may refer to section 4.3 to learn more.

*Definition:* The $r^{th}$ order Reed-Muller code, denoted $\mathcal{RM}(r, m)$, is the set of all polynomials of degree at most $r$ in the ring $\mathcal{R}_m$, as defined in the previous section. Alternatively, through the isomorphism $\psi$, it may be thought of as a subspace of $\mathbb{F}_2^{2^m}$.

*Observation:* The $0^{th}$ order Reed-Muller code $\mathcal{RM}(0, m)$ consists of the monomials $\{0, 1\}$, which is equivalent to the following vectors:

$$\{\underbrace{00\ldots0}_{2^m}, \underbrace{11\ldots1}_{2^m}\}$$

On the other extreme, the $m^{th}$ order Reed-Muller code $\mathcal{RM}(m, m) = \mathcal{R}_m \cong \mathbb{F}_2^{2^m}$.

## 3.1 Encoding and the generator matrix

For the Reed-Muller code $\mathcal{RM}(r, m)$, we define the generator matrix as follows:

$$G_{\mathcal{RM}(r,m)} = \begin{bmatrix} \psi(1) \\ \psi(x_0) \\ \psi(x_1) \\ \vdots \\ \psi(x_{m-1}) \\ \psi(x_0 x_1) \\ \psi(x_0 x_2) \\ \vdots \\ \psi(x_{m-2} x_{m-1}) \\ \psi(x_0 x_1 x_2) \\ \vdots \\ \psi(x_{m-r} x_{m-r+1} \ldots x_{m-1}) \end{bmatrix}$$

*Example:* The generator matrix for $\mathcal{RM}(1, 3)$ is:

$$G_{\mathcal{RM}(1,3)} = \begin{bmatrix} \psi(1) \\ \psi(x_0) \\ \psi(x_1) \\ \psi(x_2) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

*Example:* The generator matrix for $\mathcal{RM}(2, 3)$ is:

$$G_{\mathcal{RM}(2,3)} = \begin{bmatrix} \psi(1) \\ \psi(x_0) \\ \psi(x_1) \\ \psi(x_2) \\ \psi(x_0 x_1) \\ \psi(x_0 x_2) \\ \psi(x_1 x_2) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

*Example:* The generator matrix for $\mathcal{RM}(2,4)$ is:

$$G_{\mathcal{RM}(2,4)} = \begin{bmatrix} \psi(1) \\ \psi(x_0) \\ \psi(x_1) \\ \psi(x_2) \\ \psi(x_3) \\ \psi(x_0x_1) \\ \psi(x_0x_2) \\ \psi(x_0x_3) \\ \psi(x_1x_2) \\ \psi(x_1x_3) \\ \psi(x_2x_3) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

**Theorem 1.** *The matrix $G_{\mathcal{RM}(r,m)}$ has dimension $k \times n$, where:*

$$k = \sum_{i=0}^{r} \binom{m}{i}$$

$$n = 2^m$$

*The dimension of the code is hence $k$.*

*Proof.* The rows of the matrix may be partitioned by the degree of the monomial they represent; for the code $\mathcal{RM}(r,m)$, there are $\binom{m}{0}$ rows corresponding to monomials of degree 0 (0), $\binom{m}{1}$ rows corresponding to monomials of degree 1 $(x_0, x_1, \ldots, x_{m-1})$, $\binom{m}{2}$ rows corresponding to monomials of degree 2 $(x_0x_1, x_0x_2, \ldots, x_{m-2}x_{m-1})$, etc...

Hence, there are:

$$\binom{m}{0} + \binom{m}{1} + \ldots + \binom{m}{r} = \sum_{i=0}^{r} \binom{m}{i}$$

such rows, and thus, the dimension of the code is $k$. It is simple to see that the number of columns is $2^m$ by definition of $\psi$. □

Encoding is then easy; we are given a message $m \in \mathbb{F}_2^k$ and we simply perform the multiplication $m * G_{\mathcal{RM}(r,m)}$ to get our codeword $c$.

*Example:* Assume we want to encode the message $m = 01101001010$ using $\mathcal{RM}(2,4)$. We have:

$$c = m * G_{\mathcal{RM}(2,4)} = 1010111111111010$$

## 3.2 Decoding using majority logic

There are different techniques available for decoding Reed-Muller codes, but one of the most common and most easily implementable is majority logic decoding. We will investigate one form of this technique here.

Before we begin, we will give mention to the error-correcting capabilities of Reed-Muller codes. Given the code $\mathcal{RM}(r, m)$, the distance between any two codewords is $2^{m-r}$. This is not proven here; interested readers can refer to theorem 2 in section 4.2 of this document for a detailed proof of this fact that relies on the recursive definition of Reed-Muller codes.

To correct $\epsilon$ errors, we must have a distance strictly greater than $2\epsilon$. Thus, in the case of Reed-Muller codes, since we have distance $2^{m-r}$, we can correct $\max(0, 2^{m-r-1} - 1)$ errors.

We now look at a basic algorithm for decoding using majority logic. The essential idea behind this technique is that, for each row of the generator matrix, we attempt to determine through a majority vote whether or not that row was employed in the formation of the codeword corresponding to our message.

*Definition:* Let $p$ be any monomial of degree $d$ in $\mathcal{R}_m$ with reduced form $p'$. Then we form the set $\mathcal{J}$ of variables not in $p'$ and their complements. The *characteristic vectors* of $p$ are all vectors corresponding to monomials of degree $m - d$ over the variables of $\mathcal{J}$. *Note:* Any monomial containing a variable and its complement corresponds to the 0 vector (through $\psi$: $\psi(x_i\overline{x_i}) = \psi(x_i) * \overline{\psi(x_i)} = 0$). Since $\psi$ is a bijection and $\psi^{-1}(0) = 0$, this implies that any monomial containing both a variable and its complement is equivalent to the monomial of degree 0. Thus, without loss of generality, we only consider monomials where the variables are distinct (i.e. no variable and its complement appears).

*Example:* If we are working over $\mathcal{RM}(3, 4)$, the characteristic vectors of $x_0x_1x_3$ are the vectors corresponding to the monomials $\{x_2, \overline{x_2}\}$. The characteristic vectors of $x_0x_2$ are the vectors corresponding to the monomials $\{x_1x_3, \overline{x_1}x_3, x_1\overline{x_3}, \overline{x_1x_3}\}$.

The way in which we accomplish this is that we begin at the bottom of our generator matrix and work our way upwards. Our algorithm starts by examining the rows with monomials of degree $r$. We begin by calculating $2^{m-r}$ characteristic vectors for the row, and then we take the dot product of each of these vectors with our received message. If the majority of the dot products are 1, we assume that this row was used in constructing our codeword, and we set the position in our original message vector corresponding to this row to 1. If the majority of the dot products are 0, then we assume that this row was not used in forming our message, and hence, we set the corresponding entry in the original message vector to 0.

After we have run this technique for all rows corresponding to monomials of degree $r$, we take the vector of length $\binom{m}{r}$ corresponding to the portion of the message we have just calculated, and we multiply it by the $\binom{m}{r}$ rows of our generator matrix that we have just considered. This gives us a vector $s$ of length $n$. We add $s$ to our received message, and proceed recursively on the rows corresponding to monomials of degree $r - 1$.

This is best clarified by an example.

*Example:* We decode the message $u = 00110110$ in $\mathcal{RM}(2, 3)$. The generator matrix for this code can be found in the preceeding subsection.

As per the algorithm, we begin from the bottom of the matrix up, first considering the rows corresponding to monomials of degree $r = 2$.

Row $x_1x_2$ has characteristic vectors $x_0$, $\overline{x_0}$.

$$\left.\begin{array}{rcl} u \cdot \underline{\psi(x_0)} & = & 0 \\ u \cdot \overline{\psi(x_0)} & = & 0 \end{array}\right\} \rightarrow m = \_\_\_\_\_ 0$$

Row $x_0x_2$ has characteristic vectors $x_1$, $\overline{x_1}$.

$$\left.\begin{array}{rcl} u \cdot \underline{\psi(x_1)} & = & 1 \\ u \cdot \overline{\psi(x_1)} & = & 1 \end{array}\right\} \rightarrow m = \_\_\_\_\_ 1\ 0$$

Row $x_0x_1$ has characteristic vectors $x_2$, $\overline{x_2}$.

$$\left.\begin{array}{rcl} u \cdot \underline{\psi(x_2)} & = & 0 \\ u \cdot \overline{\psi(x_2)} & = & 0 \end{array}\right\} \rightarrow m = \_\_\_\_ 0\ 1\ 0$$

We have completed processing the rows corresponding to monomials of degree $r = 2$, so we compute $s$:

$$s = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

We add $s$ to $u$ to get $s+u = 10010110$, and we proceed to process the rows corresponding to monomials of degree $r - 1 = 1$.

Row $x_2$ has characteristic vectors $x_0x_1$, $\overline{x_0}x_1$, $x_0\overline{x_1}$, $\overline{x_0x_1}$.

$$\left.\begin{array}{rcl} u \cdot (\underline{\psi(x_0)} * \psi(x_1)) & = & 1 \\ u \cdot (\overline{\psi(x_0)} * \underline{\psi(x_1)}) & = & 1 \\ u \cdot (\underline{\psi(x_0)} * \overline{\psi(x_1)}) & = & 1 \\ u \cdot (\overline{\psi(x_0)} * \overline{\psi(x_1)}) & = & 1 \end{array}\right\} \rightarrow m = \_\_\_ 1\ 0\ 1\ 0$$

We continue to proceed in this fashion, to discover that the original message was $v = 0111010$ (we can trivially check that this is correct by verifying that $v \cdot G_{\mathcal{RM}(2,3)} = u$). In this case, $u$ had no errors (indeed, $\mathcal{RM}(2,3)$ can correct 0 errors, which makes it a poor choice of code, but it will suffice for the purposes of this example).

# 4 Different constructions for $\mathcal{RM}(r, m)$

The isomorphism $\psi$, as mentioned above, is not unique. In fact, we can derive many such isomorphisms, and our generator matrix $G_{\mathcal{RM}(r,m)}$ is dependent on which of these isomorphisms that we choose.

## 4.1 An alternate view of $\psi$

Another way that we can think of $\psi$ is as follows. Consider the ring $\mathcal{R}_m$ and the vector space $\mathbb{F}_2^m$. We can consider the vectors in $\mathbb{F}_2^m$ simply as binary representations of the elements in $\mathbb{Z}_{2^m}$, and thus we can derive a natural ordering from this observation.

*Example:* Suppose we are considering $\mathbb{F}_2^3$. Our natural ordering for the vectors in this vector space is as follows:

$$(000, 001, 010, 011, 100, 101, 110, 111)$$

Let $\mathcal{S}$ be the set of all monomials in $\mathcal{R}_m$. Let $p \in \mathcal{S}$ be a Boolean monomial with reduced form $p'$ with the form $p' = x_{i_1}x_{i_2}\ldots x_{i_r}$ ($i_j \in \mathbb{Z}_m$, $i_j = i_k \rightarrow j = k$, $0 \leq r \leq m$). We consider the function $\alpha$, defined as follows (where $\mathcal{P}(\mathcal{X})$ is the power set of $\mathcal{X}$):

$$\alpha : \mathcal{S} \rightarrow \mathcal{P}(\mathbb{Z}_m)$$

$$p \equiv p' = x_{i_1}x_{i_2}\ldots x_{i_r} \mapsto \begin{cases} \emptyset & : \text{ if } r = 0 \\ \{i_1, i_2, \ldots, i_r\} & : \text{ otherwise} \end{cases}$$

We then define a class of certain functions. Given $\mathcal{T} \in \mathcal{P}(\mathbb{Z}_m)$, we define the following function:

$$f_{\mathcal{T}} : \mathbb{F}_2^m \rightarrow \mathbb{F}_2$$

$$(x_0, x_1, \ldots, x_{m-1}) \mapsto \begin{cases} \prod_{i \in \mathcal{T}}(x_i + 1) & : \text{ if } \mathcal{T} \neq \emptyset \\ 1 & : \text{ if } \mathcal{T} = \emptyset \end{cases}$$

*Example:* Evaluate $f_{\{0,2\}}(1001)$ and $f_{\{0,2\}}(0101)$.
We begin by noting that, by definition, $f_{\{0,2\}}(x_0, x_1, x_2, x_3) = (x_0 + 1)\cdot(x_2 + 1)$.

$$f_{\{0,2\}}(1011) = (1 + 1) \cdot (0 + 1) = 0 \cdot 1 = 0$$

$$f_{\{0,2\}}(0101) = (0 + 1) \cdot (0 + 1) = 1 \cdot 1 = 1$$

Then we consider the function $\beta$, defined as follows:

$$\beta : \mathcal{S} \rightarrow \mathbb{F}_2^{2^m}$$

$$x \mapsto (f_{\alpha(x)}(0_b), f_{\alpha(x)}(1_b), f_{\alpha(x)}(2_b), \ldots, f_{\alpha(x)}((m-1)_b))$$

where $n_b$ is the binary representation of $n$ as a vector in $\mathbb{F}_2^m$.
Now we have all the tools we need in order to fully define $\psi$ within this new framework, which we will see is equivalent to the old framework in a moment. We first recognize that any element of $q \in \mathcal{R}_m$ can be written as a sum of monomials, i.e. $q = m_1 + m_2 + \ldots + m_r$, where $m_i \in \mathcal{S}$, $m_i = m_j \rightarrow i = j$, $0 \leq r \leq 2^m$. We then define $\psi$ as follows:

$$\psi : \mathcal{R}_m \rightarrow \mathbb{F}_2^{2^m}$$

$$x = m_1 + m_2 + \ldots + m_r \mapsto \begin{cases} \beta(0) & : \quad x = 0 \\ \beta(m_1) + \beta(m_2) + \ldots + \beta(m_r) & : \quad x \in \mathcal{R}_m \setminus \{0\} \end{cases}$$

This more precise definition of $\psi$ is consistent with our previous definition of $\psi$, and this is best shown with an example.

*Example:* Using $\psi$ as defined above, we construct $G_{\mathcal{RM}(2,3)}$. The rows of $G_{\mathcal{RM}(2,3)}$ correspond to the monomials $\{1, x_0, x_1, x_2, x_0x_1, x_0x_2, x_1x_2\}$, and to get each row, we evaluate $\psi$ for each monomial.

$$\begin{array}{lcllcl}
\psi(1) & = & \beta(1) & = & (f_\emptyset(000), f_\emptyset(001), \ldots, f_\emptyset(111)) & = & 11111111 \\
\psi(x_0) & = & \beta(x_0) & = & (f_{\{0\}}(000), f_{\{0\}}(001), \ldots, f_{\{0\}}(111)) & = & 11110000 \\
\psi(x_1) & = & \beta(x_1) & = & (f_{\{1\}}(000), f_{\{1\}}(001), \ldots, f_{\{1\}}(111)) & = & 11001100 \\
\psi(x_2) & = & \beta(x_2) & = & (f_{\{2\}}(000), f_{\{2\}}(001), \ldots, f_{\{2\}}(111)) & = & 10101010 \\
\psi(x_0x_1) & = & \beta(x_0x_1) & = & (f_{\{0,1\}}(000), f_{\{0,1\}}(001), \ldots, f_{\{0,1\}}(111)) & = & 11000000 \\
\psi(x_0x_2) & = & \beta(x_0x_2) & = & (f_{\{0,2\}}(000), f_{\{0,2\}}(001), \ldots, f_{\{0,2\}}(111)) & = & 10100000 \\
\psi(x_1x_2) & = & \beta(x_1x_2) & = & (f_{\{1,2\}}(000), f_{\{1,2\}}(001), \ldots, f_{\{1,2\}}(111)) & = & 10001000
\end{array}$$

As we can see, these vectors form the rows of $G_{\mathcal{RM}(2,3)}$ as defined in section 2.

## 4.2 Recursive definition of $\mathcal{RM}(r, m)$

As mentioned in the introduction, it is interesting to note that Reed-Muller codes can be defined recursively. However, the standard recursive definition of $\mathcal{RM}(r, m)$ will result in a generator matrix that differs from the one defined by $\psi$. This is of no consequence, provided that both encoding and decoding are done using the same matrix.

Here is the standard recursive definition for $\mathcal{RM}(r, m)$ ([4]):

1. $\mathcal{RM}(0, m) = \{\underbrace{00\ldots0}_{2^m}\}$
2. $\mathcal{RM}(m, m) = \mathbb{F}_2^{2^m}$
3. $\mathcal{RM}(r, m) = \{(x, x + y) \mid x \in \mathcal{RM}(r, m - 1), y \in \mathcal{RM}(r - 1, m - 1)\}, 0 < r < m$

*Example:* Using the recursive definition of Reed-Muller codes, we find $\mathcal{RM}(1, 2)$:

$$\begin{array}{rcl}
\mathcal{RM}(1,2) & = & \{(x, x + y) \mid x \in \mathcal{RM}(1, 1), y \in \mathcal{RM}(0, 1)\} \\
& = & \{(x, x + y) \mid x \in \{00, 01, 10, 11\}, y \in \{00, 11\}\} \\
& = & \{0000, 0011, 0101, 0110, 1010, 1001, 1111, 1100\}
\end{array}$$

This recursive definition of the codes translates to a recursive definition of the generator matrix, as follows:

$$G_{\mathcal{RM}(r,m)} = \begin{bmatrix} G_{\mathcal{RM}(r-1,m)} & G_{\mathcal{RM}(r-1,m)} \\ 0 & G_{\mathcal{RM}(r-1,m-1)} \end{bmatrix}$$

where we consider the following base cases:

$$G_{\mathcal{RM}(0,m)} = \{\underbrace{11\ldots1}_{2^m}\}$$

$$G_{\mathcal{RM}(m,m)} = \begin{bmatrix} G_{\mathcal{RM}(m-1,m)} \\ \underbrace{00\ldots0}_{2^m-1}1 \end{bmatrix}$$

*Example:* Using the recursive definition of generator matrices for Reed-Muller codes, we construct $G_{\mathcal{RM}(2,3)}$:

$$\begin{aligned}
G_{\mathcal{RM}(2,3)} &= \begin{bmatrix} G_{\mathcal{RM}(2,2)} & G_{\mathcal{RM}(2,2)} \\ 0 & G_{\mathcal{RM}(1,2)} \end{bmatrix} \\
&= \begin{bmatrix} G_{\mathcal{RM}}(1,2) & G_{\mathcal{RM}}(1,2) \\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1 & \\ 0\ 0\ 0\ 0 & G_{\mathcal{RM}(1,1)} \quad G_{\mathcal{RM}(1,1)} \\ 0\ 0\ 0\ 0\ 0\ 0 & G_{\mathcal{RM}(0,1)} \end{bmatrix} \\
&= \begin{bmatrix} G_{\mathcal{RM}(1,1)} & G_{\mathcal{RM}(1,1)} & G_{\mathcal{RM}(1,1)} & G_{\mathcal{RM}(1,1)} \\ 0 & G_{\mathcal{RM}(0,1)} & 0 & 0\ G_{\mathcal{RM}(0,1)} \\ 0 & 0 & 1 & 0\ 0\ 0\ 1 \\ 0 & 0 & 0 & G_{\mathcal{RM}(1,1)} \quad G_{\mathcal{RM}(1,1)} \\ 0 & 0 & 0 & 0\ 0\ G_{\mathcal{RM}(0,1)} \end{bmatrix} \\
&= \begin{bmatrix} 1&1&1&1&1&1&1&1 \\ 0&1&0&1&0&1&0&1 \\ 0&0&1&1&0&0&1&1 \\ 0&0&0&1&0&0&0&1 \\ 0&0&0&0&1&1&1&1 \\ 0&0&0&0&0&1&0&1 \\ 0&0&0&0&0&0&1&1 \end{bmatrix}
\end{aligned}$$

One interesting property of the recursive definition is that it facilitates proving certain properties of Reed-Muller codes; recursion allows us to derive inductive proofs that are, in these cases, much simpler than non-inductive proofs.

**Theorem 2.** *The distance of $\mathcal{RM}(r,m)$ is $2^{m-r}$.*

*Proof.* (from [4]) Let $wt(x)$ denote the weight of x in the code.

We proceed by induction on $r$. By definition:

$$\mathcal{RM}(r,m) = \{(x, x+y) \mid x \in \mathcal{RM}(r,m-1), y \in \mathcal{RM}(r-1,m-1)\}$$

It is easy to see that $\mathcal{RM}(r-1,m-1) \subseteq \mathcal{RM}(r,m-1)$. By combining these facts, we observe that $x+y \in \mathcal{RM}(r,m-1)$. We now consider two distinct cases.

1. $x \neq y$

   By the inductive hypothesis, the weight of $x + y$ must be at least $2^{(m-1)-r}$. We also have that the weight of $x$ is at least $2^{(m-1)-r}$. Thus:

$$wt(x, x + y) = wt(x + y) + wt(x) \geq 2 \cdot 2^{m-r-1} = 2^{m-r}$$

2. $x = y$

   We then have that $(x, x+y) = (x, 0) = (y, 0)$, and we know that $y \in \mathcal{RM}(r-1, m-1)$, so we have that:

$$wt(x, x + y) = wt(y, 0) = wt(y) \geq 2^{(m-1)-(r-1)} = 2^{m-r}$$

Thus, the distance of the code is $2^{m-r}$. □

## 4.3 Affine geometries and Reed-Muller codes

Reed-Muller codes, like many other codes, have links in design theory, and in particular, to designs from affine geometries. We will briefly investigate the link here. Readers are assumed to have some basic familiarity with designs, although definitions are given where necessary.

### 4.3.1 Affine geometries and their designs

*Definition:* Let $\mathcal{F}$ be any field, and let $\mathcal{V}$ be a vector space over $\mathcal{F}$ of dimension $n$. Then, the *affine geometry of* $\mathcal{V}$, denoted $\mathcal{AG}(\mathcal{V})$, is the set of all cosets over all subspaces of $\mathcal{V}$. More precisely:

$$\mathcal{AG}(\mathcal{V}) = \{x + \mathcal{U} \mid x \in \mathcal{V},\ \mathcal{U} \text{ is a subspace of } \mathcal{V}\}$$

*Definition:* For any coset $x + \mathcal{U}$, if $\mathcal{U}$ is a subspace of $\mathcal{V}$ with dimension $s$, we say that the coset $x + \mathcal{U}$ has dimension $s$. Then, we call $x + \mathcal{U}$ an *$s$-flat* of $\mathcal{AG}(\mathcal{V})$.

In $\mathcal{AG}(\mathcal{V})$, the points correspond to the 0-flats. The 1-flats are the lines and the 2-flats the planes, with incidence defined by containment, as we will see in an example in a moment.

*Example:* We consider the field $\mathcal{F} = \mathbb{F}_2$ and the vector space $\mathcal{V} = \mathbb{F}_2^3$. We then determine the subspaces of $\mathcal{V}$, partition them by dimension, and derive the cosets. For brevity, we write vectors of $\mathcal{V}$ as 0-1 strings.
The subspace of dimension 0 is:

$$\{000\}$$

The cosets over this subspace give us the 0-flats, or the points, of our geometry:

$$\begin{array}{cccc} \{000\} & \{001\} & \{010\} & \{011\} \\ \{100\} & \{101\} & \{110\} & \{111\} \end{array}$$

The subspaces of dimension 1 are:

$$\begin{array}{cccc} \{000,001\} & \{000,010\} & \{000,011\} & \{000,100\} \\ \{000,101\} & \{000,110\} & \{000,111\} & \end{array}$$

The cosets over these subspaces give us the 1-flats, or lines, of our geometry:

$$\begin{array}{cccc} \{000,001\} & \{000,010\} & \{000,011\} & \{000,100\} \\ \{000,101\} & \{000,110\} & \{000,111\} & \{001,011\} \\ \{001,010\} & \{001,101\} & \{001,100\} & \{001,111\} \\ \{001,110\} & \{010,011\} & \{010,110\} & \{010,111\} \\ \{010,100\} & \{010,101\} & \{011,111\} & \{011,110\} \\ \{011,101\} & \{011,100\} & \{100,101\} & \{100,110\} \\ \{100,111\} & \{101,111\} & \{101,110\} & \{110,111\} \end{array}$$

The subspaces of dimension 2 are:

$$\begin{array}{cccc} \{000,001,010,011\} & \{000,001,100,101\} & \{000,001,110,111\} & \{000,010,100,110\} \\ \{000,010,101,111\} & \{000,011,100,111\} & \{000,011,101,110\} & \end{array}$$

The cosets over these subspaces give us the 2-flats, or planes, of our geometry:

$$\begin{array}{cccc} \{000,001,010,011\} & \{000,001,100,101\} & \{000,001,110,111\} & \{000,010,100,110\} \\ \{000,010,101,111\} & \{000,011,100,111\} & \{000,011,101,110\} & \{100,101,110,111\} \\ \{010,011,110,111\} & \{010,011,100,101\} & \{011,011,101,111\} & \{001,011,100,110\} \\ \{001,010,101,110\} & \{001,010,100,111\} & & \end{array}$$

Trivially, the only subspace of $\mathcal{V}$ of dimension 3 is $\mathcal{V}$, and the only coset derivable from this subspace is $\mathcal{V}$.

We also note that if, for an affine geometry $\mathcal{AG}(\mathcal{V})$, we take the set of 0-flats and the set of $s$-flats (with $s > 0$) together, they form a *balanced incomplete block design*, with the set of 0-flats as the points, and the set of $s$-flats as the blocks.

*Example:* In the previous example, we notice that the set of 0-flats as the points with the set of 1-flats as the blocks forms a 2-$(8, 2, 1)$-BIBD. If we take the set of 0-flats as our points and the set of 2-flats as our blocks, we have a 3-$(8, 4, 1)$-BIBD.

In the next section, we briefly examine the links between these designs and Reed-Muller codes.

### 4.3.2 Reed-Muller codes and designs

Reed-Muller codes can be thought of as the designs of an affine geometry; the link between the two is briefly explored here with an example. Interested readers should refer to [1] for a more detailed examination of the ties between design theory and coding theory.

**Theorem 3.** *$\mathcal{RM}(r,m)$ is the binary code of the design of points and $(m-r)$-flats of $\mathcal{AG}(\mathbb{F}_2^m)$.*

We do not prove this theorem, but it simply requires showing that the incidence vectors of the $(m-r)$-flats of $\mathcal{AG}(\mathbb{F}_2^m)$ span $\mathcal{RM}(r,m)$. A detailed proof may be found in [1].

*Example:* We demonstrate how $\mathcal{RM}(1,3)$ is the binary code of the design of points and 2-flats of $\mathcal{AG}(\mathbb{F}_2^3)$. We have already investigated $\mathcal{AG}(\mathbb{F}_2^3)$ in the previous section, and we will rely on the results we obtained there.

We examine how the incidence vectors of the plane equations corresponding to the 2-flats of $\mathcal{AG}(\mathbb{F}_2^m)$ are actually codewords in our design. Recall that $G_{\mathcal{RM}(1,3)}$, using the isomorphism $\psi$, is as follows:

$$G_{\mathcal{RM}(1,3)} = \begin{bmatrix} \psi(1) \\ \psi(x_0) \\ \psi(x_1) \\ \psi(x_2) \end{bmatrix}$$

Then it is easy to see that the 16 messages over $\mathbb{F}_2^4$ correspond to the codeword vectors associated with the following equations:

| $x \in \mathbb{F}_2^4$ | $x \cdot G_{\mathcal{RM}(1,3)}$ | $x \in \mathbb{F}_2^4$ | $x \cdot G_{\mathcal{RM}(1,3)}$ |
|---|---|---|---|
| 0000 | $0$ | 1000 | $1$ |
| 0001 | $x_2$ | 1001 | $1 + x_2$ |
| 0010 | $x_1$ | 1010 | $1 + x_1$ |
| 0011 | $x_1 + x_2$ | 1011 | $1 + x_1 + x_2$ |
| 0100 | $x_0$ | 1100 | $1 + x_0$ |
| 0101 | $x_0 + x_2$ | 1101 | $1 + x_0 + x_2$ |
| 0110 | $x_0 + x_1$ | 1110 | $1 + x_0 + x_1$ |
| 0111 | $x_0 + x_1 + x_2$ | 1111 | $1 + x_0 + x_1 + x_2$ |

Now we examine our design; each block corresponds to a plane, and each plane has an

incidence vector, as per the following table:

| Block | Plane eqn | Inc vector |
|---|---|---|
| $001, 010, 100, 111$ | $1 + X_0 + X_1 + X_2 = 0$ | $0$ |
| $000, 011, 101, 110$ | $X_0 + X_1 + X_2 = 0$ | $1$ |
| $010, 011, 100, 101$ | $1 + X_0 + X_1 = 0$ | $x_2$ |
| $000, 001, 110, 111$ | $X_0 + X_1 = 0$ | $1 + x_2$ |
| $001, 011, 100, 110$ | $1 + X_0 + X_2 = 0$ | $x_1$ |
| $000, 010, 101, 111$ | $X_0 + X_2 = 0$ | $1 + x_1$ |
| $100, 101, 110, 111$ | $1 + X_0 = 0$ | $x_1 + x_2$ |
| $000, 001, 010, 011$ | $X_0 = 0$ | $1 + x_1 + x_2$ |
| $001, 010, 101, 110$ | $1 + X_1 + X_2 = 0$ | $x_0$ |
| $000, 011, 100, 111$ | $X_1 + X_2 = 0$ | $1 + x_0$ |
| $010, 011, 110, 111$ | $1 + X_1 = 0$ | $x_0 + x_2$ |
| $000, 001, 100, 101$ | $X_1 = 0$ | $1 + x_0 + x_2$ |
| $001, 011, 101, 111$ | $1 + X_2 = 0$ | $x_0 + x_1$ |
| $000, 010, 100, 110$ | $X_2 = 0$ | $1 + x_0 + x_1$ |

The vectors 0000 and 1111 in $\mathbb{F}_2^4$ do not have planes associated with their codewords; however, their associated geometric structures may be formed by linear combinations of the planes of the others.

Thus, Reed-Muller codes can be associated with the designs of affine geometries, and indeed, several majority-logic decoding techniques have arisen because of this association.

# A Implementation of Reed-Muller encoding and decoding

Using the techniques found in section 3, I have implemented two command-line applications to perform encoding and decoding of Reed-Muller codes.

These applications have been coded in ANSI C and hence, should compile on any platform that has an available C compiler. The source code is available on my personal web page, at

```
http://www.site.uottawa.ca/~raaphors
```

## A.1 Encoding using `rmencode`

Encoding is done with the `rmencode` application, which is run in the following way:

```
rmencode r m vector1 [vector2 [vector3 [...]]]
```

where vectors are elements of $\mathbb{F}_2^k$, represented by 0-1 strings.

Here are some examples of the encoding process:

```
~ -> ./rmencode 2 4 01101001010
1010111111111010
~ -> ./rmencode 2 4 00000000000
0000000000000000
~ -> ./rmencode 2 4 11111111111
1110100010000001
~ -> ./rmencode 0 3 0
00000000
~ -> ./rmencode 0 3 1
11111111
~ -> ./rmencode 3 3 00110011
01101110
~ -> ./rmencode 3 3 00110010
11101110
~ ->
```

## A.2 Decoding using `rmdecode`

Decoding is performed using the `rmdecode` program, which has the same basic command-line paramaters as `rmencode`:

```
rmdecode r m vector1 [vector2 [vector3 [...]]]
```

However, in this case, of course, the specified vectors must be elements of $\mathbb{F}_2^{2^m}$, written as 0-1 strings.

Here are some examples of the decoding process:

```
~ -> ./rmencode 2 5 1111111111111111
01111110111010001110100010000001
~ -> ./rmdecode 2 4 1010111111111010
01101001010
~ -> ./rmdecode 2 4 1010111011111010
01101001010
~ -> ./rmdecode 2 4 1011111111111010
01101001010
~ -> ./rmdecode 2 5 01111110111010001110100010000001
1111111111111111
~ -> ./rmdecode 2 5 01101110101010001110101010000001
1111111111111111
~ ->
```

Note that in the final example, we add an error vector of length 3 to the encoding of 1111111111111111 (namely 00010000010000000000001000000000), and the program decodes the correct message. This is as we expect in $\mathcal{RM}(2,5)$, which can correct at most $2^{5-2-1}-1 = 3$ errors.

# References


[1] Assmus, Jr., E. F. and Key, J. D. *Designs and their Codes*. Press Syndicate of the University of Cambridge, Cambridge, 1992.

[2] Cameron, P. J. and Van Lint, J. H. *Designs, Graphs, Codes, and their Links*. Cambridge University Press, Cambridge, 1991.

[3] Cooke, Ben. *Reed-Muller Error Correcting Codes*. MIT Undergraduate Journal of Mathematics Volume 1, MIT Department of Mathematics, 1999.

[4] Hankerson, D. R. et al. *Coding Theory and Cryptography: The Essentials*. Marcel Dekker, New York, 1991.

[5] MacWilliams, F. J. and Sloane, N. J. A. *The Theory of Error Correcting Codes*. North-Holland Mathematical Library Volume 16, Elsevier Science B. V., Amsterdam, 1977.