Radiation tolerant design of RLBCS system for RPC detector in LHC experiment

Wojciech M. Zabolotny[a], Ignacy M. Kudla[b], Krzysztof T. Pozniak[a],Karol Bunkowski[b], Krzysztof Kierzkowski[b], Grzegorz Wrochna[c], Jan Krolikowski[b]

[a]Institute of Electronic Systems, Warsaw University of Technology, ul. Nowowiejska 15/19, 00-665 Warszawa, Poland
[b]Institute of Experimental Physics, Warsaw University, ul. Hoza 69, 00-681 Warszawa, Poland
[c] The Andrzej Soltan Institute for Nuclear Studies, ul. Hoza 69, 00-681 Warszawa, Poland

ABSTRACT

This paper describes the design of the Link Box Control System for the RPC Detector (RLBCS), emphasizing the features needed to assure reliable operation in the irradiated environment of the RPC detector and its neighbourhood. The development process required to balance different factors - radiation hardness, reliability, flexibility, firmware upgrade possibilities, diagnostic features. The final design presented in the paper is a result of compromise between the above requirements.

Keywords: Radiation Tolerant Design, TMR, FPGA, HEP, High Energy Physics, CMS, LHC, RPC muon trigger, Detector Control System

1. INTRODUCTION

The RPC muon trigger is one of components of the level one trigger in the CMS detector in LHC experiment in CERN.[2] The electrical signals from the Resistive Plate Chambers (RPC) (Fig. 1) are received by the Link Boards (LBs). The Link System FPGA chips, located on the LBs, synchronise these signals, compress them and send via the high speed unidirectional optical link to the CMS counting room for further processing.[3]

However the correct operation of the Link System requires also a relatively slow bidirectional channel, needed for sending of control commands, operation parameters, and for reading the diagnostic data from the Link System.

This bidirectional channel is provided by the RPC Link Box Control System (RLBCS).[4] The RLBCS uses the redundant serial link based on CCU25 chips.[5] Each CCU25 chip is located on a dedicated Control Board (CB), which provides communication for maximally 9 LBs, using a specially developed Control Bus (CBus). Each CB contains also some additional FPGA chips, needed to interface CCU25 to the CBus and to implement some additional functionalities, which will be described later.

The complete system will contain 248 Control Boards (192 in the first stage), and 1808 Link Boards (1232 in the first stage).[6] Therefore the price of single CB and LB is a significant factor to be considered when designing the RLBCS.

Another requirement was to assure the reasonable flexibility of the design. It should be possible to configure most FPGAs with the special diagnostic versions of the IP cores. On the other hand the configuration of the SRAM based FPGAs with standard cores should be as fast as possible to assure fast start of the system after the power up (this requirement is even more important, as the system requires periodical reconfiguration of the SRAM based FPGAs due to reliability reasons, which will be described later). The possibility of firmware upgrade in the working system is also desirable.

The design of RLBCS must also consider fact, that LBs and CBs work in the irradiated neighbourhood of the CMS detector, so special radiation hard or at least radiation tolerant solutions are required to provide reliable operation.

These main concerns: radiation hardness or tolerance, reliability, flexibility and low cost lead to contradictory requirements. Therefore the final design is a result of compromise between all of them.

Further author information: (Send correspondence to W.M.Zabolotny)
W.M.Zabolotny: E-mail: wzab@ise.pw.edu.pl, Telephone: +48 (22) 660 7717

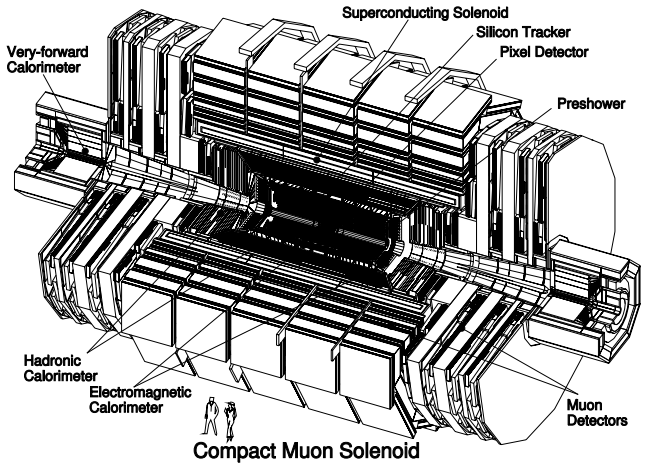

Figure 1. The view of CMS detector.[1] The "Muon Detectors" contain Resistive Plate Chambers (RPC).

2. RADIATION RELATED REQUIREMENTS FOR RLBCS ARCHITECTURE

The Link System and RLBCS will be located in the environment irradiated with thermal neutrons and high energy hadrons.[7] This conditions may lead to SEU effects in electronic systems. If the reliability was the only factor taken into consideration, both systems should be built using the radiation hard programmable chips (eg. the RT54SX family manufactured by Actel[8]), using the Triple Modular Redundancy (TMR) or even Functional Triple Modular Redundancy (FTMR) approach.[9] However such solution is not acceptable, because of many reasons:

- There are no radiation hard chips of complexity sufficient for the Link System
- The radiation hard chips are very expensive, and their use would render the price of Link System and RLBCS unacceptably high
- The radiation hard chips use antifuse based technology, which does not allow reprogramming of the chip. If such chips were used, it would not be possible do download diagnostic IP cores or to upgrade the firmware.

The analysis of SEU sensitivity of electronic components, performed in Ref. 10 leads to conclusion, that the SRAM configured FPGA chips may be used in such conditions, but periodic reconfiguration is required.

Therefore the RLBCS should be able not only to configure the Link System's FPGA chips after the power up, but also to reconfigure them periodically. However configuration of SRAM based FPGAs requires the significant amount of data. In the last version of the Link System the Xilinx XC3S1000FG456-5 chips are used. Each of them requires 3,223,488 bits (402936 bytes) of configuration data.[11] These data must be either transferred to the LB each time, when the chips are reconfigured, or must be stored locally. Unfortunately throughput of the CCU25 based communication links, connecting the RLBCS with the rest of the system, is too low to allow for retransmission of configuration data. Therefore the configuration data must be stored locally.

Storing of such amount of data in a safe way in the irradiated environment may be also a difficult task. The tests described in Ref. 10 included testing of the SDRAM memory chips and FLASH memory chips. These tests have shown, that using the FLASH memory assures sufficient reliability, especially when supplemented with the ECC technology. Such solution additionally assures that configuration data do not have to be reloaded after each power down of the system.

However efficient programming of the FLASH memories with data sent via relatively slow CCU25 link requires intelligent FLASH controller with buffering capabilities. Therefore relatively complex SRAM based FPGA chips are also required for RLBCS system. The final design of RLBCS system resulting from the above requirements is shown in the Fig. 2. The RLBCS contains single SRAM based FPGA on each Link Board. This is so called Link Board Controller

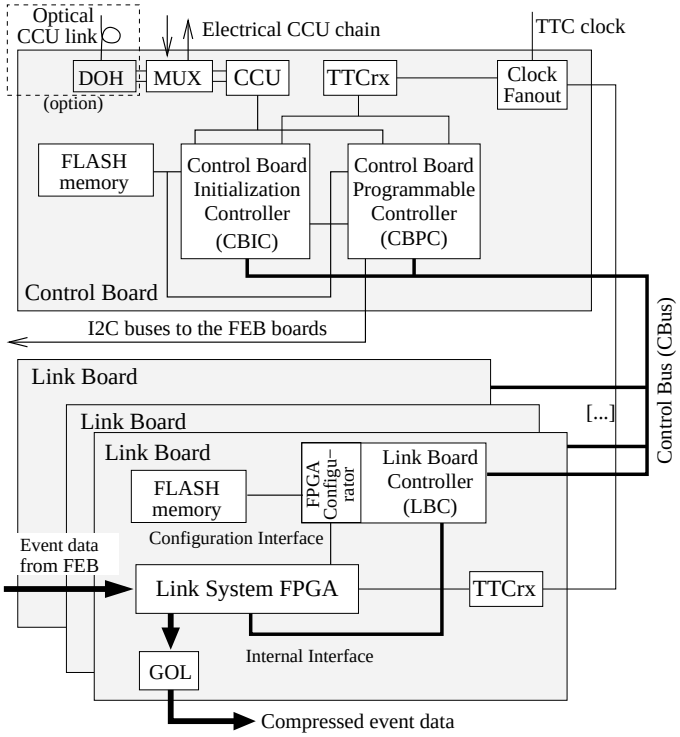

Figure 2. The final radiation tolerant architecture of the RLBCS system.[4]

(LBC), responsible for interfacing between the CBus and the LB's Internal Interface bus (II), for reconfiguration of the Link System FPGA, and for control of the local FLASH memory.

Each CB contains Control Board Programmable Controller - responsible for interfacing between the CCU25 chip and the CBus, for control of the local FLASH memory, and for some additional functionalities (eg. I2C interface). This is a relatively complex chip, which must also be implemented in the SRAM based FPGA.

The next important component is the Control Board Initialization Controller (CBIC), which is responsible for reconfiguration of CBPC and LBC, using the data stored in the local FLASH memory. If during reconfiguration the CB FLASH memory does not contain valid data, CBIC enters the emergency mode, where configuration data are sent directly via the CCU25 link. Similarly if the LB FLASH memory does not contain the valid configuration data, the LBC enters the emergency mode, where the configuration data are sent via the CCU25 link and the CBus.

The CBIC is relatively simple, and can be implemented in a radiation tolerant technology. The last version of design uses the Actel's FLASH based ProAsicPlus[12] FPGA as CBIC. The radiation tests[13] has shown, that these chips should not loose their configuration due to radiation during the operational life of the detector.

3. RADIATION TOLERANT IMPLEMENTATION TECHNIQUES

The radiation tolerance of the described RLBCS architecture has been further supported with some implementation features. These include protection of the configuration data stored in the FLASH memories and protection of the internal logic of FPGA chips.

3.1. Radiation protection of the configuration data

Special measures have been taken to protect the configuration data stored in the FLASH memories. The radiation tests described in Ref. 10 have shown, that radiation induced data errors in FLASH are mainly changes of data bits from "0" to "1". To reliably detect such data corruptions, a simple checksum, defined as "*number of zeroes in the data word*" may

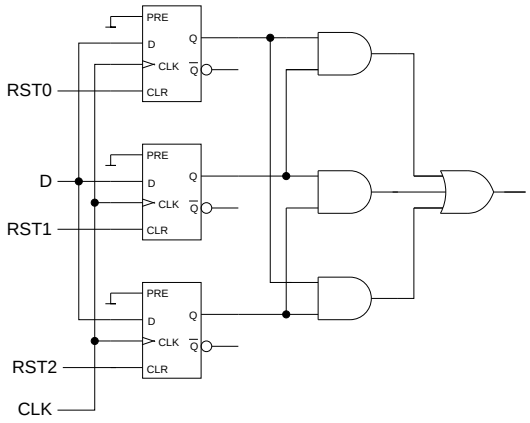

Figure 3. Basic implementation of the triple redundant flipflop.

be used. The configuration memory works with 32-bit long data words, where 27 bits store configuration data and 5 bits store the checksum. Additionally each three data words are supplemented with the parity word (also checksum protected). If only a single word in such 4 word block is corrupted, it is possible to recover it from the remaining data words and the parity word.

Another level of protection is assured by the fact that the configuration memory is able to store two sets of configuration data. If one set is found corrupted and unusable, it is possible to switch to the second one. The corrupted set may be refreshed, while the second one is used for the reconfiguration of chips.

To allow early detection of the configuration data corruption, the CBPC and LBC chips scan the contents of the FLASH memory, and notify the managing computer system if either the recoverable or unrecoverable data corruption is found.

The described mechanisms should minimize the probability of situation, when RLBCS or Link System SRAM based FPGAs can not be reconfigured due to corruption of configuration data.

3.2. Protection of internal logic in RLBCS FPGAs

The RLBCS chips store some internal data (eg. current states of the state machines, contents of the control registers, counters etc.) which are very important for the correct operation of the system. To protect this information against the radiation induced corruption, the triple redundant logic should be used. The ideal solution would be to use the Functional Triple Modular Redundancy (FTMR), described in Ref. 9 with both combinational and sequential logic triplicated. However this approach causes some portability problems related to different logic synthesis tools, and therefore another solutions were developed. Because the triplication of combinational logic leads to high increase of resources consumption in the FPGA, only the sequential logic has been triplicated.

The basic triplicated flipflop with voting circuit is shown in the figure 3. Using of redundant logic inside of FPGA chips is a fight against the optimization algorithms used by synthesis tools. It is difficult to protect the elements assuring redundancy, while allowing to perform other advantageous optimizations. Some synthesis tools use special attributes to keep some signals during the optimization, however their use is non portable and may result in preserving of not used signals, which increases consumption of resources.

In the RLBCS a portable protection of redundancy against optimization has been used, achieved by using three separate reset lines, which are tied together outside the chip. The basic implementation of redundant flipflop using this technique has been shown in the Fig. 3. The real implementation used in the RLBCS FPGAs is a little different. To avoid problems when rising slope of reset signals occurs near to the active clock slope, the synchronous resets have been used. The final VHDL implementation of the basic triple redundant flipflop is shown in the Fig. 4. Additionally dedicated VHDL components have been created to implement the registers storing multibit words and integer values in such TMR flipflops.

However usage of such redundant components affects the VHDL description of the system. The typical behavioral VHDL description heavily relies on the automatic inferring of flipflops. This is not possible when using TMR flipflops,

```
entity tripsig is
  port (
    input  : in  std_logic;                    -- input signal
    output : out std_logic;                    -- output signal
    init   : std_logic;                        -- initial state after reset
    a_rst  : in std_logic;                     -- asynchronous reset
    r_rst  : in  std_logic_vector(2 downto 0); -- synchronous resets
    clk    : in std_logic                      -- clock
    );

end tripsig;

architecture tripsig1 of tripsig is

  type  my_data is array(2 downto 0) of std_logic;
  signal d : my_data;                          -- flipflops storing the data

begin  -- trip1
    output <= (d(2) and d(1)) or (d(1) and d(0)) or (d(2) and d(0));

  g2: for j1 in 2 downto 0 generate
    process (clk, a_rst,init)
    begin
      if a_rst = '0' then                      -- asynchronous reset (active low)
        d(j1) <= init;
      elsif clk'event and clk = '1' then       -- rising clock edge
        if rst(j1) = '0' then                  -- synchronous reset (active low)
          d(j1) <= init;
        else                                   -- normal operation
          d(j1) <= input;
        end if;
      end if;
    end process;
  end generate g2;
end tripsig1;
```

Figure 4. The VHDL implementation of simple redundant flipflop

unless the synthesis tool supports triple redundant logic.[14] To obtain portable solution, all state machines in the RLBCS system had to be implemented using the combinational processes with external registers. An example implementation of the redundant state machine is shown in the Fig. 5.

Another problem associated with the use of TMR components is the implementation of asynchronously writable registers (usually implemented as latches). They are used to store information transferred from the external asynchronous bus. A possible redundant implementation of such asynchronously writable fliplop is shown in the Fig. 6. Such implementation allows for refreshing of flipflop contents after each system clock pulse. Unfortunately, such structure may not be efficiently synthesized in all used FPGA chips. For example in Actel chips such component is synthesised using the flipflops created from gates, which affects both resources consumption and speed of the design.

To avoid such problems standard TMR flipflop (as shown in the Fig. 3), with write strobe connected to the clock input, has been used to implement asynchronously writable registers.

This technique allows for convenient implementation of asynchronously writable registers in a sequential processes, as shown in the Fig. 7. The redundancy is achieved using a simple for–generate loop, and again separate reset lines are used to protect redundant flipflops against optimization. In this implementation it is possible to use the automatic inferring of flipflops, and the resulting code is both efficient and readable.

The disadvantage of this implementation, resulting from limitations of the VHDL language, is that for every size of the register it is necessary to declare another type (as it is impossible to declare a type "type TSLVR is array(2 downto 0) of std_logic_vector(natural range <>)"). This problem was solved by writing a simple bash script automatically generating a VHDL package file with all necessary type declarations.

Certainly in the above implementation the contents of flipflops is not refreshed automatically, and it is necessary to rewrite them periodically to avoid accumulation of radiation induced errors. In the case of SRAM based FPGAs this is not a problem, because anyway after periodic reconfiguration contents of all registers must be reinitialized. In the FLASH based CBIC FPGA it is necessary to periodically rewrite all the registers.

```
-- Signal in1 is of type std_logic, should be synchronized with clk
-- States ST_IDLE, ST_1, ST_2, ST_3, ST_4 are integer constants
-- Signal declarations
signal s_state_i, s_state_o, s_state_ini : integer;
signal s_rout1_i, s_rout1_o, s_rout1_ini : std_logic_vector (1 downto 0);
signal out2 : std_logic;
[ ... ]
-- Instantiation of the state register
st1 : tripstate
  generic map ( width => 3 )
  port map (
    input  => s_state_i, output => s_state_o, init   => s_state_ini,
    arst   => a_reset,   rst    => r_reset,   clk    => clk);
s_state_ini <= ST_IDLE;

-- Instantiation of the registered output
rout1 : tripreg
  generic map (width => 2 )
  port map (
    input  => s_rout1_i, output => s_rout1_o, init   => s_rout1_ini,
    arst   => a_reset,   rst    => r_reset,   clk    => clk);
s_rout_ini <= (others -> '0');
[ ... ]
-- Process for determination of state changes and outputs
 s1: process(s_state_o, s_rout1_o, in1)
   begin
   -- default assignments to avoid latches on non-registred signals
   out2 <= '0';
   -- the redundant registers are looped-back by default
   s_state_i <= s_state_o;
   s_rout1_i <= s_rout1_o;
   -- synchronous reset
   if not a_reset then
     if s_state_o = ST_IDLE then
       if in1='1' then
         s_state_i <= ST_1;  s_rout1_i <= "10"; out2 <= '1';
       end if;
     elsif s_state_o = ST_1 then
       s_rout_1 <= "10";     s_state_i <= ST_2;
     elsif s_state_o = ST_2 then
       s_state_i <= ST_3;    out <= '1';
     elsif s_state_o = ST_3 then
       s_state_i <= ST_3;
     elsif s_state_o = ST_4 then
       s_rout_1 <= "00";     s_state_i <= ST_IDLE;
     end if;
   end if;
 end process bs1;
```

Figure 5. The VHDL code implementing the redundant state machine

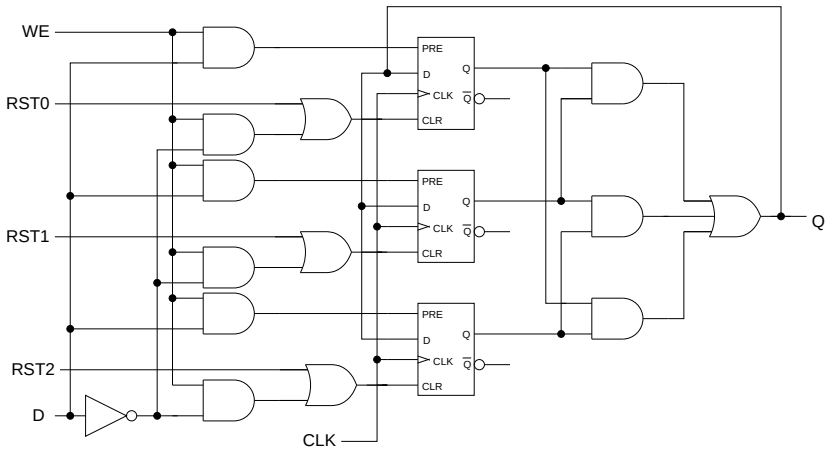

Figure 6. Extended triple redundant register with asynchronous write.

```
-- Types and function declarations in the package
type TSLVR4 is array(2 downto 0) of std_logic_vector(3 downto 0);
function r_vote (constant din : TSLVR4)
  return std_logic_vector;

type TSLVR8 is array(2 downto 0) of std_logic_vector(7 downto 0);
function r_vote (constant din : TSLVR8)
  return std_logic_vector;

-- Sample body of the r_vote function in the package
function r_vote ( constant din : TSLVR8)
  return std_logic_vector is
variable result : std_logic_vector(7 downto 0);
begin  -- wz_vote
  for i in result'range loop
    result(i) := ( din(0)(i) and din(1)(i) ) or
                 ( din(2)(i) and din(1)(i) ) or
                 ( din(0)(i) and din(2)(i) );
  end loop;  -- i
  return result;
end wz_vote;

--Signals declarations
signal reg1_r : TSLVR4;  -- redundant 4-bit register
signal reg1 : std_logic_vector(3 downto 0);
signal reg2_r : TSLVR8;  -- redundant 8-bit register
signal reg2 : std_logic_vector(7 downto 0);
signal s_nWR : std_logic; -- write strobe
signal s_nRESET : std_logic_vector(2 downto 0); -- reset signals
signal s_data : std_logic_vector(7 downto 0); -- data bus

--Processes implementing the real write access
gwr1 : for i in 0 to 2 generate
  bus1 : process (s_nCS, s_nRESET)
  begin  -- process bus1
    if s_nRESET(i) = '0' then
      -- reset registers - this protects against
      -- optimizing them out
      reg1_r(i)      <= (others => '0');
      reg2_r(i)      <= (others => '0');
    elsif elsif s_nCS'event and s_nCS = '1' then
      -- rising edge of Chip Select signal
      if s_nWR = '0' then
        -- write access
        case s_address is
          when ADDR_REG1 =>
            reg1_r(i) <= s_data(3 downto 0);
          when ADDR_REG2 =>
            reg2_r(i) <= s_data;
          when others => null;
        end case;
      end if;
    end if;
  end process bus1;
end generate gwr1;

--Conversion of redundant signals into standard ones
reg1 <= r_vote(reg1_r);
reg2 <= r_vote(reg2_r);
```

Figure 7. The VHDL code implementing the redundant register writable from external bus

4. CONCLUSIONS

The presented design of RLBCS assures protection against the radiation induced errors, while keeping the costs at reasonable level. The protection is achieved by periodical reconfiguration of SRAM based FPGAs with the configuration data stored locally in the FLASH memory.

The configuration data are protected using the ECC technique, which allows to detect data corruption when the data are still usable. Two sets of the configuration data may be stored in the FLASH memory, so another set may be used when the first one requires refreshing. The contents of FLASH memory is continuously checked to detect data corruption as early as possible and to request reloading of corrupted FLASH area.

The sequential logic in the RLBCS FPGA is protected using the TMR flipflops to minimize the probability of corruption of data stored in the internal registers.

Implementation of fundamental functionalities in the FLASH based radiation tolerant FPGAs (CBIC) assures fast startup of the system after power up or after periodical reconfiguration, while implementation of more complicated functionalities in the SRAM based FPGAs provides low cost, high flexibility, good diagnostic capabilities and possibility of firmware upgrades.

REFERENCES

1. "CMS Outreach, Detector Drawings." <http://cmsinfo.cern.ch/Welcome.html/CMSdocuments/DetectorDrawings/DetectorDrawings.html>.
2. J. Krolikowski, G. Wrochna, M. Konecki, M. Kudla, A. Ranieri, E. Pietarinen, K. Banzuzi, K. Pozniak, and P. Zalewski, "RPC Trigger," in *CMS, The Trigger and Data Acquisition project*, C. E. Board, ed., **I**, pp. 419–480, CERN, 2000. Also available as <http://cmsdoc.cern.ch/cms/TDR/TRIGGER-public/CMSTrigTDR.pdf>.
3. M.Gorski, I.Kudla, and K.Pozniak, "Resistive Plate Chamber (RPC) Based Muon Trigger System for the CMS Experiment - Data Compression/Decompression System," *Nucl. Instr. and Meth. in Phys. Res.* **A 419**, pp. 701–706, 1998.
4. W. M. Zabolotny, I. M. Kudla, K. T. Pozniak, K. Kierzkowski, M. Pietrusinski, G. Wrochna, and J. Krolikowski, "RPC Link Box Control System for RPC Detector in LHC Experiment," in *Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments*, R. S. Romaniuk, ed., *Proc. SPIE* **5775**, pp. 131–138, 2004.
5. A. Marchioro, C. Ljuslin, and C. Paillard, "CCU25 Communication and Control Unit ASIC for Embedded Slow Control." <http://cmstrackercontrol.web.cern.ch/cmstrackercontrol/documents/Sandro/CCU25Specs%20v2-1.pdf>.
6. M. Kudla, "RPC Trigger Progress Report." <http://pccms9.igf.fuw.edu.pl/users/kudla/konferencje/tridas0411/mk_0411.pdf>.
7. "CMS COTS Workshop 1999." <http://cmsdoc.cern.ch/ huu/tut1.pdf>.
8. "The Programmable Alternative to High-Density ASICs for Space Applications." <http://www.actel.com/products/aero/rtaxs.aspx>.
9. S. Habinc, "Functional Triple Modular Redundancy (FTMR)." <http://www.estec.esa.nl/microelectronics/asic/fpga_003_01-0-2.pdf>.
10. K. Bunkowski, I. Kassamakov, J. Krolikowski, K. Kierzkowski, M. Kudla, T. Maenpaa, K. Pozniak, D. Rybka, E. Tuominen, D. Ungaro, G. Wrochna, and W. M. Zabolotny, "Radiation tests of CMS RPC muon trigger electronic components," *Nuclear Instruments and Methods in Physics Research* **A538**, pp. 708–717, 2005.
11. "Spartan-3 FPGA Family: Complete Data Sheet." <http://www.xilinx.com/bvdocs/publications/ds099.pdf>.
12. "ProASIC^{PLUS} Actel's 2nd Generation Reprogrammable Flash FPGAs." <http://www.actel.com/products/proasicplus/index.html>.
13. "APA750 and A54SX32A LANSCE Neutron Test Report." <http://www.actel.com/documents/LANSCETestReportWP.pdf>.
14. "TMRTool Industry First development tool to automate generation of Triple Module Redundant (TMR) designs for re-programmable FPGAs." <http://www.xilinx.com/products/milaero/tmr/index.htm>.