Interpreting Simulation with Functional Labels

Chris Price and David Pugh

Department of Computer Science
University of Wales, Aberystwyth
Dyfed, SY23 3DB, United Kingdom
cjp@aber.ac.uk

Abstract

Functional labels provide a simple but very reusable way for defining the functionality of a system and for making use of that knowledge. Unlike more complex functional representation schemes, these labels can be efficiently linked to a behavioral simulator to interpret the simulation in a way that is meaningful to the user. They are also simple to specify, and highly reusable with different behavioral implementations of the system's functions. The combination of functional labels and behavioral simulator can be employed for a variety of tasks – simulation, failure mode effects analysis (FMEA), sneak circuit analysis, design verification – producing results that are very valuable to engineers and presented in terms that are easily understood by them. The utility of functional labels is illustrated in this paper for the domain of car electrical systems, with links to a qualitative circuit simulator. In this domain, functional labels provide a powerful way of interpreting the behavior of the circuit simulator in terms an engineer can understand. This claim has been substantiated by the development of the FLAME application, a practical automated FMEA assistant in regular use at several automotive manufacturers.

Introduction

Qualitative reasoning from the structure of a device or system has been a promising technology for many years (see papers in Weld and de Kleer 1990), but very few successful industrial applications of the technology have emerged during that time. One of the main problems has been interpretation of the results of qualitative simulation at a level which is relevant to potential users. Several researchers have used a representation of overall system behavior, sometimes referred to as function, to interpret the simulation results (Sasajima et al. 1995), (Iwasaki et al. 1995), (Hunt et al. 1993). Such schemes tend to be overly complex for use in practical applications.

This paper describes a deceptively simple scheme which has proven very effective in the domain of car electrical systems. Functional labels are employed to guide the use of component-based qualitative simulation. Instead of specifying the overall behavior of a device or system, the significant functions of the overall behavior are identified and named. These functional labels can then be linked to the state of one or more specific components in the device, in order to identify when a function is being achieved.

The linking of functional labels to qualitative simulation has been used to provide practical automated assistance for design analysis to engineers in industry. The paper gives examples of tools in use in industry for automotive design analysis. Finally, the paper considers whether functional labels are more generally applicable than the specific scheme described in detail in the paper.

Functional Labels

Functional labels identify the important states of a system or device. These are likely to be linked to user goals: either states that the user might wish to attain or states that the user wishes to avoid.

For example, in a car's central locking system, All doors locked and All doors open might be functional labels for two of the significant states. In a car's wash/wipe system, relevant functional labels might be: Screen wash, Slow wipe, Intermittent wipe and Fast wipe.

The significant overall behavior of many systems can be characterized by a few such labels. This is much simpler than other methods for defining function. There are no explicit links declared between the different functional states that the system can be in, such as those needed by the FR language (Sembugamoorthy and Chandrasekaran 1986) or by CFRL (Iwasaki et al. 1995).

It is also much more reusable. Where more complex functional representations commit themselves to explicit links to structure, they become much less reusable in future implementations of the same system. There is no such commitment in the functional labels themselves, only in the links that must be declared between functions and key component states for each version of system structure.

Despite the simplicity of the functional labelling method, combining the results of a qualitative simulation of a system with functional labels enables interpretation of the behavior of the system in terms of the overall goals of that system. The next section will illustrate this for a simple headlight system.

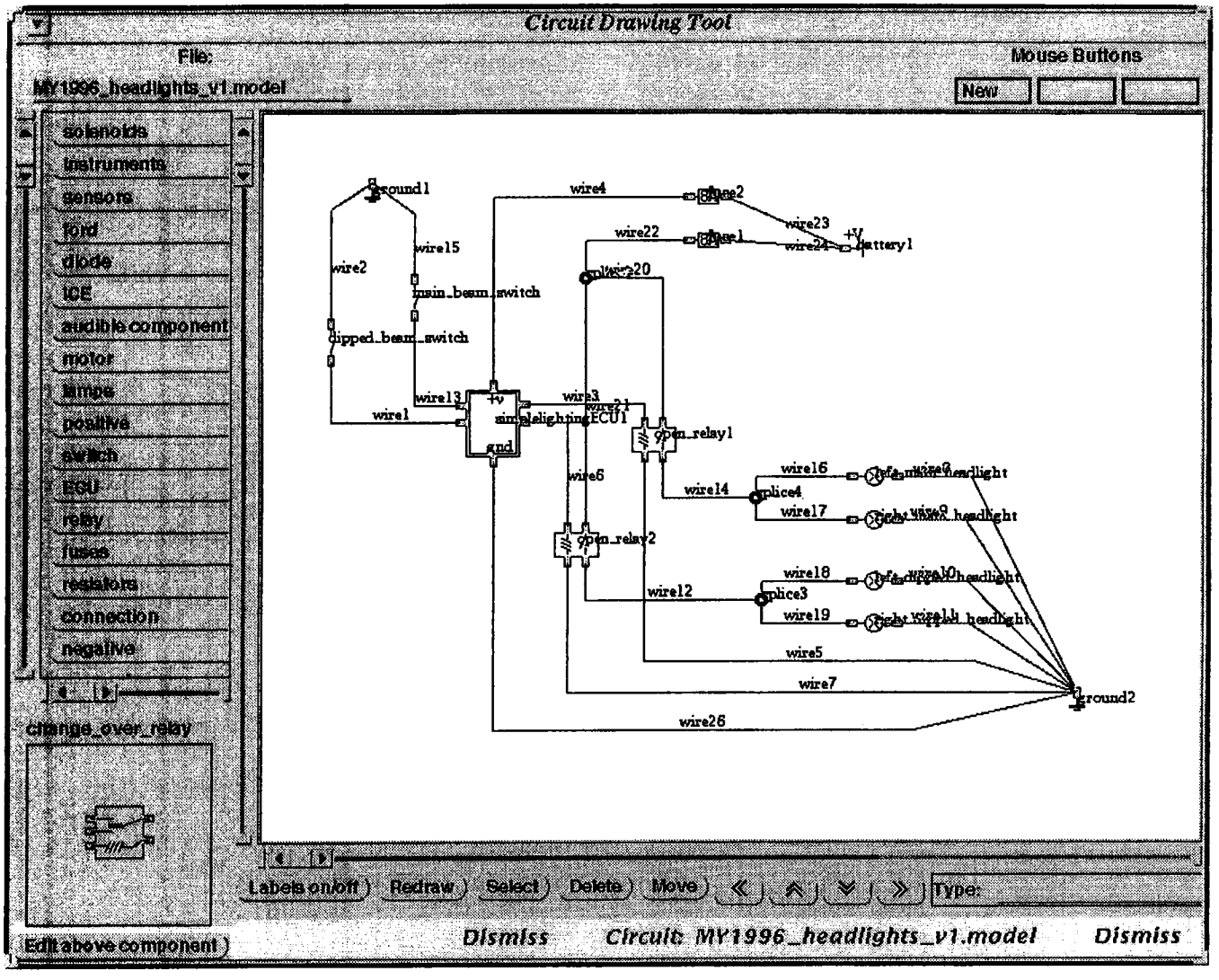

Figure 1: Simple headlight circuit

Qualitative Simulation of Electrical Circuits

Figure 1 shows a simple headlight system, where the driver can select either of two sets of headlights (main or dipped beam). This circuit has been input using the circuit drawing tool as a simple illustration – more complex electrical designs would be imported directly from the electrical CAD tools used by automotive engineers.

The behavior of the circuit can be simulated by the QCAT qualitative circuit simulator (Pugh and Snooke 1996). With knowledge of possible outside changes (switches, sensors), this can be used to map all possible states for the circuit. Each state in the envisionment has details of which components are active in that state. Where there are hundreds of components in a circuit, this is too much information for an engineer to make sense of it. Even for the simple headlight example, the results run to several pages of information.

The QCAT 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.

The network analyser, based extensively on CIRQ (Lee and 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 2 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 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.

The example used to describe the CIRQ algorithm contained only simple components; that is, components which could be represented as a simple resistance value.

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 resistance 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 resistance to infinity and re-simulate.

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

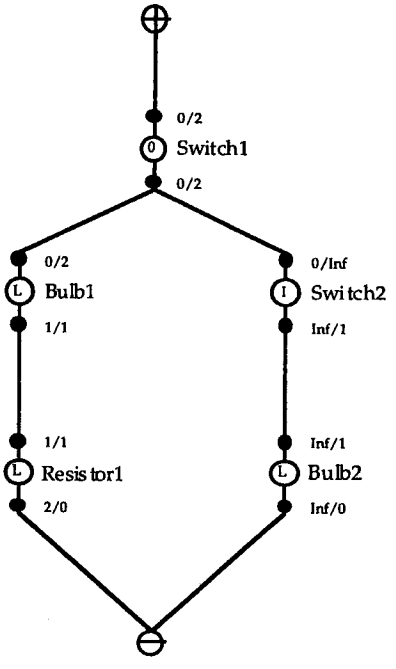

Figure 2: Resistance graph for simple circuit

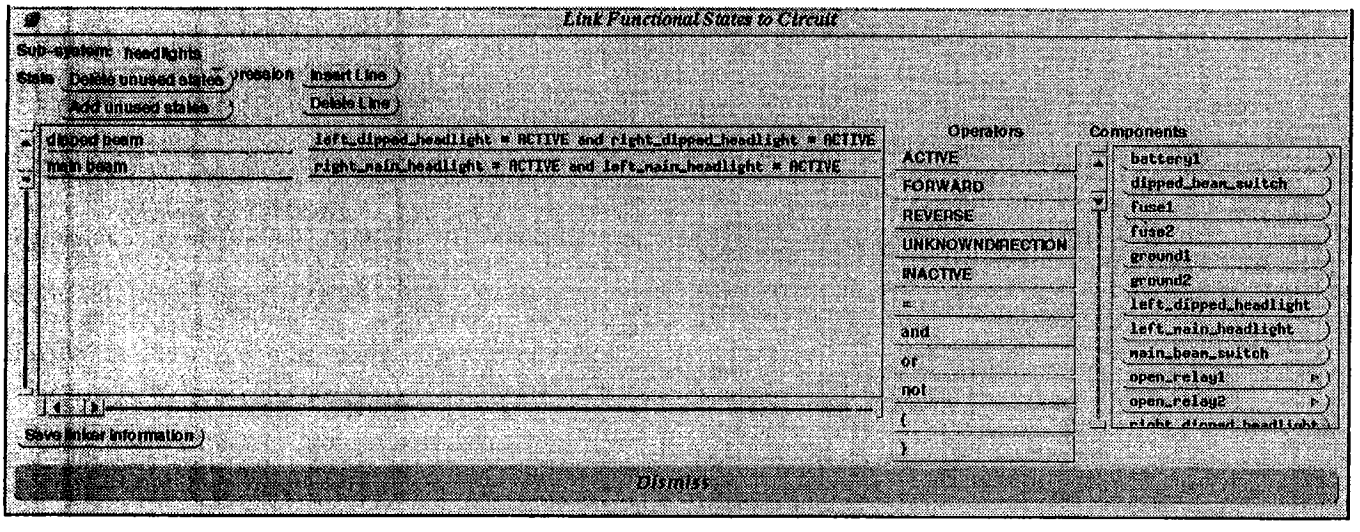

Figure 3: Linking function to behavioral states

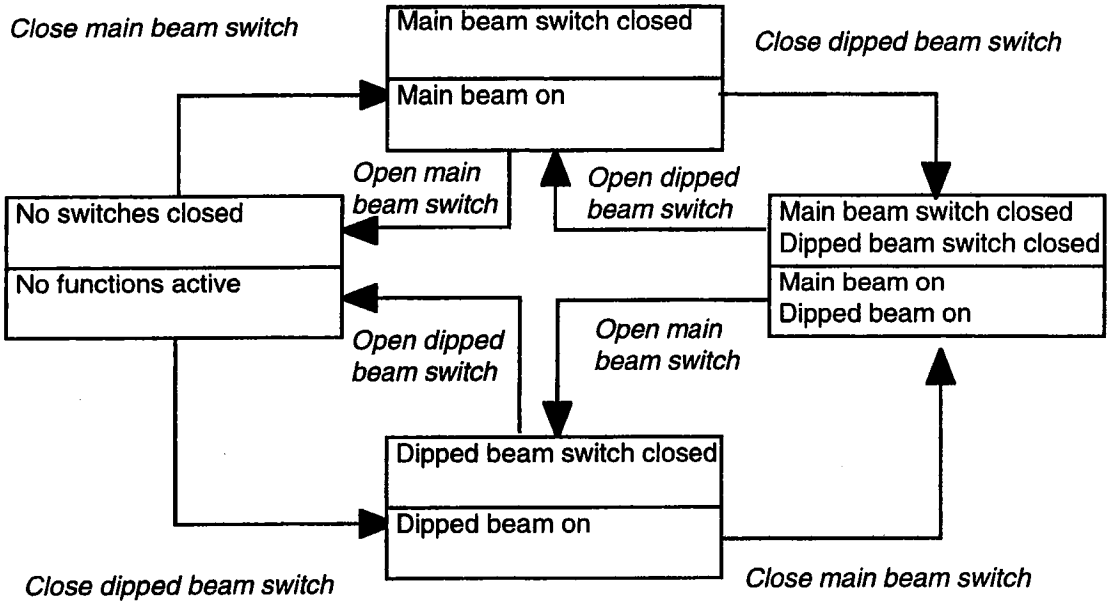

Figure 4: State summary produced by using functional labels

Interpreting behavior with functional labels

Functional labels can be used to simplify and to interpret the behaviour of the qualitative simulation.

Functions of the headlight system are: Main beam and Dipped beam. The presence of each function can be identified from the states of key components. The user declares links between function and system state using a screen such as the one in figure 3. The link can be a complex expression combining the state of several components. For example, the Main Beam function is happening when left_main_headlight = ACTIVE and right_main_headlight = ACTIVE.

A functional interpretation of each state in the qualitative simulation can then be automatically obtained by matching each possible function against the state of the relevant components. For the headlight system, a state graph like that shown in figure 4 can be generated, summarising possible states for the headlight system and how states can be attained. This is a dramatic reduction in the amount of information produced by the simulation, and the information that is left is focused on problem-solving in the domain. For a more complex circuit, the reduction from state information produced by the behavioral simulation to functional state information is even more significant.

One of the major advantages of functional labelling over explicit declaration of the relationship between different states is the high level of reusability that can be obtained. A simple example of this advantage can be shown by a change to the structure of the circuit. If an extra switch was added to the circuit in series with one of the other switches, or if both sets of headlights were operated by a single switch, then the new state graph for the system can be generated automatically, without even changing the links between function and structure. This is because the links to the functional level only involve the components that are involved in interpreting the behaviour: all generation of behaviour is done by qualitative circuit simulation. Under previous functional representations, the functional representation would need to be changed.

Automatic generation of state graphs summarising function from the results of a qualitative simulation is even more significant for reasoning about component failures for diagnosis or FMEA. It becomes possible to make a change to the structure of the system (to change the circuit, in the case of electrical systems) and then to determine which functions of the system will be affected.

For example, assume the headlight circuit contained a faulty relay which stuck open, connected to the main headlight beams. The qualitative simulation could be run for a version of the circuit containing a faulty relay, and the state graph shown in figure 5 would be automatically generated for the headlight circuit. By comparing this state graph with the one for the correctly working system, it is possible to infer that the effect of the faulty relay is that the function "main beam on" will not be achieved.

This kind of reasoning is impossible for functional systems such as FR (Sembugamoorthy and Chandrasekaran 1986), where the relationship between states is explicitly defined. In order for those kinds of systems to reason about faults, it is necessary to link the achievement of functions to assumptions about the correct working of particular sets of components. If that is done, then the functional representation becomes even less reusable for different designs of the same system. Systems such as FR are also unable to detect the unexpected achievement of a function due to a fault – an easy task for functional labels linked to structural simulation.

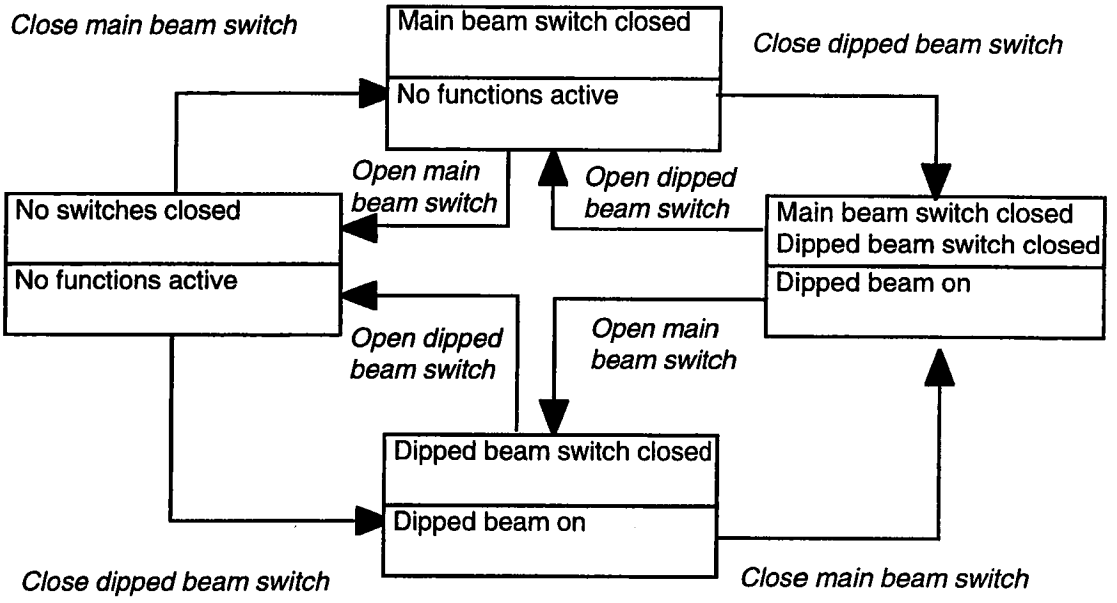

Figure 5: State summary with faulty relay to main beams

The use of functional labels along with structure has advantages over reasoning only at a structural level. As with other kinds of functional reasoning, the behavior of the system is expressed in terms of the goals of the system. This means that significant behaviour can be identified and meaningful design analysis reports can be produced.

The next section shows how this works out in practice for performing failure mode effects analysis of electrical circuit designs.

Functional reasoning for FMEA

Failure modes and effects analysis (Jordan 1972) is used extensively in the aerospace and automotive industry to verify the safety of proposed electrical and mechanical designs. There are a number of steps an engineer needs to go through in order to perform an electrical FMEA:

1. For all components in the system, identify the ways in which they can fail. These are the so called failure modes. For example, the main failure modes of a wire are short to ground, short to battery, and open circuit.

2. For each of the component failure modes, identify the effect that failure has on the system (the failure effect). For example, a wire shorting to battery may cause the headlights to stay on, when they should be off.

3. For each of the failure effects, rankings within the three classes of severity, detectability, and likelihood of occurrence need to be assigned. For example, brake failure would have a severity of 10, on a scale of 1 to 10, as this may cause death, serious injury, or damage to the car. Once values are defined for the three classes, the values are usually multiplied to generate an risk priority number (RPN). If any of the three indices have high values, or the overall RPN is greater than a certain pre-set value (often 100), then ways of improving the design will be sought.

The FLAME application automates the whole of the process described above. Having designed a circuit using a CAD tool, the engineer imports it into FLAME. Definitions of the behavior of standard components such as wires, relays and lamps already exist. The internal behavior of components such as the CPU must be described. The engineer can then run simulations of the circuit to ensure that its correct behavior is as expected. The functions for the circuit are taken from a database of functions for each subsystem of the car, and linked to the circuit as shown in figure 2.

Possible failure modes for every component in the structure are known (for example, relays can stick open or stick closed). The behavior of the circuit is simulated for each possible failure mode, and compared with the expected behavior. Functional labels will be attached to each state, identifying both expected functions that fail to occur and unexpected functions that occur unexpectedly. RPNs can be generated by assigning severity and detection values to each functional label, and combining this with a likelihood value which depends on structural complexity and component type.

This method, explored in greater detail in (Pugh et al. 1995), has enabled the automatic production of FMEA reports of a similar quality to those produced by engineers, only much more quickly and with a much greater degree of consistency. An example output page is shown in figure 6.

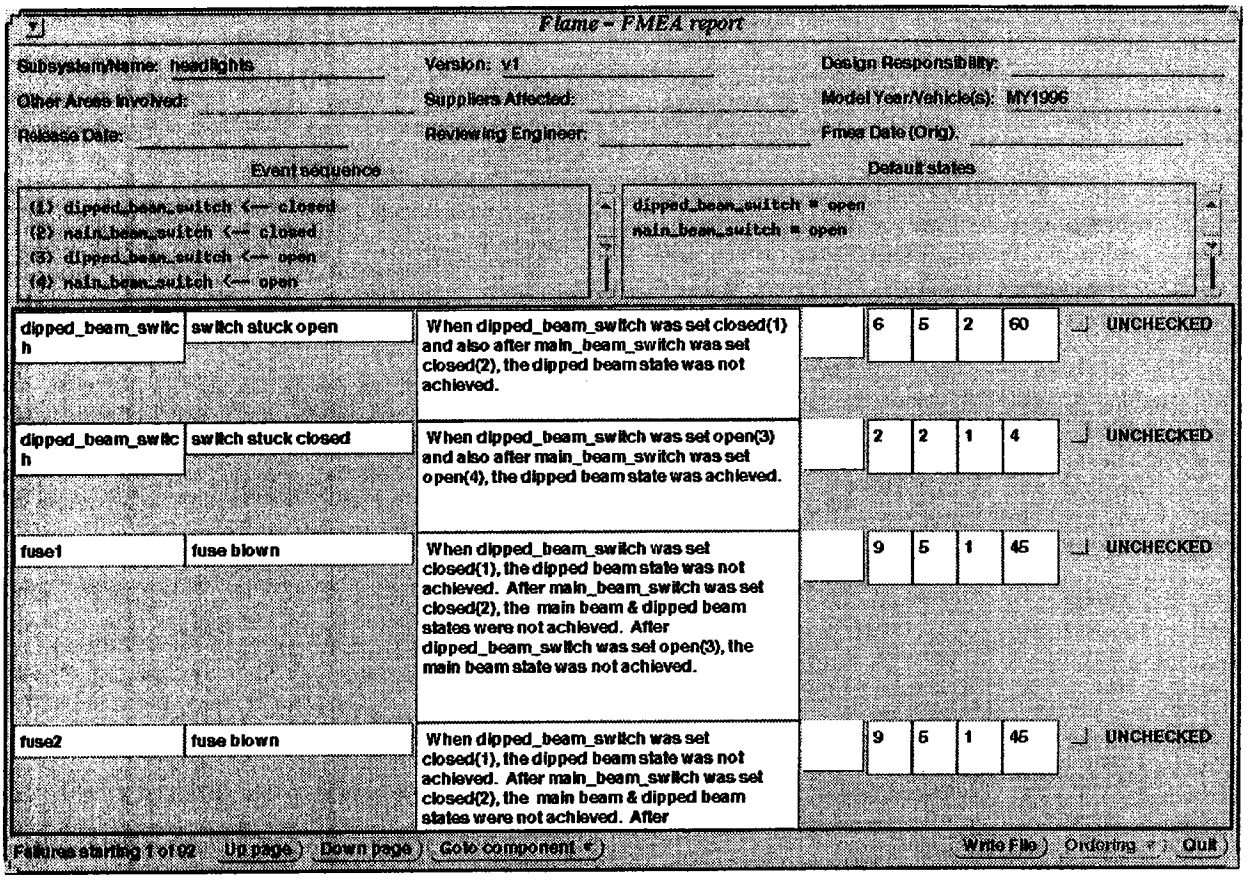

Figure 6: FMEA report generated by FLAME

As can be seen, the simple headlight circuit has 92 possible failures that must be examined. In more complex circuits, many hundreds of failures need to be examined, and manual generation of an FMEA report can take months of engineer's effort.

The simplicity of the functional label idea linked with use of the standard structural representation for the domain means that FLAME is used in industry by automotive engineers with no help from computer scientists – not the case for the more complex functional systems. It has been tried out by engineers on a complete car electrical design, and can deal adequately with almost all of the circuits.

Execution time for generating a complete FMEA report with FLAME varies from a few minutes for the circuit given as an example, to several hours for the most complex subsystem in a car, containing hundreds of components. Execution is $O(n^2)$ with number of components.

The consistency of the produced output gives benefits that had not been anticipated when building the system. Because the FMEA report is given in terms of reusable functions, incremental FMEA becomes possible: when the design of a circuit is changed, a new FMEA report can be generated, with the engineer being shown only the FMEA results that differ from the report for the previous version of the circuit. This means that a discipline which previously took several weeks of an engineer's time has been turned into an exercise which can be performed on a "what if" basis to see if a change in design produces a circuit with improved reliability.

Use for other applications

The combination of structure-based simulation and functional labels can also perform several other design analysis tasks effectively.

Sneak circuit analysis. This is the process of identifying and eliminating unexpected interactions between subsystems. A good example problem is given in (Savakoor et al. 1993) for the cargo bay doors of an aircraft, where operating the emergency switch for the cargo doors can cause the landing gear to lower unintentionally. Typically, the problem is caused when a line which was expected to provide current in one direction is used in the opposite direction, causing a sneak path. FLAME can identify sneak paths automatically. The user must declare which inputs

(switches or sensors) cause which functions to occur. All combinations of inputs are then tried, and if at any point a function is activated that should not be, then a sneak path has been detected. This strategy correctly identifies classic example sneaks such as the cargo door sneak.

Design verification. Engineers use techniques such as Statecharts (Harel et al. 1987) to handcraft a state-based specification of the operation of a system very similar to that shown in figure 3. This is used to produce a requirements definition for the system being designed. The Statechart requirement specification could be compared with the envisionment generated from qualitative simulation and functional labels, in order to identify unexpected or missing states. We are currently investigating the generation of proposed changes to structure when there are discrepancies between the two, but automated identification of discrepancies is a useful first step.

Diagnostic systems. The combination of structural simulation and functional labels is capable of generating candidate faults with the right level of symptom (functional level). The simulator can deal with multiple faults, and so there is no single fault limitation. It has been used to generate fault trees (an AND/OR tree of all combinations that can cause a particular symptom such as "Main beam fails to work").

Wider Applicability of Functional Labels

This paper has shown that functional labels linked with a qualitative circuit simulator are of great utility for building practical applications. They seem to show promise for use with other types of simulator and in other domains.

Work has been done by our collaborators at Ford Research Centre, Dearborn into using a quantitative circuit simulator for generating the effects of failures (Montgomery et al.). The same functional labels can be used to interpret the meaning of the quantitative simulation. The links are more complex, involving ranges of values as well as whether components are active, but they enable the extraction of meaningful information from a complex simulation. Even better, they can be used to link the quantitative results to situations where the qualitative results are ambiguous.

Functional labels should also be useful in other domains. The main requirement is the ability to perform a qualitative component-centred simulation and to assign function to significant components in a system. Hydraulic or pneumatic systems are an obvious area for further application, and ones where FMEA is carried out. Mechanical systems would be more difficult, because of the complex mapping between functions and components.

Conclusions

Qualitative simulation from structure can be used to generate behaviour, but practical applications of the technology have been limited because of the problem of interpreting the results of the simulation. Functional labels give a simple and effective way of highlighting significant detail in the results of a good qualitative simulator.

The use of functional labels has several advantages over more complex schemes for linking functionality to structure:

Simplicity. More complex functional schemes can be difficult for engineers to specify. Functional labels require only that the main purposes of the device are identified, and that the component structure for qualitative simulation can be imported from the design for the device.

Reusability. Where a functional representation specifies how a function is achieved (Sasajima et al. 1995), (Iwasaki et al. 1995), (Hunt et al. 1993) it is tied to a particular implementation of the function. Functional labels only specify what is done, not how, and so are much more reusable.

Capability. More complex specifications of functionality cannot identify the unexpected achievement of functions or decide on functionality under fault conditions. Functional labels identify functionality from the results of the structural simulator, and so can detect unexpected achievement of functions and deal with fault conditions.

These advantages have been illustrated in FLAME, an FMEA generation system used directly by engineers and able to produce quality FMEA reports efficiently and accurately.

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

Dijkstra, E. W. 1959. A Note on Two Problems in Connection with Graphs. *Numerishe Math.*, vol. 1, 269-271.

Hunt, J.; Price, C.; and Lee, M. 1993. Automating the FMEA Process. *Intelligent Systems Engineering* 2(2), 119-132.

Iwasaki, Y.; Vescovi, M.; Fikes, R.; and Chandrasekaran, B. 1995. Causal Functional Representation Language with Behaviour-based Semantics. *Applied Artificial Intelligence* 9(1), 5-31.

Jordan, W. 1972. Failure Modes, Effects and Criticality Analyses. In Proceedings of Annual Reliability and Maintainability Symposium, IEEE Press, 30-37.

Lee, M. and Ormsby, A. 1993. Qualitative Modelling of the Effects of Electrical Circuit Faults. *Artificial Intelligence in Engineering* vol. 8, 293-300.

Pugh, D.; Price, C.; and Snooke, N. 1995. Practical Applications of Multiple Models - the need for simplicity and reusability. In *Applications and Innovations in Expert Systems III*, 277-291.

Pugh, D. and Snooke, N. 1996. Dynamic Analysis of Qualitative Circuits. In Proceedings of Annual Reliability and Maintainability Symposium, 37-42, IEEE Press.

Sasajima, M.; Kitamura, Y.; Ikeda, M.; and Mizoguchi, R. 1995. FBRL: A Function and Behavior Representation Language. In Proceedings IJCAI-95, 1830-1836.

Savakoor, D. S.; Bowles, J. B.; and Bonnell, R. D. 1993. Combining sneak circuit analysis and failure modes and effects analysis, In Proceedings of Annual Reliability and Maintainability Symposium, 199-205, IEEE Press.

Sembugamoorthy, V. and Chandrasekaran, B. 1986. Functional Representation of Devices and Compilation of Diagnostic Problem-solving Systems. *In Experience, Memory and Reasoning*. eds. Kolodner and Riesbeck, 47-73, Lawrence Erlbaum.

Weld, D., and de Kleer, J. eds. 1990. *Readings in Qualitative Reasoning about the Physical World*. Morgan Kaufmann.

Harel, D.; Pnueli, A.; Schmidt, J. P.; and Sherman, R. 1987. On the formal semantics of Statecharts, In Proceedings 2nd IEEE Symposium on Logic in Computer Science, 54-64.

Montgomery, T.; Pugh, D.; Leedham, S.; and Twitchett, S. 1996. FMEA Automation for the Complete Design Process. In Proceedings of Annual Reliability and Maintainability Symposium, 30-36, IEEE Press.