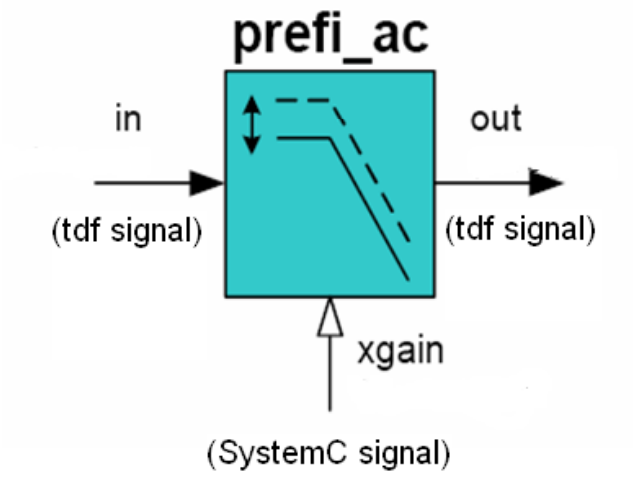
# Analog Behavior Refinement in System Centric Modeling


Yaseen Zaidi, Christoph Grimm and Jan Haase
Institute of Computer Technology
Vienna University of Technology

# The Motivation

- System level analog modeling possible with SystemC AMS:
  - Executable specification
  - AMS semantics
  - Models of computations e.g. Timed Data Flow (TDF)
  - Synchronization of MoCs of different domains
  - Constant stepping ⇨ fast simulation
- But need hooks for immediate refinement to utilize:
  - Granularity supported by HDLs and HDLs/AMS
  - Dedicated analog solvers
  - Fine/variable stepping
  - Synchronization

# Abstract modeling

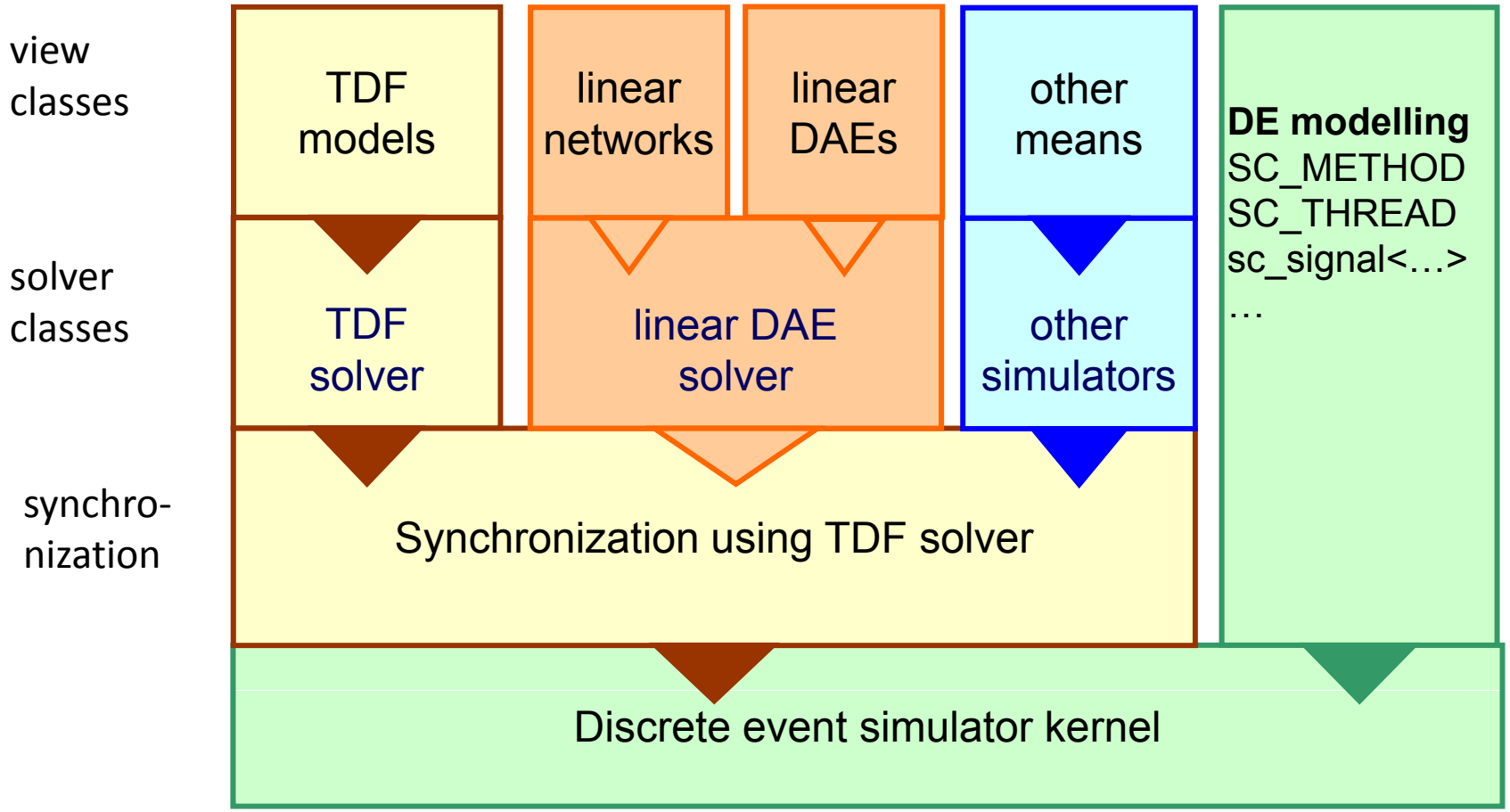



$$H(s) = \frac{1}{1 + \frac{1}{2\pi f_c} s}$$

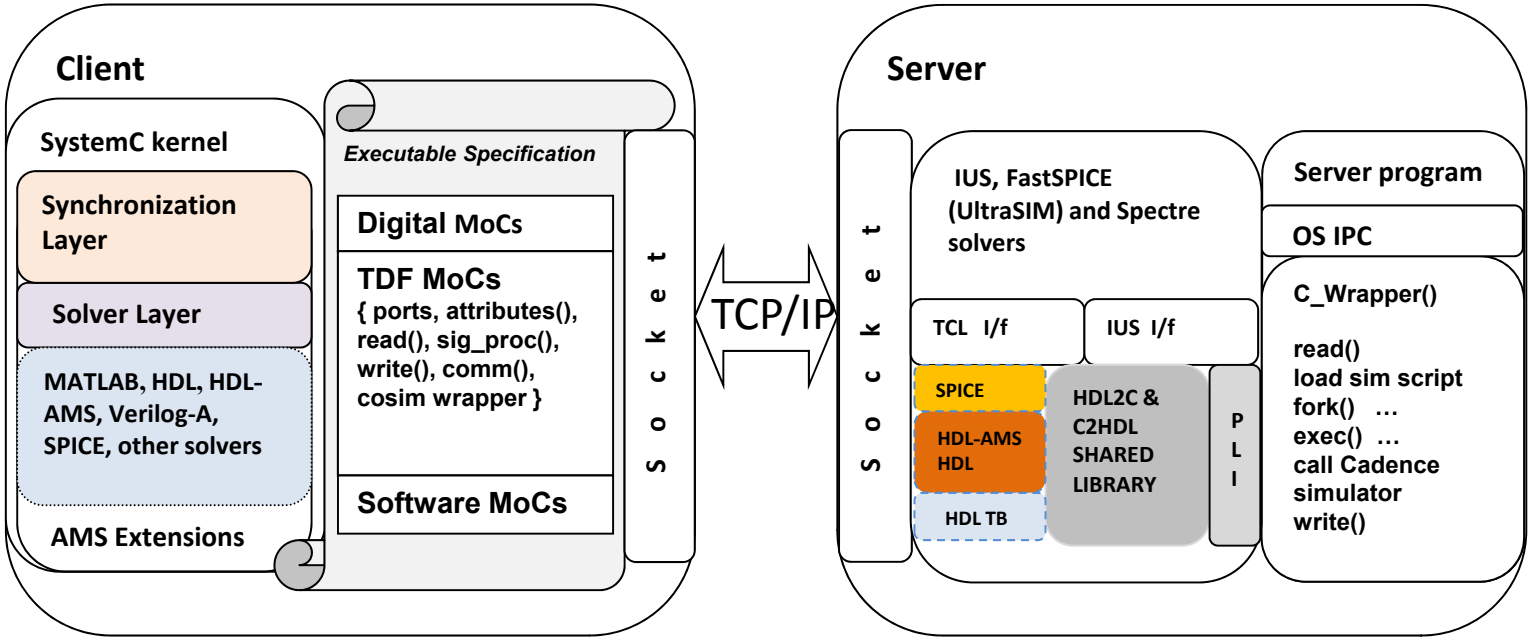
```
SCA_TDF_MODULE(prefi_ac)
{
sca_tdf_in<double> in;
sca_tdf_out<double> out;
sca_sctdf_in<bool> xgain;

// parameters
double prefi_fc; //cut-off freq
double prefi_g0; //gain !xgain
double prefi_g1; //gain xgain

// filter model
sca_ltf_nd ltf_1; //filter inst
sca_vector<double> A, B; //coeffs
sca_vector<double> S; //states
```

```
void init() {

//filter coeffs
B(0) = 1.0; A(0) = 1.0;
A(1) = 1.0/(2.0*M_PI*prefi_fc);}

void processing()
{
double tmp=ltf_1(B,A,S,in.read());
if (xgain.read())
out.write(tmp * prefi_g1);
else out.write(tmp * prefi_g0);}

SCA_CTOR(prefi_ac)
 { // defaults
prefi_fc = 1.0e6; prefi_g0 = 2.74;
prefi_g1 = 2.74 * 2.2;}
};
```

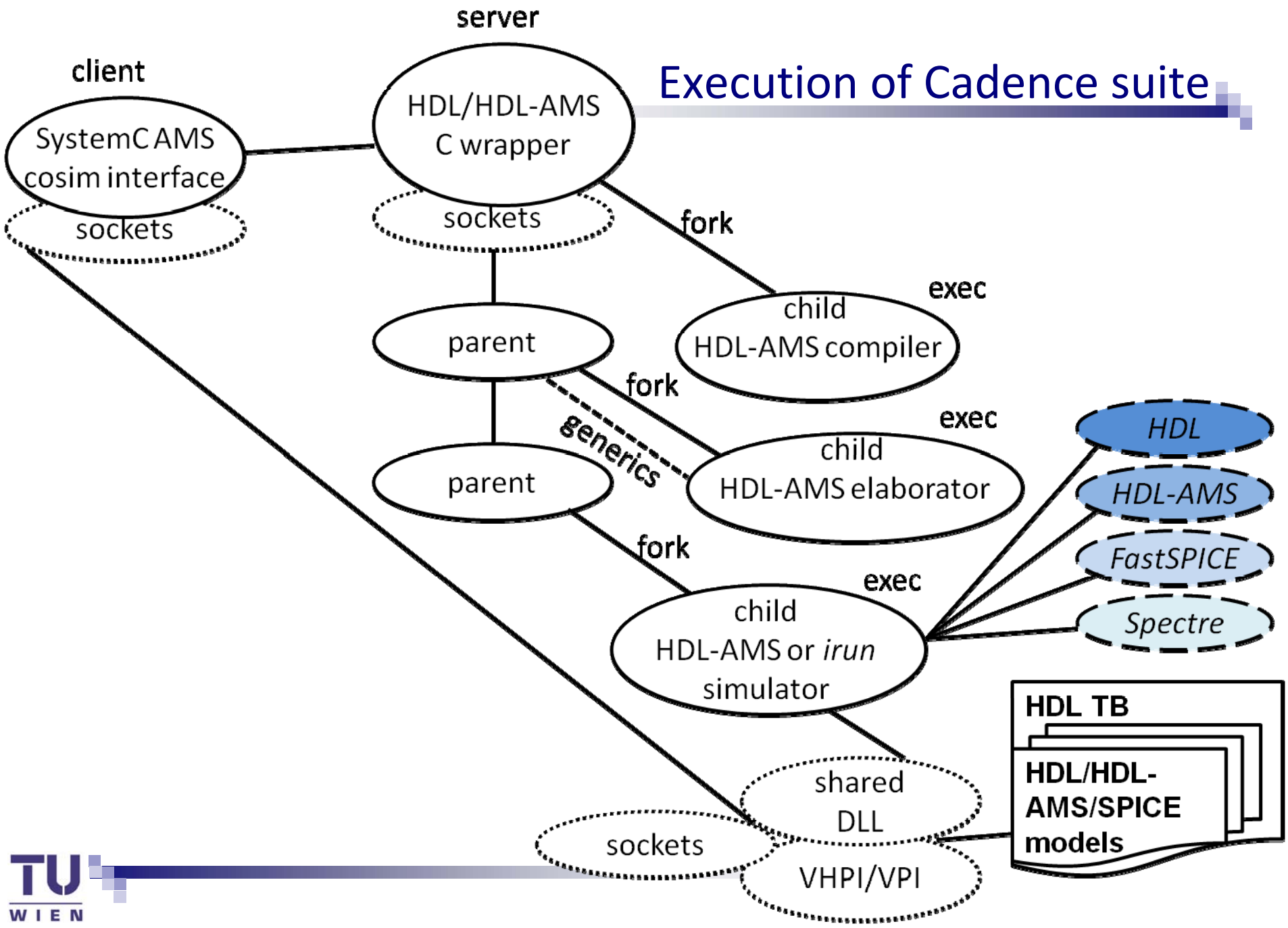
# System AMS can access and synchronize external simulators



# The Methodology

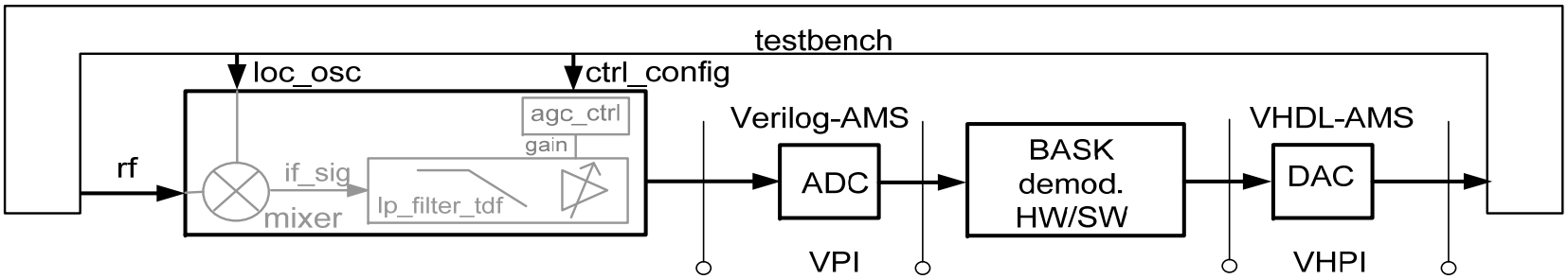


# Execution of Cadence suite



# Example system

- Simulate various ADCs and DACs in HDLs/AMS
- Use VPI for Verilog and VHPI for VHDL model access



# Cosimulation instance in SystemC AMS specification

```
SCA_TDF_MODULE(ad_converter)
{
sca_tdf::sca_in<double> in_tdf;
sca_tdf::sca_out<sc_int<12> out_de;
..
char *out_token_stream;
void processing() {
token_collection = format_and_queue(in_tdf.read());
out_token_stream = cadence_cosim(token_collection);
..
out_de.write<static_cast<sc_int<12> > format(out_token_stream);}
}
```

# Calling Cadence solver and C access of simulation

```
child_pid_ncelab = vfork();
execv("ncelab", "-amsfastspice",
"-propspath", "prop.cfg", "bench_a2d_12bit", "-snapshot",
"worklib.bench_a2d_12bit");
child_pid_ncsim = vfork();

execv("ncsim", "-input", "@tcl_script", "-status", "-analogcontrol",
"acf.scs", "worklib.bench_a2d_12bit:behav",
"+loadvhpi", "VHDL2C_DLL",
"inst=:bench_a2d_12bit", "+start=0", "+stop=62");
```

# Conclusion

- SystemC AMS TDF semantics while suited for high level analog modeling can assist in cosimulation with refined models of HDLs-AMS
- Synchronization layer allows connections of specialty simulators e.g. Spectre and FastSPICE
- A client (SystemC AMS) and server (Cadence) topology realized
- C/UNIX calls control of Cadence simulation
- Access of simulation objects is made using standardized procedural interfaces (VPI/VHPI)

Thank you