Multi -Logic Level DRAM:
A Differential Approach

Jeff Gilbert
Varghese George
EE241
March 1, 1996

1 Introduction

Improved system performance and portability of computers demand higher memory density. In order to store information more densely in a cellular array, one can either try to reduce the cell size or else store more information per cell. Although much research and development has gone into the former option, we are choosing to investigate the latter for our EE241 project. Thus we will be looking at how to store multiple values in a given RAM cell. For reasons of simplicity of cell design, we are focusing on DRAM. (multi-stable SRAM cells would be a project in themselves). This approach is equally valid for both discrete memory components and integrated on chip memories using standard CMOS processes.

DRAM cells are analog storage cells that are conventionally only used to store one of two values - "0" or "1". This simplifies and accelerates cell sensing and regeneration, but it is not an inherent limitation in the cell itself. By storing more than two different voltages in the cell, greater bit density can be achieved. Issues such as noise margin, charge leakage, and sensing time become all the more critical, however.

1.1 Problems with Multi-logic level DRAM

Many of the problems with multi-logic level DRAM stem from the fact that if N levels are being stored in a cell than the delta voltage available to distinguish each level from its neighbor is only 1/(N-1) as large as that available for a simple two level scheme. This reduces the noise margin and makes the V_T variations, and signal coupling critical factors for a robust design.

2 Background Material

The first attempt at multi-valued memory technology was the 4 valued ROM cell by Intel[1]. The different values were defined by transistors with three different widths. This kind of an approach was feasible because the nMOS design rules in existence had a metal pitch which was very large as compared with device sizes, i.e. device technology was more advanced than metal interconnect technology. Within a 7λ metal pitch it was easy to place 2λ, 3λ, and 5λ wide transistors[2].This is not the case any more.

In recent times multilevel storage has been mainly used only in non-volatile memory technology[3], for voice storage and playback, where moderate data corruption can be tolerated. 256 analog voltage levels are stored with a ~7.5mV resolution per level.

An experimental DRAM chip with 16-levels/cell[4] has been presented, but the read times are quite high with a random block selection time of 147us, and a sequential access time of 210ns. The multilevel storage limitation is given by the subhtreshold leakage current from the memory cell, which gives a false signal charge to the data line. To overcome the speed degradation due to the small signal variation on the data lines during read out, a two stage sense amplifier set-up is used. A charge transfer amplifier[5] is used amplify the differential signal voltage on the data line (~12mV) to 120mV which is fed to the inputs of a cross coupled sense amplifier. The level sense time is proportional to the number of levels, in this case it is sixteen times the worst case sensing time, i.e. when the differential signal is corresponds to one voltage step. This implementation also requires a fairly complicated reference signal generation scheme.

Conventional error correcting codes, and techniques can be employed for soft-errors-correction[6][7]. The effect of transition to lower voltages to improve the power consumption[8] has to be investigated.

3 Differential Coding

3.1 Motivation

Many of these problems can be remedied through the use of *differential coding*. Using differential coding, the signal is coded by the difference between two stored values. The simplest form of differential coding would be storing both the signal and its complement. In this way, all common-mode parasitics are avoided. This technique is in fact employed in some areas such as EEPROMs[9]. The overhead in area is approximately 30-50% using clever layout techniques.

3.2 Method

In order to combine the storage density advantages of multi-logic level with the robustness of differential coding, we must consider what it means to store multiple values in a relative sense. In a conventional multi-logic level scheme, each cell assumes one of N values for a bit density of $\log_2(N)$ bits per cell. Consider the following scheme: Instead of considering each cell in isolation, consider a group of N cells each which can assume one of N values. If we put the restriction that only one cell in each group can take on each of the N levels then all information is now contained in the ordering of the cells. For N cells and N logic levels, a total of N! combinations can be stored. This works out to a bit-density of $\log_2(N!)$ / N bits per cell. (See figure 1).

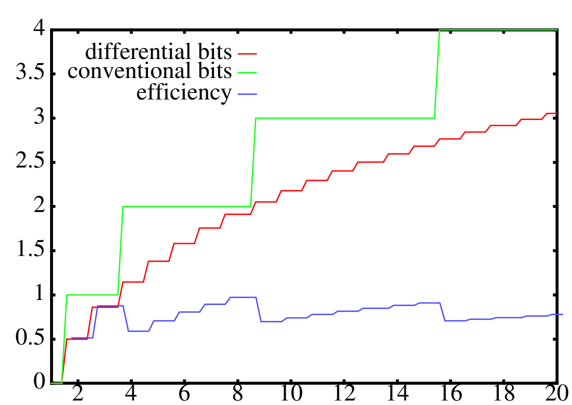


Figure 1

By examining the trade-offs we are tentatively going to use 9 logic levels to get 2 bits per cell. This would achieve approximately twice the density of standard DRAM (modulo overhead). However, to explain the principles involved using standard 8.5" by 11" paper, it is easier to consider the case with 4 logic levels. While while this would only give about 1 bit per cell, it is conceptually the same.

3.2.1 Encoding

The first problem that we have to overcome is the encoding problem - how do we map the standard binary bits for the group into independent logic levels. We computed that the 4 cell group could assume one of 4! = 24 input patterns. Thus we can break down the value of the 4-cell group into 4 number - the first which goes 0,1,2,3, the second 0,1,2, the third 0,1 and the fourth which is always 0. Thus the number of unique combinations of the set is 4*3*2*1 as desired. We can enumerate the combinations as in Table 1 under the headings "unordered."

The goal is to take this and turn it into a set of four numbers, each which is between 0 and 3 with no two numbers being the same (a requirement imposed by the differential encoding.) The result is shown under the "ordered" headings. This mapping is obtained by proceeding from left to right in the set of numbers as follows: The list of possible values for the current ordered digit is all of the values that have not been used in previous digits. For example, consider encoding the unordered value 1 2 0 0. The first ordered digit (A) is always the same as the first unordered digit (A): 1. For subsequent digits, consider the ordered digits that have not been used - i.e. 1 was used so we have 0, 2, and 3 left. Pick the one indexed (0-based) by the unordered value. Thus 2 means pick the third value in the list: 3. For the third unordered digit (c): 0, the digits 1 and 3 have been used, so we have 0 and 2 to pick from. c's value is 0 so pick the first in the list: 0. For the last digit, we only have the digit 2 to pick from. The fourth unordered digit's value is 0, (as always) so pick the first (and only) entry in the list: 2. So the ordered representation is 1 3 0 2. This method produces a 1-to-1 mapping that guarantees an output in which no two digits are the same.

TABLE 1. Mapping of standard binary bits into independent logic levels

Val	Unordered				Ordered				Val	Unordered				Ordered			
	a	b	c	d	A	B	C	D		a	b	c	d	A	B	C	D
0	0	0	0	0	0	1	2	3	12	2	0	0	0	2	0	1	3
1	0	0	1	0	0	1	3	2	13	2	0	1	0	2	0	3	1
2	0	1	0	0	0	2	1	3	14	2	1	0	0	2	1	0	3
3	0	1	1	0	0	2	3	1	15	2	1	1	0	2	1	3	0
4	0	2	0	0	0	3	1	2	16	2	2	0	0	2	3	0	1
5	0	2	1	0	0	3	2	1	17	2	2	1	0	2	3	1	0
6	1	0	0	0	1	0	2	3	18	3	0	0	0	3	0	1	2
7	1	0	1	0	1	0	3	2	19	3	0	1	0	3	0	2	1
8	1	1	0	0	1	2	0	3	20	3	1	0	0	3	1	0	2
9	1	1	1	0	1	2	3	0	21	3	1	1	0	3	1	2	0
10	1	2	0	0	1	3	0	2	22	3	2	0	0	3	2	0	1
11	1	2	1	0	1	3	2	0	23	3	2	1	0	3	2	1	0

This method also maps quite nicely to hardware as will be shown in a later section.

One might wonder how to go from the input vector which would be number between 0 and N!-1 to the unordered numbers. The exact way would be to set the first digit to the input value divided by (N-1)!, the second value to the input divided by (N-2)! modulo (N-1), the third value to the input divided by (N-3)! modulo (N-2) etc. until the final digit would be the input divided by 0! mod 1 - which is always 0. This would be a bit complex for dedicated hardware...

So we can approximate by dividing the input bit vector into fields that are comprised of as many bits as can be fully specified by each of the unordered digits. Recall that the first unordered digit can assume 4 values, the second one 3 values, the third one 2 values and the last one only 1 value (0). Thus we need 2 bits to code the first digit. For the second digit, we waste one of the 3 combinations and code the remaining 2 with one bit. The third group gets another bit, and the fourth group has no bits. Pictorially this can be seen in figure 2. Note that this approximation does waste some bits - our 9 level scheme could theoretically give 18 bits of storage per group but by inefficient use of the bits, this is reduced to 16 bits. (Actually with a little work we can increase it to 17 bits. It would take a lot of work to go to 18!) For the 4 bit example, although a total of 4! = 24 combinations are possible, only 16 directly map into bits. Combinations 4, 5, 10, 11, 16, 17, 22, and 23 contain a 2 in the second unordered position which would not be realizable with only one bit allocated to it. Note that figure 1 does not take this into account. The effect is not severe but a revised figure will take it into account.

4 Level Example:

Group:	a	b	c	d
# Vals:	4	3	2	1
# Bits:	2	1	1	0

(Total of 4 bits)

Example: $\underbrace{10}_{a}\underbrace{1}_{b}\underbrace{1}_{c}$

9 Level Example:

Group:	a	b	c	d	e	f	g	h	i
# Vals:	9	8	7	6	5	4	3	2	1
# Bits:	3	3	2	2	2	2	1	1	0

(Total of 16 bits)

Example: $\underbrace{101}_{a}\underbrace{100}_{b}\underbrace{10}_{c}\underbrace{10}_{d}\underbrace{10}_{e}\underbrace{11}_{f}\underbrace{0}_{g}\underbrace{0}_{h}$

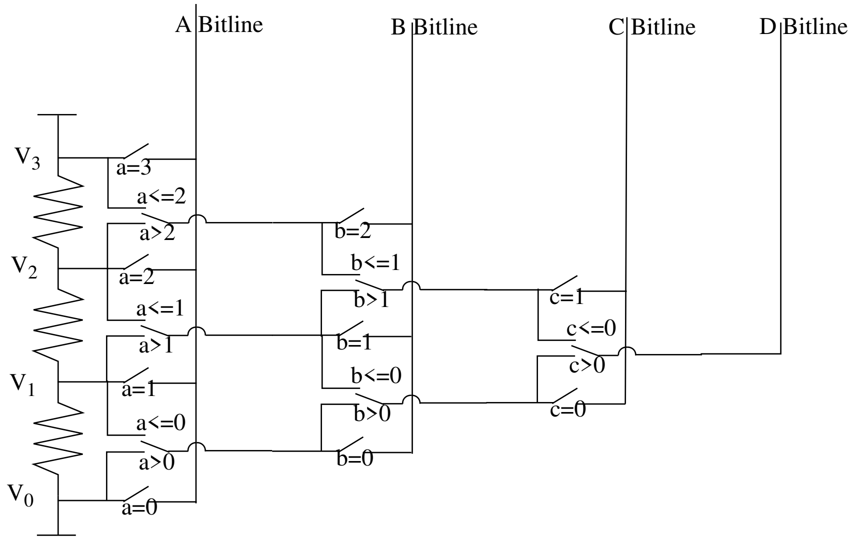
Figure 2

3.2.2 Decoding

Now that we have our differential representation stored in the DRAM array, we need to be able to convert it back into the binary bits from which it was generated. The method is quite simple: For each of the ordered digits, count the number of digits to its right that it is greater than and that is its unordered value. Thus for our example of 1 2 0 0 which mapped to 1 3 0 2 we convert the 1 3 0 2 back as follows: 1 is greater than the 0 to its right but not greater than the 3 or 2. Thus it is greater than 1 digit to its right and receives the unordered value 1. The 3 is greater than both the 0 and 2 to its right and thus is greater than two digits to its right and receives an unordered value of 2. The 0 is not greater than the 2 to its right and thus its unordered value is 0. The 2 doesn't have any values to its right and thus its unordered value is 0. Thus we have obtained the 1 2 0 0 back!

3.3 Possible Hardware Implementation

3.3.1 Encoder



4 Challenges to be addressed

More voltage levels mean less charge to differentiate between voltage levels. This will directly translate to increased time taken to sense a particular level. The sense amplifier is a critical component which will define the speed of the whole circuit.

The increase in number of levels for a fixed cell capacitance may require higher voltage rails to maintain adequate noise margins.

The increased overhead in the form of encoders/decoders and sensitive sense amplifiers may drive up the power consumption. It is hard to predict at this point if there will be any dramatic improvements in the power consumption.

The encoding scheme might put a lower limit on the minimum block size which can be selected.

5 References

[1] M. Stark, "Two bits per cell ROM," *IEEE COMPCON*, pp. 209-212, Spring 1981.

[2] D. Etiemble, "On the performance of multivalued integrated circuits: Past, present and future," *IEICE Trans. Electron.*, pp. 364-371, Vol. E76-C, March 1993.

[3] H. V. Tran, T. Blyth, et.al., "A 2.5V 256-Level non-volatile analog storage device using EEPROM technology," *ISSCC Digest of Tech. papers*, pp. 270-271, February 1996.

[4] M. Horiguchi, M. Aoki, et.al, "An experimental large-capacity semiconductor file memory using 16-levels/cell storage," *IEEE Journal of solid state circuits*, pp. 27-33, February

1988.
[5] L. G. Heller, D. P. Spampinate, and Y. L. Yao, “High sensitivity charge-transfer sense amplifier,” *IEEE Journal of solid state circuits,* pp. 596-601, October 1976.
[6] T. Mano, J. Yamada, et.al, “Circuit techniques for a VLSI memory,” *IEEE Journal of solid state circuits*, pp. 463-469, October 1983.
[7] W. W. Peterson, E. J. Weldon, *Error correcting codes*, 2nd Ed, The MIT press, 1972.
[8] K. Itoh, K. Sasaki, etal, “Trends in low-power RAM technologies,” *Proc. IEEE*, pp. 524-543, April 1995.
[9] J. M. Rabaey, *Digital integrated circuits: A design perspective*, 1st Ed., Prentice Hall, 1996.