IJCES International Journal of Computer Engineering Science , Volume1 Issue2, November 2011
ISSN : 2250:3439
https://sites.google.com/site/ijcesjournal
http://www.ijces.com/

Synthesizable Implementation of Safety Fuzzy Logic Controller of 1oo2 architecture in FPGA

Mohammed Bsiss, Amami Benaissa,

Laboratory of computer science systems and telecommunications (LIST)
Faculty of Science and Technology of Tangier,
University Abdelmalek Essaadi Tangier Morocco
90 000 BP 416 Tangier Morocco
mohbsiss@gmail.com, b_benaissa@hotmail.com

Abstract. Currently, the achievements of security systems is becoming more and more ground in different areas not only through the development of new technologies of programmable circuits, with the ability to achieve very complex systems in a single chip but thanks also a common and coherent organization of the different safety standard.This paper describes the implementation for a safety fuzzy logic controller (SFLC) on the basis of Safety Norm 61508. The SFLC is programmed with the hardware description language VHDL and implemented in FPGA.

Keywords: Safety norm 61508, Safety fuzzy logic controller, one-out-of-two architecture, field programmable gate array (FPGA), very high speed hardware description language (VHDL).

1 Introduction

Today the implementation of a safety fuzzy logic controller is necessary. This paper shows how an SFLC can be realized in FPGA. The internal structure of FPGAs is composed of an array of configurable logic blocks (CLBs) along with interconnection channels, blocks of SRAM and input/output blocks (IOBs). Certain FPGAs also contain PLLs (Phase Locked Loop), DLLs (Delay Locked Loop), (DCM: DIGITAL Clock Manager) and also simple ALUs (Arithmetic Logic Units).
However, this complex structure can contain a large number of failures such as stuck-at fault, bridging fault, interconnect defect, CLB defect.
One strategy discussed in [1] is based on creating several application circuits and testing them with test vectors developed specifically for each circuit.
The second strategy is based on testing the internal structure and reconfigurability of an FPGA and is called the Multi-Configuration Strategy (MCS) [2].
The third strategy [3] is based on the concept of Built-In Self-Test (BIST).
Generally all strategies mentioned above are normally employed by the chip manufacturer and used by unprogrammed FPGAs. The challenge is how can we detect this failure using programmed FPGAs? Or, in other words, how can we be sure that the generated VHDL code for the simple fuzzy logic controller (FLC) architecture [4], [5] is correctly operated on the device? Under these circumstances, a simple-structure for an FLC without a safety function (redundancy) does not provide reliability and the safety is only partial.
Particularly with regard to safety-related systems, model structures are necessary in order to allow safe operation in case of system failure.
This security must meet the requirements defined in Security Norm IEC61508 [6].

2 The security norm IEC61508

Safety Norm IEC61508 is a international norm for the functional safety of electrical/electronic/programmable electronic safety-related systems and consists of seven parts [8] (general requirement, requirements for E/E/PES safety-related systems, software requirements, definitions and abbreviations, examples of methods for the determination of safety-integrity levels, guidelines on the application of Part 2 and Part 3 and an overview of techniques and measures).

Safety Norm 61508 [8] defines the safety integrity levels (SIL) as an order of the magnitude levels of risk reduction. There are four SILs. SIL4 has the highest level of safety integrity and SIL1 the lowest in terms of risk reduction. The four levels are defined in Table 1 [6] for various fractions of failures leading to a safe state as follows:

Table 1. Type a safe failure fraction

Safe Failure fraction	Hardware fault tolerance (see note 1)		
	0	**1**	**2**
< 60 %	SIL1	SIL2	SIL3
60 % - < 90 %	SIL2	SIL3	SIL4
90 % -<99%	SIL3	SIL4	SIL4
>99%	SIL3	SIL4	SIL4
NOTE 1 A hardware fault tolerance of N means that N+1 faults could cause a loss of the safety function			

3 Safety fuzzy logic controllers

The safety fuzzy logic controller (SFLC) consists of the fuzzification process, rule evaluation process and defuzzification process. The details of this process can be found in the following publications [4], [5] and [7].
The difference to the simple controller FLC is that these processes are created redundantly. The used architecture of SFLC is one-out-of-two (1oo2), which means that in these particular cases the system must operate at least so that the security function can react when an error occurs, and thus bring the system to a condition of safety. This structure allows the system to meet the requirements of Safety Norm IEC61508 [6].
Figure 1 shows a basic model for a safety fuzzy logic controller with redundancy architecture (1oo2).

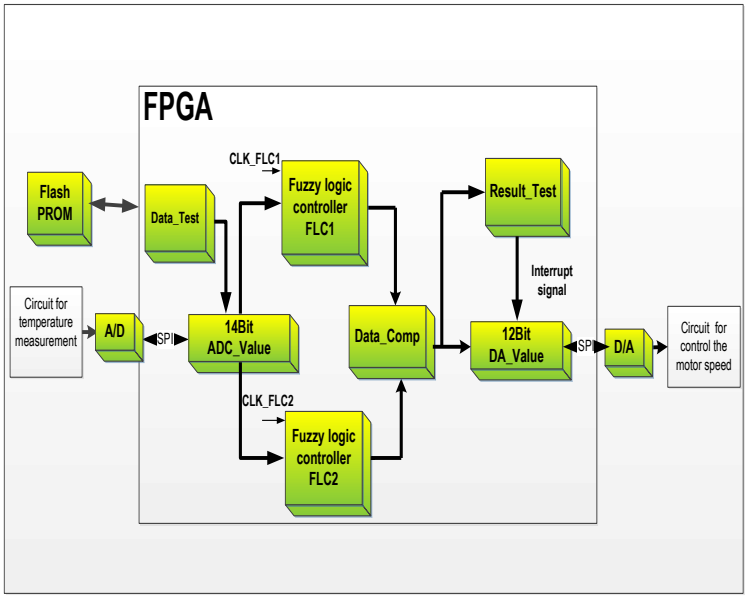

Fig. 2. Safety fuzzy logic architecture

As shown in Figure 1 the safety fuzzy logic controller consists of the following components:

- Two Fuzzy Logic Controllers.
- One Flash PROM to save Test Data Pattern.
- Register Data_Comp
- Register Data_Test.
- Register Result_Test.
- 14-bit register to save data from analog-to-digital converter LTC6912-1[9]
- 12-bit register to give data to the digital-to-analog converter LTC2624 [10]

The design of the fuzzy logic controller architecture is shown in figure 2.

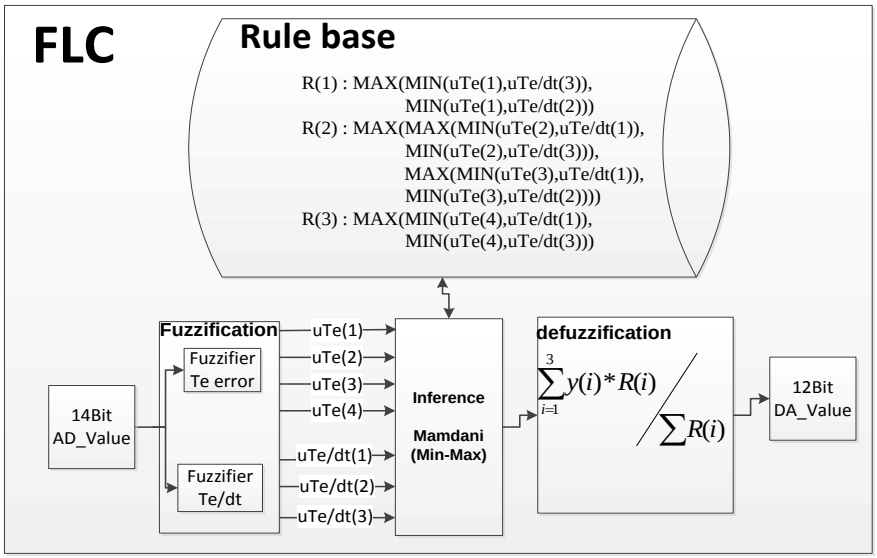

Fig.2. Fuzzy logic controller

3.1 Fuzzification

There are two variables for controlling the temperature input:

- The error of temperature Te
- And the rate of change of temperature dTe/dt.

The fuzzy set of membership functions of error of temperature is partitioned into 4 zones as shown in figure 3 below.

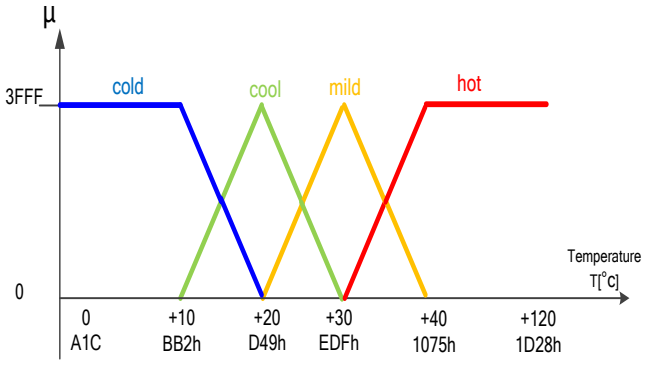

Fig. 3. The membership functions of error of temperature Te.

The analog to digital converter [9] has a 14-bit resolution which means that the membership degree μ = 1 is equal to 3FFFh or 16383 in decimal.
The second input variable for SFLC is the rate of change of temperature dTe/dt.
The membership functions of the derivation dTe/dt consist of three linguistic terms (slow, moderate and fast). The graphic representation of membership functions is shown in figure 4.

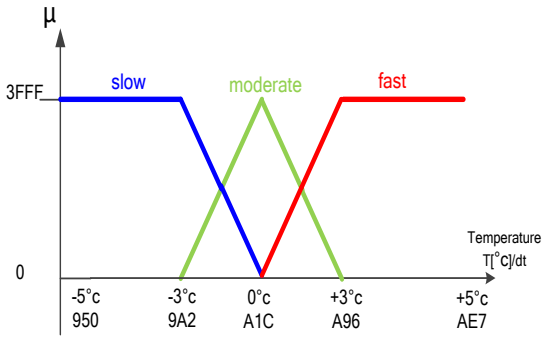

Fig. 4. The membership functions of the rate of the temperature change.

Figure 5 illustrates how part of this process is implemented in VHDL:

```
Fuzzification:process(CLK_FLC)
begin
if rising_edge(CLK_FLC) then

    if Reset = '0' then
    sig_stemp <= '0';
    for i in  1 to n  loop
    u(i) <= 0;
    end loop;
    end if;

    for i in 1 to n loop

    if  (sig_temp <= te_mf(i).point1)   then
    sig_u(i) <= 0;
    elsif (sig_temp <= te_mf(i).point2)     then
    sig_u(i) <= 1023+(sig_temp-te_mf(i).point2)*te_mf(i).tang1;
    else
    sig_u(i) <= 1023-(sig_temp-te_mf(i).point2)*te_mf(i).tang2;
    end if;

    if(sig_u(i)<=0)then
    u(i) <= 0;
    else
    u(i) <=sig_u(i);
    end if;
    end loop;

end if;
end  process Fuzzification;
```

Fig.5. The fuzzification process in VHDL

3.2 Rule inference engine

In this step, each input value is applied to its membership function in order to determine the value of the fuzzy input. In this application there are two inputs, one with four membership functions and the other with three, which makes seven degrees of membership functions to be calculated.

After that, the degrees of membership function are determined in a fuzzification step; the next step is to use linguistic rules to decide what action should be taken in response to a set of data.

The Mandani min-max technique [7] is used to calculate the numerical results of linguistic rules based on the input values of the system.

Before we commence calculating how many rules might be needed for the system, we must define the output membership functions.

The output membership functions consist of 3 singletons. The linguistic terms of the singletons output (slow, moderate and fast) are used.

The graphic representation of membership functions is shown in figure 6.

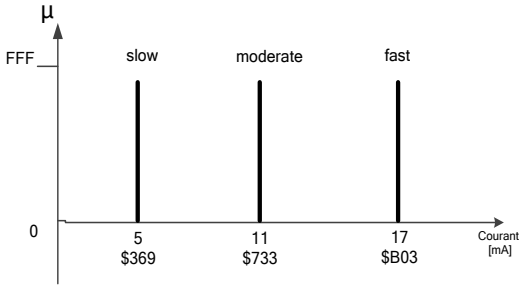

Fig.6. The membership functions of the output

The SFLC has two inputs, one with four linguistic terms and the other with three and an output with three linguistic terms. This makes a total of 4*3*3 different rules that may be used to describe the strategy of total control.

3.3 Defuzzification

After the rules for each output have been established, the next step is to combine them into a single output value that can be used to control the output.
The center of gravity [7] will be used to obtain the release of the final system. In this application, the defuzzification takes the weighted average of all fuzzy outputs. Each fuzzy output is multiplied by the corresponding singleton, and then the sum of these products is divided by the sum of all fuzzy outputs for the final result.
The following pseudo-code illustrates how this process is implemented in VHDL:

```
Defuzzification:process(CLK_FLC)
variable output,var_sum: integer;
begin
if rising_edge(CLK_FLC ) then
    if Reset = '0' then
    produit <= 0;
    sum <= 0;
    output:= 0;
    var_sum:=0;
    end if;
    for i in 1 to n loop
        output := (ac(i) * u(i)) + output;
        var_sum := u(i) + var_sum;
    end loop;
    result <= divide(conv_std_logic_vector(output,32),conv_std_logic_vector(var_sum,32))
end if;
end process Defuzzification;
```

Fig.7. Defuzzification process in VHDL

3.4 System test

The SFLC is tested by using following register ‘Data_test’, ‘Data_Comp’ and ‘Result_Test’.
The register ‘Data_Test’ compares the data from the first fuzzy logic controller FLC1 with the second FLC2. The comparison takes place during each clock cycle. The comparison of the faulty data is avoided by periodically testing the system with different data patterns. These patterns are stored in external flash PROM see Figure 1 and are based on measurements made on the system. Note that when a channel is tested, the functionality of the system is ensured by the second channel.

Therefore the SFLC allows not only to detect all stuck-at faults,, but also places the system in a safe state. The safety integrity levels (SIL) reached by this architecture is SIL1.

4 Functional and timing simulations

After converting the VHDL code into gate-level schematics, a test bench for the safety fuzzy logic controller was designed using the Isim tools [11] from Xilinx. This allows the timing and the correct functionality of the system to be verified.
A synthesizable simulation result (timing diagram) is shown in figure 8.

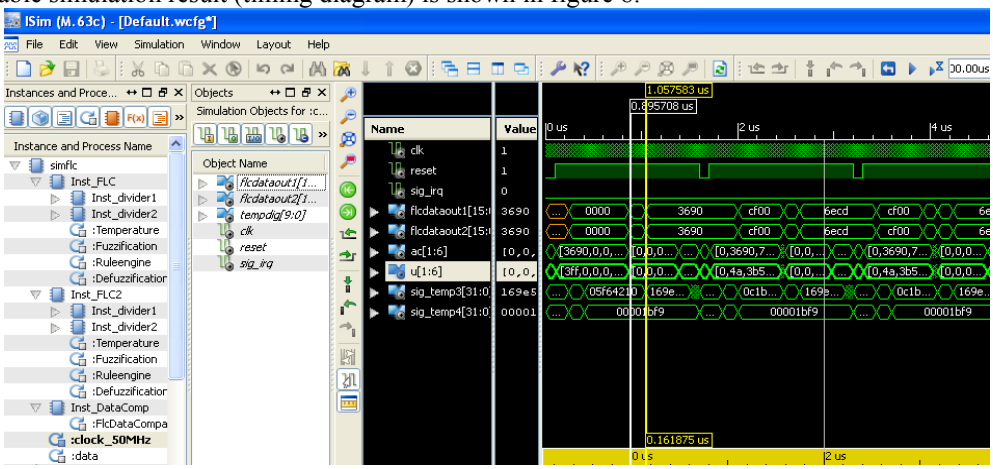

Fig.8. Waveform of functional simulation of the safety fuzzy logic controller

The waveform in figure 8 shows the values of the temperature inputs and the corresponding output current in hex form at the various instances determined by the stimuli in the test bench.
The parallelism in the FPGAs allows the fuzzification process and the defuzzification process of both fuzzy logic controllers to be carried out during one time period T = 0.9us. The result of the first FLC1 and the second FLC2 are stored within T = 1us in the registers flcdataout1 and flcdataout2.
As seen in figure 8 the result of the first FLC is exactly the same as the result of the second. If the interrupt signal (sig_irg) switches to high, it means there is a discrepancy in the results and the system goes into the safe state. When the system goes into the safe state, it means that the output is switched off.

5 Synthesis of the safety controller 'SFLC'

The synthesis processes convert the VHDL code into gate-level schematics. The synthesis process is carried out using Xilinx tools ISE Design Suite 12.2 [13]. The VHDL code of the safety fuzzy logic controller is synthesized for converting into RTL view.
Part of synthesized netlist design of the safety fuzzy logic controller is shown in figure 9.

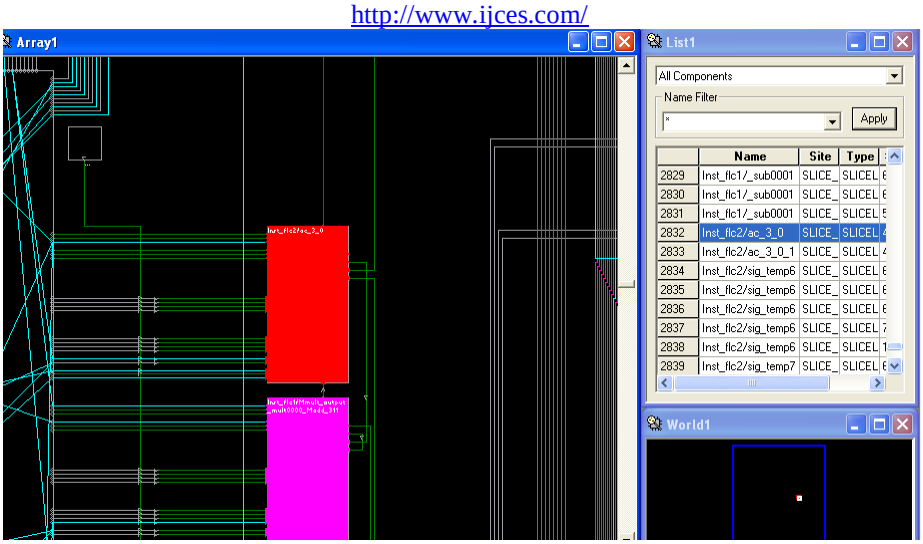

Fig. 9. Part of netlist design of SFLC

The generated synthesized netlist of the SFLC has been downloaded into the Sparan-3AN FPGA Starter Board Kit [12] for verification of the correctness of the algorithm functionality.

6 Conclusions

A novel approach to realize a safety fuzzy logic controller with a field-programmable gate array has been proposed in this paper. The design has been realized in VHDL using Xilinx12.3 [13.]
The SFLC operates at a frequency of 50 MHz and is not only able to react very quickly to the output whenever an error occurs, it can also put the systems into a safe state.
The simulation and the implementation of safety fuzzy logic controller with ISim 12.3 from Xilinx demonstrated complete functionality while meeting all the initial system requirements.

References

1. C.Jordan, W.P Marnane, 1993 . ”Incoming inspection of FPGAs”. In Proc. European Test Conf pp371-377.
2. M.B Tahoori, E J. Mc.Cluskey and M.Renovell, P. Faure, 2004. “A Multi-Configuration Strategy for Application Dependent Testing of FPGAs” In Proc. VLSI Test.
3. C.Stroud, P Chen, S.Konala, and M.Abramovici, 1995. “using ILA testing for BIST in FPGAs”. In proceedings of IEEE VLSI Test Symposium, pp.259-265
4. Philip T.Vsuong, Asad M.Madni and Jim B. Vuong 2006. “VHDL Implementation for a Fuzzy Logic Controller”, In Bei technologies, Inc.
5. Zeyad assi Obaid, Nasri Sulaiman and M. N. Hamidon July 2009. “FPGA-based Implementation of Digital Logic using Altera DE2 Board” in IJCSNS International Journal of Computer Science and Nezwork Security, VOL.9 No.8.
6. A Summary of the IEC 61508 Standard for Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems , version 2.0, january2, 2006 http://www.exida.com/articles/iec61508_overview.pdf
7. Fuzzy sets. Information and Control. 1965;8:338-353
8. David J Smith, Kenneth G L Simpson 2004. “Functional Safety” A Straightforward Guide to applying IEC 61508 and Related Standards. Elsevier Butterwoth-Heinemann, 2nd edition
9. Datasheet Analog to digital converter with 14-bit from Linear technology LTC6912 “Dual programmable gain Amplifiers with Serial Digital Interface”. http://cds.linear.com/docs/Datasheet/6912fa.pdf
10. Datasheet Digital to analog converter with 12 bit from lineartechnologyLTC2624. http://cds.linear.com/docs/Datasheet/2604fd.pdf
11. http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/plugin_ism.pdf
12. Spartan-3A/3AN FPGA Starter Kit Board User Guide UG334 (v1.1) June 19, 2008,

13.www.xilinx.com