**Georgia State University**

# ScholarWorks @ Georgia State University

Computer Science Theses

Department of Computer Science

5-12-2005

# Delay and Power Reduction in Deep Submicron Buses

Sharareh Babvey

Follow this and additional works at: https://scholarworks.gsu.edu/cs_theses

Part of the Computer Sciences Commons

## Recommended Citation

Babvey, Sharareh, "Delay and Power Reduction in Deep Submicron Buses." Thesis, Georgia State University, 2005.
https://scholarworks.gsu.edu/cs_theses/5

# COPYRIGHT BY

# Delay and Power Reduction in Deep Submicron Buses

By:

Sharareh Babvey

| | |
|---|---|
| Major Professor: | A. P. Preethy |
| Committee: | Saeid Belkasim |
| | Alex Zelikovsky |

# Delay and Power Reduction in Deep Submicron Buses

By:

Sharareh Babvey

Under the Dirction of:

A. P. Preethy

## ABSTRACT

As technology scales down, coupling between nodes of the circuits increases and becomes an important factor in interconnection analysis. In many cases like the deep submicron technology (DSM), the coupling between lines (inter-wire capacitance) is strong and the power consumed by parasitic capacitance is non-negligible [1-6].

In this work, we employ the differential low-weight encoding [1] to reduce energy and delay (transmission cost) on DSM buses. We propose an enumeration method that reduces the encoder table-size from $O(2^n)$ words to $O(n)$ words, for an $n$-bit DSM bus, and so reduces the memory complexity significantly and facilitates energy and delay reduction due to addressing and fetching data from large lookup tables. We modify the energy and delay equations for DSM buses and develop new representations in terms of number of the same and opposite direction transitions on the bus and use them in our interconnect analysis. We also use these equations to develop formulas for computing the

mean transmission cost per bit on DSM buses for both differential low-weight encoding and uncoded schemes. Using the interconnect analysis, we compute a *help codeword* (from the set of unselected codewords) on the fly and assign to each selected codeword. This low complexity step consists of simple operations and enables us to gain more cost reduction without increasing the table size or number of the bus lines. The simulation results for 16-bit, 32-bit and 64-bit buses at maximum rate (only one extra line added) show that the proposed encoding scheme achieves more than 10% cost reduction, and performs more than 2.5% better than to the original differential low-weight scheme, in the worst case.

# ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. A. P. Preethy for giving me the privilege to work under her supervision. I will always remember her kindness, help and support. I would like to thank Dr. Steven W. McLaughlin for all his contributions to this work and his valuable guidelines and comments. I would also like to thank the committee members Dr. Saeid Belkasim and Dr. Alex Zelikovsky for reviewing my thesis and their valuable comments.

I especially want to thank Dr. Sunderraman for all his help, advice and support during my graduate studies at GSU and for all his hard work and devotion to the Computer Science Department.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# Chapter 1

## Introduction

Current trend of technology requires scaling down the size of devices and increasing the clock frequencies. Such scaling leads to exponential increase in leakage current and decrease in noise immunity for high speed circuits. Shrinking feature sizes results in more interconnect levels packed closer together. Reducing the wire areas decreases the total wire capacitance. However, as technology scales down, not many lines are isolated or shielded anymore. Inductive effects and coupling between nodes of the circuit increase and become an important factor in interconnection analysis. As frequency of operation increases, additional metal patterns or ground planes may be required for inductive shielding. As supply voltage is scaled or reduced, *crosstalk* has become an issue for all clock and signal wiring levels.

In many cases, like the deep submicron technology (DSM), the power dissipation, delay and crosstalk depend on the cross-activities between the nodes, as well. For DSM buses, the coupling between lines (inter-wire capacitance) is much stronger than the coupling between individual lines and ground and the energy consumption caused by parasitic

capacitance is non-negligible. Thus, both the intrinsic capacitance and parasitic capacitance should be taken into account [1, 18].

The near term solution adopted by the industry is the use of thinner metallization, reduced aspect ratios and less aggressive scaling of dielectric constant to decrease the capacitance. There are some limitations for fulfilling this goal, as some required characteristics of certain technologies (like area, current per area, etc.) may be violated.

A wise solution is to employ encoding to reduce the effect of the capacitance both for the existing technologies, and future technologies with smaller capacitance. Encoding controls the transition activity, and so the cross-coupling effect and switching activities on the DSM bus by controlling the data sequences transmitted on the bus [1, 7-15].

In [1, 7], P. Sotiriadis, A. Chandrakasan and V. Tarokh discussed two fundamental questions. What is the minimum energy required per information bit for communicating through DSM buses? And, is the minimum energy achievable using coding? They introduced equations for the minimum energy requirements and showed that the minimum energy is asymptotically achievable using coding. They also proposed a simple differential coding scheme (the differential low-weight coding) that achieved most of the possible energy reduction. This efficient scheme employs differential coding and selects transition codewords of smaller Hamming weights for transmission on the bus. Thus, it is based on reducing the transition activity on the bus. However, this coding scheme requires tables of size $O(2^n)$ for an $n$-bit bus. Also, it does not take into account some characteristics of the DSM buses that can be exploited to reduce the transmission cost further. For example, on a DSM bus if two adjacent lines change both from 0 to 1 or vice versa (same direction transitions), less energy is consumed compared to two transitions

on two nonconsecutive lines. Conversely, if two adjacent lines change in opposite directions then more energy is consumed, compared to two transitions on two nonconsecutive lines. The same direction and opposite direction bus transitions can be exploited to achieve more energy and delay reduction on the DSM bus.

In this work, we propose an enumeration method that reduces the required table-size of the differential low-weight encoder from $O(2^n)$ words to $O(n)$ words, for an $n$-bit bus. Thus, the enumeration reduces the memory complexity significantly, especially when $n$ is large. Furthermore, it facilitates less power dissipation and delay, as it eliminates the need for addressing and fetching data from very large tables in memory.

We modify the energy and delay equations for DSM buses further and develop a new representation in terms of number of the same and opposite direction transitions on the bus. These equations are very helpful for cost analysis in the DSM buses. We also develop equations for computing the mean transmission cost per bit on DSM buses for both differential low-weight encoding and uncoded schemes. We use simple operations like complement and rotate, to compute *help codewords* (from the set of unselected codewords) on the fly, and assign them to the selected codewords. Note that the help codewords enable us to achieve more cost reduction, without increasing the memory complexity or number of the bus lines. The simulation results show that the proposed encoding scheme achieves more cost (power and energy) reduction, compared to the original differential low-weight scheme.

# Chapter 2

# Terms and Definitions

In this chapter, we present some definitions and terms that are used throughout this work.

## 2.1 Transmission Rate

The *transmission rate R* is computed as $R = m/n$, when $m$-bit data sequences $D \in \{0, 1\}^m$ are transmitted on a bus with $n > m$ lines. Set $\{0, 1\}^m$ includes all binary sequences of length $m$ and so has $2^m$ different elements.

In fact, transmission rate is a measure of how many useful bits are transmitted on the bus during each use of bus.

## 2.2 *N*-ary vs. Binary

A *binary* sequence of length $n$ is composed of elements in the set $\{0, 1\}^n$. For example, 011101 is a binary sequence of length 6. Similarly, an *N*-ary sequence of length $n$ is composed of elements in the set $\{0, 1, 2 \dots N\text{-}1\}^n$.

## 2.3 Hamming Weight

Hamming weight of a given binary sequence is the number of ones in that sequence. For example $D = 10011100$ has hamming weight equal to 4 and we show it by $W_H(D) = 4$.

## 2.4 Phrase Representation

We may break each data sequence of length $l$ to phrases, where each phrase consists of $d \geq 0$ zeros and terminates with single nonzero symbol (1 for binary sequences). The *phrase representation* [16] uses the length of each phrase to represent the data sequence. Using this representation, each $l$–bit binary data sequence $D$ is mapped to a unique phrase representation $p$ from the alphabet {1, 2…$n$}, whose length $l$ is equal to the Hamming weight of $D$ [16]. We denote the mapping function by $f_{phrase}$ and we have $f_{phrase}(D) = p$. For example, the phrases representing the 8-bit data $D$ = 10011100 are: 1 001 1 1. The phrase representation for $D$ is the sequence1311 of length 4. Thus, $f_{phrase}(10011100) =$ 1311.

For a given $l$ the function $f_{phrase}(D)$ is one to one and has an inverse. The inverse function expands a phrase representation to a unique binary sequence of length $l$. For example, for $l = 8$, $f^{-1}_{phrase}(1321) = 10010110$.

## 2.5 Lexicographic Ordering

Let {1, 2, 3… $n$}$^k$ denote all the $n$-ary permutations of $k$ symbols chosen from the set {1, 2, 3, … $n$} and let $S$ be a subset of this set. By a lexicographic ordering for $S$ we mean the usual dictionary ordering with the assumption that 1< 2< 3 <… < $n$-1 [16]. Specifically, for $x = (x_1, x_2 \ldots x_k) \in S$ and $y = (y_1, y_2 \ldots y_k) \in S$ we have $x < y$ if there exists $1 \leq i \leq k$ such that $x_i < y_i$ and $x_j = y_j$ for all $1 \leq j < i$.

Let $n_S(x_1, x_2 \ldots x_u)$ denote the number of elements in $S$ for which the first $u$ coordinates are $(x_1, x_2 \ldots x_u)$. Then, the lexicographic index or rank of $x$ is given by [17]:

$$i_S(x) = \sum_{j=1}^{k} \sum_{t=0}^{x_j - 1} n_S(x_1, x_2, \ldots, x_{j-1}, t)$$

# Chapter 3

# Related Works

In this chapter, we introduce the Deep Submicron Buses (DSM) and the important factors of energy consumption and delay on these buses. We also explain how to measure energy and delay in DSM buses. We present the concept of low transition-activity encoding [1, 7-15] for reducing power consumption and delay on DSM buses. Finally, we explain in detail, the differential low-weight encoding [1] that is a simple and efficient low transition-activity scheme upon which we base our proposed encoder.

## 3.1 Deep Submicron Buses

Current trend of technology requires scaling down the size of devices and increase clock frequencies. Such scaling leads to exponential increase in leakage current and decrease in noise immunity for high speed circuits. Shrinking feature sizes results in more interconnect levels packed closer together. Reducing the wire areas decreases the total wire capacitance. However, as technology scales down, not many lines are isolated

or shielded anymore. Coupling between nodes of the circuit increases and becomes an important factor in interconnection analysis. Figures 3.1 and 3.2 compare the old silicon technology and the deep submicron technology (DSM) [1, 18, and 20].
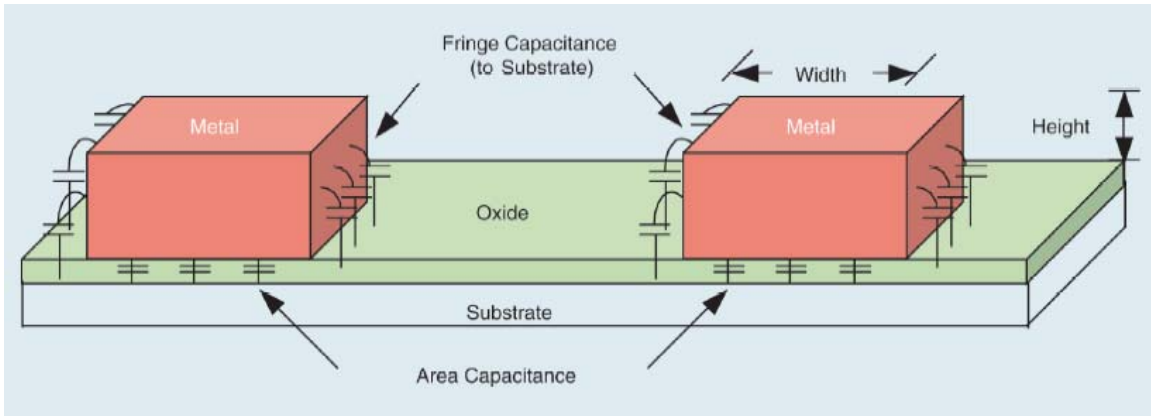


**Figure 3.1** The older silicon technology [18].
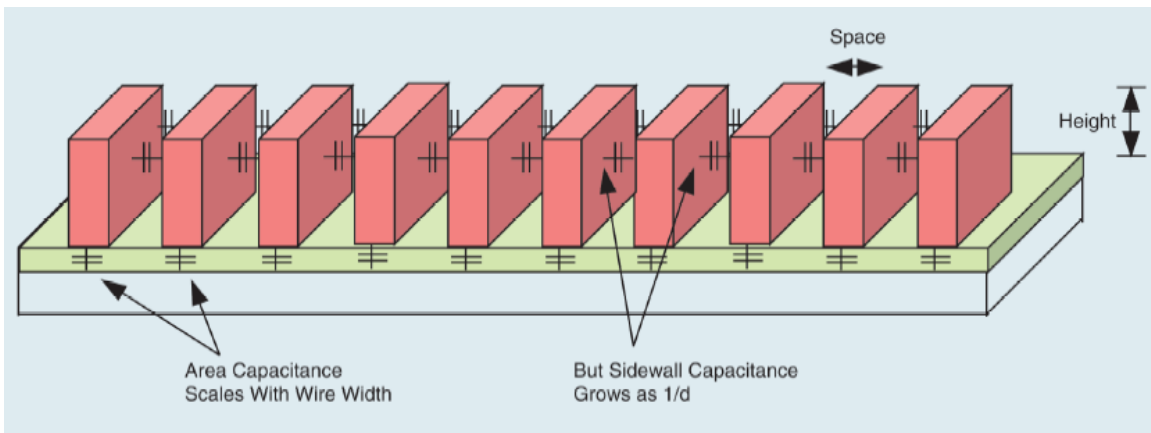


**Figure 3.2** The Deep Sub-Micron technology [18].

## 3.2 Energy Consumption in Deep Submicron Buses

Consider a bus line carrying a binary sequence $x(t)$, $x(t+1)$, $x(t+2)$ … . The energy consumption at time $k$ for charging and discharging the parasitic capacitance between the line and ground depends only on the two values $x(k-1) = u^0$ and $x(k) = u^N$. In other words, the transition cost depends on the initial and final values of the bus. Equation 3.1 generalizes this property to $n$ bus lines. In this equation, $u^0 = [u_1^0 \ u_2^0 \ ... \ u_n^0]$ is the initial state of the bus, and $u^N = [u_1^N \ u_2^N \ ... \ u_n^N]$ is the final state.

$$T_a = \sum_{i=1}^{n} u_i^0 \oplus u_i^N \tag{3.1}$$

The transition activity $T_a$ of a circuit node (or line) is a useful power measure when the node (line) is decoupled from other active nodes in the circuit [1]. For such nodes the power consumption can be simply computed by $E = T_a \ C_L \ V^2_{dd} /2$. Equation 3.2 computes the energy consumption for a bus with $n$ lines. In this equation $C_L$ is the capacitance between the node and ground.

$$E(w \rightarrow z) = T_a C_L V_{dd}^2 / 2 = \left( \sum_{i=1}^{n} u_i^0 \oplus u_i^N \right) C_L V_{dd}^2 / 2 \tag{3.2}$$

Coupling between nodes implies that power dissipation depends also on their cross-activities and therefore, Equation 3.2 is not valid anymore. In many cases like the deep Submicron Buses, coupling between lines is even stronger than the coupling between a line and the ground.

Figure 3.3 shows an approximate model for a DSM technology bus [1-3]. This model considers the boundary capacitors due to fringing effects, capacitors due to coupling between a line and the ground, and coupling between two adjacent lines.
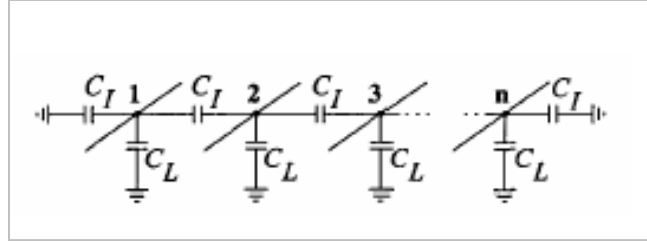


**Figure 3.3** A DSM bus model [3].

The energy consumption during transition from the initial bus state $u^0$ to the final bus state $u^N$ is given by [3, 6]:

$$E(u^0 \rightarrow u^N) = (u^0 - u^N)^T \cdot C \cdot (u^0 - u^N)^T V_{dd}^2 / 2 \tag{3.3}$$

Equation 3.4 shows the matrix C, where $\lambda = \dfrac{c_I}{c_L}$. $c_I$ is the *inter-wire capacitance* per unit length between adjacent lines and $c_L$ is the *parasitic capacitance* per unit length between each line and ground. The constant $\lambda$ is a real non-negative parameter that depends on physical parameters of lines, such as geometry, size, distance between the lines and the manufacturing technology. For modern deep sub-micrometer technologies, $\lambda$ can be as high as 8 (for example in 0.13- m technologies).

$$C = \begin{bmatrix} 1+2\lambda & -\lambda & 0 & 0 & ... & 0 \\ -\lambda & 1+2\lambda & -\lambda & 0 & ... & 0 \\ 0 & -\lambda & 1+2\lambda & -\lambda & ... & 0 \\ : & : & : & : & : & : \\ 0 & ... & 0 & -\lambda & 1+2\lambda & -\lambda \\ 0 & ... & 0 & 0 & -\lambda & 1+2\lambda \end{bmatrix} \cdot c_L \tag{3.4}$$

For the obsolete non-submicron (NSM) technologies λ is practically zero and *C* reduces to a scalar matrix. Thus, in the case of DSM technologies (λ > 0), unlike the case of NSM (λ = 0), the cost function has terms corresponding to interactions between the values transmitted on different lines. Figure 3.4 shows more details of the approximate model of a DSM bus with two parallel lines [4]. There is a driver for each line (Each line is connected to $V_{dd}$). A driver can be modeled by a voltage source with a series resistance $r_d$.
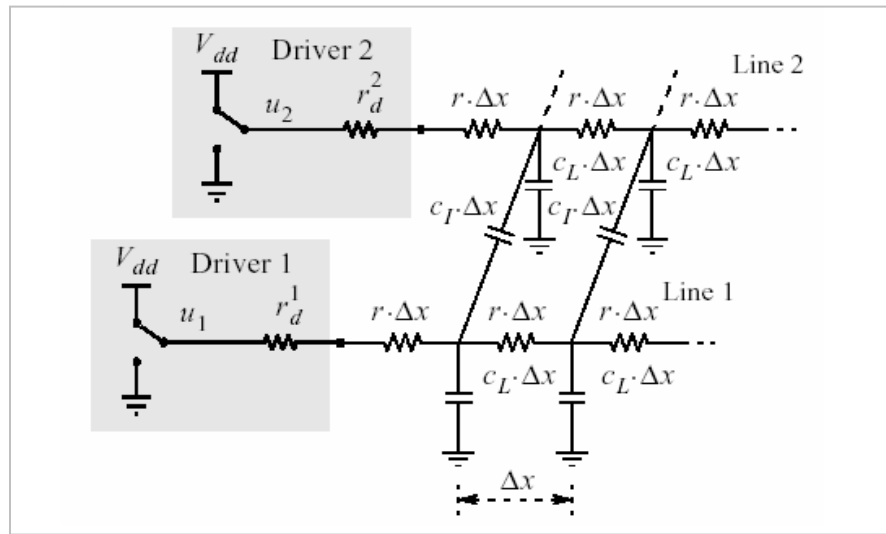


**Figure 3.4** DSM bus model for a bus with two parallel lines [4].

## 3.3 Delay in Deep Submicron Buses

A significant noise source in current processes is the coupling between lines. This coupling is even stronger than the coupling between a line and the ground. Consider a bus with *n* lines, each of length *L*. Let $u^0 = [u_1^0 \ u_2^0 \ ... \ u_n^0]$ be the initial condition of the bus, where $u_k^0 = V_k(L,0)$, $u_k^0 \in \{0,1\}$ is the voltage at the start of the line *k* of the bus. Also,

$u^N = [u_1^N \ u_2^N \ \ldots \ u_n^N]$ is the final condition of the bus, where $u_k^N = V_k(L,\infty)$, $u_k^N \in \{0,1\}$ is the voltage at the end of line $k$ of the bus (Figure 3.4).

The delay of each line $k$ is notified with $D_k$. The sum of the delays in the bus is computed by Equation 3.5, as follows [4, 5]. In this equation, $r$ is the resistance of line per unit length. Also, $r_T$ is a constant depending on resistance of each line and the line driver per unit length, and the line length.

$$\sum_{k=1}^{n} D_k\left(u^0, u^N\right) = \left(u^0 - u^N\right)^T \cdot C \cdot \left(u^0 - u^N\right) \cdot L \cdot r_T \tag{3.5}$$

$$r_T = r_d + \frac{L}{2} \cdot r$$

Equation 3.6 shows the matrix C. In this equation, $\lambda = \dfrac{c_I}{c_L}$, $c_I$ is the *inter-wire capacitance* per unit length between adjacent lines and $c_L$ is the *parasitic capacitance* per unit length between each line and ground.

$$C = \begin{bmatrix} 1+2\lambda & -\lambda & 0 & 0 & \ldots & 0 \\ -\lambda & 1+2\lambda & -\lambda & 0 & \ldots & 0 \\ 0 & -\lambda & 1+2\lambda & -\lambda & \ldots & 0 \\ : & : & : & : & : & : \\ 0 & \ldots & 0 & -\lambda & 1+2\lambda & -\lambda \\ 0 & \ldots & 0 & 0 & -\lambda & 1+2\lambda \end{bmatrix} \cdot c_L \tag{3.6}$$

## 3.4 Low Transition-Activity Encoding for Deep DSM Buses

Equations 3.3 and 3.5 show that for DSM buses, energy and delay depend on the sequence of data transmitted on the bus. Thus, we may decrease energy and delay on the bus by controlling the sequence of data transmitted on the bus.

Encoding is an efficient way for controlling the data sequence that is transmitted on the bus. Using encoding we may map the high cost codewords to some other codewords with lower costs. One approach is to add redundancy in the form of extra bus lines. To be effective, the encoder and decoder systems should have small energy consumption and delay values [1, 7-15].

*Definition* 1: Code $C$ is a mapping from elements in set $Q_m = \{0, 1\}^m$ to the elements in set $Q_n = \{0, 1\}^n$, where $n > m$. Each sequence $d \in Q_m$ is mapped to one and only one element $d' \in Q_n$ that satisfies a property of interest. As a result, some elements of $Q_n$ are not used at all.

Consider a bus with $n$ lines and let $Q_n = \{0, 1\}^n$ be the set of all binary vectors of length $n$. In an uncoded scheme, any data sequence $d \in Q_n$ can be transmitted on the bus. Figure 3.5 depicts both uncoded and coded transmission on the bus [19].

Adding extra lines to the bus, while the data stream is unchanged increases the capacity of the transmission channel (the bus). Consider an $n$-bit bus, which is supposed to transmit data sequences $d \in Q_n$. The bus has the potential to transmit $2^n$ different sequences, while there are only $2^m < 2^n$ different data sequences. Thus, some bus states are not used. One may choose the uncoded bus states to be the expensive ones.
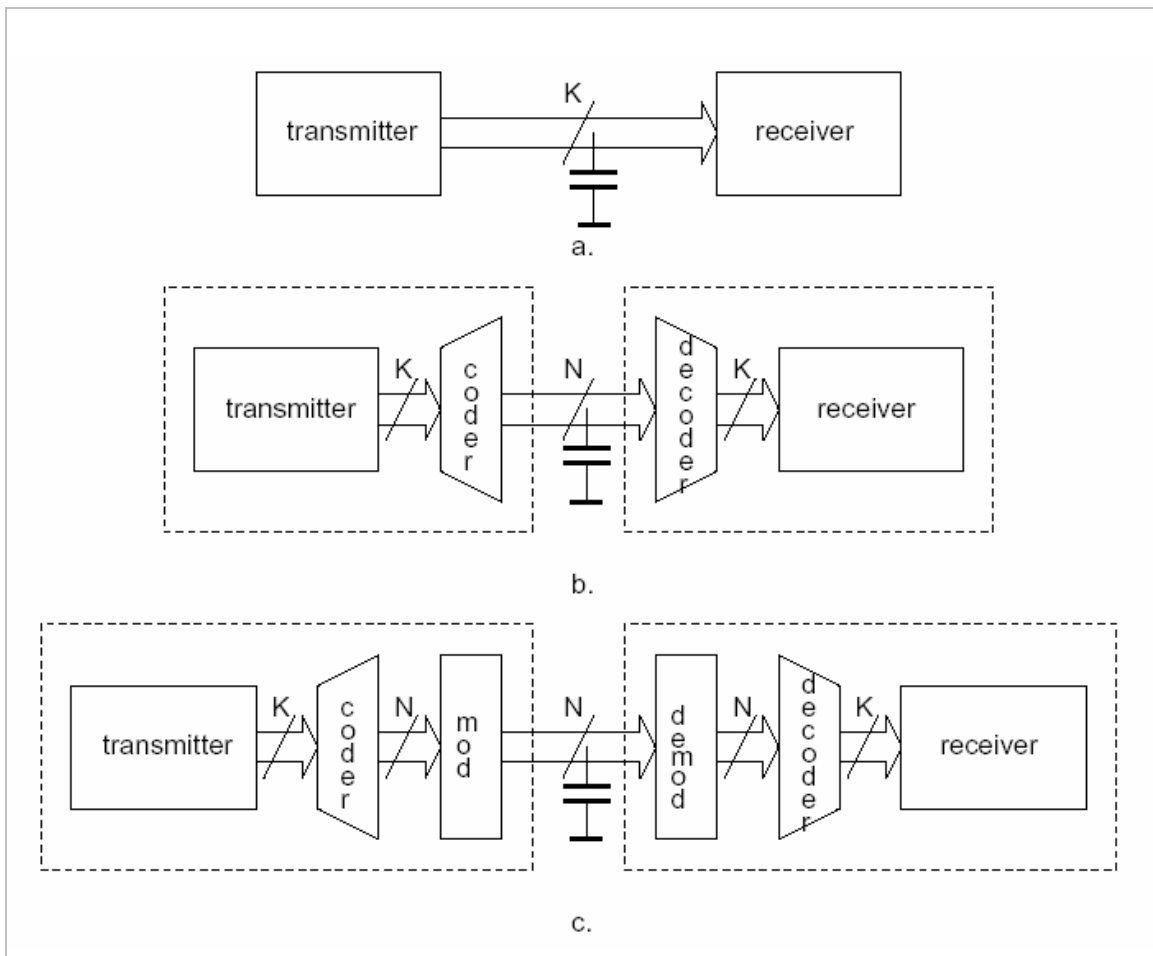
**Figure 3.5 a)** Transmit uncoded data, **b)** Transmit coded data, **c)** Signaling. [19]

## 3.5 Differential Low-Weight Encoding Approach

### 3.5.1 Encoding Scheme

The *Bus state* at time $t$ is the sequence which is on the bus at time $t$. The encoding scheme [1] includes the following steps:

1.  Extend the bus from $m$ lines to $n > m$ lines.

2.  Choose $2^m$ codewords out of all possible codewords of length $n > m$ such that the Hamming weight for the selected codewords are smaller than a given threshold $w$. (choose codewords of low Hamming weights).

3. Map each bus input data $d$ of length $m$ to a valid codeword $c$ of length $n$, using function $F$. (Function $F$ uses a table to complete the mapping).

4. XOR $s_t$, the bus state at the current time $t$ with the codeword corresponding to the bus input data. So, put $s_{t+1} = c \oplus s_t$ on the bus.

As the code words have small weight, there will be small number of transitions on the bus, and so the transmission cost decreases. Figure 3.6 depicts the encoding and decoding schemes in the differential low-weight encoding approach.

**3.5.2 Decoding Scheme**

Each received codeword ($s_{t+1}$) is decoded [1] to obtain the corresponding input data, using the following steps:

1. Use delay to access the previous bus state $s_t$ and compute codeword $c = s_{t+1} \oplus s_t$.

2. Use the function $F$ to find the corresponding data input $d$ for codeword $c$.

Figure 3.6 depicts the encoding and decoding schemes in the differential low-weight encoding approach.
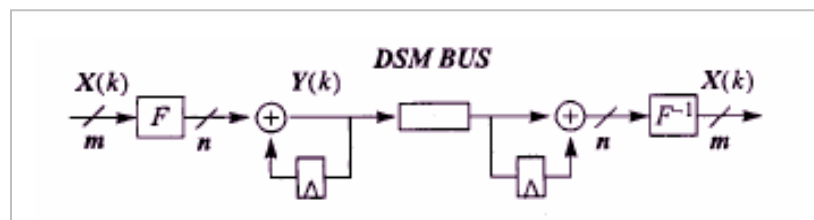


**Figure 3.6** The differential low-weight encoding and decoding [1].

## 3.6 Energy Reduction by Differential Low Weight Encoding

Figure 3.7 shows simulation results for the differential low weight scheme. In this figure $m/n$ is the code rate (number of useful bits transmitted per bus use). $E_b(Y)$ is energy per bit for the encoded scheme and $E_b(X)$ is energy per bit for uncoded data. Note that $E_b(Y) = E_b(X)$ when no bus lines or too many bus lines are added. For example, for 4-bit data the optimal energy reductions happens at rate $m/n = 0.5$, where 4 out of sixteen 4-bit sequences are chosen as codewords.
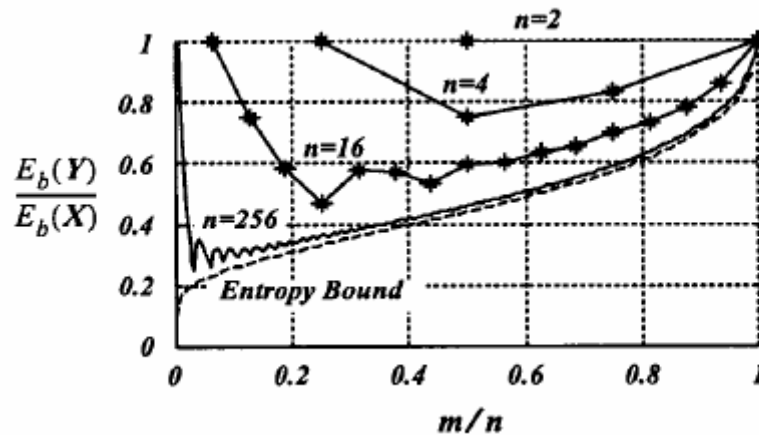


**Figure 3.7** Energy reduction by low-weight encoding and decoding [1].

# Chapter 4

# Energy and Delay Analysis for DSM Buses

In this chapter, we analyze the transmission energy and delay equations and show that the energy consumption and delay for a given DSM bus are different only by a scalar term. We introduce a common cost function which can control both. Reducing the cost function by encoding or any other approach results in both energy and delay reduction. We modify the energy and delay equations for DSM buses and develop new representations in terms of number of the same and opposite direction transitions on the bus and use them in our interconnect analysis. Then, we drive closed form formulas for computing the mean cost per bit over all possible initial bus states, for both differential low-weight encoding [1] and uncoded approaches.

## 4.1 Cost Function for Energy and Delay in DSM Buses

Using the equations introduced in Chapter 3, the energy cost for transition $T$ from the initial bus state $w = (w_1, w_2 \ldots w_n)$ to the final bus state $z = (z_1, z_2 \ldots z_n)$ is computed by Equation 4.1, where $z_i, w_i \in \{0, 1\}$, $T = z - w$, the constant $K = V_{dd}^2 / 2$ is a scalar depending on the driver voltage and $C$ is the capacity matrix.

$$E(z \rightarrow w) = f(T,C) \cdot K$$

$$f(T,C) = T^T \cdot C \cdot T \qquad (4.1)$$

$$C = \begin{bmatrix} 1+2\lambda & -\lambda & 0 & 0 & \ldots & 0 \\ -\lambda & 1+2\lambda & -\lambda & 0 & \ldots & 0 \\ 0 & -\lambda & 1+2\lambda & -\lambda & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & 0 & -\lambda & 1+2\lambda & -\lambda \\ 0 & \ldots & 0 & 0 & -\lambda & 1+2\lambda \end{bmatrix} \cdot c_L$$

Similarly, the delay is computed by Equation 4.2, where $T = z - w$ and $k' = L \cdot r_T$ is a scalar depending on the length and resistance per length for each bus line. Note that the capacitance matrix $C$ and the scalars $K$ and $K'$ depend on physical characteristics of the bus.

$$\sum_{k=1}^{n} D_k(z \rightarrow w) = f(T,C) \cdot K' \qquad (4.2)$$

$$f(T,C) = T^T \cdot C \cdot T$$

$$C = \begin{bmatrix} 1+2\lambda & -\lambda & 0 & 0 & \ldots & 0 \\ -\lambda & 1+2\lambda & -\lambda & 0 & \ldots & 0 \\ 0 & -\lambda & 1+2\lambda & -\lambda & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \ldots & 0 & -\lambda & 1+2\lambda & -\lambda \\ 0 & \ldots & 0 & 0 & -\lambda & 1+2\lambda \end{bmatrix} \cdot c_L$$

Equations 4.1 and 4.2 indicate that for a given DSM bus, the energy and delay are different only by a scalar term and depend on the function $f(T, C)$. This function is a measure of the transmission energy and delay. We call it the *cost function* and we focus on decreasing the transmission energy and delay cost on the DSM bus by decreasing this

cost function. Cost function $f(T, C)$ depends on the sequence of the data transmitted on the bus. Thus, we may decrease $f(T, C)$ and consequently, the transmission energy and delay on the DSM buses by controlling the sequence of data transmitted on the bus. For a DSM bus with $n$ lines, the transition vector is denoted by $T = (t_1, t_2 \dots t_n)$, where $t_i$ shows the transition activity for line $i$. Therefore, $f(T, C)$ can be computed as follows:

$$f(T,C) = T^T \cdot C \cdot T$$

$$= [t_1 \ t_2 \ t_3 \ \dots \ t_n] \cdot \begin{bmatrix} 1+2\lambda & -\lambda & 0 & 0 & \dots & 0 \\ -\lambda & 1+2\lambda & -\lambda & 0 & \dots & 0 \\ 0 & -\lambda & 1+2\lambda & -\lambda & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & 0 & -\lambda & 1+2\lambda & -\lambda \\ 0 & \dots & 0 & 0 & -\lambda & 1+2\lambda \end{bmatrix} \cdot c_L \cdot \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_n \end{bmatrix} \quad (4.3)$$

Divide both sides of Equation 4.3 by constant $c_L$ results in Equation 4.4.

$$f(T,C)/C_L = \begin{bmatrix} (1+2\lambda) \cdot t_1 + (-\lambda) \cdot t_2 \\ -\lambda \cdot t_1 + (1+2\lambda) \cdot t_2 + (-\lambda) \cdot t_3 \\ -\lambda \cdot t_2 + (1+2\lambda) \cdot t_3 + (-\lambda) \cdot t_4 \\ -\lambda \cdot t_3 + (1+2\lambda) \cdot t_4 + (-\lambda) \cdot t_5 \\ \vdots \\ -\lambda \cdot t_{n-1} + (1+2\lambda) \cdot t_n \end{bmatrix}^T \cdot \begin{bmatrix} t_1 \\ t_2 \\ t_3 \\ \vdots \\ t_n \end{bmatrix} \quad (4.4)$$

And, we complete the matrix multiplication and simplify further to get Equation 4.5.

$$f(T,C)/C_L = \left((1+2\lambda)\cdot t_1^2 + (-\lambda)\cdot t_2 \cdot t_1\right) + \left((-\lambda)\cdot t_1 \cdot t_2 + (1+2\lambda)\cdot t_2^2 + (-\lambda)\cdot t_3 \cdot t_2\right)$$
$$+\left((-\lambda)\cdot t_2 \cdot t_3 + (1+2\lambda)\cdot t_3^2 + (-\lambda)\cdot t_4 \cdot t_3\right) + \left((-\lambda)\cdot t_3 \cdot t_4 + (1+2\lambda)\cdot t_4^2 + (-\lambda)\cdot t_5 \cdot t_4\right) + \dots$$
$$+\left((-\lambda)\cdot t_{n-1} \cdot t_n + (1+2\lambda)\cdot t_n^2\right)$$

$$f(T,C)/C_L = (1+2\lambda)\cdot\left(t_1^2 + t_2^2 + \dots + t_n^2\right) + (-2\lambda)\left(t_1 \cdot t_2 + t_2 \cdot t_3 + t_3 \cdot t_4 + \dots + t_{n-1}\cdot t_n\right) \qquad (4.5)$$

Knowing that $T = (t_1, t_2 \dots t_n) = z\text{-}w$, each $t_i$ for $1 \le i \le n$ could have the following values:

$$t_i = \begin{cases} 0 & \text{if } w_i = z_i \ (\textit{no transition}) \\ 1 & \text{if } w_i = 0 \text{ and } z_i = 1 \ (\textit{transition from 0 to 1}) \\ -1 & \text{if } w_i = 1 \text{ and } z_i = 0 \ (\textit{transition from 1 to 0}) \end{cases} \qquad (4.6)$$

Using Equations 4.5 and 4.6, we can see that the '*no-transition* case' leaves $f(T,C)/C_L$ unchanged, and so imposes no extra energy or delay cost on the system. However, each transition on a line adds as much as $(1+2\lambda)$ to the cost function. Moreover, each same direction transition on adjacent lines (both lines go from 0 to 1 or vice versa) decreases the cost by $2\lambda$, while an opposite direction transition on two adjacent lines (one goes from 1 to 0 and the other from 0 to 1) increases the cost by $2\lambda$. In general, if codeword $c$ results in $T$ transitions on the DSM bus from which $t^+$ transitions are in the same direction and $t^-$ in the opposite direction, then the following formula gives the total transition cost on DSM buses for codeword $c$.

$$f(T,C)/C_L = (1+2\lambda)\times T + (2\lambda)\times t^+ - (2\lambda)\times t^- \qquad (4.7)$$

Our goal is to employ encoding to map high cost data sequences with large cost function values to codewords with lower costs. Thus, we are interested in codewords that result in minimum number of transitions (less transition activity). Meanwhile, we like the same direction transitions to happen on adjacent lines and the opposite direction transitions on non-adjacent lines, as far as possible. On the other hand, we cannot be very selective on the codewords, as it decreases the number of codewords selected for transmission on a DSM bus of a given length and so decreases the transmission rate.

**4.2 Transmission Cost Analysis**

In this section, we compare the mean cost for different transition codewords of Hamming weight $w_H$ to investigate how the location of ones in the codeword affects the transmission cost.

Let $\underline{0}$ denote a run of $x > 0$ zeros. $C_{cont} = \underline{0}11\ldots1\underline{0}$ denotes a transition codeword with a continuous run of 1s and $C_{discont} = \underline{0}1\underline{0}1\underline{0}\ldots\underline{0}1\underline{0}$ denotes a transition codeword for which there is at least one zero between any two consecutive 1s. Recall that a transition vector with adjacent 1s may lead to same direction or opposite direction adjacent transitions based on the initial bus state. The question is how does the transition codewords $C_{cont}$ and $C_{discont}$ compare, in terms of transmission cost. We start with computing the mean cost for a given codeword of Hamming weight $w_H$ over different initial bus states.

For a transition codeword with Hamming weight $w_H$, there are exactly $w_H$ nonzero transitions. Suppose that the number of the same direction and the opposite direction transitions are $x^+$ and $x^-$, respectively. Thus, using Equation 4.7, the cost function can be computed as follows.

$$f(T,C)/C_L = (1+2\lambda)\cdot w_H + (2\lambda)x^- - (2\lambda)x^+$$
$$= w_H + 2\lambda(w_H + x^- - x^+)$$

In the following sections, we first give closed form equations for the mean cost per bit of transition sequences like $C_{cont}$ and $C_{discont}$, and then we generalize these equations for codewords of any given form.

### 4.2.1 Cost Comparison for Transition Codewords $C_{discont}$ and $C_{cont}$

For $C_{discont}$, the transition cost is simply $(1+2\lambda) \times w_H$ independent of the initial bus state $S_o$. For $C_{cont}$ however, $S_o$ is an important factor. Any zero is equivalent to no transition and so no cost. For adjacent 1s, we should identify how many transitions are in the same direction and how many are not. Note that for a codeword of weight $w_H$, there may be at most $w_H$-1 pairs of consecutive transitions, some of which are in the same direction and some in the opposite direction.

The number of all possible combinations of selecting $i$ out of $n$ objects is computed by $\binom{n}{i} = \dfrac{n!}{i! \times (n-i)!}$ . Similarly, $N_1 = 2 \times \binom{w_H - 1}{x^-}$ gives the number of all possible combinations of $x^-$ opposite direction transitions out of $w_H$-1 pairs of consecutive transitions. Multiplication by a factor of 2 is required, as the transitions could be from 0 to 1 or from 1 to 0. Table 4.1 summarizes different values of $N_1$, as well as the cost per transition codeword for different values of $x^-$ and $x^-$.

For example, consider the transition sequence $C_{cont}$ = 111. There are $2 = 2 \times \binom{2}{0}$ possible initial bus states, namely $S_0$ = 000 and $S_0$ = 111 that result in $x^-$ = 0 and

consequently $x^+= 2$. There are four possible values that result in $x^- = 1$ and so $x^+ = 1$, which are $S_0 = 011$, $S_0 = 100$, $S_0 = 110$, $S_0 = 001$. Finally, $S_0 = 010$ and $S_0 = 101$ result in $x^- = 2$ and $x^+= 0$.

| $x^+$ | $x^-$ | #of possible combinations for $x^-$ opposite direction transitions | Transition Cost per codeword (computed by Equation 4.8) |
|---|---|---|---|
| $w_H$-1 | 0 | $\binom{w_H-1}{0}\times 2$ | $w_H+ 1\times(2\lambda)$ |
| $w_H$-2 | 1 | $\binom{w_H-1}{1}\times 2$ | $w_H+ 3\times (2\lambda)$ |
| $w_H$-3 | 2 | $\binom{w_H-1}{2}\times 2$ | $w_H+ 5\times (2\lambda)$ |
| ... | ... | ... | ... |
| 0 | $w_H$-1 | $\binom{w_H-1}{w_H-1}\times 2$ | $w_H+ (2(w_H-1)+1)\times(2\lambda)$ |

**Table 4.1** Transition cost for $C_{cont}$ and different initial bus states.

Recall the following properties of the combination operation.

$$\sum_{i=0}^{n}\binom{n}{i}= 2^n \tag{4.9}$$

$$\binom{n}{i}=\binom{n}{n-i}, \ i\leq n \tag{4.10}$$

For odd values of $n$, we have:

$$n = 2k +1 \Leftrightarrow \sum_{i=0}^{n}\binom{n}{i}=\binom{n}{0}+\binom{n}{1}+...+\binom{n}{(n-1)/2}+\binom{n}{(n+1)/2}+...+\binom{n}{n-1}+\binom{n}{n} \tag{4.11}$$

$$n = 2k + 1 \Leftrightarrow \sum_{i=0}^{n} (2i+1) \times \binom{n}{i} = 1 \times \binom{n}{0} + 3 \times \binom{n}{1} + \dots$$

$$+ (2\frac{n-1}{2}+1)\binom{n}{(n-1)/2} + (2\frac{n+1}{2}+1)\binom{n}{(n+1)/2} \qquad (4.12)$$

$$+ \dots$$

$$+ (2(n-1)+1) \times \binom{n}{n-1} + (2n+1) \times \binom{n}{n}$$

Combine Equations 4.9 - 4.11 to obtain the following equation.

$$n = 2k + 1 \Leftrightarrow \sum_{i=0}^{n} \binom{n}{i} = 2\binom{n}{0} + 2\binom{n}{1} + \dots + 2\binom{n}{(n-1)/2} = 2^n \qquad (4.13)$$

Similarly, for even values of $n$, we have:

$$n = 2k \Leftrightarrow \sum_{i=0}^{n} \binom{n}{i} = \binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{n/2-1} + \binom{n}{n/2} + \binom{n}{n/2+1} + \dots + \binom{n}{n-1} + \binom{n}{n} \quad (4.14)$$

$$n = 2k \Leftrightarrow \sum_{i=0}^{n} (2i+1) \times \binom{n}{i} = 1 \times \binom{n}{0} + 3 \times \binom{n}{1} + \dots$$

$$+ (2(n/2-1)+1)\binom{n}{n/2-1} + (2(n/2)+1)\binom{n}{n/2}$$

$$+ (2(n/2+1)+1)\binom{n}{n/2+1} \qquad (4.15)$$

$$+ \dots$$

$$+ (2(n-1)+1) \times \binom{n}{n-1} + (2n+1) \times \binom{n}{n}$$

And finally, combine Equations 4.9, 4.10 and 4.14 to obtain the following equation.

$$n = 2k \Leftrightarrow \sum_{i=0}^{n} \binom{n}{i} = 2\binom{n}{0} + 2\binom{n}{1} + \dots + 2\binom{n}{n/2-1} + \binom{n}{n/2} = 2^n \qquad (4.16)$$

Using Table 4.1, the total cost over all possible initial bus states, denoted by *COST* is computed as follows:

$$COST = \binom{w_H - 1}{0} \times 2 \times (w_H + 1(2\lambda)) + \binom{w_H - 1}{1} \times 2 \times (w_H + 3(2\lambda))$$
$$+ \binom{w_H - 1}{2} \times 2 \times (w_H + 5(2\lambda)) + ... \qquad (4.17)$$
$$+ \binom{w_H - 1}{w_H - 1} \times 2 \times (w_H + (2(w_H - 1) + 1)(2\lambda))$$

Use the associative property of summation to regroup the terms in parenthesis of Equation 4.17 and obtain two groups as factors of $w_H$ and $2\lambda$.

$$COST = \left( 2w_H \sum_{i=0}^{w_H - 1} \binom{w_H - 1}{i} + 2(2\lambda) \sum_{i=0}^{w_H - 1} (2i + 1) \times \binom{w_H - 1}{i} \right) \qquad (4.18)$$

Using Equation 4.9 we obtain:

$$COST = 2 \times w_H \times 2^{w_H - 1} + 2(2\lambda) \sum_{i=0}^{w_H - 1} (2i + 1) \times \binom{w_H - 1}{i} \qquad (4.19)$$

Now, we simplify the summation term in Equation 4.19 to simplify *COST* further.

For odd values of $w_H$-1, replace *n* with $w_H$-1 in Equation 4.12.

$$\sum_{i=0}^{w_H-1}(2i+1)\times\binom{w_H-1}{i}=1\times\binom{w_H-1}{0}+3\times\binom{w_H-1}{1}+...$$

$$+(2\frac{(w_H-1)-1}{2}+1)\binom{w_H-1}{(w_H-1-1)/2}$$

$$+(2\frac{(w_H-1)+1}{2}+1)\binom{w_H-1}{(w_H-1+1)/2}+... \qquad (4.20)$$

$$+(2(w_H-1-1)+1)\times\binom{w_H-1}{w_H-2}$$

$$+(2(w_H-1)+1)\times\binom{w_H-1}{w_H-1}$$

And use Equation 4.10 to get:

$$\sum_{i=0}^{w_H-1}(2i+1)\binom{w_H-1}{i}=(1+2(w_H-1)+1)\binom{w_H-1}{0}+(3+2(w_H-1-1)+1)\binom{w_H-1}{1}+...$$

$$+(2\frac{(w_H-1)-1}{2}+1+2\frac{(w_H-1)+1}{2}+1)\binom{w_H-1}{(w_H-1-1)/2}$$

$$=2(w_H)\binom{w_H-1}{0}+2(w_H)\binom{w_H-1}{1}+...$$

$$+2(w_H)\binom{w_H-1}{(w_H-2)/2}$$

$$=(w_H)(2^{w_H-1})$$

So, we finally obtain:

$$w_H = 2k \Leftrightarrow \sum_{i=0}^{w_H-1} (2i+1)\binom{w_H-1}{i} = (w_H)(2^{w_H-1}) \tag{4.21}$$

Similarly, for even values of $w_H$-1 we obtain Equation 4.22 as follows.

$$\sum_{i=0}^{w_H-1} (2i+1)\binom{w_H-1}{i} = (1+2(w_H-1)+1)\binom{w_H-1}{0} + (3+2((w_H-1)-1)+1)\binom{w_H-1}{1} + ...$$

$$+ (2(\frac{w_H-1}{2}-1)+1+2(\frac{w_H-1}{2}+1)+1))\binom{w_H-1}{(w_H-1)/2-1}$$

$$+ (2(w_H-1/2)+1)\binom{w_H-1}{(w_H-1)/2}$$

$$= 2(w_H)\binom{w_H-1}{0} + 2(w_H)\binom{w_H-1}{1} + ...$$

$$+ 2(w_H)\binom{w_H-1}{(w_H-1)/2-1} + (2(w_H-1/2)+1)\binom{w_H-1}{(w_H-1)/2}$$

$$= w_H(2^{w_H-1})$$

$$w_H = 2k+1 \Leftrightarrow \sum_{i=0}^{w_H-1} (2i+1)\binom{w_H-1}{i} = (w_H)(2^{w_H-1}) \tag{4.22}$$

Equations 4.21 and 4.22 indicate that:

$$\sum_{i=0}^{w_H-1} (2i+1)\binom{w_H-1}{i} = (w_H)(2^{w_H-1}) \tag{4.23}$$

Combining Equations 4.19 and 4.23, results in:

$$COST = 2 \times w_H \times 2^{w_H - 1} + 2(2\lambda) \sum_{i=0}^{w_H - 1} (2i + 1) \times \binom{w_H - 1}{i}$$
$$= w_H \times 2^{w_H} + (2\lambda) w_H \times 2^{w_H} \tag{4.24}$$
$$= (1 + 2\lambda) \times w_H \times 2^{w_H}$$

There are $2^{w_H}$ different initial bus states ($S_0$). Therefore, the mean cost for $C_{cont}$ over different initial bus states, denoted by $\overline{Cost}_{C_{cont}}$ is computed as follows.

$$\overline{Cost}_{C_{cont}} = COST / 2^{w_H}$$
$$= w_H (1 + 2\lambda) \tag{4.25}$$

Equation 4.25 shows that the mean transition cost over all possible original bus states ($S_o$) is simply $w_H (1 + 2\lambda)$, for both $C_{discont}$, and $C_{cont}$.

### 4.2.2 Mean Transition Cost for Codewords

We can generalize the above property to any codeword of hamming length $w_H$, and of the general from $C_l = \underline{1}\ \underline{0}\ \underline{1}...$, where $\underline{0}$ and $\underline{1}$ denote a run of $x > 0$ zeros and ones, respectively. The integer $l$ shows the number of runs of ones (number of $\underline{1}$s) in the codeword.

***Theorem* 1:** The mean cost per bit, over all possible initial bus states is the same for all codewords of equal Hamming weight, and is computed by $w_H (1 + 2\lambda)$.

*Proof:*

We use *mathematical inference* to prove that the cost per bit for $C_{general}$ is equal to $w_H$ $(1+ 2\lambda)$.

    a. In Section 4.2.1 we showed that the mean transition cost over all possible original bus states $(S_o)$ for $C_1 = C_{cont}$ is $w_H$ $(1+ 2\lambda)$. Thus, the initial inference condition is satisfied.

    b. We show that if the mean transition cost over all possible original bus states for $C_k$ is $w_H \times(1+ 2\lambda)$ then it is so, for $C_{k+1} = C_k \underline{0}\ \underline{1}$, as well. The proof is as follows.

Table 4.1 indicates that the transmission cost may be one of the values $c_0 = w_H+1(2\lambda)$, $c_1 = w_H+3(2\lambda)$... $c_{wH\ -1} = w_H+ (2(w_H-1) +1) (2\lambda)$ based on the number of the same and opposite direction transitions. Suppose $N_i$ is the number of initial bus states that result in cost $c_i$ and the Hamming weight for $C_k$ denoted by $w_H$ $(C_k)$ is equal to $x_1$. For transition sequence $C_k$, we have:

$$COST = N_0 \times (x_1 + 1(2\lambda)) + N_1 \times (x_1 + 3(2\lambda)) + N_2 \times (x_1 + 5(2\lambda)) + ...$$
$$+ N_{x_1-1} \times (x_1 + (2(x_1 - 1) + 1)(2\lambda)) \tag{4.26}$$

According to the inference assumption,

$$COST = x_1(1 + 2\lambda) \times 2^{x_1} \tag{4.27}$$

And so, the right hand sides of Equations 4.26 and 4.27 are equal and we have the following:

$$N_0 \times (x_1 + 1(2\lambda)) + N_1 \times (x_1 + 3(2\lambda)) + ... + N_{x_1-1} \times (x_1 + (2(x_1-1)+1)(2\lambda))$$
$$= x_1(1 + 2\lambda) \times 2^{x_1}$$

(4.28)

Suppose the length for the $k+1^{\text{th}}$ run of ones in $C_{k+1}$ is $x_2$. Thus, the Hamming weight for $C_{k+1}$ is $x_1 + x_2$ and from Equation 4.25 the following Equation holds:

$$\overline{\text{Cost}}_{C_{k+1}} = COST_{C_{k+1}} / 2^{x_1 + x_2}$$

(4.29)

Using Equation 4.26 with $w_H = x_2$, for only the last run of ones in $C_{k+1}$, we have:

$$x_2(1+2\lambda) \times 2^{x_2} = \binom{x_2 - 1}{0} \times 2 \times (x_2 + 1(2\lambda)) + \binom{x_2 - 1}{1} \times 2 \times (x_2 + 3(2\lambda)) + ...$$
$$+ \binom{x_2 - 1}{x_2 - 1} \times 2 \times (x_2 + (2(x_2-1)+1)(2\lambda))$$

(4.30)

To compute the total cost for $C_{k+1} = C_k \; \underline{0} \; \underline{1}$, we should consider $C_k$, as well as the last run of ones. We compute the cost for each of these separately, and then add up the individual costs to obtain the total cost. Equation 4.31 gives the total cost for $C_{k+1}$.

$$COST_C = N_0 \times 2\binom{x_2-1}{0} \times (x_1 + 1(2\lambda) + x_2 + 1(2\lambda))$$

$$+ N_0 \times 2\binom{x_2-1}{1} \times (x_1 + 1(2\lambda) + x_2 + 3(2\lambda)) + ...$$

$$+ N_0 \times 2\binom{x_2-1}{x_2-1} \times (x_1 + 1(2\lambda) + x_2 + (2(x_2-1)+1)(2\lambda))$$

$$+ N_1 \times 2\binom{x_2-1}{0} \times (x_1 + 3(2\lambda) + x_2 + 1(2\lambda))$$

$$+ N_1 \times 2\binom{x_2-1}{1} \times (x_1 + 3(2\lambda) + x_2 + 3(2\lambda)) + ...$$

$$+ N_1 \times 2\binom{x_2-1}{x_2-1} \times (x_1 + 3(2\lambda) + x_2 + (2(x_2-1)+1)(2\lambda)) + ...$$

$$+ N_{x_1-1} \times 2\binom{x_2-1}{0} \times (x_1 + (2(x_1-1)+1)(2\lambda) + x_2 + 1(2\lambda))$$

$$+ N_{x_1-1} \times 2\binom{x_2-1}{1} \times (x_1 + (2(x_1-1)+1)(2\lambda) + x_2 + 3(2\lambda)) + ...$$

$$\tag{4.31}$$

$$+ N_{x_1-1} \times 2\binom{x_2-1}{x_2-1} \times (x_1 + (2(x_1-1)+1)(2\lambda) + x_2 + (2(x_2-1)+1)(2\lambda))$$

Using associative property of the summation, we place the first terms in the entire parenthesis in one group, and the second terms in another one to obtain:

$$COST_{C_{k+1}} = N_0 \times 2 \times (x_1 + 1(2\lambda)) \times \left[\binom{x_2-1}{0} + \binom{x_2-1}{1} + ... + \binom{x_2-1}{x_2-1}\right]$$

$$+ N_1 \times 2 \times (x_1 + 3(2\lambda)) \times \left[\binom{x_2-1}{0} + \binom{x_2-1}{1} + ... + \binom{x_2-1}{x_2-1}\right] + ...$$

$$+ N_{x_1-1} \times 2 \times (x_1 + (2(x_1-1)+1)(2\lambda)) \times \left[\binom{x_2-1}{0} + \binom{x_2-1}{1} + ... + \binom{x_2-1}{x_2-1}\right]$$

$$+ N_0 \times \left[2\binom{x_2-1}{0} \times (x_2 + 1(2\lambda)) + 2\binom{x_2-1}{1} \times (x_2 + 3(2\lambda)) + ... + 2\binom{x_2-1}{x_2-1} \times (x_2 + (2(x_2-1)+1)(2\lambda))\right]$$

$$+ N_1 \times \left[2\binom{x_2-1}{0} \times (x_2 + 1(2\lambda)) + 2\binom{x_2-1}{1} \times (x_2 + 3(2\lambda)) + ... + 2\binom{x_2-1}{x_2-1} \times (x_2 + (2(x_2-1)+1)(2\lambda))\right] + ...$$

$$+ N_{x_1-1} \times \left[2\binom{x_2-1}{0} \times (x_2 + 1(2\lambda)) + 2\binom{x_2-1}{1} \times (x_2 + 3(2\lambda)) + ... + 2\binom{x_2-1}{x_2-1} \times (x_2 + (2(x_2-1)+1)(2\lambda))\right]$$

Then, we simplify the terms inside the brackets using Equations 4.9 and 4.30 and factor these terms out.

$$COST_{C_{k+1}} = 2 \times 2^{x2-1} \times \left[ N_0 \times (x_1 + 1(2\lambda)) + N_1 \times (x_1 + 3(2\lambda)) + ... + N_{x1-1} \times (x_1 + (2(x_1-1)+1)(2\lambda)) \right]$$
$$+ \left( x_2 (1 + 2\lambda) \times 2^{x2} \right) \times \left( N_0 + N_1 + ... + N_{x1-1} \right)$$

Note that $N_0 + N_1 + ... + N_{x1-1} = 2^{x1-1}$ and we can simplify the terms inside the brackets, using Equation 4.28.

$$COST_{C_{k+1}} = 2 \times 2^{x2-1} \times \left( x_1 (1 + 2\lambda) \times 2^{x1} \right) + \left( x_2 (1 + 2\lambda) \times 2^{x2} \right) \times 2 \times 2^{x1-1}$$
$$= 2^{x1+x2} \times x_1 (1 + 2\lambda) + 2^{x1+x2} \times x_2 (1 + 2\lambda) \qquad (4.32)$$
$$= 2^{x1+x2} (1 + 2\lambda)(x_1 + x_2)$$

And finally from Equation 4.29,

$$\overline{Cost}_{C_{k+1}} = COST_{C_{k+1}} / 2^{x1+x2}$$
$$= (1 + 2\lambda)(x_1 + x_2) \qquad (4.33)$$
$$= (1 + 2\lambda) w_H (C_{k+1})$$

Equation 4.33 proves step $b$ of the inference and so we can conclude that the mean transition cost per bit, over all possible initial bus states ($S_o$) is simply $w_H (1 + 2\lambda)$, for any given codeword $C_l$. ∎

## 4.3 Transmission Cost for Differential Low-Weight Encoding

In this scheme, only the codewords with Hamming weight smaller than a threshold $w_{max}$ are selected. As explained in the previous section, the mean cost over initial bus states is the same for all the codewords of the same Hamming weight $w_H$ and can be computed by $w_H (1+2\lambda)$. Thus, mean energy per bit for a differential low-weight code of length $n$ is as follows:

$$\overline{E}_b = \frac{\sum_{i=0}^{w_{max}} i \times \binom{n}{i} \times (1+2\lambda)}{n \sum_{i=0}^{w_{max}} \binom{n}{i}}$$

$$= \frac{1+2\lambda}{n \sum_{i=0}^{w_{max}} \binom{n}{i}} \sum_{i=0}^{w_{max}} i \times \binom{n}{i}$$

And so,

$$\overline{E}_b = \frac{1+2\lambda}{n \sum_{i=0}^{w_{max}} \binom{n}{i}} \sum_{i=0}^{w_{max}} i \times \binom{n}{i} \qquad (4.34)$$

## 4.4 Transmission Cost for Uncoded Scheme

In the uncoded scheme, every sequence in the set $S = \{0, 1\}^n$ may be transmitted on the bus. Thus, the transition vectors are all elements of set $S$ and the mean energy per bit is computed as follows:

$$\overline{E_b} = \frac{\sum\limits_{i=0}^{n} i \times \binom{n}{i} \times (1 + 2\lambda)}{n \sum\limits_{i=0}^{n} \binom{n}{i}}$$

$$= \frac{1 + 2\lambda}{n \sum\limits_{i=0}^{n} \binom{n}{i}} \sum\limits_{i=0}^{w_{\max}} i \times \binom{n}{i}$$

$$= \frac{1 + 2\lambda}{n \times 2^n} (\frac{n}{2} \times 2^n)$$

$$= \frac{1 + 2\lambda}{2}$$

$$= 0.5 + \lambda$$

$$\boxed{\overline{E_b} = 0.5 + \lambda} \tag{4.35}$$

As Equation 4.35 shows the mean cost per bit is a constant depending on $\lambda$ when no encoding is used. Note that transmission rate in this case is equal to 1. Thus, the rate is ideal, while the cost is high compared to the encoded case.

# Chapter 5

# The Proposed Encoding Scheme

As explained in the previous chapters, the differential low weight encoding is an efficient and low complexity scheme that achieves most of the transmission cost reduction possible, on DSM buses [1].

In this chapter, we propose an enumeration method to be used with the differential low-weight coding. This enumeration method reduces the mapping table size from $O(2^n)$ words to $O(n)$ words, for an $n$-bit bus. Thus, it reduces the memory complexity significantly, especially when $n$ is large. Furthermore, it facilitates less power dissipation and delay, as it eliminates the need for addressing and fetching data from very large tables in memory.

Furthermore, we exploit the same direction and opposite direction transitions to achieve more cost reduction. For each selected low weight codeword, we compute a *help codeword* (from those high weight codewords which are not selected) on the fly. The help codewords result in lower cost for some initial bus states. They are computed by simple operations like complement and rotate and so reduce the cost without any extra memory or bus lines.

This chapter is organized as follows. First we propose a new set for representing the codewords and the Low-Weight order for this set. We also define a number of operations on this set. Then, we explain the proposed encoding scheme and application of the Low-Weight order in the proposed enumeration process, in detail. At the end, we present the *Help codewords* and simulation results. For the terms and definitions used in the following sections, refer to Chapter 2.

## 5.1 Low-Weight Order

Let set $P^k = \{1, 2, 3… n\}^k$ denote all the *n*-ary permutations of $k$ symbols (not necessarily distinct) chosen from the alphabet $\{1, 2, 3… n\}$ such that the sum of the digits of each permuted sequence is smaller than or equal to *n*. Thus, permutation $p \in P^k$ is of length $k$ and may be denoted as $d_1, d_2…d_k$ where $d_i \in \{1, 2, 3… n\}$. For simplicity, we denote $p$ with $d_1 d_2…d_k$ when $n <10$ (no comma between symbols). We define the set $P^k$, $1 \leq k \leq n$ as the union of sets $P^1$, $P^2$... and $P^n$. Using the phrase representation introduced in Chapter 2.4, each permutation $p \in P^k$ may be expanded to obtain a binary sequence $c$. Note that the constraint on the sum of the symbols of $p \in P^k$ guarantees that the length of $c$ is smaller than or equal to *n* and transmitting it on a DSM bus requires no more than *n* lines. In other words, the length of the binary sequence corresponding to

each permutation is limited to *n*. We require this constraint, as we have a limited number of bus lines for data transmission.

Recall the lexicographic ordering introduced in Chapter 2.5. This ordering scheme considers the rightmost digit as the least significant digit (LSD) and the leftmost one as the most significant digit (MSD), in computing the lexicographic index (rank). Alternatively, we define the low-weight order for which the rightmost symbol is the MSD and the rank for $x = (x_1, x_2 \ldots x_k) \in P^k$, $1 \le k \le n$ is computed by the Equation 5.1. In this equation, $n^k{}_S(x_1, x_2 \ldots x_u)$ denotes the number of elements in $P^k$, $1 \le k \le n$ for which the first *u* coordinates are $(x_1, x_2 \ldots x_u)$. Also, $|P^k|$ denotes the cardinality of set $P^k$.

$$i_S(x) = \sum_{h=1}^{k-1} | P^h | + \sum_{j=1}^{k} \sum_{t=0}^{x_j - 1} n_S^k (x_1, x_2, \ldots, x_{j-1}, t) \tag{5.1}$$

Note that extend the Low-Weight order extends to different values of $1 \le k \le n$. Specifically, for $x = (x_1, x_2 \ldots x_k) \in P^{k1}$ and $y = (y_1, y_2 \ldots y_k) \in P^{k2}$ we have $x < y$ if $k1 < k2$. So, $f^{-1}{}_{phrase}(x)$ has a lower Hamming weight compared to $f^{-1}{}_{phrase}(y)$. We exploit this ordering to decrease the transmission cost by choosing the lower weight codewords before and sooner than the higher weight ones.

For example, Table 5.1 shows the elements of set $P^k$ with $1 \le k \le 5$ and from alphabet {1, 2, 3, 4, 5}, sorted in Low-Weight order.

| index | p | Index | p |
|---|---|---|---|
| 1 | 1 | 16 | 111 |
| 2 | 2 | 17 | 211 |
| 3 | 3 | 18 | 311 |
| 4 | 4 | 19 | 121 |
| 5 | 5 | 20 | 221 |
| 6 | 11 | 21 | 131 |
| 7 | 21 | 22 | 112 |
| 8 | 31 | 23 | 212 |
| 9 | 41 | 24 | 122 |
| 10 | 12 | 25 | 113 |
| 11 | 22 | 26 | 1111 |
| 12 | 32 | 27 | 2111 |
| 13 | 13 | 28 | 1211 |
| 14 | 23 | 29 | 1121 |
| 15 | 14 | 30 | 1112 |
|  |  | 31 | 11111 |

**Table 5.1** Low-Weight order for entries of set $P^k$, $1 \leq k \leq 5$.


## 5.2 Operations for Low-Weight Ordered Numbers

### 5.2.1 Low-Weight Count

Low-Weight count enables us to find the $n$-bit codeword $C_{i+1}$ that follows (appears 1codeword after) a given $n$-bit codeword $C_i$, in Low-Weight order. The algorithm is depicted in Figure 5.1. In this algorithm $C_i$ denotes the $f^{-1}_{phrase}$ of an element $p_i$ in set $P^k = \{1, 2, 3 \ldots n\}^k$, $1 \leq k \leq n$.

if $C_i$ terminates in 0   /*case 1*/

   Shift $C_i$ to right by 1 to obtain $C_{i+1}$.

else

   Find the first phrase $p$ in $C_i$.

   if there exists a 1 after phrase $p$ in $C_i$  /*case 2*/

     $C'$ = Shift only phrase $p$ to left by 1

   else   /*case 3*/

     $C'$ = Insert a 1 at the beginning of the sequence and remove a 0 in $p$

   $C_{i+1}$ = Move the zeros of the first phrase in $C'$ to the end of the sequence.

**Figure 5.1** The Low-Weight count algorithm.


Examples:

Case 1.   In Table 5.1, the $11^{th}$ entry is $C_i = f^{-1}_{phrase}$ (22) = 01010 and the $12^{th}$ entry is

    $C_{i+1}$= 00101 $= f^{-1}_{phrase}$ (32).

Case 2.   In Table 5.1, the $9^{th}$ element is $C_i = f^{-1}_{phrase}$ (41) = 00011, $p$ = 0001 (bits 1 to 4

    of $C_i$), $C'$ = 00101 and $C_{i+1}$= 10100 $= f^{-1}_{phrase}$ (12). Note that 12 is the $10^{th}$ entry in

    Table 5.1.

**Case 3.**   In Table 5.1, the $25^{th}$ element is $C_i = f^{-1}_{phrase}$ (113) = 11001, $p$ = 001 (bits 3 to

    5 of $C_i$), $C'$ = 11101 and $C_{i+1}$= 11110 $= f^{-1}_{phrase}$ (1111). Note that 1111 is the $26^{th}$

    entry in Table 5.1.

### 5.2.2 Low-Weight Division

Figure 5.2 shows the division algorithm. The Low-Weight division of an integer $x$ by a list of integers $(y_1, y_2 \ldots y_k)$ is denoted by $f_{div}(x, (y_1, y_2 \ldots y_k))$. Similar to normal integer division, this operation is implemented by successive subtractions. The only difference is that instead of subtracting a fixed integer (dividend) all the time, each time one of the elements in the dividend list is subtracted from the divider $x$.

The operation $f_{div}(x, (y_1, y_2 \ldots y_k))$ results in the quotient $q$ and remainder $r$ if and only if $x = y_1 + y_2 + \ldots + y_q + r$, where $r < y_q$.

/*Low-Weight Divide algorithm*/

i=1, q=0

While $x \geq y_i$

  $x = x - y_i$

  $q = q + 1$

  $i = i+1;$

r = x;

**Figure 5.2** The Low-Weight division algorithm.

### 5.2.3 Low-Weight Addition

Consider the first vector of weight $w_H$ in Low-Weight order for different values of $1 \leq w_H \leq n$. For example, for $n = 4$, these vectors are 1, 11, 111, and 1111. Recall that the corresponding binary codewords are 1000,1100,1110,1111, respectively. We denote such codewords with $\bar{1}_{w_H}$ to indicate that the number of ones is $w_H \leq n$.

Low-Weight addition enables us to find the $n$-bit codeword $C_{i+x}$ that appears $x$ (a given integer in base 10) codewords after $C_i = \bar{1}_{w_H}$, the first codeword of weight $w_H$ in Low-Weight order. This operation is required for the enumeration process and we explain it, in detail.

Note that the number of trailing zeros for $f_{phrase}^{-1}(\bar{1}_{w_H})$ is $n$- $w_H$. We define $y = n+1$- $w_H$. In fact, $y$ -1 is the number of trailing zeros for $f_{phrase}^{-1}(C_i)$. Consider $\bar{1}_{w_H}$ and its phrase representation $p = d_1, d_2 \dots d_{wH}$, where $d_i = 1$ for $1 \le i \le w_H$. The following rules are used to add $p$ with $a$ (an integer in base 10) in the Low-Weight addition.

**Rule 1:** if $a \in \{1, 2 \dots y\text{-}1\}$ then add integer '$a$' to the leftmost 1 in phrase representation $p$.

**Rule 2:** if $a = \sum\limits_{i=0}^{y-x} y-i$ for $y \ge x \ge 2$, then the 2$^{\text{nd}}$ leftmost digit in the phrase representation $p$ is increased by $x$ +1.

Define $f_1(y) = \sum\limits_{i=1}^{y} i = \dfrac{y(y+1)}{2}$,

**Rule 3:** if $X = \sum\limits_{i=x}^{y} f_1(i)$ for $y \ge x \ge 2$, then the 3rd leftmost digit in the phrase representation $p$ is increased by $(y\text{-}x)$ +1.

Define $f_2(y) = \sum\limits_{i=1}^{y} f_1(i)$,

**Rule 4:** if $X = \sum_{i=x}^{y} f_2(i)$ for $y \geq x \geq 2$, then the 4$^{th}$ leftmost digit in the phrase

representation $p$ is increased by $(y- x) +1$.

…

Define $f_k(y) = \sum_{i=1}^{y} f_{k-1}(i)$, $k = w_H - 2$,

**The final Rule:** if $X = \sum_{i=x}^{y} f_k(i)$ for $y \geq x \geq 2$, then the $w_H{}^{th}$ (last) leftmost digit in the

phrase representation $p$ is increased by $(y- x) +1$.

Note that $f_i(y-1) = f_i(y) - f_{i-1}(y)$ for $i = 1,2...k$.

For example, consider $p = 1111 = d_1 d_2 d_3 d_4$, the phrase representation of $c = 111100000$

for $n = 9$ and List 1 (at the end of this Chapter) which shows all codewords with $w_H = 4$

and $n = 9$ in Low-weight order. Recall that $y = n+1- w_H = 6$.

If $X = 1$ then $d_1$ increases by 1.

If $X = 2$ then $d_1$ increases by 2.

…

If $X = y -1 = 5$ then $d_1$ increases by 5.

If $X = y = 6$ then $d_2$ increases by 1.

If $X = y + (y - 1) = 6 + 5 = 11$ then $d_2$ increases by 2.

…

If $X = y + (y - 1) + … + 2 = 6 + 5 + 4 + 3 + 2 = 20$ then $d_2$ increases by 5.

If $X = f_1(y) = 6.7/2 = 21$ then $d_3$ increases by 1.

If $X = f_1(y) \; + f_1 (y - 1) = 21 + 15 = 36$ then $d_3$ increases by 2.

…

If $X = f_1(y) \; + f_1 (y - 1) + \ldots + f_1 (2) = 21 + 15 + 10 + 6 + 3 = 55$ then $d_3$ increases by 5.


If $X = f_2(y) = 21 + 15 + 10 + 6 + 3 + 1 = 56$ then $d_4$ increases by 1.

If $X = f_2(y) \; + f_2 (y - 1) = 56 + (56 - 21) = 91$ then $d_4$ increases by 2.

…

If $X = f_2(y) \; + f_2 (y - 1) + \ldots + f_2 (2) = 56 + 35 + \ldots = 125$ then $d_4$ increases by 5.


For ordinary integers (of any base), any given digit increases over equal distances determined by base. For example, let $x$ be a natural number. The integer 20 appears $x$ integers after 10 in base $x$. Similarly, the integer 30 appears $x$ integers after 20.

However, using the Low-Weight order, the digits of phrase representation $p$ of a given binary sequence $c$, do not increase on equal intervals. This fact is indicated by the rules of the Low-Weight addition introduced above. For example, consider List 1 at the end of this chapter. There are 5 integers between 1111 and 1211, while 10 integers appear between 1211 and 1311. Although a fixed interval over which a given digit increases does not exist, it is possible to predict how the interval changes for each digit of $p$ as the above rules imply.

To increase phrase $p$ of length $w_H$ by $X$ (in base 10), we compute the amount of increase for the rightmost digit in $p$ (the most significant digit in Low-Weight order sequences)

using the Low-Weight division operation. Then, we update $X$, and repeat the same procedure for the following digits until the leftmost digit (the least significant digit in Low-Weight order sequences) is processed or $X$ is zero. The following algorithm completes Low-Weight addition in $O(n \times w_H^2)$ time and $O(n)$ space.

---

/*Low-Weight increase by $X$ algorithm*/

/*$X \leq \binom{n}{w_H} - 1$ */

/* $p = d_1 d_2 \ldots d_{wH}$, $d_i = 1$ for $1 \leq i \leq w_H$*/

/*Find $y$, the number of trailing zeros of the …*/
/* sequence represented by $p$*/

$y = n - w_H + 1$

for each digit $d_i$ , $w_H \geq i \geq 2$

   $Y = (y, y\text{-}1, y\text{-}2 \ldots 2, 1)$

   for *counter* $= 1$ to $w_H\text{-}2$

$$Y_j = \sum_{i=j}^{y} Y_i$$

   Evaluate $f_{div}$ $(X, Y)$ to get $q$ and $r$

   /* Update $X$ */
   $X = r$

   /* Update phrase $p$*/
   $d_i = d_i + q$

   /* Update $y$ */
   $y = y - q$

$d_1 = d_1 + r$

---

**Figure 5.3** The Low-Weight addition algorithm.

### 5.2.4 The Low-Weight Subtract

Consider the phrase representation $p_0$ as the first vector of weight $1 \leq w_H \leq n$ in Low-Weight order. Also, consider $p$ as a given codeword of same Hamming weight as $p_0$. Suppose that codeword $p$ appears $cnt$ codewords after $p_0$, in Low-Weight order. The Low-Weight subtract operation takes $p_0$ and $p$ and computes $cnt$, the distance between $p_0$ and $p$.

/*Low-Weight subtract algorithm*/

/* $p = d_1 d_2 \ldots d_{wH}$, $d_i = 1$ for $1 \leq i \leq w_H$*/

$y = n - w_H + 1$

for each digit $d_i$, $w_H \geq i \geq 2$

    $cnt = 0$

    $Y = (y, y\text{-}1, y\text{-}2 \ldots 2, 1)$

    for $counter = 1$ to $w_H\text{-}2$

        $Y_j = \sum_{i=j}^{y} Y_i$

    $cnt = cnt + \sum_{j=1}^{d_i - 1} Y_j + d_1 - 1$

**Figure 5.4** The Low-Weight subtract algorithm.

### 5.3 The Proposed Encoding Scheme

For a transmitting $m$-bit data we require at least $m$ bus lines. We increase number of bus lines to $n > m$ lines, such that $m/n$ is greater than or equal to the desired transmission rate.

The encoding idea is to find a function that maps an $m$-bit data sequence to a low Hamming weight $n$-bit codeword.

Let $L1$ be the list of all binary sequences in $\{0, 1\}^m$, in normal ascending order. For example, using an indexing system starting at zero, the index for the all zeros sequence is zero and the index for the all ones sequence is $2^m$-1. Also, list $L2$ is the list of all $n$-ary sequences in $P^k$, $1\leq k\leq n$, in Low-Weight order. We expand (using $f^{-1}_{phrase}$) and map the $i^{th}$ element of $L2$ as the codeword for the $i^{th}$ element of $L1$, for $0\leq i\leq 2^m$-1. Table 5.2 shows the 5-bit codewords assigned to 4-bit data sequences. Note that the codewords are 5-bit binary sequences with low (0, 1 and 2) Hamming weights. This encoding scheme can be completed on the fly, using enumeration.

## 5.4 Enumeration Scheme

The question is how to map a given $m$-bit sequence to an $n$-bit codeword on the fly without storing the (data, codeword) pairs in a table. Recall that the table size grows exponentially with data length and eliminating the table is so useful in terms of memory complexity.

Let $L$ be a list whose $k^{th}$ element is the number of $n$-bit binary sequences with exactly $k$ ones (Hamming weight $w_H = k$). We denote the $k^{th}$ element of $L$ by $\binom{n}{k}$, and form another list $L_{cumulative}$ by cumulative summation of list $L$. Thus, the $k^{th}$ element of list $L_{cumulative}$ is $\sum_{i=0}^{k} \binom{n}{k}$ and shows the number of codewords with $w_H \leq k$. This list grows linearly with $w_H$ and does not need a large memory space. Furthermore, based on Equation 5.2, we may keep only half of the list and compute the rest on the fly and save more space.

$$\binom{n}{k1} = \binom{n}{k2}, k1+k2 = n \tag{5.2}$$

| Index | L1 Binary Data | L2 Codeword Phrase Representation | Codeword |
|---|---|---|---|
| 0 | 0000 | 0 | 00000 |
| 1 | 0001 | 1 | 10000 |
| 2 | 0010 | 2 | 01000 |
| 3 | 0011 | 3 | 00100 |
| 4 | 0100 | 4 | 00010 |
| 5 | 0101 | 5 | 00001 |
| 6 | 0110 | 11 | 11000 |
| 7 | 0111 | 21 | 01100 |
| 8 | 1000 | 31 | 00110 |
| 9 | 1001 | 41 | 00011 |
| 10 | 1010 | 12 | 10100 |
| 11 | 1011 | 22 | 01010 |
| 12 | 1100 | 32 | 00101 |
| 13 | 1101 | 13 | 10010 |
| 14 | 1110 | 23 | 01001 |
| 15 | 1111 | 14 | 10001 |

**Table 5.2** Mapping 4-bit data sequences to 5-bit codewords.

The proposed encoding scheme allows replacing the table of (data, codeword) pairs with $L_{cumulative}$ list and enables enumerative coding. The enumeration scheme includes the following steps for mapping data $D$ to codeword $C$.

1. Use $L_{cumulative}$ list to find the Hamming weight of codeword $C$ by Equation 5.3.

$$\sum_{i=0}^{k-1}\binom{n}{k} \leq D < \sum_{i=0}^{k}\binom{n}{k} \Leftrightarrow w_H(C) = k-1 \tag{5.3}$$

2. Use $L_{cumulative}$ list to find the number of codewords ($N$) with Hamming weights smaller than the Hamming weight of codeword $C$ denoted by $w_H(C)$.

$$N = L_{cumulative}(w_H(C) - 1) \tag{5.4}$$

3. Form the first codeword $C_0$ with Hamming weight $w_H$ in Low-Weight ordering.

$C_0$ = A string of $w_H$ ones followed by $n\text{-}w_H$ zeros with the general form: (1, 1 … 1, 0,

0 … 0)

4. Find the codeword that appears $cnt = D - N$ codewords after $C_0$ in Low-Weight

order.

For example, consider finding the 5-bit codeword corresponding to the 4-bit data $D =$

1010 which is 10 in decimal.

$L = \{1, \quad 5, \quad 10, \quad 10, \quad 5, \quad 1\}$

$L_{cumulative} = \{1, \quad 6, \quad 16, \quad 26, \quad 31, \quad 32\}$

In this example, $6 \le D = 10 < 16$, $w_H(D) = 1$ and $N = L_{cumulative}(0) = 1$, $C_0 = 10000$, and

$D - N = 9$. Thus, to find the corresponding codeword for $D = 1010$ we should find the

codeword $C$ that appears $N = 9$ codewords after $C_0 = 10000$ in Low-Weight ordering.

Thus, $C = 10100$, as Table 5.2 shows.

Similarly, at the decoder side, we can use enumeration to find the data $D$ for a received

codeword $C$, on the fly. The difference is that we have to use $C$ and compute its Low-

Weight rank which is the corresponding data. Recall that at the encoder we used the data

rank to compute $C$. The following steps complete the enumeration for at the decoder:

1. Count the number of ones of codeword $C$ to find its Hamming weight $w_H(C)$.

2. Use $L_{cumulative}$ list to find the number of codewords ($N_0$) with Hamming weights

smaller than $w_H(C)$.

$$N_0 = L_{cumulative}(w_H(C) - 1) \tag{5.5}$$

3. Form $C_0$ as before, at the encoder.

4. Codeword $C$ appears $cnt$ codewords after $C_0$. Use the Low-Weight Subtract to

find $cnt$.

5. Compute $D = cnt + N_0$.

## 5.5 Using help Codewords

We showed that the codewords of lower Hamming weights have less mean transmission cost per bit, compared to larger weight ones. On the other hand, for some initial bus states, a high weight transition codeword may result in smaller cost. For example, if the initial bus state $S_0 = 101010000$ then the transition codeword $c_1 = 111100000$ is more costly than $c_2 = 000011111$, while it has a lower Hamming weight.

Therefore, after selecting the low weight codewords, we assign a *help codeword* from the unselected higher weight codes to each selected low weight codeword. Before transmission on the bus, we compute the transmission cost for both the codeword and the help codeword. The codeword which results in smaller cost is placed and transmitted on the bus. This approach helps to gain more cost reduction, compared to the original differential low-weight coding scheme.

We add another step to this to reduce the cost more. Consider the phrase representation $p = p_1p_2...p_k$ of the extra codeword assigned to a given low weight codeword. *Phrase rotation* of $p$ to the right to obtain $p = p_k p_1p_2... p_{k-1}$ is helpful to reduce the cost, as well. For example, consider codeword $c = 010100101$ the complement is $c' = 101011010$ and $p$ $=12212$, which cannot help reduce the transmission cost of $c$ for any initial bus states. Phrase rotation of $p$ however, results in $p =21221$, or equivalently 011010110, which has a smaller transmission cost for initial bus states like *xccxxxcc*, where there are no restrictions on each $x$ and any run of consecutive bits depicted by $c$ must be all 1 or all 0.

## 5.6 Simulation Results

We employed the proposed encoding method to transmit data on 16-bit, 32-bit and 64-bit buses at the maximum possible rate, where only one extra line bus is added. Tables 5.3-5.5 compare the uncoded and the differential low-weight methods with the proposed method. As the tables show, in all cases the proposed method results in better cost (power and delay) reductions. We can achieve more power reduction by adding more bus lines. The tables show that the proposed method results in power reduction, for the most limited case (worst case), where only one extra bus line is allowed. Moreover, the proposed method requires $O(w_H)$ tables, where $w_H$ is the maximum Hamming weight of the selected codewords. Recall that the low-weight encoding scheme requires $O(2^n)$ tables for $n$-bit buses. Thus, there is a considerable reduction in size of tables and the system memory requirements.

| Method | $k$ | $n$ | $E$/codeword | $E$/bit | E Normalized |
|--------|-----|-----|--------------|---------|--------------|
| Uncoded | 16 | 16 | 88 | 5.5 | 1 |
| Dif. low-weight | 16 | 17 | 75.1181 | 4.6949 | 0.8536 |
| Proposed | 16 | 17 | 71.6958 | 4.4810 | 0.8147 |

**Table 5.3** Comparison of different transmission methods for 16-bit bus.

According to Table 5.3, the proposed encoding achieves 18.5% compared to uncoded and 1-71.6958/75.1181=4.6% improvement compared to the differential low-weight encoding. This table shows the simulation results for 1e6 randomly chosen input samples.

| Method | k | n | E/codeword | E/bit | E Normalized |
|--------|---|---|------------|-------|--------------|
| Uncoded | 32 | 32 | 176 | 5.5 | 1 |
| Dif. low-weight | 32 | 33 | 156.1017 | 4.8782 | 0.8869 |
| Proposed | 32 | 33 | 150.6102 | 4.7066 | 0.8557 |

**Table 5.4** Comparison of different transmission methods for 32-bit bus.

Table 5.4 shows that the proposed encoding achieves 14.5% compared to uncoded and (1-150.6102/156.1017)100 = 3.5% improvement compared to the differential low-weight method. This table shows the simulation results for 118e6 randomly chosen input samples.

| Method | k | n | E/codeword | E/bit | E Normalized |
|--------|---|---|------------|-------|--------------|
| Uncoded | 64 | 64 | 352 | 5.5 | 1 |
| Dif. Low-weight | 64 | 65 | 3.2195e2 | 5.0305 | 0.9146 |
| Proposed | 64 | 65 | 3.1373e2 | 4.9021 | 0.8913 |

**Table 5.5** Comparison of different transmission methods for 64-bit bus.

Table 5.5 shows that the proposed encoding reduces the transmission cost 10.87% totally and (1-3.2195e2/3.1373e2)100 = 2.6% more than the differential low-weight

method for a 64-bit DSM bus. This table shows the simulation results for 1e6 randomly chosen input samples.

We also compared two methods for finding the helping codewords: *complement and phrase rotate* and *complement* only. Table 5.6 shows the results for an 8-bit bus. As the table indicates the 'complement and phrase rotate' method results in better power reduction compared to 'complement only'.

| Extra Codeword Selection Method | Rate | E/ codeword | E/bit | Improvement vs. Dif. Low-weight |
|---|---|---|---|---|
| (Comp. & Phrase Rotate) | 8/9 | 34.0922 | 4.2615/4.4969 | 5.2% |
| Complement | 8/9 | 34.2915 | 4.2864/4.4969 | 4.6% |

**Table 5.6** Mapping 4-bit data sequences to 5-bit codewords.

Note that our simulations are for the highest possible rate were only one extra line is added. We can achieve more reduction by adding more lines. However, we cannot add too many lines as it decreases the rate (the number of useful transmitted bits per bus use). For optimal number of bus lines refer to Section 3.6.

# Chapter 6

## Conclusions

In the deep submicron technology (DSM), the coupling between lines (inter-wire capacitance) is much stronger than the coupling between individual lines and ground and the energy caused by parasitic capacitance is non-negligible [1-6]. Equations of the DSM buses indicate that the energy and delay depend on the sequence of data transmitted on the bus. Encoding is an efficient way for controlling the transmitted sequence on the bus and many encoding schemes have been used to reduce energy and delay on DSM buses, in the literature [7-15]. Note that to be effective, the encoder and decoder systems should have small energy and delay values.

The differential low-weight coding [1] is a simple differential scheme that achieves most of the possible energy reduction, from an information theory point of view. This efficient scheme employs differential coding and selects transition codewords of smaller Hamming weights for transmission on the bus. Thus, it is based on reducing the transition activity on the bus. However, this coding scheme requires tables of size $O(2^n)$ for an $n$-line bus, which is a significant overhead, especially when the number of bus lines is

large. Also, it does not take into account some characteristics of the DSM buses that can be exploited to reduce the transmission cost further.

We proposed an enumeration method that reduces the required table-size of the differential low-weight encoder from $O(2^n)$ words to $O(n)$ words, for an $n$-line bus and achieves considerable gain in terms of memory complexity. Furthermore, it facilitates less power dissipation and delay, as it eliminates the need for addressing and fetching data from very large tables in memory.

We also defined a cost function which could control both the energy and delay on DSM buses. We modified the energy and delay equations of DSM buses and developed a new representation in terms of number of the same and opposite direction transitions on the bus. We used these equations in our interconnect analysis to reduce the transmission cost further and to develop closed form formulas for computing the mean transmission cost per bit on DSM buses for both differential low-weight encoding and uncoded schemes.

We employed simple operations like complement and rotate, to compute *help codewords* (from the set of unselected codewords) on the fly, and assign them to the selected codewords. Note that the help codewords enable us to achieve more cost reduction, without increasing the memory complexity or number of the bus lines.

The simulation results for 16-bit, 32-bit and 64-bit buses at maximum rate (only one extra line added) show that the proposed encoding scheme achieves more than 10% cost reduction, and performs more than 2.5% better than to the original differential low-weight scheme, in the worst case.

# Bibliography

1. P. Sotiriadis, V. Tarokh, A. Chandrakasan, "Maximum achievable energy reduction using coding with applications to deep sub-micron buses," *IEEE International Symposium on Circuits and Systems*, Vol. 1, pp. I-85 - I-88, May 2002.

2. D. Sylvester, Wu Chenming, "Analytical modeling and characterization of deep-submicrometer interconnect," *Proceedings of the IEEE*, Vol. 89, No. 5, pp. 634 – 664, May 2001.

3. P.P. Sotiriadis, A. Chandrakasan, "A bus energy model for deep submicron technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 10, No. 3, pp. 341 – 350 , June 2002.

4. P.P. Sotiriadis, A. Chandrakasan, "Reducing bus delay in submicron technology using coding," *Proceedings of the Design Automation Conference,* pp. 109 – 114, Jan. 2001.

5. T. Sakurai, "Closed-form expressions for interconnection delay, coupling, and crosstalk in VLSI," *IEEE Transactions on Electron Devices*, Vol. 40, No. 1, pp. 118 – 124, Jan. 1993.

6. P.P. Sotiriadis, A. Chandrakasan, "Power Estimation and Power Optimal Communication in Deep Sub-Micron Buses: Analytical Models and Statistical Measures," *Journal of Circuits, Systems and Computers*, Vol. 11, No. 6, pp. 637-658, 2002.

7. P. Sotiriadis, V. Tarokh, A. Chandrakasan, "Energy reduction in VLSI computation modules: an information-theoretic approach," *IEEE Transactions on Information Theory*, Vol. 49, No. 4, pp. 790 – 808, April 2003.

8. M.R. Stan, W.P. Burleson, "Bus-invert coding for low-power I/O," *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 3 , No. 1, pp. 49 – 58, March 1995.

9. P.P. Sotiriadis, A. Chandrakasan, "Low power bus coding techniques considering inter-wire capacitances," *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 507 – 510, May 2000.

10. D. Rossi, V.E.S. van Dijk, R.P. Kleihorst, A.H. Nieuwland, C. Metra, "Coding scheme for low energy consumption fault-tolerant bus," *Proceedings of the Eighth IEEE International Workshop on On-Line Testing*, pp. 8 – 12, July 2002.

11. V. Sundararajan, K. K. Parhi, "Data transmission over a bus with peak-limited transition activity," *Proceedings of the ASP Design Automation Conference*, pp. 221 – 224, Jan. 2000.

12. S. Ramprasad, N. R. Shanbhag, I. N. Hajj, "A coding framework for low-power address and data busses," *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 7, No. 2, pp. 212 – 221, June 1999.

13. S. Ramprasad, N. R. Shanbhag, I. N. Hajj, "Signal coding for low power: fundamental limits and practical realizations," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 46, No. 7, 923 – 929, July 1999.

14. D. Marculescu, R. Marculescu, M. Pedram, "Information theoretic measures for power analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 15, No. 6, pp. 599 – 610 , June 1996.

15. N. R. Shanbhag , "A mathematical basis for power-reduction in digital VLSI systems," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 44, No. 11, pp. 935 – 951, Nov. 1997.

16. S. Datta, S. W. Mclaughlin, "Optimal Block Codes for M-ary Runlength-Constrained Channels," *IEEE Trans. on Information Theory*, Vol. 47, No. 5, July 2001.

17. T. M. Cover, "Enumerative source coding," *IEEE Trans. Inform. Theory*, Vol. IT-19, pp. 73–77, Jan. 1973.

18. M.A. Elgamel, M.A. Bayoumi, "Interconnect noise analysis and optimization in deep submicron technology," *IEEE Circuits and Systems Magazine*, Vol. 3, No. 4, pp. 6-17, 2003.

19. M.R. Stan, W. P. Burleson, "Low-power encodings for global communication in CMOS VLSI," *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 5, No. 4, pp. 444 – 455, Dec. 1997.

20. http://www.synopsys.com/products/tlr/flexroute_wp.pdf.

# Appendix

**List 1** List of the codewords with Hamming weight equal to 4 for $n = 9$, in Low-weight order.

| $X$ | $p$ | $c$ |
|---|---|---|
| 0 | 1111 | 111100000 |
| 1 | 2111 | 011110000 |
| 2 | 3111 | 001111000 |
| 3 | 4111 | 000111100 |
| 4 | 5111 | 000011110 |
| 5 | 6111 | 000001111 |
| 6 | 1211 | 101110000 |
| 7 | 2211 | 010111000 |
| 8 | 3211 | 001011100 |
| 9 | 4211 | 000101110 |
| 10 | 5211 | 000010111 |
| 11 | 1311 | 100111000 |
| 12 | 2311 | 010011100 |
| 13 | 3311 | 001001110 |
| 14 | 4311 | 000100111 |
| 15 | 1411 | 100011100 |
| 16 | 2411 | 010001110 |
| 17 | 3411 | 001000111 |
| 18 | 1511 | 100001110 |
| 19 | 2511 | 010000111 |
| 20 | 1611 | 100000111 |
| 21 | 1121 | 110110000 |
| 22 | 2121 | 011011000 |
| 23 | 3121 | 001101100 |
| 24 | 4121 | 000110110 |
| 25 | 5121 | 000011011 |
| 26 | 1221 | 101011000 |
| 27 | 2221 | 010101100 |
| 28 | 3221 | 001010110 |
| 29 | 4221 | 000101011 |
| 30 | 1321 | 100101100 |
| 31 | 2321 | 010010110 |
| 32 | 3321 | 001001011 |
| 33 | 1421 | 100010110 |
| 34 | 2421 | 010001011 |
| 35 | 1521 | 100001011 |
| 36 | 1131 | 110011000 |
| 37 | 2131 | 011001100 |
| 38 | 3131 | 001100110 |
| 39 | 4131 | 000110011 |

| | | |
|---|---|---|
| 40 | 1231 | 101001100 |
| 41 | 2231 | 010100110 |
| 42 | 3231 | 001010011 |
| 43 | 1331 | 100100110 |
| 44 | 2331 | 010010011 |
| 45 | 1431 | 100010011 |
| 46 | 1141 | 110001100 |
| 47 | 2141 | 011000110 |
| 48 | 3141 | 001100011 |
| 49 | 1241 | 101000110 |
| 50 | 2241 | 010100011 |
| 51 | 1341 | 100100011 |
| 52 | 1151 | 110000110 |
| 53 | 2151 | 011000011 |
| 54 | 1251 | 101000011 |
| 55 | 1161 | 110000011 |
| 56 | 1112 | 111010000 |
| 57 | 2112 | 011101000 |
| 58 | 3112 | 001110100 |
| 59 | 4112 | 000111010 |
| 60 | 5112 | 000011101 |
| 61 | 1212 | 101101000 |
| 62 | 2212 | 010110100 |
| 63 | 3212 | 001011010 |
| 64 | 4212 | 000101101 |
| 65 | 1312 | 100110100 |
| 66 | 2312 | 010011010 |
| 67 | 3312 | 001001101 |
| 68 | 1412 | 100011010 |
| 69 | 2412 | 010001101 |
| 70 | 1512 | 100001101 |
| 71 | 1122 | 110101000 |
| 72 | 2122 | 011010100 |
| 73 | 3122 | 001101010 |
| 74 | 4122 | 000110101 |
| 75 | 1222 | 101010100 |
| 76 | 2222 | 010101010 |
| 77 | 3222 | 001010101 |
| 78 | 1322 | 100101010 |
| 79 | 2322 | 010010101 |
| 80 | 1422 | 100010101 |
| 81 | 1132 | 110010100 |
| 82 | 2132 | 011001010 |
| 83 | 3132 | 001100101 |
| 84 | 1232 | 101001010 |
| 85 | 2232 | 010100101 |

| | | |
|---|---|---|
| 86 | 1332 | 100100101 |
| 87 | 1142 | 110001010 |
| 88 | 2142 | 011000101 |
| 89 | 1242 | 101000101 |
| 90 | 1152 | 110000101 |
| 91 | 1113 | 111001000 |
| 92 | 2113 | 011100100 |
| 93 | 3113 | 001110010 |
| 94 | 4113 | 000111001 |
| 95 | 1213 | 101100100 |
| 96 | 2213 | 010110010 |
| 97 | 3213 | 001011001 |
| 98 | 1313 | 100110010 |
| 99 | 2313 | 010011001 |
| 100 | 1413 | 100011001 |
| 101 | 1123 | 110100100 |
| 102 | 2123 | 011010010 |
| 103 | 3123 | 001101001 |
| 104 | 1223 | 101010010 |
| 105 | 2223 | 010101001 |
| 106 | 1323 | 100101001 |
| 107 | 1133 | 110010010 |
| 108 | 2133 | 011001001 |
| 109 | 1233 | 101001001 |
| 110 | 1143 | 110001001 |
| 111 | 1114 | 111000100 |
| 112 | 2114 | 011100010 |
| 113 | 3114 | 001110001 |
| 114 | 1214 | 101100010 |
| 115 | 2214 | 010110001 |
| 116 | 1314 | 100110001 |
| 117 | 1124 | 110100010 |
| 118 | 2124 | 011010001 |
| 119 | 1224 | 101010001 |
| 120 | 1134 | 110010001 |
| 121 | 1115 | 111000010 |
| 122 | 2115 | 011100001 |
| 123 | 1215 | 101100001 |
| 124 | 1125 | 110100001 |
| 125 | 1116 | 111000001 |