Hierachical Generation of Pin accurate SystemC Models based on RF Circuit Schematics

Yifan Wang, Zhimiao Chen, and Prof. Dr. Stefan Heinen

Chair of Integrated Analog Circuits and RF Systems

RWTH Aachen University, Germany

http://www.ias.rwth-aachen.de

Outline

- Motivation & Targets
- Proposed Model Generation Approach
  - Brief introduction of SystemC models
  - Comparison of different approaches
  - SKILL based model generation routine
- Application Example
- Conclusions

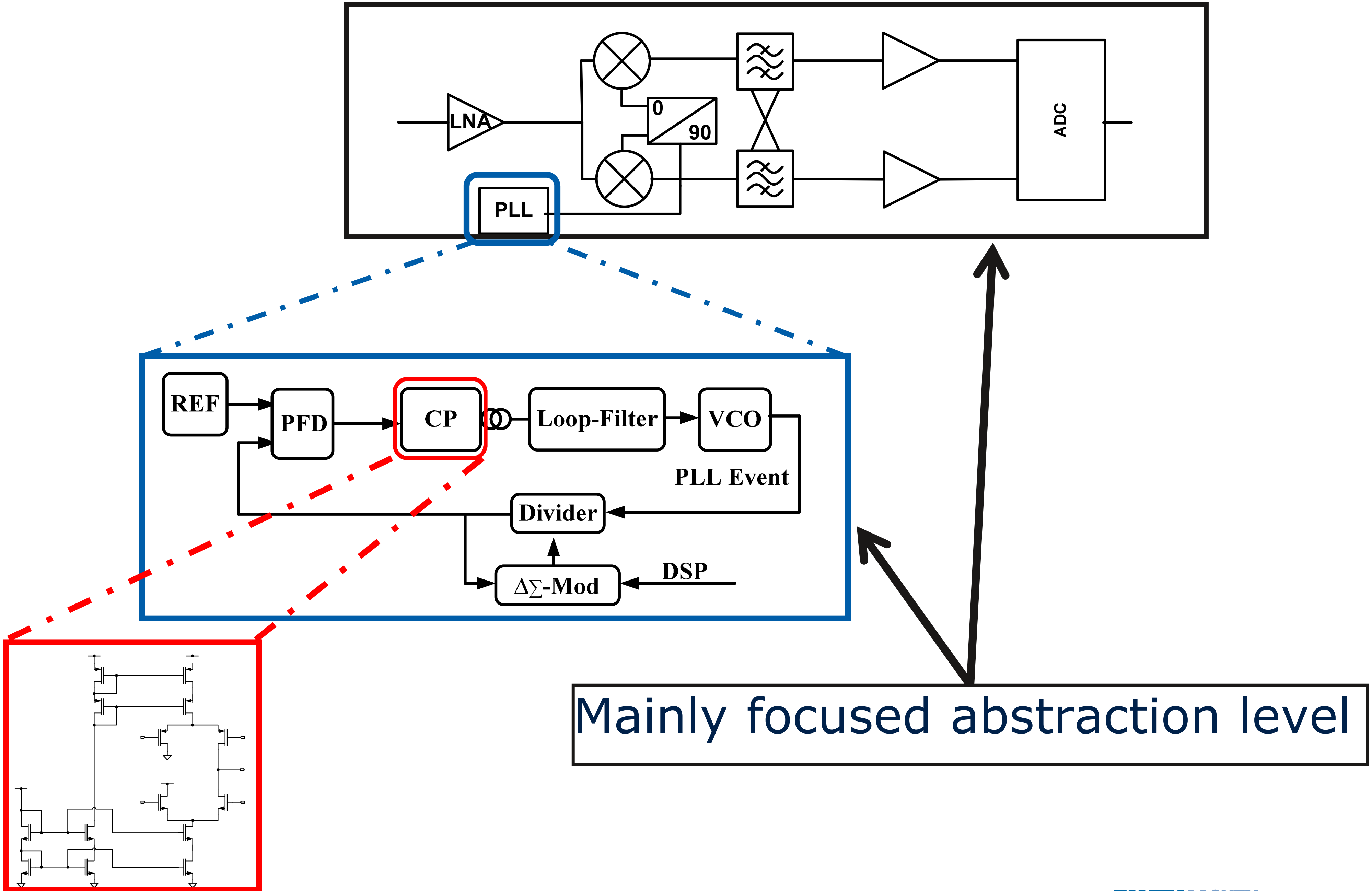
Motivation & Target

- Link different levels of design hierarchy
- System level models from circuit data base
  - Bottom up refinement of pin accurate models
  - Keep consistency of models at all hierarchy levels
- Enhance simulation efficiency
  - Real number traffic in digital domain
- Keep models flexible and portable
- Virtual prototyping

Proposed Model Generation Approach - Requirements

- Same data base for circuits and models
- No modification of the schematic allowed
  - Additional properties cause additional problems
- Pin accurate models for digital simulation
  - Real number traffic necessary to map analog spec.
  - Switchable passband / baseband signal type for RF
- Prototype
  - Reflects hierarchical structure of circuit level
  - With refined RF/analog models for digital design

Proposed Model Generation Approach - Circuit hierarchy



Proposed Model Generation Approach - SystemC language structure

```
SC_MODULE(adder) {  // module declaration

  sc_in<double> a;        // port & signal
  sc_in<T> b;             // type definition
  sc_out<T> out;

  void proc(){            // module behavioral
    out.write(a.read() + b.read());
  }

  SC_CTOR(adder){         // module constructor
    SC_METHOD(proc);
    sensitive << a << b;  // sensitiv. list of ‚proc'
  }
};
```

- Module, I/O definition
- Signal type <T> can be
  - 'double', 'logic', …
  - or user defined
- Behavioral description
- Module construction

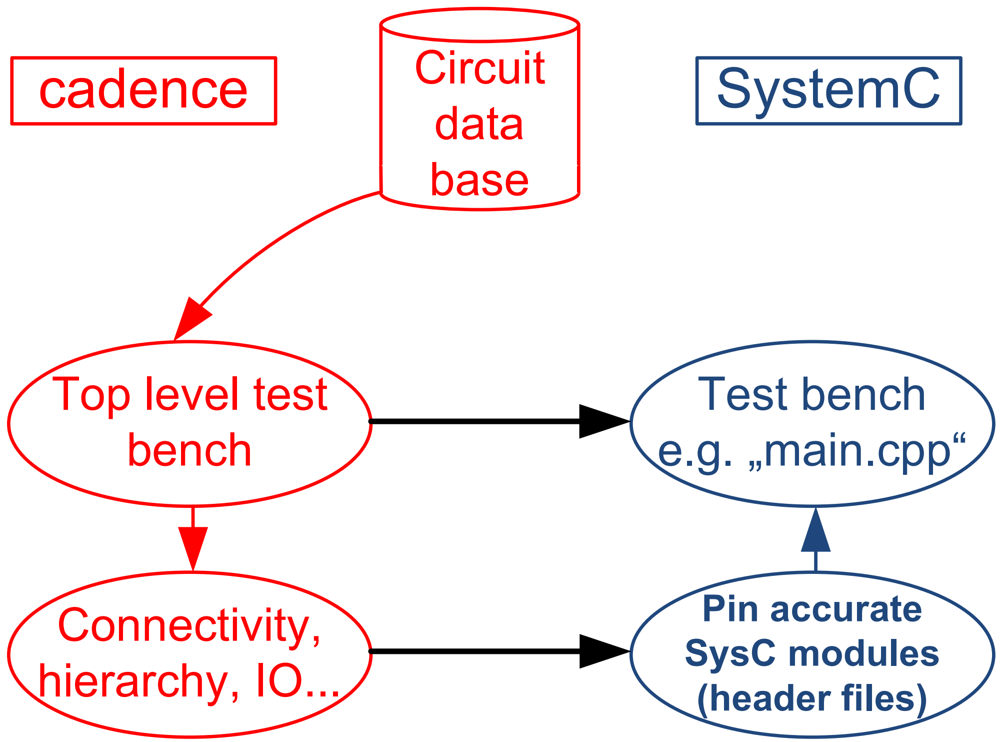
"*proc()*" and "*sensitive*" have to be hand crafted

Proposed Model Generation Approach - User defined signal structure

- User defined data type BB_double
- Vector of real numbers represent:
  - Center frequency of passband signal
  - Baseband spectral components
  - Overload the operators for BB-signal processing

```
BB_double operator * (BB_double & rhs) const{   //operator * overloading;
    BB_double result;                            //In case BB_double * BB_double;
    if(rhs.f0 == f0){                            //deduced from Matlab;
        result.f0 = f0;
        result.DC = DC * rhs.DC + I1*rhs.I1/2 + I2*rhs.I2/2 + I3/2*rhs.I3/2
                  + Q1*rhs.Q1/2 + Q2*rhs.Q2/2 + Q3*rhs.Q3/2;
        result.I1 = DC*rhs.I1 + rhs.DC*I1
                  + (Q1*rhs.Q2)/2 + (rhs.Q1*Q2)/2 + (Q2*rhs.Q3)/2 + (rhs.Q2*Q3)/2
                  + (I1*rhs.I2)/2 + (rhs.I1*I2)/2 + (I2*rhs.I3)/2 + (rhs.I2*I3)/2;
        result.Q1 = DC*rhs.Q1 + rhs.DC*Q1
                  + (rhs.I1*Q2)/2 + (rhs.I1*Q2)/2 - (rhs.I2*Q1)/2 - (rhs.I2*Q1)/2
                  + (I2*rhs.Q3)/2 + (rhs.I2*Q3)/2 - (I3*rhs.Q2)/2 - (rhs.I3*Q2)/2;
        result.I2 = DC*rhs.I2 + rhs.DC*I2
                  + (I1*rhs.I1)/2 + (I1*rhs.I3)/2 + (rhs.I1*I3)/2
                  + (Q1*rhs.Q3)/2 - (Q1*rhs.Q1)/2 + (rhs.Q1*Q3)/2;
        result.Q2 = DC*rhs.Q2 + rhs.DC*Q2
                  + (I1*rhs.Q1)/2 + (rhs.I1*Q1)/2 + (I1*rhs.Q3)/2
                  + (rhs.I1*Q3)/2 - (I3*rhs.Q1)/2 - (rhs.I3*Q1)/2;
```

Proposed Model Generation Approach - Overview

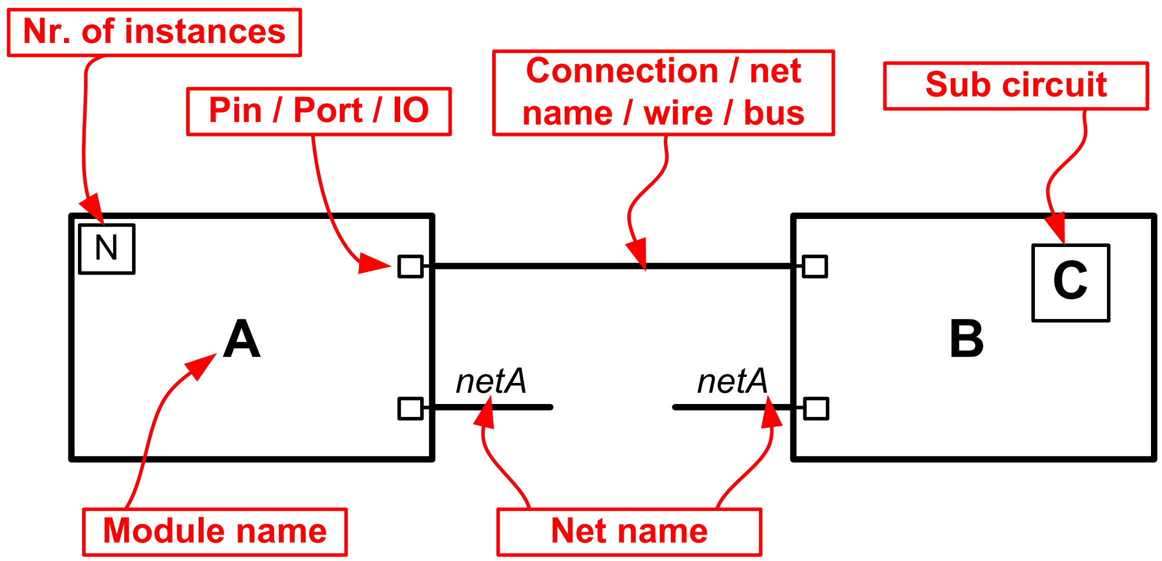


- Automatic generation of
  - equivalent SystemC test bench
  - pin accurate model frames

Proposed Model Generation Approach - Comparison

- Netlist (spectre) based approach
  - Parser needs extensive keyword training
  - Netlist is "flat" ➔ rebuild hierarchical structure
  - Sometimes, too much information for our target
    - Verilog/VHDL behavioral models, simulation controls, etc
- Direct approach
  - Hierarchical structure already exist in the data base
  - Use SKILL language to access the data base:
    - Pin / Port / IO direction
    - Connectivity

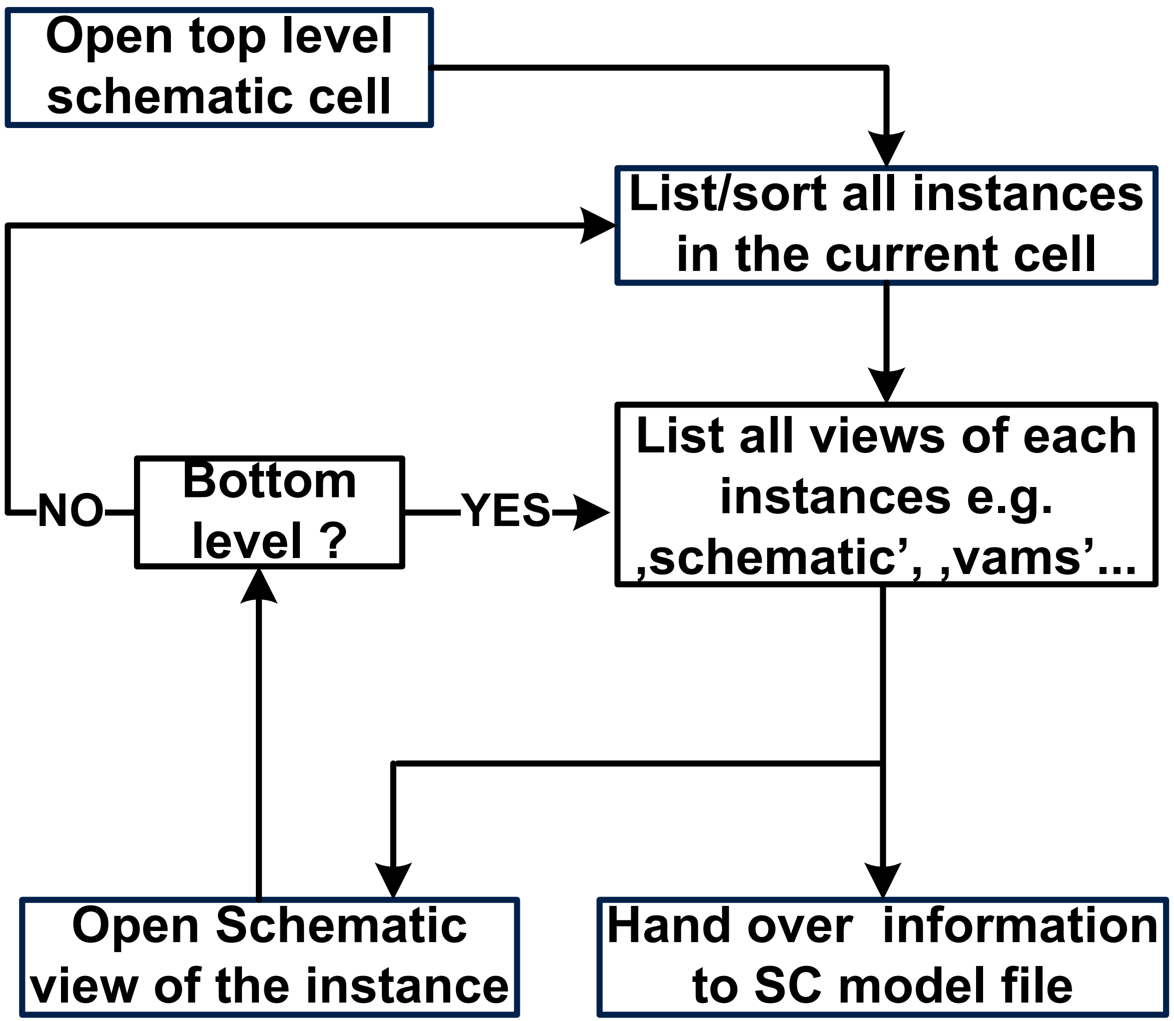
Proposed Model Generation Approach - Accessible information via SKILL



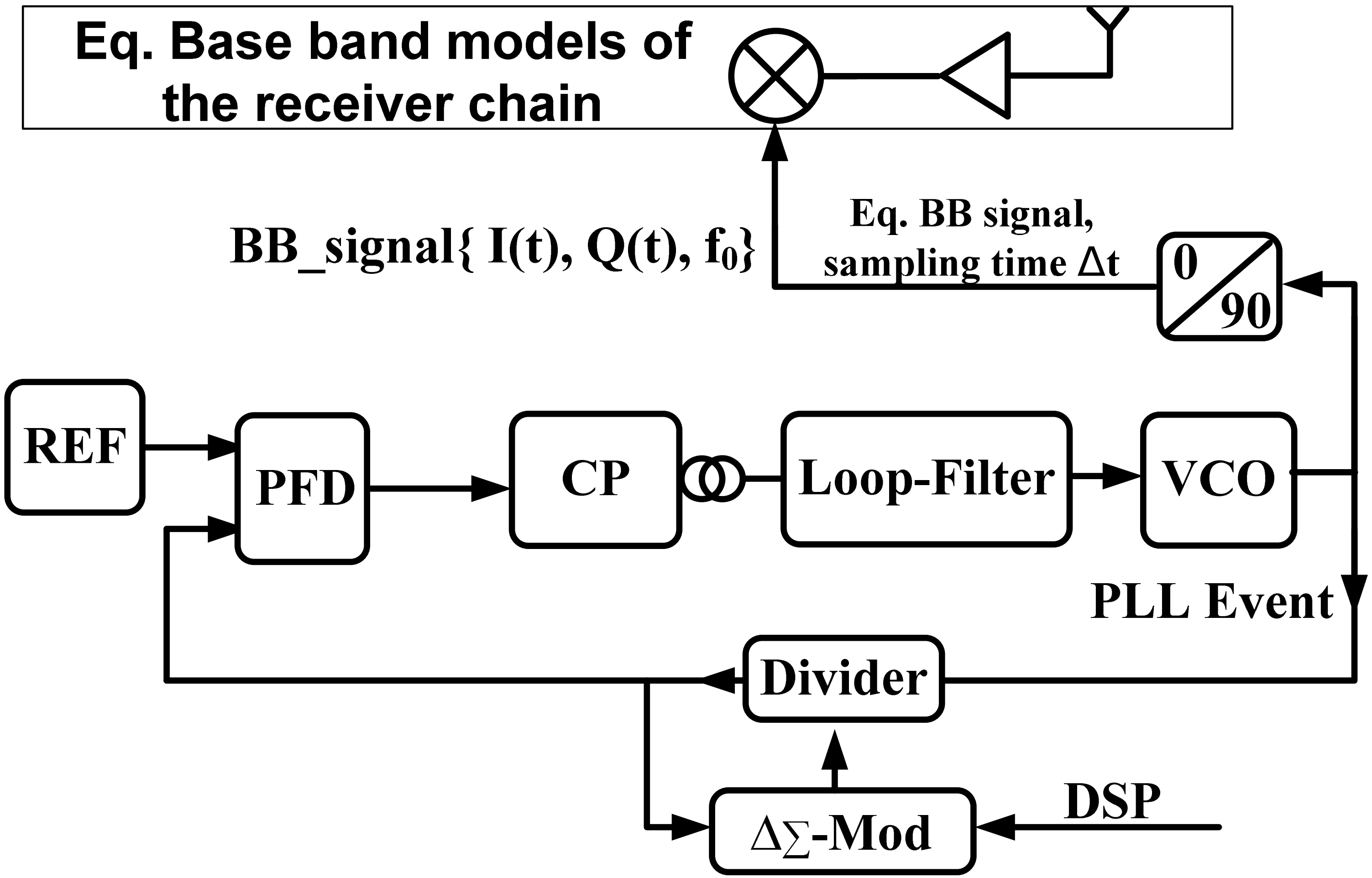
- Using SKILL under the platform of Cadence
  - High-level, interactive extension language
  - Built upon a *Lisp* interpreter
  - Access and control components of Cadence DFII

Proposed model generation approach - Flowchart

An iterative process

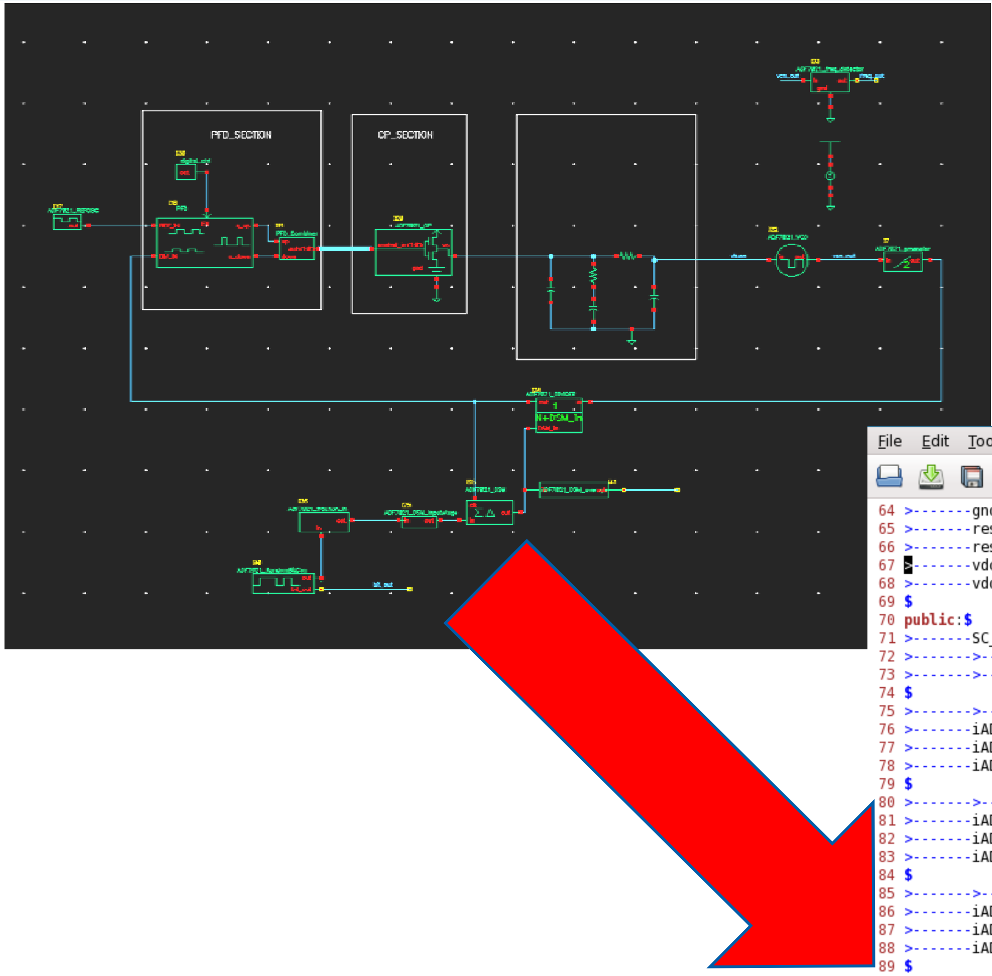


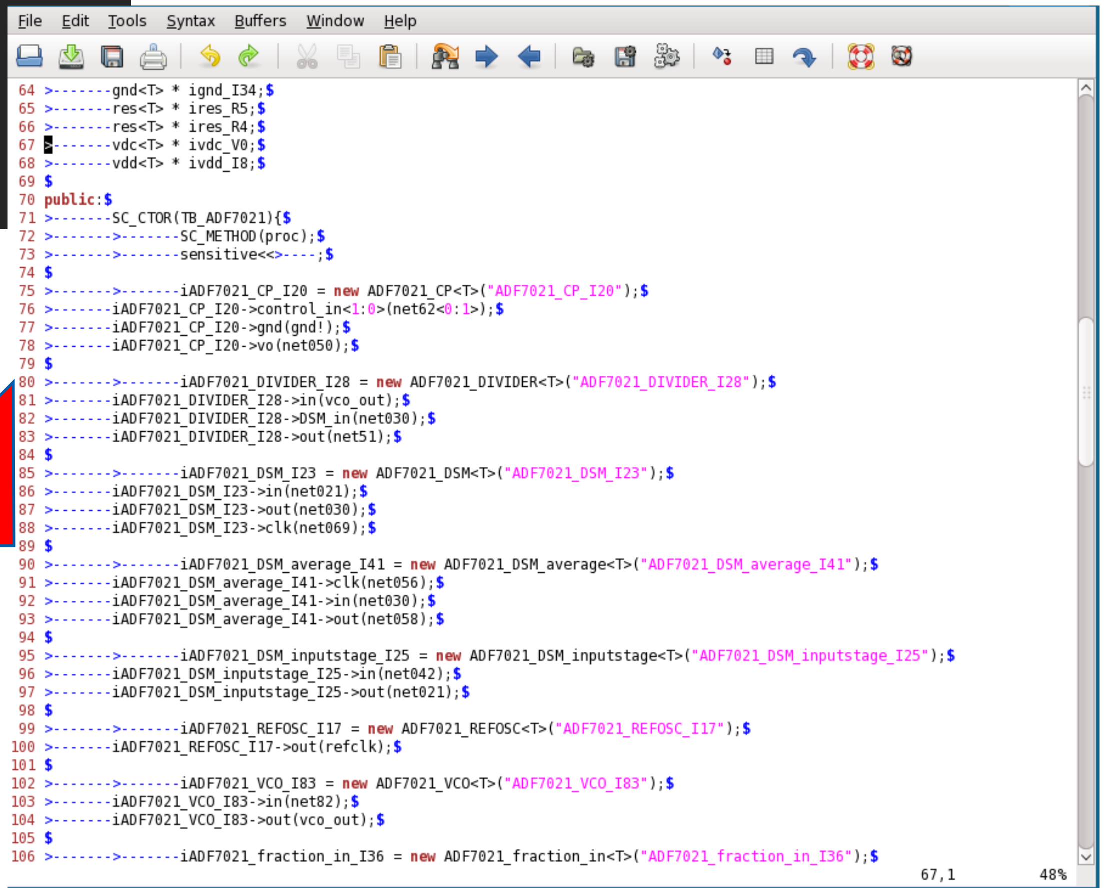
Application example



- Receiver chain contains baseband models
- PLL generates PB event and BB. phase

Application Example
- SystemC model frame generation






```
>-------gnd<T> * ignd_I34;$
>-------res<T> * ires_R5;$
>-------res<T> * ires_R4;$
>-------vdc<T> * ivdc_V0;$
>-------vdd<T> * ivdd_I8;$
$
public:$
>-------SC_CTOR(TB_ADF7021){$
>------->-------SC_METHOD(proc);$
>------->-------sensitive<<>----;$
$
>------->-------iADF7021_CP_I20 = new ADF7021_CP<T>("ADF7021_CP_I20");$
>-------iADF7021_CP_I20->control_in<1:0>(net62<0:1>);$
>-------iADF7021_CP_I20->gnd(gnd!);$
>-------iADF7021_CP_I20->vo(net050);$
$
>------->-------iADF7021_DIVIDER_I28 = new ADF7021_DIVIDER<T>("ADF7021_DIVIDER_I28");$
>-------iADF7021_DIVIDER_I28->in(vco_out);$
>-------iADF7021_DIVIDER_I28->DSM_in(net030);$
>-------iADF7021_DIVIDER_I28->out(net51);$
$
>------->-------iADF7021_DSM_I23 = new ADF7021_DSM<T>("ADF7021_DSM_I23");$
>-------iADF7021_DSM_I23->in(net021);$
>-------iADF7021_DSM_I23->out(net030);$
>-------iADF7021_DSM_I23->clk(net069);$
$
>------->-------iADF7021_DSM_average_I41 = new ADF7021_DSM_average<T>("ADF7021_DSM_average_I41");$
>-------iADF7021_DSM_average_I41->clk(net056);$
>-------iADF7021_DSM_average_I41->in(net030);$
>-------iADF7021_DSM_average_I41->out(net058);$
$
>------->-------iADF7021_DSM_inputstage_I25 = new ADF7021_DSM_inputstage<T>("ADF7021_DSM_inputstage_I25");$
>-------iADF7021_DSM_inputstage_I25->in(net042);$
>-------iADF7021_DSM_inputstage_I25->out(net021);$
$
>------->-------iADF7021_REFOSC_I17 = new ADF7021_REFOSC<T>("ADF7021_REFOSC_I17");$
>-------iADF7021_REFOSC_I17->out(refclk);$
$
>------->-------iADF7021_VCO_I83 = new ADF7021_VCO<T>("ADF7021_VCO_I83");$
>-------iADF7021_VCO_I83->in(net82);$
>-------iADF7021_VCO_I83->out(vco_out);$
$
>------->-------iADF7021_fraction_in_I36 = new ADF7021_fraction_in<T>("ADF7021_fraction_in_I36");$
```

Application example

- Output signal of PLL:
  - VCO_event for PB PLL
  - F0, I, Q for BB receiver chain

```
class PLL_out_type{
public:
        bool vco_event;        //square wave from VCO;
        double fnoise;         //PLL noisy frequency;
        double f0;             //average frequency of
                               //output of PLL;
        double I1;   //BB phase noise parameter VCO;
        double Q1;
 operator bool() const{
        return vco_event;
        }

 bool operator == (const PLL_out_type& rhs){
        //operator == overload
        return (vco_event == rhs.vco_event);
        }
 PLL_out_type & operator = (const PLL_out_type& rhs){
        vco_event = rhs.vco_event;    // = overload;
        fnoise = rhs.fnoise;
        ….
        return *this;
        }...
```

- Loopfilter: equivalent Z-domain model

Application Example
- VCO noise implementation

VCO module

- Phase noise calculation using Box-Muller algorithm:

Distribution function:

$$\Phi = \int_{-\infty}^{\infty} e^{-\frac{y^2}{2}} dy$$

$$\Phi^2 = \int_{-\infty}^{\infty} e^{-\frac{x^2}{2}} dx \cdot \int_{-\infty}^{\infty} e^{-\frac{y^2}{2}} dy = \iint_{-\infty}^{\infty} e^{-\frac{(x^2+y^2)}{2}} dxdy$$

Coordinate transformation:

$$x = rcos(\theta)\,,\ y = rsin(\theta)$$

$$\Phi^2 = \int_0^{2\pi} \int e^{-\frac{r^2}{2}}\ rdrd\theta \overset{q=\frac{r^2}{2}}{\Longleftrightarrow} = \int_0^{\infty} e^{-q}\ dq = 2\pi$$

- Normal distri. based on 2 uniform randoms (A, B)

$$P_1 = \sqrt{(-2lnA)} \cdot \cos(2\pi B)$$

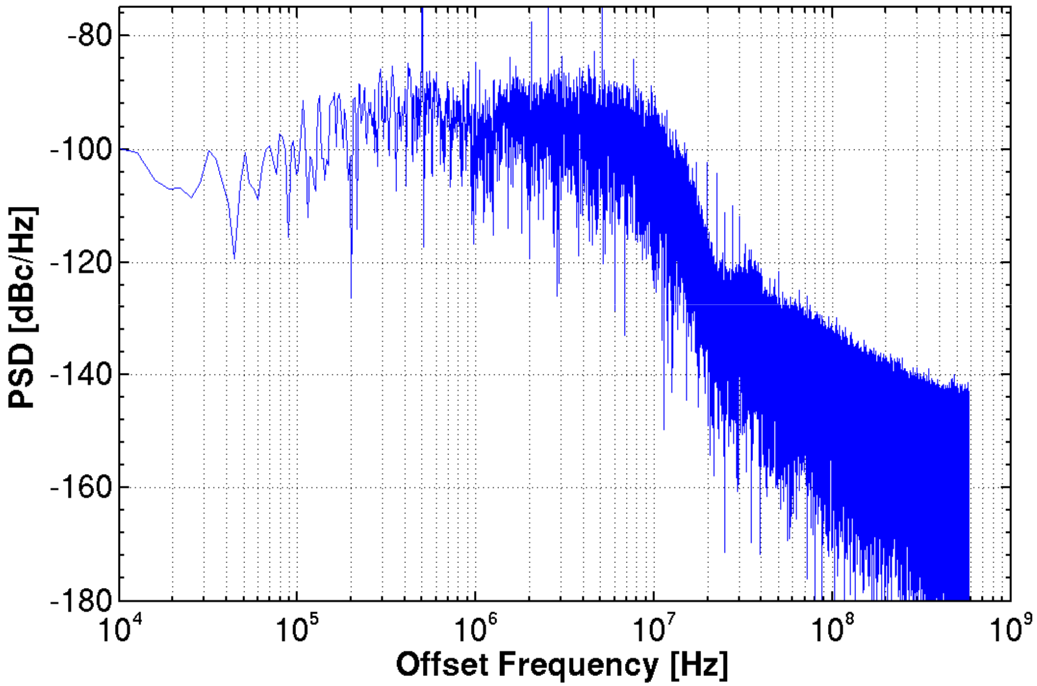
$$P_2 = \sqrt{(-2lnA)} \cdot \sin(2\pi B)$$

Application Example
- Simulation time

- PLL system implemented in
  - SystemC : purely event driven
  - SystemC-AMS: Timed Data Flow(TDF) models
  - Verilog-AMS wreal
- Simulation time

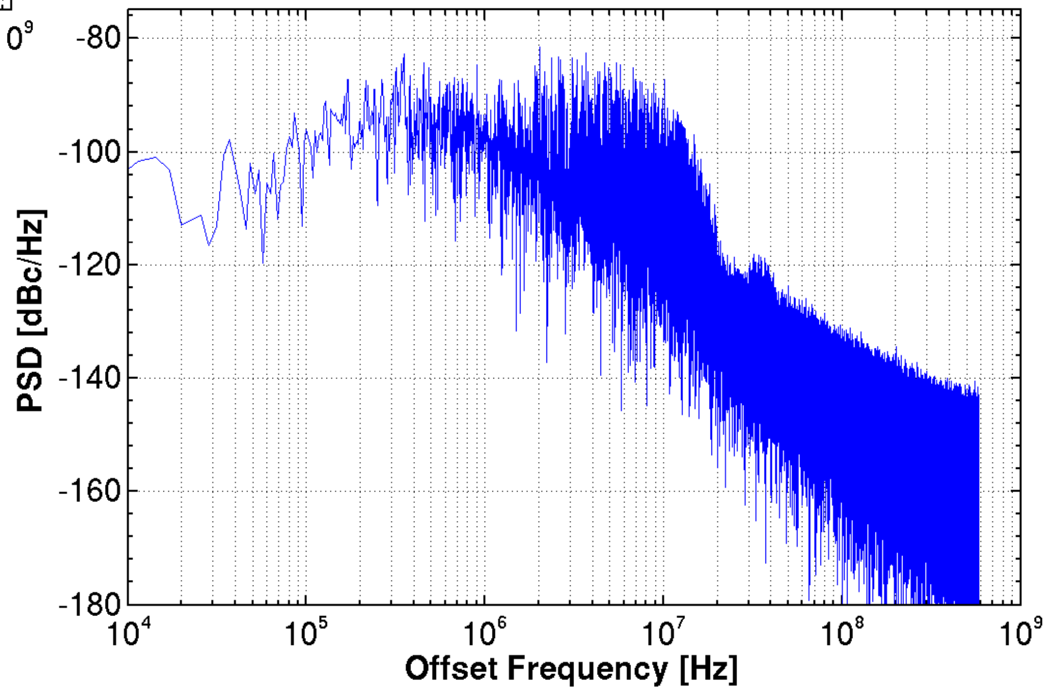
HDLs	Simulation time
SystemC 2.2.0	31 sec
SystemC-AMS 1.0	36 sec
Verilog-AMS(wreal)	40 sec

Application Example
- Phase noise simulation results



← Verilog-AMS models

SystemC Models →



Conclusions

- A method to hierarchically generate pin accurate SystemC RF models based on circuit data base
- The SystemC models are efficient and can be simulated in a pure digital environment
- Switchable PB /BB signal type without modification of the connectivity information
- Suitable for system design of digital centric nano-scale CMOS-RF system architecture

Thank you