IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 11, Issue 1, Ser. I (Jan. - Feb. 2021), PP 09-21
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Educational Introduction to VLSI Circuit Design and Simulation with Tanner-EDA Tools

George P. Patsis[1,2]
[1]*Department of Electrical and Electronics Engineering, University of West Attica, Athens, Greece*
[2]*Nanometrisis private company, Scientific & Technological Park "Leukippos", NCSR Demokritos, Neapoleos 27 & Patr. Grigoriou Str., 15341, Agia Paraskevi, Attica, Greece*
Corresponding Author: George P. Patsis

Abstract: *A compact introduction to the use of Tanner's-EDA S-Edit and T-Spice tools is mandatory for students taking their first VLSI design course. The article present a few clear examples of design and simulation of basic building blocks in VLSI design. Their study will provide the student with the basic technical skills to move to more advanced design work.*
Keywords *– VLSI, CMOS, Tanner-EDA, Inverter, SPICE, Simulation, Circuit-design*

Date of Submission: 12-02-2021 Date of Acceptance: 25-02-2021

I. INTRODUCTION – MOTIVATION

The beginner to circuit design for VLSI applications has to acquire a certain degree of knowledge, skills and experience on corresponding software design tools. Usually the information provided in the manuals is not intended so to teach the designer but to help find references on what the software at hand can do. This can be frustrating, at the beginning of the learning curve.

In the current article an educational approach is undertaken in the same detailed presentation that has been used in previous related works [**1-6**]. Based on Tanner-EDA's examples [**7**] (today, owned by Mentor Graphics), a detailed implementation is provided along with various directives on the way simple introductory circuits can be designed and simulated. Once these examples are studied in detail, one is able to move on to more complex design applications. Specifically, S-Edit, T-Spice and W-Edit are the tools to be used here. Version 16.0 will be used, but there should be no difference in newer versions of the software.

In the second section of this work a brief review of S-Edit's project structure is presented followed by several simple design examples. Specifically, a single MOS transistor is simulated, followed by examining several stimulus sources, and then a CMOS inverter is thoroughly investigated for various SPICE analyses without going into the many details of the SPICE setup files. A more detailed analysis focusing on the use of SPICE commands should be the subject of a separate follow up article.

Refer to [**7, 8**] for related examples in circuit design and simulation with the Tanner EDA tools.

II. PROJECT STRUCTURE IN S-EDIT

Initially, it is necessary to understand the basic project structure and terminology used in S-Edit. The highest-level-entity in the S-Edit schematic-database-hierarchy is the **design**. A design contains many **cells**, some of which may be referenced from a library. Most often a cell contains a single interface, with a single symbol view and a single schematic view. However, a cell can contain any number of interfaces, and each interface can contain any number of symbol views and schematic views. Therefore a design is the container for all elements in the design-database. A library if also a design, one whose cells are extremely referenced by other designs. A design can reference multiple libraries, and any given library can be referenced by multiple designs. A cell is the fundamental unit of design. So, a design contains multiple cells and cells in turn contain multiple views, of different types. Cells can be instanced by other cells (an instance is a generic reference to a representation of $cell_B$ found within $cell_A$). So in more compact terminology: `Design = {cell`$_i$`, cell`$_j$`}, cell`$_j$ `in Library`$_k$`, i, j, k: integer, Library = Design, Cell = {Interface(s)}, Interface = {symbol-view(s), schematic-view(s)}`

A **property** is an attributer of an object (cell, instance, shape). S-Edit differentiates between built-in properties (e.g. cell name, shape type) which are always defined, and user defined properties (e.g. channel length L, on a transistor cell) which are optional and not interpreted directly by the software engine. Each type of object has its own set of parameters, which are displayed and can be edited by the Properties-navigator. Finally, a **view** is simply a component of a cell definition. Each view provides a different way of depicting a

cell. S-Edit view-types are symbols, schematic, and interface. Since T-Spice can also execute Verilog-A and Verilog-AMS code, simulating a schematic with subcells that have either a Verilog-A or AMS description will simulate the Verilog code; so S-Edit has also SPICE, Verilog-A, and Verilog-AMS views. Cells can instance each other, but cyclical cell references are not allowed.

A **symbol view** is a graphical description of a cell, for use in schematic views of other cells. A symbol view contains the ports of a cell and non-electrical geometry for representation-purposes only. Usually, the most basic design components will have only a symbol view. A **schematic view** is a more detailed view of a cell, showing ports, instance, references to ports in the parent cell of instanced cells, connecting wires and graphic objects (e.g. boxes, polygons, paths, text labels) that have no electrical meaning. An **interface view** shows the definition of the electrical interface of a cell, containing port, optional information for those ports, and a set of defined parameters. Each schematic view and symbol view in a cell must be associated with a specific interface. When an instance of a cell is created a corresponding interfaced must be defined. Cells may be associated with multiple interfaces. It is often useful to create a cell with multiple interface, symbol, or schematic views. **Figure 0** shows an example of the hierarchy of views in a schematic.

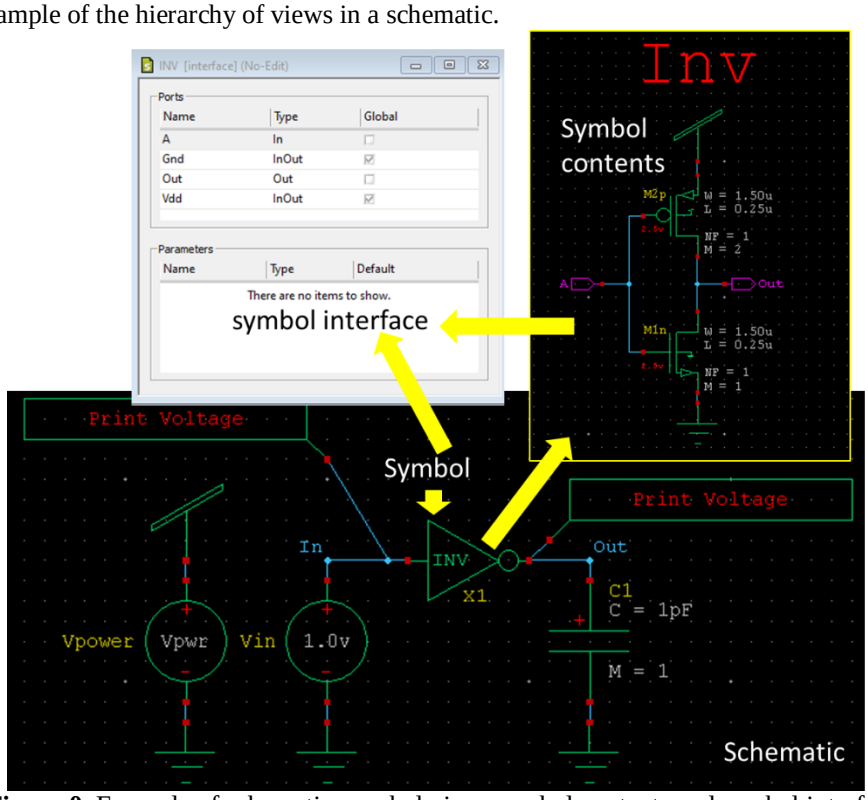

Figure 0. Example of schematic, symbol view, symbol contents and symbol interface.

Tanner's schematic and simulation tools are fully integrated to allow AC, DC, or transient analysis of the design, with interactive setup, simulation, and post-processing. The three components of this process are S-Edit, T-Spice, and W-Edit, respectively. The setup stage corresponds to entering commands and information which describe the type of simulation (DC, AC, transient, etc.), and establish the simulator options and output. In the design export and simulation stage, S-Edit generates a SPICE file (a netlist) from the design. Then, T-Spice simulates the SPICE file to create a probing data file with voltage and current values for each node and device in the design, and for each analysis specified in the SPICE file. In the probing stage, W-Edit displays traces from the probe data file corresponding to an analysis type and a specific net or device selected in S-Edit. S-Edit can also annotate the schematic with node voltages and device terminal current and charges.

III. EXAMPLE 1. MOS SIMULATION

In order to get familiar with Tanner's software environment, a built-in example is investigated. From **File ➔ Open Design** the following example design is loaded: **C:\ Users\ user\ Documents\ Tanner EDA\ Tanner Tools v16.0\Designs\ MOS_Subthreshold\ MOS_Subthreshold.tanner**. This example simulates the IV characteristic of an nMOS transistor.

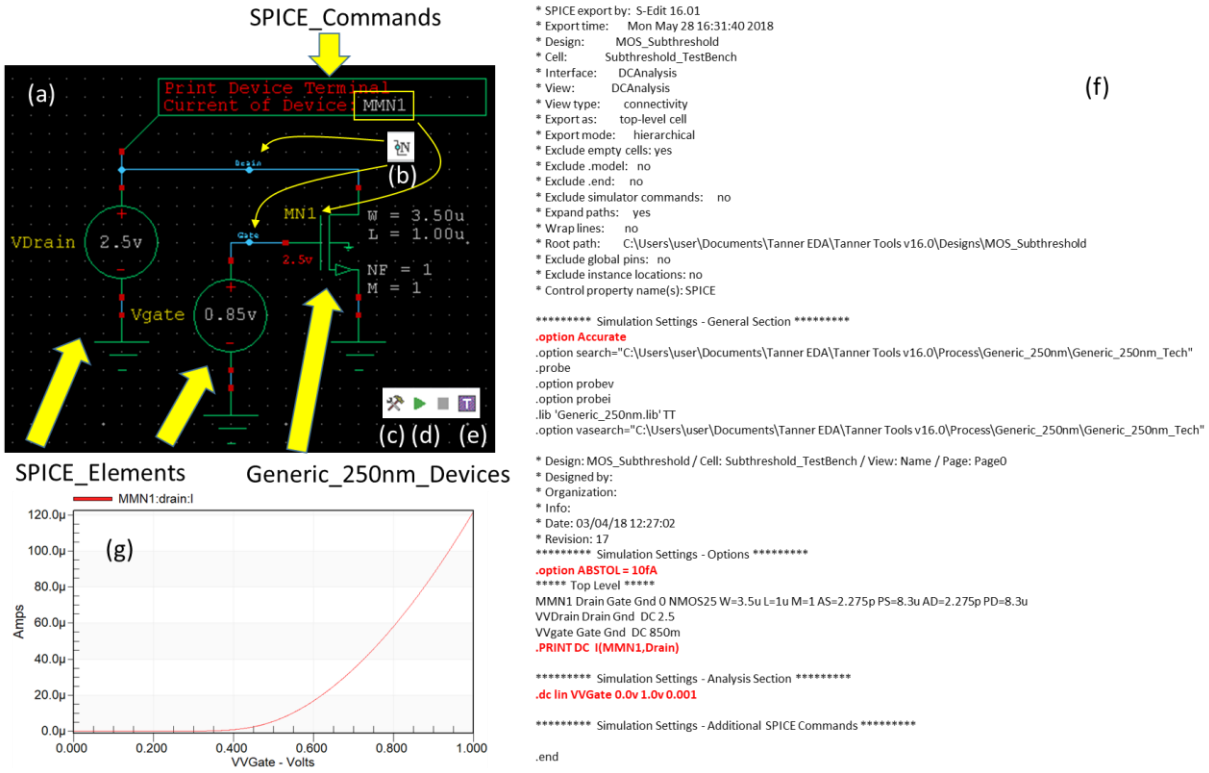

Figure 1. MOS_Subthreshold.tanner file simulation.

Figure 1(a) shows the design schematic. It consists of the nMOS symbol MN1 (located in the Generic_250nm_Devices library), two voltage sources Vdrain, and Vgate (located in SPICE_Elements library). Also a command from the SPICE_Commands library (the Print-Current) is used to return the current on the drain node of the transistor. The width W and length L of NM1 can be given by selecting the NM1 on the schematic and then entering the corresponding values on the Properties-window. The same procedure should be used to give the value on Vdrain and Vgate. Using the **Net Label button** (see **Fig. 1(b)**), nets can be given names.

The next step is to set the simulation properties. This is done by pressing the **Setup_SPICE_Simulation** button (**Fig. 1(c)**) and checking the DC Sweep Analysis. Then, set: Source or Parameter Name = VVGate, Start Value = 0.0v, Stop Value = 1.0v, Step = 0.001, and Sweep Type = lin. This way, in the background, a SPICE file is created (shown in **Fig. 1(f)**) containing all the graphical information, as well as the information for the simulation. This file can be open pressing the T-Spice button (**Fig. 1(e)**). Finally, pressing the the Start Simulation button (**Fig. 1(d)**), the T-Spice file is simulated and the W-Edit window is opened showing the corresponding IV characteristic of the device (**Fig. 1(g)**).

IV. EXAMPLE 2. STIMULUS SIMULATION

From **File ➔ Open Design** the following example design is loaded: **C:\Users\user\Documents\ Tanner EDA\ Tanner Tools v16.0\ Designs\ Stimuli\ Stimuli.tanner**.

Figure 2(a) shows a resistor with name R<1:9>. This name actually corresponds to 9 resistors, R<1>, to R<9> in parallel, each 2 kOhm. **Figure 3(a)**, shows the properties view of this device and **Fig. 3(b)** the corresponding T-Spice code. This is a compact way to create parallel resistor combinations. Notice the thicker net line originating from this resistor, indicative of a bus line and not just a single wire.

Figure 2(b) - (g) shows several voltage sources. Each one represent a two-terminal ideal voltage supply. T-Spice provides exponential, pulse, piecewise linear, frequency modulated, sinusoidal, and customizable (vectored) waveforms. Voltage sources whose waveform is described using an expression can be created using the e-element [**8**] with an expression and the time() function. Voltage sources V1 (**Fig. 2(b)**, **Fig, 3(c), (d)**) and V2 (**Fig. 2(c)**, **Fig. 3(e), (f)**) are piecewise linear and are specified by a series of (time, voltage) pairs. Their SPICE definitions as shown next:

```
VV1 N<1> Gnd PWL(0ns 0v 100ns 0v 101ns 5v 300ns 5v 301ns 0v
+500ns 0v 680ns 5v 700ns 0v 880ns 5v 900ns 0v)
VV2 N<2> Gnd  PWL(0ns 0v 100ns 0v 101ns 1v 200ns 1v 201ns 2v
+300ns 2v 301ns 3v 400ns 3v 401ns 4v 500ns 4v 501ns 5v +600ns 5v
+601ns 4v 700ns 4v 701ns 3v 800ns 3v 801ns 2v 900ns 2v 901ns 1v)
```

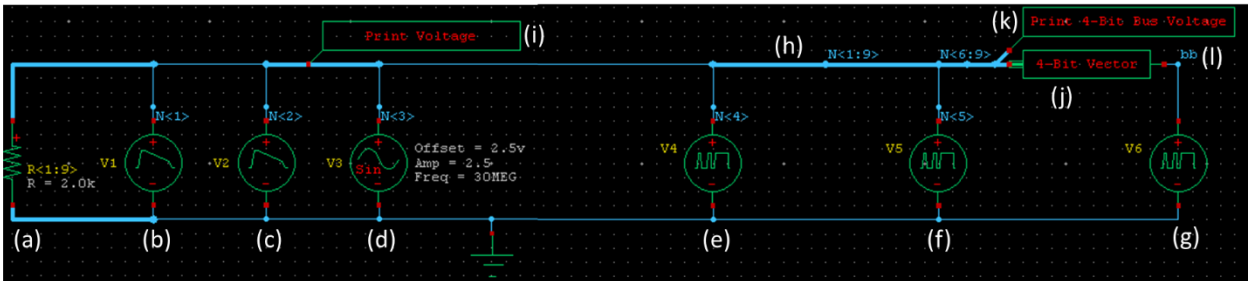

Figure 2. Schematic file: Stimuli.tanner

Voltage source V3 (**Fig. 2(d), Fig. 4(a), (b)**) is sinusoidal and for its specification requires an offset voltage (2.5 V), a peak voltage (2.5 V), a frequency (30 Meg), a delay time (100 ns), a damping factor (0 Hz), and a phase advance (0^o). Its SPICE definition is:

```
VV3 N<3> Gnd  SIN(2.5 2.5 30X 100n 0 0)
```

Voltage source V4 (**Fig. 2(e), Fig. 4(c), (d)**) has a vectored waveform, specified in SPICE as:

```
VV4 N<4> Gnd  BIT({01010 11011} PW=50n ON=5 OFF=0 RT=10n FT=30n )
```

{01010 11011} is a bit-pattern containing two bit-strings (01010 and 11011). PW=50 n is the pulse width, i.e. the time the wave is ramping up and on or dropping down and off. ON=5 is the on-voltage. OFF=0 is the off-voltage. RT=10 n is the rise time. FT=30 n is the fall-time.

Voltage source V5 (**Fig. 2(f), Fig. 4(e), (f)**) is also of vectored format:

```
VV5 N<5> Gnd  BIT({5(01010 5(1))} PW=10n ON=5 OFF=0 )
```

Here, V5 generates a repeating bit input. Two distinct patterns are given again, but no multiplier factors are included. The wave consists of two alternating patters: the first pattern contains five bits, the second is a single bit. The five-bit pattern is followed by five successive repetitions of the single-bit pattern, and this combination is repeated five times. The pulse width, and the on and off-voltages are also specified.

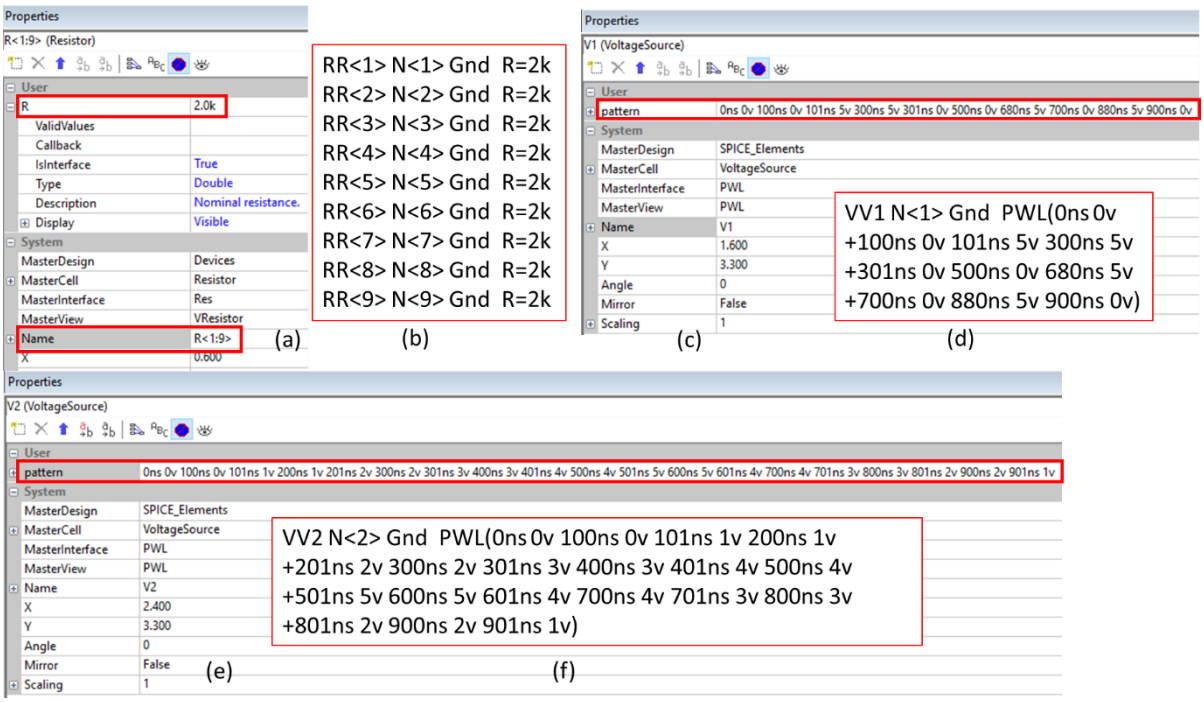

Figure 3. Properties and corresponding SPICE code for R<1:9>, V1, and V2.

The **.vector** command (**Fig. 5(a), (b)**) defines the bus waveform generated by voltage source V6 (**Fig. 2(g)**, and **Fig. 4(g), (h)**). The command assigns the bus a name (bb) and specifies by name the number of bits the bus waveform will have (four: N<6> through N<9>). The voltage source statement which contains the bus keyword, specifies waveforms with one or more patterns, along with pulse width and level information:

```
.VECTOR bb {N<6> N<7> N<8> N<9>} VV6 bb Gnd  BUS({50(Ah) 30(7d4) 20(1000)} PW=5n ON=5 OFF=0 )
```

The first pattern is `Ah (hex) = 1010` (binary). Thus, using the names given on the .vector command: `N<6>=1, N<7>=0, N<8>=1, and N<9>=0`. The pattern is repeated 50 times (that is, maintained for a time period equal to

the pulse width multiplied by 50). The next pattern is `7d4` – that is, `7`(decimal) = `111` (binary), or to four lower-order bits, `0111`. So `N<6>=0, N<7>=1, N<8>=1,` and `N<9>=1`. The pattern is repeated 30 times. The last pattern is `1000`(binary), so `N<6>=1, N<7>=0, N<8>=0,` and `N<9>=0`. The pattern is repeated 20 times.

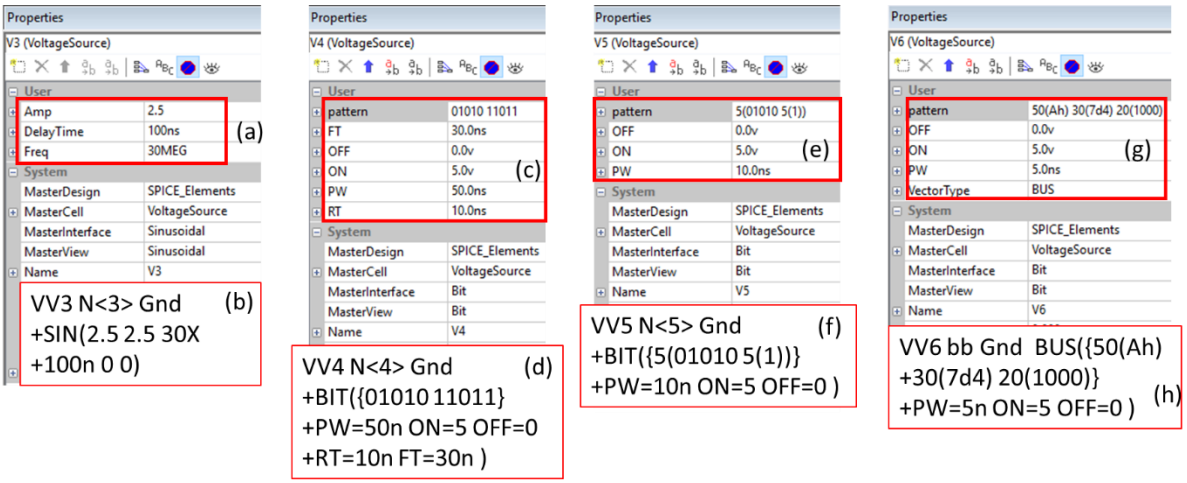

Figure 4. Properties and corresponding SPICE code for V3, V4, V5, and V6.

All sources are in library SPICE_Elements, and the resistor is in library Devices. Print-Voltage and Print-4-Bit Bus-Voltage commands are in library SPICE_Commands. 4-Bit-Vector is in library SPICE_Elements.

Nets within a design can be labeled in order to make the design more readable and to indicate connections within a cell. Connections formed by net labels exist only within a particular cell. When different cells contain nets with the same name, those nets are generally unconnected, with the important exception of global nets. S-Edit supports buses, arrays and net bundles. A net is the fundamental single unit of connection. A bus is a set of connections with the same name, plus a numerical identifier and an increment. Similarly, an array is a set of ports within an instance, with the same name plus a numerical identifier and an increment that defines related multiple connection points. A bundle is a collection of both nets and buses, where the nets do not need to share a name with the buds. A port bundle is a port defined to accept more than one wire. It is defined by the same numerical identifier and incrementing syntax that defines a bus. Buses in S-Edit can be grouped or nested to any required depth. If multiple buses with different dimensions connect to a single wire, there is the option to treat this as multiple devices in parallel or to treat it as an error.

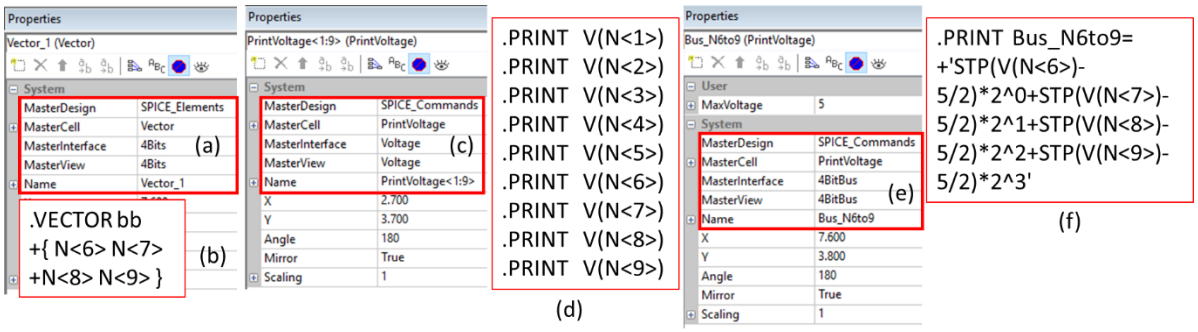

Figure 5. Properties and SPICE code for Vector_1, PrintVoltage<1:9>, and Bus_N6to9.

In **Fig. 2** there is the net-label bb (**Fig. 2(l)**), `N<1>, N<2>, N<3>, N<4>, N<5>, N<1:9>` (**Fig. 2(h)**), and `N<6:9>`. Nets `bb`, and `N<1>` to `N<5>` are simple nets. Net `N<1:9>` creates a bus of 9 bits wide with nets `N<1>` , …, `N<9>`. Net `N<6:9>` creates a bus 4-bits wide with nets `N<6>,…, N<9>`.

Command Print Voltage (**Fig. 2(i), Fig. 5(c), (d)**), when applied to a bus-net, prints all the corresponding node voltages. Its output property defined in its symbol view is:

```
[expr {[string equal $Enabled "Disabled"] ? "*" : "" }]
.PRINT ${Analysis} [se $PrintName {$PrintName=}]V(%{Vprint})
```

In the current example it is evaluated to:

```
.PRINT    V(N<1>)
.PRINT    V(N<2>)
.PRINT    V(N<3>)
.PRINT    V(N<4>)
```

```
.PRINT    V(N<5>)
.PRINT    V(N<6>)
.PRINT    V(N<7>)
.PRINT    V(N<8>)
.PRINT    V(N<9>)
```

Command Print-4-Bit Bus Voltage (**Fig. 2(k), Fig. 5(e), (f)**), when applied to a bus-net, prints all the corresponding node voltages. Its output property defined in its symbol view is:

```
[expr {[string equal $Enabled "Disabled"] ? "*" : "" }]
.PRINT ${Analysis} ${Name}='STP(V(%{Vprint<0>})-${MaxVoltage}/2)*2^0+STP(V(%{Vprint<1>})-
${MaxVoltage}/2)*2^1+STP(V(%{Vprint<2>})-${MaxVoltage}/2)*2^2+STP(V(%{Vprint<3>})-
${MaxVoltage}/2)*2^3'
```

In the current example it is evaluated to:

```
.PRINT Bus_N6to9= 'STP(V(N<6>)-5/2)*2^0+STP(V(N<7>)-5/2)*2^1+STP(V(N<8>)-5/2)*2^2+STP(V(N<9>)-
5/2)*2^3'
```

Any number in a command or statement may be replaced by an algebraic expression. Expressions must conform to the following conventions:

a) They must be enclosed by single quotes (’),

b) They may span several lines. The plus sign (+) is ignored if it appears in the first column of a continued expression,

c) They may include comments that do not interrupt numeric values or parameter names. Expressions may involve any valid combination of numbers, parameters, operations, algebraic functions, and user defined functions. T-Spice evaluates expressions according to a standard mathematical operator precedence. STP(expression) is a built-in function which evaluates to 0 if expression is negative, and 1 otherwise.

The output property of the 4-Bit Vector is:

```
.VECTOR %{VectorNode} { %{V<0:3>} }
```

and in the current example is evaluated to:

```
.VECTOR bb { N<6> N<7> N<8> N<9> }
```

Spice simulation setup is for Transient/Fourier Analysis with Stop Time of 1 us and Maximum Time Step of 1 ns. Simulation results are seen in the graph of **Fig. 6**. The corresponding T-Spice file of the previous schematic of **Fig. 2** is presented in **Listing 1**.

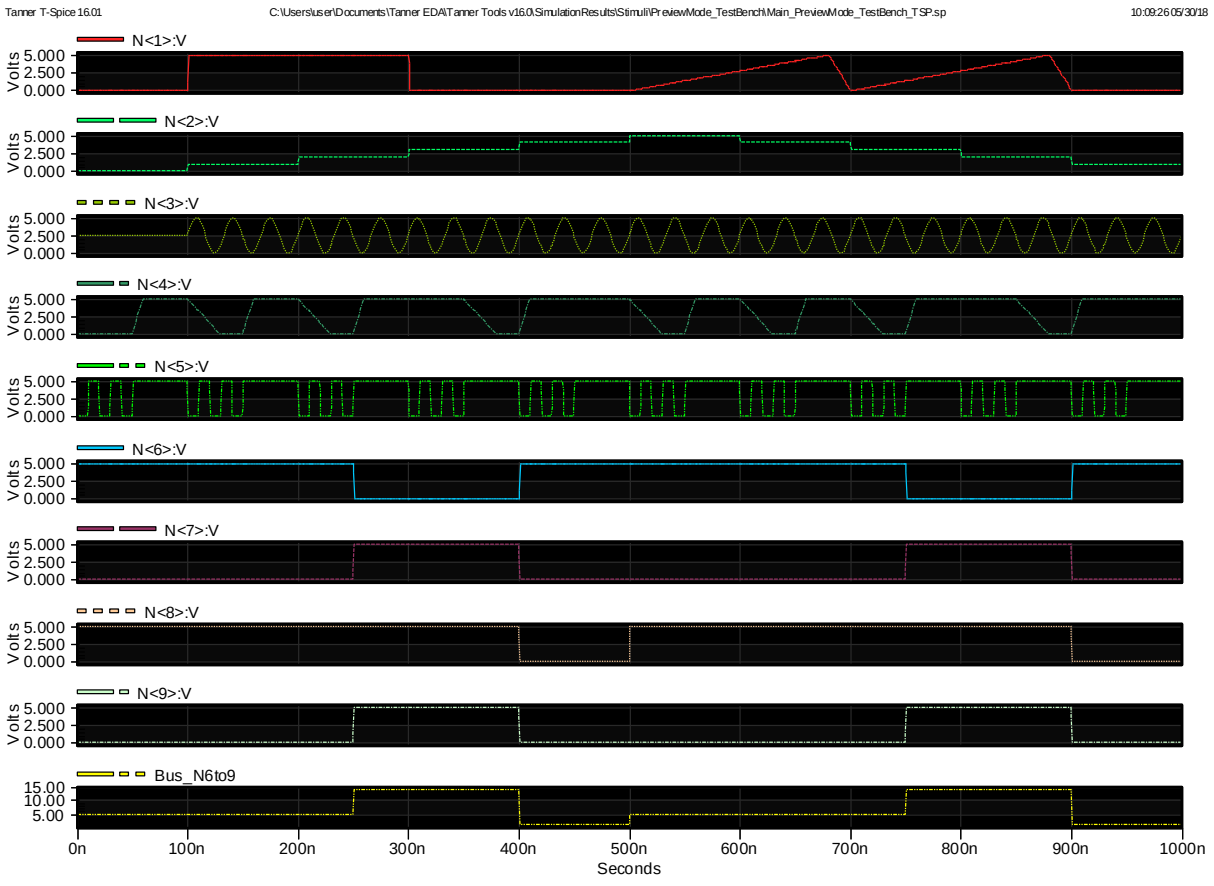

Figure 6. Transient analysis results from the simulation of the stimuli-schematic file.

Listing 1. T-Spice representation of the schematic file: Stimuli.tanner.

```
********* Simulation Settings - General Section *********
.option search="C:\Users\user\Documents\Tanner EDA\Tanner Tools
v16.0\Process\Generic_250nm\Generic_250nm_Tech"
.lib 'Generic_250nm.lib' TT
.option vasearch="C:\Users\user\Documents\Tanner EDA\Tanner Tools
v16.0\Process\Generic_250nm\Generic_250nm_Tech"

***** Top Level *****
RR<1> N<1> Gnd  R=2k
RR<2> N<2> Gnd  R=2k
RR<3> N<3> Gnd  R=2k
RR<4> N<4> Gnd  R=2k
RR<5> N<5> Gnd  R=2k
RR<6> N<6> Gnd  R=2k
RR<7> N<7> Gnd  R=2k
RR<8> N<8> Gnd  R=2k
RR<9> N<9> Gnd  R=2k
VV1 N<1> Gnd PWL(0ns 0v 100ns 0v 101ns 5v 300ns 5v 301ns 0v 500ns
+    0v 680ns 5v 700ns 0v 880ns 5v 900ns 0v)
VV2 N<2> Gnd  PWL(0ns 0v 100ns 0v 101ns 1v 200ns 1v 201ns 2v 300ns
+2v 301ns 3v 400ns 3v 401ns 4v 500ns 4v 501ns 5v 600ns
+5v 601ns 4v 700ns 4v 701ns 3v 800ns 3v 801ns 2v 900ns 2v 901ns 1v)
VV4 N<4> Gnd  BIT({01010 11011} PW=50n ON=5 OFF=0 RT=10n FT=30n )
VV5 N<5> Gnd  BIT({5(01010 5(1))} PW=10n ON=5 OFF=0 )
VV6 bb Gnd  BUS({50(Ah) 30(7d4) 20(1000)} PW=5n ON=5 OFF=0 )
VV3 N<3> Gnd  SIN(2.5 2.5 30X 100n 0 0)
.VECTOR bb { N<6> N<7> N<8> N<9> }
.PRINT   V(N<1>)
.PRINT   V(N<2>)
.PRINT   V(N<3>)
.PRINT   V(N<4>)
.PRINT   V(N<5>)
.PRINT   V(N<6>)
.PRINT   V(N<7>)
.PRINT   V(N<8>)
.PRINT   V(N<9>)
.PRINT   Bus_N6to9='STP(V(N<6>)-5/2)*2^0+STP(V(N<7>)-5/2)*2^1+STP(V(N<8>)-
5/2)*2^2+STP(V(N<9>)-5/2)*2^3'

********* Simulation Settings - Analysis Section *********
.tran/Preview 1ns 1us

.end
```

V. EXAMPLE 3. CMOS INVERTER SIMULATION

In this section, the CMOS inverter cell is examined in detail. It is the entry point circuit design of all introductory course to VLSI design. In a new schematic, first the library Generic_250nm_LogicGates should be loaded which contains the INV symbol view of the inverter (**Fig. 7**). The libraries entered have symbols for nMOS and pMOS transistors. However, all non-linear components such as MOS transistors, require a model to describe their behavior. Simply entering a MOS symbol in the schematic, will not make SPICE able to know what to do with it, since each MOS transistor fabricated in a different technology behaves differently. The connection between the symbol and the appropriate model-file is done in the SPICE setup dialog presented in detail below.

Before running the T-SPICE simulator, some basic settings should be defined related to the type of simulation to be performed and the outputs required from the simulation. These values are entered in the Setup-SPICE- Simulation-dialog (**Fig. 8**). Note that simulations are performed on the active cell, as opposed to an entire design.

All fields in the Setup-SPICE-Simulation-dialog are evaluated. This allows customization, such as setting the output file name based on the name of the cell being simulated. Note that if the Simulation- Results-Folder-field is left empty, output files are considered temporary and are deleted as the end of the S-Edit session. The default location for output files is the directory indicated in the SPICE-File-Name-field. Both S-Edit and T-SPICE have fields that control which simulation results are saved and whether or not simulation results are overwritten. If there is a value in these fields in S-Edit:Setup_Spice-Simulation, and the simulation is initiated form S-Edit, the S-Edit settings are used. However, if the simulation is from T-Spice and the Keep-all-simulation- results-option is checked, the T-Spice setting will override the S-Edit setting.

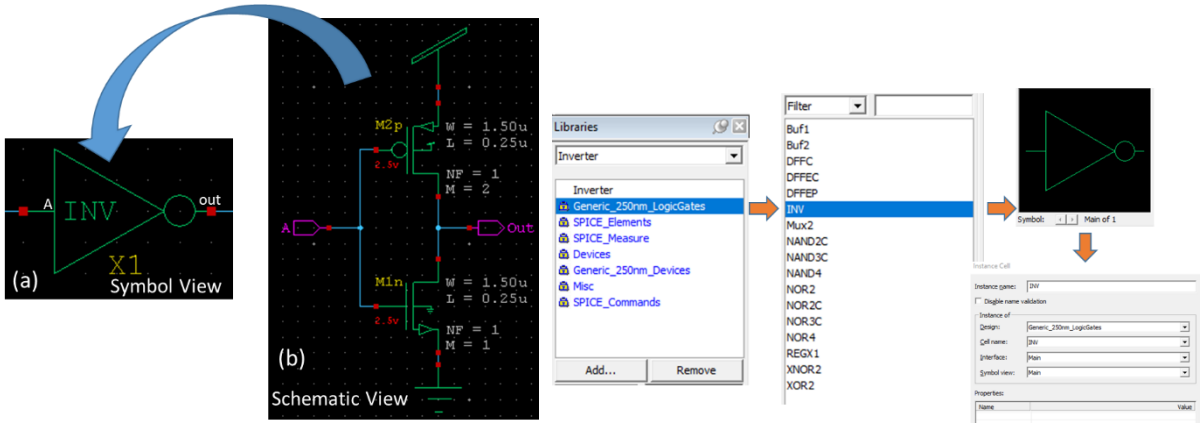

Figure 7. The inverter cell. (a) Symbol view, (b) Schematic view. INV cell is contained in Library Generic_250nm_LogicGates. So, in order to use it, this library should be added to the referenced libraries of the current design. (c) Having INV selected and pressing "i" while the cursor is in the design area, the "Instance Cell" dialog is brought-up and the cell can be placed in the new design.

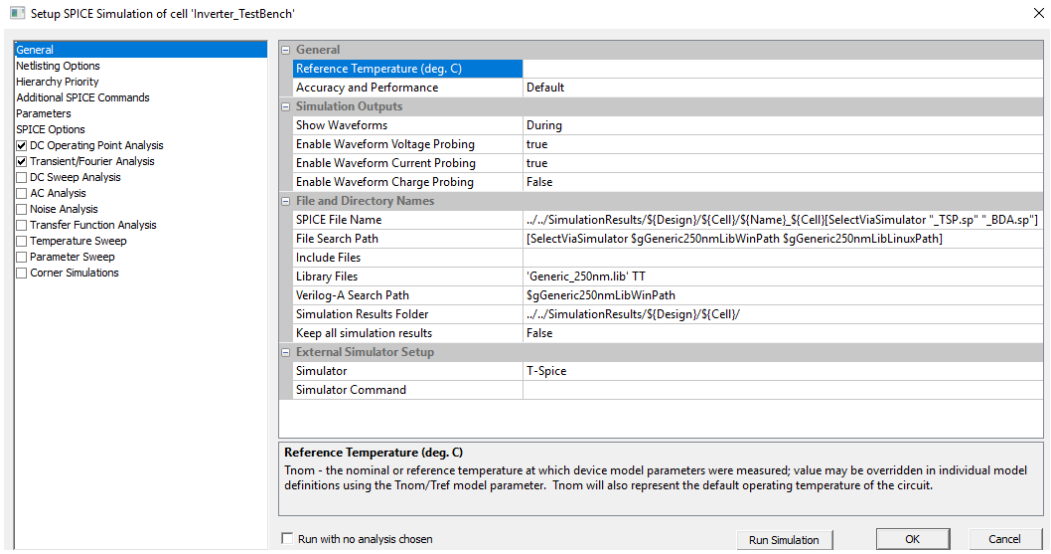

Figure 8. General SPICE simulation settings overview.

Both S-Edit and T-Spice have fields for library, include, and Verilog-A files (**Fig. 8**). However, these in S-Edit are written to the netlist, whereas those in T-Spice are written to a file header.sp that is called by the netlist.

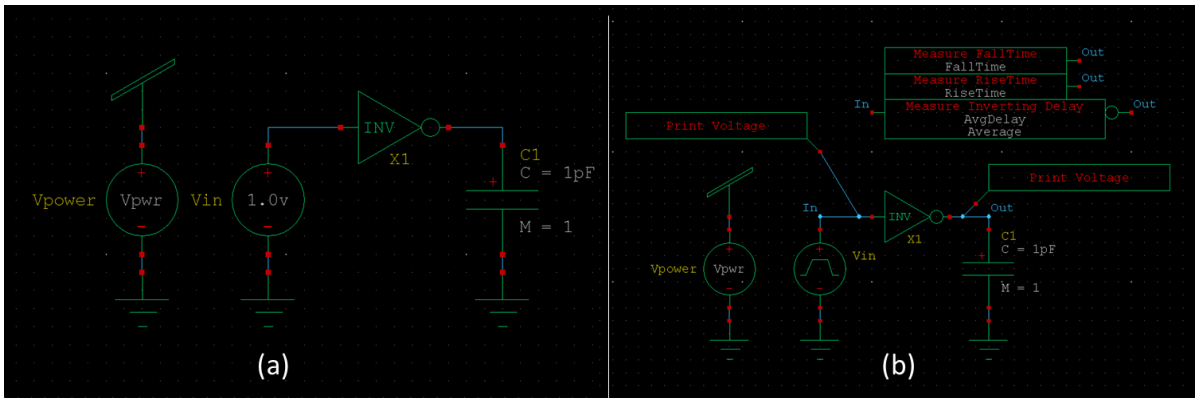

Figure 9. Circuit designs for (a) operating point analysis and DC analysis (b) transient analysis and Monte Carlo analysis.

The first simulation example is the determination of the operational point for the circuit of **Fig. 9(a)**. The sources Vpower and Vin are found within SPICE_Elements library, INV within Generic_250nm_LogicGates and the capacitor C within the Devices library. The SPICE setup requires checking the option DC-Operating-Point-Analysis. This setup file creates a SPICE file (with .sp extension) containing the circuit description as seen in **Listing 2**. A small portion of the result-file (with .op extension) is seen in **Listing 3**.

Listing 2. SPICE file for circuit of **Fig. 9(a)** for operating point analysis.

```
.subckt INV A Out Gnd Vdd
MM1n Out A Gnd 0 NMOS25 W=1.5u L=250n M=1 AS=975f PS=4.3u AD=975f PD=4.3u
MM2p Out A Vdd Vdd PMOS25 W=1.5u L=250n M=2 AS=975f PS=4.3u AD=562.5f PD=2.25u
.ends
.param Vpwr = 2.5v
XX1 N_1 N_2 Gnd Vdd INV
CC1 N_2 Gnd  1p
Vin N_1 Gnd  DC 1
VVpower Vdd Gnd  DC Vpwr
.op
.end
```

Listing 3. Partial contents of *.op results file for circuit of **Fig. 9(a)** for operating point analysis.

```
v(N_1) =   1.0000e+000
v(N_2) =   2.1518e+000
v(Vdd) =   2.5000e+000
i1(VVin) =  -0.0000e+000
i2(VVin) =   0.0000e+000
i1(VVpower) =  -1.7735e-004
i2(VVpower) =   1.7735e-004
```

Figure 9(b) shows the circuit that will be used for the transient analysis simulation. It contains several commands from the SPICE_Commands library, specifically Print-Voltage (2 instances), Measure-FallTime, Measure-RiseTime and Measure-Inverting Delay. The last command requires input and output, taken from the "In" and "Out" nets of the schematic. For this correspondence the "**Net Label**" function should be used from the menu bar of the main screen in S-Edit. **Figure 10** shows the user specified values for the setting up of Vin source. **Figure 11** shows the setting up of the SPICE commands. In the setup of SPICE simulation parameters for this design, both the DC operating point analysis, and Transient/Fourier analysis options should be selected. **Figure 12**, shows input / output voltages vs. time, where the output rise and fall delays are observed.

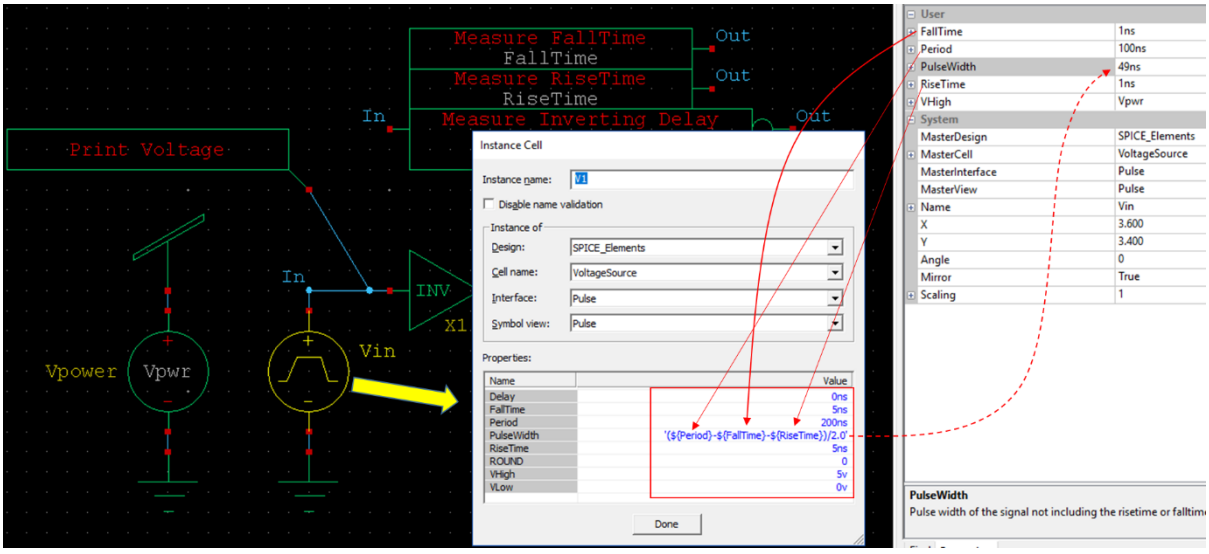

Figure 10. Instance placement of a Pulse voltage source. PulseWidth-field values is obtained in relation to Period, FallTime and RiseTime.

The circuit of **Fig. 9(a)**, will be used now for DC sweep analysis, The DC-Sweep-Analysis dialog permits the definition of up to three levels of sweeping for DC Analysis. The CD-Sweep-Analysis option should be checked in the SPICE setup dialog in S-Edit. DC sweeping is also referred to as DC-transfer-analysis. This analysis computes the DC states of a circuit while a voltage or current source is swept over a given interval. In addition to sweeping current and voltage source values, parameter values may be swept in order to yield a DC curve as a function of the parameter value. Parameters are defined in the Parameters-section of the SPICE setup dialog, include the intrinsic temperature-value, which is the operating temperature of the circuit. As shown in **Fig. 13**, Source-1 is swept for each value of Source-2. In the current example, Source-2 is Vpwr which is swept from 1.75 V to 3.25 V in linear steps of 0.375 V. For each of the values of Vpwr, Source-1, the VVin, is swept from 0 V to 2.5 V in linear steps of 0.02 V. The corresponding SPICE input file is seen in **Listing 6** and the resulting transfer graph in **Fig. 14**.

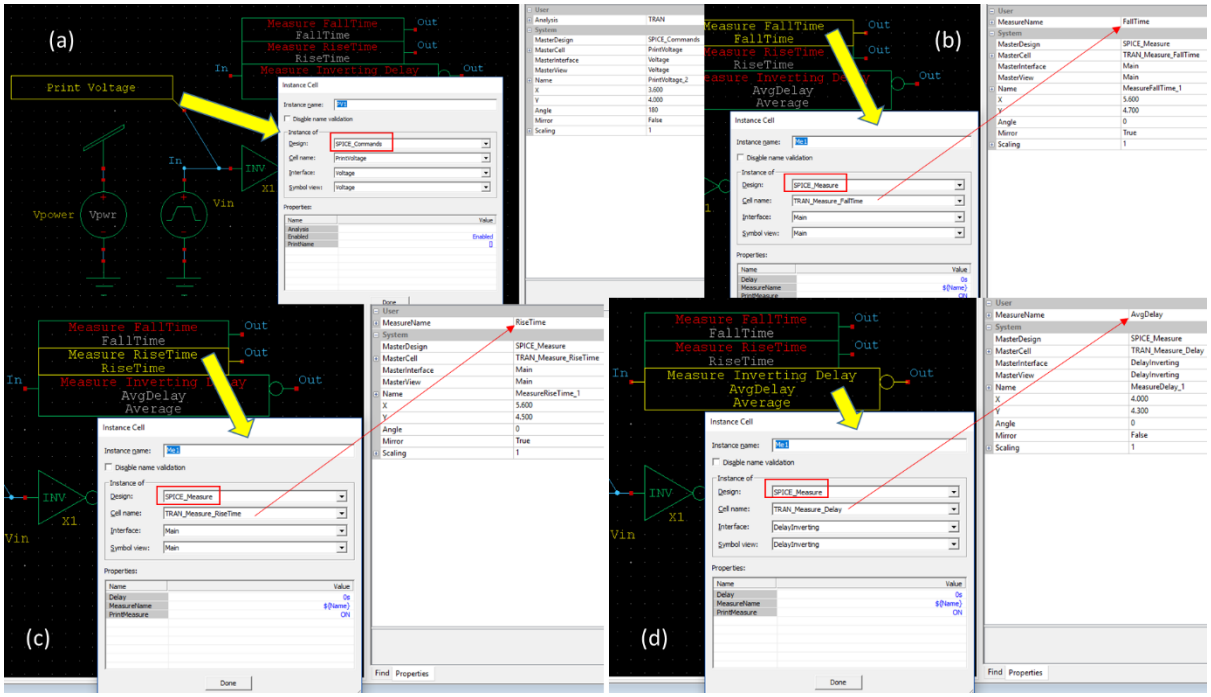

Figure 11. Instances of SPICE Commands and Measurements. (a) Print Voltage, (b) Measure FallTime, (c) Measure RiseTime, (d) Measure Inverting Delay.

Listing 4. SPICE input file for circuit of **Fig. 9(b)** for transient analysis.

```
.probe
.option probev
.option probei
.subckt INV A Out Gnd Vdd
MM1n Out A Gnd 0 NMOS25 W=1.5u L=250n M=1 AS=975f PS=4.3u AD=975f PD=4.3u
MM2p Out A Vdd Vdd PMOS25 W=1.5u L=250n M=2 AS=975f PS=4.3u AD=562.5f PD=2.25u
.ends
.param Vpwr = 2.5v
XX1 In Out Gnd Vdd INV
CC1 Out Gnd  1p
VVpower Vdd Gnd  DC Vpwr
VVin In Gnd  PULSE(0 Vpwr 0 1n 1n 49n 100n)

.PRINT TRAN V(In)
.PRINT TRAN V(Out)

.MEASURE TRAN RiseDelay_MeasureDelay_1 TRIG v(In) VAL='(Vpwr-0)*50/100+0' TD='0' RISE='1'
TARG v(Out) VAL='(Vpwr-0)*50/100+0' TD='0' FALL='1' OFF

.MEASURE TRAN FallDelay_MeasureDelay_1 TRIG v(In) VAL='(Vpwr-0)*50/100+0' TD='0' FALL='1'
TARG v(Out) VAL='(Vpwr-0)*50/100+0' TD='0' RISE='1' OFF

.MEASURE TRAN AvgDelay PARAM='(RiseDelay_MeasureDelay_1+FallDelay_MeasureDelay_1)/2.0' ON

.MEASURE TRAN RiseTime TRIG v(Out) VAL='(Vpwr-0)*10/100+0' TD='0' RISE='1' TARG v(Out)
VAL='(Vpwr-0)*90/100+0' TD='0' RISE='1' ON
```

```
.MEASURE TRAN FallTime TRIG v(Out) VAL='(Vpwr-0)*90/100+0' TD='0' Fall='1' TARG v(Out)
VAL='(Vpwr-0)*10/100+0' TD='0' FALL='1' ON

.op
.tran 250p 300n
.end
```

Listing 5. SPICE output file for the various delays, for circuit of **Fig. 9(b)** for transient analysis.

```
Measurement result summary
   AvgDelay        =    1.5751n
   RiseTime        =    3.4098n
   FallTime        =    2.5976n
```

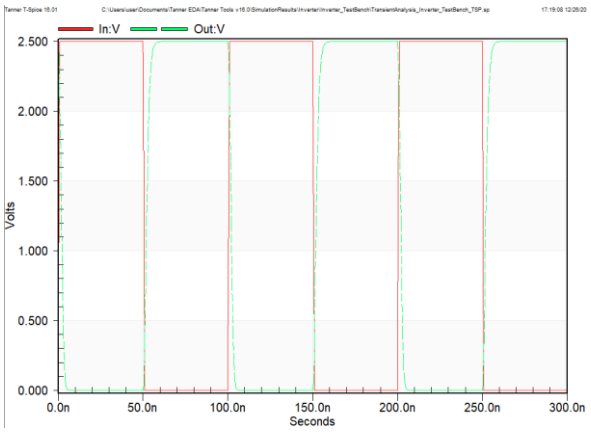

Figure 12. Input / Output voltages vs. time, showing the inverter function and the rise and fall delays.

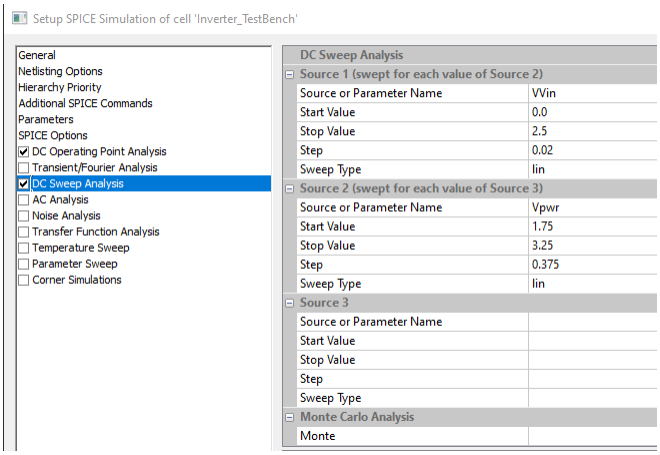

Figure 13. SPICE simulation setup dialog for the DC sweep analysis of the inverter in **Fig. 9(a)**.

Listing 6.

```
.probe
.option probev
.option probei
.subckt INV A Out Gnd Vdd
MM1n Out A Gnd 0 NMOS25 W=1.5u L=250n M=1 AS=975f PS=4.3u AD=975f PD=4.3u
MM2p Out A Vdd Vdd PMOS25 W=1.5u L=250n M=2 AS=975f PS=4.3u AD=562.5f PD=2.25u
.ends
.param Vpwr = 2.5v
XX1 In Out Gnd Vdd INV
CC1 Out Gnd  1p
VVin In Gnd  DC 1
VVpower Vdd Gnd  DC Vpwr
.PRINT DC V(In)
.PRINT DC V(Out)
.op
.dc lin VVin 0.0 2.5 0.02 SWEEP lin Vpwr 1.75 3.25 0.375
.end
```

Finally, the schematic of **Fig. 9(b)** will be used now for Monte Carlo analysis. From the SPICE simulation setup, in the Transient/Fourier analysis option, the settings seen in **Fig. 15**, produce the SPICE-file command **.tran 250ps 300ns sweep monte=200**, which performs 200 transient simulations as part of a Monte Carlo analysis. The keyword "**monte**" defines one of the many sweep options. For each of the 200 transient analyses, values are randomly chosen for circuit variables, which are assigned probability distributions. The SPICE input file commands for this analysis is seen in **Listing 7**. **Listing 8** shows the corresponding statistics and **Fig. 16** the corresponding statistical distribution for the average delay, fall time delay, and rise time delay.

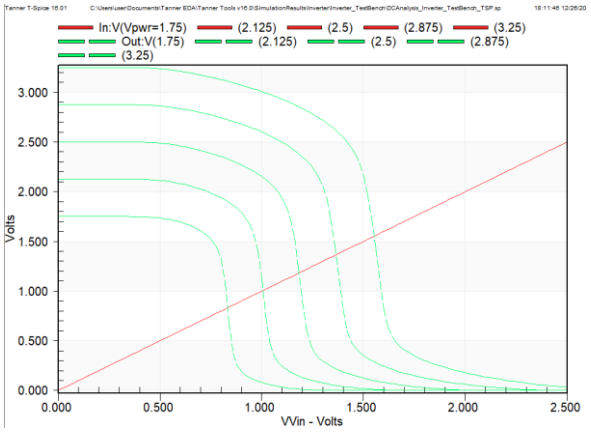

Figure 14. DC sweep analysis results for circuit of **Fig. 9(a)**.

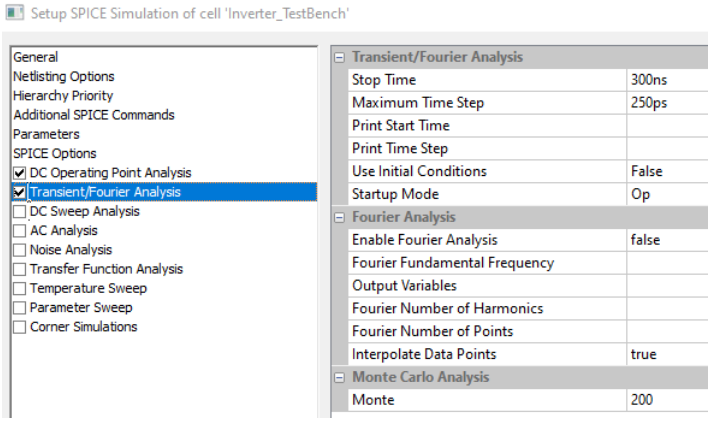

Figure 15. Setting up for Monte Carlo Analysis.

Listing 7. SPICE input file for Monte Carlo analysis of the inverter circuit **Fig. 9(b)**.

```
.subckt INV A Out Gnd Vdd
MM1n Out A Gnd 0 NMOS25 W=1.5u L=250n M=1 AS=975f PS=4.3u AD=975f PD=4.3u
MM2p Out A Vdd Vdd PMOS25 W=1.5u L=250n M=2 AS=975f PS=4.3u AD=562.5f PD=2.25u
.ends
.param Vpwr = 2.5v
.option modmonte = 1
XX1 In Out Gnd Vdd INV
CC1 Out Gnd  1p
VVpower Vdd Gnd  DC Vpwr
VVin In Gnd  PULSE(0 Vpwr 0 1n 1n 49n 100n)

.PRINT TRAN V(In)
.PRINT TRAN V(Out)

.MEASURE TRAN RiseDelay_MeasureDelay_1 TRIG v(In) VAL='(Vpwr-0)*50/100+0' TD='0' RISE='1'
TARG v(Out) VAL='(Vpwr-0)*50/100+0' TD='0' FALL='1' OFF

.MEASURE TRAN FallDelay_MeasureDelay_1 TRIG v(In) VAL='(Vpwr-0)*50/100+0' TD='0' FALL='1'
TARG v(Out) VAL='(Vpwr-0)*50/100+0' TD='0' RISE='1' OFF

.MEASURE TRAN AvgDelay PARAM='(RiseDelay_MeasureDelay_1+FallDelay_MeasureDelay_1)/2.0' ON

.MEASURE TRAN RiseTime TRIG v(Out) VAL='(Vpwr-0)*10/100+0' TD='0' RISE='1' TARG v(Out)
VAL='(Vpwr-0)*90/100+0' TD='0' RISE='1' ON
```

```
.MEASURE TRAN FallTime TRIG v(Out) VAL='(Vpwr-0)*90/100+0' TD='0' Fall='1' TARG v(Out)
VAL='(Vpwr-0)*10/100+0' TD='0' FALL='1' ON

.op
.tran 250ps 300ns sweep monte=200
.end
```

Listing 8. Monte Carlo Measurement result statistics.

	AvgDelay	RiseTime	FallTime
Minimum	1.4312n	3.0117n	2.4396n
Maximum	1.6868n	3.7612n	2.8338n
Mean	1.5759n	3.4091n	2.5945n
Avgdev	31.7032p	108.9436p	46.8844p
Variance	1.6327e-021	19.0772e-021	3.9212e-021
StdDev	40.4072p	138.1201p	62.6195p

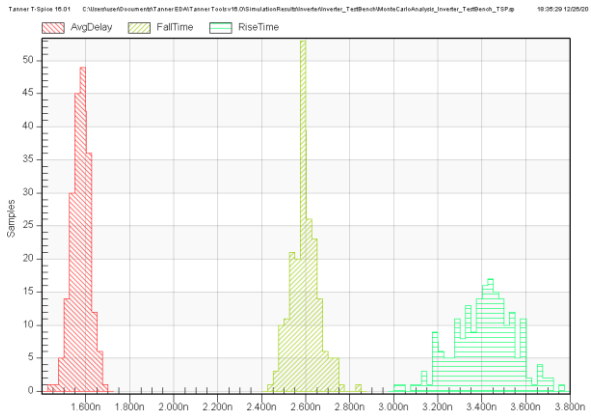

Figure 16. Monte Carlo analysis result distributions for average delay, fall and rise time.

VI. CONCLUSION

In this work an introduction to the use of Tanner's EDA tools of S-Edit and T-Spice was presented. The aim was to help the beginners in VLSI circuit design to get a first impression on simple but informative design examples. Subsequent articles could follow with more detailed analysis of the various tools and more specific circuit design applications.

REFERENCES

[1] G. P. Patsis, *Process-simulation-flow and metrology of VLSI layout fine-features*, IOSR Journal of VLSI and Signal Processing, vol. 7, no. 6, 2017, 23-28.
[2] G. P. Patsis, *VHDL-AMS macromodels of MOSFET. Consideration of gate length variability and single-electron-transistors*, IOSR Journal of VLSI and Signal Processing , vol. 7, no. 6, 2017, 29-33
[3] G. P. Patsis, *Basic topologies of MOS single-stage amplifiers. DC analysis for maximum input-voltage swing and amplification*, IOSR Journal of VLSI and Signal Processing, vol. 8, no. 1, 2018, 47-59.
[4] G. P. Patsis, *MOSFET EKV Verilog-A model implementation in Genesys*, IOSR Journal of VLSI and Signal Processing, vol. 8, no. 2, 2018, 73-86.
[5] G. P. Patsis, *Educational Introduction to VLSI Layout Design with Microwind*, IOSR Journal of VLSI and Signal Processing, vol. 8, no. 5, 2018, 18-29.
[6] G. P. Patsis, *Educational Introduction to CMOS Circuit Design with Texas Instrument ANalyzer (TINA)*, IOSR Journal of VLSI and Signal Processing, vol. 8, no. 6, 2018, 39-48.
[7] Tanner EDA, S-Edit User Guide contained in software's help directories. (S-Edit 16 User Guide.pdf) .
[8] Tanner EDA, Applications and Examples User Guide contained in software's help directories. (Tanner Tools Tutorial.pdf, Tanner Tools Examples Guide.pdf).

George P. Patsis. "Educational Introduction to VLSI Circuit Design and Simulation with Tanner-EDA Tools." *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*, vol. 11, no. 1, 2021, pp. 09-21.