KÄRNTEN University of Applied Sciences

Next Generation System Level Design for Electronic Based Systems

Lecture Master Thesis Seminar - 7th May 2020

Wolfgang Scherr

Engineering & IT, ISCD

KÄRNTEN University of Applied Sciences

Agenda

- Introduction: Why using models?
- How can IEEE1666 help?
- Examples
- Summary

WWW.FH-KAERNTEN.AT

4 typical Challenges in Chip Design

1) System complexity enabled by modern technologies

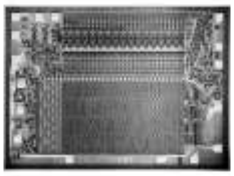
The wild 60s through a peephole…

Talbert and Widlar's µA709 high-performance operational amplifier (1965)

TTL gate design (1964)

Don Forbes, Rex Rice, and Bryant ("Buck") Rogers developed the DIL package (1965), before mainly "TO" (transistor outline) formed as metal cans where used

Micromosaic - standard cell design for GE Avionics (1968)


1024bit bipolar ROM Intel 3301 (1965)

Laurence W. Nagel

IBM 360/67 mainframe-powered CAD system at Fairchild in 1967 (for layout purposes)

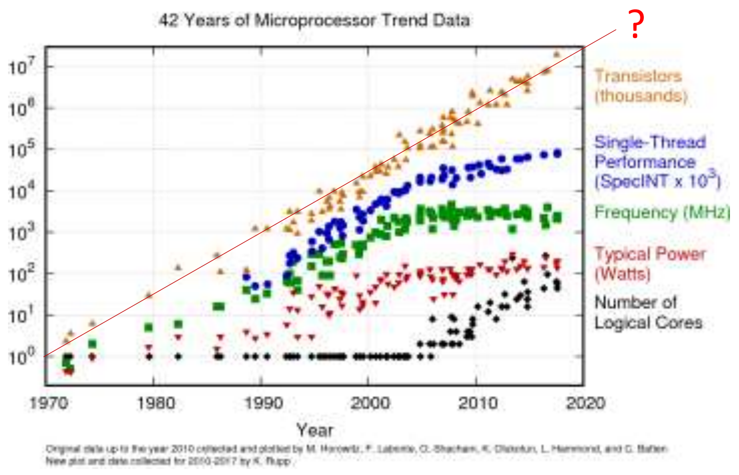
Ronald A. Rohrer and Donald O. Peterson: a practical course on circuit simulation led to development of CANCER (Computer Analysis of Nonlinear Circuits Excluding Radiation) presented by Ron and Laurence for Berkeley at IEEE ISSCC in 1971

The students developing CANCER: Laurence W. Nagel, Bob Berry, Shi-Ping Fan, Frank Jenkins, Jesse Pipkin, Steve Ratner, Lynn Weber

Gordon Moore predicts the rate of rising semiconductor complexity in 1965 (and revised it to a lower rate at a IEEE conference in 1975)

By the way…
Siemens started in Villach
In 1970 with diode production,
first fab in 1972

Transistor count is still rising…



Even when disregarding exact quantitative statistics:

.) Transistor count/chip is still rising

.) Chips can implement more and more functions, more stuff on PCBs will be integrated into the chips

.) Thus, also EBS will further increase in complexity

.) Systems get larger too (e.g. energy harvesting in cars including geoinformation and 5G comm. between cars)

Creative Commons Attribution 4.0 - graph *used from: https://www.karlrupp.net/wp-content/uploads/2018/02/42-years-processor-trend.png*
data available here: https://github.com/karlrupp/microprocessor-trend-data

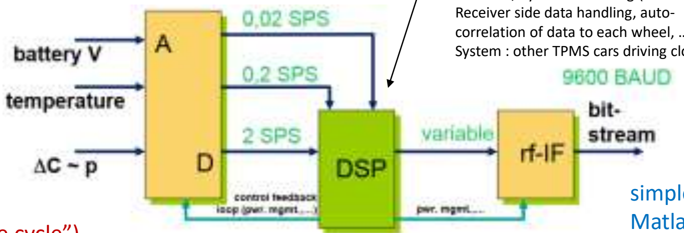
WWW.FH-KAERNTEN.AT

Why is complexity especially a problem of the system engineers?

TPMS: active measurement especially for run-flat tires; ~10yrs with a 250mAh battery
Classic principle: measure pressure and g-force to transmit of data via RF only if needed
Idea: only measure pressure, no g-sensor for a single-chip solution

Stop/go algorithms (wakeup)
Baseband modulation

Not shown:
Channel / system modelling (4 tires)
Receiver side data handling, auto-correlation of data to each wheel, …
System : other TPMS cars driving close, …



simple Matlab model

This you need for system validation (e.g. “one drive cycle”).
→ “overall system”: 4MHz/140µHz = **29·10⁹** w/o RF
434MHz/140µHz = **3·10¹²** /w RF

Required for firmware verification (e.g. “full update cycle”)
→ “software system”: 4MHz/20mHz = 200·10⁶ w/o RF
→ “algorithm check”: 2Hz/140µHz = 15·10³ w/o RF

OK for mixed-signal top level verification (e.g. “send a bit”)
→ “chip system”. 434MHz/9600Hz = 45·10³ /w RF

TPMS sensor system setup:

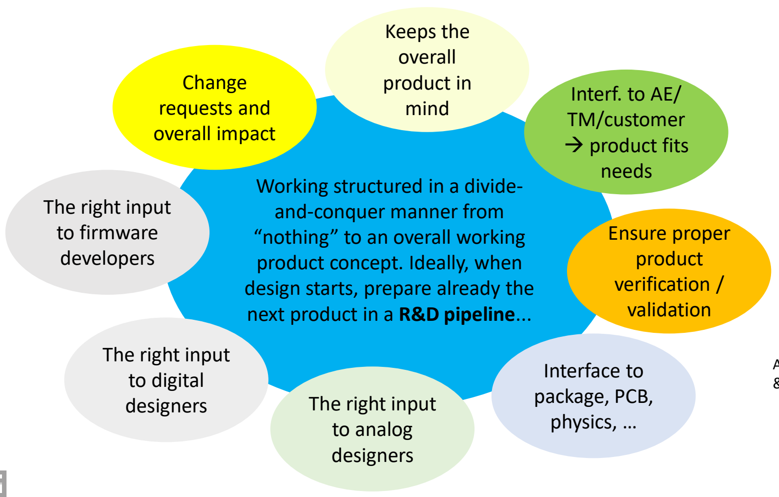
- Application (driving) 2 hrs (1/t ~ 140µHz)
- Battery voltage update rate ~20mHz
- Temperature update rate ~200mHz
- Pressure update rate (wakeup algo) ~2Hz
- Data transmission 9600 bits/s
- Analog processing rate (ADC) ~1MHz
- Digital processing rate (DSP) ~4MHz
- RF transmission (ISM band) ~434MHz

(How) is 10yrs with 250mAh feasible?

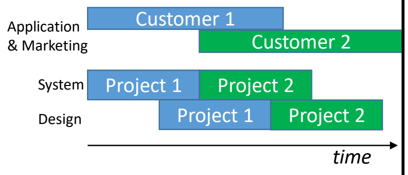
System engineers role   Designers role

WWW.FH-KAERNTEN.AT

Role of a System/Concept Engineer (not only) in Chip Development


Working structured in a divide-and-conquer manner from "nothing" to an overall working product concept. Ideally, when design starts, prepare already the next product in a **R&D pipeline**…
Change requests and overall impact
Keeps the overall product in mind
Interf. to AE/ TM/customer → product fits needs
Ensure proper product verification / validation
Interface to package, PCB, physics, …
The right input to analog designers
The right input to digital designers
The right input to firmware developers

What is a R&D pipeline ?



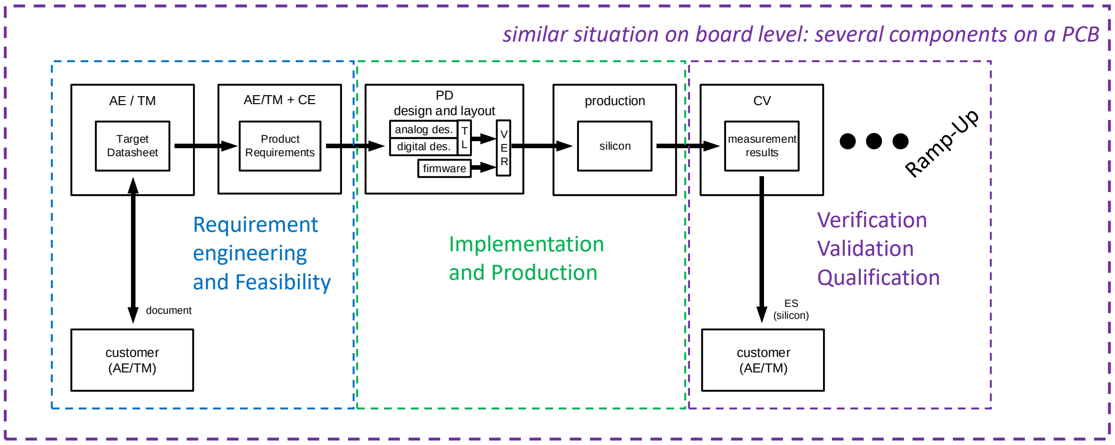
WWW.FH-KAERNTEN.AT

4 typical Challenges in Chip Design

1) High complexity enabled by modern technologies
2) Development process related topics

WWW.FH-KAERNTEN.AT

A classic R&D process



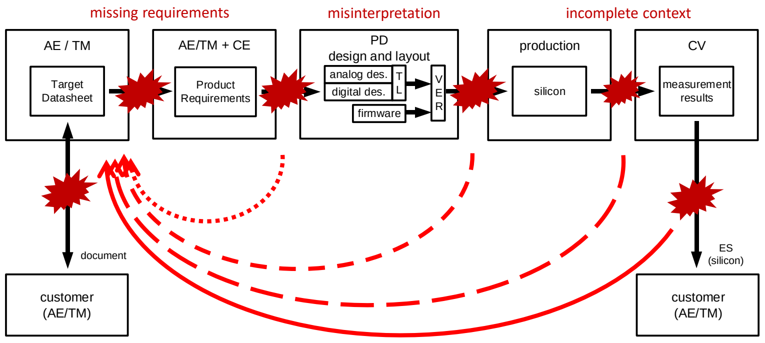
AE=Application Engineering
TM=Technical Marketing
CE=Concept Engineering
PD=Product Design
CV=Component Verification

(redrawn and modified) from: Simon Hainz, Infineon Technologies AG,
NASCUG Conference 2012

WWW.FH-KAERNTEN.AT

R&D design: the “sword of Damocles”

- A lot of things can go wrong…



(redrawn and modified) from: Simon Hainz, Infineon Technologies AG,
NASCUG Conference 2012

WWW.FH-KAERNTEN.AT

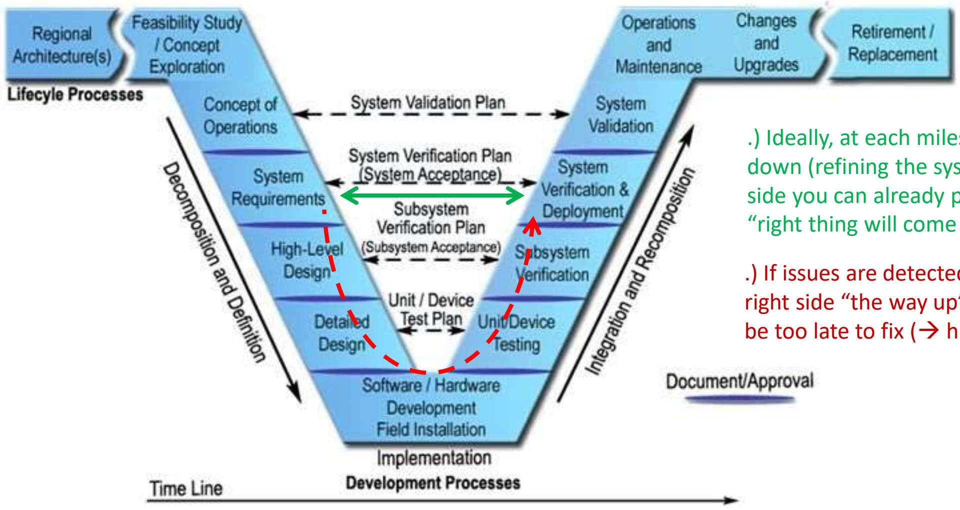
4 typical Challenges in Chip Design

1) High complexity enabled by modern technologies

2) Process related topics

3) Tool related topics in conjunction with the top-down V-model

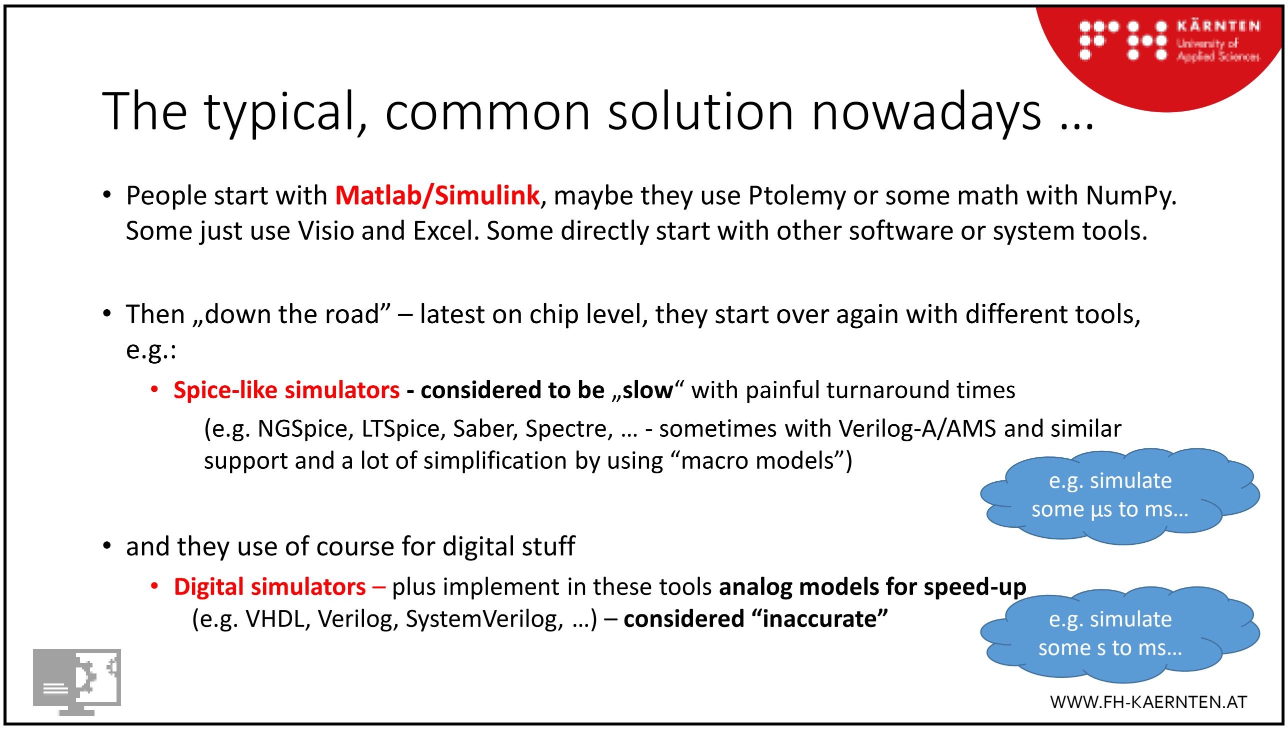
The V-Model – an ideal top-down approach?



.) Ideally, at each milestone the way down (refining the system) on the left side you can already proof the “right thing will come out at the end”

.) If issues are detected late on the right side “the way up”, it may be too late to fix (→ high risk !)

Ref.: S. Malaek at.al.,”A New Systems Engineering Model Based on The Principles of Axiomatic Design”, International Conference on Industrial Engineering and Applications, 2014



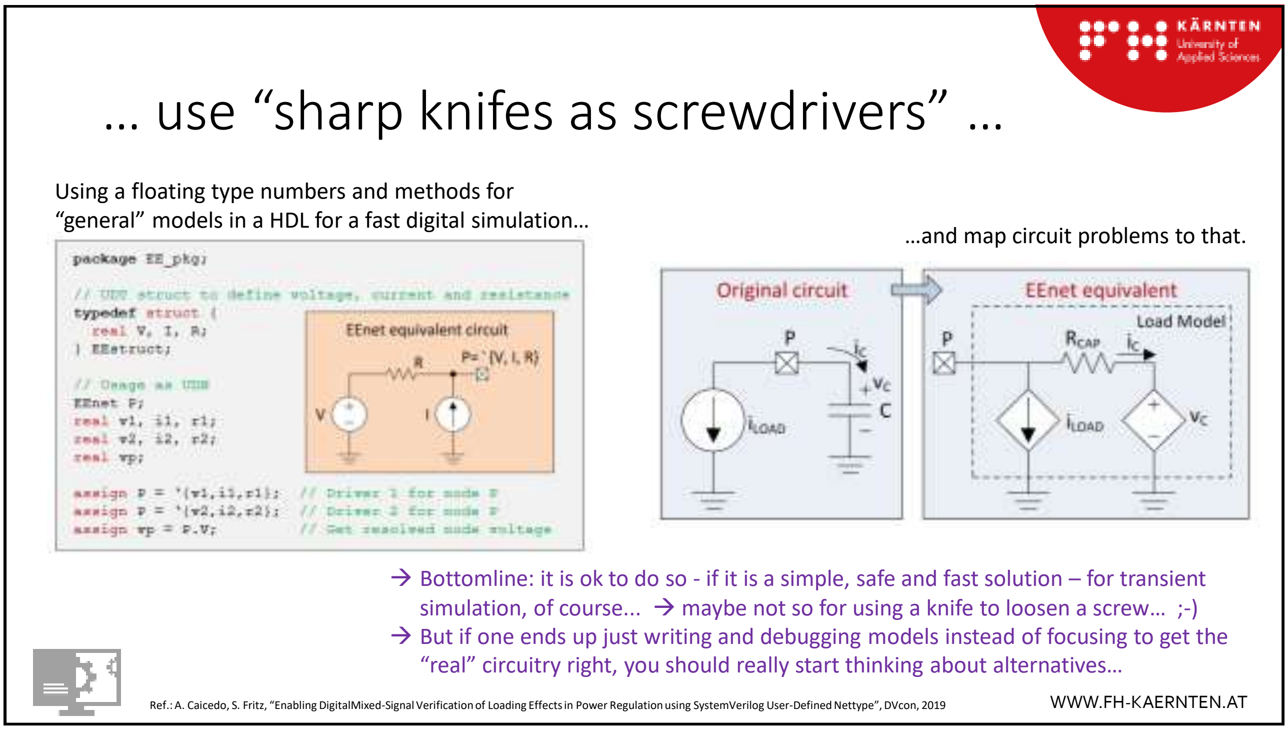
The typical, common solution nowadays …

- People start with **Matlab/Simulink**, maybe they use Ptolemy or some math with NumPy. Some just use Visio and Excel. Some directly start with other software or system tools.
- Then „down the road" – latest on chip level, they start over again with different tools, e.g.:
  - **Spice-like simulators - considered to be „slow"** with painful turnaround times
    (e.g. NGSpice, LTSpice, Saber, Spectre, … - sometimes with Verilog-A/AMS and similar support and a lot of simplification by using "macro models")
- and they use of course for digital stuff
  - **Digital simulators –** plus implement in these tools **analog models for speed-up**
    (e.g. VHDL, Verilog, SystemVerilog, …) – **considered "inaccurate"**

e.g. simulate some µs to ms…

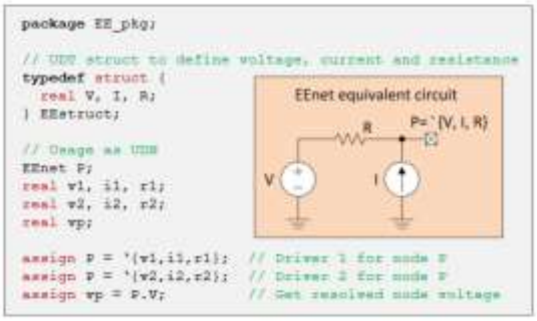
e.g. simulate some s to ms…

WWW.FH-KAERNTEN.AT



… use "sharp knifes as screwdrivers" …

Using a floating type numbers and methods for "general" models in a HDL for a fast digital simulation…



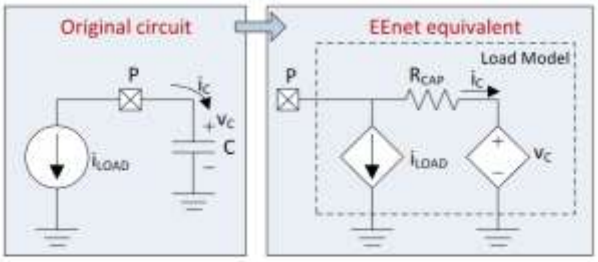
```
package EE_pkg;

// UDT struct to define voltage, current and resistance
typedef struct {
  real V, I, R;
} EEstruct;

// Usage as UDN
EEnet P;
real v1, i1, r1;
real v2, i2, r2;
real vp;

assign P = '{v1,i1,r1};   // Driver 1 for node P
assign P = '{v2,i2,r2};   // Driver 2 for node P
assign vp = P.V;          // Get resolved node voltage
```

…and map circuit problems to that.

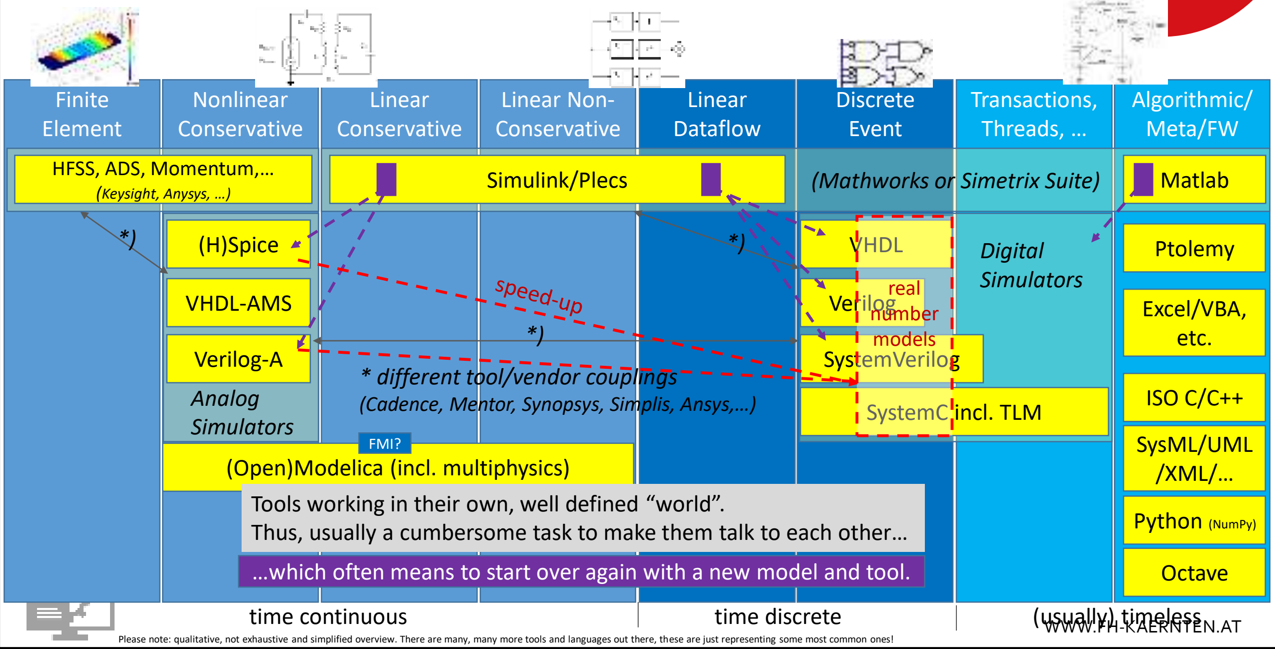


→ Bottomline: it is ok to do so - if it is a simple, safe and fast solution – for transient simulation, of course… → maybe not so for using a knife to loosen a screw… ;-)
→ But if one ends up just writing and debugging models instead of focusing to get the "real" circuitry right, you should really start thinking about alternatives…

Ref.: A. Caicedo, S. Fritz, "Enabling DigitalMixed-Signal Verification of Loading Effects in Power Regulation using SystemVerilog User-Defined Nettype", DVcon, 2019

WWW.FH-KAERNTEN.AT

… and end up doing things at least twice.



* different tool/vendor couplings (Cadence, Mentor, Synopsys, Simplis, Ansys,…)

Tools working in their own, well defined "world".
Thus, usually a cumbersome task to make them talk to each other…

…which often means to start over again with a new model and tool.

Please note: qualitative, not exhaustive and simplified overview. There are many, many more tools and languages out there, these are just representing some most common ones!

WWW.FH-KAERNTEN.AT

4 typical Challenges in Chip Design

1) High complexity enabled by modern technologies

2) Process related topics

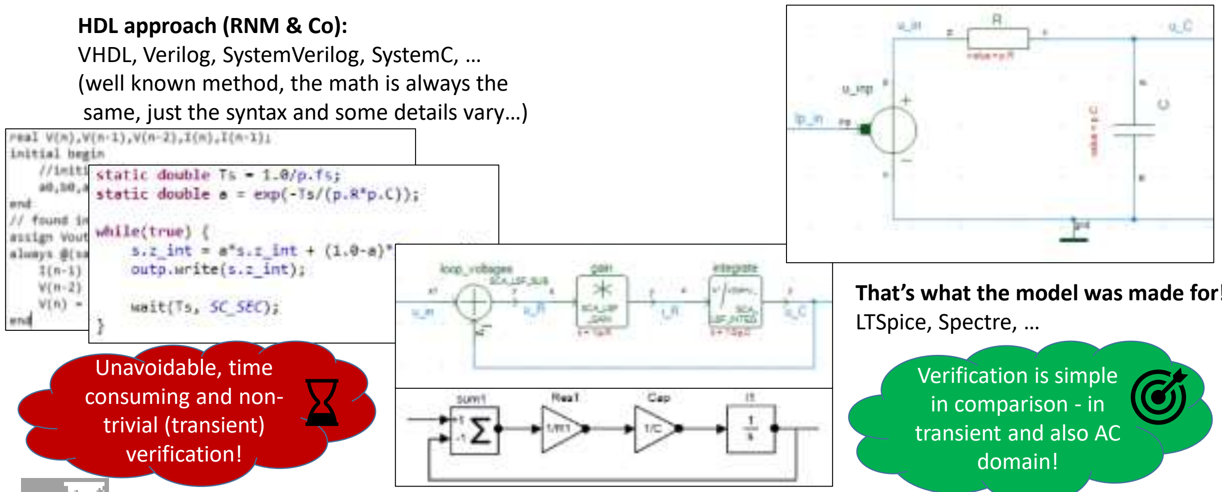
3) Tool related topics in conjunction with the top-down V-model

4) Verification and Validation

WWW.FH-KAERNTEN.AT

What is the function of this model?

HDL approach (RNM & Co):
VHDL, Verilog, SystemVerilog, SystemC, …
(well known method, the math is always the same, just the syntax and some details vary…)



```
real V(n),V(n-1),V(n-2),I(n),I(n-1);
initial begin
    //initi
    a0,b0,a
end
// found in
assign Vout
always @(sa
    I(n-1)
    V(n-2)
    V(n) -
end
```

```
static double Ts = 1.0/p.fs;
static double a = exp(-Ts/(p.R*p.C));

while(true) {
    s.z_int = a*s.z_int + (1.0-a)*
    outp.write(s.z_int);

    wait(Ts, SC_SEC);
}
```

Unavoidable, time consuming and non-trivial (transient) verification!

That's what the model was made for!
LTSpice, Spectre, …

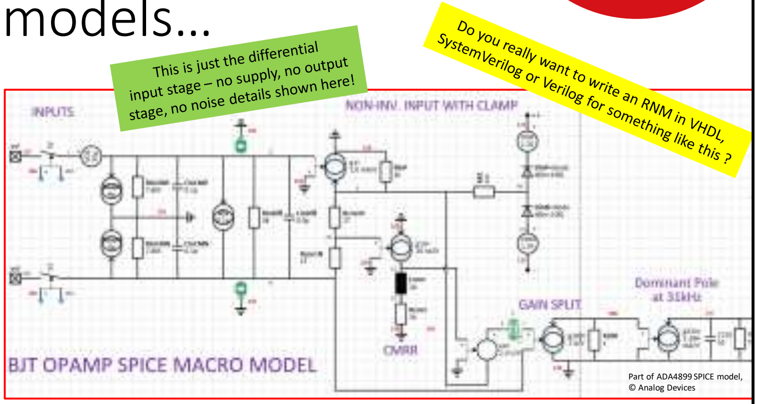
Verification is simple in comparison - in transient and also AC domain!

ODE / equation based approach, like:
Matlab/Simulink, Simetrix, …

WWW.FH-KAERNTEN.AT

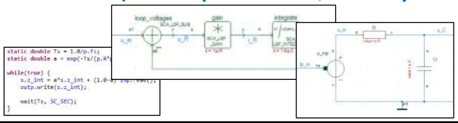
The problem with models…

- One need to spend quite some **effort** to model rather simple electrical behavior
- The **complexity** to create the model also ends up in complexity for **verification**
- Ignoring that will lead for sure in a redesign!



This is just the differential input stage – no supply, no output stage, no noise details shown here!
Do you really want to write an RNM in VHDL, SystemVerilog or Verilog for something like this ?
INPUTS
NON-INV. INPUT WITH CLAMP
GAIN SPLIT
Dominant Pole at 35kHz
CMRR
BJT OPAMP SPICE MACRO MODEL
Part of ADA4899 SPICE model, © Analog Devices

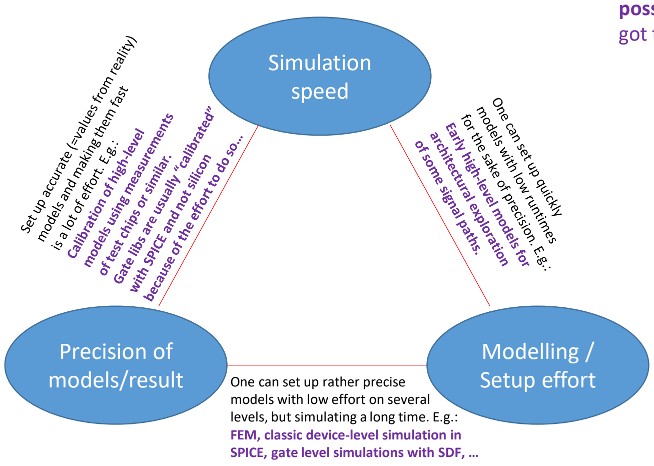
- Wouldn't it be nice to have **always the option to mix & match** the best possible representation for the model – even "correct by design" - and it is even faster than a "HDL coded" version (to implement, verify and simulate)?



WWW.FH-KAERNTEN.AT

Modelling triangle

- **Speed** versus **effort** versus **precision**



Set up accurate (=values from reality) models and making them fast is a lot of effort. E.g.: **Calibration of high-level models using measurements of test chips or similar. Gate libs are usually "calibrated" with SPICE and not silicon because of the effort to do so…**

One can set up quickly models with low runtimes for the sake of precision. E.g.: **Early high-level models for architectural exploration of some signal paths.**

One can set up rather precise models with low effort on several levels, but simulating a long time. E.g.: **FEM, classic device-level simulation in SPICE, gate level simulations with SDF, …**

BUT WHO CARES !

Selection is **rarely done by the best possible solution**, but what you have got to do the job…

… or are there any alternatives ?

WWW.FH-KAERNTEN.AT

Now we are "motivated enough"… 🙄

What is IEEE 1666 (.1) and how can it help?

WWW.FH-KAERNTEN.AT

SystemC/SystemC-AMS today (2020)

- Actively developed in Accellera working group by several semiconductor companies and EDA vendors

  (there are also WG for SystemVerilog etc….)

- IEEE Standard 1666 – 2011 (SystemC)
- IEEE Standard 1666.1 – 2016 (SystemC-AMS)

- Open-source reference implementation with examples
- New, extended users manual

http://www.es.ele.tue.nl/~heco/courses/ProcDesign/SystemCUserGuide.pdf
https://www.accellera.org/images/downloads/standards/systemc/Accellera_SystemC_AMS_Users_Guide_January_2020.pdf

WWW.FH-KAERNTEN.AT

History to latest LRMs released

1999:	Open SystemC Initiative (OSCI) announced
2000:	SystemC 1.0 released (sourceforge.net)
2002:	OSCI SystemC 1.0.2
2005:	**IEEE Std 1666-2005 LRM**
2005:	SystemC Transaction level modeling (TLM) 1.0 released
2007:	SystemC 2.2 released
2009:	SystemC TLM 2.0 standard
2009:	SystemC Synthesizable Subset Draft 1.3
2011:	**IEEE Std 1666-2011 LRM**

time

~2000:	First C-based AMS initiatives (AVSL, MixSigC)
2002:	SystemC-AMS study group started
2005:	First SystemC-AMS PoC released by Fraunhofer
2006:	OSCI AMSWG installed
2008:	SystemC AMS Draft 1 LRM
2010:	**SystemC AMS 1.0 LRM standard**
2010:	SystemC AMS 1.0 PoC released by Fraunhofer IIS/EAS
2012:	SystemC AMS 2.0 draft standard
2013:	**SystemC AMS 2.0 LRM standard**
2013:	SystemC AMS 2.0 PoC test version
2014:	IEEE 1666.1 (SystemC AMS) started
2016:	**IEEE Std 1666.1-2016 LRM**

From: COSEDA technologies presentation, K. Einwich, DVCon 2014

WWW.FH-KAERNTEN.AT

KÄRNTEN University of Applied Sciences

Definition

- **SystemC is an ANSI standard C++ class library** for system and hardware design for use by designers and architects who need to address complex systems that are a hybrid between hardware and software.
- **SystemC AMS is an ANSI standard C++ class library** for electronic system-level design and modeling for use by system architects and engineers who need to address complex heterogeneous systems that are a hybrid between analog, digital and software components.
- Example:

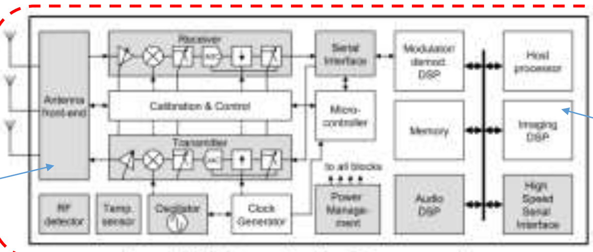


Figure 1.1—A Communication System, example of an heterogeneous AMS/HW/SW architecture

SystemC-AMS

SystemC

One seamless and fully standardized environment!

From:
https://standards.ieee.org/standard/1666-2011.html, https://standards.ieee.org/standard/1666_1-2016.html
https://www.accellera.org/images/downloads/standards/systemc/Accellera_SystemC_AMS_Users_Guide_January_2020.pdf

KÄRNTEN University of Applied Sciences

High-level System Modeling

Figure 4. Snapshots of SystemC-AMS code: (a) TDF code implementing an example of mechanical model of EV power consumption and (b) ELN circuit equivalent model of a battery.

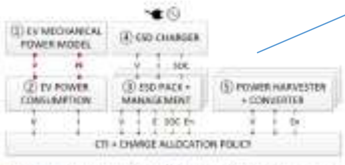
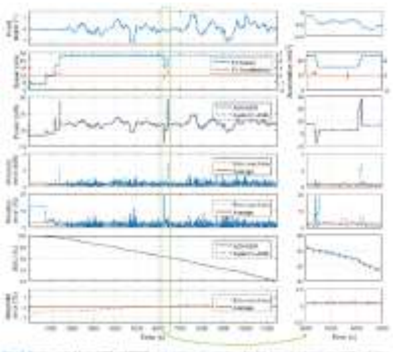

① EV MECHANICAL POWER MODEL
④ ESD CHARGER
② EV POWER CONSUMPTION
③ ESD PACK + MANAGEMENT
⑤ POWER HARVESTER + CONVERTER
CTI + CHARGE ALLOCATION POLICY

Figure 1. Structure of the proposed template architecture for EV simulation in SystemC-AMS: the figure shows the typical components of the EV energy sub-domain and their interfaces, formalizing the energy and information flows in the system. Type and meaning of the signals are described in Section III-B.

Figure 5. Example of driving route: (a) road traffic information and (b) road slope information.



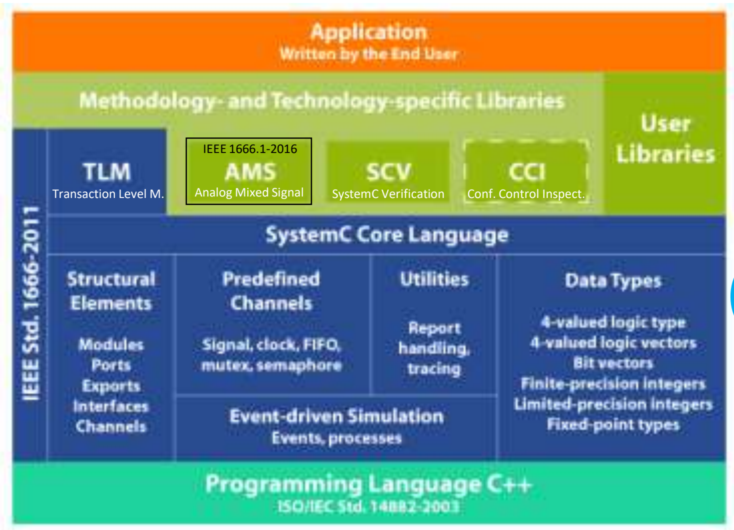
[illegible]

Table 1. Coefficients for Tesla Model 3

η	0.098	β	10.213	γ	0.989	α	[illegible]
C_D	1.800	C_1	164.868	C_2	0.01	C_3	[illegible]
[illegible]	0.850	[illegible]	7,790,000				

Ref.: Yukai Chen et al., "A SystemC-AMS Framework for the Design and Simulation of Energy Management in Electric Vehicles", as IEEE access (marked as preliminary article), 2019

IEEE 1666 is more than AMS modeling…

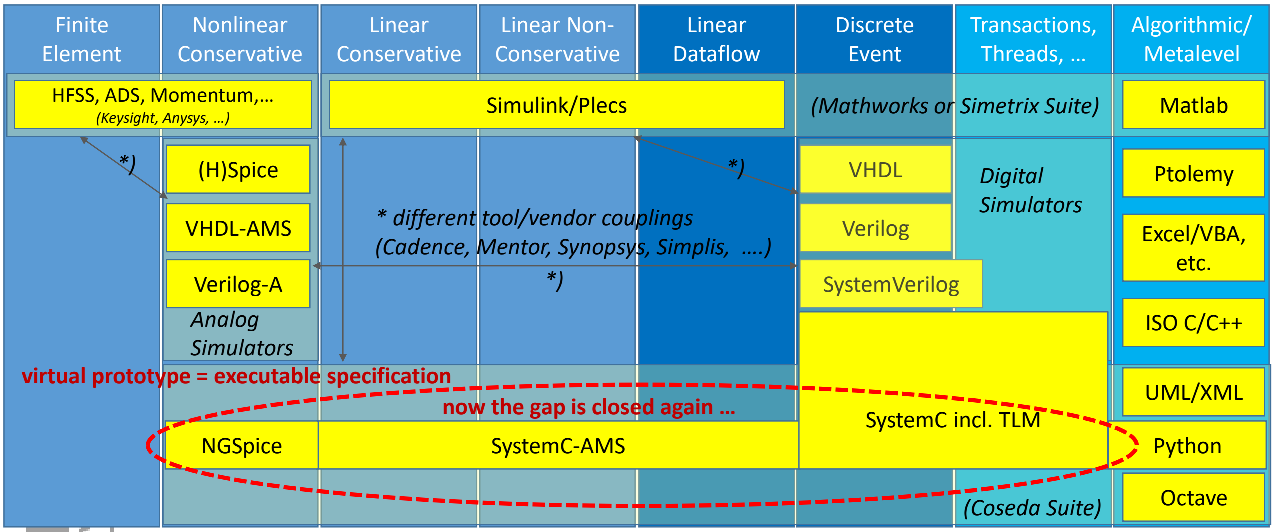


One monolithic solution – instead of side-by-side approaches like with VHDL/Verilog/.. and their –AMS addons!

From (modified): https://www.accellera.org/community/systemc/about-systemc

IEEE 1666 allows new approaches by combining MoCs and virtual prototyping

SystemC-AMS allows a truly heterogenous modelling and simulation setup

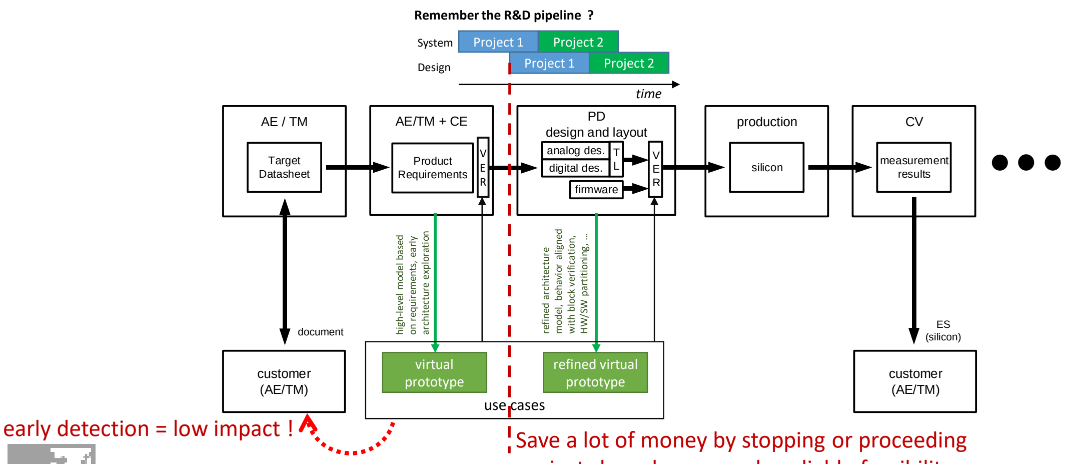


Please note: qualitative, not exhaustive and simplified overview. There are many, many more tools and languages out there, these are just representing some most common ones!

WWW.FH-KAERNTEN.AT

R&D design: early prevention measures

- Improve the flow in an early stage saves time and cost later !

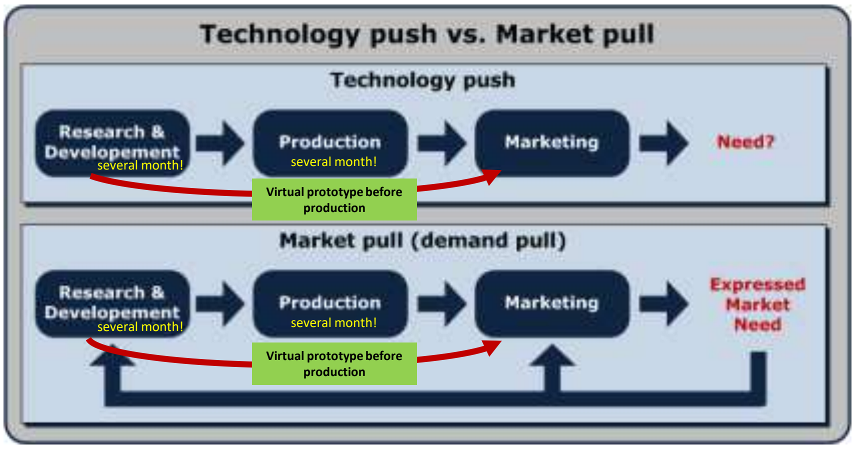


Save a lot of money by stopping or proceeding projects based on an early, reliable feasibility, before a designer even opens Cadence & Co…

(redrawn and modified) from: Simon Hainz, Infineon Technologies AG, NASCUG Conference 2012

WWW.FH-KAERNTEN.AT

Virtual prototypes are not new – but IEEE 1666 makes usability much broader



Examples:

INNOVATION:
Development and introduction of the world first smart phone.

EVOLUTION:
Next generation of a solar inverter.

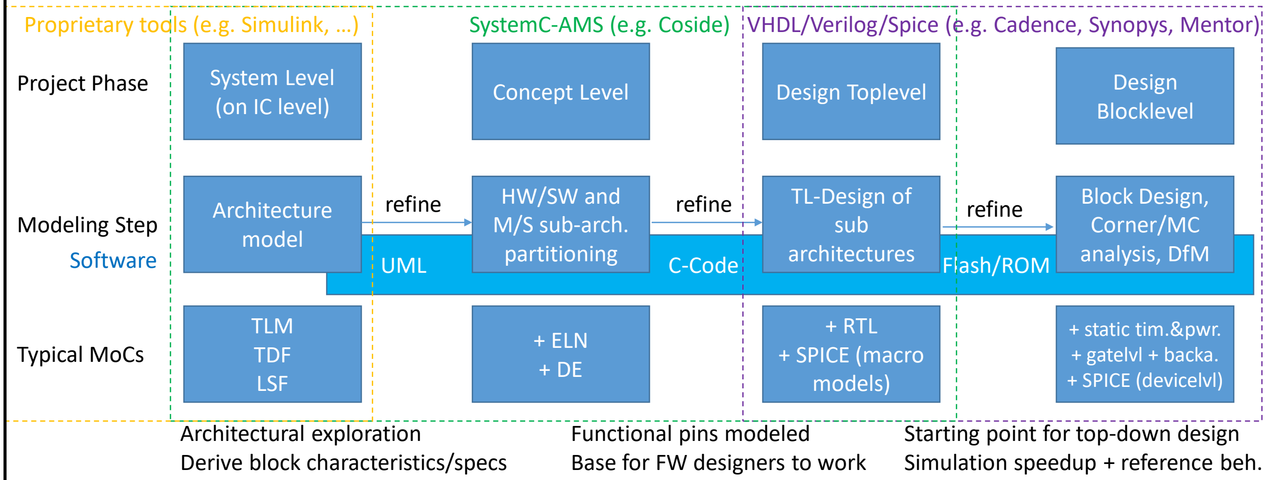
Source (modified):
https://de.wikipedia.org/wiki/Push-Pull-Strategie

WWW.FH-KAERNTEN.AT

Now we can talk "top-down" – in many different perspectives…

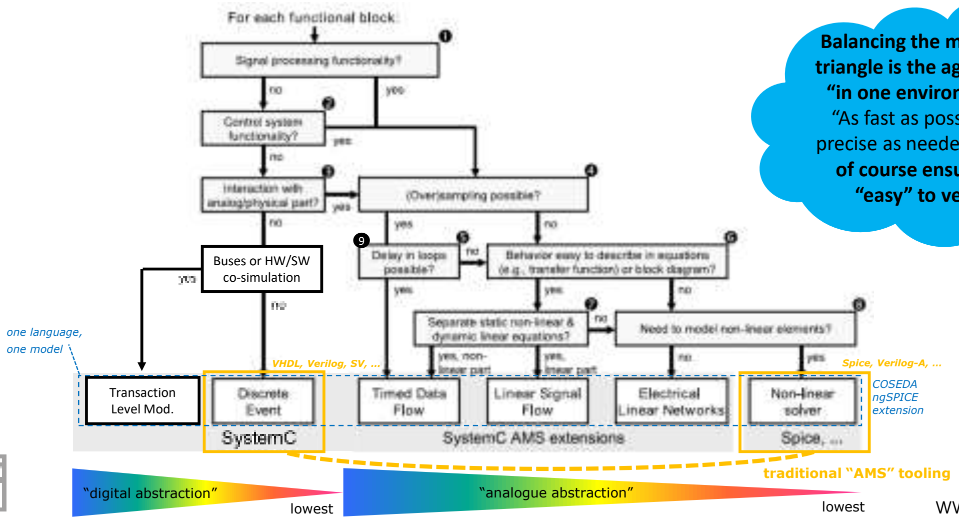
WWW.FH-KAERNTEN.AT

→ an agile top-down work flow also for analog


Proprietary tools (e.g. Simulink, …)
SystemC-AMS (e.g. Coside)
VHDL/Verilog/Spice (e.g. Cadence, Synopys, Mentor)
Project Phase
System Level (on IC level)
Concept Level
Design Toplevel
Design Blocklevel
Modeling Step
Software
Architecture model
refine
HW/SW and M/S sub-arch. partitioning
refine
TL-Design of sub architectures
refine
Block Design, Corner/MC analysis, DfM
UML
C-Code
Flash/ROM
Typical MoCs
TLM TDF LSF
+ ELN + DE
+ RTL + SPICE (macro models)
+ static tim.&pwr. + gatelvl + backa. + SPICE (devicelvl)
Architectural exploration
Derive block characteristics/specs
Functional pins modeled
Base for FW designers to work
Starting point for top-down design
Simulation speedup + reference beh.

Reworked, from SystemC-AMS users guide, 2010

Choosing the right level of abstraction within the SystemC(-AMS) environment



Balancing the modelling triangle is the again key – "in one environment"! "As fast as possible, as precise as needed." – **and of course ensure it is "easy" to verify!**

PID controller case study for virtual prototyping

- Transaction Level Model
- Discrete Event
- Timed Data Flow
- Linear Signal Flow
- Electric Linear Network

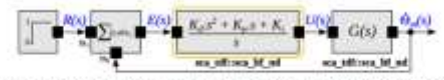


Fig. 5. Analog controller: TDF models of PID and motor in closed loop.

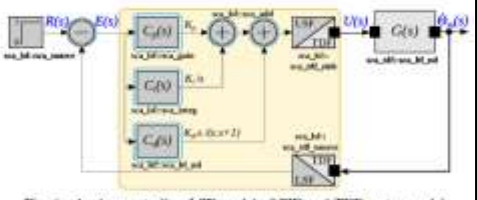


Fig. 6. Analog controller: LSF model of PID and TDF motor model.

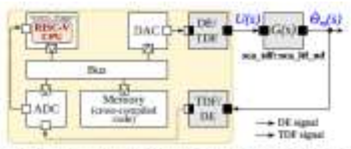


Fig. 3. Digital controller: TLM platform and TDF motor model in closed loop.

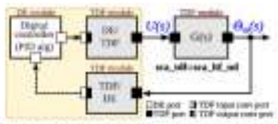


Fig. 2. Executable specification of PID controller and TDF motor model in closed loop.

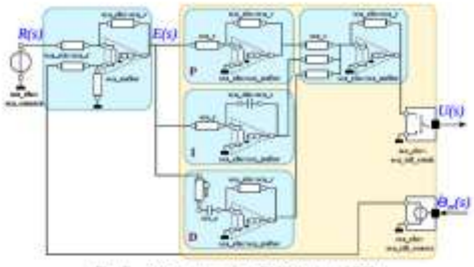


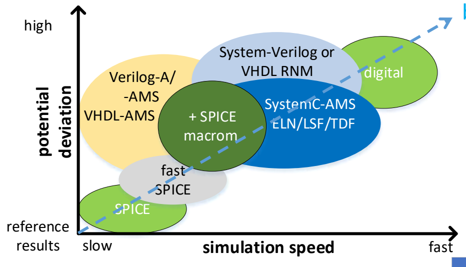
Fig. 7. Analog controller: ELN model of PID.

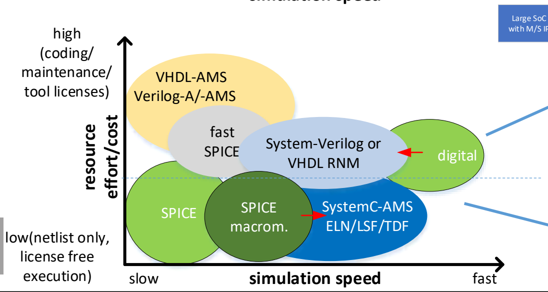
Ref.: Breytner Fern´andez-Mesa et al., "Electronic System Level Design of Heterogeneous Systems: a Motor Speed Control System Case Study", as IEEE access (marked as preliminary article), 2019

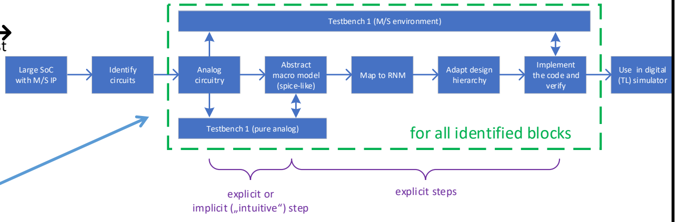
M/S modelling approaches - a qualitative view

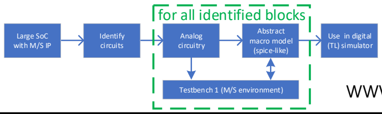
In general a path from the original design on device level to a very simplified model mainly by reduction of details and simplifying MoCs.

Some details (maybe) presented in IEEE FDL 2020





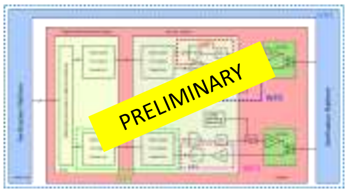

Testbench 1 (M/S environment)
Large SoC with M/S IP
Identify circuits
Analog circuitry
Abstract macro model (spice-like)
Map to RNM
Adapt design hierarchy
Implement the code and verify
Use in digital (TL) simulator
Testbench 1 (pure analog)
for all identified blocks
explicit or implicit („intuitive") step
explicit steps


for all identified blocks
Large SoC with M/S IP
Identify circuits
Analog circuitry
Abstract macro model (spice-like)
Use in digital (TL) simulator
Testbench 1 (M/S environment)

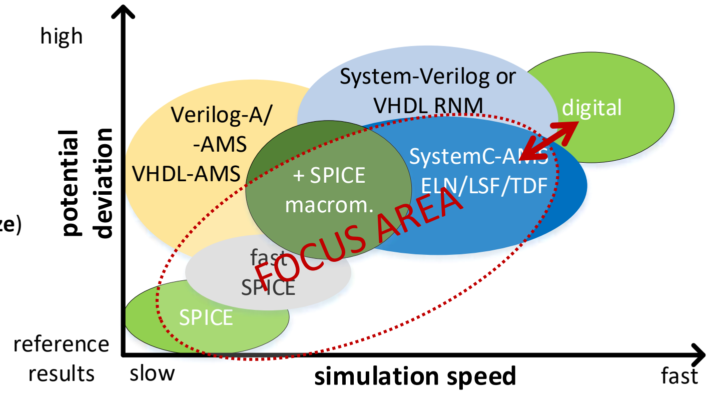
SystemC-AMS at CUAS

Potential research topics (and Master Thesis topics):

- Modeling of analog RF (started in **RFFE-Lab**)

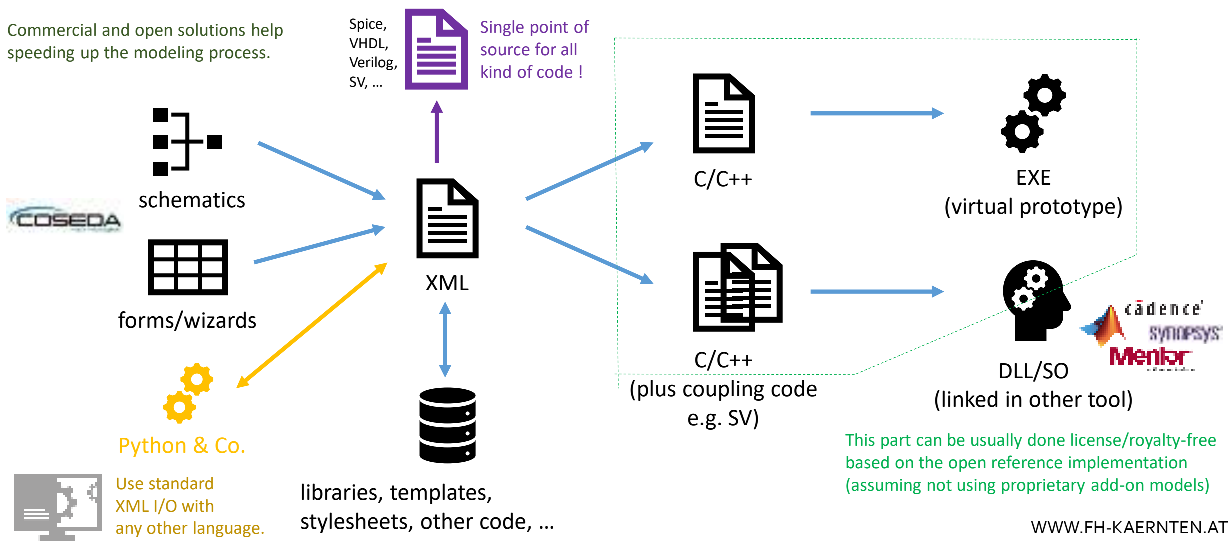


- Modeling of sensor systems (started in **Capsize**)
- Top-down methodology together with **BAG** (Berkeley Analog Generator)
- SystemC-AMS and HDLs like VHDL and SV
- SystemC-AMS and OpenModelica (https://www.openmodelica.org/)
- SystemC-AMS and FMI – functional mock-up interface (https://fmi-standard.org/)



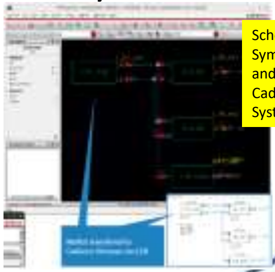
WWW.FH-KAERNTEN.AT

Another standard as key enabler: XML + code generation helps with modeling

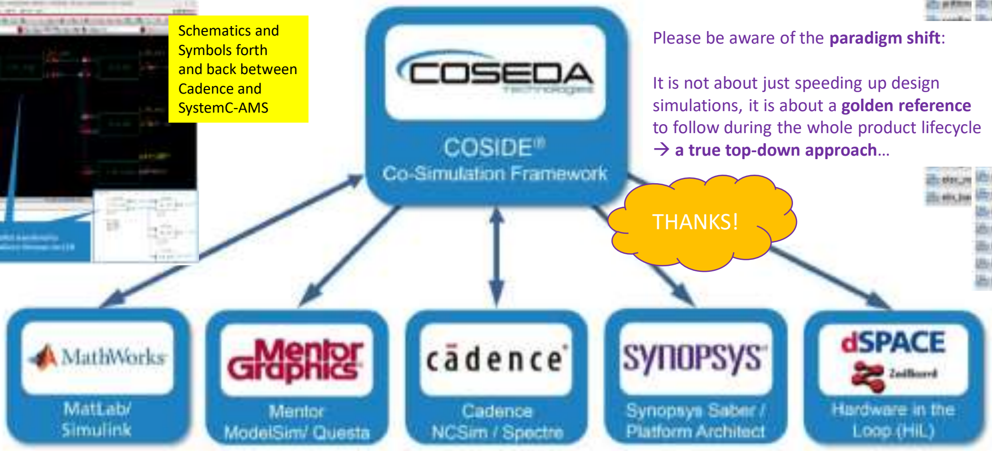

Commercial and open solutions help speeding up the modeling process.
Spice, VHDL, Verilog, SV, …
Single point of source for all kind of code !
schematics
forms/wizards
Python & Co.
Use standard XML I/O with any other language.
XML
libraries, templates, stylesheets, other code, …
C/C++
EXE (virtual prototype)
C/C++ (plus coupling code e.g. SV)
DLL/SO (linked in other tool)
This part can be usually done license/royalty-free based on the open reference implementation (assuming not using proprietary add-on models)

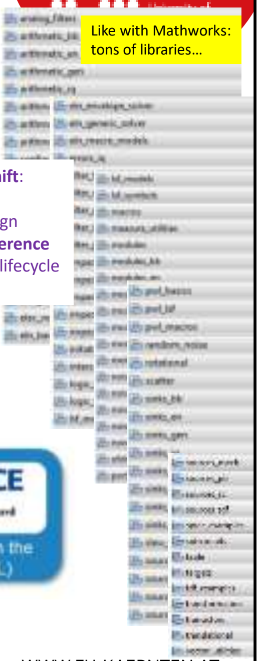
WWW.FH-KAERNTEN.AT

The new application and design-aware system-level flow available for ISCD



Schematics and Symbols forth and back between Cadence and SystemC-AMS





Like with Mathworks: tons of libraries…

Please be aware of the **paradigm shift**:

It is not about just speeding up design simulations, it is about a **golden reference** to follow during the whole product lifecycle
→ **a true top-down approach…**

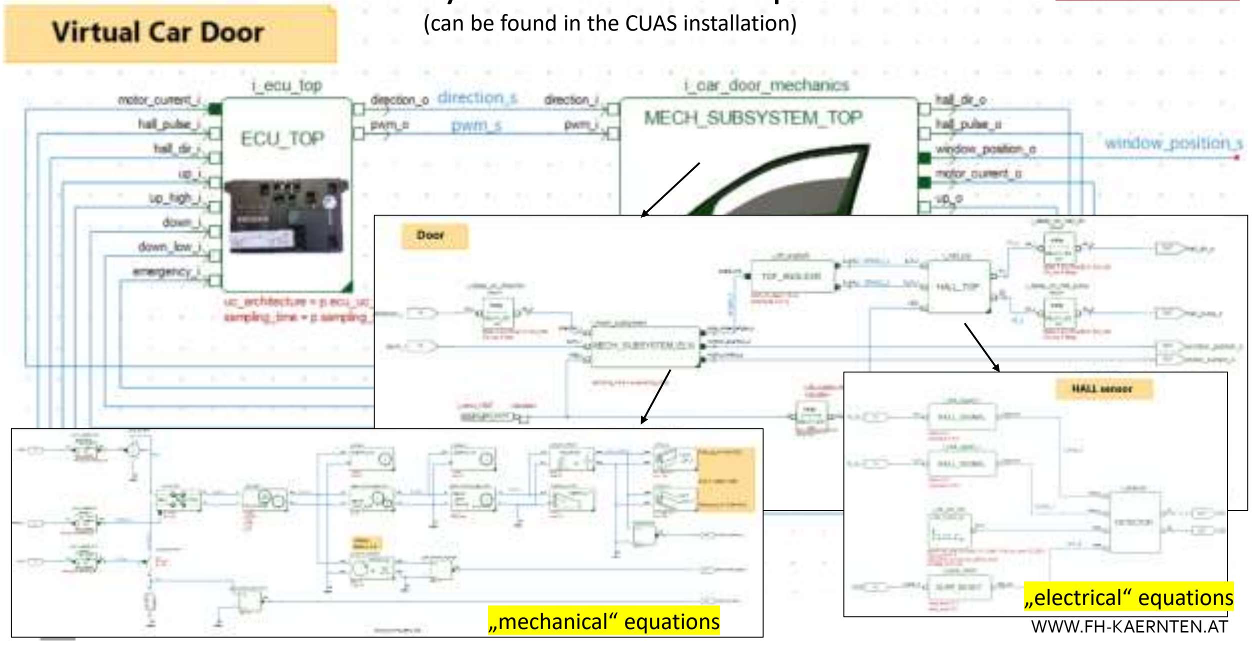
Use SystemC-AMS models in and with tools like Simulink, Matlab, Octave, Python.

From Coside presentation slides and documentation, Coseda technologies GmbH, 2019

WWW.FH-KAERNTEN.AT

System Model Example

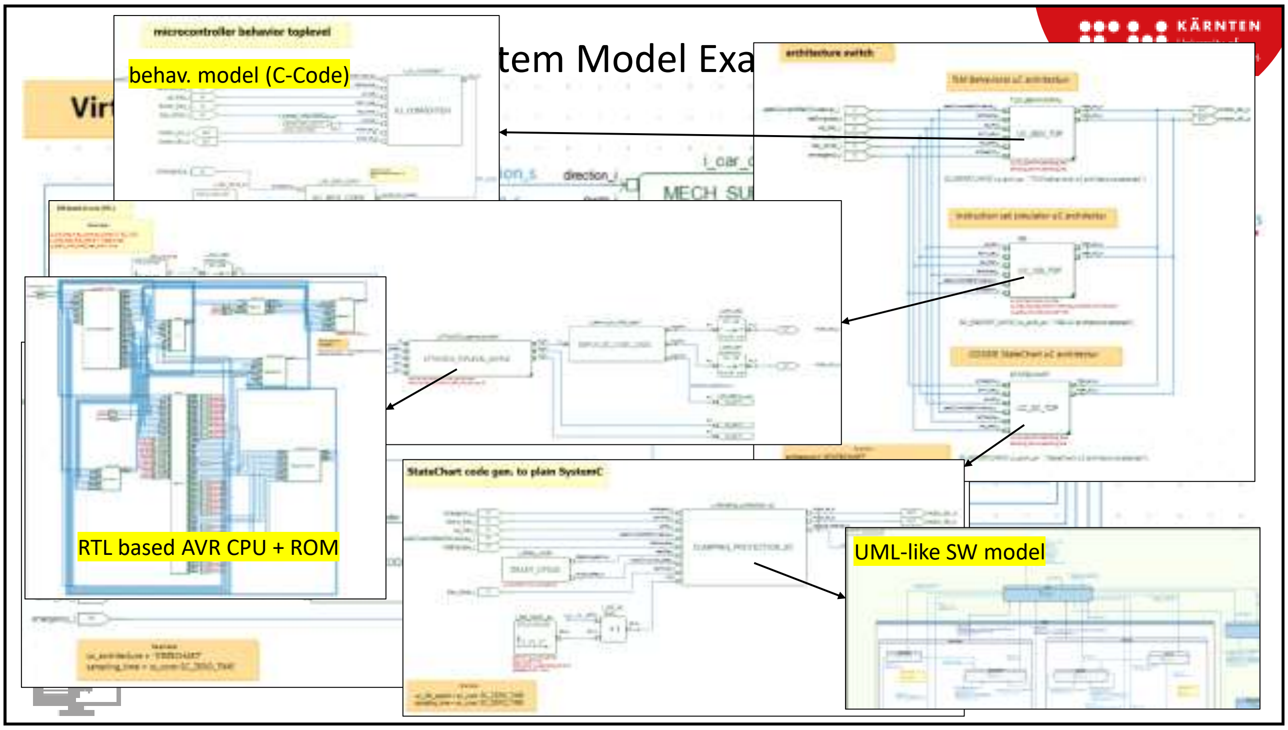
(can be found in the CUAS installation)



„mechanical“ equations

„electrical“ equations

WWW.FH-KAERNTEN.AT

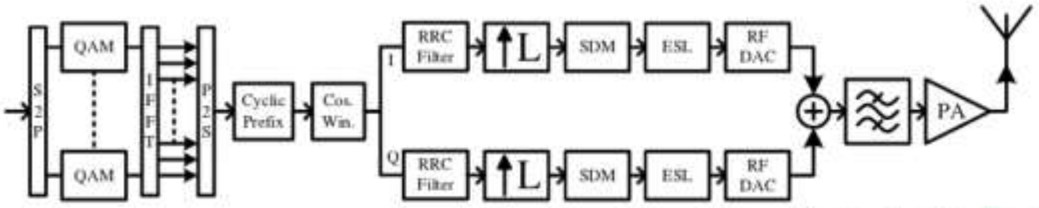


KÄRNTEN University of Applied Sciences

What other universities and companies have to say…

WWW.FH-KAERNTEN.AT

Architecture RF modeling (LTE) using SystemC-AMS



Block	Configuration
QAM	rectangular 256-QAM
OFDM	subcarriers = 1024, IFFT size = 2048, cp = 144 samples, [illegible] = 31

Fig. 1. Block diagram of the RF-DAC based transmitter

Abstract:
This paper presents the modelling and simulation of a multistandard, multimode RF-DAC based LTE transmitter. The RF-DAC combines DAC and up-conversion mixer. Therefore, more analogue blocks can be replaced by digital reconfigurable blocks. Each block of the transmitter is modelled in SystemC/AMS. A platform is developed to simulate and evaluate the different architectures. Because out-of-band emission is a critial issue regarding the regulatory specifications, several techniques are discussed to reduce these unwanted emission. Based on the simulation results, the verification and reconfigurability of this transmitter architecture are proven. Furthermore, design specifications for each block are derived.

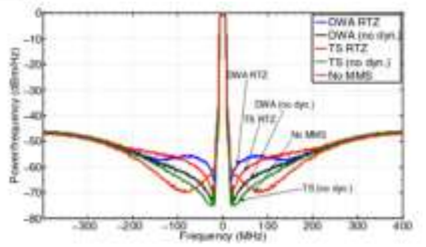


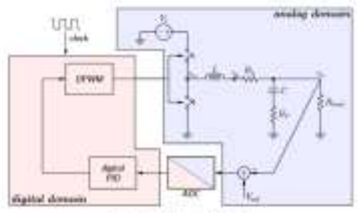
Fig. 8. Comparison of mismatch shaping approaches (σ = 0.1, $\Delta\tau$ = 0)

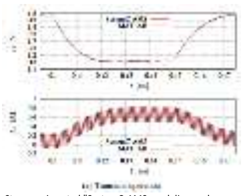
2010, 6th Conference on Ph.D. Research in Microelectronics & Electronics, RWTH Aachen, Junqing Guan et. al., „Modelling and simulation of an RF-DAC based transmitter at architectural level in SystemC/AMS"

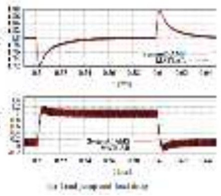
Architectural exploration: DC-DC converter (ELN/TDF)

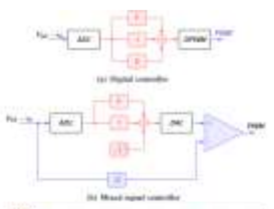
TABLE I
SIMULATION EXECUTION TIMES (IN SECONDS).

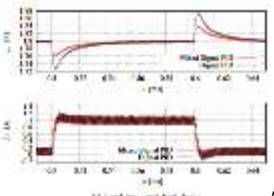
	open-loop	closed-loop
SystemC-AMS	0.17	0.75
MATLAB/Simulink	1.04	1.04
MATLAB/Simulink (Rapid Accelerator)	2.11	2.17
MATLAB/Simulink (SimPowerSystems)	1.64	1.05
MATLAB/Simulink (PLECS)	1.08	1.01





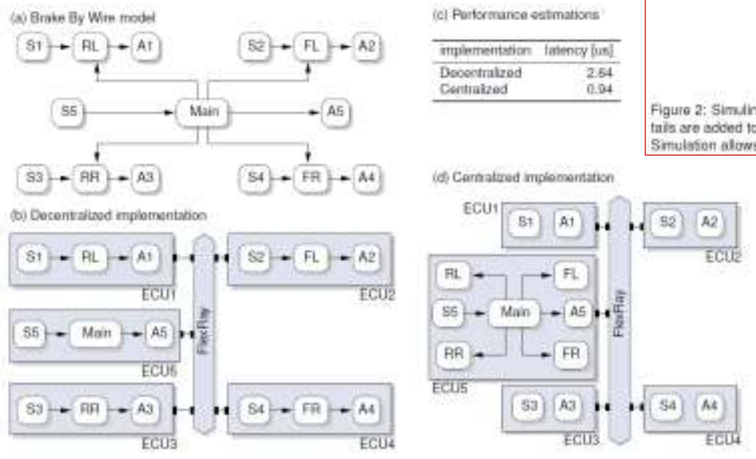






M. Agostinelli, S. Marsili, D. Straeussnigg et.al "SystemC-AMS modeling and simulation of digitally controlled DC-DC converters," in *Proc. of the 25th Annual IEEE Applied Power Electronics Conference and Exposition (APEC 2010)*

Architecture development


(a) Brake By Wire model
S1
RL
A1
S2
FL
A2
S5
Main
A5
S3
RR
A3
S4
FR
A4
(b) Decentralized implementation
ECU1
ECU2
ECU5
ECU3
ECU4
FlexRay
(c) Performance estimations
(d) Centralized implementation

implementation	latency [us]
Decentralized	2.64
Centralized	0.94

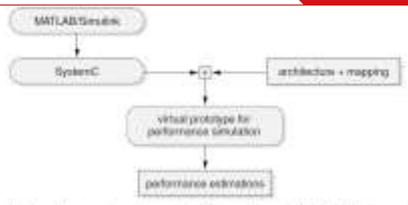
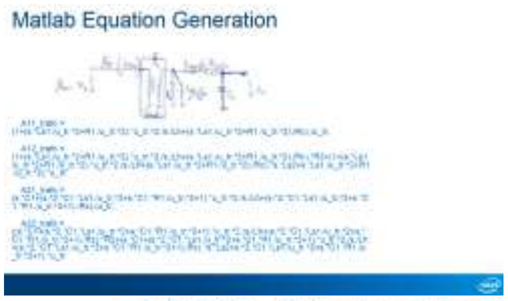


Figure 2: Simulink models are transformed to SystemC models. Architecture and mapping details are added to the SystemC model via configuration files in order to form a virtual prototype. Simulation allows evaluating the non-functional performance parameters.

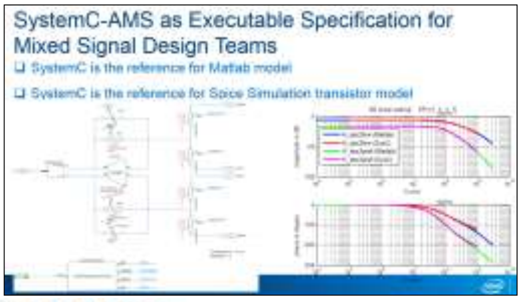
Streubühr et. al., University of Erlangen-Nuremberg
Proceedings of the Embedded World Conference,
Nuremberg, Germany, March 03-05, 2009

WWW.FH-KAERNTEN.AT

SystemC-AMS is **NOT** only time domain!



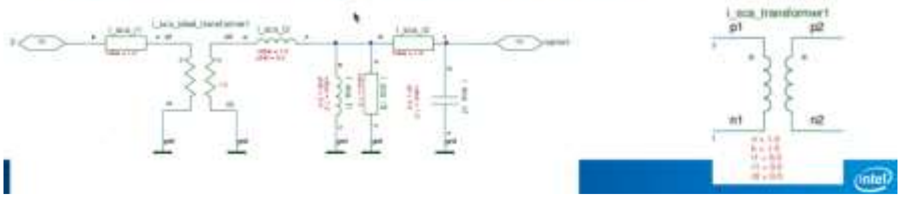
Matlab Equation Generation



SystemC-AMS as Executable Specification for Mixed Signal Design Teams

- SystemC is the reference for Matlab model
- SystemC is the reference for Spice Simulation transistor model

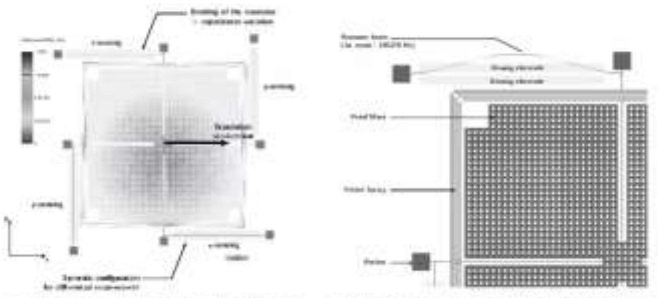
- SystemC – AMS solves all the equations for the user



AC and Noise Simulation for an Ethernet Phy
SystemC AMS & COSIDE® User Group Meeting 2017
Gerhard Nössing, Intel Corporation

WWW.FH-KAERNTEN.AT

Using model order reduction for MEMS to create a TDF model in SystemC-AMS



(a) Sensing of the tangential displacement on x-axis through resonator (b) "L-shaped" resonator first mechanical mode used for sensing application

Figure 1: Biaxial accelerometer principle and differential sensing through a resonator (MEMS+ model)

Table I: Accelerometer Parameters

Parameter	Value	Unit
Resonator length	3520.0	µm
Resonator thickness	1.51	µm
Nominal gap at rest	3.00	µm
Main spring length	320.0	µm
Folded spring length	275.0	µm
Spring thickness	1.51	µm
Proof mass	8.110 e[illegible]	kg

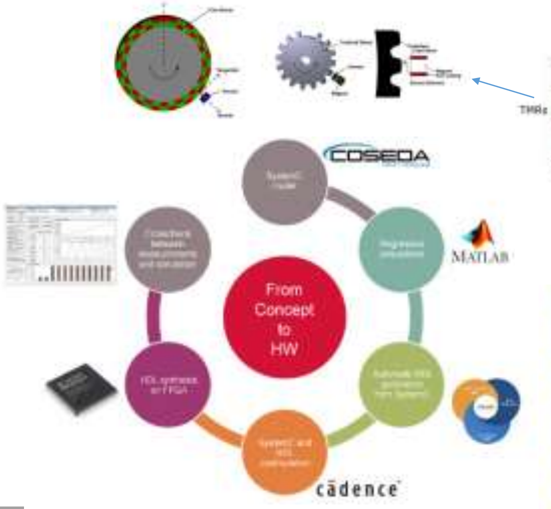
Table III: Simulation results - Ramp impulse of translation acceleration on x-axis (Amplitude: 1g)

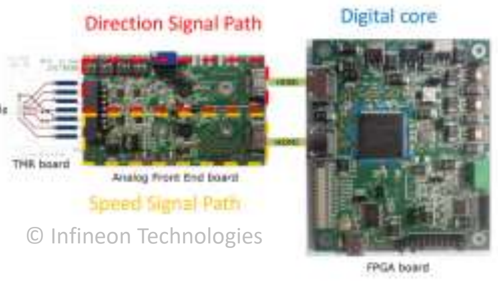
	Solver		Step size			Simulation
Simulator	Model	Method	Type	Max. (µs)	Iterations	Time (s)
MEMS+	Full	ode23t	variable	800.0	2000	2110.[illegible]
MATLAB / Simulink	ROM	ode23t	variable	800.0	43893	3.721
MATLAB / Simulink	ROM	ode23t	fixed	1.0	4000000	797.[illegible]
SystemC AMS (standard)	ROM	Backward Euler	fixed	10.0	4000	4.000
SystemC AMS (modified)	ROM	Crank Nicolson	fixed	10.0	4000	1.333

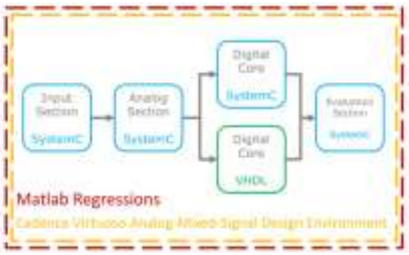
B. Vernay et. al.: 2015 Symposium on Design Test Integration and Packaging of MEMS and MOEMS, "SystemC-AMS Simulation of a Biaxial Accelerometer based on MEMS Model Order Reduction"

WWW.FH-KAERNTEN.AT

High-level Synthesis starting from SystemC







S. Fontanesi at al: 2018, DESIGN & ELEKTRONIK Messen + Testen, "Von der Idee zum Prototyp - schnell, flexibel und effizient" (Infineon Technologies AG) https://www.elektroniknet.de/design-elektronik/messen-testen/von-der-idee-zum-prototyp-schnell-flexibel-und-effizient-160955.html

WWW.FH-KAERNTEN.AT

Conclusion

What you should take away today…

- Modeling and abstraction gains importance to **cope with complex systems**
- Virtual **prototyping reduces risk**, helps sorting out early project problems towards realization and helps improving **time to market in technology push and market pull strategies**
- SystemC/SystemC-AMS:
  - An **open-source C++ extension** for electronic system level (ESL) design and verification, standardized by IEEE 1666(.1)
  - A **broad range of models of computations** (MoCs) were shown to generate an single-executable specification / virtual prototype, several examples were presented and many more are available for download
  - Improves **refinement strategies** from system level down to design, allows agile development processes (similar to the software world)
  - Of course, SystemC-AMS is "**just another option**" modeling within many existing solutions, **but shows many new ways** to incorporate more disciplines in a heterogenous modeling landscape and can work together with them.
- **Keep your goal in mind**, sometimes simulation speed is not all, but simplicity to create models (without a lot of debugging) and to **go on faster with your design task** and the problem you want to solve.

KÄRNTEN University of Applied Sciences

THANK YOU FOR YOUR ATTENTION