

S-Edit v13

事例ガイド

Tanner EDA Division
Tanner Research, Inc.
825 South Myrtle Avenue
Monrovia, CA 91016-3424
Tel: (626) 471-9700

タナーリサーチジャパン株式会社
〒102-0083
東京都千代田区麹町3-5-2
BUREX 麹町6階
Tel: 03-3239-2853 Fax: 03-3239-2848
Web: www.tanner.jp

目次

1. 交通信号コントローラ— Lights	2
2. バスとアレイ	3
2.1. シンプルバス	3
2.2. バスの分岐.....	4
2.3. ポートバンドル	5
2.4. 1次元のアレイ（並列アレイ）	6
2.5. 2次元のアレイ	7
3. Spice エクスポート	9
3.2. パラメータを下の階層のセルに引き渡す	11
3.3. サブサーキット	13
3.4. エクスポート制御用プロパティ	16
4. グローバルネット、グローバルシンボルと、ネットキャップ	18
4.1. シンプル グローバルネット	18
4.2. 別々の電圧源	20
4.3. 別々の電圧源の名前の変更	22
4.4. 別々の電圧源の名前の変更（他の方法）	24
5. マルチビュー	26
5.1. 4端子と3端子 MOSFET シンボル	26
5.2. IEEE と IEC シンボルビューを持つ NMOS セル	27
5.3. ピンが異なる方法で整理されたシンボルが2つ持つアダプター	28
6. TCL/TK スクリプト	29
6.1. テキストの編集	29
6.2. パラメータを入力する TK ダイアログを使ってテキストのサイズの変更	30

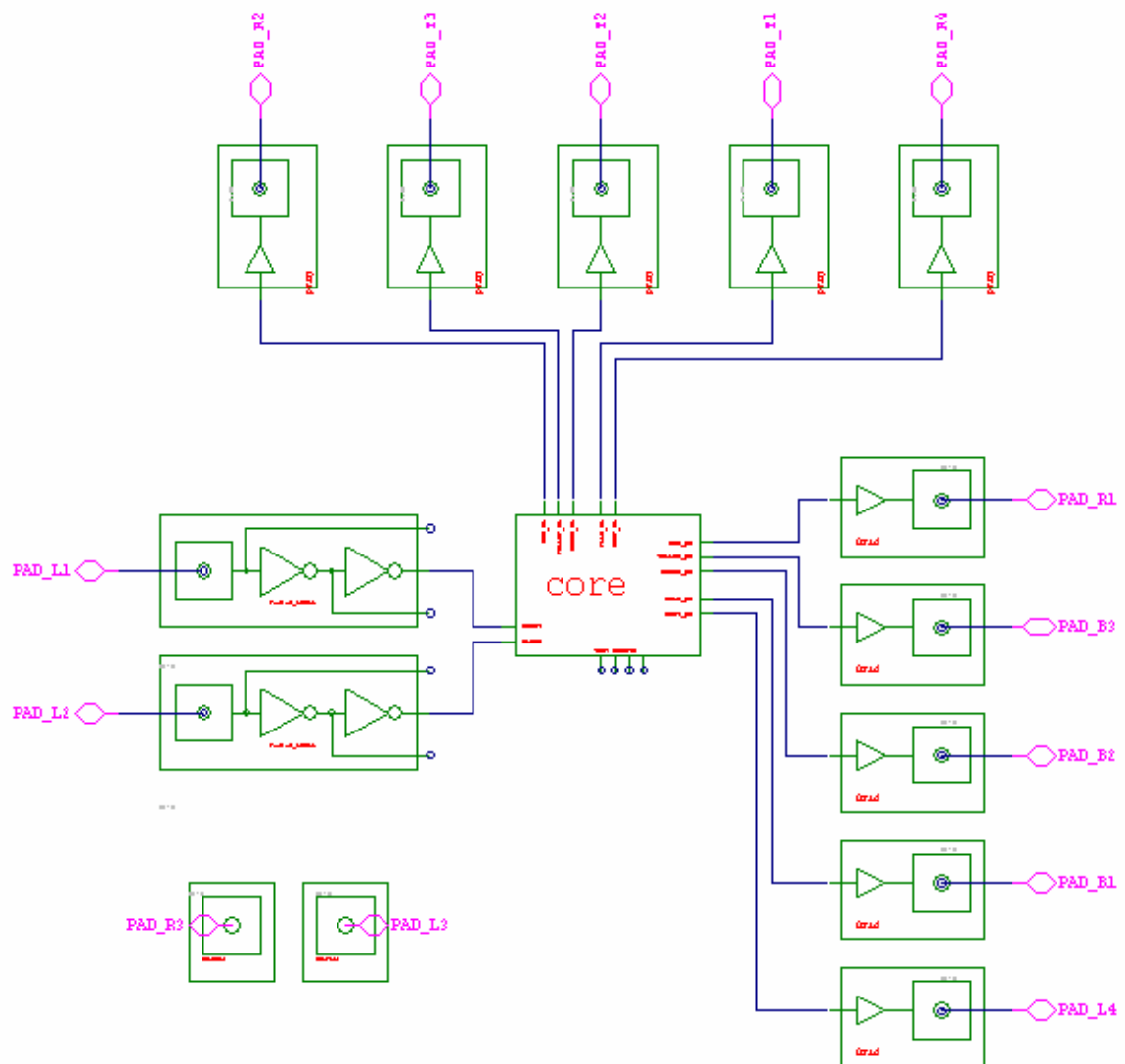
1. 信号機コントローラー Lights

デザイン Lights

セル Lights

この事例では、ライブラリによって構成されたデザインを示します。ここで Lights はメインデザイン、PADLIB と SCMOSLIB は Lights のライブラリです。そして、SPICELIB は PADLIB と SCMOSLIBに両方とも利用されるライブラリです。

回路図



Lights セルの回路図

2. バスとアレイ

2.1. シンプルバス

デザイン BusesAndArrays

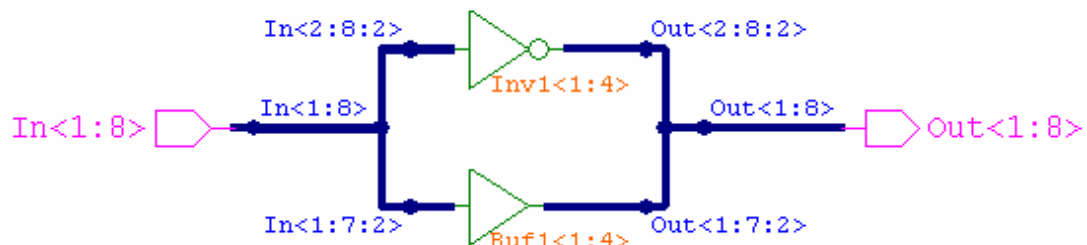
セル Example1

この事例はバスとアレイの基本的な記述と使用方法を示します。バスやアレイの記述で、最初の値は、バスのコンポーネントが始まる番号を、2番目の値はコンポーネントの最後の番号を表します。3番目の番号はステップを表します。オミットされている場合ステップが1となります。

ここで、8ビット広さのバス(In<1:8>)は、偶数ビットは一つのバスに、偶数ビットはもう一つのバスに含まれるように2つに分岐されています。この場合、分岐されたバスのステップが2ですので、両方とも4ビットのバスになります。偶数ビットのバスは4×インバータのアレイに、奇数ビットバスは4×バッファのアレイに接続されています。インバータとバッファがそれぞれ1つ入力と1つ出力を持っているので、それらのアレイは4ビット幅の入力と出力となり、接続されるバスの次元と一致します。インバータの出力とバッファは結合され、8ビット広さの出力バス(Out<1:8>)を構成しています。

バスをインスタンス、あるいはインスタンスのアレイに接続する場合、次元が一致しているかどうかとチェックすることはとても重要です。S-Edit メニューの **Tools > Design Checks** を実行すると、ミスマッチしているバスとインスタンスの次元に関して検証されます。

回路図



Spice ネットリスト

```
XInv1<1> In<2> Out<2> Gnd Vdd Inv
XInv1<2> In<4> Out<4> Gnd Vdd Inv
XInv1<3> In<6> Out<6> Gnd Vdd Inv
XInv1<4> In<8> Out<8> Gnd Vdd Inv
XBuf1<1> In<1> Out<1> Gnd Vdd Buf
XBuf1<2> In<3> Out<3> Gnd Vdd Buf
XBuf1<3> In<5> Out<5> Gnd Vdd Buf
XBuf1<4> In<7> Out<7> Gnd Vdd Buf
.end
```

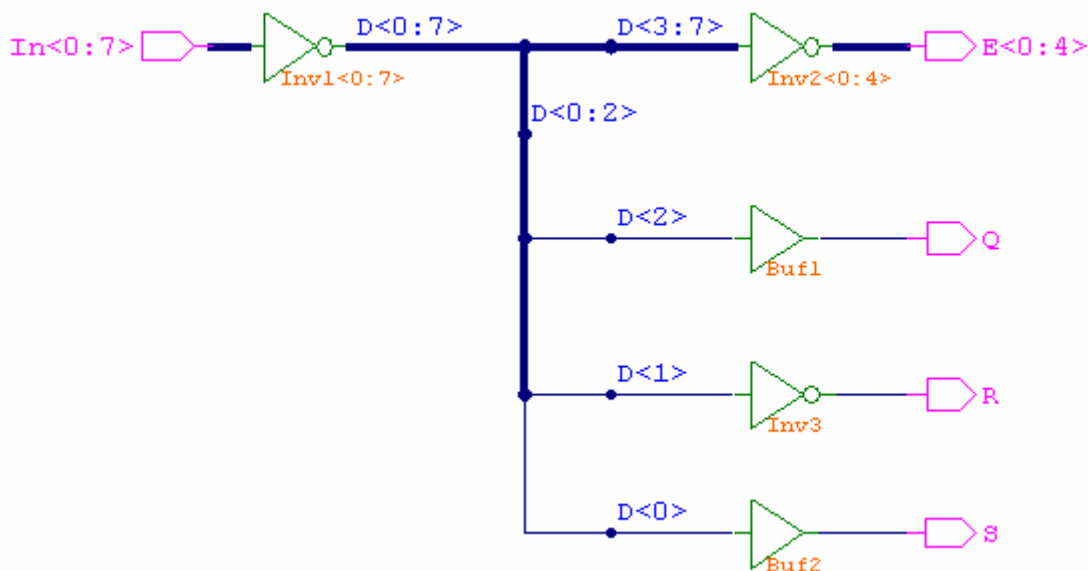
2.2. バスの分岐

デザイン BusesAndArrays

セル Example2

この事例は、バスを分岐する際に、バスやネットのラベルの設定条件を示します。8ビット幅のバス(In<0:7>)は8×インバータのアレイに入力されています。D<0:7> は出力です。8ビットのバス、D<0:7> は5ビット幅のバス(D<3:7>)と3ビットのバス(D<0:2>)に分岐されています。分岐されたバスのコンポーネントの合計と元のバスのコンポーネントの数、名前と番号が一致しなければなりません。これは各ブランチの次元はあいまいにならないように必要です。バス D<0:2>は各々のビット D<2>, D<1>, D<0> に分岐され、バッファ(Buf1)、インバータ(Inv3)、バッファ(Buf2)に接続されています。それらの出力は Q, R, S 出力ポートに接続されています。

回路図



Spice ネットリスト

```
XInv1<0> In<0> D<0> Gnd Vdd Inv
XInv1<1> In<1> D<1> Gnd Vdd Inv
XInv1<2> In<2> D<2> Gnd Vdd Inv
XInv1<3> In<3> D<3> Gnd Vdd Inv
XInv1<4> In<4> D<4> Gnd Vdd Inv
XInv1<5> In<5> D<5> Gnd Vdd Inv
XInv1<6> In<6> D<6> Gnd Vdd Inv
XInv1<7> In<7> D<7> Gnd Vdd Inv
```

```
XInv2<0> D<3> E<0> Gnd Vdd Inv
XInv2<1> D<4> E<1> Gnd Vdd Inv
XInv2<2> D<5> E<2> Gnd Vdd Inv
XInv2<3> D<6> E<3> Gnd Vdd Inv
XInv2<4> D<7> E<4> Gnd Vdd Inv
```

```
XBuf1 D<2> Q Gnd Vdd Buf
XInv3 D<1> R Gnd Vdd Inv
XBuf2 D<0> S Gnd Vdd Buf
.end
```

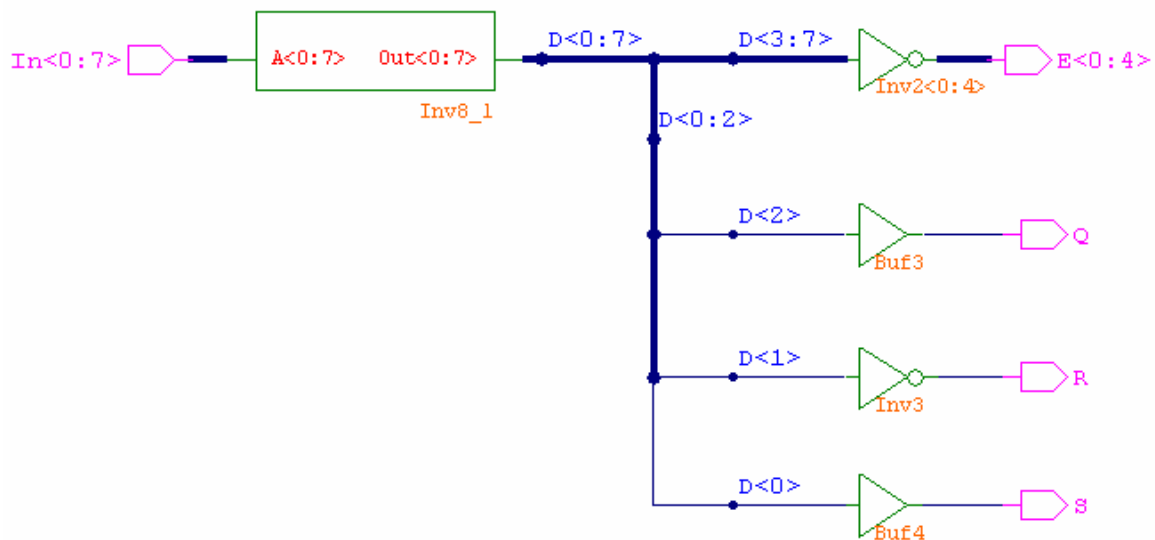
2.3. ポートバンドル

デザイン BusesAndArrays

セル Example3

この事例はシンボルのポートバンドルを示しています。Example3 は Example 2 に似ていますが、ここで8ビット入力バス(In<0:7>)はインスタンスのエイレイではなく、1つのインスタンス(Inv8_1)に接続されています。しかし、Inv8_1 のシンボルの入力、8ビットのポートバンドル(A<0:7>)に接続されているので次元一致しています。この事例にあるように、ポートバンドルは一つのバス(A<0:7>)であっていいです。または、A<0:4:2>, B<0:3>, C のようにいくつかのバスやネットの組合せでもいいです。

回路図



Spice ネットリスト

```
***** Subcircuits *****
.subckt Inv8 A<0> A<1> A<2> A<3> A<4> A<5> A<6> A<7> Out<0> Out<1> Out<2>
Out<3> Out<4> Out<5> Out<6> Out<7> Gnd Vdd
XBuf1 A<0> Out<0> Gnd Vdd Buf
XInv1 A<1> Out<1> Gnd Vdd Inv
XBuf2 A<2> Out<2> Gnd Vdd Buf
XInv2 A<3> Out<3> Gnd Vdd Inv
XBuf3 A<4> Out<4> Gnd Vdd Buf
XInv3 A<5> Out<5> Gnd Vdd Inv
XBuf4 A<6> Out<6> Gnd Vdd Buf
XInv4 A<7> Out<7> Gnd Vdd Inv
.ends

***** Simulation Settings - Analysis section *****
XInv8_1 In<0> In<1> In<2> In<3> In<4> In<5> In<6> In<7> D<0> D<1> D<2> D<3>
D<4> D<5> D<6> D<7> Gnd Vdd Inv8
XInv2<0> D<3> E<0> Gnd Vdd Inv
XInv2<1> D<4> E<1> Gnd Vdd Inv
XInv2<2> D<5> E<2> Gnd Vdd Inv
XInv2<3> D<6> E<3> Gnd Vdd Inv
XInv2<4> D<7> E<4> Gnd Vdd Inv
XBuf3 D<2> Q Gnd Vdd Buf
XInv3 D<1> R Gnd Vdd Inv
XBuf4 D<0> S Gnd Vdd Buf
.end
```

2.4. 1次元のアレイ(並列アレイ)

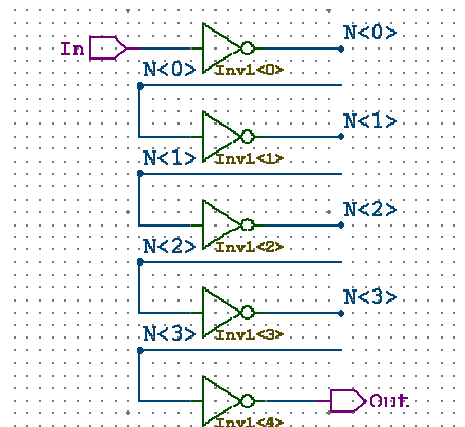
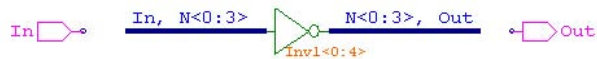
デザイン BusesAndArrays

セル Example4

この事例は、並列に接続を構成するために、アレイの入力と出力の接続仕方を示します。5×インバータアレイの入力は In ポートと N<0> バスでなるバンドルに接続されています(詳細には In, N<0:3>, In, N<0>, N<1>, N<2>, N<3> です)。アレイの出力は N<0:3> と Out ポートでなるバンドルに接続されています(詳細には N<0>, N<1>, N<2>, N<3>, Out です)。つまり一つのインバータの出力は次のインバータの入力に接続され、並列に並んだアレイとなります。

下の左図を展開すると右図が得られます。

回路図



Spice ネットリスト

```
XInv1<0> In N<0> Gnd Vdd Inv
XInv1<1> N<0> N<1> Gnd Vdd Inv
XInv1<2> N<1> N<2> Gnd Vdd Inv
XInv1<3> N<2> N<3> Gnd Vdd Inv
XInv1<4> N<3> Out Gnd Vdd Inv
.end
```

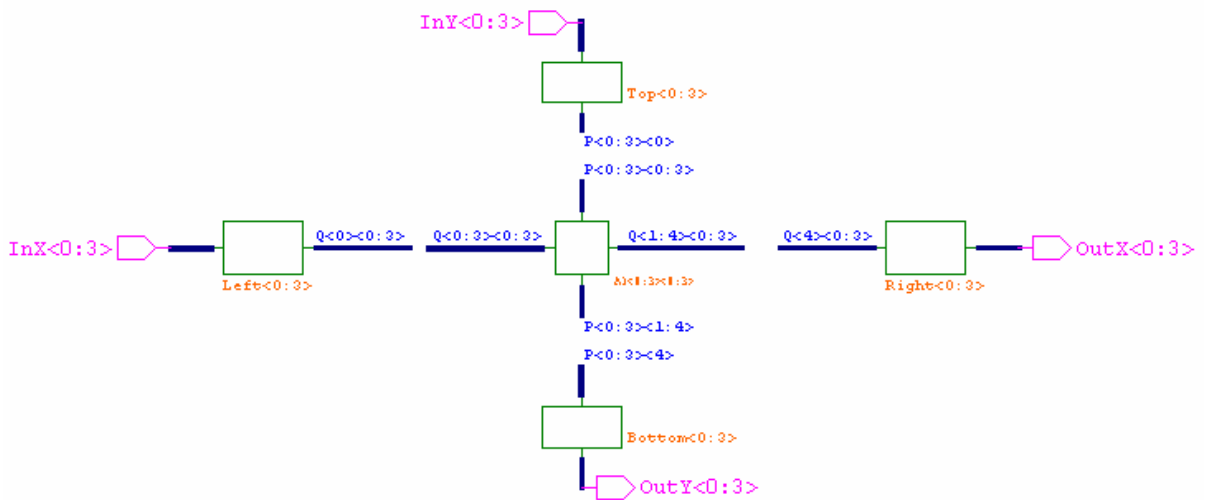
2.5. 2次元の配列

デザイン BusesAndArrays

セル Example5

この事例は、2次元の配列の記述と使用方法を示します。Left, Top, Bottom, と Right という名前のインスタンスの配列は、それぞれが4つのインスタンスでなる1次元の配列です。それらは2次元の配列 (A<0:3><0:3>) の周りに、直接ではなく、Example4 にあったように名前を定義した方法で接続されています。下記の Spice ネットリストでは、配列の構造と接続性が容易にわかるように内部のネットを色別に描いています。

回路図



Spice ネットリスト

```
XLeft<0> InX<0> Q<0><0> Left
XLeft<1> InX<1> Q<0><1> Left
XLeft<2> InX<2> Q<0><2> Left
XLeft<3> InX<3> Q<0><3> Left
```

```
XTop<0> InY<0> P<0><0> Top
XTop<1> InY<1> P<1><0> Top
XTop<2> InY<2> P<2><0> Top
XTop<3> InY<3> P<3><0> Top
```

```
XA1<0><0> Q<0><0> P<0><0> Q<1><0> P<0><1> A
XA1<0><1> Q<0><1> P<0><1> Q<1><1> P<0><2> A
XA1<0><2> Q<0><2> P<0><2> Q<1><2> P<0><3> A
XA1<0><3> Q<0><3> P<0><3> Q<1><3> P<0><4> A
```

```
XA1<1><0> Q<1><0> P<1><0> Q<2><0> P<1><1> A
XA1<1><1> Q<1><1> P<1><1> Q<2><1> P<1><2> A
XA1<1><2> Q<1><2> P<1><2> Q<2><2> P<1><3> A
XA1<1><3> Q<1><3> P<1><3> Q<2><3> P<1><4> A
```

```
XA1<2><0> Q<2><0> P<2><0> Q<3><0> P<2><1> A
XA1<2><1> Q<2><1> P<2><1> Q<3><1> P<2><2> A
XA1<2><2> Q<2><2> P<2><2> Q<3><2> P<2><3> A
XA1<2><3> Q<2><3> P<2><3> Q<3><3> P<2><4> A
```

XA1<3><0> Q<3><0> P<3><0> Q<4><0> P<3><1> A
 XA1<3><1> Q<3><1> P<3><1> Q<4><1> P<3><2> A
 XA1<3><2> Q<3><2> P<3><2> Q<4><2> P<3><3> A
 XA1<3><3> Q<3><3> P<3><3> Q<4><3> P<3><4> A

XRight<0> Q<4><0> OutX<0> Right
 XRight<1> Q<4><1> OutX<1> Right
 XRight<2> Q<4><2> OutX<2> Right
 XRight<3> Q<4><3> OutX<3> Right

XBottom<0> P<0><4> OutY<0> Bottom
 XBottom<1> P<1><4> OutY<1> Bottom
 XBottom<2> P<2><4> OutY<2> Bottom
 XBottom<3> P<3><4> OutY<3> Bottom

.end

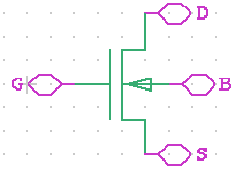
3. Spice エクスポート

3.1. Spice primitives

デザイン SpiceOutput

セル N4, P4, Inv

この事例は “primitive” デバイスを出力するための SPICE.OUTPUT プロパティの使用法を示します。“primitive” デバイスは最下階層レベルのデバイスのことであって、その下に他の回路図がありません。Spice ネットリストに出力されるテキストは、シンボルの SPICE.OUTPUT プロパティによって定義されます。つまり、シンボルの SPICE.OUTPUT プロパティは、シンボルがインスタンスされたすべての親回路で利用されるSpice出力テキストの定義をします。そして、SPICE.PRIMITIVE = True プロパティは、デバイスが primitive であって、その下に他の回路図がないことを意味します。下図に NMOS トランジスタ(N4 セル)のシンボルが表示されています。



```
W = '0.35u'  
L = '0.25u'  
match = ?match  
nfing = 1  
  
Cell = N4  
InstanceName =  
  
SPICE.OUTPUT = M${Name} % {D} % {G} % {S} % {B} NMOS W=${W} L=${L} nfing=${nfing} match=${match}  
SPICE.PRIMITIVE = true
```

NMOS トランジスタ(N4)のシンボル

このシンボルはプロパティが4つ持っています。

```
W = '0.35u'  
L = '0.25u'  
match = ?match    (“?” マークの意味について次の事例を参照してください)  
nfing = 1
```

このシンボルの SPICE.OUTPUT と SPICE.PRIMITIVE プロパティは次のようになっています。

```
SPICE.OUTPUT = M${Name} % {D} % {G} % {S} % {B} NMOS W=${W} L=${L}  
               nfing=${nfing} match=${match}  
SPICE.PRIMITIVE = True
```

Spice ネットリストをエクスポートする際に、Spice エクスポート制御専用プロパティの名前を入力しますが、それは .OUTPUT と .PRIMITIVE というサブプロパティを持っているプロパティの事です。ここでは、“SPICE” はエクスポート制御専用プロパティです。詳細は事例3.4 をご参照ください。

SPICE.OUTPUT の定義を詳細に調べるとプロパティの要素は次のようSpiceネットリストを書いているとわかります。

M	“M” と書きます
\${Name}	このインスタンスの“Name” というプロパティの値を書きます (インスタンス名)
% {D}, % {G}, % {S}, % {B}	D, G, S, と B ピンに接続しているネットの名前を書きます
NMOS	“NMOS” と書きます
W=	“W=” と書きます
\${W}	このインスタンスの“W” というプロパティの値を書きます
L=	“L=” と書きます

<code>\${L}</code>	このインスタンスの“L” というプロパティの値を書きます
<code>nfing=</code>	“nfing=” と書きます
<code>\${nfing}</code>	このインスタンスの“nfing=” というプロパティの値を書きます
<code>match=</code>	“match=” と書きます
<code>\${match}</code>	このインスタンスの“match=” というプロパティの値を書きます

セル P4 のシンボルは、セル N4 のシンボルと同じプロパティを持っています。さらに、SPICE.OUTPUT プロパティも同じです。セルInv には、セル N4 と セル P4 がインスタンスされています。Inv の Spice 出力は次のとおりです。

```
MP4_1 Out A Vdd Vdd PMOS W='0.35u' L='0.25u' nfing=1 match=match
MN4_1 Out A Gnd Gnd NMOS W='0.35u' L='0.25u' nfing=1 match=match
.end
```

上記では、インスタンス名、ネット名とプロパティの値が各々のSpiceラインに、SPICE.OUTPUT の定義が代用されていることがわかります。

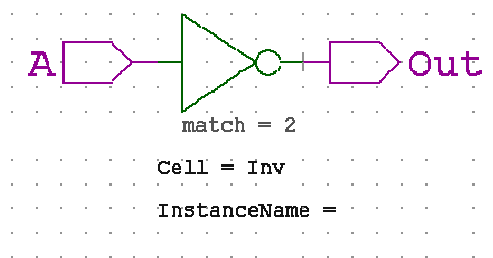
3.2. パラメータを下の階層のセルに引き渡す

デザイン SpiceOutput

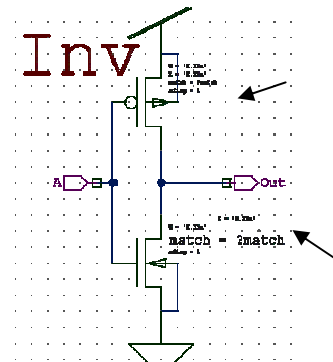
セル Inv, Inverters

この事例は、パラメータを下の階層へ引渡す方法を示します。セル Inverters はセル Inv のインスタンスを3つ含んでいます。セル Inv のシンボルビューには “match=2” というプロパティがあります (Evaluated Parameters ボタンをオフにしてください)。セル Inv のインスタンスされているセル P4 と N4 のシンボルビューには “match=?match” というプロパティがあるのです。“?”字で始まるプロパティは親回路図のプロパティを優先するという意味です。

回路図 Inverters で、一番目のインスタンス(インスタンス名Inv_1)は、“match” のデフォルトのプロパティを使っているため、パラメータの引渡しによって上書きされていません(Properties Window でそのプロパティの色が緑であることを確認してください)。一方、2番目のインスタンス(Inv_2)では、match プロパティがProperties Window で 3に変更されています(Properties Window でそのプロパティの色が黒になっていることを確認してください)。それで、セル Inv の元のパラメータ“match=2” が、インスタンス Inv_2 で、“match=3”に変更され、上書きされます。3番目のインスタンス(Inv_3)でもInv_2と同様に、“match=2” を“match=4”に変更されています。

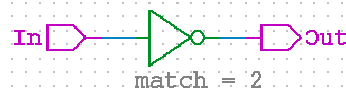


セルInvのシンボルビュー



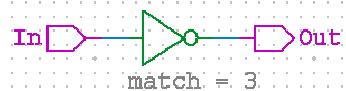
セルInvの回路図ビュー

インスタンス Inv_1 :



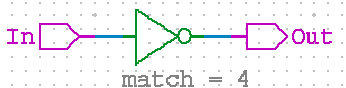
match = 2 is default value on symbol of Inv

インスタンス Inv_2 :



match = 3 is local override on instance Inv_2

インスタンス Inv_3 :



match = 4 is local override on instance Inv_3

セルInvertersの回路図ビュー

このセルのSpice 出力は次のようです

```
.subckt Inv A Out Gnd Vdd match=2
MP4_1 Out A Vdd Vdd PMOS W='0.35u' L='0.25u' nfing=1 match=match
MN4_1 Out A Gnd Gnd NMOS W='0.35u' L='0.25u' nfing=1 match=match
.ends
XInv_1 In Out Gnd Vdd Inv match=2
XInv_2 In Out Gnd Vdd Inv match=3
XInv_3 In Out Gnd Vdd Inv match=4
```

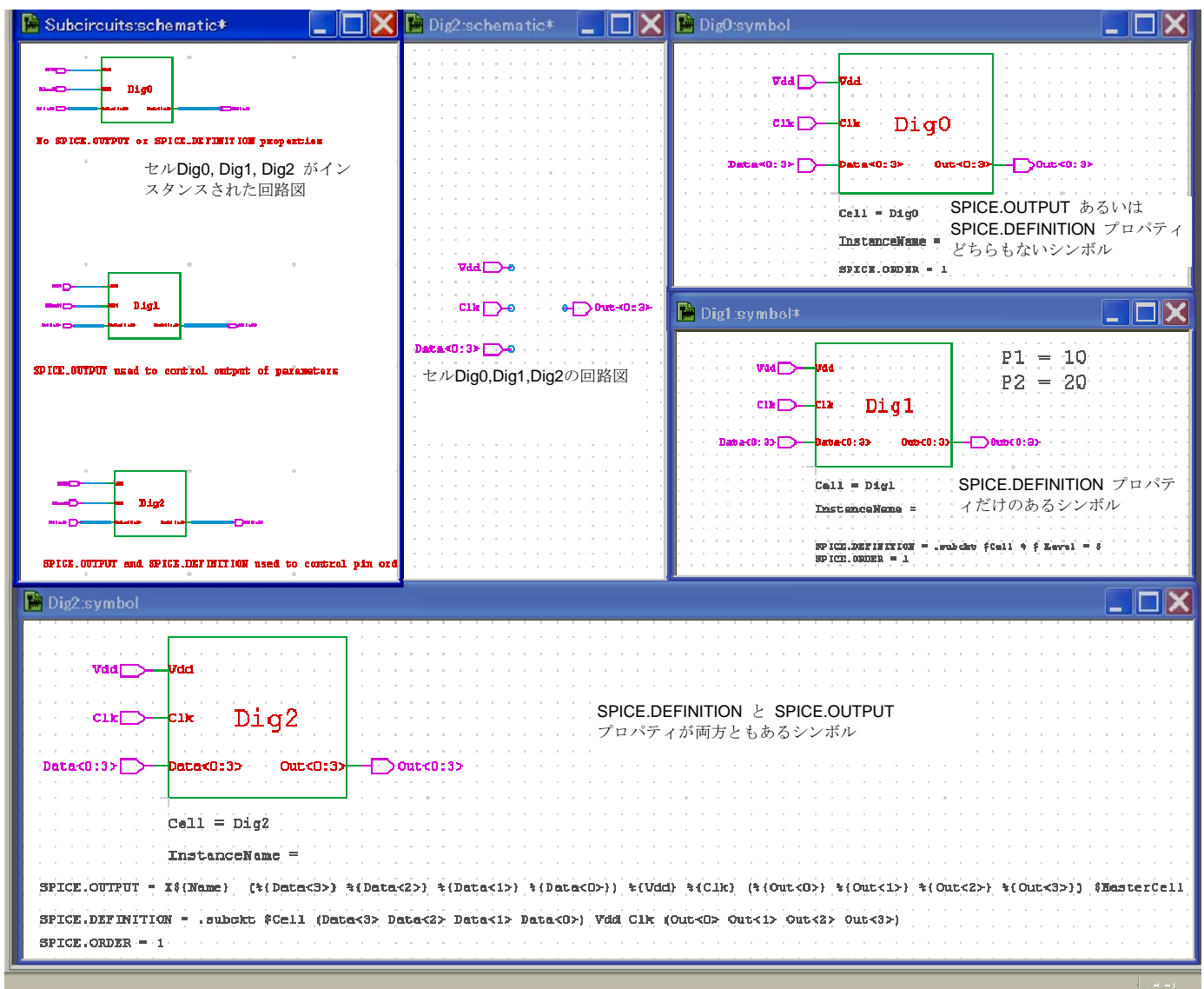
3.3. サブサーキット

デザイン SpiceOutput

セル Dig0, Dig1, Dig2, Subcircuits

この事例は SPICE.OUTPUT と SPICE.DEFINITION プロパティを使ってサブサーキットのSpice出力を制御する方法を示します。「サブサーキット」はprimitiveではないシンボルですので、SPICE.PRIMITIVE=True プロパティが不要です。

SPICE.DEFINITION プロパティはサブサーキットの構成を定義するために使われます。



◆ セル Dig0

セルDig0 のシンボルは SPICE.OUTPUT あるいは SPICE.DEFINITION プロパティどちらも持っていません。

SPICE.DEFINITION プロパティがない場合、サブサーキットの定義は全てのピンをABC順に含みます(グローバルポートは最後にABC順です)。

SPICE.OUTPUT プロパティがない場合、エクスポートされる Spice ネットリストに、インスタンスされたシンボルの全てのピンと、それに続けてインターフェースのパラメータが書かれます。インターフェースパラメータは、サブプロパティが IsInterface = True となっているパラメータです(Properties Windowで、SPICE の各々のプロパティの下にあります)。

セルSubcircuits をエクスポートすると、インスタンス Dig0 のサブサーキット定義とそのコールは次のようになります。

```
サブサーキットの定義は :
.subckt Dig0 Clk Data<0> Data<1> Data<2> Data<3> Out<0> Out<1> Out<2>
Out<3> Vdd
...
.ends
コールは :
XDig0_1 Clock A<0> A<1> A<2> A<3> B<0> B<1> B<2> B<3> PWR Dig0
```

◆ セル Dig1

セル Dig1 は、SPICE.DEFINITION プロパティを使ってサブサーキットの定義の中にパラメータの引渡し方法を示しています。

セルDig1のシンボルは次のようなSPICE.DEFINITION プロパティを持っています。

```
SPICE.DEFINITION = .subckt $Cell %% $$ Level = 5
```

ここで、SPICE.DEFINITION プロパティのステートメントの各エレメントは、Spice 定義インターフェースに次のように書かれます。

.subckt	“.subckt” と書きます
\$Cell	セルの名前を書きます
%%	シンボルのポートの名前をデフォルト順に書きます
\$\$	全てのインターフェース プロパティを次のように書きます “property_name1 = property_value1, ...”
Level = 5	“Level = 5” と書きます

セルSubcircuits をエクスポートすると、インスタンス Dig1 のサブサーキット定義と、そのコールは次のようになります。

```
.subckt Dig1 Clk Data<0> Data<1> Data<2> Data<3> Out<0> Out<1> Out<2>
Out<3> Vdd P1=10 P2=20 Level = 5
...
.ends

XDig1_1 Clock A<0> A<1> A<2> A<3> B<0> B<1> B<2> B<3> PWR Dig1 P1=10
P2=20
```

◆ セル Dig2

セル Dig2 は SPICE.OUTPUT と SPICE.DEFINITION プロパティを持ってピン順のカスタマイズと、サブサーキットの定義とサブサーキットのコールに特別な記述を追加する方法を示しています。

```
SPICE.DEFINITION = .subckt $Cell (Data<3:0>) Vdd Clk (Out<0:3>)  
SPICE.OUTPUT = X${Name} (%{Data<3:0>}) %{Vdd} %{Clk} (%{Out<0:3>}) $MasterCell
```

SPICE.DEFINITION プロパティのステートメントの各エレメントは、Spice定義インターフェースに次のように書かれます。

.subckt	“.subckt” と書きます
\$Cell	セルの名前を書きます
(“(“ と書きます
%{Data<3:0>}	デフォルト順を逆転し、データポートを書きます
)	”)” と書きます
%{Vdd}	Vdd ポートを書きます
%{Clk}	Clk ポートを書きます
(“(“ と書きます
%{Out<0:3>}	“Out” ポートを書きます
)	”)” と書きます

SPICE.OUTPUT プロパティのステートメントの各エレメントも同様に構成されています。セル“Subcircuits”をエクスポートすると、インスタンス Dig2 のサブサーキット定義と、そのコールは次のようになります。

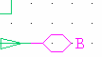
```
.subckt Dig2 (Data<3> Data<2> Data<1> Data<0>) Vdd Clk (Out<0> Out<1>  
Out<2> Out<3>)  
...  
.ends  
  
XDig2_1 (A<3> A<2> A<1> A<0>) PWR Clock (B<0> B<1> B<2> B<3>) Dig2
```

3.4. エクスポート制御用プロパティ

デザイン
セル

この事例はデバイスの Spice 出力を制御するために「エクスポート制御用プロパティ」の使用方を示します。あるデバイスの Spice 出力のために、いくつかの Spice 出力プロパティを定義することができます。例えば、デフォルトプロパティ、基本的なプロパティと詳細なプロパティなどです。それぞれのプロパティがシミュレーションの異なったレベルのために異なったパラメータを持つように設定できます。「エクスポート制御用プロパティ」は Spice を書き上げる時どの出力プロパティが使われるかを決定します。プロパティ名のリストが「エクスポート制御用プロパティ」に入力されると、Spice は最初の「エクスポート制御用プロパティ」に従ってエクスポートされます。

シンボルが下に表示されているセル P4 (PMOS トランジスタ)を考慮しましょう。下記に表示されるように、このシンボルで3つの .OUTPUT プロパティが定義されています。それぞれの .OUTPUTプロパティのためのSpiceエクスポート制御用プロパティは、SPICE, SPICE_BASIC, と SPICE_DETAILED です。



```

W = '0.35u'
L = '0.25u'
match = '?match'
nfing = 1

```

Cell = P4
 InstanceName = M = 1

```

SPICE.OUTPUT = M${Name} ${D} ${G} ${S} ${B} PMOS W=${W} L=${L} nfing=${nfing} match=${match}
SPICE.PRIMITIVE = true

SPICE_BASIC.OUTPUT = M${Name} ${D} ${G} ${S} ${B} PMOS W=${W} L=${L}
SPICE_BASIC.PRIMITIVE = true

SPICE_DETAILED.OUTPUT = M${Name} ${D} ${G} ${S} ${B} PMOS W=${W} L=${L} nfing=${nfing} match=${match} M=${M}
SPICE_DETAILED.PRIMITIVE = true

```

◆ セル INVでSPICEをエクスポートするとき、Spiceエクスポート制御用プロパティとして

「**SPICE DETAILED, SPICE, SPICE BASIC**」というリストを入力すると、次の出力が得られます。

```
MP4_1 Out A Vdd Vdd PMOS W='0.35u' L='0.25u' nfing=1 match=match M=1
MN4_1 Out A Gnd Gnd NMOS W='0.35u' L='0.25u' nfing=1 match=match
```

インスタンス P4 の出力には、SPICE_DETAILED.OUTPUT が利用されますが、それはリストで先に設定された Spice エクスポート制御用プロパティであり、P4のシンボルに定義されているからです。N4 シンボルに、SPICE.OUTPUT プロパティしか定義されていないので、インスタンス N4の出力でSPICE.OUTPUT プロパティが利用されています。

◆ セル INVでSPICEをエクスポートするとき、Spiceエクスポート制御用プロパティとして「**SPICE_BASIC, SPICE**」というリストを入力すると、次の出力が得られます。

```
MP4 1 Out A Vdd Vdd PMOS W='0.35u' L='0.25u'
```

```
MN4_1 Out A Gnd Gnd NMOS W='0.35u' L='0.25u' nfing=1 match=match
```

ここで、インスタンス P4 の出力のために、SPICE_BASIC.OUTPUT が利用されました。なぜなら、それはリストで最初に設定された Spiceエクスポート制御用プロパティであり、P4のシンボルに定義されているからです。N4 シンボルに、SPICE.OUTPUT プロパティしか定義されていないので、インスタンス N4の出力で SPICE.OUTPUT プロパティが利用されています。

◆ セル INVでSPICEをエクスポートするとき、Spiceエクスポート制御用プロパティとして単に「 SPICE 」と入力すると、次の出力が得られます。

```
MP4_1 Out A Vdd Vdd PMOS W='0.35u' L='0.25u' nfing=1 match=match
```

```
MN4_1 Out A Gnd Gnd NMOS W='0.35u' L='0.25u' nfing=1 match=match
```

P4 と N4 の インスタンスの出力に、SPICE.OUTPUT プロパティが利用されています。

4. グローバルネット、グローバルシンボルと、ネットキャップ

4.1. シンプル グローバルネット

デザイン Globals1
セル Toplevel

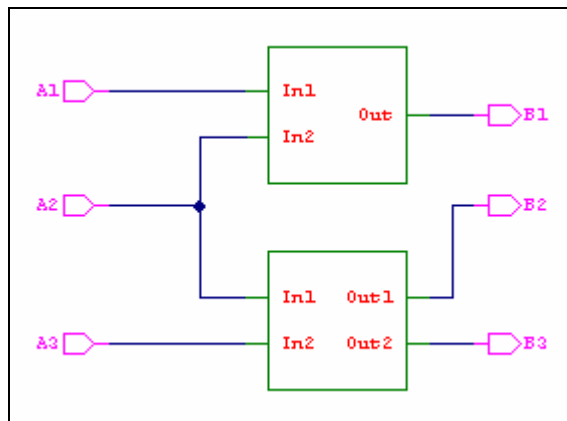
S-Edit で、グローバルネットがデザイン階層を通して接続されます。それらを接続するために各レベルでポートを配置する必要はないです。この事例では、セル Toplevel では coreHV と coreLV セルがインスタンスされています。coreHV の中に Block2 と Block3, そして coreLV の中に Block1 と Block2 のインスタンスがあります。Block1, Block2, と Block3 の回路図では Vdd (そして Gnd) というグローバルシンボルがあります。

このデザインでは、Vdd (そしてGnd) はグローバルネットであって、デザイン階層を通して接続されます。

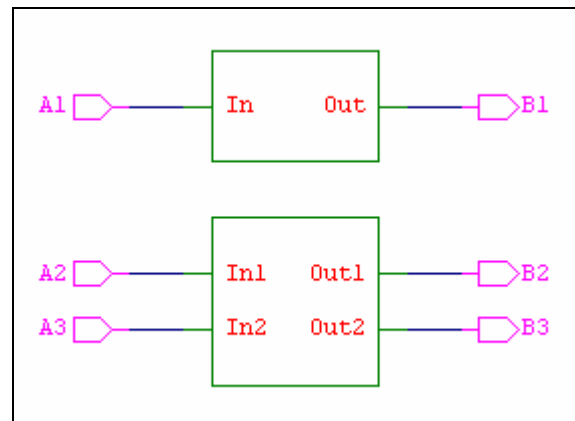
回路図



セル Toplevel の回路図



セル HVの回路図



セル LV の回路図

Spice ネットリスト

```
***** Subcircuits *****
.subckt Block1 In_NC_21 Out_NC_22 Gnd vdd
.ends

.subckt Block2 In1_NC_23 In2_NC_24 Out1_NC_25 Out2_NC_26 Gnd vdd
.ends

.subckt Block3 In1_NC_27 In2_NC_28 Out_NC_29 Gnd vdd
.ends

.subckt coreHV A1 A2 A3 B1 B2 B3 Gnd vdd
XU2 A2 A3 B2 B3 Gvdd Block2
nd XU1 A1 A2 B1 Gnd vdd Block3
.ends

.subckt coreLV A1 A2 A3 B1 B2 B3 Gnd vdd
XU1 A1 B1 Gnd vdd Block1
XU2 A2 A3 B2 B3 Gnd vdd Block2
.ends

***** Simulation Settings - Analysis section *****
Xcore_LV N_10 N_7 N_12 N_9 N_11 N_8 Gnd vdd coreLV
Xcore_HV N_4 N_1 N_6 N_3 N_5 N_2 Gnd vdd coreHV

.end
```

4.2. 別々の電圧源

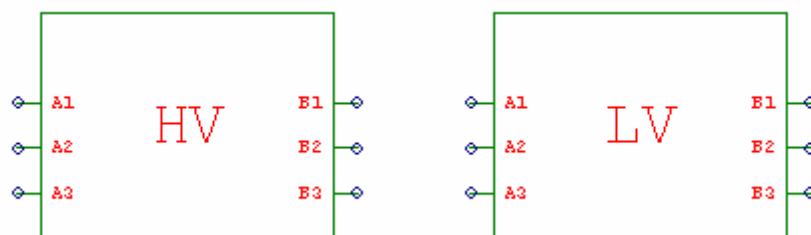
デザイン Globals2
セル Toplevel

この事例は、2つのセルに含まれているグローバル Vdd ネットの切り離す方法を示します。デザイン Globals1 にあった2つのセルを考慮しましょう。coreHV にあるグローバル Vdd を coreLV にあるグローバルVdd から切り離したいです。

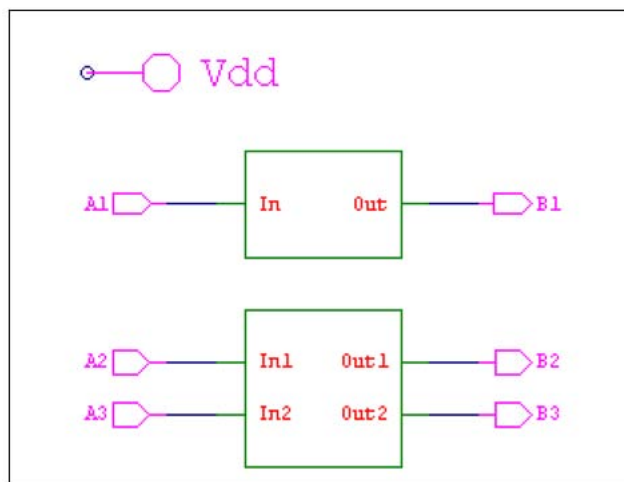
デザイン Globals2 は デザイン Globals1 の セル coreHV と coreLV で Vdd のキャップネットを配置することによって得られたものです。グローバルネットを正しくキャップするために、ネットヤップの名前とキャップされるネットの名前は一致しなければなりません(大文字と小文字の区別を含めて)。下記に Spice ネットリストが記載されています。coreHV と coreLV の定義でパラメータリストの中に Vdd はもうないです。そしてさらに、親回路 Toplevel で coreHV とcoreLV コールの中にもないことにご注意ください。従って、coreHV の中の Vdd と coreLVの中の Vdd は接続されていません。同じ名前ですが、別々のネットになっています。

coreHV と coreLV の中の Vdd ネットは、ネットキャップを削除することによって、或いは Spice ネットリストの中に `.global Vdd` というコマンドを追加することによって再接続できます。ネットリストの中に `.global Vdd` というコマンドを S-Edit によって自動的に入れたい場合、`SPICE.OUTPUT = .global Vdd` というプロパティのあるシンボルを作り、それを設計のトップレベルにインスタンスします。

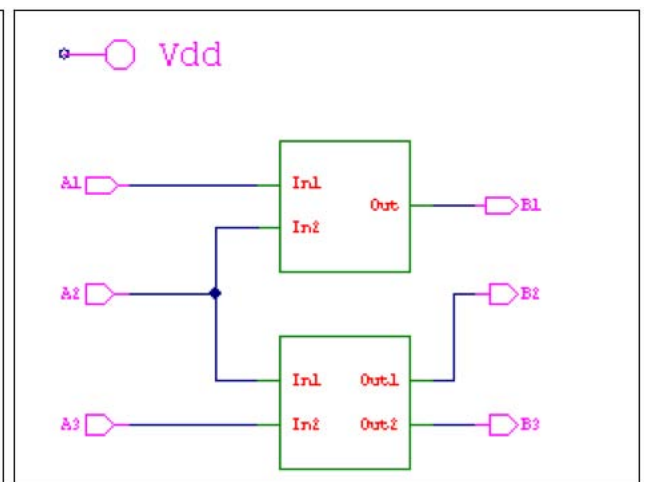
回路図



セルToplevel の回路図



セル HV の回路図



セル LV の回路図

Spice ネットリスト

```
***** Subcircuits *****
.subckt Block1 In_NC_1 Out_NC_2 Gnd Vdd
.ends

.subckt Block2 In1_NC_3 In2_NC_4 Out1_NC_5 Out2_NC_6 Gnd Vdd
.ends

.subckt Block3 In1_NC_7 In2_NC_8 Out_NC_9 Gnd Vdd
.ends

.subckt coreHV A1 A2 A3 B1 B2 B3 Gnd
XU2 A2 A3 B2 B3 Gnd Vdd Block2
XU1 A1 A2 B1 Gnd Vdd Block3
.ends

.subckt coreLV A1 A2 A3 B1 B2 B3 Gnd
XU1 A1 B1 Gnd Vdd Block1
XU2 A2 A3 B2 B3 Gnd Vdd Block2
.ends

***** Simulation Settings - Analysis section *****
Xcore_LV N_10 N_7 N_12 N_9 N_11 N_8 Gnd coreLV
Xcore_HV N_4 N_1 N_6 N_3 N_5 N_2 Gnd coreHV

.end
```

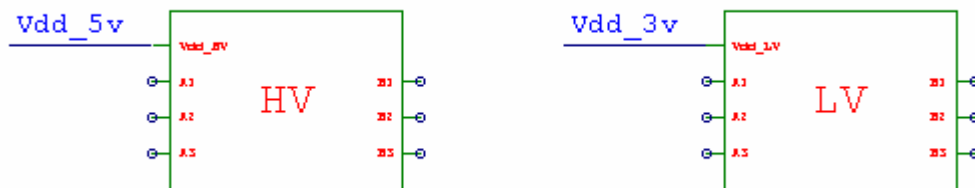
4.3. 別々の電圧源の名前の変更

デザイン Globals3
セル Toplevel

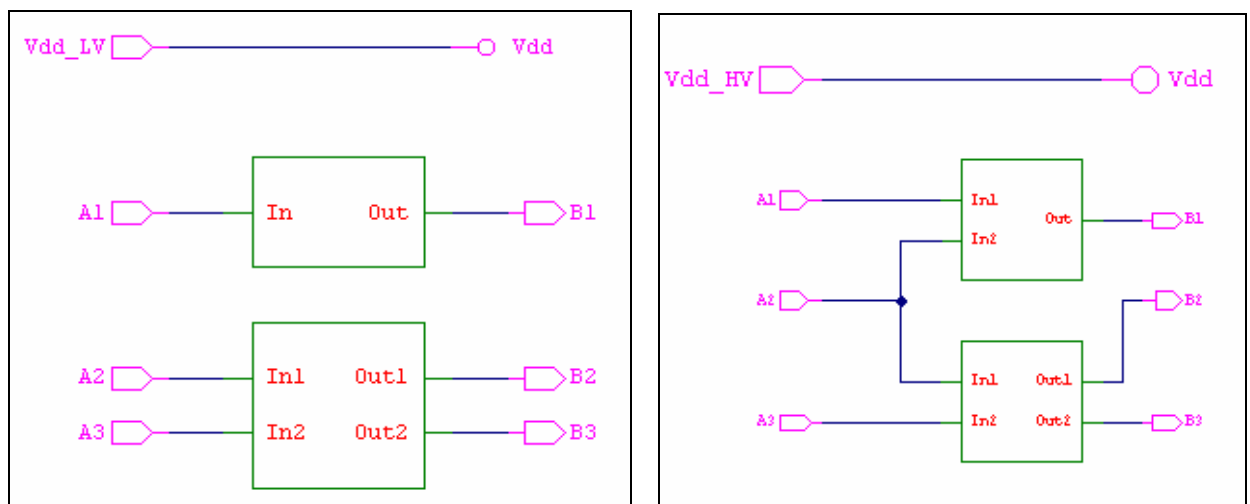
この事例は、2つのセルに含まれているグローバル Vdd ネットを切り離し、別々の名前にする方法を示します。デザイン Globals2 で、グローバル Vdd と切り離れた coreHV と coreLV にある Vdd をそれぞれ Vdd_5v と Vdd_3v という名前に変更したいです。

デザイン Globals3 は デザイン Globals2 の セル coreHV に Vdd_HV、coreLV に Vdd_LV という名前で入力ポートを回路図とシンボルビューで追加することによって得られたものです。回路図ビューで、新しいポートはネットキャップに接続されています。だから、Vdd ネットは階層の中で別の名前で接続しているということになっています。セル Toplevel の回路図の下記に表示されている Spice ネットリストで、coreHV と coreLV へのコールの行では、Vdd_5v が coreHV に、Vdd_3v が coreLV に接続されていることを確認できます。

回路図



セルToplevel の回路図



セルHV の回路図

セルLV の回路図

Spice ネットリスト

```
***** Subcircuits *****
.subckt Block1 In_NC_28 Out_NC_29 Gnd Vdd
.ends

.subckt Block2 In1_NC_30 In2_NC_31 Out1_NC_32 Out2_NC_33 Gnd Vdd
.ends

.subckt Block3 In1_NC_34 In2_NC_35 Out_NC_36 Gnd Vdd
.ends

.subckt coreHV A1 A2 A3 B1 B2 B3 Vdd Gnd
XU2 A2 A3 B2 B3 Gnd Vdd Block2
XU1 A1 A2 B1 Gnd Vdd Block3
.ends

.subckt coreLV A1 A2 A3 B1 B2 B3 Vdd Gnd
XU1 A1 B1 Gnd Vdd Block1
XU2 A2 A3 B2 B3 Gnd Vdd Block2
.ends

***** Simulation Settings - Analysis section *****
Xcore_LV N_10 N_7 N_12 N_9 N_11 N_8 Vdd_3v Gnd coreLV
Xcore_HV N_4 N_1 N_6 N_3 N_5 N_2 Vdd_5v Gnd coreHV

.end
```

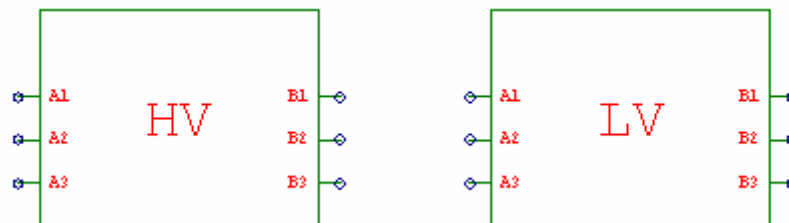
4.4. 別々の電圧源の名前の変更(他の方法)

デザイン Globals4
セル Toplevel

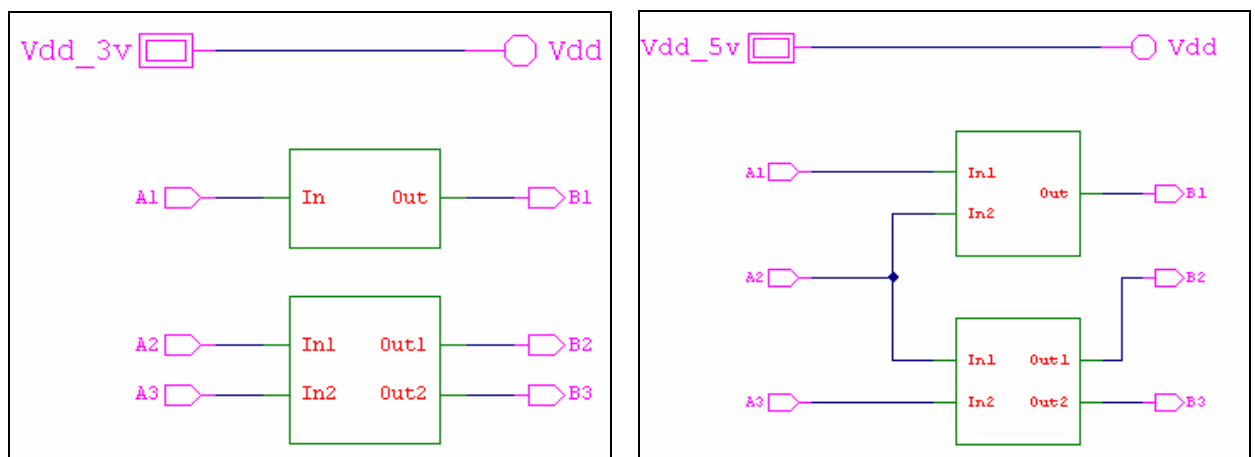
この事例は、2つのセルに含まれているグローバル Vdd ネットを切り離し、別々の名前にするもう一つの方法を示します。デザイン Globals2 でグローバル Vdd と切り離れた coreHV と coreLV にある Vdd をそれぞれ Vdd_5v と Vdd_3v という名前に変更したいです。

デザイン Globals4 は デザイン Globals2 の セル coreHV に Vdd_HV、coreLV に Vdd_LV という名前でグローバルポートを回路図ビューだけで追加することによって得られたものです。回路図ビューで、新しいポートはネットキャップに接続されています。だから、Vdd ネットは階層の中で別の名前で接続しているということになっています。グローバルポートの名前はネットキャップの名前に優先されます。セルToplevel の回路図の下記にあるネットリストで、coreHV と coreLV へのコールの行では、Vdd_5v は coreHV に、Vdd_3v は coreLV に接続されていることを確認できます。

回路図



Toplevelセルの回路図



HVセルの回路図、 LVセルの回路図

Spice ネットリスト

```
***** Subcircuits *****
.subckt Block1 In_NC_10 Out_NC_11 Gnd Vdd
.ends

.subckt Block2 In1_NC_12 In2_NC_13 Out1_NC_14 Out2_NC_15 Gnd Vdd
.ends

.subckt Block3 In1_NC_16 In2_NC_17 Out_NC_18 Gnd Vdd
.ends

.subckt coreHV A1 A2 A3 B1 B2 B3 Gnd Vdd
XU2 A2 A3 B2 B3 Gnd Vdd Block2
XU1 A1 A2 B1 Gnd Vdd Block3
.ends

.subckt coreLV A1 A2 A3 B1 B2 B3 Gnd Vdd
XU1 A1 B1 Gnd Vdd Block1
XU2 A2 A3 B2 B3 Gnd Vdd Block2
.ends

***** Simulation Settings - Analysis section *****
Xcore_LV N_10 N_7 N_12 N_9 N_11 N_8 Gnd Vdd_3v coreLV
Xcore_HV N_4 N_1 N_6 N_3 N_5 N_2 Gnd Vdd_5v coreHV

.end
```

5. マルチビュー

5.1. 4端子と3端子MOSFET シンボル

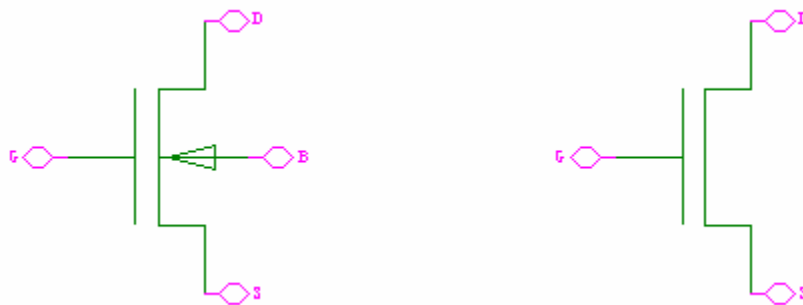
デザイン MultipleViews
セル MOSFET_N

この事例はセルの中でマルチビューの使用を示します。セル NMOS MOSFET には、4端子のシンボルと3端子のシンボルがあります。3端子のシンボルの4端子目はSPICE.OUTPUTプロパティによってGNDに接続されています。

セルMOSFET_N は次のように、2つずつのインターフェースビューとシンボルビューでできています。

4端子	nmos mosfet インターフェースビュー	: nmos4
4端子	nmos mosfet シンボルビュー	: nmos4
3端子	nmos mosfet インターフェースビュー	: nmos3
3端子	nmos mosfet シンボルビュー	: nmos3

セル MOSFET_N はprimitiveセルですので、回路図ビューはありません。



NMOS MOSFET の4端子と、3端子シンボルビュー

3端子のシンボルの4端子目は SPICE.OUTPUT プロパティで %B} の代わりに GND を書くことによってショートされています。それぞれのシンボルの SPICE.OUTPUT プロパティは下記に記載されています。

3端子 SPICE.OUTPUT プロパティ:

```
SPICE. OUTPUT = M$ {Name} % {D} % {G} % {S} GND $ {model}  
L=$ {L} W=$ {W} AD=$ {AD} PD=$ {PD} AS=$ {AS} PS=$ {PS}
```

4端子 SPICE.OUTPUT プロパティ:

```
SPICE. OUTPUT = M$ {Name} % {D} % {G} % {S} % {B} $ {model}  
L=$ {L} W=$ {W} AD=$ {AD} PD=$ {PD} AS=$ {AS} PS=$ {PS}
```

セル MOSFET_P は同様に、4端子と3端子シンボルビューを持っている PMOS MOSFET です。

5.2. IEEE と IEC シンボルビューを持つ NMOSセル

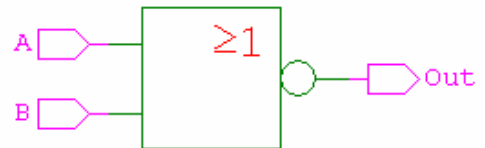
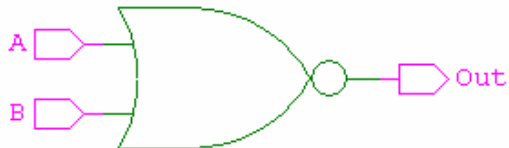
デザイン MultipleViews

セル NOR2

この事例はセルの中でマルチシンボルのビューの使用を示します。セル NOR ゲートには IEEE と IEC という2つシンボルがあります。

セルNOR2 は次のようにインターフェースビューが1つと、シンボルビューが2つと、回路図ビューが1つでできています。

インターフェース ビュー	: view_1
IEEE シンボルビュー	: IEEE
IEC シンボルビュー	: IEC
回路図 ビュー	: view_1



セル NOR の IEEE と IEC シンボル

両方のシンボルも同じインターフェースと回路図を参照しています。

5.3. ピンが異なる方法で整理されたシンボルが2つ持つアダー

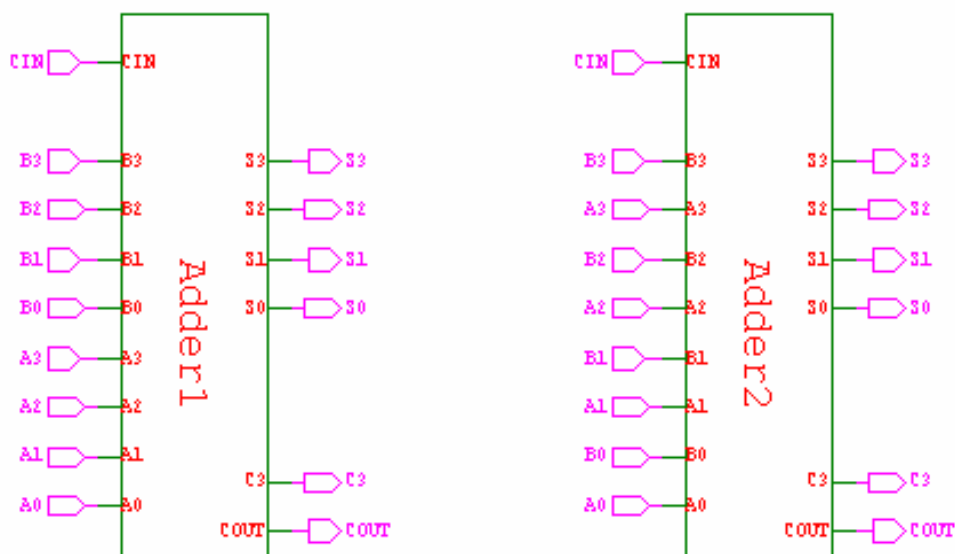
デザイン MultipleViews

セル Adder

この事例はセルの中でマルチシンボルのビューの使用方を示します。セルAdder には、次のようにインターフェースビューが1つと、シンボルビューが2つと、回路図ビューが1つあります。

インターフェースビュー : Adder
回路図ビュー : Adder
ピンが整理されたシンボルビュー: Adder1
ピンが別の方法で整理されたシンボルビュー: Adder2

回路図を描くとき、シンボルを接続するために、都合によって、あるときはシンボルのピンを一つの方法で、他の時は別の方法で整理する必要があります。これは、ピンがそれぞれに異なった方法で整理されたマルチシンボルビューを持たつことによって達成できます。ここで一つのシンボルが左側で入力ピンを A0 、 A1 、 A2 、 A3 、 B0 、 B1 、 B2 、 B3 の順で整理しています。そしてもう一つのシンボルがピンを A0 、 B0 、 A1 、 B1 、 A2 、 B2 、 A3 、 B3 の順で整理しています。これは単純に便利のためであって、Spiceに出力されるピンの順番に影響を与られません。



6. TCL/TK スクリプト

6.1. テキストの編集

スクリプト fixup_vdd_labels.tcl

このサンプルスクリプトは、アクティブセル、あるいは、デザインのすべての回路図のビューで「vdd」という名のすべてのポートとネットラベルを「VDD」に変えます。

スクリプトをロードしたら、2つのファンクションが利用できます。

fix_vdd_names: 現在アクティブのセルの中にある、すべてのポートとネットラベル'vdd'から 'VDD' に変更します

fixall: データベースにあるすべての回路図を開いて'fix_vdd_names' を実行します

このスクリプトを実行するために、コマンドウィンドーの中にスクリプトのファイルをドラッグして(または メニューの File > Open > Execute Script を使ってロードして)関数をロードします。次にコマンドウィンドーで “fixall” と入力します。コマンドウィンドーで“help”と入力するとタイプすることによって、S-Edit TCL で利用可能なコマンドの全てがリストされます。特定のコマンドやサブコマンドやオプションに関する詳細な情報を得るためにコマンド名 と続けて“-help” と入力します。

```
# find all ports and netlabels called 'vdd' and change them to 'VDD'
# (in the current view)
proc fix_vdd_names { args } {
    find port vdd
    property set -system Name -value VDD
    find netlabel vdd
    property set -system Name -value VDD
} proc ForEachSchematicView { UserProcName } {
    mode renderoff
    set count 0
    set cells [ database cells ]
    foreach cell $cells {
        # traverse all interface views
        set interfaces [ database interfaces -cell $cell ]
        foreach interface $interfaces {
            # traverse all schematic views
            set views [ database views -cell $cell -type schematic
                                -interface $interface ]

            foreach view $views {
                # puts "Opening schematic($view) of cell $cell"
                cell open -cell $cell -view $view -type schematic
                                -interface $interface
                eval "$UserProcName $cell $view $interface"
                incr count
            }
        }
    }
    mode renderon
    return $count
}
proc fixall {} {
    set n [ForEachSchematicView fix_vdd_names]
    puts "$n views processed" }
```

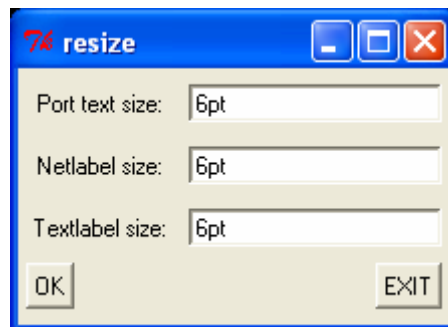
6.2. パラメータを入力する TK ダイアログを使ってテキストのサイズの変更

スクリプト `resizeText.tcl`

このサンプルスクリプトは、デザインの中の全ての回路図とシンボルビューを循環し、ポート、ネットラベル、テキストラベルのサイズを変更する方法を示しています。また、パラメータをスクリプトに入力するためにダイアログを作る TK の使用方法をも示します。

中にパラメータを入力するウィンドウを宣言するには `toplevel` コマンドを使います。デフォルトの `toplevel` ウィンドウは S-Edit アプリケーションウィンドウであるため、全ての TK スクリプトに必要であり、ユーザによって変更することは許されていません。

コマンドウィンドウの中にスクリプトのファイルをドラッグアンドドロップすることによって(または、メニューの `File > Open > Execute Script`)スクリプトをロードした後に次のダイアログが開きます。それぞれのフィールドに希望のテキストサイズを入力して OK を押すと、すべてのビューでポート、ネットラベル と テキストラベルのサイズが変わります。

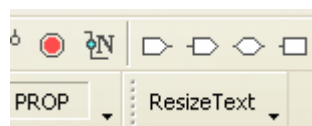


ポート、ネットラベル、テキストラベルサイズを入力するための TK ダイアログ

スクリプトを実行するために、ツールバーの上にボタンを作ることができます。そのために、`workspace userbutton set` コマンドを使います。用意されているスクリプト中最後の 2 行で、コメントアウト用のマーク “#” を 次のようにすると、スクリプトをロードしたときではなく、ボタンをクリックしたときに `ResizeText` が実行するようになります。

```
workspace userbutton set ResizeText
# ResizeText
```

ツールバーにボタンを追加するため行う変更



`ResizeText` スクリプトを実行できるツールバーの上に作られたユーザボタン

デザインが開かれるとき、デザインフォルダの直下の `scripts¥open.design¥` に置かれたスクリプトが自動的にロードされます。よく使うスクリプトを、下記のフォルダに入れると、S-Edit が起動したときに自動的にロードされるようになります(<username> は一般的にはWindowsへログインするときのログイン名です)。

`C:\Documents and Settings\<username>\Application Data\Tanner EDA\scripts\startup`

どのデザインを開いたときにでも、スクリプトが自動的にロードされるようにするために、次のフォルダに入れます。
C:\Documents and Settings\<username>\Application Data\Tanner EDA\scripts\open.design

スクリプトが、S-Edit終了時に自動的にロードされるようにするために次のフォルダに入れます。
C:\Documents and Settings\<username>\Application Data\Tanner EDA\scripts\shutdown

ResizeText スクリプトは次のとおりです。

```
# This sample script illustrates how to cycle over all schematic views in
# the design, and size all ports, netlabels, and labels to a user
# specified size. The script shows how to use TK to create a simple dialog
# to enter parameters for the script.

# resize all ports, netlabels, and labels (in the current view)
proc resize_text { new_portsize new_netlabelsize new_textlabelsize } {
    find none
    if { [string length $new_portsize] } {
        find port
        property set -system FontSize -value $new_portsize
    }
    if { [string length $new_netlabelsize] } {
        find netlabel
        property set -system FontSize -value $new_netlabelsize
    }
    if { [string length $new_textlabelsize] } {
        find textlabel
        property set -system FontSize -value $new_textlabelsize
    }
    find none
}

proc resizetext { new_portsize new_netlabelsize new_textlabelsize } {
    mode renderoff
    foreach cv [ database cellviews ] {
        set type [lindex $cv 4]
        if { $type == "schematic" || $type == "symbol" } {
            set design [lindex $cv 0]
            set cell [lindex $cv 1]
            set view [lindex $cv 2]
            set interface [lindex $cv 3]

            cell open -design $design -cell $cell -view $view
                -interface $interface -type $type
            resize_text $new_portsize $new_netlabelsize
                $new_textlabelsize
        }
    }
    mode renderon
}

proc ResizeText {} {
    global resize_portsize
    global resize_netlabelsize
    global resize_textlabelsize

    set resize_portsize 6pt
    set resize_netlabelsize 6pt
}
```

```

set resize_textlabelsize 6pt

toplevel .resize

set portprompt [label .resize.portprompt -text "Port text size:" ]
set portsize [entry .resize.portsize -textvariable resize_portsize ]

set netlabelprompt [label .resize.netlabelprompt
                    -text "Netlabel size:" ]
set netlabelsize [entry .resize.netlabelsize
                  -textvariable resize_netlabelsize ]

set labelprompt [label .resize.labelprompt -text "Textlabel size:" ]
set labelsize [entry .resize.labelsize
               -textvariable resize_textlabelsize ]

set cmdframe [ frame .resize.buttons ]
set ok [ button .resize.buttons.ok -text OK -command { \
                resizetext $resize_portsize $resize_netlabelsize
                $resize_textlabelsize ; \
                destroy .resize \
            } ]
set quit [ button .resize.buttons.quit -text EXIT -command
          {destroy .resize} ]
pack $ok -side left
pack $quit -side right

grid $portprompt $portsize -row 0 -sticky ew -padx 4 -pady {8 4}
grid $netlabelprompt $netlabelsize -row 1 -sticky ew -padx 4
    -pady {8 4}
grid $labelprompt $labelsize -row 2 -sticky ew -padx 4 -pady {8 4}
grid $cmdframe -column 0 -columnspan 2 -row 3 -sticky ew -padx 4
    -pady {4 8}
grid columnconfigure .resize 1 -weight 1
focus $portsize
}

#workspace userbutton set ResizeText
ResizeText

```