SysML based Architecture Definition and Platform Generation Flow

Ralph Görgen, NXP Semiconductors

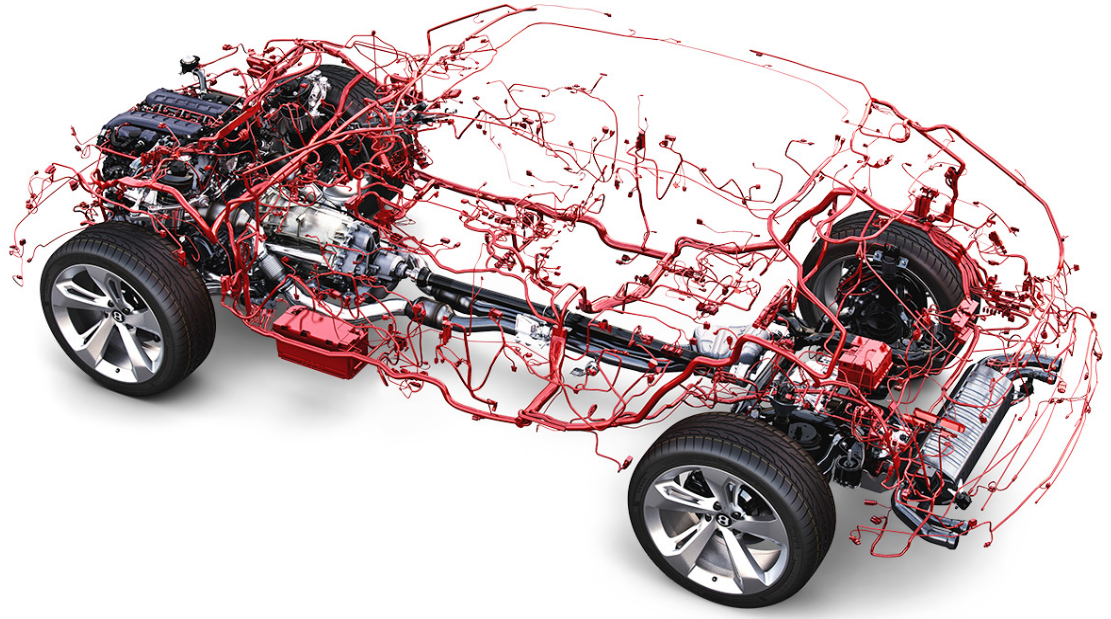
Erwin de Kock, NXP Semiconductors

SECURE CONNECTIONS FOR A SMARTER WORLD

2019 DESIGN AND VERIFICATION DVCON CONFERENCE AND EXHIBITION EUROPE

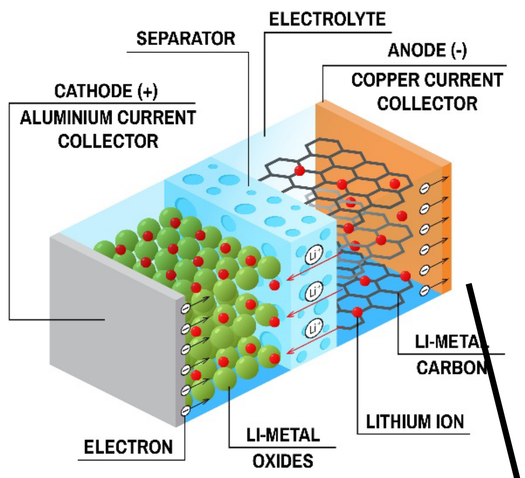
Motivation

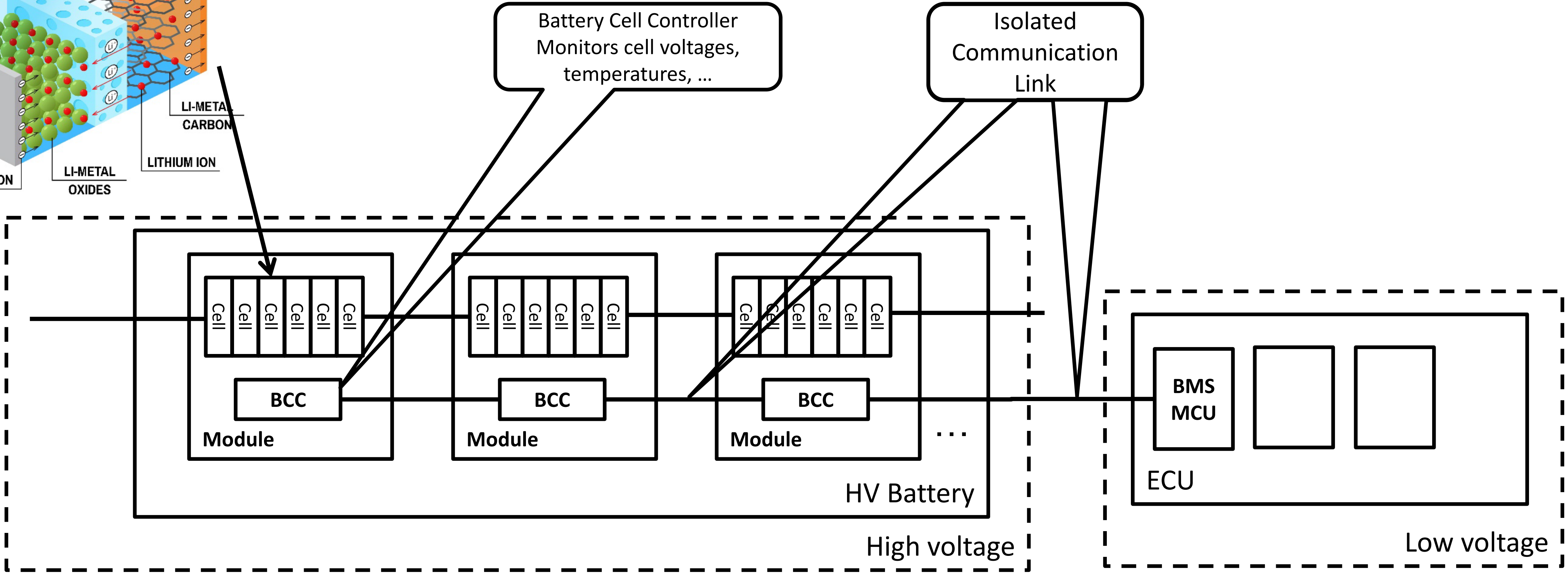
- High complexity of connected electronics systems in cars
- Functional Safety is …
  - a very important aspect in development of automotive semiconductor products
  - fulfilled by a system providing functionality and not individual components
- Model-based top-down flow enables traceability and model based safety analyses
- Executable virtual prototype enables analysis of HW/SW interaction
- VP shall represent architecture as defined



Outline

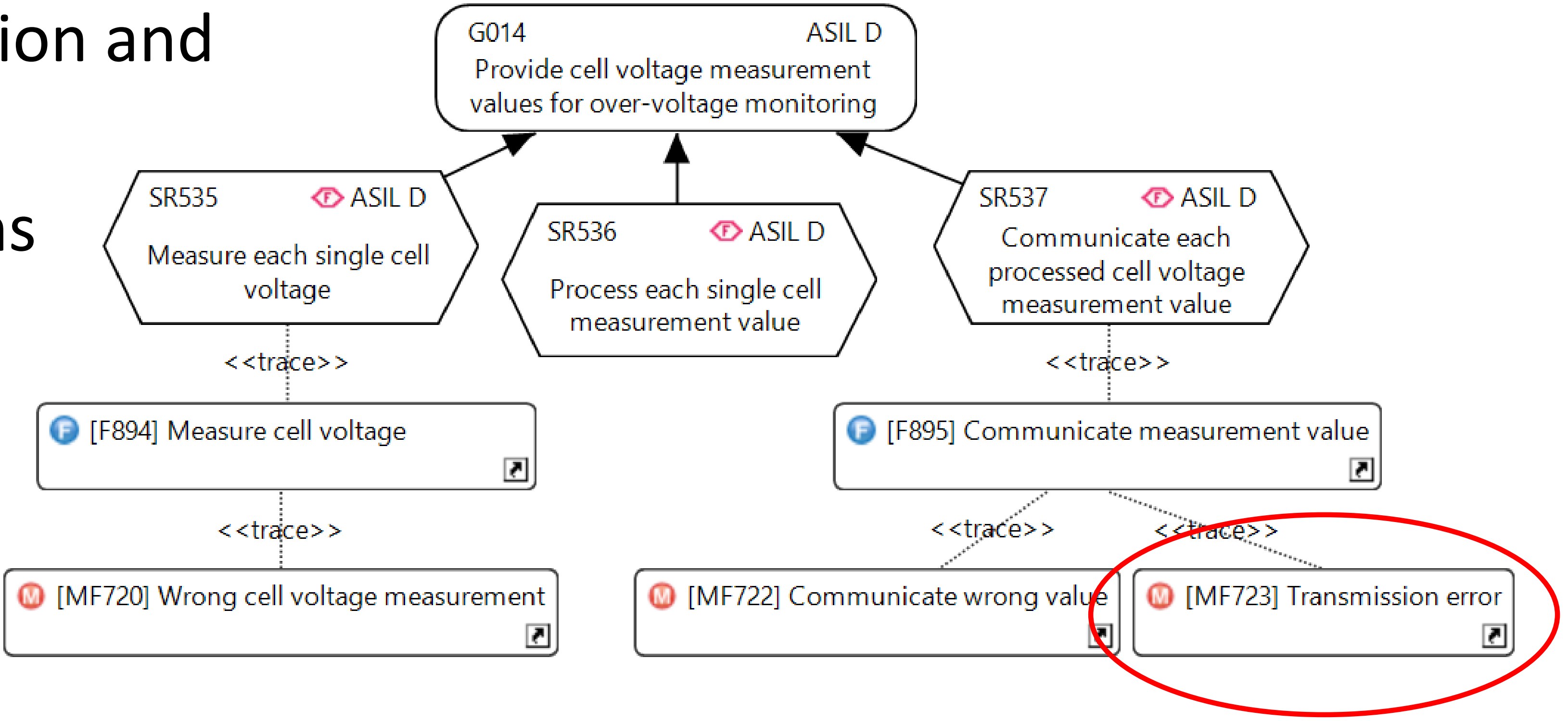
Automotive Battery System





SysML-based Safety Flow 1

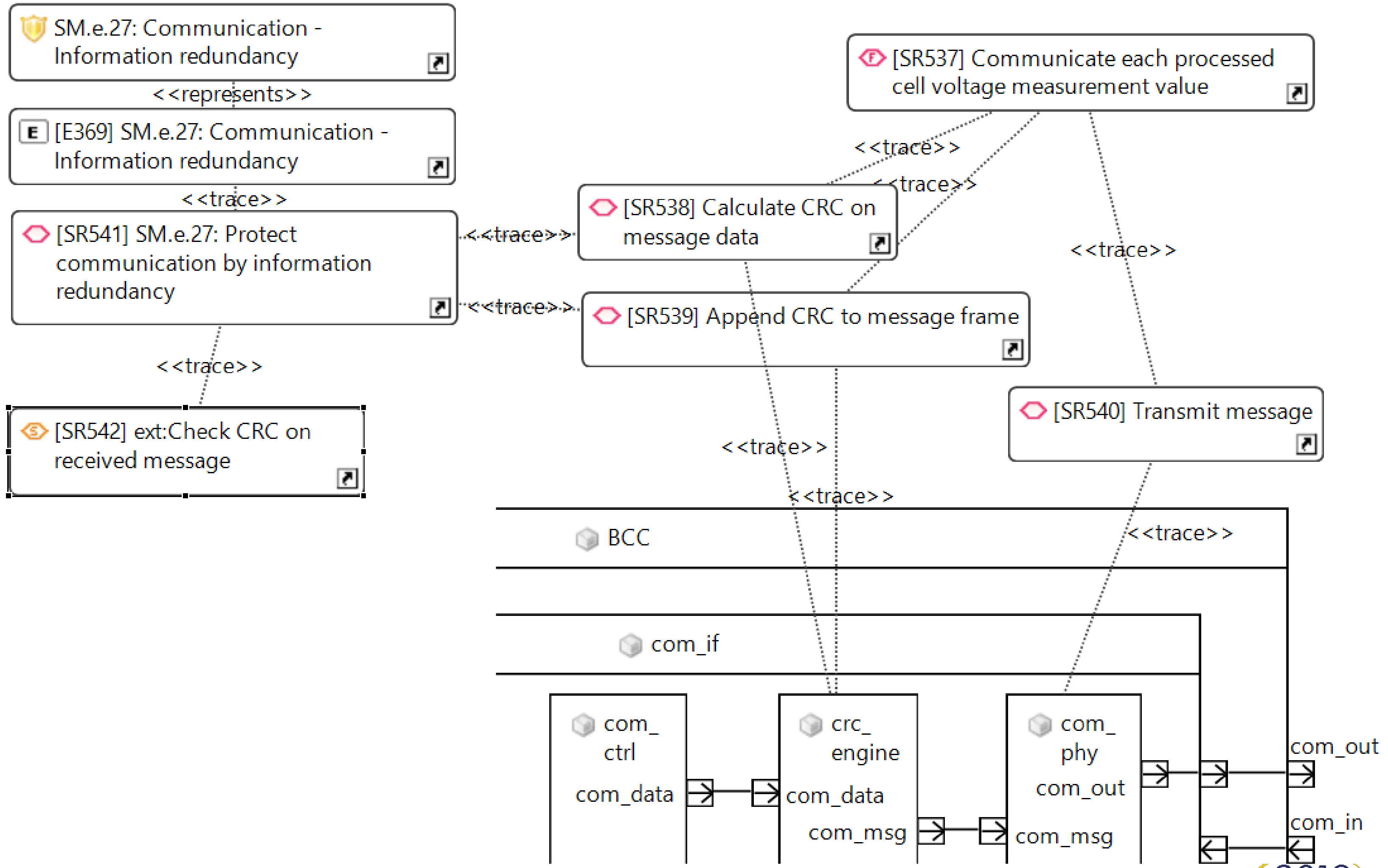
- Functional Safety Requirements
- Functional decomposition and architecture
- Annotated malfunctions
- Derive initial FTA



G014 ASIL D
Provide cell voltage measurement values for over-voltage monitoring
SR535 ASIL D
Measure each single cell voltage
SR536 ASIL D
Process each single cell measurement value
SR537 ASIL D
Communicate each processed cell voltage measurement value
<<trace>>
[F894] Measure cell voltage
[F895] Communicate measurement value
<<trace>>
[MF720] Wrong cell voltage measurement
[MF722] Communicate wrong value
[MF723] Transmission error

SysML-based Safety Flow 2

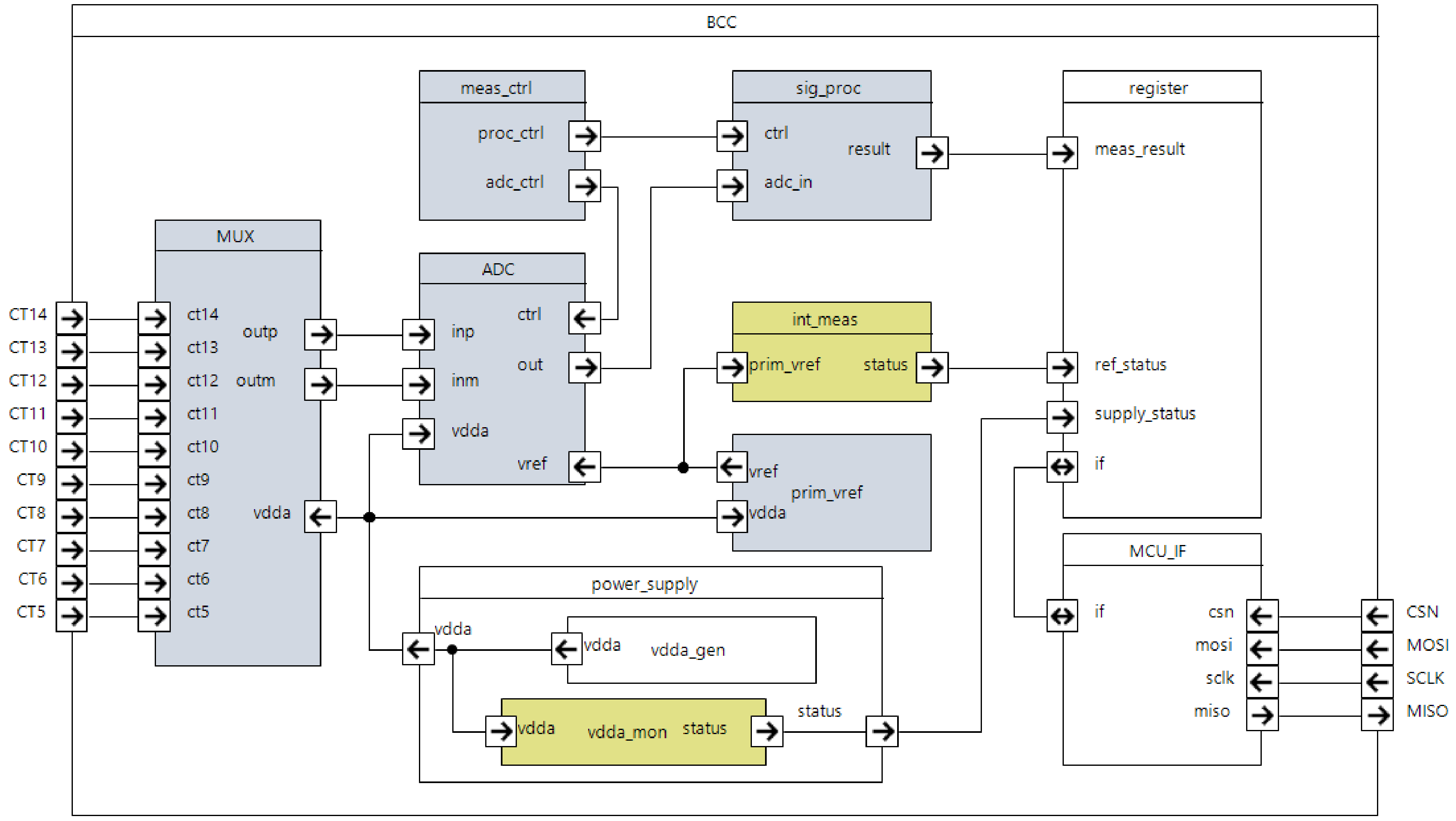
- Add safety mechanism to prevent SPF
- Split requirements in internal (HW) and external (SW)
- Allocate FSRs to HW architecture elements



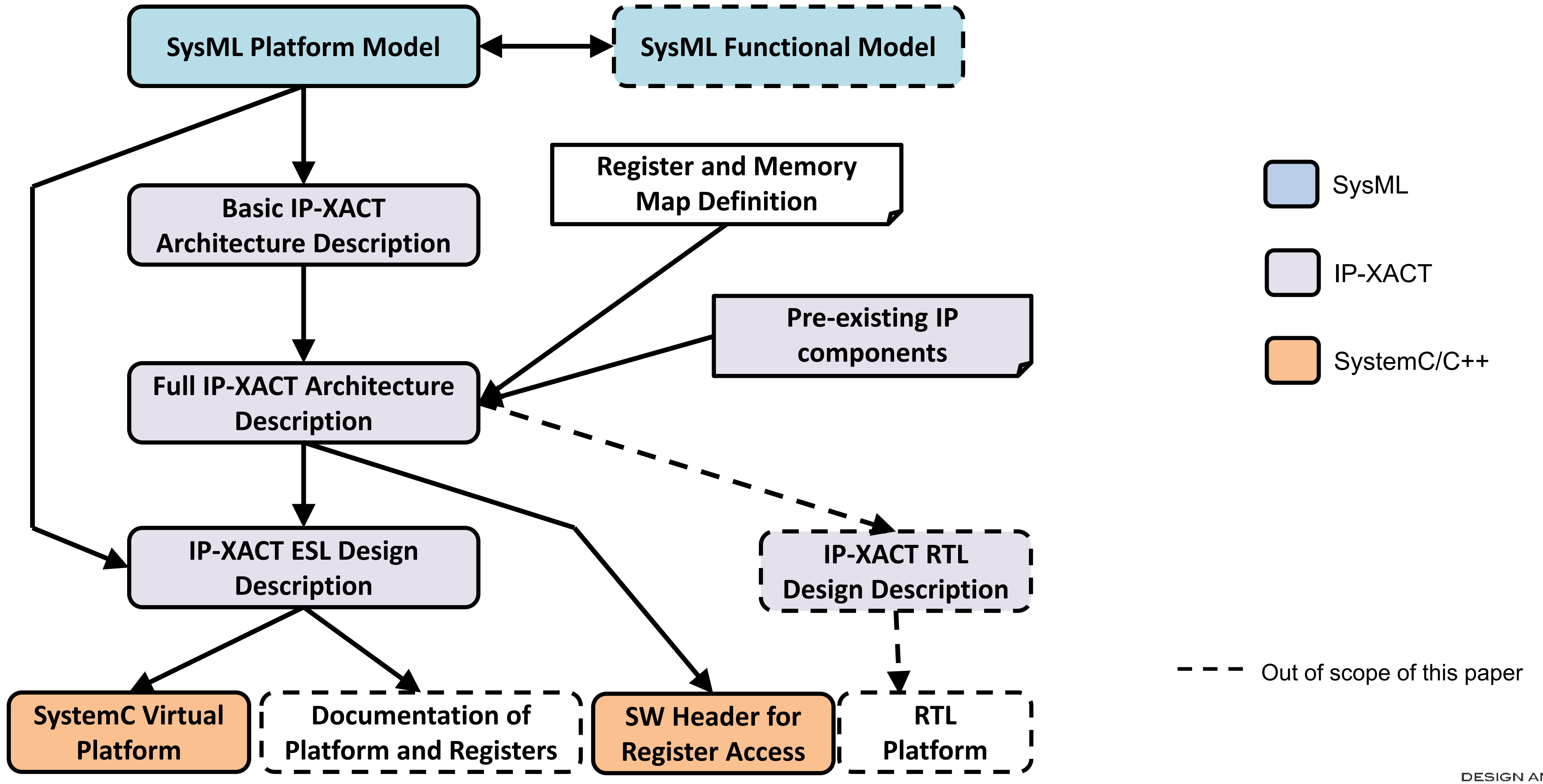
SM.e.27: Communication - Information redundancy
<<represents>>
[E369] SM.e.27: Communication - Information redundancy
<<trace>>
[SR541] SM.e.27: Protect communication by information redundancy
<<trace>>
[SR542] ext:Check CRC on received message
[SR537] Communicate each processed cell voltage measurement value
<<trace>>
<<trace>>
<<trace>>
[SR538] Calculate CRC on message data
<<trace>>
<<trace>>
[SR539] Append CRC to message frame
[SR540] Transmit message
<<trace>>
<<trace>>
<<trace>>
BCC
com_if
com_ctrl
com_data
crc_engine
com_data
com_msg
com_phy
com_out
com_msg
com_out
com_in

Platform Model in SysML

- Actual IC hardware architecture
- Decomposed hardware resources and communication links
- Real pinout



Model Generation Flow Steps


SysML Platform Model
SysML Functional Model
Register and Memory Map Definition
Basic IP-XACT Architecture Description
Pre-existing IP components
Full IP-XACT Architecture Description
IP-XACT ESL Design Description
IP-XACT RTL Design Description
SystemC Virtual Platform
Documentation of Platform and Registers
SW Header for Register Access
RTL Platform
SysML
IP-XACT
SystemC/C++
Out of scope of this paper

Architecture Description in IP-XACT

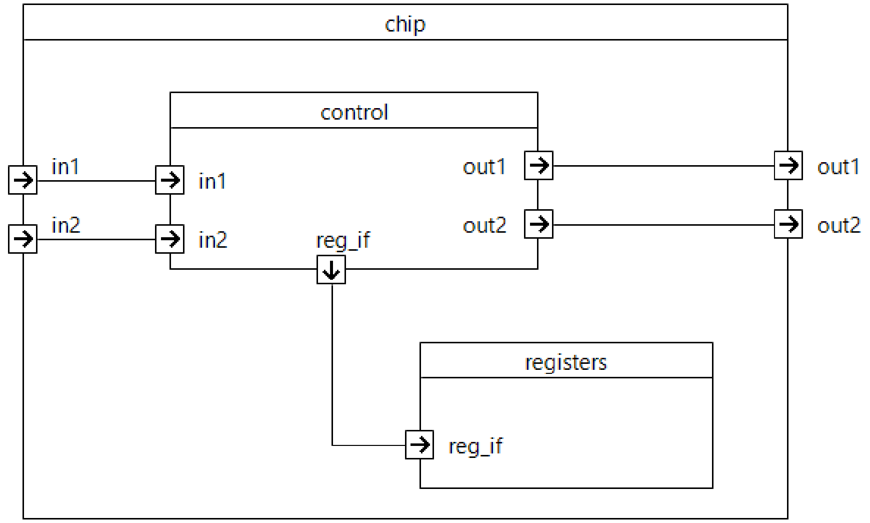
- Split between architecture and implementation
- Architecture description
  - Logical component interface, parameters, registers
  - Component instances and logical connectivity
  - Connectivity based on abstract interface connections
  - Reusable over all abstraction/implementation levels
- Implementation description
  - Addition of implementation specific details to architecture description
  - Ports, file sets
  - Communication refinement by bus abstraction definitions and port maps
  - Explicit for each abstraction level/implementation

SysML to IP-XACT Architecture

- Export plugin in SysML tool
- Reuse of existing TCL infrastructure to create IP-XACT
- Plugin traverses SysML model and generates TCL scripts
- Basic SysML to IP-XACT mapping:
  - block definition → IP-XACT component
  - block ports → bus interfaces
  - port types → bus definitions
  - Internal Block Diagrams → hier. component + interface connections
- Registers and memory map added from external source (XLS)

SysML to IP-XACT Architecture (2)

```
source ${nxp::workarea}/chip_control/magillem/run_assemble_architecture.tcl
source ${nxp::workarea}/chip_registers/magillem/run_package_architecture.tcl

nxp::createHierarchicalComponent nxp.com bms chip 0.0 architecture "" \
${nxp::workarea}/chip/METADATA/

component::createElement busif in1
component::setElementProperty busif in1 bus_type [ list nxp.com types my_type_t 1.0 ]
component::setElementProperty busif in1 bus_direction slave
component::createElement busif in2
component::setElementProperty busif in2 bus_type [ list nxp.com types my_other_type_t 1.0 ]
component::setElementProperty busif in2 bus_direction slave
component::createElement busif out1
component::setElementProperty busif out1 bus_type [ list nxp.com types my_type_t 1.0 ]
component::setElementProperty busif out1 bus_direction master

design::addComponentInstance [ list nxp.com bms chip_control 0.0 ] control
design::addComponentInstance [ list nxp.com bms chip_registers 0.0 ] registers

design::createHierarchicalConnection control out1 out1
design::createHierarchicalConnection control in1 in1
design::createHierarchicalConnection control in2 in2
design::createInterconnection control reg_if registers reg_if


if { [ file exists ${nxp::workarea}/chip/magillem/post_run_assemble_architecture.tcl ] == 1 } {
  source ${nxp::workarea}/chip/magillem/post_run_assemble_architecture.tcl
 }
```



Recursive call for subcomponents

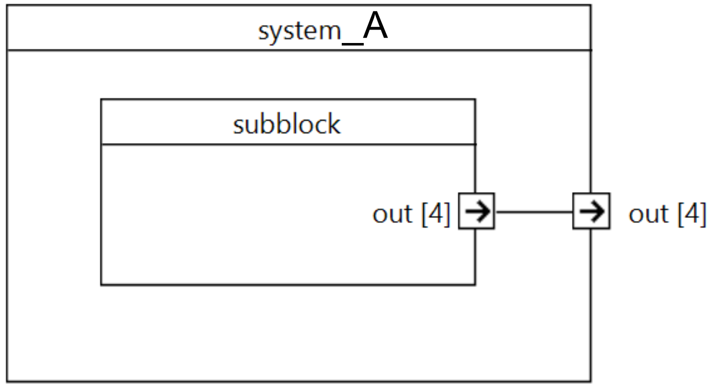
Create component

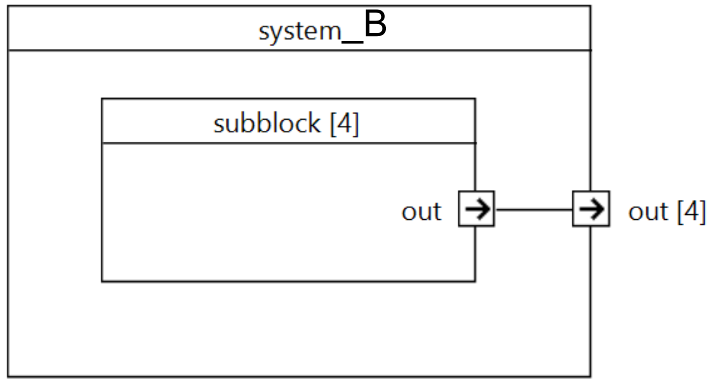
Create interfaces + assign bus type

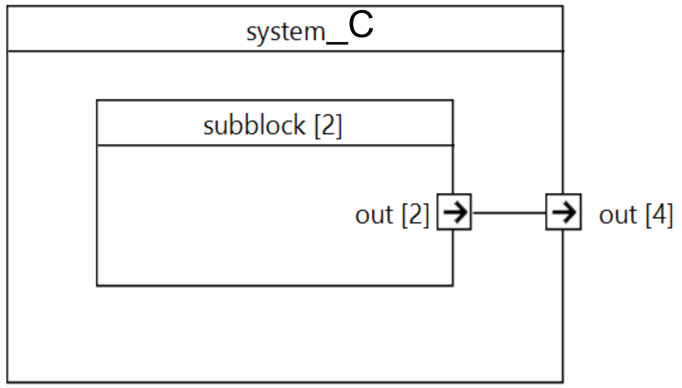
Instantiate and connect subcomponents

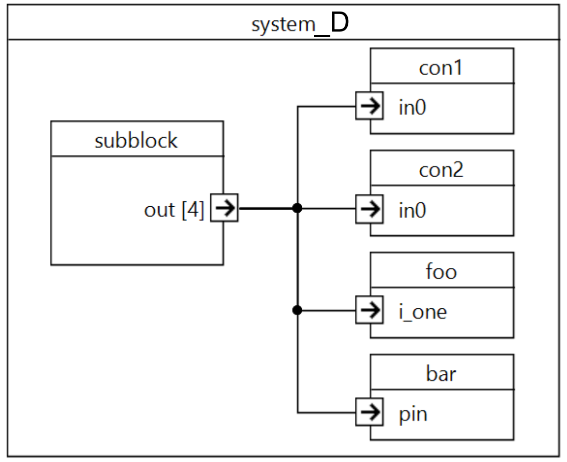
Call post processing script (if available)

Handling SysML Multiplicity



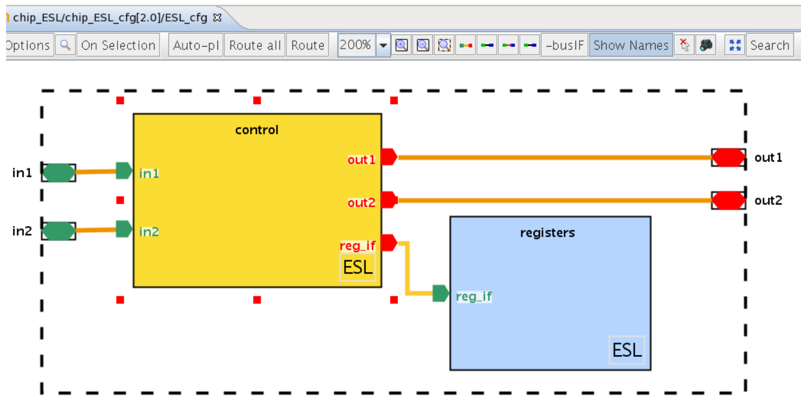






Generate ESL View

```
source ${nxp::workarea}/chip_control/magillem/run_assemble_esl.tcl
source ${nxp::workarea}/chip_registers/magillem/run_assemble_esl.tcl

nxp::copyHierarchicalComponent \
  nxp.com bms chip 0.0 architecture architecture_cfg \
  nxp.com bms chip 2.0 ESL ESL_cfg \
  ${nxp::workarea}/chip/METADATA/ systemc

component::setElementProperty busif in1 abstraction_type [ list nxp.com types my_type_t_esl 1.0 ]
component::setElementProperty busif out1 abstraction_type [ list nxp.com types my_type_t_esl 1.0 ]
nxp::createWirePort in1_pi in ""
nxp::createWireTypeDefs in1_pi [ list bms::my_type_t [ list ESL ] false [ list my_type_t.h ] ]
nxp::createWirePort out1_po out ""
nxp::createWireTypeDefs out1_po [ list bms::my_type_t [ list ESL ] false [ list my_type_t.h ] ]
component::setElementProperty busif in1 port_maps [ list [ list port "" in1_pi "" ] ]
component::setElementProperty busif out1 port_maps [ list [ list port "" out1_po "" ] ]

design::setComponentInstanceComponentRef control -v 2.0
design::setComponentInstanceComponentRef registers -v 2.0


nxp::createFileSet ESL \
  [ list \
    [ list controllib \
      [ list \
        ../SLMODEL/inc/chip_control.h \
        ../SLMODEL/src/chip_control_base.cpp \
        ../SLMODEL/src/chip_control.cpp \
      ]\
    ]\
  ]
component::setElementProperty fileset ESL dependency [ list ../SLMODEL/inc ]
```



Recursive call for subcomponents

Copy base component

Set bus abstraction

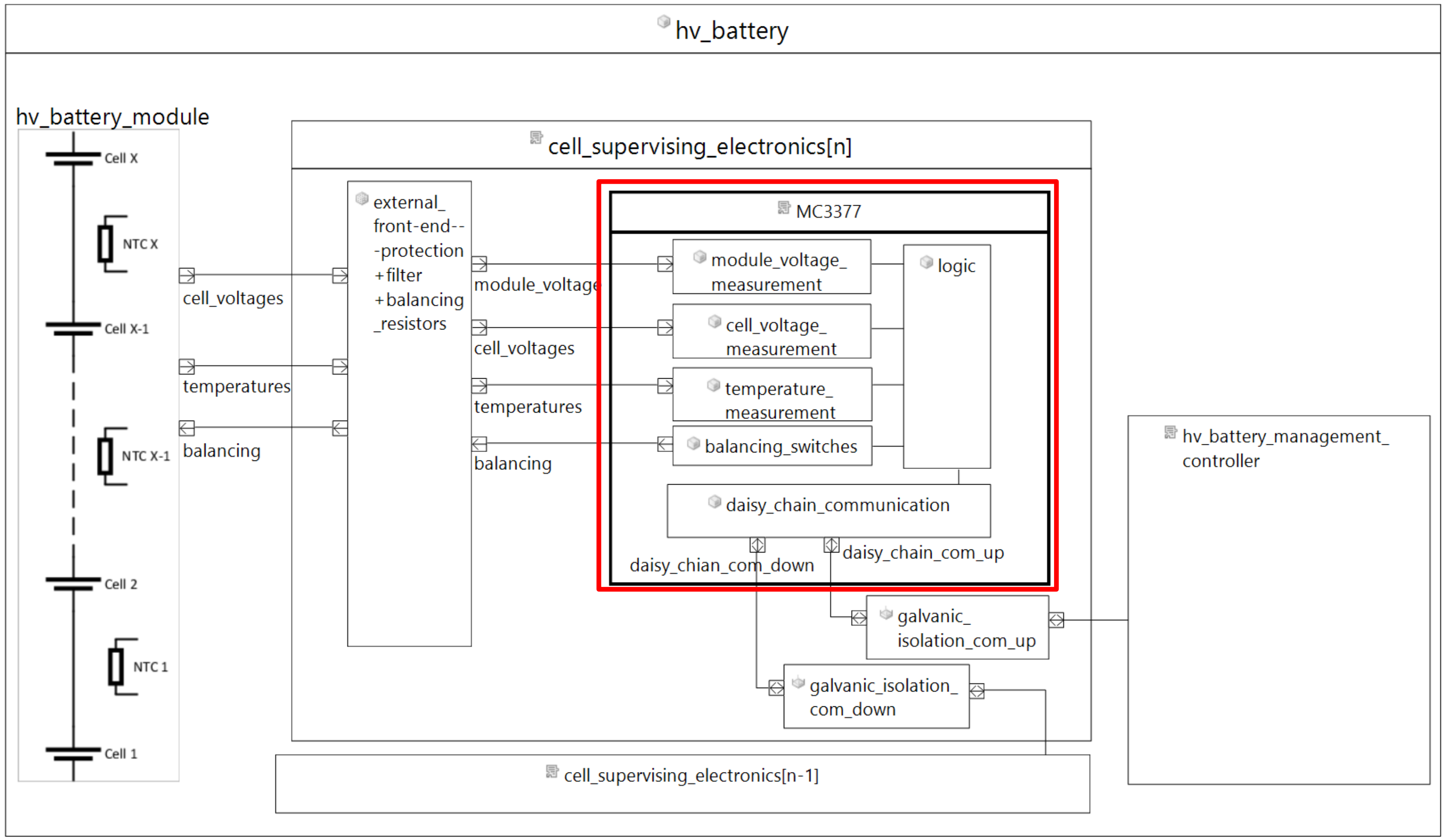
Create ports and port maps

Create fileset

Generate SystemC Model

- Leaf components
  - NXP internal tool generate modules incl. interface and registers
  - Register models based on SCML
  - Base module for generated elements and derived module to add behavior
- Hierarchical components
  - Commercial SystemC Netlister (Magillem tool suite)
- Data types for ports/signals
  - Header file containing default typedef to int
- Build files for SystemC compilation based on filesets

Flow Evaluation based on Battery Controller IC



Battery Cell Controller

- High accuracy measurement
- 14 cells + GP channels
- Integrated balancing

SysML Model:

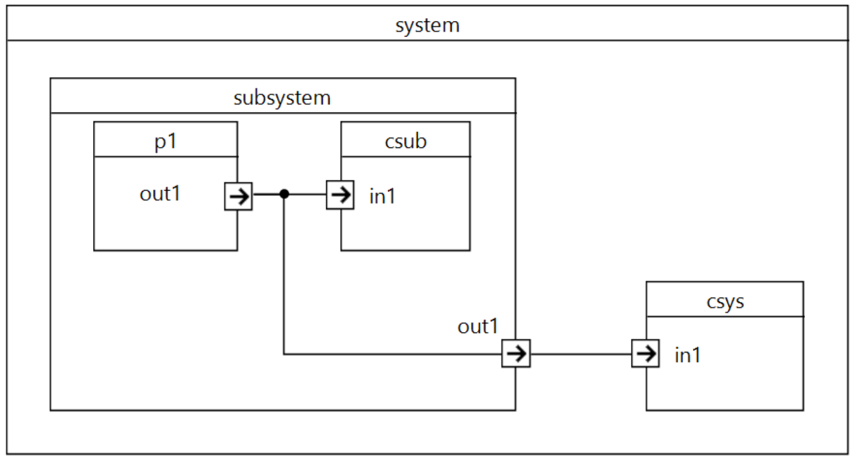
- More than 80 blocks
  - unique and re-used
  - analog, digital, mixed, memory
- More than 700 connectors
  - shared bus, point-to-point

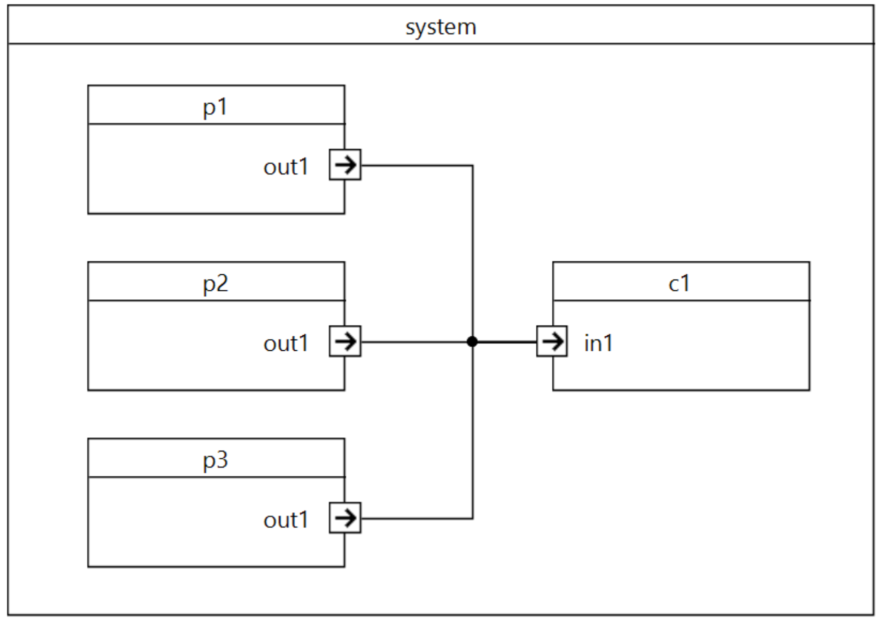
Generator Result:

- SystemC model + build scripts
- Compile and run out of the box
- Good starting point for refinement and adding behavior

Flow Limitations

- Different syntax rules (e.g. identifier, whitespace, keywords)
  - Checkers to check rules in SysML
- Missing information (e.g. vendor, library, version)
  - Can be added by generator (default values)
- Connectivity





Conclusion

- Good productivity gain
  - Creation of 65 IP-XACT components (15 hierarchical, >700 connections, excl. IP)
  - Saved effort aprox. 2 to 3 WWs
- Well-defined relation of SysML to architecture and model
  - Ensures consistency between safety analyses and implementation
- Incomporates well with existing flow (gen doc, SW headers, …)
- Flow requires modeling guidelines for SysML

Summary and Outlook

- Model based flow for safety critical semiconductor products
- System architecture definition and automated generation of VP skeleton
- Evaluation based on automotive battery controller IC

Outlook

- Refine and align modeling rules and extend automated checkers
- Sync updates in IP-XACT to SysML
- Improve AMS support

Questions