Error Correction Code in SoC FPGA-Based Memory Systems

WP-01179-1.2 White Paper, with contributions from Micron Technology

This paper examines the potential sources and implications of soft errors and a method implemented by Altera Corporation and Micron Technology to make embedded systems more resilient to these types of soft errors through error detection and correction.

Introduction

Continuously advancing semiconductor process technologies have enabled increased component integration, functionality, and performance in embedded systems. While the increased capabilities reap huge rewards, one of the side effects of higher-performance systems is that more attention must be paid to the probability of soft errors. Decreasing supply voltages cause integrated circuits to be increasingly susceptible to various types of electromagnetic and particle radiation. As memory size in embedded systems grows to 100s of megabytes, soft errors due to naturally occurring alpha particles may exceed acceptable levels. As interface speeds exceed 1 Gigabits per second, excessive noise and jitter may cause errors in the transmission lines to and from external memory.

Memory Bit Error Sources

Bit Cell Soft Errors

Commonly used memory bit cells retain their programmed value in the form of an electrical charge. Writing a memory bit cell consists of reprogramming and forcing the electrical charge to represent the new desired value. Memory bit cells will retain their value indefinitely, as long as basic requirements are met, e.g. power is applied, and—for dynamic memory types—a refresh method is active.

The stored charge can be negatively impacted by injection of a charge foreign to the memory device. Cosmic particles colliding with atoms in the atmosphere cause energetic rays which may affect the stored charge. To flip the value of a memory bit cell, enough charge has to be injected to change it to represent an incorrect logic value.

High-energy alpha particles make up about 10 percent of cosmic rays and are able to penetrate many meters of concrete. Lower-energy alpha particles may be emitted by decay of materials used in the chip package, and while lower in energy, the distance these need to travel to make an impact is small. Similarly, gamma rays are highly energetic, are naturally produced by decay, and present in cosmic rays. The earth's atmosphere is a natural, significant, but not flawless barrier to cosmic particles and rays. Consequently, at higher altitude, on mountaintops or in airborne systems, the thinner atmosphere provides less protection from these particles, and so the chance of soft errors is higher.

101 Innovation Drive
San Jose, CA 95134
www.altera.com

The event in which an external energy injection inadvertently modifies the value of a memory bit cell is referred to as a single event upset (SEU). The class of these errors is soft errors, as the error is not caused by a defect in the device, but instead by the device being subject to an outside disturbance. If the correct data is subsequently rewritten, it is not likely to undergo the same upset. As such, the likelihood of such an event is extremely small, while it increases with growing memory capacity.

Hard Errors

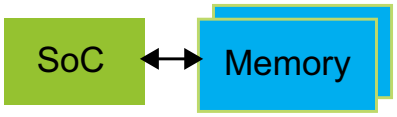
Hard errors are categorized as incorrect functionality. This is where the error is often reproducible and persistent. While a system, including the memory contained therein, is assumed to be free of faults after production, this situation may change as the device ages. Factors such as excessive temperature variation, voltage stress, high humidity, and physical stress, all contribute to increased probability that a component in the system may start to fail. These errors may show as a stuck bit caused by a defect in a memory cell or in a printed circuit board trace.

Transmission Errors

Transmission errors are those errors that occur in the path between a memory bit cell and the functional unit that is reading or writing data. This type of error can be introduced by jitter and noise in the system temporarily exceeding design margins of the transmission path, and thus are dependent on design margins, quality of components used, and the systems susceptibility to electrical energy in its environment.

Inductances, capacitances, and wire lengths of physical connections to external memory are orders of magnitude higher compared to internal wiring in the system on chip (SoC) or the memory devices. Still, transmission errors also can occur inside components. Alpha particles and gamma rays can impact sense amplifiers and memory bit lines, causing the incorrect capture of a data value.

Figure 1. Transmission Error Path



Implications of Errors

Memory data corruption is often fatal to the operation of an embedded system. In a processor-based system, memory errors result in incorrect values in either instruction or data streams. Modern processors will detect illegal instructions, commonly forcing a reboot of the system. Errors in data streams may cause the program flow to derail, which often results in illegal access to protected memory. These events have their equivalent in the desktop world as a "blue screen of death" or a "core dump."

While a crash is undesirable in embedded systems, the alternative is worse. Errors that are not immediately detected can linger in the system for an extended period of time. Undetected memory errors can multiply as the faulty data is used to calculate new data. Once faulty data has been detected, the originating point and the subsequent induced damage may be difficult to correct or even identify.

Embedded systems often operate for extended periods of time and are not frequently rebooted as one may see with desktop computers. This gives embedded systems the additional disadvantage that errors will accumulate over time.

The effects of data corruption or system crashes are numerous. Misbehaving systems will annoy users and make customers unhappy. Maintenance costs may increase, as customer complaints trigger expensive investigations for error sources that are non replicable. A sudden system failure may cause an unsafe environment around heavy machinery, and errors in secure systems may provide access via unintended backdoor entry methods.

Likelihood of Errors in Embedded Systems

The rates of hard errors and transmission errors are a function of many variables. Studies have measured such errors in larger systems, but those results may not translate to other systems.

On the other hand, various studies have published soft error rate (SER) results. As a practical example, an embedded system with 1 Gigabyte of dynamic memory is expected to have a mean time between failures (MTBF) in the range of a few times per year to once every few years.

The MTBF should be considered in view of the number of systems in the field. As a system supplier, you should regard the possible number of fails of the total devices in a given time period. Assuming 10,000 devices in the field with an MTBF of 10 years, this implies that an average of 1,000 devices per year is expected to suffer from a single bit soft error.

The acceptability of such an error rate depends on the application domain. Developers of applications used at high altitudes will be concerned with higher SERs due to cosmic rays. Military, automotive, high-performance computing, communication, and industrial customers will be concerned with degradation of safety, security, and reliability. In the consumer domain, an MTBF of one year may sometimes be acceptable. In many cases however, the added maintenance cost and the number of unhappy customers are key factors driving the need for a solution.

Improving Error Resilience

Transmission Errors

The Altera® SoC FPGA supports up to 533 MHz (1066 Gbps) DDR3. The specification for the DDR3 interface and the way this has been implemented in both the SoC FPGA and external memory device guarantee a negligible error rate. This assumes a robust board design and control of jitter and noise within the boundaries dictated by the DDR specifications.

A large-scale study performed by Google in cooperation with the University of Toronto and another study by Stanford University showed that a subset of all the systems analyzed created the majority of the errors. These errors may well be caused by excessive jitter or noise, or may be related to sub-par quality of the systems and their components.

The probability of transmission errors increases with higher interface speeds, such as defined for DDR4 and beyond. As voltages of power planes and signal levels shrink in support of reduced power consumption and higher interface speeds, jitter and noise are becoming harder to control. JEDEC points out that for next-generation memory specifications of DDR4 and GDDR5, the impact of jitter and noise has driven the specification to allow for a tradeoff between a certain bit error rate and simplification of design, characterization, and qualification. Any allowed bit error rate would effectively necessitate a method to correct occurring errors.

Soft Errors

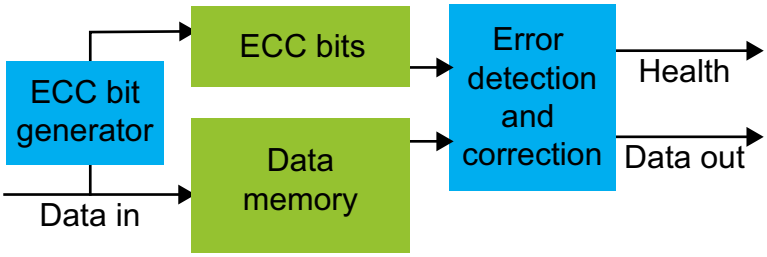
Because soft errors are unavoidable, methods have been developed to make systems resilient to many such errors. That is, when an error occurs, it can be detected, corrected, and the corrected value passed on, and thus the system continues uninterrupted. This feat is accomplished by adding bits to memory data words, whereby the widened word carries sufficient information to detect and correct errors. The more bits are added to a data word, the more errors in a word can be corrected. This makes error correction a function of cost and desired reliability.

A method that allows correction of a single error and detection of two errors in a word is both cost-effective and proven to provide excellent error resilience in embedded systems. This technology, widely deployed in the industry, is referred to as error correction code (ECC).

Basic Implementation of ECC

ECC is implemented by making the memory wider and adding a limited amount of combinatorial logic in the path to and from that extra memory. The logic required for ECC encoding is based on well-established polynomial Hamming algorithms. An ECC bit generator creates the ECC bits out of the data being stored and stores the ECC data together with the regular data. An ECC detection and correction logic function is inserted at the output of the memory. When reading the memory, this function will check the combination of ECC data and regular data. If no error is detected, it will pass the regular data through unchanged. If a single bit error is detected, it will correct the error bit pass through the regular data, now with all bits correct, and optionally raise a flag. If two errors are detected, it will raise a flag, allowing the system to gracefully respond to the event.

Figure 2. ECC-Enabled Memories Are Wider and Have Additional Logic



Advantages of ECC

The ability to correct a single error and detect double errors brings many benefits. While the introduction of ECC has been driven by the SER of large memories, it adds resilience against other types of errors as well.

A single hard error, such as a stuck bit line inside a memory or unreliable connection on a printed circuit board, may be fully covered by ECC. Single bit transmission errors are covered as well. Key is that single bit errors in a word can be corrected. That is, ECC will correctly handle many errors in the system, as long as any single word shows no more than a single bit error.

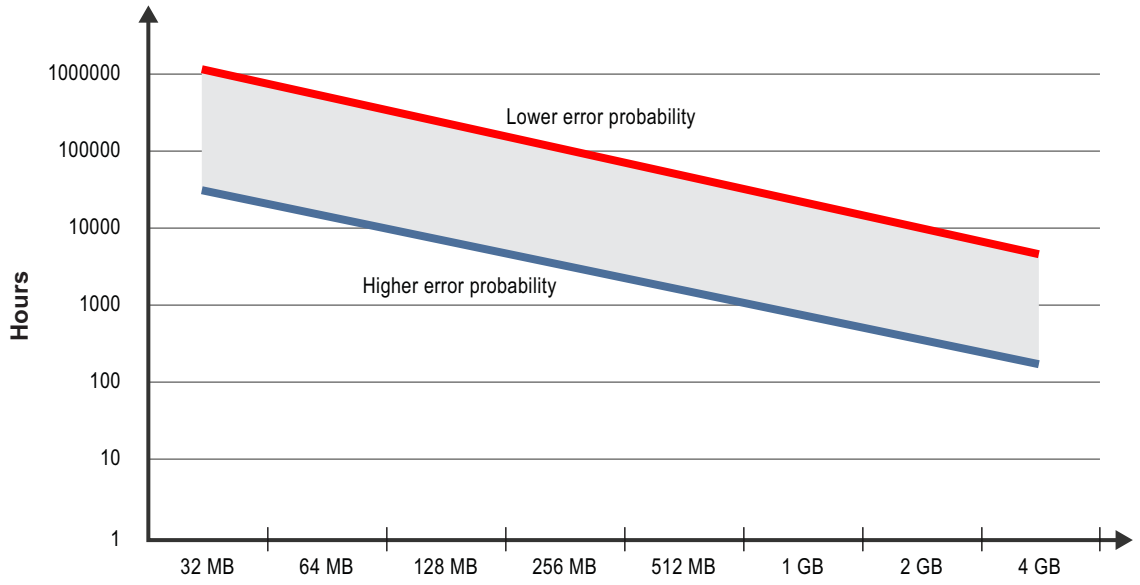
Another benefit is that the ECC logic can indicate a system health status. For any single bit error in a word, the ECC logic will correct the error. It also can signal a failure status to the processor, and the operator can take measures relevant to the required reliability of that system. This method turns system degradation into a maintenance task that can be scheduled, as opposed to a response to an unexpected fatal system error condition.

Based on heuristic probabilities, a model for estimating soft errors in systems without and with ECC is explained in the appendix. It shows that the addition of ECC effectively increases the MTBF from being shorter than the life time of the product to longer than the life time of the universe.

Table 1. ECC Brings Very High Resilience to Single Errors

1-GB DDR, 32-bit Data, 7-bit ECC	Error Probability	Failures per 1 Billion Hours (FIT)	MTBF
No ECC	Higher: 1.10×10^{-13}	10^{6}	979 hours
	Lower: 6.9×10^{-15}	59269	1.9 years
ECC	Higher: 1.19×10^{-13}	10^{-14}	10^{19} years
	Lower: 6.9×10^{-15}	$4*10^{-17}$	$3*10^{21}$ years

Figure 3. Example Soft Error MTBF for External DDR Memory Without Error Resilience



Altera and Micron ECC Solutions

ECC for External DRAM

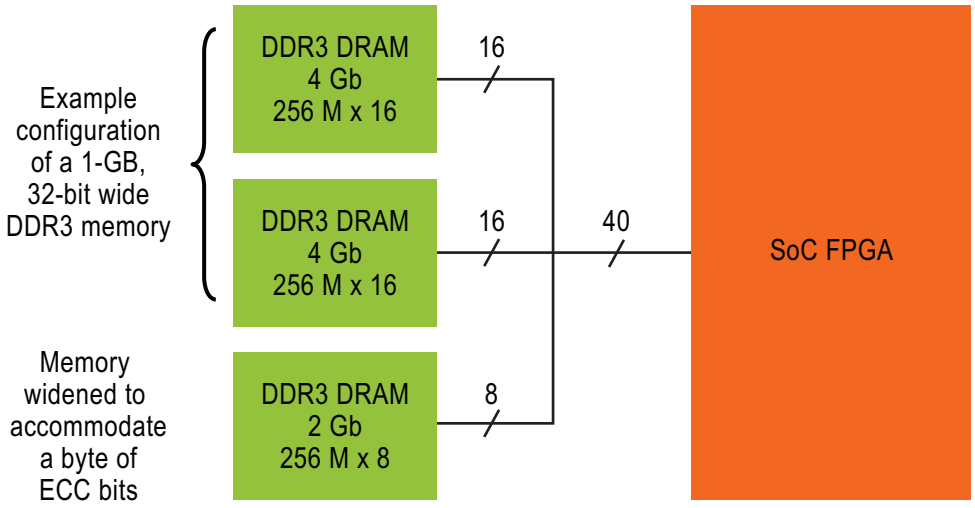
Adding ECC capability to a memory that is external to the Altera SoC FPGA only requires that the memory data width is increased. The functions to generate the ECC bits, and those for error detection and correction, are already built into the memory controller inside the SoC FPGA. With that regular, albeit wider, memories can be used. In addition, the data path between the SoC FPGA and the external memory is covered by ECC, thus including resilience to some transmission errors in the data path.

External memory systems with 16- or 32-bit wide data lanes between the SoC FPGA and the external memory require six or seven additional storage bits of ECC, respectively. Altera's architecture simplifies this by defining that one additional byte-wide memory be used for either scenario.

Typical Architecture of ECC-Protected DRAM

A modern embedded system consisting of a SoC FPGA and external memory might be required to include 1 GB of DRAM. To achieve a cost-effective, 32-bit wide memory interface for such a system, a typical selection of DRAM might include a combination of one 8-bit wide device and two 16-bit wide devices, configured in parallel.

Figure 4. Typical External DDR Memory Architecture



ECC for NAND Flash

External NAND flash can be connected to SoC FPGA devices, both as one of the possible boot configuration memories and as application as a file system. NAND flash is rather sensitive to soft errors, and there is often a desire for ECC protection. Instead of making data words in the NAND flash devices wider, as seen in external DDR memories, NAND flash devices provide additional storage buffers in the device to retain ECC data bits. ECC is done on larger data blocks, 512 or 1024 bytes, and the SoC FPGA device can correct up to 24 bit errors. While the implementation specifics are tuned to generally accepted error patterns of NAND flash, the resilience method is similar to those found on regular SRAM and DRAM.

Error Resilience Inside Altera SoC FPGAs

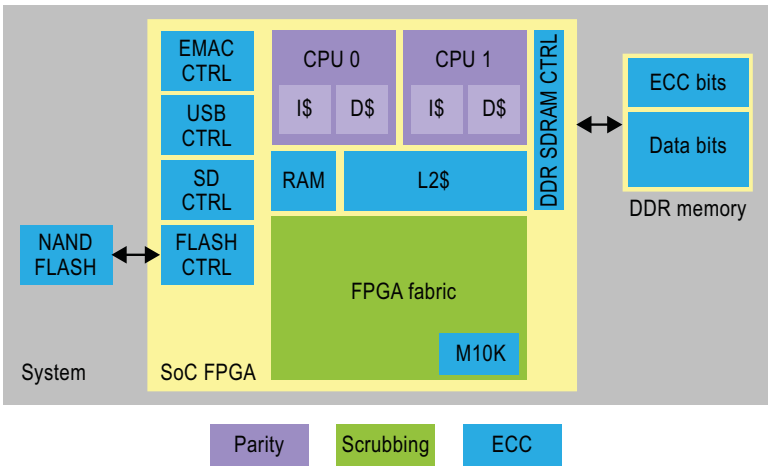
ECC functionality is built into the SoC FPGA device for a large number of on-chip memory instances. The level 2 cache, the scratch RAM, memory inside the FPGA fabric, and memories that serve as data buffer in peripherals are each widened and equipped with ECC generator and correction logic. As these are built-in, using the ECC feature carries no additional cost. Similarly, the ECC logic for external memories and NAND flash is built-in the SoC FPGA.

Data and instruction caches are relatively small in their physical size and thus less prone to soft errors. They do need to operate at high performance levels, and to avoid additional latency when reading the level 1 caches, a simple parity check method is used.

The configuration bits inside the FPGA fabric are not organized in wide data words and do not lend to ECC implementation. Instead, the FPGA fabric has a built-in hardware engine that allows for cyclically checking for the correctness of the configuration bits, raising a flag when an error is detected. This error correction method is referred to as "scrubbing," and is an industry-standard method for this class of devices.

Altera SoC FPGAs have all the required logic to support error resilience at the system level, integrated into the device. By merely widening the external DDR memory, a comprehensive system-level wide method of resilience to soft errors and transmission errors can be obtained.

Figure 5. SoC FPGA's System-Level Error Resilience



Parity
Single error detection (SED):
- Signal on parity mismatch
- Optimized for speed and performance

Scrubbing
FPGA configuration bits validation:
- Configuration bits read back by hardware engine
- Compare to know good value
- Signal on mismatch

ECC
Single error correction, double error detection (SECDED):
- Detect and correct single bit errors
- Detect double bit errors
- Signal on either error correction or detection
- Health signal while system still operational

Summary

Memories in embedded systems may suffer from cosmic energy injections causing bit flips and from transmission errors due to excessive noise and jitter that can cause incorrect transfer of data. While the chance of such errors is small, with growing memory sizes, interface frequencies, and bandwidth, the likelihood of such events is a growing challenge. System-level error resilience is increasingly becoming a design consideration in higher-performance embedded systems.

Altera SoC FPGA devices are designed for high levels of error resilience, through the use of scrubbing, parity checking, and ECC. They enable efficient addition of ECC on external memories, specifically including 32-bit-wide external DRAM memories found in higher performance systems. Adding ECC practically eliminates sensitivity to such errors in memories, contributing strongly to a system-level improvement in error resilience.

Appendix

Error Occurrence Probability

A simplified model for first order analysis of soft errors due to memory bit flips, e.g. the model assumes randomness of SEU and does not include hard or transmission errors.

n = number of bits in a data word (n_d), plus the number of bits for correction code (n_e)

m = number of bits in memory

w = number of words in memory, or m/n_d

p = probability of a single bit error in a period of one hour

Empirically determined range: $1.19*10^{-19}$ ... $6.9*10^{-15}$

Probability of Experiencing an Unrecoverable Error Without ECC

p_t = chance of an error in a memory of bit size m in a period of one hour

$$P_t = 1 - (1 - p)^m$$

Probability of Experiencing Unrecoverable Errors with ECC

p_w = probability of an uncorrectable error in a word in one hour

p_{mem} = probability of an uncorrectable error in memory in one hour

$$P_w = 1 - (1 - p)^n - (n \cdot p) \times (1 - p)^{n-1}$$

$$P_{\text{mem}} = 1 - (1 - p_w)^w$$

Further Information

- "A Second Look at Jitter: Calculating Bit Error Rates," Justin Redd, Maxim Integrated Products: www.eetimes.com/electronics-news/4164171/A-second-look-at-jitter-Calculating-bit-error-rates
- "Understanding the New Bit Error Rate DRAM Timing Specifications," Perry Kelly, Agilent Technologies: www.jedec.org/sites/default/files/Understanding%20BER%20based%20timing%20specifications%20Agilent%202011_1101.pdf
- "DRAM Errors in the Wild: A Large-Scale Field Study," Bianca Schroeder, University of Toronto, Eduardo Pinheiro and Wolf-Dietrich Weber, Google Inc.: www.cs.toronto.edu/~bianca/papers/sigmetrics09.pdf
- "Hard Data on Soft Errors: A Large-Scale Assessment of Real-World Error Rates in GPGPU," Imran S. Haque, Vijay S. Pande, Stanford University: http://cs.stanford.edu/people/ihaque/papers/gpuser.pdf
- White Paper: *Enhancing Robust SEU Mitigation with 28-nm FPGAs*, Altera Corporation: www.altera.com/literature/wp/wp-01135-stxv-seu-mitigation.pdf
- On the Need to Use Error-correcting Memory," Berke Durak: http://lambda-diode.com/opinion/ecc-memory

Acknowledgements

- Hans Spanjaart, Senior Technical Marketing Manager, Embedded Products, Altera Corporation
- Matthew Prather, DRAM Applications Engineer for Server Platforms, Micron Technology

Document Revision History

Table 2 shows the revision history for this document.

Table 2. Document Revision History

Date	Version	Changes
April 2012	1.2	■ Minor text edits.
March 2012	1.1	■ Moved figure 4 to Typical Architecture. ■ MInor text edits.
March 2012	1.0	Initial release.