A Novel Method for Obtaining the Structure and Corresponding Parity Weight Equations for the RTZ Inputs of RSC Codes: A First Step Towards Interleaver Design

Kwame Ackah Bohulu

November 26, 2020

0.1 Abstract

The knowledege of the distance spectrum, as well as the structure of the message inputs that make up the distance spectrum for a specific Recursive Systematic Convolutional (RSC) code is vital to the design of Turbo Code interleavers. Whiles the distance spectrum of a RSC code can be obtained by calculationg its transfer function, it provides no information about the structure of the message inputs.

In this paper,we present a novel low-complexity method for determining the distance spectrum of any RSC code which has the added benefit of revealing the structure of the message inputs that make up the distance spectrum. With the knowledge of the structure of the message inputs, we derive a general polynomial representation for them based on the weight of the message input after which we go a step further and show how to derive their corresponding parity-weight equations.

Finally, a summary of the structure of the message inputs as well as their corresponding parity weight equations is tabulated for different RSC codes.

0.2 Introduction

The Turbo Code (TC) [**?**], is known for being one of the forward-error correcting codes that come very close to satisfying the Shannon limit for AWGN channels. It was introduced by Claude Berrou in 1993 and steadily gained fame and has been used in many applications including Mobile and Satelite communication networks. The simplest and most common construction of the TC is via the parallel concatenation of two Recursive Systematic Convolutional (RSC) codes (of the same kind) via an interleaver. The reason why the TC has such a good error correcting capability has been attributed to the low multiplicity of its minimum distance codeword and through many years of intensive research in the field, it is common knowledge that this is largely due to the use of the interleaver in the TC construction.

Interleaver design for TCs has been a hot topic for many years and generally, they are grouped into random and deterministic interleavers. Random interleavers determine their order of permutation in a pseudo random manner and therefore require interleaver tables in both the transmitter and the receiver. Even thought TCs made with random interleavers have really good error-correcting capabilities (especially for medium and long frame sizes), the need for interleaver tables imposes huge memory constraints for many practical applications. A notable example of a random interleaver is the S-random interleaver.

On the flip side, deterministic interleavers generate their order of permutation via algorithims and as such can be generated on the fly, killing the need for permutation tables. Popular deterministic interleavers include Quadratic Permutation Polynomial (QPP) interleaver [**?**], Almost Regular Permutation (ARP) Interleaver and the Dithered Relative Prime (DRP) interleaver. Deterministic interleavers also make it possible to perform parallel decoding, once the interleaver meets certain requirements. Given all these benefits, it is a well known fact that in terms on TC error-correcting performace random interleavers always outperform deterministic interleavers, especially for long frame sizes. Another benefit of using deterministic interleavers is the ability to custom design the interleaver to a specific component code to improve the overall error-correcting capability of the TC.

The most common approach to deterministic interleaver design is the minimum free distance (d_{free}) maximization approach, where the interleaver is designed with the aim to maximize the value of d_{free}. This approach whilst simplistic, has produced some good interleavers only after considering higher weight inputs [**?**]. Therfore, using it as a general rule of thumb for all deterministic interleaver design approaches might not be the best, especially when the minimum distance codeword for the component code is generated by an input message with weight greater than 2.

To better design custom interleavers for a specific component code requires deep knowledge of its distance spectrum, more specifically the general structure of the message inputs that make up the distance spectrum. These message inputs are such that they diverge from and then return to the initial stateand are refered to as Return-To-Zero (RTZ) inputs There are many methods available for obtaining the distance spectrum of an RSC code including finding the transfer function of the RSC code [**?**]. However, the transfer function method is not a very helpful tool when it comes to custom interleaver design since aside information about the number of codewords of weight d generated by a message input of weight w, it provides zero information about the structure of the RTZ inputs. As an added downside, the complexity of calculating the transfer function for a given RSC code increases with the number of states.

In this paper,we present a novel alternate method to the Transfer Function whose complexity is independent of the number of states of the component code, and has the added benefit of making known the structure of the RTZ inputs that make up the distance spectrum. With the

knowledge of the structure of the message inputs, we derive a general polynomial representation for them based on the weight of the message input after which we go a step further and derive corresponding parity-weight equations for the codewords they generate. Finally, a summary of the structure of the message inputs as well as their corresponding parity weight equations is tabulated for different RSC codes.

0.3 Preliminaries

binary vectors are represented by bold font and $\dot{\boldsymbol{v}}$ represent an infinite repitition of the vector $\boldsymbol{v}$ whiles $(\boldsymbol{v})_j$ represents the repitition of vector $\boldsymbol{v}$ j times

$\boldsymbol{\phi}$ represent a primitive element in the extended Galois field GF(2^τ) and the vector representation for all elements in GF(2^τ) are written as $\boldsymbol{\phi}_i,\ 0 \leq i \leq 2^\tau$.

The subscript i represents the decimal value of the binary vector, which means $\boldsymbol{\phi}_0$ represents the all zero vector. All addition and multiplication operations are done in GF(2^τ).

The impulse response for a RSC code is given by $\boldsymbol{\theta}$ and the cycle of the RSC code and the associated cycle length is given by $\boldsymbol{\psi}$ and τ respectively. $\boldsymbol{\theta},\ \boldsymbol{\psi}$ and will be properly defined in the next section.

0.4 Recursive Systematic Convolutional Codes:Review

An (n, k) RSC code is a convolutional code generated by using feedback shift registers which has its input bits as part of the codeword. At each time instant it receives an input of k bits and outputs n bits. The output bits are determined by the generator function, which may be written in polynomial notation as $\left[1 \ \frac{f(x)}{g(x)}\right]$. Given the systematic nature of the RSC code, we only focus on the parity part of the RSC code and write the generator function simply as $\left[\frac{f(x)}{g(x)}\right]$ where $f(x)$ and $g(x)$ represent the feedfoward and feedback connections of the RSC encoder.

It should be noted that the prescence of the feedback loop means that the code produced will have an infinite-length impulse response denoted $\boldsymbol{\theta}$. $\boldsymbol{\theta}$ is unique for each RSC code and it is the parity output when the input $(1\ 0\ 0\ 0\ 0\ \cdots)$ is fed into the RSC encoder. Without having to feed the input into the encoder, we can find the impulse response as $\boldsymbol{\theta} = f(x)g^{-1}(x)$

Within $\boldsymbol{\theta}$ is a vector $\boldsymbol{\psi}$ (the cycle) that is repeated infinitely and the length of that cycle is represented by τ. With the impulse response it is possible to calculate the parity weight generated by any input message. This is done by noting that each input message is a summation of shifted versions of the input $(1\ 0\ 0\ 0\ 0\ \cdots)$ and therefore its parity-bit sequence is also a summation of shifted versions of the impulse response. This knowledge will be useful in the derivation of the parity weight equation for the RTZ inputs of a given RSC code.

Similar to any code, the minimum distance ($d_{\min}$) of the RSC code also determines its error-correcting capability. With the aid of the distance spectrum, it is possible to determine $d_{\min}$ as well as its multiplicity. The most common way to find the distance spectrum is via the transfer function of the RSC code. The transfer function enumerates all the paths that diverge from and then return to the initial state [**?**], ie the RTZ inputs. In other words, the distance spectrum provides information about the number of codewords of weight d generated by an RTZ input of weight w. Combined with the knowledge that all RTZ inputs are divisible by $G(D)$ [**?**], we introduce a novel method for determining what we refer to as the *input structure distance spectrum* of any RSC code in the next section. This version of the distance spectrum is a better tool for interleaver design compared to the regular distance spectrum.

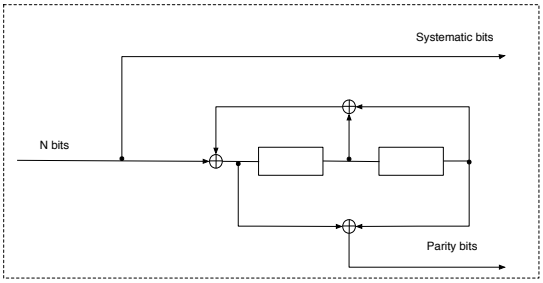

Figure 0-1: $[\frac{1+x^2}{1+x+x^2}]$ RSC Encoder

A RSC encoder is shown in Figure 0-1 with $k = 1$ and $n = 2$. Its generator function is given by $[\frac{1+x^2}{1+x+x^2}]$ which may be written as 5/7 in octal form where 5 and 7 correspond to the numerator and denomenator of the generator function respectively. For the 5/7 RSC code, $\boldsymbol{\theta} = (1\ 1\ 1\ 0\ 1\ 1\ 0\ 1\ 1\ 0\ \cdots)$ which may be written in terms of the elements of GF(8) as $\boldsymbol{\phi}_7\ \dot{\boldsymbol{\phi}}_3$. The corresponding cycle $\boldsymbol{\psi} = \boldsymbol{\phi}_6$ with a cycle length of $\tau = 3$. Moving foward all examples and discussions relating to RSC codes will be done using the 5/7 RSC code unless otherwise stated.

0.5 Input-Structure Distance Spectrum

The regular distance spectrum is an insufficient tool where it relates to interleaver design. In this section, we present a novel method that generates what we refered to as the input-structure distance spectrum. It is the distance spectrum with the structure of the RTZ inputs revealed making it a very useful tool for interleaver design.

For convinience sake we will switch to polynomial representation, where $g(x)$ and $f(x)$ represent the feedback and feedforward connections of the RSC code respectively. Also $b(x)$ and $h(x)$ are the polynomial representations of the input message and its corresponding parity-bit sequence respectively.

This method is simple and relies of the fact that a message input $b(x)$ is an RTZ input if

$$b(x) \bmod g(x) = 0 \tag{0-1}$$

Which means that we may rewrite $b(x)$ as $b(x) = a(x)g(x)$, where $a(x)$ is the polynomial representation of any binary vector or arbitrary length.

$h(x)$ is simply calculated as

$$h(x) = a(x)f(x) \tag{0-2}$$

With the above equations, finding the input-structure distance spectrum for an input message of length N is as simple as using (0-1) or (0-2) (or both) for all values of $a(x)$, $x^0 \in a(x), a(x) \in GF(2^N)$. The condition $x^0 \in a(x)$ ensures that there is no repitition of $b(x)$ or $h(x)$

For $N = 16$ we find all $b(x)$ and $h(x)$ for which $w_H(\mathbf{h}) = 2$ and $w_H(\mathbf{h}) = 4$. The results are shown in Table 1 and 2 respectively . For $w_H(\mathbf{h}) = 4$, only the first 10 results are shown. Also 3 shows all $b(x)$ and $h(x)$ that produce a codeword weight $w_H(\boldsymbol{c}) \leq 8$ for $N = 32$.

Table 1: codewords with parity bit sequence weight $w_H(\mathbf{h}) = 2$

$w_H(\mathbf{b})$	$b(x)$	$h(x)$
3	$1 + x + x^2$	$1 + x^2$
4	$1 + x + x^3 + x^4$	$1 + x^4$
5	$1 + x + x^3 + x^5 + x^6$	$1 + x^6$
6	$1 + x + x^3 + x^5 + x^7 + x^8$	$1 + x^8$
7	$1 + x + x^3 + x^5 + x^7 + x^9 + x^{10}$	$1 + x^{10}$
8	$1 + x + x^3 + x^5 + x^7 + x^9 + x^{11} + x^{12}$	$1 + x^{12}$
9	$1 + x + x^3 + x^5 + x^7 + x^9 + x^{11} + x^{13} + x^{14}$	$1 + x^{14}$

Table 2: codewords with parity bit sequence weight $w_H(\mathbf{h}) = 4$

$w_H(\mathbf{b})$	$b(x)$	$h(x)$
2	$1 + x^3$	$1 + x + x^2 + x^3$
3	$1 + x^2 + x^4$	$1 + x + x^3 + x^4$
	$1 + x^4 + x^5$	$1 + x + x^2 + x^5$
	$1 + x + x^5$	$1 + x^3 + x^4 + x^5$
4	$1 + x^2 + x^3 + x^5$	$1 + x + x^4 + x^5$
	$1 + x^2 + x^5 + x^6$	$1 + x + x^3 + x^6$
	$1 + x^4 + x^6 + x^7$	$1 + x + x^2 + x^7$
	$1 + x + x^4 + x^6$	$1 + x^3 + x^5 + x^6$
	$1 + x + x^6 + x^7$	$1 + x^3 + x^4 + x^7$
	$1 + x + x^3 + x^7$	$1 + x^5 + x^6 + x^7$

Table 3: All $b(x)$ which produce codewords with weight $w_H(\boldsymbol{c}) \leq 8$ for $M = 32$

$w_H(\mathbf{c})$	$b(x)$	$h(x)$
5	$1 + x + x^2$	$1 + x^2$
6	$1 + x^3$	$1 + x + x^2 + x^3$
	$1 + x + x^3 + x^4$	$1 + x^4$
7	$1 + x^2 + x^4$	$1 + x + x^3 + x^4$
	$1 + x + x^5$	$1 + x^3 + x^4 + x^5$
	$1 + x^4 + x^5$	$1 + x + x^2 + x^5$
	$1 + x + x^3 + x - 5 + x^6$	$1 + x^6$
8	$1 + x^2 + x^3 + x^5$	$1 + x + x^4 + x^5$
	$1 + x^6$	$1 + x + x^2 + x^4 + x^5 + x^6$
	$1 + x + x^4 + x^6$	$1 + x^3 + x^5 + x^6$
	$1 + x^2 + x^5 + x^6$	$1 + x + x^3 + x^6$
	$1 + x + x^3 + x^7$	$1 + x^5 + x^6 + x^7$
	$1 + x + x^6 + x^7$	$1 + x^3 + x^4 + x^7$
	$1 + x^4 + x^6 + x^7$	$1 + x + x^2 + x^7$
	$1 + x + x^3 + x^5 + x^7 + x^8$	$1 + x^8$

0.6 Generalization of RTZ inputs and Their Corresponding Parity-Weight Equations

With our method described in the previous section, we are able to not only determine the distance spectrum of the RSC code, but we have information with regards to the structure of the RTZ inputs. In this section, provide general information with regards to the type of RTZ inputs present in a RSC code. Then, we go a step further and provide a general representation (in polynomial form) of the RTZ inputs grouped by their weight. Finally we derive equations for calculating the parity weight for these RTZ inputs. The generalization as well as the derived equations will be very useful when it comes to designing interleavers for turbo codes.

0.6.1 Preliminaries

To aid in our proofs the following have been defined with respect to the impulse response of the 5/7 RSC code. Let

$$\boldsymbol{\phi}_1 = (0\ 0\ 1) \quad , \ \boldsymbol{\phi}'_1 = (0\ 1\ 0), \ \boldsymbol{\phi}''_1 = (1\ 0\ 0) \tag{0-3}$$

$$\boldsymbol{\phi}_2 = (0\ 1\ 1) \quad , \ \boldsymbol{\phi}'_2 = (1\ 1\ 0), \ \boldsymbol{\phi}''_2 = (1\ 0\ 1) \tag{0-4}$$

$$\boldsymbol{\phi}_7 = (1\ 1\ 1) \tag{0-5}$$

To simplify calculation, we have included an addition table for all the vectors which is shown in Table 4

	$\boldsymbol{\phi}_1$	$\boldsymbol{\phi}'_1$	$\boldsymbol{\phi}''_1$	$\boldsymbol{\phi}_2$	$\boldsymbol{\phi}'_2$	$\boldsymbol{\phi}''_2$	$\boldsymbol{\phi}_7$
$\boldsymbol{\phi}_1$	$\mathbf{0}_3$	–	–	–	–	–	–
$\boldsymbol{\phi}'_1$	$\boldsymbol{\phi}_2$	$\mathbf{0}_3$	–	–	–	–	–
$\boldsymbol{\phi}''_1$	$\boldsymbol{\phi}''_2$	$\boldsymbol{\phi}'_2$	$\mathbf{0}_3$	–	–	–	–
$\boldsymbol{\phi}_2$	$\boldsymbol{\phi}'_1$	$\boldsymbol{\phi}_1$	$\boldsymbol{\phi}_7$	$\mathbf{0}_3$	–	–	–
$\boldsymbol{\phi}'_2$	$\boldsymbol{\phi}_7$	$\boldsymbol{\phi}''_1$	$\boldsymbol{\phi}'_1$	$\boldsymbol{\phi}''_2$	$\mathbf{0}_3$	–	–
$\boldsymbol{\phi}''_2$	$\boldsymbol{\phi}''_1$	$\boldsymbol{\phi}_7$	$\boldsymbol{\phi}_1$	$\boldsymbol{\phi}'_2$	$\boldsymbol{\phi}_2$	$\mathbf{0}_3$	–
$\boldsymbol{\phi}_7$	$\boldsymbol{\phi}'_2$	$\boldsymbol{\phi}''_2$	$\boldsymbol{\phi}_2$	$\boldsymbol{\phi}''_1$	$\boldsymbol{\phi}_1$	$\boldsymbol{\phi}'_1$	$\mathbf{0}_3$

Table 4: Truth Table

0.6.2 Types of RTZ inputs

Regardless of the component code used in turbo coding, the RTZ inputs can be grouped into two basic forms. We shall refer to them as *base RTZ inputs* and *compound RTZ inputs*. The number of base RTZ inputs depends on the component code and cannot be broken down into 2 or more RTZ inputs. Compound RTZ inputs as the name implies, are formed from 2 or more base RTZ inputs and therefore can be broken down into base RTZ input form.

For the 5/7 component code, its base RTZ inputs are weight-2 RTZ inputs (W2RTZs) and weight-3 RTZ inputs (W3RTZs). Every RTZ input with a weight higher than 3 is a compound RTZ input. In general for RTZs with weight w greater than 3, if $w \bmod 2 = 0$, then the RTZ is made up of $w/2$ W2RTZs. On the other hand, if $w \bmod 2 = 1$, then the RTZ is made up of $\lfloor w/2 \rfloor - 1$ W2RTZs and 1 W3RTZ.

0.6.3 General Form of RTZ Inputs

In literature for turbo codes, it has been shown as the weight of the inputs increases, it has less effect on the bound of the BER[reference needed]. For this reason, for any RSC, we will provide a generalization for at most the first 4 distinct (by weight) RTZ inputs in the distance spectrum. These generalizations are obtained from using our novel method to list all the RTZ inputs up to weight w that generate a codeword with weight $\leq d$ and taking note of the pattern that exists. The examples shown here are done with respect to the 5/7 RSC and the at the end of the section shows the generalization for a few other RSC codes.

General Form of W2RTZs

A W2RTZ has the general form

$$\begin{aligned} P(x) &= x^{h\tau+t}(1+x^{\alpha\tau}) \\ &= x^{t}(x^{h\tau}+x^{(h+\alpha)\tau}) \end{aligned} \tag{0-6}$$

where

$$h = 0,1,2,...,\left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ \alpha = 1,2,...,\left\lfloor \frac{M}{\tau} \right\rfloor - h,\ t = (0,1,...,\tau-1)$$

Proof. The proof will be to show that any polynomial of this form is an RTZ input. This can be done either by dividing $P(x)$ by $g(x)$ or by using the impulse response

division by g(x) Without loss of generality, we can assume that $h = t = 0$ and $P(x) = 1+x^{\alpha\tau}$. For any value of α, $P(x) \bmod g(x) = 0$ is always true. This proves that $P(x)$ is a RTZ input which has weight 2.

Using the impulse response If we write $P(x)$ in binary for we have a summation of vectors that take the form

$$\begin{matrix} (\mathbf{0}_3\ \boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{\alpha-1}\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2') \\ (\mathbf{0}_3\ \ (\mathbf{0}_3)_{\alpha}\ \ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2') \end{matrix}$$

Regardless of the value of α we realize that there is an infinite summation of $\boldsymbol{\phi}_2 + \boldsymbol{\phi}_2 = \mathbf{0}_3$ after the second 1 bit leading to a low-weight codeword, proving again that that $P(x)$ is a RTZ input which has weight 2 or a W2RTZ.

This ends the proof. □

General Form of W3RTZs

A W3RTZ has the general form

$$\begin{aligned} Q(x) &= x^{h\tau+t}(1+x^{\beta\tau+1}+x^{\gamma\tau+2}) \\ &= x^{h\tau+t}+x^{(h+\beta)\tau+t+1}+x^{(h+\gamma)\tau+t+2}. \end{aligned} \tag{0-7}$$

where

$$\begin{aligned} h &= 0,1,2,...,\left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ \beta = 0,1,2,...,\left\lfloor \frac{M}{\tau} \right\rfloor - 1 \\ \gamma &= 0,1,2,...,\left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ t = (0,1,...,\tau-1) \end{aligned}$$

Notice that $h \leq \beta$ or $h \leq \gamma$ is not a necessary condition.

Proof. Similarly we prove that $Q(x)$ is a RTZ input either dividing $Q(x)$ by $g(x)$ or utilizing the impulse response.

division by g(x) Without loss of generality, we can assume that $h = t = 0$ and $Q(x) = 1 + x^{\beta\tau+1} + x^{\gamma\tau+2}$. For any value of β and γ, $Q(x) \bmod g(x) = 0$ is always true. This proves that $Q(x)$ is a RTZ input which has weight 3.

Using the impulse response If we write $Q(x)$ in binary for we have a summation of vectors that take the form

$$\begin{array}{lll}(\mathbf{0}_{3(\gamma+h)} & \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2' \ \cdots) \\ (\mathbf{0}_{3(\beta+h)} & \boldsymbol{\phi}_2 & \boldsymbol{\phi}_2'' \ \cdots) \\ (\mathbf{0}_{3h} & \boldsymbol{\phi}_7 & \boldsymbol{\phi}_2 \ \cdots)\end{array}$$

Regardless of the value of β and γ we realize that there is an infinite summation of $\boldsymbol{\phi}_2 + \boldsymbol{\phi}_2 = \mathbf{0}_3$ after the third 1 bit leading to a low-weight codeword, proving again that that $Q(x)$ is a RTZ input which has weight 3 ie a W3RTZ.

This ends the proof.

□

.

General Form of W4RTZs

A W4RTZ has the general form

$$\begin{aligned} R(x) &= x^{h\tau+t}(1 + x^{\alpha_1\tau}) + x^{h'\tau+t'}(1 + x^{\alpha_2\tau}) \\ &= x^t(x^{h\tau} + x^{(h+\alpha_1)\tau}) + x^{t'}(x^{h'\tau} + x^{(h'+\alpha_2)\tau}) \end{aligned} \tag{0-8}$$

where

$$h,\ h' = 0, 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ \alpha = 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - h$$
$$\alpha' = 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - h',\ t,\ t' \in \{0, 1, ..\tau - 1\}, h\tau + t \neq h'\tau + t'$$

It is obvious that the W4RTZ representation is valid since it is a combination of 2 W2RTZs

W5RTZs : Definitions and Breaking them

A W5RTZ has the general form

$$\begin{aligned} S(x) &= x^{h\tau+t}(1 + x^{\alpha\tau}) + x^{h'\tau+t'}(1 + x^{\beta\tau+1} + x^{\gamma\tau+2}) \\ &= x^t(x^{h\tau} + x^{(h+\alpha)\tau}) + x^{h'\tau+t'} + x^{(h'+\beta)\tau+t'+1} + x^{(h'+\gamma)\tau+t'+2} \end{aligned} \tag{0-9}$$

where

$$h,\ h' = 0, 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ \alpha' = 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - h'$$
$$\beta = 0, 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - 1,\ \gamma = 0, 1, 2, ..., \left\lfloor \frac{M}{\tau} \right\rfloor - 1$$
$$t,\ t' \in \{0, 1, ..\tau - 1\}, h\tau + t \neq h'\tau + t'$$

Again, it is obvious that the W4RTZ representation is valid since it is a combination of a W2RTZ and a W3RTZ.

0.6.4 Parity Weight Equations for RTZ Inputs

Once the general form of an RTZ input is know, we can calculate the parity weight of the codeword generated. In this section, we derive the equations for calculating the parity weight for the parity-bit sequences generated by those RTZ Inputs.

Parity Weight Equations for W2RTZs

The parity weight for a W2RTZ $w_p^{(2)}$ is given by

$$w_p^{(2)} = 2\alpha + 2 \tag{0-10}$$

Proof. For W2RTZ we consider the summation of the vectors below

$$\begin{array}{l} (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\cdots\ \boldsymbol{\phi}_2') \\ +(\mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2') \\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_3'\ \mathbf{0}_3\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3) \end{array}$$

The derived vector will be

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{\alpha-1}\ \boldsymbol{\phi}_7)$$

The parity weight $w_p^{(2)}$ is given by

$$\begin{aligned} w_p^{(2)} &= 2(\alpha-1)+4 \\ &= 2\alpha + 2 \end{aligned}$$

This ends the proof. □

Hamming Weight for W3RTZ Turbo Codewords

The parity weight for a W3RTZ $w_p^{(3)}$ is given by

$$2l + 2 \tag{0-11}$$

where $l = \max(\beta, \gamma)$

Proof.
The polynomial representation of a weight-3 RTZ input is given by

$$Q(x) = x^{h\tau+t}(1 + x^{\beta\tau+1} + x^{\gamma\tau+2})$$

With reference to the impulse response of the 5/7 RSC encoder,

Now, we consider the weight of the vector derived by the sumation of the followings vectors.

$$\begin{array}{ll} (\mathbf{0}_{3(\gamma+h)} & \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots) \\ (\mathbf{0}_{3(\beta+h)} & \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2''\ \cdots) \\ (\mathbf{0}_{3h} & \boldsymbol{\phi}_7\ \boldsymbol{\phi}_2\ \cdots) \end{array}$$

Without loss of generality, we can assume that all weight-3 RTZ inputs begin at the 0th position, ie $h = t = 0$. This is because the case where $h > 0$ or $t > 0$ is just a right-shifted version of the weight-3 RTZ. With this assumption, we we only need to consider cases where $h = 0,\ \gamma \geq h$.

Furthermore, we consider 4 general cases for all possible values of i, j, k where $i \geq k$ These cases are $(=\ \ =)$, $(=\ \ <)$, $(<\ \ =)$ and $(<\ \ <)$

Case 0: $\gamma = \beta = h$

For this case, the vectors to sum will be

$$\begin{array}{r} (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots) \\ (\boldsymbol{\phi}_2\ \boldsymbol{\phi}_2''\ \cdots) \\ (\boldsymbol{\phi}_7\ \boldsymbol{\phi}_2\ \cdots) \\ \hline (\boldsymbol{\phi}_2''\ \mathbf{0}_3\ \cdots) \end{array}$$

and the derived vector will be $(\boldsymbol{\phi}_2''\ \mathbf{0}_3\ \cdots)$ with a weight of $w_p = 2$

Case 1a: $\gamma = h < \beta$

vectors to sum:

$$\begin{array}{r} (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots) \\ (\mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2''\ \cdots) \\ +(\boldsymbol{\phi}_7\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \cdots) \\ \hline (\boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_2'\ \mathbf{0}_3\ \cdots) \end{array}$$

derived vector : $(\boldsymbol{\phi}_2'\ (\boldsymbol{\phi}_2'')_{\beta-h-1}\ \boldsymbol{\phi}_2'\ \mathbf{0}_3\ \cdots)$
Parity weight:

$$w_p = 2(\beta - h) + 2 = 2\beta + 2 \tag{0-12}$$

Case 1b: $\beta = h < \gamma$

vectors to sum:

$$\begin{array}{r} (\mathbf{0}_3\ \cdots\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots) \\ (\boldsymbol{\phi}_2\ \boldsymbol{\phi}_2''\ \cdots\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_2''\boldsymbol{\phi}_2''\ \cdots) \\ +(\boldsymbol{\phi}_7\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \cdots) \\ \hline (\boldsymbol{\phi}_1''\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7\ \mathbf{0}_3\ \cdots) \end{array}$$

derived vector : $(\boldsymbol{\phi}_1''\ (\boldsymbol{\phi}_2)_{\gamma-h-1}\ \boldsymbol{\phi}_7\ \mathbf{0}_3\ \cdots)$
Parity weight:

$$w_p = 2(\gamma - h) + 2 = 2\gamma + 2 \tag{0-13}$$

Case 2a: $h < \gamma = \beta$
vectors to sum:

$$\begin{array}{r} (\mathbf{0}_3 \cdots \ \cdots \ \mathbf{0}_3 \ \phi_1 \ \phi_2' \ \cdots) \\ (\mathbf{0}_3 \cdots \ \cdots \ \mathbf{0}_3 \ \phi_2 \ \phi_2'' \ \cdots) \\ +(\phi_3 \ \phi_2 \ \cdots \ \phi_2 \ \phi_2 \ \phi_2 \ \cdots) \\ \hline (\boldsymbol{\phi}_7 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_1 \ \mathbf{0}_3 \ \cdots) \end{array}$$

derived vector : $(\boldsymbol{\phi}_7 \ (\boldsymbol{\phi}_2)_{\gamma-h-1} \ \boldsymbol{\phi}_1 \ \mathbf{0}_3 \ \cdots)$
Parity weight:

$$w_p = 2(\gamma - h) + 2 = 2\gamma + 2 \tag{0-14}$$

Case 3a: $h < \gamma < \beta$
vectors to sum:

$$\begin{array}{r} (\mathbf{0}_3 \cdots\cdots \ \mathbf{0}_3 \ \phi_1 \ \phi_2' \ \cdots \ \phi_2' \ \phi_2' \ \phi_2' \cdots) \\ (\mathbf{0}_3 \cdots \ \cdots \ \cdots \ \cdots \ \cdots \ \mathbf{0}_3 \ \phi_2 \ \phi_2'' \cdots) \\ +(\phi_3 \ \phi_2 \ \cdots \ \phi_2 \ \phi_2 \ \phi_2 \ \cdots \ \phi_2 \ \phi_2 \ \phi_2 \cdots) \\ \hline (\boldsymbol{\phi}_7 \ \boldsymbol{\phi}_2 \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_1' \ \boldsymbol{\phi}_2'' \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \cdots) \end{array}$$

derived vector : $(\boldsymbol{\phi}_7 \ (\boldsymbol{\phi}_2)_{\gamma-h-1} \ \boldsymbol{\phi}_1' \ (\boldsymbol{\phi}_2'')_{\beta-\gamma-1} \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \ \cdots)$
Parity weight:

$$\begin{aligned} w_p &= 2(\gamma - h) + 2 + 2(\beta - i) \\ &= 2(\beta - h) + 2 \\ &= 2\beta + 2 \end{aligned} \tag{0-15}$$

Case 3b: $h < \beta < \gamma$

$$\begin{array}{r} (\mathbf{0}_3 \ \cdots \ \cdots \ \cdots \ \cdots \ \cdots \ \mathbf{0}_3 \ \phi_1 \ \phi_2' \cdots) \\ (\mathbf{0}_3 \ \cdots\cdots \ \mathbf{0}_3 \ \phi_2 \ \phi_2'' \ \cdots \ \phi_2'' \ \phi_2'' \ \phi_2'' \cdots) \\ +(\phi_3 \ \phi_2 \ \cdots \ \phi_2 \ \phi_2 \ \phi_2 \ \cdots \ \phi_2 \ \phi_2 \ \phi_2 \cdots) \\ \hline (\boldsymbol{\phi}_7 \ \boldsymbol{\phi}_2 \cdots \ \boldsymbol{\phi}_2 \ \mathbf{0}_3 \ \boldsymbol{\phi}_2' \cdots \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_7 \ \mathbf{0}_3 \cdots) \end{array}$$

derived vector : $(\boldsymbol{\phi}_7 \ (\boldsymbol{\phi}_2)_{j-k-1} \ \mathbf{0}_3 \ (\boldsymbol{\phi}_2')_{i-j-1} \ \boldsymbol{\phi}_7 \ \mathbf{0}_3 \ \cdots)$
Parity weight:

$$\begin{aligned} w_p &= 2(\beta - h) + 1 + 2(\gamma - \beta) + 1 \\ &= 2(\gamma - h) + 2 \\ &= 2\gamma + 2 \end{aligned} \tag{0-16}$$

From all the above cases we can conclude that the parity weight for a weight-3 RTZ sequence may be calculated as

$$w_p^{(3)} = 2l + 2 \tag{0-17}$$

where $l = \max(\gamma, \beta)$
This ends the proof $\square$

Hamming weight for W4RTZ Turbo Codewords

The parity weight for a W4RTZ $w_p^{(4)}$ is given by

$$w_p^{(4)} = \begin{cases} 2(\alpha+\alpha')+4, & \text{if } h\tau+t < (h+\alpha)\tau+t < h'\tau+t' < (h'+\alpha')\tau+t' \\ 2(\alpha), & \text{if } h\tau+t < h'\tau+t' < (h'+\alpha')\tau+t' < (h+\alpha)\tau+t,\ t \neq t' \\ 2(\alpha'+(h'-h)+(t'-t)-1), & \text{if } h\tau+t < h'\tau+t' < (h+\alpha)\tau+t < (h'+\alpha')\tau+t',\ t \neq t' \end{cases} \tag{0-18}$$

Proof. For all the proofs, we will rely on the parity vector for W2RTZs, which is given by

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{\alpha-1}\ \boldsymbol{\phi}_7)$$

Case 1: $h\tau+t < (h+\alpha)\tau+t < h'\tau+t' < (h'+\alpha')\tau+t'$

For this case, we have the following parity vector summation.

$$\begin{array}{l} (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3) \\ +(\mathbf{0}_3\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7\ \cdots\ \mathbf{0}_3) \\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7\ \cdots\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7\ \cdots\ \mathbf{0}_3) \end{array}$$

The derived parity vector is then $(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{\alpha-1}\ \boldsymbol{\phi}_7\ \cdots\ \boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{\alpha'-1}\ \boldsymbol{\phi}_7)$ and the weight for the derived parity vector is calculated as

$$\begin{aligned} w_p =& 2(\alpha)+2+2(\alpha')+2 \\ =& 2(\alpha+\alpha')+4 \end{aligned} \tag{0-19}$$

Case 2: $h\tau+t < h'\tau+t' < (h'+\alpha')\tau+t' < (h+\alpha)\tau+t,\ t' \neq t$

For the above case, there are 3 possible vector summations as shown below

Case 2a

$$\begin{array}{l} (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7) \\ +(\mathbf{0}_3\ \ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_7\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_1''\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3) \\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2''\ \cdots\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_1'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7) \end{array} \tag{0-20}$$

for Case 2a, the derived parity vector is

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{(h'-h)}\ \boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2'')_{(\alpha'-1)}\ \boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{((h-h')+(\alpha-\alpha')-2)}\ \boldsymbol{\phi}_7)$$

with a corresponding weight of

$$\begin{aligned} w_p &= 2(h'-h)+1+2(\alpha'-1)+1+2((h-h')+(\alpha-\alpha')-2)+4 \\ &= 2(h'-h)+1+2\alpha'-1+2(h-h')+2(\alpha-\alpha') \\ &= 2(\alpha'-\alpha'+\alpha) \\ &= 2\alpha \end{aligned}$$

Case 2b

$$\begin{array}{c}(\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7)\\ +(\mathbf{0}_3\ \ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2''\ \cdots\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_2'\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3)\\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \mathbf{0}_3\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\boldsymbol{\phi}_7)\end{array} \tag{0-21}$$

For Case 2b, the derived parity vector is

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{(h'-h)}\ \boldsymbol{\phi}_2''\ (\boldsymbol{\phi}_2)_{(\alpha-1)}\ \mathbf{0}_3\ (\boldsymbol{\phi}_2')_{((h-h')+(\alpha-\alpha')-2)}\ \boldsymbol{\phi}_7)$$

And the parity weight is

$$\begin{aligned} w_p &= 2(h'-h)+1+2(\alpha'-1)+2+2((h-h')+(\alpha-\alpha')-2)+3\\ &= 2(h'-h)+1+2\alpha'-2+2+2(h-h')+2(\alpha-\alpha')-1\\ &= 2(\alpha'-\alpha'+\alpha)\\ &= 2\alpha \end{aligned}$$

Case 2c

$$\begin{array}{c}(\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7)\\ +(\mathbf{0}_3\ \ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7\ \mathbf{0}_3\ \mathbf{0}_3)\\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7)\end{array} \tag{0-22}$$

For Case 2c, $t = t'$ and the derived vector as well as the parity weight is the same as that of the Type1 W4RTZ. Therefore for Case 2, the parity weight is given by

$$w_p = 2\alpha \tag{0-23}$$

Case 3: $h\tau + t < h'\tau + t' < (h+\alpha)\tau + t < (h'+\alpha')\tau + t',\ t' \neq t$

Similarly, there are 3 possible vector summations as shown below

Case 3a

$$\begin{array}{c}(\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_7\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3)\\ +(\mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_7\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_2\cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_1'')\\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}_2'\ \cdots\ \boldsymbol{\phi}_2'\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}_2''\ \cdots\ \boldsymbol{\phi}_2''\ \boldsymbol{\phi}_1''\ \boldsymbol{\phi}_2\cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}_1'')\end{array} \tag{0-24}$$

For Case 3a the derived parity vector is

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2')_{(h'-h)}\ \boldsymbol{\phi}_1\ (\boldsymbol{\phi}_2'')_{(h-h'+\alpha)-2}\ \boldsymbol{\phi}_1''\ (\boldsymbol{\phi}_2)_{((h'-h)+(\alpha'-\alpha))}\ \boldsymbol{\phi}_1'')$$

with a weight of

$$\begin{aligned} w_p &= 2(h'-h)+1+2(h-h'+\alpha-2)+2+2((h'-h)+(\alpha'-\alpha))+1\\ &= 2(h'-h)+2(\alpha-\alpha+\alpha')+1-1\\ &= 2((h'-h)+\alpha') \end{aligned}$$

Case 3b

$$\begin{array}{c}(\boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_7\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3)\\ (\mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}''_2\ \cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}''_2\cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}'_2)\\ \hline (\boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}'_1\ \boldsymbol{\phi}''_2\cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}'_2)\end{array} \tag{0-25}$$

For Case 3b the derived parity vector is

$$(\boldsymbol{\phi}_1\ (\boldsymbol{\phi}'_2)_{(h'-h)}\ \boldsymbol{\phi}''_2\ (\boldsymbol{\phi}_2)_{(h-h'+\alpha)-2}\ \boldsymbol{\phi}'_1\ (\boldsymbol{\phi}''_2)_{((h'-h)+(\alpha'-\alpha))}\ \boldsymbol{\phi}'_2)$$

with a weight of

$$\begin{aligned} w_p &= 2(h'-h)+1+2(h-h'+\alpha-2)+3+2((h'-h)+(\alpha'-\alpha))+2\\ &= 2(h'-h)+2(\alpha-\alpha+\alpha')+2\\ &= 2((h'-h)+\alpha'+1)\end{aligned}$$

Case 3c

$$\begin{array}{c}(\boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_7\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3)\\ (\mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_7)\\ \hline \boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_7\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_7)\end{array} \tag{0-26}$$

For Case 3c, $t = t'$ and the derived vector as well as the parity weight is the same as that of the Type1 W4RTZ.

The parity weight equations for cases 3a and 3b are different but without loss of generality if we assume that $t' > t,\ t = 0$ we see that case 2a corresponds to the case where $t' - t = 1$ whiles case 2b corresponds to the case where$t' - t = 2$ Therefore for Case 3 the parity weight is given by

$$w_p^{(4)} = 2(\alpha' + (h'-h) + (t'-t) - 1) \tag{0-27}$$

This ends the proof □

Hamming weight for W5RTZ Turbo Codewords (Proof is incomplete)

Proof. We represent each possible W5RTZ pattern by $*$ and $\cdot$, where $*$ is used to represent the W3RTZ portion of the W5RTZ and $\cdot$ is used to represent the W2RTZ part of the W5RTZ. Once the W5RTZ pattern is determined, we consider actual W5RTZ cases which fit this pattern and then determine the parity weight. Without loss of generality as well as for simplicity sake, if the first element of the W5RTZ is part of the W3RTZ portion, we assume that $h' = t' = 0$, whiles if it is part of the W2RTZ we assume $h = t = 0$.

Case1: $(* \ * \ * \ \cdot \ \cdot)$
The valid W5RTZ cases are shown below

$$h'\tau + t' < (h' + \beta')\tau + (t' + 1) < (h' + \gamma')\tau + (t' + 2) < h\tau + t < (h + \alpha)\tau + t \quad (0\text{-}28)$$
$$h'\tau + t' < (h' + \gamma')\tau + (t' + 2) < (h' + \beta')\tau + (t' + 1) < h\tau + t < (h + \alpha)\tau + t \quad (0\text{-}29)$$

Since the W2RTZ and the W3RTZ do not overlap, it is obvious that the parity weight is

$$\begin{aligned} w_p^{(5)} &= 2(l) + 2 + 2(\alpha) + 2 \\ &= 2(l + \alpha') + 4 \end{aligned}$$

Case2: $(* \ * \ \cdot \ * \ \cdot)$
The valid W5RTZ cases are

$$h'\tau + t' < (h' + \beta')\tau + (t' + 1) < h\tau + t < (h' + \gamma')\tau + (t' + 2) < (h + \alpha)\tau + t (case2a) \quad (0\text{-}30)$$
$$h'\tau + t' < (h' + \gamma')\tau + (t' + 2) < h\tau + t < (h' + \beta')\tau + (t' + 1) < (h + \alpha)\tau + t (case2b) \quad (0\text{-}31)$$

For the case where $h'\tau + t' < (h' + \beta')\tau + (t' + 1) < h\tau + t < (h' + \gamma')\tau + (t' + 2) < (h + \alpha)\tau + t$, the W3RTZ cases that are valid are W3-Case1b and W3-Case 3b.

For the case where $h'\tau + t' < (h' + \gamma')\tau + (t' + 2) < h\tau + t < (h' + \beta')\tau + (t' + 1) < (h + \alpha)\tau + t$, the W3RTZ cases that are valid are W3-Case1a and W3-Case 3a.

Case2a : W3-Case1b

Case2a : W3-Case3b

Case2b : W3-Case1a
There are 3 possible vector summation for this W3RTZ case is

Case2b-a1

$$\begin{array}{r} (\boldsymbol{\phi}'_2 \ \boldsymbol{\phi}''_2 \ \cdots \ \boldsymbol{\phi}''_2 \ \boldsymbol{\phi}''_2 \ \boldsymbol{\phi}''_2 \ \cdots \ \boldsymbol{\phi}''_2 \ \boldsymbol{\phi}'_2 \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \mathbf{0}_3) \\ +(\mathbf{0}_3 \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \boldsymbol{\phi}_7 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}''_1) \\ \hline (\boldsymbol{\phi}'_2 \ \boldsymbol{\phi}''_2 \ \cdots \ \boldsymbol{\phi}''_2 \ \boldsymbol{\phi}'_1 \ \boldsymbol{\phi}'_2 \ \cdots \ \boldsymbol{\phi}'_2 \ \boldsymbol{\phi}''_2 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}''_1) \end{array}$$

For Case2b-a1, the derived vector is $(\boldsymbol{\phi}'_2 \ (\boldsymbol{\phi}''_2)_{(h-h'+\gamma'-1)} \ \boldsymbol{\phi}'_1 \ (\boldsymbol{\phi}'_2)_{(h-h'+\beta'-1)} \ \boldsymbol{\phi}''_2 \ (\boldsymbol{\phi}_2)_{(h-h')+(\alpha-\beta')-1} \ \boldsymbol{\phi}''_1)$ and the parity weight is

$$\begin{aligned} w_p^{(5)} &= 2\gamma' + 2(h - h') - 2\gamma' - 2 + 2 + 2(h' - h) + 2\beta' - 2 + 3 + 2(h - h') + 2\alpha - 2\beta - 2 + 1 \\ &= 2(h - h') + 2(\alpha) \end{aligned}$$

Case2b-b1

$$\begin{array}{l}(\boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \mathbf{0}_3) \\ +(\mathbf{0}_3 \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2''\boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2') \\ \hline (\boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \boldsymbol{\phi}_2\boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2')\end{array}$$

For Case2b-b1, the derived vector is $(\boldsymbol{\phi}_2' \ (\boldsymbol{\phi}_2'')_{(h-h'+\gamma'-1)} \ \boldsymbol{\phi}_2' \ (\mathbf{0}_3)_{(h-h'+\beta'-1)} \ \boldsymbol{\phi}_2 \ (\boldsymbol{\phi}_2'')_{(h-h')+(\alpha-\beta')-1} \ \boldsymbol{\phi}_2')$ and the parity weight is

$$\begin{aligned} w_p^{(5)} &= 2\gamma' + 2(h-h') - 2\gamma' - 2 + 2 + 0(h'-h) + 0\beta' - 0 + 4 + 2(h-h') + 2\alpha - 2\beta - 2 + 2 \\ &= 4(h-h'+1) + 2(\alpha-\beta') \end{aligned}$$

Case2b-c1

$$\begin{array}{l}(\boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \mathbf{0}_3) \\ +(\mathbf{0}_3 \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \boldsymbol{\phi}_1 \ \boldsymbol{\phi}_2' \ \cdots \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2' \ \cdots \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_7) \\ \hline (\boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_1'' \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \mathbf{0}_3 \ \boldsymbol{\phi}_2' \ \cdots \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_7)\end{array}$$

For Case2b-c1, the derived vector is $(\boldsymbol{\phi}_2' \ (\boldsymbol{\phi}_2'')_{(h-h'+\gamma'-1)} \ \boldsymbol{\phi}_1'' \ (\boldsymbol{\phi}_2)_{(h-h'+\beta'-1)} \ \mathbf{0}_3 \ (\boldsymbol{\phi}_2')_{(h-h')+(\alpha-\beta')-1} \ \boldsymbol{\phi}_7)$ and the parity weight is

$$\begin{aligned} w_p^{(5)} &= 2\gamma' + 2(h-h') - 2\gamma' - 2 + 2 + 2(h'-h) + 2\beta' - 2 + 1 + 2(h-h') + 2\alpha - 2\beta - 2 + 3 \\ &= 2(h-h') + 2(\alpha) \end{aligned}$$

Case2b : W3-Case3a

There are 3 possible vector summation for this W3RTZ case is

Case2b-a2

$$\begin{array}{l}(\boldsymbol{\phi}_7 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_1' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2' \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \mathbf{0}_3) \\ +(\mathbf{0}_3 \ \mathbf{0}_3 \ \cdots \ \mathbf{0}_3 \ \mathbf{0}_3 \ \mathbf{0}_3 \ \cdots \ \boldsymbol{\phi}_7 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_1'') \\ \hline (\boldsymbol{\phi}_7 \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_1' \ \boldsymbol{\phi}_2'' \ \cdots \ \boldsymbol{\phi}_1' \ \boldsymbol{\phi}_2' \ \cdots \ \boldsymbol{\phi}_2' \ \boldsymbol{\phi}_2'' \ \boldsymbol{\phi}_2 \ \cdots \ \boldsymbol{\phi}_2 \ \boldsymbol{\phi}_1'')\end{array}$$

For Case2b-a, the derived vector is $(\boldsymbol{\phi}_7 \ (\boldsymbol{\phi}_2'')_{(h-h'+\gamma'-1)} \ \boldsymbol{\phi}_1' \ (\boldsymbol{\phi}_2')_{(h-h'+\beta'-1)} \ \boldsymbol{\phi}_2'' \ (\boldsymbol{\phi}_2)_{(h-h')+(\alpha-\beta')-1} \ \boldsymbol{\phi}_1'')$ and the parity weight is

$$\begin{aligned} w_p^{(5)} &= 2\gamma' + 2(h-h') - 2\gamma' - 2 + 2 + 2(h'-h) + 2\beta' - 2 + 3 + 2(h-h') + 2\alpha - 2\beta - 2 + 1 \\ &= 2(h-h') + 2(\alpha) \end{aligned}$$

Case2b-b2

$$\begin{array}{c}(\boldsymbol{\phi}_7\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}'_1\ \boldsymbol{\phi}''_2\ \cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}''_2\ \cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}'_2\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3)\\ +(\mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}''_2\ \cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}''_2\ \cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}'_2)\\ \hline (\boldsymbol{\phi}_7\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}'_1\ \boldsymbol{\phi}''_2\ \cdots\ \boldsymbol{\phi}'_2\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}''_2\ \cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}'_2)\end{array}$$

For Case2b-b2, the derived vector is $(\boldsymbol{\phi}'_2\ (\boldsymbol{\phi}''_2)_{(h-h'+\gamma'-1)}\ \boldsymbol{\phi}'_2\ (\mathbf{0}_3)_{(h-h'+\beta'-1)}\ \boldsymbol{\phi}_2\ (\boldsymbol{\phi}''_2)_{(h-h')+(\alpha-\beta')-1}\ \boldsymbol{\phi}'_2)$ and the parity weight is

$$\begin{aligned} w_p^{(5)} &= 2\gamma' + 2(h-h') - 2\gamma' - 2 + 2 + 0(h'-h) + 0\beta' - 0 + 4 + 2(h-h') + 2\alpha - 2\beta - 2 + 2 \\ &= 4(h-h'+1) + 2(\alpha-\beta') \end{aligned}$$

Case2b-c2

$$\begin{array}{c}(\boldsymbol{\phi}_7\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}'_1\ \boldsymbol{\phi}''_2\ \cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}''_2\ \cdots\ \boldsymbol{\phi}''_2\ \boldsymbol{\phi}'_2\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3)\\ +(\mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \mathbf{0}_3\ \mathbf{0}_3\ \mathbf{0}_3\ \cdots\ \boldsymbol{\phi}_1\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_7)\\ \hline (\boldsymbol{\phi}_7\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \boldsymbol{\phi}'_1\ \boldsymbol{\phi}''_2\ \cdots\ \boldsymbol{\phi}''_1\ \boldsymbol{\phi}_2\ \cdots\ \boldsymbol{\phi}_2\ \mathbf{0}_3\ \boldsymbol{\phi}'_2\ \cdots\ \boldsymbol{\phi}'_2\ \boldsymbol{\phi}_7)\end{array}$$

For Case2b-c2, the derived vector is $(\boldsymbol{\phi}'_2\ (\boldsymbol{\phi}''_2)_{(h-h'+\gamma'-1)}\ \boldsymbol{\phi}''_1\ (\boldsymbol{\phi}_2)_{(h-h'+\beta'-1)}\ \mathbf{0}_3\ (\boldsymbol{\phi}'_2)_{(h-h')+(\alpha-\beta')-1}\ \boldsymbol{\phi}_7)$ and the parity weight is

$$\begin{aligned} w_p^{(5)} &= 2\gamma' + 2(h-h') - 2\gamma' - 2 + 2 + 2(h'-h) + 2\beta' - 2 + 1 + 2(h-h') + 2\alpha - 2\beta - 2 + 3 \\ &= 2(h-h') + 2(\alpha) \end{aligned}$$

□

0.7 Conclusion

In this paper, we presented a method for listing input message which produce codewords with low-weight parity bit sequences for a for a given (n, k) RSCC. Compared to the Transfer function method, it has low complexity and provides more information about distance spectrum of the RSCC. Using a specially configured finite state machine, we can obtain a partial distance spectrum which we use to calculate an upper bound for the RSCC.

Bibliography

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo codes", Proc. Intern. Conf. Communications (ICC), Geneva, Switzerland, pp. 1064- 1070, May 1993.

[2] John G. Proakis, Masoud Salehi. "Digital Communications", Fifth Edition,Chapter 8, McGraw-Hill.

[3] Todd K. Moon. "Error Correcting Codes",Chapter 12, John Wiley & Sons.

[4] Alain Glavieux, "Channel Coding in Communication Networks",
Chapter 3, John Wiley & Son.

[5] Jing Sun, Oscar Y. Takeshita "Interleavers for Turbo Codes Using Permutation Polynomials over Integer Rings", IEEE Trans. Inform. Theory, vol. 51, pp. 101 - 119 Jan. 2005.

[6] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan, and M. Jézéquel "Designing Good Permutations for Turbo Codes: Towards a Single Model",IEEE Communications Society 2004,pp.341-345