# Incorporating Ngspice circuit models into GGI\_TLM

Contents

[Incorporating Ngspice circuit models into GGI\_TLM 1](#_Toc14084383)

[Introduction 2](#_Toc14084384)

[Theory 3](#_Toc14084385)

[Filtering the interaction of GGI\_TLM and Ngspice processes 5](#_Toc14084386)

[GGI\_TLM – Ngspice linking process 6](#_Toc14084387)

[Voltage references in GGI\_TLM-Ngspice models 6](#_Toc14084388)

[Implementation 7](#_Toc14084389)

[Interaction between GGI\_TLM and Ngspice 7](#_Toc14084390)

[Model Preparation 9](#_Toc14084391)

[Ngspice port and node numbering 9](#_Toc14084392)

[GGI\_TLM\_create\_PCB\_simulation\_model 10](#_Toc14084393)

[PCB Simulation Specification file format: 16](#_Toc14084394)

[Known problems 18](#_Toc14084395)

[Post processing 19](#_Toc14084396)

[Examples 21](#_Toc14084397)

[1. Transmission line with non-linear load. 21](#_Toc14084398)

[2. Buck converter model 28](#_Toc14084399)

[3. Buck converter model including SACAMOS cable model and LISN for conducted emissions analysis 39](#_Toc14084400)

[References 40](#_Toc14084401)

# Introduction

GGI\_TLM is open source software for the time domain simulation of Electromagnetic fields [1] with applications including EMC, Antenna design and scattering analysis. The software includes the ability to include thin wires, including shielded cables, frequency dependent materials and thin layers into models. In order to enable GGI\_TLM to simulate circuits consisting of lumped components, including active devices the circuit simulation tool Ngspice [2] has been linked into the simulation process. This enables 3D models of PCBs to be created and simulated in the time domain which allows both conducted and radiated emissions to be assessed. This capability allows dc-dc converters to be simulated for example.

In order for a circuit simulation to be performed the layout of the circuit must be specified. This is most commonly available in the gerber format [3] which is used for design and manufacture of PCBs. In addition, the circuit topology and spice component models must be specified.

There are a number of published approaches to the embedding of Spice models in TLM [4]-[8] however in order to simplify the interface between GGI\_TLM and Ngspice, lumped components are placed on faces between TLM cells in the problem space.

In order to aid the model creation process some tools have been developed which bring together the required information and automatically creates the appropriate input files for GGI\_TLM. These are:

**GGI\_TLM\_create\_PCB\_simulation\_model**

This tool creates the main input file for GGI\_TLM by bringing together the following information and processes:

1. Specify the problem space dimensions
2. Specify the number of PCB layers to include in the model. For each layer, convert the gerber file to stl format and specify the position within the problem space to place the PCB layer
3. Specify the number of dielectric layers. For each layer specify the position in space of the dielectric and specify the material properties (i.e. give the name of the appropriate material file.)
4. Specify the number of vias to include and their positions
5. Specify the number of lumped components. For each lumped component:
6. Specify the number of ports
7. Provide the ngspice node numbers corresponding to each port
8. Provide the

**GGI\_TLM\_gerber\_to\_stl**

This tool converts files in gerber format to stl format (surface triangulation format) which allows the PCB layer geometries to be imported into GGI\_TLM.

In addition to these two tools dedicated to the creation of GGI\_TLM models, two other processes have been developed which are directly relevant to this work. These processes enable the simulation of lumped components at high frequency i.e. at frequencies at where they can no longer be considered to be ideal and parasitic elements contribute to or even dominate the component response measured at its terminals. It is assumed that the real and imaginary parts of impedance as a function of frequency are available from a component measurement. The first process is to determine a rational function fit to this impedance frequency response. The vector fitting process is used here and an implementation can be found in [9]. Once the rational function impedance representation has been obtained the network synthesis process can be applied to generate a passive circuit which reproduces the required impedance function [10]. These two procedures have been combined into a single process in [11].

# Theory

A 3D model of a circuit is a combination of lumped and distributed elements. Lumped circuit elements such as diodes, transistors etc. will be simulated using Ngspice and the distributed elements such as PCB tracks, heatsinks etc. will be simulated in 3D using GGI\_TLM. Since GGI\_TLM is based on an equivalent circuit principle the link between the TLM based circuit simulator and the Spice based circuit simulation is straightforward in principle.

The most basic example of the way in which circuit and TLM simulations may be combined, is with the embedding of a single lumped component into a TLM model for example a diode load on a wire. This is illustrated in Figure 1. In this model, the wire is represented in the 3D TLM model as short circuits in the link lines between the TLM nodes. The presence of the short circuit will be taken into account in the TLM model by locally modifying the connect procedure.

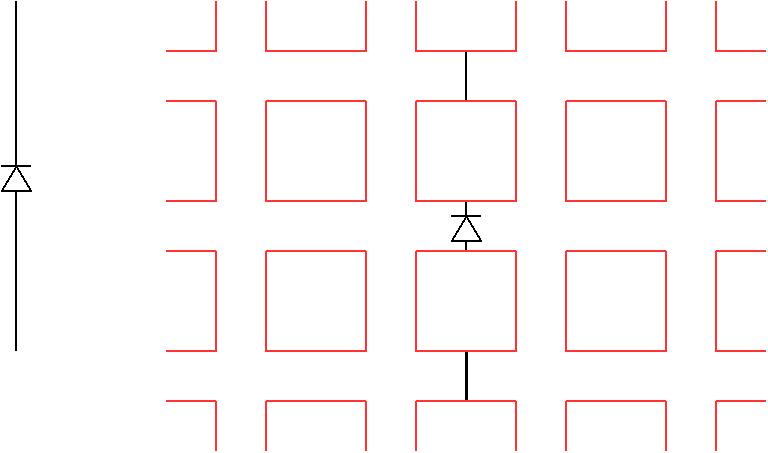


Figure 1. Diode at the centre of a conductor and the TLM model of this situation with the diode and conductors placed at cell faces.

The diode may also be placed on the link lines in a similar position only now the connect procedure must take into account the presence of the diode. In order to do this we may draw a Thevenin equivalent circuit for the connect process as seen in Figure 2.

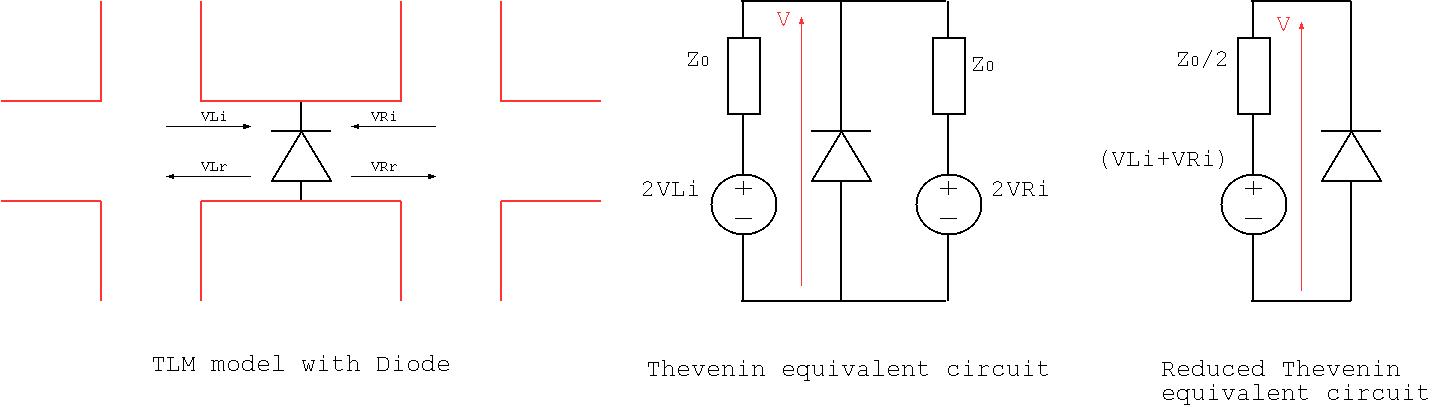


Figure 2. Example diode model and the Thevenin equivalent model of the TLM-NGspice link circuit

The central figure shows the Thevenin equivalent circuit of the two link lines interacting with the diode. These two Thevenin equivalent circuits may be combined to give the simple circuit on the right hand side of Figure 2 in which a resistance of Z0/2 is in series with a voltage source whose value depends on the voltage pulses incident on the diode from the link lines (VLi+VRi). This circuit may be solved using Ngspice for the diode voltage, V and hence the scattered voltages may be calculated from:

## Filtering the interaction of GGI\_TLM and Ngspice processes

The implementation of the GGI\_TLM-Ngspice link requires voltages to be exchanged between the Ngspice and GGI\_TLM processes i.e. GGI\_TLM must send the link line voltages, (VLi+VRi), to Ngspice, Ngspice is required to calculate the component voltage and return this to GGI\_TLM so that the scattered voltages from the component can be returned to the link lines. In this work we have implemented the TLM-Ngspice link using a low-pass filter with an adjustable parameter, α, to control the cutoff frequency.

This approach recognises the potential for the hybrid solution to become unstable whilst also bearing in mind the limited bandwidth of validity of the TLM solution due to dispersion issues. Typically the minimum wavelength is determined from the requirement for 10 cells per wavelength i.e. or )=1/(20 where c is the speed of light.

A simple first order low pass filter is implemented with the transfer function

A discrete time implementation may be derived using the bilinear transformation to the z-domain

Where z-1 indicates a delay of one timestep. The first order discrete time filter function is found to be implemented by the formula

Where

And

In practice the cutoff frequency parameter, α, is normalised to the TLM timestep, Δt such that α’=α/Δt. When α’=0 the filter passes everything unchanged i.e. it does not act as a filter at all. A value of α’=0.1 should be sufficient to prevent any high frequency oscillation and instability without introducing too much loss into the system.

## GGI\_TLM – Ngspice linking process

The voltage pulses incident on the component are available following the scattering process i.e. at time nΔt. The voltage pulses scattered from the component must be returned to GGI\_TLM for the next scattering process i.e. at (n+1) Δt thus we have a time period of Δt in which the GGI\_TLM – Ngspice link algorithm can utilise.

TLM is normally assumed to be based on the propagation of time synchronised impulses on a network of transmission lines. The Spice circuit simulation method on the other hand is based on a modified nodal analysis technique with a trapezoidal integration scheme and variable time stepping thus care must be taken in how TLM and Spice are interfaced; the propagation of impulses in non-linear and dispersive circuits is not possible. It is normally assumed that the ‘connect’ operation in TLM is instantaneous, and occurs at time (n+½)Δt however when we are coupling to a lumped circuit solution we may be required to modify this process in order to maintain stability of the coupled system. A number of different algorithms are discussed in [5] in which algorithms based on averaging and integration of the voltage calculated in Spice are proposed.

For convenience of the GGI\_TLM- Ngspice link, Ngspice is run at a timestep smaller than the TLM timestep.

The interaction of GGI\_TLM and Ngspice is summarised as follows:

Time=nΔt:

1. Scattering process at TLM nodes.
2. Calculate the Ngspice link voltage Vtlm=VLi+VLr
3. Apply the Low Pass Filter process to Vtlm as required to give Vtlm’
4. Set a breakpoint in Ngspice at (n+1)Δt and resume the Ngspice simulation using the voltage source value Vtlm’

Time=(n+½)Δt:

1. Connection process at TLM cell faces

Time=(n+1)Δt.

1. Transfer the component voltage, Vspice from Ngspice to GGI\_TLM
2. Apply the Low Pass Filter process to Vspice as required to give Vspice’
3. Calculate VLr=Vspice’-VLi, VRr=Vspice’-VRi

There is a complication that Ngspice can use variable timestepping and is therefore not guaranteed to use a specific timestep however with a sufficiently small Ngspice maximum timestep set, and the use of the Low Pass Filter process it is assumed that this will not lead to too much accumulated error. This issue does need to be studied in more detail though as for some applications it may become significant.

## Voltage references in GGI\_TLM-Ngspice models

In a circuit simulation in Ngspice for example, all voltages are referenced to a single node, node zero. Once the Ngspice circuit model is separated into a multi-port model linked by GGI\_TLM there is no common reference by which circuit voltages may be determined.

# Implementation

The implementation requires Ngspice to be compiled as a library which may be linked to GGI\_TLM. The first stage is to compile Ngspice as follows:

1. Download Ngspice (here Ngspice-30). Make sure that you download the file ngspice-30.tar.gz (the default download from the green button may be different on a windows machine)
2. cd ngspice-30
3. mkdir ngspice\_GGI\_TLM
4. cd ngspice\_GGI\_TLM
5. ../configure --with-ngshared --enable-xspice –enable-cider
6. make clean
7. make
8. sudo make install
9. cd ../..

Download GGI\_TLM then compile as follows:

1. cd GGI\_TLM/SRC
2. make clean
3. make NGSPICE=TRUE

The appropriate paths to the Ngspice libraries are set in the Makefile (Makefile\_GGI\_TLM)

Currently GGI\_TLM can only be run with the sequential version of GGI\_TLM.

There are some test cases in the directory GGI\_TLM/TEST\_DATA which demonstrate the GGI\_TLM – Ngspice link. These are:

NGSPICE\_DIODE\_TEST

PCB\_SIMULATION\_TEST

PCB\_SIMULATION\_TEST\_2

PCB\_SIMULATION\_TEST\_3

PCB\_PORT\_TEST\_1

PCB\_PORT\_TEST\_2

PCB\_CONVERTER\_MODEL

## Interaction between GGI\_TLM and Ngspice

Before the simulation starts the Ngspice circuit file is created from the template circuit file so as to include GGI\_TLM Thevenin equivalent circuit impedance values, timestep values and simulation time.

The circuit file must include the line:

Vbreak time\_node 0 DC 0.0

This voltage source is used to control the breakpoints in the Ngspice simulation. At each timestep a breakpoint is set so that Ngspice will stop after the simulation time is increased by Δt, the GGI\_TLM timestep. The breakpoint is set using the command ‘stop when time > v(time\_node)’, the voltage source connected between node 0 and time\_node is set to the breakpoint time, this eliminates the need to continuously set and remove breakpoint commands.

In the software Ngspice is run in a separate thread.

Following the scattering process the GGI\_TLM voltage pulses are updated in the Ngspice model using Ngspice commands of the form ‘alter Vtlm1 = 0.325’. Similarly the next breakpoint time is set with a command of the form ‘alter V(time\_node) = 1.02e-10’. Once the new voltages have been set the Ngspice simulation is restarted with the Ngspice command ‘bg\_resume’.

Once Ngspice stops again at the specified breakpoint the voltages on all nodes from 1 to 100 are returned to GGI\_TLM in an array (for this reason nodes between 1 and 100 should be reserved for nodes which interact with GGI\_TLM.) These node voltages are then used to calculate the voltage pulses scattered from the Ngspice component.

The interaction between Ngspice and GGI\_TLM may be controlled by two parameters in the GGI\_TLM input file.

**Ngspice\_timestep\_factor** sets the number of timesteps in Ngspice for each GGI\_TLM timestep. The default is set to 8.

Example:

**Ngspice\_timestep\_factor**

**4**

The second parameter is the low pass filter parameter, alpha. The default value is 0 (i.e. no low pass filter is applied to the ngspice link).

Example

**Ngspice\_LPF\_alpha**

**0.10**

# Model Preparation

The GGI\_TLM- Ngspice coupled simulation process required both a GGI\_TLM input file (**name.inp**) plus any associated material and cable specification files and in addition an Ngspice circuit file which must have the name **Spice\_circuit\_TEMPLATE.cir**.

As described in the Theory section above, Ngspice circuit elements are implemented in GGI\_TLM on surfaces between TLM cells they are therefore specified in the GGI\_TLM input file in the Surface\_material\_list packet.

The format to specify an Ngspice circuit element in the Surface\_material\_list is illustrated in the example below:

Surface\_material\_list

1 # Number of surface materials

1 # Surface material number

SPICE

1 #Ngspice port number

1 0 # Ngspice node numbers on + and – terminals of the surface

-y # port direction i.e. the direction of the component in 3D space

1 # number of surfaces (this should always be 1 for a SPICE surface

5 # Surface list (this is the number of the surface in the Surface\_list)

1 # Surface orientation list (this has no effect as the Spice interface is symmetrical)

## Ngspice port and node numbering

The Ngspice port number is an integer required to uniquely identify the GGI\_TLM Thevenin equivalent circuit in the Spice circuit file. This is shown in red in Figure 3. The Ngspice node numbers are the nodes associated with the + and – terminals of the Ngspice port, these are shown in blue in Figure 3. The Ngspice node numbers should be integers between 0 and 99 as currently these are the only nodes which support the transfer of voltage information between GGI\_TLM and Ngspice. It is recommended that node numbers below 100 are reserved for the link between GGI\_TLM and Ngspice and nodes which are internal to the Ngspice circuit whose voltages are not required to be communicated between processes should take numbers outside this range (>1000 in the example below).

The portion of the spice circuit file corresponding to figure 3 is shown below:

\*Voltage source with series resistance: equivalent circuit of TLM link

Vtlm1 1001 0 DC 0.0

Rtlm1 1001 1 #Z0\_TLM

(note that the source voltage is updated at every TLM timestep and the value of the resistance #Z0\_TLM is substituted with the appropriate value automatically.)

The order of the node specification is important as it specifies the orientation of the Ngspice circuit element model in the GGI\_TLM mesh. Reversing the Ngspice nodes reverses the orientation of the circuit element. This may also be achieved by reversing the port direction.

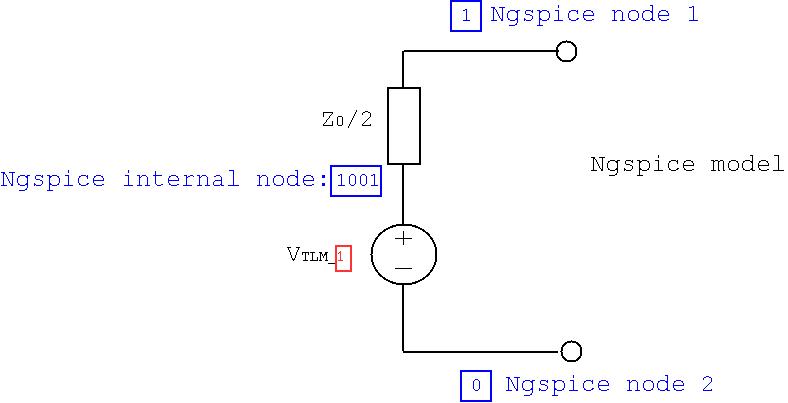


Figure GGI\_TLM-Ngspice link circuit showing port (red) and node (blue) numbering.

The port direction specified the direction of the component in the 3D TLM mesh. The component direction must be in the plane of the defined surface e.g. if the surface is normal to z then the possible port directions are +x, -x, +y and -y. Following the port direction comes the number of the geometric surface that the component model should be placed on the format of this information is the same as that for any other surface material. The surface should consist of a single TLM face i.e. Ngspice components cannot (yet) be distributed over multiple TLM faces (although this is possible in principle).

## GGI\_TLM\_create\_PCB\_simulation\_model

The **GGI\_TLM\_create\_PCB\_simulation\_model** software has been developed in order to aid the setting up of linked GGI\_TLM/ Ngspice models. The purpose of this is to simplify the setup of the simulation files for PCB simulations in GGI\_TLM.

**GGI\_TLM\_create\_PCB\_simulation\_model** generates surface geometry for the PCB structure from gerber files. Dielectric layers may then be specified in order to model the PCB substrate materials. Vias connecting different layers are then specified. Following this, lumped component models are specified by their terminal coordinates and the z position of the component (height above the PCB layer). The process creates the geometry for the connecting wires and the surface which will be used as the ‘active’ element. Models of both two terminal and three terminal devices may be generated. A heatsink model may be included in the geometry (this can be important for conducted emissions simulations for power converters).

Additional information to be included in the GGI\_TLM input file can then be given e.g. required outputs, ngspice\_timestep\_factor, simulation time etc.

Finally the part of the Ngspice circuit file which describes all of the circuit elements to be linked into the GGI\_TLM model are included.

Figure 4 shows a typical model, here a dc-dc converter, which may be readily built using the **GGI\_TLM\_create\_PCB\_simulation\_model** software. The model includes a PCB tracks on a dielectric substrate, lumped components connected to the PCB pads on both sides of the board. The model also includes a heatsink.

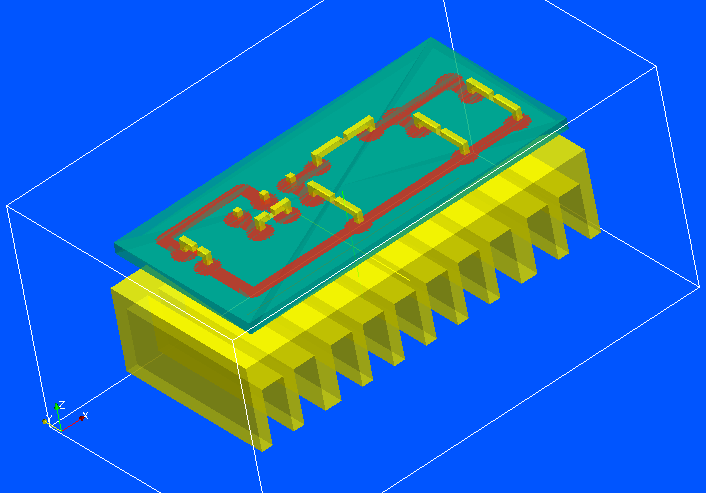


Figure Full converter model

The components on the PCB are either one port (two terminal) components as seen in Figure 5 or two port (three terminal) components as seen in Figure 6. In Figure 5 the component position in the 3D GGI\_TLM model is shown as the blue surface. This is connected to the PCB pads by conductors (which are represented by PEC surfaces in the GGI\_TLM model) shown in yellow. In Figure 6 the three conductors connecting the device model to the PCB pads are again shown in yellow and the two ports are shown by the blue and green surfaces.

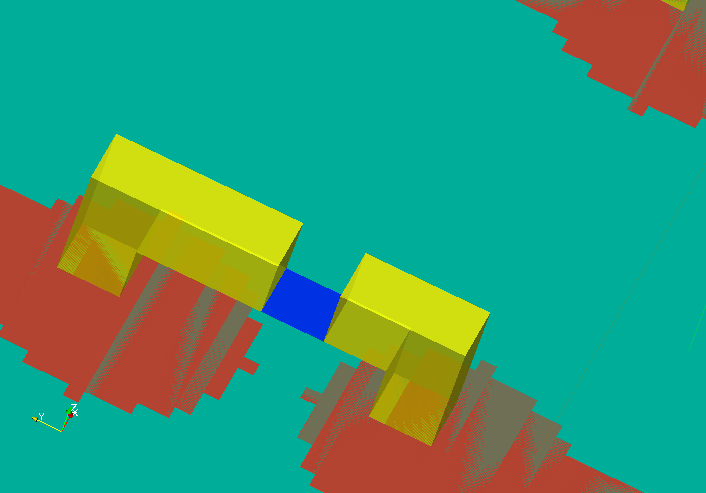


Figure One port component model

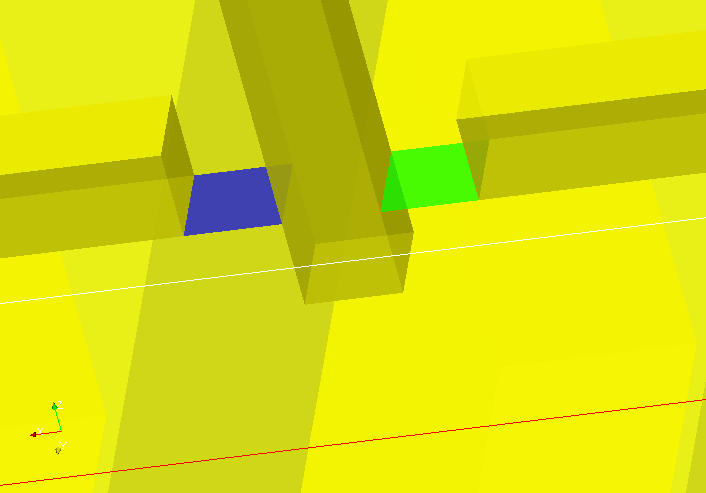


Figure Two port component model

In the **GGI\_TLM\_create\_PCB\_simulation\_model** software a one port model is specified using the port number, the Ngspice nodes corresponding to each of the two connections, the 3D coordinates of the PCB pads to which it is connected and the z position of the actual component surface in the GGI\_TLM model. A package model (at the moment this consists of a dielectric volume) may also be added. The format of the one\_port\_model component information is shown below for a one port component with a rectangular package whose dielectric properties are defined in the material file **component\_case.vmat**:

1 #component number

one\_port\_model

1 # number of ports

1 # port number

3 0 # Ngspice node numbers for connection 1 and connection 2

x1 y1 z1 # coordinates of connection 1

x2 y2 z2 # coordinates of connection 2

zc # z position of component

rectangular # package type

xmin ymin zmin xmax ymax zmax # min and max coordinates of package volume

component\_case

If no package model is to be included then the following package definition should be used:

none # package type

The software automatically creates a Spice model surface at the specified z position of the component, the x and y position of the surface is calculated as the average of the x and y coordinates of the two connections. The surfaces constituting the PEC connection wires are also generated automatically. Figure 7 shows the orientation of the component for the example of a diode where the Ngspice circuit file line is:

D1 node1 node2 diode\_model

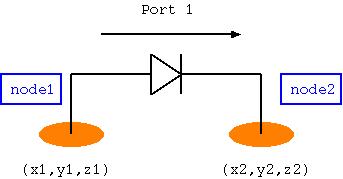


Figure one port component orientation

A two port model has two ports and three electrical connections as seen in Figure 8. Each of the ports has two Ngspice nodes associated with it therefore there are four Ngspice nodes associated with the two port model. A two port model is specified using the port numbers, the 4 Ngspice nodes corresponding to each of the two ports, the 3D coordinates of the PCB pads to which it is connected and the z position of the actual component surface in the GGI\_TLM model. A package model (at the moment this consists of a dielectric volume) may also be added. The format of the two\_port\_model component information is shown below for a two port component with a rectangular package whose dielectric properties are defined in the material file **component\_case.vmat**:

1 #component number

two\_port\_model

2 # number of ports

1 2 # port number

3 0 # Ngspice node numbers for port 1 (connection 1 and connection 2)

0 4 # Ngspice node numbers for port 2 (connection 2 and connection 3)

x1 y1 z1 # coordinates of connection 1

x2 y2 z2 # coordinates of connection 2

x3 y3 z3 # coordinates of connection 3

zc # z position of component

rectangular # package type

xmin ymin zmin xmax ymax zmax # min and max coordinates of package volume

component\_case

If no package model is to be included then the following package definition should be used:

none # package type

The software automatically creates a Spice model surface at the specified z position of the component, the x and y position of the surface is calculated as the average of the x and y coordinates of the three connection points. The surfaces constituting the PEC connection wires are also generated automatically. Figure 8 shows the orientation of the component for the example of a diode where the Ngspice circuit file lines are:

D1 node1 node2 diode\_model

V1 node3 node4 dc 1.0

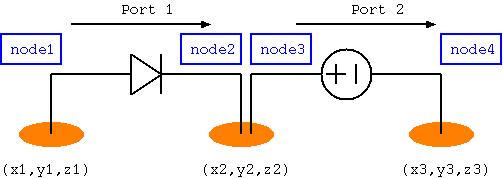


Figure two port orientation

Great care must be taken to ensure the correct orientation of components such as diodes and voltage sources in which the correct orientation of the component is significant. This includes components on which initial conditions are applied for example inductor currents or capacitor voltages. Figure 9 illustrates the orientation of a selection of components in order to clarify the component orientation with respect to the node specification in Ngspice.

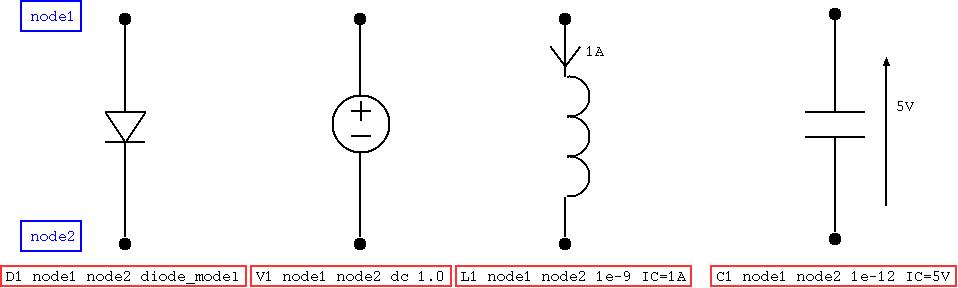


Figure Component orientations with respect to node order, including initial conditions

Additional components may be specified in the input to the **GGI\_TLM\_create\_PCB\_simulation\_model** software. Currently these additional components may be a heatsink model or a dielectric volume.

The heatsink model consists of a cuboid block with slots cut into in to form the vanes of the heatsink. The specification of a heatsink model requires the outer dimensions of the heatsink, the number of slots, the direction across the width of the slots, the direction in which the slots are cut plus the width and depth of the slots. The format of this information is shown below for the example of the heatsink shown in Figure 10:

heatsink  
-0.002  0.0 -0.020    0.062 0.03 -0.005  # outer dimensions of the heatsink  
10                                       # number of slots  
y                                        # slot width direction  
zmin                # slot depth direction (face from which slots are cut)  
0.004               # width of slots   
0.010               # depth of slots

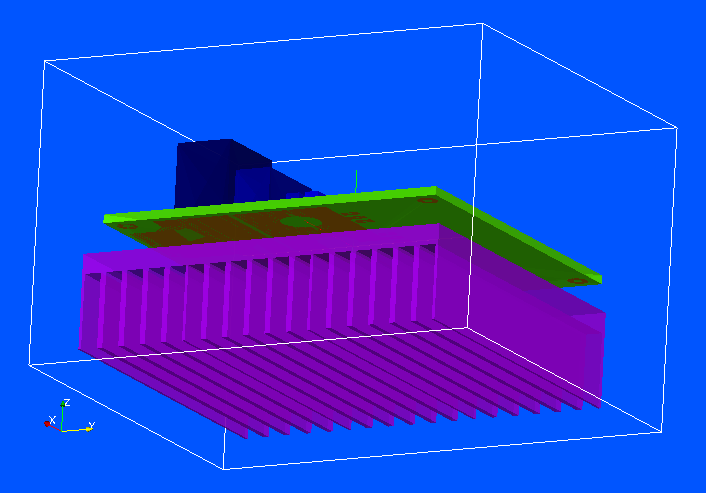


Figure Converter model highlighting the heatsink model.

A dielectric volume may also be specified as an additional component. This could be a ceramic tile used between a component package and a heatsink for example. An example of the format of this information is shown below:

dielectric  
0.015  0.010  -0.005     0.026  0.020   -0.004 # outer dimensions of the dielectric. Material name follows (without .vmat extension)  
ceramic\_tile

# PCB Simulation Specification file format:

The format of the PCB simulation specification file format is shown in the table below alongside an example of each element of the file.

|  |  |  |
| --- | --- | --- |
| **Format specification** | **Data type** | **Example** |
| GGI\_TLM input filename (\*.inp) (string) | String | circuit\_test.inp |
| dl: cell size for the TLM solution | real | 0.001 |
| TLM problem space dimensions  xmin, ymin, zmin, xmax, ymax, zmax | 6\*real | -0.05 -0.02 -0.03 0.05 0.02 0.03 |
| Number of gerber files to include | integer | 1 |
| **For each gerber file:** |  |  |
| gerber file name | string | two\_track.gbr |
| z position of layer | real | 0.003 |
| Number of dielectric layers | integer | 1 |
| **For each dielectric layer:** |  |  |
| Rectangular dielectric region dimensions  xmin, ymin, zmin, xmax, ymax, zmax | 6\*real | -0.04 -0.01 0.0 0.04 0.01 0.003 |
| material filename (without .vmat extension) | string | FR4 |
| Number of vias | integer | 1 |
| **For each via:** |  |  |
| x and y coordinates then zmin and zmax of via | 4\*real | -0.03 0.005 0.0 0.003 |
| Number of lumped components | integer | 2 |
| **For a one port lumped component:** |  |  |
| component number (should be numbered  in order) | integer |  |
| component type (‘one\_port\_model’ or  ‘two\_port\_model’) | string | one\_port\_model |
| Number of ports | integer | 1 |
| Ngspice node numbers for port | 2\*integer | 1 0 |
| connection point coordinates for connection 1 | 3\*real | 0.02 0.005 0.003 |
| connection point coordinates for connection 2 | 3\*real | 0.03 0.005 0.003 |
| z offset for component | real | 0.005 |
| Package type (none, rectangular) | string | rectangular |
| package parameters | rectangular : 6\*real | -0.006 -0.003 0.0 0.006 0.003 0.008 |
| **For a two port lumped component:** |  |  |
| component number (should be numbered  in order) | integer |  |
| component type (‘one\_port\_model’ or  ‘two\_port\_model’) | string | two\_port\_model |
| Number of ports | integer | 2 |
| Ngspice node numbers for port1 | 2\*integer | 1 0 |
| Ngspice node numbers for port2 | 2\*integer | 2 0 |
| connection point coordinates for connection 1  (port 1) | 3\*real | 0.025 -0.005 0.003 |
| connection point coordinates for connection 2  (reference of ports 1 and 2) | 3\*real | 0.03 -0.005 0.003 |
| connection point coordinates for connection 3  (port 2) | 3\*real | 0.035 -0.005 0.003 |
| z offset for component | real | 0.005 |
| Package type (none, rectangular) | string | rectangular |
| package parameters | rectangular : 6\*real | -0.004 -0.003 0.0 0.006 0.003 0.008 |
| Number of additional components (heatsinks etc) | integer | 1 |
| **For each additional component:** |  |  |
| additional component type | string | heatsink |
| outer dimensions of heatsink | 6\*real | -0.04 -0.01 -0.02  0.04 0.01 -0.001 |
| Number of slots | integer | 10 |
| slot width direction (x, y or z) | character | x |
| slot depth direction (x, y or z) | character | z |
| width of slots | real | 0.002 |
| depth of slots | real | 0.015 |
| \* START of GGI\_TLM input file text \* | string | \* START of GGI\_TLM input file text \* |
|  | n\*string | ngspice\_output\_node\_list |
| **Multiple lines of GGI\_TLM input file text**  **may follow until it is ended with the line:** |  | 1 # Number of output nodes  1 # output node number  1 0 # Ngspice output node numbers |
|  |  | Simulation\_time  1e-8 |
| \* END of GGI\_TLM input file text \* | string | \* END of GGI\_TLM input file text \* |
| \* START of Ngspice input file text \* | string | \* START of ngspice input file text \* |
| **Multiple lines of Ngspice input file text**  **may follow until it is ended with the line:** | n\*string | \*GGI\_TLM port 1 model |
|  |  | Rs1 1001 1 100.0 |
|  |  | Vs1 1001 0 exp(0 1 1e-10 1e-9 1e-10) |
|  |  | \* GGI\_TLM port 2 model |
|  |  | D1 0 2 Dsch |
|  |  | .MODEL Dsch D( IS=0.0002 ) |
| \* END of Ngspice input file text \* | string | \* END of ngspice input file text \* |

# Known problems

There is a known issue regarding memory usage with the GGI\_TLM/Ngspice linked software/ Ngpsice appears to store node voltages at all the previous time-steps of the simulation thus as a simulation proceeds, Ngspice uses more and more memory. For some simulations with very large numbers of nodes and many timesteps (of order of millions) this may cause problems if the whole of the memory is used before the simulation finishes.

The impact of this ‘memory leak’ may be reduced by using the Ngspice ‘SAVE’ command to tell Ngspice to only save data for certain nodes in the circuit (i.e. those used in the GGI\_TLM interface and those required for output).

The way in which a simulation is approached may also be used to reduce the required simulation time. An initial circuit simulation in Ngspice to allow initial slow transient circuit effects to reach their steady state may be performed and then the individual component voltages (for a capacitor) or currents (for an inductor) used as initial conditions in the GGI\_TLM – Ngspice coupled solution.

# Post processing conducted emissions data

Post processing procedures have been implemented in order to calculate conducted emissions data from raw time domain voltage data. The post processing software allows a number of processes to be applied to the data in both time and frequency domains to allow investigation of different data analysis procedures and to allow comparison with standards. The software allows the following processes to be applied to the raw time domain data:

1. Scale the voltage data (e.g. to micro volts) as required
2. Apply a low pass filter to the time domain data
3. Apply a high pass filter to the time domain data
4. Extract a series of sub-samples from the dataset i.e. divide a long period of data into a number of shorter sub-samples distributed through the time period of the input data. The sub-samples may be sampled at a different sample rate to the initial dataset. The sub-sampled data may be padded with zeros or a periodic extension of the dataset added.
5. A window function may be applied to each of the sub-sampled datasets.
6. The d.c. component of the signal can be subtracted
7. The FFT of each of the sub-samples may be calculated to give frequency domain sub-sample data
8. The frequency domain sub-sample functions may be averaged in the frequency domain
9. The response of a receiver with a specified detector bandwidth may be calculated where the receiver detector frequency response may be described by a gaussian or a rectangular frequency domain response

This processes which allows application of all the above options has been incorporated into the **GGI\_TLM\_post\_process** software, option 58. The software is run with the command

**GGI\_TLM\_post\_process**

The available post processing options are then presented. The time domain conducted emissions processing option is selected by inputting **58** at the prompt.

The filename for the time domain data must then be supplied by the user. This file should be an ascii (text) file with columns for time and voltage. There may be header lines at the top of the file but following this the data should be continuous. It is assumed that the voltage data is sampled at a uniform sample rate. This assumption is valid for time domain oscilloscope data and for GGI\_TLM simulation data however it may not be valid for raw Spice time domain output due to the variable time stepping algorithm used in Spice [ ].

The user is requested to supply the number of header lines in the file to ignore and then the column for the time data and then the column for the voltage data.

The first process which can be applied is scaling the voltage data. For example if the data is to be presented as dBµV then a scaling factor of ‘1E-6’ should be given. For no scaling, enter ‘1’.

Once the time domain data has been read and scaled appropriately, low pass and/or high pass filters may be applied. These filters may be used to prevent aliasing if the data is subsequently sub-sampled at a lower data rate for example.

The low and high pass filter functions are Butterworth filters. The filter order is chosen to be between 1 and 4 and the cutoff frequency, fc, must be specified. The low pass filters of order 1 to 4 are defined using the following functions of s where

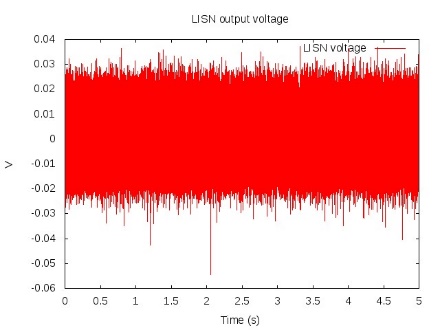
The high pass filter functions are obtained using the same formulae with

The user is asked to enter the number of filters which are required (this may be 0).

For each filter applied the user must enter the type of filter to apply (‘LPF’ or ‘HPF’) then the filter order (integer between 1 and 4, inclusive) and the filter cutoff frequency (Hz).

Following the filtering process, the dataset may be re-sampled in order to obtain a number of sub-samples over shorter timespans. The sample rate may also be changed (reduced).

The user is required to specify the number of sub-segments of data required, the time period of each sub-segment, the number of samples in each sub-segment (this should be a power of 2 due to FFT implementation), and the total time period over which the sub-samples are to be distributed (or enter ‘0’ for continuous sub-samples). These parameters are explained in Figure 11 below. Note that the sampling times at this stage will probably be different to the sampling times of the input data. In the re-sampling of the input dataset a linear interpolation is used to calculate the voltage value between input time samples.



Sub-sample 1

Sub-sample 2

Sub-sample 3

Sub-sample 4

Sub-sample 5

Sub-sample period

Total sample period

Figure . Sub-sampling the input dataset: 5 sub-samples, each of period 0.25s distributed over 5s

The sub-sampled data may be padded with zeros or alternatively a periodic extension of the dataset may be added such that the total number of samples is a power of 2. This has the effect of providing frequency samples from the FFT at a finer resolution that can be obtained with the initial sub-sampled dataset. The user must enter the padding factor (enter ‘1’ for no padding) and then either ‘z’ to pad with zeros or ‘p’ to add a periodic extension of the data. Note that the effect of zero padding on the final frequency domain output will be compensated for such that the same frequency domain amplitude will be produced whatever the extent of the zero padding.

After sub-sampling and padding the data, a time domain window function may be applied. The window function which may be applied to the time domain data can be either a rectangular window or a Hann window. The Hann window is given by the function

The Hann window is shown in Figure 12.

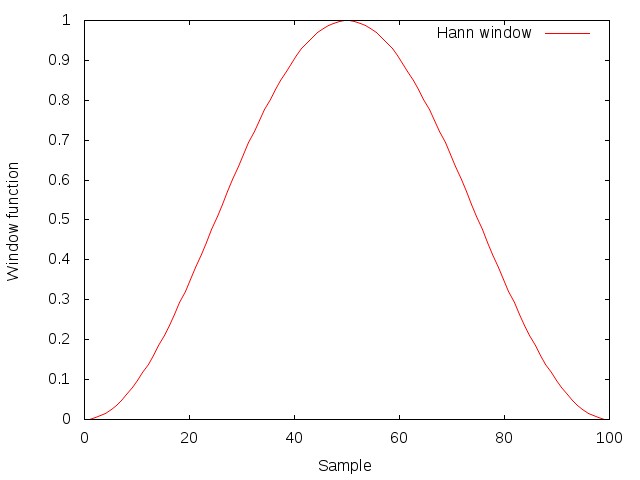


Figure . Hann Window function

The user is requested to enter the window type and must respond with either ‘r’ for a rectangular window (effectively no change to the data as the window covers the full period of the data) or ‘h’ to apply the Hann window as described above.

Following windowing, the user is asked whether to subtract the d.c. component of the signal. The user must enter ‘y’ or ‘n’ to this question.

Each of the sub-samples is transformed into the frequency domain via the FFT algorithm. The magnitude of these frequency domain functions is then averaged. This raw frequency domain data is then written to a file (filename to be provided by the user.)

The format of the raw frequency domain data output is a text file with six columns as follows:

**Frequency (Hz), Re{V(f)}, Im{V(f)}, Abs{V(f)}, VdBm(50Ω load), VdB**

Note that the real part is the average of the magnitudes and the imaginary part is zero. The data is only written for positive frequencies however the VdBm and VdB data compensates for this by the addition of 3.01dB so it can be compared to measured data etc.

In order to provide frequency domain data which can be compared with that measured using a spectrum analyser or for comparison with emissions limits from standards, frequency domain data with a specified receiver (detector) bandwidth can be produced. Different receiver bandwidths can be specified over different frequency ranges if required (see Post processing example 1 below for an example). The user must specify the number of frequency bands with different receiver bandwidths, the filename for the output and for each band:

1. Minimum frequency of the band (Hz)
2. Maximum frequency of the band (Hz)
3. Number of frequencies to output
4. ‘r’ for a rectangular receiver transfer function or ‘g’ for a Gaussian receiver transfer function
5. The detector bandwidth (Hz)

The detector transfer functions are shown in

\*\*\*\* Detector transfer function figure \*\*\*\*

The format of the frequency domain data output is a text file with three columns as follows:

**Frequency (Hz), P\_detector, VdBm(50Ω load), VdB**

Note that P\_detector is the total power in the detector band width at each frequency and that negative frequency data is taken into account in the output.

A number of test cases have been produced which exercise the conducted emissions data post processing. These are available in the GGI\_TLM directory **GGI\_TLM/TEST\_DATA/POST\_PROCESS\_CONDUCTED\_EMISSIONS/**

TEST 1:

Single cycle of 10 harmonics from a square wave expansion, 20kHz.

2048 samples, timestep=2.44e-8s, fmax=20.49GHz.

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

TEST 1B:

Single cycle of 10 harmonics from a square wave expansion, 20kHz.

2000 samples, timestep=2.5e-8s, fmax=20GHz.

Resampling adds some noise to the frequency domain output

A screenshot of a cell phone

Description automatically generated

TEST 2:

1000 cycles of 10 harmonics from a square wave expansion, 20kHz.

200 samples, timestep=2.5e-7s, fmax=2GHz.

Resampling adds some noise to the frequency domain output at period of 2GHz

A screenshot of a cell phone

Description automatically generated

A close up of a piece of paper

Description automatically generated

TEST 3:

1000 cycles of 10 harmonics from a square wave expansion, 20kHz plus 50Hz contribution

200 samples, timestep=2.5e-7s, fmax=2GHz.

Use of High pass filter at 100Hz to reduce 50Hz influence on result

A close up of a logo

Description automatically generated

A close up of a piece of paper

Description automatically generated

A close up of a map

Description automatically generated

TEST 3b:

1000 cycles of 10 harmonics from a square wave expansion, 20kHz plus 50Hz contribution

200 samples, timestep=2.5e-7s, fmax=2GHz.

No High pass filter at 100Hz to reduce 50Hz influence on result. Compare with TEST 3.

A close up of a map

Description automatically generated

TEST 4:

1000 cycles of 10 harmonics from a square wave expansion, 20kHz.

200 samples, timestep=2.5e-7s, fmax=2GHz.

Single period in sub-sample and zero pad \*128. Not recommended

A close up of a piece of paper

Description automatically generated

TEST 5:

1000 cycles of 10 harmonics from a square wave expansion, 20kHz.

200 samples, timestep=2.5e-7s, fmax=2GHz.

Single period in sub-sample and periodic extension \*128. Better than zero padding...

A screenshot of a cell phone

Description automatically generated

TEST 6:

128 cycles of 10 harmonics from a square wave expansion, 20kHz.

2048 samples, timestep=2.5e-7s, fmax=2GHz.

128 periods in sub-sample.

Shows effect of BW change between bands

A close up of a piece of paper

Description automatically generated

A close up of a piece of paper

Description automatically generated

TEST 6b:

128 cycles of 10 harmonics from a square wave expansion, 20kHz.

2048 samples, timestep=2.5e-7s, fmax=2GHz.

100 periods in sub-sample.

Shows effect of sub-sampling interpolation on high frequency response

A close up of a piece of paper

Description automatically generated

TEST7:

10000 cycles of 10 harmonics from a square wave expansion, 20kHz plus 0.5V gaussian noise

average over 100 frequency domain sub-samples, each of 100 cycles

A close up of a map

Description automatically generated

TEST7b:

1000 cycles of 10 harmonics from a square wave expansion, 20kHz plus 0.5V gaussian noise

process only 1 frequency domain sub-sample of 100 cycles

A close up of a piece of paper

Description automatically generated

MEAS:

process measured data measured at 500kHz over 0.1s

process data over 10 subsegments averaged over 10 subsamples

A screenshot of a cell phone

Description automatically generated

A close up of a piece of paper

Description automatically generated

MEAS2:

process measured data measured at 500kHz over 0.1s

process data over the whole dataset

A close up of a piece of paper

Description automatically generated

# Examples

## Transmission line with non-linear load.

In this first example a pulse voltage source drives a diode load via a transmission line. The transmission line is simulated in GGI\_TLM and the source and load lumped components are simulated in Ngspice. The configuration is shown in Figure 3.

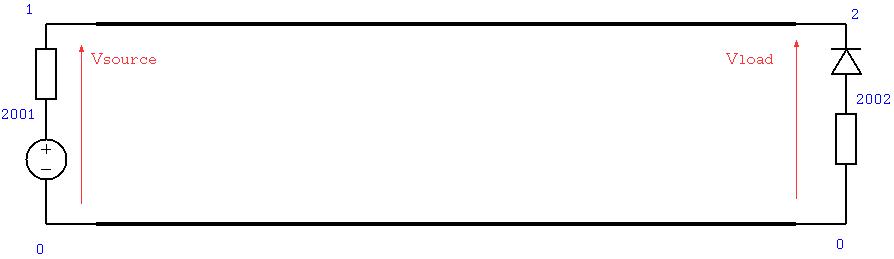


Figure 13 Example 1. Transmission line linking voltage souce and resistance to diode and resistance load

The geometry of the two conductor transmission line is specified as a gerber file (two\_track.gbr) and is shown in Figure 4.

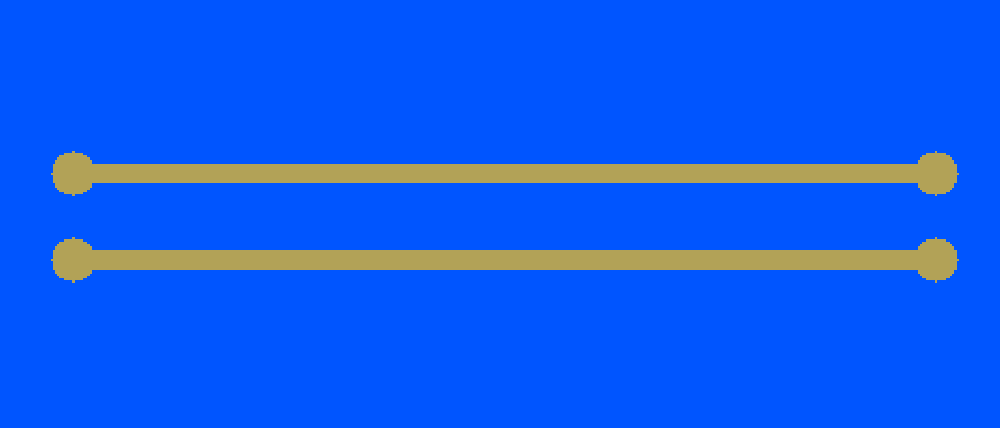


Figure 14 Two track transmission line PCB layout

The Simulation model is built using the process **GGI\_TLM\_create\_PCB\_simulation\_model**. The input file for this process is shown below.

**PCB Simulation Specification file:**

two\_wire\_diode\_test.inp

0.001 # dl: cell size for the TLM solution

-0.06 -0.02 -0.01 0.06 0.02 0.04 # TLM problem space dimensions: xmin,ymin,zmin,xmax,ymax,zmax

1 # Number of gerber files to include

two\_track.gbr

0.002 # z position for the layer specified in this gerber file

0 # Number of dielectric layers

0 # Number of vias

2 # Number of lumped components

1 # COMPONENT NUMBER

one\_port\_model # Component type for component 1

1 # Number of ports

1 # Ngspice node number for port 1

-0.05 0.005 0.002 # connection point coordinates for connection number 1, port 1

-0.05 -0.005 0.002 # connection point coordinates for connection number 2, reference for port 1

0.004 # z offset for component

none # package type

2 # COMPONENT NUMBER

one\_port\_model # Component type for component 2

1 # Number of ports

2 # Ngspice node number for port 1

0.05 0.005 0.002 # connection point coordinates for connection number 1, port 1

0.05 -0.005 0.002 # connection point coordinates for connection number 2, reference for port 1

0.004 # z position for component

none # package type

0 # Number of additional components (heatsinks etc)

\* START of GGI\_TLM input file text \*

ngspice\_node\_output\_list

2 # number of ngspice output nodes

1 # NGSPICE OUTPUT NUMBER

1 # ngspice node number

2 # NGSPICE OUTPUT NUMBER

2 # ngspice node number

ngspice\_timestep\_factor

4

Simulation\_time

1e-8

\* END of GGI\_TLM input file text \*

\* START of Ngspice input file text \*

\* Model to be included in the GGI\_TLM simulation for port 1, connected to node 1

\* in this case a voltage source with series resistance

Rs1 2001 1 100.0

Vs1 2001 0 EXP( 0.0 1.0 0.000000E+00 2.000000E-010 1.200000E-09 2.000000E-010 )

\*

\* Model to be included in the GGI\_TLM simulation for port 2, connected to node 2

\* in this case a diode with series resistance

Diode2 2002 2 Dmod

Rl2 2002 0 100.0

.model DMOD D ( is=1e-12 )

\*

\* END of Ngspice input file text \*

The GGI\_TLM\_create\_PCB\_simulation\_model process creates a GGI\_TLM input file (two\_wire\_diode\_test.inp) and a template circuit file for the Ngspice model (Spice\_circuit\_TEMPLATE.cir) file. The Ngspice circuit for the simulation is shown in Figure 5 below in which the source and load end components are driven by separate Thevenin equivalent circuits which form the links to GGI\_TLM. The 3D Simulation model is shown in Figure 6.

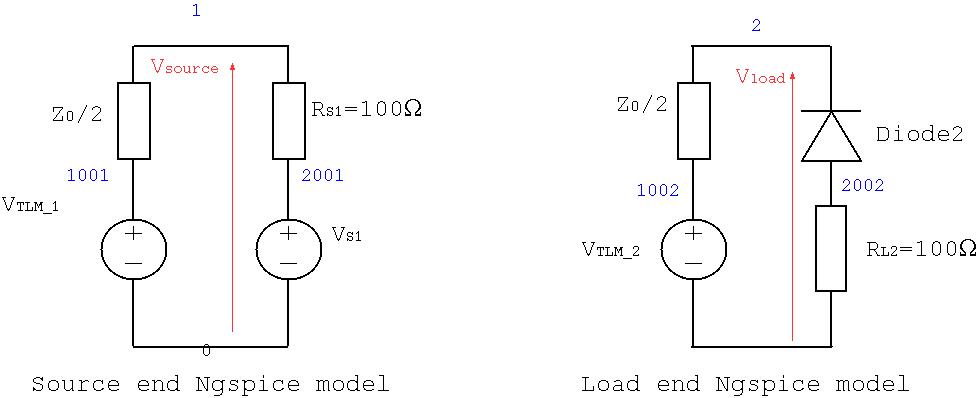


Figure 15 Thevenin equivalent circuit models of source and load end components. Ngspice node numbers are shown in blue.

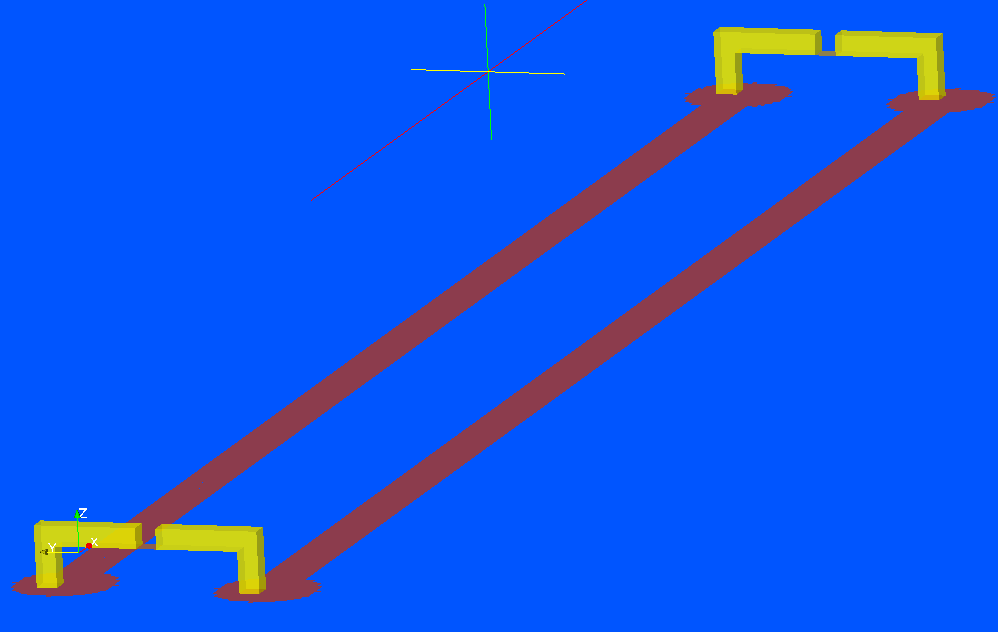


Figure 16. 3D GGI\_TLM Model of the two wire transmission line configuration with component models in place

**GGI\_TLM input file:**

Mesh\_outer\_boundary\_dimension

-6.000000E-02 6.000000E-02 -2.000000E-02 2.000000E-02 -1.000000E-02 4.000000E-02

Mesh\_cell\_dimension

1.000000E-03

Outer\_boundary\_reflection\_coefficient

0.0 0.0 0.0 0.0 0.0 0.0

Surface\_list

4 # Number of surfaces

1 # SURFACE NUMBER

stl\_triangulated\_surface

two\_track.gbr.stl

1.0 0.0

0.000000E+00 0.000000E+00 0.000000E+00

0.000000E+00 0.000000E+00 2.000000E-03

2 # SURFACE NUMBER

zplane

-5.100000E-02 8.673617E-19 4.000000E-03 -5.000000E-02 1.000000E-03 4.000000E-03

0.000000E+00 0.000000E+00 0.000000E+00

0.000000E+00 0.000000E+00 0.000000E+00

3 # SURFACE NUMBER

zplane

5.000000E-02 8.673617E-19 4.000000E-03 5.100000E-02 1.000000E-03 4.000000E-03

0.000000E+00 0.000000E+00 0.000000E+00

0.000000E+00 0.000000E+00 0.000000E+00

4 # SURFACE NUMBER

stl\_triangulated\_surface

component\_terminal\_connections.stl

1.0 0.0

0.000000E+00 0.000000E+00 0.000000E+00

0.000000E+00 0.000000E+00 0.000000E+00

Surface\_material\_list

4 # Number of surface materials

1 # SURFACE MATERIAL NUMBER

PEC

1 # Number of surfaces

1 # Surface list

1 # Surface orientation list

2 # SURFACE MATERIAL NUMBER

SPICE

1 # Ngspice port number

1 0 # Ngspice node numbers

-y # Port direction

1 # Number of surfaces

2 # Surface number

1 # Surface orientation list

3 # SURFACE MATERIAL NUMBER

SPICE

2 # Ngspice port number

2 0 # Ngspice node numbers

-y # Port direction

1 # Number of surfaces

3 # Surface number

1 # Surface orientation list

4 # SURFACE MATERIAL NUMBER

PEC

1 # Number of surfaces

4 # Surface number

1 # Surface orientation list

volume\_list

0 # Number of volumes

volume\_material\_list

0 # Number of volume materials

ngspice\_node\_output\_list

2 # number of ngspice output nodes

1 # NGSPICE OUTPUT NUMBER

1 # ngspice node number

2 # NGSPICE OUTPUT NUMBER

2 # ngspice node number

ngspice\_timestep\_factor

4

Simulation\_time

1e-8

**Spice template input file:**

Following is the template Spice circuit file. Note that this is a template file as the TLM Thevenin equivalent source impedance needs to be substituted along with the timestep information in the .TRAN line. The voltage source values (Vtlm\*) are set as the simulation runs.

Ngspice template file for GGI\_TLM - ngspice linked simulation

\*

\* GGI\_TLM link port 1 using nodes 1 and 0

\*

\* Voltage source with series resistance: equivalent circuit of TLM link

Vtlm1 1001 0 DC 0.0

Rtlm1 1001 1 #Z0\_TLM

\*

\* GGI\_TLM link port 2 using nodes 2 and 0

\*

\* Voltage source with series resistance: equivalent circuit of TLM link

Vtlm2 1002 0 DC 0.0

Rtlm2 1002 2 #Z0\_TLM

\*

\*

\* Voltage source required for the voltage source controlling the breakpoint time

Vbreak time\_node 0 DC 0.0

\*

\* Model to be included in the GGI\_TLM simulation for port 1, connected to node 1

\* in this case a voltage source with series resistance

Rs1 2001 1 100.0

Vs1 2001 0 EXP( 0.0 1.0 0.000000E+00 2.000000E-010 1.200000E-09 2.000000E-010 )

\*

\* Model to be included in the GGI\_TLM simulation for port 2, connected to node 2

\* in this case a diode with series resistance

Diode2 2002 2 Dmod

Rl2 2002 0 100.0

.model DMOD D ( is=1e-12 )

\*

\*

\* Control for transient simulation

.TRAN #dt\_out #tmax\_ngspice 0.0 #dt\_ngspice UIC

\*

.END

The model is validated by simulating the whole circuit, including the transmission line in Ngspice. The transmission line is included in the Ngspice circuit as a delay line with an appropriate characteristic impedance and time delay. The comparison of the time domain voltage at the source and load ends is shown in Figure 7. The Spice circuit file for the validation circuit including the delay line is included below.

**Ngspice validation model:**

Ngspice validation circuit for GGI\_TLM - ngspice linked simulation demonstration

\* Source end voltage source with series resistance

Rs1 2001 1 100.0

Vs1 2001 0 EXP( 0.0 1.0 0.000000E+00 2.000000E-010 1.200000E-09 2.000000E-010 )

\*

\* Delay line used to link source and load ends

\*

T1 1 0 2 0 Z0=320 TD=4.4e-10

\*

\* Load end Diode and series resistance

Diode2 2002 2 Dmod

Rl2 2002 0 100.0

.model DMOD D ( is=1e-12 )

\*

\*

\* Control for transient simulation

.TRAN 4.169551E-13 1.000267E-08 0.0 4.169551E-13

.PRINT TRAN V(1) V(2)

\*

.END

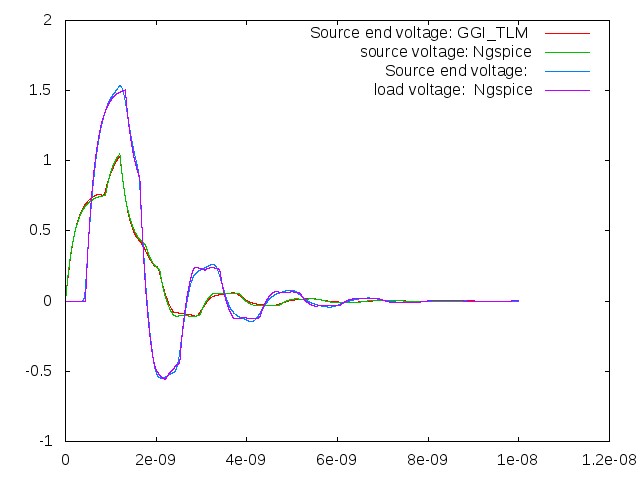


Figure 17. Comparison between GGI\_TLM – Ngspice simulation and Ngspice simulation with delay line.

## Buck converter model

Figure 16 shows the circuit for a simple dc-dc buck converter designed to convert the 20V input to 5V at the load.

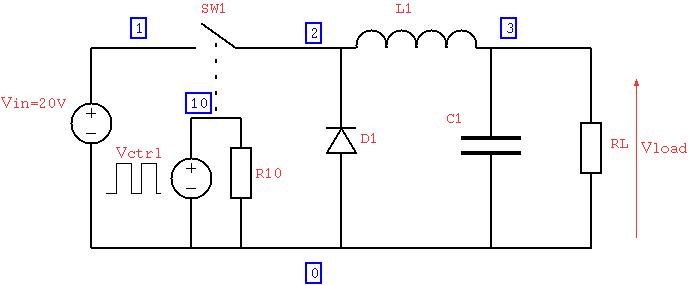


Figure dc-dc converter circuit diagram

An initial ngspice model has been created in order to test the correct operation of the idealised circuit and to calculate the initial conditions for the GGI\_TLM- Ngspice model. The calculation of initial conditions for the inductor current and the large output capacitor is due to the very large time constants of the turn on transient. The load voltage is shown as a function of time in Figure 17 and the inductor current in Figure 18. A simulation time of 200µs has been used to allow the converter to reach a steady state. The detail of the steady state inductor current is shown in Figure 19.

BASIC BUCK CONVERTER

\*

\* SWITCH DRIVER

VCTRL 10 0 PULSE(0V 6V 0 0.0001US 0.0001US 0.05US 0.2US)

R10 10 0 1MEG

\*

\* INPUT VOLTAGE

VIN 1 0 DC 20

\*

\* CONVERTER

SW1 1 2 10 0 SWitch1

D1 0 2 DSCH

L1 2 30 50UH

VL1 30 3 0.0

C1 3 0 2UF

\*

\* LOAD

RL 3 0 5.0

\*

.MODEL SWitch1 SW Vt=5V Vh=0.2V RON=0.01 ROFF=1MEG

.MODEL DSCH D( IS=0.0002 )

\*

.OPTIONS NOPAGE

\* ANALYSIS

.TRAN 0.01US 200US

\*

\* VIEW RESULTS

.PRINT tran V(3) I(VL1)

.END

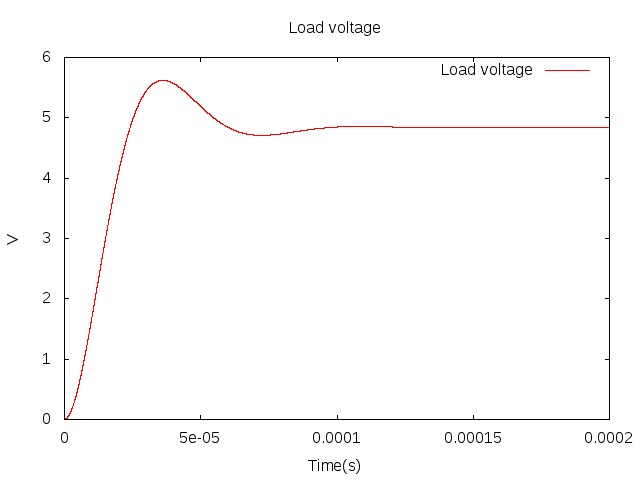


Figure Switch on transient converter output voltage

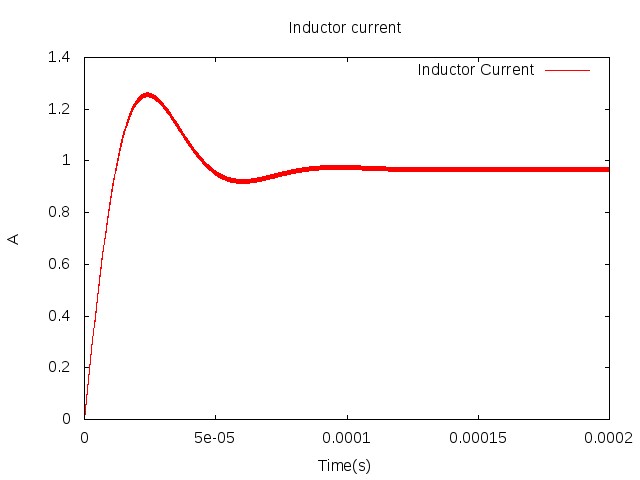


Figure Switch on transient inductor current

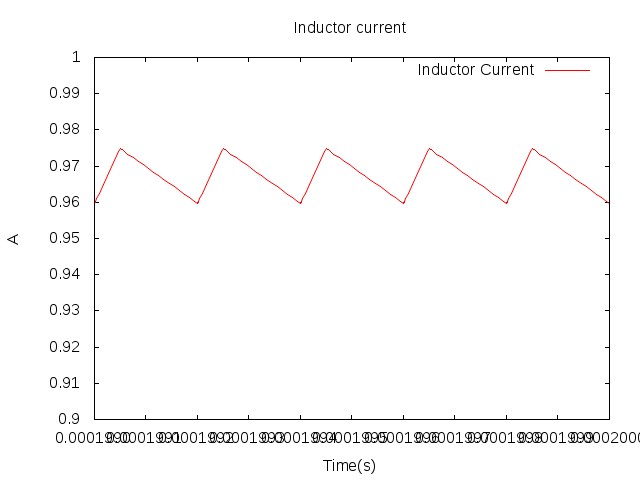


Figure Detail of switch on transient inductor current approaching steady state

A second Ngspice simulation has been performed in which the steady state output capacitor voltage and inductor current have been set as initial conditions as seen in the Ngspice circuit file below.

BASIC BUCK CONVERTER using initial conditions

\*

\* SWITCH DRIVER

VCTRL 10 0 PULSE(0V 6V 0 0.0001US 0.0001US 0.05US 0.2US)

R10 10 0 1MEG

\*

\* INPUT VOLTAGE

VIN 1 0 DC 20

\*

\* CONVERTER

SW1 1 2 10 0 SWitch1

D1 0 2 DSCH

L1 2 30 50UH IC=0.967A

VL1 30 3 0.0

C1 3 0 2UF IC=4.835V

\*

\* LOAD

RL 3 0 5.0

\*

.MODEL SWitch1 SW Vt=5V Vh=0.2V RON=0.01 ROFF=1MEG

.MODEL DSCH D( IS=0.0002 )

\*

.OPTIONS NOPAGE

\* ANALYSIS

.TRAN 0.01US 1US UIC

\*

\* VIEW RESULTS

.PRINT tran V(3) I(VL1)

.END

Figure 20 and Figure 21 show the load voltage and inductor current as a function of time with the initial conditions applied where it is seen that the steady state operation has been reached from the start of the simulation. The approach of using initial conditions to eliminate the switch on transient must be used in the GGI\_TLM simulation as it is not feasible to run GGI\_TLM for the length of time required to calculate the initial transient behaviour of the circuit.

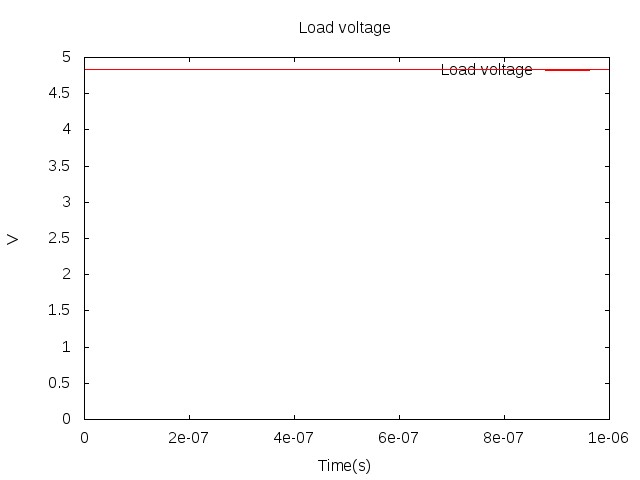


Figure Load voltage using initial conditions

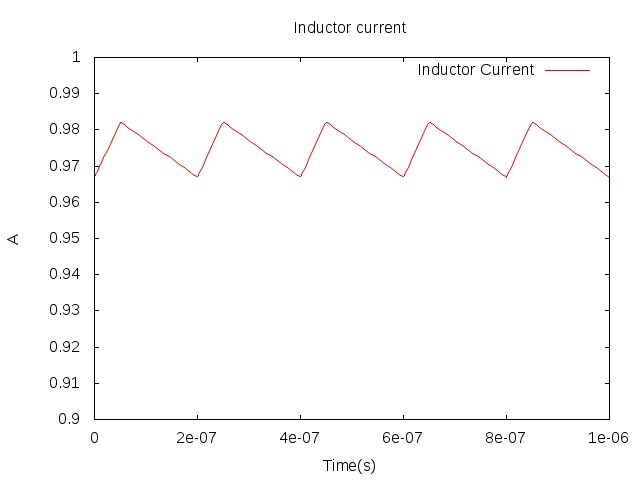


Figure Inductor current using initial conditions

The GGI\_TLM-Ngspice model development starts with the PCB layout which is specified in a gerber file. The PCB geometry is shown in Figure 22.

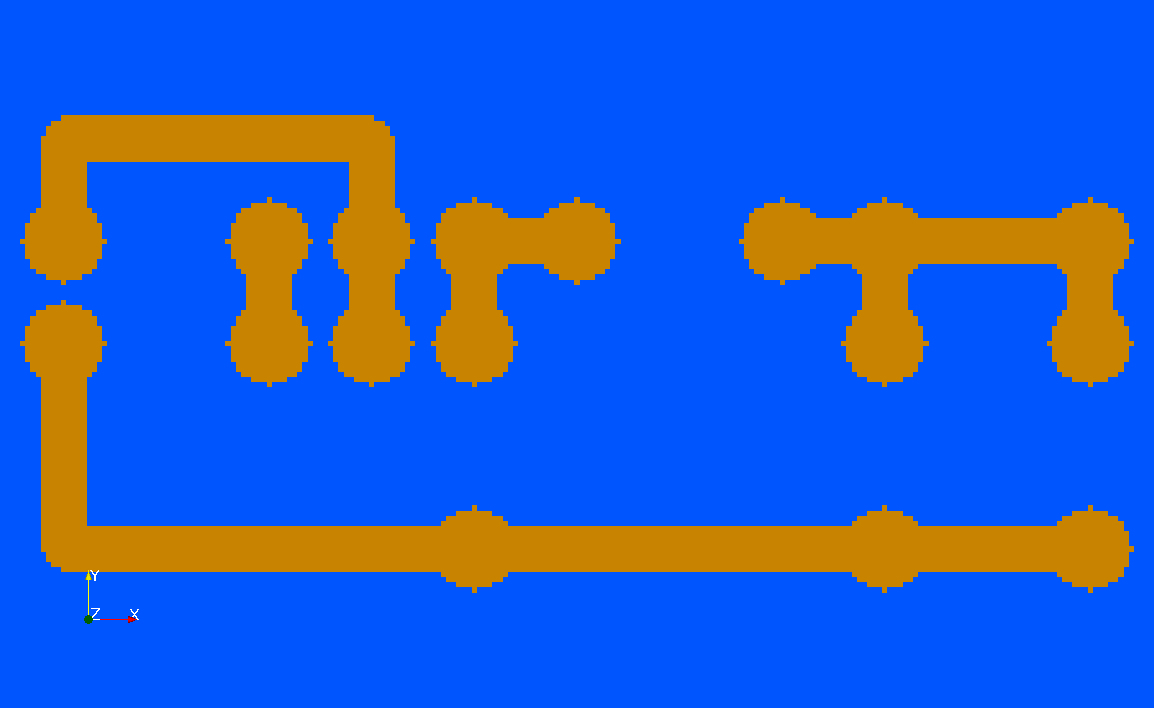


Figure PCB layout for the converter model

The model is created using the **GGI\_TLM\_create\_PCB\_simulation\_model** process. The input file to the process is shown below:

converter.inp

0.001 # dl: cell size for the TLM solution

-0.01 -0.01 -0.03 0.07 0.04 0.01 # TLM problem space dimensions: xmin,ymin,zmin,xmax,ymax,zmax

1 # Number of gerber files to include

top\_layer.gbr

0.002 # z position for the layer specified in this gerber file

1 # Number of dielectric layers

0.0 0.0 0.0 0.06 0.03 0.002 # outer dimensions of the dielectric layer. Material filename follows (without .vmat extension)

FR4

0 # Number of vias

7 # Number of lumped components

1 # COMPONENT NUMBER

one\_port\_model # Component type for component 1

1 # Number of ports

1 # port number(s)

1 0 # Ngspice node number for port 1

0.005 0.020 0.002 # connection point coordinates for connection number 1, port 1: note coordinate numbering; end, centre, end

0.005 0.015 0.002 # connection point coordinates for connection number 2, reference for port1

0.003 # z position for component

none # package type

2 # COMPONENT NUMBER

two\_port\_model # Component type for component 2

2 # Number of ports

2 3 # port number(s)

8 0 # Ngspice node number for port 1

0 2 # Ngspice node number for port 2

0.015 0.020 0.002 # connection point coordinates for connection number 1, port 1: note coordinate numbering; end, centre, end

0.020 0.020 0.002 # connection point coordinates for connection number 2, reference for port1

0.025 0.020 0.002 # connection point coordinates for connection number 2, reference for port1

-0.003 # z position for component

rectangular # package type

-0.0055 0.0005 -0.0015 0.0055 -0.0095 0.0015 # package parameters. Material filename follows (without .vmat extension)

component\_case

1 # Number of PEC surfaces

zmin 2 # PEC surface and the terminal to electrically connect to

3 # COMPONENT NUMBER

one\_port\_model # Component type for component 3

1 # Number of ports

4 # port number(s)

3 0 # Ngspice node number for port 1

0.025 0.015 0.002 # connection point coordinates for connection number 1, port 1: note coordinate numbering; end, centre, end

0.025 0.005 0.002 # connection point coordinates for connection number 2, reference for port1

0.003 # z position for component

none # package type

4 # COMPONENT NUMBER

one\_port\_model # Component type for component 4

1 # Number of ports

5 # port number(s)

4 0 # Ngspice node number for port 1

0.030 0.020 0.002 # connection point coordinates for connection number 1, port 1: note coordinate numbering; end, centre, end

0.040 0.020 0.002 # connection point coordinates for connection number 2, reference for port1

0.003 # z position for component

none # package type

5 # COMPONENT NUMBER

one\_port\_model # Component type for component 5

1 # Number of ports

6 # port number(s)

5 0 # Ngspice node number for port 1

0.045 0.015 0.002 # connection point coordinates for connection number 1, port 1: note coordinate numbering; end, centre, end

0.045 0.005 0.002 # connection point coordinates for connection number 2, reference for port1

0.003 # z position for component

none # package type

6 # COMPONENT NUMBER

one\_port\_model # Component type for component 6

1 # Number of ports

7 # port number(s)

6 0 # Ngspice node number for port 1

0.055 0.015 0.002 # connection point coordinates for connection number 1, port 1: note coordinate numbering; end, centre, end

0.055 0.005 0.002 # connection point coordinates for connection number 2, reference for port1

0.003 # z position for component

none # package type

7 # COMPONENT NUMBER

one\_port\_model # Component type for component 6

1 # Number of ports

8 # port number(s)

7 0 # Ngspice node number for port 1

0.015 0.015 0.002 # connection point coordinates for connection number 1, port 1: note coordinate numbering; end, centre, end

0.020 0.015 0.002 # connection point coordinates for connection number 2, reference for port1

0.003 # z position for component

none # package type

2 # Number of additional components (heatsinks etc)

heatsink

-0.002 0.0 -0.020 0.062 0.03 -0.005 # outer dimensions of the heatsink

10 # number of slots

y # slot width direction

zmin # slot depth direction (face from which slots are cut)

0.004 # width of slots

0.010 # depth of slots

dielectric

0.015 0.010 -0.005 0.026 0.020 -0.004 # outer dimensions of the dielectric. Material name follows (without .vmat extension)

ceramic\_tile

\* START of GGI\_TLM input file text \*

ngspice\_node\_output\_list

4 # number of ngspice output nodes

1 # NGSPICE OUTPUT NUMBER

1 # ngspice node number # Supply voltage

2 # NGSPICE OUTPUT NUMBER

2 # ngspice node number # Switch control voltage

3 # NGSPICE OUTPUT NUMBER

3 # ngspice node number # Diode voltage

4 # NGSPICE OUTPUT NUMBER

6 # ngspice node number # load voltage

ngspice\_timestep\_factor

8 (real)

Simulation\_time

1e-9

1e-7

#1e-6

\* END of GGI\_TLM input file text \*

\* START of Ngspice input file text \*

\* Model to be included in the GGI\_TLM simulation connected to node 1

\* INPUT VOLTAGE

VIN 1 0 DC 20

\* Model to be included in the GGI\_TLM simulation connected to node 2

\* Voltage controlled switch model

SW1 2 0 8 0 SWitch1

\* Model to be included in the GGI\_TLM simulation connected to node 3

\* Diode model

D1 0 3 DSCH

\* Model to be included in the GGI\_TLM simulation connected to node 4

\* Inductor model

L1 4 0 50UH IC=1A

\* Model to be included in the GGI\_TLM simulation connected to node 5

\* Capacitor model

C1 5 0 2UF IC=5V

\* Model to be included in the GGI\_TLM simulation connected to node 6

\* LOAD model

RL 6 0 5.0

\* Model to be included in the GGI\_TLM simulation connected to node 7

\* SWITCH CONTROL SIGNAL

R10 7 0 1MEG

\* Model to be included in the GGI\_TLM simulation connected to node 8

\* Gate-source impedance

RGS 8 0 1MEG

VCTRL 8 0 PULSE(0V 6V 0 0.0001US 0.0001US 0.05US 0.2US)

\*

.MODEL SWitch1 SW Vt=5V Vh=0.2V RON=0.01 ROFF=1MEG

\*

.MODEL DSCH D( IS=0.0002 )

\*

\* END of Ngspice input file text \*

This process creates a GGI\_TLM input file and a template Ngspice circuit file (with #Z0\_TLM and simulation time still to be substituted) as shown below. A diagram of this Ngspice circuit is shown in Figure 23. Note that initial conditions have been set for the output capacitor voltage and the inductor current.

Ngspice template file for GGI\_TLM - ngspice linked simulation

\*

\* GGI\_TLM link port 1 using nodes 1 and 0

\*

\* Voltage source with series resistance: equivalent circuit of TLM link

Vtlm1 1001 0 DC 0.0

Rtlm1 1001 1 #Z0\_TLM

\*

\* GGI\_TLM link port 2 using nodes 8 and 0

\*

\* Voltage source with series resistance: equivalent circuit of TLM link

Vtlm2 1002 0 DC 0.0

Rtlm2 1002 8 #Z0\_TLM

\*

\* GGI\_TLM link port 3 using nodes 0 and 2

\*

\* Voltage source with series resistance: equivalent circuit of TLM link

Vtlm3 1003 2 DC 0.0

Rtlm3 1003 0 #Z0\_TLM

\*

\* GGI\_TLM link port 4 using nodes 3 and 0

\*

\* Voltage source with series resistance: equivalent circuit of TLM link

Vtlm4 1004 0 DC 0.0

Rtlm4 1004 3 #Z0\_TLM

\*

\* GGI\_TLM link port 5 using nodes 4 and 0

\*

\* Voltage source with series resistance: equivalent circuit of TLM link

Vtlm5 1005 0 DC 0.0

Rtlm5 1005 4 #Z0\_TLM

\*

\* GGI\_TLM link port 6 using nodes 5 and 0

\*

\* Voltage source with series resistance: equivalent circuit of TLM link

Vtlm6 1006 0 DC 0.0

Rtlm6 1006 5 #Z0\_TLM

\*

\* GGI\_TLM link port 7 using nodes 6 and 0

\*

\* Voltage source with series resistance: equivalent circuit of TLM link

Vtlm7 1007 0 DC 0.0

Rtlm7 1007 6 #Z0\_TLM

\*

\* GGI\_TLM link port 8 using nodes 7 and 0

\*

\* Voltage source with series resistance: equivalent circuit of TLM link

Vtlm8 1008 0 DC 0.0

Rtlm8 1008 7 #Z0\_TLM

\*

\*

\* Voltage source required for the voltage source controlling the breakpoint time

Vbreak time\_node 0 DC 0.0

\*

\* Model to be included in the GGI\_TLM simulation connected to node 1

\* INPUT VOLTAGE

VIN 1 0 DC 20

\* Model to be included in the GGI\_TLM simulation connected to node 2

\* Voltage controlled switch model

SW1 2 0 8 0 SWitch1

\* Model to be included in the GGI\_TLM simulation connected to node 3

\* Diode model

D1 0 3 DSCH

\* Model to be included in the GGI\_TLM simulation connected to node 4

\* Inductor model

L1 4 0 50UH IC=1A

\* Model to be included in the GGI\_TLM simulation connected to node 5

\* Capacitor model

C1 5 0 2UF IC=5V

\* Model to be included in the GGI\_TLM simulation connected to node 6

\* LOAD model

RL 6 0 5.0

\* Model to be included in the GGI\_TLM simulation connected to node 7

\* SWITCH CONTROL SIGNAL

R10 7 0 1MEG

\* Model to be included in the GGI\_TLM simulation connected to node 8

\* Gate-source impedance

RGS 8 0 1MEG

VCTRL 8 0 PULSE(0V 6V 0 0.0001US 0.0001US 0.05US 0.2US)

\*

.MODEL SWitch1 SW Vt=5V Vh=0.2V RON=0.01 ROFF=1MEG

\*

.MODEL DSCH D( IS=0.0002 )

\*

\*

\* Control for transient simulation

.TRAN #dt\_out #tmax\_ngspice 0.0 #dt\_ngspice UIC

\*

.END

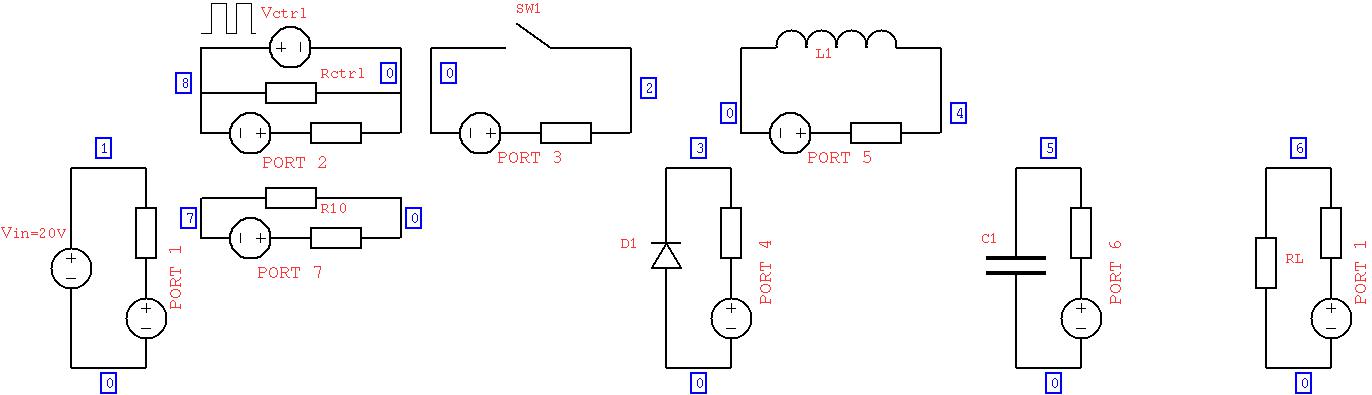


Figure Ngspice model for the converter with the GGI\_TLM linking components added.

The 3D GGI\_TLM simulation model is shown in Figure 24 and Figure 25.

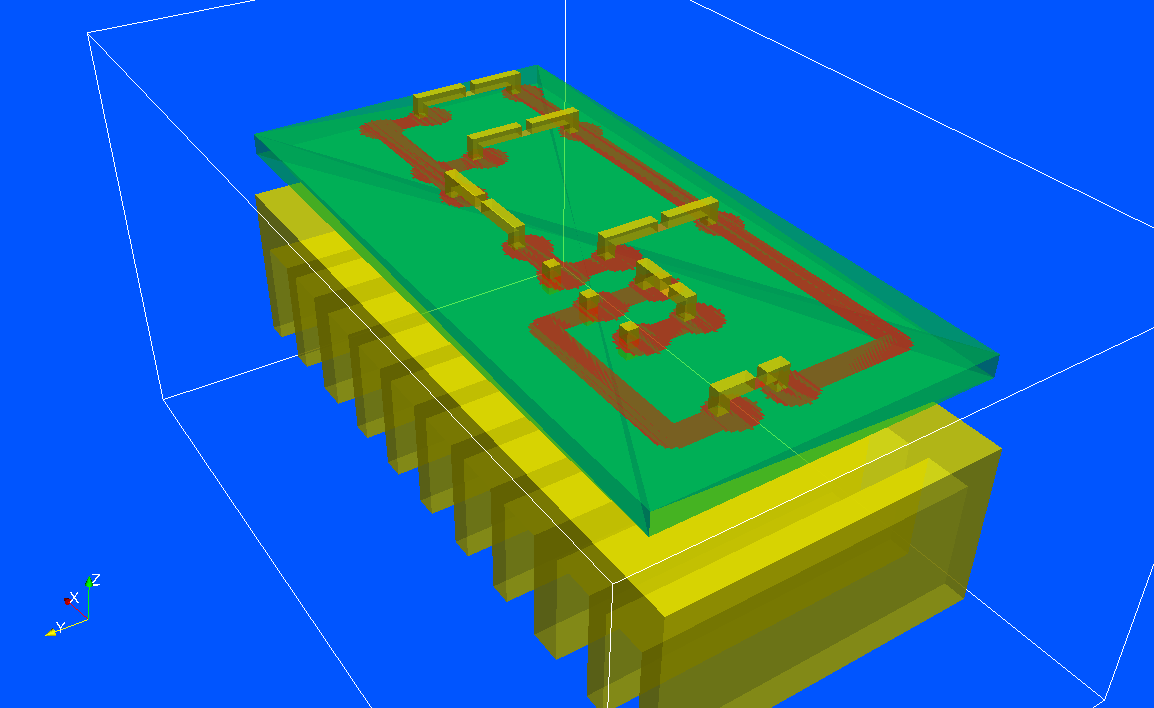


Figure 3D GGI\_TLM converter model including heatsink

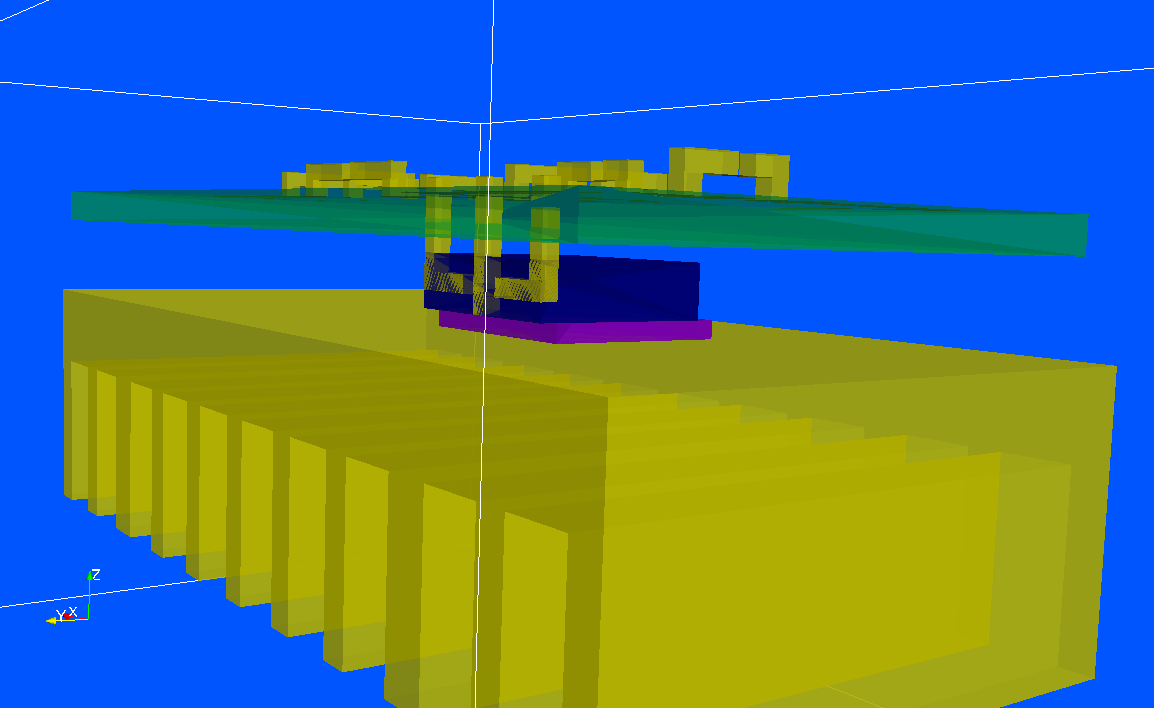


Figure 3D GGI\_TLM converter model showing the MOSFET package model and ceramic tile between it and the heatsink

- Specification of outputs for surface current on PCB tracks

- Voltage output results: show ringing on switch voltage due to inductance of loop on input side.

- Field pattern output results

- Re-run with a suitable snubber circuit in place.

## Buck converter model including SACAMOS cable model and LISN for conducted emissions analysis

- Circuit diagram

-Impedance synthesis process: Ngspice component models

- Initial ngspice model to calculate the initial conditions for the GGI\_TLM- Ngspice model

- GGI\_TLM-Ngspice model development: Port definitions and Ngspice circuit

- Specification of initial conditions

- Specification of outputs for surface current on PCB tracks

- Voltage output results

- Field pattern output results

- Comparison with measured results

# References

1. GGI\_TLM Open source project, www.github.com/ggiemr/GGI\_TLM
2. Ngspice, ngspice.sourceforge.net
3. Gerber format, www.ucamco.com/en/gerber
4. J.W. Park, P.P.M. So, W. J. R. Hoefer, “Lumped and distributed Device Embedding Techniques in Time Domain TLM Field Models,” IEEE MTT
5. P. P. M. So, W. J. R. Hoefer, “A TLM-SPICE Interconnection Framework for Coupled Field and Circuit Analysis in the Time Domain,” IEEE Trans MTT, Vol 50, No 12, December 2002
6. P.P.M. So, W. J. R. Hoefer, “A General Framework for SPICE-TLM Interconnection,” IEEE MTT
7. H. Du, P. P. M. So, W. J. R. Hoefer, “Embedding Current-Coupled Lumped Networks in TLM Models,” IEEE MTT
8. M. N. de Sousa, L. R. A. X. Menezes, P. Russer, “Hybrid TLM-SPICE Method in Electromagnetics Problems,”
9. <https://github.com/chrissmartt/vector_fit>
10. <https://github.com/chrissmartt/network_synthesis>
11. <https://github.com/chrissmartt/Spice_impedance_synthesis>