Safety Analysis of an Airbag System using Probabilistic FMEA and Probabilistic Counter Examples

Husain Aljazzar[1], Manuel Fischer[2], Lars Grunske[3], Matthias Kuntz[1], Florian Leitner[1], Stefan Leue[1]
[1] University of Konstanz, Germany
[2] TRW Automotive GmbH, Germany
[3] Swinburne University, Australia

Abstract

Failure mode and effects analysis (FMEA) is a technique to reason about possible system hazards that result from system or system component failures. Traditionally, FMEA does not take the probabilities with which these failures may occur into account. Recently, this shortcoming was addressed by integrating stochastic model checking techniques into the FMEA process. A further improvement is the integration of techniques for the generation of counter examples for stochastic models, which we propose in this paper. Counter examples facilitate the redesign of a potentially unsafe system by providing information which components contribute most to the failure of the entire system. The usefulness of this novel approach to the FMEA process is illustrated by applying it to the case study of an airbag system provided by our industrial partner, the TRW Automotive GmbH.

1 Introduction

In light of the fact that a failure of a safety-critical system can lead to injuries and even loss of life it is extremely important to provide designers with safety assessment methods that help to minimise the risk of the occurrence of such disastrous events. One of these methods is failure mode and effects analysis (FMEA) [20]. In FMEA, a team of trained engineers or system designers analyses the cause consequence relationships of component failures on system hazards. After having found such a relation, the occurrence probability of that hazard is computed. It is then checked whether this value is above a certain threshold, defined by the tolerable hazard probability or rate (THP or THR). If this is the case measures must be taken to reduce the probability of the undesired event.

To support the traditionally time-intensive and error-prone FMEA process, functional model checking techniques have been integrated into the process [6, 8, 9, 19, 16]. While these techniques are able to establish cause-consequence relationships, they are unable to calculate the actual failure probabilities. Therefore, stochastic model checking was applied to FMEA, leading to a probabilistic FMEA (pFMEA) process [14]. Currently, this pFMEA process provides no means to help the designer in reducing the risk of failures. It only supports the first step of the FMEA process, which is to identify cause-consequence relationships and compute the actual hazard probabilities.

The contributions of our paper can be described as follows.

- We illustrate the usefulness of pFMEA as supported by stochastic model checking using the real-life case study of an airbag system. We describe how to map the system architecture to a PRISM [23] model and illustrate how to perform pFMEA on this model. The airbag case study results from a collaboration with the automotive supplier TRW Automotive GmbH in Radolfzell, Germany, and is based on real data. Due to intellectual property concerns of our industrial partner, we are unable to publish the concrete values of component or overall system failure probabilities[1]. This does not affect our finding that pFMEA can lead to useful failure probability assessment values, as confirmed by our industrial partner.
- We address the inability of the current pFMEA method to give guidance in how to improve system dependability by integrating a recently developed technique for finding counter examples in stochastic models. Counter examples provide means to identify those parts that contribute most probably to the failure of the system and thus, provide valuable information for its redesign.

[1] Any concrete probability or rate values that this paper presents are either taken from publicly available specifications, such as the ISO 26262 standard [22], or plausible estimates confirmed by TRW Automotive GmbH. The PRISM model used in the analysis does contain concrete probability values for the components used.

This paper is organised as follows: In Sec. 2 we will briefly introduce FMEA, pFMEA and counter examples in stochastic model checking. Sec. 3 is devoted to the description of the airbag system and its PRISM model. In Sec. 4 we will describe possible hazard conditions and system failures, and Sec. 5 is devoted to the probabilistic FMEA of the airbag system, supported by counter example generation. In Sec. 6 we compare our approach with existing approaches in the FMEA literature. Sec. 7 describes the lessons learnt from this case study. Finally, Sec. 8 concludes the paper with a summary and an outlook on future research.

2 FMEA, pFMEA and Counter Examples

This section explains the basic concepts of the failure model and effect analysis (FMEA) and its probabilistic extension [14]. Furthermore, in Sec. 2.3 we briefly introduce an approach to counter example generation for stochastic model checking. A more detailed account of these topics can be found in [1, 3, 18].

2.1 FMEA

As described in the introduction, the aim of an FMEA is to explore the consequences, such as hazards, of known component-level failure modes and to propose countermeasures to mitigate and reduce the probability that these consequences occur. The final outcome of an FMEA is a table which documents for each component the set of relevant component failure modes and for each of these failure modes its consequences. Possible failure detection, correction or mitigation mechanisms may also be recommended in this table. The structure, number of columns and meaning of columns of the resulting FMEA table may vary in different organizations performing FMEA. However, the following column headings are commonly used [20]: investigated component, failure mode, description of the failure mode/local effect of the failure mode, possible cause for the failure, effect at the system level, recommended failure detection mechanism, recommended mitigation mechanism, and recommended design changes. For complex systems with a large number of components and a large number of failure modes per component this table can become very large. Additionally, it has been reported in [17] that the table may contain redundant information since different failure modes can lead to the same consequences. The FMEA procedure is commonly defined by an iterative process [15] that identifies for all components the possible failure modes and identifies their consequences. Recent approaches [25, 19, 8, 9, 6, 7, 16] aim to support the FMEA process, especially the identification of possible consequences with model checking. The basic idea is to formalise the system's behaviour as a state-based model and the hazard conditions as temporal logical formulae. As a result, fault injection experiments can be created where specific failure modes are injected into the system behaviour model. Model checking tools can then analyse the consequences on the formalised hazard conditions.

2.2 pFMEA

A further development of the idea to use of model checking support for FMEA is the approach referred to as probabilistic FMEA (pFMEA) presented in [14]. Instead of functional state-based models pFMEA uses stochastic models, in particular discrete and continuous time Markov chains. The hazard conditions are formulated as stochastic temporal logical formulae. As a consequence the tolerable hazard probabilities can also be integrated into the fomalisation of the hazard conditions. Furthermore, to each injected failure mode an occurrence probability can be assigned in the probabilistic model. A main benefit of pFMEA is the ability to probabilistically estimate the likelihood that an injected failure mode will lead to a violation of the hazard condition. The use of stochastic models also avoids a common shortcoming of using functional model checking in FMEA, namely that the model checker finds a relationship between the injected failure mode and a hazard that is very unlikely to occur in practice. As already noted in [14], one practical problem of pFMEA is the lack of counter examples in stochastic model checking. This impedes the explanation of property violations and hence failure mode / consequence relationships found by the stochastic model checker.

2.3 Counter Examples in Stochastic Model Checking

In stochastic model checking, the property that is to be verified is specified using a variant of temporal logic. The temporal logic used in this paper is CSL (continuous stochastic logic) [4, 5]. CSL is a stochastic variant of CTL [10]. It is tailored to reason about quantitative system behaviour, including the performance and dependability of a system. Just like in traditional model checking, given an appropriate system model and a CSL property, a stochastic model checking tool such as PRISM can verify automatically whether the model satisfies the property. If the model refutes the property, a counter example usually helps engineers to comprehend the reasons of the property violation and to devise arrangements to fix the error. The computation of counterexamples in stochastic model checking has recently been addressed in [1, 2, 3, 18].

2.3.1. Notion of Counter Examples. For our purposes it suffices to consider upper bounded properties which require the probability of a property offending behaviour not to exceed a certain upper probability bound. In CSL such properties can be expressed by formulas of the form $\mathscr{P}_{\leq p}(\varphi)$, where φ is path formula specifying undesired behaviour of the the system. Any path which starts at the initial state of the system and which satisfies φ is called a *diagnostic path*. A counter example for an upper bounded property is a set X of diagnostic paths such that the accumulated probability of X violates the probability constraint $\leq p$.

2.3.2. Generation of Counter Examples. In [1] it has been shown that counter examples for this class of properties can be efficiently computed using an explicit state space search strategy called *eXtended Best-First* (XBF). XBF is an extension of the well-known Best-First search strategy (BF) [24]. XBF explores the state space of the model on-the-fly searching for diagnostic paths. It does not explicitly compute the set X of diagnostic paths forming the counter example. Instead it computes a sub-graph of the state space of the model which covers this set, called *diagnostic sub-graph*. The diagnostic sub-graph is selected incrementally. Once the selected diagnostic sub-graph covers enough diagnostic paths so that the accumulated probability exceeds the given upper probability bound, XBF terminates and produces the diagnostic sub-graph as a counterexample.

2.3.3. Counter Example Visualisation. A counter example computed in this way is a potentially very large set of diagnostic paths. Although XBF provides the counterexample in the form of a sub-graph, it can still be very complex. An approach for supporting the analysis of complex counter examples using visualisation techniques is proposed in [2]. The counter example visualisation aims at facilitating the the discovery of causal factors for property violations and hence understanding the cause behind property violations. Portions of the model that contribute a larger portion of the probability mass to the property violation are brought out visually in order to support the discovery of causal dependencies. The visualisation presented in [2] is designed for counterexample generation methods based on K-Shortest-Paths search algorithms like K* [3] or Eppstein's algorithm [12]. For the purpose of this paper we adopted that visualisation to be used in combination with XBF, which has been proven to be significantly more scalable than the other algorithms mentioned above.

3 Case Study: Functionality and Modelling

3.1 System Functionality

Modern cars are equipped with safety systems that protect the occupants of the vehicle. Airbags are one example of an occupant protection system. In case of a crash, the airbag system will deploy airbags that reduce the risk of serious or even fatal injuries for the occupants. Current airbag systems consist of not only the front airbags but also of side, head, knee and a number of further airbags to protect both the driver and the passengers.

An airbag system can be divided into three major parts: sensors, crash evaluation and actuators. An impact is detected by acceleration sensors (front/rear/side impact) and additional pressure sensors (side impact). Angular rate or roll rate sensors are used to detect rollover accidents. The sensor information is evaluated by two redundant microcontrollers (μC) which decide whether the sensed acceleration corresponds to a crash situation or not. The deployment of the airbags is only activated if both microcontrollers decide that there was indeed a critical crash. The redundancy of the microcontroller system layout decrease the hazard of an unintended airbag deployment, which is considered to be the most hazardous malfunction of the system[2]. Upon activation of the deployment, the airbags are inflated with irreversible pyrotechnical actuators. The sensors can be either located as internal sensors inside the Airbag Electrical Control Unit, or mounted as satellites to the bumper, the a-, the b- or the c-pillar.

Our case study focuses on two variants of the airbag system. It consists of two acceleration sensors whose task it is to detect front or rear crashes, either one microcontroller or two microcontrollers to perform the crash evaluation, and an actuator that controls the deployment of the airbag. Fig. 1 gives a schematic overview of the system architecture using the two microcontroller variant. Notice that the redundant acceleration sensors are mounted into different directions so that one is measuring the acceleration in the x direction (also referred to as *main sensor*) of the vehicle and the other one is measuring the acceleration in the $-x$ (also called later on `safing sensor`) direction. The microcontrollers read the sensor values of the main and safing sensor (microcontroller 1) or the safing sensor (microcontroller 2) in a cyclic fashion. The two sensor values (x and -x acceleration) are compared after they have been read by microcontroller 1. They are then separately used for crash discrimination which is normally done by calculating mean

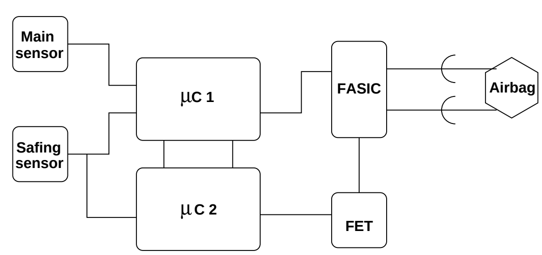

Figure 1. Schematic system architecture

[2]Older airbag systems comprise only one microcontroller.

values of the acceleration measured over certain intervals of time. If a certain number of thresholds in a given time frame are exceeded, the microcontrollers will synchronise their fire decisions and only if they both come to the conclusion that a critical crash occurred the airbags will be deployed.

The deployment of the airbag is also secured by two redundant protection mechanisms. The Field Effect Transistor (FET) controls the power supply for the airbag squibs that ignite the airbag. If the Field Effect Transistor is not armed, which means that the FET-Pin is not high, the airbag squib does not have enough electrical power to ignite the airbag. The second protections mechanism is the Firing Application Specific Integrated Circuit (FASIC) which controls the airbag squib. Only if it receives first an arm command and then a fire command from the microcontroller 1 it will ignite the airbag squib which leads to the pyrotechnical detonation inflating the airbag.

In case there is only one microcontroller, the signals from both the main and the safing sensor are evaluated by this microcontroller, also the signals to both the FET and FASIC untis are only sent by this microcontroller.

Although airbags save lives in crash situations, they may cause fatal behaviour if they are inadvertently deployed. This is because the driver may lose control of the car when this deployment occurs. It is therefore a pivotal safety requirement that an airbag is never deployed if there is no crash situation.

The international standards [21] and [22] regulate how safety critical components in cars shall be developed and how their safety shall be ensured.

3.2 System Model

The airbag system was modelled using the input language of the stochastic model checking tool PRISM [23]. The overall structure of the model corresponds directly to the system's architecture (cf. Fig 1). The behaviour of each block and each bus or connection line, which may also be subject to failures, was modelled by a separate module in PRISM.

While modelling the airbag system, the following challenges had to be met:

1. Many failures stem from corrupted signals, which are of continuous nature. Continuous signals cannot be modelled in the PRISM language and we have to resort to abstractions by discrete approximations. The sensors convert the physical signals using an A/D converter to discrete signals whose values range from -512 to 511. Notice that for the system analysis it is irrelevant whether the original signal is corrupted or whether the corruption is due to an A/D converter failure. The obtained abstraction is, however, still too fine since the induced state space is beyond what could be handled by the PRISM model checker. We therefore abstract from the concrete values of the digital signals and only consider four categories of sensor values: a) normal driving, b) rear crash, c) frontal crash, and d) borderline cases (cf. Sec. 4). Due to space restrictions, we can only deal with cases a) and b) in this paper.

2. For the microcontrollers and the sensors we can safely abstract from internal behaviour, for instance from the failure of subcomponents, since these subcomponents are not manufactured by TRW Automotive. As a consequence these failure modes fall outside of the responsibility of the organisation for which the FMEA is to be carried out. Instead, the total failure rate of the component that was determined by the supplier of these components is used. If these components turn out not to satisfy the reliability requirements, they need to be replaced by other components.

3. The probability distributions for all failure rates can be safely assumed to be exponential. Either this assumption holds due to the data provided by the manufacturers of the components, or the distribution follows a bathtub curve [26] of which only the portion where the failure rate is constant is relevant. According to TRW, we can assume that the phases where the bathtub curve is not constant are either observed early in production and filtered out during the end of line testing, or they occur very late in the lifetime of the system where it can safely be assumed that the car is inoperative at that time.

Using these abstractions, we end up with a fairly accurate basis model (no failures, critical crash) that possesses $\sim$ 55.000 states.

In Fig. 2 we find the principle state-machine model of the μC. ReadSensors is the system's initial state, sensor values are read asynchronously from the sensors. The values are stored in registers and evaluated. If consecutively for n times, the sensor values indicate that a critical crash occured, then the FET and the FASIC are armed (actions arm_FET! resp. arm_FASIC!. This reflects the fact that a critical crash has to be discriminated from borderline cases. In the latter case a high acceleration is sensed only for a very short time, which must not cause an airbag ignition. Such situations may occur when driving over a curbstone or driving fast in a curve or roundabout.

In Fig. 3 resp. Fig. 4, we find the basic state-machine model of the FET and FASIC modules (with initial states WaitFET resp. WaitFASIC). Here, the FET and FASIC synchronise with the μC, that sends command to the FET (FASIC), action arm_FET? (arm_FASIC?). For the FASIC to finally fire, it synchronises with the FET via the action fire_FASIC?, which is sent by the FET unit (fire_FASIC!)

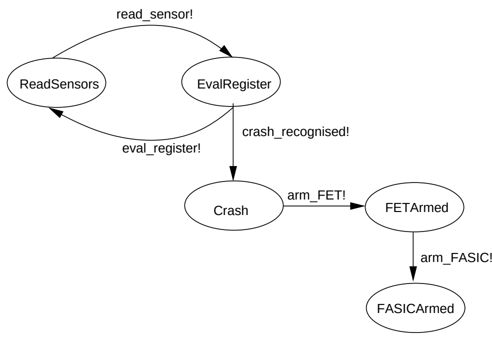

Figure 2. Basic μC model

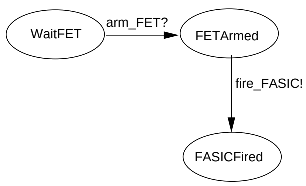

Figure 3. Basic FET model

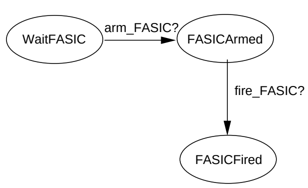

Figure 4. Basic FASIC model

4 System Failures and Hazard Conditions

In this section we describe possible failures of the system components and their respective consequences for the safe functionality of the system. The hazards, we consider in this paper are either the suppression of airbag ignition when required or the unintended deployment of the airbag, in case no crash occurred.

4.1 System Hazards and Safety Requirements

An upcoming standard ISO 26262 [22], which is an adaption of ISO 61508 [21] for road vehicles, is under development. According to ISO 26262, new airbag systems have to comply with ASIL D (Automotive Safety Integrity Level D) for unintended deployment of the airbag. ASIL D corresponds to a tolerable hazard rate (THR) of 10^{-8} per hour. Currently, airbag systems only must correspond to ASIL B which specifies a THR of 10^{-7} per hour. For our case study, we found the following hazards condition to be relevant (specified as the Probabilistic Existence pattern [13]):

1. The airbag is not ignited, although a critical crash actually occurred. This hazard can be formalised as safety requirement 1 in CSL as follows:

$$\mathscr{P}_{\leq p_1}(\mathsf{true}\ \ U^{>T_1}\ (\mathsf{critical\_crash} \wedge \neg \mathsf{fasic\_fired}))$$

   For the purpose of the analysis we let the probability bound $p_1 = 10^{-3}$ and the actual time bound $T_1 = 20\ ms$ $\mathsf{critical\_crash}$ and $\mathsf{fasic\_fired}$ are atomic properties that can be derived from the original PRISM model. $\mathsf{critical\_crash}$ is the state of the μC, when after reading and evaluating the sensor values, it is decided that the crash event requires airbag ignition. $\mathsf{fasic\_fired}$ is the state of the FASIC module that indicates, that the FASIC finally sent the fire command to the airbag squibs.

2. The airbag is ignited at latest after $T_2 = 45\ ms$, which yields safety requirement 2:

$$\mathscr{P}_{\leq p_2}(\mathsf{true}\ \ U^{>T_2}\ (\mathsf{critical\_crash} \wedge \mathsf{fasic\_fired}))$$

   With this hazard condition, we associate a tolerable violation probability p_2 of 10^{-4}.

3. The airbag is deployed unintentionally which means that it is ignited even though no crash at all or only a non-critical crash has occurred. This leads to safety requirement 3 in CSL:

$$\mathscr{P}_{\leq thp_3(T_3)}(\mathsf{true}\ \ U^{\leq T_3}\ (\neg\mathsf{critical\_crash} \wedge \mathsf{fasic\_fired}))$$

This hazard is associated with a tolerable hazard probability (THP) $thp_3(T_3)$ which depends on the mission time T_3, and the THR associated with the desired ASIL D:

- Given the following mission times $T_3 = 1\ hr$, $5\ hrs$, and $10\ hrs$ and ASIL B, we obtain: $thp_3(1\ hr) = 1.0 \cdot 10^{-7}$, $thp_3(5\ hrs) = 5.0 \cdot 10^{-7}$, and $thp_3(10\ hrs) = 1.0 \cdot 10^{-6}$. The actual THP can be computed according to the formula $THP(t) = 1 - e^{-THR \cdot t}$, where t is the mission time (here: driving time).
- Similarily, for ASIL D, we obtain: $thp_3(1\ hr) = 1.0 \cdot 10^{-8}$, $thp_3(5\ hrs) = 5.0 \cdot 10^{-8}$, and $thp_3(10\ h) = 1.0 \cdot 10^{-7}$.

4.2 Sensor Failures

For the sensors, we have identified the following failure modes:

1. Even though both sensors measure the same signal, the amplitude of this signal at both sensors is different.
2. The sensors deliver wrong amplitudes. This means that the real signal's amplitude is corrupted by sensor failures.
3. The sensors function correctly, but since the sensor values are not sampled synchronously the delay between the two samples may be so large that the amplitudes are erroneously interpreted as being different.
4. Both sensors are accelerated in the same direction. This means that the amplitudes on both sensors have the same prefixes.

4.3 Microcontroller Failures

The microcontroller is composed of different subcomponents that can fail independently of each other. In our case, these subcomponent failures are not considered separately, we are only interested in the failure of the microcontroller itself. Failures of subcomponents are outside the responsibility of TRW Automotive. Thus, the microcontroller can be considered as a blackbox.

The potential consequences of a μC failure are:

- A fire command is needlessly sent to the FET and FASIC, thus causing an unintended deployment of the airbag.
- A fire command in case of a critical crash is suppressed, thus preventing the airbag from being ignited.
- The fire command for the airbag in case of a crash is delayed, thus causing the airbag to be ignited too late.

According to TRW, the first case is considered to be the worst case.

4.4 Power Supply Failures

The power supply unit has two lines: a 5V-line connected to the microcontroller and the sensors and a 24V-line to the FET- and FASIC-units. Both lines are subject to failures:

1. 5V-line: If the voltage of this line is above a certain threshold a number of causally dependent failures can occur:
   - Both sensor amplitudes have the same value which means that the sensor signals are corrupted, and
   - the firing lines of the microcontroller can be set needlessly to high.

   If the voltage is below $4V$, then the airbag system will be set to the inactive mode, which is indicated by a warning lamp. This can be considered to be a safe operational mode.

2. 24V-line: We distinguish two failures that may lead to hazardous situations:
   - If the voltage is too high, for instance above 40V, the FET and FASICs may be destroyed.
   - If the voltage is between 7 and 19V, the airbag system is in a degraded operational mode.

   If the voltage of this line is below 7V, the airbag system is inactive which means this is a safe operational mode.

4.5 FET Failures

The Field Effect Transistor (FET) can be compared to a switch.

1. It can close inadvertently and hence enable the FASIC to fire.
2. It can be open instead of being closed as requested and hence suppress ignition of the airbag.

4.6 FASIC Failures

The Firing Application Specific Integrated Circuit (FASIC) consists of two internal switches (Highside and Lowside switch).

1. It is possible that either one or both of these switches close inadvertently, or that one or both do not close as requested. In the first case, an ignition of the airbag is not possible as long as the FET is not activated. In the latter case a correct firing may be suppressed by the FASIC.

2. For diagnostic purposes the FASIC is connected to the voltage supply. If this line is connected to the output line of the FASIC due to an internal short circuit, the FET protection becomes useless and the airbag may be fired.

4.7 Bus/Connection Line Failures

Due to environmental conditions the connection lines from the sensors to the μC and the busses from the μC to both the FET and FASIC in the airbag system are subject to failures. These signals can be corrupted, thus potentially violating all three safety requirements.

4.8 Modelling Component Failures

The failure mode matrix that describes the change from fault-free to faulty behaviour is modelled as a PRISM module. In case of single component failures, it consists of a single transition, from normal behaviour (failure mode fm 0) to the failure mode n under consideration ($n = 1 \ldots 10$, cf. Table 1).

In case of multiple-component failures, this module becomes more complex, for example, for microcontroller and FASIC failure, the failure mode transition matrix encoded as PRISM module looks as follows:

```
module FailureViewMatrix
fm:[0..11] init 0; //Ten basic failure modes //
//combined failue modes is assigned a fresh value //
[] fm = 0 -> rate_MCFailure:(fm' = 3);
[] fm = 0 -> rate_FASICFailure:(fm' = 6);
[] fm = 3 -> rate_FASICFailure:(fm' = 11);
[] fm = 6 -> rate_MCFailure:(fm'=11);
```

Where `fm = 0` (`5, 6`) is the transition guard, i.e., the transition can only fire, if the guard condition is satisfied. As a consequence of taking the transition the failure mode is changed (`fm' = 5`, 6 or 11). In case of intermittent failures (sensor or bus line failures can be of that kind in this case study), transitions back to failure mode 0 have to be added. For all other components (FET, FASIC, μC, etc.) failure recovery is not considered.

The failures are injected into the basic model by adding to the resprective component transitions that model the effect of the failure; these transitions are made guarded, and can only be taken, if the system is in the corresponding failure mode. The transitions, that model the failure-free module behaviour are also made guarded, such that these transitions can only be taken if the failure under consideration has not yet occurred.

For example, consider the potential μC-failure of suppressing a fire command in case of a critical crash, this scenario is important for safety requirements 1 and 2 (cf. Sec. 4.1). In case of a failure (`fm = 3`), in the worst case, the fire signal is not sent, represented as transition from Crash back to *ReadSensors*, labelled with $\mathsf{fm} = 3/\mathsf{skip}!$. A simple state machine representation of this model can be found in Fig. 5. If the μC fails such that the signal is de-

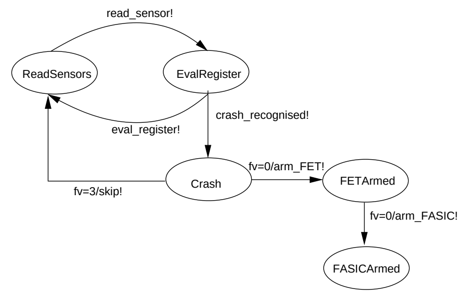

Figure 5. State machine representation of μC with injected failure

layed, this is modelled by assigning to the fire-command-tranistions a smaller rate. As no real data for this situation exists, TRW suggested to use half the rate that is applied in case of normal operation.

In the case, where a failure of the μC results in a needless deployment of the airbag (relevant for safety requirement 3), the fire command is sent, even, if no crash is recognised. In Fig 6 this situation is shown, by a transition labelled with $\mathsf{fm} = 3/\mathsf{skip}!$ from the initial state ReadSensors to the state Crash.

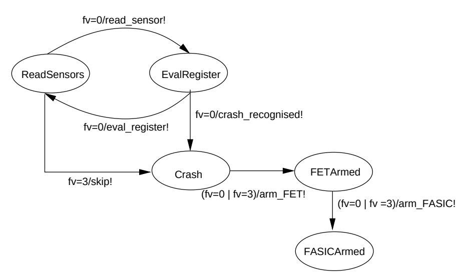

Figure 6. State machine representation of μC with injected failure

5 Analysis of the Airbag System

In order to perform the FMEA we conduct model checking experiments by injecting faults into the PRISM system model. Thereby, we proceed as described in detail in [14]. These faults are as described in Sec. 4. Faults can be single or joint faults. In other words, more than one component can fail.

5.1 Scenarios

To conduct the experiments, we assume that there is an environment which models the possible driving scenarios that there is 1) no crash, and 2) a frontal crash. For the first scenario the relevant hazard condition is the unintented ignition of the airbag. For the second scenario we are interested in the probability of a timely airbag ignition. Each scenario is considered in isolation as the analysis results would be useless otherwise. We are interested in the probability that the safety requirements from Sec. 4.1 are violated. If all possible scenarios were merged into a single model, such an analysis would be impossible.

5.2 Failure Modes and Experiments

For our analysis, we have identified one normal operation mode, which is referred to as Fm_0, and ten failure modes, Fm_1 to Fm_{10}. A short description of the failure modes can be found in table 1. While the official safety requirements standards only consider single failures it is possible in principle to experience simultaneous multiple component failures. Since our analysis approach is automated, we can easily accommodate mutliple component failures and consider the following combinations: 1) power supply- and microcontroller-failure, 2) FET- and FASIC-failure and 3) microcontroller-, FET- and FASIC-failure. Table 2 presents the results of the pFMEA for 2 microcontrollers in case of a critical frontal crash for the safety requirements 1 and 2 from Sec. 4.1 that are the relevant safety requirements for this scenario. We only considered failure modes Fm_0, Fm_1, Fm_3, Fm_4, Fm_6, and Fm_9. Where applicable, we have taken both permanent and intermittent failures into account. The system with one microcontroller also complies with ASIL D for safety requirements 1 and 2.

Failure view	Description
Fm_0	Normal operation
Fm_1	Sensor failure: Different amplitudes
Fm_2	Sensor failure: Amplitudes wrong
Fm_3	Summary failure of any μC-component
Fm_4	Power supply failure
Fm_5	FET failure
Fm_6	FASIC failure
Fm_7	μC-FET-line failure
Fm_8	μC-FASIC-line failure
Fm_9	Main sensor-line failure
Fm_{10}	Safing sensor-line failure

Table 1. Failure modes

We checked safety requirement 3 with varying time bounds for the case in which no critical crash occurs. We first analysed the one-microcontroller architecture and checked whether it complies with ASIL D. ASIL D can not be satisfied in all cases with this architecture. In the case of a microcontroller failure and power supply failures with $T_3 = 5$ and 10 hours mission time, the actual hazard rate was slightly above the THR. Therefore, the experiments were repeated with a model of the airbag system with two microcontrollers. The results of this experiment can be found in table 3.

	Requirement 1	Requirement 2
	violated (yes/no)?	
Fm_0	no	no
Fm_1, permanent failure	no	no
Fm_1, intermittent failure	no	no
Fm_3	no	no
Fm_4	no	no
Fm_6	no	no
Fm_9, permanent failure	no	no
Fm_9, intermittent failure	no	no

Table 2. Analysis results in case of a frontal crash (2 microcontrollers, ASIL D)

	$T_3 = 1h$	$T_3 = 5h$	$T_3 = 10h$
	Requirement 3 violated (yes/no)?		
Fm_0	no	no	no
Fm_1, per. failure	no	no	no
Fm_1, int. failure	no	no	no
Fm_3	no	no	no
Fm_4	no	no	no
Fm_6	no	no	no
Fm_9, per. failure	no	no	no
Fm_9, int. failure	no	no	no

Table 3. Analysis results for requirement 3, no crash (2 microcontrollers, ASIL D)

5.3 Multiple Failures and Counter Examples

We now consider the actual hazard rate for multiple failures, even if this is not demanded by the standard. We believe that it is important to go beyond the minimum safety requirements specified in the standard in order to design reliable safety critical systems.

While the airbag system with only one microcontroller complies with ASIL B, it does not comply with ASIL D in the case of multiple failures. We noticed that when microcontrollers and FET and FASIC failures occurred the actual hazard rate is significantly above the THR (cf. Sec. 4.1). In table 4 we can find the results of pFMEA analysis in case no accident occurs while considering multiple component failures. Therefore, we generated counter examples (cf. Sec. 5.5) for these cases in order to identify the primary source of the safety requirement violation. For 2 microcontrollers the actual hazard probability complies with the upcoming ASIL D.

	$T_3 = 1h$	$T_3 = 5h$	$T_3 = 10h$
	Requirement 3 violated (yes/no)?		
Fm_3 and Fm_{10}	no	yes	yes
Fm_5 and Fm_6	no	no	no
Fm_3, Fm_5 and Fm_6	no	yes	yes

Table 4. Analysis results for requirement 3, no crash, multiple failures (1 microcontroller, ASIL D)

5.4 Time and Space Complexity of Model Checking

The sizes of the models we encountered vary from 1,536 states for failure mode 0 and no crash, to 615,600 states for failure mode 2 with intermittent failures and crash. Memory thus, was not the problem here, the largest model only required 28.4 MB of storage, including iteration vectors for numerical analysis fits thus easily into main memory.

Where in the case of safety requirements 1 and 2 the model checking is very efficient despite large state spaces, it can be seen that in case of safety requirement 3 these times increase heavily. For the failure mode 2 model with intermittent failures, having 615,600 states and requirement 2, model checking took only 311 sec., whereas for safety requirement 3 and a mission time of 10 h for the same model, the model checking time exceeded 12 hours. This enormous increase can be explained by the fact that in the latter case the time bounds are extremely high in comparison to safety requirements 1 and 2. As the number of iterations for transient analysis is also determined by this value, we experience this increase in the model checking times.

5.5 Counter Example Support for pFMEA

Due to space restrictions, we only detail only the case where the μC, and both the FET and FASIC can fail. We chose especially this case, as we think it is interesting for the following reasons:

- the FASIC is the least reliable component,
- the μC is the central part of the system, and the correct airbag ignition depends on the results as delivered by the μC.

It is therefore interesting to see whether the reliability or the potential consequences of a component failure contribute more of the probability mass to the safety requirement violation.

Intuitively, one might expect that the FASIC, as the least reliable component contributes most to the violation of the ASIL-D property (cf. Table 4). Contrary to that, the computation of the counter example yielded, that in fact the μC is the contributes mostly to this violation. Based on these findings, a solution that makes the μC-part more reliable is more useful than trying to improve the reliability of the FASIC, although it is the least reliable component in the entire system.

These findings are underpinned by the fact, that TRW indeed uses a solution, that makes the microcontroller part redundant by introducing a second, safing microcontroller that alleviates the effects of a single μC failure.

A second, interesting finding of the analysis of the counter example is, is the observation, that true multi-component failures, i.e., failures where within the mission time considered more than one component fails do not contribute significantly to the violation of the ASIL-D property. In principle, these kind of failures are interesting to analyse. However, if they are unlikely, like in the airbag system, then it is valuable information that the treatment of this type of failures is dispensible since their analysis leads to a blow-up of the resulting state space.

For the generation of the counter example, we applied XBF search [1]. The generation time for this model with 6,618 states took about 50 minutes.

The results of the counter example generation may be either displayed in a purely textual way or they can be visualised. This visualisation relieves the user from browsing through potentially long textual files, that represent the error traces.

Using visualisation, the user can find at a glance the most probable error traces and interprete them. to see some pictures of the visualised counter example for μC, FET and FASIC failure. The explored part of the state space is visualised as a graph. Counter example states are surrounded with red lines and the action names of counter example transitions are rendered in red. These action names are useful in order to retrace the actions which lead to a failure state, which are rendered as diamonds. The size of the diamond is proportional to the reachability probability of the corresponding failure state. Hence, the size of the diamond indicates how significant the contribution of this particular failure state to the total system failure actually is.

In Fig. 7 we see that certain states are much bigger than other failure states. In order to see, to which component failures, these states belong, the user zooms into the graph. In Fig. 8, we see a zoomed in part of the state graph displayed in Fig. 7. This zoom reveals that that μC-failures are much more probable than FASIC failures. This can be deduced from the action labels in the path leading to failure states, that symbolise that a μC-failure occurred (action label *FailureView*3).

From Fig. 9 we can also see that the actual probability that FASIC resp. FET components are failing is quite small compared to the probability errors due to μC failures. In principle, multiple failures are interesting to analyse. However, if they are unlikely like in the air bag system, then it is valuable information that the treatment of this type of fail-

Approach & Year	Spec. Formal-ism	Tool	Prob. FMEA	Counter exam-ples	Case Studies
Heimdahl et al. 2005[19]	RSML^{-e}	NuSMV	No	Yes	Altitude Switch
Bozzano et al. 2003 [6]	NuSMV code	FSAP/ NuSMV-SA	No	Yes	Bit Adder
Cichocki & Górski 2000[8, 9]	CSP	FDR	No	No	Line Block System
Grunske et al. 2005 [16]	Behavior Trees	SAL	No	Yes	Metal Press
Elmqvist & Nadjm-Tehrani 2008 [11]	PRISM Reactive Modules	PRISM	Yes	No	Altitude Meter System
Grunske et al. 2007 [14]	PRISM Reactive Modules	PRISM	Yes	No	Metal Press
Our Approach	PRISM Reactive Modules	PRISM	Yes	Yes	Industrial case study (airbag system)

Table 5. Comparison with related approaches

ures is dispensible since their analysis leads to a blow-up of the resulting state space.

Visualisation is thus an auxiliary mean to relieve the user from reading, potentially large, textually represented error traces. Instead, visualisation provides an at-a-glance overview of failures, details can be looked up if required and are hidden by the graphical representation if the user does not need them.

6 Related Work

A considerable number of approaches have been proposed to automate or support the FMEA process with model checking [6, 7, 8, 9, 16, 19, 25]. The existing approaches are summarised in Figure 5. From this comparison it becomes evident that only the approaches described in [11] and [14] use probabilistic model checking and support a probabilistic FMEA process. All other approaches work with traditional model checking tools. The novel aspect described in this paper with respect to the approaches in [11] and [14] is the support (generation and analysis) of counter examples. These counter examples provide valuable insights in the cause consequence relations between low level component failures and system level hazards. Furthermore, while all the existing approaches only work with small academic examples, a central contribution of this paper is to provide evidence that the process also works on a medium size application taken from industry.

7 Lessons Learnt

7.1 System Modelling

With respect to system modelling, we have learnt the following lessons: At first, even PRISM is not able to deal with signal of continuous nature, and the model at first sight, requires to deal with such continuous signals, it is still possible to model such a system, by applying useful abstractions. Such abstractions were used for modelling crash events, where we could reduce a continuous signal to a discrete model with only five states. It was found, that these abstraction does not render the findings of the experiments invalid.

At second, it is trivial, but interesting, that the state space size or structure is not the only limiting factor for the applicability of stochastic model checking. We had to deal with large time bounds. The time bounds influence the number of iterations needed for transient analysis. So, even for moderate size models, here models having at most 615,600 states, the time for numerical analysis may become prohibitive. This is an observation, which is generally true for safety critical systems, that have long mission (run) times.

At third, we learnt, that PRISM itself is a language that the engineers at TRW can learn quickly, but the logic CSL itself is considered to be "exotic", therefore the pattern-based approach as, for example suggested in [13] may be a step towards further proliferation of pFMEA in the industry.

7.2 Implications for Industry

There are a number of potential benefits from the adoption of probabilistic FMEA in industry. First of all, it is a tool to to check with a reasonable effort which reliability requirements are satisfied by an existing state of the art design. In this case, we saw that although an existing single path airbag system is reliable in the field, not all new safety requirements are fulfilled. This result also corresponds to the decision that in order to fulfill all safety requirements, future systems have to be built with two redundant paths to increase the reliability.

Second, probabilistic FMEA is a technique that can be used at early stages of the system development process to evaluate the reliability of the designed system and to identify weak paths with a high failure probability in the architectural design. The upcoming standard ISO 26262 defines the goal to decrease the number of unintended behaviour of electronic components in the car and it demands the assessment of design alternatives to find the one alternative that is the most reliable one. The proposed approach facilitates and supports this assessment and provides a basis for the technical discussion and comparison of design alternatives.

Third, due to the fact that the analysis is automated and supported by tools it is possible to investigate much more complex scenarios than with a manual analysis, such as for instance multiple failures.

8 Conclusion

In this paper, we have presented a case study for applying probabilistic FMEA to an industrial airbag system. The system was modelled and analysed using the PRISM tool. We have considered a system with two different configurations (one and two microcontrollers). By applying probabilistic FMEA the two system configurations were checked whether they comply with the upcoming safety standard for road vehicles ASIL D with respect to a large number of possible component failures. For the system variant with one microcontroller, we found the ASIL D requirements to be violated. Using counter example generation and visualisation, we were able to identify the main source of the requirement violation.

Although the research presented in this paper provides evidence of the applicability of the pFMEA process to industrially relevant systems, there are still some task for future research to improve the scalability of the approach. Improvements to the performance of probabilistic model checking and counterexample generation algorithms would in particular further enhance the applicability of the pFMEA process.

Acknowledgements: The authors wish to thank Dr. Johannes Konle and Richard Cording of TRW Automotive GmbH for supporting this cooperation. We especially thank Martin Brügel for his essential assistance in the FMEA process.

References

[1] H. Aljazzar and S. Leue. Extended Directed Search for Probabilistic Timed Reachability. In *Proceedings of FORMATS '06*, volume LNCS 4202, pages 33–51. Springer, 2006.

[2] H. Aljazzar and S. Leue. Debugging of Dependability Models Using Interactive Visualization of Counterexamples. In *Proceedings of QEST 2008*, pages 189–198. IEEE Computer Science Press, 2008.

[3] H. Aljazzar and S. Leue. K*: A Directed On-The-Fly Algorithm for Finding the k Shortest Paths. Technical Report soft-08-03, University of Konstanz, Chair for Software Engineering, 2008.

[4] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Verifying continuous time Markov chains. In *Computer-Aided Verification*, volume LNCS 1102, pages 146–162. Springer, 1996.

[5] C. Baier, B. Haverkort, H. Hermanns, and J. Katoen. Model-Checking Algorithms for Continuous-Time Markov Chains. *IEEE Trans. Software Eng.*, 29(7):1–18, July 2003.

[6] M. Bozzano, A. Cavallo, M. Cifaldi, L. Valacca, and A. Villafiorita. Improving Safety Assessment of Complex Systems: An Industrial Case Study. In *Proceedings of FME 2003*, volume LNCS 2805, pages 208–222. Springer, 2003.

[7] M. Bozzano and A. Villafiorita. Improving System Reliability via Model Checking: The FSAP/NuSMV-SA Safety Analysis Platform. In *Proceedings of SAFECOMP 2003*, volume LNCS 2788, pages 49–62. Springer-Verlag, 2003.

[8] T. Cichocki and J. Górski. Failure Mode and Effect Analysis for Safety-Critical Systems with Software Components. In *Proceedings of SAFECOMP 2000*, volume LNCS 1943, pages 382–394. Springer, 2000.

[9] T. Cichocki and J. Górski. Formal Support for Fault Modelling and Analysis. In *Proceedings of SAFECOMP 2001*, volume LNCS 2187, pages 190–199. Springer, 2001.

[10] E. M. Clarke, E. A. Emerson, and A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986.

[11] J. Elmqvist and S. Nadjm-Tehrani. Formal support for quantitative analysis of residual risks in safety-critical systems. In *Proceedings of HASE 2008*, pages 154–164. IEEE Computer Society, 2008.

[12] D. Eppstein. Finding the k shortest paths. *SIAM J. Computing*, 28(2):652–673, 1998.

[13] L. Grunske. Specification patterns for probabilistic quality properties. In Robby, editor, *Proceedings of the 30th International Conference on Software Engineering (ICSE 2008), Leipzig, Germany, May 10-18, 2008*, pages 31–40. ACM, 2008.

[14] L. Grunske, R. Colvin, and K. Winter. Probabilistic Model-Checking Support for FMEA. In *Proceedings of QEST 2007*, pages 119–128. IEEE Computer Science Press, 2007.

[15] L. Grunske, B. Kaiser, and R. H. Reussner. Specification and Evaluation of Safety Properties in a Component-based Software Engineering Process. In *Embedded Software Development with Components -An Overview on Current Research Trends*, pages 737–738. Springer-Verlag, 2005.

[16] L. Grunske, P. A. Lindsay, N. Yatapanage, and K. Winter. An Automated Failure Mode and Effect Analysis Based on High-Level Design Specification with Behavior Trees. In *Proceedings of IFM 2005*, volume LNCS 3771, pages 129–149. Springer, 2005.

[17] H. A. Haapanen Pentti. Failure Mode and Effects Analysis of Software-based Automation Systems. Technical report, VTT Industrial Systems, Helsinki, STUK-YTO-TR 190, 37 pp, 2002.

[18] T. Han and J.-P. Katoen. Counterexamples in Probabilistic Model Checking. In *Proceedings of TACAS 2007*, volume LNCS 4424, pages 60–75. Springer, 2007.

[19] M. P. E. Heimdahl, Y. Choi, and M. W. Whalen. Deviation Analysis: A New Use of Model Checking. *Automated Software Engineering*, 12(3):321–347, 2005.

[20] International Electrotechnical Commission. Analysis Techniques for System Reliability - Procedure for Failure Mode and Effects analysis (FMEA), IEC 60812, 1991.

[21] International Electrotechnical Commission. Functional Safety of Electrical/Electronic/Programmable Electronic

Safety-related Systems, IEC 61508, , 2000.

[22] International Organization for Standardization. Road Vehicles Functional Safety, ISO 26262 (Commitee Draft), 2008.

[23] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer (STTT)*, 6(2):128–142, 2004.

[24] J. Pearl. *Heuristics – Intelligent Search Strategies for Computer Problem Solving*. Addison–Wesley, 1986.

[25] J. D. Reese and N. G. Leveson. Software Deviation Analysis. In *Proceedings of ICSE 1997*, pages 250–261. ACM Press, 1997.

[26] K. S. Trivedi. *Probability and statistics with reliability, queuing and computer science applications*. John Wiley and Sons Ltd., 2002.

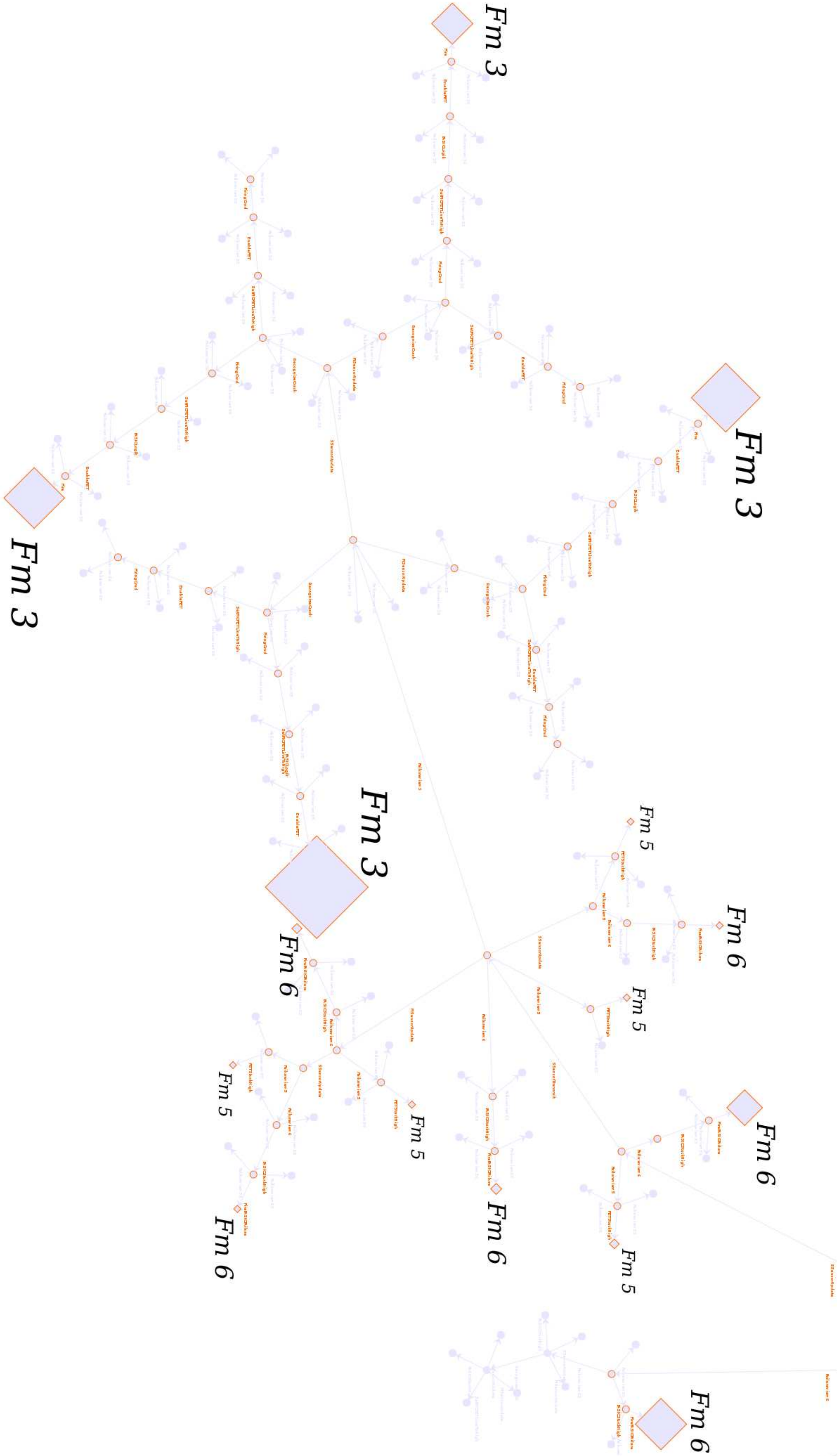

Figure 7. A portion of the counter example visualisation in the case of μC, FET and FASIC failures

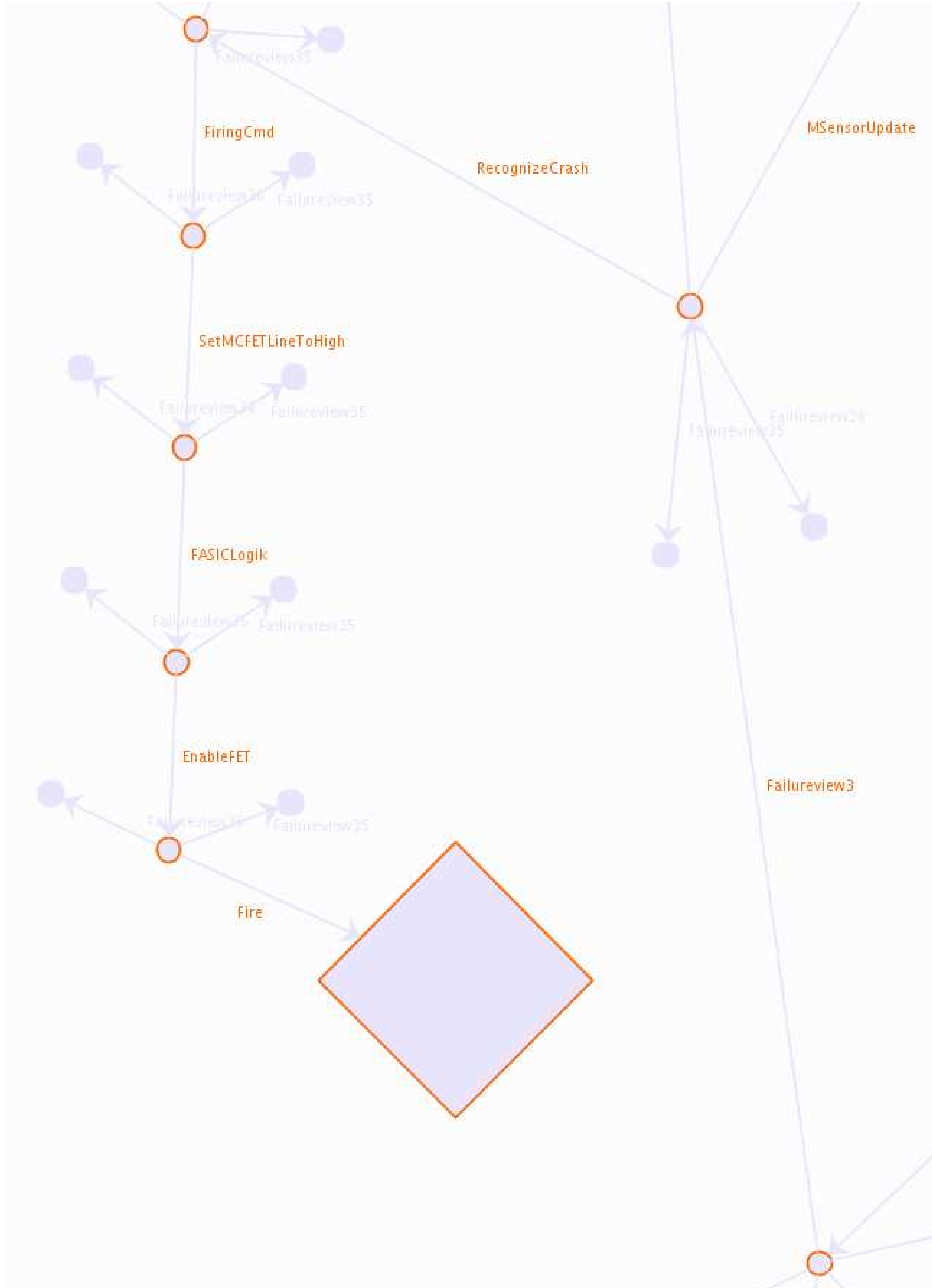

Figure 8. Zoomed portion of Fig. 7: μC failures

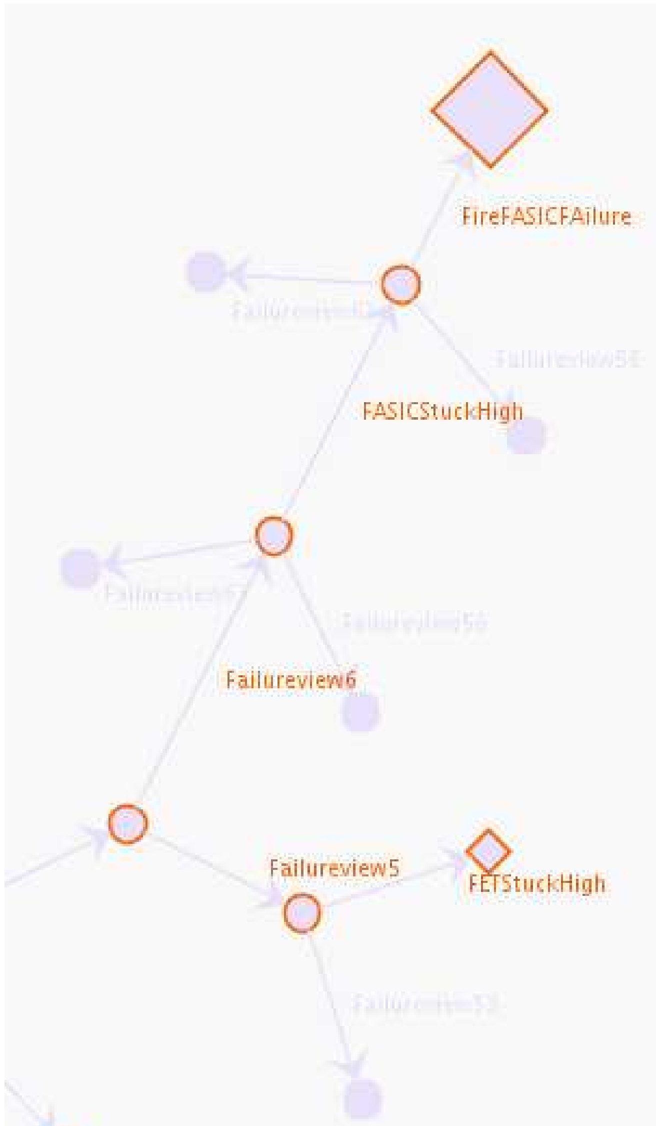

Figure 9. Zoomed portion of Fig. 7: combined FET and FASIC failures