Evaluation Platform For Testing Fault Tolerance: Testing Reliability of Smart Electronic Locks

Jakub Podivinsky, Jakub Lojda, Richard Panek, Ondrej Cekan, Martin Krcma, Zdenek Kotasek
Faculty of Information Technology, Brno University of Technology, Centre of Excellence IT4Innovations
Bozetechova 2, 612 66 Brno, Czech Republic
Email: {ipodivinsky, ilojda, ipanek, icekan, ikrcma, kotasek}@fit.vutbr.cz

***Abstract*—This research paper presents the examination of the influences of faults on a control unit of smart electronic locks. A stepper motor is often used as an actuator of such smart locks and its motor controller is usually implemented in a processor. The aim of this paper is to examine the impact of faults occurring in the control processor. It should be noted that faults in such electronic systems can also be induced artificially, usually with ulterior motives. The processor can be implemented in an FPGA (Field Programmable Gate Array) in order to be able to emulate HW faults inside the processor. This allows us to use previously developed evaluation platform for fault tolerance testing. This platform allows us to monitor the impact of faults both on electronic and mechanical parts of electro-mechanical system. In this paper, the evaluation of faults artificially injected in FPGA-based processor is proposed. Experiments with both single and multiple fault injections were performed. In our research, we found out that a fault in the same position of the design does not always affect the electronics in the same way. Also, the mechanics may still operate correctly despite the electronics failure.**

***Keywords*—Electronic Lock, Stepper Motor, FPGA, Fault Tolerance, Fault Injection.**

I. Introduction

Nowadays, *smart devices* [1] are on the rise, their goal is to make our lives more efficient, simpler and more pleasant. The smart electronic lock [2] is an example of smart device that can be met in our everyday life. Connecting electronics with mechanical elements and a remote server brings new possibilities for users to control these locks. The smart electronic locks have many advantages, since they can be controlled remotely.

Various types of faults can arise in electronic systems, especially if such systems are operated in environments with increased level of electrostatic electricity, increased occurrence of charged particles, etc. It should be noted that faults can also be injected artificially, usually with malicious intentions. For example, sensitive data from an embedded memory (parts of an algorithm, encryption keys, etc.) can be extracted by bumping attacks [3]. Attacks on smart cards based on fault injection which modifies the behavior [4] can serve as another example. Unauthorized or accidental unlocking of electronic locks may also be caused by fault injection. Opening a lock or preventing its closing can cause financial losses or endanger human health or life. Unauthorized unlocking can be caused by various types of attacks, either to the server part of the whole system, or directly to the lock in the door [5]. Intently inducing faults is one of the possible attacks which is yet an unexplored topic that we will deal with in our research.

The evaluation of the effects of faults on an electro-mechanical system is in focus of our research. Especially, we focus on systems composed of SRAM-based FPGAs. Another example of electro-mechanical system where faults can cause undesirable consequences, is an electronic lock. In this work, we are going to use previously developed tools to analyze the impact of faults on electronic locks. Electronic locks are usually controlled by an embedded processor that can be implemented in an FPGA which gives us the possibility to simulate faults. We have identified three main goals that we are going to achieve during our research targeted on electronic locks which are in detail presented in paper [6]:

1) *The evalution of faults injected directly into stepper motor control signals and the estimation of the risk of an unauthorized unlocking* (solved in [6]).
2) *To implement the stepper motor controller with a processor configured into an FPGA and evaluate: a) the impact of faults injected into processor and b) the possibility of unauthorized unlocking (topic of this paper).*
3) *To verify the possibility of using standard fault tolerance techniques for eliminating unauthorized lock unlocking.*

During solving of the goals we will use three levels of evaluation which were presented in [6] with various architectures of component realization and interconnection. This paper is based on interconnections of a PC for simulation running and an FPGA where electronic controller is implemented.

This paper is organized as follows. Electronic lock components are presented in Section II. Section III introduces an evaluation platform for monitoring the impact of faults on electro-mechanical applications. Section IV describes experimental environment architecture. Experiments with fault injection are presented in Section V. Section VI concludes the paper and mentions plans for our future research.

II. Electronic lock

Smart electronic locks are quite complex electronic devices. The basis of the lock can be divided into three blocks – control module, I/O module and motor module [7]. The important block is the motor module which performs the operation with the mechanical part of the lock. Stepper motor is very often found in electronic locks [8] [9], that is why we focus on experiments with stepper motor in our research.

The stepper motor is a DC electric motor whose full rotation can be divided into several equal steps. The stepper motor is controlled by input pulses (typically square pulses) that precisely rotate the shaft position based on an angle which is given by the number of motor steps. It consists of a cylindrical rotor, a number of stators, a number of yokes, and a set of coils [10]. We have chosen a conventional bipolar stepper motor with a permanent magnet in its rotor which operates on the attraction or repulsion between the rotor and the stator electromagnets. The particular model of this type of stepper

978-1-7281-3427-7/20/31.00/31.00 ©2020 IEEE

motor that we have chosen is 28BYJ-48 [11]. It is a small stepper motor operating at 5V which is equipped with a 1/64 transmission gearbox. It has 4 phases with a single step angle of 5.625°/64 and 4,096 steps are needed to perform the full rotation (64 steps without the gearbox).

III. The Verification-based Evaluation Platform

The main evaluation tool used in this paper is our previously developed platform [12] for the evaluation of the impact of faults injected in electronic part of electro-mechanical system. The proposed evaluation platform is based on well known functional verification technique [12]. The core of the platform is an ML506 evaluation board with Virtex 5 FPGA which allows us to inject faults directly to the FPGA in which the electronic control unit is implemented. The fault injector based on Partial Dynamic Reconfiguration (PDR) is implemented in a computer and faults are injected through a JTAG interface. The communication between the simulation of mechanical part running on the computer and the electronic controller on the FPGA is accomplished through the Ethernet.

Functional verification is used as a tool for monitoring the impact of faults both on the electronic controller and the mechanical part of an electro-mechanical system. The modification of basic functional verification is shown in Figure 1. The main difference is that Device Under Test (DUT) is moved to the FPGA, which allows to inject artificial faults directly into FPGA and monitor their impact. The versatility of the proposed platform is based on the fact that functional verification is usually used during electronic systems development. Therefore, the verification environment and the reference model (the most important elements dependent on the evaluated system) are available from the previous stage of system development and can be used for a fault tolerance evaluation. The verification scenario generation is usually a part of the verification environment or we can use our previously developed universal generator [13]. An important condition for using the platform is that an electronic controller can be implemented in an FPGA. The DUT implementation in the FPGA and proper communication with the software part of verification environment are realized by the *driver* and the *monitor* components. These components are partly universal, but they need to be customized for a particular DUT.

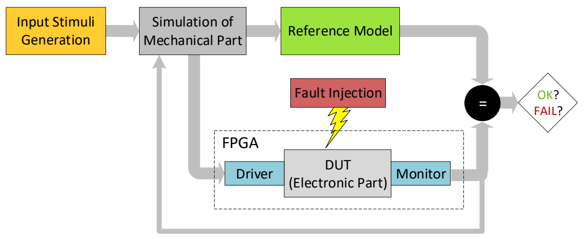

Fig. 1: The general concept of the use of functional verification for monitoring impact of faults.

The mechanical part is also an important element which allows to monitor the impact of faults not only on electronics, but also on mechanics. It is not important whether it is a real mechanical part or its simulation. The availability of sensors that provide feedback of the mechanical part behavior is important. The values provided by these sensors are monitored by the verification environment which checks if the system behaves according to its specification. Usually, the use of simulation leads to a faster testing and is usually cheaper.

Together with the evaluation platform we proposed the process of evaluation which is divided into three phases: 1) Classical simulation-based functional verification is done. After this phase we can be sure that the detected failures in following phases are caused by the injected faults. 2) During the second phase, modified verification environment (DUT implemented on FPGA) is used and artificial faults are injected. The output of this phase is a list of faults with the impact on electronic controller. 3) The third phase is focused on the evaluation of the behaviour of the mechanical part if the electronic part is corrupted by a fault.

IV. Experimental Setup

In this work, we deal with monitoring the impact of faults on the electronic lock, more precisely on the main mechanical element which is the stepper motor and its control processor. In this case, the verification environment is modified since no feedback is used when the stepper motor is controlled only by control signals. The measurement of the angle of rotation (both continuous and resulting angle) is needed for monitoring the behaviour of the mechanical part. The angle measurement can be easily realized both in the case of experiments with a real stepper motor (e.g. a stepper motor can rotate a potentiometer), and in the case of simulation. Figure 2 shows the use of functional verification-based platform for checking the impact of faults injected into the control processor implemented in FPGA.

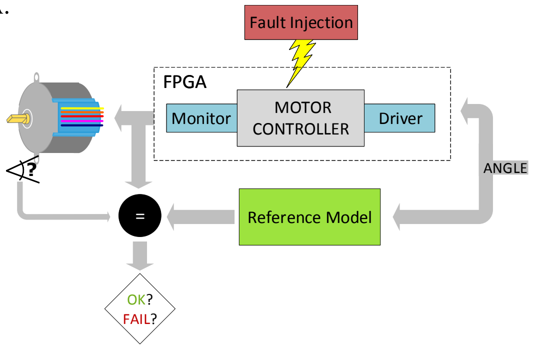

Fig. 2: The use of functional verification for monitoring impact of faults on stepper motor controller.

For our experimental purposes, we chose the NEO430 soft-core processor [14] which is based on the Texas Instruments MSP430 [15] instruction set architecture. The NEO430 is based on the Harvard architecture and uses program (IMEM) and data (DMEM) memory with configurable sizes. The processor implements configurable peripherals like a timer, a watchdog, UART and SPI serial interfaces (implemented together as a USART unit), general purpose IO ports and an internal bootloader. The peripheral modules are optional due to the possibility to reduce the size of the implemented system. In our experimental implementation (shown in Figure 3), we use UART interface for debugging purposes, Custom Functional Unit (CFU) is used as an input interface for the required number of steps and output signals for stepper motor are provided through the General-purpose inputs and outputs (GPIOs). The program is stored in ROM memory, it means that the program starts working after the processor is activated without any need to load the program from an external memory. The program itself repeatedly sets the stepper motor signals according to its specification. It is possible to parameterize the required number of steps through a signed number on the input ”STEPS”. The negative number causes opposite direction of rotation. The outputs include four signals for the stepper motor control.

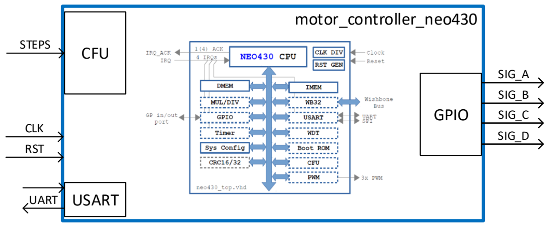

Fig. 3: The use of functional verification for monitoring impact of faults on the stepper motor controller.

We decided to use MATLAB and Simulink [16] for the simulation of the stepper motor, especially Simscape library [17] which proposes the stepper motor simulation. This model is generic, however, proper parameters from the stepper motor datasheet [11] must be used. More accurately, it is the 4-phase stepper motor with a permanent-magnet rotor. The output from the complete model is a current angle of the stepper motor. It is also necessary to model a voltage controller which converts logic inputs to the electric impulses which excite the motor coils.

V. Experiments and Experimental results

The following subsections describe experiments corresponding to the second and the third phase of the evaluation process. There is no space for the first phase which is not so important from the reliability point of view. Faults were injected during the second and the third phase into the experimental processor according to two strategies – single bit-flip faults injected at the start of experiment and multiple bit-flip faults injected at a regular interval. We injected the faults into the bits of the bitstream that are utilized by LUTs of the controller implemented in the FPGA. The total number of the utilized bits was 58496. The impact of faults was monitored on the output of the electronic controller and then on the behaviour of the mechanical part with corrupted electronic controller.

During the mechanics simulation we recorded the rotation angle. The collected information covers the minimal, maximal and the final rotation angle of the stepper motor. The minimal angle was identical during all iterations as the motor always started at the same position and it never started to rotate in the opposite direction. The maximal and the final angle depended on an injected fault effect. Almost always the angles were equal. The maximal rotation angle is the most important variable in the context of a possible unwanted locking or unlocking the lock.

A. *Single fault injection*

Since an exhaustive evaluation would be demanding, we uniformly selected at random 6000 random bit faults that were injected into the FPGA. We performed five iterations of experiments with the selected bits in order to evaluate whether the controller behaviour is affected randomly by the particular injected fault or it remains the same. We compared the iterations in order to extract common features of the fault affected controller behaviour. The experiments were repeated five times using the same set of faults. Table I illustrates the failure rate of the electronic controller in all three experiment iterations (rows 1, 2, 3, 4 and 5). The *Electronic failure—Total* column covers the number of faults that caused the electronics failure. We divided the types of the electronics failures into three classes: 1) the premature stopping—*Stuck*, 2) unending controller operation—*Time-out* and 3) the correct termination, although with mismatching values—*Mismatch*. The numbers of failures of the particular types are listed in Table I. In the third phase we evaluated the mechanics behaviour. The *Mechanic OK—Total* column shows the number of cases when the electronics has failed but the mechanics functioned correctly and reached the proper position as was evaluated in the mechanics simulation. This number of faults therefore affected the electronic controller without affecting the overall lock behaviour at the same time. As can be seen, the highest number appears in the last column, that is when the controller terminated its activity correctly with mismatching values. The reason is we evaluated the electronics failure strictly when we considered even a single mismatching output as a failure.

TABLE I: The results of single and multiple injection experiments with the failures classification.

Iter.	Electronic failure				Mechanic OK			
	Total	Stuck	Time-out	Mis-match	Total	Stuck	Time-out	Mis-match
1	633	159	260	214	210	5	2	203
2	633	174	270	189	186	5	3	178
3	572	93	145	334	331	6	3	322
4	624	172	269	183	183	5	3	175
5	574	100	147	327	327	6	3	318
Multi	5772	1248	1901	2605	592	35	26	531

The set of bits that have proven to cause a failure was always a bit different between the iterations. We compared all combinations of two (e.g. $1 \cap 2$, $1 \cap 3$...), three (e.g. $1 \cap 2 \cap 3$, $1 \cap 3 \cap 4$...) and four (e.g. $1 \cap 2 \cap 3 \cap 4$, $1 \cap 3 \cap 4 \cap 5$...) sets of results. The result statistics are listed in Table II. The table does not cover all the combinations but describes only the minimum, average and the maximum of all of the combinations. The last row contains a combination of all iterations, so it presents a precise value. The *Electronic Failure* column covers the number of faults that caused an electronic failure in all the compared iterations. The *Mechanic OK* column covers the number of faults which have not affected the mechanics during any iteration. Even though, the same fault caused the electronics failure, the electronic did not fail always in the same way (stuck, time-out and mismatch). The column *Different El. Failure* covers the number of such faults. As can be seen, the growing size of compared sets leads to a smaller number of faults that cause electronic failure for each iteration. At the same time, the number of faults that did not cause any mechanic failure is lowering while on the other hand the number of faults with different electronic failure grows. We also detected unique bits, out of the total 6000, that caused a failure of the controller when fault was injected during almost one iteration which are shown in the last column. The number of these unique bits is rising with the growing size of the compared sets.

TABLE II: The iterations results comparison (single faults).

Num. of Comp. Runs	Electronic Failure			Mechanic OK			Different El. Failure			El. Failure Unique Bits		
	min	avg	max	min	avg	max	min	avg	max	min	avg	max
2	509	527	560	46	109	260	36	146	213	633	687	715
3	430	481	511	13	55	86	112	207	292	710	728	742
4	460	467	475	30	37	44	208	216	224	740	750	755
all	395			32			312			764		

Moreover, we examined the motor rotation angle. Figure 4a contains a boxplot chart which displays the maximal angle for all the iterations. As the chart illustrates, the required rotation angle was 4500°. The majority of the electronics failures led to a smaller final rotation angle. Only a small number of faults caused a bigger rotation angle. If the goal of the fault injection is to unlock the lock unauthorized, most likely it will not be reached. On the contrary, if the goal is to prevent the door to lock properly, the chances are significantly higher.

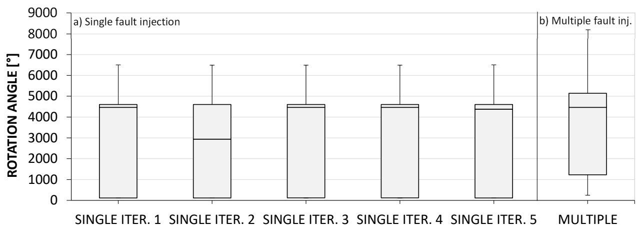

Fig. 4: Boxplot graph with rotation angle for (a) three experiment iterations with single injection and (b) multiple injection.

B. Multiple fault injection

Furthermore, we inspected the impact of a multiple fault injection on both the electronics and the mechanics. We performed 6000 experiments in this scenario as well. We uniformly-at-random injected a single fault into utilized bits of the motor controller design each five seconds. The particular experiment was terminated when a fault impact was detected or after 280 seconds if the faults proved to have no effect. The time limit is important as some faults do not cause a significant number of incorrect transactions on the controller output as other faults, but they cause that the controller would not stop the motor rotation when expected. The multiple injection proved to cause a significantly higher rate of the controller failures. Out of the 6000 total, 5772 experiments resulted in incorrect controller output transactions when multiple faults were injected.

The last row of Table I shows the number of faults that led to the electronics failure and their classification. It also contains the number that did not lead to the mechanics failure as well as the relations with the electronics failures types. As can be seen, the overall behaviour is similar to single fault cases, therefore the mechanics did not fail despite the inputs mismatch. The mechanics reactions on the incorrect controller outputs are illustrated in Figure 4b. As can be seen, unlike in the case of single faults, the smallest rotation angles did not occur and the span between the minimum and maximum grew.

VI. Conclusions and Future Research

In this paper we realized the second phase of our evaluation of fault injections effects on the electronic controller of an electronic lock implemented in HW. The stepper motor controlled by the controller implemented in a processor was used as our experimental platform. The processor was implemented in FPGA which allowed us a repeated and nondestructive testing of the faults effects. We examined the faults effects on electronics and mechanics of the lock, when we inspected how the faults affected the rotation of the motor. For these purposes, we further developed our fault tolerant system testing platform that we presented in our previous papers. The results indicate that random single faults cause a failure in about 10 % cases which is not suitable for an exploitation. The same faults proved not to always cause the same failures during the multiple iterations, moreover they did not lead to the mechanics failure every time. The mechanics failures most often led to a smaller angle rotation than desired, therefore these failures may prevent the door to lock. The failures proved to be more probable when injecting multiple faults.

In our future research, we will focus on the third phase which will utilize simulation accelerated on an FPGA in order to speed up the evaluation.

Acknowledgements

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II), the project IT4Innovations excellence in science – LQ1602, the BUT project FIT-S-17-3994 and the JU ECSEL Project SECREDAS (Product Security for Cross Domain Reliable Dependable Automated Systems), Grant agreement No. 783119.

References

[1] C. Salzmann, S. Govaerts, W. Halimi, and D. Gillet, "The smart device specification for remote labs," in *Proceedings of 2015 12th International Conference on Remote Engineering and Virtual Instrumentation (REV)*. IEEE, 2015, pp. 199–208.

[2] Y. T. Park, P. Sthapit, and J.-Y. Pyun, "Smart digital door lock for the home automation," in *TENCON 2009-2009 IEEE Region 10 Conference*. IEEE, 2009, pp. 1–6.

[3] S. Skorobogatov, "Flash memory bumpingattacks," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2010, pp. 158–172.

[4] J.-B. Machemie, C. Mazin, J.-L. Lanet, and J. Cartigny, "Smartcm a smart card fault injection simulator," in *2011 IEEE International Workshop on Information Forensics and Security*. IEEE, 2011, pp. 1–6.

[5] M. Pavelić, Z. Lončarić, M. Vuković, and M. Kušek, "Internet of Things Cyber Security: Smart Door Lock System," in *2018 International Conference on Smart Systems and Technologies (SST)*. IEEE, 2018, pp. 227–232.

[6] O. Cekan, J. Podivinsky, J. Lojda, R. Panek, M. Krcma, and Z. Kotasek, "Testing Reliability of Smart Electronic Locks: Analysis and the First Steps Towards," in *Proceedings of the 2019 22nd Euromicro Conference on Digital System Design (DSD)*. IEEE, 2019, pp. 506–513.

[7] Y. T. Park, P. Sthapit, and J. Pyun, "Smart digital door lock for the home automation," in *TENCON 2009 - 2009 IEEE Region 10 Conference*, Jan 2009, pp. 1–6.

[8] Z. Fonea, "Electronic lock system," Nov. 14 2000, uS Patent 6,147,622.

[9] G. K. Verma and P. Tripathi, "A digital security system with door lock system using RFID technology," *International Journal of Computer Applications*, vol. 5, no. 11, pp. 6–8, 2010.

[10] G. Saether, "Electric stepper motor," Nov. 29 1994, uS Patent 5,369,324.

[11] Kiatronics®, "28BYJ-48 - 5V Stepper Motor," http://robocraft.ru/files/datasheet/28BYJ-48.pdf, 2015, accessed: 2019-03-26.

[12] J. Podivinsky, O. Cekan, J. Lojda, M. Zachariasova, M. Krcma, and Z. Kotasek, "Functional Verification based Platform for Evaluating Fault Tolerance Properties," *Microprocessors and Microsystems*, vol. 52, pp. 145 – 159, 2017.

[13] O. Cekan, J. Podivinsky, and Z. Kotasek, "Random Stimuli Generation Based on a Stochastic Context-Free grammar," in *Proceedings of the 2016 International Conference on Field Programmable Technology*. IEEE Computer Society, 2016, pp. 291–292.

[14] S. Nolting, "NEO430 Processor," https://github.com/stnolting/neo430, 2018.

[15] Texas Instruments. (2018, Feb.) Msp430 ultra-low-power sensing & measurement mcus. [Online]. Available: http://www.ti.com/microcontrollers/msp430-ultra-low-power-mcus/overview.html

[16] MathWork®, "MATLAB and Simulink," https://www.mathworks.com/, 2018, accessed: 2019-03-20.

[17] MathWork®, "Stepper motor," https://www.mathworks.com/help/physmod/sps /powersys/ref/steppermotor.html, 2019, accessed: 2019-03-20.