NON-VOLATILE MEMORY MANAGER UNIT FOR SPACE APPLICATIONS

Radek Hajek
Doctoral Degree Programme (1), FEEC BUT
E-mail: xhajek29@stud.feec.vutbr.cz

Supervised by: Lukas Fujcik
E-mail: fujcik@feec.vutbr.cz

Abstract: The paper deals with HDL design methodology and methods of data protection against the influence of the Single Effect Upset (SEU) for the space applications. Presented methodology is applied to design of non-volatile memory manager unit, which is presented in this paper. This unit is responsible for transfer data between memories and must be fault-tolerant to the SEU. To ensure high reliability the data in the memories are protected by EDAC logic.

Keywords: Memory Manager Unit, Space Design Methodology, Single Effect Upset, SEU

1 INTRODUCTION

Every object in the space is exposed to a radiation. Even microelectronic devices are not an exception and radiation can influence the functionality of the device. Single ionizing particle striking the microelectronic device can change the state of the device and it can create a free charge which can lead to logic value change inside the device. This change is commonly known a the single event upset (SEU). Effects of the SEU are especially critical for the outer space applications where the exposing to the ionized particles and radiation generally is distinctly higher than on the Earth. The SEU may alter the logic state of any static memory element (register or RAM cell) or cause transient pulses in combinatorial logic. The effect of the SEU can be reduced by using special radiation hardened (rad-hardened or rad-hard) devices. However, the influence of the SEU can never be completely reduced. Therefore, digital circuits for space applications must be designed with respect to the possible SEU.

Capacity and performance of the Field Programmable Gate Arrays (FPGAs) suitable for space applications are constantly increasing. In critical applications, the FPGAs are more being used and they are even replacing the Application Specific Integrated-Circuits (ASICs). The main advantages of using FPGA over discrete logic are decreasing the weight of the board, increasing reliability with reduced solder connections and flexibility to make modifications without board layout changes. In comparison with ASICs the design does not have to be completed a long time in advance of delivery. However, there are differences between space suitable FPGA design and commercial application design. FPGAs are generally perceived as easy to modify and correct late in the development process and the design methods are chosen respectively. This is however not true for the space flight applications where very expensive and often only one-time programmable FPGAs are in use and Hardware Description Language (HDL) rules and recommendation should be followed to minimize the costs. [1]

This paper deals with space suitable HDL design methodology and methods of data protection against the influence of the SEU. It also presents practical example of the design of non-volatile memory manager unit described in VHDL language. This unit was designed for FPGA for the space flight application.

2 DESIGN METHODOLOGY

Special rules and practices to achieve space suitable FPGA design and implementation should be used. Only space-grade (rad-hard) FPGA must be selected. These FPGAs are very expensive and in most cases, only one-time programmable. Good practice is to prototype the design with low-cost commercial-grade FPGA and to program the final space-grade FPGA once the design is tested and fully verified with a commercial-grade FPGA. However, radiation effects such as SEUs cannot be tested with commercial-grade FPGAs and it is not feasible to perform test of radiation hardness of design on space-grade FPGAs due to high price and longtime of such tests. The methods introduced in this chapter are the most common methods that are used to protect both critical and non-critical parts of design. Before the methods are applied, following points need to be considered to choose suitable protection method (or combination of methods):

1. Frequency of SEU effects appearance – analysis of radiation environment needs to be performed to establish impact of particles to design
2. Critical and non-critical parts of design - establish where special attention should be paid and where protection can be omitted
3. Set of available protection methods - choose meaningful combination of methods to be used.

2.1 TRIPLE MODULE REDUNDANCY

The most widespread method of SEU protection is to use triple module redundancy (TMR). The TMR may be applied to registers or even parts of the design. When applied to registers, each register is implemented as three flip-flops and the correct value of the register is selected by the majority of registers. The inclusion of TMR can be done directly in the HDL description. Space-grade FPGA may contain registers with built-in TMR [2] or the inclusion may be done directly by the synthesis tool without any HDL modification [3]. Implementing TMR is very simple however there is always need to do the refresh of corrupted register to correct value before any other SEU occurs. To achieve an efficient TMR protection, it requires that the refresh rate is higher than the frequency of SEUs.

2.2 CODE LEVEL CONSIDERATIONS

The most commonly used digital structure are counters. Especially critical are counters that are used as timers. When such a counter is not protected, there might be even 50% period change due to SEU. Protection of the counter can be done using TMR. For modulo counters it is necessary to check for greater or equal than (>=) maximal value instead of equal to (=). When SEU happens, the counter can skip the maximal value and enter a state with no recovery. Without the protection of registers, this can lead to 100% period change due to SEU. [1]

Another challenge is the protection of Finite State Machines (FSM) against the SEU. The main issue is handling of unused states that can be entered by the SEU. Unused states are formed when there are less used states than number of all possible states given by encoding. There always must be implemented some recovery logic which moves FSM from unused state to the defined recovery state. Omitting the implementation of recovery logic may lead to dead locking of the FSM. The recovery logic of FSMs shall is usually implemented with “when others” statement in the FSM VHDL description. Designer can choose the recovery state of the FSM and even define behavior in the unused state (e.g. report SEU error of FSM). In this case, one must be careful and ensure that this logic was not removed in the synthesis as this logic may seem unnecessary to the synthesis tool. [1]

Important is also to choose appropriate encoding of the FSM. The recommended options of FSM encoding are Gray code and one-hot encoding. When SEU is critical to the FSM, one-hot encoding should be used as the SEU always means transition into unused state. The disadvantage of one-hot encoding is higher number of registers. For the Gray code encoding SEU can set the FSM into some other working state and it can be used for FSM where SEU is not critical and registers can be saved. [1]

2.3 DATA PROTECTION METHODS

The SEU can affect also memories and therefore also data in the memories must be protected. In this case TMR can be used again. However using more memories increases costs, weight and size of the board. For error detection, simple methods can be used, for example, the parity bit, checksum or Cyclic Redundancy Check (CRC). These methods are capable only of error detection. It is preferable to use Error Detection And Correction (EDAC) logic, which can detect a limited number of errors in corrupted data and often even correct them. EDAC logic works with the addition of several redundancy bits.

The widespread EDAC logic is Hamming code, which can be designed with different Hamming distances. Hamming distance between two sets of data is a number of positions where these data are different. In other words, it measures the minimum number of changes between these two sets of data. For the EDAC logic, it measures the bit error rate. In general, a code with Hamming distance k can detect $k-1$ errors. Single Error Correction (SEC) Hamming code uses several redundancy parity bits to enable correction of the data. Those bits are computed from particular data bits and are arranged such that incorrect bits produce diverse error result. The combination of errors in parity bits can identify error position in the word and data can be corrected. For example, Hamming code (15, 11) can be used - 11 data bits are protected by 4 redundancy bits. [4]

3 NON-VOLATILE MEMORY MANAGER UNIT

Almost all advanced digital circuit applications require access to data that are typically stored in external Non-Volatile Memories (NVM). Utilization of an NVM memory brings advantage of availability of data in the system right after power-up and flexibility of design even in the late phase of development with simple update of configuration stored in the external NVM.

The proposed architecture of the NVM Manager Unit is intended to autonomously load data from external NVM into an internal memory in FPGA after power on and update data in the external NVM after dedicated command from a control unit is received. Block diagram of proposed unit is shown in Figure 1. Design has been done with respect to the space flight application methodology.

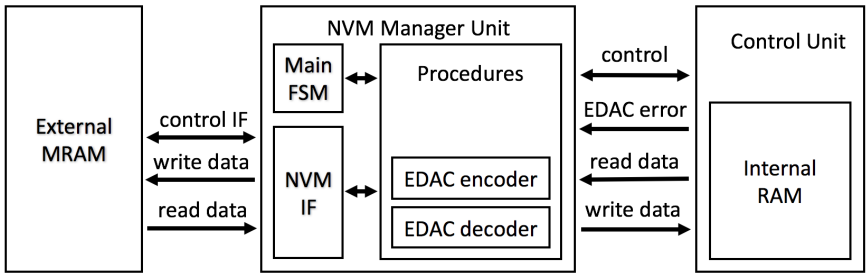

Figure 1: Block diagram of the NVM Manager Unit.

The Magnetoresistive Random Access Memory (MRAM) module MR0A08B from EVERSPIN Technologies [5] has been selected as the external non-volatile memory. The memory is organized in 131,072 words of 8 bits and offers SRAM compatible 35 ns read/write timing with unlimited

endurance. According to the manufacturer description, the MR0A08B is the ideal memory solution for applications that must permanently store and retrieve critical data and programs quickly.

The MRAM memory cells are fault-tolerant to SEUs in general, however radiation can influence the control logic in the memory or data during the manipulation in NVM manager unit. A data protection technique must be included to ensure the correctness of the data on entire data path. Error Detection And Correction (EDAC) logic was chosen to use in this case, concretely the Single-Error Correction and Double Error Detection (SEC-DED) technique [4] using Hamming code with extra parity bits (24, 18). This means that every 18-bit data word is protected by 6 redundant bits and the result (24b) is stored in the memory.

The design was divided into several parts by the function. The base part is the main finite state machine which controls other parts - procedures. All procedures are implemented as one finite state machine which controls accesses to the memories. Last finite state machine is NVM interface which is responsible for access and correct timing to the external NVM. One-hot encoding was selected for all finite state machines and unused states were treated with "when others".

3.1 MAIN FSM

The main finite state machine controls all functionality of NVM manager unit. A part of this FSM is status register RWC which selects actual procedure to be run. FSM diagram of the main FSM is shown in Figure 2. After the reset, state of FSM is set to PWR_OFF and RWC is set to READ. When started by setting the NVM_start, FSM moves to PWR_ON state. In this state counting of 2 ms is started. After the 2 ms, NVM is powered-on and the FSM moves to START_PROC state, where counter for memory address is initialized. Next clock cycle the FSM goes to WAIT_PROC where it is waiting until all data from the external NVM are read. When read procedure finishes, FSM goes to PWR_OFF and set RWC to WRITE.

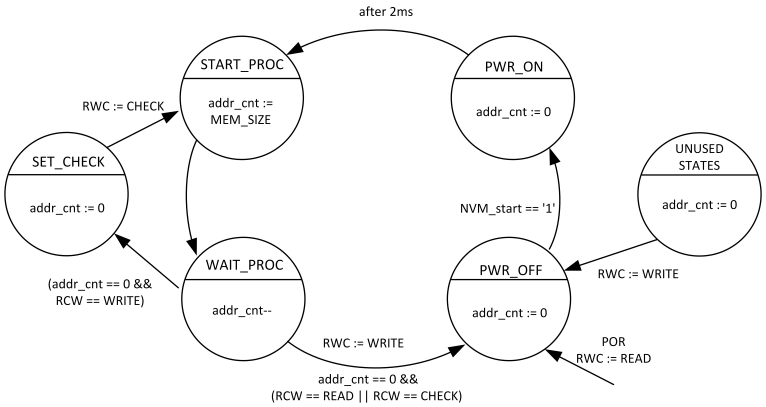

Figure 2: State diagram of the main FSM.

When main FSM has started again, now with RWC set to WRITE, processing of the FSM is similar. In state WAIT_PROC writing into the NVM is done. When writing is finished, FSM moves to state SET_CHECK and set RWC to CHECK. In state WAIT_PROC check procedure is running. CHECK procedure loads written data from the external memory and compares them to the data in internal memory. After finishing CHECK procedure, the FSM goes to PWR_OFF and set RWC to WRITE again.

3.2 PROCEDURES

As mentioned in the previous chapter, NVM manager unit contains 3 procedures: READ, WRITE and CHECK. Since only one procedure is used in the time, all procedures were implemented as one

FSM. Merging the procedures together reduces the number of state registers which reduces the probability of SEU.

The task of the **READ procedure** is reading data from the external NVM, check data in the EDAC decoder and storing encoded data into the internal memory. READ procedure is started only once after reset of the system. The word from the NVM is loaded (via NVM interface) and it is send to EDAC decoder. Decoded data are written into the internal memory. The same procedure is repeated for all data sets. Error flags are generated in case when error is detected or unused state occurs.

The task of the **WRITE procedure** is reading data from the internal memory, encoding using EDAC logic and writing to the external NVM. WRITE procedure is immediately stopped in case of EDAC error detection in internal memory. At the same time error flags are generated. Data from the internal memory are encoded in EDAC logic and encoded data are sequentially written into the NVM.

The task of the **CHECK procedure** is checking that the data written in the external NVM are the same as data in the internal memories. Data are read from both the internal and the external memory. Encoded data from external NVM are compared to the data from the internal memory. When an error in comparison or EDAC error is detected, error flags are set.

3.3 NVM INTERFACE

The NVM interface is a simple finite state machine that controls access to the external NVM and ensures the proper timing of the access. Its separation from the other parts of the design allows easy change of the external memory and therefore changing of a timing of the memory access. The direction of the NVM access is set by RWC state (write for RWC set to WRITE, read for RWC set to READ or CHECK).

4 CONCLUSION

The methodology of design for space flight applications and data protection methods were described in this paper. The architecture of the non-volatile memory manager unit for space applications was presented. This unit was designed to be fault-tolerant to SEU with high reliability. Registers was implemented with TMR protection against the SEU and for all FSMs recovery logic was defined. The EDAC logic (SEC-DED technique) was implemented to correct possible SEU in the memories. When an uncorrectable SEU happens, the error flags are issued. This unit will be verified with respect to the possible SEU and then tested in FPGA.

ACKNOWLEDGEMENT

The article was supported by project no. FEKT-S-17-3934, Utilization of novel findings in micro and nanotechnologies for complex electronic circuits and sensor applications

REFERENCES

[1] Gaisler Research: Lessons Learned from FPGA Developments, Gaisler Research, 2002

[2] Microsemi: RTAX-S/SL FPGAs. Microsemi.com [online]. 2017 [cit. 2017-03-10]. Available from: https://www.microsemi.com/products/fpga-soc/radtolerant-fpgas/rtax-s-sl

[3] Xilinx: Xilinx TMRTool, Xilinx, 2015

[4] TAM, Simon. Single Error Correction and Double Error Detection, Xilinx, 2006

[5] EVERSPIN Technologies: MR0A08 datasheet, EVERSPIN Technologies, 2015