Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 07, July-2022

FMEDA based Fault Injection to Validate Safety Architecture of SPI

Vinod Sai E
Department of Electronics and Communication
R V College of Engineering
Bengaluru, India.

Praneeth Uddagiri
Staff Design Verification Engineer
Analog Devices, Inc.
Bengaluru, India.

Namita Palaeha
Assistant Professor
Department of Electronics and Communication
R V College of Engineering
Bengaluru, India.

***Abstract*— The integration of advanced technologies into Electrical Vehicles (EV) has been increasing in recent times, so it has become crucial to evaluate the risk of the technologies that are deployed into it. Functional safety is something which is required in automobiles for ensuring safety of human lives. Battery Management System (BMS) chip is one such important component of EV which uses Serial Peripheral Interface (SPI) as a peripheral to communicate with external ICs to monitor battery state used in an EV. To get this chip functional safety certification, every block in on the chip should possess safety architecture around it and functional safety verification for the same should be done. This paper performs Failure Mode Effects and Diagnostic Analysis (FMEDA) based Fault Injection to validate the safety architecture of SPI as recommended by ISO 26262. Diagnostic coverage of 97.2 % is achieved for the single point faults in the SPI block which makes it sufficient to achieve Single Point Fault Metric (SPFM) of 99% for the entire chip.**

Keywords*— *BMS, EV, FMEDA, Fault Injection, Functional Safety, ISO-26262, SPI, SPFM.

I. INTRODUCTION

With advanced electronics bringing the automotive industry to higher levels, automotive Original Equipment Manufacturer (OEM)s require safety-certified semiconductors. The automation of E/E systems in the automobiles is evolving into a complex process which are designed to deliver many advanced features like electric power steering, ADAS, braking system, airbags and many more, these need safety assurance. Due to the incorporation of these cutting-edge technology into automobiles, it is now necessary for manufacturers to evaluate and examine the risk related to the technology they want to use. Functional safety is something which is required in automobiles for ensuring safety of human lives. Functional Safety (FuSa) is the idea that an overall system will continue to work reliably and as intended even if an unexpected event, occurs. Additionally, the systems guarantee that there is no unacceptable danger of physical harm or damage. Automotive features are possible because of the electronics that is going into that. This led to the requirement of FuSa semiconductor chips in the automotive industry. For System on Chip (SoC)s, especially as one moves into sub-micron designs, susceptibility becomes greater. High levels of safety can distinguish this product and change consumers opinions of it.

BMS is one such SoC used in EVs which should meet functional safety standards. A BMS is any electronic system that controls a rechargeable battery's environment, protects the battery from running outside of its safe operating parameters, monitors the battery's condition, reports derived secondary data, balances, and authenticates the battery. Consider some scenarios where a BMS chip that doesn't have the capability of measuring current. An external IC that measures current tries to send current data that needs to be fed to the BMS chip. Or, monitoring the voltages of battery cells. To send and/or receive information from one another, these ICs must be able to communicate with one another. So, communication protocols are vital for a BMS with multiple ICs to be able to communicate with each other. SPI is a protocol that provides an easy to implement and very low-cost interface between a micro-controller and its peripherals. The SPI protocol uses a serial clock that is generated by the master to synchronize the master and slave devices for transmissions and receptions. One device is considered the master of the bus (BMS is the master in this case) and all the other devices (peripheral ICs or even other micro-controllers) are considered as slave devices.

A computed estimate of the rate of hazards caused by random hardware failures is required by ISO 26262. During the product development phase, hardware and software are actually developed by first trying to analyze the BMS at the system level, then at the component level, deriving the safety requirements from the functional safety concept, developing a system architecture, and defining safety mechanisms for failure detection and avoidance. Using an FMEDA, which identifies possible failure modes and the impacts of those failures on multiple system levels, it is typically the first step in comprehending system safety [2]. Faults, which can result in errors and failures later on, are where analysis begins. To ensure the safety requirements of ISO 26262, safety mechanisms are designed for every function in the BMS. SPI, a peripheral in a BMS has a safety architecture designed around it, a safety mechanism for every failure mode in the function. Any safety architecture designed for various purposes in automotive industry to meet the safety standards of highest level should be verified using fault injection. By introducing faults into a design and monitoring it to see how it reacts to a fault, the dependability of a design under test that

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 07, July-2022

can be evaluated using the fault injection technique. Digital fault injection runs on RTL/GLS (netlist) by injecting faults at each block's input/output ports and internal nodes during top-level verification. Injected faults are then checked whether they can be detected by safety mechanisms. Diagnostic coverage (DC) is a benchmark of the safety measure to detect dangerous failures which will be calculated for the SPI safety architecture.

II. FUNCTIONAL SAFETY STANDARD

International Electrotechnical Commission (IEC) 61508 is a single standard that addresses functional safety for all products and industries. The International Electrotechnical Commission has published a global standard that outlines techniques for using, designing, deploying, and maintaining safety-related technologies. Functional Safety of Electrical, Electronic, and Programmable Electronic Safety Related Systems (E/E/PE or E/E/PES) is the title of the document. IEC 61508 is the fundamental Functional Safety standard, and it is applicable to all sectors. Despite the fact that this standard applies to all industries, each one has its own subtleties, which is why so many industries have created their own standards based on IEC 61508. The design under validation is a BMS which comes under the Automotive industry.

A. ISO-26262

A worldwide standard for the automotive industry that focuses on safety-critical systems is ISO 26262, which is mainly derived from IEC 61508. It is used for E/E systems, which include both hardware and software components, in automobiles. It outlines the requirements that must be met by the system's safety-relevant function as well as by the procedures, techniques, and tools used during the development process. As the automotive sector becomes more complex, more effort must be made to provide safety-compliant systems. The objective of ISO 26262 is to offer a single safety standard for all E/E systems in automobiles. Functional safety of the product is managed systematically by ISO 26262 at the system, hardware, and software levels during development. It features an automobile safety lifecycle that outlines every stage of production, from management to development to production to operation to service to decommissioning. Automobile Safety Integrity Levels (ASILs), which are used to establish the applicable standards of ISO 26262 in order to reduce unacceptable residual risk, this is further used as a distinctive risk-based approach for identifying risk classes for the automotive industry. Defines specifications for the architecture, design development, verification, integration, validation, and confirmation procedures to guarantee the achievement of an acceptable level of safety.

B. Automotive Safety Integration Levels

The ASIL is a vital component of ISO 26262 compliance. The system's design and development must adhere to the ASIL, which is determined at the start of the chip's development phases. The planned functionalities of the system are examined in light of any potential dangers. The estimation of this risk, based on a combination of the probability of exposure, the possible controllability by a driver, and the possible outcome's severity if a critical event occurs, leads to the ASIL. Regardless of the technologies used in the system, the ASIL is exclusively dependent on the harm to the driver and other road users. Any safety requirement is assigned an ASIL rating of A, B, C, or D. Systems with a "D" are considered to be the most safety-critical and are subject to the strictest testing standards, while processes with a "A" are considered to be the least safety-critical. The minimum testing criteria are outlined in the ISO 26262 standard. This makes picking the testing methods simpler. Based on the ASIL level, the Single Point Fault Metric (SPFM), Latent Fault Metric (LFM), and Probabilistic Metric for Hardware Failure (PMHF) should all be computed and further met, as shown in Table I.

TABLE I. TARGET VALUES FOR HARDWARE ARCHITECTURAL METRICS FOR EACH ASIL

ASIL	SPFM	LFM	PMHF
A	Not Relevant	Not Relevant	<1000FIT
B	≥90%	≥60%	<100FIT
C	≥97%	≥80%	<100FIT
D	≥99%	≥90%	<10FIT

SPFM measures how safety is the system towards Single Point Faults (SPF) and Residual Faults (RF), higher the SPFM better is the safety. For example, if SPI receives corrupted data, if not detected, will always lead to incorrect outputs and surely result in a critical situation. For ASIL D, the SPFM of 99% should be achieved, in simple terms over 99% of Single Point Faults should be detected. SPFM can be calculated as weighted average of diagnostic coverage calculated for SPFs of each block on chip under consideration. The weights are directly proportional to the area occupied by the block on the chip. Latent-fault Metric measures how safety is the system towards Multi Point Faults (MPF) faults, higher the LFM better is the safety. For ASIL D, over 90% of MPF should be detected and mitigated. Probabilistic Metric for random Hardware Failures is the average probability of failure of system per hour throughout the operational lifetime of the system. For ASIL D, PMHF of ≤ 10 FIT should be achieved. Where, the failures-in-time (FIT) rate is determined by the number of random failures that can be expected to happen in one billion (10^9) device-hours of operation [1].

This paper focuses on performing functional safety verification of SPI by estimating the safeness of the design at SoC level, to meet FuSa standards in accordance with ISO26262:2018. At the top-level targets to achieve ASIL D. Here, to achieve this level a DC of SPFs of greater than 97% should be achieved, the value is decided based on the weighted average formula used for the calculation of SPFM, based on this there are industry standard tools available, which calculates LFM and PMHF. To define verification procedure some of the clauses of ISO 26262:2018 adopted:

1. Clause 7.4.3 of ISO 26262-5:2018 - Safety analyses of the hardware design to identify the causes of failures and the effects of faults shall be applied by performing deductive and inductive analysis.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 07, July-2022

2. Clause 7.4.4 of ISO 26262-5:2018 - The hardware design shall be verified by using hardware design verification methods like hardware design walk-through, inspection and simulation in accordance with Clause 7.4.3 of ISO 26262-5:2018 to fulfill the safety related special characteristics to achieve functional safety during production and service [3].
3. Clause 7.4.5 of ISO 26262-5:2018 - Safety related special characteristics shall be specified if safety analysis has shown these characteristics to be relevant.
4. Clause 4.8.1 of ISO 26262-5:20182-11:2018 - Fault injection at the semiconductor component level is a known methodology which can be used to support several activities of the lifecycle when the safety concept involves semiconductor components [4].
5. Clause 4.8.2 of ISO 26262-11:2018 - Characteristics or variables of fault injection that help the verification planning [4].
6. Clause 4.8.3 of ISO 26262-11:2018 - Results of fault injection can be used to verify the safety concept and the underlying assumptions as listed in Clause 4.8.1 of ISO 26262-11:2018 (e.g., the effectiveness of the safety mechanism, the diagnostic coverage and number of safe faults) [4].
7. Clause 5.1.10 of ISO 26262-11:2018 - Verification using fault injection simulation can be used for both permanent and transient faults. Fault injection utilizing design models can be successfully used to assist in verification of safe faults and computation of their amount and failure mode coverage. Injecting faults and utilizing well-specified observation points to determine if the fault caused a measurable effect. Moreover, it can be used to assist the computation and to verify the values of failure mode coverage. injecting faults that were able to cause a measurable effect and determining if those faults were detected or controlled by the safety mechanisms within the maximum fault handling time interval [4].

C. Fault Injection Terminologies:

Fault injection is an integral step and valuable technique for functional safety verification. This paper, which is an automotive application, with a particular focus on the simulation-based fault injection platform. Simulation Based Fault Injection is an automated fault injected simulation is used to mimic" in the field faults" to verify the safety architecture. With the system Verilog, the random fault injection simulation platforms can implement fault models such as: Stuck at 0, Stuck-at 1 and Bit flip, and insert them randomly into the design to analyze fault coverage. By performing fault injection, a wrong output signal is/ are produced which is an error, which will further lead to a failure. The simulation-based fault injection environment consists of a Fault Injector, Fault Library, Controller and data analyzer.

In simulation-based fault injection faults can be modelled as permanent and transient faults. A permanent fault is a persistent failure, such as a short circuit between wires, pins, or tracks which continues to exist until the faulty component is repaired or replaced. Real world examples of permanent faults include disk head crashes, software bugs, and burnt-out power supplies. Permanent fault that continues to exist within a system until that error is fixed or repaired. In simulation-based fault injection further, permanent faults can be modelled in two different ways:

1. Stuck at 0 or SA0: Forcing a signal value to be 0 from the start of fault injection to the end of simulation.
2. Stuck at 1 or SA1: Forcing a signal value to be 1 from the start of fault injection to the end of simulation.

A transient fault is a fault that is no longer present if power is disconnected for a short time and then restored or which are seen for short time. In simulation-based fault injection further, transient faults can be modelled in two different ways:

1. Single Event Upset or SEU: this model inverts the value of the output of a sequential element and holds the modified value it is assigned a new value. This fault model can be applied on the outputs of sequential elements such as memories, flipflops and latches.
2. Single Event Transient or SET: this model inverts the value of a signal and holds the modified value for a specified period of time. This fault model can be applied to any kind of signal, such as nets or registers.

In fault injection campaign multiple fault injection runs are executed to generate safety metrics. Every fault injection simulation run performs good simulation and fault simulation. Good simulation is the initial simulation that you must run to generate reference values for fault injection and fault simulation is the subsequent simulation that classifies any faults injected into the design. In this process to capture reference values for observation points (that are defined before the good simulation) during the good simulation run. These observation points help to better classify injected faults at one or more nets/nodes.

1. Functional Strobes: All the primary outputs in the design under consideration will be used to detect whether a fault injected on a node/port causes functional failure.
2. Checker strobes: These are the outputs which will be used to detect whether a fault injected on a node/port is detected by safety mechanism. All the outputs of the safety mechanisms are checker strobes.

Functional and checker strobes are required to categorize the injected faults. Fault propagating to Functional strobes are dangerous and Fault propagating to Checker strobes are detected. The injected faults can be classified into following ways:

1. Dangerous and Detected (DD): Any injected fault propagates to both functional strobe and checker strobe is considered to be DD. If a fault is classified as DD, then, safety architecture is able to detect dangerous faults.
2. Dangerous and Undetected (DU): Any injected fault propagates to only functional strobe but not checker strobe is considered to be DU. If a fault is classified as DU, then, safety architecture is not able to detect dangerous faults.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 07, July-2022

3. Unobserved and Detected (DU): Any injected fault propagates to only checker strobe but not functional strobe is considered to be UD. If a fault is classified as UD, then, safety architecture is more cautious.
4. Unobserved and Undetected (UU): Any injected fault does not propagate to any functional strobe and checker strobe is considered to be UU. If a fault is classified as UU, then, this is due to lack of stimuli, on providing proper stimuli may end up as DDs/DUs/UDs.

Diagnostic coverage is a benchmark of the safety measure to detect dangerous failures. It can be expressed as:

III. SAFETY ARCHITECTURE AND ITS FSV

A. *SPI Safety Architecture.*

A single-master communication protocol is known as SPI. This implies that just one device begins communications with other slave devices. It is a high-speed synchronous serial IO port that shifts the length of a serial bit stream (data) and transfers or receives it at a programmed bit-transfer rate. The serial clock is activated at a clock frequency that may be controlled by both the master and slave when the SPI master wants to send data to a slave. The slave makes the selection by pushing the corresponding slave selection line low. SPI can support duplex communication between the master and its peripheral devices because the master generates data onto the MOSI line while sampling the MISO line. It is crucial to remember that for a communication to work, a master and slave pair must employ the same set of parameters, such as the SCLK frequency, CPOL, and CPHA. Status, control, and data registers, shifter logic, a baud rate generator, master/slave control logic, and port control logic make up the majority of the SPI design. The SPI designed is extremely similar to the one detailed in Motorola, Inc.'s SPI block guide.

In the safety concept, functional safety requirements are to be defined for the SPI design at system level. Functional Safety Requirement of SPI is to have data loss protection through SPI path (both transmit and receive) which needs to be functionally safe and protected from malfunctions. The data flow through BMS can happen either from the external IC into BMS SPI or from BMS internal storage to external IC through BMS SPI. Any fault in SPI path that could lead to functional failures that could violate the functional safety requirement are mentioned in FMEDA. Any fault in SPI, its configuration / mode selection, I/O pins may lead to data integrity, authenticity, timeliness (e.g., data transfer initiation and completion) and configuration errors. Also, any fault in interrupt generation, recognition and servicing may show in an inability to recognize events and modify the signal flow which results in failures in data acquisition, transmission and processing. So, safety mechanisms should be designed in order to protect functional safety requirement that is defined.

B. *SPI FMEDA*

Firstly, FMEDA is performed by asking the questions" What are we trying to avoid??" and" How can it occur??" and also adds a "Diagnostic" section by asking the question beyond "what can go wrong??". FMEDA includes analysis of diagnostic coverage to identify failure mode that has potential to violate safety goal in absence of safety mechanism and then identify the safety mechanism that prevent the failure mode from violating the safety goal. After the netlist is available, FMEDA at block level is done which helps in implementations of these mechanisms in order to detect all the failure modes at bottom level. The objective is to calculate the failure mode coverage wrt safety goal violation. Functional analysis forms the basis for FMEDA.

TABLE II. SPI FMEDA

Failure Modes	Malfunction	Safety mechanisms
Transmit shift register, which holds the latest data to be transmitted to external device.	Data corruption	CRC Check on data.
Receive shift register, which holds the latest of data received from external device.	Data corruption	CRC Check on data.
Clock selection logic, select the clock polarity and phase based on CPOL/CPHA configuration or master/slave modes.	Wrong decode of clock selection	Register toggle verification
Data transfer type logic, which selects the data to be transmitted with LSB/MSB bit first.	Wrong data transmission.	CRC Check on data and Register toggle verification
Interrupt generation logic, which enables SPI interrupt requests upon corresponding flag bits set in status register.	Wrong interrupt being serviced or interrupt may not be serviced.	Interrupt count and Register toggle verification
Protocol generation logic, generates SPI protocol signals: SCLK, SS.	Wrong protocol signals generated	Timeliness check and CRC check in external device.
SPI state machine, provides control signals to shift registers and transfers & Wrong control signals generated and state machine is corrupted	Wrong control signals generated and state machine is corrupted	Timeliness check and CRC check on data.
Input/Output enable controls, provides IO control to SPI ports.	Incorrect IO control - MISO, MOSI, SCLK and SS.	Timeliness check.

IV. IMPLEMENTATION

A. *Safety Mechanisms*

Safety mechanisms are on-chip features that detect and mitigate or make the design tolerate faults and report them when they are detected These can be pure hardware, pure software or hybrid. The safety mechanisms proposed during the design phase to protect SPI from failures are purely software safety mechanisms. After performing FMEDA at block level these mechanisms are implemented in order to detect all the failure modes at bottom level. These have to be implemented in parallel when SPI operations are happening. The proposed safety mechanisms are:

1. CRC check on the data: CRC provides error detection on serial interfaces as data moves across

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 07, July-2022

chips. Detects faults that lead to any mismatch in the expected data.

2. Interrupt Count and timeliness check: Detects any missing or spurious interrupts resulting random failures in interrupt generation.
3. Register toggle verification: Detects faults leading to incorrect configuration that cause incorrect data transfers

B. Simulation based fault Injection

To support functional safety verification, Cadence vManager Safety Client is used to automate fault injection campaign using well-defined flows. By using input files, one can drive both the safety client and the internal core engine running within the client, the Xcelium Fault Simulator. The input files are, configuration file that defines the overall parameters, the fault list that identifies targets for fault instrumentation, the fault list that identifies targets for fault instrumentation, the test list that specifies one or more tests to run in a single session and other script files required for running a campaign.

The vManager Safety Client supports various campaign flow types, the ones used are serial and concurrent. In Serial: flows the injection of one fault during a fault simulation run (i.e., the next fault can be injected after the run is closed and reopened). Number of fault simulations is equal to the number of faults to be injected, which is a disadvantage in this flow and consumes more CPU memory because of large number of simulations. In Concurrent flow, injection of multiple faults per run for the fault simulation session. Advantage of this flow is it is a throughput solution where multiple faults are injected and simulated together and also simulate good and fault simulation simultaneously. Tool keeps a list of faulty values along with good value at every node in the circuit. Disadvantage of this flow is, the hyperactive faults, which cause very large number of simulation events will not propagate to their respective observation point(s). Such faults are flagged in the fault database for concurrent simulations as Not Simulatable (NS). Also, if faulty simulation doesn't converge within the time good simulation value is calculated then the faults are reported as DU.

For the SPI block all the functional outputs are defined as functional strobes – MISO, MOSI, SCLK, SS and corresponding input & output enables. All the safety mechanism outputs are defined as checker strobes – CRC check, timeliness check, register toggle configuration check and interrupt count and timing check. All the test cases implemented for various SPI operating modes with safety mechanisms. Fault instrumentation of the SA0 and SA1 fault types for all cell ports in SPI netlist and these are injected at 80ns after simulation started and timeout factor of 10 is given, this means fault simulation should be executed maximum up to 10 times the good simulation time.

V. RESULTS AND DISCUSSIONS

A. Concurrent Run

The concurrent simulation is done considering group size of 2000, this indicates for multiple faults injected and simulated are restricted to 2000 per group. A total of 9702 faults can be instrumented into the design out of which 119 were reported safe after structural analysis, 9583 as testable out of which 6264 are prime faults. After fault simulation is completed a session report is generated, shown in figure I, DC of 83.84 % is achieved.

```
--- Fault Classification Summary (Dual Strobe) Total                Prime
-------------------------------------------------------------------------------
nr faults SAFE                  [S] :        119 [  1.2%]         115 [  1.8%]
nr faults DANGEROUS_UNDETECTED  [DU] :      1041 [ 10.7%]         851 [ 13.6%]
nr faults DANGEROUS DETECTED    [DD] :      5400 [ 55.6%]        3299 [ 52.7%]
nr faults NOT CLASSIFIED        [NC] :      3142 [ 32.3%]        2002 [ 31.9%]
-------------------------------------------------------------------------------
--- Fault Classification Detail (Dual Strobe)            Total               Prime
-------------------------------------------------------------------------------
SAFE                              [S] :        119 [  1.2%]         115 [  1.8%]
  UNTESTABLE                     [UT] :        119 [  1.2%]         115 [  1.8%]
  UNOBSERVABLE UNDETECTED       [U+U] :          0 [  0.0%]           0 [  0.0%]
  UNOBSERVABLE DETECTED         [U+D] :          0 [  0.0%]           0 [  0.0%]
DANGEROUS UNDETECTED             [DU] :       1041 [ 10.7%]         851 [ 13.6%]
DANGEROUS DETECTED               [DD] :       5400 [ 55.6%]        3299 [ 52.7%]
NOT CLASSIFIED                   [NC] :       3142 [ 32.3%]        2002 [ 31.9%]
  UNOBSERVED UNDETECTED          [UU] :       3142 [ 32.3%]        2002 [ 31.9%]
  UNOBSERVED DETECTED            [UD] :          0 [  0.0%]           0 [  0.0%]
  UNPROCESSED UNPROCESSED     [NP,NS] :          0 [  0.0%]           0 [  0.0%]
NS := NOT SIMULATABLE
```

FIGURE I CONCURRENT SIMULATION LOG

B. Serial Run

Though concurrent flow is a throughput solution it has many disadvantages as mentioned earlier in this paper. So, the fault nodes which are reported as DU's in the previous run are extracted and given as input to serial run in fault list file. After fault simulation is completed a session report is generated, shown in figure II, DC of 93.57 % is achieved after merging the concurrent and serial run results.

```
--- Fault Classification Summary (Dual Strobe) Total                Prime
-------------------------------------------------------------------------------
nr faults SAFE                  [S] :          0 [  0.0%]           0 [  0.0%]
nr faults DANGEROUS_UNDETECTED  [DU] :       414 [ 39.8%]         387 [ 45.8%]
nr faults DANGEROUS DETECTED    [DD] :       627 [ 60.2%]         458 [ 54.2%]
nr faults NOT CLASSIFIED        [NC] :          0 [  0.0%]           0 [  0.0%]
-------------------------------------------------------------------------------
--- Fault Classification Detail (Dual Strobe)            Total               Prime
-------------------------------------------------------------------------------
SAFE                              [S] :          0 [  0.0%]           0 [  0.0%]
  UNTESTABLE                     [UT] :          0 [  0.0%]           0 [  0.0%]
  UNOBSERVABLE UNDETECTED       [U+U] :          0 [  0.0%]           0 [  0.0%]
  UNOBSERVABLE DETECTED         [U+D] :          0 [  0.0%]           0 [  0.0%]
DANGEROUS UNDETECTED             [DU] :       414 [ 39.8%]         387 [ 45.8%]
DANGEROUS DETECTED               [DD] :       627 [ 60.2%]         458 [ 54.2%]
NOT CLASSIFIED                   [NC] :          0 [  0.0%]           0 [  0.0%]
  UNOBSERVED UNDETECTED          [UU] :          0 [  0.0%]           0 [  0.0%]
  UNOBSERVED DETECTED            [UD] :          0 [  0.0%]           0 [  0.0%]
  UNPROCESSED UNPROCESSED        [NP] :          0 [  0.0%]           0 [  0.0%]
-------------------------------------------------------------------------------
```

C. DU Analysis

The obtained DC is not sufficient and has to be improved, this can be done by reducing number of faults that are reported as DU. To do this a detailed DU analysis for every node that is reported has to carried out and which is possible only with the good understanding of the architectural design, and find a way to detect them or classify them as safe, even though they disturb the functional strobes but have no safety violation. vManager safety client also supports debugging ways through which the reported DU's can be rerun with waves, this helps in comparing good simulation waves with fault simulation waves, when is the injected fault effecting different signals and logic blocks of the SPI.

To achieve the targeted ASIL D, diagnostic coverage of 97.2 % is achieved for the single point faults in the SPI block by performing detailed DU analysis and moving them to DD category by identifying the detection mechanism or to safe category by analyzing the effect of the fault, which makes it sufficient to achieve SPFM of 99% for the entire chip.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 11 Issue 07, July-2022

VI. CONCLUSIONS

Based on the FMEDA, safety mechanisms are implemented and simulation based digital fault injection is performed for safety mechanisms (diagnostics) verification using vManager safety client. Here, permanent faults at every cell port of the netlist are injected, set of functional and checker strobes are defined which classify faults as DD, DU, UD, UU and Safe. Both serial and concurrent flows supported by the tool were used to achieve a diagnostic coverage of 93.5% for the total 9702 instrumented faults, with 6047 reported as DD and 414 reported as DU. To achieve the targeted ASIL D, diagnostic coverage of 97.2 % is achieved for the single point faults in the SPI block by performing detailed DU analysis and moving them to DD category by identifying the detection mechanism or to safe category by analyzing the effect of the fault, which makes it sufficient to achieve SPFM of 99% for the entire chip.

VII. FUTURE SCOPE

Based on the challenges faced to complete the functional safety verification faster, there are some ways possible to achieve the same results in the less time frame and recommendations for other designs [5][6].

1. Running a concurrent run and then extracting the fault list to run serial run involves more time for setup creation and execution. So, a hybrid flow can be created by asking the tool vendor to run concurrent and serial campaigns simultaneously with the single setup and internally extract the fault list for serial run as soon as concurrent run reports a fault as NS and DU. Then generated a merged data report which saves a lot of execution time.
2. For more complex designs, having > 40,000 faults, performing fault injection on all possible faults is not a good way, rather sampling method should be used where tool randomly injects faults in the design for the specified value. Then analyze the results for the fewer faults and modify the test cases to achieve desired DC for this set of faults. Perform this for 2 to 3 iterations, usually the DC value will converge.
3. To perform the DU analysis faster, formal methods can be used to know the stimuli for which the fault injected can be detected or classified as safe.

REFERENCES

[1] S. Chonnad, R. Iacob, and V. Litovtchenko, “A quantitative approach to soc functional safety analysis,” in 2018 31st IEEE International System-on-Chip Conference (SOCC), IEEE, 2018, pp. 197–202.

[2] K.-L. Lu, Y.-Y. Chen, and L.-R. Huang, “Fmeda-based fault injection and data analysis in compliance with iso-26262,” in 2018 48th Annual IEEE/IFIP international conference on dependable systems and networks workshops (DSN-W), IEEE, 2018, pp. 275–278.

[3] ISO, “26262 road vehicles-function safety-part 5: Product development at the hardware level,” International Standardization Organization Std, 2018.

[4] ISO, “26262 road vehicles-function safety-part 11: Guidelines on applying the standard to semiconductors,” International Standardization Organization Std, 2018.

[5] F. A. Da Silva, A. C. Bagbaba, S. Hamdioui, and C. Sauer, “Efficient methodology for iso26262 functional safety verification,” in 2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS), IEEE, 2019, pp. 255–256.

[6] F. A. da Silva, A. C. Bagbaba, S. Hamdioui, and C. Sauer, “Combining fault analysis technologies for iso26262 functional safety verification,” in 2019 IEEE 28th Asian Test Symposium (ATS), IEEE, 2019, pp. 129–1295.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)