Probabilistic Optimization and Assessment of Voting Strategies for X-By-Wire Systems

Markus Kucera[1], Hans Mauser[2]

[1]University of Applied Sciences, Regensburg
markus.kucera@informatik.fh-regensburg.de
[2]Siemens AG, München
hans.mauser@siemens.com

Abstract. Signal voting of redundant sensor values and communication channels is of central importance in today's X-by-wire systems. The required degree of sensor redundancy, the type of redundancy, and finally the voting strategy must be designed to meet the system's dependability requirements. These design decisions depend on an analysis of the probabilities and effects of all underlying fault scenarios. Given a probabilistic fault model and a communication model, the voting step can be formally stated as a maximum-likelihood estimation of the correct input signal. With an example of an X-by-wire system we show how GTEFT can be used to derive the failure probabilities of different fault scenarios for various systems architectures and different voting strategies. Thus the capability of GTEFT to support system development and system assessment is demonstrated.

Keywords. Fault-tolerance, Dependability, Voting, Embedded System, Automotive

1. Introduction

The driving innovation force in today's cars has become the area of Information Technology. Powerful new vehicle systems like Brake-by-Wire, Steer-by-Wire, or Park-Assistant are not viable without powerful functionality in software and electronics [1]. This increase in functionality comes together with an increase in system complexity.
Complexity is thus one major challenge to face when dealing with future embedded systems.

The increasing ubiquitous use of embedded systems directs many applications into an area where problems regarding human health and life emerge. This leads to the topic of dependability and safety in particular. Safety has become a topic of special importance in the automotive area, where system cost, penetration rates, and short innovation cycles are the driving factors.

Thus, meeting dependability requirements whilst meeting cost targets and development schedule is the second major challenge to face in that area.

The idea to support dependability assessment by tools is not new. Many approaches exist to evaluate dependability figures for distributed systems [7-10].
However, two main problems remain when dealing with above mentioned challenges:
(1) countering the state space explosion problem
(2) efficient integration into a product development cycle to allow for professional use

In [3] an approach is presented that synthesizes dynamic fault trees from UML System Models. The author's main motivation to use UML as modelling and specification language was to integrate into their sponsor's development toolchain. In [4] another approach is presented, that aims at efficient integration into the product development cycle. For that purpose Matlab/Simulink is used for system modelling. Both approaches provide semi-automatic fault tree synthesis for reliability assessment. In [6] Grunske and Kaiser present an approach that offers the possibility for automatic fault tree generation by providing a special Transformation Notation between interacting components. The state space explosion problem, however is not countered following above presented approaches.

In [11] a method is presented that relieves the problem of state space explosion by combining formal and informal techniques. Amari et al. [5] propose a method to analyze dynamic fault trees in order to find the best strategy for avoiding or minimizing the state space explosion problem. The problem of efficient integration into the product development cycle, however is not solved following these methods.

In contrast to these approaches, we presented GTEFT in [2]. GTEFT is an approach that solves both of above mentioned challenges. To do so, we combine simulative and analytic techniques. GTEFT makes use of a COTS GUI (Matlab/Simulink) for efficient integration into the product development cycle. For dependability evaluation GTEFT makes use of classical Markov theory. The problem of state space explosion is avoided by means of a dependability module that uses locality traversing. GTEFT not only allows to derive reliability figures for a given system architecture automatically. It also generates and analyses all failure sequences possible in a given system. New system development or system optimisation is thus strongly supported.

Voting strategies are a central part of today's safety related systems. In general, voting strategies, and thus voting decisions, can be classified as exact, or probabilistic. The distinction between probabilistic and exact voting decisions depends on the underlying fault-model. If the fault-model contains complex faults like conspiracy scenarios, then every voting strategy can be corrupted and exact voting decisions become impossible.

For safety-related systems the question whether a voting strategy can guarantee exact decisions for a reasonable fault-model is crucial.

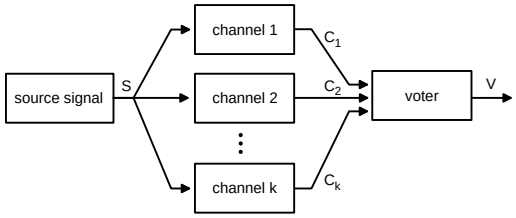
On the other hand, the questions whether probabilistic voting decisions can be applied, and what error-probabilities are acceptable, depend on the system under investigation. In some cases probabilistic voting decisions are acceptable as a last resort to make best-effort decisions in the presence of severe fault scenarios. If the system and consequently the voter output has a safe state, it is advisable to vote ambiguous input signal combinations to the safe-state signal.

In this paper we present a way to derive probabilistic voting strategies with GTEFT. Given a probabilistic fault model and a communication model, the voting step is formally stated as a maximum-likelihood estimation of the correct input signal. This estimation can then be exploited in order to develop a suitable voting strategy.

2. Signal Voting

We show the usefulness of exhaustive fault-state-space enumeration in the context of the generation and analysis of voting strategies. The necessity of voting arises in redundant systems, when input signals are transmitted over several independent communication channels. Ultimately, the redundant communication signals have to be voted into one authoritative signal that drives an actuator.

We state the voting problem as a channel-decoding problem as follows:



A source signal S is generated by a sensor, by manual input or by an automated control system and has to be transmitted to an actuator which reacts upon the value of the signal. In order to achieve fault-tolerance, the source signal S is split and communicated redundantly over k independent communication channels. The outputs of these channels are denoted $C_1 \ldots C_k$ and are processed by a voter which generates an output V. In the fault-free case all signals S, $C_1 \ldots C_k$ and V agree. We assume that faults can affect the communication channels, such that the voter input signals $C_1 \ldots C_k$ can differ from S. In this case the voter has to make its decision based on faulty input data. The voter decision is correct if the value of the output signal V is identical to S. A voting strategy is a function $V(C_1, \ldots, C_k)$ that maps all possible input signal values to a voted output signal.
We classify the voting decisions for a given input signal combination $C_1, \ldots, C_k$ as follows:

- **exact voting decisions**: The value of the source signal S can be inferred with absolute certainty from the voter input. This means that no fault-combination will cause the voting strategy to make a wrong decision.
- **Probabilistic voting decisions**: The combination of input signals does not allow to infer the correct source signal with absolute certainty. Based on the values of the input signals, however, the correct value of the source signal can be reconstructed with a high probability. The decision involves an error-probability which should be small.

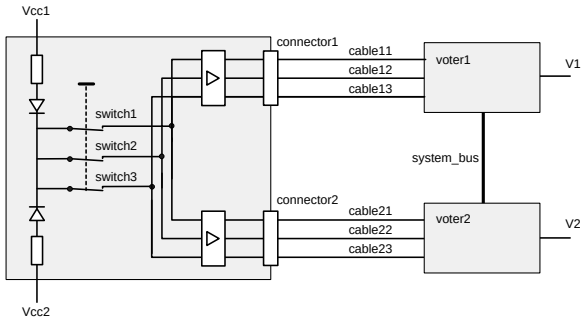
Note that even in the case of probabilistic voting decisions, the voting strategy remains a deterministic function.

The distinction between probabilistic and exact voting decisions depends on the underlying fault-model. If the fault-model contains complex faults like conspiracy scenarios between independent communication channels, then every voting strategy can be corrupted and exact voting decisions become impossible. Conversely, by adding redundant communication channels the voting strategy can be made more robust to communication faults and more combinations of input signals can be voted by an exact voting decision.

In the following we present a general approach that employs exhaustive fault-state-space exploration to automatically generate voting strategies for a given system architecture. For every input-signal combination an optimal voting decision is found. Furthermore, for every voting decision we derive whether the decision is exact or calculate the conditional probability of the voting decision being correct given the correct value of the source signal.

3. System Description

We present a simple system architecture as a case study for constructing and analyzing the voting strategy. The example was chosen to be simple enough for an exhaustive treatment, yet to contain the key elements and features of a real-world system.



The system has a typical dual-channel architecture as is frequently employed in X-by-wire applications. A switch unit generates the source signal from a mechanical input by an array of three redundant switches. The switches have normally-closed (NC) contacts, the switch unit has redundant power supply. The three redundant signals from the switches are transmitted to two independent actuators, which react upon the signals after a voting decision has been made. The input stages of the voters are assumed to have pull-down resistors so that open input lines are read as logical 0. The independent actuators also communicate over a system bus and transmit their individual input signal over the system bus to each other. Hence, the voter receives 6 input signals: Three input signals S_1, S_2, S_3 directly from the switch unit and three signals R_1, R_2, R_3 relayed from the peer voter.

For the system we consider the following fault model:

fault identifier	fault description	failure-rate [fit]
vcc1_down	voltage supply from channel one insufficient	10
vcc2_down	voltage supply from channel two insufficient	10
switch1_stuck_open	the switch has always disconnected terminals	4
switch1_stuck_closed	the switch has always connected terminals	1
switch2_stuck_open	… same for other switches	4
switch2_stuck_closed		1
switch3_stuck_open		4
switch3_stuck_closed		1
connector1_disconnected	the connector1 disconnects all cables	5
connector2_disconnected	the connector2 disconnects all cables	5
cable11_disconnected	cable11 is open	5
cable12_disconnected	cable12 is open	5
cable13_disconnected	cable13 is open	5
cable21_disconnected	cable21 is open	5
cable22_disconnected	cable22 is open	5
cable23_disconnected	cable23 is open	5

The failure rates are stated in fit, i.e. failures / 10^9 hours of operation. All faults are assumed to be independent and to have exponential lifetime distributions. Obviously stuck_open and stuck_closed failure modes are mutually exclusive.

Since the system architecture and the fault-model were simplified to allow a self-contained and comprehensive presentation, we conclude with some comparative remarks on real-world applications:

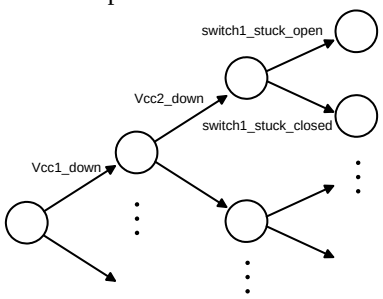
Usually a sensor unit should have diverse sensors and generate dynamic signals which can easily be checked for validity. The fault model for a real system must be developed systematically by an FMEA (failure mode and effect analysis) of the system and all its components. We have omitted system-bus failures. Since the system-bus communication will be protected by CRC-codes, communication errors will be detected with a high probability. The voter would then have to process detectably unavailable signals at its input. Though this poses no conceptual problem, it increases the set of possible input combinations. Shorted communication lines were also omitted.

4. Automatic Generation of Voting Strategies

In order to construct a voting strategy, we explore the failure-state-space exhaustively and analyse how the different failure-scenarios affect the communication of the source signal to the voter input. This allows to determine which failure combinations can affect the communication in such a way that a given input signal combination arrives at the voter.

We have presented efficient methods for enumerating the failure states and calculating their corresponding time-dependent probabilities in [2]. The method is based on

a depth-first exploration of all possible component-fault-sequences beginning from the global intact state, where all components are intact.



The diagram shows a part of the generated state space. Nodes represent failure states, edges represent component faults and are labelled with the fault identifier. By replacing the edge labels with the associated failure rates, the event graph can be analysed as a Markov-Chain to calculate the state probabilities.

Brute-force attempts at exhaustive state-space exploration suffer from the problem of state-space explosion. It is necessary to define reasonable truncation criteria where the exploration of further fault-events is aborted. The following truncation criteria are useful:

- If we reach a failure state where the continued operation of the system becomes impossible, we can truncate the depth-first search, because the system will have to be repaired immediately and therefore will not encounter further faults before the repair.
- It is reasonable to limit the length of considered fault-sequences. This is possible because long fault-sequences have small probabilities and can therefore be neglected in the sense of a rare-event-approximation. Also to meet a specified safety and integrity level it suffices to consider fault sequences up to a required length.

With these truncation criteria it is possible to analyse a realistic fault model with several hundred single component faults up to a reasonable search depth.

For the example model we have analyzed all fault-sequences of length up to three and calculated their time-dependent probabilities. What we get from this analysis is a set of fault scenarios $F = \{f_1, f_2, \ldots, f_n\}$ and their associated probabilities $P(f_1)$, $P(f_2)$, … $P(f_n)$. Since X-by-wire systems usually have to survive a given mission time without repair, the probabilities $P(f_i)$ are calculated for the time at the end of the mission assuming that all components were intact at the beginning of the mission and that there was no repair during the mission. Reasonable mission times can be inspection periods, maintenance periods or for non-safety-critical systems warranty periods. For the purpose of this study we have chosen a mission time of 2 years of continued operation.

Note that the correct procedures for calculating the probabilities $P(f_i)$ depend on the system under investigation. Alternative methods and assumptions for calculating $P(f_i)$ can be perfectly suited for other systems and do not affect the following procedure for constructing voting strategies.

Given the set of considered fault scenarios F and the associated probabilities $P(f_i)$, the voting strategy is constructed as described in the following pseudo code:

```
Initialize the following array data-structures to zero:
n[s, C1, …, Ck]
p[s, C1, …, Ck]
// traverse fault scenarios
For all fault scenarios f in F
 {
  For all values s of the source signal
   {
    Calculate the input signal combination
    (C1, …, Ck) to the voter, resulting from
    f and s.
    increment n[s, C1, …, Ck] by 1
    increment p[s, C1, …, Ck] by P(f)
   }
 }
// output voting strategy
For every input signal combination (C1, …, Ck)
 {
  if (0 < n[s, C1, …, Ck]
      and for all t != s: 0 == n[t, C1, …, Ck]
     )
   {
    The exact voting decision is s
   }
  else if (for all t != s:
            p[t, C1, …, Ck] < p[s, C1, …, Ck]
           )
   {
    The probabilistic voting decision is s
   }
  else
   {
    No voting decision possible
   }
 }
```

The values of the arrays can be readily interpreted:

- `n[s, C1, …, Ck]`
  represents the number of considered fault scenarios that cause the input signal combination C_1, …, C_k to appear at the voter input given the condition that s is the correct input signal. This is used to make exact voting decision by ruling out all possible source signal values but one.
- `p[s, C1, …, Ck]`
  is the conditional probability that the input signal combination C_1, …, C_k appears at the voter input given the condition that s is the correct input signal. This is used to make probabilistic voting decisions by maximum likelihood estimation of the true source signal value. This maximum likelihood decision is optimal, if we make no a priori assumptions on the probability of the source signal values.

The above procedure includes the possibility that for a given input combination C_1, …, C_k to the voter no well-justified voting decision can be made. This can be the case if:

- The input combination C_1, …, C_k does not arise for any source signal s and any considered fault scenario f.
- The input combination C_1, …, C_k is equally likely for different values of the source signal. This can occur if the system architecture shows a certain degree of symmetry.

If no voting decision can be made, we recommend to refine the fault model or to set the voter output to a safe state for unresolvable voter inputs.

For the example system model we show the complete voting table. Since the voters are symmetric their voting strategies will be identical up to the naming convention of the input signals. The constructed voting table can easily be coded into software or hardware for a real system implementation.

The analysis has been explicitly carried out for first-fault scenarios, double-fault scenarios and triple-fault scenarios separately.

For the source signal the signal values 0 and 1 have the following interpretation:
0: button released, i.e. switches closed
1: button applied, i.e. switches open

For the communication signals s_i, r_i the values 0 and 1 have the following interpretation:
0: signal from switch unit connected to ground.
1: signal from switch unit connected to Vcc.

The following values are reported in the table:
`n(0)` corresponding to `n[0, C1, …, Ck]`
`n(1)` corresponding to `n[1, C1, …, Ck]`
`p(0)` corresponding to `p[0, C1, …, Ck]`
`p(1)` corresponding to `p[1, C1, …, Ck]`

The column ‘exact’ marks all exact voting decisions with the label ‘exact’ and all probabilistic decisions with ‘-‘.

Note in interpreting the n(0), n(1) values that our implementation counts all permutations of a fault sequence individually.

	voter input							single component fault scenarios					double component fault scenarios					triple component fault scenarios				
#	s1	s2	s3	r1	r2	r3	V	n(0)	p(0)	n(1)	p(1)	exact	n(0)	p(0)	n(1)	p(1)	exact	n(0)	p(0)	n(1)	p(1)	exact
0	0	0	0	0	0	0	1	0		13	1,26E-03	exact	4	3,83E-08	169	1,26E-03	-	178	3,84E-08	1975	1,26E-03	-
1	0	0	0	0	0	1	1	0		0			0		4	3,07E-09	exact	30	2,60E-12	106	3,07E-09	-
2	0	0	0	0	1	0	1	0		0			0		4	3,07E-09	exact	30	2,60E-12	106	3,07E-09	-
3	0	0	0	0	1	1	0	0		0			4	1,38E-08	0		exact	106	1,38E-08	6	2,68E-14	-
4	0	0	0	1	0	0	1	0		0			0		4	3,07E-09	exact	30	2,60E-12	106	3,07E-09	-
5	0	0	0	1	0	1	0	0		0			4	1,38E-08	0		exact	106	1,38E-08	6	2,68E-14	-
6	0	0	0	1	1	0	0	0		0			4	1,38E-08	0		exact	106	1,38E-08	6	2,68E-14	-
7	0	0	0	1	1	1	0	1	8,75E-05	0		exact	17	8,75E-05	0		exact	185	8,75E-05	0		exact
8	0	0	1	0	0	0	1	0		0			0		4	3,07E-09	exact	30	2,60E-12	106	3,07E-09	-
9	0	0	1	0	0	1	1	0		1	1,75E-05	exact	2	4,91E-09	17	1,75E-05	-	56	4,91E-09	179	1,75E-05	-
10	0	0	1	0	1	0	0	0		0			0		0			6	5,37E-13	0		exact
11	0	0	1	0	1	1	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
12	0	0	1	1	0	0	0	0		0			0		0			6	5,37E-13	0		exact
13	0	0	1	1	0	1	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
14	0	0	1	1	1	0	0	0		0			0		0			6	6,71E-13	0		exact
15	0	0	1	1	1	1	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
16	0	1	0	0	0	0	1	0		0			0		4	3,07E-09	exact	30	2,60E-12	106	3,07E-09	-
17	0	1	0	0	0	1	0	0		0			0		0			6	5,37E-13	0		exact
18	0	1	0	0	1	0	1	0		1	1,75E-05	exact	2	4,91E-09	17	1,75E-05	-	56	4,91E-09	179	1,75E-05	-
19	0	1	0	0	1	1	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
20	0	1	0	1	0	0	0	0		0			0		0			6	5,37E-13	0		exact
21	0	1	0	1	0	1	0	0		0			0		0			6	6,71E-13	0		exact
22	0	1	0	1	1	0	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
23	0	1	0	1	1	1	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
24	0	1	1	0	0	0	0	0		0			4	1,38E-08	0		exact	106	1,38E-08	6	2,68E-14	-
25	0	1	1	0	0	1	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
26	0	1	1	0	1	0	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
27	0	1	1	0	1	1	0	1	7,00E-05	0		exact	15	7,00E-05	2	3,07E-10	-	129	7,00E-05	32	3,07E-10	-
28	0	1	1	1	0	0	0	0		0			0		0			6	6,71E-13	0		exact
29	0	1	1	1	0	1	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
30	0	1	1	1	1	0	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
31	0	1	1	1	1	1	0	1	8,75E-05	0		exact	11	8,75E-05	0		exact	65	8,75E-05	0		exact
32	1	0	0	0	0	0	1	0		0			0		4	3,07E-09	exact	30	2,60E-12	106	3,07E-09	-
33	1	0	0	0	0	1	0	0		0			0		0			6	5,37E-13	0		exact
34	1	0	0	0	1	0	0	0		0			0		0			6	5,37E-13	0		exact
35	1	0	0	0	1	1	0	0		0			0		0			6	6,71E-13	0		exact
36	1	0	0	1	0	0	1	0		1	1,75E-05	exact	2	4,91E-09	17	1,75E-05	-	56	4,91E-09	179	1,75E-05	-
37	1	0	0	1	0	1	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
38	1	0	0	1	1	0	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
39	1	0	0	1	1	1	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
40	1	0	1	0	0	0	0	0		0			4	1,38E-08	0		exact	106	1,38E-08	6	2,68E-14	-
41	1	0	1	0	0	1	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
42	1	0	1	0	1	0	0	0		0			0		0			6	6,71E-13	0		exact
43	1	0	1	0	1	1	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
44	1	0	1	1	0	0	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
45	1	0	1	1	0	1	0	1	7,00E-05	0		exact	15	7,00E-05	2	3,07E-10	-	129	7,00E-05	32	3,07E-10	-
46	1	0	1	1	1	0	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
47	1	0	1	1	1	1	0	1	8,75E-05	0		exact	11	8,75E-05	0		exact	65	8,75E-05	0		exact
48	1	1	0	0	0	0	0	0		0			4	1,38E-08	0		exact	106	1,38E-08	6	2,68E-14	-
49	1	1	0	0	0	1	0	0		0			0		0			6	6,71E-13	0		exact
50	1	1	0	0	1	0	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
51	1	1	0	0	1	1	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
52	1	1	0	1	0	0	0	0		0			2	6,13E-09	0		exact	44	6,14E-09	6	2,68E-14	-
53	1	1	0	1	0	1	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
54	1	1	0	1	1	0	0	1	7,00E-05	0		exact	15	7,00E-05	2	3,07E-10	-	129	7,00E-05	32	3,07E-10	-
55	1	1	0	1	1	1	0	1	8,75E-05	0		exact	11	8,75E-05	0		exact	65	8,75E-05	0		exact
56	1	1	1	0	0	0	0	1	8,75E-05	0		exact	17	8,75E-05	0		exact	185	8,75E-05	0		exact
57	1	1	1	0	0	1	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
58	1	1	1	0	1	0	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
59	1	1	1	0	1	1	0	1	8,75E-05	0		exact	11	8,75E-05	0		exact	65	8,75E-05	0		exact
60	1	1	1	1	0	0	0	0		0			2	7,66E-09	0		exact	32	7,67E-09	0		exact
61	1	1	1	1	0	1	0	1	8,75E-05	0		exact	11	8,75E-05	0		exact	65	8,75E-05	0		exact
62	1	1	1	1	1	0	0	1	8,75E-05	0		exact	11	8,75E-05	0		exact	65	8,75E-05	0		exact
63	1	1	1	1	1	1	0	5	4,02E-04	0		exact	23	4,02E-04	0		exact	65	4,02E-04	6	5,37E-15	-

5. Analysis of the Voting Strategy

The derived voting strategy can make exact voting decisions for all single component fault scenarios. This means that no single fault from our fault-model can defeat the voting strategy.

In the columns for double component fault scenarios we find some input signal combinations were the voting strategy cannot make exact decisions. This could be ex-

pected, since the degree of redundancy of our system is two with respect to power supply and connectors and the degree of redundancy is three with respect to switches. By probabilistic voting the voter can, however, make a best-effort decision for all input signal combinations.

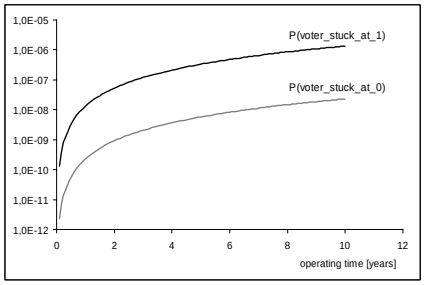
Note that the voting strategy makes exact decision, where a simple majority voter would have failed: In line number 24 the majority of four input signals has value 0 indicating that the button of the switch unit is applied. The exact voting result, however, is 0, indicating a released button.

The columns for the triple component fault scenarios show that the voter can still make exact decisions for half the input signal combinations.

The error probability of the derived voting strategy can be analyzed by summing up the probability of all states were the voter makes wrong decisions for either value of the source signal. For the example model the time-dependent error probabilities are plotted in the following diagram. The voter output shows a bias towards the applied switch position (voter output v = 1). Obviously there are more fault combination which simulate an applied switch than a released switch. An inspection of the critical fault sequences shows that identical stuck-at faults at two switches defeat the voting strategy. In addition, the voter_stuck_at_1 error can also be caused by double faults in the power supply (Vcc1_down, Vcc2_down) or in the two connectors (connector1_disconnected, connector2_disconnected).

Replacing the normally-closed switches by normally-open switches would allow to shape the error probabilities.

Since the error-probabilities approach 0 for short mission times, the total failure rate of the system can be kept acceptably low by enforcing short inspection and maintenance intervals to detect and remove first component faults.



6. Conclusion

This paper presented a practical approach to develop and assess voting strategies that make use of probabilistic voting decisions. For safety-related systems the question whether a voting strategy can guarantee exact decisions for a reasonable fault-model is crucial. In systems where probabilistic voting decisions are acceptable such deci-

sions could be used, e.g. as a last resort to make best-effort decisions in the presence of severe fault scenarios.

References

[1] M. Kucera: Drive-By-Wire Applications for future vehicles (in German). 20. Tagung Elektronik im Kraftfahrzeug, Haus der Technik, Essen, Juni 2000.

[2] M. Kucera, H. Mauser: Semi-Automatic Reliability Assessment of Safety Related Embedded Systems. Proceedings of the 18th IASTED International Conference on Parallel and Distributed Computing and Systems, 2006. 13-15 Nov. 2006. Page(s): 495 - 502

[3] Pai, G.J.; Dugan, J.B.; Automatic synthesis of dynamic fault trees from UML system models. Proceedings of the 13th International Symposium on Software Reliability Engineering, ISSRE 2002, 12-15 Nov. 2002 Page(s):243 – 254.

[4] Papadopoulos, Y.; Grante, C.; Techniques and tools for automated safety analysis & decision support for redundancy allocation automotive systems. Proceedings of the 27th Annual International Computer Software and Applications Conference, COMPSAC 2003. 2003. 3-6 Nov. 2003 Page(s):105 – 110

[5] Amari, S.; Dill, G.; Howald, E.; A new approach to solve dynamic fault trees. Proceedings of the annual Reliability and Maintainability Symposium, 2003. 27-30 Jan. 2003 Page(s):374 - 379

[6] Grunske, L.; Kaiser, B.; Automatic generation of analyzable failure propagation models from component-level failure annotations. Fifth International Conference on Quality Software, 2005. (QSIC 2005). 19-20 Sept. 2005 Page(s):117 – 123

[7] Allan Johnson, Miroslaw Malek: Survey of Software Tools for Evaluating Reliability, Availability, and Serviceability. ACM Computing Surveys, Vol. 20, No. 4, December 1988

[8] Sheldon, Greiner, Benzinger: Specification, Safety and Reliability Analysis Using Stochastic Petri Net Models. Proceedings of the 10th International Workshop on Software Specification and Design (IWSSD'00), 2000, Page 123

[9] Zhengzhu Feng, Richard Dearden, Nicolas Meuleau, Richard Washington: Dynamic Programming for Structured Continuous Markov Decision Problems. In Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, 2004, pages 154-161

[10] Frederick T. Sheldon, Kshamta Jerath: Assessing the Effect of Failure Severity, Coincident Failures and Usage-Profiles on the Reliability of Embedded Control Systems. 2004 ACM Symposium on Applied Computing, SAC'04 March 14-17, 2004, Nicosia, Cyprus, pages 826-833

[11] D. Karlsson, P. Eles, Z. Peng: Validation Of Embedded Systems Using Formal Method Aided Simulation. Proceedings of the 8th Euromicro Conference on Digital System Design, 2005. 30 Aug.-3 Sept. 2005 Page(s):196 – 199