Using Bulk Built-In Current Sensors and Recomputing Techniques to Mitigate Transient Faults in Microprocessors

Franco Leite[1], Tiago Balen[1], Marcos Hervé[2], Marcelo Lubaszewski[1], Gilson Wirth[1]

[1]Departamento de Engenharia Elétrica, [2]Instituto de Informática
Universidade Federal do Rio Grande do Sul - Porto Alegre – RS, Brazil

ABSTRACT

This work presents the application of a recomputing-based correction technique to mitigate radiation effects on integrated processors. The recomputing process is associated to Bulk Built-In Current Sensors (BICS) capable of detecting variations in the bulk current due to a particle strike in the circuit silicon area. An 8051 microprocessor is considered as case study. This work focuses on the mitigation of Single Event Transient (SET) faults affecting the execution of the microcontroller instructions. VHDL descriptions of the microcontroller and of the bulk BICS are simulated and results show that recomputing the instruction, when the BICS indicates a particle strike, is an efficient way to prevent processing errors. The resulting SET-resistant microcontroller presents low area and performance overheads.

1. INTRODUCTION

The incidence of cosmic radiation in integrated circuits can affect the correct functioning of the system in which it is inserted. This radiation is originated mainly from solar activity and can be divided in electrical charged particles, like protons and heavy ions, and electromagnetic radiation [1].

With the microelectronics technology scaling, circuits are more vulnerable to this effect, since transistors dimensions shrink as the technology evolves and, consequently, lower amounts of charge are needed to change the state of a transistor. Although this effect is more likely to occur at high altitudes (in avionic and space applications, for example), it can also affect modern circuits operating at the sea level [2].

Microcontrollers and microprocessors are widely used in several applications, including avionics and space ones, in which the incidence of radiation or electrical particles is more intense. For this reason, it is desirable that the microprocessors employed in such applications embed some kind of fault tolerance mechanism, to avoid computing errors, potentially caused by Single Event Upset (SEU) and Single Event Transient (SET).

Previous works have described the use of ECC (Error Correction Codes) to protect microprocessors memory against Single Event Upset (SEU) [2]. This work focuses on the mitigation of SET faults affecting the execution of microprocessor instructions. The technique proposed in this paper consists in a recomputing-based fault tolerance mechanism, associated to the use of Bulk Built-In Current Sensors (BICS) [3]. The bulk current sensor provides a flag signal to the control part of the microprocessor, which indicates a possible strike of an electrical particle. This signal is used to start the recomputing process in a way that a possible corruption in the instruction being executed by the microprocessor (possibly caused by a SET) is corrected.

In order to validate the technique, an 8051 microcontroller is considered as case study. For this purpose, a VHDL description of such device is used and RTL simulations are performed.

The rest of this work is organized as follows: in section 2 some related works are presented. Section 3 describes the proposed recomputing scheme, while section 4 shows the modifications implemented at the 8051 control part. Results are shown in section 5 and section 6 concludes the work.

2. RELATED WORK

Single Event Upset (SEU) and Single Event Transient (SET), as well as their effects on digital circuits, have been widely studied from more than a decade and several techniques to implement fault tolerant digital circuits have been devised, for example [2, 4]. All techniques are based on redundancy, in different levels. The redundancy can be classified in terms of area (spatial) or time (temporal). Next, some related works that propose the use of redundancy schemes are discussed. Previous works that proposed and used bulk BICS are also discussed.

2.1 Area Redundancy

A well-known hardware fault tolerance technique is the Triple Modular Redundancy (TMR) [4]. This technique consists in triplicating the circuit logic and using a majority voter. Considering a single fault hypothesis, if a fault affects one of the replicas of the logic (changing the value of the signal delivered to the voter) the other two voter inputs are correct and their values are passed to the voter output. The weak points of this technique are the high area overhead (at least 200%, considering the replicated logic) and the vulnerability to multiple faults in different paths [4]. Additionally, faults can also occur in the voter, hence, to guarantee the reliability of TMR schemes the voter must be fault tolerant as well.

Other levels of redundancy can also be employed in fault tolerant schemes. By duplicating the circuit a single fault can be detected but not corrected. Quadruplicating the circuit or by using even higher degrees of redundancy can also mitigate multiple faults [4].

Several techniques to achieve protection in RAM-type memories have been studied in last years since, contrarily to Flash and ROM memories, RAM memories are sensitive to SEE (Single Event Effects) [4]. Some techniques to protect RAM memories are based on correction codes, as Hamming code, for example [2]. By using such schemes, BIT inversions in the memory content can be detected and corrected. The use of correction codes also implies in area overhead since the number of BITs of the memory elements is increased.

2.2 Time Redundancy

Recomputing techniques are mainly based on time redundancy, although some degree of area redundancy is often required. In such schemes, a given operation is recomputed multiple times, and the results of these computations are compared in order to detect faults. Recomputing can be performed at operator level (low-level recomputing) or at algorithm level [4].

Partial recomputing can also be used to reduce the time overhead. In this technique some instructions are recomputed while others are not, this way the program execution delay is reduced [4]. Also in [4] the dynamic operator allocation is proposed, i.e., the order of mathematical operator during the recomputing phase is different from the order employed in the first time computing. This avoids that permanent faults remain undetected, since such faults may affect the final result in different ways if one compares the computing to the recomputing process.

2.3 The Bulk Built-In Current Sensor

The use of Bulk Built-In Current Sensors (bulk-BICs) to detect transient errors was proposed in [3]. Its functioning is based on the detection of bulk current, caused by the strike of ionizing particles. In this work the design of the sensor itself is not considered, since it is well described in previous works [3].

The bulk-BICS analyzes the current that appears at the bulk terminal. During normal operation, the current in the bulk is approximately zero. Only the leakage current flows through the biased junction, which is still very low compared to the current generated by energetic particles. So, when an energetic particle generates a current in the bulk, it is very clear to the bulk-BICS that a SET has happened [3]. The bulk-BICS generates thus an asynchronous signal indicating that a current peak in the IC Bulk was detected.

Once a SET is detected, the bulk-BICS output that signals the occurrence of a SET is activated, and remains in this state until the reset input is activated. The reset signal must be activated by the recomputing system.

3. MITIGATING TRANSIENTS IN MICROPROCESSORS: 8051 CASE STUDY

This work proposes the use of bulk-BICS to detect the incidence of radiation in integrated microprocessors. An 8051 microcontroller is considered as case study. If a transient current pulse is detected, the microcontroller must repeat the reading and execution of the instruction being processed, in order to avoid an erroneous computing.

By the time of the current pulse detection, the data cannot be stored in any memory element, since it may be corrupted. Therefore, in order to re-execute the last instruction, the program counter (PC) must be stopped or decremented. For this reason, the microcontroller control part must be modified. After the re-execution, the microcontroller sends a reset signal to the BICS, and if no more current activity is signaled by the sensors, the microcontroller continues its normal operation.

In this work a VHDL description of an 8051 microcontroller is used. The considered description [5] presents some enhancements if compared to the 8051 original architecture and control procedures. In classical 8051 devices the instruction execution takes 12 or 24 clock cycles. However, in the architecture described in [5] the execution of the instructions is performed in less clock cycles, as showed below:

- Instructions like INC_A, that comprise only the FETCH state take one clock cycle.
- Instructions like ADD, that have the FETCH and EXEC1 states need two clock cycles.
- Instructions like LJMP, that comprise also the EXEC2 state take 3 clock cycles.
- Instructions like INC_DPTR and XCH_A_D, which have the EXEC3 state, need four clock cycles to accomplish its execution.

In this work the VHDL description presented in [5] is modified in order to add the fault tolerance functionalities to the 8051 core. Descriptions and simulations are performed by using the ModelSim tool.

Besides the 8051, a simplified description of the bulk-BICS is also used to perform the simulation. In this description a process that generates a transient pulse is inserted, in order to simulate the detection of radiation incidence. Figure 1 shows the VHDL BICS description.

```
architecture rtl of BICS is

signal s_radiacao : std_logic := '0';
signal s_particula : std_logic := '0';

begin                -- architecture structural

set_reset_sensor: process (s_particula,resetbics2)

  begin
    if s_particula = '1' then
        s_radiacao <= '1';
    else
      if resetbics2 = '1' then
        s_radiacao <= '0';
      end if;
    end if;
 end process set_reset_sensor;

 p_tb_bics : process

 begin
    s_particula <= '0';
    wait for one_period*16.9;
    s_particula <= '1';
    wait for one_period*0.2;
    s_particula <= '0';
    wait;
 end process p_tb_bics;

radiacao <= s_radiacao;

end rtl;
```

Figure 1: VHDL description of the bulk- BICS

The process *p_tb_bics* modifies the signal *s_particula* and can simulate the radiation incidence at any time and with any duration. When the signal *s_particula* goes high the bulk-BICS logic generates a logic "1" at the signal *radiacao*, indicating that a transient pulse at the bulk was detected. This signal is only reset if the signal *s_particula* returns to zero and if the signal *resetbics2* is high. In this work the signal *resetbics2* is provided by the microcontroller and indicates that the recomputing process is finished.

4. MODIFICATIONS IN THE 8051 CONTROL PART

The recomputing process consists in interrupting the PC if a transient pulse is detected. Then, the PC must be decremented in some positions in order to re-execute the instruction that was under execution at the time of the transient pulse detection.

To comply with the proposed fault tolerance scheme the control part of the microcontroller must be modified. These modifications are inserted in the FSM (Finite State Machine) and also into the memory and registers control logic. In the VHDL description used in this work the FSM and memory controls are divided in two distinct files.

The signal *s_pc_inc_en* is responsible for enabling the PC increment. If this signal is "0" then the PC is not incremented, otherwise, if this signal is "1" then the PC can be incremented in the next clock rising-edge. This signal is crucial to the recomputing process.

Concerning the instructions with more than one execution cycle the radiation incidence may occur between the FETCH and EXEC1 states. In this case the recomputing can be performed in two different ways: simply stopping the PC or decrementing it, in such a way that the entire instruction is re-processed. Considering the 2-cycle instructions, the former option is preferred in this implementation because in the first state (FETCH) there are no control signals to corrupt. If some signal is corrupted in the EXEC1 state it is automatically recomputed, since this is the last state of the instruction execution.

For the 3-cycle instructions, if the SET occurs between the EXEC1 and EXEC2 states, it is possible that the control signals of EXEC1 are corrupted. In this particular case it is mandatory to decrement the PC, in order to return to the EXEC1 state. An example is shown in Figure 2 considering the modified long jump instruction with a 16 BIT address as reference.

```
when IC_LJMP =>                -- LJMP addr16
  if state=FETCH then
    if s_radiacao = '0' then
       s_pc_inc_en <= "0001";     -- increment program-counter
       s_nextstate <= EXEC1;
    else
       s_pc_inc_en <= "0000";
       s_nextstate <= FETCH;
    end if;
  elsif state=EXEC1 then
    s_help_en <= "0001";       -- help = rom_data_i
    if s_radiacao = '0' then
       s_pc_inc_en <= "0001";     -- increment program-counter
       s_nextstate <= EXEC2;
    else
       s_pc_inc_en <= "0000";
       s_nextstate <= EXEC1;
    end if;
  elsif state=EXEC2 then
    if s_radiacao = '0' then
       s_pc_inc_en <= "0111";     -- load program-counter
       s_nextstate <= FETCH;
    else                           -- returns to EXEC1
       s_pc_inc_en <= "1111";    -- PC = PC-1
       s_nextstate <= EXEC1;
    end if;
  end if;
```

Figure 2: Modification in a 3-cycle instruction: the example of a long jump

A necessary modification to implement is the possibility of decrementing the PC, since this functionality is not present at the 8051 original architecture. This is implemented by changing the microcontroller memory control part.

Besides the architecture modifications detailed above, the memory writing logic is also modified. This is necessary to avoid the storing of a potentially corrupted data if a SET is detected.

Finally, another modification at the memory control part is the insertion of the signal *resetbics2* responsible for resetting the current sensor after the recomputing process finishes. This signal is synchronous with the next state logic of the control part. It is necessary to guarantee that the re-execution process has finished before the transition to the next state.

5. EXPERIMENTAL RESULTS

Several simulations were performed considering the modified architecture of the 8051 microcontroller. Results showed that every time a transient is detected by the bulk-BICS the recomputing process is started. The program counter is stopped or decremented if necessary and the instruction is recomputed.

Figures 3 and 4 show a simulation in which a simple program is loaded into the 8051 microcontroller. This program executes five accumulator increments, sums 20 to the ACC and returns to the third line. These Figures focus on the "long jump" instruction. Figure 3 shows the simulation considering no SET occurrence, while Figure 4 shows the simulation considering a transient fault detection. One can see in Figure 4 that after the detection of the transient fault (during the EXEC2 state) the control FSM returns to the EXEC1 state. Additionally, the PC is decremented (14 to 13) and the bulk BICS is reset. This scheme assures that a detected transient fault will not affect the program normal execution.

Simulations concerning the area overhead were also performed. For this purpose the Synopsis tool "Design Vision" was used. The synthesis report showed that the synthesized original architecture comprises 6645 logic cells. The modified description with the fault tolerance functionalities takes 6662 logic cells, representing an overhead of 0.26%. It is important to point out that this overhead is calculated considering only the control part modifications.

If the memory protection (by using ECC) is also considered, the area overhead will be higher. In [2] the area overhead achieved by using Hamming code to prevent SEU faults in an 8051 core was 46%. Additionally, previous works that studied the use of bulk BICS showed that for logic circuits with relatively high complexity (which is the case of the 8051 core) the area overhead due to the implementation of bulk BICS is about 10% to 15% [3]. Therefore, the overall overhead achieved in a fully protected 8051 core, when using this approach, can be estimated as 60% to 70%.

The timing reports resulting from the synthesis of the VHDL descriptions showed that the original and the modified circuits have a maximum operation frequency of 44 MHz and 40.2MHz respectively. This represents that the modified circuit is 8.5% slower than the original microcontroller. As reported in [2], if the ECC memory protection is also considered, a careful VHDL description of

the fault-tolerance mechanisms can lead to no performance impact in the final microcontroller architecture. Additionally, previous works on the use of bulk-BICS have also shown that they have negligible impact on the circuit timing performance [3].

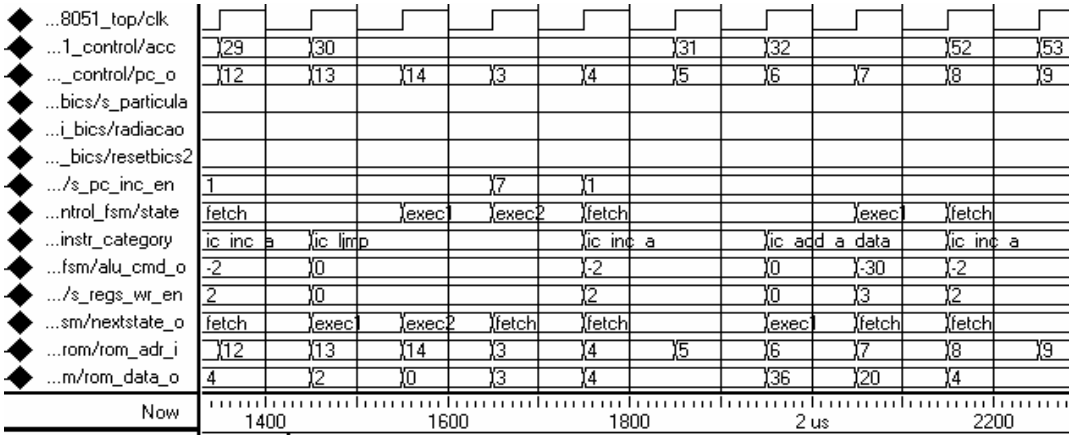

Figure 3: Test program execution without transient fault

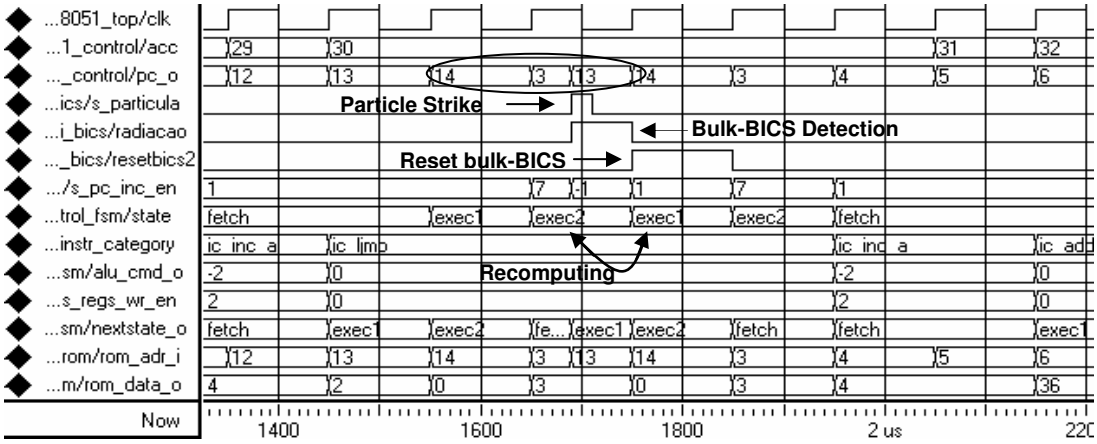

Figure 4: Test program execution simulating a transient fault during the "EXEC2" state of a "long jump" instruction

6. CONCLUSIONS

This work focuses the protection of integrated microprocessors in respect to the execution of instructions. The protection of the memory elements can be made by using error correction codes, as proposed in previous works.

The detection of the transient pulse is achieved by means of integrated bulk built-in current sensors (bulk-BICS). When the microcontroller receives the BICS indication, a recomputing process is started in order to avoid the processing of potentially corrupted data.

An 8051 microcontroller is considered as case study. The main modifications are made on the program counter control (allowing it to be decremented) and also in the memory writing control logic, in order to avoid the storing of corrupted data.

Simulation results show that radiation effects affecting the execution of microcontroller instructions can be eliminated with little processing time. Synthesis and simulations also show that low area and performance overheads are imposed by such scheme, which indicates the feasibility of the technique. Additionally, gathering previous works results with the ones presented in this paper, it is possible to achieve a full SET and SEU protected 8051 with significant reduction in the area overhead if compared to TMR solutions.

7. REFERENCES

[1] Ziegler, James F. *Terrestrial Cosmic Rays*. IBM Research Division, Thomas J. Watson Research Center, P. 0. Box 218, Yorlcown Heights, New York 10598. Volume 40 nº 1, Janeiro de 1996.

[2] Kastensmidt, Fernanda. L.; Cota, Érika ; Rezgui, Sana; Carro, Luigi; Velazco, Raoul; Lubaszewski, Marcelo; Reis, Ricardo. *Synthesis of an 8051-Like Micro-Controller Tolerant to Transient Faults*. Journal of Electronic Testing: Theory and Applications 17, 149-161, 2001.

[3] Neto, Egas H.; Ribeiro, Ivandro; Vieira, Michele; Wirth, Gilson; Kastensmidt, Fernanda L.; Carro, Luigi. *Using Built-in Sensors to Cope with Long Duration Transient Faults in Future Technologies.*

[4] Wu, Kaijie; Karri, Ramesh. *Algorithm Level Recomputing Using Allocation Diversity: A Register Transfer Level Approach to Time Redundancy-Based Concurrent Error Detection.* IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems, Vol. 21, No. 9, September 2002, P.1077-1087.

[5] Oregano Systems – Design and Consulting web page. http://www.oregano.at/ip/8051.htm, accessed in march, 2007.