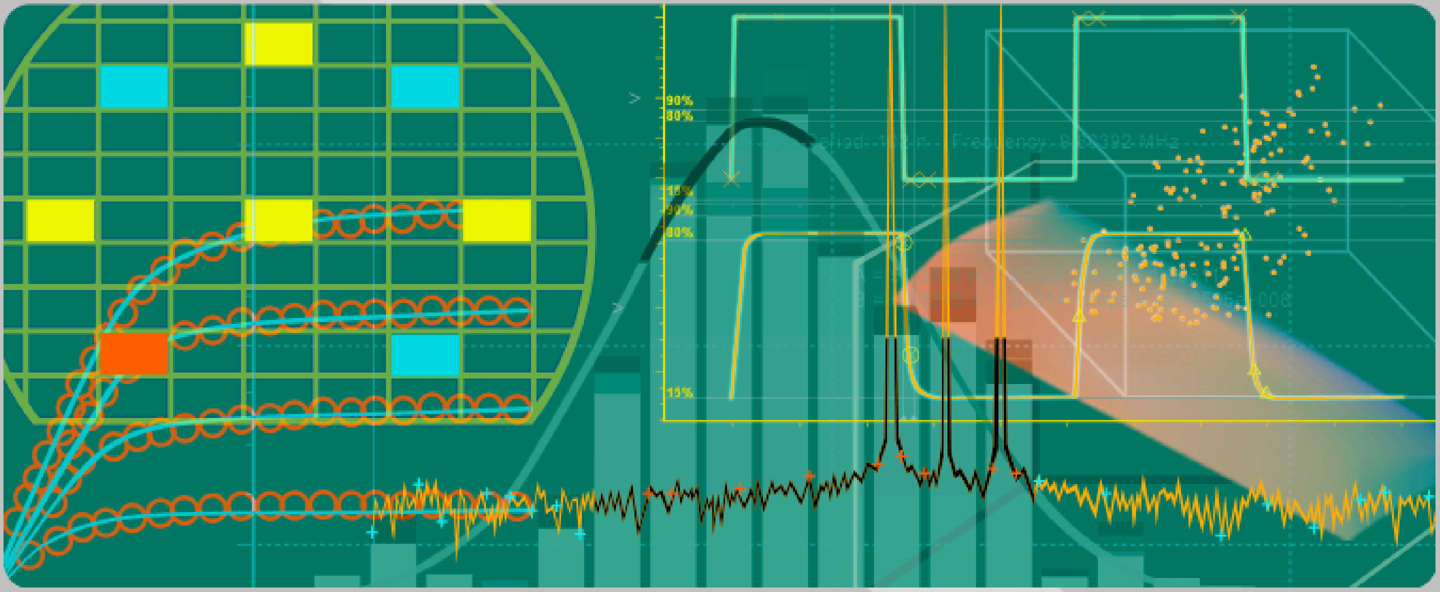
We Deliver Simulation and CAD Solutions

# SmartSpice Training Program



Part 4: SmartSpice Optimizer

SIMUCAD DESIGN AUTOMATION

6/20/06

## What Is SmartSpice Optimizer?

- The SmartSpice Optimizer is a fully integrated feature of SmartSpice that performs parametric optimization of circuits
- At the core of the optimizer, is a general purpose optimizing engine that requires initial and target parameter values to be set
- The optimizer then iterates these parameters until the target values are reached
- The optimizer provides a comprehensive interface and an interactive display system for visualizing the optimization process as it is executed

**Performance Measure Optimization**

- In performance measure optimization, the values and parameters of circuit components are automatically altered so that individual electrical characteristics meet specifications
- The Optimizer can fine-tune delay, rise/fall times, trip point, maximum and minimum current, and any other circuit performance measurement that can be calculated by a SmartSpice .MEASURE statement
- Simultaneous multi-target optimization of several performance measures is also possible

**Function Optimization**

- Function optimization matches calculated curves with desired curves for DC, AC, and transient analyses. The desired curves can represent the results of theoretical research or physical measurement
- There are no restrictions on the type of circuit analysis that can be performed. Circuits can be optimized in steady state, frequency, and time domain

## Optimizer Using `.MODIF`

- The **`.MODIF`** statement is a powerful design feature, which can be used to perform **parametric optimization and statistical analysis**
- Two stop conditions are available to investigate a circuit for sets of parameters that change over a specified range:
  - A limit on the number of repetitions of an analysis.
  - A logical condition on a circuit performance measurement
- A parameter set contains any number of the following parameter types:
  - parameter labels defined in a global **`.PARAM`** statement
  - model parameters
  - device parameters
  - temperature parameters
  - parameter GMIN/DCGMIN

## SmartSpice Optimizer Syntax

- To perform optimization in SmartSpice, the .MODIF statement must be used. For a circuit with a known topology, the .MODIF statement allows you to:
  - Solve a separate optimization problem
  - Sequentially solve several optimization problems
  - Sequentially solve optimization problems and perform parametric analysis
- **Syntax**

```
.MODIF <PROFF> <PRTBL> <RESTORE> <constpar_spfc>+ OPTIMIZE
param_spfc + TARGETS targ_spfc + OPTIONS <opt_spfc>
<<modif2_spfc> ... >
```

- **`PROFF:`** This flag suppresses online printing of .MEASURE statement results
- **`PRTBL:`** This flag causes SMARTSPICE to print the final table of parameter and measure values obtained from all iterations
- **`RESTORE:`** This keyword causes SMARTSPICE to restore all modified parameters in the specified set to their original values
- **`constpar_spfc:`** Definitions for all parameters that will be held constant through the optimization process
- **`OPTIMIZE:`** This keyword is followed by the list of optimization parameters
- **`param_spf:`** Definitions for all parameters that will be changed during optimization. For each parameter a minimum, maximum, and initial value must be specified

## SmartSpice Optimizer Syntax (cont)

- **`TARGETS:`** This keyword is followed by the list of targets for optimization
- **`targ_spf:`** Definition of all targets that will be optimized. For each target, the name of a measured and a desired value must be specified
- **`OPTIONS:`** This keyword is followed by the list of optimization control options
- **`opt_spfc:`** Optimizer control options. You can specify an option and value to replace a default option.
- **`modif2_spfc:`** This definition follows the keyword MODIF. The procedure defined here is called immediately after the final iteration of the current optimization. The procedure could be another optimization or a parametric analysis. In either case, the circuit parameters correspond to those just calculated by the current optimization

# SmartSpice Optimizer Input Deck

- A typical input deck created for optimization contains:
  - One or more analysis statements to simulate a circuit in either steady state, frequency, or time domain
  - One or more .MEASURE statements to calculate performance measurements or differences between simulated and desired curves
  - One or more .DATA statements to describe desired curves for function optimization
  - One or more .PARAM statements to define parameter labels if needed
  - A .MODIF statement to define the names of device and model parameters; their minimum, maximum and initial values; targets; and control options for optimization
  - A .IPLOT statement to plot output variables while the optimization is in progress
- In order to match circuit specifications, the Optimizer performs a number of iterations, beginning with the user-defined initial values of parameters
- On each iteration, the Optimizer simulates the circuit, calculates target electrical specifications, updates parameters using defined mathematical strategies, and simulates the circuit again
- The Optimizer will stop the optimization process when one or more of the stop criteria specified in the Optimizer control statements is satisfied
- If SmartSpice is running in window mode, the intermediate results of the optimization process are immediately visible
- SmartSpice allows you to observe the history of parameter, and target values obtained from all iterations
- A **Stop** button can be used to stop the optimization process on any iteration; change parameter values, targets, or control options; and continue the optimization process from the last parameter values calculated

## Timing Optimization Using Bisection

- Bisection is a method of optimization that uses a binary search to find the value of an input variable for specified value of an output variable (goal)
- Over some interval the function is known to pass through goal value because “function value - goal” changes sign
- Evaluate the “measured function value - goal” at interval’s midpoint and examine its sign
- Use the midpoint to replace whichever limit has the same sign
- After each iteration the bounds containing the root decrease by factor of two
- When the size of interval (distance between bounds) is within the error tolerance and latest “measured function value” > “goal”, bisection has succeeded, and stop
- **Note**: Bisection search is applied to only one parameter

## SmartSpice Bisection Optimization Syntax

```
.MODIF <PROFF> <PRTBL>
+ OPTIMIZE parname = OPT(lower upper <initval>)
+ TARGETS measname = goalval
+ OPTIONS METHOD=val <MAXERR=val> <NUMITER=val>
```

- **PROFF**: This flag suppresses online printing of .MEASURE statement results.
- **PRTBL**: This flag causes SMARTSPICE to print the final table of parameter and measure values obtained from all iterations.
- **OPTIMIZE**: This keyword is followed by the bisection optimization parameter.
- **parname**: Name of the bisection optimization parameter.
- **OPT**: Mandatory keyword followed by lower, upper and initval in parentheses.
- **lower**: Left bound for bisection optimization parameter.
- **upper**: Right bound for bisection optimization parameter.
- **initval**: Initial value for bisection optimization parameter. Default initval=(lower+upper)/2.
- **TARGETS**: This keyword is followed by the target for optimization.
- **measname**: Name of the measure calculated by a .MEASURE statement. It can be a circuit performance measure like delay, rise/fall time, maximum/minimum value, or difference between a desired and a simulated curve calculated by the ERR .MEASURE statement.
- **goalval**: Goal value.
- **OPTIONS**: This keyword is followed by the list of optimization control options.

## SmartSpice Bisection Optimization Syntax (cont)

- **METHOD=val:** Keyword to indicate which bisection optimization method to use:
  1. Bisection method, the measure results for lower and upper bounds of optimization parameter must be on opposite sides of goal value.
  2. PASSFAIL method, the measure must pass for one limit and fail for the other limit.
- **MAXERR=val:** Relative optimization parameter error tolerance. When the difference between the two latest test input values is smaller, then:
  - parTol = max(interval x MAXERR, 10E–16)
- where:
  - Interval = max(initval - lower, upper - initval)
  - Bisection optimization process will be terminated. Default is 0.01.
- `NUMITER=val`**:** Maximum number of iterations. The bisection optimization process will be terminated, when the number of iterations reaches:
  - ITERlimit = max (log(1 / MAXERR) / log(2) + 10, NUMITER)
  - Default is NUMITER=15.
- **Example**:

```
.PARAM vddv=5
.PARAM DelayTime=0
.MODIF
+ OPTIMIZE DelayTime = OPT(0.0n, 5.0n, 0.0n)
+ TARGETS MaxVout = vddv
+ OPTIONS METHOD=1 MAXERR=1.e-3 NUMITER=20
.MEASURE TRAN MaxVout MAX ‘vddv-v(D_Output)’
```

# HSPICE Compatibility Syntax

- To perform Bisection optimization using HSPICE compatibility syntax, the .PARAM, .TRAN, .MODEL and .MEASURE statements must be used.

```
.PARAM parname = OPTxxx(initval lower upper)
.TRAN tstep tstop <tstart> <tmax> <UIC>
+ <CALLV> <SAVEV < =tsave>> <TRANOP < =top>> <STORE=num>
+ SWEEP OPTIMIZE = OPTxxx RESULT = measname MODEL=modname
.MODEL modname OPT <METHOD = BISECT<ION> | PASSFAIL>
+ <RELIN=val> <ITROPT=val>
.MEASURE TRAN measname ... <GOAL < |=| > val>
```

- **parname**: Name of the Bisection optimization parameter.
- **OPTxxx**: Optimization parameter reference name must agree with the OPTxxx name given in the .TRAN statement associated with the keyword OPTIMIZE.
- **initval**: Initial value for bisection optimization parameter.
  - Default is initval=(lower+upper)/2.
- **lower**: Left bound for bisection optimization parameter.
- **upper**: Right bound for bisection optimization parameter.
- **OPTIMIZE**: This keyword is followed by the Bisection optimization parameter.
- **RESULT**: This keyword is followed by the target for optimization.
- **measname**: Name of the measure calculated by a .MEASURE statement. It can be a circuit performance measure like delay, rise/fall time, maximum/minimum value, or difference between a desired and a simulated curve calculated by the ERR .MEASURE statement.
- **MODEL**: This keyword is followed by the optimization model.
- **modname**: The model name. It is used by Bisection optimization to reference a particular model.

## HSPICE Compatibility Syntax (cont)

- METHOD: Keyword to indicate which bisection optimization method to use. Default is BISECTION.
- BISECTION: The measure results for lower and upper bounds of optimization parameter, must be on opposite sides of goal value.
- PASSFAIL: The measure must pass for one limit and fail for the other limit.
- RELIN=val: Relative optimization parameter error tolerance. When the difference between the two latest test input values is smaller, then:
- parTol = max(interval x RELIN, 10E-16)

where

  - Interval = max(upper – lower),
  - and for "=" relation case
  - |val – goal|<max(|goal| x RELIN, 10E-16) with val > goal,
  - Bisection optimization process will be terminated. Default is RELIN value is 0.001.
- **`ITROPT=val`**: Maximum number of iterations. The Bisection optimization process will be terminated, when the number of iterations reaches:
  - ITERlimit = max(log(1 / MAXERR) / log(2) + 10), ITROPT)
  - Default is NUMITER=20.
- **`GOAL=val:`** Goal value. Default is 0.
- **Example**:

```
.PARAM vddv=5
.PARAM DelayTime=Opt1(0.0n, 0.0n, 5.0n)
.TRAN .1n 8n
+ SWEEP OPTIMIZE = Opt1 RESULT = MaxVout MODEL = OptMod
.MODEL OptMod OPT METHOD=BISECTION
.MEASURE TRAN MaxVout MAX 'vddv-v(D_Output)' Goal = 'vddv'
```