RAM Characteristics Exploration with an MVEDR Model

Fernando Consigli[1], Enrique Gallego[1], Hugo Ramón[1], Horacio Villagarcía Wanza[1, 2]

III LIDI – Facultad de Informática – UNLP[1]
Calle 50 y 120 – 2do piso – (1900) La Plata, Argentina

CIC – Comisión de Investigaciones Científicas Pcia. Bs As.[2]
526 entre 10 y 11 – (1900) La Plata, Argentina

consigli@argentina.com, kike.unlp@gmail.com
{hramon, hvw}@info.unlp.edu.ar

Abstract. Nowadays, motor vehicle safety has gained a widespread interest. Technological breakthroughs have allowed more efficient system implementations as regards life protection elements for both passengers and pedestrians. Among them, Event Data Recorders (EDR) can be found.
In this paper, we present the results of simulations performed with an MVEDR model in certain scenarios.

Keywords: data collection, MVEDR, co-design, SystemC, simulation

1 Introduction

In a previous work [1] we showed how Transaction Level Modeling (TLM) in combination with a hardware description language like SystemC provides a highly flexible methodology for developing a real time embedded system. The system developed consisted of a Motor Vehicle Event Data Recorder (MVEDR) that captures data according to the IEEE Standard for Motor Vehicle Event Data Recorders [2].

An MVEDR is a device that registers car variables in real time as an accident takes place in order to obtain crash event data. This device is in charge of storing information that can be employed to identify accident causes and rebuilt it to analyze it [3].

Many constraints such as memory size, processing capacity, communication requirements and regulations were taken into account during the construction of the model.

The usage of TLM and SystemC made it possible to test and tune the model without the need of constructing a full implementation of the whole system.

Once the model was set up, it was executed with different configurations over three scenarios in order to test extreme cases.

In section 2, we briefly describe our MVEDR architecture, its components and details of its operation. In section 3, we start by depicting how much data is collected, and what percentage is given to every sensor. Then, we present our three test scenarios, explaining each of them and detailing percentages of data lost in every case. Finally, we show the obtained results. In section 4, we give our conclusions on the matter.

2 MVEDR architecture

The architecture for the data acquisition device (Fig. 1) is represented with SystemC syntax [4].

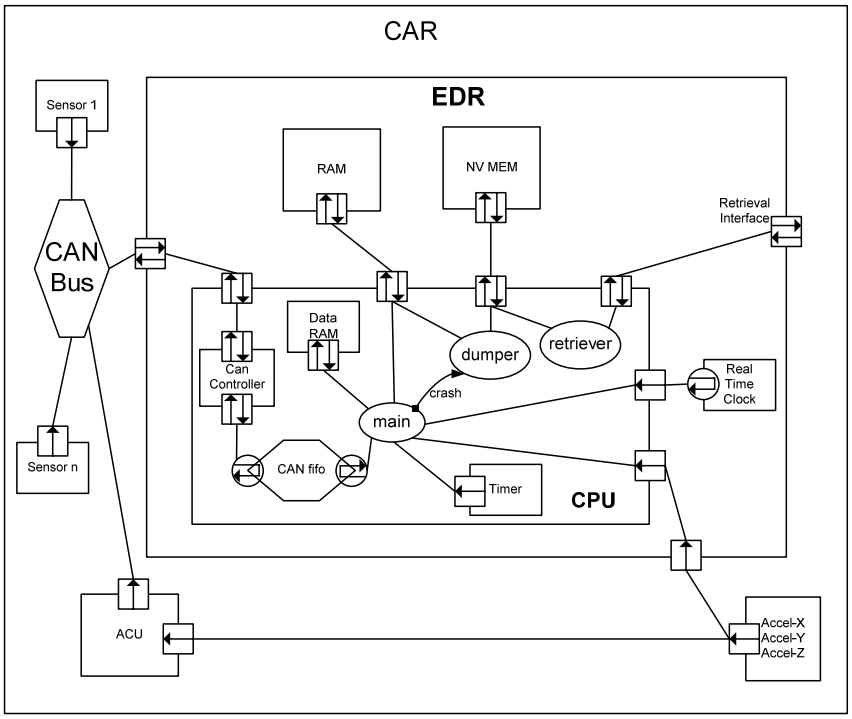

Figure 1: MVEDR Architecture

The presented device contains a CPU, a volatile RAM, a non-volatile memory, an accelerometer and a real time clock. It is connected to a CAN bus and it also provides an interface to retrieve the acquired data.

The characteristics of the model, such as CPU speed, size and speed of both RAM and non-volatile memory and CAN buffer size are parameterized so as to test different configurations of hardware.

2.1 Operation

MVEDR operates as the engine runs. Sensors in the car sample information and send it to the MVEDR through the CAN bus channel. Inside the MVEDR, the data is stored in circular buffers in RAM. Besides, the MVEDR is wired to an accelerometer from which receives acceleration data used to determine the occurrence of an accident. When the sensed acceleration exceeds the threshold established by the IEEE Standard, the information stored in RAM is dumped to the non-volatile memory.

Finally, saved data can be downloaded through the retrieval interface.

3 Simulations

The IEEE Standard states that any compliant MVEDR should be capable of capturing up to three events in a multi-event crash. For any given event that generates a change in velocity that equals or exceeds the trigger threshold, the MVEDR is required to record and capture that event and any subsequent events, up to a total of three, that begin within a 5 second window from time zero of the first event.

Subsequent events are events that meet the trigger threshold more than 500 ms after the immediately preceding event. To prevent unassociated events from being captured, the maximum time from the beginning of the first event to the beginning of the third one shall be limited to 5 seconds.

The MVEDR captures data from many sensors (Fig. 2). Each accelerometer sends 1000 samples per second. From each simulation the percentage of data lost is obtained. Considering that 600 samples is the minimum requirement for simple reconstruction [5], given a certain scenario, a simulation with a hardware configuration is considered to be successful if the dump process finishes having lost less than 40% of the accelerometer data.

Data loss can occur because of insufficient CAN buffer size, not enough CPU processing power, high memory latencies or a combination of all these factors.

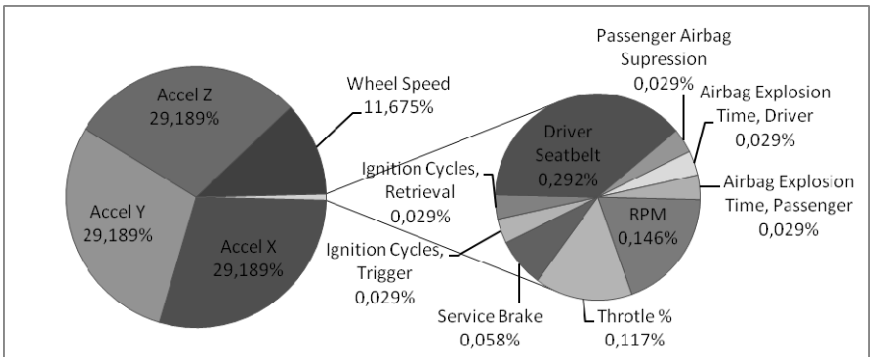

Figure 2: Data captured by EDR per second

3.1 Simulated scenarios description

Our model behaves as described; hence, scenarios were specially designed to test extreme cases.

The first scenario consists of a single-event crash in which there is only one collision. The second scenario is a multi-event crash in which the car is hit three times within a period of time of 5 seconds and there are more than 500 milliseconds between each hit. Finally, the third scenario extends the second one by adding an extra event 6 seconds after the first collision, which is considered as a different crash.

Scenario 1

In this case, as only one event occurs, the main aspect tested is the ability to capture data from sensors and to dump it from RAM to the non-volatile memory (Fig. 3). Therefore the amount of data to be dumped is the smallest possible.

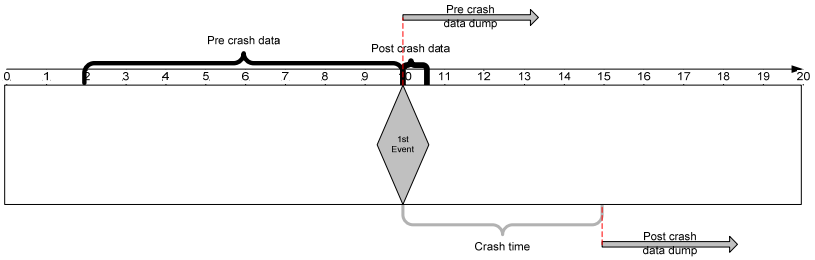

Figure 3: Scenario 1

Given the obtained results for scenario 1 (Fig. 7), it can be seen that when the system is configured with a low performance non-volatile memory, i.e. an EEPROM, it is not possible to achieve a tolerable data loss when write time is greater than 140μs per word. This is due to the fact that the pre-crash data dumping is performed while post-crash data is being captured. The later will also be subsequently stored in permanent memory. As the CPU gets locked due to memory writes, it cannot process CAN packages or accelerometer data.

One solution to this problem, partially deviating from the IEEE Standard, is to delay the dumping of data, both pre and post crash, until it is over (Fig. 4). According to the IEEE Standard, crashes can last up to five seconds. If the black box is configured to start dumping immediately after that period of time, there is no data loss but the whole process of dumping takes more time than before. In addition, it will be required a larger amount of RAM memory to prevent data in circular buffers from getting corrupted.

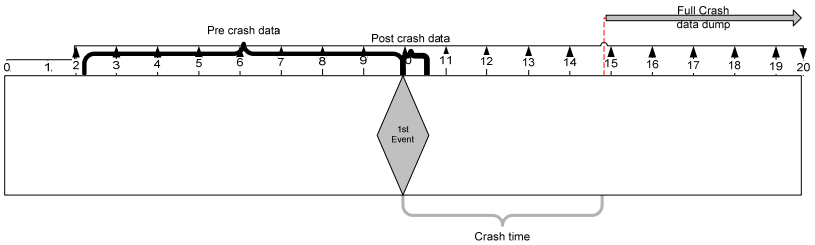

Figure 4: Scenario 1 with delayed dump

Another option would be to arrange the functionality of the black box to dump information only from one event, ignoring what the IEEE determines. In that case it can be set up to start the data dumping 500 milliseconds after the event, which is the time in which the capture of data required for accelerometers is completed. This way, the elapsed time since the occurrence of the event until the end of the dump process is less than in the previous cases.

Scenario 2

This scenario is the worst case of a multi-event crash (Fig. 5). Not only does it simulate the occurrence of three events, but also the elapsed time between the first and the third one (4950 ms) is almost the maximum admitted by the IEEE Standard (5000 ms). This way the amount of information to be saved in permanent memory is maximized. This includes data received between the first and the last event plus those received before the first and after the last one according to the sampling periods set by the IEEE Standard.

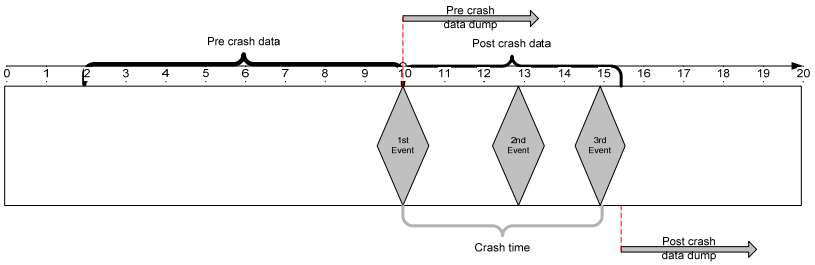

Figure 5: Scenario 2

Similarly to what happens on the first scenario, if the black box is configured with a low speed non-volatile memory, most of the data is lost. However, as in this scenario the volume of data to be stored in non-volatile memory is much higher, if the box is configured to start the data dump at the end of the crash, this process will require between 20 and 24 seconds and between 256kb and 512kb of RAM depending on the speed of the CPU to copy all the data to permanent memory.

On the other hand, if the box is configured with a faster non-volatile memory, it takes only between 64kb to 128kb of RAM depending on the CPU processing speed. With a 10Mhz CPU frequency, significant data loss (about 10%) is produced. With a 25Mhz CPU it decreases to as much as 0% data loss.

Scenario 3

In this case, it represents a similar scenario to the previous one, with the difference that airbags are triggered on a second crash immediately after the first one concluded (Fig. 6).

For the aforementioned reason, as provided by the IEEE Standard, data from the first collision is to be discarded and only that from the second one should be kept.

The main problem here is that pre-crash data from the second crash, which should be kept in permanent memory, is captured while performing data dump from the first crash.

Thus, not only post-crash data is compromised as in scenario 2, but it could also be a loss of pre-crash data.

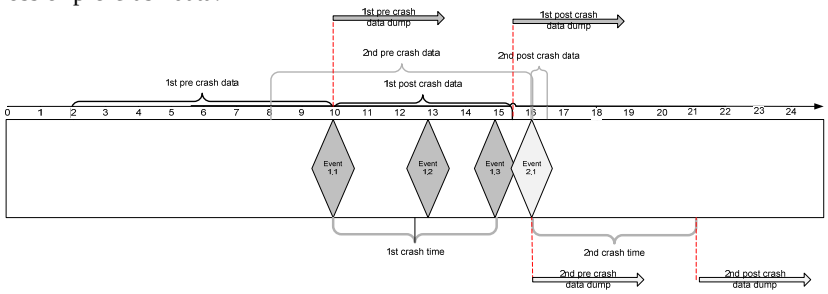

Figure 6: Scenario 3

In this case, delaying the dump process is not an option because there is always the possibility that a new crash occurs. That would cause pre-crash data to overlap with the previous data dump.

3.2 Simulations results

The results shown in the chart (Fig. 7) are from simulations that were configured to run with a CPU speed of 50MHz and a RAM memory size of 128Kb. The non-volatile memory write time per word is the variable on the horizontal axis.

It is worth to mention that much more simulations with different CPU speeds and memory sizes were run to support conclusions.

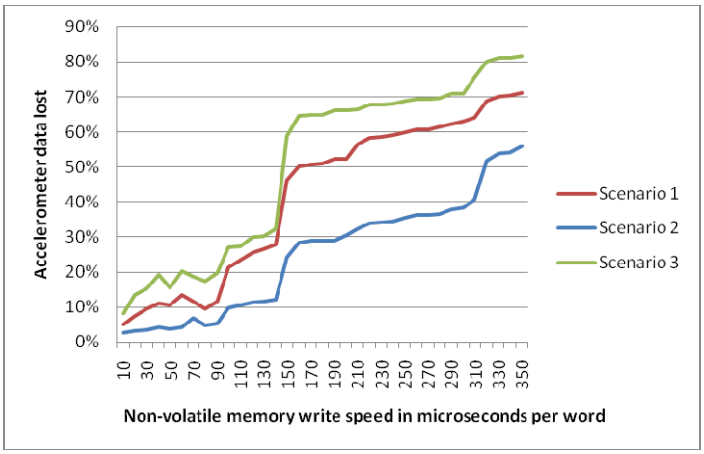

Figure 7: Acceleration data lost

The fact that the percentage of data lost in scenario 2 is lower than in the other scenarios is because the interval in which data is captured in this scenario is longer. Thus, the time in which the data dumping process and the capturing process overlap is proportionally smaller.

Moreover, although the percentage of data lost is the lowest, the amount of data lost in bytes is significantly higher.

4 Conclusions

After analyzing simulation results in the presented scenarios, we can affirm that for most cases, to accomplish the process of dumping information from RAM memory to permanent memory in the shortest time possible and without losses, it is necessary to have a non-volatile memory significantly faster than the traditional EEPROMs.

However, if for economic reasons EEPROMs are employed in the design of a system, the start of the dump process can be delayed to reduce data loss, taking the risk of losing data in case of failure of power supply. In addition, there may be loss of data in an accident involving more than three events in more than one crash, as in scenario 3, although this type of accident is unlikely [6].

In contrast, if the system is built with a high speed non-volatile memory, such as ferroelectric RAM (F-RAM) [7], CPU frequency becomes the most influential element regarding the percentage of data lost. According to the information obtained from simulations, the maximum write speed with which the amount of data loss is tolerable is 140µs per word.

References

1. Consigli F., Gallego E., H. Ramón, H. Villagarcía Wanza. *Description of an Architecture for Critical Distributed and Real-Time Data Collection Applied to MVEDR* Actas del XV Congreso Argentino de Ciencias de la Computación, Sección WARSO Trabajo 2842. 2009.

2. IEEE. *IEEE Standard 1616 – 2004*: Standard for Motor Vehicle Event Data Recorders (MVEDRs).

3. Ching-Yao Chan. *Trends in Crash Detection and Occupant Restraint Technology*. Proceedings of the IEEE, vol. 95, no. 2, pp 388-396 (2007)

4. David C. Black, Jack Donovan. *SystemC: From the Ground Up*. Kluwer Academic Publishers, Boston (2004)

5. NHTS.: *Summary of Finding.s* NHTSA EDR Working Group. Volume II. 2002.

6. NHTSA: *NASS/CDS database guide*. 2000.

7. Ramtron. Ramtron Products - Nonvolatile Memory. Web. 2010. http://www.ramtron.com/products/nonvolatile-memory/.