Top-Down Radar System Design with COSIDE: an overview

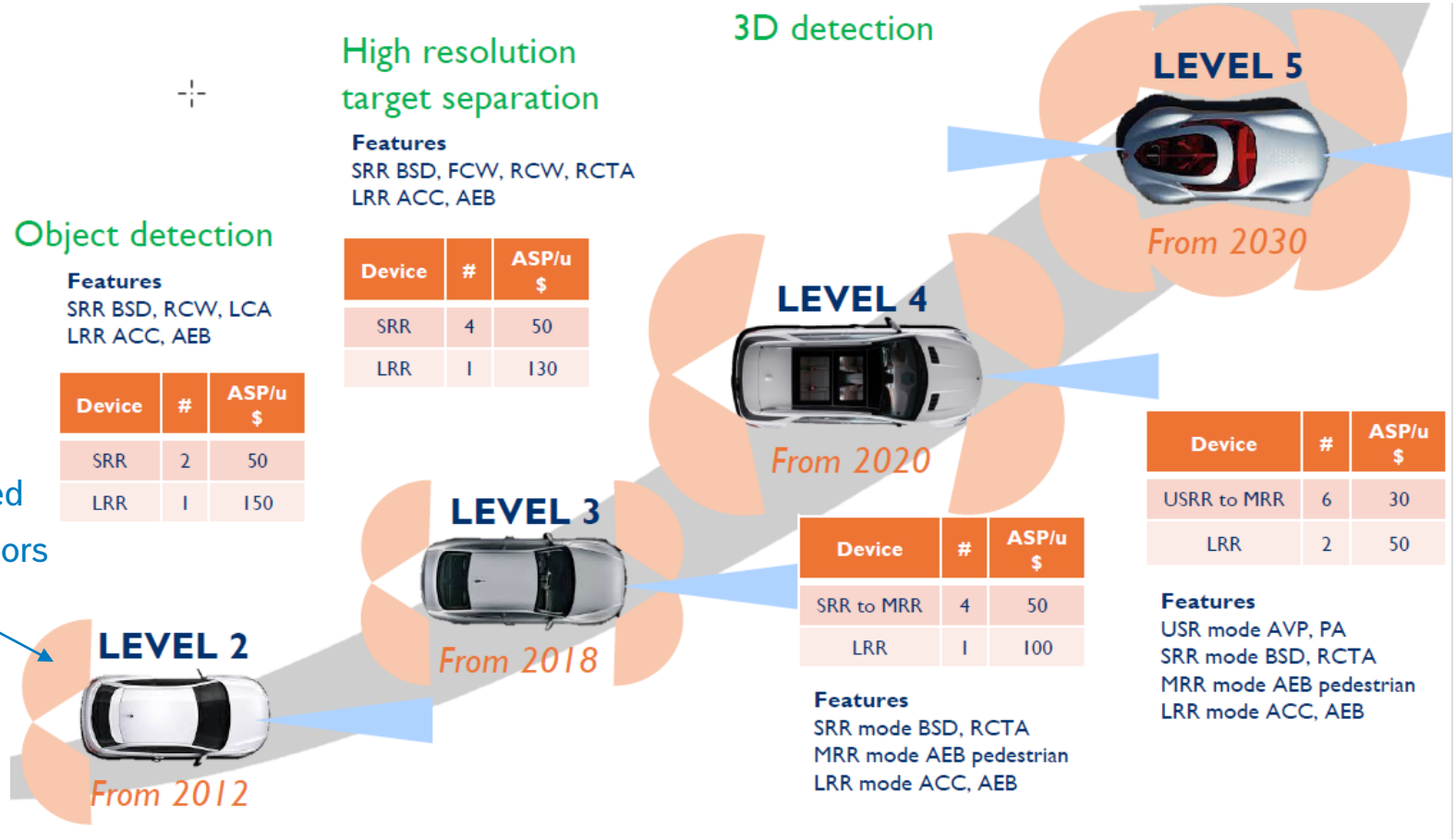
Federico Passerini
Robert Bosch Automotive Electronics
25/11/2022

BOSCH

Summary

- Introduction to the Radar System
- Top-Dow Chip Design
- Simulation Approach Using Coside
- Conclusions

Radar System
Application to Automotive

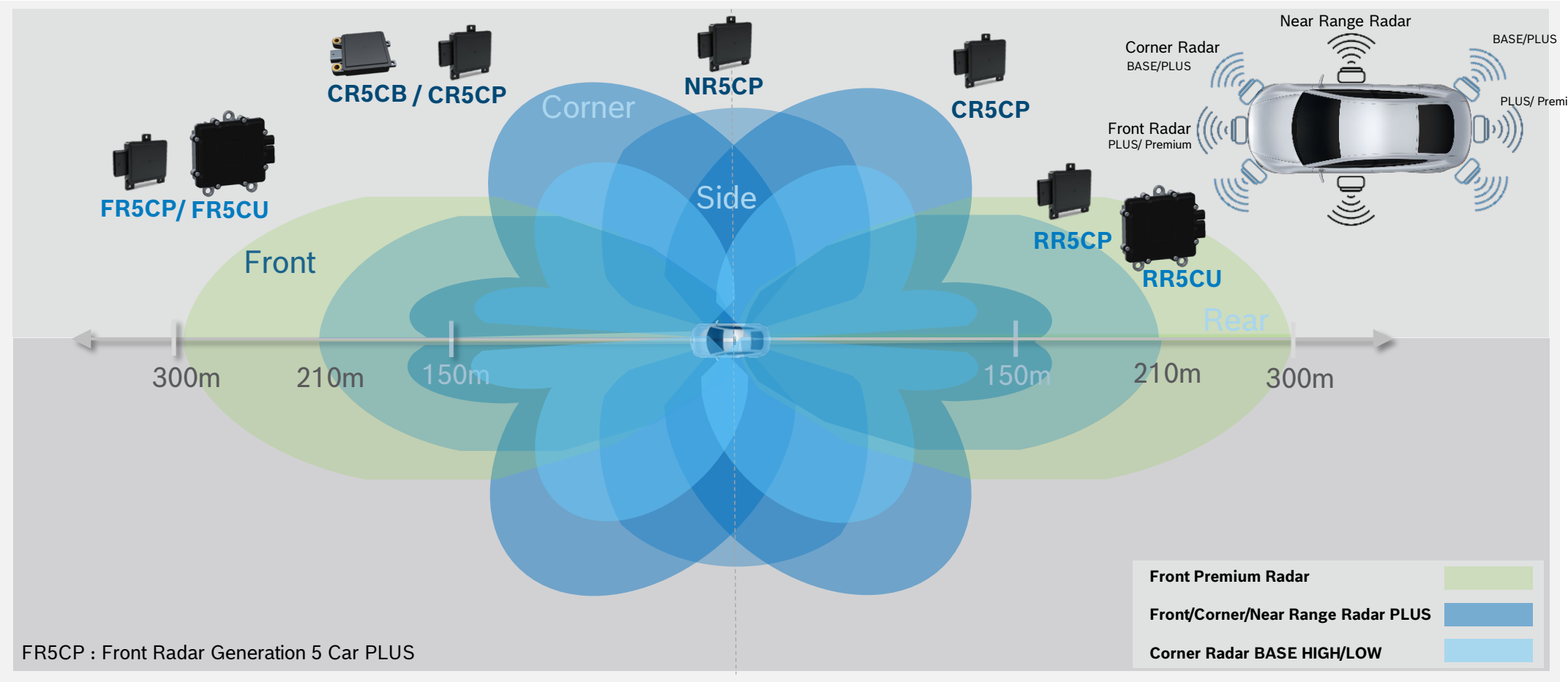


Acronyms:
SRR=short range Radar
MRR=mid range Radar
LRR=long range Radar
USRR=ultra SRR

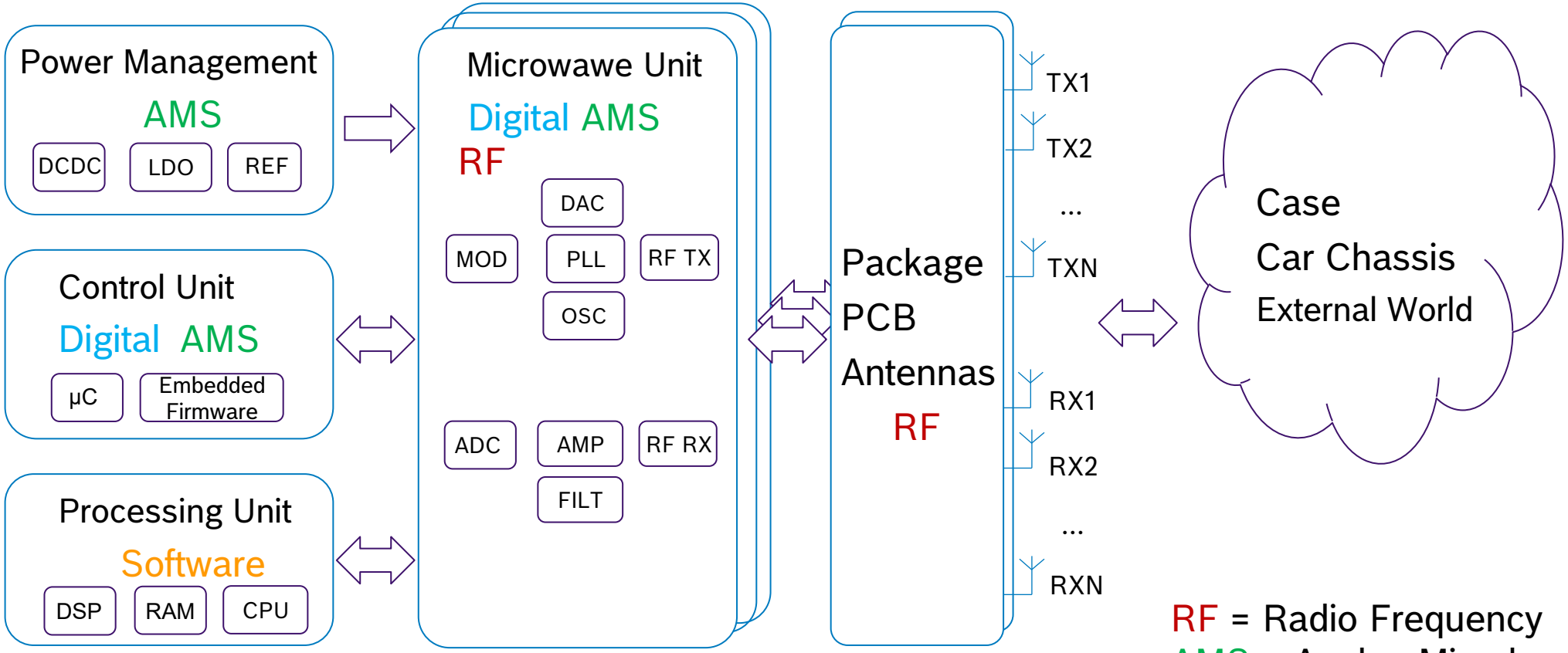
Applications: e.g.
Adaptive cruise control, automatic emergency brake, automated valet parking etc.

Source: Yole Radar Report 2018

Radar System
Types of Automotive Radars



Radar System
Different Types of Sub-System inside the Case

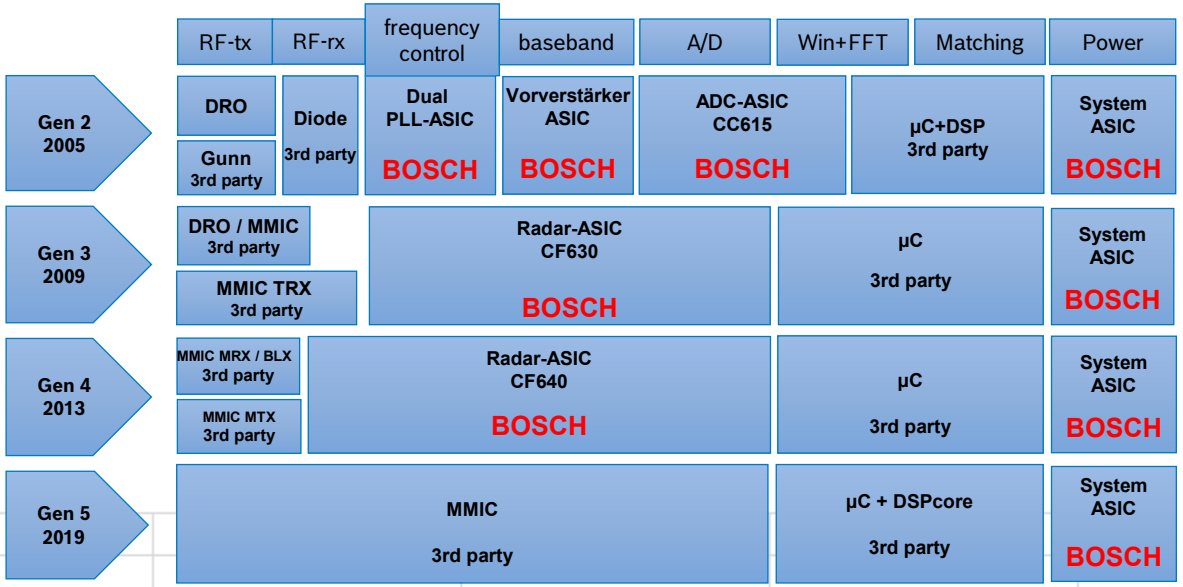


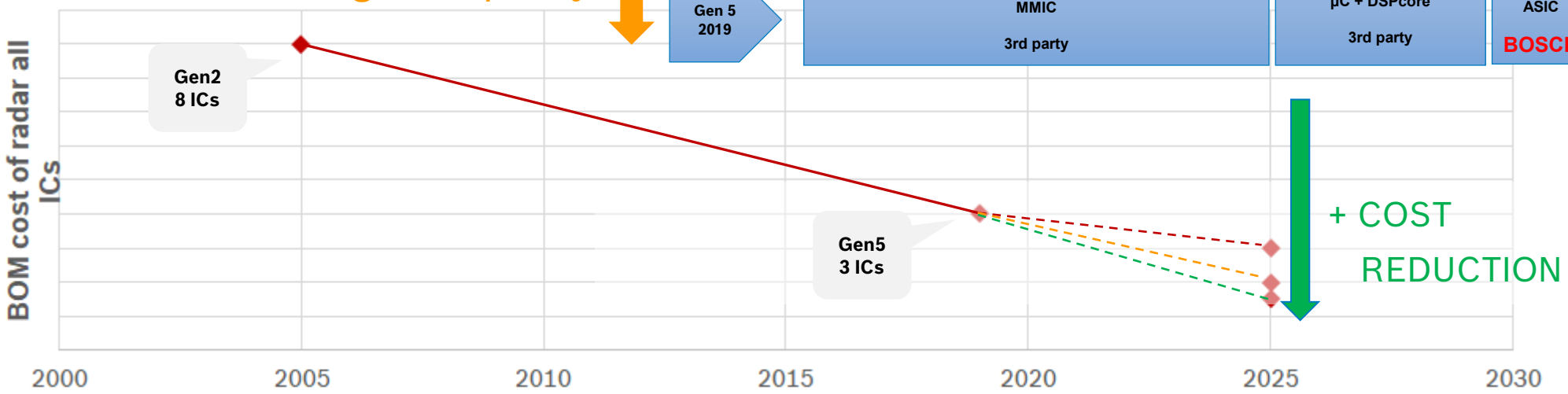
RF = Radio Frequency
AMS = Analog Mixed Signal

Radar System
Evolution

Increasing Sub-System Integration on a Single Chip

Increasing Design Complexity





Top-Down Chip Design

Component Design (Gen 2)

either Digital or AMS
or RF or Software

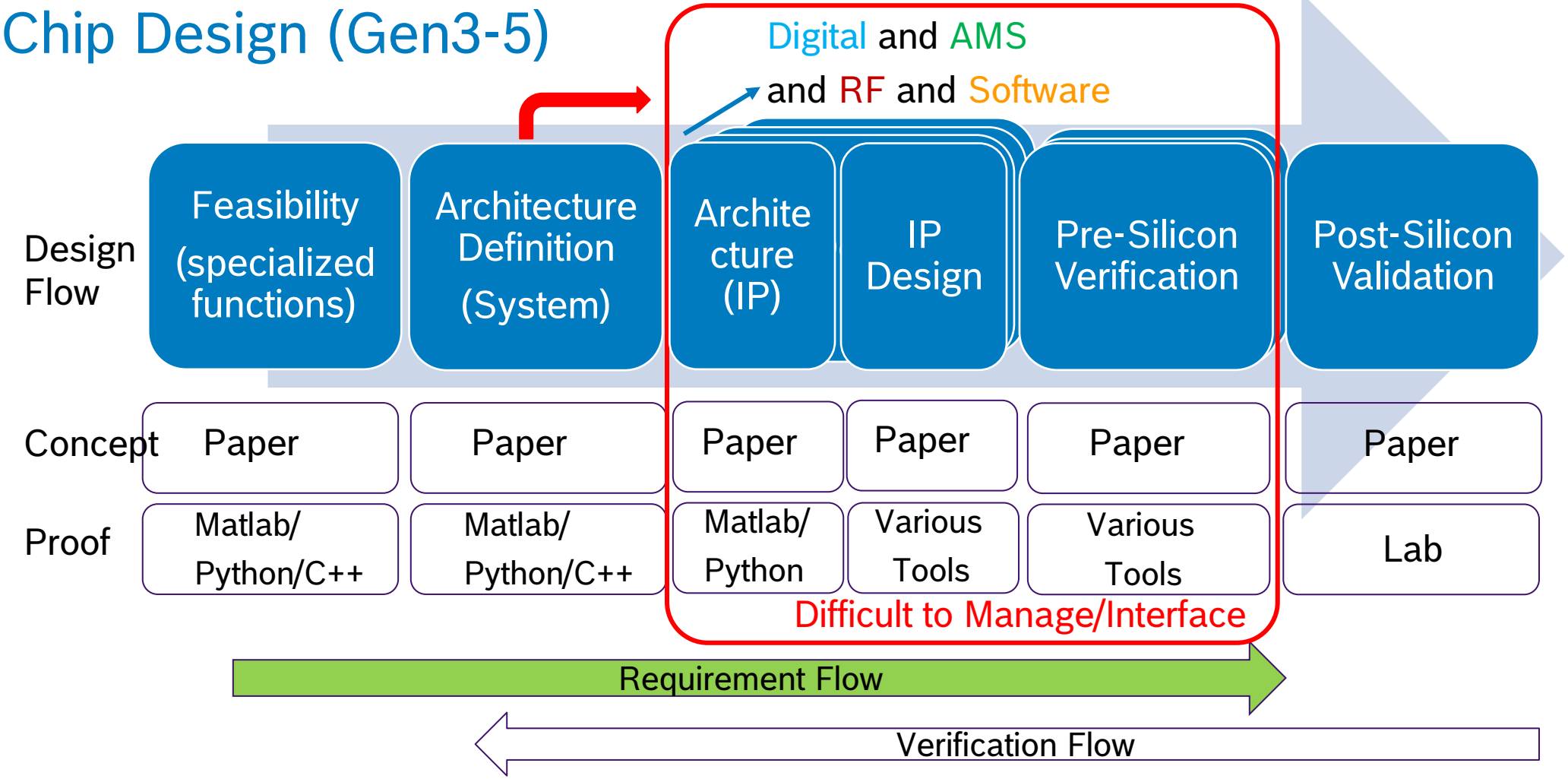
Design Flow	Feasibility (electronic functions)	Architecture Definition (IP)	IP Design	Pre-Silicon Verification	Post-Silicon Validation
Concept	Paper	Paper	Paper	Paper	Paper
Proof	Matlab/ Python/C++	Matlab/ Python/C++	Few Specific Tools	Few Specific Tools	Lab

Requirement Flow

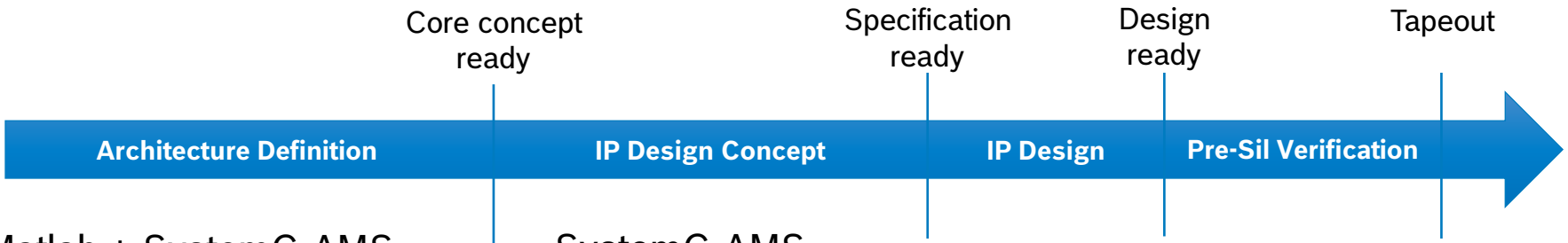
Verification Flow

Top-Down Chip Design

Chip Design (Gen3-5)



Top-Down Chip Design
Holding Everything Together



Matlab + SystemC-AMS

- High-level design of functions and algorithms
- Base development of functional behavior
- Architecture decisions
- Development of 1st part of test-case database

SystemC-AMS

- Proof of concept
- Development of a reference model
- Development of 2nd part of test-cases
- Development of block specifications
- Verification
  - Reuse of test-case database (1st and 2nd part) from concept phase
  - Reuse of environment for design- and verification-phase

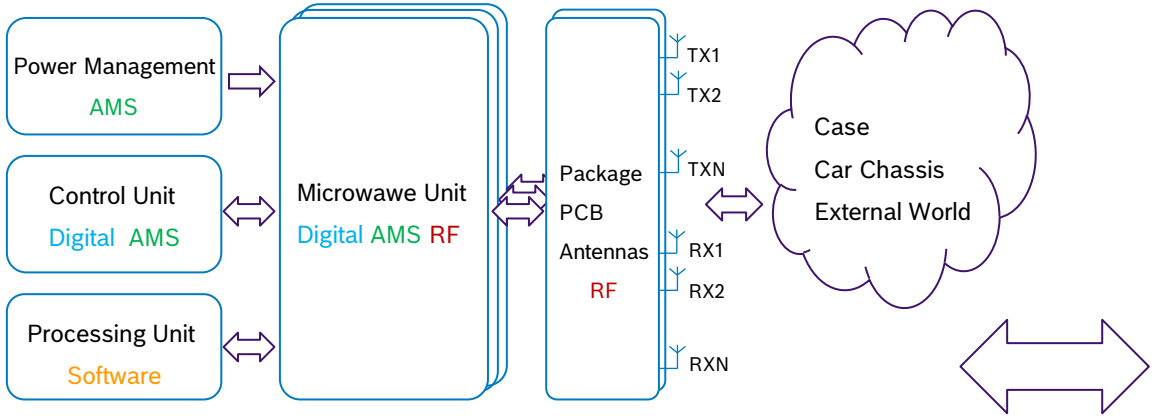
+ quick simulations

+ different levels of abstraction in the same environment

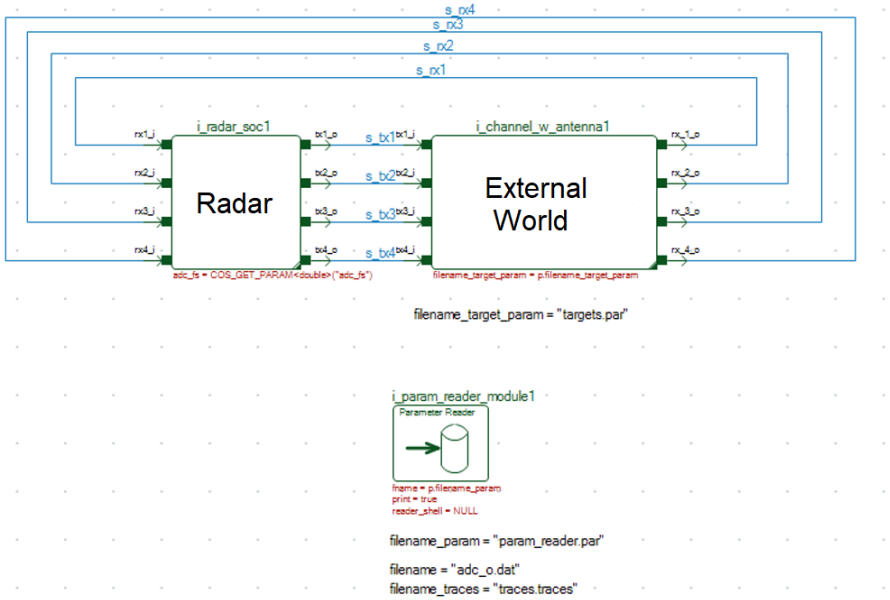
+ high integration of different types of sub-systems

Simulation Approach Using Coside
Holding Everithing Together with Coside

Radar Concept

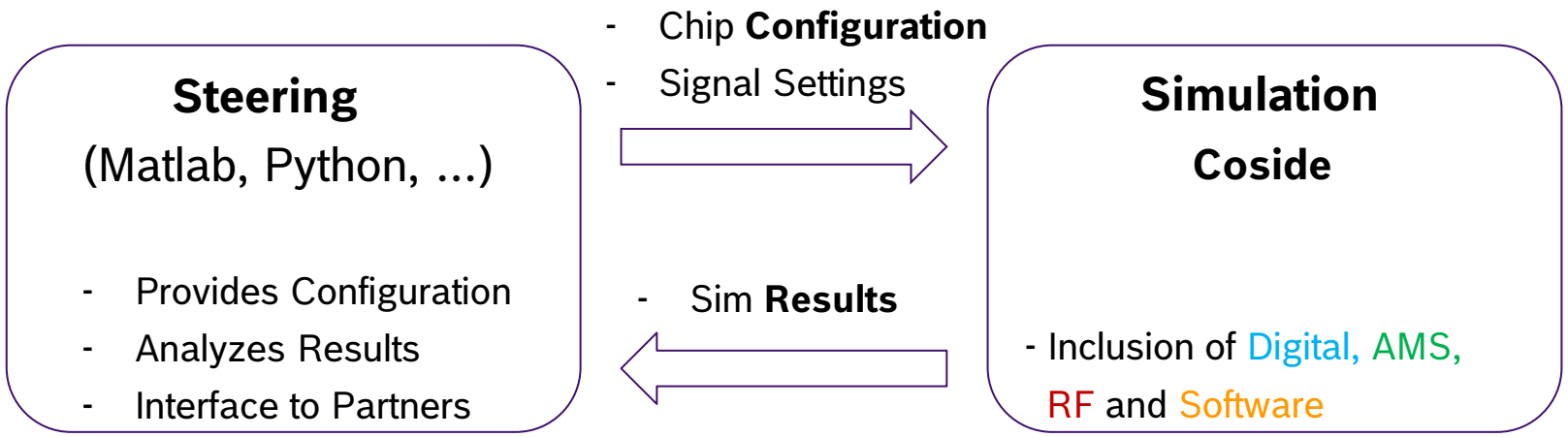


Coside Implementation



- Full system handling using Coside
- Use of parameter reader function of Coside
- External software interface (steering/processing)

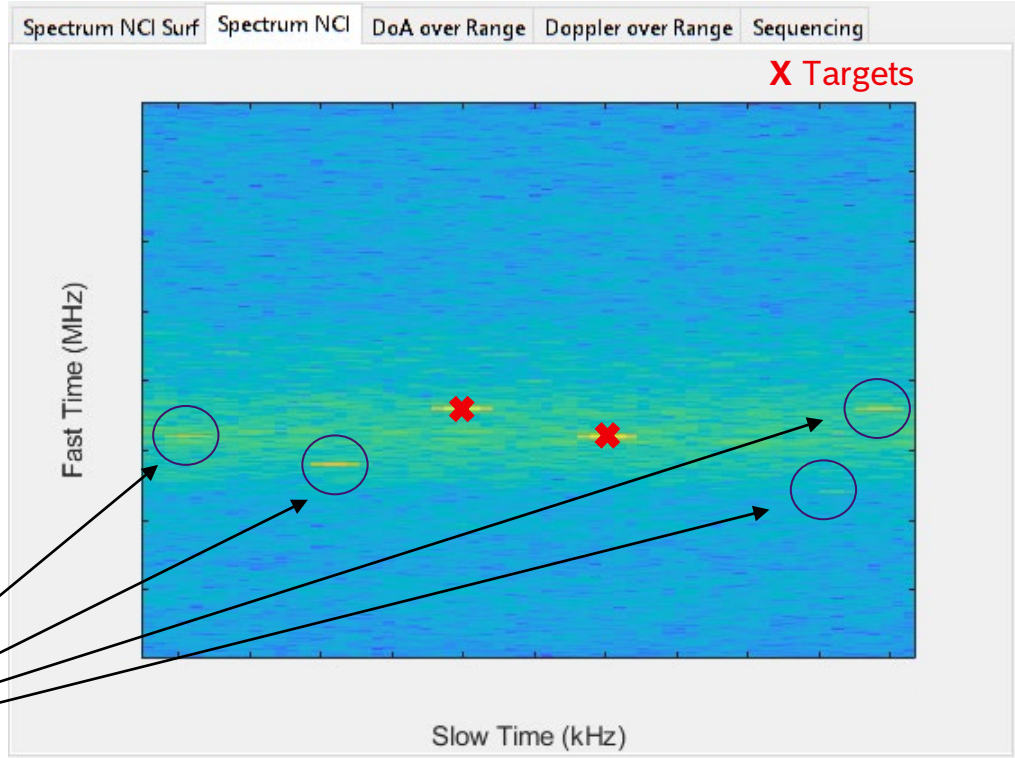
Simulation Approach Using Coside
Software Interface



- Combination of simulation power of Coside and versatility of general purpose software

Simulation Approach Using Coside
Full System

- Use Case Setup
- Full System Simulation
- From Feasibility to Implementation
  - Deep Control over all IP params
  - Very quick simulation (~sec)



- Unwanted spurs
- We can trace them back and reproduce them
- Identify root cause

Simulation Approach Using Coside
Implementation of IP Models

1. Full SystemC
2. Translation from external code to C++
3. Using Coside blocks
4. Importing external binary files

Simulation Approach Using Coside

1) Full SysC vs 2) External Code – ADC example

Ideal ADC in SysC

```
void adc::sampling()
{
    std::normal_distribution<double> distribution(0.0, 1.0);

    double out = 0.0;
    out = sig_i.read();

    // input clipping
    out = std::max(p.ref_n, std::min(p.ref_p, out));

    // add ENOB based noise
    double Vqn = (1.0/pow(2.0,p.enob))/sqrt(12);
    out = out + Vqn * distribution(s.generator);

    // scale to PNOB, signed and centered
    out = (out - p.ref_n) / (p.ref_p - p.ref_n) * pow(2.0, p.pnob) - pow(2.0, p.pnob-1);
    //out = out * pow(2.0, p.pnob);

    // quantization
    out = round(out);

    // output clipping
    //out = std::max(pow(2.0, p.pnob-1), std::min(-pow(2.0, p.pnob-1), out));

    data_o.write(out);
}
```

- Ideal ADC takes few lines -> quick feas
- Bosch ADC model developed by ADC experts

Bosch ADC imported from Simulink

Simulink Model → (Matlab Conversion Tool) → C++

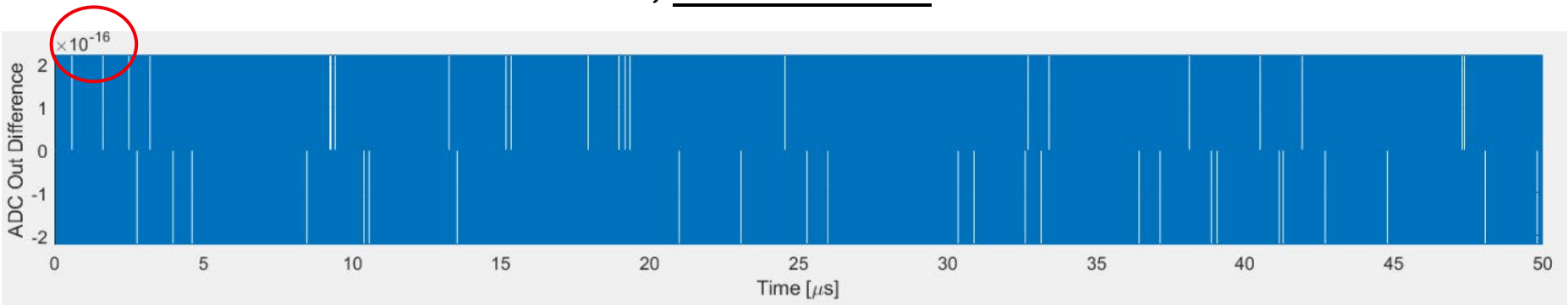
```
//-----------------------------------------------------------------//
// states                                                          //
//-----------------------------------------------------------------//
struct bosch_adc::states
{
    RT_MODEL_bosch_adc_s_T* i_bosch_adc_sc;

    states(const params& p)
    {

////////////////////////////////////////////////
// SC_METHOD calc_meth
// sensitive << clk.pos();
////////////////////////////////////////////////
void bosch_adc_ba::calc_meth()
{
    // read SystemC wrapper pin and assign to C-model I/O data structure pin
    s.i_bosch_adc_ba_sc->inputs->sig_i = sig_i.read();
    s.i_bosch_adc_ba_sc->inputs->reset_i = reset_i.read();
    // do a C-model step
    bosch_adc_ba_sc_step(s.i_bosch_adc_ba_sc);
    // assign C-model I/O data structure pin and write to SystemC wrapper pin
    data_o.write(s.i_bosch_adc_ba_sc->outputs->data_o);
}
```

Simulation Approach Using Coside
Performance comparison

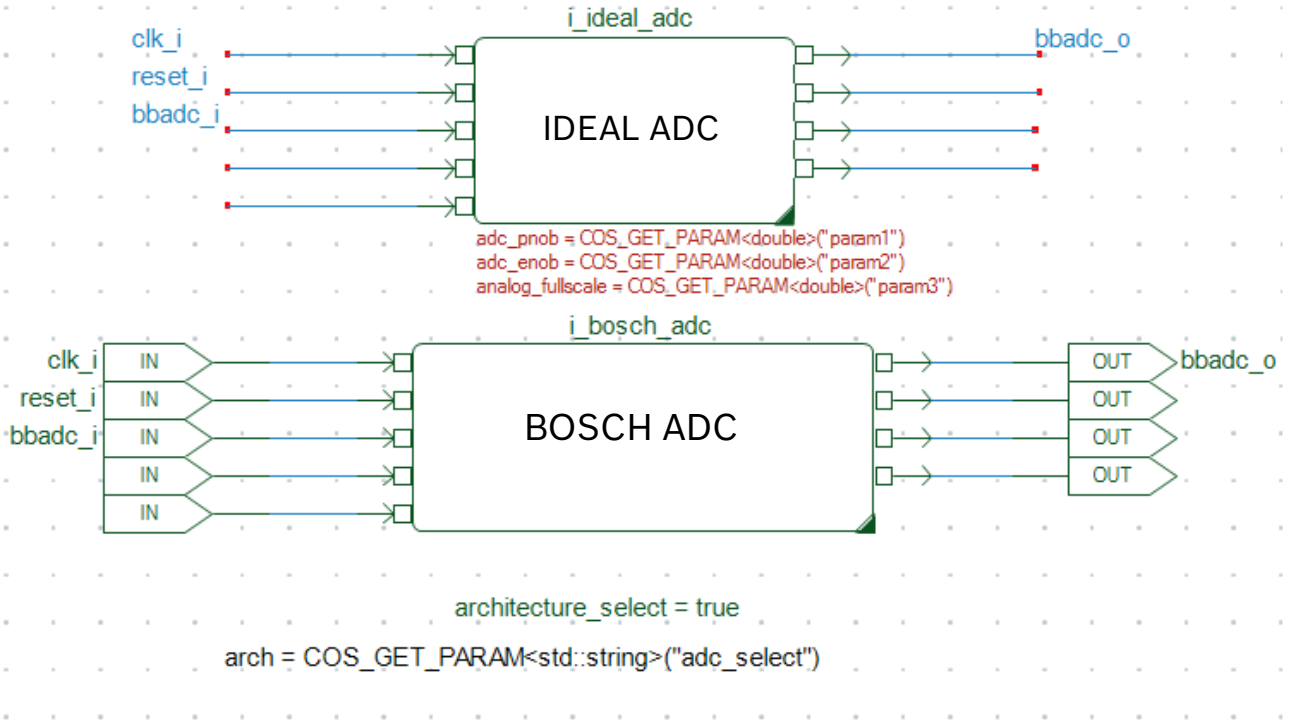
- Test signal with normalized amplitude 1
- Bosch ADC considered
- Simulation run on both Simulink and Coside, separately

- Results
  - Negligible performance difference (~1e-16)
  - Simulation time: Simulink ~10 sec, Coside ~1 sec

⇨ 10-fold sim time reduction!

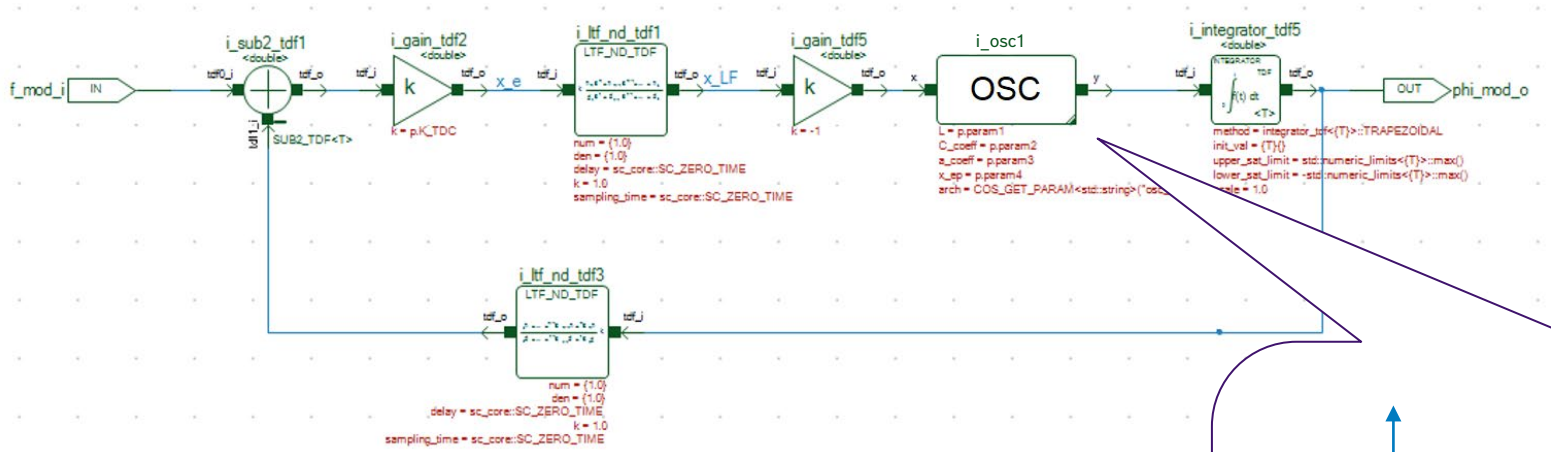


Simulation Approach Using Coside
Select Model



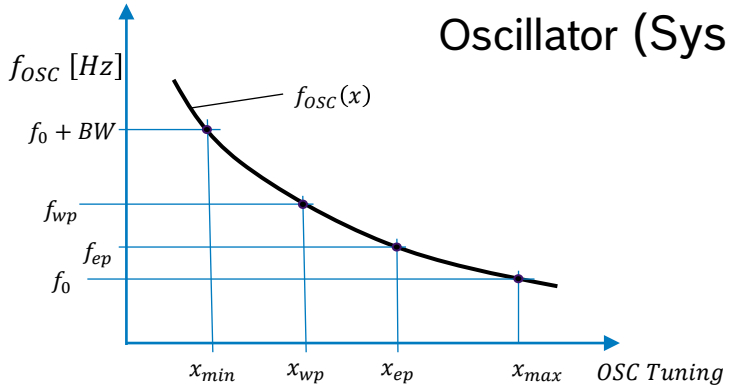
- Usage of Coside parameter reader and COS_GET_PARAM functions
- Usage of architecture selection functionality

Simulation Approach Using Coside
3) Full Coside Model – PLL example



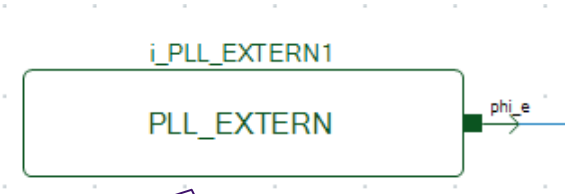
- Mostly done using standard Coside blocks
  LinearTransferFunction, Integrator, Gain
- Sub-blocks singularly expanded to more complex models
- Full implementation down to IP verification enabled

Non-Linearities in Oscillator (SysC)



Simulation Approach Using Coside

4) Hook for External Binary File – PLL example



- Few lines
- Include external simulator data
- **Include Test data**
- Direct comparison w/ other models

```
//////////////////////////////////////////////
// method initialize                        //
//////////////////////////////////////////////
void PLL_EXTERN::initialize()
{
    /*// test
    int i;
    for(i=0;i<MAX_SIM_STEPS;i++)
        myphase[i] = 1.0*i;
        */

    char fname[] = "rawCosideDataFile.bin";
    FILE *fptr;
    size_t readCount;

    fptr = fopen(fname,"rb");
    readCount = fread(&myphase, sizeof(myphase[0]), MAX_SIM_STEPS, fptr);

    fclose(fptr);
```

```
//////////////////////////////////////////////
// method processing                        //
//////////////////////////////////////////////
void PLL_EXTERN::processing()
{
    //phi_mod_o = f_mod_i*1e-6;
    //phi_mod_o = myphase[12];
    if (sim_ctr<MAX_SIM_STEPS)
        phi_mod_o = myphase[sim_ctr++];
    else
        phi_mod_o = 0.0;
    phi_e = 0.0;
    x_rtwo = 0.0;
}
```

Conclusion

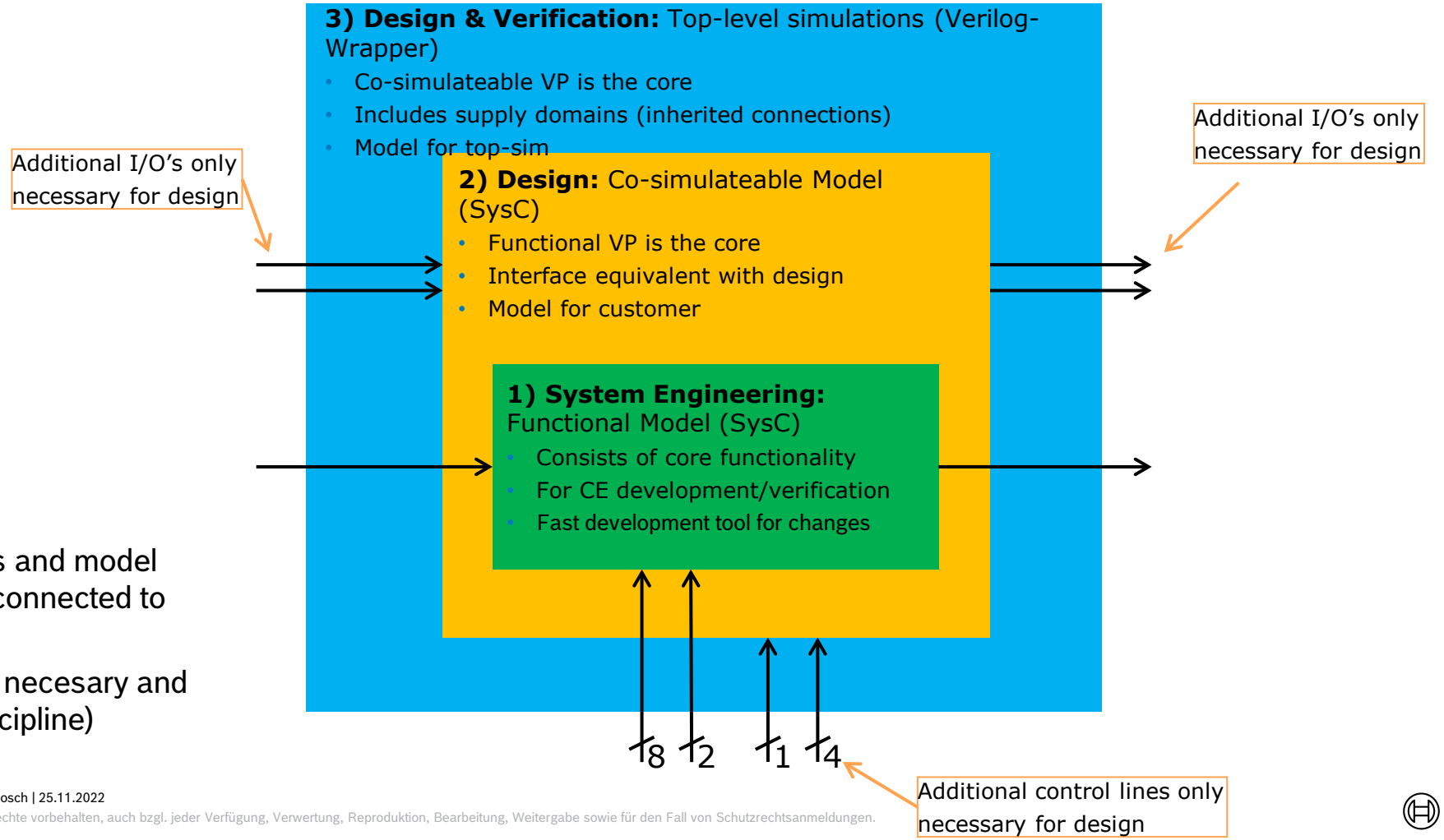
- Bosch Automotive Electronics relies on **Coside for Radar System Design**
- Capable to Support the Design Process **from Feasibility to Verification**
- Capable to **Hold Together** Digital, AMS, RF and Software Modules
- **Investigation** of **Sub-Module Influence** onto the Radar Functionality

THANKS FOR YOUR ATTENTION

BOSCH

System Simulator
Backup - Model Layers

- Necessary for CADENCE-Co simulation
- **Analog** Schematics are hierarchically and arbitrary **replaced by models**
- **Digital** is **VHDL** and/or Gate-Level



Layers show:

- How the models and model granularity are connected to each other
- Where they are necesary and used (which discipline)