



Fault Injection for Multi Chip Simulation of Battery Management System

Frank Poppen

System Modelling Engineer
BL-AA | DES/BMS | Hamburg



Outline

- Motivation
- High Voltage Battery Management System
- Full System Simulation Setup
- Fault Injection
- Conclusion and Future Work

Motivation



Motivation

- Automotive Microelectronic Systems
 - high complexity
 - high degree of interaction inside them
 - high degree of interaction between them

- Strong Requirements
 - functional safety
 - reliability
 - cost
 - time to market

- Modeling and Simulation-Based Systems Engineering

https://en.wikipedia.org/wiki/List_of_software_bugs

Transportation [\[edit \]](#)

- By some accounts ██████████ [electronic throttle control system](#) (ETCS) had bugs that could cause [sudden unintended acceleration](#).^[63]
- The ██████████ experienced an [integer overflow](#) bug which could [shut down all electrical generators](#) if the aircraft was on for more than 248 days.^[64] A similar problem was found in ██████████ which need to be powered down before reaching 149 hours of continuous power-on time, otherwise certain avionics systems or functions would partially or completely fail.^[65]
- In early 2019, the transportation-rental firm ██████████ discovered a firmware bug with its [electric scooters](#) that can cause them to [brake very hard unexpectedly](#), which may hurl and injure riders.^[66]
- ██████████ had all [cockpit displays go blank](#) if a specific type of instrument approach to any one of seven specific airports was selected in the [flight management computer](#).^[67]
- ██████████ equipped with flight management systems by ██████████ would [make wrong turns during missed approach](#) procedures executed by the autopilot in some specific cases when temperature compensation was activated in cold weather.^[68]
- In June 1996, ██████████ failed less than a minute after launch, because the horizontal bias value was too big for a 16 bit register.

Motivation

Models are the Basis for Engineering

- Contain more detail than text specifications
 - no need to reread (reinterpret) text specifications repeatedly
 - less ambiguous (execute and observe)
- Can be shared
 - virtual integration of component models build larger systems (reuse and integrate)
 - early customer involvement
- Evaluate safety functions
 - fault injection
- Enable hardware/software co-design
 - improve time-to-market (shift left)

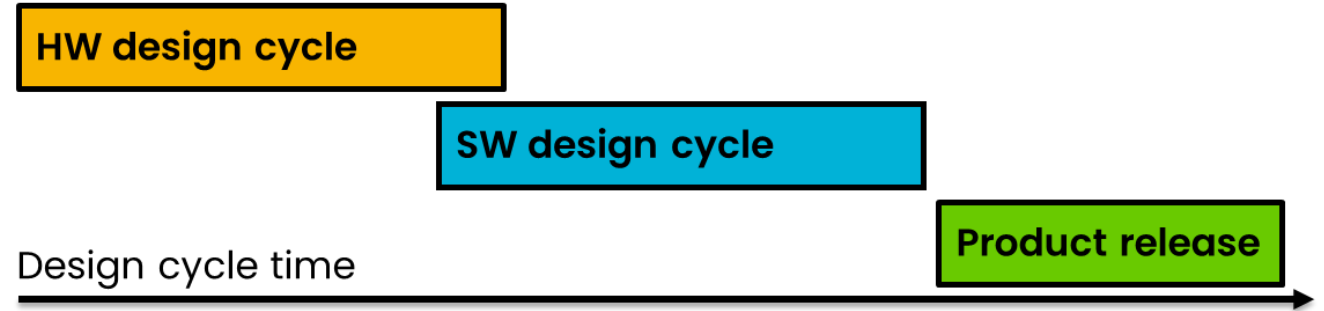


Motivation

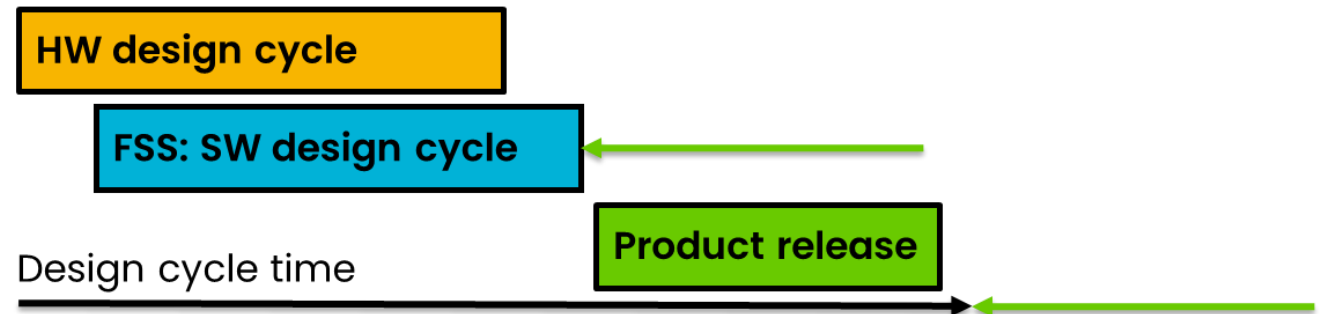
Shift Left with HW/SW Co-Design

- Developing HW and SW in parallel
 - traditional: first develop ECU then SW
 - faster: start early SW development with virtual prototype
 - supports collaboration and quality of results reducing time to production
- Early SW testing
 - HW and lab space limit test cycles
 - fast and more test cycles enable increased coverage
 - fault injection to test safety features

1.) HW based development of embedded SW



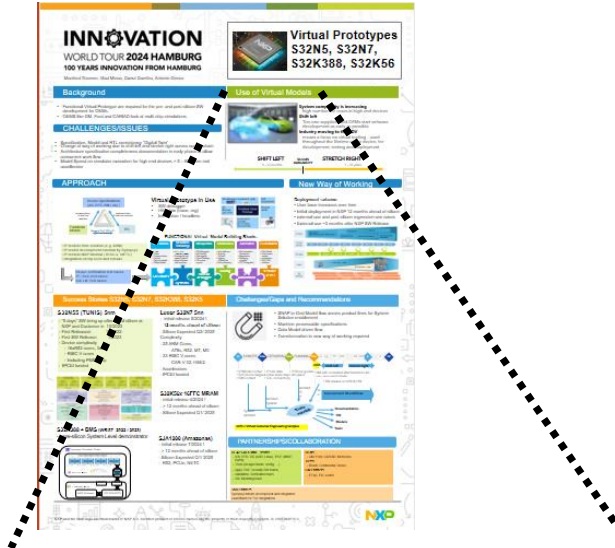
2.) Shift left: HW/SW co-design



Motivation

Shift Left with HW/SW Co-Design

- Developing HW and SW in parallel
 - traditional: first develop ECU then SW
 - faster: start early SW development with virtual prototype
 - supports collaboration and quality of results reducing time to production



Use of Virtual Models

- Early SW testing
 - HW and lab space limit test cycles
 - fast and more test cycles enable increased coverage
 - fault injection to test safety features



System complexity is increasing
 high numbers of cores in high end devices

Shift left
 Tier-one suppliers and OEMs start software development as early as possible

Industry moving to the SDV
 means a focus on virtual tooling - used throughout the lifetime of the device, for development, testing and deployment

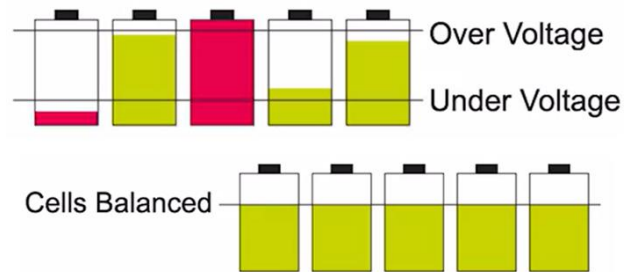


High Voltage Battery Management System



High Voltage Battery Management System

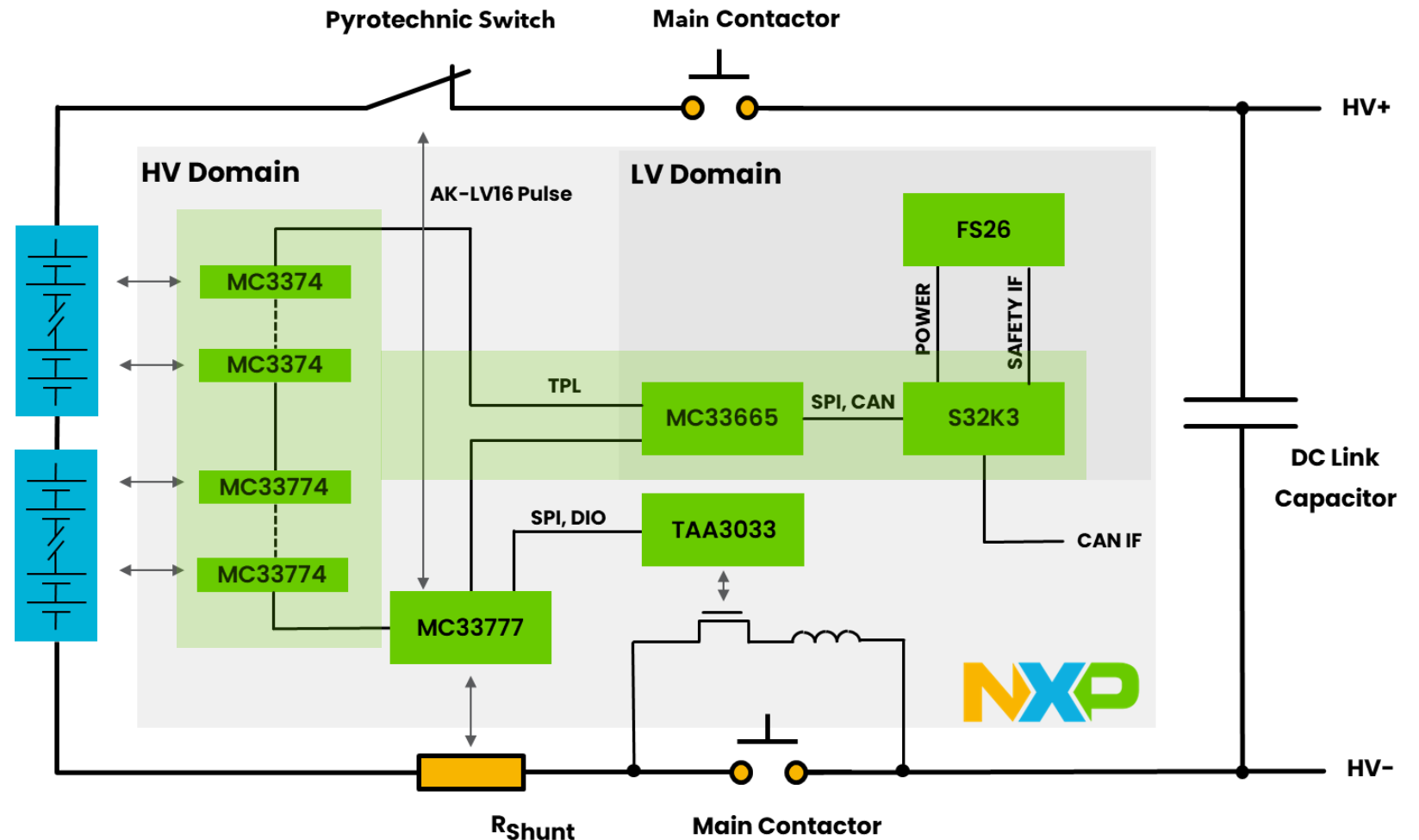
- No one size fits all BMS solution
 - scooters, motorcycles, buses, trucks, trains, boats or airplanes
 - battery packs as low as 14V or high voltages of 800V and more
- BCC directly connected to cells
 - no means of “turning off and on again”
 - low power consumption to avoid discharge of the cells
- Fire in an electric vehicle is an ASIL-D hazard
- Supervises voltage and temperature of cells
- Cell balancing
 - lowest voltage cell limits discharge
 - highest voltage cell limits charging



High Voltage Battery Management System

System Solution of a HV BMS

- **MCU**
Microcontroller Unit
- **GTW**
Gateway
- **BCC**
Battery Cell Controller
- **TPL**
Transport Protocol Link
- **BJB**
Battery Junction Box
- **PRE**
Precharge
- **SBC**
Safety System Basis Chip



Full System Simulation Setup

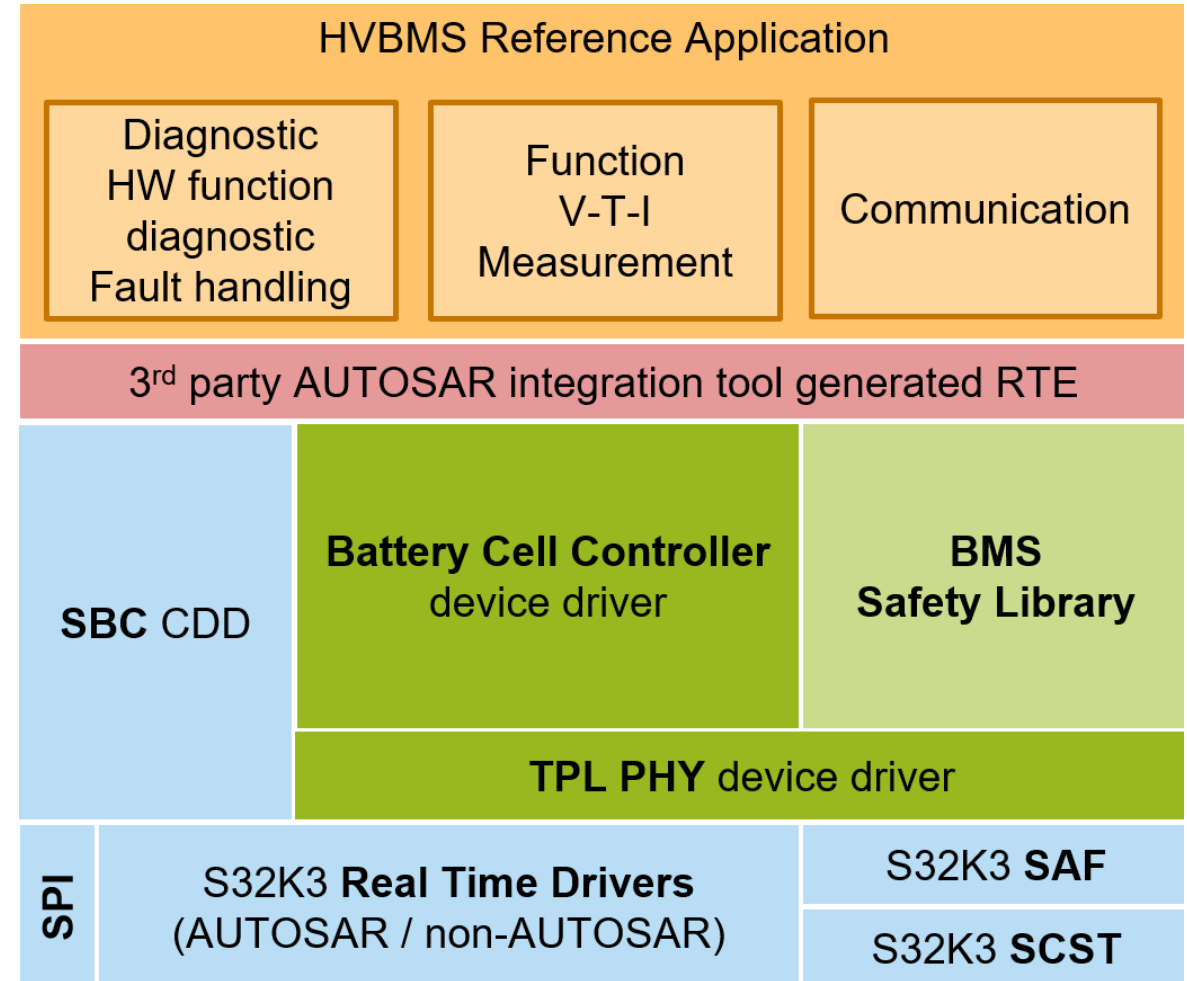


Full System Simulation Setup

Software Stack

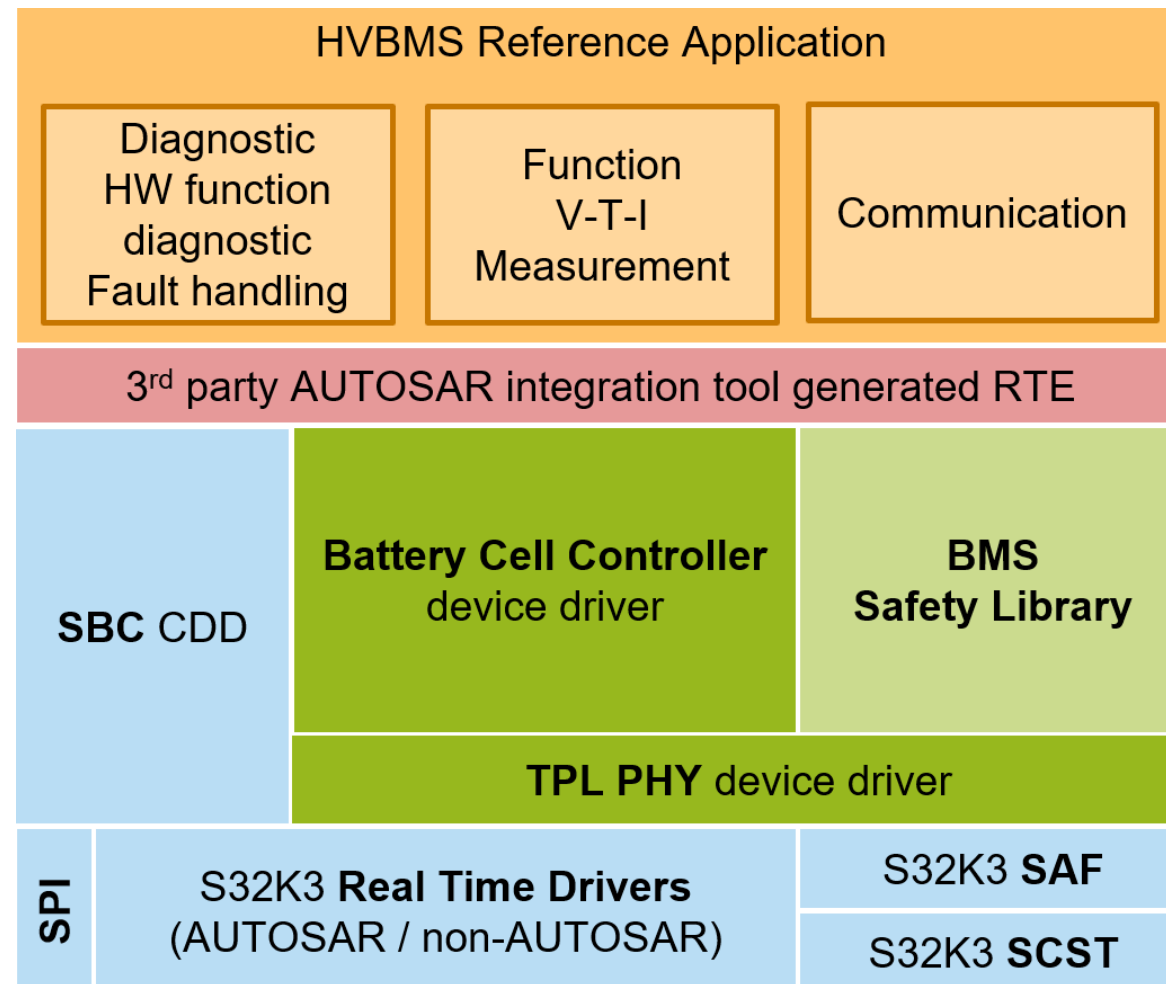
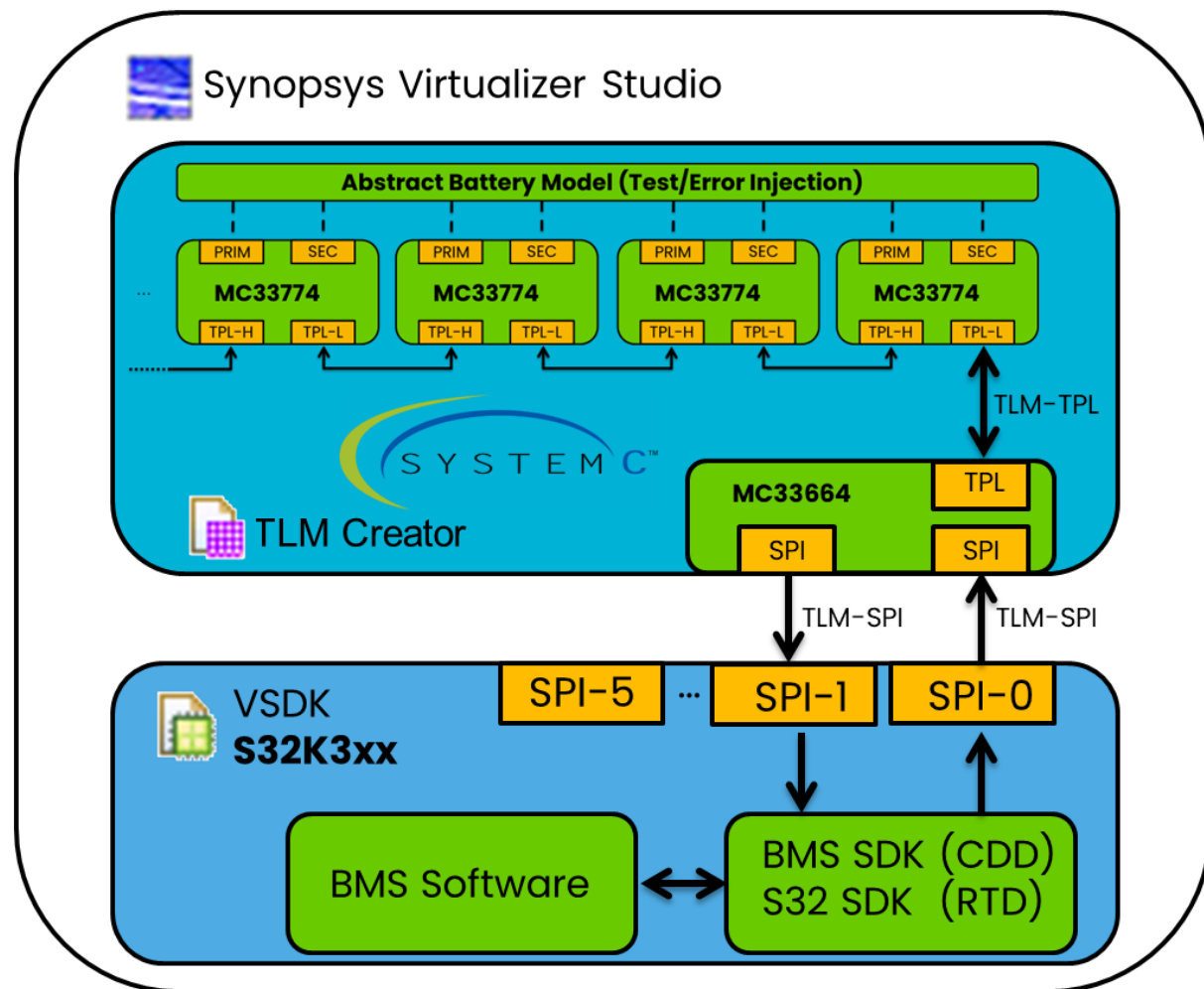
- Real time drivers
 - real time communication
- Complex device drivers
 - system basis chips
 - battery cell controller
 - ...
- Safety library
- AUTOSAR
- Application

Software Stack Based on AUTOSAR Standard



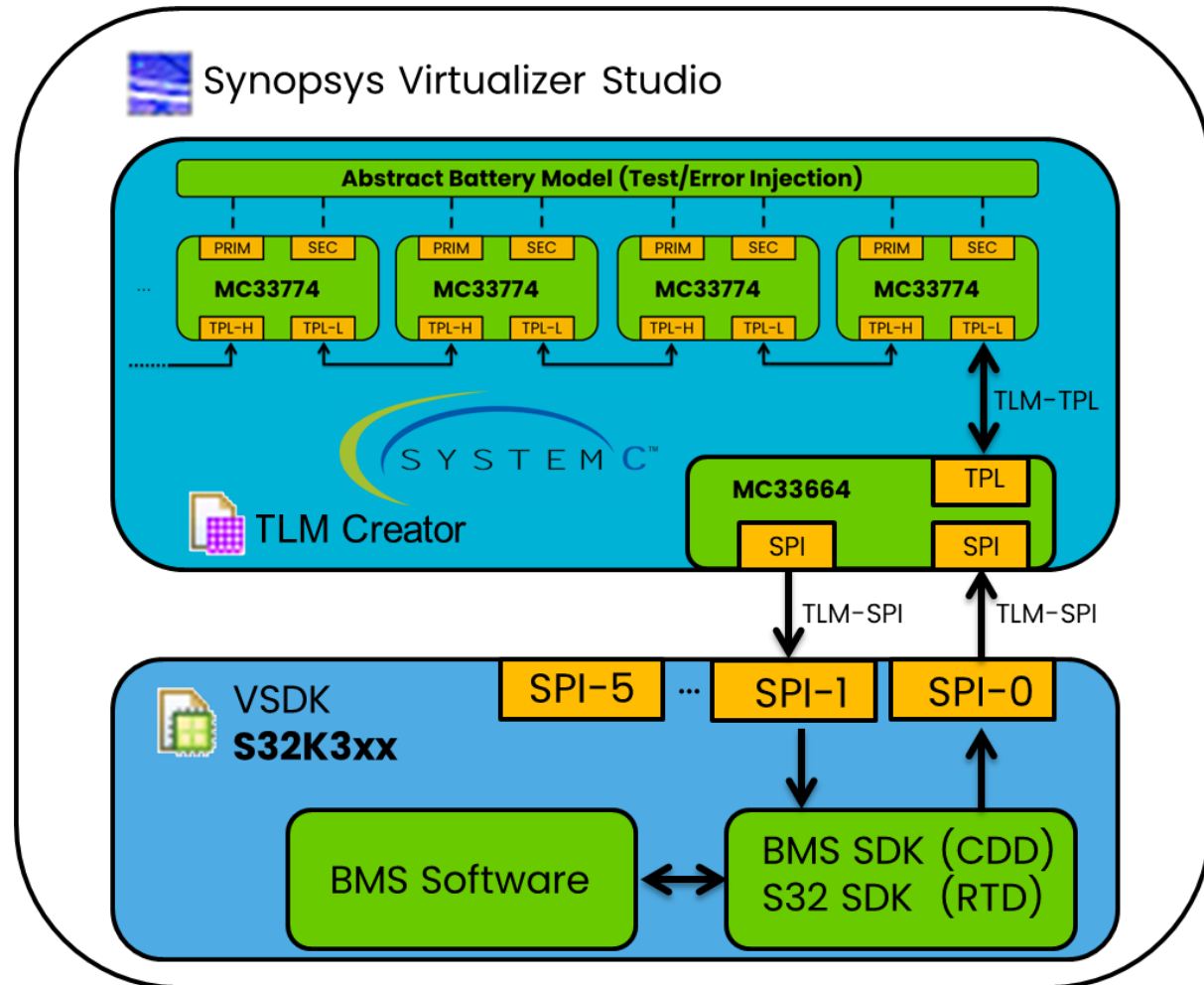
Full System Simulation Setup

Hardware



Full System Simulation Setup

Hardware

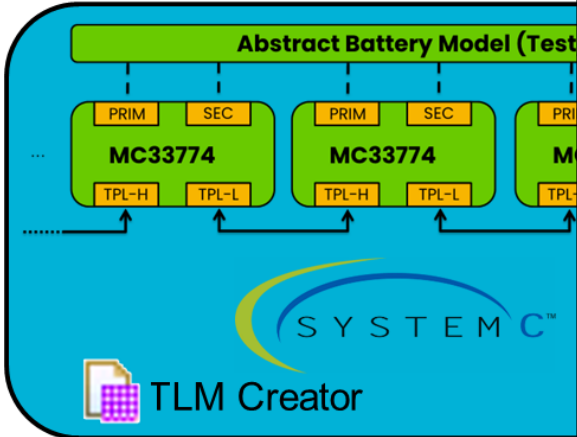


- Synopsys VSDK of NXP's MCU
 - complete hardware perspective with registers and peripherals (GPT, UART, CAN, SPI)
- BCC functional SystemC model
 - SystemC architecture generated from SysML
 - functional behavior added manually
 - importing SystemC into TLM creator using python script (300 path- and filenames)
- Gateway
 - helpful Synopsys tutorial on SPI controller
 - translates TLM-SPI to TLM-TPL
- Drag and drop into VSDK
 - components in Virtualizer Studio Library

Full System Simulation Setup

Hardware

Synopsys Virtualizer Studio



VSDK
S32K3xx

SPI-5

BMS Software

workspace_synopsys

File Edit Navigate Search Project Run Creation VDK Debug Window Help

Project Exp... BCC18_ADC_SP... S32K3_Sys... my_S32K3... BCC18_SP...

VDK - S32K3_System Build: success

Design Hierarchy S32K388_System

type filter text

S32K3_System

S32K3_MCU

BCC18_SPI_1

IP Model Libraries

Use this section to choose the model libraries and define the search paths. The found models are shown in the 'Library' view and the 'Add Building Block' dialog.

type filter text

S:\SNPS_VP_HOME\virtualizerstudio\lib

S:\NXP_S32_IPDIR\vd_k_cwr.lib

S:\NXP_S32_IPDIR\S32K3_MCU\2.0\Synopsys_S32K3_MCU_S32K3_MCU.pctlib

S(workspace)

S(proj:BCC18_ADC_SPI)

BCC18_ADC_SPI.tmc

NXP/BCC18_SPI/BCC18_SPI/V1.0

NXP_BCC18_SPI

Library Bro... type filter text

Building Blocks Examples

Core

ARM

Synopsys

Cores

ARM

CPU

ARM

Synopsys

DesignWare Core

Virtualizer Stu

>>>

Configure Network SPI_NW_2

Interconnect selection: Synopsys/tim_ft_spi_bus/tim_ft_spi_bus

Parameters:

Name	Design Value	Type
ActAsStub	false	Boolean
baudrate_in_s	10000	Integer
DefaultPayload	0	Integer
DefaultSlave	0	Integer [0..2]
DisplayIfMessages	true	Boolean
SynchronousMode	true	Boolean

0 items selected

Show 'Library Manager'

S32K388_System

SpecFlow Settings Packaging Library Manager Interfaces Parameters

Tables

type filter text

Memory Map

Interrupt Table

Reset Tree

SSI Network

CAN Network

EPHY Network

I2C Network

LIN Network

SPI Network

SPI_NW_1

SPI_NW_2

SPI_NW_3

SPI_NW_4

Connections

type filter text

BCC18_ADC_SPI_1.interfaceMasterSPI - /BCC18_ADC_SPI_1

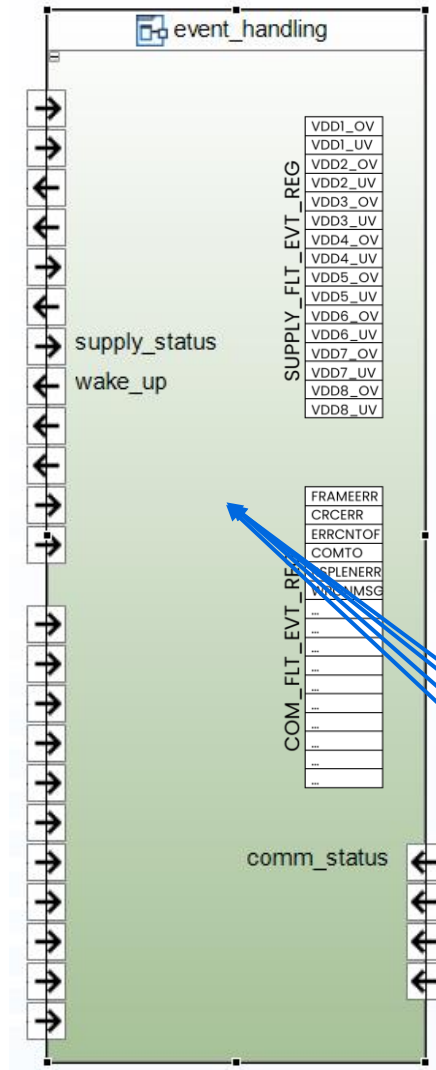
S32K388_MCU.lpspi_1_p_spi_device_socket - /S32K388_MCU

OK Cancel

Fault Injection



Coseda Fault Injection



```

print("PY: Ensure the proper initialisation of the simulation model.")
tpl.mcu.scWaitUS(1000)

print("PY: Reading 'SUPPLY_FLT_EVT_REG' register. Should be 0x0.")
reset_reason = tpl.getRegisterVal(bcc.SUPPLY_FLT_EVT_REG, 1)
assert (reset_reason == 0x0), f"Expected '0x0', got: {reset_reason}"

print("PY: Reading 'COM_FLT_EVT_REG' register. Should be 0x08.")
com_rst_cfg = tpl.getRegisterVal(bcc.COM_FLT_EVT_REG, 1)
assert (com_rst_cfg == 0x08), f"Expected '0x08', got: {com_rst_cfg}"

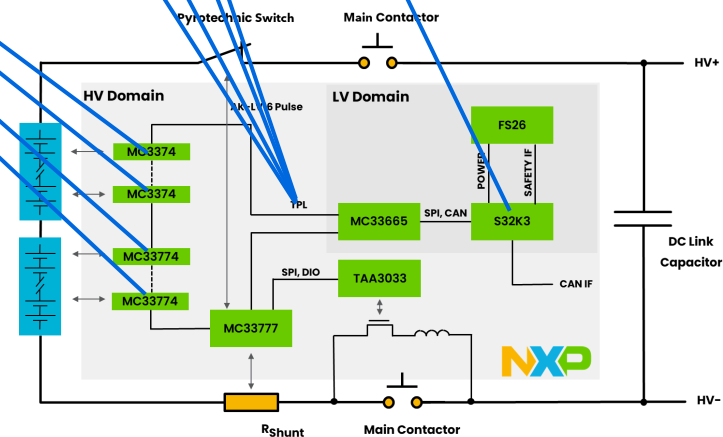
...

print("PY: Idling for 10 msec to start injecting faults.")
tpl.mcu.scWaitUS(10000)

# Under Voltage
print("PY: Reading 'SUPPLY_FLT_EVT_REG' register. Should be 0x08.")
reset_reason = tpl.getRegisterVal(bcc.SUPPLY_FLT_EVT_REG, 1)
assert (reset_reason == 0x08), f"Expected '0x08', got: {reset_reason}"

print("PY: Reading 'SUPPLY_FLT_EVT_REG' register. Should be 0x0 after prv read.")
reset_reason = tpl.getRegisterVal(bcc.SUPPLY_FLT_EVT_REG, 1)
assert (reset_reason == 0x0), f"Expected '0x0', got: {reset_reason}"

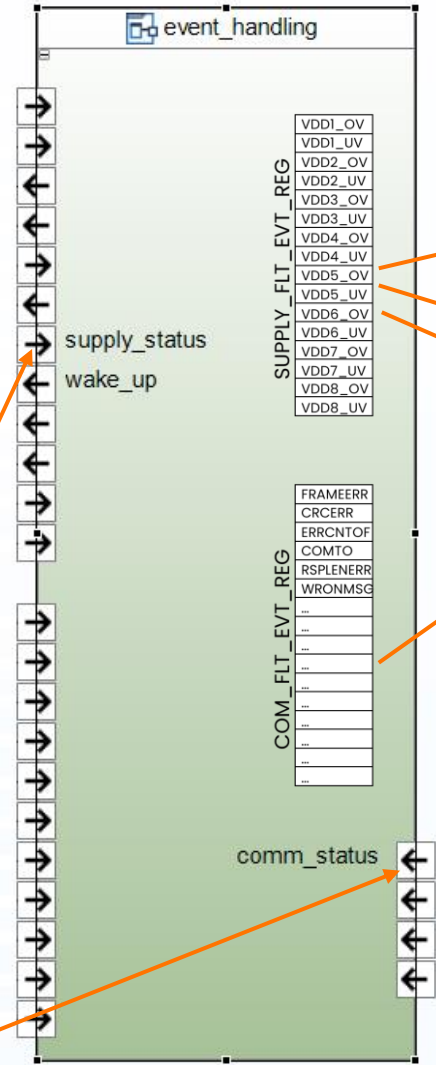
print("PY: Idling for 10 msec, for next injected fault.")
tpl.mcu.scWaitUS(10000)
    
```



Coseda Fault Injection

```
int sc_main(int argc, char** argv) {
    ...
    std::cout << "Init model." << std::endl;
    bcctb_fi bcctb("bcctb");
    ...
    /*****
     * Fault Injection controlled by file validating event_handling module
     *****/
    if (scheduled_fi){
        ...
        cos_register_fault_type<cos_stuck_at<bms::supply_status_t>>
            ("cos_stuck_at_supply");
        cos_fault_injector injector;
        injector.read_faults("fault_list_POR.fi");
        injector.inject_faults();
        sc_core::sc_start(sc_core::sc_time(simDurationTime));
        std::cout << "Simulation end at " << sc_time_stamp() << std::endl;
    }
    ...
}
```

```
// fault_list_POR.fi.
...
cos_stuck_at_supply stuck_power_supply ( xxxxx.yyyyy.event_handling.supply_status_pi ) value:0000000000
cos_stuck_at_comm stuck_comm_status ( xxxxx.yyyyy.event_handling.comm_status_pi ) value:00000000000000000000
#Sequence
wait 10e-3
// VDD1 Under Voltage
stuck_power_supply active value:000010000000
wait 10e-3
// VDD1 Over Voltage
stuck_power_supply active value:000100000000
wait 10e-3
...
```



```
print("PY: Ensure the proper initialisation of the simulation model.")
tpl.mcui.scWaitUS(1000)

print("PY: Reading 'SUPPLY_FLT_EVT_REG' register. Should be 0x0.")
reset_reason = tpl.getRegisterVal(bcc.SUPPLY_FLT_EVT_REG, 1)
assert (reset_reason == 0x0), f"Expected '0x0', got: {reset_reason}"

print("PY: Reading 'COM_FLT_EVT_REG' register. Should be 0x08.")
com_rst_cfg = tpl.getRegisterVal(bcc.COM_FLT_EVT_REG, 1)
assert (com_rst_cfg == 0x08), f"Expected '0x08', got: {com_rst_cfg}"

...

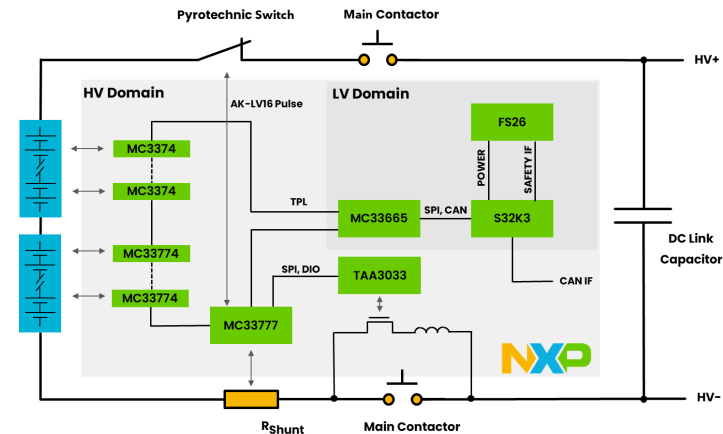
print("PY: Idling for 10 msec, to start injecting faults.")
tpl.mcui.scWaitUS(10000)

# Under Voltage
print("PY: Reading 'SUPPLY_FLT_EVT_REG' register. Should be 0x08.")
reset_reason = tpl.getRegisterVal(bcc.SUPPLY_FLT_EVT_REG, 1)
assert (reset_reason == 0x08), f"Expected '0x08', got: {reset_reason}"

print("PY: Reading 'SUPPLY_FLT_EVT_REG' register. Should be 0x0 after prv read.")
reset_reason = tpl.getRegisterVal(bcc.SUPPLY_FLT_EVT_REG, 1)
assert (reset_reason == 0x0), f"Expected '0x0', got: {reset_reason}"

print("PY: Idling for 10 msec, for next injected fault.")
tpl.mcui.scWaitUS(10000)

...
```



Coseda Fault Injection

```
SystemC 2.3.4-Accellera --- May 16 2024 13:27:03
Copyright (c) 1996-2022 by all Contributors,
ALL RIGHTS RESERVED

Init model.
[I] [ 0 s ]cos_panel_controller :
Open:
      http://127.0.0.1:8081/files/site/frontend.html
in web browser

LWS server is starting | visit http://127.0.0.1:8081
PYTB: start script ../../../../bms1bcc18tb/accelerasc_FI/py_use_cases/interactive_fault_injection_sample_usecase.py
PY: Executing script interactive_fault_injection_sample_usecase.py
*****
Measurement Fault Status = 0x0
Supply Fault Status = 0x0
Cell Voltage-Primary = : 6.094848 Volts
Cell Voltage-Secondary =0.019902 Volts
*****
Measurement Fault Status = 0x0
Supply Fault Status = 0x0
Cell Voltage-Primary = : 6.094848 Volts
Cell Voltage-Secondary =3.599844 Volts
*****
Measurement Fault Status = 0x0
Supply Fault Status = 0x0
Cell Voltage-Primary = : 6.094848 Volts
Cell Voltage-Secondary =3.599844 Volts
*****
Measurement Fault Status = 0x0
Supply Fault Status = 0x381
Cell Voltage-Primary = : 6.094848 Volts
Cell Voltage-Secondary =3.599844 Volts
*****
Measurement Fault Status = 0x0
Supply Fault Status = 0x381
Cell Voltage-Primary = : 6.094848 Volts
Cell Voltage-Secondary =3.599844 Volts
*****
```

COSEDA Interactive Simulator

127.0.0.1:8081/files/site/frontend.html

general

COSEDA technologies

5.3831 sec

Cell Voltage

3.600000

SUPPLY FLTS

on

COMM FLTS

off

MEASSYNC FLT SEC

off

PRIM CRC FLT

off

MEASSYNC FLT PRIM

off

CMP FAULT MEAS

off

LOGIC FLT

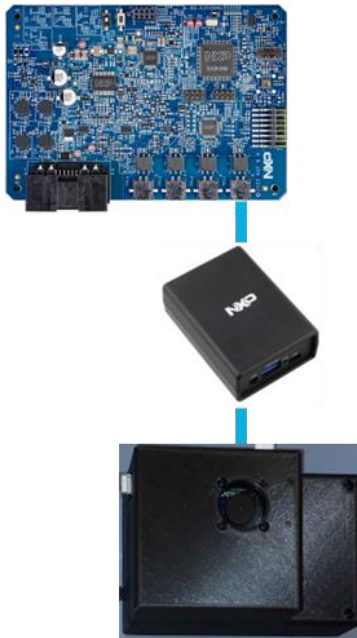
off

Conclusion and Future Work

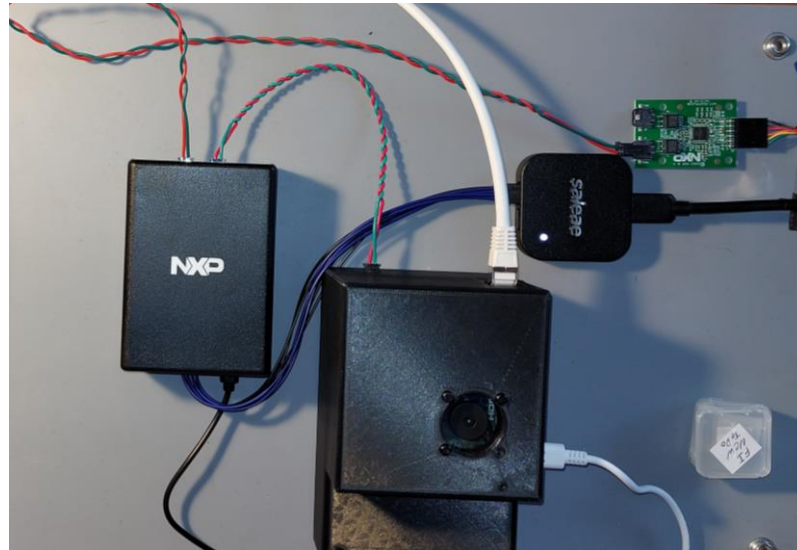


Conclusion and Future Work

- Experience and results are very convincing
- Investment of resources is justified
- Involved RTD/CDD developers:
 - “when to make use of this in our projects?”
 - “would help to clean out laboratory space!”



- Product variants to be modelled
 - reuse potential expected between models
 - Using SysML MBSE (Model Based System Engineering)
- Model In the Loop with Fault Injection
 - first experiments with Raspberry Pis 4 and 5
 - real, physical MCU and simulated BCC





Frank Poppen

frank.poppen@nxp.com

[nxp.com](https://www.nxp.com)

| **Public** | NXP and the NXP logo are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2024 NXP B.V.