Temporal aspects of functional modelling for design analysis

Jon Bell

October 8, 2003

Abstract

Qualitative simulation interpreted by the use of functional labels to identify significant system behaviour has been shown to be a useful technique for automating design analysis tasks such a Failure Modes and Effects Analysis. However, increasing sophistication of the systems to be analysed means that it is becoming increasingly important that the correct timing of these system functions be included in the analysis but the existing functional labels do not capture this information. This paper discusses difficulties arising from this lack and the problems of modelling timing of achievement of functions. It proposes a simple approach to modelling temporal aspects of system function that allows both non achievement and unexpected achievement of a system function to be detected in design analysis as well as showing situations when a function is achieved later (or earlier) than it should be.

1 Introduction

Automating design analysis by generating a description of the system behaviour and interpreting it in terms of 'functional labels' that identify the significant states of the system has been shown to be a useful approach, (Price 1998). This approach is in use in the automotive sector for the automation of design analysis, principally Failure Modes and Effects Analysis (FMEA) and Sneak Circuit Analysis (SCA). The increasing complexity of modern systems in the automotive sector and the increasing reliance on their correct behaviour for correct operation of the vehicle leads to a need for more detailed analysis of these system, especially in the area of ensuring that systems achieve their required functions in a timely manner. At present, functional labels have not been implemented in such a way as to allow the recognition that a required system function has been achieved late, for example, and this paper suggests that a simple extension of the technique is sufficient to allow this to be done.

As described in (Price 1998), functional labels are used to interpret the results of a behavioural simulation of a system in terms of the states of significant components. The functional labels identify these significant states or, in practice, system outputs. As they are not linked to specific inputs, these functional labels are simple and reusable from system to system. Their identification of significant behaviour provides a link between the system's behaviour and its purpose. As a function can fail either by not being achieved when it should be or by being achieved when it shouldn't, the fact that functional labels have the advantage over more complex representations of function of being better able to capture unexpected achievement of the function is of value.

In FMEA the effects of known component failure modes on the whole system are investigated (Jordan 1972). In this case, the simulated behaviour of the correctly functioning system is compared with that of the system with a component failure. The functional labels are used to identify significant differences between these behaviours. In Sneak Circuit Analysis (Price, Snooke, and Landry 1996) the system's behaviour is simulated to ensure that system functions are achieved only when expected, not by some unexpected combination of switch positions

leading to an output. The ability of functional labels to identify unexpected achievement of a function is valuable here, of course. For SCA it is, of course, necessary to associate outputs with inputs in drawing up the functional model so that finding outputs associated with different inputs identifies possible sneak circuits.

A full description of function might contain the necessary inputs, the necessary outputs and some description of internal system state, to show how the outputs are to be achieved. How many of these need specifying in a particular case might depend on the design analysis task (as outlined above) and the nature of the system. For example, a system that incorporates fault mitigation might require the modeller to differentiate between the normal and fault mitigation functions, even though both use the same inputs and outputs. It seems possible that such elaborate modelling of function might require more complexity in the full description of the temporal constraints.

The paper is arranged such that the reasons for adding timing of functions to the analysis are introduced in Section 2, followed by a discussion of the difficulties associated with adding this information. A simple approach to adding temporal aspects of function is introduced in Section 5 and the usefulness of this approach is discussed.

2 The significance of timing

While functional labels have been shown to be a useful method of interpreting the results of a behavioural simulation, they have not specified any temporal aspects that might be required in correctly achieving a system function. There are two aspects to this where output is concerned. Firstly, there is the simple case that it might be desirable to capture the requirement that a function be achieved at a specified time relative to the trigger (typically a user input) that results in the function's achievement. This will typically (but not necessarily) be a requirement that the function is achieved within a certain time. This suggests a requirement to be able to distinguish situations where a function is achieved late (or early) and to do so distinctly from those cases where the function is not achieved at all.

The other, more complex, case is when achievement of a required system function depends on behaviour whose correct description depends on capturing information about timing. This is particulary important where there is a sequence of required outputs, such as a warning buzzer sounding intermittently for a known (required) period of time. Another point in this case is the possibility that the behaviour will have finished once the simulation has completed. If the simulation is run until the system settles into a steady state, then any intermittent behaviour will inevitably be completed once a steady state is reached. Of course, it may be the case that such intermittent behaviour may not complete during a simulation step, as it is intended to be stopped by further external input, which will wait until the next simulation step. A car's direction indicators are a case in point. It will be appreciated that the behavioural simulator should detect these cases and mark them accordingly. It can be suggested that the description of such intermittent behaviour might require some specification of timing simply because it must be ascertained that the output lasts a sensible time. For example, it might be desired to describe the flashing of a car's direction indicators in such a way as to specify that the timing of the individual flashes is within legal requirements for them. There is little point in describing an intermittent output without specifying that the output is useful. This is perhaps more important for design verification than FMEA.

Another aspect of the modelling of functions that depend on transient behaviour is the need to recognise the correct termination of the behaviour. It is not sufficient to describe flashing direction indicators, for example, purely in terms of the requirement that both the front and rear indicators being lit, and then not both being lit, as in this case if one stays on, the both lit function will no longer be achieved. This is not the intended behaviour, of course — it is

necessary to specify that both be unlit so that this failure is recognised. Another point worth raising here is that in this case the flashing of each indicator should be simultaneous. Whether this requirement needs modelling might depend on the design analysis task being undertaken, and for FMEA whether there is any failure mode that could result in the two indicators becoming "out of sync". There is some further discussion of this point in section 4.

Another possibility is that the timing of successive inputs might be significant. An example is the temporary over-ride feature in belt minder, where if the driver's seat belt in buckled and unbuckled within a certain time, this temporarily disables the system. As the simulation steps are typically run following one input even, this requires some means of temporally relating the input events as well as some way of relating the inputs in the specification of the input for the functional model.

3 Difficulties capturing timing of function

This section introduces some problem areas and questions that arise with regard to capturing the information necessary to model untimely achievement of system functions.

One feature of functional labels, as used for FMEA, is that they are not expressly linked to the required input. This means that if the function's output is achieved with different input from that expected (that is, the input in the simulation with no failures), the function is recognised as being achieved unexpectedly. To preserve this simplification of functional modelling, it is therefore advantageous if the timing constraints on a function are not explicitly tied to a specific input event. If they are then the unexpected achievement of a system function is lost as the loss of the input means the function is no longer achieved. There are perhaps interesting questions about the correctness of defining a function purely in terms of its output, and it will be appreciated that when a system behaviour is compared with the designer's intended behaviour, rather than (assumed) correct behaviour (as is the case for design verification and sneak circuit analysis) then the required system level input needs to be specified.

This in turn might lead to difficulties. It is simple specifying the required input to the system if it can be expressed in terms of switch positions, but problems arise if the input uses a component with state, such as a toggle switch. Such a switch might be used as the dip switch in a car's headlamp circuit, for example, and the effect of its being pulled and released (being used both for dipping the lamps and returning to main beam) will depend not only on the actions of the switch but also on the previous state of the system as a whole. One possible approach to this problem is to use a state transition diagram to describe the required inputs and their effects, rather than a simple mapping between switch position and expected output. There are cases where this problem can be sidestepped by modelling the internal change to a component in terms of its input. A toggle switch's actions could be 'release', 'pull on' and 'pull off', where pull on closes the switch and pull off opens it, so pull on changes the switch resistance to zero, for example. This raises the slight difficulty that at any given time only one of the three actions are available to the user, and arguably this constraint needs modelling. Of course this option is not available if the switch is actually a passing contact switch and the toggling between states is managed by another component so there are cases where this is a real difficulty.

One possible approach to capturing temporal aspects of system function for FMEA, where the analysis is in terms of correct system behaviour, would be to have the reasoning system detect significant deviations in timing between the correct, no failures, behaviour of the system and the failure mode behaviour. The problem with this approach is deciding what constitutes such a significant deviation. It could, of course, be specified by the model builder (the designer of the subject system) but this seems less intuitive than having the user specify the required behaviour in terms of deadlines by which the function should be achieved, or before which the function should not be achieved in the relatively rare cases where a delay is required.

This leads on the the question of how time is to be represented. In view of the qualitative nature of the circuit analyser used (Lee and Ormsby 1994), it would seem natural to attempt to use a qualitative notion of time. There are two possible approaches here and both seem to raise problems. One approach would be to use a notion of time based on ordering, so it would be possible to specify that, say, a buzzer should not start until after the warning lamp has lit. This does, of course, fail to capture any requirement that there should be a delay between these two events. However, this approach might conceivably be made to work for FMEA if the sequence of events linking the input event (switch being thrown, for example) and the achievement of function were compared and any differences flagged, especially if they resulted in a change in timing of the function's achievement.

The other approach to qualitative modelling of time is to use an order of magnitude model, in which all events in a given time slot (such as millisecond) are carried out before any events in the next longer time slot (such as seconds). This is consistent with the orders of magnitude in (Raiman 1991) and is already available for simulation in the design analysis tool developed at Aberystwyth. The problem with this approach is the simple one that the likely orders of magnitude are so far apart that expressing the timing of function is insufficiently sensitive. If a function should be achieved in a few milliseconds, say, then it will be regarded as being achieved late well before its timing crosses the threshold to the seconds time slot. However, it is possible that such a model of time would be sufficient to detect cases where a function is caused by some other behaviour whose timing is very different. As such a behaviour is likely to depend on several input events (such as a user imitating flashing by repeatedly throwing the switch) it seems possible that these behaviours will not be discovered except by specifying a new sequence of inputs.

4 The functional labelling language

If we are to relate several outputs together as being required for correct achievement of a function, we clearly need a language to allow these relationships to be specified. This idea was introduced for creating a hierarchy of function in (Snooke 1998) but the need to model transient outputs and temporal relation entails some additions to this. The relations used herein are those suggested in (Bell 2003), as listed below.

AND Both subsidiary functions are required for the function to be achieved. For example 'main beam' consists of 'left main' and 'right main'.

OR For the function to be achieved, one or more of the subsidiary functions must be achieved.

XOR For the function to be achieved either of the subsidiary functions must be achieved, but not both.

NOT For the function to be achieved the subsidiary function must not be achieved. This is most useful in specifying transient behaviours, but needs careful use.

SEQUENCE For the function to be achieved, each subsidiary function should be achieved in turn, in the order specified. The sequence end with the last of the specified subsidiary functions.

RPT_SEQ The function consists of an ongoing cycle on the specified subsidiary functions which continues until some external event causes it to stop, such as the indicators flashing on and off until cancelled.

It is at least arguable that if time is to be modelled, some of these relations need some temporal qualification. Is AND satisfied if the two outputs occur at any stages of the simulation (so if

transient need not occur simultaneously) or should it imply the outputs are simultaneous at some stage of the simulation? If the former, we arguably need a relation that does require the outputs to be simultaneous, if the latter we need one that doesn't. Cases where the outputs should definitely not be simultaneous can be modelled using the sequence relation but cases where one output must come before the other but the order doesn't matter needs some thought.

A suggested approach is to leave the conventional logical relations free of time constraints and to add a SIMULTANEOUS relation to specify synchronisation between the required subsidiary functions. Could this be used as a qualifier for any binary logical relation? Is this sufficient or do we also need a synchronise relation to specify that not only are the subsidiary functions to occur at the same time, they must also be achieved simultaneously, allowing cases where both must occur together but it does not matter if one starts first be described using SIMULTANEOUS. This approach leaves the meaning of AND unchanged for compatibility with the current language and for use in cases where no time needs modelling, as there is no danger of timing constraints being violated. his approach also allow AND to be used instead of 'loose sequence' in the current AutoSteve version of the language.

5 Adding temporal labels

The advantages claimed for functional labels in (Price 1998) suggest that any attempt to add temporal information should be simple and not interfere with the capability of recognising unexpected achievement of function as well as preserving the labels' reusability. The suggested approach is simply to specify one or two deadlines for achievement of the function. These specify the time by which the function should be achieved and the time before which the function should not be achieved, when a delay is required. In many cases, of course, the delay will not be specified, and it need not be the case that either is specified if there are no temporal constraints on the function. If no temporal labels are used then neither late nor early achievement of the function will be identified, of course, just as at present. In the figures, the label 'after' has been used to specify the time after which the function should occur (the minimum delay) and 'before' has been used to identify the time before which the function should be achieved, the maximum allowable delay. If a time is to be specified then an inevitable requirement is a start point from which the time passing before achievement of the function can be measured. It is suggested that this need not be specified (at least for FMEA) as time can be measured from the input event (change of switch position) that stimulates achievement of the function. This will typically be the start of the step in in the sequence of simulations. Therefore a functional label for a headlamp system function might look like figure 1. In this case there is no need for any delay in achievement of the

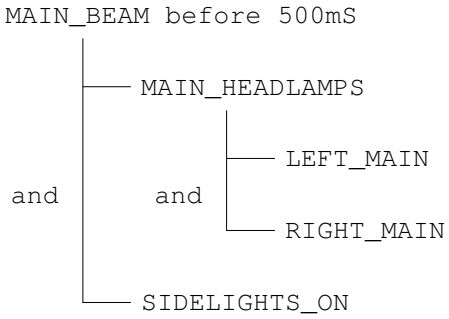

Figure 1: A car lighting system function, with deadline

function, so none is specified. Also the use of a hierarchy of functions is assumed, as described in

(Snooke 1998), and also that functions can be reused, so the necessary outputs for the sidelights subsidiary function are defined elsewhere. The reuse of functions (either as subfunctions or for other systems) raises the question of whether the temporal labels should themselves be re-used. In the case of a functional label being re-used as a subfunction (as in figure 1), the timing information will not be needed, as it will be attached to the top-level function. One exception to this is when a function that requires complex behaviour is re-used, but as it seems unlikely that identical behaviours will be specified for different purposes (so in different functions) the implicit need to specify the timing information should not be a problem. It is also suggested that in general timing constraints will not be reused between systems, so are specified anew on linking an existing functional model (set of labels) to a new system. This is because of the likelihood that the timing requirements will change between systems and also allows a functional model to be used without such constraints in cases where they are felt unnecessary, such as when there is no likelihood of a failure mode that results in delay to achievement of a system function.

It will be appreciated that the function is still achieved whenever the required outputs occur, but if there is a delay greater than that specified, then this can be noted and the fact that the function is achieved late can be included in the resulting report. As the function is not explicitly linked to any input state, the unexpected achievement of the function will be detected just as it is with no temporal labels. One point to be made is that there must be an implicit definition of when the function is achieved. In this case, as the child functions are specified using logical and, then it will naturally be when all the child functions' outputs are found. A function is defined as being achieved as soon as the required outputs have been identified as present. This is more interesting when a function depends on more complex behaviour, as will be discussed later. Naturally, for some design analysis tasks, such as SCA, there is a need to link a function's outputs with the correct inputs but it is suggested that adding the timing information does not affect this.

FMEA reports identify the significance of a failure effect by allocating it a risk priority number (RPN) of which the severity of the fault is a component. There will also be a remark in the FMEA report on the effect of the failure. Untimely achievement of a function is likely to result in a different effect and different value for the failure's severity, so where temporal labels are used, the user needs an opportunity to add this extra information. This is no different from what is already expected of the user when entering the effects of failure of the function but is additional to it. It seems reasonable to suggest that for FMEA, if the user cannot identify such effects (of late or early achievement of the function) then the temporal information need not be added, as it will add little extra to the final report.

The use of a specified deadline eliminates any need to relate variations in timing of function between the correct and failure mode behaviours, unless, of course, the timing constraints are violated, and is also arguably simpler for the user. This example is, of course, trivial in that the only consequence of late achievement of the function is a momentary lack of vision of the road, while the headlamps stay dipped. There are, however, cases where the deadlines will be much more important, such as the response time of an engine management system. In the case illustrated, the late achievement of the function is itself unlikely, but not impossible if the lighting system makes use of a data bus for passing its instructions, and this data bus uses a protocol such as CANbus, whose message latency is not deterministic. Of course it might be that no deadlines are specified, so late achievement of a function will not be detected. This has the advantage of being backwards compatible with the current functional labels.

As it appears impractical to use a qualitative notion of time to model late achievement of a function, as suggested in Section 3 above, the deadline is specified numerically.

As inputs are not specified (at least for FMEA) as at present, the proposed approach need not affect the use of inputs for SCA or design verification, in general. In other words there is no need to enter any intended inputs unless SCA is to be undertaken. There are cases where it

might be felt necessary to specify the trigger for a function as will be discussed later.

Having looked at how the idea of adding temporal labels to simple "steady state" function works, we can now consider how they work in more complex cases where the achievement of a function depends on a sequence of (possibly intermittent) outputs.

There is some discussion of functional labelling and transient outputs in (Bell 2003). This discusses the need to distinguish between sequences that run to completion before the system enters a steady state and those that continue until some external event (such as a new user input) triggers the end of the sequence. There is also some discussion there on the need to specify the required behaviour for all the intermittent states, even those associated with the absence of any output. Therefore these questions are only considered in so far as they affect the use of temporal labels.

If a function depends on a sequence of intermittent outputs then the functional labels might look like figure 2. In this case, a warning buzzer sounds three times before stopping. There are

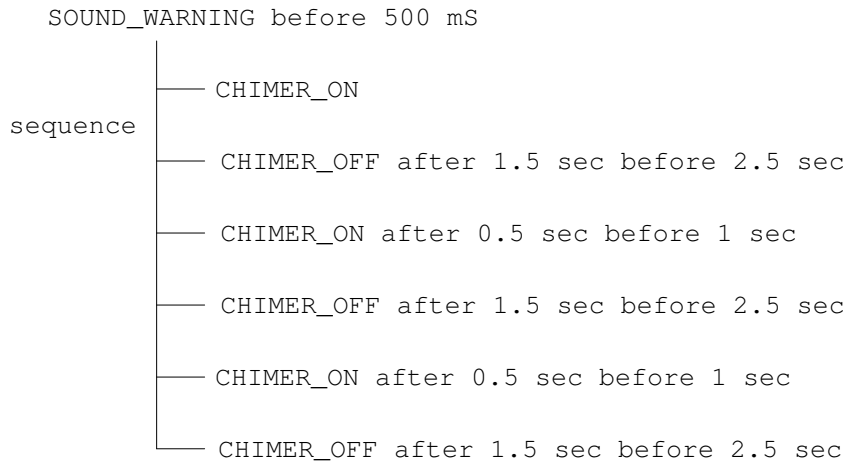

Figure 2: A function that requires a sequence of intermittent outputs

a few points worth discussing. As suggested earlier, the function deadline is defined as being when the function is first achieved. An extension to this idea is used here, in that it is required that the chiming sequence starts before the deadline specified in the top level function (500mS in this case). Therefore the first chime must start within 500mS of the trigger and needs no further timing information. All timing labels within the sequence are defined as being triggered by the previous event in the sequence, so the first "chimer off" function must start no sooner than 1.5 seconds after the start of the preceding "chimer on" function and no later than 2.5 seconds. This therefore specifies an acceptable range of duration for the chime. This continues throughout the sequence, of course. The use of the start of the sequence as the point for timing the achievement of the top level function is workable, as if any errors in the sequence are found, the top level function will not be achieved, as at present, so its timing is no longer of interest. It might be desirable to distinguish between cases where a top level function fails merely because of errors in timing of subsidiary functions. It does not seem impossible to distinguish such cases.

As an aside, in this case the functions depend on the presence or absence of output of one component, the chimer, so specifying the required outputs is simple.

In figure 2, the sequence ends with the system in a steady state, associated with the final function in the sequence. In other cases, the sequence should continue indefinitely, until some other external event triggers a change of behaviour. In (Bell 2003) this case was distinguished by the use of a "repeating sequence". As figure 3 illustrates, this slightly complicates the specification of temporal labels. In this case the duration of each element in the sequence is

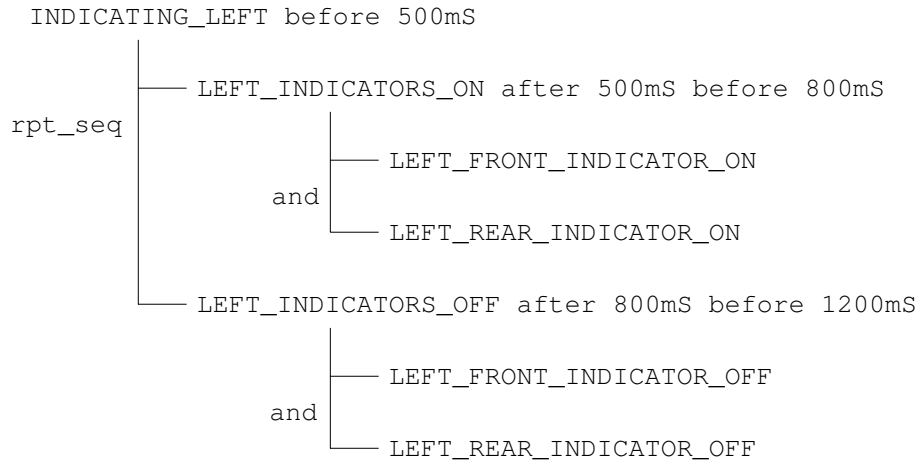

Figure 3: A function that depends on an indefinite sequence of outputs

defined by specifying the start point for the succeeding element, as in figure 2. The timing of the first element, however (the first "indicators on") is still derived from the deadline of the top level function and the timing of the first "indicators off". As each temporal label is used to define when the function should start relative to the previous member of the sequence, it is not used in this first instance.

In (Bell 2003) it was suggested that it might be desirable to define the behaviour a function depends on in terms of time. For example, it might be intended that a chimer sounds intermittently for say five seconds before stopping. It will be appreciated that this behaviour cannot be modelled without the use of some sort of temporal information. A possible approach using temporal labels as proposed here and the two different sequence relations (sequence and repeat sequence) is illustrated in figure 4. Of course, it might be the case that the overall time runs

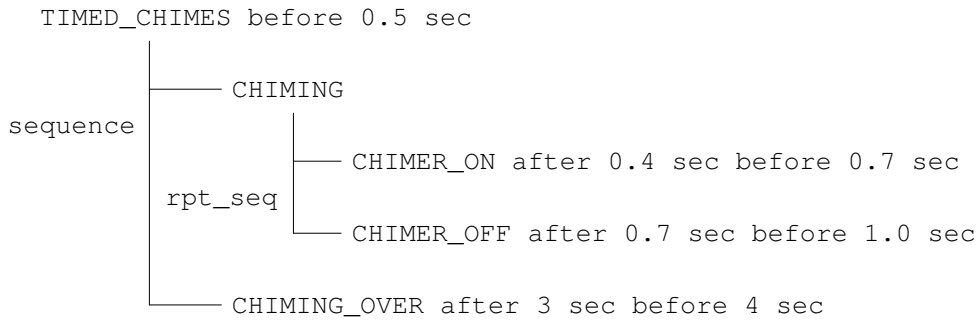

Figure 4: A sequence that runs for a specified time

out while the chimer is off so the output might seem slightly too short. It is suggested that this is insignificant and can be ignored.

One possible use of the 'after' condition is to allow the ordering of child functions to be specified, as illustrated in figure 5. In this case, a warning should be sounded between three and five seconds after a telltale light comes on. In this case, the timings have been associated with the child functions. It is not appropriate to attach the timing to the top level function, as there are distinct timing conditions for each child. If timing errors are found the FMEA report will

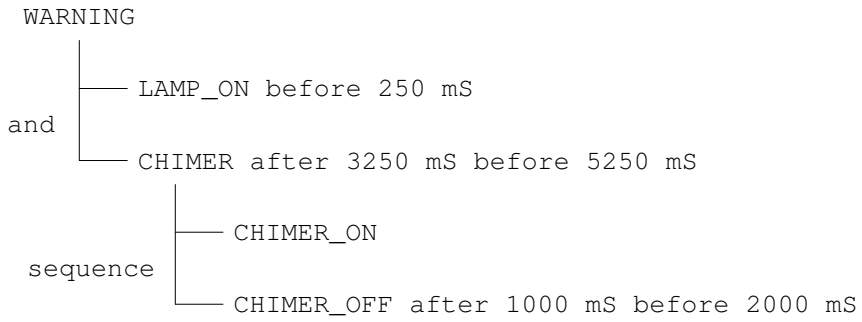

Figure 5: Combining child functions, specifying the order using AND

read something like "function WARNING not achieved because function LAMP_ON achieved late".

It is suggested that the proposed approach to adding temporal labels and the language seems to be sufficient for modelling complex behaviour where necessary without demanding any more effort from the user than at present where such complexity is not needed.

because no inputs need be specified for a function for FMEA, as at present, problems arise if a function should be triggered by achievement or non-achievement of some other function. There is some discussion of this in (Bell 2003). It is suggested that while in general there seems to be no need to specify a function's intended input for FMEA a trigger could be specified in individual cases. This is quite simple where a function is triggered by the achievement of its trigger but less so where it is triggered by failure of some function, as discussed in (Bell 2003).

6 Discussion

It seems worth summarising the effect of adding timing information to the functional labels in terms of their simplicity, reusability and capability; the three advantages claimed for the approach in (Price 1998).

It is suggested that the proposed approach to using functional labels, combined with the hierarchical function description language, allows the user to specify cases where a required system function depends on behaviour of considerable complexity, but as the proposed additions can be regarded as options, there is no need for this extra complexity to modelled except where it is felt necessary. It will be appreciated that adding extra information must reduce the simplicity of the model and demand some extra effort from the user, but the suggested approach does reduce this as far as possible. The timing information itself is simple and if it is not to be explicitly linked to the expected input, then the essential simplicity of the approach is preserved as far as possible. It is also the case that there is no need to add this information in all cases, and if there is no timing constraint on a function, then the present simplicity of the approach is unaffected.

There is some question as to how the additional information might affect reuse of a functional model. It seems not unlikely that different systems will have essentially identical functions but with different time constraints. The suggested approach here is to attach the timing information to a specific system's use of a set of functional labels, so the library of reusable functional models has no timing information. This also allows a functional model (set of labels) to be used with no timing constraints in cases where the complication is felt unnecessary. For example a lighting system could be used with no timing information except in cases where the use of a data bus in the lighting system introduces the possibility of late messages delaying achievement of functions.

Also, where a subsidiary function is reused (as in the headlamp function in figure 1) then timing constraints would not be reused, as they will generally be attached to the top level function.

As function output is still not explicitly linked to input, the ability of this approach to functional modelling to detect unexpected achievement of a function (or, strictly, unexpected outputs) is unaffected.

Naturally, the addition of temporal information to functional labels makes demands on the behavioural simulator and on the behavioural models used. If the late achievement of a function is to be recognised then the behavioural simulator needs to make use of models that have some suitable notion of time, so the time taken to achieve a function can be obtained from the simulator. As we use state charts to describe components' behaviour, as discussed in (Snooke 1999), then such information can readily be captured though the state charts will, of course, need to use quantitative time rather than the order of magnitude qualitative notion of time that is used as an alternative. This qualitative time could, of course, still be used but the temporal labels will be ignored and late or early achievement of functions will not be identified.

When these temporal labels are used to describe functions that depend on complex behaviour then the behavioural description of the system must be sufficient to capture any transient state or outputs. The simplest approach is to have the behaviour checked against the functional label at each step in each simulation, not simply once at the end. This is, of course, necessary whenever a function depends on a sequence of transient outputs whether or not the timing constraints are themselves specified. Adding these is advantageous, as it provides a way of ensuring that the outputs occur for a useful period of time, so that a warning lamp does not flash for a few milliseconds, say.

7 Conclusion

The approach to adding temporal constraints to the existing technique of using functional labels to identify significant system behaviour allows a design analysis tool to recognise cases where a function is achieved but in an untimely manner. The proposed approach does this while preserving, as far as possible, the advantages of the functional labels. Naturally, more information needs to be entered into the analysis tool, but this is kept to a minimum and need not affect the reusability of the functional labels. The use of the start of the simulation step (so a change in input to the system) as the starting point for measuring time avoids the need to link function to input (at least for FMEA) so maintaining the capability of recognising the unexpected achievement of a function, and avoiding problems specifying the input in cases where this depends on the previous state of the system.

8 Questions

These will, of course, not appear in a finished version.

- Is there a paper in here?
- Is this approach to temporal aspects of function sufficient?
- Do we need more background in the introduction? — More on FMEA, adoption of functional labels?
- More on the specification of expected behaviour for SCA?, see Section 3. Might this be another paper? There seems to be some mildly interesting arguments about full definitions of function — input, output, required behaviour between them. We raised this at the meeting on 23/7 and this might well become more relevant if we do look at fault mitigation.

- Are there any relevant papers on temporal modelling?
- Is there a suitable mathematical notation that might be used here?
- There is related material in (Bell 2003). Are they better combined for possible publication? It seems so to me. Only problem might be that there are too many loose ends, especially if we are adding stuff about definition of function — input, output and internal state. Should we come up with a reasonable subset of all this for a conference paper first?

References

Bell, J. (2003). A language for hierarchical function. SoftFMEA report ref. SD/TR/FR/05.

Jordan, W. E. (1972). Failure modes, effects and criticality analyses. In *Proceedings Annual Reliability and Maintainability Symposium*, pp. 30–37. IEEE Press.

Lee, M. H. and A. R. T. Ormsby (1994). Qualitatively modelling the effects of electrical circuit faults. *Artificial Intelligence in Engineering 8*, 293–300.

Price, C. J. (1998). Function-directed electrical design analysis. *Artificial Intelligence in Engineering 12*(4), 445–456.

Price, C. J., N. Snooke, and J. Landry (1996). Automated sneak identification. *Engineering Applications of Artificial Intelligence 9*(4), 423–427.

Raiman, O. (1991). Order of magnitude reasoning. *Artificial Intelligence 51*, 11–38.

Snooke, N. A. (1998). Hierarchical functional reasoning. *Knowledge-Based Systems 11*, 301–309.

Snooke, N. A. (1999). Simulating electrical devices with complex behaviour. *AI Communications 12*(1,2), 45–58.