

ECE 7670

Lecture 9 – Reed-Muller codes

Objective: To examine basic attributes of Reed-Muller codes

Reading:

- Chapter 7.

This chapter gets us closest to the concepts of digital design for these digital codes that we will get. We will be looking at functions of Boolean variables. As we know, we can represent these using truth tables. We will m represent the number of inputs. We will list the input variables with v_m being the most significant bit.

Example 1

$$\begin{array}{r}
 \nu_4 = 0000000011111111 \\
 \nu_3 = 0000111100001111 \\
 \nu_2 = 0011001100110011 \\
 \nu_1 = 0101010101010101 \\
 \hline
 f_1 = 0110100101101001 \\
 \hline
 f_2 = 1010101110100100
 \end{array}$$

Without ambiguity, we can represent the functions simply using the bit strings as

$$\mathbf{f}_1 = (0110100101101001)$$

$$\mathbf{f}_2 = (1010101110100100)$$

□

The set of Boolean functions, of which there are 2^{2^m} , forms a vector space V_{2^m} over $GF(2)$.

We will interchange a functional and vector notation. Here are some examples of important functions:

$$\begin{array}{l}
 1 \leftrightarrow \mathbf{1} = 1111111111111111 \\
 v_1 \leftrightarrow \mathbf{v}_1 = 0101010101010101 \\
 v_2 \leftrightarrow \mathbf{v}_2 = 0011001100110011 \\
 v_3 \leftrightarrow \mathbf{v}_3 = 0000111100001111 \\
 v_4 \leftrightarrow \mathbf{v}_4 = 0000000011111111 \\
 v_1 v_2 \leftrightarrow \mathbf{v}_1 \mathbf{v}_2 = 0001000100010001 \\
 v_1 v_2 v_3 v_4 \leftrightarrow \mathbf{v}_1 \mathbf{v}_2 \mathbf{v}_3 \mathbf{v}_4 = 0000000000000001
 \end{array}$$

In the RM codes, we will be interested in monomial Boolean functions of the sort (in functional notation) $v_1 v_2 v_3$ or $v_2 v_4$ or $v_1 v_2 v_3 v_4$. We will let M denote the set of all monomial functions:

$$M = \{1, v_1, v_2, \dots, v_m, v_1 v_2, \dots, v_{m-1} v_m, v_1 v_2 v_3, \dots, v_1 v_2 \cdots v_m\}.$$

These functions are linearly independent, as are their vector representations.

Definition 1 The binary Reed-Muller code $R(r, m)$ of order r and length 2^m consists of all linear combinations of vectors \mathbf{f} associated with Boolean functions f that are monomials of degree $\leq r$ in m variables. □

Example 2 The $R(1, 3)$ code: length $= 2^3 = 8$. The monomials are $\{1, v_1, v_2, v_3\}$, with associated vectors

$$\begin{aligned} \mathbf{1} &= (11111111) \\ v_3 &= (00001111) \\ v_2 &= (00110011) \\ v_1 &= (01010101) \end{aligned}$$

These vectors can be considered as taken as the rows of a generator matrix. This is an $(8, 4, 4)$ code; it is single-error correcting and double-error detecting. This is also the extended Hamming code (obtained by adding an extra parity bit to the $(7, 4)$ Hamming code). \square

Example 3 The $R(2, 4)$ code. The monomials up to degree 2 are

$$\{1, v_1, v_2, v_3, v_4, v_1v_2, v_1v_3, v_1v_4, v_2v_3, v_2v_4, v_3v_4\}$$

Corresponding are the following binary vectors.

$$\begin{aligned} \mathbf{1} &= (1111111111111111) \\ \mathbf{v}_4 &= (0000000111111111) \\ \mathbf{v}_3 &= (0000111100001111) \\ \mathbf{v}_2 &= (0011001100110011) \\ \mathbf{v}_1 &= (0101010101010101) \\ \mathbf{v}_3\mathbf{v}_4 &= (0000000000001111) \\ \mathbf{v}_2\mathbf{v}_4 &= (000000000110011) \\ \mathbf{v}_1\mathbf{v}_4 &= (000000001010101) \\ \mathbf{v}_2\mathbf{v}_3 &= (000001100000011) \\ \mathbf{v}_1\mathbf{v}_3 &= (0000010100000101) \\ \mathbf{v}_1\mathbf{v}_2 &= (0001000100010001) \end{aligned}$$

This is a $(16, 11)$ code, with min. distance 4. \square

In general for an $R(r, m)$ code, we have

$$k = 1 + \binom{m}{1} + \binom{m}{2} + \cdots + \binom{m}{r}.$$

We can recursively construct an $R(r+1, m+1)$ code — twice the length — from an $R(r, m)$ and $R(r+1, m)$ code.

Theorem 1

$R(r+1, m+1) = \{(f, f+g) \text{ for all } f \in R(r+1, m) \text{ and } g \in R(r, m)\}.$

Proof The codewords of $R(r+1, m+1)$ are associated with Boolean functions in $m+1$ variables of degree $\leq r+1$. If $c(v_1, \dots, v_{m+1})$ is such a function, we can write

$$c(v_1, \dots, v_{m+1}) = f(v_1, \dots, v_m) + v_{m+1}g(v_1, \dots, v_m).$$

Then f is a Boolean function in m variables of degree $\leq r+1$ and g has degree $\leq r$, so the corresponding vectors \mathbf{f} and \mathbf{g} are in $R(r+1, m)$ and $R(r, m)$, respectively. \square

Theorem 2

The minimum distance of $R(r, m)$ is 2^{m-r} .

Proof By induction. When $m = 1$ the $R(0, 1)$ code is the length-2 repetition code; in this case $d_{\min} = 2$. The $R(1, 1)$ code has four codewords of length 2; hence $d_{\min} = 1$.

As an inductive hypothesis, assume that up to m and for $0 \leq r \leq m$ the minimum distance is 2^{m-r} . We will show that d_{\min} for $R(r, m+1)$ is 2^{m-r+1} .

Let $\mathbf{f}, \mathbf{f}' \in R(r, m)$, and let $\mathbf{g}, \mathbf{g}' \in R(r - 1, m)$. By the previous theorem, the vectors $\mathbf{c}_1 = (\mathbf{f}, \mathbf{f} + \mathbf{g})$ and $\mathbf{c}_2 = (\mathbf{f}', \mathbf{f}' + \mathbf{g}')$ must be in $R(r, m + 1)$.

If $\mathbf{g} = \mathbf{g}'$ then $d(\mathbf{c}_1, \mathbf{c}_2) = 2d(\mathbf{f}, \mathbf{f}') \geq 2 \cdot 2^{m-r}$. If $\mathbf{g} \neq \mathbf{g}'$ then

$$d(\mathbf{c}_1, \mathbf{c}_2) = w(\mathbf{f} - \mathbf{f}') + w(\mathbf{g} - \mathbf{g}' + \mathbf{f} - \mathbf{f}')$$

Claim: $w(\mathbf{x} + \mathbf{y}) \geq w(\mathbf{x}) - w(\mathbf{y})$. Proof: Let w_{xy} be the number of places in which the nonzero digits of \mathbf{x} and \mathbf{y} overlap. Then $w(\mathbf{x} + \mathbf{y}) = (w(\mathbf{x}) - w_{xy}) + (w(\mathbf{y}) - w_{xy})$. But since $2w(\mathbf{y}) \geq 2w_{xy}$ the result follows.

By this result,

$$d(\mathbf{c}_1, \mathbf{c}_2) \geq w(\mathbf{f} - \mathbf{f}') + w(\mathbf{g} - \mathbf{g}') - w(\mathbf{f} - \mathbf{f}') = w(\mathbf{g} - \mathbf{g}')$$

But $\mathbf{g} - \mathbf{g}' \in R(r - 1, m)$, so that $w(\mathbf{g} - \mathbf{g}') \geq 2^{m-(r-1)} = 2^{m-r+1}$. □

1 The Reed decoding algorithm

The most interesting codes are those for which effective decoding algorithms exist. The decoding algorithms for RM coding rely upon an interesting and powerful idea in coding, the majority vote, more technically known as majority logic decoding. We will demonstrate this first for a $R(2, 4)$ code.

Let us write the generator for this code as follows:

$$G = \begin{bmatrix} \mathbf{1} \\ \mathbf{v}_4 \\ \mathbf{v}_3 \\ \mathbf{v}_2 \\ \mathbf{v}_1 \\ \mathbf{v}_3\mathbf{v}_4 \\ \mathbf{v}_2\mathbf{v}_4 \\ \vdots \\ \mathbf{v}_1\mathbf{v}_2 \end{bmatrix} = \begin{bmatrix} G_0 \\ G_1 \\ G_2 \end{bmatrix}$$

We will write the 11 input bits partitioned to correspond to the rows of this matrix as

$$\mathbf{m} = (m_0, m_4, m_3, m_2, m_1, m_{34}, m_{24}, \dots, m_{12}) = (\mathbf{m}_0, \mathbf{m}_1, \mathbf{m}_2).$$

Thus the bits in \mathbf{m}_0 are associated with the zeroth order term, the \mathbf{m}_1 bits are associated with the first order terms, and the second-order terms are associated with \mathbf{m}_2 . The encoding operation is

$$\mathbf{c} = (c_0, c_1, c_2, \dots, c_{15}) = \mathbf{m}G = [\mathbf{m}_0, \mathbf{m}_1, \mathbf{m}_2] \begin{bmatrix} G_0 \\ G_1 \\ G_2 \end{bmatrix}$$

The general operation of the algorithm is as follows: given a received vector \mathbf{r} , estimates are first obtained for the highest-order block of message bits, \mathbf{m}_2 . Then $\mathbf{m}_2 G_2$ is subtracted off from \mathbf{r} , leaving only lower-order code words. Then the message bits for \mathbf{m}_1 are obtained, which are subtracted, and so forth.

The key to finding the message bits comes from writing *multiple* equations for the same quantity, and taking a majority vote. Observe the following:

$$\begin{aligned} c_0 &= m_0 \\ c_1 &= m_0 + m_1 \\ c_2 &= m_0 + m_2 \\ c_3 &= m_0 + m_1 + m_2 + m_{12}. \end{aligned}$$

Adding these code bits together we obtain

$$c_0 + c_1 + c_2 + c_3 = m_{12}.$$

We can also add together the next four code bits:

$$c_4 + c_5 + c_6 + c_7 = m_{12}$$

and

$$c_8 + c_9 + c_{10} + c_{11} = m_{12}$$

and

$$c_{12} + c_{13} + c_{14} + c_{15} = m_{12}.$$

Now what we actually have to deal with is the received sequence $\mathbf{r} = (r_0, r_1, \dots, r_{15})$. We use this in conjunction with the equations above to obtain four estimates of m_{12} :

$$\begin{aligned}\hat{m}_{12} &= r_0 + r_1 + r_2 + r_3 \\ \hat{m}_{12} &= r_4 + r_5 + r_6 + r_7 \\ \hat{m}_{12} &= r_8 + r_9 + r_{10} + r_{11} \\ \hat{m}_{12} &= r_{12} + r_{13} + r_{14} + r_{15}\end{aligned}$$

Expressions such as this, in which the check sums all yield the same value, are said to be *orthogonal* on the message bit m_{12} (which is a different usage from the vector-space sense). From these four estimates, we determine the value of m_{12} by majority vote: if only one is incorrect we get it right; if two are incorrect, we can at least detect that an error has occurred.

It is similarly possible to set up multiple equations for the other second-order bits $m_{13}, m_{14}, \dots, m_{34}$. Based upon these equations (which I will not write here), we determine an estimate of the entire second-order block:

$$\hat{\mathbf{m}}_2 = (\hat{m}_{34}, \hat{m}_{24}, \hat{m}_{14}, \hat{m}_{23}, \hat{m}_{13}, \hat{m}_{12})$$

We then “peel off” this layer to get

$$\mathbf{r}' = \mathbf{r} - \mathbf{m}_2 G_2.$$

Now we repeat for the first-order bits. We have eight orthogonal check sums on each of the first-order message bits. For example,

$$\begin{array}{ll} m_1 = c_0 + c_1 & m_1 = c_8 + c_9 \\ m_1 = c_2 + c_3 & m_1 = c_{10} + c_{11} \\ m_1 = c_4 + c_5 & m_1 = c_{12} + c_{13} \\ m_1 = c_6 + c_7 & m_1 = c_{14} + c_{15} \end{array}$$

From eight estimates obtained from the received codeword, we estimate m_1 by voting.

Having obtained all the bits in \mathbf{m}_1 , we remove it,

$$\mathbf{r}'' = \mathbf{r}' - \mathbf{m}_1 G_1$$

and look for m_0 . But

$$\mathbf{r}'' = m_0 \mathbf{1} + \mathbf{e},$$

so the parity sums in this case are just the bits r_0 through r_{15} .

Up to this point, we have dealt with the special case of the $R(2, 4)$ code, and we could look at the exhaustive list of generator vectors and at least justify that the parity sums work. But we need to work out a general way to proceed to other codes. This will lead us into some interesting geometric questions.

First, for the codeword $\mathbf{c} = (c_0, c_1, \dots, c_{n-1})$, we associate c_i with the m -tuple P_i that is the complement of the binary equivalent of i . For example, the following applies to $R(2, 4)$.

i	binary	P_i
0	0000	1111
1	0001	1110
2	0010	1101

These m -tuples P_i can be regarded as points in an m -dimensional geometry known as the Euclidean geometry $EG(m, 2)$. In this geometry, we can represent “cubes” by connecting the points P_i those points that differ in exactly one coordinate. In our example, every vertex has four immediate neighbors.

Each codeword from $R(r, m)$ is an *incidence vector* that defines a subspace in $R(r, m)$, in which the nonzero coordinates of the codeword to points lying in the subspace. For example, the codeword $(0110011001100110) \in R(2, 4)$, is an incidence vector for the subspace containing the points $\{P_1, P_2, P_5, P_6, P_9, P_{10}, P_{13}, P_{14}\}$.

Based upon this geometry we can find equations for orthogonal checksums as follows. To find a set of orthogonal checksums that provide an estimate for the k th order message bit m_{i_1, i_2, \dots, i_k} corresponding to the basis vectors $\mathbf{v}_{i_1} \mathbf{v}_{i_2} \cdots \mathbf{v}_{i_k}$:

1. Let S be the subspace of points associated with the incidence vectors $\mathbf{v}_{i_1} \mathbf{v}_{i_2} \cdots \mathbf{v}_{i_k}$.
2. Form the “set difference”

$$j_1, j_2, \dots, j_{m-k} = \{1, 2, \dots, m\} - \{i_1, i_2, \dots, i_k\}.$$

Let T be the subspace of point associated with the incidence vector $\mathbf{v}_{j_1} \mathbf{v}_{j_2} \cdots \mathbf{v}_{j_{m-k}}$. The space T is said to be the “complementary subspace to S ”.

3. The first checksum consists of the sum of the codeword coordinates specified by T .
4. The rest of the codewords are obtained by translating T with respect to the nonzero elements of S .

Whew! Is that cryptic or what?

Example 4 Checksums for $R(2, 4)$. Let us find checksums for $\mathbf{v}_2\mathbf{v}_4 = (0000000000001111)$. The subspace for which $\mathbf{v}_3\mathbf{v}_4$ is an incidence vector contains the

$$S = \{P_{12}, P_{13}, P_{14}, P_{15}\} = \{(0011)(0010)(0001)(0000)\}$$

(Dotted line).

The difference set is

$$\{j_1, j_2\} = \{1, 2, 3, 4\} - \{3, 4\} = \{1, 2\}$$

Then we use $\mathbf{v}_1\mathbf{v}_2 = (0001000100010001)$. This corresponds to the set of points

$$T = \{P_3, P_7, P_{11}, P_{15}\} = \{(1100)(1011)(0100)(0000)\}$$

(The darker region). From this we obtain one checksum

$$m_{34} = c_3 + c_7 + c_{11} + c_{15}.$$

The translations of T by the nonzero elements of S are:

$$\begin{aligned} \text{by } P_{12} = (0011) &\rightarrow \{(1111)(1011)(0111)(0011)\} = \{P_0, P_4, P_8, P_{12}\} \\ \text{by } P_{13} = (0010) &\rightarrow \{(1110)(1010)(0101)(0010)\} = \{P_1, P_5, P_9, P_{13}\} \\ \text{by } P_{14} = (0001) &\rightarrow \{(1101)(1001)(0101)(0001)\} = \{P_2, P_6, P_{10}, P_{14}\} \end{aligned}$$

These correspond to the checksums

$$\begin{aligned} m_{34} &= c_0 + c_4 + c_8 + c_{12} \\ m_{34} &= c_1 + c_5 + c_9 + c_{13} \\ m_{34} &= c_2 + c_6 + c_{10} + c_{14} \end{aligned}$$

Other checksums are obtained similarly. □

2 Algorithms for first-order RM codes

Some efficient algorithms have been developed for first-order $R(1, m)$ codes. Consider the $R(1, 3)$ code generated by

$$\begin{aligned} \mathbf{c} = (c_0, c_1, \dots, c_7) &= \mathbf{mG} = (m_0, m_3, m_2, m_1) \begin{bmatrix} \mathbf{1} \\ \mathbf{v}_3 \\ \mathbf{v}_2 \\ \mathbf{v}_1 \end{bmatrix} \\ &= (m_0, m_3, m_2, m_1) \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \end{aligned}$$

The columns of G are simply the 4-tuples (1000) increasing to (1111) in binary counting order. We can generate these using a simple counter circuit.

Decoding: We can translate the binary received signal \mathbf{r} to a ± 1 sequence by the mapping

$$\mathcal{F}(\mathbf{r}) = F = ((-1)^{r_0}, (-1)^{r_1}, \dots, (-1)^{r_{n-1}}).$$

Define the correlation between two of the sequences as

$$\text{cor}(F, G) = \text{cor}((F_0, F_1, \dots, F_{n-1}), (G_0, G_1, \dots, G_{n-1})) = \sum_{i=0}^{n-1} F_i G_i.$$

Finding the the codeword \mathbf{c} that minimizes the distance $d(\mathbf{r}, \mathbf{c})$ is equivalent to finding the codeword that maximizes the correlation $\text{cor}(\mathcal{F}(\mathbf{r}), \mathcal{F}(\mathbf{c}))$. The operation of computing the correlation can be performed efficiently by means of the Hadamard transform.

Let $\hat{F} = FH_{2^m}$ denote the Hadamard transform of F , where H_{2^m} is the $2^m \times 2^m$ Hadamard matrix. For decoding the $R(1, 3)$ code we would use the Hadamard matrix

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

Observe that the first, second, and fourth columns (starting counting from the zeroth column) are $\mathcal{F}(\mathbf{v}_1)$, $\mathcal{F}(\mathbf{v}_2)$ and $\mathcal{F}(\mathbf{v}_3)$, respectively. The i th column of H_8 , where $i = (i_3 i_2 i_1)$ in binary is the vector

$$\mathcal{F}(\mathbf{c}_i) = \mathcal{F}(i_1 \mathbf{v}_1 + i_2 \mathbf{v}_2 + i_3 \mathbf{v}_3).$$

For example, the fifth column ($5 = (101)$) consists of the vector

$$\mathcal{F}(1 \cdot \mathbf{v}_1 + 0 \cdot \mathbf{v}_2 + 1 \cdot \mathbf{v}_3) = \mathcal{F}((10101010) + (11110000)) = \mathcal{F}(01011010) = (1 - 1 - -1 - 1).$$

The Hadamard transform of $F = \mathcal{F}(\mathbf{r})$ thus computes the correlation of $\mathcal{F}(\mathbf{r})$ with the mappings of all the codewords. We simply find the coordinate in the transform with the largest value and determine the codeword from it.

There is one last trick. The $R(1, m)$ code has 2^{m+1} codewords in it, and we only get 2^m values out of the Hadamard transform just described. However, we note that if we complement each bit, by

$$\mathcal{F}(\mathbf{1} + \mathbf{c}) = \mathcal{F}(\mathbf{1} + c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 + \dots + c_m \mathbf{v}_m)$$

then the correlation is the negative of $\mathcal{F}(\mathbf{c})$. Thus we can look at the sign and on that basis get the other half of the codewords.

1. Given \mathbf{r} , compute $F = \mathcal{F}(\mathbf{r})$.
2. Compute the Hadamard transform $\hat{F} = FH_{2^m}$.
3. Determine the coordinate $\mathbf{a} = (a_m \dots a_2 a_1)$ where \hat{F} has the greatest magnitude (starting count from zero). Let \hat{F}_a denote the correlation value at that coordinate.

4. If \hat{F}_a is negative, then $\mathbf{c} = \mathbf{1} + a_1\mathbf{v}_1 + \cdots + a_m\mathbf{v}_m$. if \hat{F}_a is positive, then $\mathbf{c} = a_1\mathbf{v}_1 + \cdots + a_m\mathbf{v}_m$.

Example 5 We receive $\mathbf{r} = (10000011)$:

1. $F = (-111111 \ - \ -)$
2. $\hat{F} = FH_8 = (2, -2, 2, -2, 2, -2, -6, -2)$.
3. $\hat{F}_a = -6$ (starting count from zero), and $a = 6 = (110)$.
4. The codeword is

$$\mathbf{c} = \mathbf{1} + 1\mathbf{v}_3 + 1\mathbf{v}_2 + 0\mathbf{v}_1 = (11000011).$$

□

Fast algorithms exist for the Hadamard transform, making the decoding operation fairly efficient.