VHDL Simulation considering Single Event Upsets (SEUs)

M. Grecki

Department of Microelectronics and Computer Science
Technical University of Lodz, Poland, grecki@dmcs.pl

ABSTRACT

This paper presents the methods enabling VHDL simulation of digital circuits taking into account SEUs generation and propagation. The way of the VHDL model conversion and the SEUSIM software library that allows simulating the operation of digital circuits together with SEUs impact is described. The SEUSIM library is written in pure VHDL so it can be used by most VHDL simulators including ModelSim, Active-HDL and open source project gHDL. The worked out methods and tools are applied to the simulation of control system of neutron radiation monitoring device. The results of simulation are compared with measurements of real circuit exposed to neutron radiation.

Keywords: VHDL, simulation, SEU, FPGA, radiation influence

1 INTRODUCTION

The SEEs (Single Event Effects) are caused by elementary particle interaction (ionization) with semiconductor structure [1][2][3]. They can be destructive for semiconductor devices (Single Event Gate Rupture, Single Event Burnout, Single Events Latchup), however non-destructive SEEs called SEU (Single Event Upsets) resulting of bit flips in flip-flops and memory cells seem to be the main problem in forthcoming years [4][5]. The SEU hazard is approximated to rise up about ten times every five years, then much faster than complexity of the integrated circuits (two times every two years according to the Moore's law). Due to reduction of feature size the SEUs play an increasing role in failures observed during operation of digital circuits, particularly in environments with remarkable radiation level (avionics, nuclear industry, High Energy Physics instrumentation etc.). In a few years SEUs will significantly affect the digital electronics not only in these special radiation environment but at the normal operating conditions as well. Therefore the countermeasures against SEUs gains importance as technology feature size drops down. The electronic system can be build using radiation-hardened semiconductor devices [6] but they skyrocket the system cost [7]. Methods that allow the application of commercial off-the-shelf (COTS) electronic components seem to be much more adequate for commercial electronics. Those methods are based on hardware and software redundancy that allows detecting and correct radiation influenced SEUs during system operation, making such a system radiation-tolerate.

The design of radiation-tolerate systems require the simulation tools that can predict the behavior of radiation exposed circuits. The interactions between elementary particles and semiconductor crystal, that is a source of SEUs, have a random nature described by exponential distribution of density of probability. The simulation of SEUs is usually performed on microscopic level with application of physical based, multi dimensional models of semiconductor device [1][8][9][10]. Such tools are very valuable when the device is optimized for radiation hardness in the factory but are useless for the designers of FPGA based and microprocessor systems. Those designers need tools that can simulate the behavior of the whole system [11]. For this purpose the specialized languages for description of digital circuits (like VHDL and Verilog) are commonly used. . On the other hand the widely used simulators for simulation of digital circuits (ModelSim, Active-HDL, gHDL) are purely deterministic, their operation is based on evaluation of signals and scheduling events.

Therefore the method for simulation of the digital whole system together with SEU generation using common languages and simulation tools is required.

2 VHDL MODEL OF SEU

The occurrence of SEU is a random process described by exponential density probability (1). The equation (1) allows calculating the probability of SEU occurrence during given time t.

$$p(t) = \frac{1}{\tau} e^{-\frac{t}{\tau}} \qquad F(t) = \int_0^t p(t)dt = 1 - e^{-\frac{t}{\tau}} \qquad (1)$$

The τ parameter is a time after that the probability of SEU occurrence is $F(\tau) \cong 0.632$. It depends on kind of particle and particle energy. It can be estimated for given semiconductor technology on the base of measurements in destination environment. The fig. 1 presents the histogram of SEU frequency in 1MB SRAM memory exposed to the constant neutron flux coming from

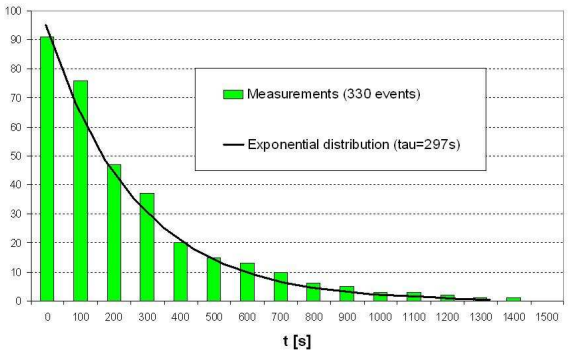

Figure 1: Histogram of SEU occurrence frequency in SRAM memory (1Mb) exposed to ^{241}AmBe source

^{241}AmBe source. The τ estimated from this data for the whole memory is $\tau \cong 297s$.

Using random numbers generator it is possible to generate SEUs in VHDL program in a stochastic way. Having the probabilistic variable *random* with uniform distribution of probability density in the range of $(0, 1)$ the SEU can be drown from condition (2).

$$random < F(t) \tag{2}$$

The fulfilled condition (2) generates SEU during the time period t. However the direct VHDL implementation of equation (1) can not be possible, since not all VHDL simulators distributes *ieee.math_real* library that implements function *random* (*random* is called $uniform$ in the *ieee.math_real* library) and e^x.

The SEUSIM library implements two different random number generators. First one is a pseudo-random generator based on RANROT W [12] program translated to VHDL function. This random number generator has 63 bits resolution and uniform distribution. It has to be initiated by *seed*. Initialization by the same *seed* generates the same pseudo-random sequence helping to debug the system. The second random number generator uses Linux device */dev/random* to get randomness. This method is very fast and really random in contrast to the first one. The exponential function in (1) can be substituted in VHDL by simpler to implement function (3).

$$F(t) = 1 - \frac{1}{1 + \frac{t}{\tau} + \frac{1}{2}\frac{t^2}{\tau^2} + \frac{1}{6}\frac{t^3}{\tau^3}} \tag{3}$$

The accuracy of the approximation (3) is better than 0.5% for all values of t. For typical operating conditions (when the operation cycle t of the circuit is much shorter than τ) the equation (3) can be even further simplified to the form (4).

$$F(t) = 1 - \frac{1}{1 + \frac{t}{\tau}} = \frac{t}{t + \tau} \tag{4}$$

The SEUSIM library includes function *readSEU* that draws the SEU occurrence for signals of *std_logic_vector* type. The *readSEU* function is called during each readout of the signal. It scans the signal history and computes time duration from the last access to the signal. This time is used in condition (2) and if SEU was generated during this time period the *readSEU* function returns SEU mask (type of *std_logic_vector* with randomly set single bit). The non-zero SEU mask value changes the signal value (flips single bit). This causes the signals updates can happen in all parts of the model, not only in the update process. Therefore the VHDL model considering SEUs has to be converted to the form allowing multiple-source signal updates.

3 VHDL MODEL CONVERSION

The VHDL converted model uses shared variables to solve the problem of multiple-source signal updates. Every signal has corresponding variable that is updated instead of the signal during readout or write. After each update of the variable the special process is activated (*seu_update*) that updates signals value as well. The example of VHDL source conversion for simple case of file of four working registers that can be written and read is presented in fig. 2.

The VHDL model conversion can be done automatically however the preprocessor is not finished yet. It is under development using open-source VHDL parser worked out by D.Scarpazza [13].

4 SIMULATION EXAMPLE

The simulation example concerns the control system applied in RadMon neutron detector (fig.3). The detector uses Static Random Access Memory (SRAM) as a neutron sensor. The SEUs in SRAM are registered and counted by control system. The operation of the control system should be failure-free, however short breaks in system operation are allowed. Therefore the control system uses double modular redundancy SRAM readout system connected by RS232 line to the PC computer placed in radiation free environment. Both modules of readout circuit operate in parallel and external circuit supervises their synchronous operation. When non-synchronous operation is detected external host computer restarts the readout system. The control circuit was realized in flash-based Actel APA300 FPGA. The results of simulation are presented in fig. 4 and fig. 5. The fig. 4 presents the detection of SEU in the SRAM. After detection the SEU (different values in *va* i *vb* registers) the memory contents is corrected during WRITE cycle (mem_wr signal goes active). The fig. 5 presents the distribution of the time between successive SEUs in SRAM sensor. It corresponds well to the theoretical distribution.

Oryginal

```
subtype byte is std_logic_vector(7 downto 0);

ENTITY reg_file IS
  PORT (
    clock    : IN  STD_LOGIC;
    write_en : IN  STD_LOGIC;
    out_en   : IN  STD_LOGIC;
    address  : IN  STD_LOGIC_VECTOR(4 DOWNTO 0);
    data_in  : IN  byte;
    data_out : OUT byte
    );
END reg_file;

ARCHITECTURE mixed OF reg_file IS
SIGNAL r8  :  byte;
SIGNAL r9  :  byte;
SIGNAL r10 :  byte;
SIGNAL r11 :  byte;

BEGIN
  data_out<=r8  WHEN address="01000" AND out_en='1' ELSE "ZZZZZZZZ";
  data_out<=r9  WHEN address="01001" AND out_en='1' ELSE "ZZZZZZZZ";
  data_out<=r10 WHEN address="01010" AND out_en='1' ELSE "ZZZZZZZZ";
  data_out<=r11 WHEN address="01011" AND out_en='1' ELSE "ZZZZZZZZ";

  write: PROCESS (clock)
    BEGIN
      IF clock'EVENT AND clock = '1' THEN
        IF write_en = '1' THEN
          case address IS
            WHEN "01000" => r8  <= data_in;
            WHEN "01001" => r9  <= data_in;
            WHEN "01010" => r10 <= data_in;
            WHEN "01011" => r11 <= data_in;
            WHEN OTHERS => NULL;
          END CASE;
        END IF;
      END IF;
    END PROCESS;
END mixed;
```

Converted

```
... byte definition and entity declaration: see oryginal VHDL model
ARCHITECTURE mixed OF reg_file_seu IS
SIGNAL seus1:std_logic:='0'; SIGNAL seus2:std_logic:='0';
SIGNAL r8  : byte; shared variable r8_v  : byte;
SIGNAL r9  : byte; shared variable r9_v  : byte;
SIGNAL r10 : byte; shared variable r10_v : byte;
SIGNAL r11 : byte; shared variable r11_v : byte;
BEGIN
  read: PROCESS (address, out_en) BEGIN
    IF out_en = '1' THEN CASE address IS
      WHEN "01000" =>
        r8_v:=r8 xor readSEU(r8,r8'path_name); data_out<=r8_v;
      WHEN "01001" =>
        r9_v:=r9 xor readSEU(r9,r9'path_name); data_out<=r9_v;
      WHEN "01010" =>
        r10_v:=r10 xor readSEU(r10,r10'path_name); data_out<=r10_v;
      WHEN "01011" =>
        r11_v:=r11 xor readSEU(r11,r11'path_name); data_out<=r11_v;
      WHEN OTHERS =>
        data_out <= (OTHERS => 'Z');
      END CASE;
      seus1 <= not seus1;
    ELSE data_out <= (OTHERS => 'Z');
    END IF; END PROCESS;
  write: PROCESS (clock) BEGIN
    IF clock'EVENT AND clock = '1' THEN IF write_en = '1' THEN
      CASE address IS
        WHEN "01000"=>r8_v :=data_in; WHEN "01001"=>r9_v :=data_in;
        WHEN "01010"=>r10_v:=data_in; WHEN "01011"=>r11_v:=data_in;
        WHEN OTHERS =>NULL;
      END CASE;
      seus2 <= not seus2;
    END IF; END IF; END PROCESS;
  seu_update: PROCESS (seus1, seus2) BEGIN
    IF r8_v /= r8 THEN r8 <= r8_v; END IF;
    IF r9_v /= r9 THEN r9 <= r9_v; END IF;
    IF r10_v /= r10 THEN r10 <= r10_v; END IF;
    IF r11_v /= r11 THEN r11 <= r11_v; END IF;
  END PROCESS;
END mixed;
```

Figure 2: Example of VHDL source conversion

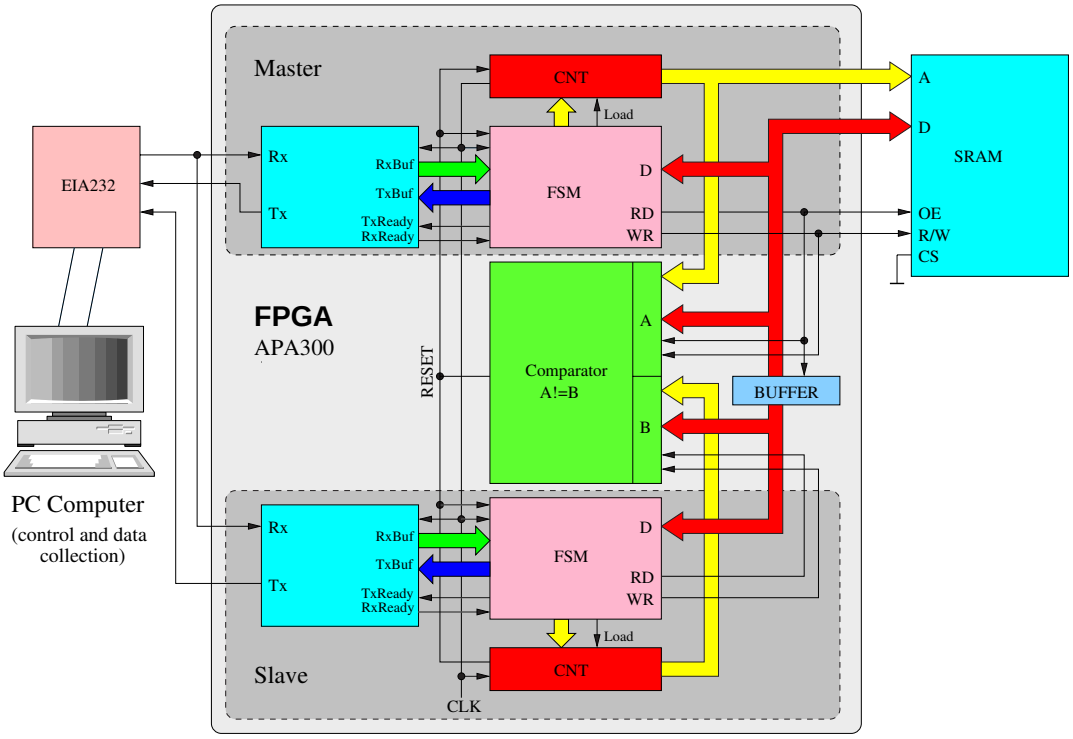

Figure 3: The block diagram of RadMon system

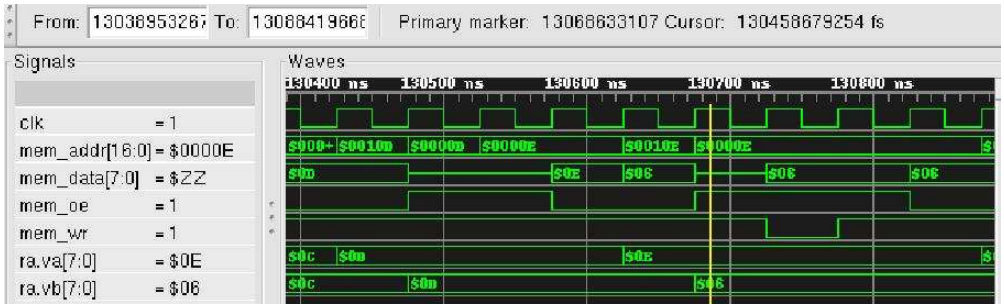

Figure 4: Simulation results of RadMon operation SEU in SRAM detector

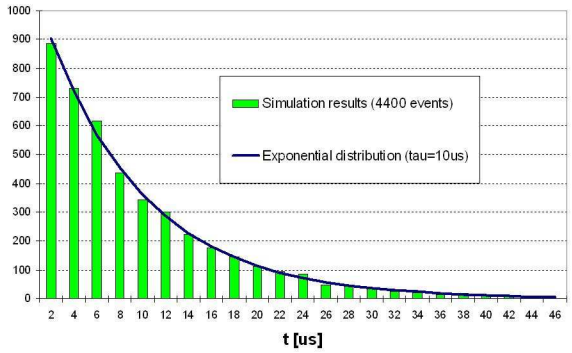

Figure 5: Simulation results of RadMon operation time between registered SEUs

5 CONCLUSION

The VHDL language and simulators can be used for description and modeling of digital circuits that are jeopardized to the ionizing particle radiation, however additional functions (SEUSIM library) and source code modifications are needed. The SEUSIM library allows simulating the results of SEU generation and propagation through the circuit thus enabling evaluation of proficiency of applied mitigation techniques.

6 ACKNOWLEDGEMENTS

We acknowledge the support of the European Community-Research Infrastructure Activity under the FP6 "Structuring the European Research Area" program (CARE, contract number RII3-CT-2003-506395), and Polish National Science Council Grant "138/E-370/SPB/6.PRUE/DIE 354/2004-2007".

REFERENCES

[1] Dodd P. E., Massengill W. L.: "Basic Mechanisms and Modelling of Single-Event Upset in Digital Microelectronics", IEEE Trans. on Nuclear Science, Vol. 50, No.3, s. 583-602, June 2003

[2] Hareland S., Maiz J., Alavi M., Mistry K., Walstra S., C.Dai.: "Impact of CMOS Scaling and SOI on soft error rates of logic processes", Proc. Symp. VLSI Tech., pp.73-74, 2001

[3] Baumann R.: "The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction", IEDM Tech. Dig., pp.329-332, 2002

[4] Normand E.: "Single event upset at ground level," IEEE Trans. Nucl. Sci., vol. 43, pp. 2742-2750, Dec. 1996.

[5] Jensen D. W.: "An Error Correction Code to Address Neutron Single Event Upsets in Semiconductor Memory", 13th Single Effects Symposium, Manhattan Beach, April, 2002

[6] Mavis D.G., Eaton P.H.: "Soft error rate mitigation techniques for modern microcircuits", Proc. Int. Reliability Physics Symp. Apr. 2002, pp.216-225

[7] http://www.intersil.com/military/

[8] Turowski M., Raman A., Mostrom M.: "Analysis of Ion-Strike Effects In Deep Submicron CMOS Devices", 12th Int. Conf. MIXDES 2005, pp. 291-296, 2005.

[9] Roche Ph., Palau J. M., Belhaddad K., Bruguier G., Ecoffet R., Gasiot J.: "SEU response of an entire SRAM cell simulated as one contiguous three dimensional device domain", IEEE Trans. Nucl. Sci., vol. 45, pp.2534-2543, Dec. 1998

[10] Dodd P. E., Massengill W. L.: "Basic Mechanisms and Modelling of Single-Event Upset in Digital Microelectronics", IEEE Trans. on Nuclear Science, Vol. 50, No.3, s. 583-602, June 2003

[11] Makowski D., Grecki M., Jablonski G.: "Application of a Genetic Algorithm to Design of Radiation Tolerant Programmable Devices", 11th Int. Conf. MIXDES 2004, pp.463-467, Szczecin, 2004

[12] Fog A.: "Chaotic random number generators with random cycle lengths", http://www.agner.org/random/theory

[13] http://www.scarpaz.com/vhdl_grammar