On the Complexity of Error Detection Functions for Redundant Residue Number Systems

Tsutomu Sasao [1] and Yukihiro Iguchi [2]

[1] Dept. of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka 820-8502, Japan
[2] Dept. of Computer Science, Meiji University, Kawasaki 214-8571, Japan

Abstract

This paper considers a single-digit error detection in a Redundant Residue Number System (RRNS). Let f be the function that denotes the set of legitimate codes of an RRNS. To analyze the complexity of the error detection circuit, C-measure, the maximum value of the column multiplicity for f is considered. We show that the C-measure is much smaller than the dynamic range of the RRNS. In this way, we show that f can be implemented by a small Look-up table (LUT) cascade.

1 Introduction

Residue Number Systems (RNSs) are useful for high-speed computation, since each digit can be computed independently, and no carry propagation is necessary [4, 6, 7, 13, 14].

By adding a redundant digit to an RNS, we have a Redundant Residue Number System (RRNS) that detects a single-digit error. In an RRNS which detects a single-digit error, if there is no error, then the code denotes a number in the legitimate range (i.e., the dynamic range). On the other hand, if there is a single-digit error, then the code denotes a number in the illegitimate range (i.e., out of the dynamic range). Let f be the set of RRNS codes that show the numbers in the legitimate range. Then, f can be used as an error detection function for the RRNS. In an RRNS, the magnitude comparisons are not easy. So, in a conventional error detection method, RRNS codes are converted into ones for the Mixed Redundant Residue Number System (MRNS), where the magnitude comparisons are easy [6, 7]. Unfortunately, the conversion from the RRNS to the MRNS is rather complex, and the estimation of circuit size is difficult.

Fig. 1.1 shows an example of a digital signal processing system using an RRNS, where two inputs X and Y are represented by $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{n+1})$ and $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_{n+1})$. Note that $\hat{x}_{n+1}$ and $\hat{y}_{n+1}$ are redundant digits. To detect an error in the arithmetic circuits, we append an error detection circuit. In this paper, we will consider the complexity of the error detection circuit. We will derive upper bounds on the C-measure of the error detection function for a given RRNS. With these bounds, we can estimate the size of the error detection network implemented by a look-up table (LUT) cascade. We will also show that the C-measure is much smaller than the dynamic range. This means that the error detection circuit can be easily implemented by a series connection of small LUTs. This paper is organized as follows: Section 2 introduces basic properties of RNS. Section 3 shows a method to detect error in RRNS. Section 4 introduces functional decomposition and defines C-measure of logic functions. Section 5 derives upper and lower bounds on the C-measure of error detection function for RRNS. Section 6 shows the C-measures for several RRNSs, and shows an example of error detection circuit implemented by an LUT cascade. And finally, Section 7 concludes the paper.

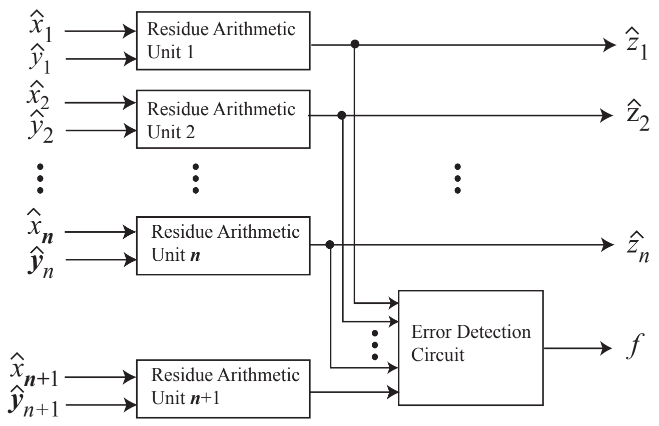

Figure 1.1. Digital signal processing system using RRNS.

2 Residue Number System

2.1 Definition and Basic Properties [14]

Definition 2.1 *Let X be an integer, r_i be the least non-negative remainder when X is divided by an integer m_i called* **modulus***, where $i = 1, 2, \ldots, n$. We denote this by $r_i = |X|_{m_i}$.*

Definition 2.2 *Given a set of mutually prime moduli $(m_1, m_2, \ldots, m_n)$, and an integer X, let (r_1, r_2,. . ., r_n) be an n-tuple of reminders, where $r_i = |X|_{m_i}$. This is the RNS(m_1, m_2,. . ., m_n) representation of an integer X, and denoted by $X_{RNS(m_1,m_2,\ldots,m_n)} = (\hat{x}_1,\ \hat{x}_2,\ldots,\ \hat{x}_n)$, where $\hat{x}_i$ denotes the variable that assume the values in $[0, 1, \ldots, m_i - 1]$. $M = \prod_{i=1}^{n} m_i$ is the* **dynamic range** *of the $RNS(m_1, m_2, \ldots, m_n)$.*

Example 2.1 *The RNS$(3, 5, 7)$ representation of an integer 12 is $12_{RNS(3,5,7)} = (|12|_3, |12|_5, |12|_7) = (0, 2, 5)$.*
(End of Example)

Lemma 2.1 *In the RNS(m_1, m_2,. . ., m_n), the RNS representation of an arbitrary integer in the range $[0, M - 1]$ is unique, where M is the dynamic range. For any n-tuple ($\hat{x}_1$, $\hat{x}_2$,. . ., $\hat{x}_n$), where $\hat{x}_i \in [0, 1, \ldots, m_i - 1]$, there exist a unique integer in $[0, 1, \ldots M]$. Thus, there exists one-to-one correspondence between an RNS representation and an integer in $[0, 1, \ldots, M - 1]$.*

That is, the RNS numbers uniquely represent the numbers in the dynamic range.

Example 2.2 *The dynamic range of the RNS$(3, 5, 7)$ is $M = 3 \times 5 \times 7 = 105$. Thus, any number in the range $[0, 104]$ can be uniquely represented by the RNS.*
(End of Example)

2.2 Arithmetic Operations in RNS

In the ordinary binary number system, an addition can produce *carry signals* from the lower digits to the upper digits. Thus, the delay time is proportional to the number of digits. However, in an RNS, the addition, the subtraction, and the multiplication can be done digit-by-digit in parallel, independently of other digits, and no carry signal exist. With this property, the computation can be done by module circuits. Thus, the networks tend to be simple. On the other hand, division and magnitude comparisons in RNS representation require quite complex circuits [4]. Thus, RNS is suitable for the applications where additions, subtractions and multiplications are major operations.

3 Error Detection in Redundant Residue Number System

In an RNS, additions, subtractions, and multiplications can be done digit-by-digit, independently of other digits. Thus, an error of a digit does not propagate to other digits. In an RRNS, the following property is known [5]:

Theorem 3.1 *An RRNS code with r redundant digits can detect up to r-digit errors, and correct up to $\lfloor \frac{r}{2} \rfloor$-digit errors.*

In this paper, for the simplicity of the network realization, we only consider the case for $r = 1$. In this case, the redundant digit is $\hat{x}_{n+1}$. Note that the corresponding modulus m_{n+1} must be greater than other moduli m_i $(i = 1, 2, \ldots, n)$.

Definition 3.1 *[8] Consider the $RRNS(m_1, m_2, \ldots, m_n, m_{n+1})$, where $m_i < m_j$, $(i < j)$ and m_i and m_j are mutually prime. $M_T = \prod_{i=1}^{n+1} m_i$ denotes the dynamic range when the redundant digit is used. That is, the number of distinct values represented by the $RNS(m_1, m_2, \ldots, m_n, m_{n+1})$. On the other hand, $M = \prod_{i=1}^{n} m_i$ denotes the dynamic range when the redundant digit is not used. That is, the number of distinct values represented by the $RNS(m_1, m_2, \ldots, m_n)$. $[0, M_T - 1]$ denotes the* **total range***, $[0, M - 1]$ denotes the* **legitimate range***, and $[M, M_T - 1]$ denotes the* **illegitimate range***. An RRNS code that denotes a number in the legitimate range is a* **legitimate code***.*

In this paper, we assume that the number of errors in the digits is at most one. In an RRNS, let the redundant modulus be m_{n+1}. In this case, the number of of different values represented by the $n+1$ digits is $M_T = \prod_{i=1}^{n+1} m_i$. Among these, we will use $M = \prod_{i=1}^{n} m_i$ values for computation. In this case, we have the following [8]:

Theorem 3.2 *In the $RRNS(m_1, m_2, \ldots, m_{n+1})$, let m_{n+1} be the redundant modulus. When at most one digit-error exist, we have the following:*

1. *Any single-digit error can be detected.*
2. *The total range, $[0, M_T - 1]$, is partitioned into the legitimate range $[0, M - 1]$ and the illegitimate range $[M, M_T - 1]$. When no error exist, the tuple denotes the value in the legitimate range, while when single-digit error exist, the tuple denotes the value in the illegitimate range, where $M_T = \prod_{i=1}^{n+1} m_i$, and $M = \prod_{i=1}^{n} m_i$.*

(Proof) By Lemma 2.1, an arbitrary integer a in $[0, M - 1]$ can be uniquely represented by an n-tuple. Assume that only the i-th digit has an error. In this case, n digits other

than the i-th digit represent the correct value of a. Also note that the $(n+1)$-tuple including the i-th digit will represent an erroneous value in the illegitimate range. We prove this by contradiction. On the contrary, assume that the tuple represents a value in the range $[0, M-1]$, when there is an error in one digit. By the assumption that $m_{n+1} > m_i$, after removing one digit, any n-tuple represents the same number. However, this contradicts to the property that the RNS represent a number uniquely. Thus, in an RRNS, when a digit has an error, the $(n+1)$-tuple represents the value in $[M, M_T - 1]$. (Q.E.D.)
From this, we have the following:

Corollary 3.1 *Consider the $RRNS(m_1, m_2, \ldots, m_n, m_{n+1})$, where m_{n+1} is the redundant modulus. Let f be the logic function such that $f = 1$ when $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n, \hat{x}_{n+1})$ denotes an integer in the range $[0, M-1]$, and $f = 0$ otherwise, where $M = \prod_{i=1}^{n} m_i$. Then, f is an* **error detection function in the RRNS**.

Note that the error detection function defined above denotes a single error when $f = 0$ and no error when $f = 1$.

4 Functional Decomposition and C-Measure of Logic Function

An arbitrary logic function can be implemented by a single memory. The size of the memory increases exponentially with the number of inputs. However, with the functional decomposition shown in this section, many of practical functions can be implemented by networks with much smaller memories.

4.1 Functional Decomposition and Column Multiplicity

Definition 4.1 *[1] Consider a logic function $f(X) : B^N \rightarrow B$, where $B = \{0, 1\}$, and $X = (x_1, x_2, \ldots, x_N)$. Let (X_1, X_2) be a partition of X, where $X_1 = (x_1, \ldots, x_k)$ and $X_2 = (x_{k+1}, x_{k+2}, \ldots, x_N)$. The* **decomposition chart of** *f is the two-dimensional matrix satisfying the following conditions: Column labels have all possible assignments of B to X_1, Row labels have all possible assignments of B to X_2, and the corresponding matrix value denotes $f(X_1, X_2)$. The number of distinct column patterns in the decomposition chart is the* **column multiplicity**.

Theorem 4.1 *[10] For a given function f, let X_1 be variables that denote rows, and let X_2 be variables that denote columns. Let μ be the column multiplicity of the decomposition chart. Then, the function f can be realized with the network shown in Fig. 4.1. In this case, the number of signal lines connecting two blocks H and G is $\lceil \log_2 \mu \rceil$.*

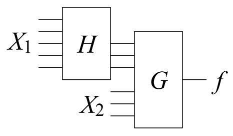

Figure 4.1. Realization of a logic function by decomposition.

When the number of signal lines connecting two blocks is smaller than the number of input variables in X_1, we can often reduce the total amount of memory by the realization in Fig. 4.1. Such technique is a **functional decomposition** [9].

Definition 4.2 *When we consider a decomposition chart of a logic function, a partition of variables X is denoted by $(X_1|X_2)$, where X_1 denotes the row variables, and X_2 denotes the column variables of the decomposition chart. In this case, $\mu(f : X_1|X_2)$ denotes the column multiplicity of the decomposition chart for f.*

First, consider the column multiplicity of an arbitrary logic function.

Lemma 4.1 *Let $X = (X_1, x, X_2)$, and let*

$$\begin{aligned} \mu_1 &= \mu(f : X_1|x, X_2), \\ \mu_2 &= \mu(f : X_1, x|X_2). \end{aligned}$$

Then, the following relations hold:

$$\begin{aligned} \mu_2 &\leq 2\mu_1, \\ \mu_1 &\leq (\mu_2)^2. \end{aligned}$$

(Proof) In the partition $(X_1|x, X_2)$, when a row variable x is moved to a column variable, we have the partition $(X_1, x|X_2)$. In this case, the the number of columns will be double. Thus, the column multiplicity will be at most twice of the original column multiplicity.
In the partition $(X_1, x|X_2)$, when a column variable x is moved to a row variable, we have the partition $(X_1|x, X_2)$. In this case, the number of rows in the decomposition chart will be double. Let the column multiplicity of the original function be μ_1, then the column multiplicity of the decomposition chart after moving the variable will be at most $(\mu_1)^2$. (Q.E.D.)

4.2 C-Measure

Definition 4.3 *[12] The* **C-measure** *(complexity measure) of a logic function $f(X)$ is the maximum value of $\mu(f : X_1|X_2)$, where $X = (x_1, x_2, \ldots, x_N)$, $X_1 = (x_1, x_2, \ldots, x_k)$, and $X_2 = (x_{k+1}, \ldots, x_N)$.*

By repeatedly applying functional decompositions to a given function, we have an **LUT cascade** [11] shown in Fig. 4.2. An LUT cascade consists of **cells**, and the signal lines connecting adjacent cell are **rails**. A logic function with a small C-measure can be realized by a compact LUT cascade. To obtain the C-measure, a decomposition chart is not necessary. A **binary decision diagram** (BDD) that represent the logic function directly shows the C-measure: The C-measure is equal to the maximum width of the BDD, where the variable ordering is $(x_1, x_2, \ldots, x_N)$ [12].

4.3 LUT Cascade

An LUT cascade realizes a given multiple-output logic function by a series connection of LUTs shown in Fig. 4.2. An LUT cascade has a regular structure, and is easier to design than a random logic network [11, 12].

Theorem 4.2 *[11] Let μ be the C measure of a function f. Then, f can be implemented by an LUT cascade, whose cells have at most $\lceil \log_2 \mu \rceil + 1$ inputs, and $\lceil \log_2 \mu \rceil$ outputs.*

Theorem 4.3 *[12] In an LUT cascade that realizes the function f, let N be the number of input variables; s be the number of cells; w be the maximum number of rails (i.e., the maximum number of signal lines between cells); u be the number of inputs for a cell; μ be the C measure of the function f; and $u \geq \lceil \log_2 \mu \rceil + 1$. Then, an LUT cascade satisfying the following condition exists:*

$$s \leq \left\lceil \frac{N - w}{u - w} \right\rceil .$$

5 C-measure of Error Detection Function in RRNS

Let N be the number of input variables of a function. Then, for almost all functions, C-measures are proportional to 2^N. However, for many practical functions, C-measures are relatively smaller [12]. Functions with small C-measures can be efficiently implemented by LUT cascades. From here, we will show the following:

Property 5.1 *The C measures of single-digit error detection functions for RRNSs are much smaller than the dynamic range $M = \prod_{i=1}^{n} m_i$.*

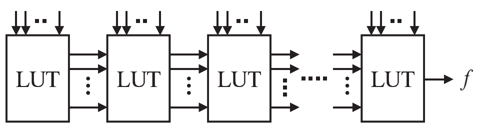

Figure 4.2. LUT cascade.

This means that error detection functions in RRNSs can be efficiently implemented by LUT cascades.

5.1 Lower Bound on Column Multiplicity

In the case of legitimate codes, for each tuple $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$, there exist a unique value for $\hat{x}_{n+1}$. Therefore, the number of legitimate codes having the form $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n, \hat{x}_{n+1})$ is M. Furthermore, we have the following:

Lemma 5.1 *In the $RRNS(m_1, m_2, \ldots, m_{n+1})$, if $\prod_{i=1}^{k} m_i \leq \prod_{i=k+1}^{n+1} m_i$, then for each tuple $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k)$, there exist a unique tuple $(\hat{x}_{k+1}, \ldots, \hat{x}_{n+1})$ that represents the same integer, where $\hat{x}_i \in [0, 1, \ldots m_i - 1]$*

(Proof) Consider an arbitrary integer a in the range $[0, M_U - 1]$, where $M_U = \prod_{i=1}^{k} m_i$. Let $(\hat{x}_{k+1}, \ldots, \hat{x}_{n+1})$ be the RNS representation of a in the $RNS(\hat{x}_{k+1}, \hat{x}_{k+2}, \ldots, \hat{x}_{n+1})$. From the relation $\prod_{i=1}^{k} m_i < \prod_{i=k+1}^{n+1} m_i$ and by Lemma 2.1, the corresponding tuple $(\hat{x}_{k+1}, \ldots, \hat{x}_{n+1})$ is unique in the $RNS(\hat{x}_{k+1}, \ldots, \hat{x}_{n+1})$. (Q.E.D.)

Example 5.1 *In the $RNS(3, 5, 7, 8)$, the relation $3 \times 5 < 7 \times 8$ holds. Thus, the $RNS(7, 8)$ representations of the integers in $[0, 14]$ are unique. In fact, in the $RNS(7, 8)$ representation, all the integers in $[0, 55]$ are uniquely represented by the tuples $(\hat{x}_3, \hat{x}_4)$.*

Theorem 5.1 *Let μ be the C-measure of the error detection function in the $RRNS(m_1, m_2, \ldots, m_{n+1})$. Then, the following relations hold:*

$$M_U + 1 \leq \mu$$

and

$$M_L + 1 \leq \mu,$$

where

$$M_U = \prod_{i=1}^{k} m_i;$$

$$M_L = \prod_{i=k+2}^{n+1} m_i;$$

and k is the largest integer satisfy the relation $M_U \leq M_L$.

(Proof) **1.** Consider a decomposition chart with the partition $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k | \hat{x}_{k+1}, \ldots, \hat{x}_{n+1})$. Consider the RNS code representing a value in $[0, M_U - 1]$. Then, by Lemma 5.1, for each tuple $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k)$, there exist a unique tuple $(\hat{x}_{k+1}, \ldots, \hat{x}_{n+1})$. This means that in each row of the decomposition chart, there exists a unique 1. Note that the number of columns for the codes $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k)$ is M_U. Thus, the column multiplicity of the decomposition chart is $1 + M_U$. The constant 1, corresponds to the columns representing the values [1] that are not in the range $[0, M_U - 1]$: These columns consist of all 0's.

2. From the hypothesis of the lemma, we have $\prod_{i=1}^{k+1} m_i > \prod_{i=k+2}^{n+1} m_i$. Consider the decomposition chart with the partition $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_{k+1} | \hat{x}_{k+2}, \ldots, \hat{x}_{n+1})$. Consider the RNS code that represents a value in $[0, M_L - 1]$. For each tuple $(\hat{x}_{k+2}, \ldots, \hat{x}_{n+1})$, there exist a unique tuple $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k)$. Similarly to Lemma 5.1, all the tuples $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k)$ are distinct. This means that in each column of the decomposition chart, there exists a unique 1. Note that the number of rows for the codes $(\hat{x}_{k+2}, \ldots, \hat{x}_{n+1})$ is M_L. Thus, the column multiplicity of the decomposition chart is $1 + M_L$. The constant 1 corresponds to the column that represents the values not in the range $[0, M_L - 1]$: These columns consist of all 0's.

(Q.E.D.)

Example 5.2 *Consider the error detection function f for $RRNS(2, 3, 5, 7)$. Let $k = 2$. Then, $M_U = 2 \times 3 = 6$, $M_L = 5 \times 7 = 35$. Note that k is the largest integer satisfying $M_U \leq M_L$. $RNS(2, 3)$ representations of integers 0,1,2,3,4, and 5 are $(0, 0), (1, 1), (0, 2), (1, 0), (0, 1)$ and $(1, 2)$, respectively. $RNS(5, 7)$ representations of integers 0,1,2,3,4, and 5 are $(0, 0), (1, 1), (2, 2), (3, 3), (4, 4)$ and $(0, 2)$, respectively. Table 5.1 shows the decomposition chart $(\hat{x}_1, \hat{x}_2 | \hat{x}_3, \hat{x}_4)$ for f. Note that there are 30 1's in the decomposition chart. Consider the rows $(0, 0), (1, 1), (2, 2), (3, 3), (4, 4)$ and $(0, 2)$. Each row has a unique 1. Note that rows $(0, 2), (1, 3), (2, 4), (3, 5)$, and $(4, 6)$ have only 0's. Also, the last column headed by $(*, 3)$ denotes $(0, 3)$ and $(1, 3)$. The column $(*, 3)$ has only zero elements. We can confirm that the column multiplicity is $6 + 1 = 7$.*

Next, consider the decomposition chart $(\hat{x}_1, \hat{x}_2, \hat{x}_3 | \hat{x}_4)$ for f shown in Table 5.2. Note that each column, except for the last column, has a unique 1. The last column headed by $(*, 3, *)$ denotes unused codes, and has only zeros. We can confirm that the column multiplicity is $7 + 1 = 8$. (End of Example)

5.2 Upper Bound on Column Multiplicity

Theorem 5.2 *Let μ be the C-measure of the error detection function of the $RRNS(m_1, m_2, \ldots, m_{n+1})$. Then, the following relation holds:*

$$\mu \leq \max_{t=0}^{s} \{min[2^t M_U, (M_L)^{2^{s-t}}]\} + 1,$$

where

$$M_U = \prod_{i=1}^{k} m_i;$$

$$M_L = \prod_{i=k+2}^{n+1} m_i;$$

k is the largest integer satisfy the relation $M_U \leq M_L$; and s denotes the number of bits to represent $\hat{x}_{k+1}$.

Table 5.1. Decomposition chart $(\hat{x}_1, \hat{x}_2 | \hat{x}_3, \hat{x}_4)$

	0	0	0	1	1	1	*	$\hat{x}_1$
$\hat{x}_3\hat{x}_4$	0	1	2	0	1	2	3	$\hat{x}_2$
00	1	0	0	0	0	0	0	
01	0	0	0	1	0	0	0	
02	0	0	0	0	0	0	0	
03	0	1	0	0	0	0	0	
04	0	0	0	0	1	0	0	
05	0	0	0	0	0	1	0	
06	0	0	1	0	0	0	0	
10	0	0	0	1	0	0	0	
11	0	0	0	0	1	0	0	
12	0	1	0	0	0	0	0	
13	0	0	0	0	0	0	0	
14	0	0	0	0	0	1	0	
15	0	0	1	0	0	0	0	
16	1	0	0	0	0	0	0	
20	0	0	0	0	1	0	0	
21	0	1	0	0	0	0	0	
22	0	0	1	0	0	0	0	
23	0	0	0	0	0	1	0	
24	0	0	0	0	0	0	0	
25	1	0	0	0	0	0	0	
26	0	0	0	1	0	0	0	
30	0	1	0	0	0	0	0	
31	0	0	1	0	0	0	0	
32	0	0	0	0	0	1	0	
33	0	0	0	1	0	0	0	
34	1	0	0	0	0	0	0	
35	0	0	0	0	0	0	0	
36	0	0	0	0	1	0	0	
40	0	0	1	0	0	0	0	
41	0	0	0	0	0	1	0	
42	0	0	0	1	0	0	0	
43	1	0	0	0	0	0	0	
44	0	1	0	0	0	0	0	
45	0	0	0	0	1	0	0	
46	0	0	0	0	0	0	0	

[1] Since at least one number in $(m_1, m_2, \ldots, m_k)$ is an odd number, there exist binary codes that represent values not in $[0, M_U - 1]$. For example, suppose that $m_2 = 3$. Then, $\hat{x}_2$ is represented by two bits: 0 is denoted by 00, 1 is denoted by 01, and 2 is denoted by 10. However, 11 is an unused code. So, if $\hat{x}_2 = 3$, then the value is not in the range $[0, m_U - 1]$.

Table 5.2. Decomposition chart $(\hat{x}_1, \hat{x}_2, \hat{x}_3 | \hat{x}_4)$

	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	*	$\hat{x}_1$
	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	3	$\hat{x}_2$
$\hat{x}_4$	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	*	$\hat{x}_3$
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	
2	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	
3	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	
4	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	
5	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	
6	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	

(Proof) Let $X_1 = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_k)$ and $X_2 = (\hat{x}_{k+2}, \ldots, \hat{x}_{n+1})$. Let f be the error detection function of the RRNS. Let

$$\mu_1 = \mu(f : X_1 | \hat{x}_{k+1}, X_2),$$

and

$$\mu_2 = \mu(f : X_1, \hat{x}_{k+1} | X_2).$$

Then, the following relation hold:

$$\mu_1 \leq 1 + M_U.$$

The constant 1 corresponds to the columns with all 0's, which denote the codes other than legitimate codes $(\hat{x}_1, \hat{x}_2, \ldots \hat{x}_k)$. When the row variables are $X_2 = (\hat{x}_{k+2}, \ldots, \hat{x}_{n+1})$, each column has at most one 1. Thus, we have the following relation:

$$\mu_2 \leq 1 + M_L.$$

The constant 1 corresponds to the columns with all 0's, which denote the codes other than legitimate codes $(\hat{x}_1, \hat{x}_2, \ldots \hat{x}_k)$.

Next, consider the column multiplicity of the decomposition chart of the function f, where the bits of the variables $\hat{x}_{k+1} = (y_1, y_2, \ldots, y_s)$ are partitioned into two. Let

$$\mu_3 = \mu(f : X_1, y_1, y_2, \ldots, y_t | y_{t+1}, \ldots, y_s, X_2).$$

By applying Lemma 4.1 t times, we have the relation:

$$\mu_3 \leq 2^t M_U + 1.$$

This is because the number of columns for the legitimate codes will be twice by moving one variable, but the the number patterns for other codes (i.e., unused codes) remains the same, since such columns consist of all 0's. And, by moving $(s - t)$ variables in the columns into rows, we have

$$\mu_3 \leq (M_L)^{2^{s-t}} + 1.$$

Thus, the column multiplicity of the decomposition chart that partitions the variables in $\hat{x}_{k+1}$ satisfies the relation of the theorem. (Q.E.D.)

Example 5.3 *As an example for $n = 3$, consider the error detection function in the $RRNS(2, 3, 5, 7)$. The dynamic range is $M = 2 \times 3 \times 5 = 105$. First, obtain the lower bound on μ. Since, $2 \times 3 < 5 \times 7$, we have $k = 2$ in Lemma 5.1. From this, we have $1 + 2 \times 3 = 7 \leq \mu$ and $1 + 7 = 8 \leq \mu$.*
Second, obtain the upper bound on μ. From Theorem 5.2, we have $M_U = 2 \times 3 = 6$, and $M_L = 7$. Consider the function $g(t) = min\{2^t M_U + 1, (M_L)^{2^{3-t}} + 1\}$. This function takes its maximum when $t = 2$. Thus, we have $\mu \leq 2^2 \times M_U + 1 = 25$. By constructing the BDD, we found that the C-measure is $\mu = 15$. (End of Example)

Example 5.4 *As an example for $n = 5$, consider the error detection function in the $RRNS(2, 3, 5, 7, 11, 13)$. The dynamic range is $M = 2310$. First, obtain the lower bound on μ. Since, $2 \times 3 \times 5 < 7 \times 11 \times 13$, we have $k = 3$ in Lemma 5.1. From this, we have $1 + 2 \times 3 \times 5 = 31 \leq \mu$ and $1 + 11 \times 13 = 144 \leq \mu$.*
Second, obtain the upper bound on μ. From Theorem 5.2, we have $M_U = 2 \times 3 \times 5 = 30$, and $M_L = 11 \times 13 = 143$. Note that $s = 3$. Consider the function $g(t) = min\{2^t M_U + 1, (M_L)^{2^{3-t}} + 1\}$. This function takes its maximum when $t = 3$. Thus, we have $\mu \leq M_L + 1 = 144$. In this case, the upper bound and the lower bound are the same. Thus, the C measure is $\mu = 144$. (End of Example)

5.3 Changing the Order of Variables

In Example 5.3, the difference of the upper bound (25) and the lower bound (8) is so large, that they are not so useful to estimate the value of the C-measure. Note that Lemma 5.1 and Theorem 5.2 hold independently of the ordering of the variables. Thus, by changing the ordering of the variables, we can obtain tighter bounds on the C-measure. Note that the C-measure depends on the ordering of the input variables.

Example 5.5 *In Example 5.3, consider the error detection function in the $RRNS(3, 5, 2, 7)$, where the ordering of the*

Table 6.1. C-measure of error detection functions in RRNSs.

n	m_1	m_2	m_3	m_4	m_5	m_6	M	M_T	μ	UB	LB
3	17	31	32	33			16864	556512	528	4217	528
3	19	27	32	35			16416	574560	516	4105	514
4	8	11	13	15	17		17160	291720	490	705	256
4	7	11	15	16	17		18480	314160	371	617	273
5	3	5	7	11	13	17	15015	195195	422	841	222
5	3	5	8	11	13	17	17160	223080	397	961	222

variables has been changed. First, obtain the lower bound on μ. Since $3 < 5 \times 2 \times 7$, we have $k = 1$ in Lemma 5.1. From this, we have $1+3 = 4 \leq \mu$, and $1+2\times 7 = 15 \leq \mu$. Next, obtain the upper bound on μ. From Theorem 5.2, we have $M_U = 3$, and $M_L = 2 \times 7 = 14$. Note that $s = 3$. Consider the function $g(t) = min\{2^t M_U + 1, (M_L)^{2^{3-t}} + 1\}$. This function takes its maximum when $t = 3$. Thus, we have $\mu \leq M_L + 1 = 15$. In this case, the upper bound and the lower bound are the same. So, the C-measure is $\mu = 15$. (End of Example)

6 Experimental Results

For several RRNSs, we obtained C-measures of the error detection functions. Table 6.1 shows the results, where n denotes the number of non-redundant moduli; m_i denotes the value of modulus; M denotes the dynamic range of the legitimate range; M_T denotes the dynamic range of the total range; μ denotes the C-measure; UB denotes the upper bound obtained by Theorem 5.2; and LB denotes the lower bound obtained by Theorem 5.1. Table 6.1 shows that Property 5.1 holds: **C-measure is much smaller than the dynamic range**. In this experiment, M, the dynamic ranges of RRNSs are chosen so that they match 14-bit precision, since many AD converters have 14-bit precision. From Table 6.1 and Theorem 4.3, we can estimate the size of an error detection network for a given RRNS.

Example 6.1 *Design the cascade for the error detection function for $RRNS(8, 11, 13, 15, 17)$. In this case, $\hat{x}_1$ is represented by 3 bits, $\hat{x}_2, \hat{x}_3$, and $\hat{x}_4$ are represented by 4 bits, and $\hat{x}_5$ is represented by 5 bits. Thus, the total number of primary inputs is $N = 20$. Here, the most significant bit of $\hat{x}_1$ is denoted by $\hat{x}_1(2)$, while the least significant bit of $\hat{x}_1$ is denoted by $\hat{x}_1(0)$. Table 6.1 shows that $\mu = 490$. Since $w = \lceil \log_2 \mu \rceil = 9$, the number of rails in the LUT cascade is at most 9. When cells with $u = 12$ inputs are used, by Theorem 4.3, we have $s = \left\lceil \frac{N-w}{u-w} \right\rceil = \lceil \frac{20-9}{12-9} \rceil = \lceil \frac{11}{3} \rceil = 4$. This means that the function can be implemented by the LUT cascade with 4 cells. However, detailed analysis of the column multiplicities shows that this function requires only 3 cells as shown in Fig 6.1. The first cell has 12 external inputs: $\hat{x}_1(2,1,0), \hat{x}_2(3,2,1,0), \hat{x}_3(3,2,1,0)$ and $\hat{x}_4(3)$, the most significant bit of $\hat{x}_4$. Let $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4(3)|\hat{x}_4(2,1,0), \hat{x}_5)$ be the partition of the input variables. Since the column multiplicity of the decomposition chart is 256, the first cell has 8 outputs. The second cell has 8 rail inputs and four external inputs $\hat{x}_4(3,2,1,0)$. Let $(\hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4|\hat{x}_5)$ be the partition of the input variables. Since the column multiplicity of the decomposition chart is 18, the second cell has 5 rail outputs. The last cell has 5 rail inputs and 5 external inputs $\hat{x}_5(4,3,2,1,0)$. (End of Example)*

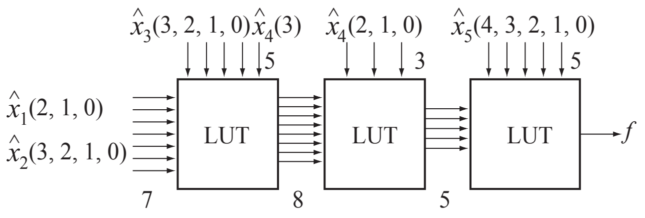

Figure 6.1. LUT cascade for error detection function.

7 Conclusion and Comments

In this paper we have

1. derived upper and lower bounds on the C-measure of the error detection function for a given RRNS.

2. obtained C-measures of the error detection functions for several RRNSs, and demonstrated that C-measures are much smaller than the dynamic ranges. Thus, error detection circuits can be implemented by compact LUT cascades.

3. shown that the upper and lower bounds can be improved by changing the ordering of the input variables.

In this paper, we only consider the case for $r = 1$. An extension for the case of $r \geq 2$ is straightforward, although the complexity of the circuit will be large.

In the past, to detect an error in RRNS, only the method using MRNS or a method using Chinese Reminder Theorem (CRT) were known. Thus, the estimation of the size of the error detection circuit was difficult. This paper presents a design method without using MRNS or CRT. By using the C-measure of the error detection function, we can estimate the size of the circuit.

Acknowledgments

This research is partly supported by The Japan Society for the Promotion of Science (JSPS) Grant in Aid for Scientific Research, and the Knowledge Cluster Initiative (the second stage) of MEXT (Ministry of Education, Culture, Sports, Science and Technology).

References

[1] R. L. Ashenhurst, "The decomposition of switching functions," *International Symposium on the Theory of Switching*, pp. 74-116, April 1957.

[2] F. Barsi and P. Maestrini, "Error correcting properties of redundant residue number systems," *IEEE Trans. Comput*, Vol. 22 No. 3, March 1973, pp. 307-315.

[3] W. K. Jenkins, "The design of error checkers for self-checking residue number arithmetic," *IEEE Trans. Comput*, Vol. 32 No. 4, April 1983, pp. 388-396.

[4] I. Koren, *Computer Arithmetic Algorithms*, 2nd Edition, A. K. Peters, Natick, MA, 2002.

[5] D. Mandelbaum, "Error correction in residue arithmetic," *IEEE Transactions on Computers*, 1972, Vol. C-21, No. 6, pp. 538-545

[6] P. V. A. Mohan, *Residue Number Systems: Algorithms and Architectures,* Kluwer Academic Pub. 2002.

[7] A. Omondi and B. Premkumar, *Residue Number Systems: Theory and Implementation*, Imperial College Press, Sept. 2007.

[8] G. A. Orton, L. E. Peppard, and S. E. Tavares, "New fault tolerant techniques for residue number systems," *IEEE Trans. Comput*, Vol. 41, No. 11, Nov. 1992, pp. 1453 - 1464.

[9] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.

[10] T. Sasao, "FPGA design by generalized functional decomposition," In *Logic Synthesis and Optimization*, Sasao ed., Kluwer Academic Publisher, pp. 233-258, 1993.

[11] T. Sasao, M. Matsuura, and Y. Iguchi, "A cascade realization of multiple-output function for reconfigurable hardware," *International Workshop on Logic and Synthesis (IWLS01)*, Lake Tahoe, CA, June 12-15, 2001, pp. 225-230.

[12] T. Sasao, "Analysis and synthesis of weighted-sum functions," *IEEE TCAD, Special issue on International Workshop on Logic and Synthesis*, Vol. 25, No. 5, May 2006, pp. 789 - 796.

[13] M. A. Soderstrand, W. K. Jenkins, G. A. Jullien, and F. J. Taylor (e.d.), *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing* IEEE Press, 1982.

[14] F. J. Taylor, "Residue arithmetic: A tutorial with examples," *IEEE Computer*, pp. 50-62, Vol. 17, May 1984.