Combining Functional and Structural Reasoning for Safety Analysis of Electrical Designs

Price, Chris; Pugh, Dave; Snooke, Neal; Hunt, John; Wilson, Myra

Published in:
Knowledge Engineering Review

Publication date:
1997

Citation for published version (APA):
Price, C., Pugh, D., Snooke, N., Hunt, J., & Wilson, M. (1997). Combining Functional and Structural Reasoning for Safety Analysis of Electrical Designs. *Knowledge Engineering Review*, *12*(3), 271-287.

The Knowledge Engineering Review, Vol. 12:3, 1997, 271–287

Combining functional and structural reasoning for safety analysis of electrical designs

C. J. PRICE, N. SNOOKE, D. R. PUGH, J. E. HUNT and M. S. WILSON

Department of Computer Science, University of Wales, Aberystwyth, Ceredigion, SY23 3DB, UK, email: cjp@aber.ac.uk

Abstract

Increasing complexity of design in automotive electrical systems has been paralleled by increased demands for analysis of the safety and reliability aspects of those designs. Such demands can place a great burden on the engineers charged with carrying out the analysis. This paper describes how the intended functions of a circuit design can be combined with a qualitative model of the electrical circuit that fulfils the functions, and used to analyse the safety of the design. FLAME, an automated failure mode and effects analysis system based on these techniques, is described in detail. FLAME has been developed over several years, and is capable of composing an FMEA report for many different electrical subsystems. The paper also addresses the issue of how the use of functional and structural reasoning can be extended to sneak circuit analysis and fault tree analysis.

1 Introduction

Failure Mode and Effects Analysis (FMEA) of a design involves the investigation and assessment of the effects of all possible failure modes on the system being designed. For electrical design, it means examining what would happen if any component failed or if any wire in a circuit went open circuit, shorted to ground or shorted to battery unexpectedly. This kind of analysis is of growing importance in the automotive, aerospace and other advanced manufacturing industries, where increasingly complex electrical, electronic and mechanical systems are being combined in safety-critical applications.

Automation of FMEA is an attractive proposition, as it is a tedious and repetitive task, yet one which is time consuming and needs to be done by skilled engineers. Previous research on automation of FMEA has concentrated on automating a few aspects of the FMEA process. Some work has concentrated on assisting the engineer with management of FMEA information (Legg, 1978), or on manipulating significance values entered by engineers (Kara-Zaitri et al., 1992). A second important strand of research has automated the generation of effects for each failure mode (Bell et al., 1992; Palumbo, 1994; Price et al., 1992). The work on the FLAME system described in this paper combines these aspects with an automatic generation of significance values, thus providing a complete automated FMEA assistant.

FLAME has evolved over a period of five years, moving gradually from research prototypes that addressed only the difficult issue of generating failure effects (Price & Hunt, 1991), through systems capable of covering the whole FMEA process but demanding much modelling effort from the user (Price et al., 1995) to the present system, capable of producing FMEA reports for most of the electrical systems in a modern car but demanding little modelling input from the user. The paper describes both the technology underlying the FLAME system, and the kind of user interface needed to make it usable by engineers. Finally, it addresses the use of the technology in the FLAME system for other types of safety analysis.

The rest of the paper is arranged as follows. Section two of the paper deals with the demands that an automated FMEA system must meet in order for the engineer to regard it as an improvement over generating an FMEA report by hand, or using a clerical FMEA tool to help produce a neatly printed FMEA report. Section three describes the FLAME system from the engineer's point of view. Section four describes the technology underlying the FLAME system, including details of the functional representation used and the qualitative circuit simulation techniques applied. Section five evaluates how well the FLAME system meets the criteria put forward in section two. Section six shows how the technology used here for FMEA can also be applied to other design analysis techniques. Section seven examines related work by other researchers, and compares it with the work described here.

2 Challenges for an automated FMEA system

Failure Mode and Effects Analysis (FMEA) involves the investigation and assessment of the effects of possible failure modes on a system (Coker et al., 1989; IEC, 1995). For electrical system designs, this involves investigating what happens when each wire in the circuit goes open circuit, shorts to ground, or shorts to battery, or when other components fail in expected ways (for example, a pump might stall or a relay be stuck at open). The significance of each failure is assessed and the failure is assigned a Risk Priority Number (RPN). The RPN value for a failure is obtained by multiplying three separate values each on a scale of 1 to 10 (where 10 is the most serious):

- Severity: how bad the consequences of the failure are.
- Detection: whether the failure can be detected.
- Occurrence: how frequently the failure will occur.

Automation of the record-keeping aspects of FMEA is fairly widely accepted, acting as a sort of cross between a word processor and a spreadsheet program. It can make the entry of information much simpler, but leaves the main burden of filling the form—generating all the effects and assigning significance values—squarely on the shoulders of the engineers.

On the other hand, the efforts to automate the generation of effects and significance values have not been developed enough to use for practical tasks. Experience of evaluating prototype effect generation systems with engineers has generated a list of requirements for an automated FMEA assistant. These requirements need to be met for such systems to be accepted in everyday practice.

- **Help for the whole FMEA report generation process**. It is not enough just to generate effects for failure modes. Engineers are capable of generating answers for themselves. The maximum benefit comes from integrating the effects generation with the significance assignment and the form filling.
- **Timely assistance**. FMEA is often performed late in the design cycle, once detailed information, such as wire lengths, is known. However, the effectiveness of FMEA depends on reacting to any problems that are highlighted. That means that it should be possible to perform the automated FMEA without having the information needed for an accurate simulation in a quantitative simulator such as Saber (Montgomery et al., 1996).
- **Reuse of information**. Where information already exists, the engineer should be able to access it rather than reenter it into the automated system. This both increases efficiency and takes away any possibility of erroneous entry.
- **Usable by engineers**. The engineers should be able to enter any extra information needed themselves rather than requiring computer experts to do it for them.
- **Greatly improved efficiency**. Resistance to process change means that it needs to be significantly less effort to use the automated system than to perform the analysis by hand. Such efficiency must be an improvement of at least an order of magnitude in order for new technology to be adopted enthusiastically.

Any attempt to automate the FMEA process needs to be assessed against the criteria outlined above.

3 The FLAME system

The FLAME system provides automated assistance to the engineer for all aspects of FMEA report generation. Use of the FLAME system can be broken down into three stages:

- Model construction.
- FMEA generation.
- Interactive FMEA verification.

The relationship between the three stages is depicted in Figure 1. The rest of this section will explain what happens during each stage, illustrating the discussion with the example of a car headlight system. This system is simpler than the kind of system that would be considered in practice. The lighting system would be considered as a whole, including items such as fog lamps, sidelights and indicator lights. The FLAME system is capable of dealing with that degree of complexity. This simpler system has been used as an illustration because it can be fully explored within the limitations of the paper.

3.1 Model construction

3.1.1 Capturing the circuit design

The model construction phase of using FLAME has been streamlined to use existing information as much as possible. The first stage is to obtain a description of the circuit to be analysed. The circuit will have been designed using an Electrical Computer Aided Design (ECAD) tool to draw the electrical components and the connections between components. The FLAME system contains routines to import circuit descriptions from an ECAD tool and to verify that FLAME's component library contains descriptions of the operation and failure modes for each type of component. The headlight circuit is composed of common components such as wires, relays and lamps, and so there is a description of all components except for the operation of the electronic control unit (ECU). The engineer uses the Component Builder (described in section 4.2) to describe the operation of the ECU.

Once the engineer has a complete circuit description, as illustrated in Figure 2, it is possible to simulate the operation of the circuit under different conditions by opening and closing switches. The circuit simulator will show which paths are active by changing the colour of components. This facility can be used to ensure that the circuit has the correct behaviour when all components are working properly.

3.1.2 Describing the system's functionality

The circuit design is not enough to be able to generate meaningful effects for failures of the system being examined. Knowledge of the intended functions of the system is also necessary to interpret the state of the circuit in ways that are meaningful in an FMEA report.

In earlier versions of the FLAME system (e.g. Price et al., 1995), the functionality of a system was described as a set of states of the system and details of the transitions between the states. Evaluation of this scheme with engineers showed that it had several drawbacks:

- The engineers found it difficult to construct functional descriptions of how a system worked.
- Linking the functional description to states of the circuit was also a challenge.
- If the functional descriptions were badly constructed, then they were not reusable between different versions of the same design.

These problems have been overcome by a radical simplification of the functional layer. For each subsystem in a car, FLAME has a set of functional labels that describe what the system is intended to do. These labels are held in a database, along with numbers used in RPN generation that describe the severity and detection values for absence of the function when it is expected (columns headed *fail* in Figure 3), and presence of the function when it is not expected (columns headed *succ* in Figure 3).

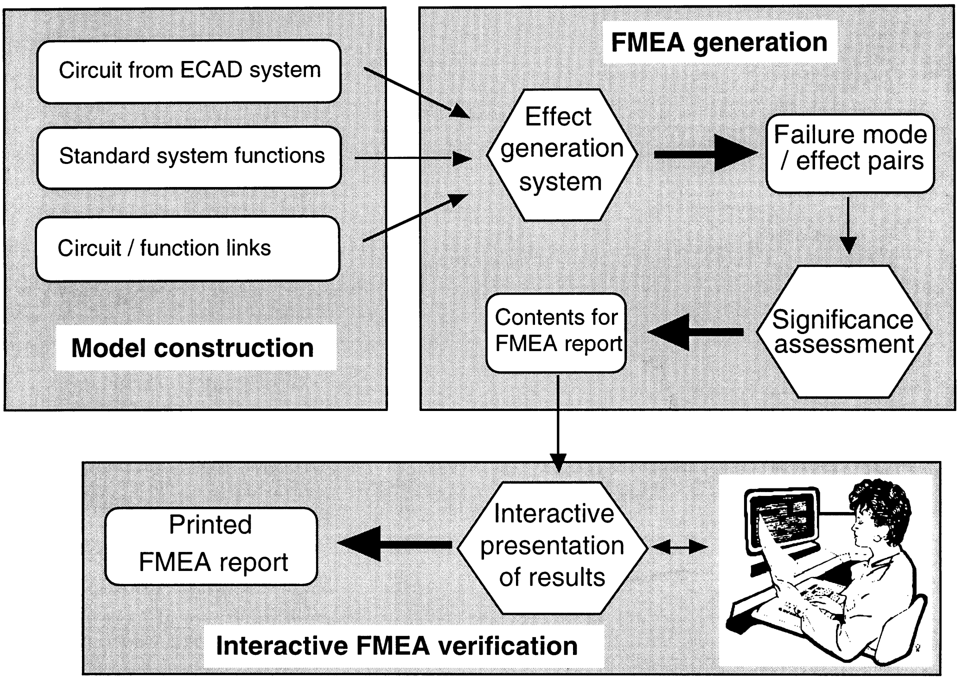
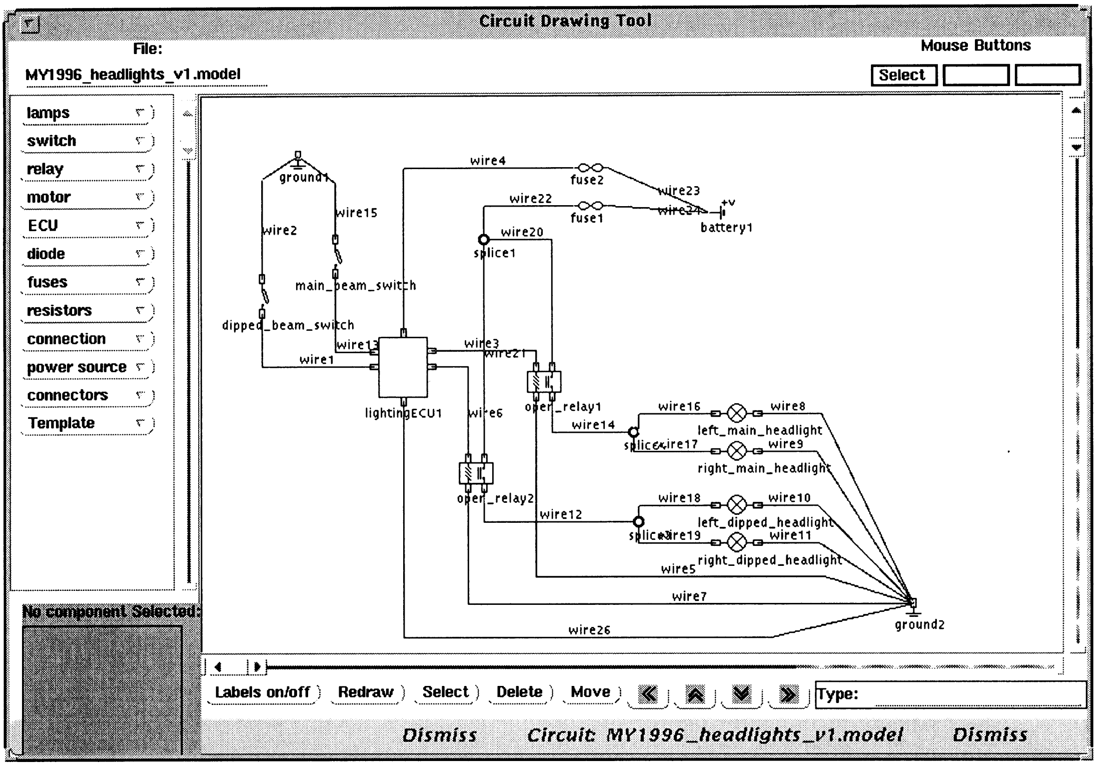


Figure 1 Stages of the use of the FLAME system

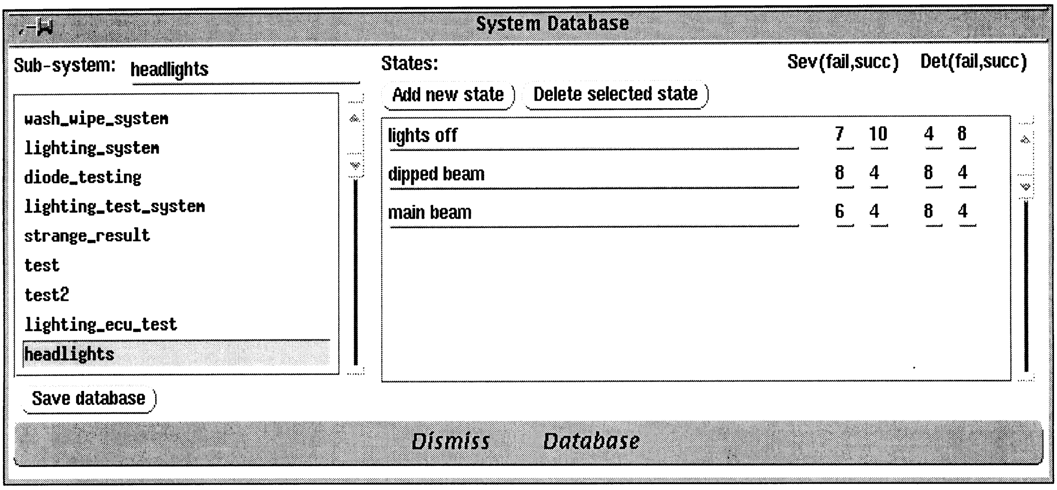

Circuit Drawing Tool

File:
MY1996_headlights_v1.model

Figure 2 Headlight circuit imported to FLAME circuit drawing tool

These functional labels are highly reusable provided that the automobile designers decompose the electrical system into subsystems in the same manner for each car. This seems like a reasonable restriction, given the benefits in reusability that it provides.

The reusable functions for the headlight system are shown in Figure 3. The engineer can add extra



Sub-system: headlights

wash_wipe_systen
lighting_systen
diode_testing
lighting_test_systen
strange_result
test
test2
lighting_ecu_test
headlights

States:	Sev(fail,succ)		Det(fail,succ)	
lights off	7	10	4	8
dipped beam	8	4	8	4
main beam	6	4	8	4

Figure 3 Functions for the headlight system

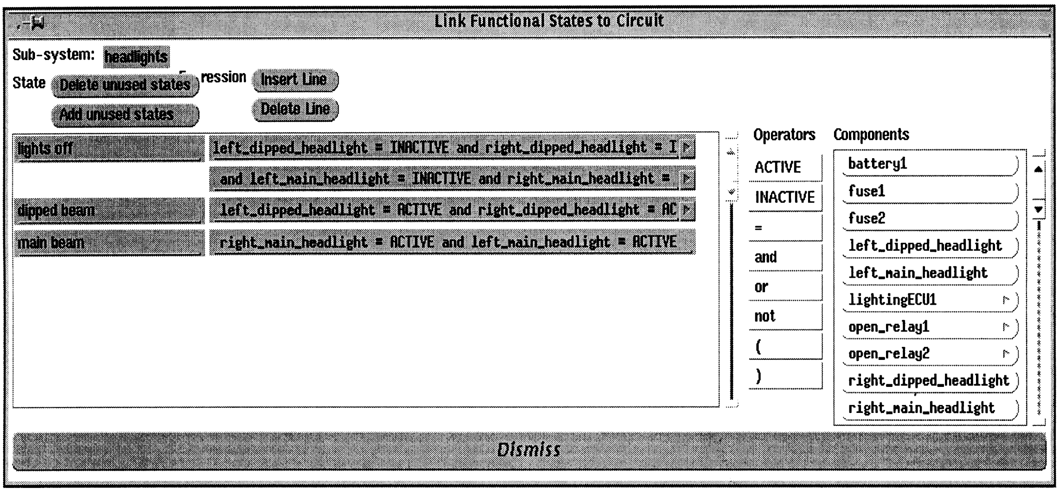
functions or change the values if necessary, but in this case they can be used as they are, so having examined them, the engineer dismisses the display.

3.1.3 Linking function to structure

To be able to interpret the state of the circuit, the engineer must link the functional labels to the state of significant components in the circuit. In this particular example, it is easily done. The main headlights are on when the main lamps are active, the dipped headlights are on when the dipped lamps are active, and no lights are on when no lamps are active. This linking of structure to function must be done for each new design of the subsystem, and it is shown for the headlight system in Figure 4. The expressions are constructed by clicking on a list of available components, legal operators and allowable states for each component. This makes the linking much simpler and less error-prone than it would be if the engineer had to type in the text of the expression.

3.1.4 Constructing FMEA tests

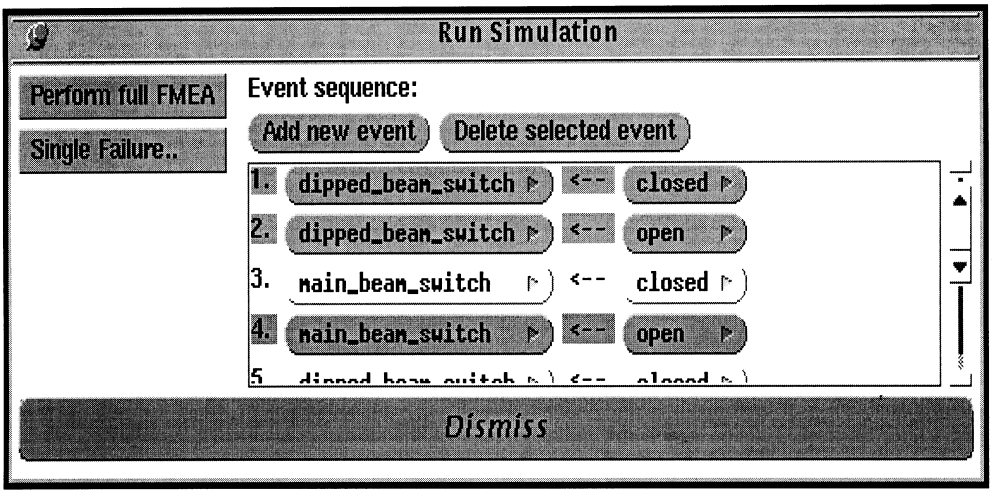
To decide on the interesting effects of any failure, it is necessary to describe how the headlight system will be used, so that the described use can be simulated for each possible failure. Figure 5 shows that the headlight system will be tested by switching the dipped headlights on, then switching them off,



Sub-system: headlights

lights off	left_dipped_headlight = INACTIVE and right_dipped_headlight = I
	and left_main_headlight = INACTIVE and right_main_headlight =
dipped beam	left_dipped_headlight = ACTIVE and right_dipped_headlight = AC
main beam	right_main_headlight = ACTIVE and left_main_headlight = ACTIVE

Figure 4 Linking headlight functions to structure for headlight system



Event sequence:

1. dipped_beam_switch <-- closed
2. dipped_beam_switch <-- open
3. main_beam_switch <-- closed
4. main_beam_switch <-- open
5 dipped_beam_switch <-- closed

Figure 5 Simulation states for testing headlight system

switching the main headlights on, then switching them off, and finally switching the dipped headlights on again.

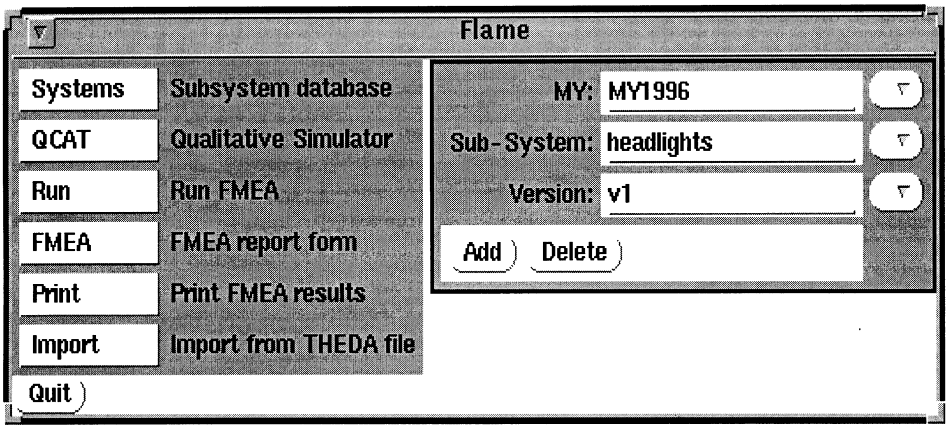
In more realistic cases than the simple headlight example shown here, it is impossible to provide complete coverage of all possible combinations of switch states in a circuit. The engineer needs to identify all combinations of states likely to be interesting for a failure. In practice, this is likely to be all combinations that occur as normal operating states.

3.2 FMEA generation

FMEA generation is performed in batch mode. Once the FLAME system has all the model components described above, the engineer can choose "Run FMEA" from the main control panel of the FLAME system (see Figure 6).

The expected behaviour of the circuit can be obtained by running the circuit simulator through the switch states set up in section 3.1.4 with a correct version of the circuit.

For each failure mode of each component in the circuit, the FLAME system will simulate the behaviour of the circuit under each of the changes described in section 3.1.4. The functional labels are used to interpret the state of the circuit after each change, and differences from the expected functions in each state are noted. The difference between the expected functions in each state and the actual functions in each state are the effects for that failure mode. For example, if the dipped



MY: MY1996

Sub-System: headlights

Version: v1

Figure 6 Main control panel of FLAME system

headlight switch was turned on and the dipped headlight function did not occur, the failure effect would be that the dipped headlight function did not occur when expected.

Once all failure effects have been generated, the FLAME system calculates significance values for each effect. For some complex effects, the FLAME system might not be able to decide on the effects or might be unable to decide on the significance values. Where that is the case, the entry is left blank, and the engineer is expected to fill in the answer during the interactive FMEA verification phase.

3.3 *Interactive FMEA verification*

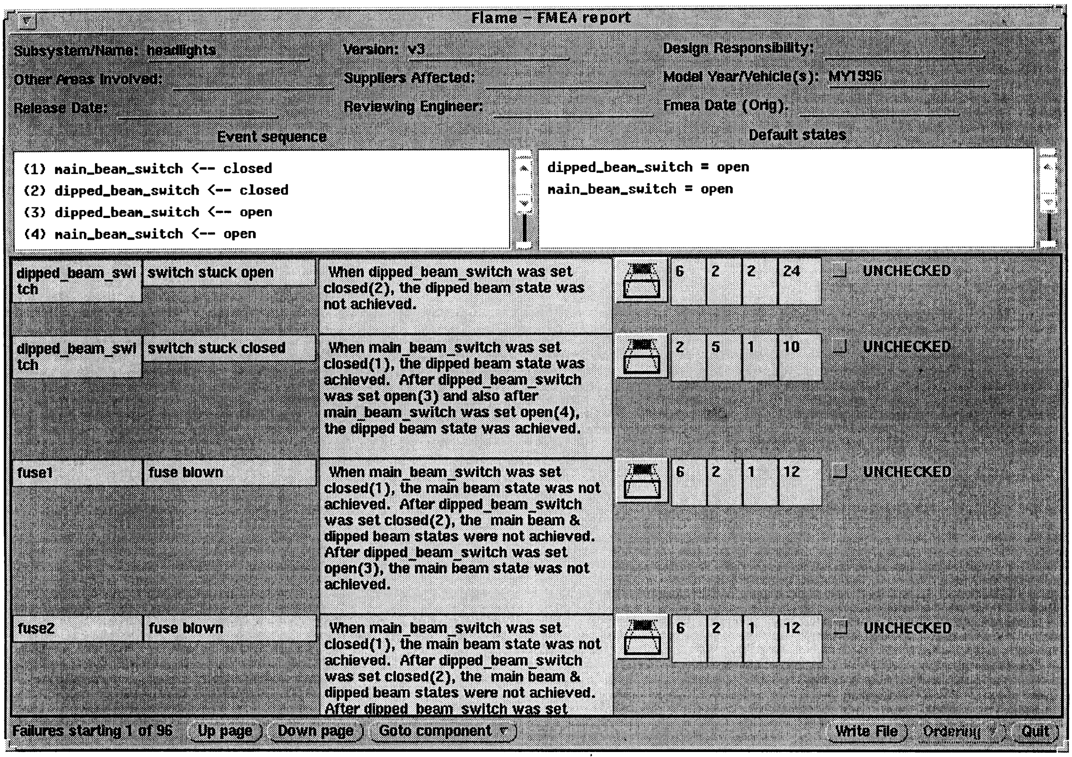
The reason for performing an FMEA is to identify potential problems in a design and to rectify them. This means that the engineer must examine the results and identify where action needs to be taken. The FLAME system removes much of the effort of producing the FMEA report, but responsibility for the correctness of the report is left with the engineer.

The results of the FMEA can be examined on the screen by the engineer, who can fill in missing details, amend reports or alter significance values. Such changes are annotated with the name of their originator. An example screen of the FMEA report for the headlight system is shown in Figure 7. The results of the FMEA can be manipulated in several ways: the most significant failures can be shown first, or those with missing values, or the failures which have changed since the previous design for the circuit. When the engineer is satisfied with the contents of the FMEA, it can be turned into a printed report.

4 Technical aspects of FLAME

The previous section dealt with the way in which the engineer used the FLAME system. This section deals with important aspects of the technology underlying the ability to produce an FMEA report.

The most challenging aspect is being able to generate failure effects for an electrical design given a



Subsystem/Name: headlights
Version: v3
Design Responsibility:
Other Areas Involved:
Suppliers Affected:
Model Year/Vehicle(s): MY1996
Release Date:
Reviewing Engineer:
Fmea Date (Orig).

Event sequence

```
(1) main_beam_switch <-- closed
(2) dipped_beam_switch <-- closed
(3) dipped_beam_switch <-- open
(4) main_beam_switch <-- open
```

Default states

```
dipped_beam_switch = open
main_beam_switch = open
```

dipped_beam_switch	switch stuck open	When dipped_beam_switch was set closed(2), the dipped beam state was not achieved.	6	2	2	24	□ UNCHECKED
dipped_beam_switch	switch stuck closed	When main_beam_switch was set closed(1), the dipped beam state was achieved. After dipped_beam_switch was set open(3) and also after main_beam_switch was set open(4), the dipped beam state was achieved.	2	5	1	10	□ UNCHECKED
fuse1	fuse blown	When main_beam_switch was set closed(1), the main beam state was not achieved. After dipped_beam_switch was set closed(2), the main beam & dipped beam states were not achieved. After dipped_beam_switch was set open(3), the main beam state was not achieved.	6	2	1	12	□ UNCHECKED
fuse2	fuse blown	When main_beam_switch was set closed(1), the main beam state was not achieved. After dipped_beam_switch was set closed(2), the main beam & dipped beam states were not achieved. After dipped_beam_switch was set	6	2	1	12	□ UNCHECKED

Failures starting 1 of 96

Figure 7 FMEA output for headlight system

specific failure mode. This is achieved by qualitative simulation of the circuit design combined with descriptions of the overall functions of the circuit.

The other noteworthy area is the automatic generation of RPN values from the deduced effects. This is accomplished by rule-based combination of significance values assigned to different effects and by reuse of previous results.

4.1 Deciding on effects of a failure mode

The effects generation software takes as input the following products of the model construction stage:

- the functions of the subsystem to be analysed;
- a description of the structure of the circuit to be analysed;
- a set of links between the functions and the circuit structure;
- a list of events which change the state of the circuit (e.g. closing a switch).

The following additional information can be obtained by the system:

- **A list of failure modes for each component in the circuit**. The FLAME system has a library containing details of how each component operates, and this includes details of the different ways in which that component can fail. The component library has been constructed within the user companies, and appropriate failure modes for each type of component have been identified by the engineers.
- **The correct behaviour of the circuit**. This is obtained by loading the description of the circuit into the qualitative circuit simulator, then applying in turn each event in the list of events. After each event, the possible functions are matched against the state of the circuit, in order to ascertain which functions are achieved by each event. For each event, this produces the expected effects of that event, for example:

  When the dipped beam switch is closed, the “dipped beam” function occurs.
  When the dipped beam switch is opened, the “no lights” function occurs.
  When the main beam switch is closed, the “main beam” function occurs, etc.

The generation of each failure effect is done by the following method:

```
For each failure mode of each component:
  Load a version of the circuit with that failure included into the circuit simulator.
  For each event in the list of events:
    Apply the event to the circuit simulator.
    Match the possible functions against the state of the circuit in order to identify which functions occur.
    Compare the functions that occur with the functions that were expected to occur (the correct behaviour for that event).
    Add the differences to the list of unexpected effects.
```

So, for example, take the failure mode where wire 1, the output wire from the dipped beam switch, fails open. The circuit simulator loads a circuit where wire 1 has infinite resistance, and then applies the set of events as was done for the correct behaviour. The results of the simulation are:

When the dipped beam switch is closed, the “no lights” function occurs.
When the dipped beam switch is opened, the “no lights” function occurs.
When the main beam switch is closed, the main beam function occurs, etc.

When these results are compared with the expected behaviour, the following differences are obtained:

When the dipped beam switch is closed, the "dipped beam" function was expected, but the "no lights" function occurred.

When the dipped beam switch is opened, the expected functions occurred.

When the main beam switch is closed, the expected functions occurred, etc.

The circuit simulator is vital to the process of generating failure effects, and merits a more complete description of its operation.

4.2 The qualitative circuit simulator

FLAME'S qualitative circuit simulator consists of two parts: a network analyser, and a controller which converts a circuit description into a form understood by the analyser, and dynamically monitors and updates component interdependencies.

4.2.1 The network analyser: CIRQ

The network analyser, based on CIRQ (Lee & Ormsby, 1993), takes as input a graph made up of qualitative resistances which represent either a component or part of a component; the resistances can take the values of zero, load or infinity. The output generated by CIRQ consists of a qualitative description of the electrical state of each resistance. This description will indicate active and inactive paths, as well as any short paths in the circuit. The CIRQ algorithm is based upon Dijkstra's shortest distance algorithm (Dijkstra, 1959).

The analysis of a circuit network is split into two stages. The first stage labels the terminals of each resistance in the graph with a forward/reverse (F/R) value; this specifies the number of loads which will be traversed to reach the negative and positive terminals respectively. In some instances, it will not be possible to reach one of the terminals, in which case "infinity" is used instead. Figure 8 shows a simple CIRQ graph labelled with F/R values. To aid readability, nodes for wires have been omitted from the example network. Switch1 is closed (resistance 0), switch2 is open (resistance infinity).

In the second stage, deciding which paths are active, short, and inactive, the network is traversed using a form of depth first traversal. All components whose terminal F/R values include an infinity are immediately marked inactive. Short paths are identified by a branch of the circuit having the same (non-infinite) F/R value at both ends; this implies that no load is being drawn by this part of

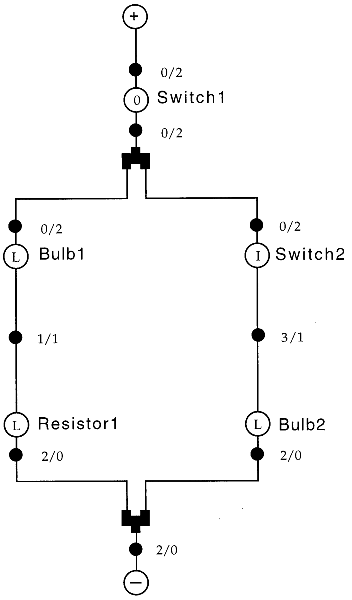


Figure 8 Resistance graph and F/R values for simple circuit

the circuit. Components on other branches between these two points will be marked inactive (assuming that these branches are not short paths also) as the zero-resistance branch will draw all current. All other paths are marked active, i.e., they have not been shorted out, and are not inactive.

4.2.2 Dynamically analysing a circuit: QCAT

The example used to describe the CIRQ algorithm contained only simple components; that is, components which could be represented as a simple resistance value. The question is, how can CIRQ be used to analyse circuits which contain components whose behaviour cannot be represented in this way? For example, relays, ECUs, multi-way switches, etc.

QCAT provides a computational layer above CIRQ which allows circuits to be analysed dynamically. For example, if a circuit contained an open relay, the relay would be represented as two resistance values—one for the coil, one for the switch. The status of the switch will be defined as being dependent upon the state of the coil; that is, if the coil is active then the switch will be closed, otherwise the switch will be open. To find out the state of a circuit containing an open relay, QCAT does the following:

- Find out the electrical state of the circuit using CIRQ with the relay switch resistance set to infinity (representing an open switch).
- If CIRQ returns with the result that the relay coil is active, then set the switch resistance to zero. Pass the new resistive network to CIRQ for analysis.
- If the coil is still active, then closing the switch did not affect the state of the coil. Stop processing.
- If the coil is inactive, set the switch state to infinity and re-simulate.

The processing will continue until the circuit reaches equilibrium or a feedback loop is identified.

4.2.3 Building realistic components: the Component Builder

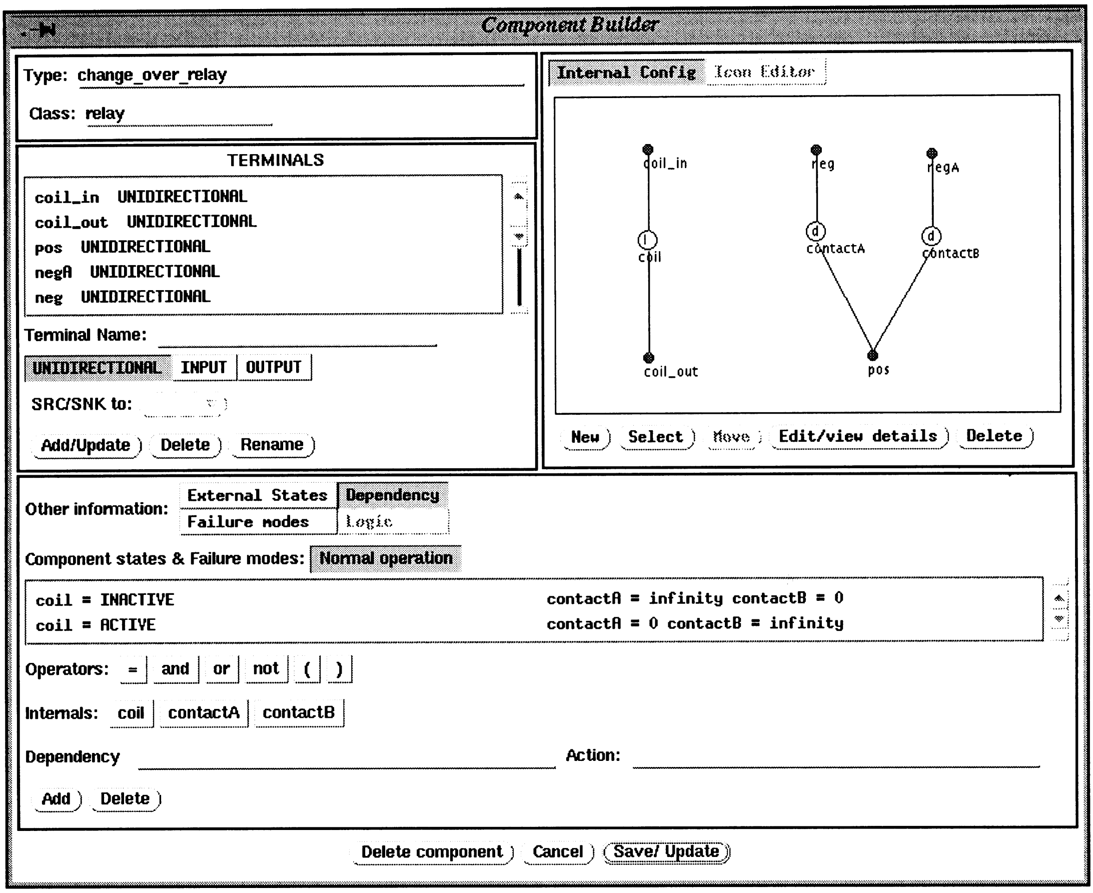
QCAT has a library containing descriptions of behaviour for common types of components. Where new components are needed for a circuit, the Component Builder provides the ability to specify the behaviour of components interactively, including failure mode behaviour, and to add the new component to the component library. The structure of a new component is defined in terms of resistance values and internal dependencies.

Figure 9 shows the component builder description of a change-over relay. In the top left of the figure, the component type, and component class have been defined. Below this is a description of each of the terminals associated with the component (the input and output options are not relevant to this example, they are used in association with ECUs). At the top right, the internal configuration of the component is defined in terms of terminals and qualitative resistance values. Here, the coil is defined as drawing a load and the switch contacts are defined as having dependent behaviour which is described at the bottom of the screen. The dependency expression indicates that when the coil is inactive, contactA has a resistive value of infinity (closed), and contactB has a resistive value of zero (open). The switch state is reversed when the coil is active. At the top right of the figure an icon editor can be invoked which allows the user to describe how the component will appear in the circuit drawing tool.

The component builder also allows the engineer to define failure modes for a component. Figure 10 shows the failure mode definition for relay switch fused to contactA. This failure mode is described by changing the qualitative resistance values for the component. In this case, the resistance for contactA is set to zero, representing the switch being closed, and contactB is set to infinity. Therefore, this failure mode would cause the component to allow current flow through contactA irrespective of the state of the coil.

4.3 Assessing significance

The Risk Priority Number (RPN) assigned to each failure effect is made up of three values: severity, detection and occurrence. The easiest of these to generate is occurrence. It can be calculated by



Type: change_over_relay

Class: relay

TERMINALS

```
coil_in   UNIDIRECTIONAL
coil_out  UNIDIRECTIONAL
pos  UNIDIRECTIONAL
negA  UNIDIRECTIONAL
neg  UNIDIRECTIONAL
```

Terminal Name:

SRC/SNK to:

Other information:

Component states & Failure modes: Normal operation

```
coil = INACTIVE                    contactA = infinity contactB = 0
coil = ACTIVE                      contactA = 0 contactB = infinity
```

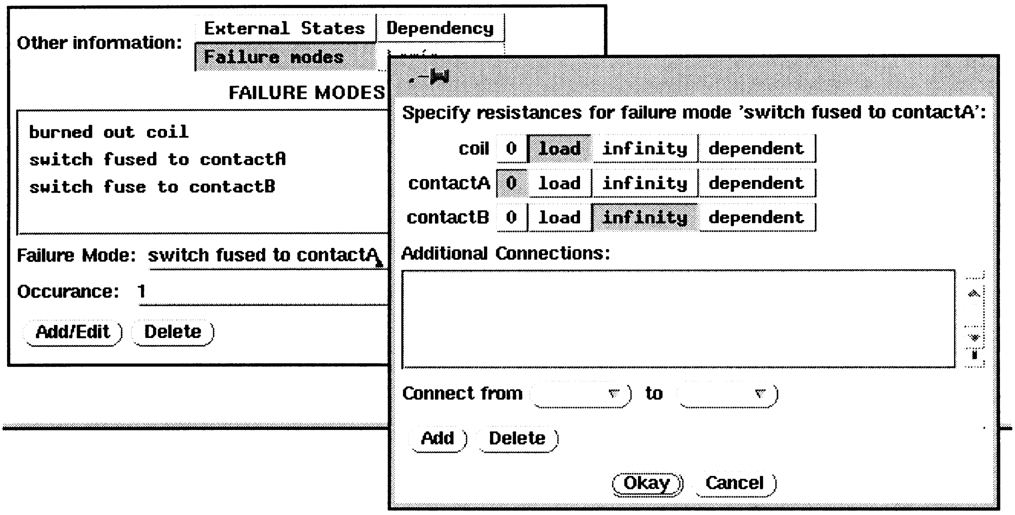
Operators:

Internals:

Dependency

Action:

Figure 9 Component builder description of a change over relay



Other information:

FAILURE MODES

```
burned out coil
switch fused to contactA
switch fuse to contactB
```

Failure Mode: switch fused to contactA

Occurance: 1

Specify resistances for failure mode 'switch fused to contactA':

coil	0	[✓ load]	infinity	dependent
contactA	[✓ 0]	load	infinity	dependent
contactB	0	load	[✓ infinity]	dependent

Additional Connections:

Connect from to

Figure 10 Failure mode description for change over relay

mapping component reliability figures onto the severity scale of 1 to 10 using the company's usual mapping. For example, an occurrence of 1 might mean that there is a 1:50,000 possibility of a relay failing by staying closed during the lifetime of the car. Wires give a slight complication, as several wires joined in series are considered as a single failure mode. Automatic rules can be applied here for combining wire reliabilities and producing a composite value.

Severity and detection values for single effects can be obtained by taking the relevant values from the function database—for example, if the dipped beam function is expected but absent then take the relevant values for that effect declared in the database.

Multiple values are more complex. For example, in a windscreen wiper system, achieving slow wipe when fast wipe is expected is not as bad as the wipers not working at all. Values for these more complex effects could be obtained by declaring entries for each more complex effect. FLAME takes a simpler possibility, leaving the entry empty and allowing the engineer to examine the effect and produce a value.

The FLAME system has a second way of producing severity and detection values, by reusing values assigned by the engineer in previous designs for the same functional system. This is only possible because of the consistency of the failure effects generated. This method is most effective for previous versions of the same design, but because the effects are related to function, not specific to a particular design for the circuit, it is possible to reuse values between different model years as well.

5 Evaluating the FLAME system

Section two stated a number of different criteria against which a system to automate the FMEA process could be judged. How does the FLAME system fare against these criteria?

- **Help for the whole FMEA report generation process**. The FLAME system is capable of generating effects for each failure mode in a readable form that can be automatically filled in to an FMEA report form. It also generates significance values and allows the engineer to annotate its results. After the interactive session with the engineer, the FLAME system is capable of printing a completed analysis.
- **Timely assistance**. The qualitative nature of the FLAME system enables FMEA to be performed early in the design process. However, the ability to perform an incremental FMEA (Price, 1996) means that the analysis can be repeated each time the design is changed, just showing the engineer the entries that have changed since the previous FMEA was run.
- **Reuse of information**. The circuit information is imported from the ECAD tool used by the engineers to design the circuit. This has been done for three different ECAD tools in the user companies. The functional information is generic to each subsystem (lighting, central locking, etc.), and so can be reused from model year to model year. Descriptions of common components are kept in a component library. All of this reuse makes model construction a fairly simple process.
- **Usable by engineers**. Where components do need to be built for a circuit, construction can be done pictorially, and tested by running the simulator and checking that the component works correctly.
- **Greatly improved efficiency**. The simple headlight circuit used as an example earlier in the paper is a fairly trivial circuit, and yet a conservative estimate is that it would take a week to produce an FMEA report on the circuit without the use of a tool such as the FLAME system (it has 92 failure modes). Using the FLAME system to perform FMEA on the lighting subsystem speeds up the generation of an FMEA report significantly. Model construction should only take a few minutes of the reliability engineer's time. The FLAME system takes about 10 minutes on a Sparc 5 to identify all failure modes, generate all failure effects, and assign RPN values for each failure effect. While this does take some time, the engineer need not be present, and could be carrying out some other duty, or just taking a coffee break. The major commitment of time comes during the interactive examination phase. For a circuit of the complexity of the example, this process might take a couple of hours. Thus it can be seen that the FLAME system reduces the task of producing this FMEA report from one which previously took a week to one which can be performed within half a day at the most. For more complex circuits, the time savings are even more significant.

There are other benefits in using the FLAME system which cannot be quantified as easily as the engineer's time.

- **Consistency**. The FLAME system always reports the same failure effect in the same way, and ascribes the same RPN values to it. This makes it a more objective report than is the case when it is produced by hand, where reporting of effects can be variable, and RPN values can depend on how an engineer is feeling.
- **Analysis of multiple failures**. No work has been carried out on generating FMEA reports for multiple failures, but the simulator is capable of working with multiple failures. It is clear that it would be possible to investigate the implications of all pairs of failures on a circuit (exploring all permutations of failures is not feasible because of the exponential effect of considering all fault combinations). The report could then be pruned to present only effects which did not occur for a single failure. This would be valuable information, and is not possible when performing FMEA "by hand": even for the headlight example, it would mean considering 8372 failure modes.
- **Changes to the way FMEA is performed**. The ability to perform incremental FMEA (considering only the changes caused by a change to the circuit) means that FMEA can be performed early in the design cycle and then repeated whenever the circuit is changed. For example, changing the lighting circuit so that the main beams and dipped beams are fused separately changes just eight of the entries in the FMEA report. Showing the engineer just the changes means that incremental FMEA takes up just a few minutes of engineer time. This may well cause the FLAME system to be used as a "what-if" tool, answering questions such as: *what are the reliability implications if this change to the design is made?*

The FLAME system has been in experimental use by Jaguar Cars and Ford Motor Company Ltd for more than a year. It can satisfactorily simulate about 85% of the circuitry of a modern car (complex analog circuits can have behaviour too dependent on quantitative values to be amenable to qualitative simulation), and produce useful FMEA results. We are currently reimplementing the FLAME system in C++, to improve the speed of simulation.

5.1 Scope and limitations of the FLAME system

FLAME is intended to be used early on in the design of electrical systems, where FMEA results can be fed back into the design to improve both the safety and reliability of a circuit. Qualitative reasoning is used to generate circuit behaviour, and this gives FLAME both its greatest strengths and weaknesses. Many of the strengths have already been discussed:

- FMEAs can be performed without knowing about quantitative information such as wire lengths, fuse ratings etc.
- The qualitative analysis is much quicker than a comparable quantitative analysis; for example, performed by Saber.
- The operation of very complex components can be simulated, including Electronic Control Units (ECUs).
- The analysis can identify where a short circuit may cause a fuse to blow, or a fire to occur.
- The technique scales up: it can generate reports for the largest subsystems in modern cars within an hour.

The weakness of this technique lies in the fact that some failure modes require a quantitative analysis to be performed to discover the effect on the circuit. An example of such a failure is where a contact on a switch is partially corroded. The immediate effect on the circuit is that less current flows through the switch than was intended by the designer. The overall effect on the circuit could be that a bulb does not shine as brightly as it should—an effect that FLAME cannot identify.

This weakness can be overcome by using a quantitative analysis for cases where a qualitative analysis is not sufficient; this could be done late in the design cycle when detailed information about the circuit is known. We are currently investigating how to best combine these two types of analyses. See Montgomery et al. (1996) for further details.

6 Application to other design techniques

Research into automation of design techniques other than FMEA is less advanced, but early experiments indicate that the use of modelling and simulation of electrical systems provides a potential basis for automatic production of results for other design techniques too. This section will briefly describe work done towards the automation of sneak circuit analysis and fault tree analysis.

6.1 Sneak circuit analysis

In complex electrical systems, the interaction of several subsystems can cause further systems to be activated unexpectedly. A good example is given in Savakoor et al. (1993) of the cargo bay doors of a particular aircraft design, where operating the emergency switch for the cargo doors can cause the landing gear to lower unintentionally. Typically, such problems are caused when a line which was expected to provide current in one direction is used in the opposite direction, causing a *sneak path*.

Sneak Circuit Analysis (SCA) is the process of identifying and eliminating such sneak paths where they might occur. If a line is enabling current flow in an unexpected direction, this can often be prevented by the addition of a diode to the design. Cost considerations mean that diodes should not be added to the circuit unless they are really needed.

The QCAT circuit analyser can be used to perform sneak circuit analysis between a number of car subsystems. Say that the engineer wanted to find out whether there were any sneaks between the lighting systems, the central locking system and the wash/wipe system for a car design. It can be done in the following way:

- Perform FMEA on each of the subsystems separately (if not already done). This provides much of the information needed to do SCA.
- Load the circuit composed of all of those subsystems (linked together to the battery and to any common grounds) into the circuit analyser.
- Load the links from circuit state to function for each separate subsystem.
- Identify the functions activated by each switch in each of the subsystems (from the correct behaviour description for each individual subsystem generated when doing the FMEA).
- Activate each combination of switches in the composite circuit. Use the function-to-circuit links to identify which functions are active in each state. Compare the active functions with the functions that are expected to be active (the ones that were active when those switches were closed in the individual subsystems).
- If any additional functions occur, then there is a significant sneak between the subsystems, and a report to that effect should be generated, giving the switches that were closed at the time, and the additional effect that happened. The engineer can then use the circuit simulator to study the effect in detail.

Experiments with several classic documented sneak problems indicate that this method is capable of finding all significant sneaks in a circuit (assuming that sneaks which do not activate any extra functions are not significant). This has only dealt with sneaks in correctly behaving circuits, but it should extend to faulty versions of circuits fairly simply.

6.2 Fault tree analysis

Fault Tree Analysis (FTA) is a method for identifying the reliability of a function (IEC, 1990). Unlike FMEA, which looks at all possible effects in a circuit, FTA identifies all of the combinations of failures that might cause one effect of interest to occur. It is driven by a single question. An example of such a question for the simple headlight circuit might be "what failures might cause no headlights at all (either dipped beam or main beam) to be available when requested?"

Such questions can be answered from the output of the FLAME system, if they are rephrased in terms of events and functions. The rephrased question would be: "What single failure modes or combinations of failure modes can cause the effect that when the dipped beam switch is turned on, the dipped beams do not work, and when the main beam switch is turned on, the main beams do not work?

The FTA software constructs a tree where one branch contains all single failures which can cause both sets of headlights to fail, and the other has an "AND" function combining the possibility that the dipped beams will fail and the possibility that the main beams will fail. Under each of the single branches, are put the failure modes that can cause the single effect.

Having constructed the tree, it is possible to generate cut sets and do the analysis in the normal way.

7 Related work

FLAME is the first FMEA tool to automate the production of effects descriptions and risk priority numbers for electrical system design. Previous research on automation of FMEA has concentrated on automating individual parts of the process:

- **Managing FMEA information**. At its simplest, computerised assistance for FMEA involves using spreadsheet like tools which allows the results to be presented and stored in a neat tabular form; an example of this is FMEAplus. Other tools in this category have been developed to ensure that results are consistent (Coker et al., 1989), automatically documented (Goddard & Davis, 1985), or produced in a more readable form (Legg, 1978; Lind, 1985). This work complements the work done on FLAME, and the latest version of FLAME can export its results to FMEAplus.
- **Automatic generation of effects**. There are two main strands to this work. The first is best illustrated by Lehtela (1990) on an FMEA tool which uses quantitative reasoning to generate effects descriptions. Each component has both normal and failure mode behaviours; however, there is no concept of the function that a system performs. Therefore, the results are described simply in terms of altered component behaviour—there is no description of the effect of the failure on the system as a whole. In addition, as the circuit size increases, the time a quantitative analysis would take to perform a complete FMEA on a circuit would be immense.

  The second strand of automatic effects generation is documented in Bell et al. (1992). They use a qualitative circuit analysis tool to generate the effect description—the model is described using a graphical representation, which is converted directly into a causal model of the circuit. This system does not contain the concept of the effect of the failure on the system as a whole; therefore, the results would not be as detailed as those found in FLAME. Processing times for this work would be very similar to those found in FLAME.
- **Manipulating significance values**. Kara-Zaitri et al. (1992) have shown that it is possible to generate significance values automatically based upon the probabilistic combinations of RPNs. Because there is no functional level description of system behaviour, results are generated from combinations of values at the component level. In FLAME, the severity and detection values are generated at the functional level. This is a more intuitive way to define these indices; for example, we can define the severity of "wash screen" failure without worrying about the component which causes this failure. It also makes the results more re-usable because the indices are not directly attached to specific components.
- **A task structure for an FMEA tool**. One of the most interesting papers on automating the FMEA process has been written by Russomanno et al. (1994). The paper contains summaries of several experimental effects generation systems, such as those mentioned above. It also proposes a richer architecture for building an automated FMEA system. As FMEA is often done by a team of specialists working together, it proposes an FMEA system based on a blackboard architecture. The main agents in this architecture are: the design expert, managing the FMEA process; the simulation expert, providing the modelling abilities of the system; the analysis generator, giving the report generation facilities; the mission expert, with knowledge of how changes in simulation affect the outside world; the discourse manager, providing an intelligent user interface. We would

disagree with the necessity for separate agents in an automated FMEA system, but the paper gives some insight into the different processes that take place when performing an FMEA, and how these processes can be related to various artificial intelligence techniques.

8 Conclusions

The FLAME system demonstrates that it is possible to automate much of the tedium of producing an FMEA report. This has been accomplished, not just on simple circuits such as the one given as a detailed example in this paper, but on actual automobile circuits imported from ECAD tools. The techniques discussed have been proven to work on a wide range of automobile circuits, and to scale up to systems with hundreds of components. The report produced by the automated FMEA is similar to those produced by engineers, with the added advantage of a greater degree of consistency in effect description and RPN generation.

The techniques used in producing an automated FMEA system also provide great leverage for automating two other design techniques, fault tree analysis and sneak circuit analysis. The research on automation of those techniques is at an earlier stage, but the final goal is to provide an automated design analysis system capable of applying a number of design analysis techniques.

Acknowledgements

This work has been supported by the UK Engineering and Physical Sciences Research Council (grant number GR/H96973), the Ford Motor Company Ltd, Jaguar Cars, the Motor Industry Research Association, and Integral Solutions Ltd, using the Poplog program development system.

References

Bell, D, Cox, L, Jackson, S and Schaefer, P, 1992. "Using causal reasoning for automated failure modes and effects analysis" *Proc. Annual Reliability and Maintainability Symposium* 343–353.

Coker, AO, Smith, JA, Higgins, S and Cameron, DC, 1989. "Computer-based failure mode and effects analysis for quality management—a case study" *Quality Assurance* **15** (3) 89–94.

Dijkstra, EW, 1959. "A note on two problems in connection with graphs" *Numerishe Math* **1** 269–271.

Goddard, PL and Davis, RW, 1985. "The automated, advanced matrix FMEA technique" *Proc. Annual Reliability and Maintainability Symposium* 77–81.

Kara-Zaitri, C, Keller, A and Fleming, P, 1992. "A smart failure mode and effect analysis package" *Proc. Annual Reliability and Maintainability Symposium* 414–421.

Lee, MH and Ormsby, ART, 1993. "Qualitative modelling of the effects of electrical circuit faults" *Artificial Intelligence in Engineering* **8** 293–300.

Legg, JM, 1978. "Computerized approach for matrix-form FMEA" *IEE Transactions on Reliability* **R-27** (4) October.

Lehtela, M, 1990. "Computer-aided failure mode and effects analysis of electronic circuits" *Microelectronics Reliability* **30** (4), 761–773.

Lind, JA, 1985. "Improved methods for computerized FMEA" *Proc. Annual Reliability and Maintainability Symposium* 213–216.

Montgomery, T, Pugh, DR, Leedham, S and Twitchett, S, 1996. "FMEA automation for the complete design process" *Proc. Annual Reliability and Maintainability Symposium.*

Palumbo, D, 1994. "Automating failure modes and effects analysis" *Proc. Annual Reliability and Maintainability Symposium* 304–309.

Price, CJ, 1996. "Effortless incremental design FMEA" *Proc. Annual Reliability and Maintainability Symposium.*

Price, CJ and Hunt, JE, 1991. "Automating FMEA through multiple models" In: *Research and Development in Expert Systems VIII* 25–39, Cambridge University Press.

Price, CJ, Hunt, JE, Lee, MH and Ormsby, ART, 1992. "A model-based approach to the automation of failure mode effects analysis for design" *Proc. IMechE, Part D: the Journal of Automobile Engineering* **206** 285–291.

Price, CJ, Pugh, DR, Wilson, MS and Snooke, N, 1995. "The FLAME system: automating electrical failure modes and effects analysis (FMEA)" *Proc. Annual Reliability and Maintainability Symposium* 90–95, IEEE.

Russomanno, DJ, Bonnell, RD and Bowles, JB, 1994. “Viewing computer-aided failure mode and effects analysis from an artificial intelligence perspective” *Integrated Computer-Aided Design* **1** (3) 209–228.

Savakoor, DS, Bowles, JB and Bonnell, RD, 1993. “Combining sneak circuit analysis and failure modes and effects analysis” *Proc. Annual Reliability and Maintainability Symposium* 199–205, IEEE.

IEC 812 (1995) *Analysis Techniques for System Reliability—Procedure for failure mode and effects analysis (FMEA).*

IEC 1025 (1990) *Fault Tree Analysis.*