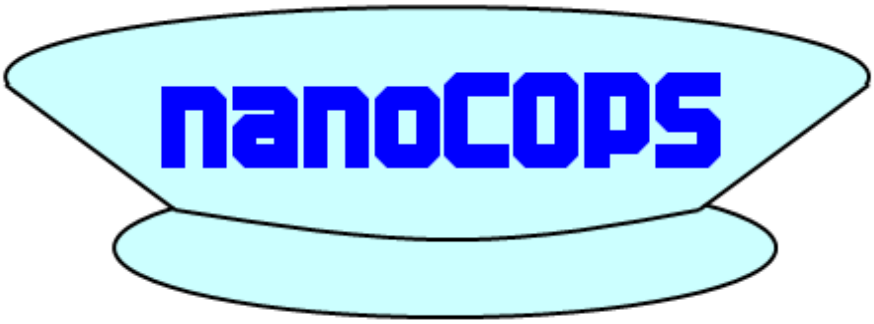

Project:	**nanoCOPS**
Project Number:	FP7-ICT-2013-11/619166
Work Package:	WP3: Validation, Demonstration and Measurements
Task:	T3.1 Tool development test set and data format specification
Deliverable:	D3.1b + D3.1c (version 1.0, Final)
Scheduled:	M6

Title:	Tool Development Test Set (a) and data format specification (b); Interfaces implementations, languages (c). **Part b+c: Data Format Specification and Interfaces Implementations, Languages**
Author(s):	Wim Schoenmaker, Rick Janssen, Frederik Deleu, Aarnout Wieers, Roland Pulch, Jan ter Maten
Affiliation(s):	MAG, NXP, ONN, UGW, TUD, HUB, MPG, FHO, KUL, ACC, BUW
Date:	7 May, 2014

Table of Contents

1 Introduction

The format of IO files used to store descriptions of layout, circuits, models, components and S-parameters at several levels of abstraction will be specified or selected from standards (such as GDSII, Spice, SnP, XML, etc.). A repository of specifications and standard references will be provided and maintained during the project. Several tools, such as GDSII/XML, SnP/Spice translators or other utility modules from public domain or developed by nanoCOPS partners can be shared by the consortium.
In the first section 2.1, possible circuit netlist formats are specified. Output from the nanoCOPS toolset will be a SPICE-compatible network, which in principle can be analyzed with all circuit simulators to be used in the project. We will use the standard HSPICE format. Spectre and ADS are the most important simulators in the RF field and therefore also Spectre and ADS output formats are possible.
The next section 2.2 about input file formats (Layout), gives an overview of the GDS II format, together with port information and process information, which forms a complete input description for EM simulation. Then an overview is given of the XML format, which can accommodate all relevant data for field solving as well, such as geometric, physical and solver data.
In section 2.3 some examples are given of existing compact models of active devices, which can be used as a starting point for the nanoCOPS development. A compact model description for bipolar transistors, the Mextram model is given. The Mextram model was developed within Philips/NXP, but has become public domain. Next a description is given of the recently developed PSP model for MOSFETS, which is based on the SP model (Penn State University) and MOS Model 11 (Philips/NXP). It has been chosen to be the new industry standard by the Compact Modeling Council, succeeding the previous industry standards BSIM3 and BSIM4. Finally, the SiMKit, a simulator-independent compact transistor model library, containing the most recent versions of the Philips/NXP transistor models, such as the Mextram and PSP models, is described. This library is available and can be downloaded from the NXP website, both for Spectre and ADS circuit simulation.
In the next section 2.4 a description is given (as an example of a component description) of the compact modeling of spiral inductors.
In section 2.5, a specification of the Touchstone® file format (also known as an SnP file) is given. Touchstone files have been accepted as a de-facto standard for the transfer of frequency dependent n-port network data. This section, based upon information from Agilent Corporation (the originator of Touchstone), is a formal specification of the Touchstone file format.
In the next session 2.6 a compact (binary, sparse matrix) format, able to describe a linear, time invariant system (e.g. as output of a field solver) is discussed. In the exchange of matrix files between the different partners in nanoCOPS, 3 matrix formats will be made available. These formats are the compressed-row storage format CRS, coordinate storage (CS) and matrix market format (MMF).
In the last section **Error! Reference source not found.** of chapter 2, specifications are given for the use of statistical data, describing the two cases of input data occurring in the project, random variables for scalar parameters (e.g. capacitances, inductances,

resistances, etc.) or random fields for spatial effects (e.g. material parameters, geometries, etc.).
Finally, chapter 3 deals with coding styles, compiler choices, library development, linking third party software, tool integration and GUI design. This chapter can be used to check if your code is compliant to integrate with the MAGWEL software environment.

2 Data format specification

2.1 *Circuits*

The output netlist format must be specified. The most important simulators in the RF field and therefore the minimum output formats are Spectre, ADS(Gemini) and Spice.
At the beginning of the 90's there were two main categories of circuit simulation methods: time domain methods with Spice as representative software and frequency domain methods, most known package being MICROWAVE HARMONICA [1]. It has been established that harmonic balance was the most efficient for weakly nonlinear circuits with signals having relatively few harmonic components, and the time domain simulation was the best choice for circuits with strong non-linearities and/or signals with abrupt transitions. Several methods for speeding up the steady state computation by time domain analysis have been developed, such as shooting by linear extrapolation [2], and shooting by exponential extrapolation [3].
The advent of huge expansion of mobile and digital communications came with new simulation requirements for strongly nonlinear circuits with frequency modulated, pulse modulated, and digitally modulated signals. These circuits cannot be simulated with the above mentioned methods using a reasonable amount of CPU time in order to obtain the accuracy required by the designers. Therefore, some new simulation methods and software packages have been developed.

2.1.1 Simulation Methods

A typical RF-IC application is strongly nonlinear and has carrier frequencies in the GHz-range with modulating signals in the KHz-range. Three new approaches implemented in commercial software seem to fit, at least partly, the requirements of RF-IC analysis: transient envelope or envelope following method [1],[4], Fourier envelope or circuit envelope [5], [6] and harmonic balance with Krylov subspace solver [1].

2.1.1.1 Envelope Following Method

It reduces the simulation time without compromising accuracy by exploiting the property that the behavior of the circuits in a given high frequency clock cycle is similar, but not identical, to the behavior in the preceding and following cycles. In particular, the "envelope" of the high-frequency clock can be followed by accurately computing the circuit behavior over occasional cycles, which accurately capture the fast transient behavior. The slow varying modulation is accurately followed by a piecewise polynomial. As a result, the spectrum of the circuit response can be obtained by combining the piecewise polynomial and the integration of occasional clock cycles.
The analysis can be applied to efficiently and accurately analyze modulation signals in large communication circuits. Important applications include prediction of spectral re-

growth of amplifiers or mixers, design of feedback loops such phase-locked loops or AGC loops, and transient behavior of switching power converters or switched capacitor filters.
The approach in [1] has been implemented in Spectre RF [7].

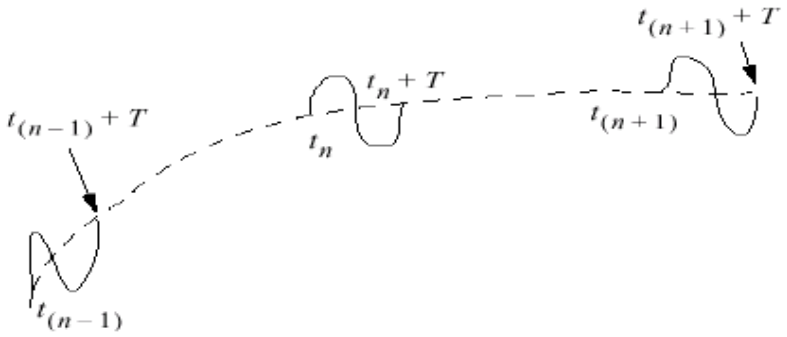

Fig 4

2.1.1.2 Fourier Envelope Method

The Envelope simulator combines features of time- and frequency-domain representation, offering a fast and complete analysis of complex signals such as digitally modulated RF signals. This simulator permits input waveforms to be represented in the frequency domain as RF carriers, with modulation "envelopes" that are represented in the time domain.

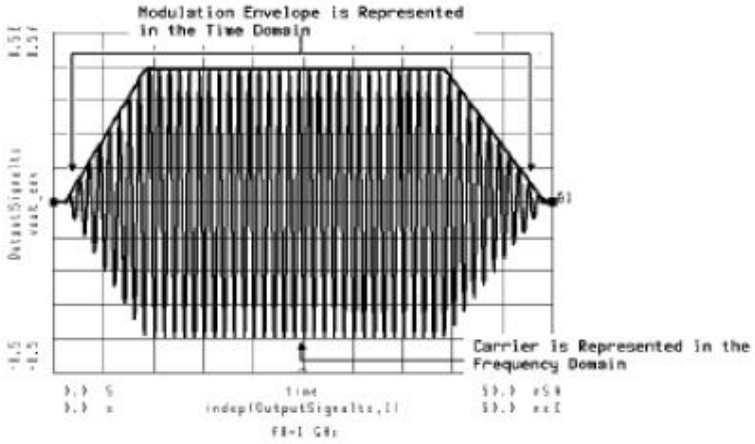

Fig 5

Each node voltage is represented by a discrete spectrum having time-varying Fourier coefficients.
The set of spectral frequencies is user-defined; the amplitude and phase at each spectral frequency can vary with time, so the signal representing the harmonic is no longer limited to a constant, as it is with harmonic balance. Each spectral frequency can be thought of as the center frequency of a spectrum; the width of each spectrum is 0.5/*Time step*. In most

cases the bandwidth of the modulation signal is much smaller than the lowest user-defined spectral frequency (which corresponds to the "carrier" frequency). The band-limited signal within each spectrum can contain periodic, transient, or random tones. This spectrum may represent transient signals with continuous spectra, such as a digital modulation envelope over an RF carrier, or periodic signals with discrete spectral lines, such as the two RF tones required for intermodulation distortion analysis.
As the applications for wireless communications continue to grow, it is no longer possible to satisfy all modeling needs with standard, pre-configured models. Users need methods to define their own nonlinear models in either the time or the frequency domain. The frequency-domain-defined device (FDD) has been developed to allow a user to describe current and voltage spectral values directly, in terms of the algebraic relationships of other voltage and current spectral values.
Another trend in digital communication systems is the issue of timing, since it is increasingly common to encounter subsystems that behave in ways that cannot be modeled as time-invariant. Clocked systems, sampled systems, TDMA pulsed systems, and digitally controlled systems are all becoming more common, even in the RF and microwave area, and behavioral models must be able to include these effects. So, in addition to its frequency-domain modeling attributes, the FDD also allows the modeler to define trigger events, to sample voltages and currents at trigger events, and to generate outputs that are arbitrary functions of either the time of the trigger or of the complex spectral voltage and current values at these trigger events.
This approach is implemented in ADS.

2.1.1.3 Harmonic Balance with Krylov subspace solver

The harmonic balance method is based on the assumption that for a given sinusoidal excitation there exists a steady-state solution that can be approximated to satisfactory accuracy by means of a finite Fourier series. Consequently, the circuit node voltages take on a set of amplitudes and phases for all frequency components. The currents flowing from nodes into linear elements, including all distributed elements, are calculated by means of a straightforward frequency-domain linear analysis. Currents from nodes into nonlinear elements are calculated in the time-domain. Generalized Fourier analysis is used to transform from the time-domain to the frequency-domain. A frequency-domain representation of all currents flowing away from all nodes is available. According to Kirchhoff's Current Law (KCL), these currents should sum to zero at all nodes. The probability of obtaining this result on the first iteration is extremely small. Therefore, an error function is formulated by calculating the sum of currents at all nodes. This error function is a measure of the amount by which KCL is violated and is used to adjust the voltage amplitudes and phases. If the method converges (that is, if the error function is driven to a given small value), then the resulting voltage amplitudes and phases approximate the steady-state solution.
Many harmonic balance simulators rely on the Newton-Raphson technique to solve the nonlinear systems of algebraic equations that arise in large-signal frequency-domain circuit simulation problems. Each Newton-Raphson iteration requires an inversion of the Jacobian matrix associated with the nonlinear system of equations. When the matrix is factored by direct methods, memory requirements climb as $O(H^2)$, where H is the number of harmonics. An alternative approach to solving the linear system of equations

associated with the Jacobian is to use a Krylov subspace iterative method such as GMRES (generalized minimum residual). This method does not require the explicit storage of the Jacobian matrix J. It turns out that the information needed to carry out such an operation can be stored in O(H) memory, not in O(H^2), in the context of harmonic balance. Thus, Krylov subspace solvers offer substantial savings in memory requirements for large harmonic-balance problems. Similar arguments show that even larger increases in computational speed can be obtained.
This approach is implemented in ADS [8]. In certain cases the results are better than those obtained with Fourier envelope method.

2.1.2 SPICE format

Although several different flavors/dialects of Spice exist, we will standardize on the HSPICE format (version 2008.03). For a detailed description of this format, see [9].

SPICE has built-in models for the semiconductor devices, and the user need specify only the pertinent model parameter values. The model for the BJT is based on the integral-charge model of Gummel and Poon; however, if the Gummel- Poon parameters are not specified, the model reduces to the simpler Ebers-Moll model. In either case, charge-storage effects, ohmic resistances, and a current-dependent output conductance may be included. The diode model can be used for either junction diodes or Schottky barrier diodes. The JFET model is based on the FET model of Shichman and Hodges. Six MOSFET models are implemented.

2.1.2.1 HSPICE Files

Suffixes:
HSPICE has many files that it can take as input or that it can produce. These files must contain these suffixes:

- HSPICE Input
  - *input netlist: .sp*
  - design configuration: .cfg
  - initialization: hspice.ini

  Note: The italicized file you must have.
- HSPICE Output
  - *run status: .st0*
  - *output listing: .lis*
  - graph data, transient: .tr# (e.g. .tr0)
  - graph data, dc: .sw# (e.g. .sw2)
  - graph data, ac: .ac# (e.g. .ac1)
  - measure output: .m*# (e.g. .mt0)

  Note: The italicized files are those that you will always have.

File Descriptions:

Netlist file (.sp)
The netlist file is the input file. This file contains the circuit description and all options and analysis you wish the simulator to deal with. The structure of the file looks like this:

title	-Implicit first line; becomes netlist file title
.options	-set conditions for simulation
ANALYSIS and TEMP	-statements to sweep variables
.print/.plot/Analysis	-set print, plot, and analysis variables
.initial conditions	-input state of system
sources	-stimulus
netlist	-circuit descriptions
.model libraries	-.lib and .inc
.end	-terminates the simulation

Note: This is just a suggested structure. Though the title always comes first and .end always comes last, the rest can come in any order in between. Another popular structure that is more implicit is to put the analysis statements after the circuit descriptions.

An example Netlist structure:

```
This is a netlist                    $must have a title line
.options post acct opts node         $HPSICE options

l6      6       16      .05          $inductor
c6      16      0       .05          $capacitor
r16     16      0       40           $resistor
c4      4       14      .1           $another capacitor
l5      data    15      1            $another inductor
c5      15      0       .2           $another capacitor

v4      4       0       dc 0 ac 0 0 pulse 0 1 0 .15 .15 .4 2   $voltage source
vdata   data    0       sin(1.0 1.0 1.0 0.0 1.0)               $voltage source
v6      6       0       exp(1 0.1 .02 .6 .2)                   $voltage source

.model ...                           $models and subckts

.tran   .1 5                         $transient analysis
.print  v(6) i(r16)                  $print statement
.plot   v(4) v(14) v(data)           $plot statement

.end                                 $Must have an end
```

Another example of a bjt amplifier bias by a voltage divider is given here:

bjtamp.sp

BJT Example

```
.OPTION      $ The options are listed below:
+ ingold=1   $ Use fixed-point and exponential form on output
+ numdgt=4   $ print 4 digits

.INC /nfs/guille/analog1/m/moon/ece323/npn322.inc


Vcc    vcc    0      =12.0
q1     c      b      e       npn322
Rb1    vcc    b      2.1k
Rb2    b      0      300
Rc     vcc    c      100
Re1    e      e1     9.5
Re2    e1     0      6.5
Cout   e1     0      63u
Cin    vi     b      1u
Vin    vi     0      AC=1 SIN 0 amp 100K 0 0 180

.OP

.AC DEC 10 1 1G
.NET V(c) Vin
.PRINT AC Vdb(c)

.MEASURE AC A0 FIND VR(c) AT=100K                $ Measure the gain at 10kHz.
.MEASURE AC AdB FIND VdB(c) AT=100K              $ Measure the gain at 10kHz.
.MEASURE AC mzi FIND PAR('v(vi)/i(vin)') AT=100K $ Measure the input imped.
.MEASURE AC mzo FIND ZOUT(M) AT=100K             $ Measure the out. imped.
@10kHz.
.MEASURE AC fl WHEN VM(c)='.707*ABS(A0)'         $ Find the lower 3dB
frequency.
.MEASURE AC fu WHEN VM(c)='.707*ABS(A0)' TD=1e5 $ Find the upper 3dB
frequency.

.TRAN .1u 20u SWEEP amp POINTS 2 110m 500m
.PRINT TRAN V(c)

.END
```

Design configuration (.cfg)
Configuration files are used by HSPICE, GSI and HSPLOT to describe the available terminals and hardcopy devices.

Initialization (hspice.ini)
The initialization file deals primarily with setting up HSPICE itself and has no need to be modified by users.

Run status (.st0)
The run status file keeps track of all that HSPICE performs on a netlist file. Since no results are stored here, just execution steps, this file is not very useful and can be deleted without any great loss to a user. The does sometimes offer small benefit for debugging purposes as it can show which steps were not performed by HSPICE.

Output listing (.lis)
This is one of the most important files in HSPICE as this file lists all results obtained from the simulation. This file contains (in order of listing in the file):
HSPICE licensing information
Listing of the circuit
Results from the analysis of the circuit (.op, .print, .plot, .measure, .ac, and .tran in order of their appearance in the netlist file)

Graph data files (.tr#, .sw#, .ac#)
Graph data files are created by .OPTION POST command and contain the data to be graphed by HSPLOT, GSI, or HSPICE. Basically these files retain the data to be graphed from the .lis file in a different graphing format.

Measure output (.m*#)
The measure output file hold the result from .MEASURE commands. These files are not very useful either as the data is hard to read and already exists in a nicely formatted way in the .lis file.

2.1.2.2 Netlist Notation:

Naming conventions:

Every node and element within the HSPICE netlist must have its own unique name. These names can exist of the following:

- Node and Element Identification
  - Either Names or Numbers (e.g. n1,33,in1,100)
  - Numbers: 1 to 99999999 (99 million)
  - Nodes with number followed by letter are all the same (e.g. 1a=1b)
  - 0 is ALWAYS ground
  - Capitals do not matter (e.g. node1=NODE1=Node1=nOdE1)

- Allowable Characters and Conventions (if nodes have names and not just numbers)
    - Begin with letter or /
    - Max of 16 characters (after 16 ignored)
    - May contain: +-*/:;$#.[]!<>_%
    - May not contain: (),=<space>
    - Ground may be either 0, gnd, or !gnd
- Every node must have at least 2 connections.

HSPICE units:

HSPICE will automatically assign ohms to resistors, Farads to capacitors, and Henries to inductors. However, the units can be manually assigned by:

R - ohm
C - Farad
L - Henry

The scaling for the all units in HSPICE is done by:

F=1e-15
P=1e-12
N=1e-9
U=1e-6
M=1e-3
K=1e3
MEG=X=1e6
G=1e9
T=1e12

Note: Capitals do not matter (e.g. F=f and M=m), therefore many people have problems with M and MEG. Please be careful with these two units!

Global Variables:

- Syntax
    - .GLOBAL node1 node2 node3 .....Globally defined nodes
    - .GLOBAL VBIAS VCC .....Globally defined sources
- Usage
    - When subcircuits are included in the data file.
    - Assigns common node name to subcircuit nodes
    - Power supply connection of all subcircuits often done this way
        - .GLOBAL VCC
        - Connects all nodes named VCC (all circuits have a common node)

Essentially, globals are used most often for subcircuits, though they are great for sources. This way all power sources do not have to be separately defined.

Note: The .GLOBAL statement has a somewhat different meaning in TITAN than in HSPICE, where .GLOBAL defines global nodes and connects local nodes to these global nodes. But with .GLOBAL TITAN only checks that there is a connection of these nodes through subcircuit pins to the next hierarchy level. For this reason it would be good if the usage of the .GLOBAL statement in benchmark netlists is avoided, if possible.

Misc. Syntax:
In general, HSPICE notation is very simple. All statements beginning with a <.> are commands (e.g. .OPTION, .PRINT, .AC, etc.) while those that don't are treated as elements (e.g. r51, cc, lload, etc.). There are, however, a few special commands:

- <*> - indicates a full line comment
- <$> - indicates an end of line comment
- <+> - indicates a continuation of the previous line

Examples of these can be found in the BJT amplifier example bjtamp.sp above. Here the .OPTION command is extended for four lines and the end of line comment is used after the .MEASURE commands. Notice too that by using tabs for the elements and their nodes, it makes the netlist much easier to read. It is not required that tabs be used, but it is very helpful.

Algebraic Syntax:
HSPICE also has the added benefit of being able to do some algebraic manipulation of some variables. The variables can be user defined or just the voltages and currents at nodes (e.g. v(1), I(q2), etc.). Also the algebra must be enclosed within single quotes (see examples below). The algebraic functions used by HSPICE in addition to (+,-,*,/) are as follows:

- sin(x)
- cos(x)
- tan(x)
- atan(x)
- exp(x)
- log(x) - this function will use the abs value of x and then apply the sign of x to the results.
- pwr(x,y) - x to power y, see also log.
- sinh(x)
- cosh(x)
- tanh(x)
- sqrt(x) - see also log
- db(x)
- log10(x) - see also log
- abs(x)
- min(x,x)
- max(x,x)

Some examples of using algebraic functions in HSPICE are:

- Parameterization (variable declaration)

```
.PARAM x='y+3'
```

- In elements

```
r1      1       0       r='abs(v(1)/i(m1))+10'
```

- In .MEASURE statements

```
.MEAS vmax MAX V(1)
.MEAS imax MAX I(q2)
.MEAS ivmax PARAM='vmax*imax'
```

Topology Rules:
It is important to know when constructing the circuit description that HSPICE will not allow certain topologies. It will not allow:

- Every node must have a DC path to ground (i.e. all circuits must a have at least one ground not in series with a capacitor).
- No dangling nodes (i.e. all nodes must have at least two connections).
- No voltage loops (i.e. no voltage sources in parallel with no other elements).
- No ideal voltage source in closed inductor loop.
- No stacked current sources (i.e. no current sources in series).
- No ideal current source in closed capacitor loop.

2.1.3 Spectre RF format

For a detailed description of this format, see [10] and [11]. We will standardize on version 6.0 of Spectre.

Capacitor – Capacitance with
• temperature dependence
• width and length and pin to pin distance for use with a model
• quality factor with analytical frequency dependence (several choices)

Inductor – Inductance with
• temperature dependence
• polynomial current controlled nonlinear inductor

Mutual inductor
• either coupling coefficient or mutual inductance is specified

Resistor – Resistance with
• temperature dependence
• polynomial voltage controlled nonlinear resistor
• width and length for use with a model
• diffusion resistor model incorporated

Circuit reduced order model (see also [11])
An m-port with port voltages in u, port currents in i and internal state variables in x described by the linear equations:
$x'=Ax+Bu$
$y=Cx+Du$

In WP3 there is a more general appearance realized of the reduced-order modeling.
Let **X** be the vector of state-space variables and let **u** be the port voltages (input signals) and **y** the port currents (output signal). Then the following structure is obtained:

$\boldsymbol{A_2 X'' + A_1 X' + A_0 X = B u}$
$\boldsymbol{y = C_0 X + C_1 X' + D u}$

This convention would imply that the first-order system is given as

$-Ax + 1 * x' = Bu$
$y=Cx+Du$

$\boldsymbol{A_0 = 1}$ and $\boldsymbol{A_1 = -A}$ and $\boldsymbol{A_2 = 0}$

Magnetic core with hysteresis
• frequency and temperature effects are taken into account

Microstrip line
• based on the equations of Hammerstad and Jensen
• dispersion equations are those of Kirschning and Jansen

Transmission line

2.1.4 ADS format

For a detailed description of this format, see [12]. We will standardize on version 2008A of ADS.

Capacitor – Capacitance with
• temperature dependence
• width and length and pin to pin distance for use with a model
• quality factor with analytical frequency dependence (several choices)

Provided models
• Dielectric Laboratory Di-cap capacitor (lossy parallel transmission line behavior)
•Dielectric Laboratory Multi-Layer Chip Capacitor (open ended transmission line behavior)

Inductor – Inductance with
• temperature dependence
• 4 loss modes
• quality factor with analytical frequency dependence
• width of pad and spacing and pin to pin distance for use with a model
Remark: for time domain analysis the frequency domain analytical model is used.

Mutual inductor
• either coupling coefficient or mutual inductance is specified

Resistor – Resistance with
• temperature dependence
• width and length and pin to pin distance for use with a model
Remark: analytical resistance variation w.r.t. time may be used in time domain analysis.
Series and parallel groups RC, RL, LC, RLC

Linear Data File Components
• one-port, two-port, up to 99 port files
• S, Y, or Z parameter specified as functions of frequency
• Frequency dependence is obtained using linear, cubic spline, or cubic interpolation
• for time domain analysis the frequency domain s- parameters are used

Equation Based Linear Components
• 2-port user defined linear chain
• 2-port user defined linear hybrid
• 1 to 6 port S parameter equation based
• 1 to 6 port Y parameter equation based
• 1 to 6 port Z parameter equation based

2.1.5 Verilog-A for compact circuit modeling

Verilog-A might also be a potential candidate for compact model development. If the need arises to make additions to compact models, it might be easier to do so in the verilog-A version, which might also be more portable. Most circuit simulators accept it and e.g. the standard PSP device model has a verilog-A description as well (see 2.2.2.3).

2.2 *Models*

2.2.1 A compact model for bipolar transistors: Mextram

The Mextram model gives an excellent description of vertical bipolar transistors in all kinds of processes, amongst which are modern SiGe processes and robust HV processes. It is very efficient in modeling the lowly doped collector epilayer of a bipolar transistor where effects like velocity saturation, base widening, Kirk effect, and impact ionization play a role. Effects due to having Germanium in the base are also modeled. Furthermore, it contains a full description of the extrinsic regions of a transistor, including substrate current and capacitance. Mextram has formulations for temperature scaling and is easily scalable over geometry. Mextram, level 504, includes full self-heating, in contrast to having self-heating handled by the circuit-simulator. Self-heating is completely implemented in the source code.

2.2.1.1 General

Mextram is an advanced compact model for the description of bipolar transistors. It contains many features that the widely-used Spice-Gummel-Poon model lacks. Mextram can be used for advanced processes like double-poly or even SiGe transistors, for high-voltage power devices, and even for uncommon situations like lateral NPN-transistors in LDMOS technology. It provides an excellent description of the forward and reverse modes of operation for both digital and analogue circuits.

Mextram has proved itself during intensive use within the NXP design community, because of its combination of good effects description and parameter set with good convergence. Mextram models a great number of effects enabling realistic transistor predictions to be made. The full documentation and source code of the Mextram Bipolar Transistor Model is available on this web site.
Good IC design is, of course, dependent on good simulation, which in turn is dependent on the availability of good physical device models. The development and continuous refinement of Mextram by NXP Research, using feedback from practical design tasks, make it an excellent and robust model that has proved its accuracy, good convergence and wide applicability. Mextram is suitable for use in both circuit design and process technology, as well as CAD tool development.

2.2.1.2 History

Mextram has been developed by Philips Research. It was first released for use within Philips in 1985. Mextram level 503 was released into the public domain by Koninklijke Philips Electronics N.V. in 1994, has had a small model update in 1995, and has been unchanged ever since. Mextram level 504 has been developed as an update to level 503 for several reasons, the main ones being the need for even better description of transistor characteristics and the need for an easier parameter extraction.

2.2.1.3 Latest update

The improved description of transistor characteristics of Mextram 504 compared to Mextram 503 is achieved by changing some of the formulations of the model. For instance, Mextram 504 contains the Early voltages as separate parameters, whereas in Mextram 503 they were calculated from other parameters. This is needed for the description of SiGe processes and improves the parameter extraction (and hence the description), in the case of pure Si transistors. An even more important improvement is the description of the epilayer. Although the physical description has not changed, the order in which some of the equations are used to get compact model formulations has been modified. The result is much smoother behavior of the model characteristics, i.e. the model formulations are now such that the first and higher-order derivatives are better. This is important for the output-characteristics and cut-off frequency, but also for (low-frequency) third order harmonic distortion. To achieve the same smoothness, some other formulations, like that of the depletion capacitances, have been changed.
In Mextram almost all of the parameters have a physical meaning. This has been used in Mextram 503 to relate different parts of the model to each other, using a limited number of parameters. Although this is the most physical way to go, it makes it difficult to do parameter extraction, since some parameters have an influence on more than one physical effect. In Mextram 504 as much of this interdependence as possible has been removed, without losing the physical basis of the model. To do this some extra parameters have been added. At the same time some parameters of Mextram 503, that were introduced long ago but which had a limited influence on the characteristics and were therefore difficult to extract, have been removed. The complete Mextram model has been thoroughly revised. The documentation of the model has also been extended.

2.2.1.4 Unsurpassed modeling

The sophisticated modeling of the avalanche multiplication, the epitaxial collector layer resistance and charge storage of advanced high frequency transistors, is unsurpassed. When compared to existing Spice models, this results in improved modeling, of all relevant transistor characteristics, such as: current gain, output conductance/conductivity, cut-off frequency, distributed high-frequency effects, noise figures and temperature behavior.
Mextram does not contain extensive geometrical or process scaling rules (only a multiplication factor to put transistors in parallel). The model is well scalable, however, especially since it contains descriptions of the various intrinsic and extrinsic regions of the transistor. By exploiting the solid physics at the basis of the Mextram model, the statistical analysis of the spread in the processing and geometrical layout of the transistor is brought to a very high level of accuracy and reliability. This also enables the calculation of predictive parameter sets, derived from the process and layout data of the transistor.
Some parts of the model are optional and can be switched on or off by setting flags. These are

- the extended modeling of reverse behavior,
- the distributed high-frequency effects in the base, and

- the increase of the avalanche current when the electron density in the epilayer exceeds the doping level (including snap-back).

Besides the NPN transistor, a PNP model description is available also. Both three-terminal devices (discrete transistors) and four-terminal devices (IC-processes which also have a substrate), can be described. An extra terminal is present when including self-heating. This makes it possible to embed the transistor in a heating network and perform combined electro-thermal simulations.

2.2.1.5 Effects modeled by Mextram

As the following table shows, Mextram enables a number of effects to be modeled that older models do not address, including Early voltage bias dependency, explicit modeling of inactive regions, current crowding and conductivity modulation for base resistance and quasi-saturation.

2.2.1.6 Effects modeled by Ebers-Moll, Spice-Gummel-Poon, and Mextram

EM	SGP	MXT	Effects
		X	Bias-dependent Early-voltage
		X	Explicit modeling of inactive regions
		X	Substrate effects and parasitic PNP
		X	Current crowding (both DC and AC)
		X	Quasi-saturation and Kirk effect
		X	Velocity saturation in the collector epilayer
		X	Weak avalanche, including optional snap-back effect
		X	Self-heating
		X	Early effect in case of a graded Ge-profile
		X	Neutral-base recombination
	X	X	High-injection effects in the base
	X	X	Low-level non-ideal base currents
	X	X	Split base-collector depletion capacitance
	X	X	Excess phase shift
X	X	X	Thermal noise, shot noise and flicker noise
X	X	X	Temperature scaling

EM	SGP	MXT	Effects
X	X	X	Charge storage
X	X	X	Series resistances

2.2.1.7 Model definition of Mextram

The model definition of Mextram can be found in [13].

2.2.1.8 Source Code & Library

The source code can also be found on [13].
The models are included in a dynamically loaded library called SiMKit.

2.2.1.9 Additional reports

Further information on Mextram is available in the following additional reports, related publications and presentations can also be found on [13].

2.2.2 The MOS Model PSP

The PSP model has been developed jointly by Philips Research and the group of Prof. Gildenblat at Penn State University. The development goals have been:

- suitable for digital, analog, and RF;
- suitable for modern and future bulk CMOS technologies;
- physics-based;
- combining the best features of SP model (Penn State University) and MOS Model 11 (Philips/NXP);
- number of parameters and simulation time comparable to MOS Model 11;
- simple parameter extraction.

2.2.2.1 Model definition of MOS Model PSP

The model definition and additional information can be found on
http://www.nxp.com/models/mos_models/psp

The PSP model is a new compact MOSFET model, which has been jointly developed by Philips Research and Penn State University. It is a surface-potential based MOS Model, containing all relevant physical effects (mobility reduction, velocity saturation, DIBL, gate current, lateral doping gradient effects, STI stress, etc.) necessary to model present-day and upcoming deep-submicron bulk CMOS technologies. Unlike previous Philips/NXP MOS models, the source/drain junction model, c.q. the JUNCAP2 model, is an integrated part of the PSP model.

2.2.2.2 Non-quasistatic RF model

Introduction
For high-frequency modeling and fast transient simulations, a special version of the PSP model is available, which enables the simulation of non-quasistatic (NQS) effects, and includes several parasitic resistances.

NQS-effects
In the PSP-NQS model, NQS-effects are introduced by applying the one dimensional current continuity equation ($\partial I / \partial y \alpha - \partial \rho / \partial t$) to the channel. A full numerical solution to this equation is too inefficient for compact modeling, therefore an approximate technique is used. The channel is partitioned into *N+1* sections of equal length by assigning equidistant *collocation points*. The charge along the channel is then approximated by a cubic spline through these collocation points, assuring that both the charge and its first and second spatial derivatives are continuous along the channel.
Within this approximation, the current continuity equation reduces to a system of *N* coupled first order ordinary differential equations, from which the channel charge at each collocation point can be found:

$$\frac{dQ_1}{dt} = f_1(Q_1,...,Q_N)$$
$$\vdots \qquad \vdots$$
$$\frac{dQ_N}{dt} = f_N(Q_1,...,Q_N)$$

Here, Q_i is the charge in the i-th collocation point and f_i are known functions, which contain the *complete* PSP-charge model. These equations are implemented by the definition of appropriate subcircuits and solved by the circuit simulator. Finally, the four terminal charges are calculated from the channel charges, using the Ward-Dutton partitioning scheme for the source and drain charges.
A more detailed description of the spline collocation method as it is implemented in the PSP-NQS model can be found in [14]. Note however that the functions fi above contain the full PSP-charge model (including, e.g., velocity saturation and poly-depletion) [15].

Parasitics circuit
To facilitate RF-simulations, the PSP-NQS model contains a small network of parasitic elements: agate resistance and four bulk resistances. The junction diodes are no longer directly connected to the bulk terminal of the intrinsic MOS-transistor. The PSP-NQS model has a few additional parameters.

2.2.2.3 Source Code & Library

The source code (i.e., C-code and Verilog-A code) of MOS Model PSP is available on [13]. The files containing the global and local model code of level 100 are

"device_psp1000.c" and "device_psp100e.c" respectively. If you use the included solver, please make sure that you compile on a system that supports Fortran 77.
The source code for MOS Model PSP has been automatically generated from a Verilog-A description of the model. The models are included in a dynamically loaded library called SiMKit.

2.2.2.4 Additional reports

Further information on PSP is available in the following additional reports and related publications can also be found on [13] and in [16].

2.2.2.5 Physical effects

The PSP model is a symmetrical, surface-potential-based model, giving an accurate physical description of the transition from weak to strong inversion. The PSP model includes an accurate description of all physical effects important for modern and future CMOS technologies, such as:

- mobility reduction
- bias-dependent series resistance
- velocity saturation
- conductance effects (CLM, DIBL, etc.)
- lateral doping gradient effect
- mechanical stress related to STI
- gate leakage current
- gate-induced drain leakage
- gate depletion
- quantum-mechanical effects bias-dependent overlap capacitances

In addition, it gives an accurate description of charges and currents and their first-order derivatives (transconductance, conductance, capacitances), but also of their higher-order derivatives. In other words, it gives an accurate description of MOSFET distortion behavior, and as such the PSP model is suitable for digital, analog as well as RF circuit design.

2.2.2.6 SiMKit library

SiMKit is a simulator-independent compact transistor model library. Simulator-specific connections are handled through so-called adapters that provide the right interfacing to Pstar, Spectre and ADS. The SiMKit library contains the most recent versions of the NXP transistor models, such as the Mextram and PSP models.

The NXP SiMKit is supported for LINUX, HPUX 11, SUNOS 5.7 and systems. There is one set-up file available, called simkit_2.2.tar With this set-up file you can compile your own SiMKit library. SiMKit can be downloaded from [13].

SiMKit is related to the following circuit simulators used within NXP:

- Pstar, the circuit simulator from NXP
- Spectre, the circuit simulator from Cadence
- ADS, the circuit simulator from Agilent.

2.2.2.7 Verilog-A

For some NXP compact models the Verilog-A code is available. The functionality of this Verilog-A code is the same as that of the SiMKit C-code.
The Verilog-A code is primarily intended as source for C-code generation and can be downloaded as well.

2.3 *Layout*

2.3.1 Input file formats

The input format for the end-user will be either GDS II or XML, whereas EM simulation tools can use their own native data formats defined in XML. However, the GDS II information is not sufficient to create the full geometry of the structure. Indeed, more information on the process and port information is needed in the form of a technology file.

2.3.1.1 GDS II

The GDSII stream data format is the standard circuit description file format. It contains a hierarchy of cells (structures) that contain geometry or other cell references.
The goal is to be able to read a cell, which can be a small structure, from a GDSII stream, add some extra information (ports, layer materials,...) and use it as input for the solver.
For more info on the GDSII stream format see [17].

2.3.1.2 Process Layer Description

Besides the GDS II data, we need the information regarding the fabrication process of the device (technology data). The Process Layer Description (PLD) gives the physical information on the different metal and dielectric layers and vias. An example of such a PLD as proposed by austriamicrosystems, which was used in the Codestar project, is given in Appendix E. In Appendix F a description is given of the Agilent RFDE/ Momentum .tch technology file format.

2.3.1.3 Port Information

To be complete the field analysis problem should include the port geometric description and their excitations. In the case of large integrated circuits the port should be generated in an automatic manner. The port description file can have the simple format described in [18].

2.3.2 XML format

This format has the advantage that it is standardized and it will become the main format for data transfer in the near future. The definition of a specific XML input format is done in XML Schema format, of which the present version (by MAGWEL, and described in Appendix D) is an extension, based on the work done in the Codestar project [19]. This meta-data defines the structure of the XML data and its datatypes. XML Schema gives a detailed definition of the syntax of the system. It defines the tags that can be used and how they can be embedded in other tags. It also defines the datatype of the different data items.
Using this approach is very advantageous for several reasons:

♦ If your data is structured as XML, and there is a XML Schema, then you can hand the data-checking task off to a schema validator. A lot of code usually deals with the data-checking task, and this can now be taken over by the schema validator.
♦ If organizations agree to structure their XML documents in conformance with an XML Schema, it acts as a contract between the organizations.
♦ The XML format definition (XML Schema) is public. The different input files however, will have a copyright tag in order to take care of property issues.
♦ Expanding and/or upgrading the input file format definition is very easy.
♦ Parsers are available to construct a data structure in most of the popular languages.
The input for a field solver can be divided in four different parts. The <geometry> part describes the geometrical information of the structure. All the physics related parameters that have nothing to do with the solution method are bundled in <physics>, while the solver specific information is give in the <solver> section. A <copyright> section contains the status of the file regarding confidentiality.

2.3.2.1 <geometry>

First there is the geometrical definition of the problem. For the applications we have in mind, in first instance, the solver uses a so-called layer-based geometry. This is closely related to the actual processing and is represented in the industrial format GDS II. A box shaped simulation domain is constructed by adding different layers. These layers are perpendicular to the z-axis.

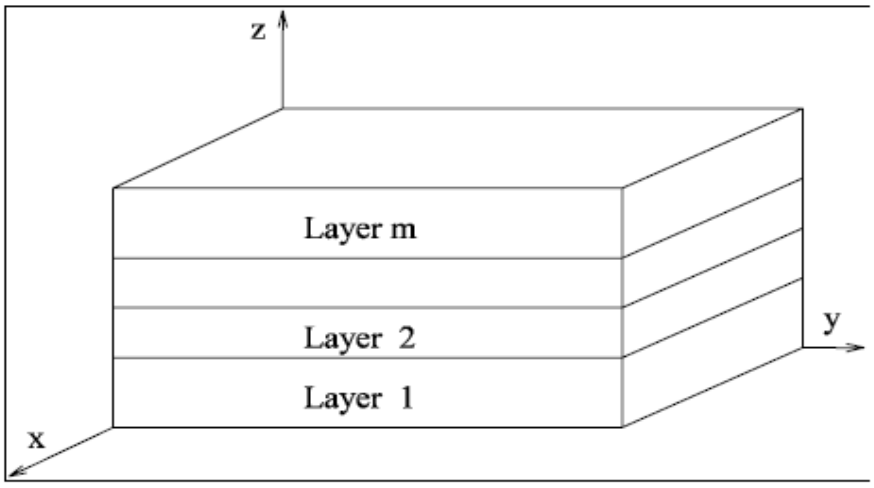

Fig. 1: The structure consists of layers perpendicular to the z-axis.

Each layer can include bricks that have the same height as the layer, but with possibly another material than the layer. Each brick is characterized by two sets of co-ordinates in the x-y plane.

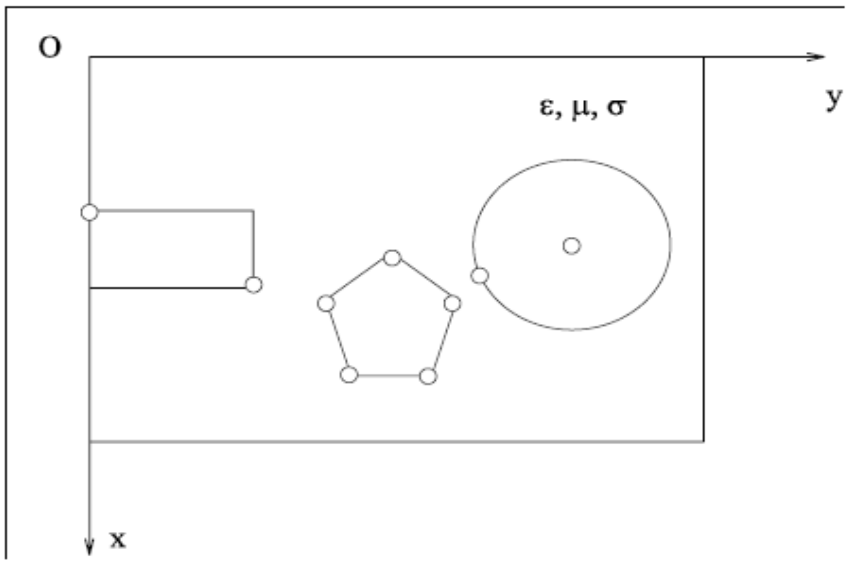

Fig. 2: Top view of a layer that contains a brick (box), a polygon (prism) and a circle (cylinder).

Each layer can contain polygons (actually prisms because they have the same height as the layer and a polygon base) with possibly another material than the layer. This requires an unstructured meshing or a Manhattan approximation. Each polygon is defined by a set of boundary points (Fig 2). Each layer can contain circles (actually cylinders because they have the same height as the layer and a circular base) with possibly another material than the layer. This requires an unstructured meshing or a Manhattan approximation. Each circle is defined by its centre and one of its points (Fig 2).
On the boundary of the computational domain, terminals are defined. These terminals are rectangular and can be located either on top, on the bottom or at the side of the simulation domain (Fig 3).

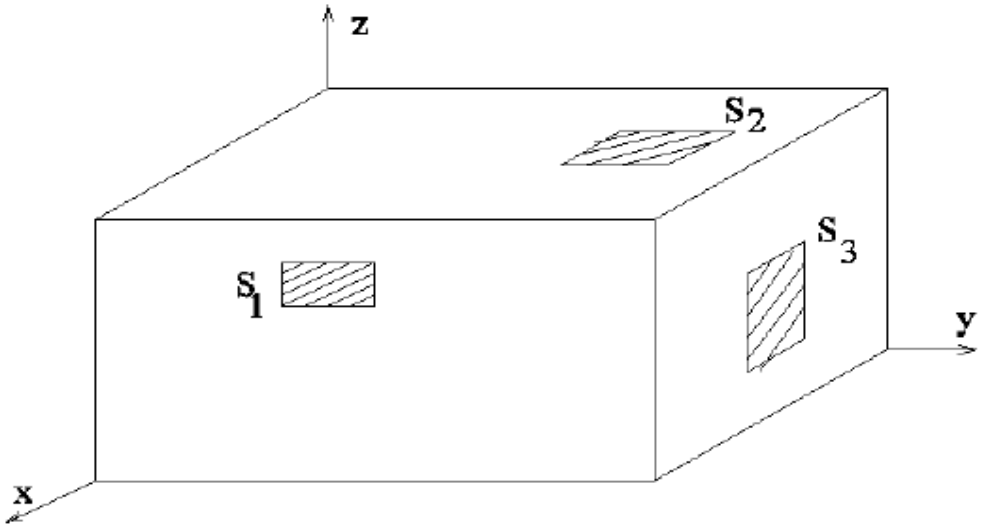

Fig. 3: Contact on the boundary of the simulation domain.

2.3.2.2 <physics>

<materials>

Electromagnetically, the layers, bricks, polygons and circles are defined as LIH (linear, isotropic and homogenous) blocks of material, characterized by their material name. We distinguish three different types of material (conductor, dielectric and semiconductor). For **conductors**, three constants are given: the electric permittivity ε, the magnetic permeability μ, and the electric conductivity σ. For **dielectrics** we do not specify the conductivity (σ=0), while **semiconductor** blocks, need specific information concerning their doping and mobility in order to define their conductance behavior.
Each material is characterized by the:
- Conductivity (σ) for conductors
- Permittivity (ε)
- Permeability (μ)
- loss angle (tan δ)
- intrinsic carrier density for semiconductors
- doping level, doping profile for semiconductors
- Mobility for semiconductors

A method to define also LAH (linear, anisotropic and homogenous) blocks of material is required. These blocks can be defined by:

- Principal values of the conductivity tensor (σx, σy, σz)
- Principal values of the permittivity tensor (εx, εy, εz)

A flexible method (with version control) for extension of the XML definition to take this into account, needs to be defined .

<boundary>

It is well known that, in order to ensure the existence and uniqueness of the field problem solution, the conditions of a uniqueness theorem have to be fulfilled by the input data. A very important element in this context is a proper definition of the boundary conditions. In order to be able to analyze the electromagnetic field in a computational domain it is necessary to place a boundary around the region of interest, in order to limit the region that will be subsequently discretized. The boundary conditions on the edges of the simulation domain are crucial for accurate simulation.

Classical boundary conditions

Among the classical boundary conditions, only the one related to the tangential component of the electric field strength, $Et = 0$, is often needed, to model metallic enclosures or ground planes. For modeling purposes, the perfect magnetic wall $Ht = 0$ may also be necessary.

Absorbing boundary conditions

At RF frequencies, radiation phenomena can occur and, if the real structure is not placed within a metallic box, then a fictitious boundary is necessary. On this fictitious boundary, the "classical" types of boundary conditions ($Et = 0$ or $Ht = 0$) cannot be applied, since they would introduce reflections even if the fictitious boundary would be placed

relatively far away from the region of interest. A special type of artificial boundary conditions must therefore be imposed. If we assume that no electromagnetic energy is entering the simulation domain, these boundary conditions should simulate the unperturbed propagation of the waves towards the exterior of the fictitious boundary and they are known as *absorbing, transparent* or *open* boundary conditions. There are three main types of absorbing boundary conditions (ABC) known in the technical literature: global ABCs, local ABCs and absorbing layers.

1. The *global ABCs* are based on integrals of the field quantities on the whole fictitious boundary. Their numerical discretization leads to relations in which all the field values on the boundary nodes appear, therefore, the localized character and the sparsity of the resulting matrix are lost even for numerical methods of essentially local type, such as the finite element method. Even if the global absorbing boundary conditions have this serious drawback, they offer the advantage of "exactness", i.e. they do not introduce supplementary errors as to the error of the numerical method used for computing field values inside the computational domain. These ABCs are relatively widely used.
2. The *local ABCs* are based on differential equations, often obtained by decomposing the waves incident on the boundary in solutions of the wave equations and imposing conditions such that only solutions representing "outgoing" waves are allowed. They are known in the scientific literature by the name of the author who proposed them. Among the local ABCs, the best known (and widely used) ones are the ones proposed by Engquist and Majda [20], with its numerical discretization done by Mur [21]. Stemming from differential equations, the local ABCs contain spatial derivatives of the field quantities. The highest order of the derivatives present in the ABC formula gives also the so–called *order of the ABC*.
3. The boundary conditions of *absorbing layer* type consist in adding a supplementary lossy layer around the domain of interest, in order to attenuate the outgoing waves to an amplitude level at which reflections on a perfect conducting boundary would not influence the numerical solution too much. Holland [22] seems to be the first author to propose such a boundary condition for electromagnetic field computation. Rappaport & Gürel proposed the "anechoic chamber ABC" [23], inspired from the reflectionless chambers used in practice. Perhaps the most used, to date, absorbing layer boundary condition is the one first introduced by Bérenger in 1994 [24], the Perfectly Matched Layer (PML). Among the numerous papers on this boundary condition, an excellent explanation is given in [25]. At the boundary of the PML layer, typically a perfect conducting boundary is used.

For the semiconductor applications above boundary conditions have limited value because these conditions originate for wave fields analysis and are designed to deal with FDTD simulations. The transient problem as it appears in semiconductor simulations requires the Euler time integration scheme in order to keep the number of time steps feasible. Moreover, the formulation of EM solving needs the scalar potential in order to make the connection to SPICE modeling. Finally, the companying field of the scalar

potential is the vector potential. As a consequence, appropriate boundary conditions for the scalar potential and the vector potential must be given. In the end one must guarantee that all boundary conditions do not violate the gauge invariance. In other words, if a gauge condition is selected, the boundary conditions should be compliant with it. Putting all these demands together we end up with boundary conditions for the scalar potential and the vector potential that are *inspired* by above ABC but are definitely not equivalent.

Let is first consider the surface of the simulation domain. We divide the surface into two classes of surface elements.

1) The surface element is a contact.
2) The surface element is a not a contact.

1 At the contact surface element we can apply Dirichlet boundary conditions for the scalar potential. For the vector potential the *tangential part* of the vector potential is assumed to be a constant. It means that the magnetic field component perpendicular to this surface element is equal to zero. Thus only a tangential magnetic field component is present. One may imagine the contact surface element as part of a port to which a TEM wave arrives from a coax cable. The perpendicular component of the vector potential is determined by the gauge condition.

2 If the surface element is not a contact, then we may apply the Neumann boundary conditions for the scalar potential (as is done in TCAD) and apply Dirichlet boundary conditions for the tangential part of the vector potential. The perpendicular part of the vector potential is again determined by the gauge condition. The combined set of boundary conditions leads to a description of the RF or transient simulation free of radiation loss. In other words: energy can only leave or enter the simulation domain via the ports.
Alternatively, if we want to mimic something like ABC or radiation loss we must use boundary conditions that at least allow for incoming or outgoing radiation through surface elements that are not contact surface elements. This is achieved by having a tangentially varying vector potential over the surface of the simulation domain. Thus the tangential component of the vector potential is submitted to Neumann boundary conditions. Since the radiation loss is carried by EM waves the Maxwell equation in free space must by valid. This gives a constraint for the scalar potential at the same time. Finally the perpendicular component is assumed to be zero (Dirichlet type boundary condition) such that the out/in going radiation is transversal. The latter is an idealization but for practical semiconductor applications and integrated passive element it is appropriate. For example, for an integrated inductor, the currents flow in the metal layers and since the vector potential is an integral over the current density, the vector field induced by the current in the integrated inductor also only will have planar components.

Boundary conditions related to ports/terminals
Special boundary conditions are needed when some of the field sources (excitations) are placed on, or near to the boundary. In RF circuits, two such types of boundary conditions appear to be of interest: Electromagnetic-Circuit-Element boundary conditions, and port-type boundary conditions.

The Electromagnetic Circuit Element (ECE) was proposed in 1966 as a generalization of the circuit element model. The main idea was to determine which set of boundary conditions would guarantee that a (possibly very complicated) spatial domain is "seen" by an external circuit coupled at its terminals as yet another circuit element. In other words, the ECE-type boundary conditions ensure that the device under study is allowed to "communicate" with the exterior world only through its terminals, i.e. it can be characterized from outside through currents and voltages only.
It must be noted that a coupled circuit-field problem is **not** well posed (the solution exists, but it is not unique) if these boundary conditions do not hold! The concept was later extended by numerous authors.
The port-type boundary conditions are needed when the field source is excited by means of 2D modes at a structure's port. Typically, the structure's S-parameters are of interest in these cases.
Special treatment of the port surface is needed, in order to detect the reflected component of the field on the port surface.
A convenient solution for this problem in the time domain is to expand the boundary field into the 2D eigenmodes of the boundary-waveguide. The coefficients in these expansions are the amplitudes of incoming and reflected waves for each mode. A low-order digital filter can model the wave propagation. This technique is explained in [26].
In [27], its application to reduced-order modeling of electromagnetic devices is described. See also [28], [29].

Finally, we would like to emphasize that the above set of boundary conditions are all related to the surface of the simulation domain. However, of equal (if not more) practical relevance are 'internal' boundary conditions. These consist of two-dimensional plane segments on which prescribed voltages are found. Using internal boundary conditions removes the need for artificial extensions to the surface of the simulation domain that then need to be de-embedded again.

<analysis>
We can make a distinction between different solver analysis modes. Either we look at the static solution of the field problem or we look at the time dependent problem. The latter can be subdivided in the time domain of the frequency domain. Also modal analysis is possible. In the system analysis mode, the linear system is saved in the format usable for order reduction.
- Static analysis
- Time domain transient analysis
- Frequency domain analysis
- Modal analysis
- System analysis

<excitation>
The types of excitation of the ports are either voltage excitation, or current excitation. Furthermore, model excitation will be possible.
- Voltage boundary conditions (pulse, step, gauss, harmonic)
- Current boundary conditions (pulse, step, gauss, harmonic)

- Modal excitation

2.3.2.3 <solver>

Extra detailed input is needed for internal solver purposes. These parameters include the specific meshing information, the output information also solver specific parameters.
- In the meshing section common information on the mesh is set up.
- The output information specifies which electromagnetic information of the solver will be written in the different output files.
- Specific information, including the linear system choice, accuracy, solver specific remeshing options, etc. can be provided.

2.3.2.4 <copyright>

This part of the XML file contains the status of the file regarding confidentiality. The file is either public, or nanoCOPS restricted. The latter means that only the nanoCOPS consortium can use the data in the file for work related to the project. Also the owner of the file is specified in this section.

2.3.2.5 Detailed syntax description

For the detailed description of the input file look at Appendix A, B and C.

2.4 *Components*

2.4.1 Compact modeling of spiral inductors

2.4.1.1 Spiral inductors

The compact model for a spiral inductor contains a number of parameters that are directly mapping the geometrical properties of the spiral layout into its RF compact parameters. An early model was presented by Long and Copeland [30]. The underlying model considers the spiral inductor as a (folded) transmission line. The model for the Q-factor results into:

$$Q = \frac{\omega L}{r_s + \dfrac{\left(\dfrac{\omega}{\omega_{ox}}\right)^4 \cdot r_{Si1}}{1+\left(\omega C_{ox1} r_{Si1}\right)^2}} \qquad (2.1)$$

The parameters are: ω_{Ox} is the oxide resonant frequency defined by L and C_{ox1}. The substrate resistance is given by r_{Si1}. In the work of Long and Copeland the lay-out aspects are all contained in L. and C and r_s. The latter is the series resistance of the inductor. The environment is taken here into account through the presence of the (passive) Silicon substrate. It should also be noted that current redistribution is not accounted for, since the parameters are not frequency dependent.

A substantial improvement of spiral inductor modeling was achieved by Niknejad and Meyer [31]. Their approach derives a circuit-equation formulation of the spiral inductor by re-interpretation of the equation (2.1). As is done in the model of Long and Copeland, the spiral inductor is broken in n times p parts, where n is the number of windings and p is the number of segments on each winding. This results in a network with n times p nodes.

The quality factor is given in this model as

$$Q = \frac{-\mathrm{Im}((Y_{11})}{\mathrm{Re}(Y_{11})} \qquad (2.2)$$

This expression includes the skin effect and proximity effect originating from coupling of parallel current segments.

The compact modeling of spiral inductors has been improved by Yue and Wong [32] who give explicit expressions for the various compact model parameters in terms of geometrical variables. For some additional models, see [33], [34].
The parameters have been used recently by C.Y. Lee et al. [35] to propose a scalable spiral inductor model.

The series resistance is:

$$R_s = \frac{1}{w \cdot \sigma \cdot \delta\left(1 - e^{-t/l}\right)} \qquad (2.3)$$

where *w* and *l* are the width and the length of the inductor body and σ is the conductivity and δ the skin-depth.

The series feed-forward capacitance of the spiral inductor is:

$$C_s = (n-1).w^2 \frac{\varepsilon_{ox}}{t_{ILD(M1/M2)}} \qquad (2.4)$$

where $t_{ILD(M1/M2)}$ is the thickness of the dielectric layer between the metal of the spiral and the metal that is used for building the under or over path. Moreover, n is the number of turns.
Next the capacitance to the substrate is considered.

$$C_s = w.l \frac{\varepsilon_{ox}}{t_{ox}} \qquad (2.5)$$

The Silicon substrate resistance and parasitic capacitance are given by

$$R_{Si} = \frac{2}{w \cdot l \cdot G_{sub}}$$
$$C_{Si} = \frac{w.l.C_{sub}}{2} \qquad (2.6)$$

The quality factor in terms of these variables becomes:

$$Q = \frac{\omega L_s}{R_s} \frac{R_p}{R_p + \left[\left(\omega \frac{L_s}{R_s}\right)^2 + 1\right].R_s} \left[1 - \left(C_p + C_s\right) \cdot \left(\omega^2 L_s + \frac{R_s^2}{L_s}\right)\right] \qquad (2.7)$$

where

$$R_p = \frac{1}{\omega^2 C_{oc}^2 R_{Si}} + \frac{R_{Si}\,(C_{ox} + C_{Si})^2}{C_{ox}^2} \qquad (2.8)$$

and

$$C_p = C_{ox} \cdot \frac{1 + \omega^2 \cdot (C_{ox} + C_{Si}) \cdot C_{Si} \cdot R_{Si}^2}{1 + \omega^2 \cdot (C_{ox} + C_{Si})^2 \cdot R_{Si}^2} \qquad (2.9)$$

2.5 *SnP file format*

2.5.1 Introduction

A Touchstone® file (also known as an SnP file) is an ASCII text file used for documenting the n-port network parameter data of an active device or passive interconnect network. Touchstone files have been accepted as a de-facto standard for the transfer of frequency dependent n-port network data. This section, based upon information from Agilent Corporation (the originator of Touchstone), is a formal specification of the Touchstone file format [36].

2.5.2 General syntax rules and guidelines

Following are the general syntax rules and guidelines for a Touchstone file.
1. Touchstone files are case-insensitive.
2. Only ASCII characters, as defined in ANSI Standard X3.4-1986, may be used in a Touchstone file. The use of characters with codes greater than hexadecimal 07E is not allowed. Also, ASCII control characters (those numerically less than hexadecimal 20) are not allowed, except for tabs or in a line termination sequence (carriage-return or carriage-return/line-feed combination).
3. Numbers are given in IEEE floating point notation (e.g. 1, 0.1, 1e-01).

***Note**: The use of Tab characters is strongly discouraged*.
4. Comments are preceded by an exclamation mark (!). Comments may appear on a separate line, or after the last data value on a line. Comments are terminated by a line termination sequence (i.e. multi-line comments are not allowed).
5. By convention, Touchstone filenames use a file extension of '.snp', where "n" is the number of network ports of the device or interconnect being described. For example, a Touchstone file containing the network parameters for a two port device would be named 'filename'.s2p, while a Touchstone file containing the data for a 3-port network would be 'filename'.s3p, and so on.
6. By convention, angles are measured in degrees.

2.5.3 File format description

2.5.3.1 Introduction

Touchstone data files consist of an 'option line' followed by one or more sets of network parameter data, where each set of data is taken at a specific frequency. The option line specifies (among other things) the kind of network data the file contains (S-parameter, Z-parameter, etc.), the format of the data values (magnitude-phase, real-imaginary, etc.) and the normalizing impedance. Data sets are arranged into one or more 'data lines', where the first line of network data is preceded by the frequency at which the data was taken or derived. Data for a 1-port or 2-port network are contained on a single data line while data

for 3-port and above networks are arrayed in a matrix format. The Touchstone format supports matrixes of unlimited size. (Note, however, many application programs and/or available computer memory will set an upper bound on how much data a Touchstone file can contain). Only one option-line/data-set pair is allowed per file.
In addition to the above option lines and data lines, files that describe 2-port devices may also contain noise parameter data. Finally, comment lines can be interspersed in the file as necessary.
The option line, data line, comment line and noise data are described in detail in the following sub-sections.

2.5.3.2 Option Line

Each Touchstone data file must contain an option line (additional option lines after the first one will be ignored). The option line must be the first non-comment line of the file, and is formatted as follows:

<*frequency unit*> <*parameter*> <*format*> R <*n*>

where

marks the beginning of the option line.
frequency unit specifies the unit of frequency. Legal values are
GHz, MHz, KHz, Hz. The default value is GHz.
parameter specifies what kind of network parameter data is contained in the file. Legal values are:
S for Scattering parameters,
Y for Admittance parameters,
Z for Impedance parameters,
H for Hybrid-h parameters,
G for Hybrid-g parameters.
The default value is S.
format specifies the format of the network parameter data pairs. Legal values are:
DB for dB-angle (dB = 20*log10|*magnitude*|)
MA for magnitude-angle,
RI for real-imaginary.
Angles are given in degrees. Note that this format does not apply to noise parameters. (Refer to the "Adding Noise Parameters" section at the end of this document). The default value is MA.
R *n* specifies the reference resistance in ohms, where n is a positive number of ohms (the real impedance to which the parameters are normalized). The default reference resistance is 50 ohms.

Option line parameters are separated by one or more white spaces; the option line itself is terminated with a newline character (either CR or CR/LF). If a parameter is missing it assumes the default value. With the exception of the opening # (hash mark) symbol and the value following "R", option line parameters can appear in any order.
In summary, the option line should read:

For 1-port files: # [HZ/KHZ/MHZ/GHZ] [S/Y/Z/G/H] [MA/DB/RI] [R n]
For 2-port files: # [HZ/KHZ/MHZ/GHZ] [S/Y/Z/G/H] [MA/DB/RI] [R n]
For n-port files: # [HZ/KHZ/MHZ/GHZ] [S/Y/Z/G/H] [MA/DB/RI] [R n]

where the square brackets ([]) indicate optional information; .../.../.../ means select one of the choices; and n is replaced by a positive number.

2.5.3.3 Option Line Examples

Minimum required option line (using all default values)
#
Frequency in GHz, S-parameters in real-imaginary format, normalized to 100 ohms:
GHz S RI R 100
Frequency in KHz, Y-parameters in real-imaginary format, normalized to 100 ohms:
KHz Y RI R 100
Frequency in Hz, Z-parameters in magnitude-angle format, normalized to 1 ohm:
Hz Z MA R 1
Frequency in KHz, H-parameters in real-imaginary format normalized to 1 ohm:
KHz H RI R 1
Frequency in Hz, G-parameters in magnitude-angle, format normalized to 1 ohm:
Hz G MA R 1

2.5.3.4 Data Lines

Following the option line is the data set. Data sets contain the data for the network parameters (S-parameter, Z-parameter, etc.) specified by the option line. Network data for 1-port and 2-port networks is contained on one data line, while data for 3-port and above are arranged on multiple data lines in a matrix format. Each set of network data is preceded by a frequency value (i.e. the first entry in the first (or only) data line of a data set is a frequency value), and the network data itself is formatted as pairs of values (magnitude-angle, dB-angle or real-imaginary).
There are three general rules for formatting data lines and data sets:

1. No more than four pairs of network data are allowed per data line.
2. Individual entries in a data lines are separated by white space
3. A data line is terminated by a newline character (CR or CR/LF combination).
4. All data sets must be arranged in increasing order of frequency.

Detailed descriptions for arranging the data for various n-port networks follows.

1-port and 2-port networks
Network parameter data for 1-port and 2-port networks at a single frequency can be contained on a single data line. As shown below, the data line consists of a frequency value followed by either one or four pairs of data values.
1-port data set (line)
<*frequency value*> <*N11*>

2-port data set (line)
<*frequency value*> <*N11*>, <*N21*>, <*N12*>, <*N22*>
Where
frequency value frequency at which the network parameter data was taken or derived.
N11, *N21*, *N12*, *N22* network parameter data points, where N11, N21, etc. represent pairs of data values .
Network parameter data points will be in magnitude-angle, dB-angle or real-imaginary format (i.e. pairs of values) as specified by the option line. For 1-port networks only '11' data is allowed, while for 2-port networks all four combinations are required.
Note the order in which 2-port network data is entered – '21' data precedes '12' data.
All entries in a data line are separated by one or more white space; a data line itself is terminated by a newline character (CR or CR/LF). Multiple data lines (sets) are allowed, but as mentioned above they must be arranged in increasing order of frequency.

Following are some examples of Touchstone files for 1-port and 2-port networks. Lines beginning with a bang (!) symbol are comments.

Example 1:
!1-port S-parameter file, single frequency point
MHz S MA R 50
!freq magS11 angS11
2.000 0.894 -12.136
In the above example, the value of S11 at a frequency of 2MhZ is given in magnitude-angle format. The reference impedance is 50 ohms.

Example 2:
!1-port Z-parameter file, multiple frequency points
MHz Z MA R 75
!freq magZ11 angZ11
100 0.99 -4
200 0.80 -22
300 0.707 -45
400 0.40 -62
500 0.01 -89
Note that in the above example Z11 (the input impedance) is normalized to 75 ohms, as given by the reference impedance (R 75) in the option line.

Example 3:
!2-port H-parameter file, single frequency point
KHz H MA R 1
! freq magH11 angH11 magH21 angH21 magH12 angH12 magH22 angH22
2 .95 -26 3.57 157 .04 76 .66 -14
In the above example the H-parameters are given in magnitude-angle format, with a reference impedance of 1 ohm.

Example 4:
!2-port S-parameter file, three frequency points
GHZ S RI R 50.0
!freq RelS11 ImS11 ReS21 ImS21 ReS12 ImS12 ReS22 ImS22
1.0000 0.3926 -0.1211 -0.0003 -0.0021 -0.0003 -0.0021 0.3926 -0.1211
2.0000 0.3517 -0.3054 -0.0096 -0.0298 -0.0096 -0.0298 0.3517 -0.3054
10.000 0.3419 0.3336 -0.0134 0.0379 -0.0134 0.0379 0.3419 0.3336
In the above example the S-parameter data is given in real-imaginary format.

3-port and 4-port networks
The network parameter data for a 3-port or 4-port network is arranged in a matrix format, with each line of data representing one row of the matrix. In other words (as shown below), the data for a 3-port network
is entered as three lines of data, with each line containing three data pairs (i.e. a 3x3 matrix of network parameter values). Likewise, the data for a 4-port network is entered as four lines with four data pairs per line (a 4x4 matrix). As required by the general rules, the first data line of each network parameter data set is preceded by the frequency value at which the data was taken.
3-port network description
<*frequency value*> <*N11*> <*N12*> <*N13*>
<*N21*> <*N22*> <*N23*>
<*N31*> <*N32*> <*N33*>
4-port network description
<*frequency value*> <*N11*> <*N12*> <*N13*> <*N14*>
<*N21*> <*N22*> <*N23*> <*N24*>
<*N31*> <*N32*> <*N33*> <*N34*>
<*N41*> <*N42*> <*N43*> <*N44*>
Where
frequency value frequency at which the network parameter data
was taken or derived.
N11, *N12*, etc. network parameter data points, where N*ij*
represent pairs of data values.
As usual, network parameter data points are entered in magnitude-angle, dB-angle or real-imaginary format (i.e. pairs of values) as specified by the option line. All entries in a data line are separated by one or more white spaces; a data line itself is terminated by a newline character (CR or CR/LF). Multiple data sets are allowed, but as mentioned above they must be arranged in increasing order of frequency.

Following is an example of an S-parameter description of a 4-port network.

Example 5:
! 4-port S-parameter data, taken at three frequency points
GHZ S MA R 50
5.00000 0.60 161.24 0.40 -42.20 0.42 -66.58 0.53 -79.34 !row 1
0.40 -42.20 0.60 161.20 0.53 -79.34 0.42 -66.58 !row 2
0.42 -66.58 0.53 -79.34 0.60 161.24 0.40 -42.20 !row 3

0.53 -79.34 0.42 -66.58 0.40 -42.20 0.60 161.24 !row 4
6.00000 0.57 150.37 0.40 -44.34 0.41 -81.24 0.57 -95.77 !row 1
0.40 -44.34 0.57 150.37 0.57 -95.77 0.41 -81.24 !row 2
0.41 -81.24 0.57 -95.77 0.57 150.37 0.40 -44.34 !row 3
0.57 -95.77 0.41 -81.24 0.40 -44.34 0.57 150.37 !row 4
7.00000 0.50 136.69 0.45 -46.41 0.37 -99.09 0.62 -114.19 !row 1
0.45 -46.41 0.50 136.69 0.62 -114.19 0.37 -99.09 !row 2
0.37 -99.09 0.62 -114.19 0.50 136.69 0.45 -46.41 !row 3
0.62 -114.19 0.37 -99.09 0.45 -46.41 0.50 136.69 !row 4
Note that the data pairs do not have to be aligned in columns; the only requirement is that there be 3 (3-port networks) or 4 (4-port networks) pairs of network data per data line.

5-port and above Networks
The network data for 5-port and above networks is also arranged in a matrix format. However, because the Touchstone format is limited to a maximum of 4 network parameter data points per line, additional entries beyond the first four in the matrix row must be continued on the following line(s). Each row of the matrix must start on a new line. As usual, the first entry in the first data line of a data set is the frequency value. These rules are illustrated by showing the format for a 6-port network:
6-port network format (single frequency point)
<*frequency value*> <*N11*> <*N12*> <*N13*> <*N14*> !row 1
<*N15*> <*N16*>
<*N21*> <*N22*> <*N23*> <*N24*> !row 2
<*N25*> <*N26*>
<*N31*> <*N32*> <*N33*> <*N34*> !row 3
<*N35*> <*N36*>
<*N41*> <*N42*> <*N43*> <*N44*> !row 4
<*N45*> <*N46*>
<*N51*> <*N52*> <*N53*> <*N54*> !row 5
<*N55*> <*N56*>
<*N61*> <*N62*> <*N63*> <*N64*> !row 6
<*N65*> <*N66*>
Where
frequency value frequency at which the network parameter data
was taken or derived.
N11, *N12*, etc. network parameter data points, where N*ij*
represent pairs of data values.
As shown, each row of matrix data extends over two lines of the file, and each new row of the matrix starts on a new line. As usual, network data values are entered in pairs according to the format specified in the option line and each entry is separated by white spaces.

2.5.3.5 Comment Lines

A Touchstone data files can be documented by preceding a comment with the exclamation mark (!). A comment can be the only entry on a line or can follow the data on any line.

2.5.3.6 Adding Noise Parameters

Noise parameters can be included in Touchstone data file; however, they can only be included in 2-port network descriptions. Noise data follows the G-, H-, S-, Y-, or Z-parameters data for each frequency point.
Each line of a noise parameter has the following five entries:
<x1> <x2> <x3> <x4> <x5>
where
x1 Frequency in units. The first point of noise data must have a frequency less than the frequency of the last S-parameter frequency.
x2 Minimum noise figure in dB.
x3 Source reflection coefficient to realize minimum noise figure
(MA).
x4 Phase in degrees of the reflection coefficient (MA).
x5 Normalized effective noise resistance. A simulator requires this parameter to meet physical requirements. If the user-supplied x5 value is less than allowed for this requirement, then a simulator may force this x5 value to the lowest physical limit.
Note that the frequencies for noise parameters and network parameters need not match. The only requirement is that the lowest noise-parameter frequency be less than or equal to the highest network-parameter frequency. This allows the file processor to determine where network parameters end and noise parameters begin.
The source reflection coefficient and effective noise resistance are normalized to the same resistance as specified in the option line for the network parameters.

Example 7:

```
!2-port network, S-parameter and noise data
# GHZ S MA R 50
2 .95 -26 3.57 157 .04 76 .66 -14
22 .60 -144 1.30 40 .14 40 .56 -85
! NOISE PARAMETERS
4 .7 .64 69 .38
18 2.7 .46 -33 .40
```

2.6 *Linear system format*

A compact (binary, sparse matrix) format, able to describe the following linear, time invariant system (e.g. as output of a field solver)

$$A_2 X'' + A_1 X' + A_0 x = Bu$$
$$y = C X + Du$$

with

- **X** : a vector of internal states (n)
- **u** : a vector of input variables (N_i input variables)
- **y** : a vector of output variables (N_o output variables)

is also needed. Order reduction methods (ROM) can then be applied to these systems. The format can be used as solver output, ROM input and ROM output. The ROM input and output formats are the same, but the latter have smaller sizes. Moreover, the file describing simulation data is the same (ROM and initial system having the same input/output vector size).

2.6.1 Introduction

In the exchange of matrix files between the different partners in nanoCOPS, 3 matrix formats will be made available. These formats are the compressed-row storage format CRS, coordinate storage (CS) and matrix market format (MMF). For a detailed overview, see [37].

2.6.2 Compressed-row storage

This storage scheme is also known as Boeing matrix storage. However, for our purposes we use a slightly less strict formulation. (The Boeing format is rather old and therefore strict in counting characters on a line). Here we use tabs separated dumping.

As far as the storage scheme is concerned, the compressed-row storage (CRS) scheme views a matrix as one array of entries (all rows are placed behind each other). The matrix elements are then contained in one array A. The total number of non-zero matrix elements are NNZ. Note that zeros are not stored. However, some care must be taken with this statement. Usually a sparse storage system can be designed before an actual computation is done. It informs us about which variable are coupled to other variables due to a grid structure. So we may lay out the storage scheme and next compute the entries. During computation it can turn out that a zero value results. If we stick to our original storage scheme we allow for a limited set of zeros in the stored matrix.

Next of each A-entry a pointer is given referring to the column index of that entry. All these pointers are listed in a second array JA.

Finally a third array whose size is N+1, where N is the number of rows (this explains the name CRS) is given that contains for each row a pointer to the first matrix element at that row into the arrays A and JA.
The motivation behind taking one more entry than there are rows is that do-loops can be easily programmed.

Summary

array A(1,NNZ) : containing all matrix elements A
array JA(1, NNZ) : containing all column indices
array IA(1,NSIZE+1) containing the pointers of the first elements in each row into A and JA

file for Right hand side (RHS)

Example

$$A = \begin{vmatrix} a11 & . & a13 & a14 & . & . \\ . & a22 & . & a24 & & 26 \\ . & . & a33 & . & a35 & . \\ a41 & . & a43 & . & . & a46 \end{vmatrix}$$

N=4
NNZ=11

A= { a11, a13, a14, a22, a24, a26, a33, a35, a41, a43, a46}

JA= { 1, 3, 4, 2, 4, 6, 3, 5, 1, 3, 6}

IA = { 1, 4, 7, 9, 12}

These arrays can be stored in three different files. This is made available in the project. We refer to this storage scheme as the compressed-row storage scheme (CRS).

Example of the A file:

MATRIX SYSTEM PRINTOUT
BigSize : 11
values
(30373933473.5358,56.3258815713584)
(-134995259.882384,-0.250337251428265)
(-75934833.6838411,-0.140814703928399)
(-30373933473.5358,-56.3258815713584)
(-134995259.882384,-0.250337251428265)
(60747866947.0716,112.651763142717)

(-151869667.367682,-0.281629407856798)
(-60747866947.0716,-112.651763142717)
(60747866947.0716,112.651763142717)
(-134995259.882384,-0.250337251428265)
(-151869667.367682,-0.281629407856798)

Example for the IA file:

MATRIX SYSTEM PRINTOUT
Size +1 : 5
IA
1
3
4
7
9
12

As is seen the first 3 lines contain comments.

Example for the JA file:

MATRIX SYSTEM PRINTOUT
BigSize : 11
JA
1
3
4
2
4
6
3
5
1
3
6

The vector RHS is a simple 1D array of numbers that can be either real, double or complex<double>.

2.6.3 Coordinate storage

The coordinate storage is built from three one-dimensional arrays, where the first array contains the row index, the column index and the matrix value.
These arrays can be dumped in three different files and then we refer to them as the coordinate-storage scheme (CS).

For the above example the file will appear as

% matrix system print out: file of matrix values for CS
% number of non-zero matrix entries: 11
(30373933473.5358,56.3258815713584)
(-134995259.882384,-0.250337251428265)
(-75934833.6838411,-0.140814703928399)
(-30373933473.5358,-56.3258815713584)
(-134995259.882384,-0.250337251428265)
(60747866947.0716,112.651763142717)
(-151869667.367682,-0.281629407856798)
(-60747866947.0716,-112.651763142717)
(60747866947.0716,112.651763142717)
(-134995259.882384,-0.250337251428265)
(-151869667.367682,-0.281629407856798)

The row indices are

% matrix system print out: file of row pointers for CS
% number of non-zero matrix entries: 11
% number of rows in matrix: 4
1
1
1
2
2
2
3
3
4
4
4

whereas the column indices are in the file which looks as:

% matrix system print out: file of column pointers for CS
% number of non-zero matrix entries: 11
% number of columns in matrix: 6
1
3
4
2
4
6
3

5
1
3
6

In general it is also convenient to store the matrix in a single file. Then we refer to it as the matrix-market format storage (MMF).

For the example above, the single file will look like

```
% File in matrix - market format
% number if non-zeros: 1368
%       row    col      value
4   6   11
1       1       (30373933473.5358,56.3258815713584)
1       3       (-134995259.882384,-0.250337251428265)
1       4       (-75934833.6838411,-0.140814703928399)
2       2       (-30373933473.5358,-56.3258815713584)
2       4       (-134995259.882384,-0.250337251428265)
2       6       (60747866947.0716,112.651763142717)
3       6       (-151869667.367682,-0.281629407856798)
3       5      (-60747866947.0716,-112.651763142717)
4       1       (60747866947.0716,112.651763142717)
4       3       (-134995259.882384,-0.250337251428265)
4       6       (-151869667.367682,-0.281629407856798)
```

The entries in these examples are chosen to be of the type complex<double>.
At this instance it seems that the matrix-market format will be used most extensively in the project since there are convenient MatLab tools to convert one format into another.

2.6.4 Matrix Naming conventions

We also consider a naming convention for the matrices. A matrix will be named after a number of key-words and numerical parameters. In general the matrix name is composed of the following string:

filename = "matrix-KEYWORD1-KEYWORD2-KEYWORD3-integer1-KEYWORD4-integer2"

Example:

matrix-VA-FULL-MMF-1-MATRIX is the name of some file.

Explanation:

KEYWORD1 = { VA | VFA}

VA means: only voltages (V) and vector potential (A) variables participate in the state-space description.
VFA means: voltages (V), Fermi potentials (F) and vector potential (A) variables participate in the state-space description.

KEYWORD2 = { FULL | MTE | ROM }

FULL means: The full matrix of the state-space description is dumped. Actually, this corresponds to composing the Newton-Raphson system in frequency space by defining

$$\boldsymbol{F(X) = 0}$$

leading to the update equation

$$\boldsymbol{DX = - \mid dF/dX \mid (X - X^*)}$$

or

$$\mathbf{M.x = b}, \text{ where } \mathbf{M = - \mid dF/dX \mid}$$

In the frequency space M = M(□) and initially b describes the coupling to a given potential at the contacts.

MTE means: Matrix-Taylor Expansion. Now we revisit the above composition of M. However, we write M as

$$\mathbf{M = M_0 + M_1 * (j \blacklozenge) \square + M_2 * \square (j \blacklozenge)^2.}$$

where ◆ = 2 □ *f* and *f* is the frequency (j is the imaginary unit).

ROM means: Reduced-Order Modeling. This keyword says that the matrix refers to a matrices A, B, C and D of the following ROM system:

$$\begin{vmatrix} Dt\,X \\ \\ Y \end{vmatrix} = \begin{vmatrix} A & B \\ \\ C & D \end{vmatrix} * \begin{vmatrix} X \\ \\ U \end{vmatrix}$$

where U are the input variables of the ROM system and Y are the output variables of the ROM system.

The matrix A is extractable from the MTE matrices and does not need to be re-created.

KEYWORD3 = { CS | CRS | MMF }

This key word refers to the storage scheme.

integer1 = { 1 | 2 | 3 | .....}

This integer corresponds to the dump cycle, where the cycle counts the iterative refinement sweeps of the frequency problem solving.

KEYWORD4 = { F0 | F1 | F2 | B0 | B1 | C0 | C1 | C2 | D0 | D1 }

meaning:

F0: constant matrix of the MTE dump
F1: linear term of the MTE dump
F2: square term of the MTE dump
B0 constant term of B matrix of the ROM dump
B1: linear term of B matrix of the ROM dump

KEYWORD5 = { A | IA | JA | ROW | COL | MATRIX | RHS }

meaning

A : The values array for a CRS or CS dump
IA: The pointer array for a CRS dump
JA: The column pointer array for a CRS dump
ROW: the row array for a CS dump
COL: the column array for a CS dump
MATRIX: the matrix of a MMF dump
RHS: the right-hand side dump

2.7 *Statistical data formats*

2.7.1 Introduction

In the near future, large parameter variations are expected within the manufacturing process of electric circuits and electric devices, which make the design and functionality of the end products critical, see [38],[39], for example. Thus the mathematical models have to include the variabilities of the parameters. In the project, the variations are modelled by random variables for scalar parameters (e.g. capacitances, inductances, resistances, etc.) or random fields for spatial effects (e.g. material parameters, geometries, etc.) to achieve an uncertainty quantification.
The models require the specification of the probability distributions as input. On the one hand, traditional distributions can be chosen (e.g. Gaussian, uniform, etc.). On the other hand, this task will be accomplished by fitting the random distributions to samples, which are obtained by physical measurements of real devices in cooperation with the industrial partners.
From the measurements, key figures like the expected value, the variance, the skew and the kurtosis or other statistical information can be calculated, see [40], for example. Furthermore, correlations of the measurements for different quantities may appear. The numerical simulation of the stochastic models yield results, which allow for approximating the key figures and probability distributions of the outputs.
In the following, the statistical data formats are described for the two cases of the input data.

2.7.2 Scalar random parameters

For the scalar quantities, the measurements consist either of a list of scalar numbers or of a list of vectors including the scalars component-wise, i.e.,

$$(m_1^1,....,m_k^1),(m_1^2,....,m_k^2),....,(m_1^l,....,m_k^l),$$

where l measurements are done for k different quantities simultaneously. It should be specified separately which different settings have been chosen to obtain the measurements. For example, (i) each dataset $(m_1,....,m_k)$ can be associated to different electric devices from the same manufacturing process, or, (ii) each dataset $(m_1,....,m_k)$ is taken from the same device at different time points $t_1,....,t_l$ in the magnitude of weeks or years to investigate ageing. In case (i), a permutation of the measurements does not change the statistical properties or key figures. In case (ii), additional specifications of the setting or coordinates play a role and the ordering of the measurements is intrinsic.

2.7.3 Spatial random fields

For spatial processes, the correlations shall be analyzed. The one-, two- and three-dimensional case is feasible. In the following, the three-dimensional case is described,

since the other two settings appear by simply omitting dimensions. Furthermore, just a scalar quantity is considered in each space point, because the extension to vector valued formats is straightforward.

2.7.3.1 Structured measurements

In a cuboid, a uniform grid is given, where a measurement is done in each grid point. Hence the data is written in the form

$$(x_i, y_j, z_k, m_{\{ijk\}}) \text{ for } i = 1,...,n_1,\ j = 1,...,n_2, k = 1,...,n_3,$$

where $m_{\{ijk\}}$ represents the measurement of the quantity at the space point (x_i, y_j, z_k).
Alternatively, the measurements can be arranged in a three-dimensional field of numbers and the information of the space points is stored separately.
Often the spatial domain is designed using a uniform grid of points or cells, whereas only a few measurements are done in the domain. In this case, the above data format is still feasible, where cells without a measurement just include an empty entry for $m_{\{ijk\}}$. This data format often agrees to the structure required for routines of graphical illustration (post-processing).

2.7.3.2 Unstructured measurements

Now the position of the space points for the measurements are arbitrary. Thus the data is written as

$$(x_i, y_i, z_i, m_i) \text{ for } i = 1,...,l,$$

with m_i being again the measurement at space point (x_i, y_j, z_k). In this case, an arrangement of the measurements in a three-dimensional field of numbers does not make sense.

3 Interface implementations & languages

This chapter deals with coding styles, compiler choices, library development, linking, third party software, tool integration and GUI design. This chapter can be used to check if your code is compliant to integrate with the MAGWEL software environment.

3.1 *Coding styles*

When producing code, always keep in mind that someone else has to read and understand the code without you be present to ask questions. Code is written once and read many times. It implies that some guidelines of good code writing should be respected.

Algorithmic fragment should be commented in order to give a short summary of what the code should do. Each method/subroutine should include a small functionality description in a few lines. Also each input/output parameter should be described in a single line: its type and its purpose.

Use variable names that are descriptive but not excessively long. At all times try to avoid "humor" in the naming of variables. What may be funny to you is just distracting the focus from the real content and acts very disturbing to the independent reader! It is also recommended to add references to deliverables, papers (accessible information) inside the comments of the code. Indentation is very helpful in reading/understanding the code and should be applied carefully. Also keep in mind the following important software development stages.

3.1.1 Design and test creation

Before actually starting to write the code, a clear plan needs to be designed how the various modules interact, which data is needed , what is the output, and what is the best possible data storage/addressing scheme. This plans should be added when delivering the code.
Before actually starting to write the code, some small (unit) tests should be designed. These tests are pathetic examples that show/test parts of the functionality/modules. The outcome of these tests can easily be formulated and verified. It is key that you design tests before writing the code and not the other way around.

3.1.2 Implementation and testing

The implementation phase represents the actual code creation that materializes the design concepts. Here above style guidelines apply. After implementation all unit tests run and give the right results.

Testing is an essential part in the software development process. At any stage it should be prohibited to release software that did not undergo a severe testing phase. Testing takes place at various levels.

1. Modules and module fragments (units) are submitted to dedicated unit test calls. For that purpose one needs to define some input, formulate the expected output and check the agreement.
2. Combining fragments in complete modules requires its own testing to check the data communication and to check the expected behavior of the combined fragments (modules). It is highly recommended to keep a track record of the test results and to be able to activate/de-active test loops easily.
3. Finally, part of testing deals with usage for "real-life" examples. These example are not the actual industrial test cases but simplifications thereof in order to reduce run times and at the same time to provide access to code issues.

3.2 *Compiler choices*

The following languages for coding are accepted;

- C++
- Fortran
-
-

Development and showing feasibility can be done in MatLab/python/octave. But keep in mind that it is hard/impossible to perform a tight integration of the software into the MAGWEL environment for technical and licensing reasons.

This means that before handing over code, a rewrite should be done in C++ or Fortran, keeping the rules above in mind.

From the exploitation point-of-view it is required that the end product can be maintained and supported. As a consequence, the final product may depend on third party libraries provided that these libraries can be linked in statically and compiler options exist to account for machine dependency. GNU compilers are generally well suited for this requirement. It is desired to select as a preferred language C++. However it is well known that MATLAB is an ideal vehicle for quickly exploring scientific ideas, but once that these ideas are consolidated (from an end-users' perspective) a conversion to C++ / Fortran source must be foreseen.

3.3 *Library development*

Auto-tools will be applied to generate the libraries. (Automake)

3.4 *Linking*

make files are provided and auto-tools will be used to provide the final executable.

3.5 *3rd party software*

Third party software is permissible if support and licensing can be assured. In practice, the NAG software can be used because NAG provided adequate on-line help and support.

- The GPL is a copyleft license, which means that derived works can only be distributed under the same license terms. Libraries under GPL cannot be used.

- The GNU Lesser General Public license (LGPL) was created to have a weaker copyleft than the GPL, in that it does not require own custom-developed source code (distinct from the LGPLed parts) to be made available under the same license terms. Libraries under LGPL can be used.

3.6 *Tool integration*

The generic guideline for tool integration will be object-orientation. It means that different functionalities and development tracks will be clearly separated. The integration of the different work package results will be addressed in the design phase of the software.

3.7 *Input parameters*

The input of parameters of the code will be in xml, described in a schema (.xsd) file. This requires further that the MAGWEL API (Codestar) will be extended to incorporate the new functionalities (Performed by MAGWEL). A list of required input parameters, needs to be provided with the software.

The Codestar API header files are the link to the computational engines which are developed in the various work packages.

3.8 *Graphical User Interface (GUI) design*

Since the resulting software is intended to be exploited by MAGWEL in a commercial end use, and since there is an existing GUI design already operational, the GUI design of the new tools needs to be compliant with the MAGWEL-GUI environment.

The GUI for the output (if needed) will be developed in Qt, to be compliant with the current strategy at MAGWEL.

3.9 *Final Remarks*

Projects, such as nanoCOPS, have competing (from the partners' perspective) interests that require special attention. Whereas the research partners will mainly focus on discovering new or improving algorithms, the exploitation requirements require robustness, absence of code errors, data consistency, CPU time consumption, minimal end-user interference (dummy proof code). Therefore, it will be unavoidable that conflicts of interest will arise at runtime of the project, where a trade-off between the 'science' effort and the 'business' effort must be made. In order to manage such conflicting interests, dedicated implementation tasks are defined in the technical work packages. The primary goal of these tasks is to guarantee that the work load is fairly shared amongst the two effort branches,

The modus-operandi is that in a first stage the emphasis will be on algorithm development. The implementation of this stage will be done with less or without emphasis on the various end user requirements. The developers of algorithms produce their results in their preferred tool suite. MAGWEL will provide the interfaces to these tools but as far as MAGWEL is concerned these algorithms are 'black boxes' (*). In parallel MAGWEL will explore market opportunities and if there is industrial interest triggered a joint effort will be done by the developers of the algorithms and MAGWEL to turn the 'black boxes' into 'white' (**) or 'grey' boxes.
If it turns out that the market shows manifest lack of interest, the boxes will stay black.

(*) A 'black box' is piece of software that the exploitation partner will provide 'as-is' without support.
(**) A 'white box' is a piece of software that the exploitation partner has integrated in his product suite, will provide support and maintenance and is (re-) programmed such that the business requirements are fulfilled.

4 References

[1] K. S. Kundert, A. Sangiovanni-Vincentelli, T. Tsugawara, Techniques for finding the periodic steady state response of circuits, in T. Ozawa, Analog methods for computer aided circuit analysis and diagnosis, Marcel Dekker, N. Y., 169-203 (1988).

[2] S. Skelboe, Computation of the Periodic Steady State Response of Nonlinear Networks by Extrapolation Methods, IEEE Trans. Circuits and Systems, **CAS-27** , 161-175 (1980).

[3] F. Constantinescu, A. Ionescu, M. Nitescu, A new extrapolation method for fast finding of the periodic steady state in nonlinear circuits, Symposium on Nonlinear Theory and Applications (NOLTA'98), Crans-Montana, Switzerland, September 14-17, 1998.

[4] A. Brambila, P. Maffezzoni, Envelope following method for the transient analysis of electrical of electrical circuits, IEEE Trans. Circuits and Systems, **CAS-47**, 7, 999-1016 (2000).

[5] D. Sharrit, New method of analysis of communication systems, Proc. MTTS'96 WMFA: Nonlinear CAD Workshop, June 1996.

[6] How-Siang Yap, HP EEsof Division, Hewlett-Packard, Designing to Digital Wireless Specifications Using Circuit Envelope Simulation (1999).

[7] A User's Guide to Envelope Following Analysis, Cadence Application Note (1999).

[8] Agilent Technologies, Advanced Design System 1.5 – Circuit Simulation, December 2000.

[9] http://www.ee.washington.edu/class/cadta/hspice

[10] Virtuoso Spectre Circuit Simulator User Guide, Product Version 5.1.41, July 2004, Cadence Design Systems.

[11] Virtuoso Spectre Circuit Simulator Components and Device Models Manual, Product Version 5.1.41, November 2004, Cadence Design Systems.

[12] Using Circuit Simulators, August 2005, Agilent Technologies.

[13] http://www.nxp.com/models/mos_models

[14] H. Wang, T.-L. Chen, and G. Gildenblat, Quasistatic and Non-quasistatic Compact MOSFET Models Based on Symmetric Linearization of the Bulk and Inversion Charges, IEEE trans. Electron Dev. **50**(11), 2262-2272 (2003).

[15] H. Wang et al., Unified Non-Quasi-Static MOSFET Model for Large-Signal and Small-Signal Simulations, CICC 2005.

[16] G. Gildenblat, X. Li, H. Wang, W. Wu, R. van Langevelde, A.J. Scholten, G.D.J. Smit and D.B.M. Klaassen, *Introduction to PSP MOSFET Model*, Nanotech2005.

[17] http://www.xs4all.nl/~kholwerd/interface/bnf/gdsformat.html and http://www.rulabinsky.com/cavd/text/chapc.html

[18] http://www.cadence.co.in/whitepapers/sca446apn.pdf - Substrate Coupling Analysis for RF circuits.

[19] http://www.imec.be/codestar/d4.pdf - Interface format to the CAD environment.

[20] B. Engquist, A. Majda, Absorbing boundary conditions for the numerical simulation of waves, Math. Comp., **31**, 629-651 (1977).

[21] G. Mur, Absorbing boundary conditions for the finite-difference approximation of the time-domain electromagnetic field equations, IEEE Trans. Electromagn. Compat. **EMC-23**, 4, 377-382 (1981)

[22] R. Holland, J.W. Williams, Total-field versus scattered-field finite-difference codes: A comparative assessment, IEEE Trans. Nucl. Sci. **NS-30**, 6, 4583-4588 (1983).

[23] C.M. Rappaport, T. Gürel, Reducing the computational domain for FDTD scattering simulation using the sawtooth anechoic chamber ABC, IEEE Trans Magnetics, **31**, 3, 1546-1549 (1995).

[24] J. P. Bérenger, A perfectly matched layer for the absorption of electromagnetic waves, J. Comput. Phys., **114,** *1*, 185–200 (1994).

[25] James R. Wait, On the PML Concept: A View from the Outside, Antennas and Propagat. Magazine, **38**, 2, 48-51 (1996).

[26] H. Wolter, M. Dohlus, T. Weiland, Broadband Calculation of Scattering Parameters in the Time-Domain, IEEE Trans. Magn., **30**, *5*, 3164-3167 (1994).

[27] R. Schuhmann, T. Wittig, I. Munteanu, B. Trapp, T. Weiland, Time and Frequency Domain Simulations of highly resonant RF Devices, Proc. Of Int. Conf. On Electromagnetics in Advanced Applications, Torino, Italy, Sep. 2001.

[28] D. Ioan, G. Ciuprina, M. Radulescu, Absorbing Boundary Conditions for Compact Modeling of on-chip Passive Structures, Proc. ISEF 2005, Sept. 15-17, Baiona, Spain, 2005 – The best paper award.

[29] D. Ioan, I. Munteanu, Missing link rediscovered: The electromagnetic circuit element concept, Proc. of Japanese Romanian Joint Seminar on Applied Electromagnetics and Mechanics JRJSAEM'98, p. 1. Kiryu, Gunma, Japan, Nov. 1998.

[30] J. Long and M. Copeland, The modeling, characterization and design of monolithic Inductors for Silicon RF ICs, IEEE Journal of Solid-State Circuits, **32**, 3, 357-369 (1997).

[31] A. Niknejad and R. Meyer, Analysis, Design and Optimization of Spiral Inductors and Transformers for Si RF ICs, IEEE Journal of Solid-State Circuits, **33**, 10, 1470-1481 (1998).

[32] C.P. Yue and S.S. Wong, Physical modeling of spiral inductors on Silicon, IEEE Trans. On Electron Devices, **47**, *3*, 560-568 (2000).

[33] S.S. Mohan, M. Hershenson, S.P. Boyd and T.H. Lee, *Simple Accurate Expressions for Planar Spiral Inductances*, IEEE Journal of Solid-State Circuits, Oct. 1999, pp. 1419-24, available at the Integrated Spiral Inductor Calculator (http://www-smirc.stanford.edu/spiralCalc.html)

[34] http://rfic.eecs.berkeley.edu/~niknejad/asitic.html

[35] C-Y. Lee, T-S Chen, J.D-R. Deng and C-H Kao, A simple systematic spiral inductor design with perfected Q-improvement for CMOS RFIC application, IEEE Transactions on Microwave Theory and Techniques., **53**, *2*, 523-528 (2005).

[36] Touchstone® File Format Specification Rev 1.1, available at www.eda.org/pub/ibis/connector/touchstone_spec11.pdf

[37] http://math.nist.gov/MatrixMarket/formats.html

[38] P. Li, F. Liu, X. Li, L.T. Pileggi, S.R. Nassif, Modeling interconnect variability using efficient parametric model order reduction. In: Design, Automation and Test Conference in Europe, 2005, pp. 958—963.

[39] Z. Wang, X. Lai, J. Roychowdhury, PV-PPV: Parameter variability aware, automatically extracted, nonlinear time-shifted oscillators macromodels. Proc. IEEE Design Automation Conference, June 2007, pp. 142—147.

[40] D. Freedman, R. Pisani, R. Purves, Statistics. 4th ed. Norton & Company, 2007.