White Paper

Correcting Data Errors and Protecting Sensitive Applications with ECC DRAM

Executive Summary

ECC DRAM modules utilizing Hamming code bring single-bit error-code correcting functionality to applications that demand high reliability and system stability. ECC DRAM only has a negligible impact on system performance, and because it can be produced at scale with few differences from regular DRAM modules, costs are kept low while compatibility remains high. These factors make ECC DRAM modules an attractive solution to use at scale in a wide variety of applications.

Introduction

Data errors pose a serious threat to computer systems worldwide. While the occasional computer crash or software malfunction caused by a data error may not be a huge cause for concern in consumer products, the same cannot be said for industrial applications. In such applications, where data integrity and system uptime are critical, a single zero flipped to a one – or vice versa – can cause substantial damage, economic or otherwise.[1]

However, single-bit errors are practically impossible to avoid. Caused by everything from cosmic radiation to electromagnetic interference from nearby circuitry, they are a fact of life in computing.[2]

Error-correcting code memory, or ECC DRAM for short, presents a compelling solution for tackling this challenge. It mitigates the negative impact of single-bit errors, allowing the system to continue operating as if nothing has happened, with a negligible impact on performance, and only with a minor increase in cost.

Background

The Problem with Single-bit Errors

Single-bit errors, or so-called bit flips, occur when a single data bit "flips" to the incorrect value, i.e., a zero flipping to a one or vice versa. This type of error can cause serious problems, including corrupting data, causing software errors, and crashing devices. Depending on the type of application, single-bit errors can have catastrophic consequences.

Single-bit errors are typically categorized as either soft errors or hard errors. Hard errors refer to single-bit errors caused by hardware defects – circuit errors that repeatedly cause bit flips. Hard errors are always caused by circuit errors, which in turn can be caused by anything from improper circuit design, manufacturing defects, and mishandling.[3]

In contrast, soft errors occur when a particle hits the memory cell, causing the cell state to change, thus resulting in the data changing values. Typically, the particle only carries enough charge to change the cell state, i.e., causing a bit flip, while the cell itself remains unscathed.[4] Consequently,

Impactful Bit Flips

While a single flipped bit may not sound like it could cause any significant problems, real-world cases of bit flips tell otherwise.

Autopilot: Disengaged

In 2008, a Qantas aircraft dove 690 feet in 23 seconds after a bit flip caused the autopilot to disengage. The plane was forced to land at a nearby airstrip after about a third of passengers got injured in the incident.[7]

More Votes than Voters

In 2003, the town of Schaerbeek in Belgium saw a bit flip in a voting machine result in 4,096 extra votes for a candidate. The only reason the city discovered the error was that candidates had received more votes than there were eligible voters.[8]

single-bit soft errors tend to go unnoticed unless they cause any problems that are clearly visible to the user, such as a software crash, and even then, it can be close to impossible to attribute the error specifically to a bit flip. Unlike hard errors, soft errors tend to be completely unpredictable.

Thus, it is virtually impossible to shield computer equipment from particles that cause soft errors. One of the most significant sources of such particles is cosmic radiation, which emits energetic neutrons and protons that can cause single-bit errors. Naturally occurring radiation from radioactive contaminants in computer hardware and equipment also risk causing single-bit errors by emitting alpha particles that can interfere with digital signals.[5]

Hamming Code and XOR

Since isolating computer equipment from cosmic radiation and ensuring that computer hardware does not contain even trace amounts of radioactive contaminants is practically impossible, alternative methods of dealing with single-bit errors are necessary.

Instead, the most common method of managing the risk of single-bit errors is to apply logical functions that automatically detect and correct single-bit errors when they occur.

Hamming codes, invented by Richard W. Hamming in 1950, are a type of linear error-correcting codes that are effective in automatically correcting errors when the error rate is low, e.g., single-bit errors in computing.[6]

Putting Hamming Codes and XOR to Work

To understand how Hamming codes and the XOR logical operation work in practice, we will look at an 8-bit data string with error-correcting code.

First, we must know how many – and which – bits are allocated to the Hamming code. With Hamming code, the parity bit positions are powers of two, e.g., $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, and so forth. Thus, the Hamming code (denoted as H) will occupy the following positions:

Position	1	2	3	4	5	6	7	8	9	10	11	12	13
Data Type	H_0	H_1		H_2				H_3					
Data Value													

For the sake of simplicity, we will only look at a short data string instead of a full 72-bit data string.

In the remaining cells, we can then fill in our eight data cells, denoted as D, in sequential order. Finally, we add a simple parity bit for error-checking in the last cell, denoted as P.

Position	1	2	3	4	5	6	7	8	9	10	11	12	13
Data Type	H_0	H_1	D_1	H_2	D_2	D_3	D_4	H_3	D_5	D_6	D_7	D_8	P
Data Value													

As an example, we will use 11001011 as our data string. Since we already know which cells are not occupied by parity bits, we can easily fill our string into the memory as follows:

Position	1	2	3	4	5	6	7	8	9	10	11	12	13
Data Type	H_0	H_1	D_1	H_2	D_2	D_3	D_4	H_3	D_5	D_6	D_7	D_8	P
Data Value			1		1	0	0		1	0	1	1	

Thereafter, we need to calculate the Hamming codes (H_0, H_1, etc.). To do this, we first need to convert the position of all the 1s in our data string into binary.

Position	1	2	3	4	5	6	7	8	9	10	11	12	13
Data Type	H_0	H_1	D_1	H_2	D_2	D_3	D_4	H_3	D_5	D_6	D_7	D_8	P
Data Value			1		1	0	0		1	0	1	1	
Position in Binary			0011		0101				1001		1011	1100	

We then apply the XOR function, which we can illustrate by vertically lining up these positions converted into binary. We then add all the 1s together vertically and denote even numbers of 1s as 0 and odd numbers of 1s as 1.

0	0	1	1
0	1	0	1
1	0	0	1
1	0	1	1
1	1	0	0
Odd	Even	Even	Even
1	0	0	0

The result of the XOR calculation is the four-digit data string 1000 which we then have to plug into the still empty Hamming code cells in backward order. Lastly, to finish our data string with the error-correcting code, we fill out the final cell with a parity bit. Since the number of 1s in our data string is even, and assuming the system uses even parity, P=0, giving the final 13-bit data string of 0010100110110.

Position	1	2	3	4	5	6	7	8	9	10	11	12	13
Data Type	H_0	H_1	D_1	H_2	D_2	D_3	D_4	H_3	D_5	D_6	D_7	D_8	P
Data Value	0	0	1	0	1	0	0	1	1	0	1	1	0
Position in Binary			0011		0101			1000	1001		1011	1100	

After adding the Hamming code, when vertically lining up the binary position of each 1 and using the XOR function, we will now always get the result 0000 – no matter the previous location of each 1. This is what allows the XOR function to identify the location of single-bit errors as we will soon look into.

0	0	1	1
0	1	0	1
1	0	0	0
1	0	0	1
1	1	1	1
1	1	0	0
Even	Even	Even	Even
0	0	0	0

If a single-bit error has not occurred, the result of the XOR function always adds up to 0000.

To understand how the final parity bit and the Hamming code parity bits are helping us, let us now imagine that one bit, D_3, gets flipped from 0 to 1:

Position	1	2	3	4	5	6	7	8	9	10	11	12	13
Data Type	H_0	H_1	D_1	H_2	D_2	D_3	D_4	H_3	D_5	D_6	D_7	D_8	P
Data Value	0	0	1	0	1	1	0	1	1	0	1	1	0

The final parity bit, P=0, indicates that the number of 1s should be even, but we are now finding an uneven number of 1s in the data string – indicating that a data error has occurred.

To find the position of the flipped bit, we use the same XOR function as before, i.e., vertically line up all the binary locations of each 1 and count the number of 1s:

Position	1	2	3	4	5	6	7	8	9	10	11	12	13
Data Type	H_0	H_1	D_1	H_2	D_2	D_3	D_4	H_3	D_5	D_6	D_7	D_8	P
Data Value	0	0	1	0	1	1	0	1	1	0	1	1	0
Position in Binary			0011		0101	0110		1000	1001		1011	1100	

0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	0	1
1	0	1	1
1	1	0	0
Even	Odd	Odd	Even
0	1	1	0

The result of the XOR function, 0110, shows the position (0110 = 6) of the flipped bit, allowing the error-correcting code to immediately flip it back to its intended state, in this case, 0:

Position	1	2	3	4	5	6	7	8	9	10	11	12	13
Data Type	H_0	H_1	D_1	H_2	D_2	D_3	D_4	H_3	D_5	D_6	D_7	D_8	P
Data Value	0	0	1	0	1	0	0	1	1	0	1	1	0
Position in Binary			0011		0101	0110		1000	1001		1011	1100	

Returning the original – correct – data string, 0010100110110.

DRAM Susceptibility

In modern computing applications, DRAM modules stand out as the type of component most susceptible to single-bit errors.[9] Due to memory modules' storage density and the longevity of data in memories, they represent a larger share of susceptible device surface area than logic circuits.[10] Thus, DRAM modules are a key focus for innovations that help systems avoid or mitigate the risk of single-bit errors.

Single-bit errors in DRAM modules also have a significant impact on system stability, thus elevating the importance of addressing single-bit errors in DRAM modules.

Challenges

Introducing error-code correcting memory that leverages Hamming code means that many challenges need to be addressed in order for ECC DRAM to become a viable alternative for most applications.

- **Easy Integration**

DRAM modules are practically used in every conceivable computing application, from simple computer workstations to small form factors in the most demanding applications. With vastly different applications, requirements (e.g., performance and capacity), and environments, providing satisfactory reliability and performance is a considerable challenge. Any addition of complex circuitry or logic components add potential weak points that may jeopardize the ECC DRAM module's stability and reliability, making it an impracticable alternative to standard DRAM modules.

- **Low Costs and Scalability**

DRAM is a volume product where cost has a huge influence on purchasing decisions. Therefore, outside truly low-volume niched applications, ECC DRAM must be available at an attractive price point – which in return requires high-volume production and only smaller changes to the module's architecture for achieving the error-correcting code functionality. that may jeopardize the ECC DRAM module's stability and reliability, making it an impracticable alternative to standard DRAM modules.

- **Limited Impact on Performance**

The viability of implementing error-correcting code also hinges on the potential performance impact on the computer system. Software-based ECC, for instance, has a considerable impact on system performance, with high overhead rates and much slower transfer rates than hardware-based solutions.[11] If the impact on performance is too high, the benefits of ECC may be outweighed by the downsides in the vast majority of computing applications.

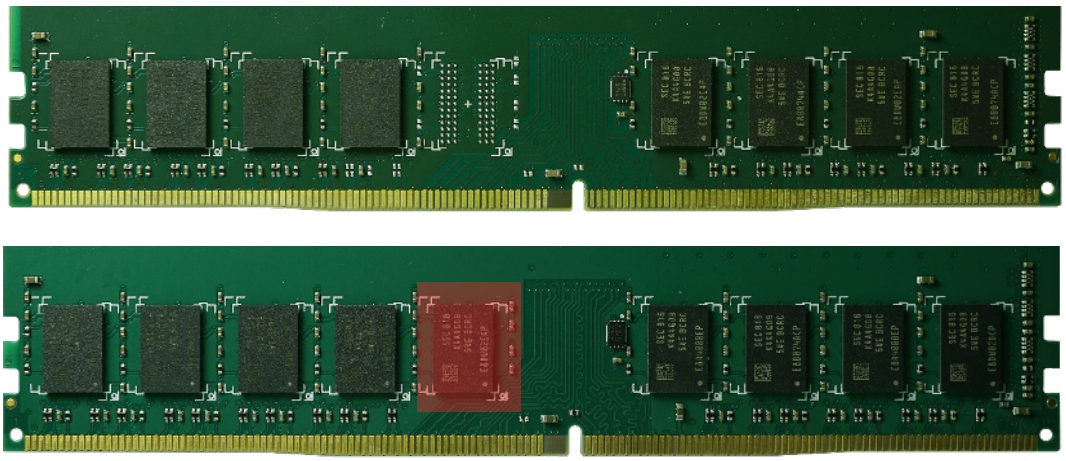
Solutions

Modern ECC DRAM modules fully address the challenges of implementing Hamming code-enabled error-code correcting functionality in enterprise computing applications while keeping costs low and allowing a high degree of flexibility.

- **Simple Design with Powerful Implications**

ECC-enabled DRAM modules retain a remarkably similar design to their non-ECC counterparts. Instead of adding additional controllers or logical functions that implement the ECC function, modern ECC DRAM is equipped with an additional integrated circuit (IC) for each memory rank. This addition, in turn, grants the DRAM module the extra data bits required by the ECC function.

The result is that manufacturing processes and module designs can remain largely unaltered for ECC DRAM compared to regular DRAM – enabling high-volume production and making it easy for ECC DRAM to keep up with advancements in DRAM technology. Because the module design is largely the same, ECC DRAM can also easily be paired with other DRAM features and technologies, e.g., coatings and side fill – allowing ECC DRAM to be used in challenging environments and applications just like their regular industrial-grade DRAM counterparts.



Two DRAM modules – one with ECC and one without. Note the extra IC on the ECC DRAM module (highlighted in red).

- **High Performance and Wide Compatibility**

Since ECC memory essentially only means an additional IC embedded on the PCB – and not any added logic components – that means that the actual XOR calculations have to occur elsewhere, in this case, in the computer's central processing unit (CPU). This means additional computing load on the CPU, but due to the relatively simple calculations required by the XOR function, the impact on overall computer performance is close to negligible. For example, in a comprehensive 2014 study on the performance differential between standard DRAM, ECC DRAM, and registered ECC DRAM by Puget Systems, ECC DRAM was found to cause a 0.25 percent decrease in system performance, with registered ECC DRAM decreasing system performance by 0.44 percent.[12]

Instead, a more common concern is that ECC DRAM modules may not be available in the highest-tier specifications, which underlines the importance of working with DRAM suppliers that can provide high-performance ECC DRAM with higher-tier specifications.

Because of the high penetration of ECC DRAM in server applications, close to all server CPUs and motherboards fully support ECC DRAM modules, making them easy to integrate into any application. Consumer-grade products, while traditionally not a substantial market for ECC DRAM, also increasingly support ECC DRAM – allowing for experimentation with and implementation of ECC DRAM even with off-the-shelf consumer products.[13]

Conclusion

ECC DRAM from reputable industrial-grade DRAM module vendors provides exceptional stability and reliability ideal for mission-critical applications. Using Hamming code, ECC DRAM provides powerful single-bit error-correcting functionality with a negligible impact on system performance and without introducing any unnecessary module complexity. Because of ECC DRAM's simple design, DRAM with ECC functionality can easily be combined with other DRAM features and technologies, including coatings and side fill – making them an ideal choice for demanding and mission-critical applications. All this combined with the exploding volumes of data generated in applications such as 5G and edge computing makes ECC an attractive means of reducing the risk for single-bit errors occurring at scale and in the critical smart city infrastructure of the future.

References

1. https://www.fasthosts.co.uk/blog/ecc-ram-keeping-critical-data-error-free/
2. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01179-ecc-embedded.pdf
3. https://www.ti.com/lit/wp/spna109/spna109.pdf
4. https://www.intelligentmemory.com/support/faq/ecc-dram/what-is-the-root-cause-for-these-single-bit-errors.php
5. https://link.springer.com/article/10.1007/BF02660131
6. http://www.eecs.umich.edu/courses/eecs373.w05/lecture/errorcode.html
7. https://www.cio.com/article/3172790/radiation-from-outer-space-could-sabotage-your-smartphone.html
8. https://www.wnycstudios.org/podcasts/radiolab/articles/bit-flip
9. https://books.google.com.tw/books?id=WCqrOkMExu8C&pg=PA24#v=onepage&q&f=false
10. https://books.google.com.tw/books?id=uyGsewzo1jcC&pg=PA240
11. https://www.micron.com/-/media/client/global/documents/products/technical-note/nand-flash/tn2971_software_bch_ecc_on_linux.pdf
12. https://www.pugetsystems.com/labs/articles/ECC-and-REG-ECC-Memory-Performance-560/
13. https://hardwarecanucks.com/cpu-motherboard/ecc-memory-amds-ryzen-deep-dive/

Innodisk Corporation

5F., NO. 237, Sec. 1, Datong Rd., Xizhi Dist., New Tapei City, 221, Taiwan
Tel : +886-2-7703-3000
Fax : +886-2-7703-3555
E-Mail : sales@innodisk.com
Website : www.innodisk.com