100 Elwood Davis Road ♦ North Syracuse, NY 13212 ♦ USA

SonnetLab Project Decompiler

Specialists in High-Frequency Electromagnetic Software
(315) 453-3096 Fax: (315) 451-1694 http://www.sonnetsoftware.com

Version 7.0

Introduction

The project decompiler tool that ships with the SonnetLab toolbox is designed to give Sonnet users the ability to reverse engineer a Sonnet Project file into a series of SonnetLab compatible Matlab commands. The project decompiler can reverse engineer Sonnet project files generated with SonnetLab or any of the Sonnet Suites.

The project decompiler tool will read a Sonnet project object from the hard drive, analyze its internal structure and generate a series of SonnetLab compatible commands that, when executed, will generate an identical Sonnet Project object in Matlab.

The project decompiler can be used as an educational tool to learn how to call SonnetLab to add or modify various elements a project. The project decompiler may also be used to analyze a template project file which could then be used as a basis to build variations of the project file.

Requirements

The SonnetLab Project Decompiler ships with and is compatible with SonnetLab version 6.0 and higher. Before using the SonnetLab or the decompiler users will need to add the location of the SonnetLab "scripts" to their Matlab path. For detailed installation instructions see the installation guide present in the "Documentation" folder that ships with SonnetLab.

General Instructions

The SonnetLab Project Decompiler tool can be called with the "SonnetProjectDecompile" function. The decompiler accepts a string as an argument which is the path and filename of the file to be analyzed by the project decompiler.

```
>> [aCommands aCommandsForEachBlock]=SonnetProjectDecompile(Filename);
```

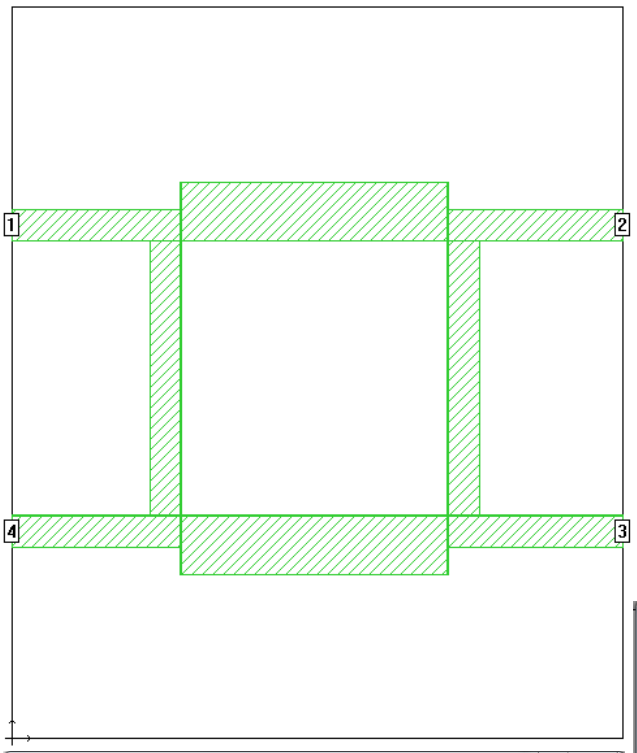
The SonnetLab Project Decompiler produces two output variables. The first output variable is a vertically concatenated vector of strings. Each row in the string vector is a SonnetLab compatible command which adds or modifies a Sonnet project feature. A user may use this list of commands to generate a SonnetLab object which is identical to the input project file.

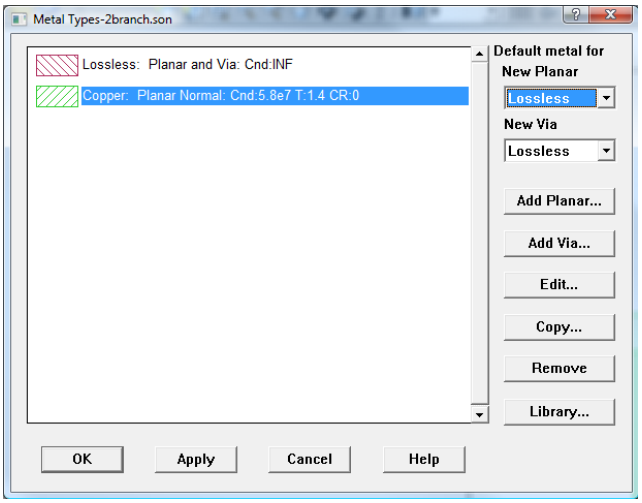
The second output parameter is a cell array. There is a single cell for each file block in the project and each cell in the array is a vertically concatenated string of commands necessary to import the settings for one particular block of the Sonnet project file to a Matlab SonnetLab project object. This can be useful for many tasks which involve a subset of the Sonnet project file. For example a user may wish to have a list of the SonnetLab commands necessary to modify a Project object's frequency sweep information with that of a saved Sonnet project file.

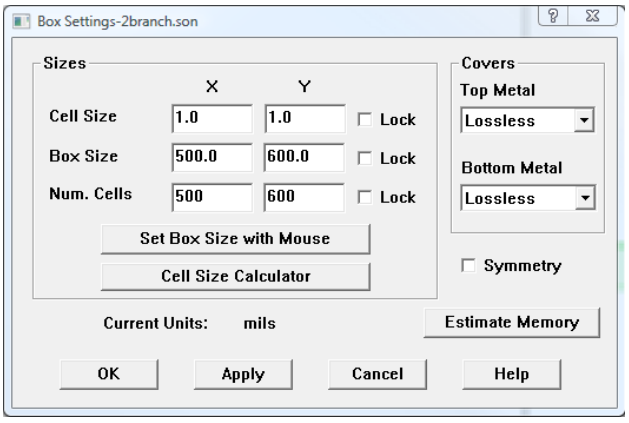
Most users will primarily be interested in the first output parameter and will wish to ignore the second output parameter. The second output parameter is useful for more experienced users who would like to only import a small subset of a Sonnet project rather than the commands for the entire project.

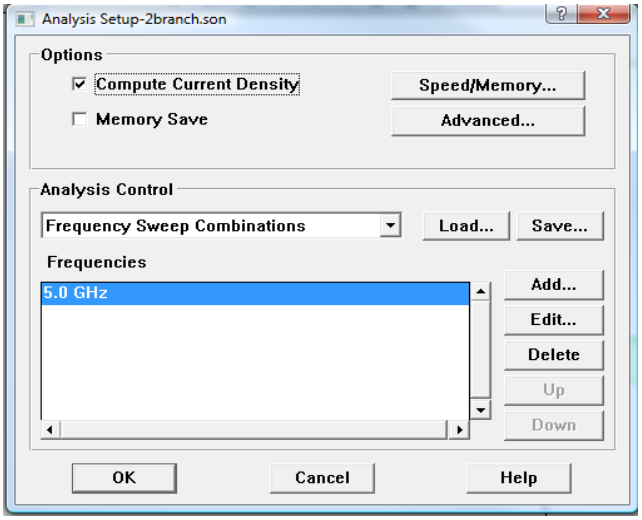
Example – Branchline Coupler

In this example we will examine the output of the following simple branchline coupler. The transmission lines in the coupler are realized in copper metal. The project is simulated at a single frequency of 5 GHZ with current calculations enabled.









The following is the output of the project decompilation. Note that the output includes several data elements that are not critical to a successful duplication of a Sonnet project. For example the LicenseString value is not critical to a project's construction. The project decompiler tool strives to be complete whenever possible and prefers to provide extra information rather than suppress information.

```
>> SonnetProjectDecompile('2branch.son')


  ans =

  Project=SonnetProject();
  Project.VersionOfSonnet='14.0.alpha';
  Project.HeaderBlock.LicenseString=' sonnet9.aa.99999';
  Project.HeaderBlock.DateTheFileWasLastSaved=' 11/21/2011 12:42:52';
  Project.HeaderBlock.InformationAboutHowTheProjectWasCreated=' example 12.56 11/23/2009
  22:31:05';
  Project.HeaderBlock.InformationAboutHowTheProjectWasLastSaved=' xgeom "14.01.002 alpha"';
  Project.HeaderBlock.DateTheProjectWasSavedWithMediumImportanceChanges=' 11/21/2011 12:42:51';
  Project.HeaderBlock.DateTheProjectWasSavedWithHighImportanceChanges=' 11/21/2011 12:42:51';
  Project.changeLengthUnit('mils');
  Project.addSimpleFrequencySweep(4,5.5,0.1);
  Project.addAbsFrequencySweep(4,5.5);
  Project.addStepFrequencySweep(5);
  Project.changeSelectedFrequencySweep('STD');
  Project.enableCurrentCalculations();
  Project.ControlBlock.CacheAbs=1;
  Project.ControlBlock.TargetAbs=300;
  Project.ControlBlock.QFactorAccuracy='N';
  Project.defineNewNormalMetalType('Copper',58000000,0,1.4);
  Project.changeTopCover('Lossless');
  Project.changeBottomCover('Lossless');
  Polygon=Project.addMetalPolygonEasy(0,[138.5 356.5 356.5 138.5 138.5],[144.5 144.5
  192 192 144.5],'Copper');
  Polygon.DebugId=9;
  Polygon=Project.addMetalPolygonEasy(0,[138.5 356.5 356.5 138.5 138.5],[417.5 417.5
  465.5 465.5 417.5],'Copper');
  Polygon.DebugId=10;
  Polygon=Project.addMetalPolygonEasy(0,[113.5 138.5 138.5 113.5 113.5],[192 192 417.5 417.5
  192],'Copper');
  Polygon.DebugId=11;
  Polygon=Project.addMetalPolygonEasy(0,[356.5 382 382 356.5 356.5],[192 192 417.5 417.5
  192],'Copper');
  Polygon.DebugId=12;
  Polygon=Project.addMetalPolygonEasy(0,[357 500 500 357 357],[166.5 166.5 192 192
  166.5],'Copper');
  Polygon.DebugId=13;
  Polygon=Project.addMetalPolygonEasy(0,[356.5 500 500 356.5 356.5],[417.5 417.5 443 443
  417.5],'Copper');
  Polygon.DebugId=15;
  Polygon=Project.addMetalPolygonEasy(0,[0 138.5 138.5 0 0],[417.5 417.5 443 443
  417.5],'Copper');
  Polygon.DebugId=17;
  Polygon=Project.addMetalPolygonEasy(0,[0 138.5 138.5 0 0],[166.5 166.5 192 192
  166.5],'Copper');
  Polygon.DebugId=19;
  Project.changeBoxSizeX(500);
  Project.changeBoxSizeY(600);
  Project.changeNumberOfCellsX(500);
  Project.changeNumberOfCellsY(600);
  Project.deleteLayer(2);
  Project.deleteLayer(1);
  Project.addDielectricLayer('Air',100,1,1,0,0,0,2);
  Project.addDielectricLayer('Alumina',25,9.8,1,0,0,0,2);
  Polygon=Project.getPolygon(5);
  Project.addPortStandard(Polygon,2,50,0,0,0,2);
  Polygon=Project.getPolygon(6);
  Project.addPortStandard(Polygon,2,50,0,0,0,3);
  Polygon=Project.getPolygon(7);
  Project.addPortStandard(Polygon,4,50,0,0,0,4);
  Polygon=Project.getPolygon(8);
  Project.addPortStandard(Polygon,4,50,0,0,0,1);
  Project.GeometryBlock.LocalOrigin.X=0;
  Project.GeometryBlock.LocalOrigin.Y=600;
  Project.GeometryBlock.LocalOrigin.Locked='U';
```

Additional Examples

SonnetLab version 8.0 includes several examples of using the Sonnet Project Decompiler. These can be found in the “Tutorials” and “Examples” folders where the SonnetLab archive has been extracted.

Contact

Your feedback is important to us. If you have any questions or comments about SonnetLab, please contact Sonnet Support by email at support@sonnetsoftware.com.

Please make sure you are using the most up to date version of SonnetLab before submitting a bug report. When submitting a bug report please include the Sonnet project file that generated the error (Sonnet project files have the extension .son) and the output from the command “SonnetMatlabVersion”. The more information that that we receive the faster it will be for us to resolve the issue and contact you back.