SonnetLab Attenuator Tutorial

By Bashir Souid

This work was supervised by Dr. Serhend Arvas.

In this tutorial we will build a single network Sonnet netlist project. The circuit design for this tutorial will match the attenuator tutorial presented in Chapter 13 of the Sonnet User Guide titled "Netlist Project Analysis". This tutorial will show users how to build the four netlist circuits presented in the user guide: att_cascade.son, att_combine.son, att_lumped.son, att_lumped2.son.

The .m files for this tutorial can be found at

```
<SonnetLab directory>\Tutorials\Attenuator\att_cascade.m
<SonnetLab directory>\Tutorials\Attenuator\att_combine.m
<SonnetLab directory>\Tutorials\Attenuator\att_lumped.m
<SonnetLab directory>\Tutorials\Attenuator\att_lumped2.m
```

att_cascade

This section of the tutorial will build att_cascade.son. In order to build the project we will first need to instantiate a Sonnet project in the Matlab workspace by using the following command:

```
Project=SonnetProject();
```

The above command will create a Sonnet geometry project; we want this project to be a Sonnet netlist project. SonnetLab makes it easy to convert Sonnet geometry projects into Sonnet netlist projects. This can be accomplished using the following command:

```
Project.initializeNetlist();
```

The above command will convert the Sonnet project into a default Sonnet netlist project that has the same settings as a Sonnet netlist project built using the Sonnet project editor. The initializeNetlist() method can be used on any Sonnet project to convert it to a default Sonnet netlist project.

The circuit designed in the Sonnet User Guide used megahertz as the frequency unit instead of gigahertz. The frequency unit selection can be easily changed to megahertz using the following command:

```
Project.changeFrequencyUnit('MHZ');
```

When a new Sonnet netlist project is created a network element is automatically added to the project. In this tutorial we will delete the default network and add a new network that matches the network from the user guide. The following command will delete the first network element from the Sonnet project.

```
Project.CircuitElementsBlock.ArrayOfNetworkElements(1)=[];
```

The new network should be named `'RESNET'`, have two external ports numbered one and three, and have a port resistance of 50 ohms. The desired network can be added to the project using the following command:

```
Project.addNetworkElement('RESNET',[1 3],50);
```

One of the greatest strengths of the Sonnet netlist engine is that it has support for incorporating touchstone files and Sonnet project files as netlist elements. These features allow users to incorporate many different design models in a network. The user guide adds two S2P files to the project; they can be added with the following commands:

```
Project.addDataResponseFileElement('att_res16.s2p',[1,2]);
Project.addDataResponseFileElement('att_res16.s2p',[2,3]);
```

The first command will incorporate the touchstone file `'att_res16.s2p'` into the network at nodes one and two. The second command will incorporate the touchstone file `'att_res16.s2p'` into the network at nodes two and three.

This tutorial will simulate the netlist using a linear frequency sweep from 200MHz to 400MHz in steps of 100MHz. The following command will add the desired frequency sweep to the project and select it as the chosen frequency sweep for simulation.

```
Project.addSimpleFrequencySweep(200,400,100);
```

When a new project is created using SonnetLab it will need to be saved to the hard drive before it can be simulated. The first time a user wants to save a particular Sonnet project to the hard drive they must specify a filename for the project; this can be accomplished using the saveAs() method as follows:

```
Project.saveAs('att_cascade.son');
```

The above command will save the Sonnet project specified by the variable Project to the hard drive with a file named 'SingleStub.son'. After the user does a single saveAs() command they are then able to use the save() command which will save the file to the hard drive using the same filename that was specified by the most recent call to saveAs(). If a Sonnet project file is opened from the hard drive (rather than being created from scratch from within Matlab) the user does not need to call saveAs() in order to specify a filename for the project file; the save() function will overwrite the read project file. At any time any Sonnet project may be saved with saveAs() in order to specify a different filename; the result of which will also cause all later calls of save() to be saved to the new filename.

Now that we have specified the simulation settings and have saved the project we can simulate the project using the command:

```
Project.simulate();
```

When the simulation is complete the user may examine the results using the Sonnet response viewer. SonnetLab can automate opening the Sonnet response viewer by issueing the following command:

```
Project.viewResponseData();
```

att_combine

This section of the tutorial will build att_cascade.son. In order to build the project we will first

need to instantiate a Sonnet project in the Matlab workspace by using the following command:

```
Project=SonnetProject();
```

The above command will create a Sonnet geometry project; we want this project to be a Sonnet netlist project. Just like in the att_cascade tutorial, this tutorial will use the initializeNetlist() method to convert the geometry project into a netlist project.

```
Project.initializeNetlist();
```

The circuit designed in the Sonnet User Guide used megahertz as the frequency unit instead of gigahertz. The frequency unit selection can be easily changed to megahertz using the following command:

```
Project.changeFrequencyUnit('MHZ');
```

The att_combine circuit has the same network as att_cascade. The desired network can be utilized by first deleting the default network and adding a new network to the project that has the desired values. The following command will delete the first network element from the Sonnet project.

```
Project.CircuitElementsBlock.ArrayOfNetworkElements(1)=[];
```

The new network should be named `'ATTEN'`, have two external ports numbered one and three, and have a port resistance of 50 ohms. The desired network can be added to the project using the following command:

```
Project.addNetworkElement('ATTEN',[1 3],50);
```

The circuit in att_combine will incorporate the same two touchstone files that were incorporated in att_cascade along with a Sonnet project element. The touchstone file elements can be added to the project with the following commands:

```
Project.addDataResponseFileElement('att_res16.s2p',[1,2]);
Project.addDataResponseFileElement('att_res16.s2p',[2,3]);
```

The first command will incorporate the touchstone file `'att_res16.s2p'` into the network at nodes one and two. The second command will incorporate the touchstone file `'att_res16.s2p'` into the network at nodes two and three.

A Sonnet project file can be added to a netlist using the addProjectFileElement() method. The addProjectFileElement() method requires the following arguments:

1. The filename for the project
2. The nodes at which to place the project file element
3. Whether to use the project's frequency sweep or the subproject's frequency sweep

The third argument should be either a zero or a one; a value of zero indicates that the simulation should use the frequency sweep from the main project and a value of one inducates that the simulation should use the frequency sweep from the project file element. The following command

is used in this tutorial to add the project file 'att_res67.son' as an element in the netlist

```
Project.addProjectFileElement('att_res67.son',[2,0],1);
```

This tutorial will simulate the netlist using a linear frequency sweep from 200MHz to 400MHz in steps of 100MHz. The following command will add the desired frequency sweep to the project and select it as the chosen frequency sweep for simulation.

```
Project.addSimpleFrequencySweep(200,400,100);
```

When a new project is created using SonnetLab it will need to be saved to the hard drive before it can be simulated. The first time a user wants to save a particular Sonnet project to the hard drive they must specify a filename for the project; this can be accomplished using the saveAs() method as follows:

```
Project.saveAs('att_combine.son');
```

Now that the simulation settings have been specified and the project can be simulated using the following command:

```
Project.simulate();
```

When the simulation is complete the user may examine the results using the Sonnet response viewer. SonnetLab can automate opening the Sonnet response viewer by issueing the following command:

```
Project.viewResponseData();
```

att_lumped

This section of the tutorial will build att_lumped.son. In order to build the project we will first need to instantiate a Sonnet project in the Matlab workspace by using the following command:

```
Project=SonnetProject();
```

The above command will create a Sonnet geometry project; we want this project to be a Sonnet netlist project. This tutorial will use the initializeNetlist() method to convert the geometry project into a netlist project.

```
Project.initializeNetlist();
```

The circuit designed in the Sonnet User Guide used megahertz as the frequency unit instead of gigahertz. The frequency unit selection can be easily changed to megahertz using the following command:

```
Project.changeFrequencyUnit('MHZ');
```

In this tutorial we will delete the default network and add a new network that matches the one from the user guide. The following command will delete the first network element from the Sonnet project.

```
Project.CircuitElementsBlock.ArrayOfNetworkElements(1)=[];
```

The new network should be named `'ATTEN'`, have two external ports numbered one and two, and have a port resistance of 50 ohms. The desired network can be added to the project using the following command:

```
Project.addNetworkElement('ATTEN',[1 2],50);
```

This tutorial will add three resistors to the project. The value for the units is defined globally for the project; the default unit selection is ohms. The first resistor is 16.11 ohms and should be placed between ports three and four. The second resistor will have a resistance of 50 ohms and will be placed between ports five and six. The third resistor is will have a resistance of 67.77 ohms and be placed between ports seven and eight.

```
Project.addResistorElement(3,4,16.77);
Project.addResistorElement(5,6,16.77);
Project.addResistorElement(7,8,67.11);
```

The tutorial adds a Sonnet project file element to the project. The following command is used in this tutorial to add the project file `'att_lgeo.son'` as an element in our netlist.

```
Project.addProjectFileElement('att_lgeo.son',[1 2 3 4 5 6 7 8],1);
```

This tutorial will simulate the netlist using a linear frequency sweep from 200MHz to 400MHz in steps of 100MHz. The following command will add the desired frequency sweep to the project and select it as the chosen frequency sweep for simulation.

```
Project.addSimpleFrequencySweep(200,400,100);
```

When a new project is created using SonnetLab it will need to be saved to the hard drive before it can be simulated. The first time a user wants to save a particular Sonnet project to the hard drive they must specify a filename for the project; this can be accomplished using the saveAs() method as follows:

```
Project.saveAs('att_lumped.son');
```

Now that the simulation settings have been specified and the project can be simulated using the following command:

```
Project.simulate();
```

When the simulation is complete the user may examine the results using the Sonnet response viewer. SonnetLab can automate opening the Sonnet response viewer by issueing the following command:

```
Project.viewResponseData();
```

att_lumped2

This section of the tutorial will build att_lumped.son. In order to build the project we will first need to instantiate a Sonnet project in the Matlab workspace by using the following command:

```
Project=SonnetProject();
```

The above command will create a Sonnet geometry project; we want this project to be a Sonnet netlist project. This tutorial will use the initializeNetlist() method to convert the geometry project into a netlist project.

```
Project.initializeNetlist();
```

The circuit designed in the Sonnet User Guide used megahertz as the frequency unit instead of gigahertz. The frequency unit selection can be easily changed to megahertz using the following command:

```
Project.changeFrequencyUnit('MHZ');
```

In this tutorial we will delete the default network and add a new network that matches the one from the user guide. The following command will delete the first network element from the Sonnet project.

```
Project.CircuitElementsBlock.ArrayOfNetworkElements(1)=[];
```

The new network should be named `'ATTEN'`, have two external ports numbered one and two, and have a port resistance of 50 ohms. The desired network can be added to the project using the following command:

```
Project.addNetworkElement('ATTEN',[1 2],50);
```

We will now define three variables that will be used to store the impedance values for the resistor elements:

```
Project.defineVariable('Z3',16.77);
Project.defineVariable('Z4',16.77);
Project.defineVariable('Z5',67.11);
```

We will now add the resistors to the project with the following commands:

```
Project.addResistorElement(3,[],'Z3');
Project.addResistorElement(4,[],'Z4');
Project.addResistorElement(5,[],'Z5');
```

The value for the second node in the resistors is an empty matrix ([]). When the node index is an empty matrix the other end of the resistor is not connected to any node of the circuit.

The tutorial adds a Sonnet project file element to the project. The following command is used in this tutorial to add the project file `'att_lgeo2.son'` as an element in our netlist

```
Project.addProjectFileElement('att_lgeo2.son',[1 2 3 4 5],1);
```

This tutorial will simulate the netlist using a linear frequency sweep from 200MHz to 400MHz in steps of 100MHz. The following command will add the desired frequency sweep to the project and select it as the chosen frequency sweep for simulation.

```
Project.addSimpleFrequencySweep(200,400,100);
```

When a new project is created using SonnetLab it will need to be saved to the hard drive before it can be simulated. The first time a user wants to save a particular Sonnet project to the hard drive they must specify a filename for the project; this can be accomplished using the saveAs() method as follows:

```
Project.saveAs('att_lumped2.son');
```

Now that we have specified the simulation settings and have saved the project we can simulate the project using the command:

```
Project.simulate();
```

When the simulation is complete the user may examine the results using the Sonnet response viewer. SonnetLab can automate opening the Sonnet response viewer by issueing the following command:

```
Project.viewResponseData();
```