cādence™

Allegro Constraint Manager: Advanced Constraints Tutorial

Product Version 16.01
December 2007

Contents

Preface

Purpose of This Tutorial

The *Advanced Constraints Tutorial* provides lessons, sample design files, and text files to help you address situations in Constraint Manager where:

- There is no existing constraint for a particular requirement.
- The constraint is not a simple static value or is dependent on other values or objects.
- The validation of a constraint cannot be done using existing measurements.

This tutorial is based on Release 16.01. Its purpose is to introduce you to the following advanced constraints features: Formulas, Predicates, User-defined Properties, and User-defined Measurements.

Audience

This tutorial is intended to help users address the advanced features introduced in Release 16.01. To work successfully, you must have advanced knowledge of the SPB tools.

Sample Files

Included with the tutorial are two board designs and four text files for use in performing the tasks in this tutorial. You must follow these tasks in the order in which they are presented to have a successful results.

To locate the sample files, see the *`<install_dir>/doc/cmadvcnstut/examples`* directory.

Syntax Conventions

This list describes the syntax conventions used in the tutorial.

`literal`	Key words that you must enter literally. These keywords represent commands (functions, routines) or option names.

`Courier` font	Indicates command line examples.
UI	Words in this font refer to menus, labels, fields, or tabs on the user interface.
`variable`	Words in this font refer to arguments for which you must substitute a value.

Introduction

The *Advanced Constraints Tutorial* covers the following tasks:

- ❑ Creating user-defined constraints
- ❑ Customizing any property or constraint with a formula
- ❑ Creating formulas with pre-defined Predicates
- ❑ Using Formulas with pre-defined constraints
- ❑ Creating user-defined measurements to populate user-defined actuals

Creating User-defined Constraints

In this first example, the requirement is to constrain nets to a particular target length with different plus and minus tolerances. Currently, this constraint does not exist in Constraint Manager as a pre-defined capability.

Using Allegro PCB Editor XL or GXL:

1. Open the `start.brd` located in the `<install_dir>/doc/cmadvcnstut/examples` directory.

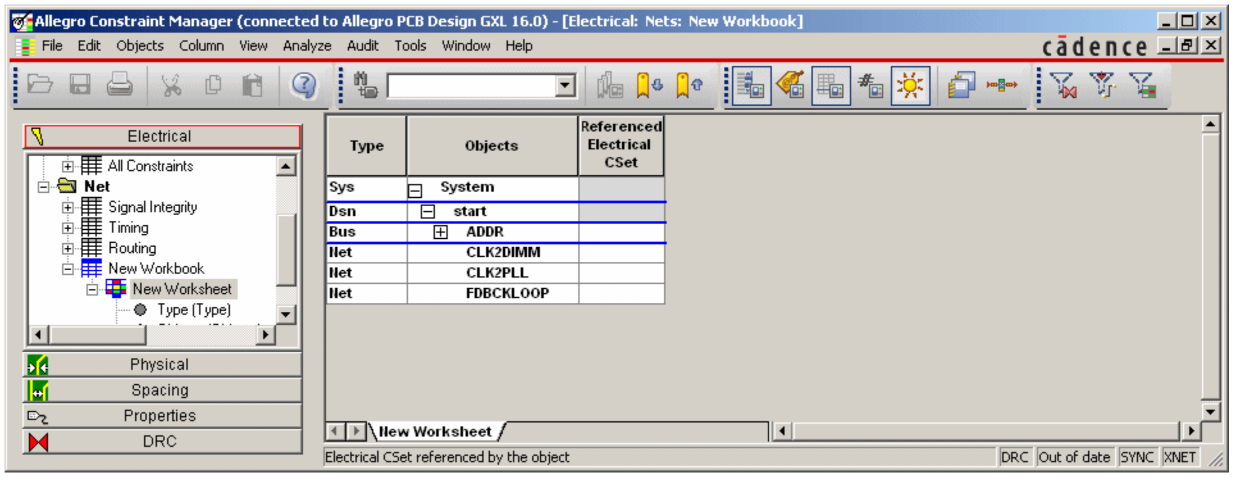
Important

The board and text files for this tutorial are in the experimental stage and continue to be developed, internally. Once complete, these revised and verified support files will accompany the tutorial in a follow-on patch to this release. Check the *Constraint Manager Known Problems and Solutions* at

http://sourcelink.cadence.com.

2. Launch Constraint Manager. In the *Worksheet Selector*, choose the *Electrical* domain, right-click on the *Net* folder, and choose *Customize Worksheet* to enter *Customize* mode.
3. Right-click on the *Net* folder again, and choose *Add New Workbook.*
4. Open the associated *New Worksheet* by double clicking on it in the *Worksheet Selector*.

Figure 1-1 New Worksheet


Allegro Constraint Manager (connected to Allegro PCB Design GXL 16.0) - [Electrical: Nets: New Workbook]
File Edit Objects Column View Analyze Audit Tools Window Help
cādence
Electrical
All Constraints
Net
Signal Integrity
Timing
Routing
New Workbook
New Worksheet
Type (Type)
Physical
Spacing
Properties
DRC
New Worksheet
DRC
Out of date
SYNC
XNET

Type	Objects	Referenced Electrical CSet
Sys	System	
Dsn	start	
Bus	ADDR	
Net	CLK2DIMM	
Net	CLK2PLL	
Net	FDBCKLOOP	

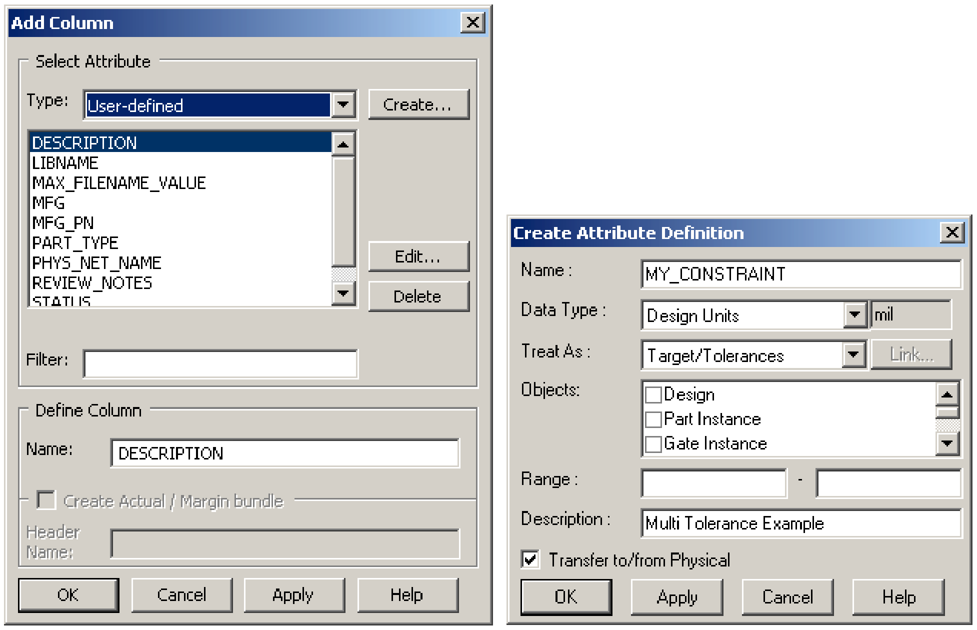
Electrical CSet referenced by the object

5. Right-click on *New Worksheet* in the *Worksheet Selector* and choose *Add Column.*

   Adding workbooks and worksheets was part of SPB Release 15.5 with the introduction of the `Customize` mode. This also changed the way to add user-defined properties.

6. In the Add Column dialog box, leave the *Type* field set to `User-defined`, and click the *Create* button.

Figure 1-2 Add Column Dialog Box/Create Attribute Definition Dialog Box



Type: User-defined

DESCRIPTION
LIBNAME
MAX_FILENAME_VALUE
MFG
MFG_PN
PART_TYPE
PHYS_NET_NAME
REVIEW_NOTES
STATUS

Filter:

Name: DESCRIPTION

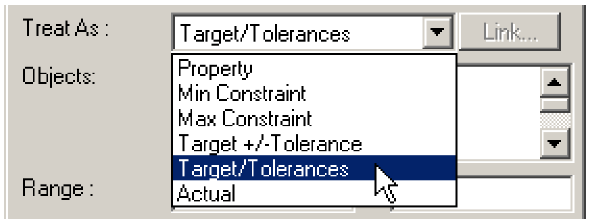
Name : MY_CONSTRAINT
Data Type : Design Units mil
Treat As : Target/Tolerances
Objects: ☐ Design ☐ Part Instance ☐ Gate Instance
Range : -
Description : Multi Tolerance Example
☑ Transfer to/from Physical

7. Complete the Create Attribute Definition dialog box as shown in Figure 1-2.

   Make sure that there are no spaces in the *Name* field and set the *Data Type* to `Design Units`. Set the *Treat As* drop-down list as shown. Leave the *Objects* field as is; this field will be discussed in the Creating User-defined Measurements to Populate User-defined Actuals on page 26. The *Range* field is used in cases where the legal values for a constraint need to be defined.

   Notice the six options available in the *Treat As* drop-down list. Prior to this release, *Property* was the only option. *Actual* is just a measurement with no associated constraint. *Target/ Tolerances* allows for separate plus and minus tolerance values whereas *Target +/- Tolerance* allows for a single tolerance value.

   Figure 1-3 Treat As Drop-Down List



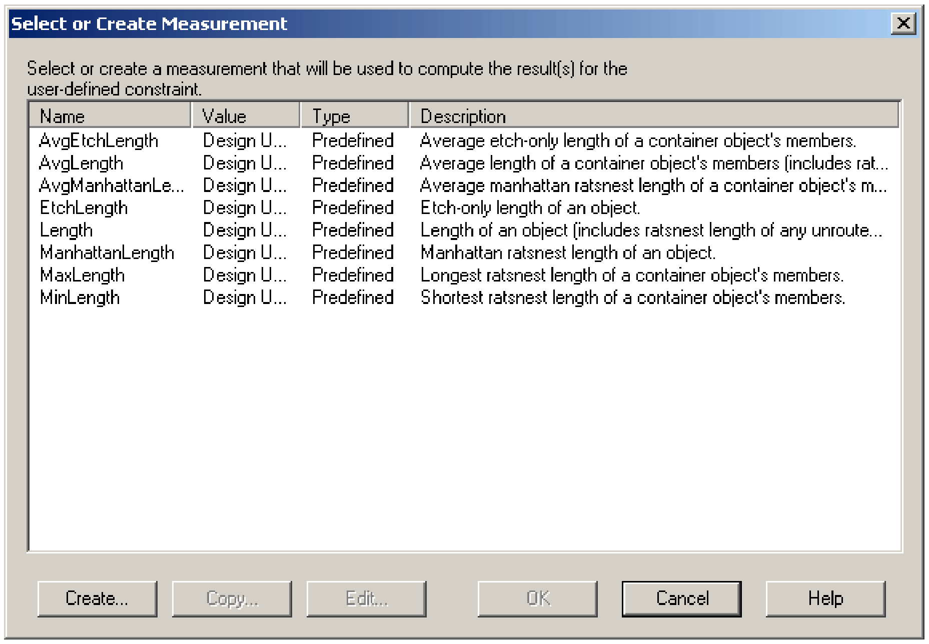
   Since different plus and minus tolerance values are required for this example, choose *Target/Tolerances*.

8. Click *OK*.

   This launches the Select or Create Measurement dialog box. The choices listed are based on the *Data Type* that you chose in the Create Attribute Definition dialog box (Figure 1-2). At this point, there are no user-defined Measurements so the dialog box lists the supplied pre-defined Measurements.

   Note: It is possible that this dialog box does not list any Measurements based on the *Data Type* you previously set.

Figure 1-4 Select or Create Measurement Dialog Box



Select or Create Measurement

Select or create a measurement that will be used to compute the result(s) for the user-defined constraint.

Name	Value	Type	Description
AvgEtchLength	Design U...	Predefined	Average etch-only length of a container object's members.
AvgLength	Design U...	Predefined	Average length of a container object's members (includes rat...
AvgManhattanLe...	Design U...	Predefined	Average manhattan ratsnest length of a container object's m...
EtchLength	Design U...	Predefined	Etch-only length of an object.
Length	Design U...	Predefined	Length of an object (includes ratsnest length of any unroute...
ManhattanLength	Design U...	Predefined	Manhattan ratsnest length of an object.
MaxLength	Design U...	Predefined	Longest ratsnest length of a container object's members.
MinLength	Design U...	Predefined	Shortest ratsnest length of a container object's members.

Notice that the *Copy* and *Edit* buttons are grayed out, even after selecting a measurement. These buttons apply to user-defined Measurements only. This topic will be discussed in the Creating User-defined Measurements to Populate User-defined Actuals on page 26.

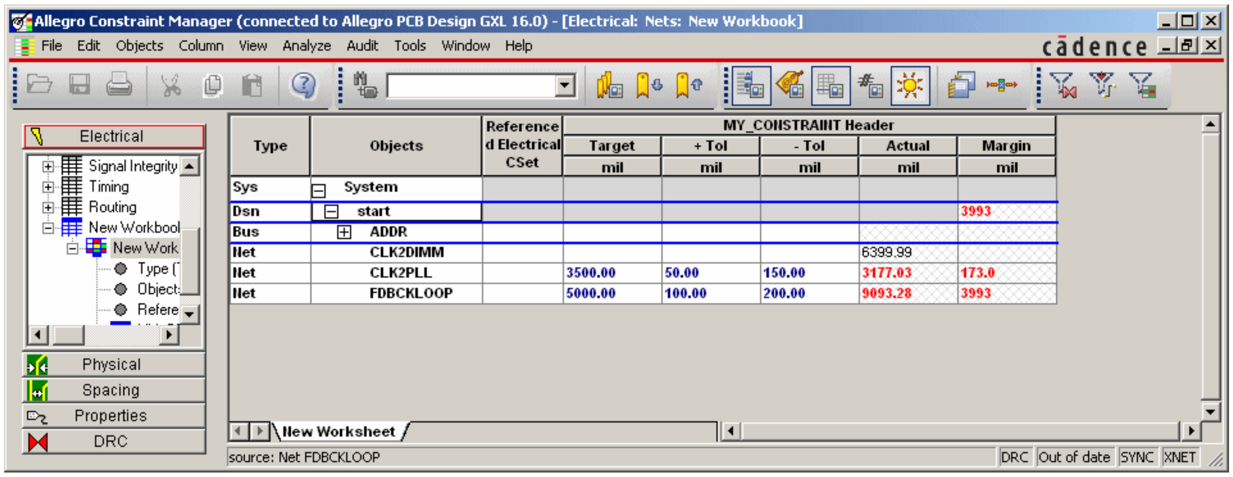
9. Choose *EtchLength* from the list and click *OK.*

The constraint value will be compared to the total etch length of the constrained net.

10. In the Add Column dialog box (Figure 1-2) with the new columns added and the bottom of the dialog box updated, click *OK*. Then click *OK* in any confirmer dialog box that appears.

11. In the new columns, add some constraint and tolerance values to the *FDBCKLOOP* and *CLK2PLL* nets, then right-click on design name (*start)* in the *Objects* column, and choose *Analyze*.

Figure 1-5 Adding Constraints and Tolerance Values


Allegro Constraint Manager (connected to Allegro PCB Design GXL 16.0) - [Electrical: Nets: New Workbook]
File Edit Objects Column View Analyze Audit Tools Window Help
cadence
Electrical
Signal Integrity
Timing
Routing
New Workbook
New Work
Type
Objects
Refere
Physical
Spacing
Properties
DRC
New Worksheet
DRC
Out of date
SYNC
XNET

Type	Objects	Referenced Electrical CSet	MY_CONSTRAINT Header				
			Target	+ Tol	- Tol	Actual	Margin
			mil	mil	mil	mil	mil
Sys	System						
Dsn	start						3993
Bus	ADDR						
Net	CLK2DIMM					6399.99	
Net	CLK2PLL		3500.00	50.00	150.00	3177.03	173.0
Net	FDBCKLOOP		5000.00	100.00	200.00	9093.28	3993

source: Net FDBCKLOOP

Notice in Figure 8 that the measurement (*Actual*) is calculated even if the *Object* is not constrained. This is true for all added user-defined *Actuals* (whether or not they are part of a user-defined constraint).

12. If the constraint values that you entered all pass, adjust one *Target* constraint value to cause an error (as shown in Figure 1-5).

You may notice that there is no DRC in PCB Editor to indicate this situation. At this point, Pass / Fail status is indicated in Constraint Manager only.

13. Make sure to undo any etch changes that you made in PCB Editor. Verify this by re-analyzing and comparing the values in the *Actual* column of your Constraint Manager session to the values in the *Actual* column shown in Figure 1-5.

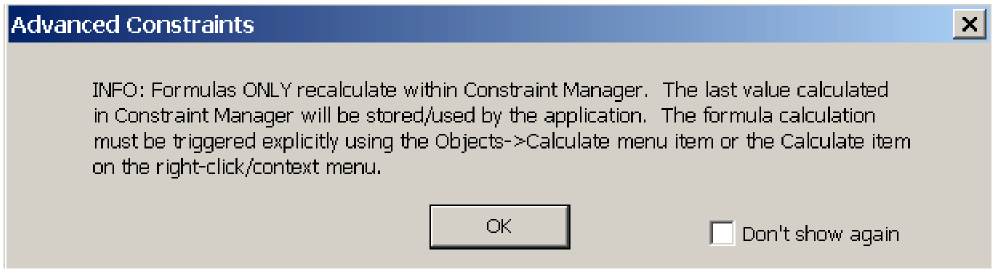
Customizing a Property or Constraint Using Formulas Created with Cell Selections

In the previous example, you created a user-defined constraint bundle where the constraint is a static value like other constraints in Constraint Manager.

But what happens if there is a case where the constraint value depends on other values? For example, what if the *FDBCKLOOP* net must equal the sum of nets *CLK2PLL* and *CLK2DIMM*? This is a good example of a target constraint where the target is of the "netA + netB" type.

Continuing from the previous example:

1. Right-click in the cell corresponding to the *Target* constraint for the *FDBCKLOOP* net and choose *Formula*. This launches the following confirmer dialog box.

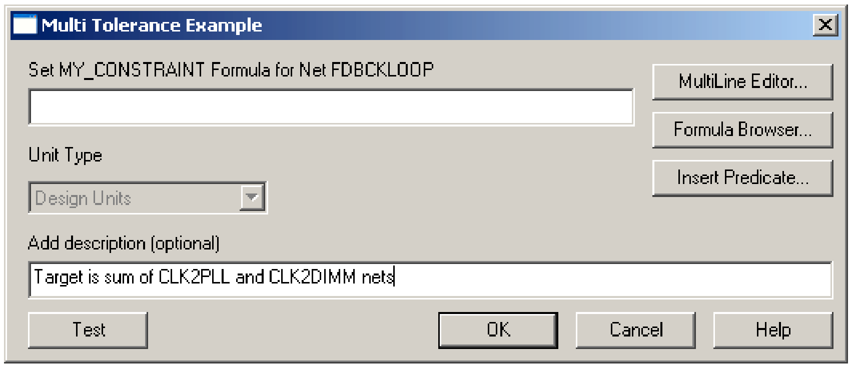


INFO: Formulas ONLY recalculate within Constraint Manager. The last value calculated in Constraint Manager will be stored/used by the application. The formula calculation must be triggered explicitly using the Objects->Calculate menu item or the Calculate item on the right-click/context menu.

   It is important to note that formulas must be calculated. You can calculate a single formula or calculate all formulas (*Edit – Calculate All*).

2. Check the *Don't show again* box and click *OK*.

Figure 1-6 Formula Dialog Box



Target is sum of CLK2PLL and CLK2DIMM nets

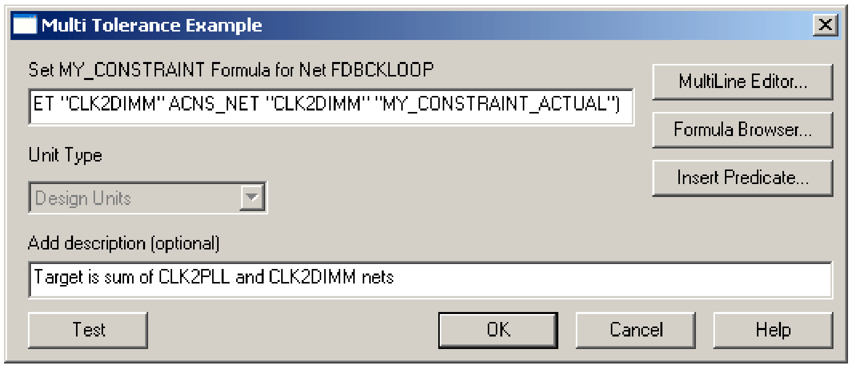
3. Type a description in the *Add description* field. Leave the dialog box open (Figure 2).

   Later on (see Figure 3), you will see that descriptions are a good way to identify particular formulas in a design for reuse.

   At this point, you can generate the formula. You can type in the top fill-in field (for example, 2+3) or use the (default) Cell Selection mode.

4. Select the cell in the main Constraint Manager canvas corresponding to the *Actual* value for the *CLK2DIMM* net (should contain the value 6399.99).
   This adds the SKILL equivalent of that cell in the Formula dialog box.

Figure 1-7 Formula Dialog Box with SKILL Code



```
ET "CLK2DIMM" ACNS_NET "CLK2DIMM" "MY_CONSTRAINT_ACTUAL")
```

Target is sum of CLK2PLL and CLK2DIMM nets

You could have entered this text manually, although it is unlikely. Even though there is only a single cell value in the formula, a dependent constraint value has already been created. If the length of the *CLK2DIMM* net changes, the value of the constraint for the *FDBCKLOOP* net changes as well.

5. Add a "+" to the end of the text in the formula field and then select the cell corresponding to the *Actual* value for the CLK2PLL net (should contain 3177.03). The formula field now contains the string:

```
(acGetValue_poa ACNS_NET "CLK2DIMM" ACNS_NET "CLK2DIMM"
"MY_CONSTRAINT_ACTUAL")+(acGetValue_poa ACNS_NET
"CLK2PLL" ACNS_NET "CLK2PLL" "MY_CONSTRAINT_ACTUAL")
```

Resize the dialog box to view the text added in the formula line.

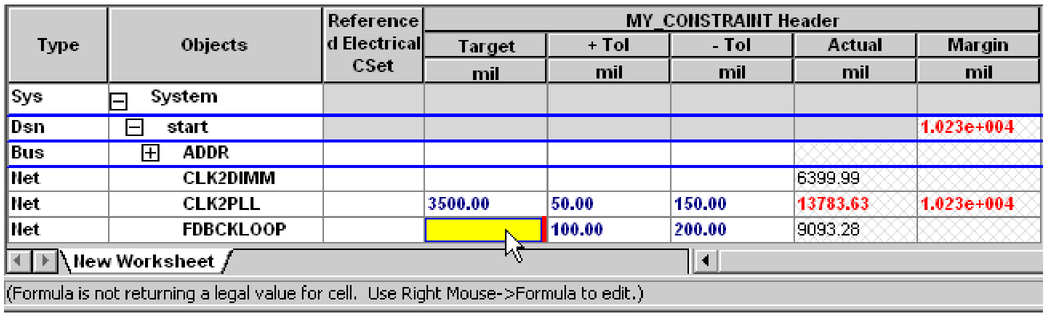
In the formula string, `acGetValue_poa` is an example of a pre-defined Predicate and the rest of the text within each set of parentheses are parameters identifying the value for the cell that you selected.

`acGetValue` gets the value of a specific cell. There are several variants (each with a different suffix) that require different information, in this case, the **P**arent, **O**bject, and **A**ttribute.

Aside from typing a simple equation (for example, 3*4), Cell Selection is the easiest way to construct formulas.

6. Click *OK* in the Formula dialog box.

Figure 1-8 Cell with Formula



Type	Objects	Referenced Electrical CSet	MY_CONSTRAINT Header				
			Target	+ Tol	- Tol	Actual	Margin
			mil	mil	mil	mil	mil
Sys	⊟ System						
Dsn	⊟ start						1.023e+004
Bus	⊞ ADDR						
Net	CLK2DIMM					6399.99	
Net	CLK2PLL		3500.00	50.00	150.00	13783.63	1.023e+004
Net	FDBCKLOOP		🟨	100.00	200.00	9093.28	

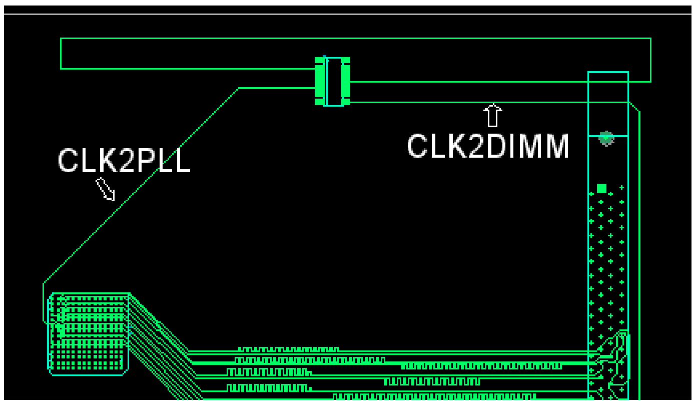
New Worksheet

(Formula is not returning a legal value for cell. Use Right Mouse->Formula to edit.)

Note: The cell now has a red bar along the right edge indicating that the cell contains a formula. Also the value has the appropriate sum (9577.02 = 6399.99 + 3177.03).

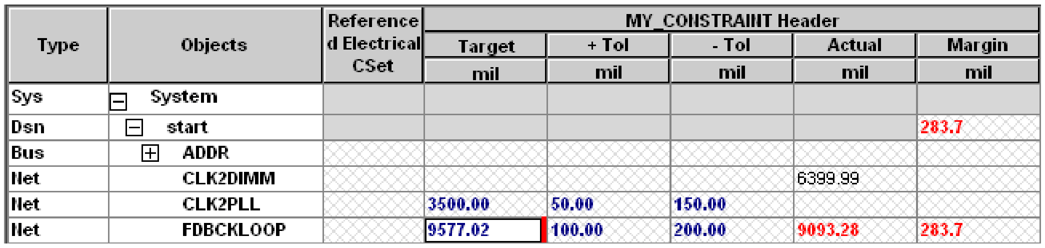
7. In PCB Editor, use either the `slide` or `delay tune` commands to add length to either the *CLK2DIMM* net or the *CLK2PLL* net.

Figure 1-9 Adding Length to a Net



Notice that the values in Constraint Manager corresponding to the *CLDK2PLL* net that you lengthened are not automatically updated, but the *Actual* and *Margin* values are cleared. Also notice (Figure 1-10) that the cell with the formula remains unchanged.

Figure 1-10 Actual and Margin Values Cleared



Type	Objects	Referenced Electrical CSet	MY_CONSTRAINT Header				
			Target	+ Tol	- Tol	Actual	Margin
			mil	mil	mil	mil	mil
Sys	⊟ System						
Dsn	⊟ start						283.7
Bus	⊞ ADDR						
Net	CLK2DIMM					6399.99	
Net	CLK2PLL		3500.00	50.00	150.00		
Net	FDBCKLOOP		9577.02	100.00	200.00	9093.28	283.7

8. Re-run *Analyze* to generate the values in Constraint Manager for the two CLK* nets.

You may need to right-click and choose *Done* to exit the `slide` or `delay tune` commands in PCB Editor.

9. Right-click on the cell with the formula and choose *Calculate* to get the updated value in the cell.

User-defined constraints must be analyzed and the results will be automatically cleared when an object changes. Formulas also must be calculated.

10. In PCB Editor, choose *Undo* to restore the lengthened net to its original state.

11. Re-open the Formula dialog box for the same cell.

Right-click and choose from the menu or left-click in any cell that has a formula.

12. Clear the top fill-in field that has the formula.

The text should be highlighted when the dialog box opens. Either press the `Delete` key or right-click in that field, and choose *Cut.*

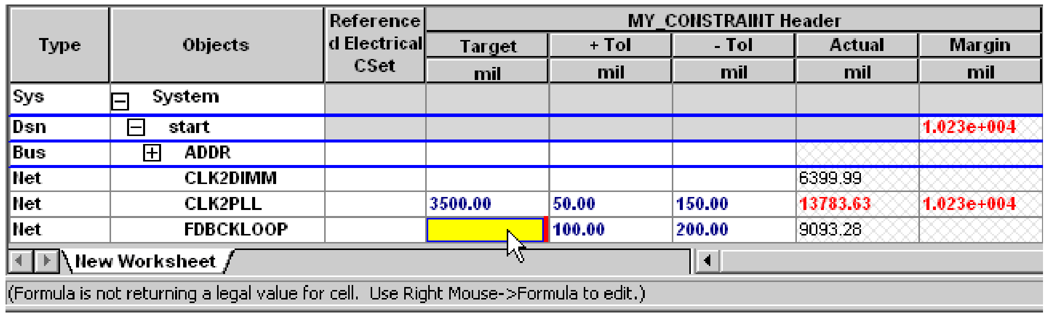
13. Add some white space with the space bar.

The purpose is to create an error with a formula with some blank text.

14. Click *OK* in the formula dialog box.

A yellow cell indicates an error with a formula. Notice the message in the *Status Line*.

Figure 1-11 Error in the Formula



Type	Objects	Referenced Electrical CSet	MY_CONSTRAINT Header				
			Target	+ Tol	- Tol	Actual	Margin
			mil	mil	mil	mil	mil
Sys	⊟ System						
Dsn	⊟ start						1.023e+004
Bus	⊞ ADDR						
Net	CLK2DIMM					6399.99	
Net	CLK2PLL		3500.00	50.00	150.00	13783.63	1.023e+004
Net	FDBCKLOOP		🟨	100.00	200.00	9093.28	

New Worksheet

(Formula is not returning a legal value for cell. Use Right Mouse->Formula to edit.)

Creating Formulas with Pre-defined Predicates

In this example, *FDBCKLOOP* represents a feedback loop for a PLL, which is used to adjust the timing for the clock signal sent to the DIMM. This feedback loop is based on the length of *CLK2PLL* (call this "A") plus the length of *CLK2DIMM* (call this "B") less the length of net *ADDR<10>* (this is "C" and is the target of the Match Group for the ADDR bus). In other words, the equation for the length of the feedback loop is:

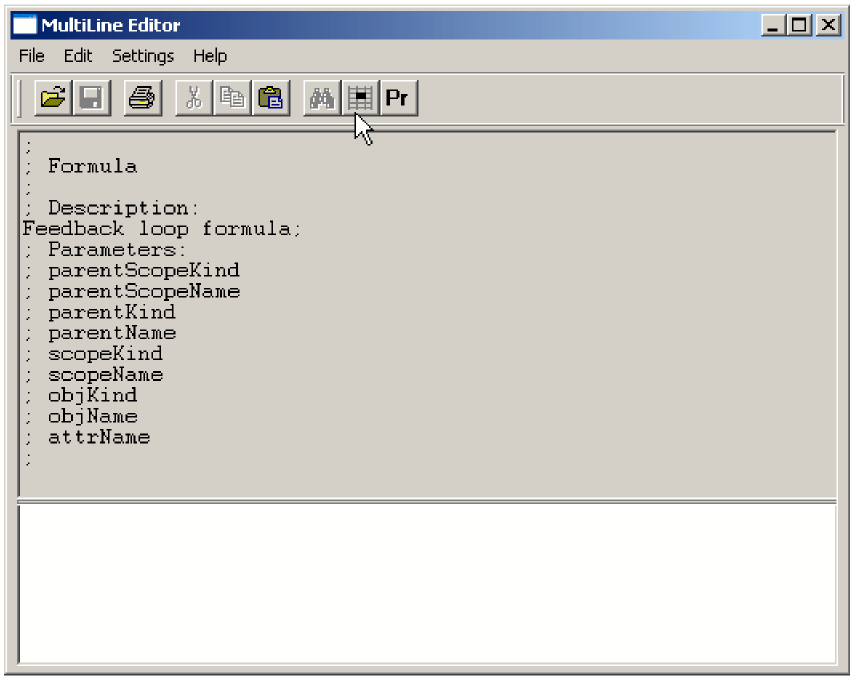
```
(A+B)-C
```

Continuing with the results of the last example:

1. In Constraint Manager, right-click in the cell corresponding to the *Target* constraint value for *FDBCKLOOP* and choose *Clear*.
2. Right-click again in the same cell, and choose *Formula*.
3. In the *Add description* field, enter `Feedback loop formula.`
4. Click the *MultiLine Editor* button.

   You can create simple formulas in the single fill-in line in the main (previous) dialog box, but the MultiLine Editor provides a more comprehensive editing environment.

Figure 1-12 MultiLine Editor



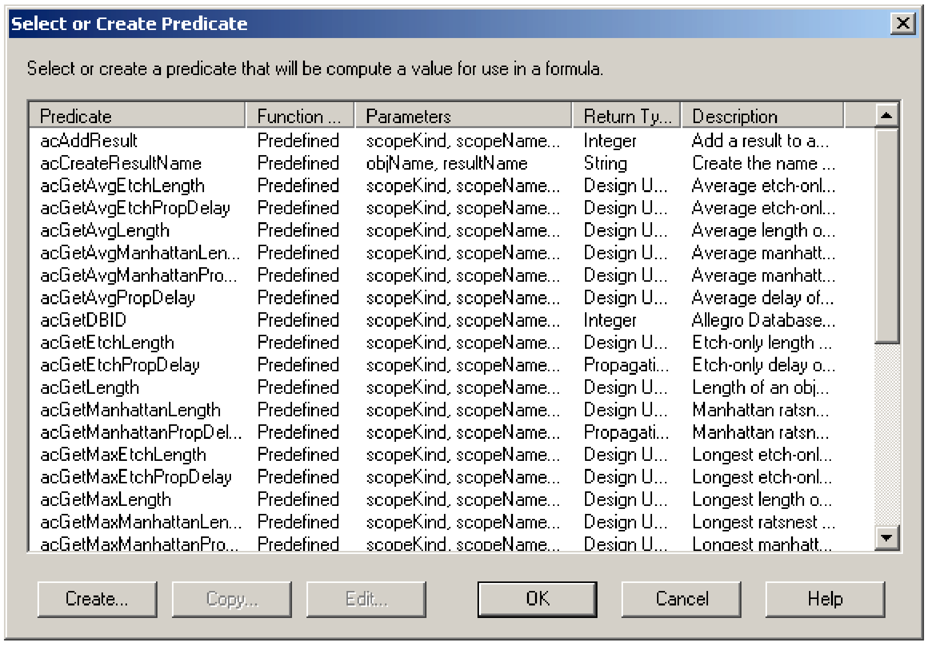
```
;
; Formula
;
; Description:
Feedback loop formula;
; Parameters:
; parentScopeKind
; parentScopeName
; parentKind
; parentName
; scopeKind
; scopeName
; objKind
; objName
; attrName
;
```

The gray area at the top (expanded here for visibility) is the read-only section of the formula and lists the parameters associated with all formulas and any description that was entered in the main formula dialog box.

The cursor in Figure 4 is pointing to the *Insert Cell Value* icon and closes this dialog box until you select a cell in Constraint Manager; then it reappears. This allows you to create formulas, as in the previous example. You can also toggle between the single and multiline view with options under the *File* menu (the single line view is the main formula dialog box previously used).

5. Click the *Pr* (*Insert Predicate*) icon at the end of the toolbar. This icon opens the Select or Create Predicate dialog box.

Figure 1-13 Select or Create Predicate Dialog Box



Select or create a predicate that will be compute a value for use in a formula.

Predicate	Function ...	Parameters	Return Ty...	Description
acAddResult	Predefined	scopeKind, scopeName...	Integer	Add a result to a...
acCreateResultName	Predefined	objName, resultName	String	Create the name ...
acGetAvgEtchLength	Predefined	scopeKind, scopeName...	Design U...	Average etch-onl...
acGetAvgEtchPropDelay	Predefined	scopeKind, scopeName...	Design U...	Average etch-onl...
acGetAvgLength	Predefined	scopeKind, scopeName...	Design U...	Average length o...
acGetAvgManhattanLen...	Predefined	scopeKind, scopeName...	Design U...	Average manhatt...
acGetAvgManhattanPro...	Predefined	scopeKind, scopeName...	Design U...	Average manhatt...
acGetAvgPropDelay	Predefined	scopeKind, scopeName...	Design U...	Average delay of...
acGetDBID	Predefined	scopeKind, scopeName...	Integer	Allegro Database...
acGetEtchLength	Predefined	scopeKind, scopeName...	Design U...	Etch-only length ...
acGetEtchPropDelay	Predefined	scopeKind, scopeName...	Propagati...	Etch-only delay o...
acGetLength	Predefined	scopeKind, scopeName...	Design U...	Length of an obj...
acGetManhattanLength	Predefined	scopeKind, scopeName...	Design U...	Manhattan ratsn...
acGetManhattanPropDel...	Predefined	scopeKind, scopeName...	Propagati...	Manhattan ratsn...
acGetMaxEtchLength	Predefined	scopeKind, scopeName...	Design U...	Longest etch-onl...
acGetMaxEtchPropDelay	Predefined	scopeKind, scopeName...	Design U...	Longest etch-onl...
acGetMaxLength	Predefined	scopeKind, scopeName...	Design U...	Longest length o...
acGetMaxManhattanLen...	Predefined	scopeKind, scopeName...	Design U...	Longest ratsnest ...
acGetMaxManhattanPro...	Predefined	scopeKind, scopeName...	Design U...	Longest manhatt...

Your list of Predicates may be different from the list shown here.

Predicates are the building blocks of formulas and user-defined Measurements. The complete list of Predicates is presented.

Predicates are similar to Measurements in that they return a particular value from the design. They differ in several ways:

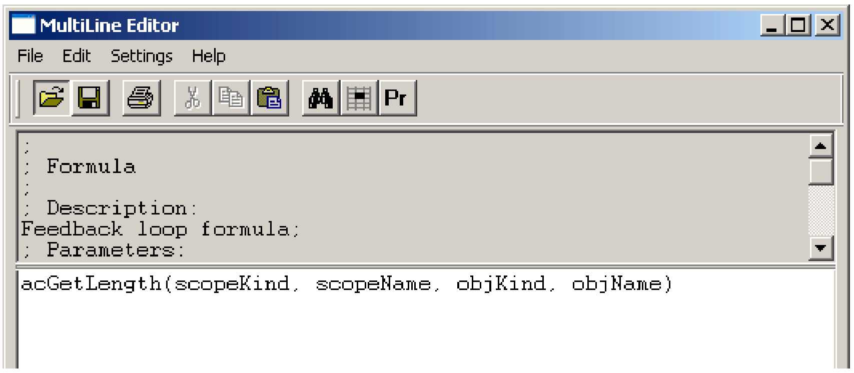
❑ Predicates only return a single value; Measurements can return many values.

❑ Measurements are used for *Actuals*; Predicates are used for Formulas, Measurements, or other Predicates.

❑ The list of Predicates presented is complete; the list for Measurements is based on the *Data Type* of the *Actual.*

6. Choose the `acGetLength` predicate in the list and then click *OK*. See Figure 6.

Figure 1-14 MultiLine Editor



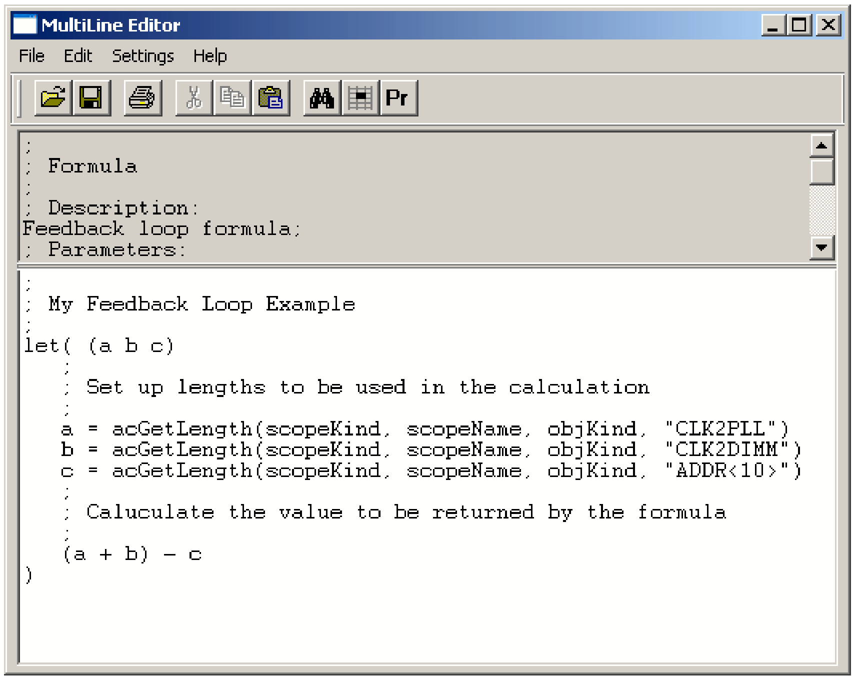
```
;
; Formula
;
; Description:
Feedback loop formula;
; Parameters:
```

```
acGetLength(scopeKind, scopeName, objKind, objName)
```

The constraint is based on length (the length of A + the length …) so a predicate for `GetLength` makes sense.

7. Use *Copy* and *Paste* or re-insert the same predicate and then edit the resulting *objName* parameters for each to create the formula shown below.

Figure 1-15 Feedback Loop Example



```
;
; Formula
;
; Description:
Feedback loop formula;
; Parameters:
```

```
;
; My Feedback Loop Example
;
let( (a b c)
   ;
   ; Set up lengths to be used in the calculation
   ;
   a = acGetLength(scopeKind, scopeName, objKind, "CLK2PLL")
   b = acGetLength(scopeKind, scopeName, objKind, "CLK2DIMM")
   c = acGetLength(scopeKind, scopeName, objKind, "ADDR<10>")
   ;
   ; Caluculate the value to be returned by the formula
   ;
   (a + b) - c
)
```

If you prefer, load the `fb_loop_example.txt` file found in the *`<install_dir>/doc/cmadvcnstut/examples`* directory with the *File – Load from file* menu command.

For this formula, the `acGetLength` predicate was used. Notice that it has four parameters and for this case, only the *objName* parameter was changed to the net name of interest. Depending on the specific application, you may need to edit additional parameters, most likely *scope* parameters for system configurations and *parent* parameters for *Match Groups*.

Also notice that variables were used to clean up the display. This formula works just as well when written as:

```
(acGetLength(scopeKind, scopeName, objKind, "CLK2PLL") +
acGetLength(scopeKind, scopeName, objKind, "CLK2DIMM")) -
acGetLength(scopeKind, scopeName, objKind, "ADDR<10>")
```

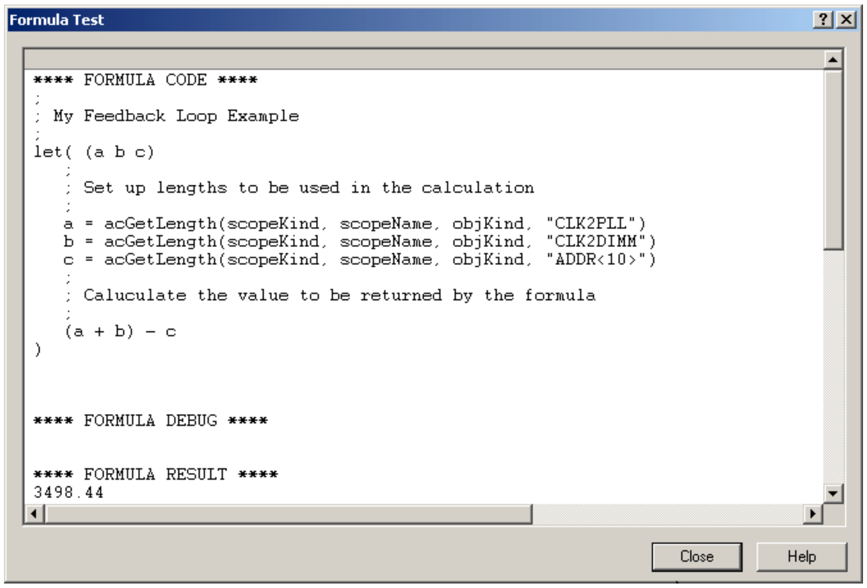
Tip

The entire function is wrapped in a `let` statement. Use the `let` statement any time that variables are used. These variables are being protected from becoming global and the potential side effects from having them accidentally accessed elsewhere.

8. From the menu, choose *File – Test*.

Choosing this menu item is the same as clicking the *Test* button in the lower-left corner of the main (single line) Formula dialog box (Figure 2).

Figure 1-16 Formula Test Dialog Box



```
**** FORMULA CODE ****
;
; My Feedback Loop Example
;
let( (a b c)
   ;
   ; Set up lengths to be used in the calculation
   ;
   a = acGetLength(scopeKind, scopeName, objKind, "CLK2PLL")
   b = acGetLength(scopeKind, scopeName, objKind, "CLK2DIMM")
   c = acGetLength(scopeKind, scopeName, objKind, "ADDR<10>")
   ;
   ; Caluculate the value to be returned by the formula
   ;
   (a + b) - c
)


**** FORMULA DEBUG ****


**** FORMULA RESULT ****
3498.44
```

In the Formula Test dialog box, there are four main sections of the output:

- A listing of the Formula itself.
- A debug message section, which is currently blank.
- The formula result. In this case, verify that 3498.44 is the expected result.
- SKILL LINT output to further check the SKILL code for possible errors.

Tip

When you create any custom code for a formula, always test the code, no matter how simple the code seems.

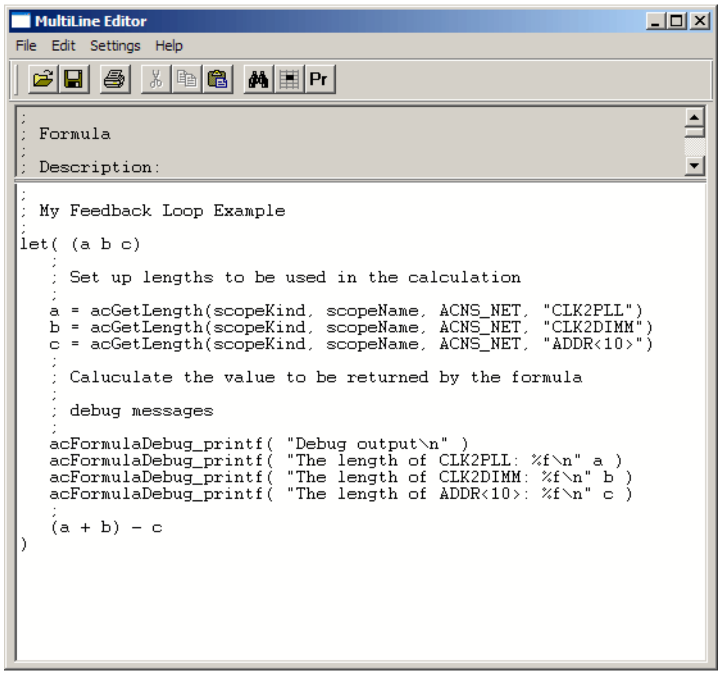
9. Close the Formula Test dialog box.

As mentioned above, verify that the result (3498.44) is as expected by checking the lengths and doing the math. However, there is another way to do this earlier in the process.

10. In the MultiLine Editor, use *File – Load from file*, and load the `debug_example.txt` file.

Notice the added `acFormulaDebug_printf` predicates.

Figure 1-17 Loading the debug_example File



```
;
; Formula
;
; Description:
```

```
;
; My Feedback Loop Example
;
let( (a b c)
    ;
    ; Set up lengths to be used in the calculation
    ;
    a = acGetLength(scopeKind, scopeName, ACNS_NET, "CLK2PLL")
    b = acGetLength(scopeKind, scopeName, ACNS_NET, "CLK2DIMM")
    c = acGetLength(scopeKind, scopeName, ACNS_NET, "ADDR<10>")
    ;
    ; Caluculate the value to be returned by the formula
    ;
    ; debug messages
    ;
    acFormulaDebug_printf( "Debug output\n" )
    acFormulaDebug_printf( "The length of CLK2PLL: %f\n" a )
    acFormulaDebug_printf( "The length of CLK2DIMM: %f\n" b )
    acFormulaDebug_printf( "The length of ADDR<10>: %f\n" c )
    ;
    (a + b) - c
)
```

11. Choose *File – Test* from the menu bar.

Figure 1-18 FORMULA DEBUG Section in the Formula Test Dialog Box

```
**** FORMULA DEBUG ****
Debug output
The length of CLK2PLL: 3177.030000
The length of CLK2DIMM: 6399.990000
The length of ADDR<10>: 6078.580000


**** FORMULA RESULT ****
3498.44
```

The added print commands output text in the FORMULA DEBUG section also aid in verifying the RESULT.

12. Close the Formula Test dialog box.

13. Close the MultiLine Editor window and click *Save* if prompted.

    The value of the formula in Constraint Manager should be 3498.44.

Using Formulas with Pre-defined Constraints

Until now, the focus has been on user-defined constraints. While this opens a new set of possibilities for constraining objects and validating the results in Constraint Manager, many PCB designers prefer to work in the PCB Editor canvas with real-time feedback.

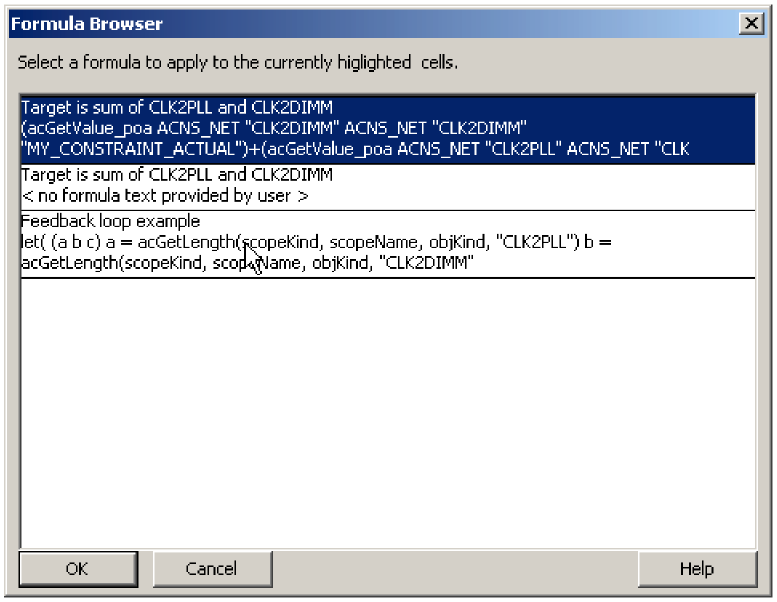
1. Open the *Net – Routing – Min/Max Propagation Delays* worksheet in the *Electrical Domain*.

   You will leverage the formula that you just created. While the formula returns a single value based on the requirements (the length of (a+b) - c), it does not make sense as both the minimum and maximum constraint value. In other words, there needs to be some tolerance which was handled in the user-defined constraint previously used. For this example, you will add the tolerance directly to the formula.

2. Right-click on the *Min Prop Delay* cell for *FDBCKLOOP*, and choose *Formula.*

3. In the Formula dialog box, click the *Formula Browser* button.

Figure 1-19 Formula Browser Dialog Box



Select a formula to apply to the currently higlighted cells.

Target is sum of CLK2PLL and CLK2DIMM
(acGetValue_poa ACNS_NET "CLK2DIMM" ACNS_NET "CLK2DIMM" "MY_CONSTRAINT_ACTUAL")+(acGetValue_poa ACNS_NET "CLK2PLL" ACNS_NET "CLK

Target is sum of CLK2PLL and CLK2DIMM
< no formula text provided by user >

Feedback loop example
let((a b c) a = acGetLength(scopeKind, scopeName, objKind, "CLK2PLL") b =
acGetLength(scopeKind, scopeName, objKind, "CLK2DIMM"

Your list may be different.

The Formula Browser is one way to use existing formulas in the design. Notice that as you hover over each formula, a tool tip window appears showing the full formula.

4. Choose one of the Feedback Loop examples and click *OK* in the Formula Browser dialog box (the formula from the previous exercise).

   Notice that the *Unit Type* drop-down list is enabled. This field applies only in cases such as where the constraint supports different units (database units, time, or % manhattan).

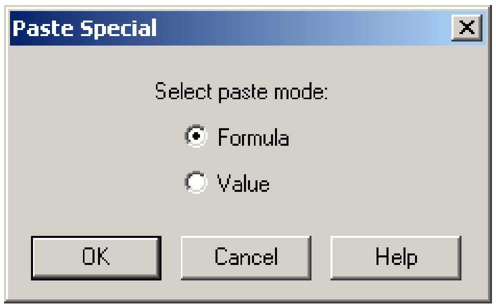
5. Click the *MultiLine Editor* button in the main Formula dialog box, and change the last line of the formula to:

   ```
   ((a + b) - c) - 25
   ```

   The *Min Prop Delay* constraint value is set to 25 mils (design units) less than the target length.

6. Save and close the MultiLine Editor.

7. Right-click on the *Min Prop Delay* cell for *FDBCKLOOP*, and choose *Copy* (this is the cell in which you just entered the formula).



8. Right-click in the *Max Prop Delay* cell for *FDBCKLOOP* and choose *Paste Special.*

   The default action is to paste the formula.

9. Make sure that you select the *Formula* radio button and click *OK* in the Paste Special dialog box.

10. Edit the formula for the *Max Prop Delay* and adjust the last line of the formula to:

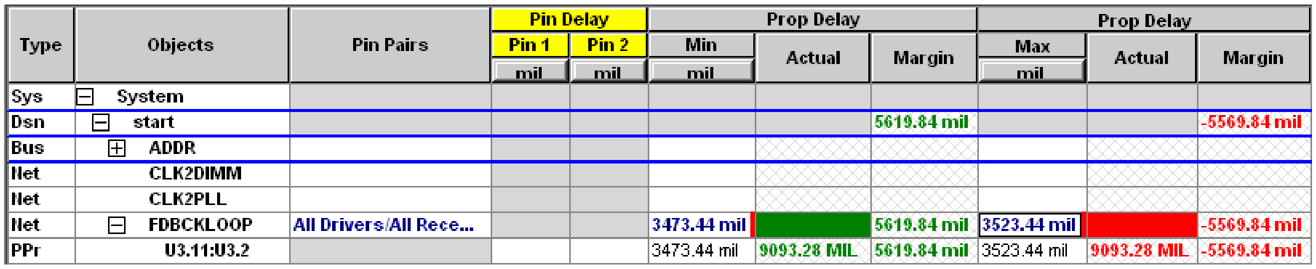
    ```
    ((a + b) - c) + 25
    ```

Notice that the Formula dialog box opens in the MultiLine Editor. It opens in whichever dialog box you last used in creating the formula.

11. Save and close the MultiLine Editor.

Your view should look similar to what is shown in Figure 1-20.

Figure 1-20 View in Constraint Manager

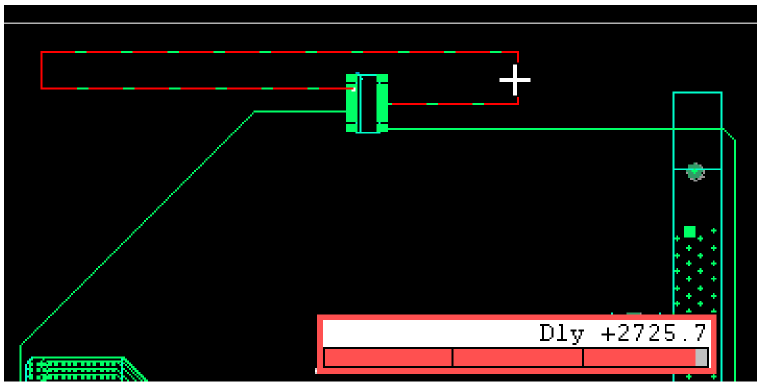


Type	Objects	Pin Pairs	Pin Delay		Prop Delay			Prop Delay		
			Pin 1	Pin 2	Min	Actual	Margin	Max	Actual	Margin
			mil	mil	mil			mil		
Sys	⊟ System									
Dsn	⊟ start						5619.84 mil			-5569.84 mil
Bus	⊞ ADDR									
Net	CLK2DIMM									
Net	CLK2PLL									
Net	⊟ FDBCKLOOP	All Drivers/All Rece...			3473.44 mil	🟩	5619.84 mil	3523.44 mil	🟥	-5569.84 mil
PPr	U3.11:U3.2				3473.44 mil	9093.28 MIL	5619.84 mil	3523.44 mil	9093.28 MIL	-5569.84 mil

Look for the values 3473.44 and 3523.44 in the *Min* and *Max* formula cells, respectively.

12. In PCB Editor, run the `slide` command for the *FDBCKLOOP* net (the top net in the design) and notice that the Heads-up Display can be used.

Figure 1-21 Sliding the FDBCKLOOP Net



13. Cancel the `slide` command (do not commit to any etch changes).

Creating User-defined Measurements to Populate User-defined Actuals

User-defined Measurements and Predicates represent the final pieces of the customization process.

- A user-defined Predicate is a SKILL function with arbitrary parameters that returns a single value for use in a Formula or Measurement.
- A user-defined Measurement is a SKILL function with a pre-determined set of input parameters (the object being measured) that must update result information and return success or failure. Result information includes pushing of the *Actual* values. You can push the *Actual* values on the object being measured, or you can push the values on result objects that you associate with the object being measured.

With the ability to code your own Measurements to associate with *Actuals* either as is or in user-defined Constraints coupled with the ability to make user-defined Predicates as the building blocks for both Measurements and Formulas, you can theoretically create any scenario in Constraint Manager.

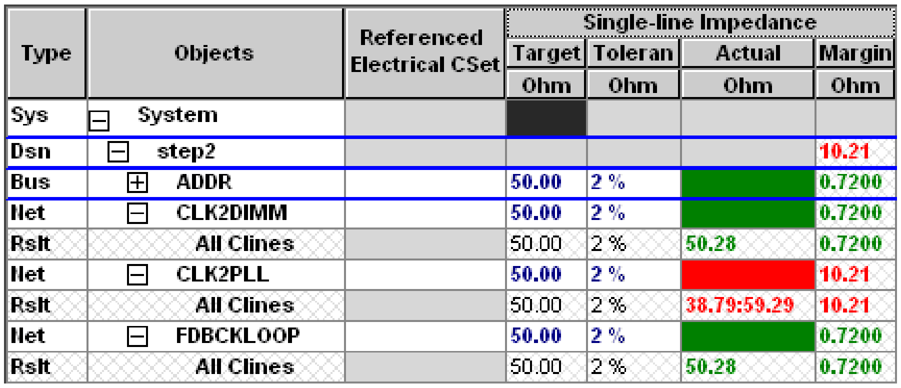
For this next example, examine adding an *Actual* (associated with a user-defined Measurement) to aid in debugging an existing constraint.

1. In PCB Editor, open the `step2.brd` design in the *`<install_dir>/doc/cmadvcnstut/examples`* directory.

   Do not save the current design.

2. Launch Constraint Manager and open the *Net – Routing – Impedance* worksheet. Right-click on the design name*,* and choose *Analyze* to generate results.

Figure 1-22 Impedance Worksheet



Type	Objects	Referenced Electrical CSet	Single-line Impedance			
			Target	Toleran	Actual	Margin
			Ohm	Ohm	Ohm	Ohm
Sys	⊟ System		■			
Dsn	⊟ step2					10.21
Bus	⊞ ADDR		50.00	2 %	🟩	0.7200
Net	⊟ CLK2DIMM		50.00	2 %	🟩	0.7200
Rslt	All Clines		50.00	2 %	50.28	0.7200
Net	⊟ CLK2PLL		50.00	2 %	🟥	10.21
Rslt	All Clines		50.00	2 %	38.79:59.29	10.21
Net	⊟ FDBCKLOOP		50.00	2 %	🟩	0.7200
Rslt	All Clines		50.00	2 %	50.28	0.7200

The *CLK2PLL* net has at least two errors since the *Actual* value is reporting a range of results from 38.79 to 59.29 Ohms. Adding a user-defined measurement to this worksheet can aid in debugging the errors for this net.

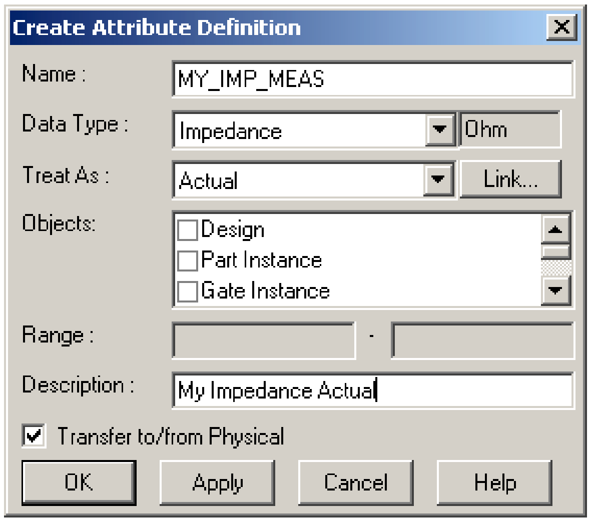
3. Right-click in the *Worksheet Selector,* and choose *Customize Worksheet.*

4. Right-click on the *Net – Impedance* worksheet (the colored icon indicates it is open), and choose *Add Column.*

5. Click the *Create* button in the Add Column dialog box.

6. Fill in the Create Attribute Definition dialog box as shown in Figure 6.

Figure 1-23 Create Attribute Definition Dialog Box



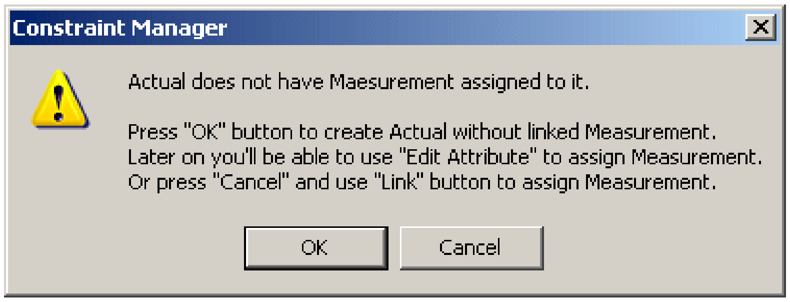
Create Attribute Definition

Name :	MY_IMP_MEAS	
Data Type :	Impedance	Ohm
Treat As :	Actual	Link...
Objects:	☐ Design ☐ Part Instance ☐ Gate Instance	
Range :	-	
Description :	My Impedance Actual	

☑ Transfer to/from Physical

You are creating an *Actual* with the *Data Type* set as *Impedance.*

7. Click *OK*.



Constraint Manager

Actual does not have Maesurement assigned to it.

Press "OK" button to create Actual without linked Measurement.
Later on you'll be able to use "Edit Attribute" to assign Measurement.
Or press "Cancel" and use "Link" button to assign Measurement.

Previously, as when you were creating a constraint, clicking *OK* launched a dialog box to select the desired associated measurement. In this confirmer dialog box, when creating an *Actual*, the correct action is to click *Cancel* and then click the *Link* button (which is now active).

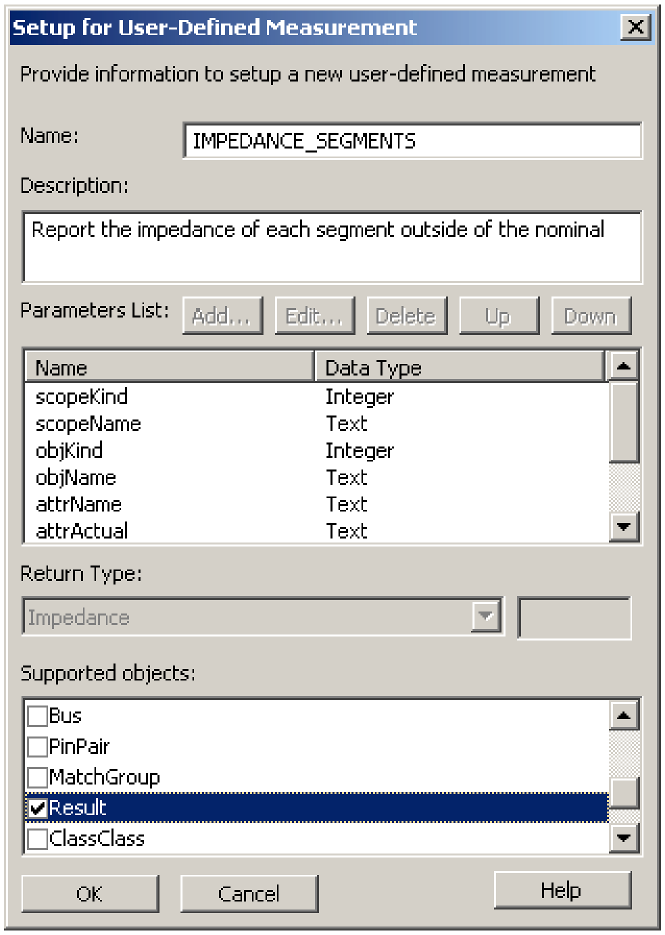
8. Click *Cancel* in the confirmer dialog box.

9. Click the *Link* button next to the *Treat As* field.

   As seen earlier when creating a constraint and choosing an associated Measurement for the *Actual*, the list of Measurements provided is driven by the *Data Type*. At this point, there are no pre-defined Measurements with this *Data Type* (*Impedance*).

10. Click *Yes* in the confirmer dialog box.

Figure 1-24 Setup for User-Defined Measurement Dialog Box


Setup for User-Defined Measurement
Provide information to setup a new user-defined measurement
Parameters List:
Add...
Edit...
Delete
Up
Down
Return Type:
Supported objects:
OK
Cancel
Help

Name: IMPEDANCE_SEGMENTS

Description: Report the impedance of each segment outside of the nominal

Name	Data Type
scopeKind	Integer
scopeName	Text
objKind	Integer
objName	Text
attrName	Text
attrActual	Text

Return Type: Impedance

Supported objects:
- [ ] Bus
- [ ] PinPair
- [ ] MatchGroup
- [✓] Result
- [ ] ClassClass

11. Enter a *Name* and *Description*.

A similar dialog box is used for both User-Defined Measurement and User-Defined Predicates. In the case of User-Defined Measurement, the *Parameters List* is fixed so that portion of the dialog box has the buttons grayed out.

12. In the *Supported objects* section, make sure that only the following are selected:

❑ Net

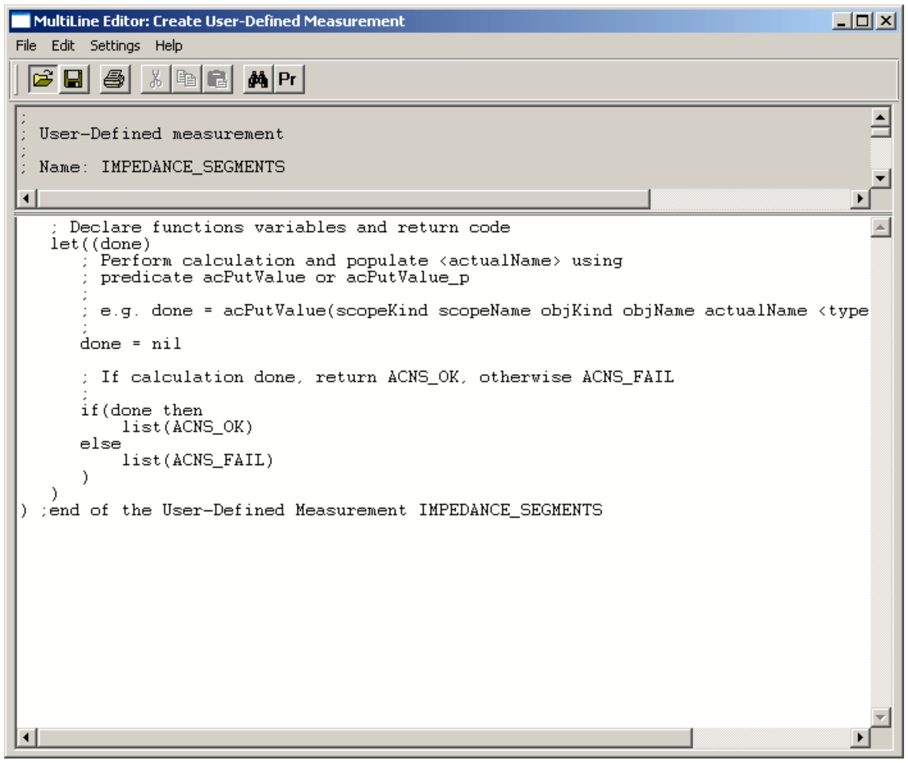
❑ Xnet

❑ Result

For Measurements, the *Supported objects* represent which kind of objects the measurement will analyze. Up to this point, you have ignored setting the *Objects* but it is a good habit to make sure that anything created in Constraint Manager has the correct *Objects* setting. This ensures that the appropriate (and desired) cells are enabled in Constraint Manager.

13. Click *OK* to display the MultiLine Editor specific to Measurement.

Figure 1-25 MultiLine Editor (Measurement)



```
;
; User-Defined measurement
;
; Name: IMPEDANCE_SEGMENTS
```

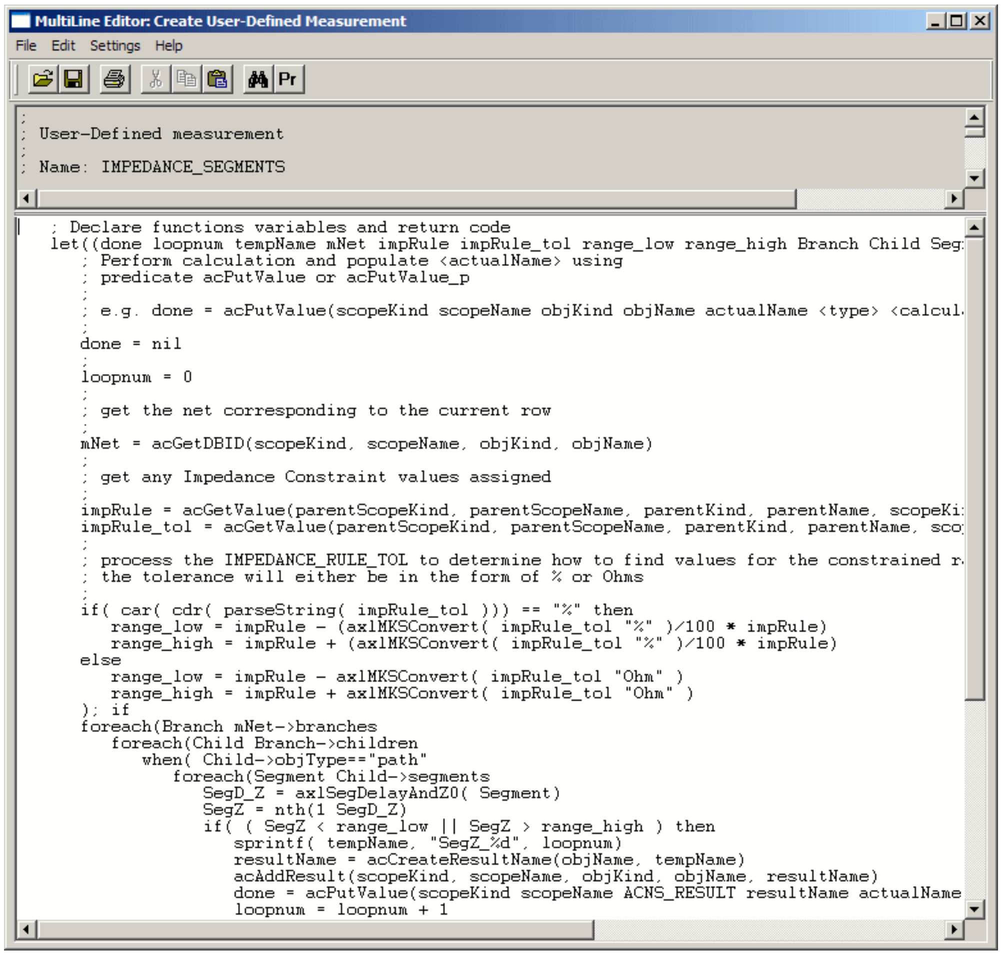
```
   ; Declare functions variables and return code
   let((done)
      ; Perform calculation and populate <actualName> using
      ; predicate acPutValue or acPutValue_p
      ;
      ; e.g. done = acPutValue(scopeKind scopeName objKind objName actualName <type
      ;
      done = nil

      ; If calculation done, return ACNS_OK, otherwise ACNS_FAIL
      ;
      if(done then
          list(ACNS_OK)
      else
          list(ACNS_FAIL)
      )
   )
) ;end of the User-Defined Measurement IMPEDANCE_SEGMENTS
```

14. Expand or scroll through the gray portion (read-only) at the top of the MultiLine Editor.

At the end of the gray section is a procedure call. Also notice that there is a template provided in the editing section of the dialog box. The template defines a variable *done* to indicate successful completion. In the comments section is a suggestion to assign done the value of the `acPutValue` predicate.Choose *File – Load from file* or click the associated icon and open the `impedance_meas.txt` file (located in the

<install_dir>/doc/cmadvcnstut/examples directory).

Figure 1-26 impedance_meas.txt File



```
;
; User-Defined measurement
;
; Name: IMPEDANCE_SEGMENTS
```

```
  ; Declare functions variables and return code
  let((done loopnum tempName mNet impRule impRule_tol range_low range_high Branch Child Seg
     ; Perform calculation and populate <actualName> using
     ; predicate acPutValue or acPutValue_p
     ;
     ; e.g. done = acPutValue(scopeKind scopeName objKind objName actualName <type> <calcul
     ;
     done = nil
     ;
     loopnum = 0
     ;
     ; get the net corresponding to the current row
     ;
     mNet = acGetDBID(scopeKind, scopeName, objKind, objName)
     ;
     ; get any Impedance Constraint values assigned
     ;
     impRule = acGetValue(parentScopeKind, parentScopeName, parentKind, parentName, scopeKi
     impRule_tol = acGetValue(parentScopeKind, parentScopeName, parentKind, parentName, sco
     ;
     ; process the IMPEDANCE_RULE_TOL to determine how to find values for the constrained r
     ; the tolerance will either be in the form of % or Ohms
     ;
     if( car( cdr( parseString( impRule_tol ))) == "%" then
        range_low = impRule - (axlMKSConvert( impRule_tol "%" )/100 * impRule)
        range_high = impRule + (axlMKSConvert( impRule_tol "%" )/100 * impRule)
     else
        range_low = impRule - axlMKSConvert( impRule_tol "Ohm" )
        range_high = impRule + axlMKSConvert( impRule_tol "Ohm" )
     ); if
     foreach(Branch mNet->branches
        foreach(Child Branch->children
           when( Child->objType=="path"
              foreach(Segment Child->segments
                 SegD_Z = axlSegDelayAndZ0( Segment)
                 SegZ = nth(1 SegD_Z)
                 if( ( SegZ < range_low || SegZ > range_high ) then
                    sprintf( tempName, "SegZ_%d", loopnum)
                    resultName = acCreateResultName(objName, tempName)
                    acAddResult(scopeKind, scopeName, objKind, objName, resultName)
                    done = acPutValue(scopeKind scopeName ACNS_RESULT resultName actualName
                    loopnum = loopnum + 1
```

In this example, notice:

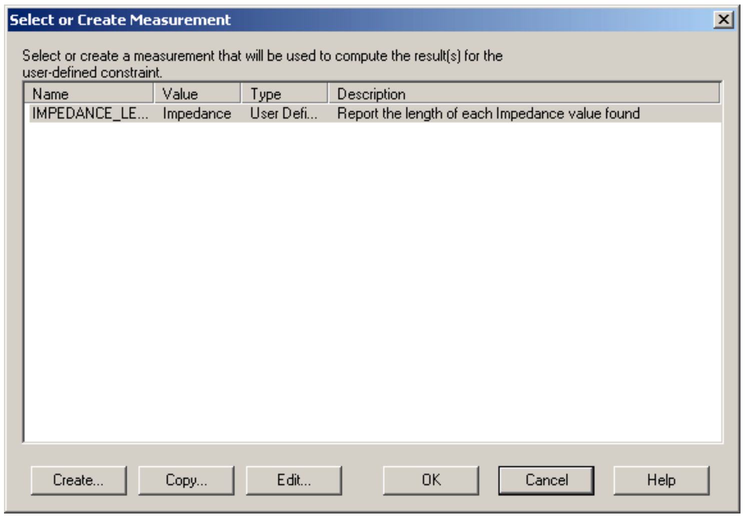
- ❑ The list of variables added to the `let` statement
- ❑ The `acGetDBID` predicate; this gets DBID of an object to pass to AXL functions
- ❑ The use of an AXL function (`axlSegDelayAndZ0`)
- ❑ Conditional statements to filter the results
- ❑ The `acCreateResultName()` predicate used to define the name of a result object

- ❑ The `acAddResult()` predicate used to create the result object for the object being measured
- ❑ The `acPutValue()` predicate used to populate the *Actual* value on the result object

Note: There is currently no Test capability for User-Defined Measurement or Predicates. An error in the coding results in yellow cells in Constraint Manager, but may also result in a SKILL error in the Allegro command line window. These error messages can help pinpoint problems so it is a good idea to enable the SKILL stracktrace in your `allegro.ilinit` file. You can also add standard SKILL print commands to send data to the Allegro CIW. Be careful not to change the logic or have a print as the last command, or the return from a print command is nil.

15. Save and close the MultiLine Editor.

Figure 1-27 Select or Create Measurement Dialog Box



Select or create a measurement that will be used to compute the result(s) for the user-defined constraint.

Name	Value	Type	Description
IMPEDANCE_LE...	Impedance	User Defi...	Report the length of each Impedance value found

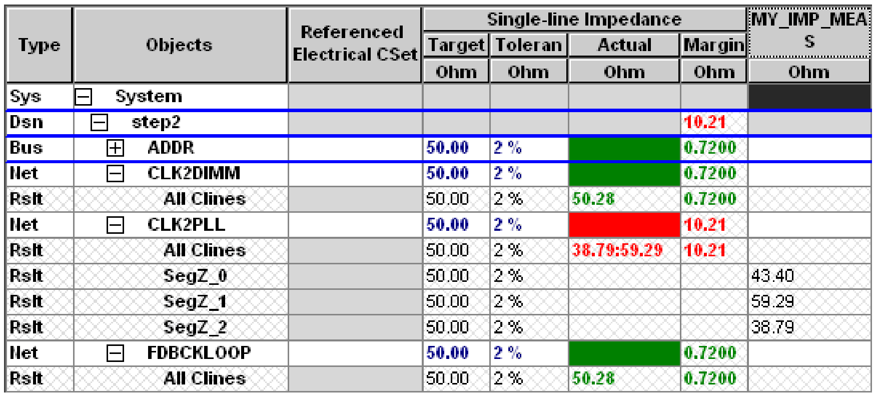
Notice that the *Copy* and *Edit* buttons are now enabled.

16. Click *OK* in the Select or Create Measurement dialog box.

17. Click *OK* in the remaining two dialog boxes and any confirmer dialog box that appears.

18. Right-click on the *MY_IMP_MEAS* column header, and choose *Analyze.*

Figure 1-28 Analysis of step2.brd Design



Type	Objects	Referenced Electrical CSet	Single-line Impedance				MY_IMP_MEAS
			Target	Toleran	Actual	Margin	
			Ohm	Ohm	Ohm	Ohm	Ohm
Sys	⊟ System						⬛
Dsn	⊟ step2					10.21	
Bus	⊞ ADDR		50.00	2 %	🟩	0.7200	
Net	⊟ CLK2DIMM		50.00	2 %	🟩	0.7200	
Rslt	All Clines		50.00	2 %	50.28	0.7200	
Net	⊟ CLK2PLL		50.00	2 %	🟥	10.21	
Rslt	All Clines		50.00	2 %	38.79:59.29	10.21	
Rslt	SegZ_0		50.00	2 %			43.40
Rslt	SegZ_1		50.00	2 %			59.29
Rslt	SegZ_2		50.00	2 %			38.79
Net	⊟ FDBCKLOOP		50.00	2 %	🟩	0.7200	
Rslt	All Clines		50.00	2 %	50.28	0.7200	

Even though you analyzed the entire design, you got results only for the *CLK2PLL* net, which is expected since it is the only net with errors. Notice that in fact there are three segments that are outside the *Target / Tolerance* range.

Although what you created and added is only an *Actual*, it could also be used in a user-defined constraint.