

Extending a SysML based Platform Model Generation Flow to support AMS

Ralph Görge

28.10.2019



SECURE CONNECTIONS
FOR A SMARTER WORLD

Company Public – NXP, the NXP logo, and NXP secure connections for a smarter world are trademarks of NXP B.V. All other product or service names are the property of their respective owners. © 2019 NXP B.V.

Outline

- Motivation: Automotive Battery Safety
- Model-based Top-Down Flow
- Automated Platform Generation Flow
- Extension to Support AMS
- Summary and Conclusion



Li-Ion traction battery incl. BMS example

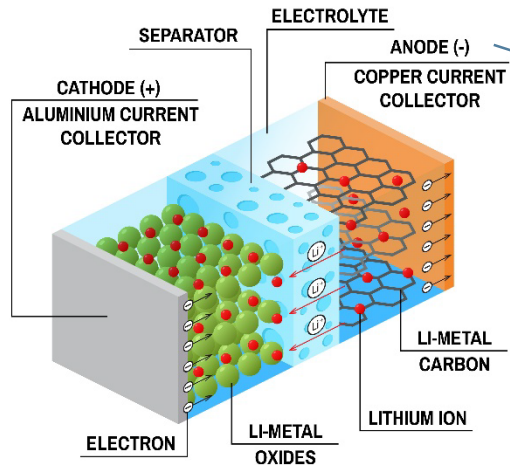
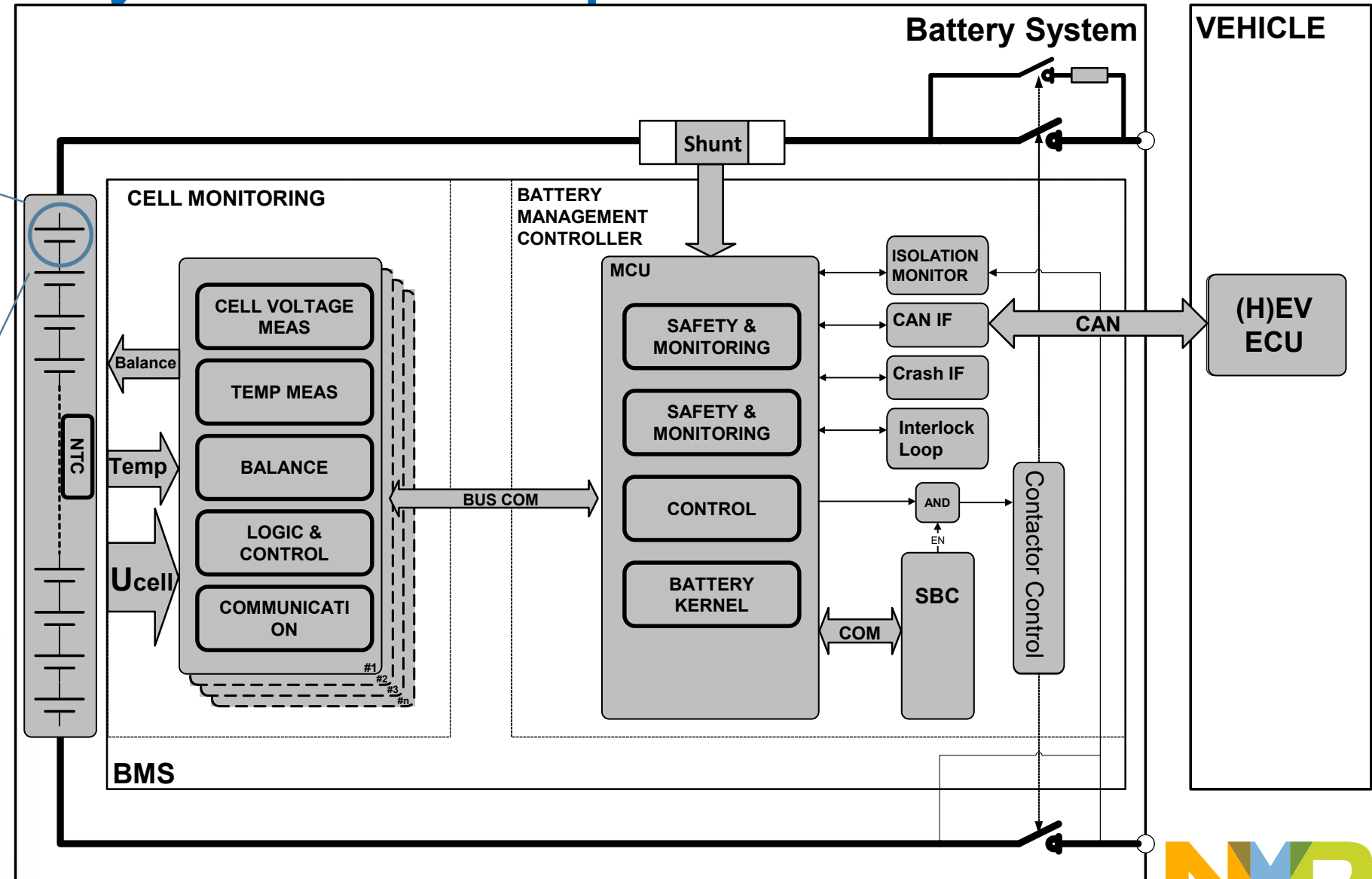
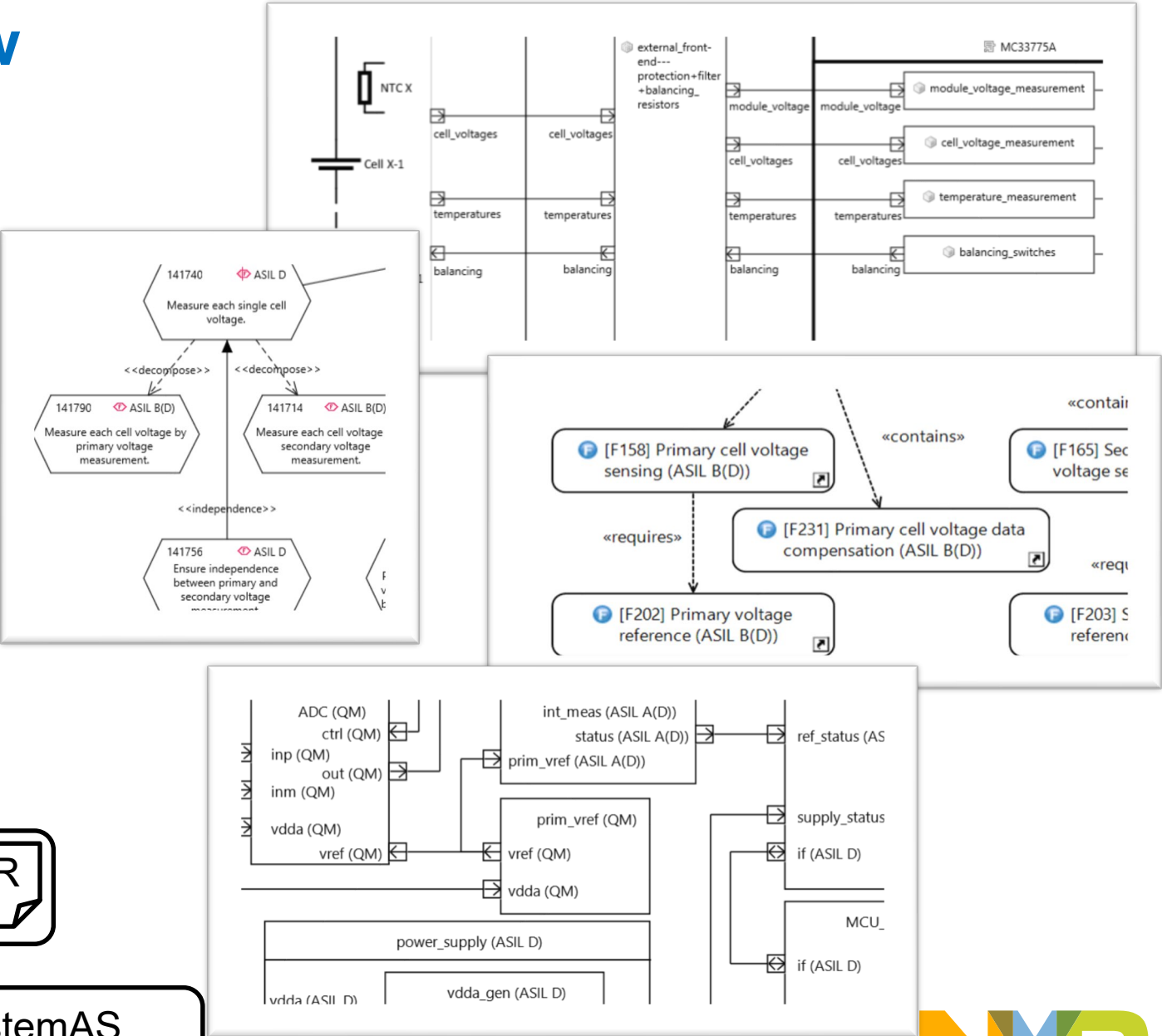
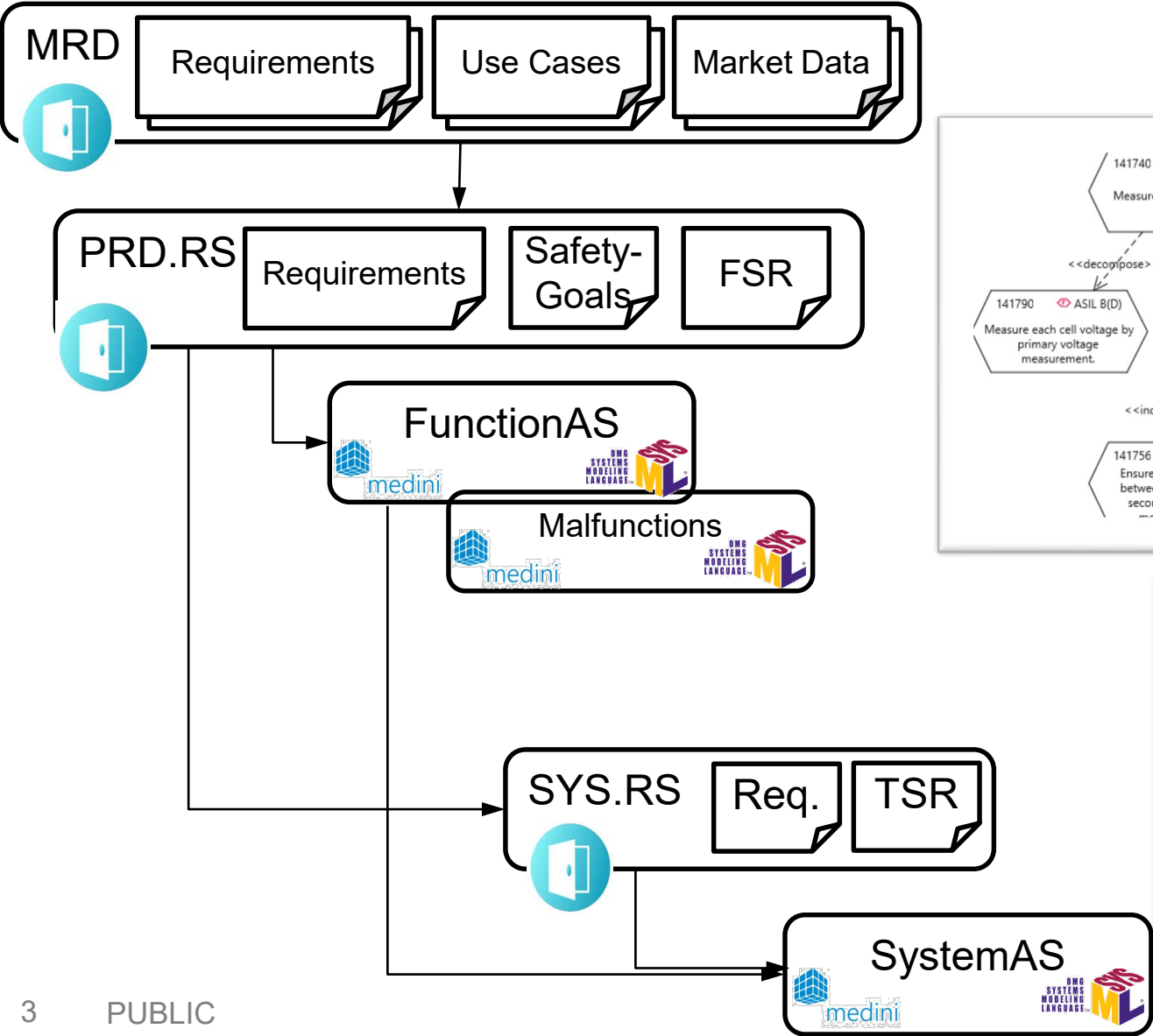


Figure 1

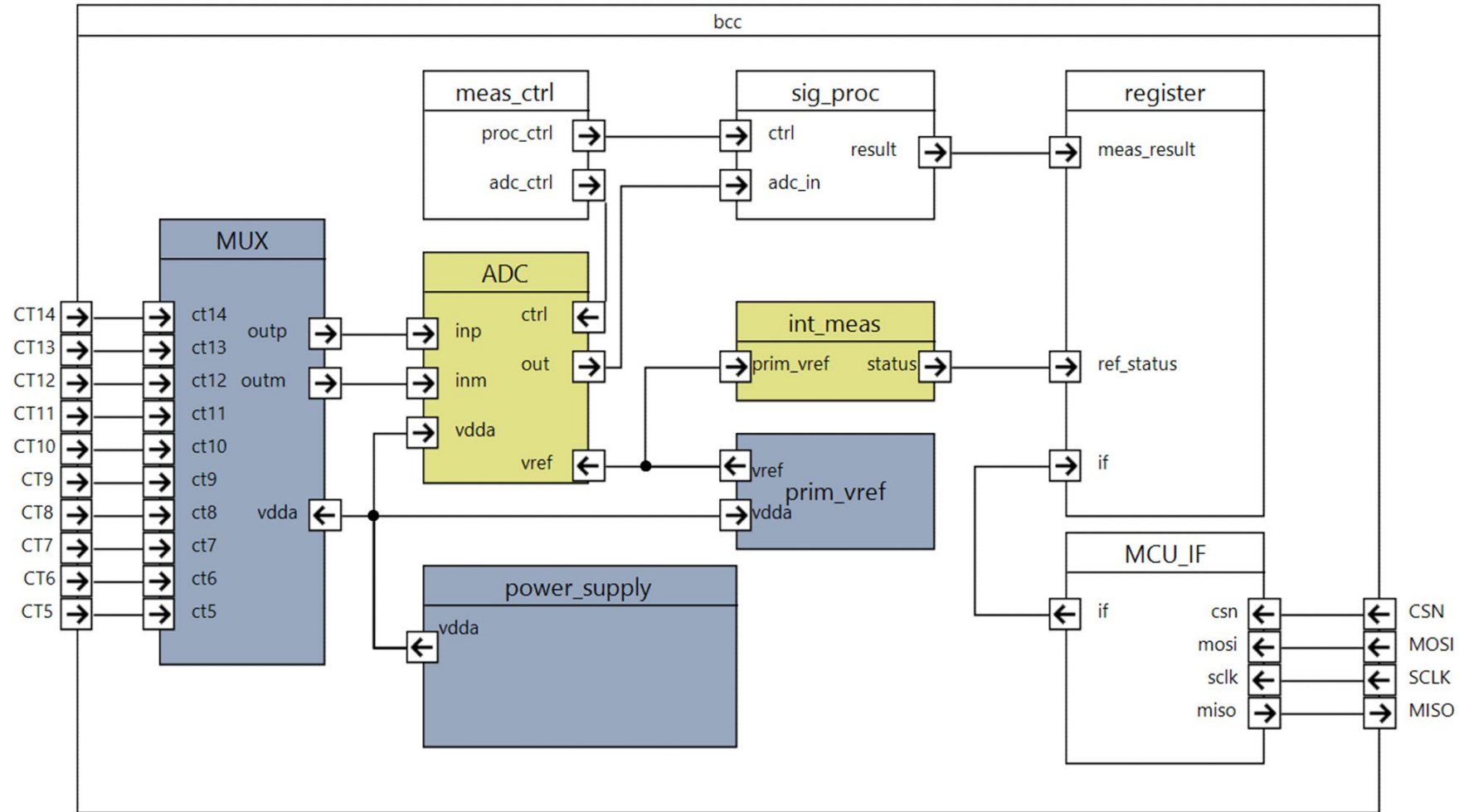


Model-Based Top-Down Flow

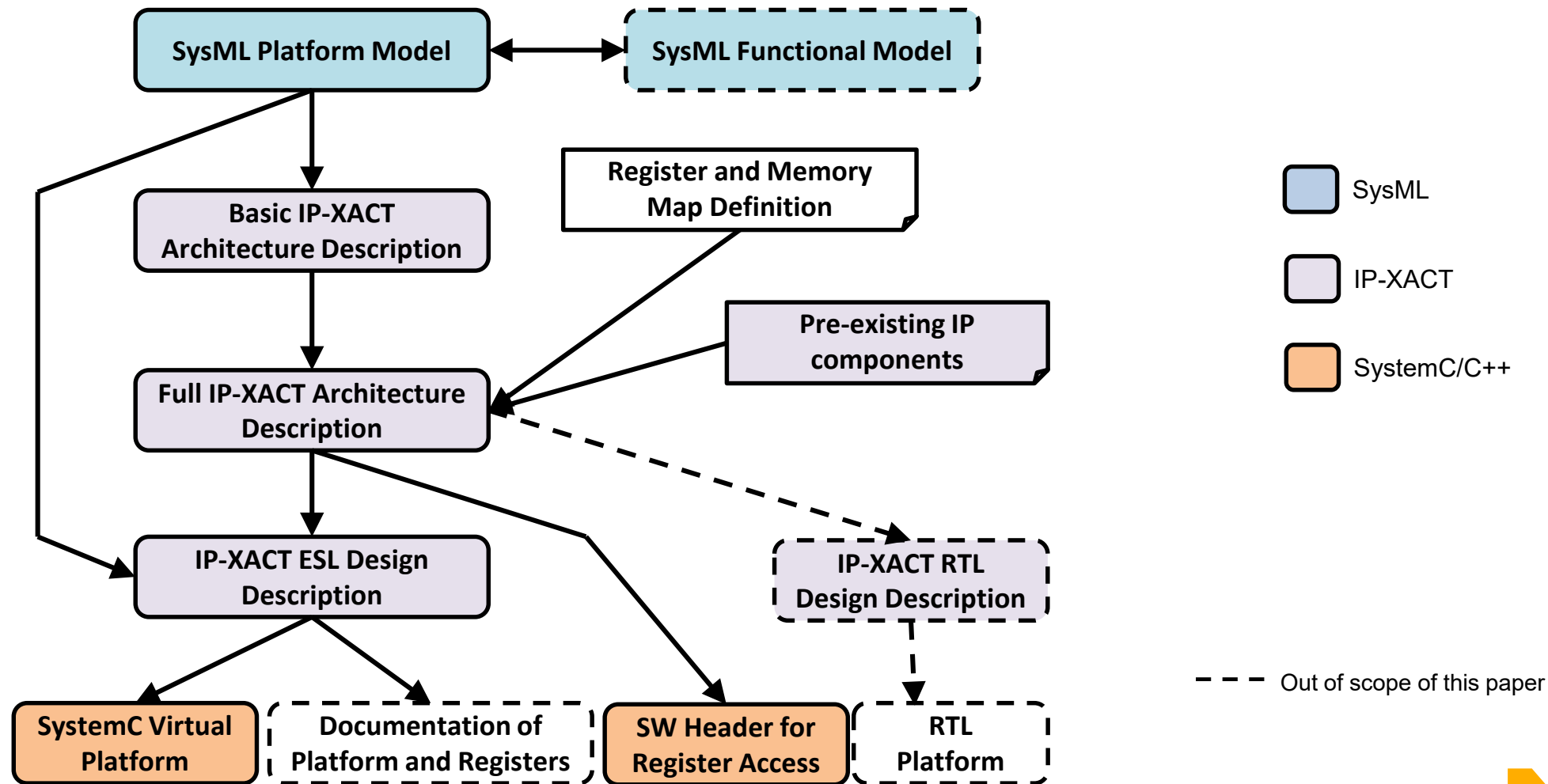


Hardware Architecture Model (Platform Model)

- IC level hardware architecture described in SysML Internal Block Diagram
- Decomposed hardware resources and communication links

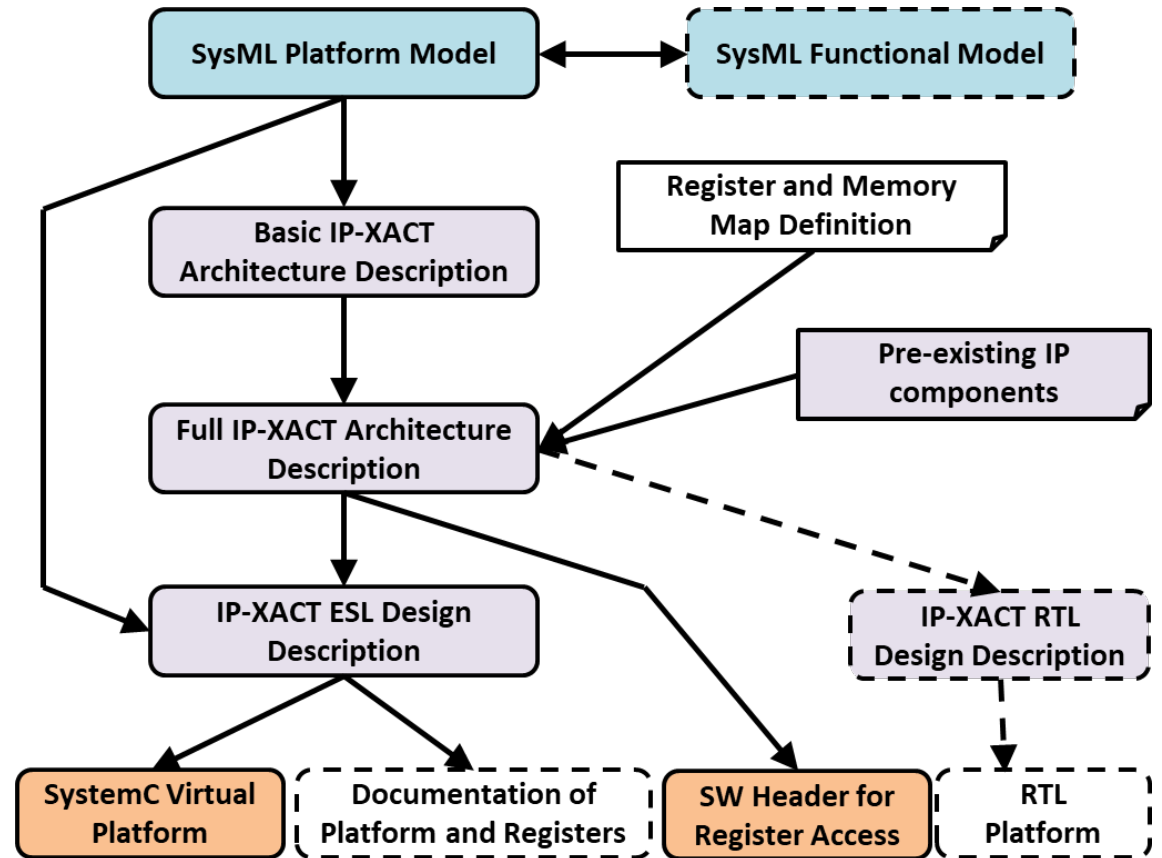


Platform Model Generation Flow



Platform Model Generation Flow (II)

- Integrated in existing design and system integration framework
 - Design tools
 - Directory structures
 - Documentation rules
 - ...
- All extensions need to fit in this framework

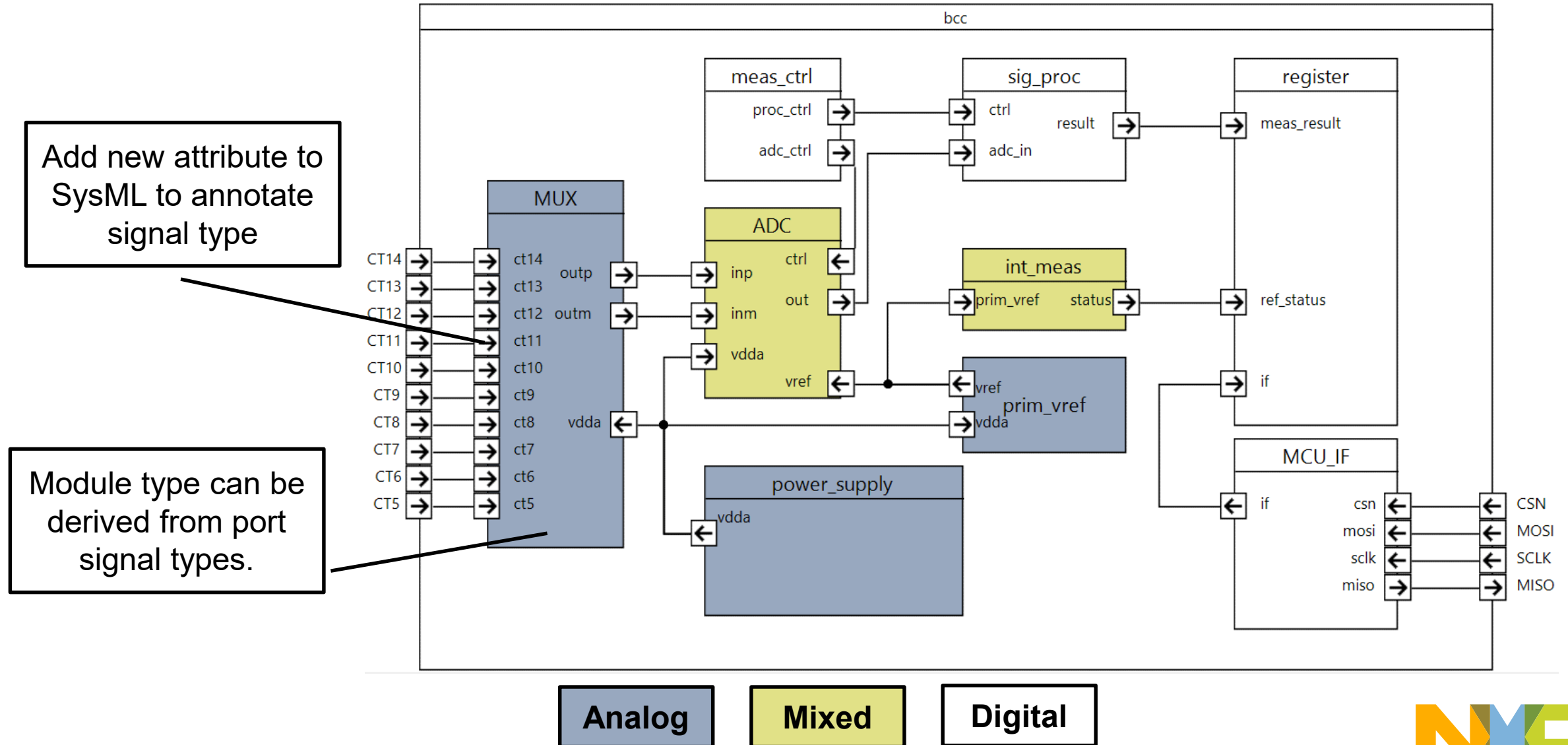


AMS Extensions for IP-XACT

- Accellera Vendor Extensions for Analog and Mixed Signal
 - Will become native element in next version of IP-XACT standard (IEEE 1685-202x)
- Extends IP-XACT wire port
- domainTypeDefs
 - Describes domain type of a port per view, e.g. electrical
- signalTypeDefs
 - Describes the signal semantics of a port per view

signalType	Verilog AMS	SystemC AMS
continuous conservative	electrical outp;	sca_eln::sca_terminal outp;
continuous non-conservative	sf_voltage outp;	sca_lsf::sca_out outp;
discrete	wreal outp;	sca_tdf::sca_out<double> outp;

Annotate AMS Attributes in Platform Model



Generation of IP-XACT Description

```
1  nxp::createComponent nxp.com bms bcc_ADC 2.0 ${nxp::workarea}/chip_lib/bcc_ADC/METADATA/
2  nxp::createWirePort inp_pi in ""
3  nxp::createWireTypeDefs inp_pi [ list bms::analog [ list ESL ] false [ list analog.h ] ]
4  nxp::createWirePort inm_pi in ""
5  nxp::createWireTypeDefs inm_pi [ list bms::analog [ list ESL ] false [ list analog.h ] ]
6  nxp::createWirePort vdda_pi in ""
7  nxp::createWireTypeDefs vdda_pi [ list bms::analog [ list ESL ] false [ list analog.h ] ]
8  nxp::createFileSet ESL \
9    [ list \
10     [ list ADClib \
11       [ list \
12         ../coside/lib/bcc_ADC.h \
13         ../coside/lib/bcc_ADC.cpp ]\
14     ]\
15  ]
16 nxp::generateXmlAmsExtension nxp.com bms bcc_ADC 2.0 \
17   [ list ] \
18   [ list \
19     [ list inp_pi discrete ESL ] \
20     [ list inm_pi discrete ESL ] \
21     [ list vdda_pi discrete ESL ] \
22   ]
```

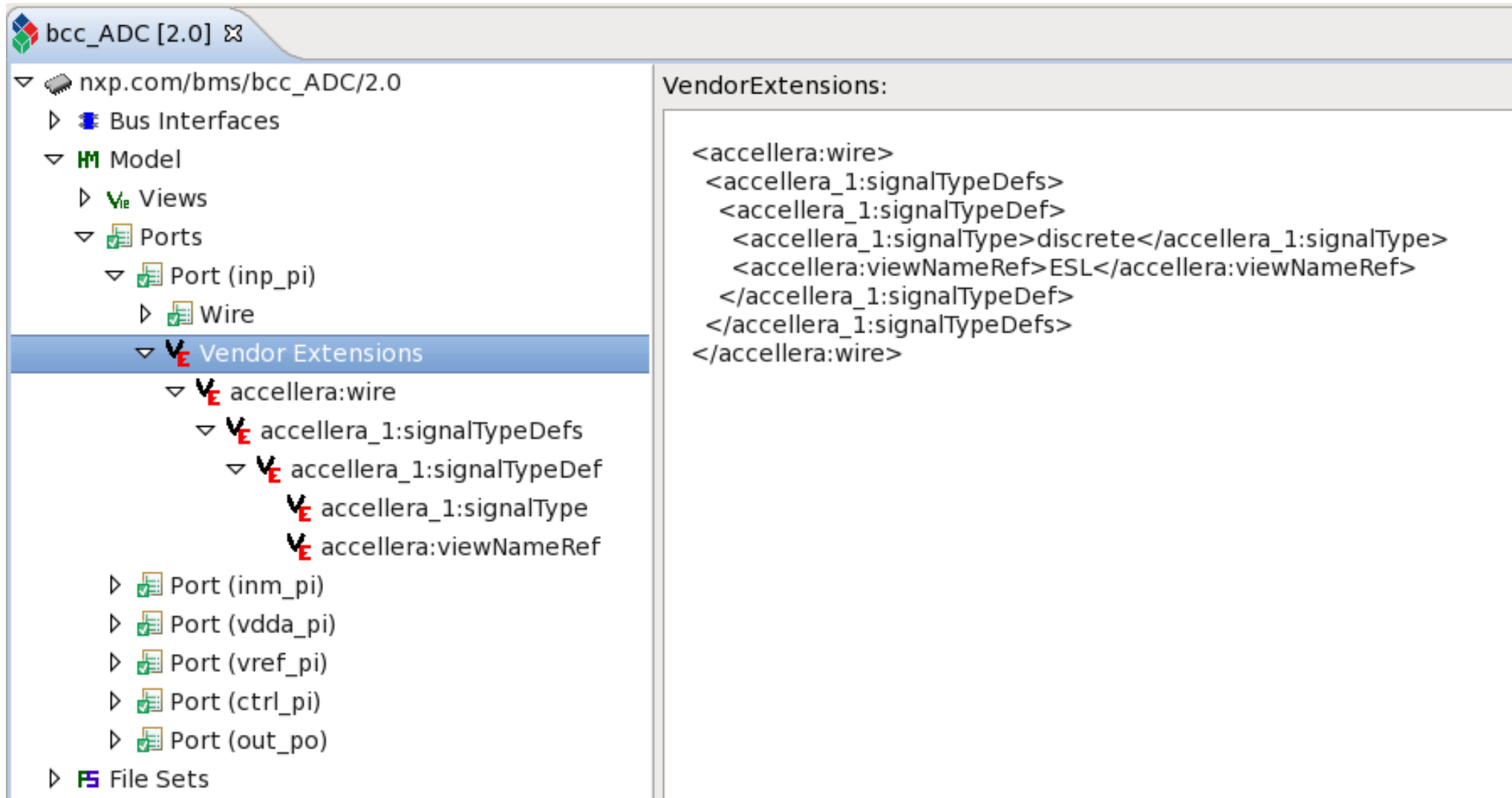
Create component

Create ports

Create fileset

Create port signal types

AMS Component in IP-XACT



The screenshot displays the IP-XACT editor interface for the component `bcc_ADC [2.0]`. The left pane shows a hierarchical tree of the component's structure:

- `nxp.com/bms/bcc_ADC/2.0`
 - Bus Interfaces
 - Model
 - Views
 - Ports
 - Port (inp_pi)
 - Wire
 - Vendor Extensions**
 - `accellera:wire`
 - `accellera_1:signalTypeDefs`
 - `accellera_1:signalTypeDef`
 - `accellera_1:signalType`
 - `accellera:viewNameRef`
 - Port (inm_pi)
 - Port (vdda_pi)
 - Port (vref_pi)
 - Port (ctrl_pi)
 - Port (out_po)
 - File Sets

The right pane displays the XML content for the selected `VendorExtensions` element:

```
<accellera:wire>
  <accellera_1:signalTypeDefs>
    <accellera_1:signalTypeDef>
      <accellera_1:signalType>discrete</accellera_1:signalType>
      <accellera:viewNameRef>ESL</accellera:viewNameRef>
    </accellera_1:signalTypeDef>
  </accellera_1:signalTypeDefs>
</accellera:wire>
```

Generate SystemC Model from IP-XACT

- Digital 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

Create Analog Components in COSIDE

The screenshot displays the COSIDE software interface, which is used for creating and configuring analog components. The main window is divided into several panes:

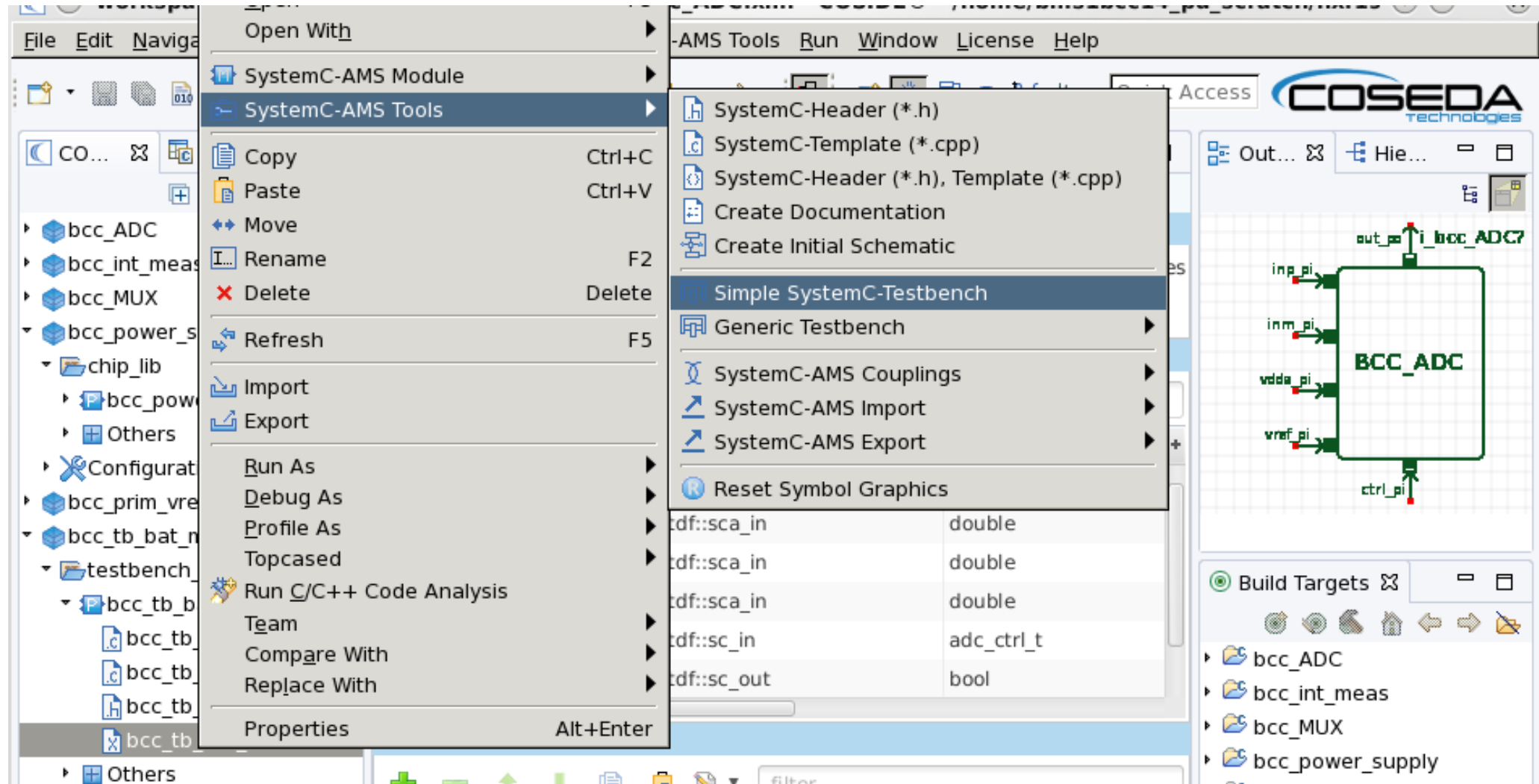
- Left Pane:** A project tree showing the hierarchy of components. The **bcc_ADC** component is selected under the **chip_lib** folder.
- Top Pane:** A menu bar with options: File, Edit, Navigate, Search, Project, Module Editor, SystemC-AMS Tools, Run, Window, License, Help. Below the menu is a toolbar with various icons for file operations and simulation.
- Center Pane:** The configuration window for the **bcc_ADC** component. It has tabs for Overview, Ports, Parameters, and Documentation. The **Overview** tab is active, showing the component's type (**sca_tdf::sca_module**) and a description field. Below this, the **Ports** section lists the component's inputs and outputs:

Name	Type	Data Type
inp_pi	sca_tdf::sca_in	double
inm_pi	sca_tdf::sca_in	double
vdda_pi	sca_tdf::sca_in	double
vref_pi	sca_tdf::sca_in	double
ctrl_pi	sca_tdf::sc_in	adc_ctrl_t
out_po	sca_tdf::sc_out	bool

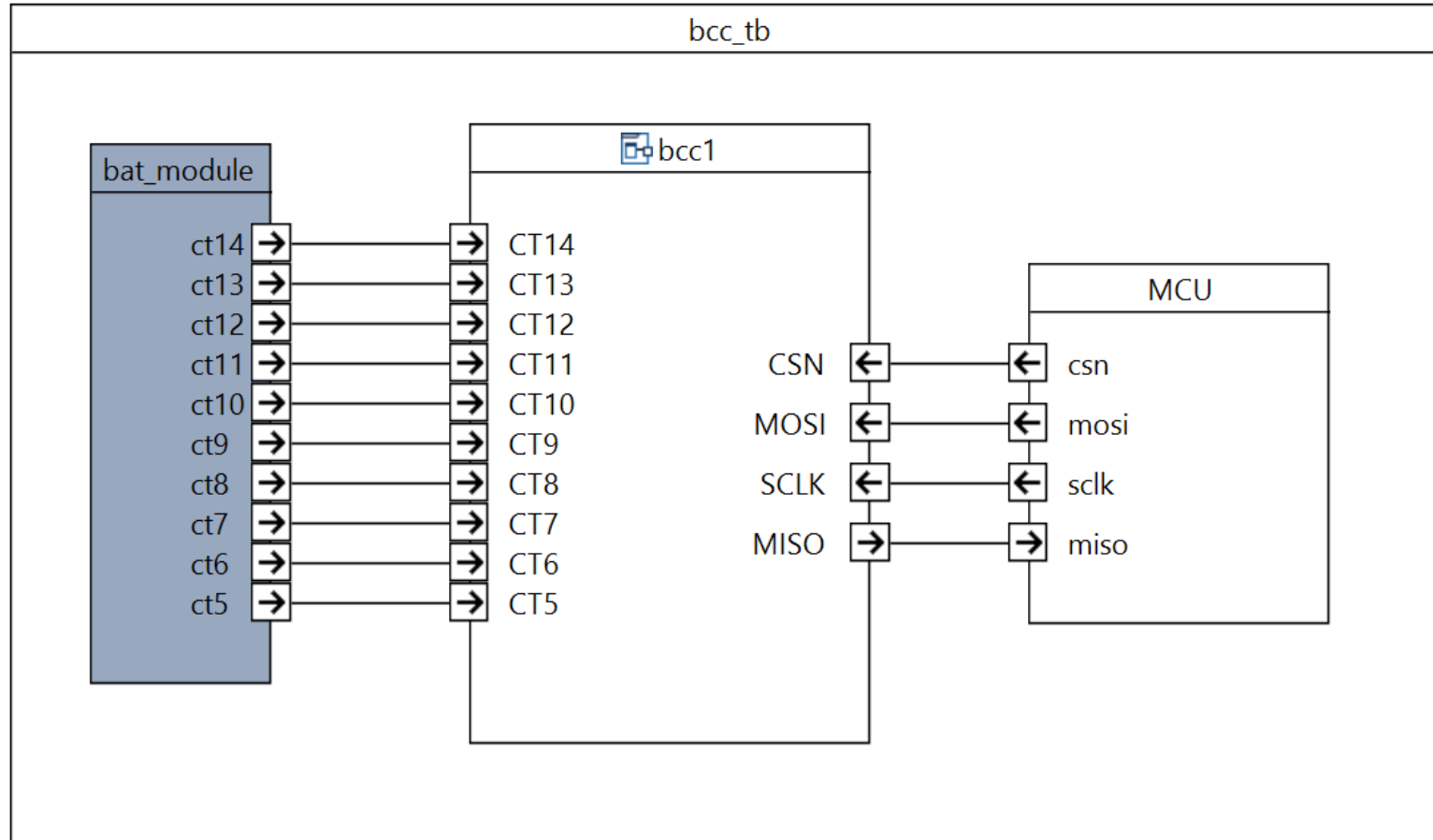
Below the ports section, the **Parameters** section is visible, showing a list of parameters with their names and default values.

- Right Pane:** A diagram showing the **BCC_ADC** component connected to other components. The component is represented by a green box with the label **BCC_ADC**. It has several input ports (inp_pi, inm_pi, vdda_pi, vref_pi, ctrl_pi) and one output port (out_po). The output port is connected to a component labeled **i_bcc_ADC7**.

Starting Point to Implement Component Behavior



Add Testbench to Complete Model



Compile and Run Virtual Prototype

```
File Edit View Scrollback Bookmarks Settings Help
ra/build $ bin/BMS1BCC

SystemC 2.3.1-Accellera --- Nov  6 2018 17:45:13
Copyright (c) 1996-2014 by all Contributors,
ALL RIGHTS RESERVED

SystemC AMS extensions 2.1.0-COSED A Release date: 20160404
Copyright (c) 2010-2014  by Fraunhofer-Gesellschaft IIS/EAS
Copyright (c) 2015-2016  by COSEDA Technologies GmbH
Licensed under the Apache License, Version 2.0

Info: SystemC-AMS:
  6 SystemC-AMS modules instantiated
  1 SystemC-AMS views created
  6 SystemC-AMS synchronization objects/solvers instantiated

Info: SystemC-AMS:
  1 dataflow clusters instantiated
    cluster 0:
      6 dataflow modules/solver, contains e.g. module: bcc.bat_module
      6 elements in schedule list,
      1 us cluster period,
      ratio to lowest: 1          e.g. module: bcc.bat_module
      ratio to highest: 1 sample time e.g. module: bcc.bat_module
      2 connections to SystemC de, 1 connections from SystemC de
```

Summary and Conclusion



- Model-based top-down flow for safety-critical semiconductor products
- Automated flow to generate skeleton for executable VP form platform model
- Flow extension for AMS based on IP-XACT AMS extensions and COSIDE

Conclusion

- Good integration in existing system integration environment
- COSIDE fits well for AMS component modeling and refinement
- Manual creation of projects and modules in COSIDE breaks automation
 - Script interface to control basic COSIDE functions would be helpful

Questions?

Context of this work:

SysML Based Architecture Definition and Platform Generation Flow

DVCon Europe 2019 | Oct 30, 15:15 - 16:45 | Forum 6: System Level Design



SECURE CONNECTIONS
FOR A SMARTER WORLD