Politecnico di Torino

Collegio di Elettronica, Telecomunicazioni e Fisica

Assignment report for the course Testing and Fault Tolerance

Master degree in Computer Engineering

Group: 16

Alessandro Vargiu s314294
Giovanni Santangelo s308882
Federico Tartaglia s319857

Contents

CHAPTER 1

Introduction

There are multiple methods for hardening modern day processors. Specifically, Hardware and Information redundancy are exploited to increase fault coverage of a starting basic processor.

1.1 Basic processor

The first step consists in analyzing the RTL of the base processor and evaluating the fault coverage. This means figuring out how well it can handle different types of faults. This preliminary step provides a starting point against which data from later simulations will be compared.
To generate the fault coverage report, the following workflow was employed:

1. Synthesis of the CV32E40P using Nangate45 as the technology library with Synopsys Design Compiler.
2. Cross-compilation of the STL for the CV32E40P RISC-V ISA.
3. Compilation of CV32E40P RT-level and Gate-level descriptions for functional logic simulation using QuestaSIM.
4. RT-level simulation of the SBST through QuestaSIM.
5. Gate-level simulation of the SBST through QuestaSIM.
6. Compilation of the sources for Z01X usage in functional mode.
7. Generation of the SAF fault list in SFF for Z01X.
8. Logic simulation to validate eVCD stimuli.
9. Fault simulation of CV32E40P using the FAULT_LIST previously generated.

The results obtained from the first analysis of the processor are derived from the file `cv32e40p_top_saf.rpt.hier` and are shown in Table 1.1. In addition to the results in the table, in the file `cv32e40p_top_saf.rpt.hier`, it is possible to observe that the most critical components of the processor are the register file, the ALU, and the multiplier.

	Prime		**Total**	
Total Faults	138676		138676	
Dropped Detected (DD)	40175	28.97%	40175	28.97%
Dropped Potential (PD)	685	0.49%	685	0.49%
Not Detected (ND)	12758	9.20%	12758	9.20%
Not Strobed (NS)	1474	1.06%	1474	1.06%
Not Observed (NO)	68364	49.30%	68364	49.30%
Not Controlled (NC)	15220	10.98%	15220	10.98%
Test Coverage	29.22%		29.22%	
Fault Coverage	29.22%		29.22%	

Table 1.1: Fault Coverage of the basic processor

CHAPTER 2

Hardware Redundancy

2.1 Triple Module Redundancy

Hardware redundancy involves designing a system with additional hardware beyond what is strictly necessary for implementing the system's functionalities. This methods aim at enhancing reliability and fault tolerance, at the cost of increasing significantly the area of the system.
In particular, Triple Module Redundancy (TMR) is a technique in which a specific module is replicated three times. Each replicated module receives identical inputs and generates a corresponding output. The three outputs are then fed into a `majority voter`, which determines the final output based on a majority rule. This technique can be effectively applied to the ALU and the multiplier of the processor to improve the fault tolerance of the system.

2.2 Hardening the ALU

Triple Module Redundancy has been applied to the ALU of the processor after recognizing it as a critical component in the system.
To implement the TMR technique, the ALU has been instantiated three times in the `ex_stage` module, situated in the `cv32e40p_ex_stage.sv` file. Additionally, a new component, the `alu_voter`, has been instantiated in the `ex_stage`. The `alu_voter` comprises two 1-bit `voter` modules and one 32-bit `voter_generic` module, with each voter corresponding to an output of the ALUs. Each `voter` produces a final output using the majority rule and provides a faulty output to indicate the presence of a fault in any of the three ALU outputs.

2.2.1 Fault Coverage Analysis

The first four fault coverage analyses has been conducted using the fault list generated for the processor without hardening. This approach allows for a clearer observation of the results of the hardening, as the new faulty locations introduced in the circuit are not taken into consideration.
The first analysis is executed without the inclusion of any strobe in the file `strobe.sv`. In the second analysis, the outputs of the first ALU are strobed. The third analysis involves strobing the data output of the voter, while the fourth analysis is performed strobing the faulty output of the voter.
Table 2.1 shows the outcomes obtained from the initial four analyses:

	No Strobe	**Strobe ALU1**	**Strobe Voter Data**	**Strobe Voter Faulty**
Total Faults	92354	92354	92354	92354
Dropped Detected (DD)	25898	29124	25919	29103
Dropped Potential (PD)	619	624	623	624
Not Detected (ND)	12150	8926	12125	8947
Not Strobed (NS)	1422	1422	1422	1422
Not Observed (NO)	40797	40785	40797	40785
Not Controlled (NC)	11468	11473	11468	11473
Test Coverage	28.38%	31.87%	28.40%	31.85%
Fault Coverage	28.38%	31.87%	28.40%	31.85%

Table 2.1: Fault coverage of the hardened processor with TMR of the ALU

From Table 2.1, several observations can be made. Specifically, when adding the strobe to the ALU1 outputs, the number of detected faults increases, while the number of undetected faults decreases compared to the case with no strobe. When strobing the data outputs of the voter, the number of detected and undetected faults returns to the same level as the first case. This occurs because the `alu_voter` selects the correct outputs, thereby masking the faults. Adding the strobe to the faulty output of the voter reveals that the faults previously masked are now detected.

Voter Fault Coverage Analysis

Further analysis can be conducted to evaluate fault coverage and the effectiveness of the applied technique. Specifically, an analysis of the `alu_voter` has been carried out by injecting faults only into the voter, and the results can be observed in Table 2.2.

	Voter No Strobe	**Voter Strobe Faulty**
Total Faults	1484	1484
Dropped Detected (DD)	184	774
Dropped Potential (PD)	0	0
Not Detected (ND)	406	65
Not Strobed (NS)	428	0
Not Observed (NO)	289	387
Not Controlled (NC)	177	258
Test Coverage	12.40%	52.16%
Fault Coverage	12.40%	52.16%

Table 2.2: Fault coverage of the `alu_voter`

As observed in the table, strobing the faulty output of the voter detects the majority of undetected faults, with 590 faults detected, 98 faults not observed, and only 65 faults remaining undetected.

Permanent Fault Injection

The last analysis that can be done is injecting the faults only in the three ALUs and then injecting a permanent fault in the voter. The fault injected is a non-critical fault (not detected) choosed from the `cv32e40p_top_saf.rpt.fsim` file of the analysis of the voter only. The two fault used in this step are:
`top_i.core_i.ex_stage_i.alu_voter.voter_2.U6.A` (SAF0) and
`top_i.core_i.ex_stage_i.mult_voter.voter_1.U92.A1` (SAF1).
The result of the analysis can be seen in table 2.3

	No Injection	**SAF0 Injection**	**SAF1 Injection**
Total Faults	54926	54926	54926
Dropped Detected (DD)	15502	14400	15502
Dropped Potential (PD)	5	5	5
Not Detected (ND)	7164	6676	7164
Not Strobed (NS)	184	184	184
Not Observed (NO)	29592	30686	29592
Not Controlled (NC)	2479	2975	2479
Test Coverage	28.23%	26.22%	28.23%
Fault Coverage	28.23%	26.22%	28.23%

Table 2.3: Fault coverage analysis with permanent fault injection

From this final analysis, we observe that injecting SAF0 into the voter reduces the number of detected faults, indicating that the fault is not masked. However, when injecting SAF1, the number of detected and undetected faults remains the same as in the case without injection, suggesting that the fault is correctly masked and the system remains safe.

2.3 Hardening the Multiplier

Multiplier hardening, along with the ALU, was done through the use of TMR. The multiplier is instantiated on the `EX_stage` part of the pipeline, present in the `cv32e40p_ex_stage.sv` file. The instance of the multiplier was triplicated and a *mult_voter* module was added. The simulations of the multiplier were conducted with the TMR model of the ALU already present in the processor, the results are to be taken with this into account.

The voter module for the multiplier contains four voter modules for each output of the multiplier. The `mult_voter` module provides a onebit signal as faulty output, which indicates the presence of a fault, without specifying which components caused it.

The same series of simulations as for the ALU, were conducted again for the multiplier.

The addition of the strobe point in the output of the first multiplier leads to an increase of detected faults, along with an increase of notobserved faults. Fault coverage improves, even considering the base processor results. Analysis of the voter faulty data output and faulty signal, lead to an increase in fault coverage as well.

	No Strobe	**Strobe MUL1**	**Voter data**	**Voter faulty**
Total Faults	91518	91518	91518	91518
Dropped Detected (DD)	25718	27084	26111	26691
Dropped Potential (PD)	621	670	669	670
Not Detected (ND)	12222	12398	13345	12791
Not Strobed (NS)	1414	1413	1414	1414
Not Observed (NO)	40055	36975	37023	36975
Not Controlled (NC)	11488	12977	12956	12977
Test Coverage	28.44%	29.96%	28.90%	29.53%
Fault Coverage	28.44%	29.96%	28.90%	29.53%

Table 2.4: Hardened processor with TMR multiplier

CHAPTER 3

Information Redundancy

3.1 The Hamming Code

The modular redundancy technique is not always the most beneficial for hardening purposes. In fact, if applied to significantly larger components like memories or register files, the resulting area of the processor could be significantly increased making the processor more expensive. The hamming code is a popular type of information redundancy technique, specifically known for its ability to correct the output data in the case of a Single error detection as well as detecting the presence of double errors. In the analyzed Risc V core this technique was applied to the Register File inside the decode stage.

3.2 Hardening the Register File

3.2.1 Motivation

The Register File is the most important element of the decode stage. It is a component utilized for every instruction of a program running in the processor. For its relevant and frequently used role, probabilities of incongruities of stored data may become relevant, hence why it is important to harden this module.

3.2.2 Implementation

The design of the hardened register file was made by introducing two types of new circuits

1. *Hamming Code Generator Circuit*: which serves for creating the parity bits for a particular signal and provides the encoded output to be stored in the register file.
2. *Error Checking Circuit*: for confronting the stored parity bits against new computed ones in the register file and fixing the error if it has occurred. This module provides the decoded version of the signal.

3.2.3 Fault Coverage Analysis

After the correct synthesis and compilation of the new modules, several tests have been performed thanks to the Zoix simulator. For these specific tests, it was decided to generate a new fault list with respect to the golden model component. This was also done to test Zoix's ability of detecting the new fault sights of the hardened design. From the Table 3.1 below, some main observations are remarked.

The Golden Test Scenario

The results in the first column called Golden, indicate the fault simulation results of the original register file component. For the generation of the fault list, all the possible faults regarding the register file instance in system Verilog were specified in `"gen_saf_cv32e40p_top.sv"` file by including the ** at the end of the constraint line.

The First Test Scenario

The first test consists of the fault simulation of the hardened register file along a new fault list considered. Because of the new fault list, Zoix has an increment in the Total Faults parameter and with respect to the golden model do to it discovering new fault sights, the not detected faults increased. However the detected faults have also proportionally increased because of these new fault sights.

The Second Test Scenario

In this Test scenario more primary outputs were added to the strobe. Particularly the second test scenario involves the addition of the double error detection signals in the strobe. The results forecast that finding two bits of error in a specific location of the register file is improbable, hence why a small increment in the detected faults.

The Third Test Scenario

Lastly the last column indicates the results obtained after indicating the single error detection bits in the strobe file. This shows us about ten percent increase in the fault coverage with respect to the golden scenario. This remarks that it is very probable that single error occur in the register file location and thanks to the strobing points more faults can be detected making the processor more secure.

	Golden	**1st Test**	**2nd Test**	**3rdTest**
Total Faults	43178	67858	67858	67858
Dropped Detected (DD)	15882	24497	24508	30857
Dropped Potential (PD)	517	614	662	1296
Not Detected (ND)	5368	9858	9801	2831
Not Strobed (NS)		2000	1964	1944
Not Observed (NO)	11885	15177	15209	15203
Not Controlled (NC)	9526	15712	15714	15727
Test Coverage	37.38%	36.55%	36.60%	46.43%
Fault Coverage	37.38%	36.55%	36.60%	46.43%

Table 3.1: Register File with Hamming Code results

In conclusion from the register File with Hamming it was possible to achieve a normalized fault coverage gain of about 1.24, while still maintaining a reasonable Area overhead of 1.14 (obtained via the confrontation of the `"syn_nandgate.log"` files.

CHAPTER 4

Conclusion and Final Results

4.1 Final Fault Analysis

In conclusion after synthesizing and compiling the processor with the additional hardened Arithmetic Logic Unit and Multiplier along with their voters, and the hardened register file the final results of the can be depicted in the table 4.1

	Strobe	**No Strobe**
Total Faults	240928	240928
Dropped Detected (DD)	76529	46227
Dropped Potential (PD)	1552	812
Not Detected (ND)	15908	32875
Not Strobed (NS)	3558	4520
Not Observed (NO)	110679	132406
Not Controlled (NC)	32702	24088
Test Coverage	32.09%	19.36%
Fault Coverage	32.09%	19.36%

Table 4.1: Final Test Coverage Analysis with all hardening components

The Hardened core results to be more testable as depicted with a resulting Fault Covarage gain of 1.65 (strobed versus not strobed).