SELF-REPARABLE MEMORIES FOR LOW COST ERROR DETECTION AND CORRECTION

Paola Vega-Castillo[1] , *Wolfgang H. Krautschneider*[2]

[1] Instituto Tecnológico de Costa Rica, [2] Hamburg University of Technology

pvega@ietec.org, krautschneider@tu-harburg.de

ABSTRACT:

A self-reparable memory approach for error detection and correction is discussed, and the specific case study of small, low cost single poly non volatile memories in 0.35µm technology is presented.

1. INTRODUCTION

Speed and area consumption play an important role when error detection and correction is implemented, especially for single-poly non-volatile memories (SPNVMs). A self-reparable memory is described and discussed for case studies of SPNVMs in 0.35µm technology for applications such as calibration memories, product identification, configurable products, multiple-time programmable memories (MTPs), one-time programmable memories (OTPs) and small data/code storage for ASICs and System on Chip ICs. Thus, memory sizes of 256 bit, 1 kbit, 4 kB and 32 kB were considered. Details about their architecture and the single-poly memory cells can be found in [1].

2. SELF-REPARABLE MEMORY

As a compromise between parity checking and parity-based error correction, a self-reparable system can be implemented which replaces defective locations by additional words reserved for replacement of the defective memory locations. Figure 1 shows the block diagram of the memory system including the stages for the self-reparation feature.

When programming conventional memories, the input data is read into input registers and kept ready for programming. Once the address data are read in, the programming bias is applied and the program-verify operation starts. In contrast, in the self-reparable memory the input data is read out and the parity bit is generated. After the address data are read in, they are compared with the contents of a bank of registers storing the address of defective locations, if ant. If the address differs from that of a defective location, the input data are programmed to the location addressed by the user.

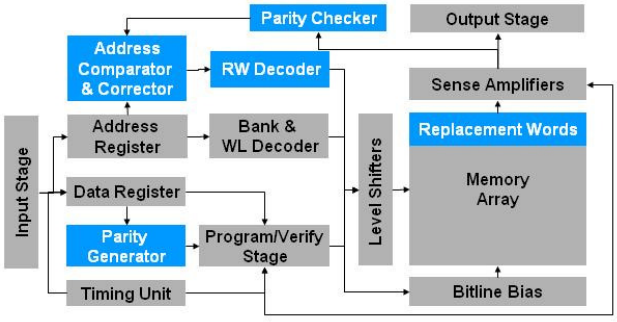

Figure 1. Block diagram of memory system for the self-reparable memory

Figure 2 shows a flow diagram of the programming operation of the self-reparable memory. If the user attempts to write data to a defective location, the system issues a flag to redirect the input data in order to program them into one of the replacement locations. In this way, the memory repairs itself by preventing data storage into defective memory cells, without any effort from the user. For reading, the address of the contents to be accessed is read in. After this, the outputs of the sense amplifiers are transferred to a parity checker before they are passed to the output registers. The parity checker compares the outputs of the sense amplifiers with the parity bit stored for that word. If the memory contents are correct, the outputs of the sense amplifiers are sent to the output registers; if not, the address of the defective location is stored, the memory contents are still sent to the output buffers and an error flag is activated, indicating the user that the data at the output of the memory contains an error. Additionally, the corrupted location is replaced by a replacement address, so that every time the user attempts to access the contents of that address, the memory fetches the replacement location. When this correction takes place, an error flag and a correction flag are activated, indicating the user that the output data contains an error but the defective location has been replaced.

Figure 3 illustrates the read procedure including the steps performed when a replacement location needs to be accessed. If the user attempts to access the corrupted location in the next read operation, the replacement

location is addressed, and the error flag and correction flag are deactivated.

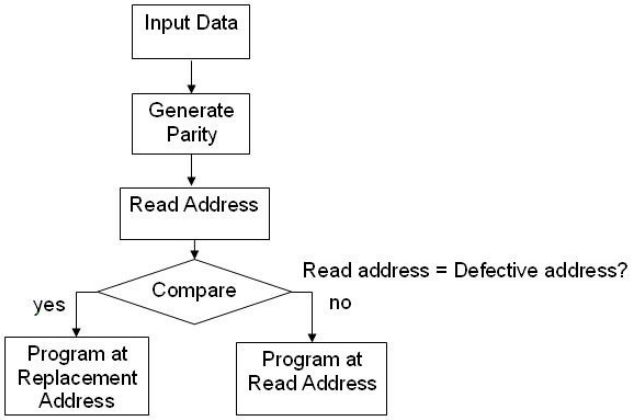

Figure 2. Flow diagram of programming operation of the self-reparable memory

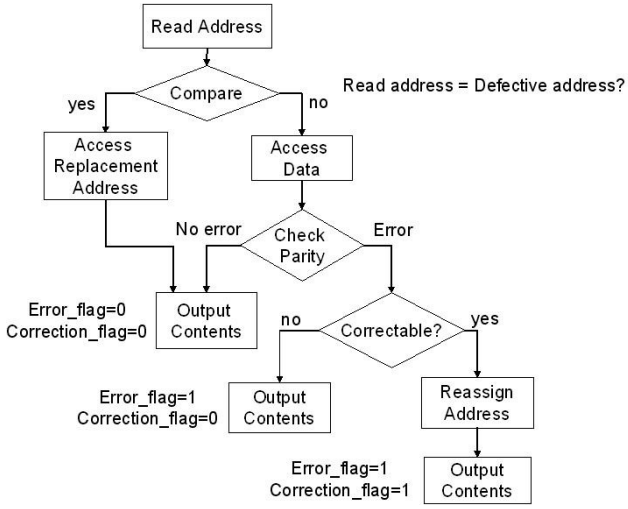

Figure 3. Read procedure of self-reparable memory

3. CASE STUDY

To estimate the area consumption of the digital peripherals, Verilog descriptions were synthesized and the area consumption was estimated using Silicon Ensemble defining a row utilization of 75%. Table 1 presents a summary of the area overhead for the self-reparable memories. For the memory sizes of 256 bits, 1 kbit, 4 kB and 32 kB, a number of 16, 64 and 128 and 1024 replacement words were included, respectively, for a total of 16 replacement words per memory block.

Figure 4 presents a comparison of the area overhead of the self-reparable memory and Hamming code error detection and correction (EDC). Most of the area overhead of Hamming EDC is caused by the area increase of the memory array. In contrast, the area consumption of the self-reparable memory can be defined by the designer, who can vary the number of replacement words according to the application. For self-reparable memories, the area overhead of the digital peripherals may be reduced by optimizing the implementation of some stages, i.e., a manual layout of the address decoders.

Table 1. Area overhead for self-reparable memory systems

Size	Digital area overhead (μm^2 / %)	Analog area overhead (μm^2 / %)	Cell array area overhead (μm^2 / %)
256 b	12960/22%	1 664/4.8%	2790 / 69%
1 kbit	23773/29.7%	3 435/23%	10 886 /63%
4 kB	59836/22.5%	13315/10%	52 530/11%
32kB	478688/22%	106520/10%	598958/15.1%

For all case studies the self-reparable memory saves area and power in comparison with the Hamming code-EDC, since no parity bits must be programmed/erased. This is of great importance for SPNVMs, since the programming and erasing mechanism of the cells is drain avalanche hot carrier injection, and thus high cell currents. Design flexibility is one additional valuable feature of the self-reparable memory.

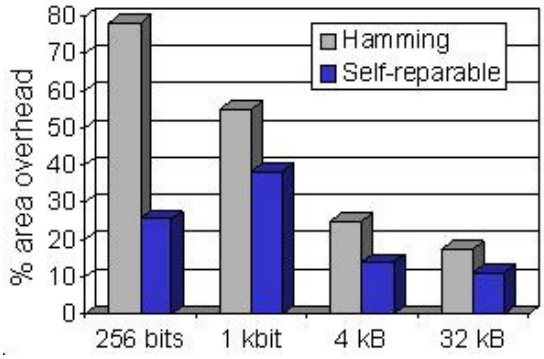

Figure 4. Comparison of area overhead Hamming-code EDC and self-reparable memories

4. CONCLUSIONS

Self-reparable memories offer the advantage of design flexibility, lower area consumption and lower power consumption. The comparison with Hamming code strategies was presented for the particular case of single-poly non-volatile memory systems.

5. REFERENCES

[1] P. Vega-Castillo. "Potentials and Constraints of Single-Poly Non-volatile Memories", ISBN 3-8322-5533-8, Shaker Verlag, October 2006.