Radix Converters: Complexity and Implementation by LUT Cascades

Tsutomu Sasao

Department of Computer Science and Electronics,
Kyushu Institute of Technology,
Iizuka 820-8502, Japan

Abstract

In digital signal processing, we often use higher radix system to achieve high-speed computation. In such cases, we require radix converters. This paper considers the design of LUT cascades that convert p-nary numbers to q-nary numbers. In particular, we derive several upper bounds on the column multiplicities of decomposition charts that represent radix converters. From these, we can estimate the size of LUT cascades to realize radix converters. These results are useful to design compact radix converters, since these bounds show strategies to partition the outputs into groups.

1 Introduction

Most digital systems use binary arithmetic. However, some digital signal processing systems use r-nary numbers ($r > 2$) for high-speed operations. In such cases, a special circuit is required for the conversion between binary numbers and r-nary numbers [4, 7]. Various methods exist to convert p-nary numbers into q-nary numbers. In general, the computational complexity is large. In particular, a combinational logic circuit that converts radix tends to be complex [5]. Radix conversion can be done by table lookup. The radix conversion using memory is very fast, but the size of the table tends to be large: The size increases exponentially with the number of inputs.

A function whose multiplicity of the decomposition chart for f is small, can be efficiently implemented by an LUT cascade [10]. The LUT cascade has a regular structure and is easy to design and modify. To implement a radix converter, we use several LUT cascades, because implementing an entire radix converter by a single cascade requires a very large LUT, and is often impractical.

In this paper, we derive upper bounds on the column multiplicities of decomposition charts for radix converters. With these, we can estimate the size of an LUT cascade that represents consecutive digits in a radix converter. Experimental results show that the bound is tight, and we can efficiently design radix converters with LUT cascades.

2 Radix Converters

2.1 Radix Conversion

Definition 2.1 *The integer representation of a p-valued vector $\vec{x} = (x_{n-1}, x_{n-2}, \ldots, x_0)$, where $x_i \in P$, and $P = \{0, 1, \ldots, p-1\}$, is $N(\vec{x}) = \sum_{i=0}^{n-1} x_i p^i$.*

Definition 2.2 *Let $\vec{x} = (x_{n-1}, x_{n-2}, \ldots, x_0)_p$ be a p-nary number of n digits, and let $\vec{y} = (y_{m-1}, y_{m-2}, \ldots, y_0)_q$ be a q-nary number of m digits. Given $\vec{x}$, the operation to derive $\vec{y}$ that satisfies the relation*

$$\sum_{j=0}^{m-1} y_j q^j = \sum_{i=0}^{n-1} x_i p^i \qquad (2.1)$$

*is **radix conversion**, where $x_i \in P, P = \{0, 1, \ldots, p-1\}, y_j \in Q$, and $Q = \{0, 1, \ldots, q-1\}$.*

Definition 2.3 *Consider an integer function $f : P^n \to R$, where $P = \{0, 1, \ldots, p-1\}$ and $R = \{0, 1, \ldots, r-1\}$. A pair of vectors $(\vec{a}_1, \vec{a}_2)$, where $\vec{a}_1, \vec{a}_2 \in P^n$, satisfying $N(\vec{a}_1) + 1 = N(\vec{a}_2)$, and $f(\vec{a}_1) \neq f(\vec{a}_2)$, is a **transition pair** of a function f. A sequence of s vectors $(\vec{a}_1, \vec{a}_2, \ldots, \vec{a}_s)$ is a **run** of f, if $f(\vec{a}_0) \neq f(\vec{a}_1) = f(\vec{a}_2) = \ldots = f(\vec{a}_s) \neq f(\vec{a}_{s+1})$, where the inequalities are omitted if extreme indices are non existent (beyond the endpoints), and $\vec{a}_0, \vec{a}_1, \vec{a}_2, \ldots, \vec{a}_s, and \vec{a}_{s+1}$ are consecutive vectors. The number of runs in f is denoted by $\rho(f)$.*

Example 2.1 *Consider the case, where $p = 2$, $r = 3$, and $n = 3$. In the integer function $f(x_2, x_1, x_0)$ shown in Table 2.1, the transition pairs are $((001), (010))$, $((011), (100))$, and $((100), (101))$. The runs are $\{(000), (001)\}, \{(010), (011)\}$, and $\{(100)\}, \{(101), (110), (111)\}$. The number of transition pairs is 3, and $\rho(f) = 4$.* ∎

Lemma 2.1 *The number of transition pairs $= \rho(f) - 1$.*

Definition 2.4 *[12] Consider the integer function $f(\vec{x})$: $P^n \to R, where P = \{0, 1, \ldots, p - 1\}$ and $R = \{0, 1, \ldots, r - 1\}$. Let $(\vec{x}_H, \vec{x}_L)$ be a partition of $\vec{x}$, where $\vec{x}_H = (x_{n-1}, x_{n-2} \ldots, x_k)$ and $\vec{x}_L = (x_{k-1}, x_{k-2}, \ldots, x_o)$. The **decomposition chart** for f is a two-dimensional matrix, where the column labels have all possible assignments of elements of P to $\vec{x}_L$, the row labels have all possible assignments of elements of P to $\vec{x}_H$, and the corresponding matrix value is equal to $f(\vec{x}_H, \vec{x}_L)$. Among the decomposition charts for f, the one whose column label values and row label values increase when the label moves from left to right, and from top to bottom, is the **standard decomposition chart**. The number of different column patterns in the decomposition chart is the **column multiplicity**.*

Note that in an ordinary decomposition chart, the partitions of variables and the order of labels in the columns and rows are arbitrary. However, in the standard decomposition chart, the labels of the rows are in increasing order of $\vec{x}_H = (x_{n-1}, x_{n-2}, \ldots, x_k)$, and the labels of the columns are in increasing order of $\vec{x}_L = (x_{k-1}, x_{k-2}, \ldots, x_0)$.

Lemma 2.2 *Let the number of runs of the integer function $f(\vec{x})$ be $\rho(f)$. Then, the column multiplicity of the standard decomposition chart for $f(\vec{x})$ is at most $\rho(f)$.*

(Proof) In the standard decomposition chart for f, when we move from the left to the right, the column pattern changes only when the value of the function changes. Thus, the column multiplicity is at most $\rho(f)$. $\square$

Definition 2.5 *Let α be a real number. The maximum integer that is not greater than α is denoted by $\lfloor \alpha \rfloor$, and the minimum integer that is equal to or greater than α is denoted by $\lceil \alpha \rceil$.*

Theorem 2.1 *Let $f_i(\vec{x})$ be the p-valued input q-valued output function represented by the i-th digit $(i = 0, 1, \ldots, m - 1)$ of a p-nary to q-nary converter. Then, the column multiplicity of the standard decomposition chart for the function $f_i(\vec{x})$ is at most q^{i+1}.*

Table 2.1. A function showing the run.

x_2	x_1	x_0	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	2
1	0	1	0
1	1	0	0
1	1	1	0

(Proof) We consider three cases.

1. $i = 0$ (the least significant digit):
   Let $N(\vec{x})$ be the value represented by the input $\vec{x}$. Then, we have

$$f_0(\vec{x}) = N(\vec{x}) \pmod{q}.$$

In the standard decomposition chart, let n_L be the number of variables in $\vec{x}_L$, and consider a column pattern. Then, the top q rows of the column can be represented as follows:

$$\begin{pmatrix} a & \pmod{q} \\ a + b_0 & \pmod{q} \\ a + 2b_0 & \pmod{q} \\ a + 3b_0 & \pmod{q} \\ \vdots & \vdots \\ a + (q-1)b_0 & \pmod{q} \end{pmatrix}$$

In the above column, $a \in \{0, 1, \ldots, b_0 - 1\}$ and $b_0 = p^{n_L}$, and the same pattern appears repeatedly in every q row.

Furthermore, in each column, for the given a and b_0, the column pattern is unique.

Since the values of the column are mod q, the number of different column patterns is at most q. Thus, the column multiplicity of the standard decomposition chart is at most q.

2. $i = 1$:
   Let $N(\vec{x})$ be the value for the input $\vec{x}$. Then, we have

$$f_1(\vec{x}) = \lfloor N(\vec{x}) \pmod{q^2}/q \rfloor.$$

In the decomposition chart, the same numbers appear q times consecutively, and then the numbers increase by 1 mod q. In this case, the top q^2 rows of the column pattern can be represented as follows:

$$\begin{pmatrix} \lfloor a & \pmod{q^2}/q \rfloor \\ \lfloor a + b_1 & \pmod{q^2}/q \rfloor \\ \lfloor a + 2b_1 & \pmod{q^2}/q \rfloor \\ \vdots & \vdots \\ \lfloor a + (q^2 - 1)b_1 & \pmod{q^2}/q \rfloor \end{pmatrix}$$

Here, $a \in \{0, 1, \ldots, b_1 - 1\}$ and $b_1 = p^{n_L}$.

In this case, the column pattern is uniquely specified by the values of a and b_1. Since the values are computed mod q^2, the number of different column patterns is at most q^2. Therefore, the column multiplicity of the standard decomposition chart is at most q^2.

3. $i > 1$:

The i-th digit represents the function

$$f_i(\vec{x}) = \lfloor N(\vec{x}) \pmod{q^{i+1}}/q^i \rfloor.$$

Similarly, consider the top q^{i+1} rows of the column. Since the column is computed mod q^{i+1}, the number of different column patterns is at most q^{i+1}. Thus, the column multiplicity of the standard decomposition chart is at most q^{i+1}. $\qquad\square$

Example 2.2 *Consider the case of $p = 2$, $q = 3$, $n_L = 4$, and $n_H = 2$. Let $\vec{x}_H = (x_5, x_4)$ and $\vec{x}_L = (x_3, x_2, x_1, x_0)$. For $i = 0$, we have the standard decomposition chart shown in Fig. 2.1. In this case, the column multiplicity is 3. For $i = 1$, we have the standard decomposition chart shown in Fig. 2.2. In this case, the column multiplicity is $3^2 = 9$.* $\qquad\blacksquare$

The next three theorems give upper bounds on the column multiplicity for the consecutive digits of a radix converter. A radix converter usually has many outputs. If it is implemented as one circuit, then it will be very large. However, if we partition the outputs into groups, and implement each group separately, then each circuit is reasonably small. The following theorem are useful for such realization.

Theorem 2.2 *Let $f(\vec{x})$ be the p-valued input q^k-valued output function that represents the i-th to the $(i + k - 1)$-th digits of the p-nary-to-q-nary converter. Then, the column multiplicity of the standard decomposition chart for the function $f(\vec{x})$ is at most q^{i+k}.*

(Proof) When the circuit represents simultaneously the i-th to the $(i + k - 1)$-th digits, the integer function $f(\vec{x})$: $P^n \to R$ represent the digits, where $R = \{0, 1, \ldots, r - 1\}$ and $r = q^k$. In a similar way to the proof of Theorem 2.1, we can prove that the column multiplicity of the standard decomposition chart is at most q^{i+k}. $\qquad\square$

Theorem 2.3 *Let $f(\vec{x})$ be the p-valued input q^k-valued output function that represents from the i-th to the $(i + k - 1)$-th digits $(i = 0, 1, \ldots, m - 1)$ of the p-nary-to-q-nary converter of n input digits. Then, the column multiplicity of the standard decomposition chart is at most*

$$\max_{n_H + n_L = n} \{\min(p^{n_L}, q^k \cdot p^{n_H})\}$$

(Proof) Let $\vec{x}_H = (x_{n-1}, x_{n-2}, \ldots, x_{n_L})$ be the row variables, and $\vec{x}_L = (x_{n_L - 1}, x_{n_L - 2}, \ldots, x_0)$ be the column variables of the standard decomposition chart. Let n_H be the number of row variables, and n_L be the number of column variables. It is clear that the column multiplicity at most p^{n_L}, the total number of the columns. From here, we will show that the column multiplicity is at most $q^k \cdot p^{n_H}$.

$\vec{x}_H$		$\vec{x}_L$																		
		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1		x_3
		0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1		x_2
		0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1		x_1
		0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1		x_0
0	0	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0		
0	1	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1		
1	0	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2		
1	1	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0		
x_5	x_4																		

Figure 2.1. The standard decomposition chart for the least significant digit of the binary-to-ternary converter.

By modifying the left-hand side of the equation (2.1) of Definition 2.1, we have

$$\sum_{j=0}^{n_L - 1} x_j p^j + \sum_{j=n_L}^{n-1} x_j p^j. \qquad (2.2)$$

In this case, the value of the first term is fixed if we fix the column, and the value of the second term is fixed if we fix the row.

When we consider the column multiplicity of the standard decomposition chart, the first term can be considered as a constant. The second term may assume p^{n_H} different values. The output values of the i-th through the $(i + k - 1)$-th digits are obtained as follows:

1. Compute the values from equation (2.2) by $mod\ q^{i+k}$, and
2. Derive the quotient by q^i.

That is, the change of the second term will change the output values in at most q^k different ways. Thus, the column multiplicity is at most $q^k \cdot p^{n_H}$. $\qquad\square$

Theorem 2.4 *Let $f_i(\vec{x})$ be the p-valued input q^k-valued output function that represents from the i-th to $(i + k - 1)$-th $(i = 0, 1, \ldots, m - 1)$ digits of the p-nary-to-q-nary converter of n input digits. Then, the column multiplicity of the standard decomposition chart is at most $\lceil p^n/q^i \rceil$.*

(Proof) In the decomposition chart, when we scan from left to right, and from top to bottom, the same numbers occur q^i times consecutively, and then increase by one with mod q^k. Thus, the number of runs is $\lceil p^n/q^i \rceil$. From Lemma 2.2, we can see that the column multiplicity of the standard decomposition chart is at most $\lceil p^n/q^i \rceil$. $\qquad\square$

2.2 Experimental Results

Tables 2.2 and 2.3 compare experimental results (EXP) and theoretical values (UB) of the column multiplicities of

3

$$\begin{array}{cc}
 & \vec{x}_L \\
\end{array}$$

```
                         x⃗_L
        0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1   x3
        0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 1   x2
        0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1   x1
        0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1   x0
     0 0 |0 0 0 1 1 1 2 2 2 0 0 0 1 1 1 2|
x⃗_H 0 1 |2 2 0 0 0 1 1 1 2 2 2 0 0 0 1 1|
     1 0 |1 2 2 2 0 0 0 1 1 1 2 2 2 0 0 0|
     1 1 |1 1 1 2 2 2 0 0 0 1 1 1 2 2 2 0|
       x5 x4
```

Figure 2.2. The standard decomposition chart for the 2nd least significant digit of the binary-to-ternary converter.

the standard decomposition charts for the radix converters. The theoretical value (UB) is the minimum of the upper bounds that are derived from Theorems 2.1 ∼ 2.4.

Tables 2.4 and 2.5 compare the column multiplicities of the standard decomposition charts that represent k consecutive digits of radix converters.

From these tables, we can observe that when the radix is a prime number, theorems give values that are the same as the experimental values. On the other hand, when the radix is 10, the differences of UB and the experimental values are large.

In these tables, the orderings of the variables are fixed to $\vec{x}_H = (x_{n-1}, x_{n-2}, \ldots, x_k)$ and $\vec{x}_L = (x_{k-1}, x_{k-2}, \ldots, x_0)$. However, in the general decomposition chart, the partition of the variables need not be restricted as above. So, for some functions, a different partition can yield decomposition charts with smaller column multiplicities. Thus, these tables just show upper bounds on the column multiplicities of the decomposition charts, and give upper bounds on the sizes of LUT cascades (to be explained later).

These tables also show that the column multiplicities for the middle digits are large, while the column multiplicities for the least significant digits and the most significant digits are small.

3 Design of Radix Converters using LUT Cascades

3.1 LUT Cascade [10, 11, 6]

An arbitrary logic function can be implemented by a single memory. However, with the increase of the number of input variables, the size of the memory increases exponentially.

In general, practical functions often have decomposition charts with small column multiplicities.

Table 2.2. Column multiplicities of standard decomposition charts for binary-to-q-nary converter.

i	$n=16$ $p=2$ $q=3$		$n=16$ $p=2$ $q=5$		$n=16$ $p=2$ $q=7$		$n=16$ $p=2$ $q=10$	
	Exp	UB	Exp	UB	Exp	UB	Exp	UB
0	3	3	5	5	7	7	10	10
1	9	9	25	25	49	49	50	100
2	27	27	125	125	343	343	250	640
3	81	81	512	512	192	192	66	66
4	243	243	105	105	28	28	7	7
5	270	270	21	21	4	4		
6	90	90	5	5				
7	30	30						
8	10	10						
9	4	4						
10	2	2						

Table 2.3. Column multiplicities of standard decomposition charts for p-nary-to-binary converter.

i	$n=8$ $p=3$ $q=2$		$n=6$ $p=5$ $q=2$		$n=5$ $p=7$ $q=2$		$n=5$ $p=10$ $q=2$	
	Exp	UB	Exp	UB	Exp	UB	Exp	UB
0	2	2	2	2	2	2	2	2
1	4	4	4	4	4	4	2	4
2	8	8	8	8	8	8	4	8
3	16	16	16	16	16	16	5	16
4	32	32	32	32	32	32	8	32
5	64	64	64	64	64	64	16	64
6	81	81	125	125	98	98	25	128
7	52	52	123	123	98	98	32	200
8	26	26	62	62	66	66	64	200
9	13	13	31	31	33	33	125	196
10	7	7	16	16	17	17	98	98
11	4	4	8	8	9	9	49	49
12	2	2	4	4	5	5	25	25
13			2	2	3	3	13	13
14					2	2	7	7
15							4	4
16							2	2

Theorem 3.1 *For a given function f, let $\vec{x}_A$ be the variables for the columns, and let $\vec{x}_B$ be the variables for the rows, and let μ be the column multiplicity of the decomposition chart. Then, the function f is realizable with the network shown in Fig. 3.1. In this case, the number of (two-valued) signal line that connect two blocks H and G is $\lceil \log_2 \mu \rceil$.*

When the number of signal lines that connect two blocks is smaller than the number of variables in $\vec{x}_A$, we can often reduce the size of memory to implement the function. This technique is **functional decomposition** [1, 8, 9].

By applying functional decomposition repeatedly to the given function, we have the **LUT cascade** shown in Fig. 3.2.

4

Table 2.5. Column multiplicities of standard decomposition charts for k consecutive digits of ternary-to-binary converter.

i	$n=8,p=3$ $q=2,k=2$		$n=8,p=3$ $q=2,k=3$		$n=8,p=3$ $q=2,k=4$		$n=8,p=3$ $q=2,k=5$		$n=8,p=3$ $q=2,k=6$		$n=8,p=3$ $q=2,k=7$	
	Exp	UB	Exp	UB	Exp	UB	Exp	UB	Exp	UB	Exp	UB
0	4	4	8	8	16	16	32	32	64	64	128	128
1	8	8	16	16	32	32	64	64	128	128	256	256
2	16	16	32	32	64	64	128	128	256	256	512	512
3	32	32	64	64	128	128	256	256	512	512	821	821
4	64	64	128	128	243	243	288	288	411	411	411	411
5	108	108	206	206	206	206	206	206	206	206	206	206
6	103	103	103	103	103	103	103	103	103	103	103	103
7	52	52	52	52	52	52	52	52	52	52		
8	26	26	26	26	26	26	26	26				
9	13	13	13	13	13	13						
10	7	7	7	7								
11	4	4										

Table 2.4. Column multiplicities of standard decomposition charts for k consecutive digits of binary-to-ternary converter.

i	$n=16$ $p=2$ $q=3$ $k=2$		$n=16$ $p=2$ $q=3$ $k=3$		$n=16$ $p=2$ $q=3$ $k=4$		$n=16$ $p=2$ $q=3$ $k=5$	
	Exp	UB	Exp	UB	Exp	UB	Exp	UB
0	9	9	27	27	81	81	243	243
1	27	27	81	81	243	243	729	729
2	81	81	243	243	729	729	2187	2187
3	243	243	729	729	2048	2048	2428	2428
4	576	576	810	810	810	810	810	810
5	270	270	270	270	270	270	270	270
6	90	90	90	90	90	90	90	90
7	30	30	30	30	30	30		
8	10	10	10	10	10	10		
9	4	4						

The cascade consists **cells**, and the wires connecting adjacent cells are **rails**. Functions with small column multiplicities have compact LUT cascade realizations. To derive column multiplicities, we need not use decomposition charts. We can efficiently obtain the column multiplicity by a binary decision diagram (BDD_for_CF) that represents the characteristic function for the multiple-output function [11].

Theorem 3.2 *[10] Let μ be the maximum width of the BDD for the function f. Then, f can be implemented by the LUT cascade consisting of cells with at most $\lceil \log_2 \mu \rceil + 1$ inputs.*

Theorem 3.3 *Consider an LUT cascade for a function f. Let n be the number of primary inputs, s be the number of cells, r be the maximum number of rails (i.e., the number of lines between cells), k be the maximum number of inputs of a cell, μ be the maximum width of the BDD for f, and $k \geq \lceil \log_2 \mu \rceil + 1$. Then, there is an LUT cascade for f that*

satisfies the relation:

$$s \geq \lceil \frac{n-r}{k-r} \rceil$$

(Proof) From the design method of the LUT cascade, we have

$$k + (k - r)(s - 1) \geq n.$$

Here, k in the left-hand side of the inequality denotes the number of inputs of the left-most LUT, and $(k - r)(s - 1)$ denotes the sum of inputs for the remaining $(s - 1)$ LUTs. When the actual number of rails is smaller than r, we append dummy rails to make the number of rails r. From this, we have

$$s - 1 \geq \frac{n-k}{k-r}, \text{ and } s \geq \frac{n-r}{k-r}.$$

Since s is an integer, we have

$$s \geq \lceil \frac{n-r}{k-r} \rceil.$$

When the above equation holds, we can realize an LUT cascade for f having cells with at most k inputs. $\square$

From here, we will consider design of binary-to-ternary converters, ternary-to-binary converters, binary-to-decimal converters, and decimal-to-binary converters.

3.2 Binary-to-Ternary Converters

Let $\vec{y} = (y_{m-1}, y_{m-2}, \ldots, y_0)$ be the outputs of the converter, where $y_i \in \{0, 1, 2\}$. Then y_i, in general, depends on all the inputs $x_i (i = 0, 1, \ldots, n - 1)$. Thus, the network will be quite complex.

Example 3.1 *Consider the case of $n = 16$ and $q = 3$, a 16-bit binary number to an 11-digit ternary number converter. From Table 2.2, we can see that the column multiplicities*

of the standard decomposition chart representing 0-th, 1-st, 2-nd, 7-th, 8-th, 9-th and 10-th digits are small. So, the cascade realization is easy. On the other hand, for the 3-rd, 4-th, 5-th and 6-th digits, the column multiplicities of the standard decomposition chart are large. So, we must be careful to realize the functions. Especially, the simultaneous realization of the 4-th and the 5-th digits will produce a very large LUT cascade. So, we implement these digits separately.

Single-Memory Realization: *The most significant bit of the outputs is constant 0, so the actual number of outputs is 21. The necessary amount of memory is $21 \times 2^{16} = 1,376,256$ (bits).*

Realization using LUT Cascades: *Assume that cells with 10 inputs are used to implement cascades.*

Fig. 3.3 shows the cascades for the converter. The outputs are partitioned into four groups. The first cascade realizes the 0-th, 1-st, 2-nd and 3-rd digits; the second cascade realizes the 4-th digit; the third cascade realizes the 5-th digit; and the fourth cascade realizes the 6-th, 7-th, 8-th, 9-th and 10-th digits. Note that each output digit is represented by two bits.

For the first cascade that realizes the 0-th, 1-st, 2-nd, and 3-rd digits, the column multiplicity is at most 81, which can be observed from the entry for $i = 0$ and $k = 4$ in Table 2.4. Thus, the number of rails of the cascade is at most $\lceil \log_2 81 \rceil = 7$.

For the fourth cascade which realizes the 6-th, 7-th, 8-th, 9-th and 10-th digits, the column multiplicity is at most 90, which can be observed from the entry for $i = 6$ and $k = 5$ in Table 2.4. Thus, the number of rails of the cascade is at most $\lceil \log_2 90 \rceil = 7$. From Theorem 3.3, we can see that all the cascade are realized with three cells.

In the case of the cascades for $i = 4$ and $i = 5$, the column multiplicities of the standard decomposition charts are 243 and 270, respectively, which can be observed from the entries for $q = 3$ in Table 2.2. Thus, the numbers of rails are 8 and 9, respectively. We can reduce the number of rails by changing the ordering of the variables. The number of rails can be reduced to at most 7, and both cascades can be realized with three cells. From Fig. 3.3, we can see that the necessary amount of memory is $20^{10}(7 + 7 + 8 + 7 + 5 + 7 + 5 + 7 + 7 + 10) + 2^8(2 + 2) = 72,704$ (bits), which is much smaller than the single-memory realization. ∎

3.3 Ternary-to-Binary Converter

Let $\vec{y} = (y_{m-1}, y_{m-2}, \ldots, y_0)$ be the outputs of the converter, where $y_i \in \{0, 1\}$. Then, in general, y_i depends on all the inputs $x_i (i = 0, 1, \ldots, n - 1)$. When this converter is implemented by a two-valued logic circuit, unused combinations occur. So, we have incompletely specified functions. Usually, constant zeros are assigned to the undefined
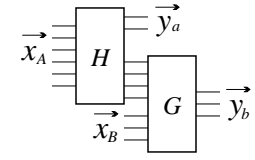

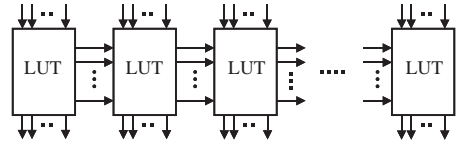**Figure 3.1. Realization of logic functions by decomposition.**

Figure 3.2. LUT cascade with intermediate outputs.

parts. However, we can often reduce the column multiplicities by considering the assignment to *don't cares*.

Example 3.2 *In the case of ternary-to-binary converters, we often use the binary-coded-ternary code to represent a ternary digit. That is 0 is represented by (00); 1 is represented by (01); and 2 is represented by (10). (11) is an unused code.*

To construct the decomposition chart, the input variables are grouped into pairs. The truth table of the 2-digit ternary to 4-bit binary converter is shown in Table 3.1. In the case of binary-coded-ternary representation, (11) is an undefined input, and the corresponding outputs are don't cares. In Table 3.1, the binary-coded-ternary representation is denoted by $\vec{w} = (w_3, w_2, w_1, w_0)$, the ternary representation is denoted by $\vec{x} = (x_1, x_0)$, and the binary representation is denoted by $\vec{y} = (y_3, y_2, y_1, y_0)$. ∎

Table 3.1. Truth table for a ternary-to-binary converter.

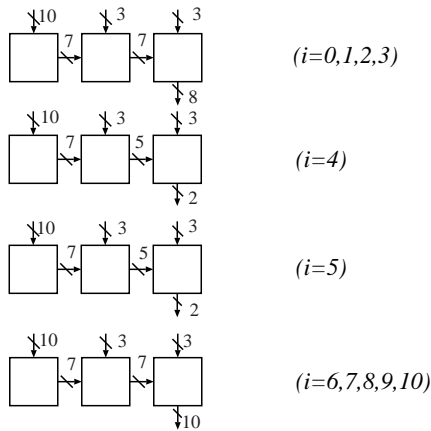| Binary − Coded Ternary				Ternary		Binary				Decimal
w_3	w_2	w_1	w_0	x_1	x_0	y_3	y_2	y_1	y_0	
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	1	1
0	0	1	0	0	2	0	0	1	0	2
0	1	0	0	1	0	0	0	1	1	3
0	1	0	1	1	1	0	1	0	0	4
0	1	1	0	1	2	0	1	0	1	5
1	0	0	0	2	0	0	1	1	0	6
1	0	0	1	2	1	0	1	1	1	7
1	0	1	0	2	2	1	0	0	0	8

Figure 3.3. Binary-to-ternary converter (16-bit inputs, 11-digit output).

Example 3.3 *Consider the converter for an 8-digit ternary number to a 13-digit binary number. When we assign 0's to the outputs for the don't care inputs, the column multiplicity of the standard decomposition chart representing all the outputs will be up to $3^8 = 6561$.*

Single-memory Realization*: It requires $2^{16} \times 13 = 851,968$ (bits), since a ternary digit requires two bits, and the total number of inputs is $2 \times 8 = 16$.*

Realization by LUT Cascades*: Assume that we use cells with 10 inputs. Fig. 3.4 shows the cascade realization. The 13 outputs are divided into two groups: The upper cascade realizes the least significant 6 bits, while the lower cascade realized the most significant 7 bits. From entry $k = 6$ and $i = 0$ of Table 2.5, we can see that the column multiplicity of the decomposition chart for the least significant 6 bits is 64. Thus, the number of rails is $\lceil \log_2 64 \rceil = 6$. From entry of $k = 7$ and $i = 6$ of Table 2.5, we can see that the column multiplicity of the decomposition chart of the most significant 7 bits is 103. Thus, the number of rails is $\lceil \log_2 103 \rceil = 7$.*

In this case, we can optimize the ordering of variables to obtain smaller cascades.

From Fig. 3.4, we can see that the necessary amount of memory of the cascades is $2^{10}(7 + 7 + 7 + 6 + 6) + 2^8(6) = 35,328$ (bits), which is much smaller than the single-memory realization. ∎

3.4 Binary-to-Decimal Converter

Decimal arithmetic circuits are often used in pocket calculators. Thus, various binary to decimal converters have been designed [5]. Unfortunately, the upper bounds obtained by Theorems are not so tight. This is because 10 is a non-prime number.
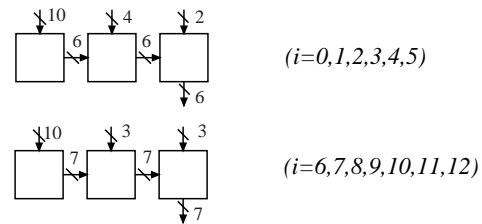

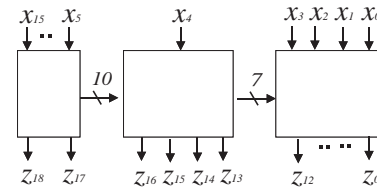Figure 3.4. 8-digit ternary to 13-digit binary converter.

Figure 3.5. 16-digit binary to 5-digit decimal converter.

Example 3.4 *Consider a 16-bit binary to 5-digit decimal converter. When the output decimal number is represented by the BCD code, the number of output bits is 20. Note that the most significant bit z_{19} of the output is always constant zero, so the actual number of outputs is 19. Let $\vec{x} = (x_{15}, x_{14}, \ldots, x_0)$ be the input binary number, and let $\vec{z} = (z_{18}, z_{17}, \ldots, z_0)$ be the output binary-coded-decimal number. When all the outputs are represented at the same time, the column multiplicity is 1250, which is obtained by the BDD_for_CF.*

In the case of binary-to-decimal converter, specific outputs depend on a part of the input variables. Thus, we can produce some outputs in the middle of the cascades. With 11-input cell, we have the converter shown in Fig. 3.5 [11]. ∎

3.5 Decimal-to-Binary Converter

When decimal numbers are represented by binary numbers such as BCD code, many *don't care's* occur.

Example 3.5 *Consider a 5-digit decimal to 17-bit binary converter. When the decimal numbers are represented by BCD codes, the number of binary inputs is $4 \times 5 = 20$. Note that the number of different combinations represented by inputs is 10^5. So the number of don't care combinations is $2^{20} - 10^5$. Thus, the ratio of the don't care is $(2^{20} - 10^5)/2^{20} \simeq 0.90$, In other words, about 90% of the input combinations are don't cares. This means that the*
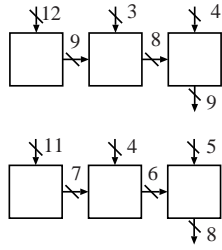
Figure 3.6. 5-**digit decimal to** 17-**digit binary converter.**

assignment of don't care values greatly influences the complexity of radix converters.

From the entry of $p = 10$ and EXP in Table 2.3, we can see that the maximum value of the column multiplicity is 64. When, we implement each digit separately, the number of rails is at most 6.

From Theorem 3.2, we can see that each output can be realized by a cascade using cells with at most 7 inputs. If we realize each digit separately, we need 17 cascades. With 12-input cells, we can implement the radix converter consisting of a pair of cascades as shown in Fig. 3.6. To design this converter, we used the new tool [6]. With an appropriate assignment of don't cares, we can reduce the number of dependent variables [13], and we may reduce the size of cascade. ■

Conclusion

This paper has derived the upper bounds on the column multiplicity for the standard decomposition chart that converts a p-nary representation into a q-nary representation. It has also shown methods to design radix converters by using LUT cascades.

This paper has shown only the upper bounds on the column multiplicity of the standard decomposition chart. Remaining challenging problems are to derive lower bounds on the column multiplicities, and to derive column multiplicities for ordinary decomposition charts. Similar technique can also be used to design radix converters for residue number systems [4].

Acknowledgments

References

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

[2] T. Hanyu and M. Kameyama, "A 200 MHz pipelined multiplier using 1.5 V-supply multiple valued MOS current-mode circuits with dual-rail source-coupled logic," *IEEE Journal of Solid-State Circuits 30, 11*, 1995, pp. 1239-1245.

[3] S. Hassoun and T. Sasao (eds.), *Logic Synthesis and Verification*, Kluwer Publishers, 2001.

[4] C. H. Huang, "A fully parallel mixed-radix conversion algorithm for residue number applications," *IEEE Trans. Comput.*, vol. 32, pp. 398-402, 1983.

[5] S. Muroga, *VLSI System Design*, John Wiley & Sons, 1982, pp. 293-306.

[6] H. Nakahara, T. Sasao, and M. Matsuura, "A design algorithm for sequential circuits using LUT rings," *12th SASIMI Workshop*, Kanazawa, Japan, Oct 18-19, 2004, pp. 430-437.

[7] D. Olson and K. W. Current, "Hardware implementation of supplementary symmetrical logic circuit structure concepts," *30th IEEE International Symposium on Multiple-Valued Logic* (ISMVL 2000) Portland, Oregon, May 23-25, 2000, pp. 371-376.

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

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

[10] 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.

[11] T. Sasao and M. Matsuura, "A method to decompose multiple-output logic functions," *41st Design Automation Conference*, San Diego, CA, USA, June 2-6, 2004, pp. 428-433.

[12] T. Sasao, J. T. Butler, and M. Riedel, "Application of LUT cascades to numerical function generators," *12th SASIMI Workshop*, Kanazawa, Japan, Oct 18-19, 2004, pp. 422-429.

[13] T. Sasao and M. Matsuura, "BDD representation for incompletely specified multiple-output logic functions and its applications to functional decomposition," *Design Automation Conference*, Anaheim, CA., June 13-17, 2005, (submitted).