I. INTRODUCTION
All types of electronic systems can malfunction due to external factors. The main sources causing faults within electronic components are radiation, electromigration and electromagnetic interference. The evaluation of a fault-tolerant system is a complex task that requires the use of different levels of modeling. Compared with other possible approaches such as proving or analytical modeling, fault injection is particularly attractive.

Fault injection is a key requirement of functional safety standards like the Automotive ISO 26262 and is highly recommended during hardware integration tests to verify the completeness and correctness of the safety mechanisms implementation with respect to the hardware safety requirements.

This article reviews the use of processors in the automotive industry, the origin of faults in processors, architectures of fault tolerant processors and techniques for processor verification with fault injection.

We then propose an emulation based framework for performing fault-injection experiments on embedded processor architectures.

II. THE USE OF ELECTRONICS IN THE AUTOMOTIVE INDUSTRY
The integration of electronics in automobiles began during the 1970’s and became well established by the 1980’s. Today’s top-of-the-line vehicle uses over 80 microcontrollers, and an even greater number of power semiconductors and smart power ICs.

1. Electronic Driver-Assisting Systems
In these cases the existing mechanical systems are supported by electronics. Examples include antilock braking system, traction control system, electronic stability program and brakes assist. This is a fail-safe design which means that at least a basic part of the system’s functionality is provided in case the electronic system fails.

2. Drive-by-Wire Systems with Mechanical Backup
Examples include electric power steering, electronic braking and throttle systems. While these systems are electronically controlled and operated, there is still a mechanical backup system if an electrical problem develops which makes the system fail-safe.

3. Drive-by-Wire Systems without Mechanical Backup
The successful use of fly-by-wire systems in aviation along with the positive experience of drive-by-wire systems with mechanical backup for braking and power steering have led to the development of complete drive-by-wire systems that reduce the cost of a vehicle.

4. Higher-Level Automotive Control
These are systems that may influence several other basic systems on the vehicle. As an example, adaptive cruise control allows a vehicle to maintain a certain gap between itself and the vehicle in front, or to maintain a speed previously set by the driver by controlling both the engine and the brakes.

III. THE ORIGIN AND MITIGATION OF FAULTS IN PROCESSORS
The main causes for faults within electronic components are radiation, electromigration and electromagnetic interference.

1. Radiation Effects on Integrated Circuits
One source of alpha particles is from the radioactive impurities found mainly in the package materials and to a lesser extent in the materials used for the fabrication of the semiconductor device, with uranium and thorium having the highest radioactivity among them. The second source comes from extraterrestrial cosmic rays that bombard the earth’s surface. The cosmic rays mainly consist of protons, neutrons, pions and muons with different energies. Cosmic rays also include particles that originate from the sun, with relatively low energies. When they penetrate the Earth’s atmosphere, some start reacting with other particles of the atmosphere thus the levels of radiation depend heavily on...
the altitude. Approximately 1% of the cosmic ray’s neutrons reach the Earth’s surface and they present a very wide energy spectrum.

2. EMC for Integrated Circuits
There are two main issues concerning electromagnetic compatibility (EMC) and ICs. The first is the electromagnetic energy emission of ICs while they are operating and the second is the susceptibility of ICs to electromagnetic waves from the operational environment. With the ever increasing use of electronic systems the electromagnetic environment becomes more complex, making the EMC requirements of systems more challenging to meet.

Crosstalk between the metal lines within the chip is also a significant source of errors, especially in multi-layered chips.

3. Electromigration
Over a period of time the flow of electrical current through metal tends to displace metal ions. In some places voids open up in the wires leading to open circuits and in other places ions are deposited causing shorts. This phenomenon is known as electromigration.

4. Impact of New CMOS Technologies
The stored charge that is used within digital circuits to represent data has decreased dramatically in recent years as a direct result of the decreased power supply. This means that the critical charge has also decreased with a negative impact on the error susceptibility to particle hits.

As frequency increases, the errors observed will be dominated by transient faults originating from combinational logic rather than single event upset (SEU) on sequential logic. The increasing frequency will also tend to increase the occurrence of multiple-bit errors, since the duration of the transient pulse may overlap more than one clock edge.

Higher currents flow through the power supply lines, consequently increasing the susceptibility to electromigration. Furthermore, the increased number of metal layers makes crosstalk between the interconnection lines more probable as the distance between them decreases.

IV. ARCHITECTURE OF FAULT TOLERANT PROCESSORS
Processors for Fault Tolerant applications are typically required to achieve the following targets: high performance, low cost, low power dissipation, and reliability. The problem is that most available processors and integrated systems-on-chip achieve only some of the targets and fail on others. This is indicated below when relative advantages and disadvantages are listed, and exemplified in later sections. The following sections survey different industry approaches to this tradeoff.

1. Radiation Hardened (RH) Processors
In this approach processors are fabricated on dedicated RH processes. Advantages include: High tolerance to radiation effects, thanks to the RH process. In some cases, such processors achieve high performance. This can be especially true when using custom design methods similar to those employed for the design of COTS high-performance processors. This approach can offer high level of integration, including the inclusion of special I/O controllers dedicated to fault tolerant applications.

Disadvantages of using RH processes include: High cost—RH processes have limited use and the high price of modern fab; Not widely available—there are only about two RH fabs in the USA and no similar advanced processes elsewhere. Use of the RH processes in the USA is International Traffic in Arms Regulations (ITAR) controlled and is not widely available to non-USA customers; Lags several generations behind commercial off-the-shelf (COTS) processors, in terms of performance and power—typical RH processes are based on 150nm CMOS, while high-end COTS processes belong to the 28nm generation, about six processing generations more advanced; SEU rate getting worse—the RH process enables a fixed SEU per bit but as the chips become more advanced and contain more memory and more flip-flops, the total SEU per chip is higher.

2. Radiation Hardening by Design (RHBD) Processors
In this approach radiation hardness is achieved by design techniques in the layout, circuit, logic and architecture areas, hence the name. Advantages include high tolerance to radiation effects, medium cost—more expensive
than COTS processors, mostly due to low production quantities and high cost of qualification, but at the same time, they are less expensive than RH processors thanks to using a regular commercial fabrication process. Finally, RHBD processors can offer high integration since they are designed as ASIC and since typically the CPU itself takes only a small portion of the silicon die.

Disadvantage is RHBD processors are usually slower than COTS processors since they are designed as ASIC chips and not as custom processors.

3. Single COTS Processor with Time Redundancy (SIFT)
In this approach, a single COTS processor is used together with Software Implemented Fault Tolerance (SIFT), which executes the entire software or certain software sections twice or more. There are two levels of granularity: Instruction level redundancy, where each instruction is executed twice and additional instructions compare the results, requiring compiler transformation of the code, and procedure level redundancy, where the programmer writes the code to invoke certain procedures twice, compare the results and use software for recovery in case of mismatch. The latter approach may also require some additional hardware to protect the critical data and the critical software. The main advantage of this approach is that it is relatively inexpensive.

Disadvantage is the major performance penalty due to the computational overhead.

4. Duplex COTS: DMR
This architecture employs two equal COTS processors (aka Dual Modular Redundancy, DMR), a matching hardware, and software for recovery from mismatches. There is no voting as there are only two copies of execution. On mismatch, computation is cancelled and repeated by software control. DMR offers high performance and is relatively inexpensive.

The disadvantages are that DMR requires special hardware and software for matching and recovery, and that modern COTS processors are sometimes unpredictable at the clock cycle level, due to methods of internal branch speculation and other algorithms that are designed to boost performance. Forcing two such processors to execute in lock-step every clock cycle may require significant slowdown of the processors.

5. Triple COTS: TMR at the system level
Triple Modular Redundancy (TMR) architectures combine three COTS processors and voting logic. The processors do not need to be stopped on SEU. TMR offers high performance and high SEU tolerance.

Disadvantages of TMR is high cost, requiring large area and power, as well as special hardware for voting and usually additional hardware and software for recovery from internal SEU errors (inside the processors) that cannot be fixed by voting and require scrubbing or reset.

6. TTMR on COTS VLIW processors
COTS VLIW processors execute multiple instructions in parallel, and the parallel instruction streams are pre-programmed. Each instruction can be executed three times and the results can be compared and voted, all within the same VLIW processors. TTMR offers high performance (in fact, TTMR processors are the fastest available space processors today) and high SEU tolerance, thanks to embedded TMR mechanism, but it is expensive, is limited to VLIW processors, and is hard to generate code for. The code executes two copies of an instruction, compares the result, on mismatch executes the same instruction the third time and compares for majority voting.

V. PROCESSOR VERIFICATION WITH FAULT INJECTION TECHNIQUES
Fault injection is the deliberate change of the state of an element within a computer system. FI is critical in the development of fault tolerant systems. They are mainly used to assess the effectiveness of fault and error detection mechanisms and to help predict the system's error rate. 6

1. Software Implemented Verification (SWIFI)
The use of SWIFI tools has been very popular mainly because they are easy to implement and adapt to a target system. They are also cost-effective since they do not
require extra hardware. Furthermore they are usually fast since they do not introduce significant delay in the execution of the target applications.

2. Simulation Based Verification
Simulation is favored since it allows the testing of fault tolerant systems very early in the design stage. If a HDL description of the system is available, testing through simulation can be performed in great detail and it is potentially very accurate since it gives realistic emulation of faults and detailed monitoring of their consequences on the system.

3. Physical Level Validation
Injection of physical faults on the actual target system hardware can be achieved through pin-level fault injection, heavy-ion radiation, electromagnetic interference and laser fault injection. The major advantage of these approaches is that the environment is realistic and the results obtained can give accurate information on the behavior of the system under such conditions.

4. FPGA Based Verification
This technique can allow the designer to study the actual behavior of the circuit in the application environment, taking into account real-time interactions. However, when an emulator is used the initial VHDL description must be synthesizable

5. Emulation Based Verification
Emulation enables pre-silicon fault injection and debug at hardware speeds, using real-world data. The scenarios of real-time software and hardware fault injection debug with simulation-like visibility are achieved.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical Level</td>
<td>• High time-resolution for hardware triggering and monitoring</td>
<td>• Limited set of injection points and limited set of injectable faults</td>
</tr>
<tr>
<td></td>
<td>• Well suited for the low-level fault models</td>
<td>• Requires special hardware</td>
</tr>
<tr>
<td></td>
<td>• Not intrusive</td>
<td>• Debug is hard</td>
</tr>
<tr>
<td></td>
<td>• No model development or validation required</td>
<td>• Limited coverage</td>
</tr>
<tr>
<td></td>
<td>• Able to model permanent faults at the pin level</td>
<td></td>
</tr>
<tr>
<td>Physical Level,</td>
<td>• Can be targeted to applications and operating systems</td>
<td></td>
</tr>
<tr>
<td>(cont.)</td>
<td>• Experiments can be run in near real-time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• No specific hardware</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• No model development or validation required</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Can be expanded for new classes of faults</td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td>• Limited set of injection instants</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• It cannot inject faults into locations that are inaccessible to software</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Require instrumentation of the source code</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Limited observability and controllability</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Difficult to model permanent faults</td>
<td></td>
</tr>
<tr>
<td>Simulation</td>
<td>• Support all abstraction levels</td>
<td>• Slow</td>
</tr>
<tr>
<td></td>
<td>• Non-intrusive</td>
<td>• Model is not readily available.</td>
</tr>
<tr>
<td></td>
<td>• Full control of both fault models and injection mechanisms</td>
<td>• No real-time faults</td>
</tr>
<tr>
<td></td>
<td>• Does not require any special-purpose hardware</td>
<td>• Coverage is limited</td>
</tr>
<tr>
<td></td>
<td>• Maximum observability and controllability</td>
<td></td>
</tr>
</tbody>
</table>
VI. A FRAMEWORK FOR VELOCE® BASED FAULT INJECTION

We suggest the following scheme for implementing a generic fault injection system using the Veloce emulator.

1. Faults, Errors, Failures

a) **Fault**—A fault is a deviation in a hardware or software component from its intended function. Faults can be categorized into permanent and transient faults by their duration.

b) **Error**—An error is the manifestation of a fault on the observed interfaces.

c) **Failure**—A failure is defined as the deviation of the delivered service from the specified service.

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| Simulation, (cont.) | • Allows performing reliability assessment at different stages in the design process  
• Able to model both transient and permanent faults | |
| FPGA Prototype | • Injection time is faster than simulation based techniques  
• The experimentation time can be reduced by implementing the input pattern generation in the FPGA; these patterns are already known when the circuit to analyze is synthesized | • High effort of partition and synthesis and limited signal visibility resulting in long debug cycles  
• Intrusive instrumentation techniques  
• Testing functional behavior of injected fault.  
• Unanticipated behavior analysis is hard |
| Emulation | • Support most abstraction levels  
• Supports Netlist  
• Real-time faults  
• Full coverage is achievable  
• Mostly Non-intrusive  
• Full control of both fault models and injection mechanisms  
• Maximum observability and controllability  
• Allows performing reliability assessment at different stages in the design process | • Requires special hardware and expertise  
• Analog design is not directly supported  
• Fault injection TB overhead can hamper acceleration performance |

Figure 1: Fault, errors, system failures

2. Flow

1. Setup phase

   a. Determine an injection fault distribution function: Uniform Random (UR), Activity Based Random (ABR), and Manual Direct (MD).

   b. For ABR:

      i. Run a test and measure activity with the Switching Activity Interchange Format (SAIF).

      ii. Extract subset of FF’s for fault injection.
c. Create a Fault Injection DB.
d. Instantiate in top with a Golden Model (GM) DUT and Fault Injected Model (FIM) DUT.

2. Emulation Phase
   a. Run a Golden Model (GM) and capture all interface signals to log.
   b. Run Fault Injected Model (FIM) and capture interface signals to log.
   c. Post-process compares GM vs FIM.

3. Evaluation phase
   a. Analyze results.
   b. Create reports.

VII. CONCLUSIONS

Reliability and safety are of major importance to the introduction of automotive drive-by-wire ISO 26262 compliant systems. Their required high safety integrity necessitates that all electronic components will be fault tolerant with regard to failures in electronic hardware and software. Fault-tolerant processors properties can be obtained primarily by static or dynamic redundancy, leading to systems that are fail-operational for at least one failure.

The comparison of different fault injection techniques leads to the conclusion that Emulation based approach has key advantages for achieving the goals required for fault-tolerant tolerance.

Figure 2: Fault Injection Testbench Architecture
VIII. REFERENCES

1. ISO 26262 Road vehicles – Functional safety – Part 5: Product development: hardware leve
2. ISO 26262 Road vehicles – Functional safety – Part 10: Guideline
VERIFICATION ACADEMY
The Most Comprehensive Resource for Verification Training

21 Video Courses Available Covering
- Intelligent Testbench Automation
- Metrics in SoC Verification
- Verification Planning
- Introductory, Basic, and Advanced UVM
- Assertion-Based Verification
- FPGA Verification
- Testbench Acceleration
- Power Aware Verification
- Analog Mixed-Signal Verification

UVM and Coverage Online Methodology Cookbooks
Discussion Forum with more than 5000 topics
UVM Connect and UVM Express Kits

www.verificationacademy.com