MunEDA Corporate Overview

• EDA Software Vendor – Tools SPT & WiCkeD™ for Migration, Sizing and Verification of full custom nanometer IC designs.

• Worldwide Sales & Support Offices in USA, Korea, China, Taiwan, Japan, UK, Ireland, Scandinavia, South America

• Worldwide Customer & Partner Base with Semiconductor IDMs, Fabless Design Houses & Foundries

For more information visit www.muneda.com
MunEDA EDA Solutions for Full Custom IC Migration, Sizing & Verification

MunEDA – Comprehensive EDA Tools for Migration, Sizing and Verification of Nanometer Custom IC Designs

MunEDA SPT
Schematic Porting Tool for IC & IP Migration

Circuit Migration with SPT 2.0
MunEDA Schematic Porting Tool for circuit schematic migration and IP porting between different process technologies

MunEDA WiCkeD™
Circuit Sizing & Tuning

Circuit Sizing & Tuning with WiCkeD™
Meet performance specifications over all types of variation, design and operation parameters with MunEDA sizing & optimization tools

MunEDA WiCkeD™
Circuit Analysis & Verification

Circuit Verification with WiCkeD™
Analyze and verify your circuit designs for constraints, performance specifications, operating and process parameters, global and mismatch variation, reliability and others
MunEDA WiCkeD™ Tool Suite for Circuit Sizing & Tuning

Selected Features:
- Circuit Constraint & Feasibility Optimization
- Deterministic Nominal Circuit Optimization
- Global Nominal Circuit Optimization
- Discrete Circuit Optimization
- Yield Optimization
- Circuit sizing under consideration of degradation and reliability effects
- Circuit Design History & Database
- Parallel simulation environment (supporting industry-standard SPICE simulators)
- Multitestbench-Support

MunEDA WiCkeD™ Circuit Sizing & Tuning: simulation-based tools to improve and optimize custom circuits

Targets constraint fulfilment, performance specifications, robustness, targets for yield, area, stability, timing, power, reliability, aging, degradation, stress, self heating and more!
MunEDA WiCkeD™ Circuit Analysis & Verification

**Selected Features**
- Parallel simulation environment (supporting industry-standard SPICE simulators)
- Multitestbench-Support
- Circuit Constraint Feasibility Check
- Circuit Sensitivity Analysis for performances, gain, frequency, slewrates, power, stability, area, others
- Fast PVT & Operating Corner Analysis: influence of corner cases on given circuit performances
- Parameter Screening
- Fast & Enhanced Monte Carlo Analysis (3-5 sigma)
- High Sigma Worst Case Analysis (6-9 sigma)
- Importance Sampling & Robustness Verification
- Global and local (mismatch) variation and yield analysis
- Reliability & aging analysis for degradation effects
- Yield plot sweeps for global variation
- RSM Circuit Modelling & Model Generation

**MunEDA WiCkeD™ Circuit Analysis & Verification:** comprehensive and powerful tools for simulation-based analysis and verification of Custom IC

**Targets circuit constraints, design parameters, PVT, operating corners, specifications, sensitivities, correlations, process dependencies, sweeps, global and mismatch variation, Monte-Carlo, Worst-Case, High Sigma, reliability and more!**
Motivation

**Complexity**

- Analysis and optimization requires multiple simulations
- Works well with simulation times in the range of seconds/minutes
- Successfully applied with spice simulations at block level (OpAmps, Bandgaps, Memory cells, standard cells, etc.)
- For more complex circuits with simulation times of several hours (i.e. PLL, ADC) → no fun
Motivation

**Approaches**
- Methodologies to create behavioural models of complex circuits → stuck in tradeoff between effort to create the model vs. accuracy
  - Methodology to break up in blocks, model and simulate mixed-mode → cumbersome, mixed-tool environment

**Requirements**
- Simplified creation of AMS Models
- Mixed-Level Simulation for optimization, verification, and debugging

**Solution via Coside**
- System C / System C AMS enables simulation of complex circuits X times faster than spice level
- Coside practical tool for modeling, mixed-simulations, and debugging
Mixed-level analysis / optimization

Initial export is system level, but as soon as individual blocks get implemented, optimization on mixed level is possible

Note: Mixed level is possible within Coside via SystemC / SystemC AMS Spice Integration
Mixed-level optimization within normal EDA design framework
STM paper in 2010
optimization of VCO within Dithered PLL
Using Coside to Apply Optimization to System

- Full hierarchcial flow: use WiCkeD @ system level
- WiCkeD / Coside interface → optimize using SystemC simulation

- Mixed level is possible by Coside SystemC / SystemC AMS Spice Integration
- Optimization can include „other components“ (mechanical, etc.)
System-Level Optimization

- System-level optimization with SystemC AMS
- Conceptual test: DC-DC converter

Analysis and optimization done at system level
(prior to implementation)

DC-DC converter example

DCDC Systemlevel model
DC-DC converter example

Testbench

Parameterized
(Poles, Zeros, Gain, Frequency)

\[ \text{fp1} = 7.5 \times 10^3 \]
\[ \text{fp2} = 531 \times 10^3 \]
\[ \text{fz1} = 806.0 / 2.0 \]
\[ \text{fz2} = 806.0 / 2.0 \]
\[ \text{loop_gain} = 100.0 \]
\[ \text{rload} = 10.0 \]
\[ \text{vref} = 5.0 \]
\[ \text{vsupply} = 24 \]
\[ \text{saw_freq} = 25 \times 10^3 \]
DC-DC converter example

Input parameter file

//parameter for optimization
fp1 = 7.5e3; //loop filter poles and zeros
fp2 = 531e3;
fz1 = 403.0;
fz2 = 403.0;
loop_gain = 10000.0; //loop gain
saw_freq = 25e3; //sawtooth generator frequency

//configuration parameter
rload = 10.0; //load resistance
vref = 5.0; //reference voltage
vsupply = 24; //supply voltage

trace_file="onchar.dat"; //trace file

//measurement configuration
settling_tolerance=0.1; // +/- vref to be considered as inbound
settle_time=5e-3; // inbound time to be considered as settled
max_settle_time=50e-3; //maximum measurement time -> if not settled within this time -> unsettled
DC-DC converter example

Testsequence (to calculate settling time)

```c
void dcdc_tb_stim_on::stimulus_sequence()
{
    std::cout << sc_core::sc_time_stamp() << " Stimulus sequence started...
    " << std::endl;

    const sc_core::sc_event& change_ev=vout.get_event(); /* get event for dc dc out*/
    wait(1.0,SC_MS);

    s.dcon.set_value(true); //switch converter on
    double dcon_time=sc_time_stamp().to_seconds();

    check_state cstate=OUTBAND;
    double last_outtime=0.0;
    double settling_time=1e9;
    double max_overshot=0.0;
    double max_undershot=0.0;
    bool finished=false;
    bool shot=false;

    while(true)
    {
        wait(change_ev); /* continue if next dcdc out value available*/
        double vout= s.vout.get_value();

        if(vout > dut_p->vref) shot=true; //if once over vref -> start shot measurement
        if(shot)
        {
            if(vout-dut_p->vref > max_overshot) max_overshot=vout-dut_p->vref;
            if(vout-dut_p->vref < max_undershot) max_undershot=vout-dut_p->vref;
        }
        double ctime=sc_time_stamp().to_seconds();

        switch(cstate)
        {
        case INBAND:
            if(fabs(vout-dut_p->vref)>s.settling_tolerance) cstate=OUTBAND;
            if((ctime-last_outtime)>s.settle_time) //settled
                {
                    settling_time=cite-time-dcon_time.s.settle_time;
                    settled=true;
                }
            break;

        if(finished)
            {
                std::cout << "Settling time: " << settling_time << std::endl;
                std::cout << "overshot: " << max_overshot << std::endl;
                std::cout << "undershot: " << max_undershot << std::endl;
                break;
            }
        }
        break;

        if(sc_time_stamp().to_seconds()-dcon_time > s.max_settle_time)
            {
                std::cout << "Not settled" << std::endl;
                break;
            }

        double dut_p=vref;
        double settling_tolerance=s.settling_tolerance;

        double ctime=vout; /* get event for dc dc out*/

        fstream fstr("dcdc_result.dat");
        fstr << std::setprecision(15);
        if(finished)
            {
                fstr << "Settling time = " << settling_time << std::endl;
                fstr << "overshot = " << max_overshot << std::endl;
                fstr << "undershot = " << max_undershot << std::endl;
            }
        else
            {
                fstr << "Settling time = " << settling_time << std::endl;
                fstr << "overshot = " << max_overshot << std::endl;
                fstr << "undershot = " << max_undershot << std::endl;
            }
        fstr.close();
        std::cout << sc_core::sc_time_stamp() << " Stimulus sequence finished. " << std::endl;
        wait(1.0,SC_MS);
        sc_core::sc_stop();
    }
};
```

Confidential
DC-DC converter example

Example for simulation result

![Graph showing simulation results for a DC-DC converter example.](image)
DC-DC converter example

Simulation Result
**Optimization Example**

**Parameters:**
- two poles (fp1, fp2)
- two zeros (fz1, fz2)
- loop gain
- saw frequency

**Optimization Target:**
- Minimize Settling Time
- Minimize Settling Time
  - Over full operating range: (2Ω ≤ load resistance ≤ 200Ω)

**Performances:**
**Settling Time**

- **Original**: Settling time = 4.17 ms
- **Manual Optimization**: Settling time = 1.6 ms
- **WiCkeD Optimization**: Settling time = 0.76 ms

149 Simulations, 55s run time (single core machine, no parallel simulation)

vs. manual: ~1 hour

170 Simulations, 59s run time (single core machine, no parallel simulation)

Manual estim.: >1 day

**too complex**
Summary

- Simple demonstrator showed tool coupling example
- Optimization worked fast and showed good results (applicable to far higher complexity than this simple demonstrator)
- Interesting learnings / observations from this simple example:
  - Difference between implementation and system level not just different abstraction: Parameters at system level are not physical dimensions (as transistor widths)
  - Parameters at system level can be performances (results) of sub-blocks → completely new field of applications opening up
- Outlook:
  - Same environment can be extended to more complex circuits
  - Interfaces at system level enable cross domain simulation (analog/digital, hardware, software, embedded, mechanical, etc.)
Thank You!