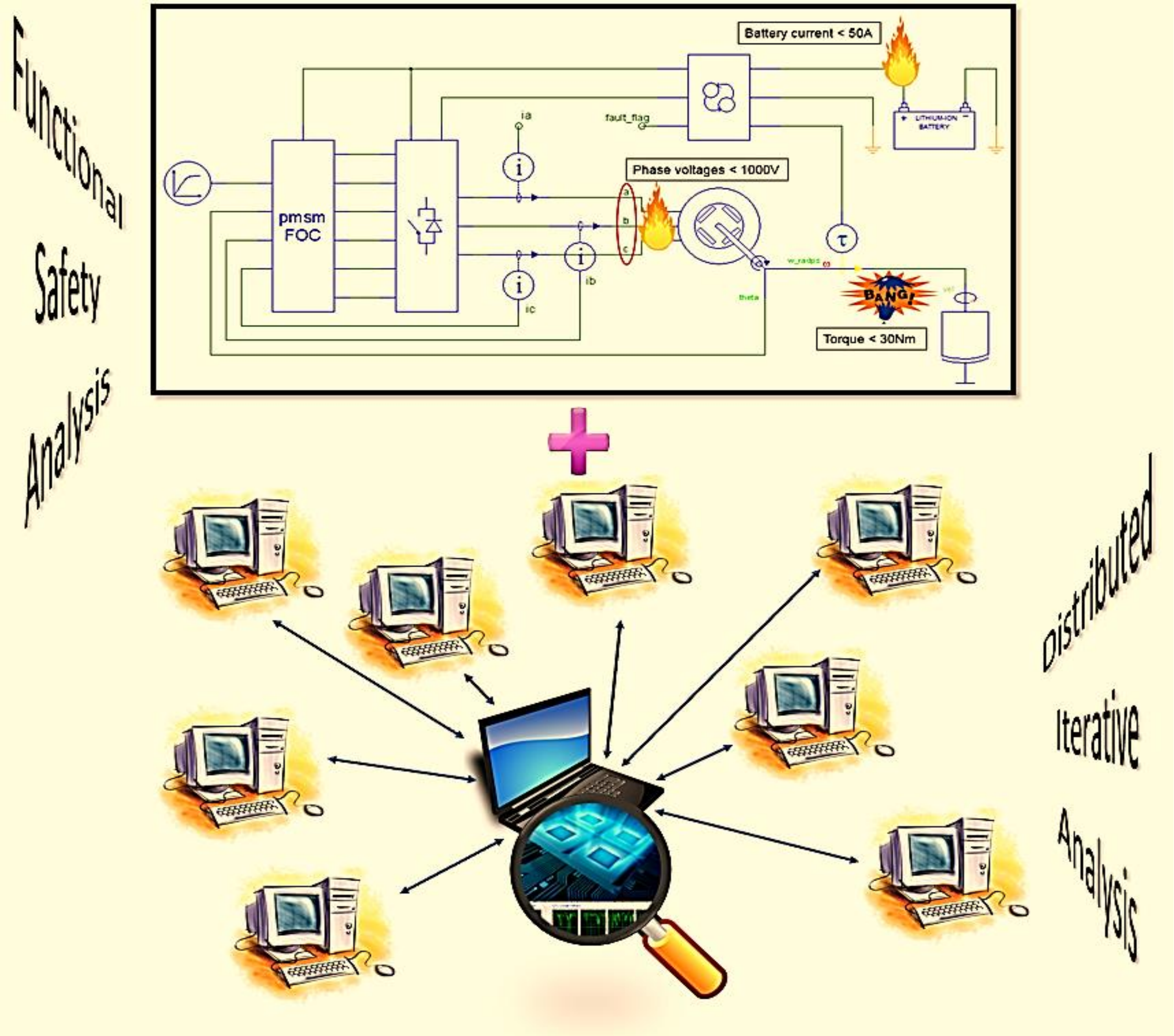


Functional Safety Analysis for a Motor Drive Design

SaberRD Design Example

Synopsys Inc. 700 East Middlefield Road, Mountain View, CA 94043 www.synopsys.com

Functional Safety Analysis for a Motor Drive Design

1 Introduction

The Saber design example describes a torque-controlled motor drive (Figure 1). When running a transient analysis, the motor will apply a torque of 10Nm to the inertia load, given by the torque reference. To get familiar with the design behavior, just open “motor_drive_FS.ai_dsn” in SaberRD and run the Experiment “Nominal_Behavior”. Have a look at the results (Figure 2).

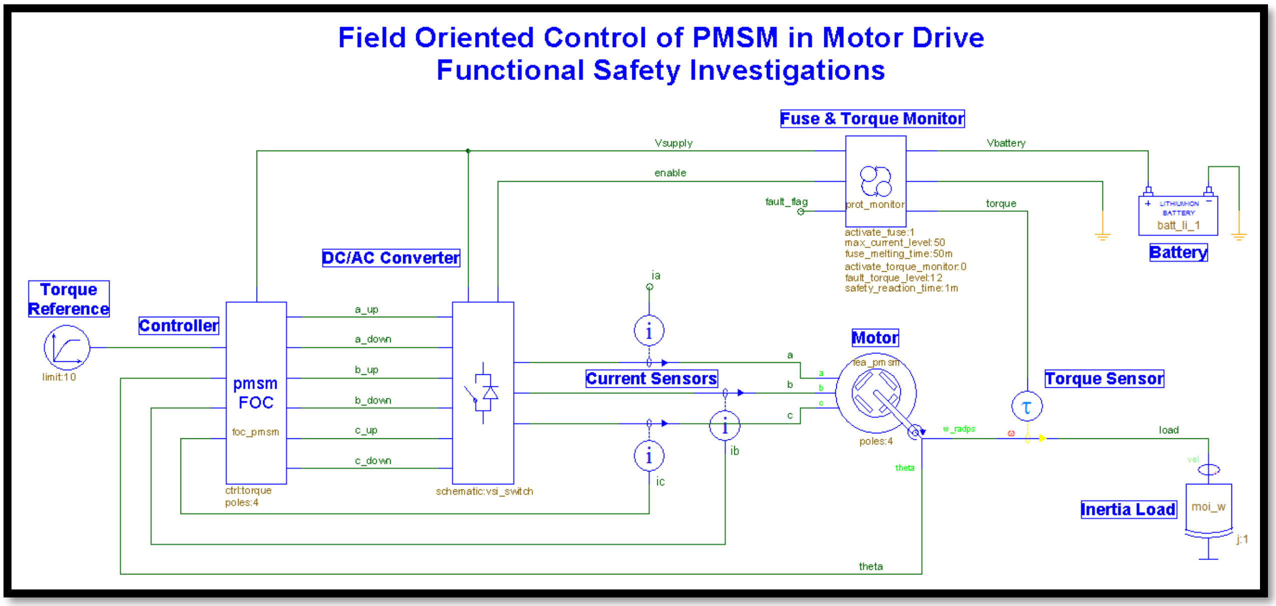
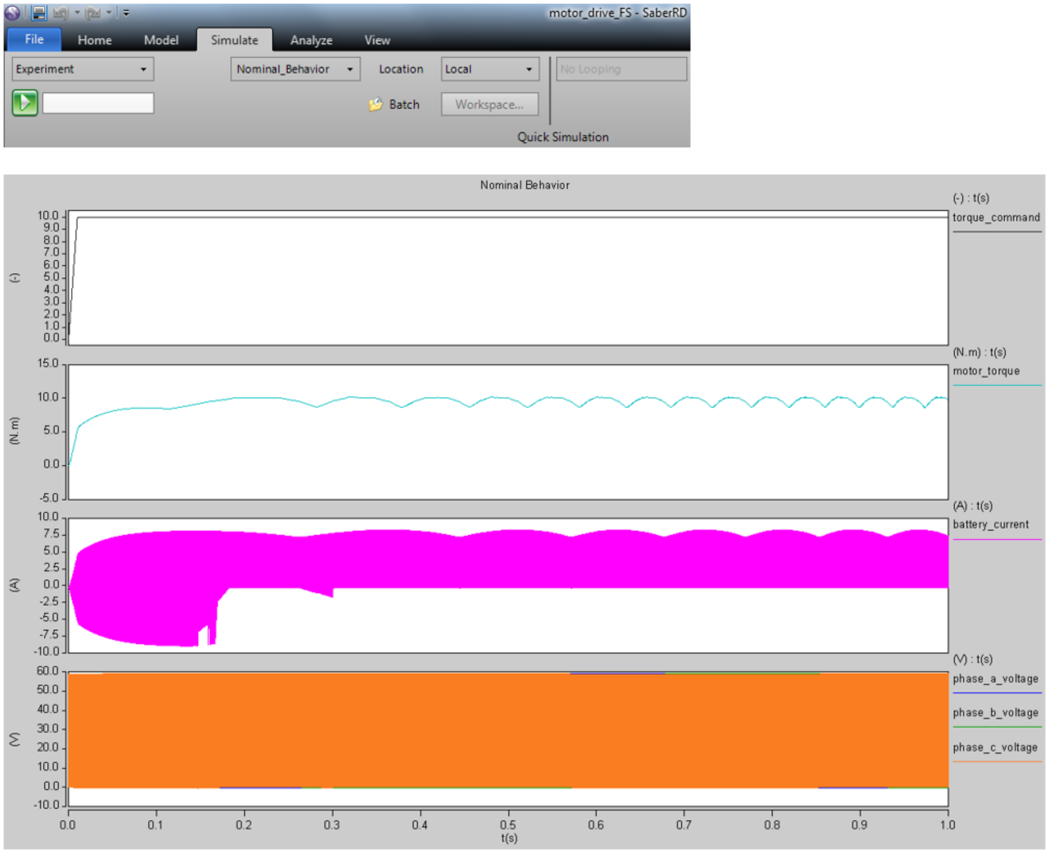


Figure 1: Motor Drive Design

The permanent magnet synchronous motor is powered by a bridge (DC/AC Converter) with ideal switches and diodes. The control-signals for the bridge are generated by the FOC-block (field oriented control), that uses motor-angle and motor-current information as input signals. A battery supplies the system. A fuse protects the motor drive. Additionally, a Torque Monitor can be activated that uses the information from a torque-sensor.

This design example details the use of Functional Safety Analysis solution available in SaberRD. Several faults are applied on the design and their impact on safety is evaluated. When a fault appears safety critical, design changes can be made to mitigate such faults during the design phase through simulation. In addition, the Distributed Iterative Analysis feature is demonstrated to increase productivity by reducing the simulation time by paralleling the simulation runs.


Experiment: Nominal_Behavior

Location: Local

No Looping

Figure 2: Experiment Results for Nominal Behavior

2 Functional Safety

2.1 Motivation

By using SaberRD’s Fault Analysis, various hardware- and software-faults will be injected in the motor drive system. The resulting fault-effects will be analyzed and tested if they violate safety criteria.
In case of violation of safety criteria, a mechanism will be developed to detect injected faults and to bring back the system into a safe state.

Fault Analysis is an iterative analysis where each fault is looped and the required analysis is performed in each loop. The DIA feature in SaberRD can be used to run various iterations of the iterative analysis in parallel reducing the total time taken for simulation. The simulation engine can use the different cores in the processor of the workstation or the grid, that contains several computers configured as grid.

2.2 SAFETY CRITERIA

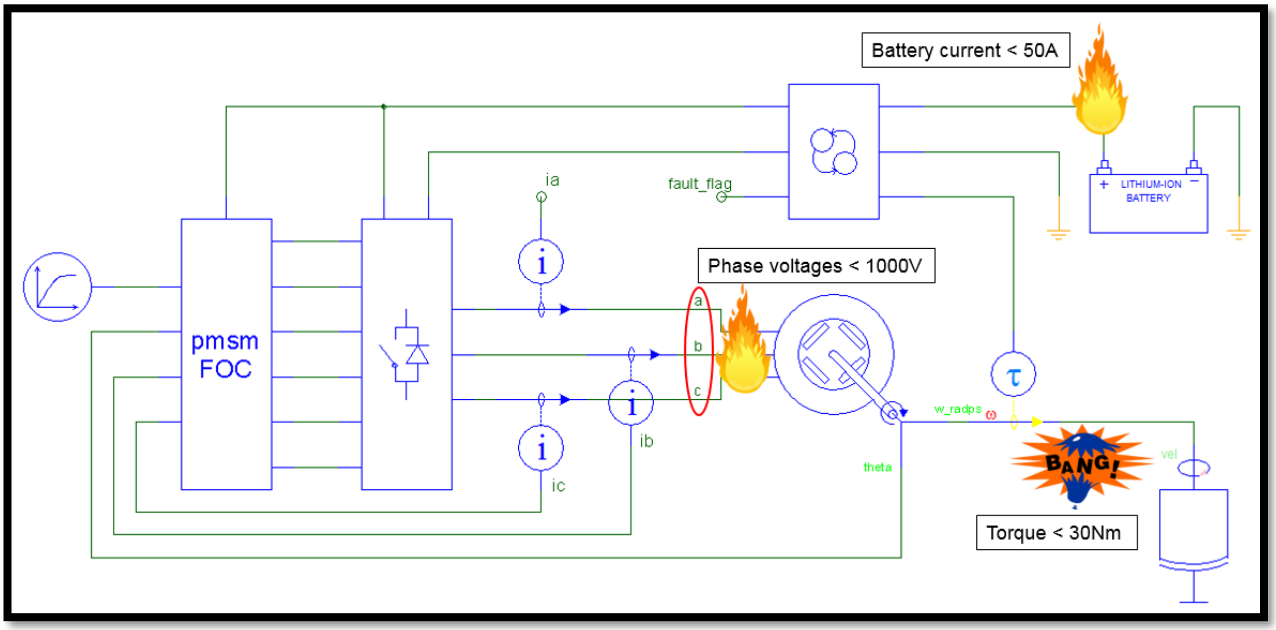


Figure 3: Safety Criteria

For the motor drive example, 3 safety criteria are defined (Figure 3):

1. Faults have to be detected and the system must be shut down.
2. Motor-torque<30Nm.
   The motor-torque that is applied to the load must not exceed 30Nm all the time (from fault injection until system shutdown).
3. Phase-voltage<1000V.
   Voltages, which are measured on motor-phase pins, must not exceed 1000V (protection against destruction of components and electrical shock).

In the motor drive design, the following signals are monitored to detect faults:

- Battery-current: is “measured” by fuse-model. In case of an overcurrent, the fuse will melt and open the circuitry after the fuse-melting-time of 50ms. The fault is then considered to be detected.
- Motor-torque: is calculated basing on motor-currents in FOC-block. In case of an activated torque-monitor, the torque is additionally measured and observed by a torque-sensor.
- Phase-voltages: are not actively monitored. The voltages will only be measured and tested after the simulation was done (postprocessing).

2.3 Faults to Be Investigated

There are 7 faults that will be injected in the motor drive system (Figure 4 and Figure 5). To define the faults, SaberRD’s Fault Tool is used.

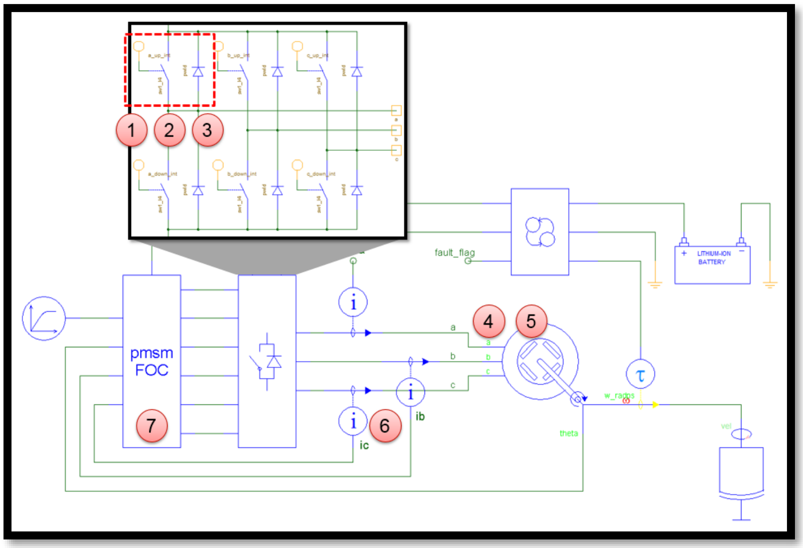
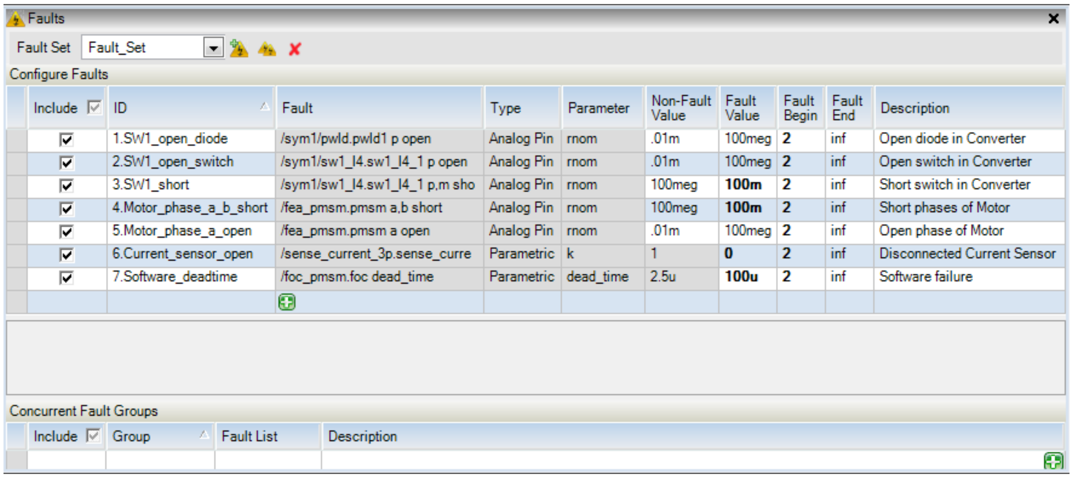


Figure 4: Injected faults in motor drive design



	Include ☑	ID	Fault	Type	Parameter	Non-Fault Value	Fault Value	Fault Begin	Fault End	Description
	☑	1.SW1_open_diode	/sym1/pwld.pwld1 p open	Analog Pin	rnom	.01m	100meg	**2**	inf	Open diode in Converter
	☑	2.SW1_open_switch	/sym1/sw1_l4.sw1_l4_1 p open	Analog Pin	rnom	.01m	100meg	**2**	inf	Open switch in Converter
	☑	3.SW1_short	/sym1/sw1_l4.sw1_l4_1 p,m sho	Analog Pin	rnom	100meg	**100m**	**2**	inf	Short switch in Converter
	☑	4.Motor_phase_a_b_short	/fea_pmsm.pmsm a,b short	Analog Pin	rnom	100meg	**100m**	**2**	inf	Short phases of Motor
	☑	5.Motor_phase_a_open	/fea_pmsm.pmsm a open	Analog Pin	rnom	.01m	100meg	**2**	inf	Open phase of Motor
	☑	6.Current_sensor_open	/sense_current_3p.sense_curre	Parametric	k	1	**0**	**2**	inf	Disconnected Current Sensor
	☑	7.Software_deadtime	/foc_pmsm.foc dead_time	Parametric	dead_time	2.5u	**100u**	**2**	inf	Software failure

	Include ☑	Group	Fault List	Description

Figure 5: SaberRD’s Fault Tool

All faults are activated after 2s and last till the end of the simulation. The first 5 faults are hardware faults, where pins in the design are either opened or shorted. Fault number 6 describes a broken current-sensor that transmits the information of 0A to the control-block. The last fault in the list is a software-fault. There is a dead time defined for the bridge. This parameter represents the time between opening a high-side switch and closing a low-side switch (and vice-versa). Due to an issue in the software, this value is changed from 2.5µs to 100µs.

3 FAULT ANALYSIS IN SABERRD

3.1 FAULT SIMULATION WITH FUSE

In a next step, we activate the 7 faults at simulation-time 2s above described. Each fault run takes 2.2 seconds. The focus is on the design behavior after fault injection and on the automatic check for violation of safety criteria by using Experiment Analyzer in SaberRD.

The only activated safety mechanism is the fuse: if the battery current exceeds 50A, the fuse will open the circuitry after 50ms. The optional torque-monitor is disabled. This functionality is described in the "Fuse & Torque Monitor" block (Figure 6). Additionally, this block has a pin called "fault_flag" that turns into "1" if the circuitry was shut down (by fuse or later by torque-monitor).

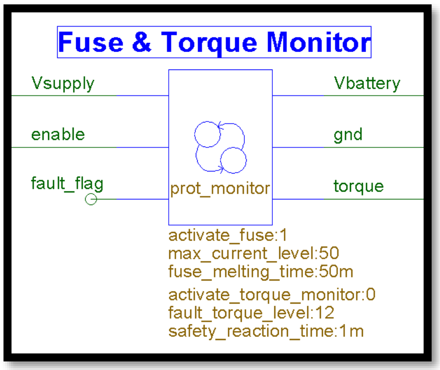
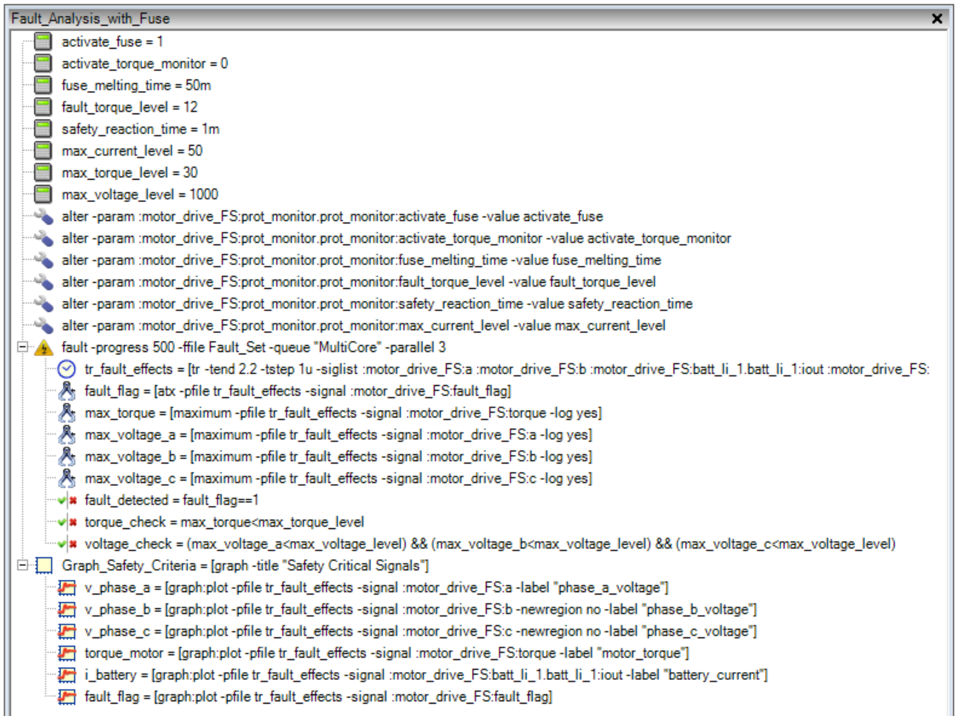


Figure 6: Safety mechanism Fuse

Please run the Experiment "Fault_Analysis_with_Fuse" (Figure 7).

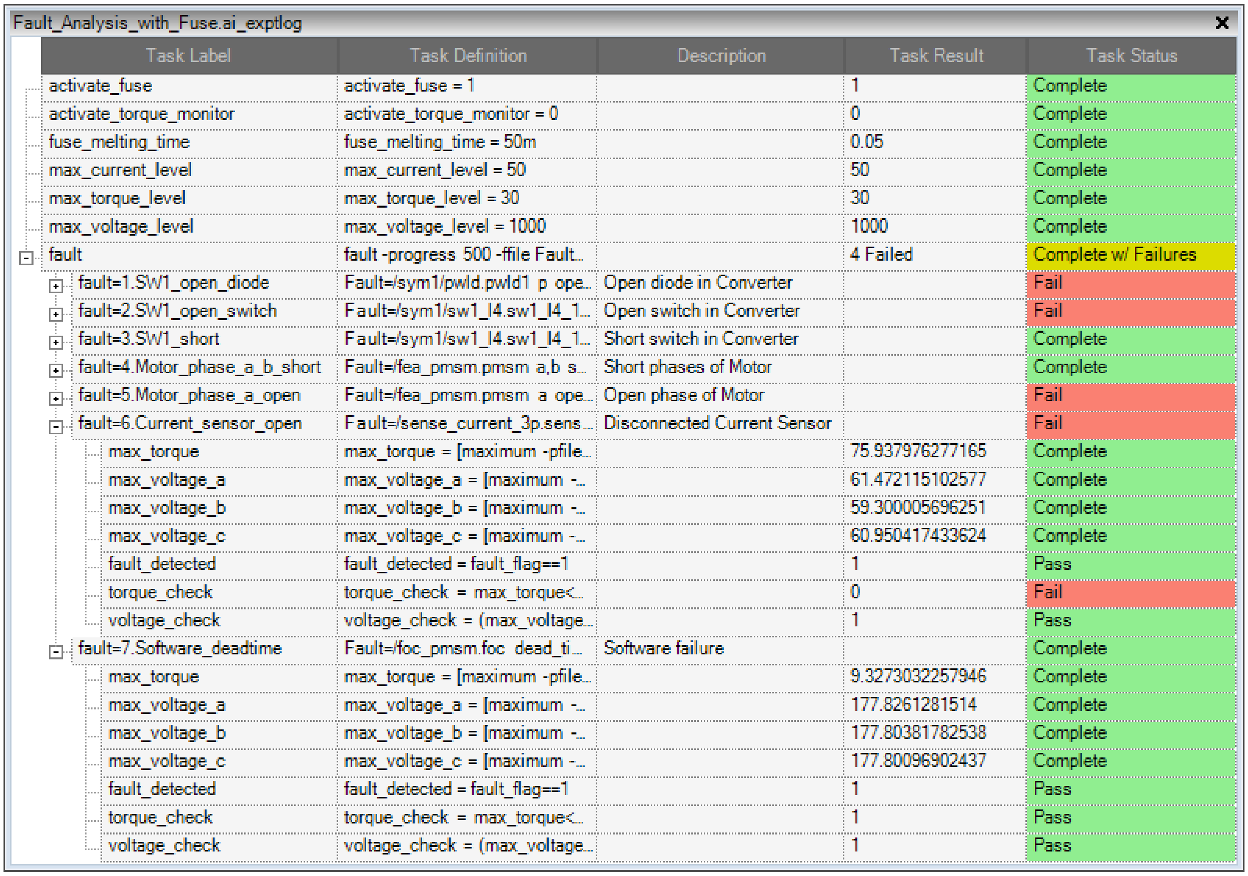


```
Fault_Analysis_with_Fuse
activate_fuse = 1
activate_torque_monitor = 0
fuse_melting_time = 50m
fault_torque_level = 12
safety_reaction_time = 1m
max_current_level = 50
max_torque_level = 30
max_voltage_level = 1000
alter -param :motor_drive_FS:prot_monitor.prot_monitor:activate_fuse -value activate_fuse
alter -param :motor_drive_FS:prot_monitor.prot_monitor:activate_torque_monitor -value activate_torque_monitor
alter -param :motor_drive_FS:prot_monitor.prot_monitor:fuse_melting_time -value fuse_melting_time
alter -param :motor_drive_FS:prot_monitor.prot_monitor:fault_torque_level -value fault_torque_level
alter -param :motor_drive_FS:prot_monitor.prot_monitor:safety_reaction_time -value safety_reaction_time
alter -param :motor_drive_FS:prot_monitor.prot_monitor:max_current_level -value max_current_level
fault -progress 500 -ffile Fault_Set -queue "MultiCore" -parallel 3
    tr_fault_effects = [tr -tend 2.2 -tstep 1u -siglist :motor_drive_FS:a :motor_drive_FS:b :motor_drive_FS:batt_li_1.batt_li_1:iout :motor_drive_FS:
    fault_flag = [atx -pfile tr_fault_effects -signal :motor_drive_FS:fault_flag]
    max_torque = [maximum -pfile tr_fault_effects -signal :motor_drive_FS:torque -log yes]
    max_voltage_a = [maximum -pfile tr_fault_effects -signal :motor_drive_FS:a -log yes]
    max_voltage_b = [maximum -pfile tr_fault_effects -signal :motor_drive_FS:b -log yes]
    max_voltage_c = [maximum -pfile tr_fault_effects -signal :motor_drive_FS:c -log yes]
    fault_detected = fault_flag==1
    torque_check = max_torque<max_torque_level
    voltage_check = (max_voltage_a<max_voltage_level) && (max_voltage_b<max_voltage_level) && (max_voltage_c<max_voltage_level)
Graph_Safety_Criteria = [graph -title "Safety Critical Signals"]
    v_phase_a = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:a -label "phase_a_voltage"]
    v_phase_b = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:b -newregion no -label "phase_b_voltage"]
    v_phase_c = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:c -newregion no -label "phase_c_voltage"]
    torque_motor = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:torque -label "motor_torque"]
    i_battery = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:batt_li_1.batt_li_1:iout -label "battery_current"]
    fault_flag = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:fault_flag]
```

Figure 7: Experiment "Fault_Analysis_with_Fuse"

The Experiment parameterizes the “Fuse & Torque Monitor” block. The fuse is activated, the torque-monitor disabled. The fuse opens the supply line after “fuse_melting_time=50m” if “max_current_level=50” is exceeded. This addresses the first safety criteria to make sure that the current is less than 50A. For the other two safety criteria, “max_torque_level” is set to 30 (Nm) and “max_voltage_level” is set to 1000 (V).

The fault analysis is set with parallel runs to save simulation time. At the end of each fault run, the motor-torque, phase-voltages, and the fault-flag are measured and tested. The results are shown in Experiment Report (Figure 8). Additionally, a Graph (Figure 9) is generated showing typical signals, like phase-voltages, motor-torque, battery-current and fault-flag.



Fault_Analysis_with_Fuse.ai_exptlog

Task Label	Task Definition	Description	Task Result	Task Status
activate_fuse	activate_fuse = 1		1	Complete
activate_torque_monitor	activate_torque_monitor = 0		0	Complete
fuse_melting_time	fuse_melting_time = 50m		0.05	Complete
max_current_level	max_current_level = 50		50	Complete
max_torque_level	max_torque_level = 30		30	Complete
max_voltage_level	max_voltage_level = 1000		1000	Complete
fault	fault -progress 500 -ffile Fault...		4 Failed	Complete w/ Failures
fault=1.SW1_open_diode	Fault=/sym1/pwld.pwld1 p ope...	Open diode in Converter		Fail
fault=2.SW1_open_switch	Fault=/sym1/sw1_l4.sw1_l4_1...	Open switch in Converter		Fail
fault=3.SW1_short	Fault=/sym1/sw1_l4.sw1_l4_1...	Short switch in Converter		Complete
fault=4.Motor_phase_a_b_short	Fault=/fea_pmsm.pmsm a,b s...	Short phases of Motor		Complete
fault=5.Motor_phase_a_open	Fault=/fea_pmsm.pmsm a ope...	Open phase of Motor		Fail
fault=6.Current_sensor_open	Fault=/sense_current_3p.sens...	Disconnected Current Sensor		Fail
max_torque	max_torque = [maximum -pfile...		75.937976277165	Complete
max_voltage_a	max_voltage_a = [maximum -...		61.472115102577	Complete
max_voltage_b	max_voltage_b = [maximum -...		59.300005696251	Complete
max_voltage_c	max_voltage_c = [maximum -...		60.950417433624	Complete
fault_detected	fault_detected = fault_flag==1		1	Pass
torque_check	torque_check = max_torque<...		0	Fail
voltage_check	voltage_check = (max_voltage...		1	Pass
fault=7.Software_deadtime	Fault=/foc_pmsm.foc dead_ti...	Software failure		Complete
max_torque	max_torque = [maximum -pfile...		9.3273032257946	Complete
max_voltage_a	max_voltage_a = [maximum -...		177.8261281514	Complete
max_voltage_b	max_voltage_b = [maximum -...		177.80381782538	Complete
max_voltage_c	max_voltage_c = [maximum -...		177.80096902437	Complete
fault_detected	fault_detected = fault_flag==1		1	Pass
torque_check	torque_check = max_torque<...		1	Pass
voltage_check	voltage_check = (max_voltage...		1	Pass

Figure 8: Experiment Report of “Fault_Analysis_with_Fuse”

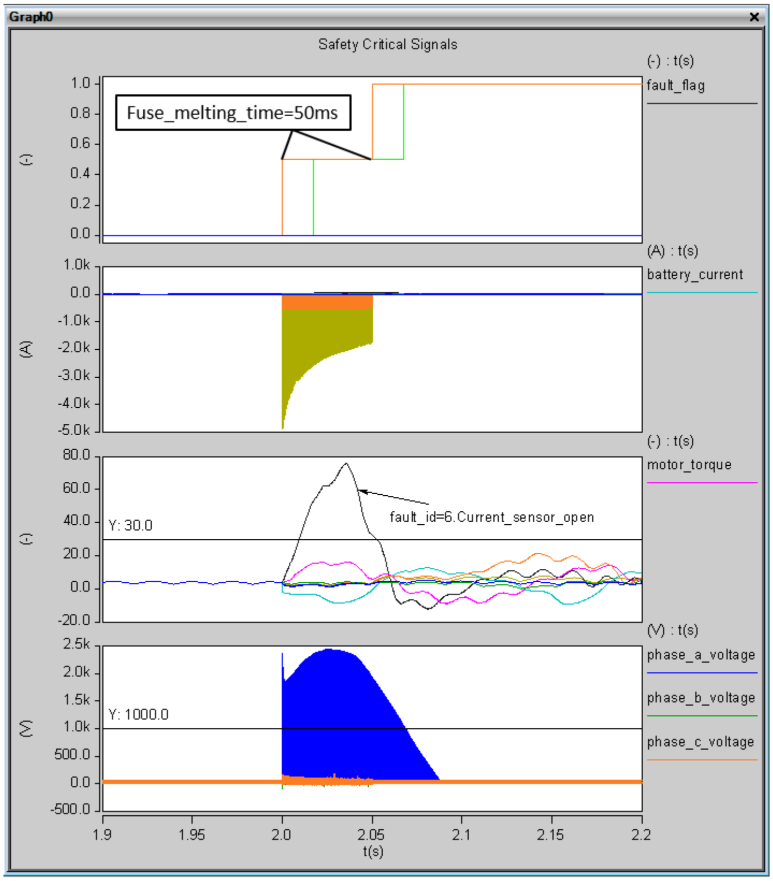


Figure 9: Graph of Experiment "Fault_Analysis_with_Fuse"

In Experiment Report, 4 of 7 faults violate safety criteria. As an example, the maximum motor-torque for faults 6 is 75.94Nm that exceeds the allowed 30Nm. Phase-voltages are within the safe range. Fault 6 is detected by a melted fuse. Nevertheless, this fault violates functional safety due to overtorque.

Fault 7 as another example is also detected by an overcurrent (fuse). The motor-torque and phase-voltages are within the allowed range. This fault fulfills safety criteria completely.

3.1.1 Summary:

Fault	Comply with safety criteria	Comments
Fault 1: Open diode in Converter	No	Overvoltage No detection of the fault
Fault 2: Open switch in Converter	No	No detection of the fault
Fault 3: Short switch in Converter	Yes	
Fault 4: Short phases of Motor	Yes	
Fault 5: Open phase of Motor	No	No detection of the fault
Fault 6: Disconnected Current Sensor	No	Overtorque
Fault 7: Software failure	Yes	

Table 1: Summary of fault analysis with fuse-protection

The table (Table 1) shows a summary of the fault simulation using the Experiment “Fault_Analysis_with_Fuse”.

The most critical fault is fault 6, which describes a broken current sensor. To protect the system against over-torque, a new safety mechanism is required.

Fault 1 shows an overvoltage. This effect needs more investigations: a change of the abstraction-level for the wiring and power-switches in the design to gain more precise simulation-results. Afterwards, EMC-components could be added. Overvoltage investigations will not be covered by this design-example.

Fault 2 is not detected but this is not a safety critical fault.

3.2 Safety Mechanism: Torque Monitor

The results of the Experiment “Fault_Analysis_with_Fuse” have shown that an unexpected high torque is applied to the load in case of a broken current-sensor (fault 6). The current-information is used by the FOC-block to control the motor-torque to exact 10Nm. The resulting fault-effect is an overtorque of 75.94Nm, which completely violates the safety criteria of 30Nm.

To overcome this safety issue, redundancy should be introduced to the motor drive system. An additional torque-sensor will measure the motor-torque and provide this information to the torque-monitor (Figure 10).

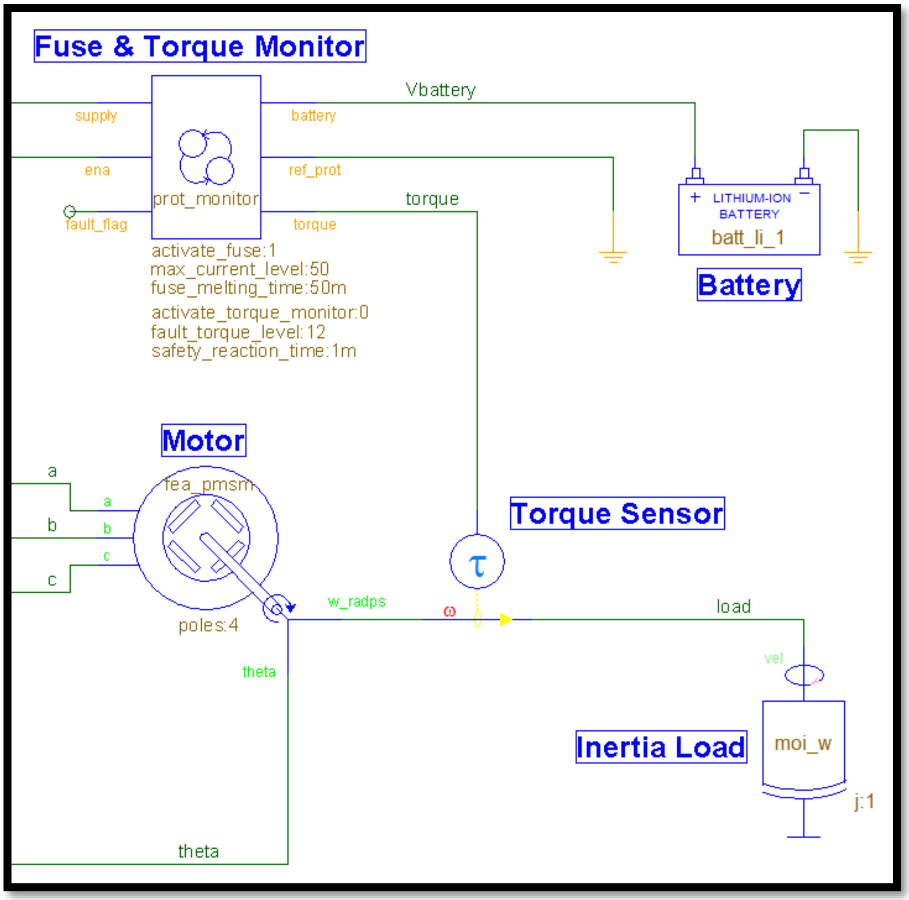


Figure 10: Safety mechanism: torque-sensor and torque-monitor

The torque-monitor consists of a µC that is capable to turn-off the system much faster than the fuse could do. If a torque is detected that is larger than 12Nm, the circuitry will open after 1ms. This means that 12Nm is the torque-threshold when a fault is detected and the shutdown is initiated. At the end, the motor-torque must not exceed the safety limit of 30Nm.

The torque-monitor is modeled in SaberRD's modeling-tool called StateAMS (Figure 11). Model-parameters related to the torque-monitor are:

- activate_torque_monitor (0: disabled, 1: activated)
- fault_torque_level=12 (Nm)
- safety_reaction_time=1m (s)

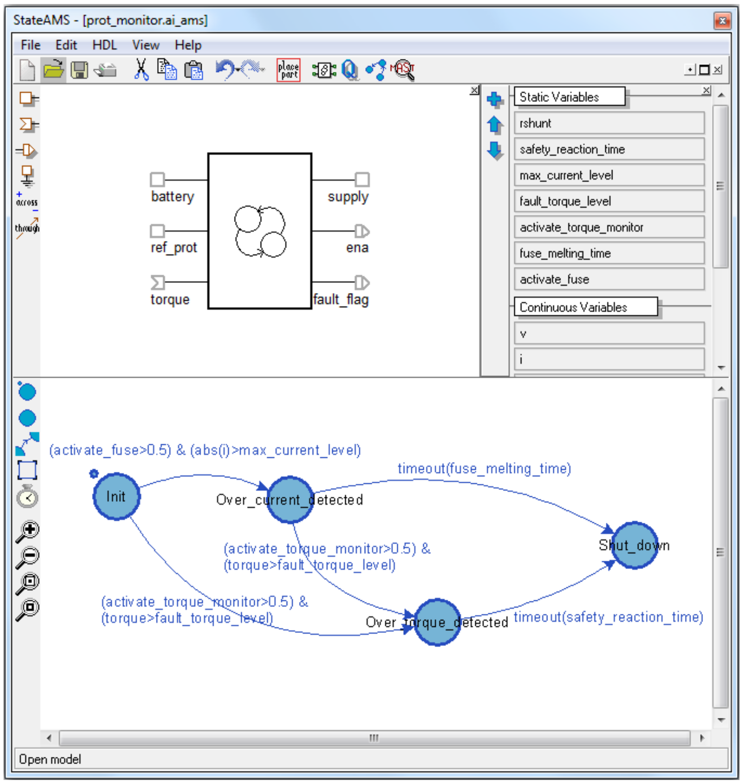


Figure 11: StateAMS-model for "Fuse & Torque Monitor" block

The simulation always starts in the "Init"-state that describes a low-ohmic connection across the pins "battery" and "supply".

The upper path covers the fuse functionality. In case of an activated fuse (activate_fuse>0.5), the current through the supply-pin is monitored. If the current exceeds the value of parameter "max_current_level", the "Over_current_detected"-state gets active and the signal "fault-flag" changes from 0 to 0.5. This state starts a timer, which will shut down the system after time "fuse_melting_time".

The lower path from state "Init" describes the torque-monitor. In case of an activated torque-monitor (activate_torque_monitor>0.5) and a higher torque-sensor signal than "fault_torque_level", the

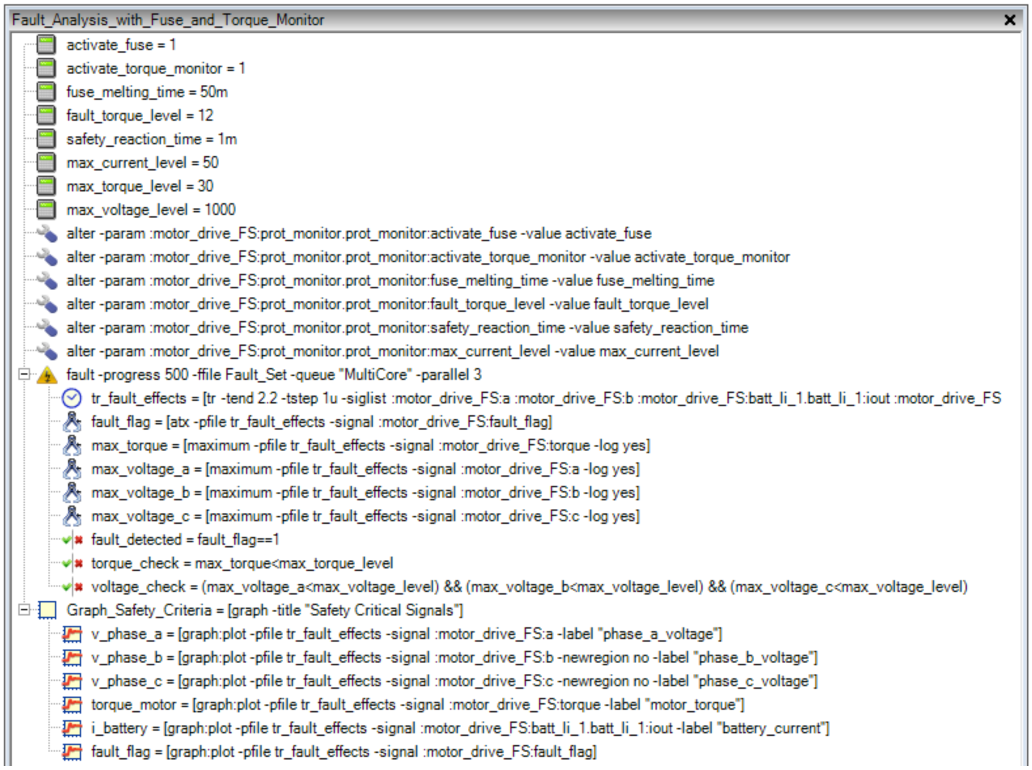
"Over_torque_detected"-state gets active and the signal "fault-flag" changes from 0 to 0.5. This state also starts a timer, which will shut down the system after time "safety_reaction_time".

The shutdown is modeled in the "Shut_down"-state by a high-ohmic connection across the pins "battery" and "supply". The signal "fault-flag" is set to 1.

In next section, the new safety mechanism will be tested.

3.3 Fault Simulation With Fuse And Torque Monitor

In this step, we will repeat the fault-simulation, but this time with activated fuse and torque-monitor. Please run the Experiment "Fault_Analysis_with_Fuse_and_Torque_Monitor" (Figure 12).



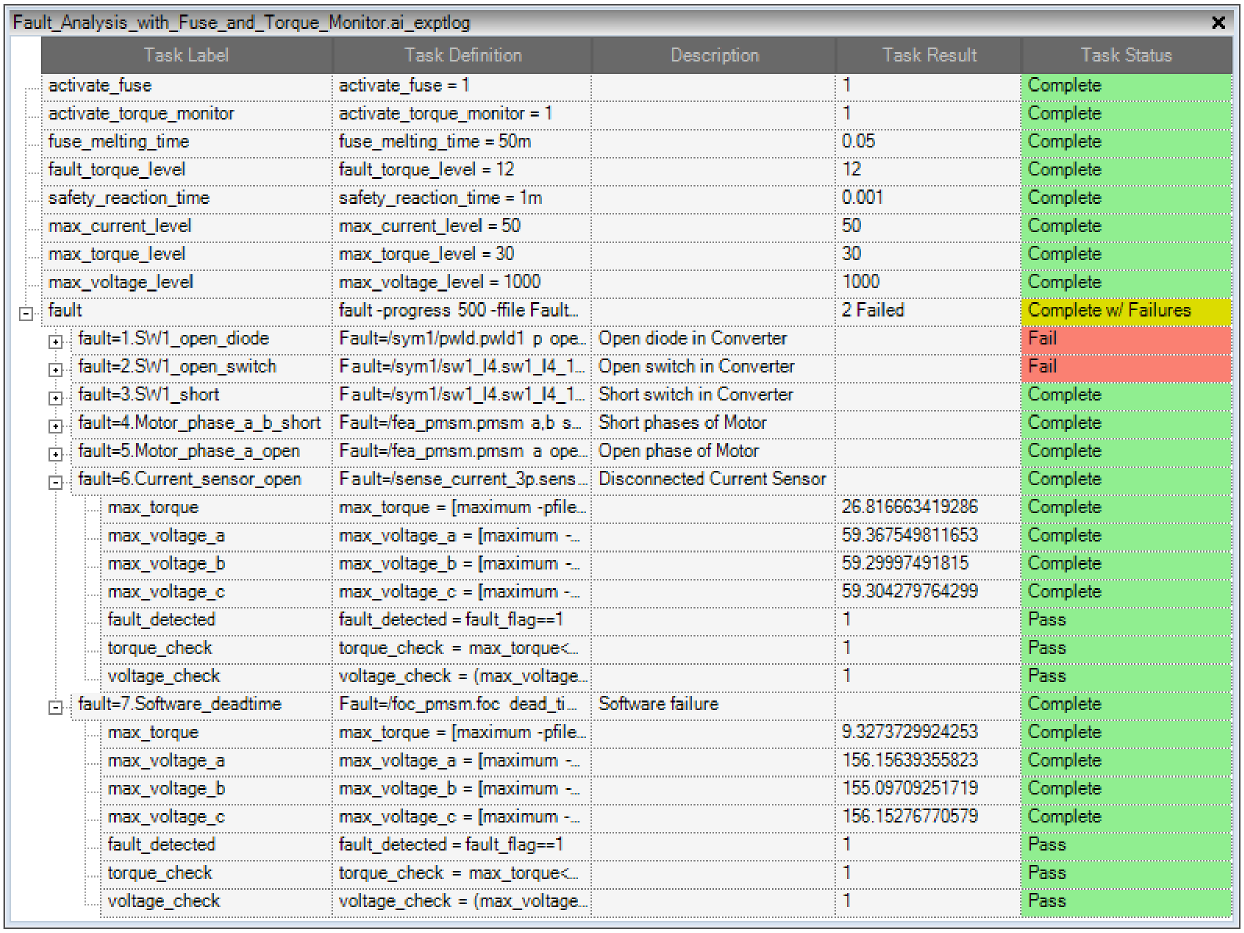
```
activate_fuse = 1
activate_torque_monitor = 1
fuse_melting_time = 50m
fault_torque_level = 12
safety_reaction_time = 1m
max_current_level = 50
max_torque_level = 30
max_voltage_level = 1000
alter -param :motor_drive_FS:prot_monitor.prot_monitor:activate_fuse -value activate_fuse
alter -param :motor_drive_FS:prot_monitor.prot_monitor:activate_torque_monitor -value activate_torque_monitor
alter -param :motor_drive_FS:prot_monitor.prot_monitor:fuse_melting_time -value fuse_melting_time
alter -param :motor_drive_FS:prot_monitor.prot_monitor:fault_torque_level -value fault_torque_level
alter -param :motor_drive_FS:prot_monitor.prot_monitor:safety_reaction_time -value safety_reaction_time
alter -param :motor_drive_FS:prot_monitor.prot_monitor:max_current_level -value max_current_level
fault -progress 500 -ffile Fault_Set -queue "MultiCore" -parallel 3
    tr_fault_effects = [tr -tend 2.2 -tstep 1u -siglist :motor_drive_FS:a :motor_drive_FS:b :motor_drive_FS:batt_li_1.batt_li_1:iout :motor_drive_FS
    fault_flag = [atx -pfile tr_fault_effects -signal :motor_drive_FS:fault_flag]
    max_torque = [maximum -pfile tr_fault_effects -signal :motor_drive_FS:torque -log yes]
    max_voltage_a = [maximum -pfile tr_fault_effects -signal :motor_drive_FS:a -log yes]
    max_voltage_b = [maximum -pfile tr_fault_effects -signal :motor_drive_FS:b -log yes]
    max_voltage_c = [maximum -pfile tr_fault_effects -signal :motor_drive_FS:c -log yes]
    fault_detected = fault_flag==1
    torque_check = max_torque<max_torque_level
    voltage_check = (max_voltage_a<max_voltage_level) && (max_voltage_b<max_voltage_level) && (max_voltage_c<max_voltage_level)
Graph_Safety_Criteria = [graph -title "Safety Critical Signals"]
    v_phase_a = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:a -label "phase_a_voltage"]
    v_phase_b = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:b -newregion no -label "phase_b_voltage"]
    v_phase_c = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:c -newregion no -label "phase_c_voltage"]
    torque_motor = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:torque -label "motor_torque"]
    i_battery = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:batt_li_1.batt_li_1:iout -label "battery_current"]
    fault_flag = [graph:plot -pfile tr_fault_effects -signal :motor_drive_FS:fault_flag]
```

Figure 12: Experiment "Fault_Analysis_with_Fuse_and_Torque_Monitor"

The Experiment parameterizes the "Fuse & Torque Monitor" block. The fuse and torque-monitor are activated. The torque-monitor opens the supply line after "safety_reaction_time=1m" if "fault_torque_level=12" is exceeded. Additionally, a melting fuse can turn off the system.

The fault analysis is set with parallel runs to save simulation time.

At the end of each fault run, the motor-torque, phase-voltages, and the fault-flag are measured and tested. The results are shown in Experiment Report (Figure 13). Additionally, a Graph (Figure 14) is generated showing typical signals, like phase-voltages, motor-torque, battery-current and fault-flag.



Fault_Analysis_with_Fuse_and_Torque_Monitor.ai_exptlog

Task Label	Task Definition	Description	Task Result	Task Status
activate_fuse	activate_fuse = 1		1	Complete
activate_torque_monitor	activate_torque_monitor = 1		1	Complete
fuse_melting_time	fuse_melting_time = 50m		0.05	Complete
fault_torque_level	fault_torque_level = 12		12	Complete
safety_reaction_time	safety_reaction_time = 1m		0.001	Complete
max_current_level	max_current_level = 50		50	Complete
max_torque_level	max_torque_level = 30		30	Complete
max_voltage_level	max_voltage_level = 1000		1000	Complete
fault	fault -progress 500 -ffile Fault...		2 Failed	Complete w/ Failures
fault=1.SW1_open_diode	Fault=/sym1/pwld.pwld1 p ope...	Open diode in Converter		Fail
fault=2.SW1_open_switch	Fault=/sym1/sw1_l4.sw1_l4_1...	Open switch in Converter		Fail
fault=3.SW1_short	Fault=/sym1/sw1_l4.sw1_l4_1...	Short switch in Converter		Complete
fault=4.Motor_phase_a_b_short	Fault=/fea_pmsm.pmsm a,b s...	Short phases of Motor		Complete
fault=5.Motor_phase_a_open	Fault=/fea_pmsm.pmsm a ope...	Open phase of Motor		Complete
fault=6.Current_sensor_open	Fault=/sense_current_3p.sens...	Disconnected Current Sensor		Complete
max_torque	max_torque = [maximum -pfile...		26.816663419286	Complete
max_voltage_a	max_voltage_a = [maximum -...		59.367549811653	Complete
max_voltage_b	max_voltage_b = [maximum -...		59.29997491815	Complete
max_voltage_c	max_voltage_c = [maximum -...		59.304279764299	Complete
fault_detected	fault_detected = fault_flag==1		1	Pass
torque_check	torque_check = max_torque<...		1	Pass
voltage_check	voltage_check = (max_voltage...		1	Pass
fault=7.Software_deadtime	Fault=/foc_pmsm.foc dead_ti...	Software failure		Complete
max_torque	max_torque = [maximum -pfile...		9.3273729924253	Complete
max_voltage_a	max_voltage_a = [maximum -...		156.15639355823	Complete
max_voltage_b	max_voltage_b = [maximum -...		155.09709251719	Complete
max_voltage_c	max_voltage_c = [maximum -...		156.15276770579	Complete
fault_detected	fault_detected = fault_flag==1		1	Pass
torque_check	torque_check = max_torque<...		1	Pass
voltage_check	voltage_check = (max_voltage...		1	Pass

Figure 13: Experiment Report of "Fault_Analysis_with_Fuse_and_Torque_Monitor"

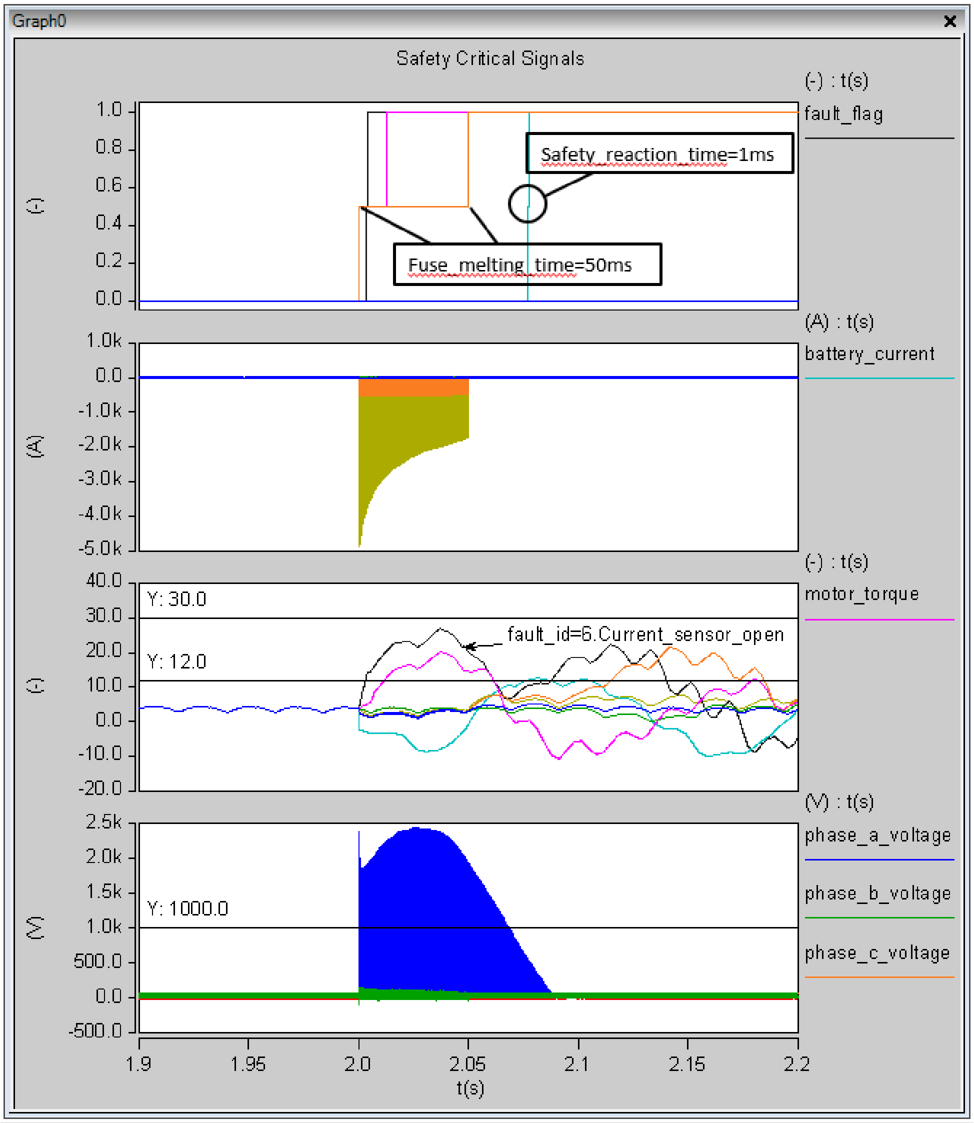


Figure 14: Graph of Experiment "Fault_Analysis_with_Fuse_and_Torque_Monitor"

In Experiment Report, only 2 of 7 faults still violate safety criteria (fault 1 and 2).

For fault 5 (open motor-pin), the "fault_torque_level" of 12Nm is exceeded and recognized by the torque-monitor. That makes this fault detectable and the system is shut-down.

For the most critical fault 6, the broken current-sensor, the torque-monitor also turns off the system much faster than a fuse could do. The resulting maximum motor-torque is 26.82Nm instead of 75.94Nm. With activated torque-monitor, both faults are detected and the system remains in a safe state all the time.

3.3.1 Summary:

Fault	Comply with safety criteria	Comments
Fault 1: Open diode in Converter	No	Overvoltage No detection of the fault
Fault 2: Open switch in Converter	No	No detection of the fault
Fault 3: Short switch in Converter	Yes	
Fault 4: Short phases of Motor	Yes	
Fault 5: Open phase of Motor	Yes	
Fault 6: Disconnected Current Sensor	Yes	
Fault 7: Software failure	Yes	

Table 2: Summary of Fault Analysis with Fuse Protection and Torque Monitor

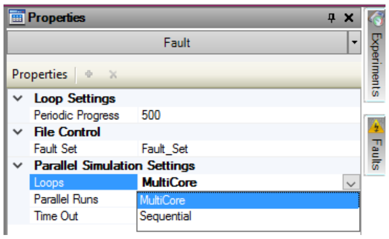
Table 2 shows a summary of the fault simulation using the Experiment “Fault_Analysis_with_Fuse_and_Torque_Monitor”.

Fault 1 shows an overvoltage. This effect needs more investigations: a change of the abstraction-level for the wiring and power-switches in the design to gain more precise simulation-results. Afterwards, EMC-components could be added. Overvoltage investigations will not be covered by this design-example.

Fault 2 is not detected but this is not a safety critical fault.

3.4 Advantage of Using Distributed Iterative Analysis (DIA)

The cores available in a multicore computer can be leveraged to speed up the simulation time for iterative analyses by using the DIA feature in Saber. As shown in the previous sections, the fault loop is set to run fault simulations in parallel using the DIA feature to save the simulation time. The DIA feature settings are available in the “Parallel Simulation Settings” as shown in Figure 15.



Loop Settings
Periodic Progress 500
File Control
Fault Set Fault_Set
Parallel Simulation Settings
Loops MultiCore
Parallel Runs
Time Out

Figure 15: Parallel Simulation Settings in Experiment Analyzer

Loops (queue)	Parallel Runs (parallel)	CPU Time Taken	% Speed Improvement (w.r.t. Sequential run)
Sequential	---	9 min, 25 secs.	---
Multicore	2	5 min, 54 secs.	37%
Multicore	4	3 min, 37 secs.	63%

Note: These results are from performing the simulations in a quad core machine (No of Cores = 4).

Table 3: CPU (Simulation) Time Reduction with DIA

The results shown in Table 3 can be verified by running the experiment Fault_Analysis_with_Fuse with settings mentioned in the table. When the simulation is performed with DIA, it can be observed that there is an overhead time for distributing the runs in parallel and in collating the results. Due to this overhead, the improvement in the total simulation time would be more prominent for the designs that have iterative analyses each with long run time. For such designs, if you want to increase the simulation speed beyond what the multicore PC can achieve, you can consider setting up a separate grid for DIA.

4 CONCLUSION

Functional Safety is increasingly important in development processes of innovative products in the industry, i.e. automotive industry.

SaberRD support standards like ISO26262 by Fault Analysis that helps to analyze fault effects in an automated way. Additionally, safety mechanism can be developed and validated by using for instance SaberRD's Modeling Tools, like StateAMS. The shown motor drive example has illustrated how fault-effects can be analyzed and checked against safety criteria in SaberRD. Basing on the results, a safety mechanism was developed and tested. With the use of Distributed Iterative Analysis feature, the simulation time is reduced drastically for iterative analyses. Hence, the improvement in simulation performance is also demonstrated.

5 REFERENCES

- International Standard ISO 26262 Functional Safety
  ISO copyright office Geneva, Switzerland, 2011