1 Materials and Methods

This section describes the theory for the encoding and Meggitt decoding of a cyclic code. Both encoding and decoding is explained with concrete examples which are also testable with the enclosed MATLAB scripts. Knowledge is obtained through TI-INCO course lessons, the "Essentials of Error-Control Coding" text book by Jorge Castiñeira Moreira and Patrick Guy Farrell and through delivered material about decoding of cyclic codes (chapter 5.5).[1]

1.1 Encoding

When a code is in systematic form the message vector can be directly extracted from the encoded information.
Encoding a message vector into a linear cyclic code, $C_{cyc}(n,k)$, in systematic form can be achieved as follows:
First the message polynomial (in the form of $m(X) = m_0 + m_1X + ... + m_{k-1}X^{k-1}$) should be converted into a vector of the length n. This is attained by multiplying the message vector by X^{n-k}, which results in a shift of the message $n-k$ to the right and forms the right-shifted message polynomial $X^{n-k}m(X)$.
Secondly, the right-shifted message polynomial should be divided by a generator polynomial, $g(X)$, fitting the conditions for being a generator polynomial of a linear cyclic code. The result of this division is a remainder polynomial (redundancy polynomial).
Finally, the code polynomial in systematic form can be obtained by adding the right-shifted message polynomial and the remainder polynomial:

$$c(X) = X^{n-k}m(X) + p(X) \tag{1}$$

The following equations describe the relationships between the right-shifted message polynomial, $X^{n-k}m(X)$, the factor, $q(X)$, the generator polynomial, $g(X)$, and the remainder polynomial, $p(X)$.

$$X^{n-k}m(X) = q(X)g(X) + p(X) \tag{2}$$

$\Leftrightarrow$

$$X^{n-k}m(X) + p(X) = q(X)g(X) \tag{3}$$

1.1.1 Encoding Example

A concrete example with the generator polynomial $g(X) = 1 + X^4 + X^6 + X^7 + X^8$ is used to encode the following message polynomial: $m(X) = 1 + X^2 + X^4 + X^6$, corresponding to $m = [1010101]$. The code is a cyclic code, $C_{cyc}(15,7)$.
First the message is right shifted $n-k$ times.

$$X^{15-7}m(X) = X^8 + X^{10} + X^{12} + X^{14} \tag{4}$$

Find $p(X)$ by taking the remainder from $X^{n-k}m(x)$ divided by $g(X)$.

[1] FiXme Note: Make reference

$$
\begin{array}{cccccc|ccccccccc}
 & & & & & & X^6 & +X^5 & +X^4 & +X^2 & +1 & & & & \\
\hline
X^8 & +X^7 & +X^6 & +X^4 & +1 & & X^{14} & +X^{12} & +X^{10} & +X^8 & & & & & \\
 & & & & & & X^{14} & +X^{13} & +X^{12} & +X^{10} & +X^6 & & & & \\
\hline
 & & & & & & & X^{13} & & +X^8 & +X^6 & & & & \\
 & & & & & & & X^{13} & +X^{12} & +X^{11} & +X^9 & +X^5 & & & \\
\hline
 & & & & & & & & X^{12} & +X^{11} & +X^9 & +X^8 & +X^6 & +X^5 & \\
 & & & & & & & & X^{12} & +X^{11} & +X^{10} & +X^8 & & +X^4 & \\
\hline
 & & & & & & & & & X^{10} & +X^9 & +X^6 & +X^5 & +X^4 & \\
 & & & & & & & & & X^{10} & +X^9 & +X^8 & +X^6 & +X^2 & \\
\hline
 & & & & & & & & & & X^8 & +X^5 & +X^4 & +X^2 & \\
 & & & & & & & & & & X^8 & +X^7 & +X^6 & +X^4 & +1 \\
\hline
 & & & & & p(X) = & & & & & & X^7 & +X^6 & +X^5 & +X^2 & +1
\end{array}
$$

Now $c(X)$ can be calculated:

$$
\begin{aligned}
c(X) &= X^8 m(X) + p(X) && (5)\\
&= 1 + X^2 + X^5 + X^6 + X^7 + X^8 + X^{10} + X^{12} + X^{14} && (6)\\
c &= [101001111010101] && (7)
\end{aligned}
$$

It is easily seen that the encoded vector is in systematic form, hence the message vector corresponds to the last 7 bits.

1.2 Decoding

In general when decoding a received vector, $r(X)$, the syndrome vector, $S(X)$, is needed to find out if any errors have happen. To find this the generator polynomial, $g(X)$, is used. The formula to find the syndrome vector for a received vector is:

$$S(X) = r(X) \; mod \; g(X) \tag{8}$$

Where $S(X)$ is the remainder from the division is done in this way.

$$
\begin{array}{cc|c}
S(X) = & g(X) & \overline{r(X)}
\end{array}
$$

If the syndrome vector is an all zero vector there is no error detected in the received vector. If the syndrome vector is not an all zero vector it can tell where one or more errors are placed. Every possible error is mapped to a specific syndrome. The error capability and the error detection can be found in this way:

$$Error\ Detection = d_{min} - 1 \tag{9}$$

$$Error\ Correction = t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor \tag{10}$$

When the error is found it can be corrected. It is known that

$$r(X) = q(X) \cdot g(x) \oplus S(X) \tag{11}$$

where $q(X)$ is the quotient. When encoding $c(X)$ and combining equation 1 and 3 it is seen that $q(X) \cdot g(X) = c(X)$. It is also known that $S(X)$ is the error therefore $S(X) = e(X)$. With this knowledge equation 11 can be changed to:

$$r(X) = c(X) \oplus e(X) \tag{12}$$

In the decoding process $r(X)$ is known and $e(X)$ is calculated, and $c(X)$ is the desired result. Therefore $c(X)$ can be isolated:

$$c(X) = r(X) \oplus e(X) \tag{13}$$

This means that when the error has been located it can be removed by multiplying it to $r(X)$.

1.2.1 Meggitt decoding

The Meggitt decoder is used to decode any cyclic code. The error patterns can be specified by a decoding table which is made as a lookup table. However this procedure is complex and the decoding circuit tends to grow exponentially with the code length and the number of errors which can be corrected. The Meggitt decoders are useful to correct up to 3 random errors. The Meggitt decoder are shown on Figure 1. It consist of 3 main component:

- Buffer register

  Is buffering the received vector

- Syndrome register

  Containing the specific syndrome for the buffer

- Error pattern detection circuit

  Containing the lookup table with all the error patterns

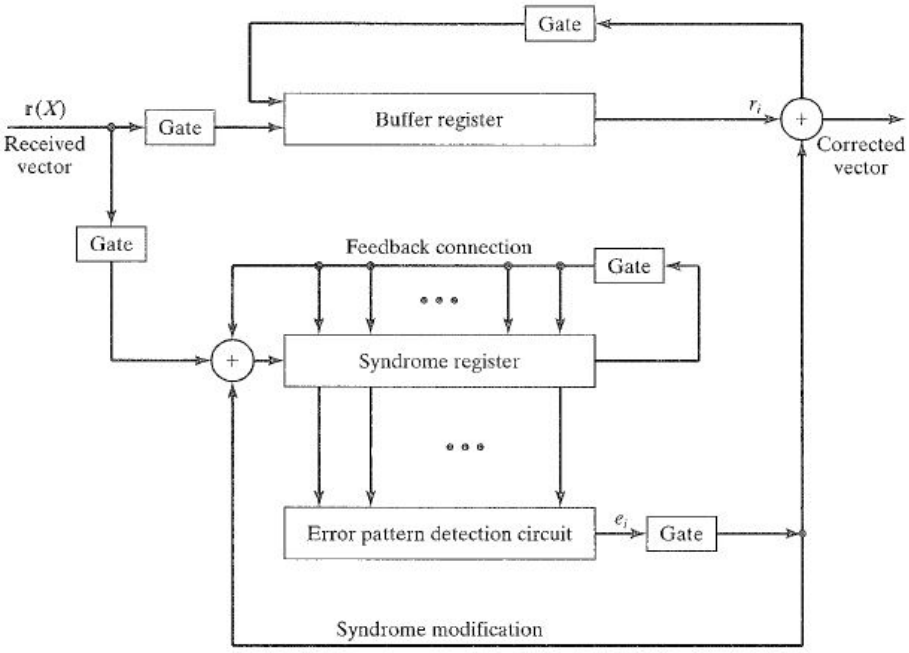

Figure 1: Meggitt decoder overview

Figure 2 shows in more details how the syndrome register could be build together with the error pattern detection circuit. The feedback in the syndrome register is depending on the general polynomial. The number of bits in the syndrome is equal to the highest degree in g(X). The non zero places in the general vector will get the feedback. In this example the generator polynomial is $1 + X + X^3$. Therefore there will be feedback before 1 and X and the feedback is given by X^3. The generator polynomial used in this project is: $g(X) = 1 + X^4 + X^6 + X^7 + X^8$. The feedback in the syndrome register will therefore happen before memory block 1, 5, 7 and 8. [2]

The Meggitt decoder contains of 5 steps:

1. The received vector is shifted into the buffer register and syndrome register. At this point the feedback from the error pattern circuit is not used.

2. The syndrome is check in the error pattern detection circuit. If the syndrome is in the error pattern lookup table there is an error in the last bit in the buffer register and the error pattern detection circuit output will be 1. If the syndrome don't match any error pattern it output will be 0.

3. The rightmost symbol in the buffer register is read out at the same time as the syndrome register is shifted once. If an error has been detected it is feedback to he shifted syndrome register which modify the syndrome. The error is also multiplied to the symbol which has been read out of the buffer.

4. The new syndrome is match the new rightmost symbol in the buffer. Step 2 and 3 is repeated.

5. The received vector is decoded symbol by symbol until the whole receive vector has been read out of the buffer register.

[2] FiXme Note: **See Figure ...**

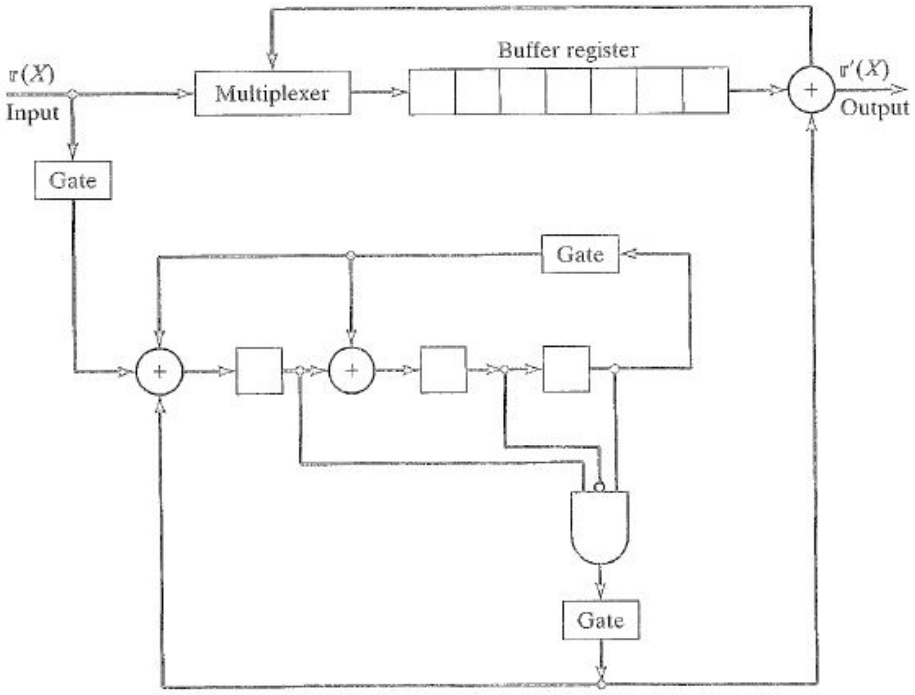

Figure 2: Meggitt decoder details

1.2.2 Meggitt decode example

In this example the codeword $c = [011110001001101]$ is used. An error is made on bit 7 which mean X^6 is flipped. The received vector is: $r = [011110101001101]$. At first r(X) is shifted into the syndrome register. The first 8 step is straight forward because where will be no feedback at the syndrome register is initialized with all zeroes at the beginning.

shift no.	**input**		**syndrome**
$shift_1:$	$input = 1$	$\Rightarrow$	10000000
$shift_2:$	$input = 0$	$\Rightarrow$	01000000
$shift_3:$	$input = 1$	$\Rightarrow$	10100000
$shift_4:$	$input = 1$	$\Rightarrow$	11010000
$shift_5:$	$input = 0$	$\Rightarrow$	01101000
$shift_6:$	$input = 0$	$\Rightarrow$	00110100
$shift_7:$	$input = 1$	$\Rightarrow$	10011010
$shift_8:$	$input = 0$	$\Rightarrow$	01001101

The next 7 shift is it possible to get feedback. The bit coloured light blue is the one which receive feedback.

shift no.	**input**	**old syn**		**new syn**
$shift_9:$	$input = 1$	01010**1**1**0**1	$\Rightarrow$	00101101
$shift_{10}:$	$input = 0$	00**1**01**1**01	$\Rightarrow$	10011101
$shift_{11}:$	$input = 1$	10**0**11**1**01	$\Rightarrow$	01000101
$shift_{12}:$	$input = 1$	01**0**00**1**01	$\Rightarrow$	00101001
$shift_{13}:$	$input = 1$	00**1**01**0**01	$\Rightarrow$	00011111
$shift_{14}:$	$input = 1$	00**0**11**1**11	$\Rightarrow$	00000100
$shift_{15}:$	$input = 0$	00**0**00**1**00	$\Rightarrow$	00000010

After shifted $r(X)$ into the syndrome register the syndrome has a 1 on the 7'th bit, which is the syndrome pattern matching the error pattern for X^6. This bit was flipped in the received vector to introduce an error. An other way to find the syndrome matching $r(X)$ is to make the polynomial division as in Equation 8. SKAL DETTE BEVISES??

The $r(X)$ in the buffer match the $S(X)$ in the syndrome register, but what happens with $S(X)$ when the $r(X)$ is shifted in the buffer? When $r(X)$ is cyclic right shifted it becomes $r^{(1)}(X)$ and has a cosponsoring $S^{(1)}(X)$. This $S^{(1)}(X)$ can be found by multiply it with X and make a polynomial division with $g(X)$ as shown below.

$$S^{(1)}(X) = X \cdot S(X) \; mod \; g(X) \tag{14}$$

The list of shifts below shows how the syndrome is changing when right shifting. The input is no longer from $r(X)$ but is the error output, which is zero when no error has been detected in the last bit. Looking at shift 3 as an example of Equation 14, the syndrome register contains $10001011 \Rightarrow 1 + X^4 + X^6 + X^7$ before shifting. When right shifting it becomes $X + X^5 + X^7 + X^8 = X \cdot S(X)$. Finding $S^{(1)}(X)$ with the polynomial division of $g(X)$ gives $1 + X + X^4 + X^5 + X^6 \Rightarrow 11001110$. This $S^{(1)}(X)$ match the $r^{(1)}(X)$ which is in the buffer after the shift.

shift no.	**input**	**old syn**		**new syn**
$shift_1:$	$input = 0$	00000010	$\Rightarrow$	00000001
$shift_2:$	$input = 0$	00000001	$\Rightarrow$	10001011
$shift_3:$	$input = 0$	10001011	$\Rightarrow$	11001110
$shift_3:$	$input = 0$	11001110	$\Rightarrow$	01100111
$shift_4:$	$input = 0$	01100111	$\Rightarrow$	10111000
$shift_5:$	$input = 0$	10111000	$\Rightarrow$	01011100
$shift_6:$	$input = 0$	01011100	$\Rightarrow$	00101110
$shift_7:$	$input = 0$	00101110	$\Rightarrow$	00010111
$shift_8:$	$input = 1$	00010111	$\Rightarrow$	00000000

After 7 shift 7 the syndrome match the error pattern for one error at the last bit. After the 8'th bit the syndrome becomes a all zero vector, which indicates that there is no more errors in the buffer.

The only error that is interesting is the last error, therefore when correcting 1 error only the error pattern for error in the rightmost symbol is used. In our example the error pattern will be $e(X) = [000000000000001]$. When introducing 2 errors correcting the error pattern is extended to the one shown in list: (INDÆST LINK HER MAIKEN!).

$$syndromeErrorPattern = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{15}$$

Even though the code is looking for 2 error it is still only errors on the rightmost bit in the buffer that are of interest. Therefore the error pattern will contain all possible 2 errors where the one is fixed on the to the right. To find the corresponding syndrome error pattern matching a specific error pattern the error pattern is divided with the general polynomial. This is done for all the interesting error patterns and the syndromes used for matching in the error pattern detection circuit.