Technical Report 1232

Noise Reduction Using Low Weight and Constant Weight Coding Techniques

Jeff F. Tabor

MIT Artificial Intelligence Laboratory

This blank page was inserted to preserve pagination.

Noise Reduction Using Low Weight and Constant Weight Coding Techniques

by

Jeffrey F. Tabor

Submitted to the
Department of Electrical Engineering and Computer Science

May 11, 1990

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Electrical Engineering

Abstract

Signalling off-chip requires significant current. As a result, a chip's power-supply current changes drastically during certain output-bus transitions. These current fluctuations cause a voltage drop between the chip and circuit board due to the parasitic inductance of the power-supply package leads. Digital designers often go to great lengths to reduce this *transmitted noise*. Cray, for instance, carefully balances output signals using a technique called differential signalling to guarantee a chip has constant output current. Transmitted-noise reduction costs Cray a factor of two in output pins and wires. Coding achieves similar results at smaller costs.

In a circuit using parallel-terminated transmission lines such as the Cray-1, a chip exhibits nearly constant output current throughout a single clock cycle. However, since current for an output 1 differs greatly from that for an output 0, output current can change drastically at the end of a cycle generating an intolerable amount of transmitted noise. Hence, a coding scheme which maintains a constant number of output 1's will reduce transmitted noise as successfully as differential signalling. Experimental results substantiate this.

Information capacity of a word is greatest when the word has half 1's. Coding requires fewer output signals than differential signalling. For example, an 8-bit byte can be encoded as eleven bits with four 1's at an output savings of 30%. Of course, fewer outputs do not come for free; coding costs time and chip area. Coding achieves a smaller percentage output expansion for words longer than eight bits but requires a more costly coding scheme.

In a circuit using series-terminated or un-terminated transmission lines, output current flows when a signal changes state to charge/discharge the capacitive load. Hence, reducing transmitted noise requires minimizing output transitions. Representing transitions as 1's, a coding strategy which minimizes 1's will reduce transmitted noise. Unfortunately, words with few 1's have little information capacity. Coding in this case requires an exponential increase in output signals. Hence, only a $2\times$ to $3\times$ noise reduction is practical.

Thesis Supervisor: Thomas F. Knight, Jr.
Title: Assistant Professor, MIT AI Lab

This empty page was substituted for a blank page in the original document.

Acknowledgments

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-88-K-0825 and in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-85-K0124.

I have learned more from Tom Knight than anyone else at MIT. Tom's contribution to this thesis was irreplaceable (most of the ideas were his) but represents only a small fraction of how he has helped me during the past few years. I have learned much about electronics while working with Tom on various projects and in several courses, but the most important lesson I have learned has been through his example. Tom's clear and simple expression of his thorough understanding of many fields makes him the best teacher I have ever had. He has taught me, in the words of M.A. Biot, "the tradition of clarity, simplicity, intuitive understanding, unpretentious depth, and a shunning of the irrelevant." Thanks, Tom, and good luck in the future.

Many students and members of the faculty and staff have contributed to this thesis. I would like to thank the following people for invaluable discussions and other help: Gill Pratt, Peter Elias, Pierre Humblet, Ron Rivest, Priscilla Cobb, Fred Drenckhahn, Curt Fennell, Ricardo Jenez, Jonathan Meyer, Jeanne Speckman, Ron Wiken, Fred Chong, Mike Noakes, and John Pezaris.

Finally, to my family and friends, thanks for your love and support.

This empty page was substituted for a blank page in the original document.

Contents

1 Introduction 1

1.1 Transmitted Noise in Digital Circuits 1

1.2 Coding to Reduce Transmitted Noise 3

1.3 Implications of Coding 4

2 Explanation of Terminology 7

2.1 Segregating Digital Systems for Coding 7

2.2 A Bit of Coding Jargon 10

3 Reducing Transmitted Noise Using Ration Coding 14

3.1 An Example of Transmitted Noise in an ECL Circuit 14

3.2 Ration Coding as a Solution 17

3.3 Theoretic Limits of Ration Coding 19

3.4 Implementations of Ration Coding 23

3.4.1 Table Look-Up 23

3.4.2 Algorithmic Implementations 26

3.5 Ration Coding: Noise Reduction for All Ages 32

3.6 Demonstration of Ration Coding Reducing Noise 36

4 Reducing Transmitted Noise Using Starvation Coding 42

4.1 An Example of Transmitted Noise in a CMOS Circuit 42

4.2 Starvation Coding as a Solution 45

4.3 Theoretic Limits of Starvation Coding 47

4.4 Implementations of Starvation Coding 50
4.4.1 Algorithmic Implementations 50
4.4.2 Table Look-Up 53

5 Conclusions and Future Investigations 56

A More Demonstrations of Ration Coding Reducing Noise 60

List of Figures

1-1 For circuits signalling with parallel-terminated transmission lines, a coding strategy which maintains a constant number of output 1's reduces transmitted noise. 3

1-2 For circuits signalling with series-terminated of un-terminated transmission lines, a coding strategy which minimizes 1's reduces transmitted noise. 4

1-3 The information capacity of 32-bit words plotted as a function of the number of 1's in the word. 5

2-1 A model for signalling with a parallel-terminated transmission line. 8

2-2 A model for signalling with an un-terminated transmission line. 9

2-3 A model for signalling with a series-terminated transmission line. 9

2-4 Coding and information theory's concept of weight leads to three coding terms. 11

2-5 Starvation coding reduces a word's weight. 11

2-6 Indulgence coding increases a word's weight. 12

2-7 Ration coding keeps a word's weight constant. 12

3-1 40-bit parallel output circuit of the B3011 driving parallel-terminated transmission lines. 15

3-2 Transistor model used in emitter-follower analysis. 17

3-3 Emitter-follower transfer curve with noise and without noise. 18

3-4 The number of different, constant-weight, 35-bit words plotted as a function of weight, $C(35,W)$. 20

3-5 The information capacity of a constant-weight, 35-bit word plotted as a function of weight, $log_2 C(35,W)$. 22

3-6 Description of parameters for ration coding using table look-up. 24
3-7 Ration coding implemented with a recursive algorithm. 27
3-8 Decoding a bit which was encoded using the algorithm with three steps of recursion. 30
3-9 Fast, analog, count-and-compare circuit for implementing recursive algorithm . 31
3-10 Differential signalling exhibiting common mode rejection. 34
3-11 A ration-coding implementation exhibiting common mode rejection. 35
3-12 A block diagram of the ration-coding demonstration board. 37
3-13 The worst-case transition experiment requires PCB1 to toggle between three different pairs of output words. 39
3-14 Noise generated during a transition from a word with weight zero to a word with weight thirty-two. 39
3-15 Noise generated after pseudo-implementation of ration-coding algorithm with one step of recursion. 40
3-16 Noise generated when implementing differential signalling. 41

4-1 96-bit parallel output circuit of the i860 driving unterminated transmission lines. 43
4-2 A circuit for implementing transition signalling. 46
4-3 The number of different 23-bit words plotted as a function of weight, $C(23, W)$. 49
4-4 A plot of $C(23, W)$ expanded to show detail near $W = 2$. 49
4-5 An algorithmic implementation of starvation coding which reduces transmitted noise by 50%. 51

A-1 Noise generated by a transition from an output word with weight seven to a word with weight twenty-five. 61
A-2 A pseudo-implementation of the ration-coding algorithm with one step of recursion reducing the noise from Figure A-1. 61
A-3 Differential signalling reducing the noise from Figure A-1. 62
A-4 Worst-case noise generation with one step of recursion. 62
A-5 A pseudo-implementation of ration-coding with two steps of recursion reducing the worst-case noise from Figure A-4. 63

A-6 Worst-case noise generation with two steps of recursion. 63
A-7 Worst-case noise generation with three steps of recursion. 64
A-8 Noise generated by a transition from an output word with weight zero to a word with weight thirty-two. 64
A-9 Table look-up implementation reducing worst-case noise from Figure A-8. Here, 8-bit words are encoded as eleven bits with weight four. 65
A-10 Table look-up implementation reducing worst-case noise from Figure A-8. Here, 4-bit words are encoded as six bits with weight three. 65
A-11 Table look-up implementation reducing worst-case noise from Figure A-8. Here, 2-bit words are encoded as four bits with weight one. 66
A-12 Table look-up implementation reducing worst-case noise from Figure A-8. Here, 6-bit words are encoded as eight bits with weight four. 66
A-13 Table look-up implementation reducing worst-case noise from Figure A-8. Here, 3-bit words are encoded as five bits with weight two. 67
A-14 Differential signalling reducing the worst-case noise from Figure A-8. 67

This empty page was substituted for a blank page in the original document.

List of Tables

3.1 Presented are the trade-offs between noise generation, output pins, and table area for several implementations of ration coding using look-up tables. 25
3.2 Tests required by recursive algorithm. 29
3.3 Trade-offs to consider when using recursive algorithm to implement ration coding. 29

4.1 Output signals required for various amounts of information when reducing transmitted noise by a factor of four. 50
4.2 The transitions saved on the average using the starvation-coding algorithm with various segment sizes. 53
4.3 Parameter combinations with their influence on various design characteristics. . 54

This empty page was substituted for a blank page in the original document.

Chapter 1

Introduction

Noise hampers the performance of digital circuits by corrupting otherwise valid logic signals. For the purposes of this document, two types of noise exist: internal (or self) noise and transmitted noise. As the names suggest, the former refers to noise confined to a single integrated circuit (or chip) whereas the latter refers to noise in the communication between chips [Dil88]. This document discusses techniques for reducing a common form of transmitted noise.[1]

1.1 Transmitted Noise in Digital Circuits

In digital circuits chips communicate with signals at one of two voltage levels. A chip's power supplies act as sources of transmitted signals and references for received signals. Chip-to-chip communication can be successful only if each chip operates with the same source and reference voltages. Transmitted noise can cause power-supply voltages from chip to chip to disagree and, hence, contribute to miscommunication.

Due to $V = L(dI/dt)$ and the parasitic inductance of package leads, a voltage drop occurs between chip and circuit board during a power-supply current fluctuation. Since all chips in a circuit will not suffer from the same voltage drop, this is a form of transmitted noise. My thesis attempts to alleviate this problem.

[1] Since the cause of the transmitted noise can be identified and removed, theorists may dispute the use of the word *noise* in this case. However, this is noise from a digital designer's perspective and is referred to as such in electronics literature.

Since signalling off-chip requires significant current, a chip's output drivers can generate a prohibitive amount of transmitted noise. For this reason ECL chip manufacturers provide a separate ground pin, often labeled V_{CCO}, for the output drivers as an attempt to isolate them from the internal logic circuits. "When switching all the outputs on a V_{CCO} pin except one, a noise pulse of about 2ns wide is coupled to the non-switched outputs due to cross-talk and lead inductance." [PB88] Digital designers often go to great lengths to reduce transmitted noise. Cray, for instance, carefully balances output signals using a technique called differential signalling to guarantee a chip has constant output current. Transmitted noise reduction costs Cray a factor of two in output pins and wires [Kol81]. Coding achieves similar results at smaller costs.

Digital designers employ many techniques to reduce transmitted noise. These techniques fall in two categories based on $V = L(dI/dt)$: reducing L and reducing dI/dt. The parasitic inductance of package leads is approximately equal to $1.0nH/mm$. (This value can be derived from basic principles.) Although this seems small, since current through power and ground pins on parts can reach ampere levels, the package lead inductance is a significant noise source [TK85]. The magnitude of L can be reduced in two ways. ECL chip designers shorten power-supply package leads by moving them from their traditional TTL locations (top, right and lower, left) to the center of the package [Blo88]. Adding power-supply pins also reduces L since adding inductors in parallel reduces total inductance. Most large chips employ this technique. The intel i860 microprocessor, for instance, has twenty-four ground pins and twenty-four power pins [Int89].

Digital designers achieve a reduction in dI/dt in several ways. Intel uses weak output drivers in its i860 to slow output-signal edges and, hence, reduce dI/dt [Int89]. The Motorola ECL gate array guidelines suggest using the *output edge rate slow down option* to reduce the noise generated by the output drivers. "This option adds 500ps of delay to the output because it slows the output signal edge rates; however, it also slows the rate of change of current pulled through the V_{CCO} pins, which cuts the maximum noise amplitude by 50mV." [PB88] The staggering of output signals and the slowing of clock edges also reduces dI/dt.

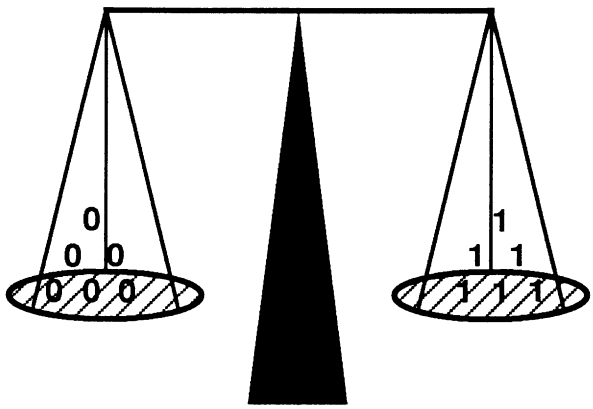

Figure 1-1: For circuits signalling with parallel-terminated transmission lines, a coding strategy which maintains a constant number of output 1's reduces transmitted noise.

1.2 Coding to Reduce Transmitted Noise

This document presents techniques for reducing power-supply lead dI/dt through the coding of output signals. For the purposes of this document, two types of digital circuits exist: circuit where chips signal with parallel-terminated transmission lines and circuits where chips signal with either series-terminated or un-terminated transmission lines. A different noise-reduction coding strategy will be developed for each case.

In a circuit using parallel-terminated transmission lines such as the Cray-1 [Kol81], a chip exhibits nearly constant output current throughout a single clock cycle. However, since current for an output 1 differs greatly from that for an output 0, output current can change drastically at the end of a cycle generating an intolerable amount of transmitted noise. Hence, a coding strategy which maintains a constant number of output 1's will reduce transmitted noise as successfully as differential signalling. My experimental results substantiate this.

In a circuit using series-terminated or un-terminated transmission lines, output current flows when a signal changes state to charge/discharge the capacitive load. Hence, reducing transmitted noise requires minimizing output transitions. Representing transitions as 1's, a coding strategy which minimizes 1's will reduce transmitted noise.

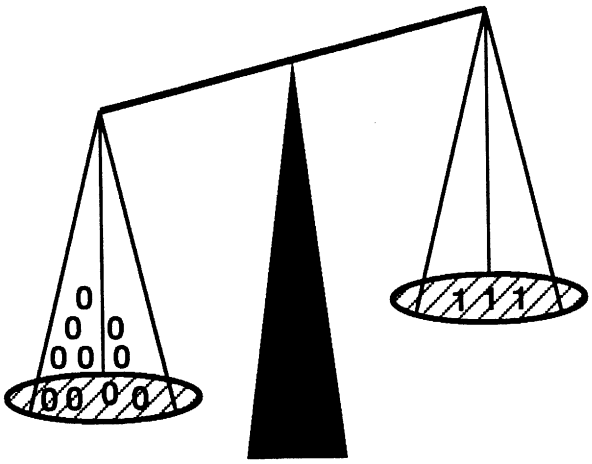

Figure 1-2: For circuits signalling with series-terminated of un-terminated transmission lines, a coding strategy which minimizes 1's reduces transmitted noise.

1.3 Implications of Coding

Coding to reduce transmitted noise does not come without costs. Coding requires design time, execution time, chip area, and output pins. This document discusses general coding approaches but does not recommend an all-purpose coding strategy. Therefore, a considerable engineering effort will be required to customize the general ideas presented here to a specific application. The other costs can be discussed more concretely. Time is required to encode the output signals prior to transmission and decode the input signals after reception. In a pipeline environment, however, the cost of an additional stage or two should not be too severe. Encoding and decoding circuitry occupy chip area. This cost will be examined in depth, later, when discussing specific coding schemes.

Coding requires additional output pins. As an example, consider a microprocessor chip with an N-bit output bus. The chip suffers from transmitted noise, and we have decided to use coding to reduce the generation of noise. Coding restricts the chip's output signals. Information theory dictates that, because of this restriction, more than N signals are required to transmit N bits of information. The reasoning is as follows.

Define p as the probability of an output signal being a 1. On the average, the amount of

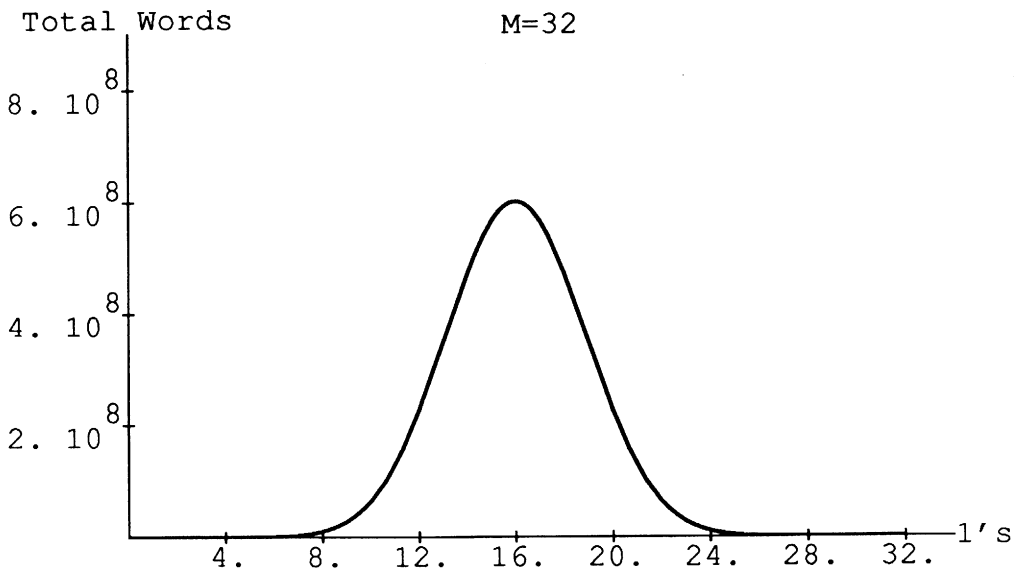

Figure 1-3: The information capacity of 32-bit words plotted as a function of the number of 1's in the word.

information (in bits) transmitted by each signal is $H(p)$, where $H(p) = -p * log_2(p) - (1-p) * log_2(1-p)$. The amount of information transmitted by the entire word is therefore $N * H(p)$. Prior to coding, we have output signals which are equally likely to be 1 or 0, so $p = 1/2$. $H(p = 1/2) = 1$, so one bit of information is transmitted per signal and N bits for the entire word. Coding places a restriction on usable output words, so p is no longer equal to $1/2$. This forces $H(p)$, the average amount of information per signal, to be less than one bit. Therefore, more than N signals are required to transmit N bits of information [Hum85].

What does all this mean for the two types of coding discussed above? The entropy (or information capacity) plot for a 32-bit word shown in Figure 1-3 will help explain. The plot shows good news for the parallel-terminated case and bad news for the series-terminated/unterminated case. The information capacity of a word is greatest where the word has half 1's. Therefore, for the parallel-terminated case, to maintain a constant number of output 1's, choose the constant to be approximately half the word length. Since so many words of this type exist, few extra output signals are necessary. For example, an 8-bit byte can be encoded

as eleven bits with four 1's. This achieves the noise reduction of differential signalling at an output savings of more than 30%. Coding achieves a smaller percentage output expansion for words longer than eight bits. For example, a 32-bit word can be encoded as thirty-five bits with seventeen 1's. Unfortunately, longer words require a more costly coding scheme.

For the series-terminated/un-terminated case minimizing 1's cannot be as successful since such a small number of words with few 1's exists. Here, coding requires an exponential increase in output signals. Hence, only a $2\times$ to $3\times$ noise reduction is practical.

No noise-reduction scheme is without its sacrifices. (The only alternative to reducing noise is to slow a circuit's clock and live with it. Of course, this represents a substantial performance loss.) The techniques discussed for reducing L restrict package design and increase the pinout of a chip. The techniques discussed for reducing dI/dt hurt a chip's performance by by reducing its output bandwidth. Coding requires time, chip area, and additional output pins but allows a designer to trade-off these characteristics to suit an application.

The next chapter includes preliminary details not appropriate for an introductory chapter. It explains in more detail some of the ideas already presented as well as introduces several noise-reduction coding terms coined to make the remainder of the document more readable.

Chapter 2

Explanation of Terminology

This document discusses the development of totally new coding techniques. Scientific literature contains little investigation of low weight and constant weight coding and certainly nothing directly applicable to this problem. For this reason coding terms have been coined to ease the reading of this document.

Coding's primary goal here is to reduce current fluctuations on a chip's power-supply pins. As mentioned previously the cause of these fluctuations fall in one of two categories, depending on a circuit's chip-to-chip signalling technology. The next section of this document examines this distinction. Later, separate noise-reduction coding schemes will be developed for each category.

2.1 Segregating Digital Systems for Coding

The coding strategy depends on the interchip communication technology of a digital circuit. Three communication technologies will be discussed in this document: parallel-terminated, series-terminated, and un-terminated transmission lines. (*Transmission line* refers to any type of wire connecting two chips, including coaxial cable, PC-board trace, and wire-wrap wire.) Figures 2-3 through 2-5 depict the three technologies schematically. Each contains a transmitting-chip output driver, a transmission line, and a capacitive load. C_{LOAD} represents the capacitance of the interconnect as well as that of the receiving-chip input. The output driver and capacitive load are identical in all three cases.

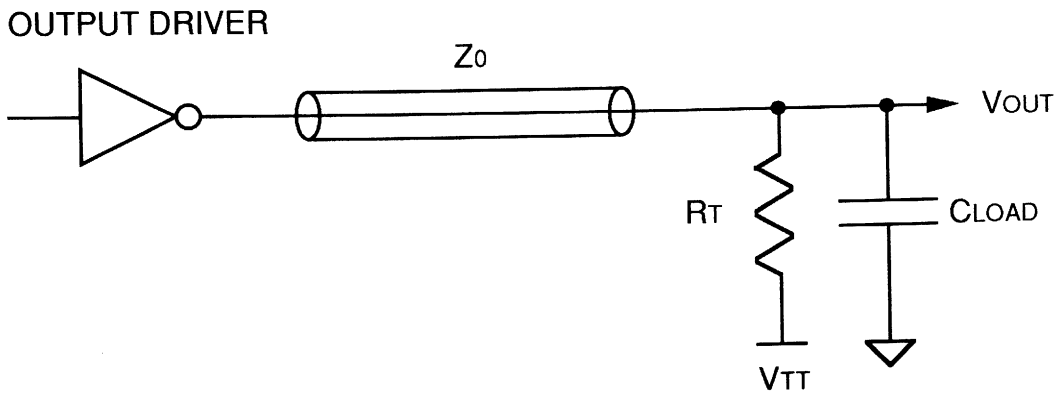

Figure 2-1: A model for signalling with a parallel-terminated transmission line.

This document categorizes circuits according to interchip, steady-state current flow. Circuits using parallel-terminated transmission lines form one group- those that exhibit steady-state current. When signalling over parallel-terminated lines, a path of low resistance exists in the steady state between a transmitting-chip power supply and that of the terminating voltage. Referring to Figure 2-3, when V_{OUT} changes logic value, current flows to/from the output driver to charge/discharge C_{LOAD}. Even after C_{LOAD} is fully charged/discharged, current continues to flow because of R_T. The terminating voltage (V_{TT}) is not equal to either logic-value voltage. Hence, steady-state current flows between the output driver and V_{TT}.

Circuits using series-terminated or un-terminated transmission lines form the second group- those that exhibit negligible steady-state current. Minimal steady-state current flows in such circuits because of the high input resistance of receiving gates. A large current flows when an output switches logic value, but this current decays to near zero in the steady state. Referring to Figures 2-4 and 2-5, when V_{OUT} switches, current flows to/from the output driver charging/discharging C_{LOAD}. Once C_{LOAD} is fully charged/discharged, however, current virtually ceases. There is no resistive element to sustain steady-state current. The input resistance of a receiving gate is so large when compared with R_T that it is effectively infinite and has been ignored in the figures.

The coding strategy is independent of the circuit technology of a digital system. Reference to circuit technology (e.g., ECL, TTL, or CMOS) is unnecessary since it is not a determining

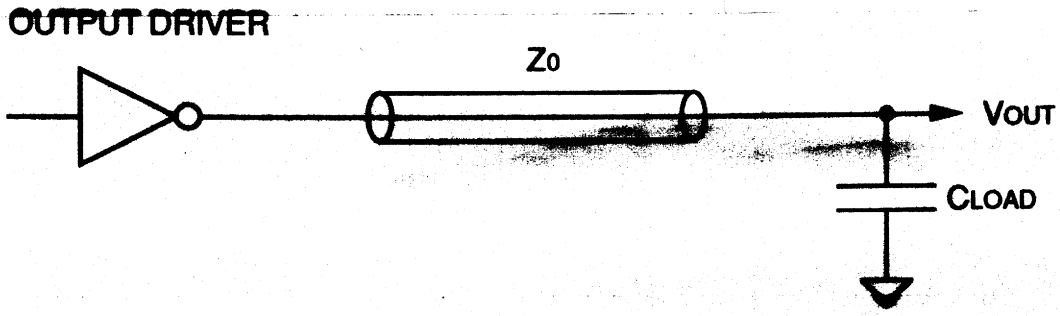

Figure 2-2: A model for signalling with an un-terminated transmission line.

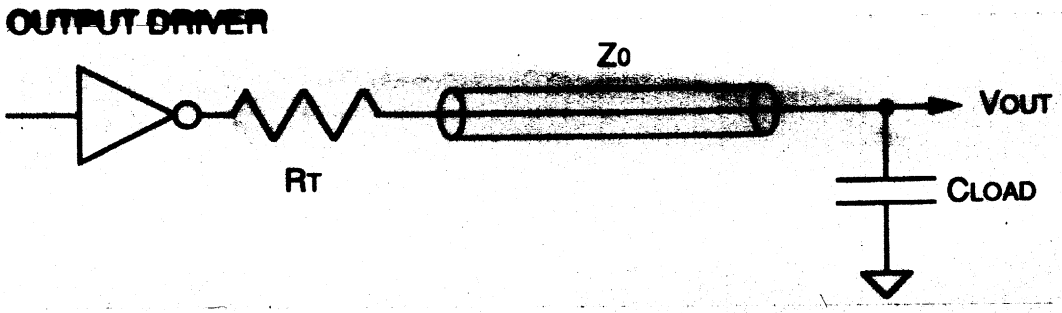

Figure 2-3: A model for signalling with a series-terminated transmission line.

factor as far as coding to reduce noise is concerned.

ECL is a popular, high-performance circuit technology. Chips in ECL circuits are commonly connected with parallel-terminated transmission lines. These circuits fall in the category of systems which exhibit steady-state current in their interconnection network and are a primary motivation of this document. It is important to understand, however, that a CMOS circuit using parallel-terminated transmission lines falls in the same category. An often proclaimed virtue of CMOS circuits is that they dissipate no static power. However, if CMOS chips are connected with parallel-terminated lines, then static power will be dissipated in the interconnection network.

CMOS is the dominant, high-performance circuit technology of our time. Chips in a CMOS circuit are often connected with un-terminated transmission lines. These circuits fall in the category of systems which exhibit negligible steady-state current in their interconnection networks and are another primary motivation of this document. Remember, chips of any circuit technology connected with series-terminated or un-terminated transmission lines fall in this category as well.

2.2 A Bit of Coding Jargon

Coding accomplishes noise reduction by limiting the variability of successive transmitted words. This in turn reduces the current variability on the transmitting chip's power-supply pins, the fundamental cause of the noise. Coding reduces the variability of words by restricting the number of 1's in the word. Coding and information theory define this number of 1's as the *weight* of the word. Because of this definition and to ease the reading of this paper, three *weight-based*, noise-reduction coding terms have been coined.

N-bit words of unrestricted weight have maximum weight N and minimum weight zero. Coding so as to reduce the maximum weight will be called *starvation* coding. specifically, starting with words of unrestricted weight, coding so as to limit words to weight J or below will be referred to as *starving* the words to weight J. In contrast, coding so as to increase the minimum weight will be called *indulgence* coding. Limiting words to weight K or above will be referred to as *indulging* the words to weight K.

Another useful technique involves coding so as to keep the weight at a constant or nearly

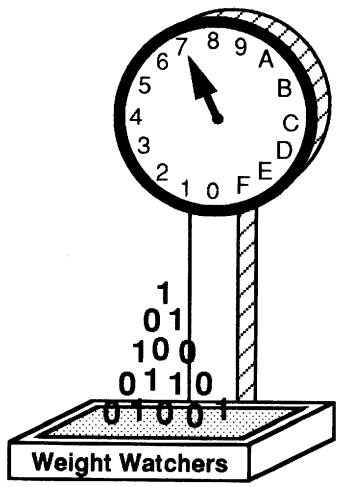

Figure 2-4: Coding and information theory's concept of weight leads to three coding terms.

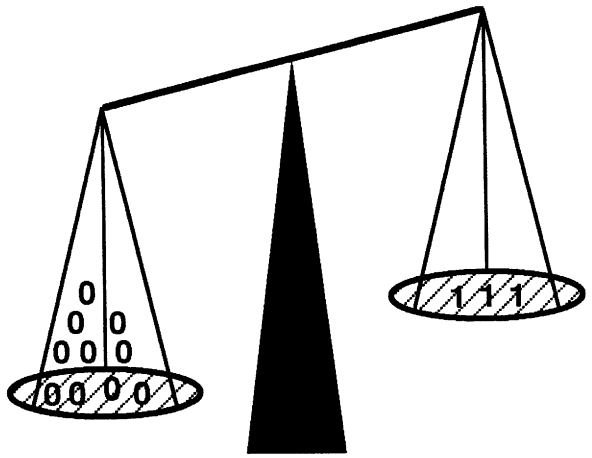

Figure 2-5: Starvation coding reduces a word's weight.

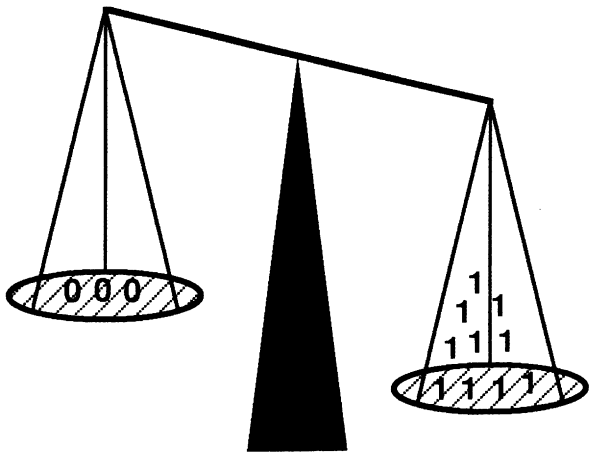

Figure 2-6: Indulgence coding increases a word's weight.

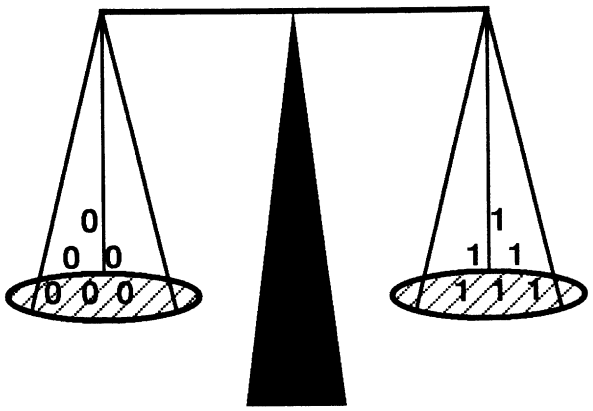

Figure 2-7: Ration coding keeps a word's weight constant.

constant level. This type of coding will be called *ration* coding. Limiting the weights of words to the range J to K will be referred to as *rationing* the words to weights J through K. Keep it in mind that all three techniques involve discarding coded words which lie outside a desired range. As a result the coded words require a greater number of bits to convey the same amount of information as their uncoded predecessors.

Throughout this document I will use the word *bit* in two ways, both as a digit in the number base 2 (often synonymous with the word *signal* or *wire*) and as a unit of information. I will use the phrase *bit of information* when referring to the latter definition and there could be confusion [Ham80]. Also, I will use N for the number of bits in the output word prior to coding and M for the expanded number of bits after coding (hence, $M > N$).

Chapter 3

Reducing Transmitted Noise Using Ration Coding

This chapter is devoted to coding to reduce transmitted noise in circuits exhibiting steady-state current in their interconnection networks. The conclusions reached apply to all such circuits. Because of its popularity, this document will often refer to a particular technology, ECL chips communicating with parallel-terminated transmission lines, though the results are more generally applicable.

3.1 An Example of Transmitted Noise in an ECL Circuit

Bipolar Integrated Technology's B3011 16x16 multiplier-accumulator is a 132-pin ECL chip capable of driving forty signals off chip simultaneously [Bip86]. The wide output bus makes the B3011 susceptible to generating transmitted noise. The output, emitter-follower circuits are shown schematically in Figure 3-1. Because all ECL chips have similar output circuitry, the problem described here pertains in varying degree to the entire ECL family. For this example, the chip is driving 50Ω transmission lines terminated in parallel to $V_{TT} = -2.0V$. Furthermore, assume there is just one output ground pin. This pin is labeled V_{CC2} with pin and bonding wire inductance L as shown.

Consider a transmitted-noise, worst-case scenario; all forty outputs are making low to high transitions. These transitions cause a large change in current on the V_{CC2} pin. With

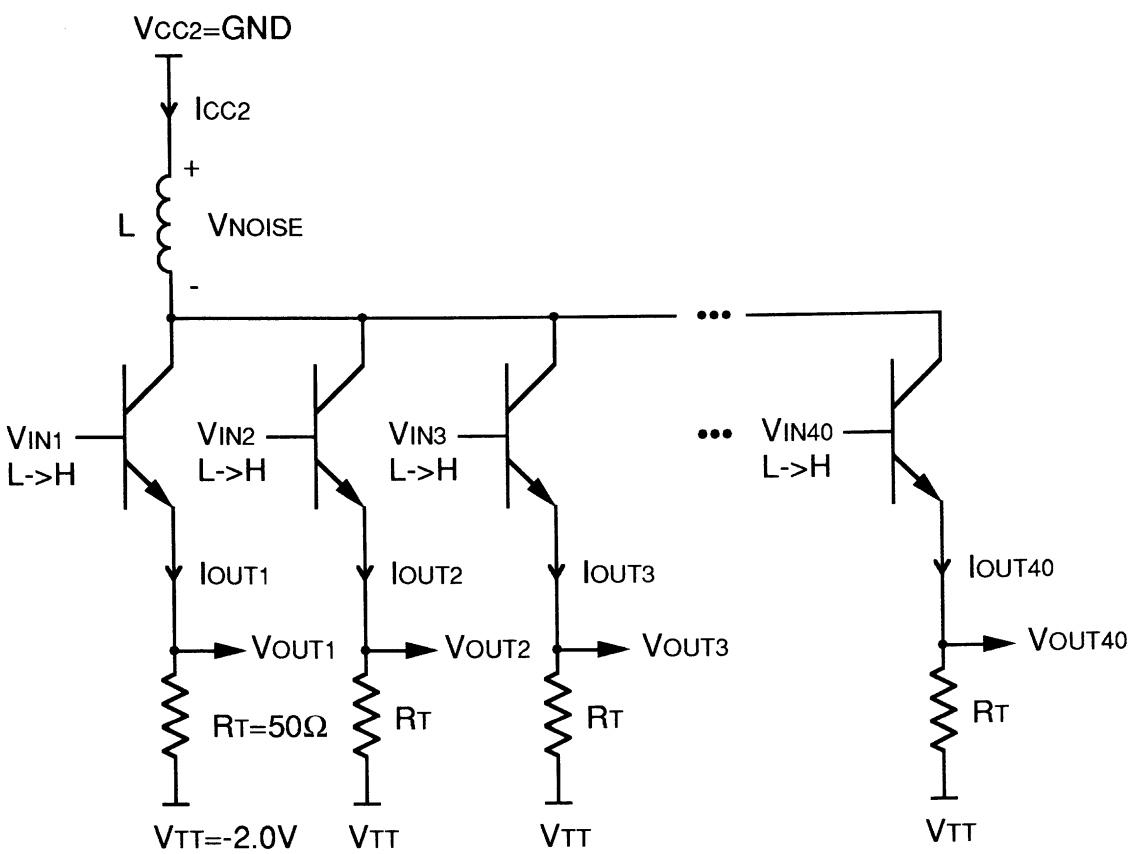

Figure 3-1: 40-bit parallel output circuit of the B3011 driving parallel-terminated transmission lines. Figure assumes one output ground pin (V_{CC2}).

$V_{OH} = -0.9V$, $V_{OL} = -1.75V$, and a transition time $T_{LH} = 1.5ns$ [Blo88], the output current rate of change, dI_{OUT}/dt, can be approximated as follows:

$$I_L = \frac{(2.0 - 1.75)V}{50\Omega} = 5.0mA \tag{3.1}$$

$$I_H = \frac{(2.0 - 0.90)V}{50\Omega} = 22.0mA \tag{3.2}$$

$$\frac{dI_{OUT}}{dt} \approx \frac{I_H - I_L}{T_{LH}} = \frac{(22.0 - 5.0)mA}{1.5 \times 10^{-9}s} = 1.13 \times 10^7 A/s \tag{3.3}$$

There are forty outputs changing, hence, the V_{CC2} current rate of change, dI_{CC2}/dt, can be approximated as follows:

$$\frac{dI_{CC2}}{dt} \approx 40(\frac{I_H - I_L}{T_{LH}}) = 4.53 \times 10^8 A/s \tag{3.4}$$

According to $V = L(dI/dt)$ this change in current will cause a voltage drop across the pin and bonding wire inductance. Package wiring has an inductance of roughly $10.0nH/cm$. Assuming the pin and bonding wire together measure 1.0cm, the voltage drop (V_{NOISE}) can be approximated as follows:

$$V_{NOISE} = L(\frac{dI_{CC2}}{dt}) \approx (10.0nH)(4.53 \times 10^8 A/s) = 4.6V \tag{3.5}$$

This amount of noise forces the emitter-follower circuits completely out of their intended region of operation. Let's examine one emitter-follower output circuit with a more instructive amount of noise (say, $V_{NOISE} = 1.0V$). Figure 3-2 depicts the transistor model used in this analysis [SS88, Blo88]. Figure 3-3 shows the resulting transfer curve, with noise (dashed lines) and without noise (solid line). Considering first the circuit when there is no noise, notice that the transistor is well within the linear region when signalling high and low logic values. However, with $1.0V$ of noise, the transistor saturates before the output can reach a valid logic 1 level. One must wait for the noise to go away before a valid 1 can be signalled.

BIT handles this problem by including six V_{CC2} pins on the chip [Bip86]. This reduces

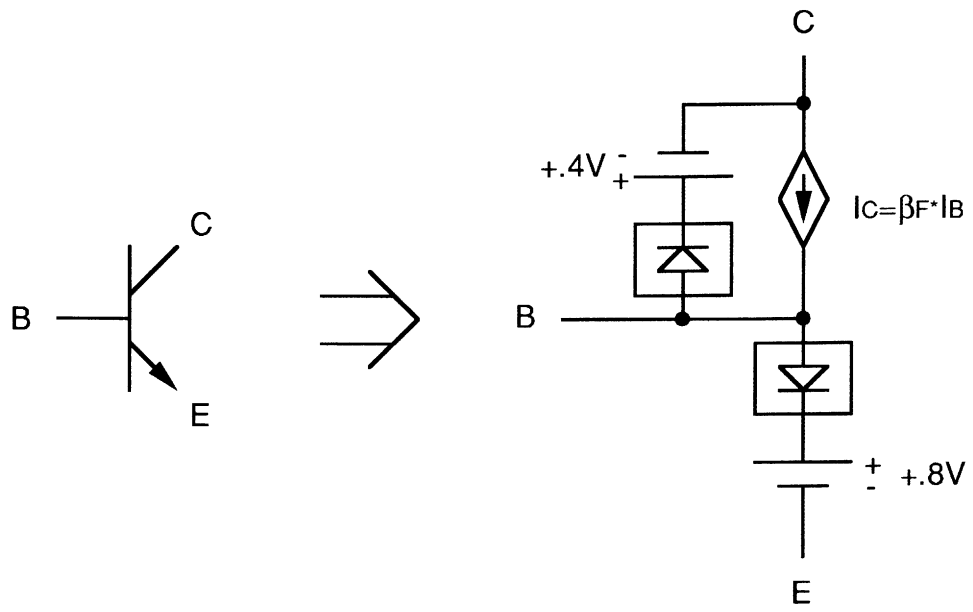

Figure 3-2: Transistor model used in emitter-follower analysis.

the worst-case noise by about a factor of six (from $2.3V$ to about $0.4V$). With $0.4V$ of noise, the emitter-follower transistor barely remains in the linear region when signalling a logic 1 (see Figure 3-3 again). With noise-reduction coding and less than six additional pins, it is possible to virtually eliminate the worst-case noise.

3.2 Ration Coding as a Solution

A type of noise-reduction coding, termed *ration* coding earlier, will nearly eliminate the noise in the example of the previous section. From equation 3.5 it is apparent that no noise will be generated if the current on the V_{CC2} pin does not change. The V_{CC2} pin carries the current for all forty output signals. In order to keep the V_{CC2} current constant, the sum of the forty output currents must remain the same. The outputs need not remain static, but the total demand from the group must not change.

Notice from equations 3.1 and 3.2 that more than four times as much current flows when signalling a 1 as when signalling a 0. A large change in current results from a transition such

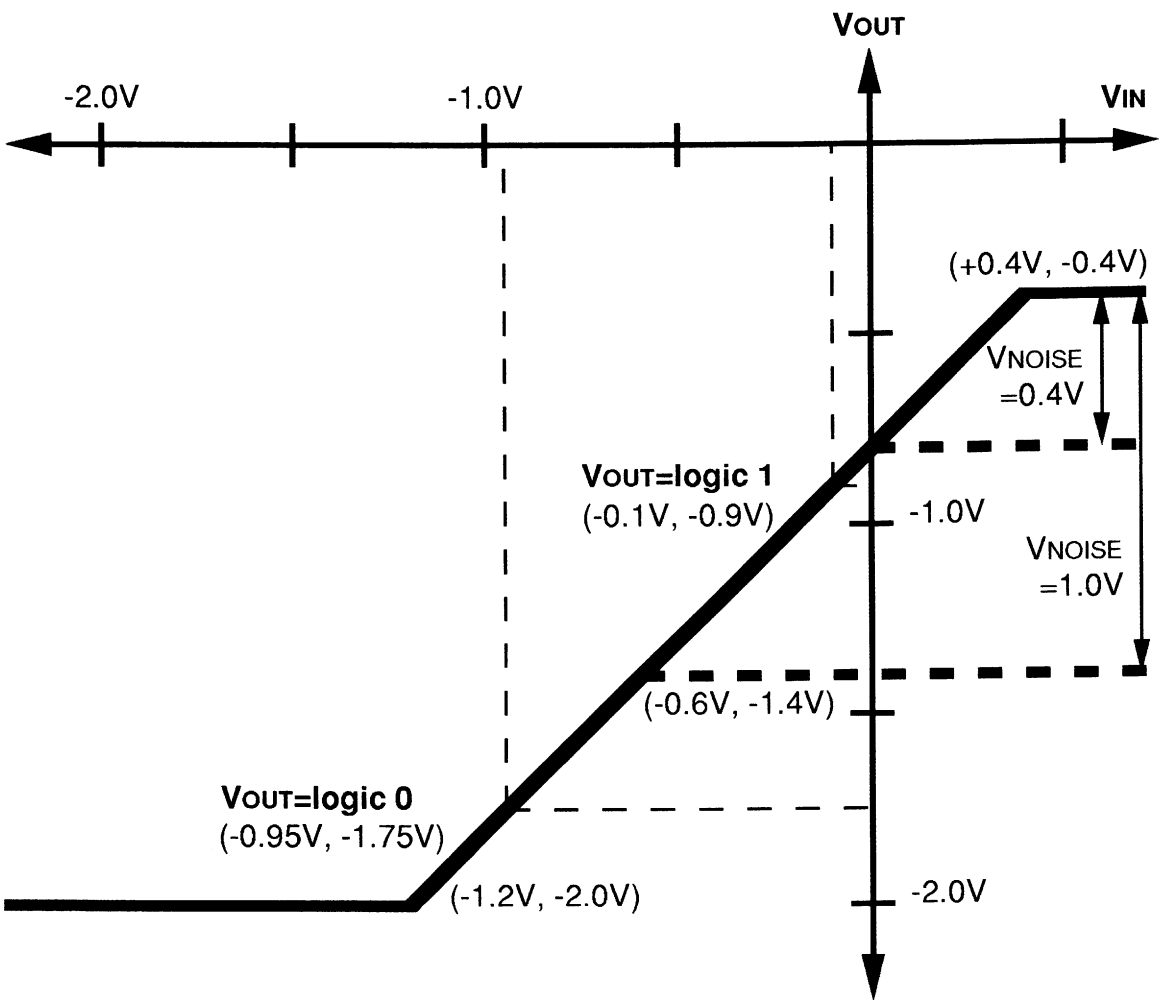

Figure 3-3: Emitter-follower transfer curve with noise (dashed lines) and without noise (solid line).

as $WORD(t) = 0000...00$ to $WORD(t+1) = 1111...11$, whereas, a negligible change results from a transition such as $WORD(t) = 0101...01$ to $WORD(t+1) = 1010...10$. Therefore, if for every high-to-low transition there is a corresponding low-to-high transition, then the V_{CC2} current will be nearly constant, and negligible noise will be generated. Another way to view this is if the chip is signalling the same number of 1's from cycle to cycle (this, of course, guarantees signalling the same number of 0's as well) then negligible noise will be generated. This can be achieved with ration coding. (The V_{CC2} current required to charge the receiving-gate input and interconnect capacitances, C_{LOAD} from section 2.1, cannot be simply re-directed from one output to another. For this reason, even when signalling the same number of 1's from cycle to cycle, small changes in current will occur.)

A technique called differential signalling was employed to nearly eliminate transmitted noise in the Cray-1 computer system [Kol81]. Here, a signal and its inverse transmit one bit of information. Differential signalling reduces transmitted noise to its limit since it guarantees a constant output current for each information bit but requires a $2\times$ increase in output pins and wires. Ration coding achieves similar results at smaller pin count.

3.3 Theoretic Limits of Ration Coding

Ration coding was defined earlier as a mapping from words with unrestricted weight (recall *weight* equals the number of 1's in a word) to words with constant or nearly constant weight. From a power conservation point of view, since in the ECL technology signalling a 1 requires more current than signalling a 0, it is desirable to make the constant weight a small number. Unfortunately, most of the information-carrying capacity of an M-bit word is contributed by words with weight near $M/2$. This fact will be demonstrated shortly.

It was argued in section 1.1.4.1. that noise-reduction coding required the expansion of the output-word width (from N bits to M bits). Since a total of $M - N$ additional output pins are required, it is desirable to restrict the magnitude of M. Throughout this document the notation $C(M,W)$ will be used to represent the number of different M-bit words with weight W. Therefore,

$$C(M,W) = \frac{M!}{(M-W)!W!} \tag{3.6}$$

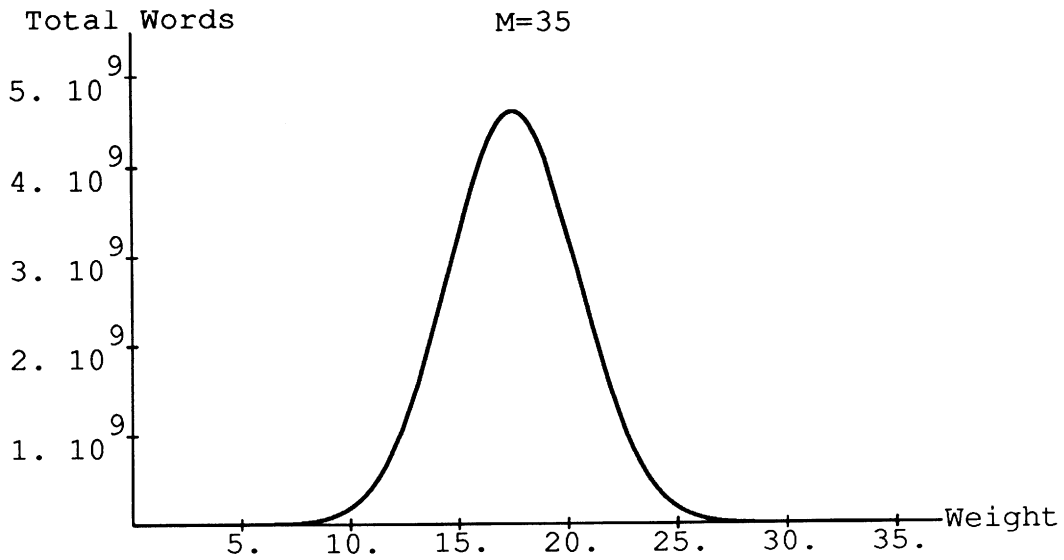

Figure 3-4: The number of different, constant-weight, 35-bit words plotted as a function of weight, $C(35, W)$.

[Dra67] Figure 3-4 plots this relationship as a function of W with $M = 35$. Notice that most of the information-carrying capacity resides near $W = 35/2$.

Figure 3-4 should give the reader a qualitative understanding of how ration coding works. Starting with an N-bit word with unrestricted weight, ration coding expands the word to M bits while restricting the weight to near $M/2$. Of course, ration coding will most successfully reduce transmitted noise when words are restricted to a single weight. Ignoring implementation details for the moment, what is the minimum number of extra bits (define $E = M - N$) needed when rationing an N-bit word to a single weight near $M/2$?

The total number of different N-bit words is equal to 2^N. In order to encode each of these N-bit words as a different M-bit, constant-weight word, enough such words must exist. This is to say, M-bit words with constant weight $M/2$ will have the information-carrying capacity to replace N-bit uncoded words, only if the following holds:

$$C(M, M/2) \geq 2^N \tag{3.7}$$

or, using equation 3.6,

$$\frac{M!}{(M/2)!(M/2)!} \geq 2^N \tag{3.8}$$

Using Stirling's approximation,

$$M! \approx \sqrt{2\pi} M^{M+(1/2)} e^{-M} \tag{3.9}$$

an expression for the additional bits required by the encoded word, E, can be derived. (Stirling's approximation improves as M gets larger. It is within 1% for $M = 9$ [Ham80].)

Starting with equation 3.8, Stirling's approximation gives

$$\frac{\sqrt{2\pi} M^{M+(1/2)} e^{-M}}{2\pi((M/2)^{(M/2)+1/2} e^{-M/2})^2} \geq 2^N \tag{3.10}$$

simplifying,

$$(\frac{1}{\sqrt{2\pi}}) \frac{M^{M+(1/2)}}{(M/2)^{M+1}} \geq 2^N \tag{3.11}$$

$$(\frac{1}{\sqrt{2\pi}}) \frac{2^{M+1} M^{M+(1/2)}}{M^{M+1}} \geq 2^N \tag{3.12}$$

$$(\frac{1}{\sqrt{2\pi}}) \frac{2^{M+1}}{M^{1/2}} \geq 2^N \tag{3.13}$$

and, since $M = 2^{log_2 M}$,

$$(\frac{1}{\sqrt{2\pi}}) \frac{2^{M+1}}{(2^{log_2 M})^{1/2}} \geq 2^N \tag{3.14}$$

further simplifying,

$$(\frac{1}{\sqrt{2\pi}}) \frac{2^{M+1}}{2^{(1/2) log_2 M}} \geq 2^N \tag{3.15}$$

$$(\frac{1}{\sqrt{2\pi}}) 2^{(M+1)-(1/2) log_2 M} \geq 2^N \tag{3.16}$$

taking the logarithm of both sides,

$$-log_2 \sqrt{2\pi} + M + 1 - (1/2) log_2 M \geq N \tag{3.17}$$

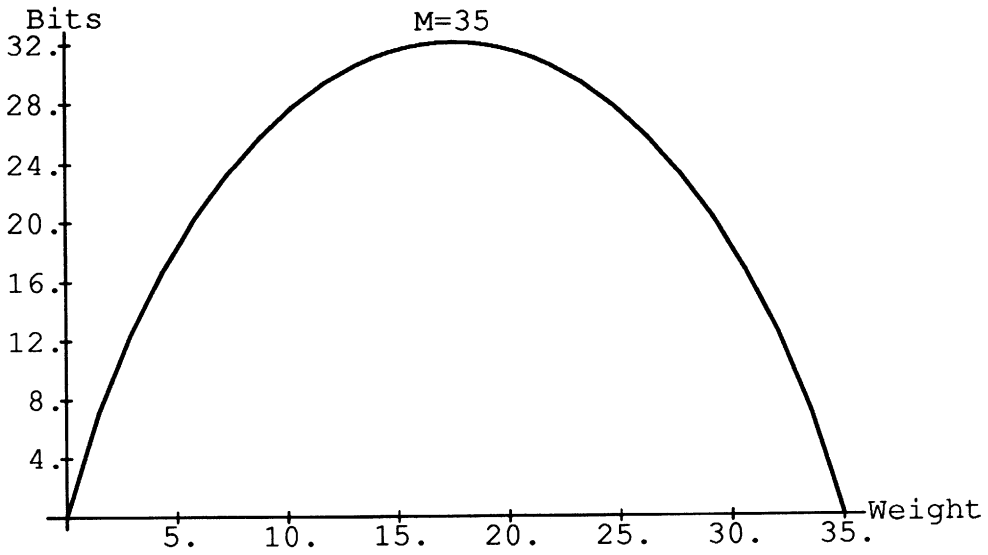

Figure 3-5: The information capacity (in bits) of a constant-weight, 35-bit word plotted as a function of weight, $log_2 C(35, W)$.

finally, substituting $M = N + E$ and solving for E yields

$$-log_2\sqrt{2\pi} + N + E + 1 - (1/2)log_2(N + E) \geq N \tag{3.18}$$

$$E \geq (1/2)log_2(N + E) + log_2\sqrt{2\pi} - 1 \tag{3.19}$$

$$E \geq (1/2)log_2(N + E) + log_2\sqrt{\pi/2} \tag{3.20}$$

Unfortunately, this expression is not a simple function of N. Hence, the minimum value of E can only be found through iterative solution.

Using this expression, only three additional bits are required when rationing a 32-bit word to weight 17. This is depicted in Figure 3-5 which plots the information capacity of 35-bit, constant-weight words. Notice that 32 bits of information can be encoded as 35-bit words with weight always equal to 17. (Since $C(35, 17) = C(35, 18)$, 32 bits of information can also

be encoded as 35-bit words with weight always equal to 18. It is true for all odd M that the same number of constant-weight words exist for the two integer weights nearest $M/2$. To conserve power the smaller weight should be implemented.)

3.4 Implementations of Ration Coding

Ration coding may be implemented in many ways. The implementation technologies fall in two categories: table look-up and algorithmic implementations. The remainder of this chapter contains examples of these techniques. This section is somewhat incomplete; the topic remains an active area of research.

3.4.1 Table Look-Up

Ration coding can be implemented in a very straightforward manner using look-up tables. Design simplicity is one of the chief advantages of this approach. Tough decisions must be made regarding the chip area, speed, and additional pins demanded by a particular application. These trade-offs will be examined in a moment.

We can choose positive integers P, Q, R, S, and V such that the following holds:

$$C(Q,R-V)+...+C(Q,R-1)+C(Q,R)+C(Q,R+1)+...+C(Q,R+S)\geq 2^P \quad (3.21)$$

This enables us to encode a P-bit word of unrestricted weight as a Q-bit word with weight from $R-V$ to $R+S$. This, of course, is ration coding. We are rationing a P-bit word to weights $R-V$ through $R+S$.

Since look-up table area grows exponentially with the number of address bits, it would be impractical to build a single ration-coding table in cases like the B3011. A table with forty address bits would require a ridiculous amount of chip area (on the order of a square meter). In cases where the output-word width, N, is larger than about ten, it will be necessary to segment N into manageable size sections.

Referring to equation 3.21 and to Figure 3-6, we can segment an N-bit output word into N/P sections. Each P-bit section can be rationed to Q bits with weight from $R-V$ to $R+S$. In total our N-bit words will be rationed to M-bit words, where $M=(N/P)Q$, with weights

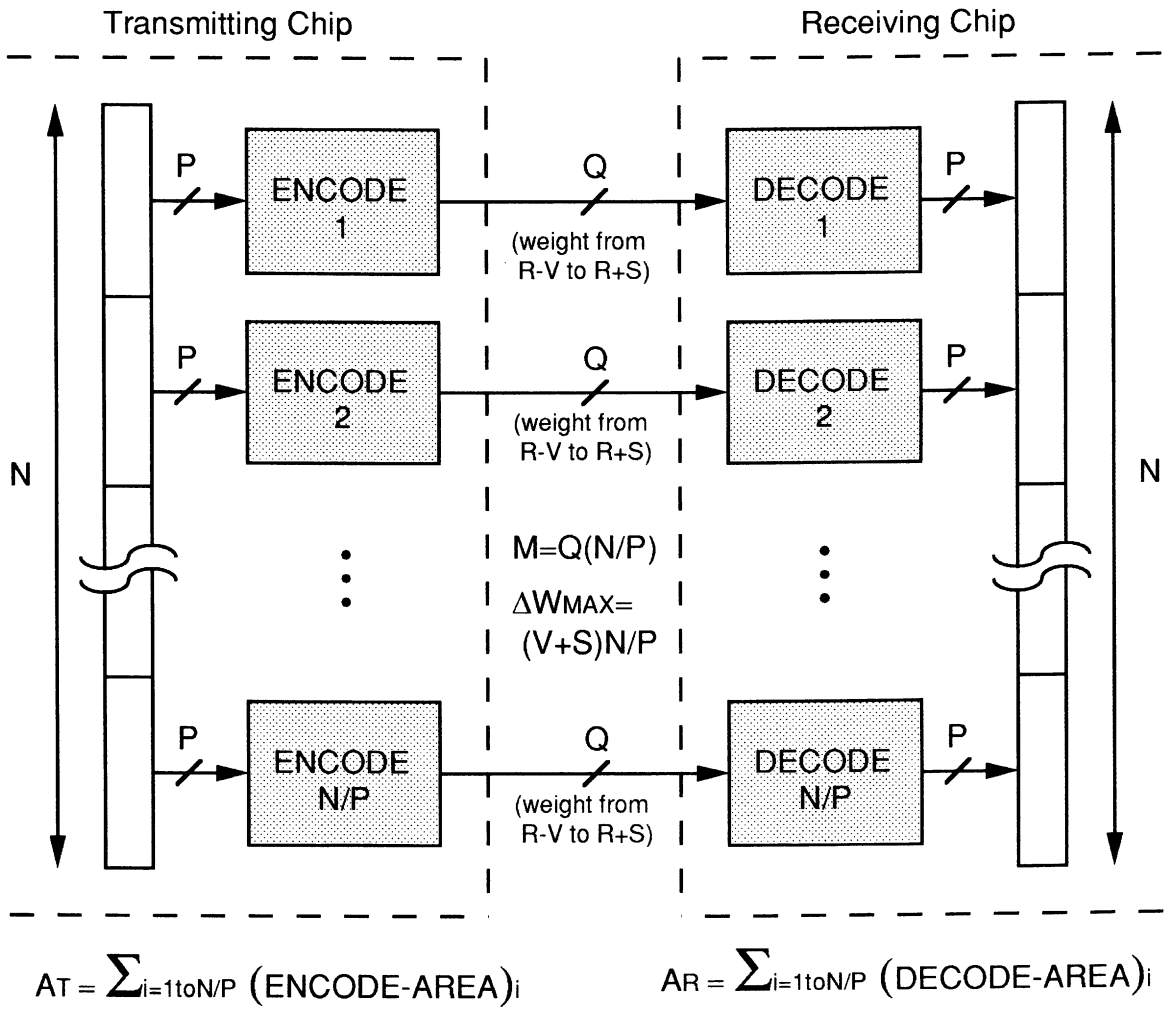

Figure 3-6: Description of parameters for ration coding using table look-up.

Table 3.1: Presented are the trade-offs between noise generation, ΔW_{MAX}, output pins, M, and table area, A_T and A_R, for several implementations of ration coding using look-up tables ($N = 64$).

P	Q	N/P	R	S	V	ΔW_{MAX}	M	A_T	A_R
9	10	7.11	4	1	1	15	71	35,840	64,512
8	11	8	4	0	0	0	88	22,528	131,072
8	10	8	3	1	0	8	80	20,480	65,536
8	9	8	4	1	1	16	72	18,432	32,768
8	9	8	2	2	2	32	72	18,432	32,768
7	9	9.14	4	0	0	1	82	10,368	32,256
7	8	9.14	3	1	1	19	73	9216	16,128
6	8	10.67	4	0	0	0	88	5632	16,896
6	7	10.67	3	1	1	22	77	4928	8448
6	7	10.67	1	2	1	33	77	4928	8448
5	7	12.8	3	0	0	0	91	2912	8320
5	6	12.8	2	1	0	13	78	2496	4160
4	6	16	3	0	0	0	96	1536	4096
4	5	16	2	1	0	16	80	1280	2048
4	5	16	1	1	1	32	80	1280	2048
3	5	21.33	2	0	0	1	106	840	2016
3	4	21.33	1	1	0	22	85	672	1008

$(R - V)(N/P)$ through $(R + S)(N/P)$. Table 3.1 contains the design characteristics resulting from a variety of P, Q, R, S, and V combinations for $N = 64$.

Worst-case, transmitted-noise generation will be proportional to ΔW_{MAX}. Ideally $S = V = 0$, there is no weight variation, and transmitted noise is virtually eliminated. The noise generated when using ration coding under these circumstances should be equivalent to that generated when using differential signalling. The number of output pins required by ration coding is M. The number of output pins required by differential signalling is $2N$. In all cases $M < 2N$. Of course, the price paid for fewer output pins is chip area and encoding and decoding time.

ΔW_{MAX}, M, A_T, and A_R were a little tricky to calculate. (A_T and A_R are proportional to the total look-up table area required by the transmitting and receiving chips, respectively. Actual area can be determined by multiplying these numbers by the area of a single bit cell.)

In cases where N is evenly divisible by P, these values were calculated as follows:

$$\Delta W_{MAX} = (S + V)(N/P) \tag{3.22}$$

$$M = (N/P)Q \tag{3.23}$$

$$A_T = (2^P * Q)(N/P) \tag{3.25}$$

When (N/P) is not an integer, a decision must be made regarding the treatment of the *remainder* bit(s). This decision affects all four values. For instance, in the case where $P = 7$, it was decided to segment the 64 bits into 9 groups of 7 and pass the remaining bit without coding. (Since $9*7 = 63$, one bit must bypass the tables.) Therefore, $\Delta W_{MAX} = 9(S+V)+1$, $M = 9Q + 1$, $A_T = 9(2^7 * Q)$, and $A_R = 9(2^Q * 7)$. In the case where $P = 5$, it was decided to segment the 64 bits into 13 groups of 5 creating one dummy table-input bit. (Since $13*5 = 65$, there is one extra table input.) Therefore, $\Delta W_{MAX} = 13(S+V)$, $M = 13Q$, $A_T = 13(2^5 * Q)$, and $A_R = 13(2^Q * 5)$. The table look-up scheme is less efficient when N/P is not an integer.

While examining Table 3.1, one should keep in mind the following:

- Look-up table area increases exponentially with the size of the sections, P.
- The number of pins, M, increases with the number of sections, N/P.
- Large look-up tables are slow, so speed is inversely proportional to P.
- Speed is so critical that, despite the fact each of the N/P tables is identical, it is unlikely the tables could be multiplexed.

3.4.2 Algorithmic Implementations

Ration coding can be implemented using a recursive algorithm. Figure 3-7 contains a block-diagram showing two recursions of this algorithm. Starting with an N-bit output word, segment the word into N/P sections of P bits. During the first recursion each P-bit section is processed separately as follows:

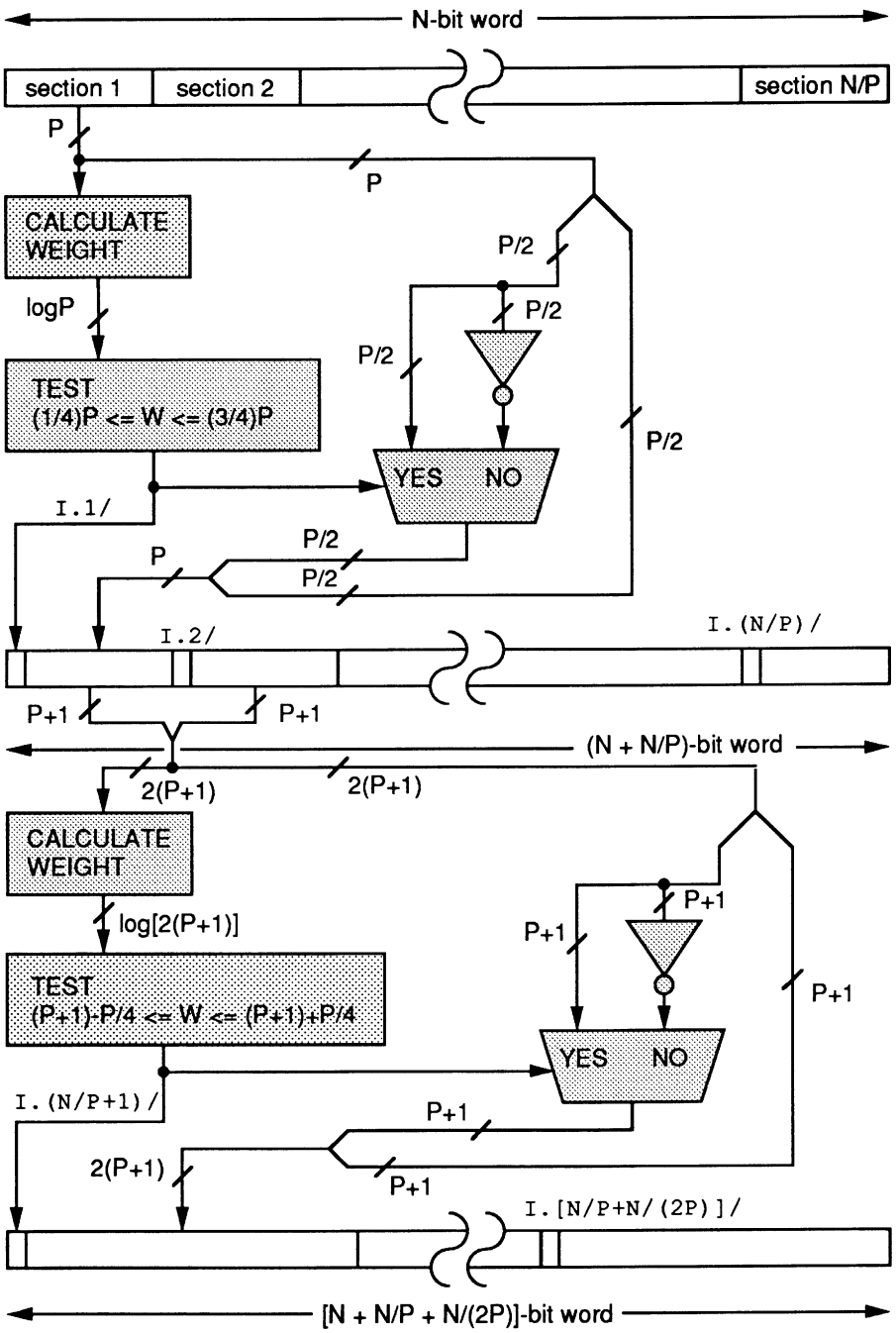

Figure 3-7: Ration coding implemented with a recursive algorithm.

- Calculate the weight, W_1, of the section ($0 \leq W_1 \leq P$).
- Test whether W_1 is less than or exactly $(1/4)P$ away from half the total number of bits in the section (i.e., test $(1/2)P - (1/4)P = (1/4)P \leq W_1 \leq (1/2)P + (1/4)P = (3/4)P$).
- If the test returns false, signal I.1/ = 0. This selects the inverted form of half the section's bits. (I is an initial for *invert*, 1 refers to the first of P sections, and / denotes that the signal is active low.) The inversion forces the section's weight into the given range. Also, include I.1/ = 0 in the section's bits to inform the receiver of the inversion.
- If the test returns true, signal I.1/ = 1. This selects the non-inverted form of half the section's bits. Include I.1/ = 1 in the section's bits.

After one recursion each section is $(P + 1)$ bits long. Furthermore, we are guaranteed for each section $(1/4)P \leq W_1 \leq (3/4)P$. A second recursion proceeds as follows:

- Combine two $(P + 1)$-bit sections to form one larger section.
- Calculate the weight, W_2, of the section. We are guaranteed $(1/2)P \leq W_2 \leq (3/2)P$, but we would like to restrict the weight further.
- As before, test whether W_2 is less than or exactly $(1/4)P$ away from half the total number of bits in the section (i.e., test $(P + 1) - (1/4)P \leq W_2 \leq (P + 1) + (1/4)P$).
- If false, invert half and include I.$(N/P) + 1/ = 0$ in the section's bits.
- If true, just include I.$(N/P) + 1/ = 1$ in the section's bits.

After the second recursion each section is $2(P + 1) + 1$ bits long with $(P + 1) - (1/4)P \leq W_2 \leq (P + 1) + (1/4)P$. Continue recursing until there is only one section. When finished, the number of possible different weights will equal $(1/2)P + 1$. Table 3.2 contains the tests required for various values of P with $N = 64$.

Implementing ration coding using this algorithm involves trade-offs similar to those discussed for table look-up. When choosing P one must realize that while small sections lead to a more constant output weight they require more recursions and, hence, more time, chip area, and additional output pins. Table 3.3 shows these trade-offs for $N = 64$.

Table 3.2: Tests required by recursive algorithm for various values of P with $N = 64$.

Recursion	$P = 64$	$P = 32$	$P = 16$
1	$16 \leq W_1 \leq 48$	$8 \leq W_1 \leq 24$	$4 \leq W_1 \leq 12$
2	-	$25 \leq W_2 \leq 41$	$13 \leq W_2 \leq 21$
3	-	-	$31 \leq W_3 \leq 39$
4	-	-	-
5	-	-	-
6	-	-	-

Recursion	$P = 8$	$P = 4$	$P = 2$
1	$2 \leq W_1 \leq 6$	$1 \leq W_1 \leq 3$	$W_1 = 1$
2	$7 \leq W_2 \leq 11$	$4 \leq W_2 \leq 6$	$W_2 = 3$
3	$17 \leq W_3 \leq 21$	$10 \leq W_3 \leq 12$	$W_3 = 7$
4	$37 \leq W_4 \leq 41$	$22 \leq W_4 \leq 24$	$W_4 = 15$
5	-	$46 \leq W_5 \leq 48$	$W_5 = 31$
6	-	-	$W_6 = 63$

Table 3.3: Trade-offs to consider when using recursive algorithm to implement ration coding.

P	*Recursions Required*	*Output Weight*	*Additional Pins*
2	6	$63 \leq W \leq 64$	63
4	5	$46 \leq W \leq 48$	31
8	4	$37 \leq W \leq 41$	15
16	3	$31 \leq W \leq 39$	7
32	2	$25 \leq W \leq 41$	3
64	1	$16 \leq W \leq 48$	1
NA	0	$0 \leq W \leq 64$	0

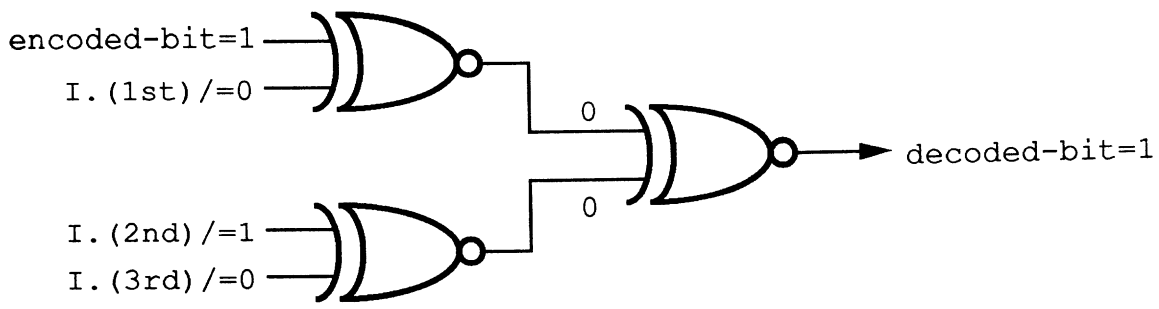

Figure 3-8: Decoding a bit which was encoded using the algorithm with three steps of recursion. (Note: I.$(Kth)/ = 0$ means an inversion took place during the Kth recursion.)

One distinct advantage of this implementation of ration coding is that it requires simple decode circuitry. Table look-up requires a set of decode tables. These tables consume more time and chip area than the encode tables. The recursive algorithm only requires circuitry to invert each encoded bit according to its set of invert signals. (In the case of K recursions, each bit has a set of K invert signals, I.$(1st)/$, I.$(2nd)/$, ..., I.$(Kth)/$.) The decode can be done in a time proportional to $log_2 K$ with a tree of XNOR gates (see Figure 3-8).

However, the recursive algorithm is not as efficient as table look-up; a constant output weight cannot be achieved even if the maximum number of recursion steps is implemented. Referring again to Table 3.3, the maximum number of recursion steps for $N = 64$ is six. Even with this many steps, the output can still differ in weight from cycle to cycle. (In this case the output weight can be either sixty-three or sixty-four.) Constant output weight can be achieved through several different implementations of ration coding using table look-up. (Refer to Table 3.1 again if necessary.)

Another disadvantage of the ration-coding, recursive algorithm is that it requires counting and comparing; these operations are slow. Since the operations need not be precise for this application- a fast, approximate count-and-compare circuit can be built to mitigate the disadvantage. Figure 3-9 depicts an example of such a circuit.

Implementing the algorithm with one step of recursion for a 32-bit output bus requires testing $8 \leq W_1 \leq 24$. Figure 3-9 shows schematically a fast circuit which performs part of this test. This circuit is fast because the count-and-compare operation is done in an analog

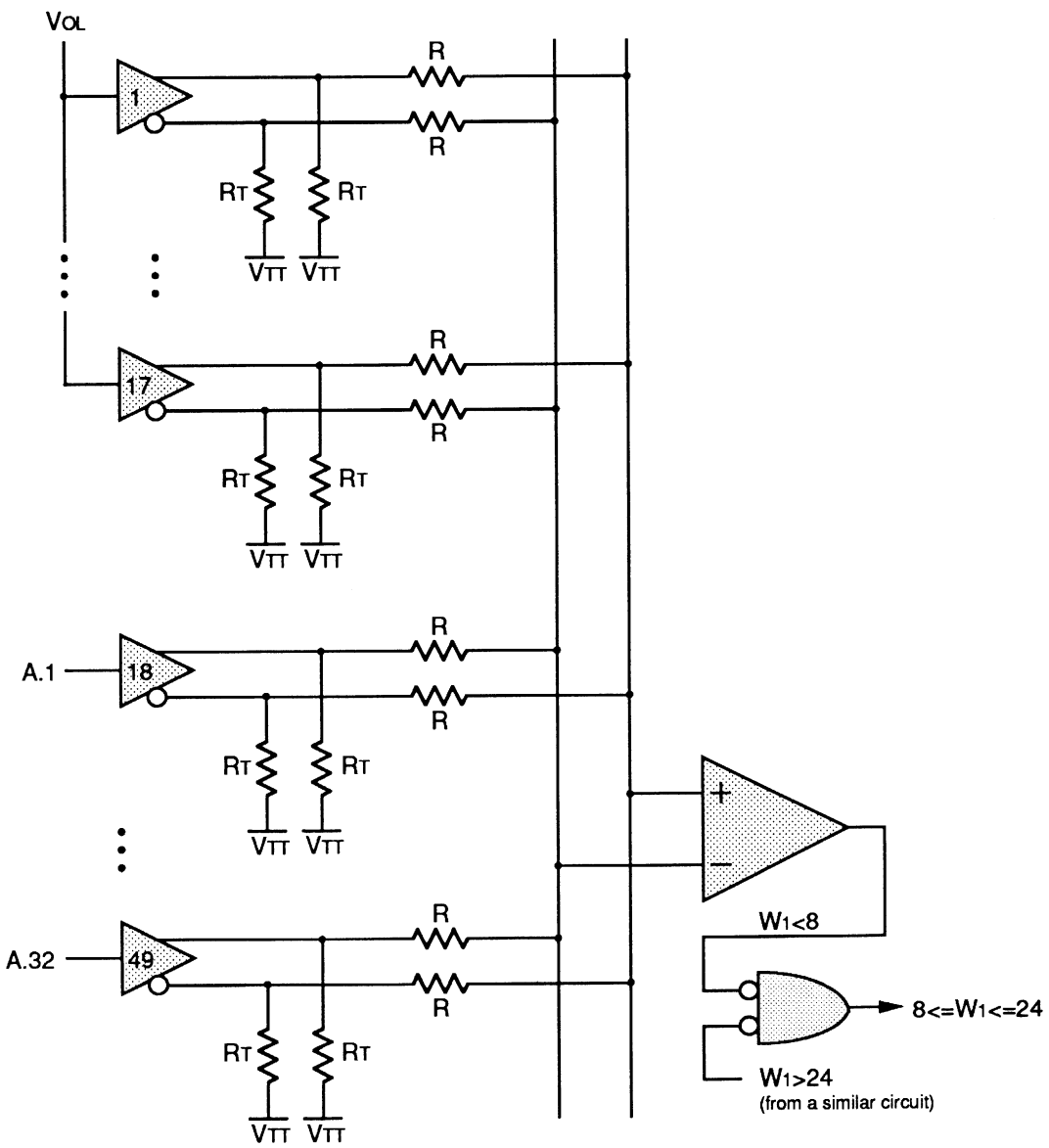

Figure 3-9: Fast, analog, count-and-compare circuit for implementing recursive algorithm

manner. The digital output of the comparator has the same logic state as the majority of the signals tied to its non-inverting input. (The voltage at each comparator input is the analog average of the forty-nine buffer/inverter digital outputs tied to it through the R resistors.) Seventeen additional buffer/inverter devices are used since the majority function is not desired here. They shift the switching point of the comparator to the desired location. Holding all seventeen at one logic level enables the comparator to differentiate between twenty-four high, of the remaining thirty-two buffer/inverter outputs, and twenty-five high. This, of course, permits a desired weight test (i.e., $W_1 > 24$) on the 32-bit output word ($A.1$ - $A.32$). A second test, $W_1 < 8$, can be similarly performed using another comparator and the inverting outputs of another set of buffer/inverter devices. NORing these results together produces the desired $8 \leq W_1 \leq 24$ signal.

A trade-off is involved when choosing the resistor values. R_T must be close to the characteristic impedance of the transmission line. This value should not be so small that it demands too much current from the buffer/inverter devices. The circuit will work for a range of R values, but keep it in mind that large R leads to a slow circuit while small R puts more current demand on the buffer/inverter devices.

Other algorithmic implementations of ration coding have been considered. Unfortunately, all have been found to involve sophisticated computations (such as multiplication) and, hence, too much time.

3.5 Ration Coding: Noise Reduction for All Ages

Without changing its functionality, varying amounts of transmitted noise may be generated by a chip due to the signalling technology employed by a circuit. At one extreme of noise generation, there is single-ended signalling (i.e., one output signal for each information bit). This situation results in the maximum amount of transmitted noise.

At the other extreme there is differential signalling. This is the *brute-force* approach to reducing the output-driver current variability; transmit the inverse of the output signals also, doubling the number of signals and reducing current variability (and, hence, transmitted noise) to its limit. Ration coding can be customized to reduce noise anywhere in the range from single-ended signalling to differential signalling. (In fact, differential signalling is exactly

an implementation of ration coding where each bit is encoded as two signals with a constant weight of one.)

The advantages of single-ended signalling are obvious. If a circuit can handle the noise its chips are creating, then it makes no sense to waste time (both design time and execution time), chip area, and output pins on a nonexistent problem.

However, if a circuit cannot handle the noise its chips are creating, it does not make sense to blindly double the output signals and employ differential signalling. One should consider using the ration coding techniques discussed previously. Time, chip area, and output signals can be balanced through ration coding to suit any application.

Common mode rejection is a virtue of differential signalling. Referring to Figure 3-10, an output signal, $A.x$, and its inverse, $A.x/$, encounter the same disturbances as they traverse from transmitting chip to receiving chip. For this reason the noise component, V_N, for each signal is the same. Because this component is effectively subtracted out by the receiving chip's differential amplifier, the two signals are compared as though the noise did not exist. (Comparing $A.x$ with $A.x/$ is the same as comparing $A.x + V_N$ with $A.x/ + V_N$.) Common mode rejection is fundamental to differential signalling.

In many cases simple additional input circuitry can be added to a ration-coding implementation for it to exhibit common mode rejection. One example is shown in Figure 3-11. Imagine this is a table look-up implementation where the output word has been expanded from N bits with unrestricted weight to M bits with constant weight $M/2$. The lower input to all the differential amplifiers is the same. Due to the resistors (labeled R) the potential at this node (V_{AVG}) is the average of all the signal-node potentials. Since half of the signal nodes are at $V_{OL} + V_N$ and half are at $V_{OH} + V_N$ (a constant weight of $M/2$ guarantees this) the potential of the common node is $V_{AVG} = 1/2(V_{OL} + V_{OH}) + V_N$. This implementation of ration coding exhibits common mode rejection since the noise (V_N) has no effect on the comparison made by the differential amplifiers. (Comparing $A.x$ with $1/2(V_{OL} + V_{OH})$ is the same as comparing $A.x + V_N$ with $1/2(V_{OL} + V_{OH}) + V_N$.)

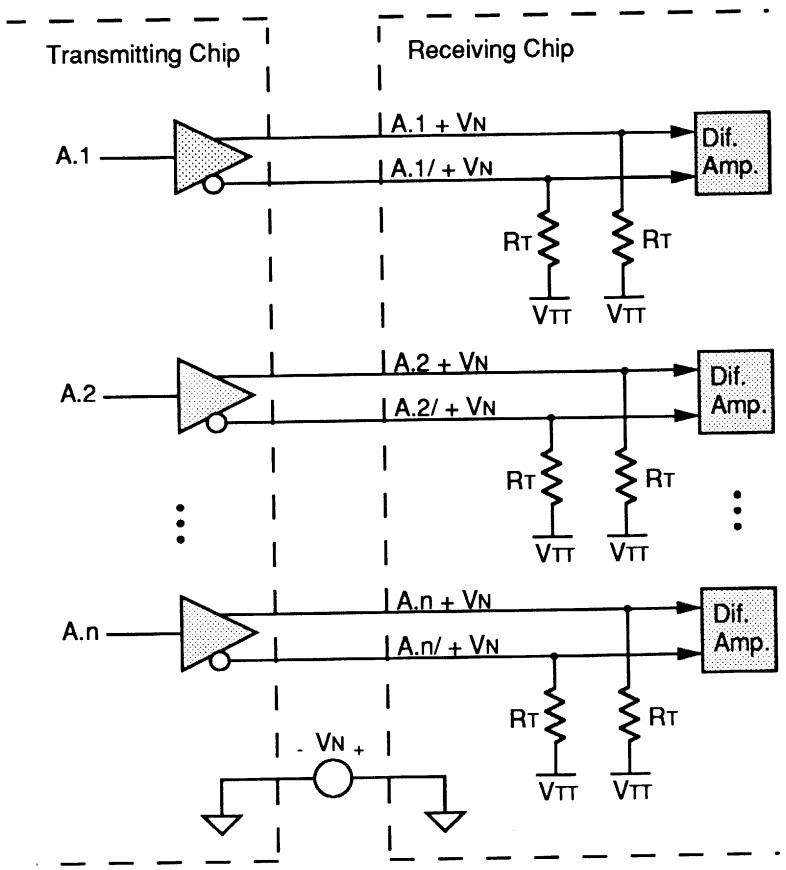

Figure 3-10: Differential signalling exhibiting common mode rejection.

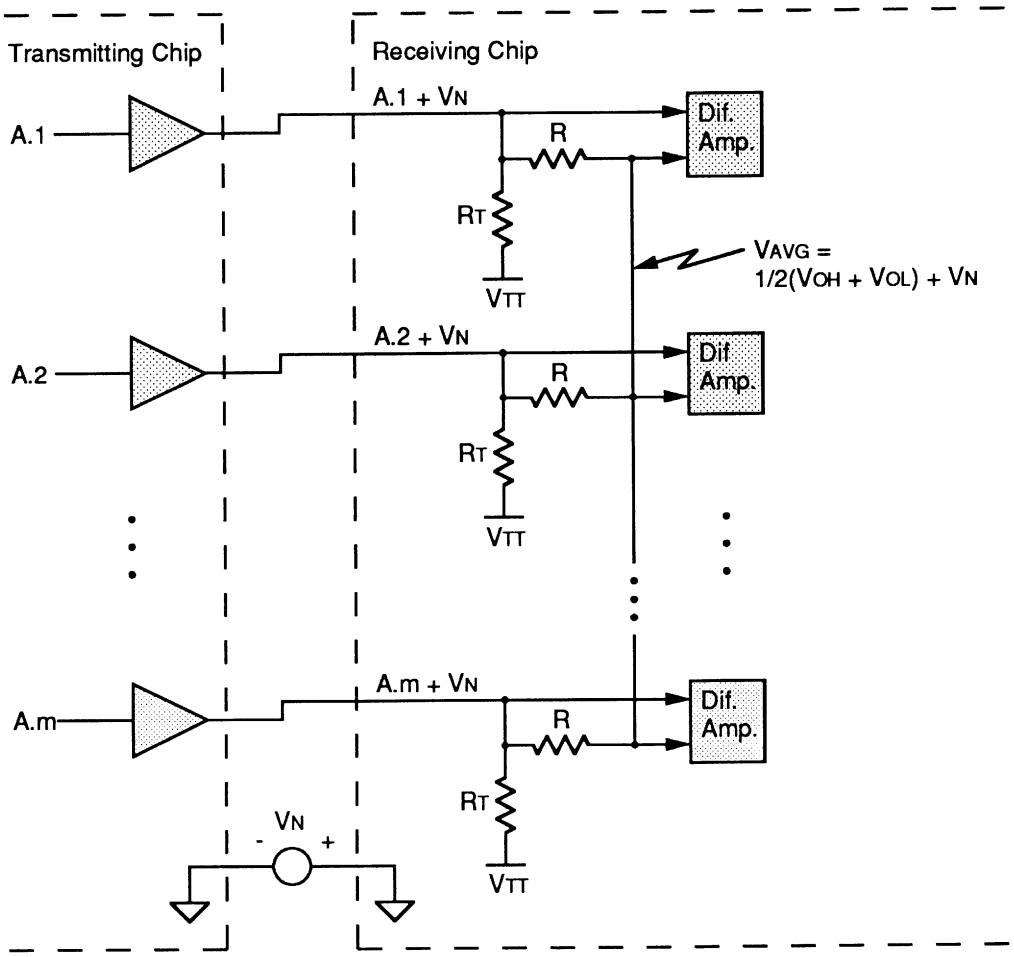

Figure 3-11: A ration-coding implementation exhibiting common mode rejection.

3.6 Demonstration of Ration Coding Reducing Noise

I built a digital circuit to demonstrate ration coding reducing transmitted noise. A block diagram of this circuit appears as Figure 3-12. This circuit does not directly prove ration coding to reduce the type of transmitted noise discussed throughout this document (i.e., the voltage drop between chip and power supply due to the parasitic inductance of pin and bonding wire). Time constraints forbid this experiment. The circuit does demonstrate ration coding reducing another form of transmitted noise. The two types of transmitted noise have similar causes. Hence, the noise reduction measurements are instructive here. The analogy between the two forms of transmitted noise will be discussed after the operation of the circuit has been presented.

The circuit consists of two component-carrying printed circuit boards: a board transmitting signals (PCB1) and a board receiving signals (PCB2). The experiment proceeds by, first, transmitting information with uncoded signals from PCB1 to PCB2 and measuring the noise generated. Next, use ration coding to encode the signals, transmit the same information, and measure the noise again. Comparison of the two measurements should show ration coding to have reduced the generation of noise. When applicable, a comparison with differential signalling should also be made.

PCB1 contains TTL and ECL devices and operates as follows (see Figure 3-12): The counter addresses 256 consecutive bytes stored in the EPROM repetitively. The data bytes are sequentially loaded into octal flip-flops 1.1 through 1.8; the one-of-eight selection is made by the three LSB's of the counter. Every eight clock cycles the data in octal flip-flops 1.1 through 1.8 is dumped into octal flip-flops 2.1 through 2.8. At this time sixty-four bits of data are transmitted over ribbon cable to PCB2. The ribbon cable is made of 128 wires. Every other wire in the cable is grounded so as to help eliminate the crosstalk which could potentially confuse results [Blo88]. PCB2 contains only resistors since its job is simply to parallel-terminate the sixty-four received signals.

The noise measured in the experiments was the difference in potential between the ground plane of PCB1 and that of PCB2. Ideally, both ground planes would be at zero volts resulting in no potential difference between the two. This voltage drop is analogous to the drop across the pin and bonding wire of a chip. In both cases the difference in potential occurs due to the

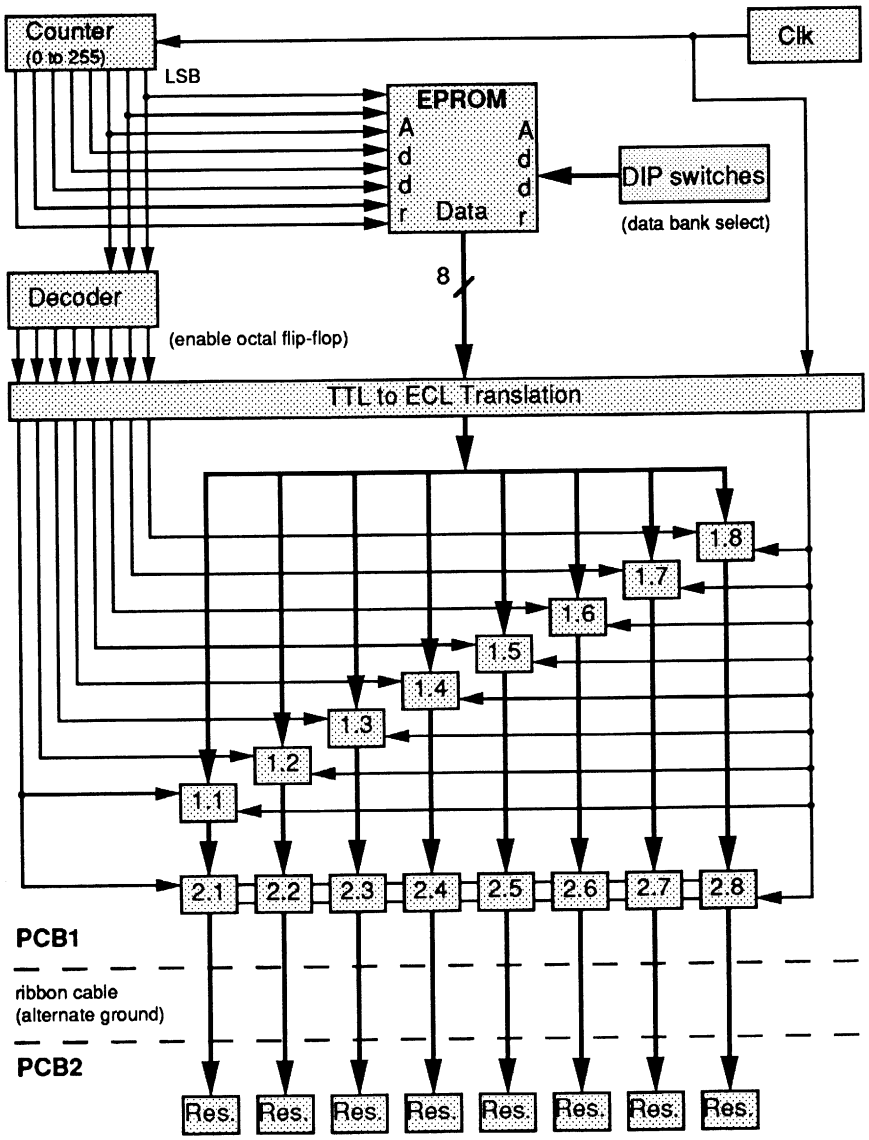

Figure 3-12: A block diagram of the ration-coding demonstration board.

parasitic inductance of wire.

Current flows from PCB1 to PCB2 over the signal wires of the ribbon cable. Current must flow in a continuous loop. Therefore, the current returns to PCB1 over the ground wires of the ribbon cable. A large change in current over these ground wires will cause a voltage drop due to $V = L(dI/dt)$. A large change in current results when successive output words have significantly different number of ones. Therefore, keeping the weight of successive output words relatively constant through ration coding should minimize the current changes. This, in turn, should minimize the potential difference between the ground planes (i.e., the noise measured in the experiment). Therefore, if noise reduction can be achieved through ration coding for my demonstration board, then it could also be achieved for integrated circuits.

Prior to making any noise measurements, the EPROM must be programmed. Three EPROM data banks (of 256 bytes each) were programmed for each experiment: one with uncoded data, one with ration-coded data, and one with differential-signalling data. The programming was done so that in all three cases the output of PCB1 toggled between two 64-bit words. This simplified result interpretation since one could examine the noise generated by a known change in weight. Of course, in each case the same amount of information was transmitted from PCB1 to PCB2.

To make matters clear, let's look closely at the specific experiment where noise generated by worst case transitions was investigated. Figure 3-13 shows the pairs of output words PCB1 toggled between during each of the three parts of the experiment. Figures 3-14 through 3-16 include photographs of the resulting noise measured with an oscilloscope.

In all experiments the first EPROM bank was programmed with uncoded data. For the specific, worst-case experiment the bank was programmed so that the output of PCB1 would toggle between a 64-bit word with weight zero and one with weight thirty-two. (Only thirty-two bits of information could be transmitted if a comparison with differential signalling was to be made. Recall, differential signalling requires two signals for each bit of information.) The noise measured during a transition from a word with weight zero to one with weight thirty-two is shown in Figure 3-14.

The second EPROM bank was always programmed with the data resulting from a pseudo-implementation of ration coding. (I say *pseudo-implementation* since the coding was done on

	Output of PCB1				
uncoded	0000	0000	0000	0000	toggle
	0000	0000	FFFF	FFFF	
ration coding	0000	0001^	FFFF	0000	toggle
	0000	0001^	0000	FFFF	
differential signalling^^	5555	5555	5555	5555	toggle
	AAAA	AAAA	AAAA	AAAA	

^ one signal (active high) to inform PCB2 of the first-half inversion

^^ 5 = 0101, A = 1010

Figure 3-13: The worst-case transition experiment requires PCB1 to toggle between three different pairs of output words (shown in hex).

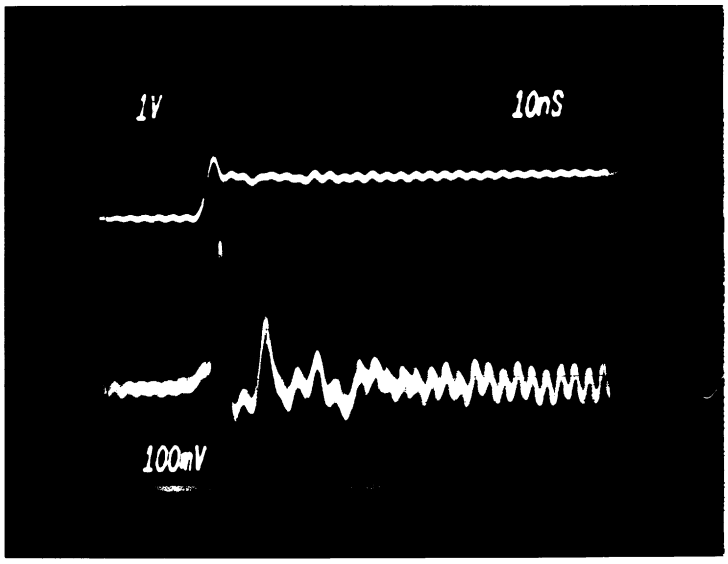

Figure 3-14: Noise generated during a transition from a word with weight zero to a word with weight thirty-two. The top oscilloscope trace shows one of the thirty-two output signals making a transition. The bottom trace shows the potential difference between the ground plane of PCB1 and that of PCB2.

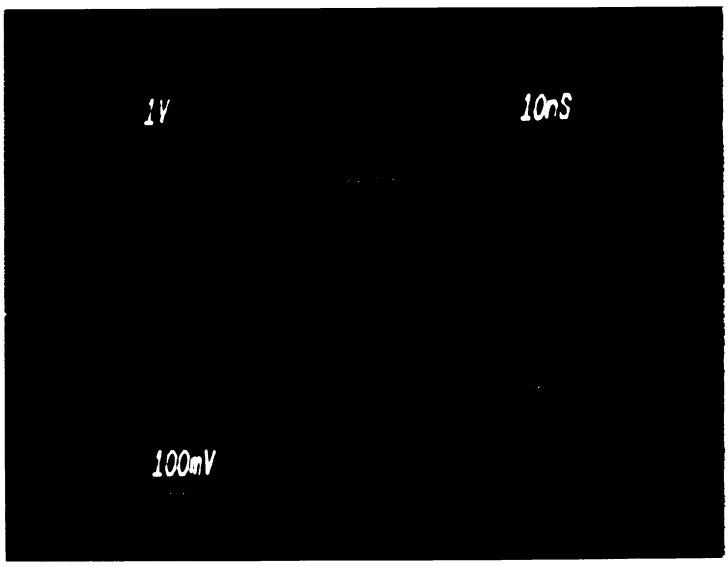

Figure 3-16: Noise generated when implementing differential signalling. The top oscilloscope trace shows one of the sixty-four output signals making a transition. The bottom trace shows the potential difference between the ground plane of PCB1 and that of PCB2.

of recursion and table look-up with various segment sizes). The resulting noise measurements uncovered no surprises. That is, in all experiments the noise measured was proportional to the weight variation of output words. Therefore, in any application the most extensive use of ration coding affordable is advised. Interested readers may consult the appendix for the results of the other experiments.

Chapter 4

Reducing Transmitted Noise Using Starvation Coding

This chapter presents a coding strategy for reducing transmitted noise in circuits exhibiting negligible steady-state current in their interconnection network. The conclusions reached apply to all such circuits. Because of its popularity, this document will often refer to a particular technology, CMOS chips communicating with unterminated transmission lines, though the results are more generally applicable.

4.1 An Example of Transmitted Noise in a CMOS Circuit

On a write operation the Intel i860 microprocessor can drive sixty-four data lines and thirty-two address lines simultaneously [Int89]. Because as many as ninety-six output signals can change at once, precautions must be taken to prevent the i860 from generating an intolerable amount of transmitted noise. First, to better understand the problem, let's look closely at a write operation as if no precautions had been taken. Next, we will examine Intel's solution to the problem. Finally, we will evaluate a coding solution.

Consider a worst-case write operation; ninety-five output signals are high and are changing to low (H->L) and one output signal is low and is staying low (L->L). The output circuitry of the i860 is shown schematically in Figure 4-1. For now, we will assume the i860 has one V_{CC} (or $5V$) pin and one V_{SS} (or GND) pin, each exhibiting some parasitic inductance (labeled

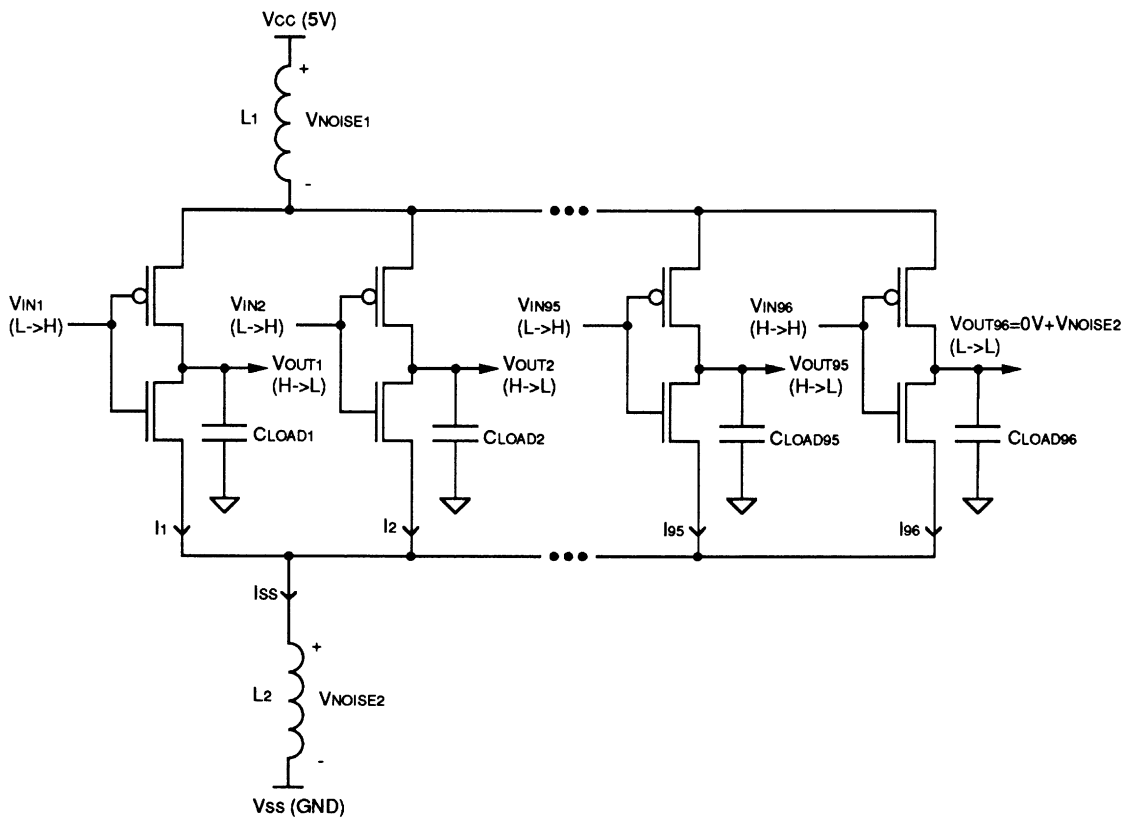

Figure 4-1: 96-bit parallel output circuit of the i860 driving unterminated transmission lines. Figure assumes one ground pin (V_{SS}) and one power pin(V_{CC}).

L_1 and L_2, respectively). Furthermore, we will assume each output is loaded with $50pF$ of interconnect and input capacitance (C_{LOAD}). Intel specifies the slew time of the i860 outputs (i.e., transition time between $V_{OUT} = 0.8V$ and $V_{OUT} = 2.0V$) as a function of the load capacitance. It is not unreasonable to want the slew time (T_S) to be 1.0ns for a capacitive load of $50pF$ [TK85]. (This is not the slew time Intel specifies. Slowing this edge time is one of the ways Intel reduces noise. We will proceed here with a desirable and achievable slew time of $1.0ns$.) These numbers mean that each output can discharge its load capacitance from $2.0V$ to $0.8V$ in $1.0ns$. The resulting current through each of the ninety-five NFETs (labeled I_n for $n = 1$ to 95) can be approximated as follows:

$$I_n = C_{LOAD}(\frac{dV_{OUTn}}{dt}) \tag{4.1}$$

$$I_n \approx C_{LOAD}(\frac{\Delta V_{OUTn}}{T_S}) \tag{4.2}$$

$$I_n \approx 50pF(\frac{2.0V - 0.8V}{1.0ns}) = 60.0mA \tag{4.3}$$

The sole V_{SS} pin sinks all ninety-five output currents. That is,

$$I_{SS} = I_1 + I_2 + I_3 + ... + I_{95} = 95(60.0mA) = 5.70A \tag{4.4}$$

The changing current on the V_{SS} pin causes a voltage drop due to the parasitic inductance L_2. Assume the pin and bonding wire measure $1.0cm$. As before, this yields an inductance of $10.0nH$. Using these numbers, the voltage drop across L_2 can be approximated as follows:

$$V_{NOISE2} = L_2(\frac{dI_{SS}}{dt}) \tag{4.5}$$

$$V_{NOISE2} \approx L_2(\frac{I_{SS}}{T_S}) \tag{4.6}$$

$$V_{NOISE2} \approx 10.0nH(\frac{5.70A}{1.0ns}) = 57.0V \tag{4.7}$$

According to this argument, the voltage at the *static* output (V_{OUT96} of Figure 4-1) should get as high as $57.0V$. Of course, this is absurd since none of the capacitive loads would be discharging if this were the case. Nonetheless, the noise is obviously large enough to warrant attention.

Intel takes this problem seriously. The i860 has twenty-four V_{CC} pins and twenty-four V_{SS} pins [Int89]. Depending on how many of these pins are used by the output drivers, this should reduce the transmitted noise by as much as a factor of twenty-four. Notice, however, in the worst case this is not enough.

The output bandwidth of the i860 suffers severely as a result of Intel's other attempt to reduce transmitted noise; the output drivers are intentionally designed to have slow transition times. Intel specifies the output slew time of the i860 to be about $6.0ns$ with a $50pF$ load capacitance [Int89]. The output drivers are substantially slower than they could be; above we assumed reasonably a slew time of $1.0ns$. This reduces noise by as much as a factor of thirty-six (notice T_S enters into the above analysis twice) but costs a factor of six in output bandwidth.

Unfortunately, there is no coding scheme which alone can solve this noise problem. For circuits in this class (i.e., circuits exhibiting negligible steady-state current) it is impractical to use coding to reduce noise by a factor of four or more. This fact will be demonstrated shortly. However, it still makes sense to consider coding techniques as a partial solution to problems like above or as a total solution to less severe problems.

4.2 Starvation Coding as a Solution

A type of noise-reduction coding, termed *starvation* coding earlier, can reduce noise in circuits like the one described in the previous section. From equation 4.5 it is apparent that no noise will be generated if the current on the V_{SS} pin does not change. Unfortunately, this requires the output signals not to change. Of course, static signals convey no information. Therefore, for circuits exhibiting negligible steady-state current in their interconnection network, only limited noise reduction can be achieved through coding.

Starvation coding cannot eliminate noise, but it can reduce it. Referring to equation 4.5 again, less noise will be generated if the current on the V_{SS} pin changes less drastically. This

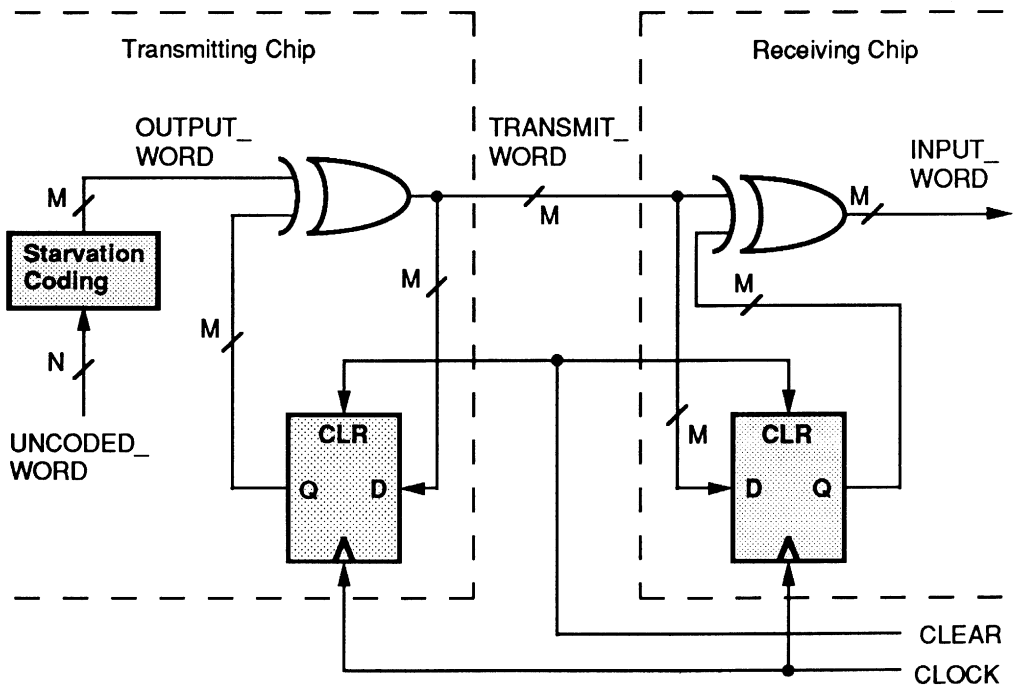

Figure 4-2: A circuit for implementing transition signalling.

requires fewer transitions by the output signals. Therefore, a coding strategy can reduce the transmitted noise generated by a set of outputs if it can reduce the maximum number of transitions experienced by the set while preserving its information-carrying capacity.

Reducing transitions is a complex problem; it requires memory. Before transmitting a new set of outputs, the set must be compared with the previous set transmitted. The number of transitions must be counted prior to taking any coding actions (e.g., inverting some of the bits). It would be nice to avoid counting since it is slow. This can be achieved through a technique called *transition signalling.*

A circuit for implementing transition signalling is shown schematically in Figure 4-2. This circuit conceptually simplifies the problem of reducing transitions to the problem of reducing 1's. Notice a bit of TRANSMIT_WORD changes state only if a 1 appears in the corresponding bit position of OUTPUT_WORD. Since transitions are encoded as 1's, starvation coding techniques can minimize the 1's in OUTPUT_WORD and, hence, reduce transmitted noise. (For this scheme to work the transmitting and receiving registers must always contain the same

word. Equality is established prior to transmitting any data by CLEARing the registers. I encourage readers to convince themselves that OUTPUT_WORD and INPUT_WORD are always the same.)

Power conservation is a desirable side effect of starvation coding. Since fewer output transitions occur, the capacitive loads need to be charged less frequently. This, of course, results in a power savings. Indulgence coding (i.e., coding so as to maximize the number of 1's in a word) is the the logical inverse of starvation coding; It can reduce output transitions and, hence, transmitted noise as effectively as starvation coding.

4.3 Theoretic Limits of Starvation Coding

As mentioned earlier, starvation coding cannot eliminate noise. In fact, reducing transmitted noise by a factor of four or more appears impractical. A rigorous mathematical proof of this statement is complex. I will demonstrate this fact with two examples. First, I will examine the maximum noise reduction achievable. Then, I will examine noise reduction by a factor of four.

Since static signals convey no information, transmitted noise is at a minimum when a chip's outputs experience one transition per cycle. This can be achieved with transition signalling and a starvation-coding scheme which limits words to a weight of one. Refering to Figure 4-2 again, this corresponds to having a starvation-coding box with 2^N outputs; there must be one output for each possible N-bit word. Of course, it is impractical to exponentially increase the number of outputs to reduce transmitted noise.

Unfortunately, reducing transmitted noise by as little as a factor of four appears to be impractical as well. Starving an 8-bit output word to weight two reduces transmitted noise by a factor of four. Unfortunately, this requires increasing the output word to twenty-three bits. This is demonstrated by the following analysis.

Starving an N-bit output word to weight W requires expanding the word to M bits. In order for M-bit words of maximum weight W to have the information capacity to replace N-bit words of unrestricted weight the following must hold:

$$C(M,0) + C(M,1) + C(M,2) + ... + C(M,W) \geq 2^N \tag{4.8}$$

Setting $N = 8$ and $W = 2$ and solving for M yields,

$$C(M,0) + C(M,1) + C(M,2) \geq 2^8 = 256 \tag{4.9}$$

$$\frac{M!}{0!(M-0)!} + \frac{M!}{1!(M-1)!} + \frac{M!}{2!(M-2)!} \geq 256 \tag{4.10}$$

$$1 + M + (1/2)M(M-1) \geq 256 \tag{4.11}$$

$$(1/2)M^2 + (1/2)M - 255 \geq 0 \tag{4.12}$$

The quadratic formula gives the following conditions for M:

$$(M > 22.1) \vee (M < -23.1) \tag{4.13}$$

Of course, M must be positive. Therefore, if the output weight is restricted to two, then at least twenty-three signals are required to transmit eight bits of information.

This is no surprise when one considers the information capacity of a 23-bit word. Figure 4-3 plots the number of different 23-bit words as a function of weight (i.e., $C(23,W)$). Notice that much fewer low-weight words exist than average-weight words. Figure 4-4 contains the same plot expanded in the area of interest. I have shaded the region corresponding to words of maximum weight two. Starvation coding uses less than 0.004% of the total number of 23-bit words.

Table 4.1 contains the output signals required (M) to transmit various amounts of information (N, in bits) when reducing transmitted noise by a factor of four. The required number of output signals for large N appears tolerable. (For $N = 32$ it is less than twice N.) We will see in the next section, however, that no implementation technology we are aware of exists to make these cases practical. (Of course, one may be invented.)

For circuits using series-terminated or un-terminated transmission lines, coding to reduce transmitted noise by a factor of four or more is fundamentally inefficient. This should be

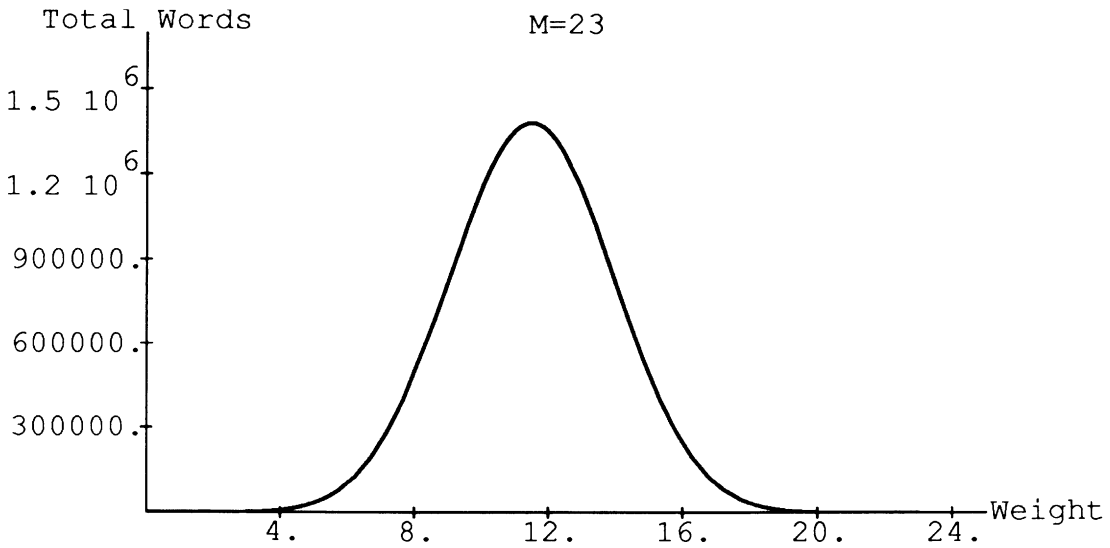

Figure 4-3: The number of different 23-bit words plotted as a function of weight, $C(23, W)$.

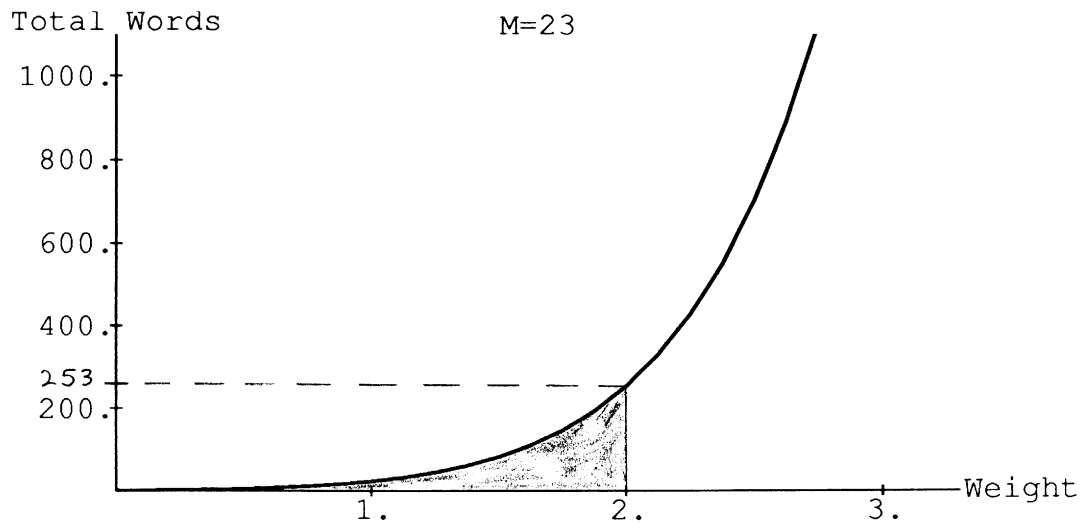

Figure 4-4: A plot of $C(23, W)$ expanded to show detail near $W = 2$.

Table 4.1: Output signals required (M) for various amounts of information (N, in bits) when reducing transmitted noise by a factor of four.

N	W_{MAX}	M
4	1	15
8	2	23
16	4	36
32	8	63

obvious from Figures 4-3 and 4-4; simply not enough low-weight words exist. However, this should not discourage the reader from using coding in general. The next section contains examples of starvation coding practically implemented to reduce transmitted noise by less than a factor of four.

4.4 Implementations of Starvation Coding

As with ration coding, starvation coding can be implemented in many ways. The implementation technologies fall in two categories: algorithmic implementations and table look-up. The remainder of this chapter contains examples of these techniques. The section on algorithmic implementations is, unfortunately, quite incomplete.

4.4.1 Algorithmic Implementations

Starvation coding can achieve a 50% noise reduction with a simple algorithm. Figure 4-5 shows schematically a circuit implementing this algorithm. This circuit reduces the worst-case number of transitions by 50%. It does not require the transition-signalling circuit of Figure 4-2. The scheme is very straightforward:

- Store the last transmitted word in a register.
- Compare this with the next word to be transmitted.
- If the words differ in more than half the bit positions, bit-wise invert the entire word prior to transmitting it.

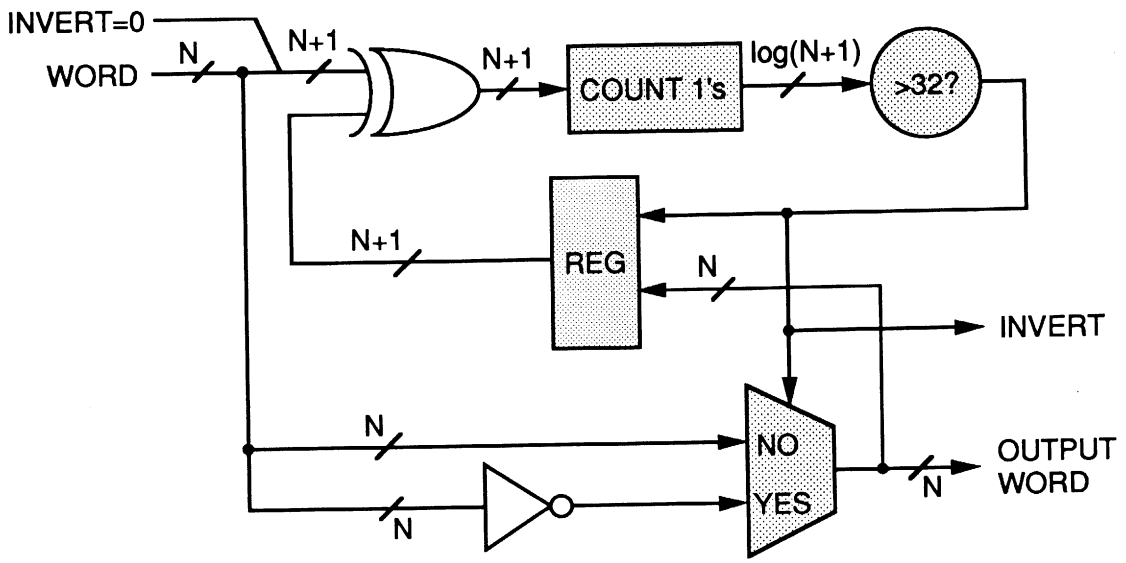

Figure 4-5: An algorithmic implementation of starvation coding which reduces transmitted noise by 50%.

Table 4.2: The transitions saved on the average using the starvation-coding algorithm with various segment sizes.

N/K	T_{AVG}	$T_{AVG,STARVE}$	*Savings*
4	2	1.56	21.9%
8	4	3.27	18.3%
16	8	6.83	14.6%
32	16	14.2	11.3%

- Transmit an additional signal (INVERT) to inform the receiver whether or not the inversion took place.

This scheme is attractive since it reduces the worst-case number of transitions by a factor of two with the addition of only one signal. Furthermore, the receiving chip requires only simple circuitry to interpret the transmitted word. Unfortunately, one cannot recurse on the idea to achieve greater reductions with the addition of more signals.

An additional disadvantage of this scheme is it requires counting and comparing; these operations are slow. Since in this application the operations need not be accurate- fast, approximate count-and-compare circuitry can be built to mitigate the disadvantage. This idea was discussed previously with reference to a ration-coding implementation. An analog count-and-compare circuit could be built in a style similar to that of the circuit shown in Figure 3-9.

An issue to consider with this scheme is whether or not to segment the output word and compare cycle-to-cycle words in sections rather than in entirety. For an N-bit output bus, building K circuits like the one shown in Figure 4-5 where each processes a single (N/K)-bit section has advantages. This would require more chip area and the transmission of more signals (K INVERT signals, one for each section) but would simplify the necessary count-and-compare circuitry since only N/K bits would need to be counted and compared. Furthermore, this would reduce the *average* number of transitions per cycle and, hence, the power consumption.

Table 4.2 contains data describing the transitions saved on the average using the starvation-coding algorithm with various segment sizes. An (N/K)-bit uncoded section experiences, on the average, $T_{AVG} = N/(2K)$ transitions per cycle. The starvation-coding algorithm reduces the average number of transitions to $T_{AVG,STARVE}$. Notice *Savings* increases as the section

size decreases. This translates directly into a power savings. Since it is common for half or more of the dissipated power in a high speed CMOS system to be devoted to driving a highly capacitive output bus [TK85] this savings is significant. However, since the worst case remains at $N/(2K)$ transitions, segmenting the output does not further alleviate the transmitted noise problem.

4.4.2 Table Look-Up

Starvation coding can be implemented with look-up tables. We can choose positive integers P, Q, and R such that the following holds:

$$C(Q,0) + C(Q,1) + C(Q,2) + ... + C(Q,R) \geq 2^P \tag{4.14}$$

This enables us to encode a P-bit word of unrestricted weight as a Q-bit word with maximum weight R. This, of course, is starvation coding. We are starving a P-bit word to weight R.

In cases where the output word is greater than about ten, it will be necessary to segment it into manageably sized sections. Using this idea we can segment our N-bit output word into (N/P) sections. Each P-bit section can be starved to Q bits of maximum weight R by its own look-up table. In total N/P look-up tables can starve the N-bit output word to M bits with maximum weight W_{MAX}, where $M = (N/P)Q$ and $W_{MAX} = (N/P)R$.

Choosing the values of P, Q, and R involves many trade-offs. Table 4.3 shows possible combinations of these parameters for $N = 64$ as well as their influence on various design characteristics including maximum output weight (W_{MAX}), total pins (M), table area for the transmitting chip (A_T), and table area for the receiving chip (A_R). Using the transition-signalling circuit of Figure 4-2, a look-up scheme of Table 4.2 will reduce the worst-case noise by a factor of $64/W_{MAX}$. A_T and A_R were calculated as discussed in section 3.4.1 with the following formulas:

$$A_T = (2^P * Q)(N/P) \tag{4.15}$$

$$A_R = (2^Q * P)(N/P) \tag{4.16}$$

Table 4.3: Combinations of P, Q, and R with their influence on various design characteristics.

P	Q	N/P	R	W_{MAX}	M	A_T	A_R	*Savings*
10	19	6.4	3	22	118	116,736	3.15×10^7	44.1%
10	13	6.4	4	28	82	79,872	491,520	29.5%
10	11	6.4	5	34	70	67,684	122,880	17.1%
9	32	7.11	2	15	225	114,688	2.71×10^{11}	57.0%
9	15	7.11	3	22	106	53,760	2.06×10^6	39.3%
9	11	7.11	4	29	78	39,424	129,024	24.7%
8	23	8	2	16	184	47,104	5.37×10^8	52.4%
8	12	8	3	24	96	24,576	262,144	34.1%
8	9	8	4	32	72	18,432	32,768	18.3%
7	16	9.14	2	19	145	18,432	4.13×10^6	46.9%
7	9	9.14	3	28	82	10,368	32,256	27.0%
6	11	10.67	2	22	121	7744	135,168	40.1%
6	7	10.67	3	33	77	4928	8448	19.3%
5	8	12.8	2	26	104	3328	16,640	32.5%
4	15	16	1	16	240	3840	2.10×10^6	53.1%
4	5	16	2	32	80	1280	2048	21.9%
3	7	21.33	1	22	148	1176	8064	41.7%
2	3	32	1	32	96	384	512	25.0%

Table 4.3 contains combinations of P, Q, and R where P is no larger than ten and transmitted noise is reduced by about a factor of two or more. Some combinations are obviously impractical because they require too many output signals or too much chip area. The last column of the table indicates the power savings I expect from a particular implementation. The entries are actually equal to the savings in average number of output transitions, but, as argued previously, the two are proportional. *Savings* was calculated as follows:

$$Savings = 1 - \frac{W_{AVG}}{(P/2)} \tag{4.17}$$

W_{AVG} is the average weight of the section and was calculated as follows:

$$W_{AVG} = \frac{(0)C(Q,0) + (1)C(Q,1) + (2)C(Q,2) + ... + (R-1)C(Q,R-1)}{2^P} + \frac{(R)(2^P - [C(Q,O) + C(Q,1) + C(Q,2) + ... + C(Q,R-1)])}{2^P} \tag{4.18}$$

The average-weight calculation is complicated since often not all of the maximum-weight words need to be used.

Chapter 5

Conclusions and Future Investigations

This document developed a weight-based, noise-reduction coding scheme termed ration coding which reduces transmitted noise in circuits exhibiting steady-state current in their interchip signalling networks such as ECL circuits with parallel-terminated transmission lines. Ration coding achieves the noise reduction of differential signalling at smaller costs. Differential signalling requires doubling output pins and wires. This is a big price to pay especially in state-of-the-art digital systems; accommodating the wires is a primary concern in large, multiprocessor computers.

Ration coding permits trading output wires for design time, execution time, and chip area. Design time is a fixed cost associated with all new ideas. In a pipeline environment the cost of adding an extra stage or two to hide the execution time should not be too severe. It would often make sense to trade chip area for output pins even if extra chip area was not readily available. Therefore, in many cases replacing differential signalling with an implementation of ration coding provides an increase in benefits.

This document also developed a weight-based, noise-reduction coding scheme termed starvation coding which reduces transmitted noise in digital circuits exhibiting negligible steady-state current in their interchip signalling networks such as CMOS circuits with un-terminated transmission lines. Starvation coding can reduce transmitted noise by as much as a factor of

three. A fundamental limitation makes greater reductions impractical. Nonetheless, starvation coding is a valuable addition to the noise-reduction arsenal.

The development of new ration-coding and starvation-coding techniques will greatly enhance the use of coding to reduce transmitted noise. Unfortunately, no coding research effort concerning the regulation of binary weight has produced a scheme applicable here. At this time the most applicable strategy appears to be table look-up. Look-up tables require small design and execution times but significant chip area.

Those interested in furthering these investigations should consider two interesting possibilities. The codes described here have advantages not only at the chip boundary but also at the card/backpanel boundary. In particular, if a memory system is built to handle coded data, it might be easier to store data in coded rather than decoded form. Combining ration or starvation coding with coding for error detection/correction might make an attractive code for storing in a memory system. The combined codes, if they can be developed, might be able to combine both features into a smaller codeword than would be required by simply summing the extra code bits used [TK85].

The second idea is somewhat of a fantasy. Since eight bits of data can be encoded as eleven bits with weight four, it is conceivable to build an entire ECL computer with virtually no transmitted noise. In this computer each byte would be expanded to eleven bits with weight four. If it were possible to build modules that operated on 11-bit words of weight four and produced 11-bit words of weight four, then an entire ECL computer could be built where the number of 1's in the data paths was always constant. This would lead to virtually no transmitted noise. This idea may seem a little far-fetched since it would be difficult to build a module that, say, added two 11-bit words of weight four and produced an 11-bit result of weight four, but it is conceivable nonetheless.

Bibliography

[Bip86] Bipolar Integrated Technology, Inc., Beaverton, OR. *BIT Product Summary*, 8 1986.

[Blo88] William R. Blood. *MECL System Design Handbook.* Motorola Semiconductor Products, Inc., 1988.

[Dil88] Thomas E. Dillinger. *VLSI Engineering.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1988.

[Dra67] Alvin W. Drake. *Fundamentals of Applied Probability Theory.* McGraw-Hill Book Co., New York, NY, 1967.

[Ham80] Richard W. Hamming. *Coding and Information Theory.* Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1980.

[Hum85] Pierre A. Humblet. *Binary Codewords with Few Ones.* Laboratory for Information and Decision Systems, MIT, 6 1985.

[Int89] Intel Corp. *i860 64-Bit Microprocessor*, 4 1989.

[Kol81] James S. Kolodzey. Cray-1 computer technology. *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, CHMT-4(2):181–186, 6 1981.

[PB88] Jerry Prioste and Alan Bass. *MECL MCA II Macrocell Array Design Guidelines.* Motorola Semiconductor Products, Inc., 1 1988.

[SS88] Campbell L. Searle and Stephen D. Senturia. *6.002 Notes, Preliminary Edition January, 1988.* MIT Introductory Electronics Course, 1 1988.

[TK85] Jr. Thomas F. Knight. *Power and Noise Reduction in VLSI Circuits Using Coding Techniques.* MIT AI Lab, 1985.

Appendix A

More Demonstrations of Ration Coding Reducing Noise

The appendix contains the experimental results promised in section 3.6. Each picture shows the noise generated when transmiting thirty-two bits of information. For the pictures containing two oscilloscope traces, the top trace shows a representative signal being transmitted from PCB1 and the bottom trace shows the noise. For the pictures containing three oscilloscope traces, the top trace shows a representative signal being transmitted from PCB1, the bottom trace shows a representative signal being received by PCB2, and the middle trace shows the noise.

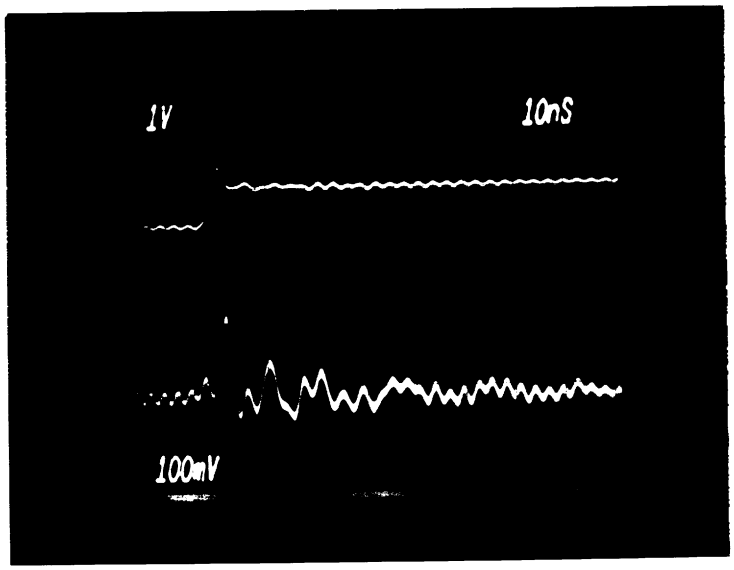

Figure A-1: Noise generated by a transition from an output word with weight seven to a word with weight twenty-five.

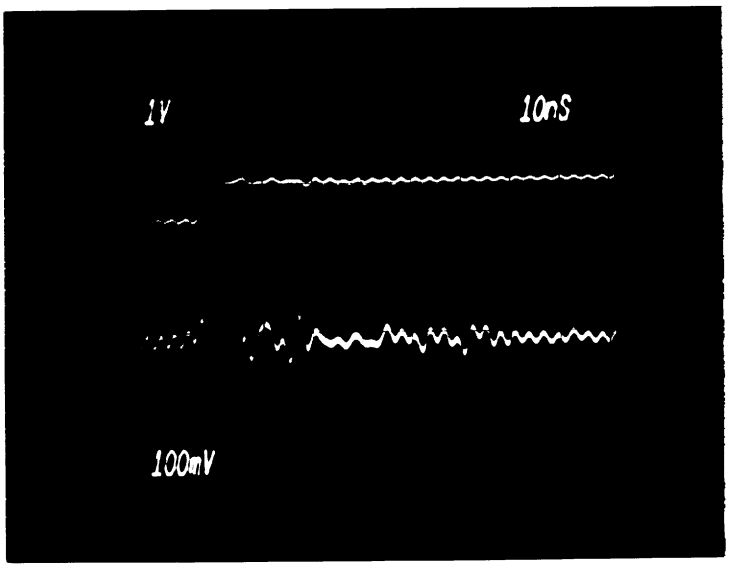

Figure A-2: A pseudo-implementation of the ration-coding algorithm with one step of recursion reducing the noise from Figure A-1.

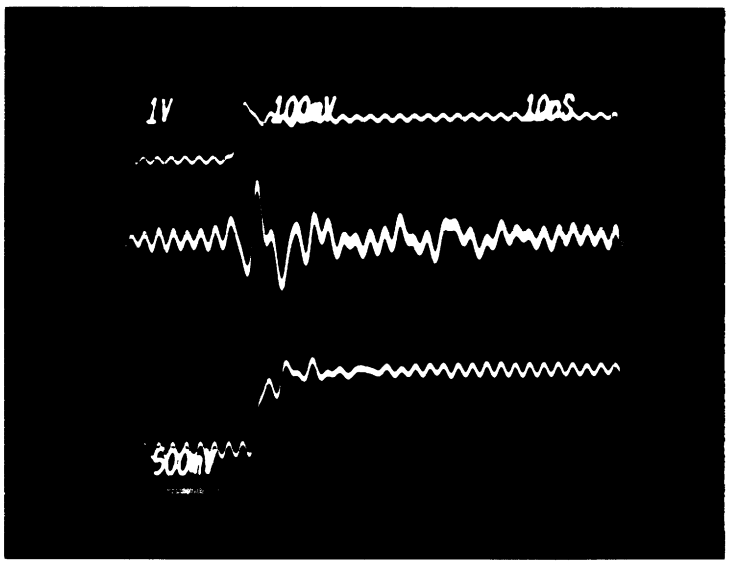

Figure A-5: A pseudo-implementation of ration-coding with two steps of recursion reducing the worst-case noise from Figure A-4.

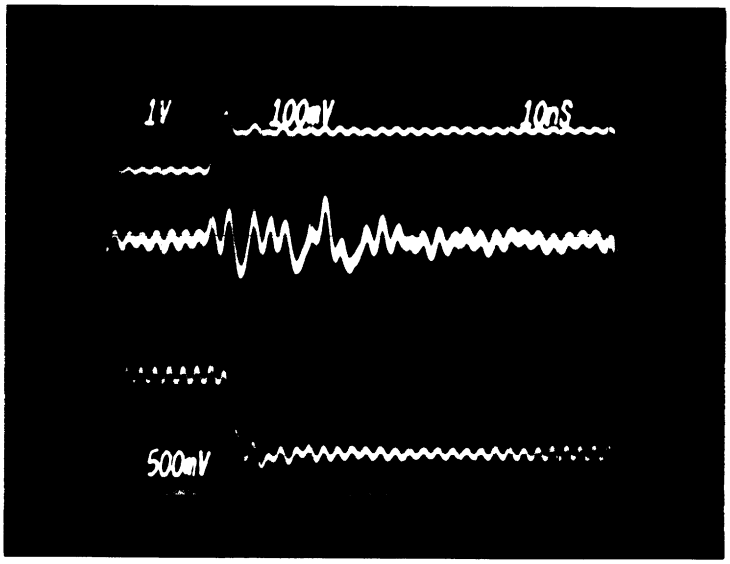

Figure A-6: Worst-case noise generation with two steps of recursion. This occurs when an output word with weight twenty-one is transmitted after a word with weight thirteen.

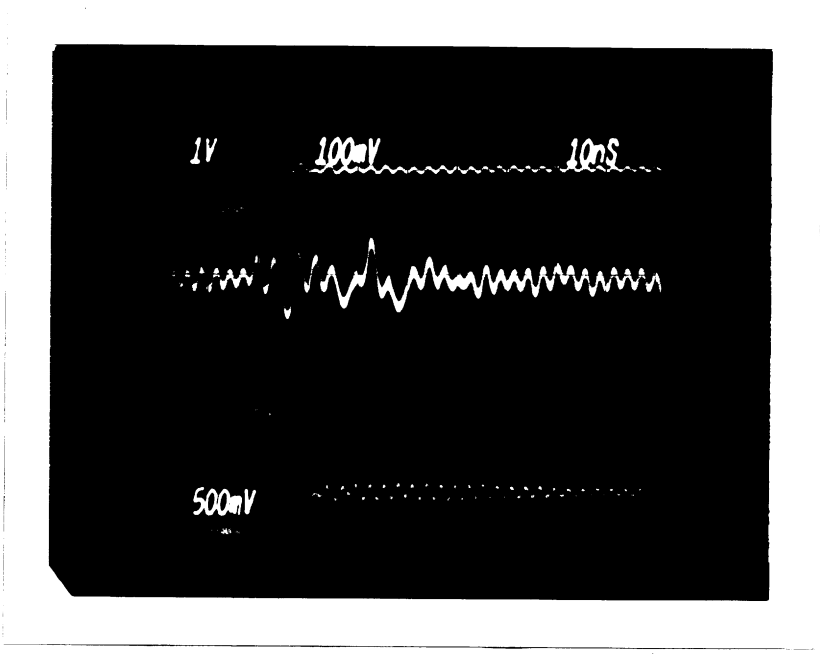

Figure A-7: Worst-case noise generation with three steps of recursion. This occurs when an output word with weight twenty-one is transmitted after a word with weight seventeen.

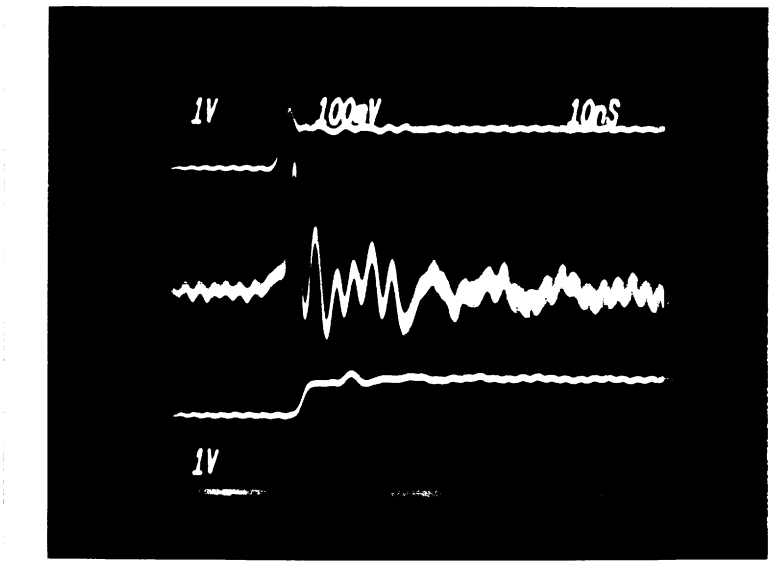

Figure A-8: Noise generated by a transition from an output word with weight zero to a word with weight thirty-two.

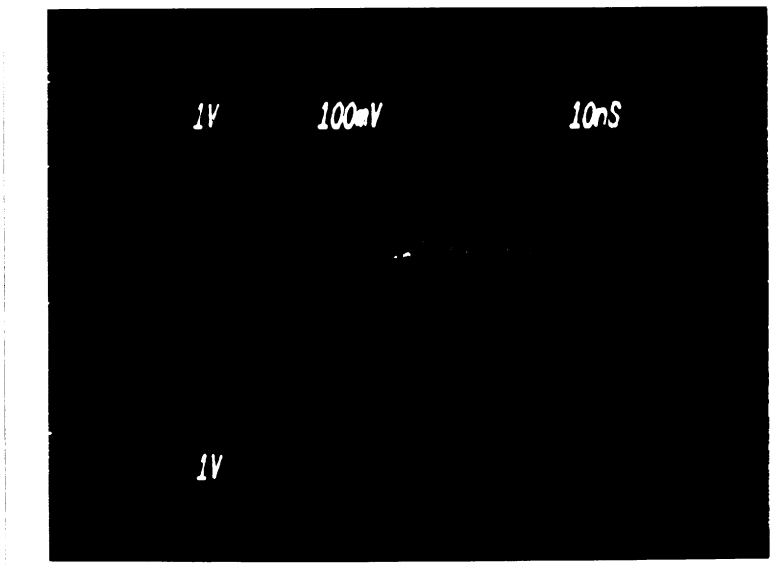

Figure A-9: Table look-up implementation reducing worst-case noise from Figure A-8. Here, 8-bit words are encoded as eleven bits with weight four.

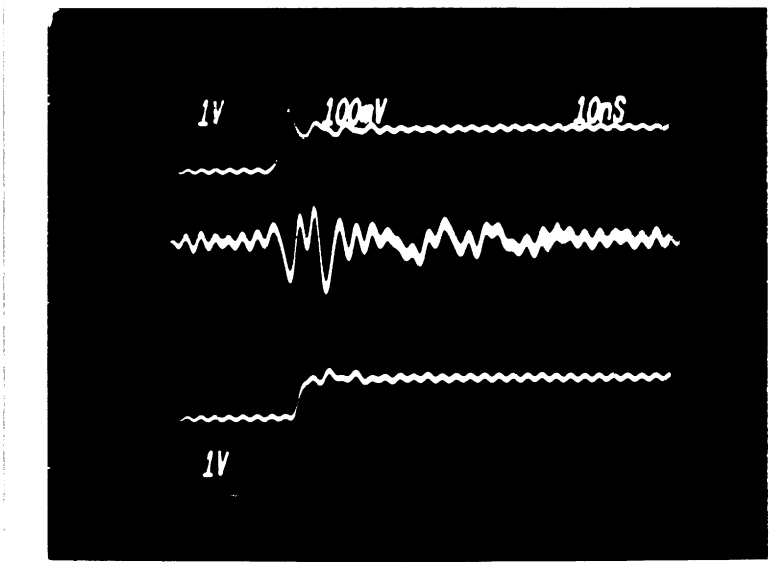

Figure A-10: Table look-up implementation reducing worst-case noise from Figure A-8. Here, 4-bit words are encoded as six bits with weight three.

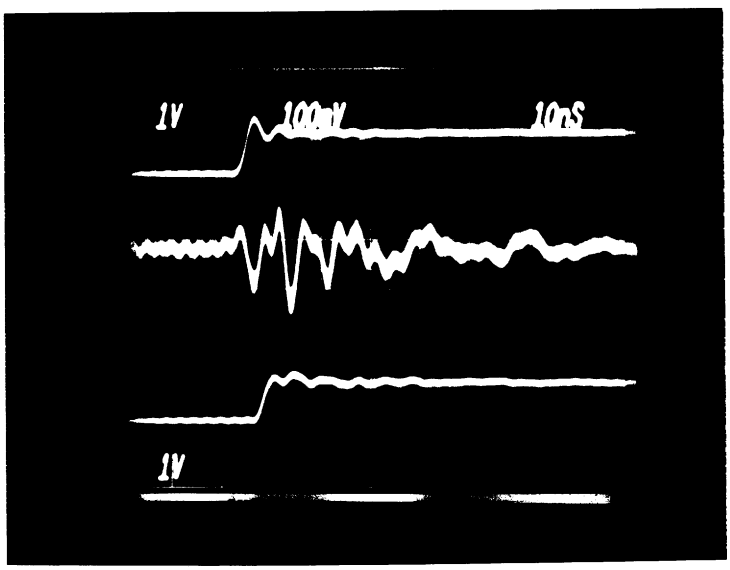

Figure A-11: Table look-up implementation reducing worst-case noise from Figure A-8. Here, 2-bit words are encoded as four bits with weight one.

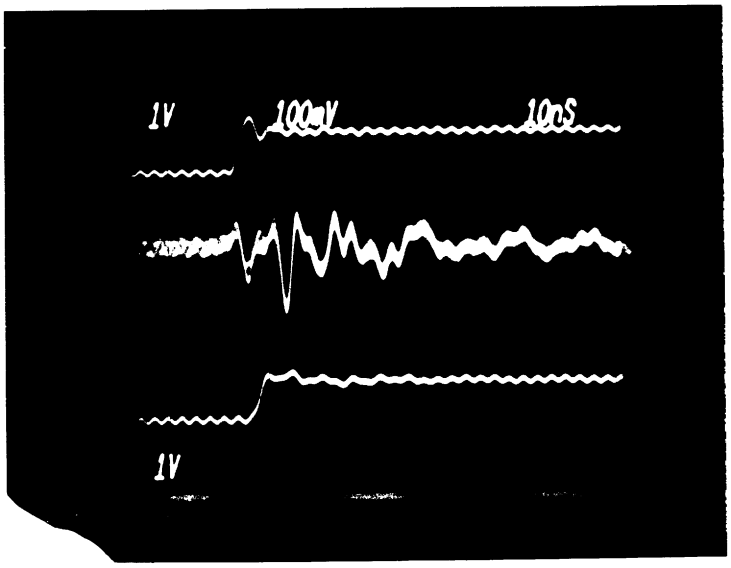

Figure A-12: Table look-up implementation reducing worst-case noise from Figure A-8. Here, 6-bit words are encoded as eight bits with weight four.

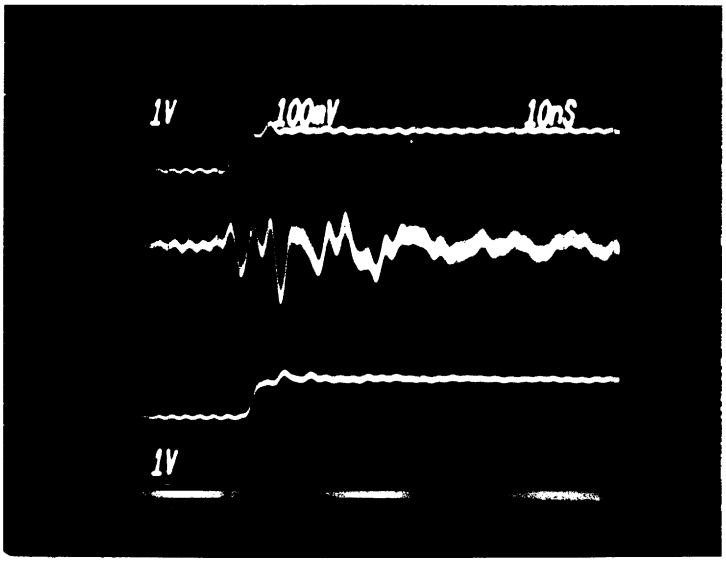

Figure A-13: Table look-up implementation reducing worst-case noise from Figure A-8. Here, 3-bit words are encoded as five bits with weight two.

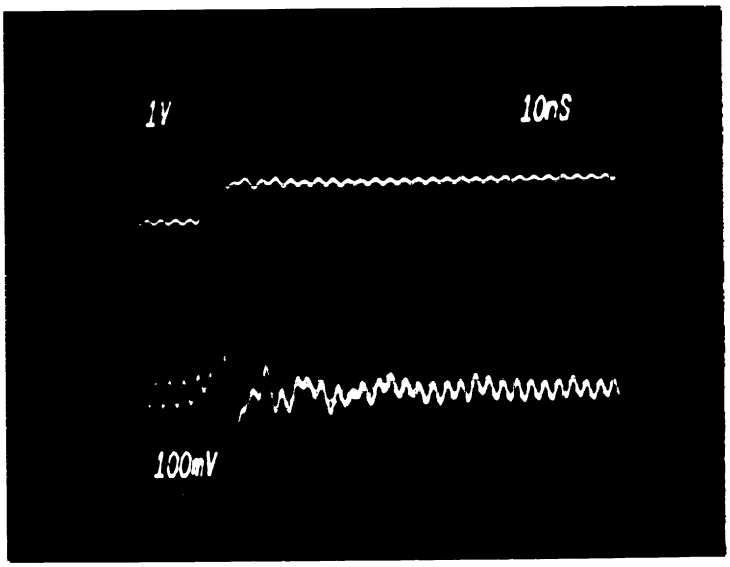

Figure A-14: Differential signalling reducing the worst-case noise from Figure A-8.

This blank page was inserted to preserve pagination.

CS-TR Scanning Project
Document Control Form

Date : 6/22/95

Report # AI-TR-1232

Each of the following should be identified by a checkmark:
Originating Department:

- [x] Artificial Intellegence Laboratory (AI)
- [ ] Laboratory for Computer Science (LCS)

Document Type:

- [x] Technical Report (TR)
- [ ] Technical Memo (TM)
- [ ] Other: _____

Document Information

Number of pages: 79 (87-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:

- [x] Single-sided or
- [ ] Double-sided

Intended to be printed as :

- [ ] Single-sided or
- [x] Double-sided

Print type:

- [ ] Typewriter
- [ ] Offset Press
- [x] Laser Print
- [ ] InkJet Printer
- [ ] Unknown
- [ ] Other: _____

Check each if included with document:

- [x] DOD Form (2)
- [ ] Funding Agent Form
- [x] Cover Page
- [x] Spine
- [ ] Printers Notes
- [ ] Photo negatives
- [ ] Other: _____

Page Data:

Blank Pages (by page number): PAGES FOLLOWING ii, iii, viii, ix.

Photographs/Tonal Material (by page number): 39-41, 61-67

Other (note description/page number):

Description :	Page Number:
(A) IMAGE MAP (1-12)	PAGES #'ED ii, UN#'ED BLANK, iii UN# BLANK, iv-viii, UN#'ED BLANK, ix, UN#'ED BLANK.
(13-79)	PAGES #'ED 1-67
(80-87)	SCANCONTROL, COVER, SPINE, DOD (2), TRGTS (3)
(B) CUT & PASTE FIGS ON PAGES 3-5, 8-9, 11-12, 15, 17-18, 20, 22, 24, 27, 30-31, 34-35, 37, 39-41, 43, 46, 49, 51 61-67	

Scanning Agent Signoff:

Date Received: 6/22/95 Date Scanned: 6/28/95 Date Returned: 6/29/95

Scanning Agent Signature: Michael W. Cook

Rev 9/94 DS/LCS Document Control Form cstrform.vsd

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AI-TR 1232	2. GOVT ACCESSION NO: AD-A228346
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): Noise Reduction Using Low Weight and Constant Weight Coding Techniques	5. TYPE OF REPORT & PERIOD COVERED: technical report
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Jeff Tabor	8. CONTRACT OR GRANT NUMBER(s): N00014-88-K-0825, N00014-85-K-0124
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA 02139	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: Advanced Research Projects Agency, 1400 Wilson Blvd., Arlington, VA 22209	12. REPORT DATE: June 1990
	13. NUMBER OF PAGES: 75
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Office of Naval Research, Information Systems, Arlington, VA 22217	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

Distribution is unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

None

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

noise
coding
output driver
constant weight codes
differential signalling
low weight codes

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Signalling off-chip requires significant current. As a result, a chip's power-supply current changes drastically during certain output-bus transitions. These current fluctuations cause a voltage drop between the chip and circuit board due to the parasitic inductance of the power-supply package leads. Digital designers often go to great lengths to reduce this *transmitted noise*. Cray, for instance, carefully balances output signals using a technique called differential

(continued on back)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20 continued:

signalling to guarantee a chip has constant output current. Transmitted-noise reduction costs Cray a factor of two in output pins and wires. Coding achieves similar results at smaller costs.

In a circuit using parallel-terminated transmission lines such as the Cray-1, a chip exhibits nearly constant output current throughout a single clock cycle. However, since current for an output 1 differs greatly from that for an output 0, output current can change drastically at the end of a cycle generating an intolerable amount of transmitted noise. Hence, a coding scheme which maintains a constant number of output 1's will reduce transmitted noise as successfully as differential signalling. Experimental results substantiate this.

Information capacity of a word is greatest when the word has half 1's. Coding requires fewer output signals than differential signalling. For example, an 8-bit byte can be encoded as eleven bits with four 1's at an output savings of 30%. Of course, fewer outputs do not come for free; coding costs time and chip area. Coding achieves a smaller percentage output expansion for words longer than eight bits but requires a more costly coding scheme.

In a circuit using series-terminated or un-terminated transmission lines, output current flows when a signal changes state to charge/discharge the capacitive load. Hence, reducing transmitted noise requires minimizing output transitions. Representing transitions as 1's, a coding strategy which minimizes 1's will reduce transmitted noise. Unfortunately, words with few 1's have little information capacity. Coding in this case requires an exponential increase in output signals. Hence, only a 2× to 3× noise reduction is practical.

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**

Scanned

Date: 6/28/1995

M.I.T. Libraries
Document Services