Coding a Terminated Bus for Low Power

Mircea R. Stan Wayne P. Burleson
ECE Dept., Univ. of Massachusetts, Amherst, MA 01003
email: mstan@risky.ecs.umass.edu
www homepage: http://www.ecs.umass.edu/ece/vspgroup/mstan.html

Abstract

Coding was proposed as a general method of decreasing power dissipation for the I/O [4, 5]. Lower power dissipation can be obtained by using extra bus lines for coding the data. This paper presents an application of the general theory of limited-weight codes *for a class of parallel terminated buses with pull-up terminators (e.g. Rambus). Power dissipation on such a bus-line is larger for a logical 1 and it follows that patterns with few* 1s *should be chosen. A perfect $\frac{k}{2}$-limited weight code equivalent to the previously proposed Bus-Invert method and a novel non-perfect 3-limited weight code are described. Both codes can be algorithmically generated and practical issues related to their implementation on the Rambus are discussed.*

1 Introduction

Coding buses for low-power and low-noise was independently proposed in [1, 4, 5, 6]. For unterminated buses the power dissipated is mainly dynamic (charging and discharging of bus line capacitances) and the codes minimize the bus activity (number of transitions). The main idea in [5] was to recognize that for each bus cycle an unterminated bus line can go through one of two different "states":

- a dissipative transition (LO to HI or HI to LO),
- a non-dissipative steady state (no transition).

Encoding transitions as 1*s* and non-transitions as 0s was the next logical step that enabled the formulation of *limited-weight codes* (LWC) [5] as a new family of codes useful for low-power. If the length (number of bits) of a codeword is n then by definition an m-LWC is a code whose codewords have weight $\leq m$. For efficiency it is desirable that (almost) all codewords with weight $\leq m$ be used as codewords (a code with fewer 1*s* translates into a code with fewer transitions and thus with lower power dissipation for an unterminated bus). If the number of possible patterns to be transmitted over the bus is 2^k by considering the information entropy of the source the following inequality must be satisfied for an m-LWC:

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{m} \geq 2^k$$

A *perfect* LWC satisfies the above with equality (uses *all* the patterns of length n with weight $\leq m$ as codewords). A *semi-perfect* LWC uses all the patterns with weight $< m$ and only some of the patterns with weight $= m$ as codewords. There is a clear trade-off between m and n: for low-power m should be as small as possible but that leads to an exponential increase of n which is generally unacceptable. Examples of a 2-LWC ($n = 5$) and a 1-LWC ($n = 15$) for $k = 4$ that reflect this trade-off are given in table 1. Perfect and semi-perfect limited-weight codes are optimal in the sense that any other code with the same length cannot have better statistical properties for low power. Except for special cases the algorithmic generation of arbitrary limited-weight codes is hard.

Table 1: Perfect 2-LWC and 1-LWC for $k = 4$.

uncoded $k = 4$	2-LWC $n = 5$	1-LWC $n = 15$
0000	00000	000000000000000
0001	00001	000000000000001
0010	00010	000000000000010
0011	00011	000000000000100
0100	00100	000000000001000
0101	00101	000000000010000
0110	00110	000000000100000
0111	11000	000000001000000
1000	01000	000000010000000
1001	01001	000000100000000
1010	01010	000001000000000
1011	10100	000010000000000
1100	01100	000100000000000
1101	10010	001000000000000
1110	10001	010000000000000
1111	10000	100000000000000

In this paper we'll show how the theory of limited-weight codes can be applied in the case of terminated buses. We'll focus on a special class of parallel terminated buses but similar techniques can be used whenever one of the two different bus states is more dissipative than the other. Where appropriate we'll use *Rambus* [1] as a special case study for our techniques. In section 1.1 we look at different ways of computing power dissipation on a terminated bus. Section 2 looks at limited-weight coding for decreasing the power on a terminated bus. Since an algorithmic approach for

[1]Rambus is a trademark of Rambus Inc.

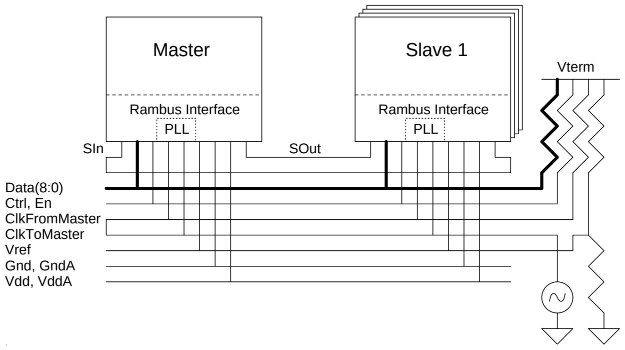

Figure 1: A Rambus system. The bus lines are terminated with pull-up resistors equal to the line's characteristic impedance.

code generation is desired we present two such methods: in section 2.1 an equivalent of the Bus-Invert method [4] and in section 2.2 a 3-LWC which is convenient to implement even if the resulting code is non-perfect.

1.1 Power dissipation on a parallel terminated bus

The parallel terminated buses considered in this paper use active-low level signaling with pull-up resistors: when a 1 is transmitted the bus driver must pull the line low, for a zero the driver shuts and the pull-up resistor takes over. The Rambus is a byte-wide high performance synchronous bus standard that uses such termination (see fig. 1). There are two different general formulas that compute the power dissipated on such a bus:

$$P = V_{term}^2/R_{term} \tag{1}$$

$$P = CV_{term}^2 f \tag{2}$$

Although they look rather different the above formulas can be interchangeably used. To show this we need a short review of basic transmission line terminology [2]. A transmission line possesses a uniformly distributed series inductance L_0 and a uniformly distributed shunt capacitance C_0 (we consider a lossless line with no series resistance R_0 or shunt conductance G_0). A wave will travel on such a line with speed $v_0 = 1/\sqrt{L_0 C_0}$. The characteristic impedance will be $Z_0 = \sqrt{\frac{L_0}{C_0}}$ and in order to avoid reflections the bus must be terminated with its characteristic impedance $R_{term} = Z_0$.

In order to understand how power is dissipated on such a bus we consider three cases:

1. an extreme case when the length of the line is 0 which means the terminators are coupled directly to the drivers (formula 1),
2. the other extreme case of a line of infinite length (formula 2),
3. a line of length l terminated at the end by a matching resistor.

It should be noted that both case 1 and 2 are the normal recommended abstractions for terminated lines and we merely want to see here that different interpretations will give the same results.

Consider the energy dissipated in one period T:

- in the case of the line with zero length the energy dissipated will be: $P_1 = TV^2/R_{term} = TV^2\sqrt{\frac{C_0}{L_0}}$.
- in the case of the line of infinite length the capacitance charged in time T will be $C = Tv_0C_0 = TC_0/\sqrt{L_0C_0} = T\sqrt{\frac{C_0}{L_0}}$ and so the energy dissipated in one period will be $P_2 = V^2C = V^2T\sqrt{\frac{C_0}{L_0}} = P_1$.
- in the case of a line of finite length l we can consider that power is first dissipated for charging a capacitance $C = lC_0$ followed by dissipation in the terminator for a time $T - l/v_0 = T - l\sqrt{L_0C_0}$. The energy for one period is then $P_3 = CV^2 + (T - l\sqrt{L_0C_0})V^2/Z_0$. After simple substitutions it follows easily that $P_3 = P_2 = P_1$.

The above show that the simple method of computing the power dissipated on the bus as if the drivers were directly connected to the terminating resistors works fine.

For a bus line with only a pull-up resistor power will be dissipated only when the line is pulled low. For the Rambus for example the voltage-swing on the bus is $0.6V$. Rambus lines are terminated with matched impedance resistors connected to $V_{term} \approx 2.5V$. The power dissipation for a logical 0 is practically nil (output current $I_{OH} \leq 10\mu A$ [3]) and the voltage on the line is $V_{OH} = V_{term}$. A logical 1 is obtained by pulling the line low to $V_{OL} \approx 1.9V$. The maximum current in that case is $I_{OL} \approx 35mA$ [3] and the power dissipated on a line at logical 1 is $P_{OL} = (V_{term} - V_{OL}) \cdot I_{OL} \approx 21mW$. For the worst case (all $1s$ pattern) the power dissipated on the byte-wide Rambus is $P_{worst} = 8 \cdot P_{OL} \approx 168mW$. Although this value may seem small compared to the power dissipated by other system components it must be remembered that the power dissipated by an IC can be theoretically made arbitrarily small (by technology, circuit and algorithmic advances) but the power dissipated over a bus line remains constant as long as the bus electrical specification is not changed and consequently can become dominant. It is desirable then to be able to decrease the bus power dissipation for example by coding techniques.

2 Limited-weight coding for a parallel terminated bus

Since a parallel terminated bus with only pull-up resistors dissipates power only for $1s$, a LWC which has codewords with small weight will help decrease power dissipation. Although it is theoretically possible to generate such a code with a look-up table (that will store table 1 for example) this is generally unacceptable for low-power applications because the look-up table will likely dissipate more power than the savings due to the LWC. For this reason an *algorithmic* generation of limited-weight codes is desired. This is an unsolved problem in the general case but in the following sections two particular methods with different performance are presented.

2.1 Equivalent Bus-Invert coding for a parallel terminated bus

For *unterminated* lines the Bus-Invert method [4] looks at two consecutive bus transfers and if the Hamming distance between the two successive values is $> k/2$ than the next value is complemented and this is signaled on an extra bus line called *invert*. In this way the maximum number of transitions (maximum Hamming distance between two consecutive values) is decreased from k to $k/2$. The method is efficient, uses only one extra line ($n = k+1$), and is relatively easy to implement. By analyzing the different possible Hamming distances it can be seen that they can be viewed as the codewords of a perfect $k/2$-limited-weight code.

In order to adapt the Bus-Invert method for our class of terminated buses the analysis is simpler because now the 1 values on the bus are directly dissipative. The purpose of a low-power coding then is to have few 1*s* which is exactly what limited-weight codes provide. The steps of an equivalent of Bus-Invert method for this case will then be:

1. compute the uncoded word's weight,
2. *IF* weigt$>k$ complement the bits *AND invert*$= 1$,
3. *ELSE* let the bits un-inverted *AND invert*$= 0$.

At the receiving end the data must be recovered by conditionally complementing the data according to the *invert* line. For $k = 4$ and $n = k + 1 = 5$ the resulting code will be exactly the one in the 2^{nd} column of table 1.

For the Rambus our purpose is to code the data for low power at the *logical* level without affecting the *physical* specification. The Rambus has $8 + 1$ data bits, the 9^{th} bit's use being left to the system designer. We'll take advantage of this 9^{th} bit and use it as the invert line. The derived code will have codewords of length 9. With 9 bits there are $2^9 = 512$ possible patterns out of which $2^8 = 256$ are needed in order to transmit the same amount of information as in the uncoded case. It can be observed that:

$$\binom{9}{0} + \binom{9}{1} + \binom{9}{2} + \binom{9}{3} + \binom{9}{4} = 2^8 = 256$$

It follows that a perfect *4-limited weight code* [5] that uses *all* 9-bit patterns with at most four 1*s* is optimal. The data can be decoded at the receiver and stored uncoded as 8-bit values. Another option is to store the data in coded form in a 9-bit wide Rambus DRAM (RDRAM). This would have the great advantage of using only off-the-shelf RDRAMs, with modifications needed only on the Rambus ASIC (RASIC) side. Because the resulting codewords have at most four 1*s* the worst case power dissipated on the data lines is

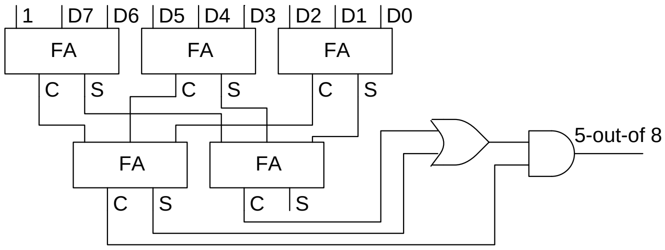

Figure 2: 5-out-of-8 majority voter.

decreased by 50% (from $168mW$ to $84mW$). The decrease in average power dissipation depends on the statistics of the bus transfers and is generally smaller [4, 5]. If the data is random uniformly distributed the average power dissipation is only reduced by approx. 18% (from $84mW$ to $68mW$). The circuit for computing the weight is a *majority voter* which can be easily implemented as a tree of full-adders (fig. 2) or as an analog circuit [4] included in the RASIC at the master. The circuit has no feedback and can be easily pipelined for achieving the high throughput of the Rambus.

2.2 Algorithmic generation of a novel limited-weight code

Sometimes it is desirable to obtain a code with smaller maximum weight at the expense of a larger number of extra lines. For small k it is possible to use a 1-LWC code which is essentially a 1-hot encoding of the $2^k - 1$ nonzero data words (see 3^{rd} column of table 1). The 1-LWC is a perfect code like the Bus-Invert code and these two families of limited-weight codes represent two extremes:

- the Bus-Invert has the minimum *redundancy* for a limited-weight code (one extra line),
- the 1-limited-weight code achieves the minimum *weight* (one for nonzero codewords).

Unfortunately between these two extremes algorithmic methods for generating other perfect or semi-perfect codes are lacking. In what follows we'll show a novel algorithmic method of generating a 3-limited-weight code with weight distribution very close to a semi-perfect code of the same length.

The steps in obtaining the novel 3-limited-weight code are:

1. divide the k bits of the uncoded word into two halves of length $k/2$. We'll call them the UP and LOW halves.
2. encode the two halves as 1-limited-weight codes of length $n = 2^{k/2} - 1$.
3. do a bitwise OR of the two encoded values. The result will be a word of length $n = 2^{k/2} - 1$ with weight ≤ 2. In order to be able to uniquely decode this ORed word two extra bits are needed:

Table 2: Numerical examples of different cases for the novel 3-limited-weight code

uncoded word	UP-> bit	zero bit	coded UP half	coded LOW half	coded word
0011 0011	0	0	000000000000100	000000000000100	0 0 000000000000100
0111 0011	1	0	000000001000000	000000000000100	1 0 000000001000100
0011 0111	0	0	000000000000100	000000001000000	0 0 000000001000100
0000 0011	0	1	000000000000000	000000000000100	0 1 000000000000100
0011 0000	1	1	000000000000100	000000000000000	1 1 000000000000100
0000 0000	0	0	000000000000000	000000000000000	0 0 000000000000000

- a *zero* bit that signals when exactly one of the two halves is zero,
- an *UP-greater* bit that signals when the UP half is greater than the LOW half (this choice is arbitrary, a *LOW-greater* bit could be also used).

In order to exemplify this we'll look again at the Rambus for which the length of the uncoded words is $k = 8$. The steps of the code generation for $k = 8$ are then:

1. divide the 8 bits into two 4-bit halves.
2. encode the two halves into 1-LW codewords of length 15 (as in the 3^{rd} column of table 1).
3. the two 15-bit words are to be ORed and appended with the *zero* bit that signals if one of the two halves is zero and the *UP-greater* bit which signals if the UP half is greater than the LOW.

For some numerical examples of this procedure see table 2. From the table and the code generation algorithm it can be seen that for $k = 8$ a *3-limited weight* code of length $n = 17$ was generated. In order to see how efficient this code is we'll compare it with an optimal *semi-perfect* 3-limited-weight code of length $n = 17$. The semi-perfect code will have: $\binom{17}{0} = 1$ codewords with weight 0, $\binom{17}{1} = 17$ codewords with weight 1, $\binom{17}{2} = 136$ codewords with weight 2, $256 - 1 - 17 - 136 = 102$ remaining codewords with weight 3. The novel 3-limited-weight code proposed here has: 1 codewords with weight 0 (the zero word), $\binom{15}{1} = 15$ codewords with weight 1, $\binom{15}{2} + 15 = 120$ codewords with weight 2, $\binom{15}{2} + 15 = 120$ codewords with weight 3.

As can be seen the statistics of the new code are not much worse as those of the semi-perfect one which coupled with it's algorithmic generation make it suitable for low-power I/O. For random uniformly distributed data the semi-perfect 3-LWC will have an average of $(17+136\cdot 2+102\cdot 3)/256 = 2.32$ $1s$ per transfer (42% savings vs. the uncoded bus) while our non-perfect 3-LWC will have $(15 + 120 \cdot 2 + 120 \cdot 3)/256 = 2.4$ $1s$ per transfer (40% savings vs. the uncoded bus). For using this code with the Rambus we need to use two successive transfers over the 9-bit wide bus. Of course this will decrease the maximum bandwidth by half but for low-power applications this might be acceptable since the Rambus maximum bandwidth is a large 500 MB/s.

3 Conclusions and future work

We presented a coding methodology for decreasing the power dissipation on a class of parallel terminated buses. Future efforts will concentrate on the algorithmic generation of limited-weight codes based on a relationship with the well understood field of error-correction codes.

For the Rambus the most important aspect is that the coding can be done within the constraints of the Rambus specification. Bus-Invert coding the data uses the 9^{th} extra bus line in order to get a code with patterns with weight ≤ 4. The coding and decoding can be done at the master only and then standard 9-bit wide RDRAMs can be used. The RASIC on the master can be easily modified to support a pipelined transparent coding and decoding. All these modifications can be done in a downward compatible manner. In the future we'd like to see these ideas used in a real Rambus system.

References

[1] R. J. Fletcher, "Integrated Circuit Having Outputs Configured for Reduced State Changes", U.S. Patent no. 4,667,337, May, 1987.

[2] Sol Rosenstark, *Transmission Lines in Computer Engineering*, McGraw-Hill, 1994.

[3] *Rambus documentation*, Rambus Inc., Mountain View, CA, 1993. Contact `ray@rambus.com`.

[4] M. R. Stan, W. P. Burleson, "Bus-Invert coding for low power I/O", to appear in *IEEE Transactions on VLSI*, March 1995.

[5] M. R. Stan, W. P. Burleson, "Limited-weight codes for low-power I/O", Int. Workshop on Low-Power Design, Napa, CA, April 1994.

[6] J. Tabor, *Noise Reduction Using Low Weight and Constant Weight Coding Techniques*, Master's Thesis, EECS Dept., MIT, May 1990.