

ispLEVER CLASIC 1.2 Startup Manual for MACH4000

Rev.1.0

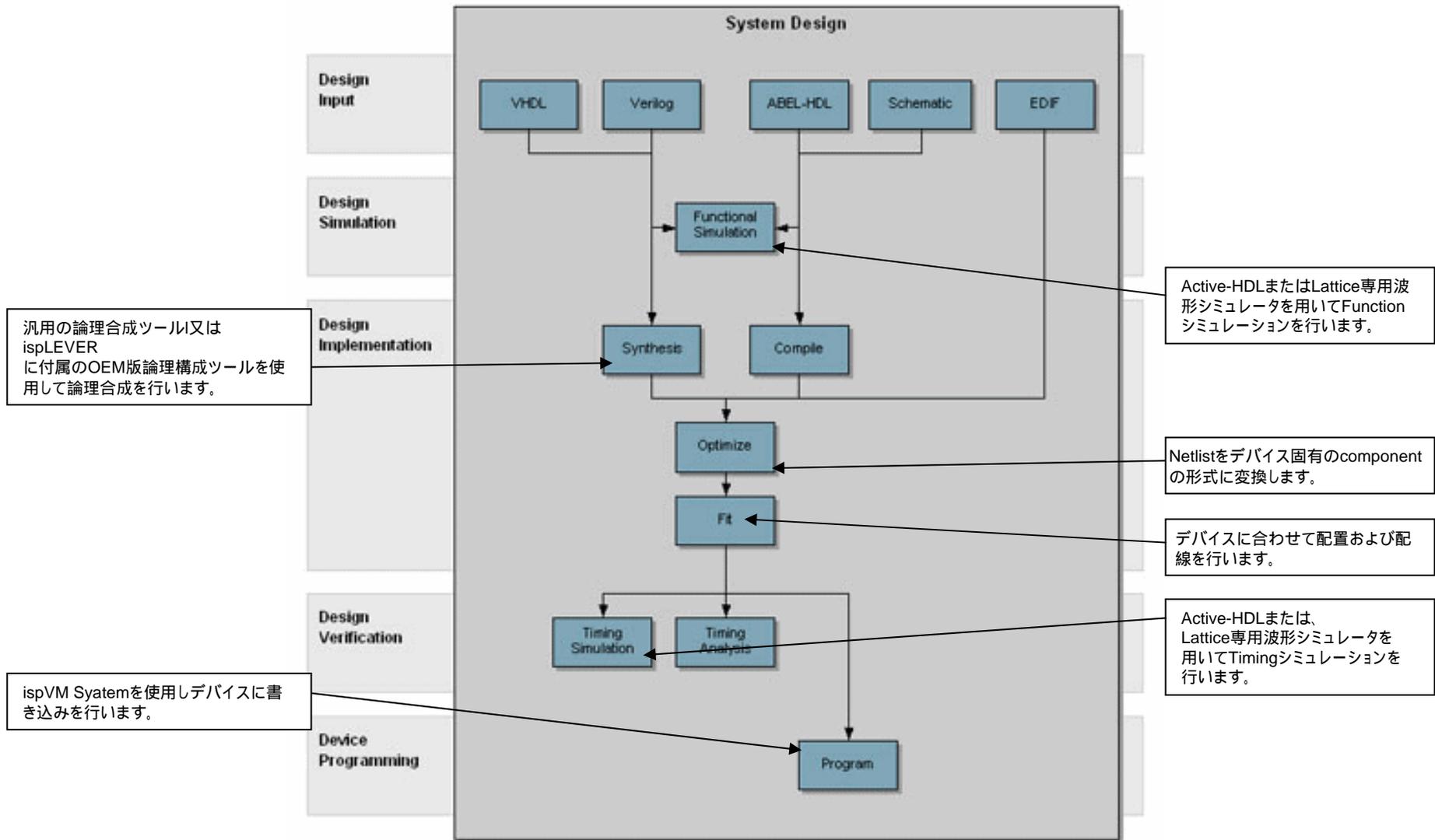
1.	はじめに	Page 3
2.	Lattice ispLEVER Design Flow	Page 4
3.	ツールの起動	Page 5
	3-1 新規プロジェクト作成	Page 6
	3-2 ターゲットデバイス選択	Page 7
	3-3 デザインエントリ	Page 8
	3-4 VHDLソースの入力	Page 9
	3-5 回路図作成	Page 11
4.	コンパイル作業の開始	Page 16
	4-1 論理合成ツールについて	Page 17
	4-2 論理合成時のオプション設定について	Page 18
5.	ピン固定 (Package Viewを用いた設定)	Page 22
	5-1 Location Assignmentを用いた設定	Page 23
	5-2 Compile後できる便利な設定	Page 24
	5-3 回路図上での設定	Page 25
	5-4 IOの詳細設定について	Page 26
	5-5 ピンの属性の設定について	Page 30
6.	再コンパイルとレポートファイルの生成	Page 32
7.	タイミング解析について	Page 33
8.	波形シミュレーションについて	Page 34
9.	おわりに	Page 39

- ・このマニュアルはispLEVERのオペレーションフローマニュアルです。
- ・簡単な回路図及びVHDLソースコードの作成からデバイスへのフィッティング、波形シミュレーションまでの一連のフローを解説したものです。
- ・各項目の詳細については、別途ツール取り扱いマニュアルもしくは弊社技術サポートまでお問い合わせください。



株式会社マクニカ
テクスター カンパニー
TEL: 045-470-9841
lattice@macnica.co.jp

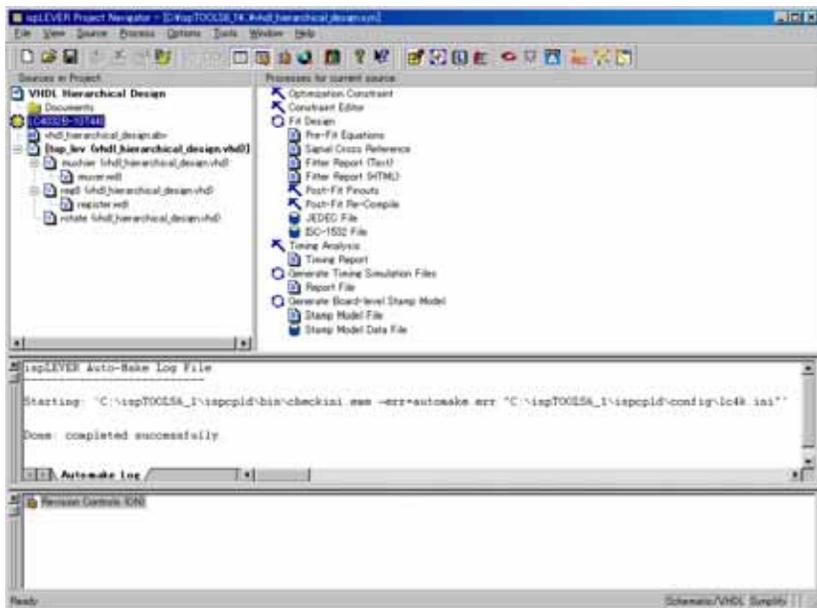
2. Lattice isp LEVER Design Flow



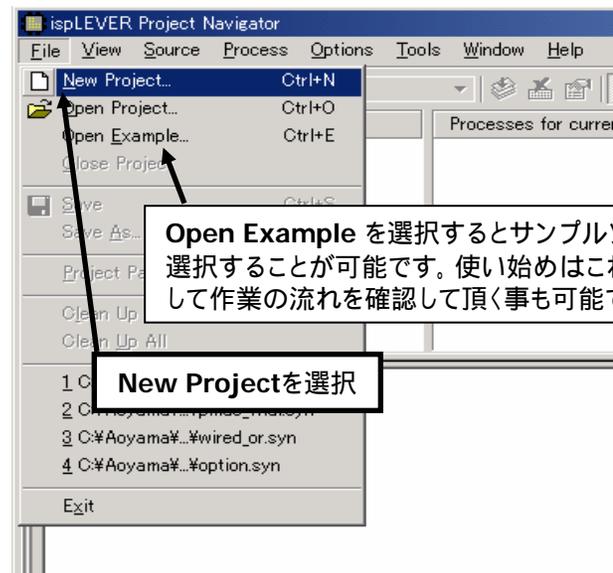
3. ispLEVER起動

ispLEVERを起動させます。

1. スタートメニューより、[プログラム] [Lattice Semiconductor] [ispLEVER]を実行します。

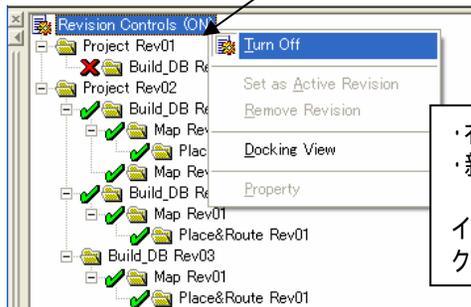


2. [File] [New Project]を選択して、新規プロジェクトを作成します。



新規以外のプロジェクトの場合は、Open Project を選択してください。

Revision管理Tree



・右クリックでRevision管理をOn,Offできます。
・新しくコンパイルする度にRevisionが変わります
以前のRevisionに戻す際には該当するRevisionをハイライトさせて右クリックし、Set as Active Revisionをクリックしてください。

3.1 新規プロジェクトの作成

新規プロジェクトの作成を行う前に、“新しいフォルダ”を必ず作成してください！

1. 任意のドライブを表示させ、フォルダの新規作成アイコンをクリックします。

sampleと命名

保存するロケーションを選択します。

Schematic/VHDLを選択して下さい。

論理合成ツールを選択します。

次へをクリック！！

プロジェクト名には「日本語」を使用しないでください。右図のようにプロジェクト名を“sample.syn”として、プロジェクトのタイプは“Schematic/VHDL”を選択してください(プロジェクトタイプはエントリー方法に応じてお選びください)
プロジェクトフォルダの設定にはSchematicがございませがFPGAデバイスはSchematicをサポートしておりませ
プロジェクトフォルダを配置する場所としてマイドキュメントやデスクトップを選択しないで下さい(プロジェクトが保存されているパスに日本語や空白スペースが入っていると問題になることがあります)。

Design Entry Typeについて	
Schematic/ABEL	回路図及びABELを用いたデザイン
Schematic/VHDL	回路図及びVHDLを用いたデザイン
VHDL	VHDLを用いたデザイン
Schematic/Verilog HDL	回路図及びVerilog HDLを用いたデザイン
Verilog HDL	Verilog HDLを用いたデザイン
EDIF	EDIFを用いたデザイン
GDF	GDFを用いたデザイン

3.2 ターゲットデバイスの選択

新規プロジェクトで用いるターゲットデバイスを選択します。2.元々あるソースをインポートします。
(必要な場合を除く)

1. デバイスの情報を選択します。

このスクリーンショットは、Latticeのソフトウェアでターゲットデバイスを選択するための「Select Device」ダイアログボックスを示しています。以下の項目が選択されています：

- デバイスファミリーの選択:** ispMACH 4000
- デバイスの選択:** LC4032B
- パッケージタイプの選択:** 44TQFP
- 温度グレードの選択:** Industrial
- スピードグレードの選択:** -10

その他の表示されている情報には、I/O pins: 30, Dedicated input: 2, Output enable: -, Icc: 11.3 mA などが含まれます。

Add Sourceをクリック

このスクリーンショットは、「Add Source」ダイアログボックスを示しています。以下の操作が行われます：

- ソースを選択:** 既存のソースファイルを選択する。
- 次へをクリック:** 次のステップに進む。

3. 完了をクリックします。

このスクリーンショットは、「Project Wizard - Project Information」ダイアログボックスを示しています。以下の操作が行われます：

- 完了をクリック:** プロジェクトの生成を完了させる。
- プロジェクト情報が表示:** 生成されるプロジェクトの仕様を確認する。

表示されているプロジェクト情報は以下の通りです：

```

Project Name: Untitled
Project Title: Untitled
Project Location: c:\v_desin
Project Type: Schematic/VHDL
Device: LC4032B-10T44I
Synthesis: Synplify
    
```

3.3 デザインエントリー

新規プロジェクトのエントリーをします(今回はVHDLで行ないます)

1. Project Navigatorから[Source] [New]を選択します。

新規作成時は[New]を選択して下さい。

既にソースが作成済みの場合は、[Import...]を選択してください。

プロジェクトからソースを外したいときには[Remove]を選択してください。
この作業ではフォルダ内からファイルは削除されません。

2. VHDL Moduleを選択し、OKをクリックします。

VHDL Moduleを選択

3. 表示されたダイアログボックスの各項目を入力してOKをクリックします。

入力参考例

File Name : counter
Entity : counter
Architecture : behavioral

* Importの場合は使用したいファイルを選択します。

- ABEL Test Vectors
 - ABEL HDL Module
 - User Document
 - Schematic
 - Waveform Stimulus
 - Verilog Module
 - Verilog Test Fixture
 - VHDL Module
 - VHDL Test Bench
 - ABEL HDL のシミュレーション記述
 - ABEL HDL のソース
 - コメント等のオプション機能
 - 回路図エディタ
 - 波形シミュレーション
 - Verilog HDL のソース
 - Verilog HDL のシミュレーション記述
 - VHDL のソース
 - VHDL のシミュレーション記述
- 選択できるファイルタイプは、プロジェクトタイプにより変わります。

ソースファイルを作成します(簡単な4ビットのアップ/ダウンカウンタを作成しましょう)

1. 前ページで入力した項目が記載されたテキストエディタが開きます。
入力するサンプルソースは次ページ以降を参照してください。

Saveアイコンを選択して保存してください

もしくは

【File】【Save】を選択して保存してください

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity counter is
end;

architecture behavioral of counter is
begin
end behavioral;
    
```

3.4 サンプルソース(4bitアップダウンカウンタ)

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counter is
port(
    clk          :in std_logic;           --"port"でピン定義をします
    rst          :in std_logic;
    c_en        :in std_logic;
    up_dw       :in std_logic;
    count       :out std_logic_vector(3 downto 0));

end;

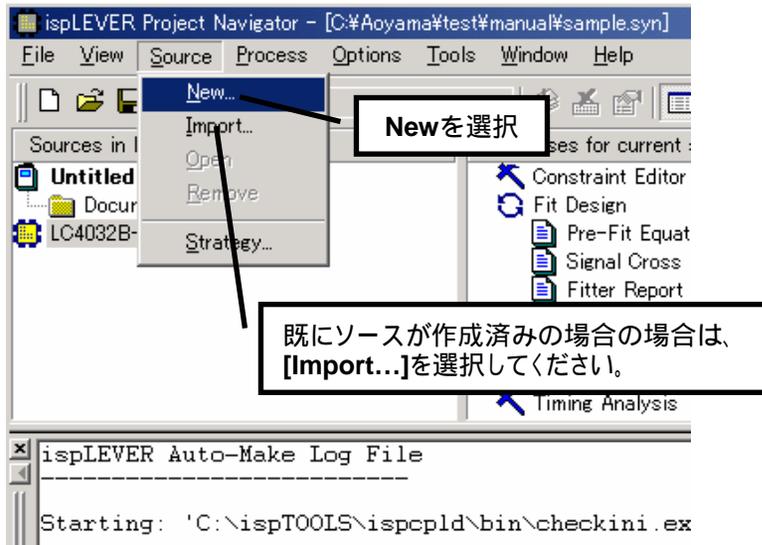
architecture behavioral of counter is
    signal n_count:std_logic_vector(3 downto 0);
begin
    process(clk,rst)
    begin
        if(rst='1')then                    --リセット条件(High active)
            n_count <= "0000";
        elsif(clk 'event and clk = '1')then --クロックの立ち上がり動作
            if (c_en = '1') then          --クロックイネーブルで動作
                if(up_dw = '1')then       --アップ/ダウン信号で動作
                    n_count <= n_count + 1; --カウントアップ
                elsif(up_dw = '0')then    --カウントダウン
                    n_count <= n_count - 1;
                end if;
            end if;
        end if;
    end process;
    count <= n_count;                    --ピンに出力します
end;

```

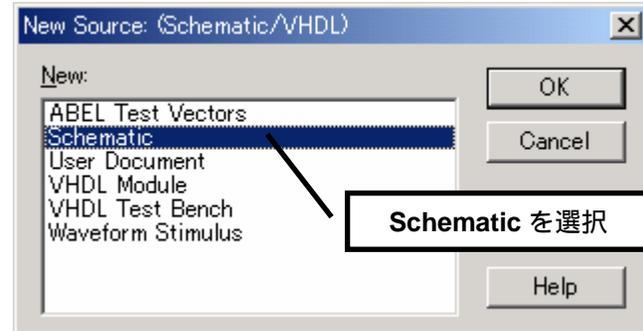
入力が終了しましたら前ページを参考に保存して下さい。

前のページで作成したVHDLソースをトップの階層で回路図としてシンボル登録します

1. Project Navigatorから[Source] [New]を選択します。

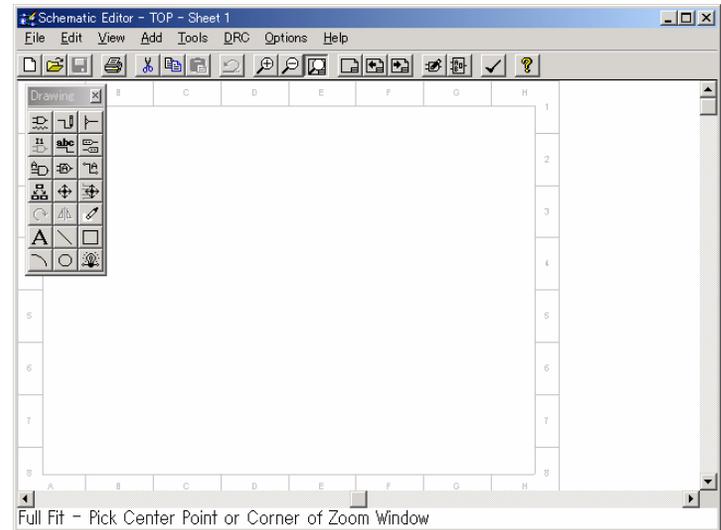


2. Schematicを選択し、OKをクリックします。



4. 以下のような回路図エディタが開きます。

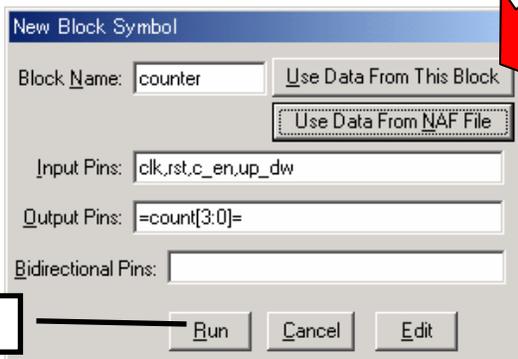
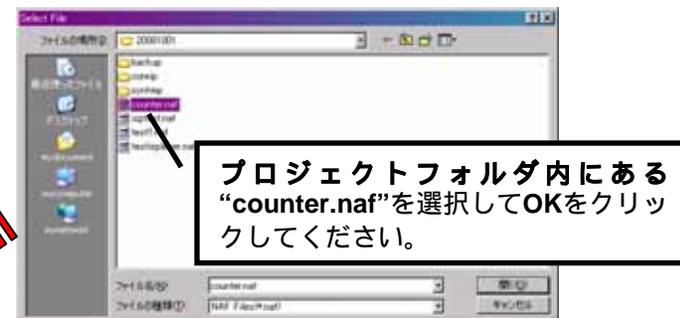
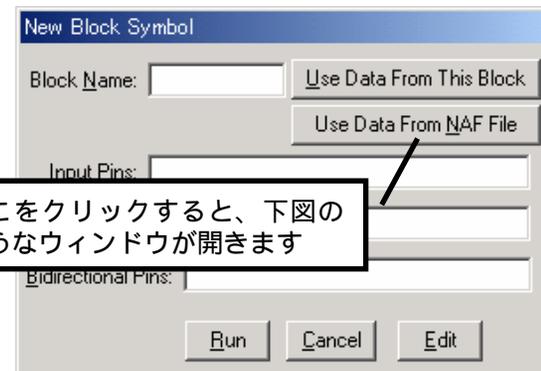
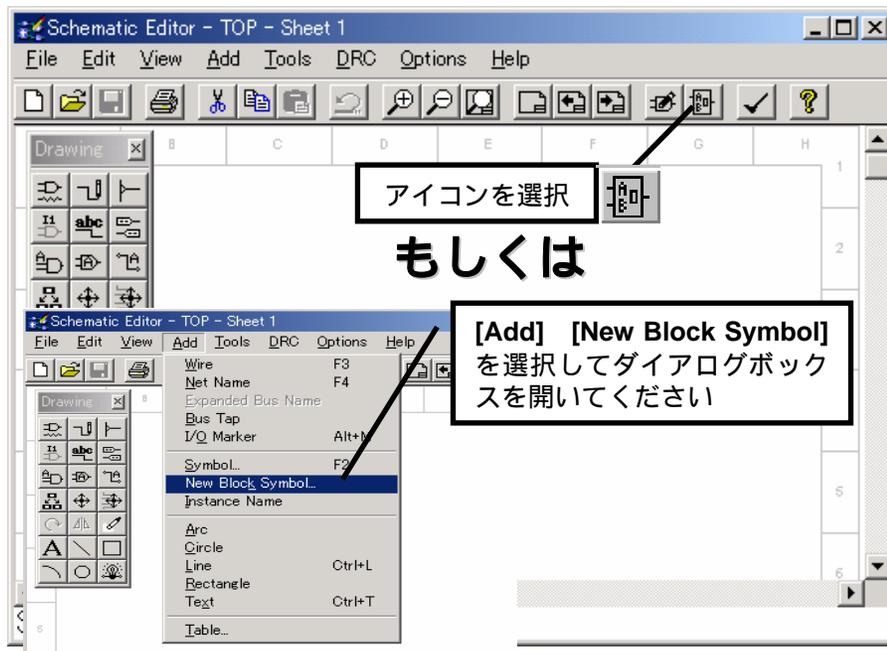
3. 表示されたダイアログボックスのファイル名を入力してOKをクリックします。



3.5 回路図の作成

実際にシンボルブロックの登録作業を行ないます

1. メニューバーから[Add] [New Block Symbol...]を選択します。
2. 以下のようなウィンドウが開きます。



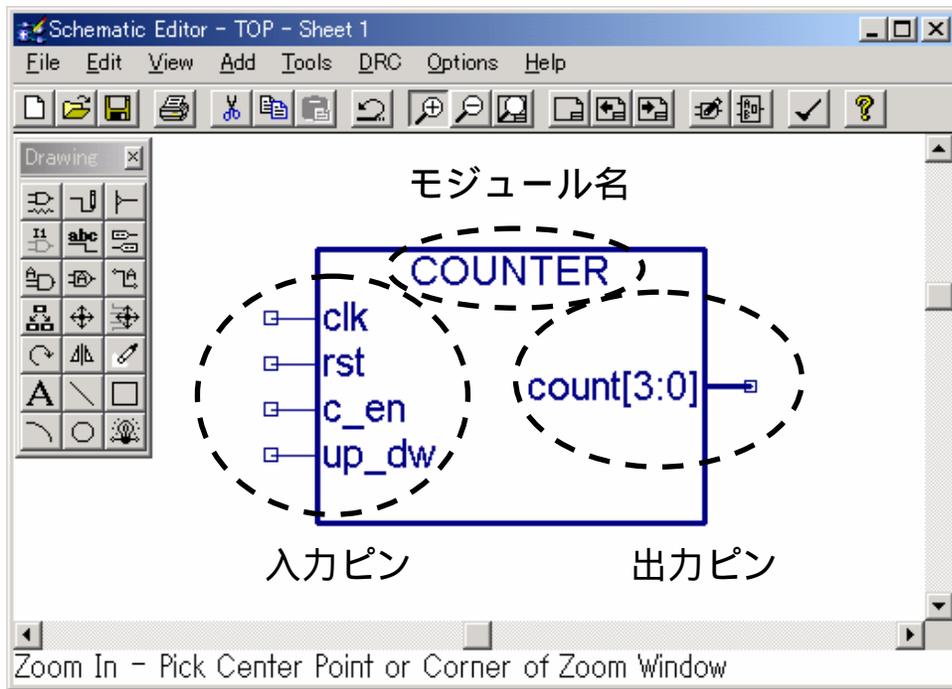
3. VHDLソースで設計したモジュールの入出力部分が各ボックスにインポートされてきます。このダイアログはユーザー側で入力することも可能です。

“*.naf” は、モジュールのI/F情報を含んだファイルです。

3.5 回路図の作成

回路図エディタを使って回路設計を行ないます

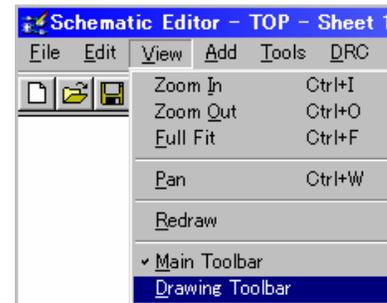
1. 前ページの方法で作ったシンボルがエディタ上に表示されます。



2. エディタでの作業に用いるツールバーの説明を行ないます。ここでは今回の設計に用いる操作のみを書きますので、詳細については別途マニュアルを参照してください。



今回は赤色で囲った部分のみの説明をします



ツールバーが表示されていない場合、メニューバーより[View] [Drawing Toolbar]を選択してください

Generate Schematic Symbolを行うことでもシンボルを自動的に生成することも可能です。

3.5 ツールバーの説明

ツールバーの機能説明をします

1. ツールバーアイコンの各機能は以下の通りになります。

[Add Symbol] : ゲートやレジスタをエディタ上に配置する際に選択します

このようなウィンドウが開きます。ispM4Aシリーズの設計の場合には、“Generic”ライブラリ及び“Vantis”マクロを選択して設計をしてください

[Add Net Name] : 信号名(I/Oピン)及びネット名(内部ノード名)を付ける際に選択します

[Delete] : 消しゴムツールです

[Add Wire] : 配線する際に選択します

[Add Bus Tap] : バス信号の配線を行う際に選択します

[Add I/O Marker] : 信号のモード(方向)を設定します

[Add Symbol Attribute] : シンボルにアトリビュートを設定します。回路図上でピン固定をする際に有効です

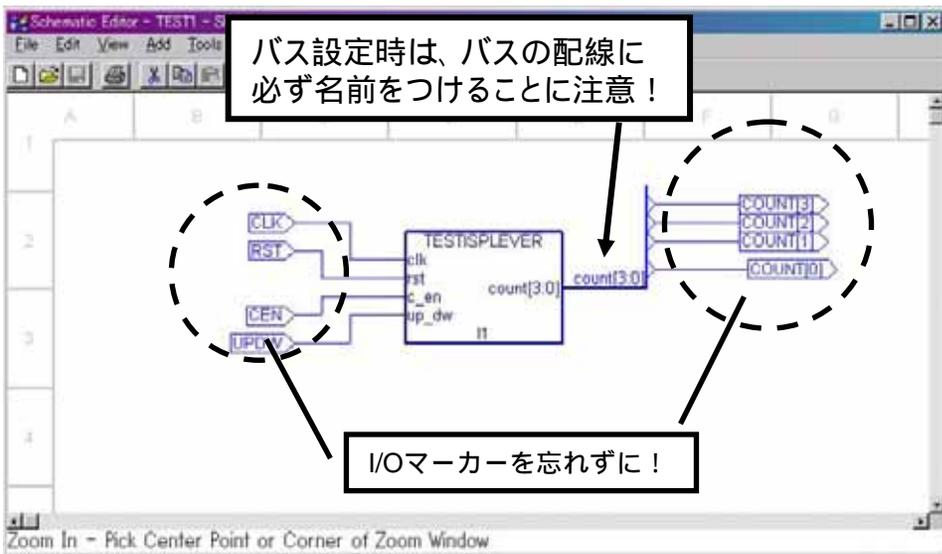
作成したシンボルはライブラリ [Local]の中に保存されています

上図“Symbol Attribute Editor”が開きます。このウィンドウ上で左図のようなI/Oパッドのピン固定を行ないます。

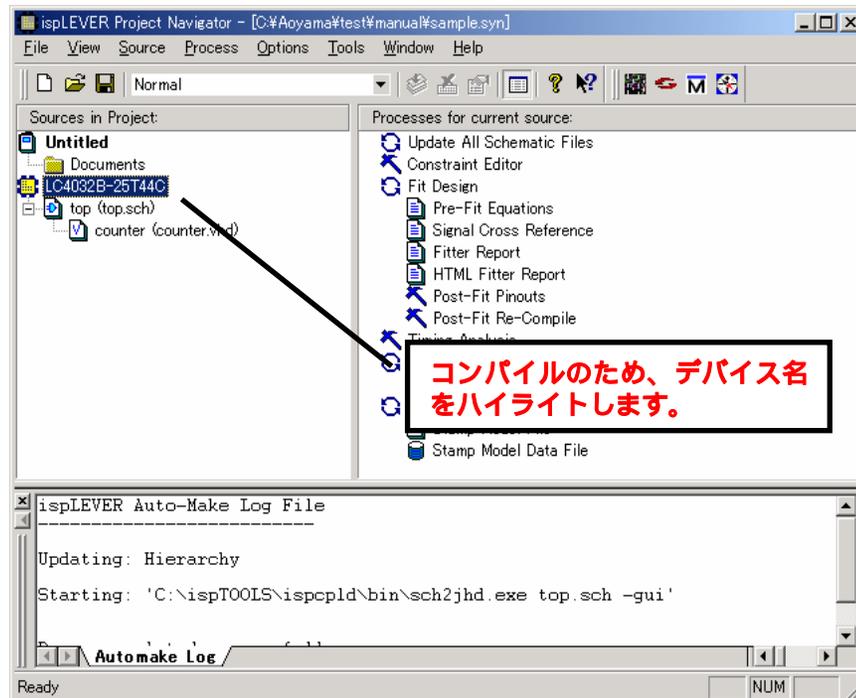
3.5 回路図の配線/信号名をつける(回路図設計の完了)

回路図作成作業を完了します

1. 前ページで説明したツールバー機能を使って以下のように設計を完成させます。
2. 作成した回路図を保存し、エディタを閉じてください。



回路図のみの設計を行ないたい場合には、ツールバーより必要なゲートやレジスタを配置・配線して設計を完了してください。



(回路図エディタのメニューアイコン)

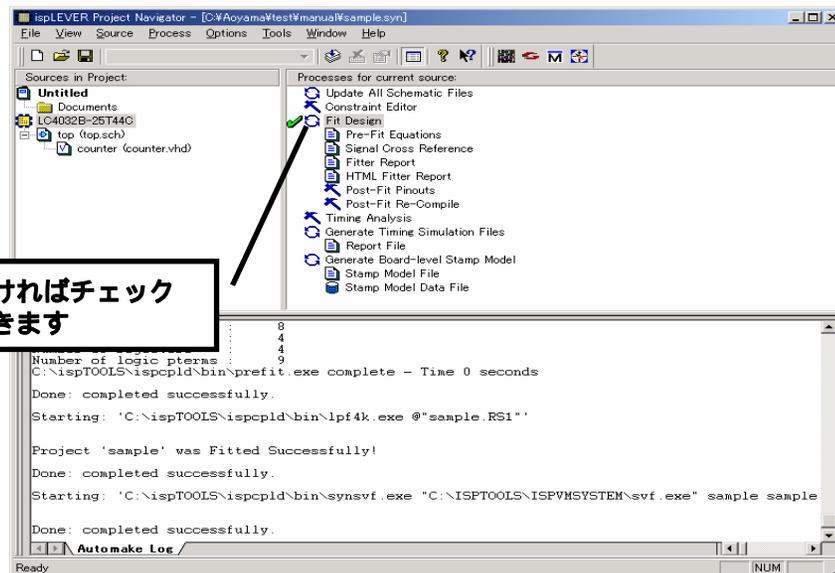
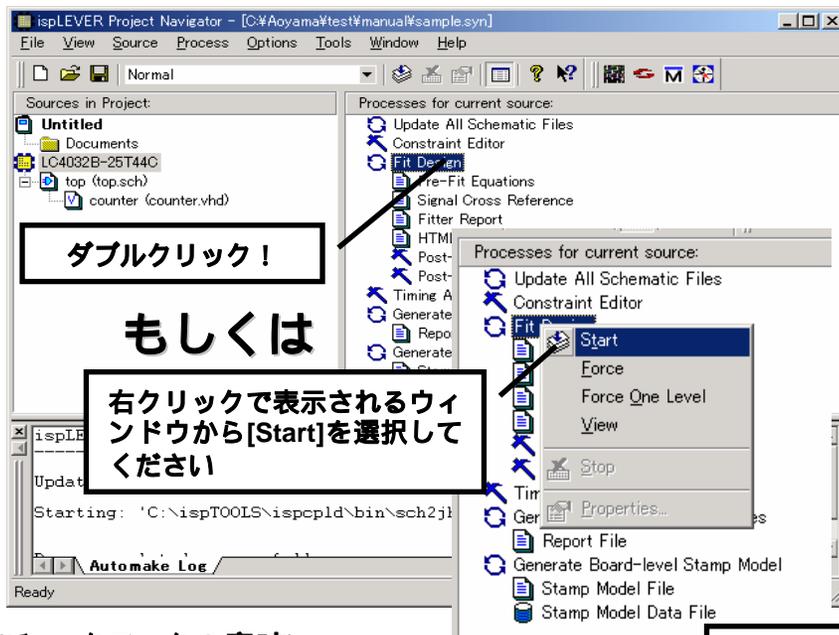


4. コンパイル作業の開始

ソースファイルをコンパイルします(ソースのチェックを行ないます)

1. Fit Designをダブルクリック。もしくはハイライト表示して、Startボタンをクリックしてください。

2. コンパイルが始まります。



(チェックマークの意味)

-  コンパイル成功です(エラーはありません)
-  コンパイル成功です(ワーニングがあります)
-  コンパイルエラーです(ログを確認して下さい)

エラーがなければチェックマークが付きます

記述したソースに文法的な誤りがあるかどうかをチェックします。エラーが発生した場合はもう一度記述等を確認してください。文法エラーの場合は、ログのエラー勧告箇所をダブルクリックするとソース上のエラー個所にジャンプします。

『Fit Design』により論理合成からFitting(書き込み用ファイル生成)まで完了します。

4.1 論理合成ツールについて

論理合成を行います。

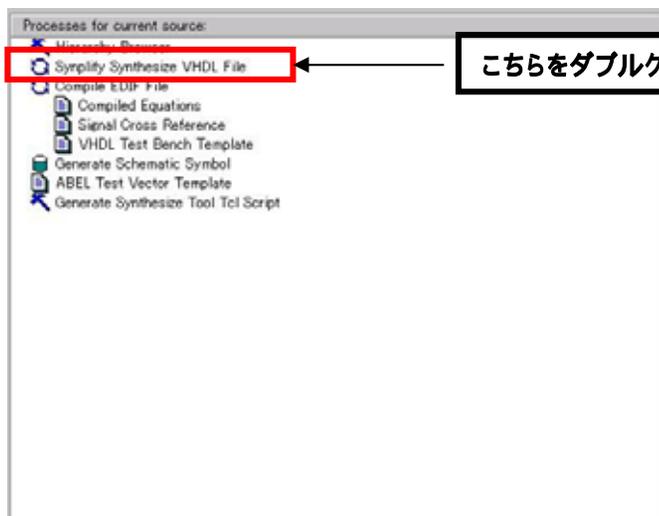
Lattice社のispLEVERでは、Synplicity社の論理合成ツールSynplify proがバンドルされています。

1. Project NavigatorのProcesses for current source画面の『Synplify Synthesize VHDL File』で論理合成を行えます。

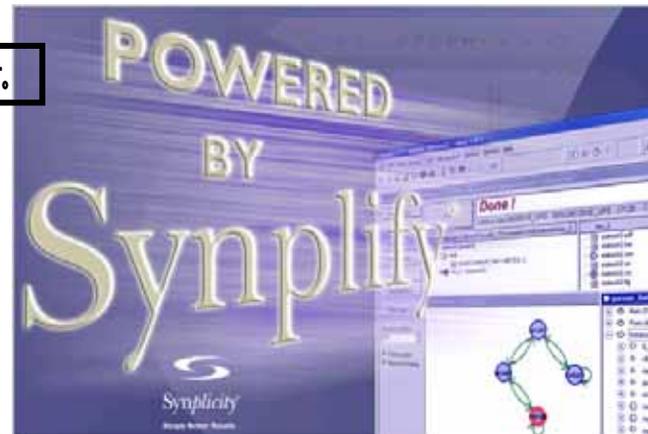
ダブルクリックした後、バックグラウンドでSynplifyが起動し論理合成を行います。

チェックがついたら論理合成は完了です。

記述ミスの場合はlogファイル内に赤字で表示されているエラーメッセージをダブルクリックすることで、ソース内の該当箇所付近にとぶことができます。

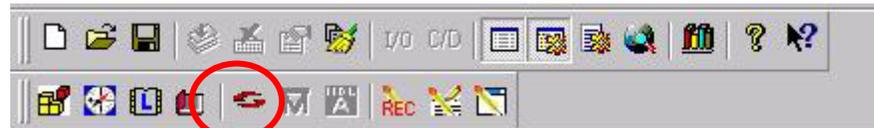


こちらをダブルクリックします。



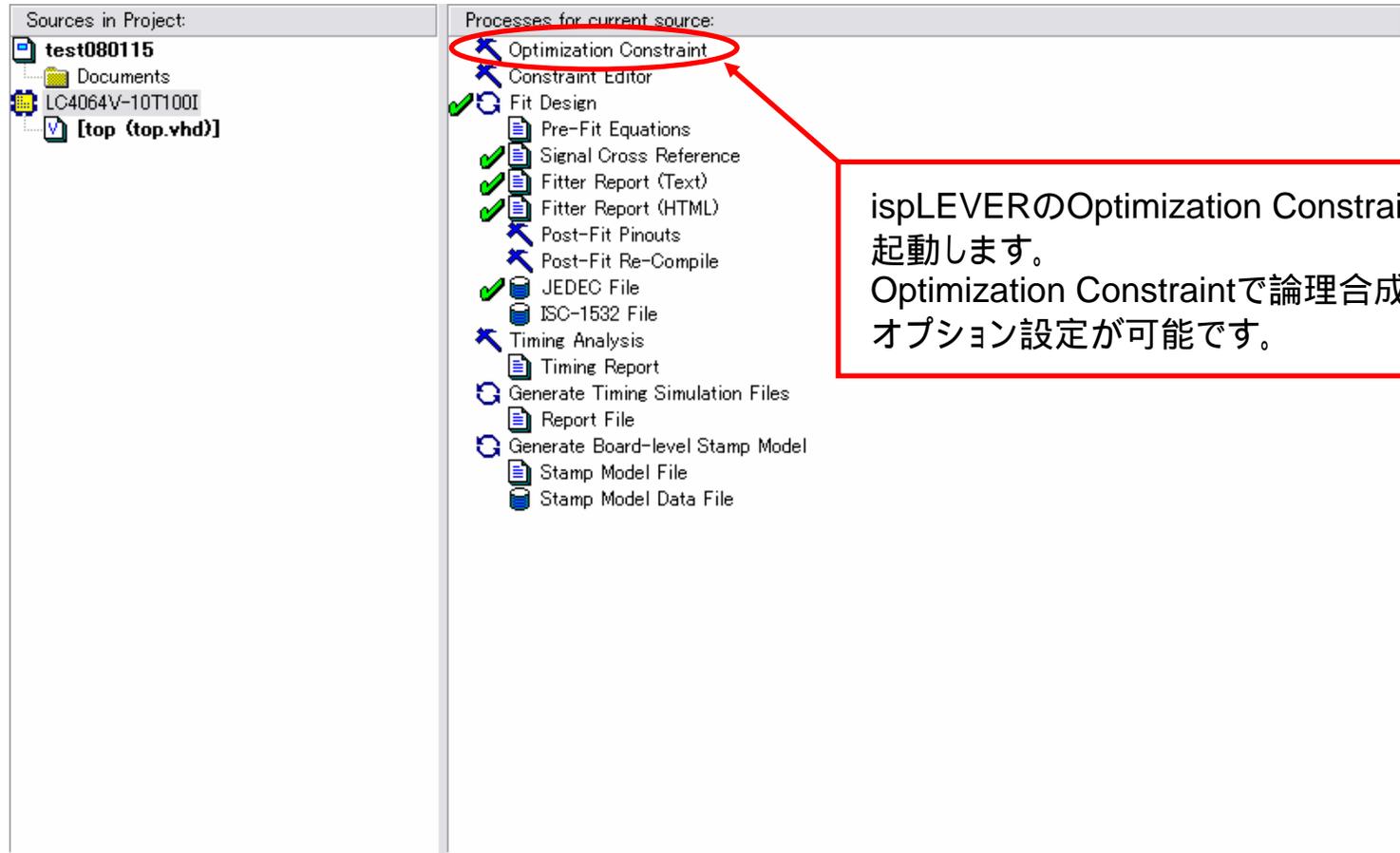
論理合成ツールはLattice社OEM版となっており、
選択したデバイスのアーキテクチャに最適な論理合成を提供致します。
論理合成ツールは、単体での起動も可能です。

-Project Navigator画面のTools Synplify



4.2. 論理合成時のオプション設定について

論理合成する前にオプション設定を行うことができます。



ispLEVERのOptimization Constraintを
起動します。
Optimization Constraintで論理合成時の
オプション設定が可能です。

4.2. 論理合成時のオプション設定について

各項目で以下の設定を行うことができます。

	Constraint Name	Constraint Value
1	Max_area	32
2	Logic_reduction	Yes
3	Dt_synthesis	Yes
4	Xor_synthesis	Yes
5	Node_collapse	Yes
6	Nodes_collapsing_mode	Fmax
7	Fmax_logic_level	1
8	Max_pterm_collapse	16
9	Max_fanin	24
10	Max_pterm_split	80
11	Max_fanin_limit	28
12	Max_pterm_limit	80
13	Clock_enable_optimization	Keep_all
14	Logic_optimization_effort	2

Max_area

PT(プロダクトターム)の接続本数に制約をかけます。数値を小さくするとPTの使用率が削減されますが、Logic LEVELが増加しFmaxの特性が悪くなります。

Logic_reduction (Yes / No)

冗長回路を削減し、インプリメントします。使用リソースの削減に効果があります。

Dt_synthesis (Yes / No)

D-FF、T-FFを使用し、積数項が最小になるようにインプリメントします。使用リソースの削減に効果があります。"No"に設定した場合、D-FFのみの使用となります。

Xor_synthesis (Yes / No)

マクロセル内のXORリソースを優先的に使用します。"No"の場合、積項でXORを生成します。

Nodes_conllapse (Yes / No)

レジスタと出力ピン間の無駄なノードを削減し、Fmaxの向上、ロジックリソースの削減を行います。Node_collapsing_modeの設定によって論理合成の結果が変わります。

Nodes_collapsing_mode (Speed / Area / Fmax)

Speed・・・設定された積項数の制限値までなら内部ノードを削減します。Tpd、Tco、Fmaxが向上される傾向があります。

Area・・・ロジックリソース使用効率が向上されるように論理合成を行います。リソース不足の際に効果がありますが、スピードの特性が悪くなる傾向があります。

Fmax・・・で設定されたロジックレベルに応じて論理合成を行います。

4.2. 論理合成時のオプション設定について

各項目で以下の設定を行うことができます。

	Constraint Name	Constraint Value
1	Max_area	32
2	Logic_reduction	Yes
3	Dt_synthesis	Yes
4	Xor_synthesis	Yes
5	Node_collapse	Yes
6	Nodes collapsing mode	Fmax
7	Fmax_logic_level	1
8	Max_pterm_collapse	16
9	Max_fanin	24
10	Max_pterm_split	80
11	Max_fanin_limit	28
12	Max_pterm_limit	80
13	Clock_enable_optimization	Keep_all
14	Logic_optimization_effort	2

Fmax_logic_level

設定されたロジックレベルで論理合成を行います。ロジックレベルが大きくなるとスピードの特性が悪くなります。(Nodes_collapsing_modeをFmaxに設定した場合のみ有効)

Max_pterm_collapse

PTの接続本数制限を設定します。本数を増やすとマクロセルの使用率を削減し、スピードを向上する傾向があります。(Nodes_collapsing_modeをSpeedかAreaに設定した場合のみ有効)

Max_fanin

ノードの入力本数制限を設定します。本数を減らすとマクロセルの使用率、スピード特性が共に悪くなる傾向があります。(Nodes_collapsing_modeをSpeedかAreaに設定した場合のみ有効)

Max_pterm_split

PTの接続本数制限を設定します。制限本数を超えたPTに関しては別のマクロセルに分割されます。(Nodes_collapsing_modeをSpeedかAreaに設定した場合のみ有効)

Max_fanin_limit

ノードの入力本数制限を設定します。本数を減らすとマクロセルの使用率、スピード特性が共に悪くなる傾向があります。(Nodes_collapsing_modeをFmaxに設定した場合のみ有効)

Max_pterm_limit

PTの接続本数制限を設定します。制限本数を超えたPTに関しては別のマクロセルに分割されます。(Nodes_collapsing_modeをFmaxに設定した場合のみ有効)

4.2. 論理合成時のオプション設定について

各項目で以下の設定を行うことができます。

	Constraint Name	Constraint Value
1	Max_area	32
2	Logic_reduction	Yes
3	Dt_synthesis	Yes
4	Xor_synthesis	Yes
5	Node_collapse	Yes
6	Nodes_collapsing_mode	Fmax
7	Fmax_logic_level	1
8	Max_pterm_collapse	16
9	Max_fanin	24
10	Max_pterm_split	80
11	Max_fanin_limit	28
12	Max_pterm_limit	80
13	Clock_enable_optimization	Keep_all
14	Logic_optimization_effort	2

Clock_enable_optimizations(Warp_all / Warp_all_opt / Keep_all / Auto)

Warp_all ……CLK_ENをD入力ポートで生成します。

Warp_all_opt ……CLK_ENをD入力ポートに生成、もしくはD入力ポートに最適化します。

Keep_all ……CLK_ENを保持します。

Auto ……CLK_ENの保持、最適化をツールが決定します。

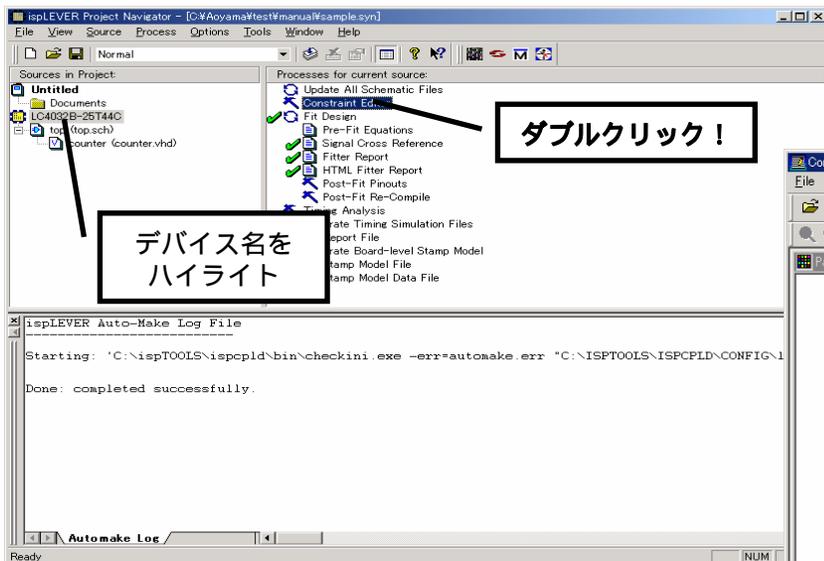
Logic_optimization_effort

論理合成のレベルを設定します。数字を大きくするほど、内部ノードの削減が見込めます。

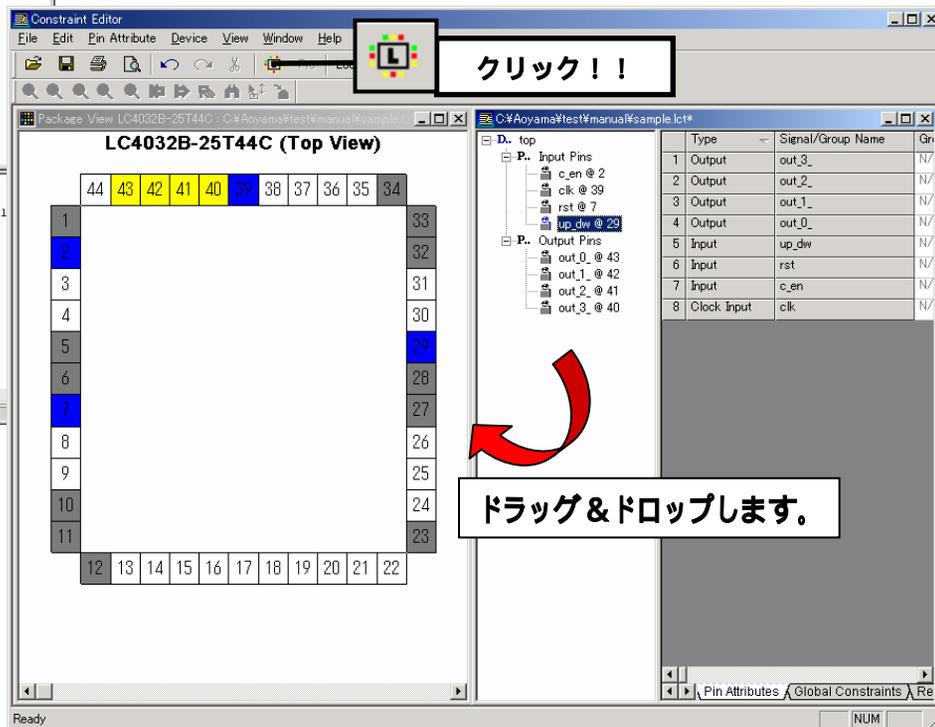
5. ピン固定 (Package Viewを用いた設定)

コンストレイント・エディタを使ってピン固定を行ないます

1. デバイス名をハイライトして、画面右のConstraint Editorをダブルクリックしてください。



2. 下のような画面が開いたら、アイコンメニューから“Package View”を選択してください。



ピン配置の方法

1. 右画面に信号名の一覧が表示されます。
2. 配置したい信号をドラッグして、左画面の配置したい場所でドロップします。
3. 配置が完了すると、左画面のピン位置に色がつきます。



入力ピン

(黄色)



出力ピン

(青色)

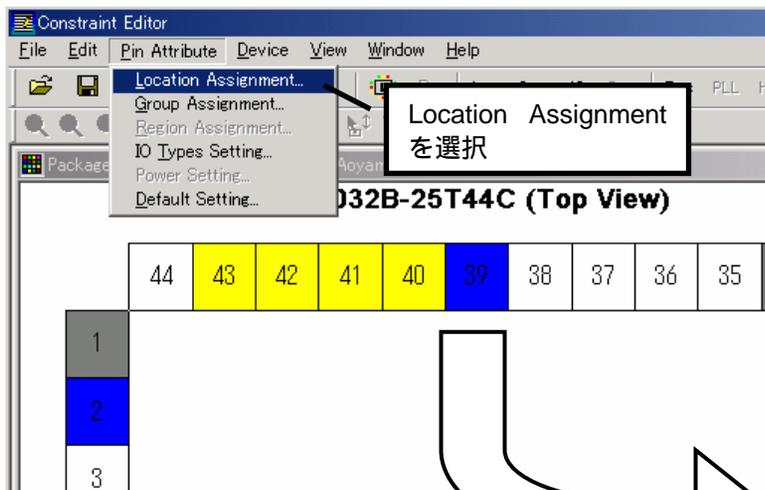


双方向ピン

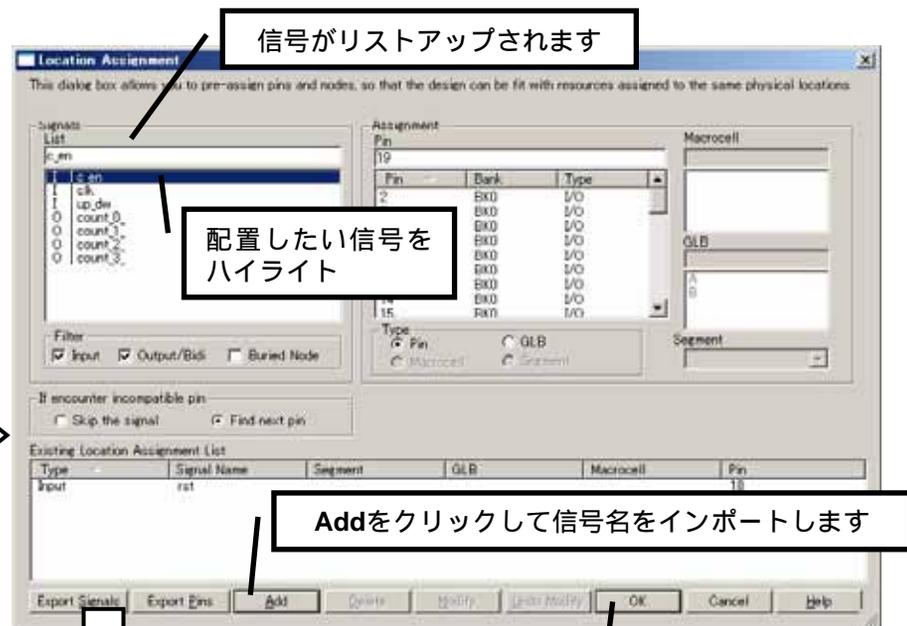
(ピンク)

5.1 ピン固定 (Location Assignmentを用いた設定)

メニューリストからLocation Assignmentを選択します



1. 以下のようなウィンドウが開きます。



信号がリストアップされます

配置したい信号を
ハイライト

Addをクリックして信号名をインポートします

良ければ“OK”をクリック

ピン配置の修正・削除

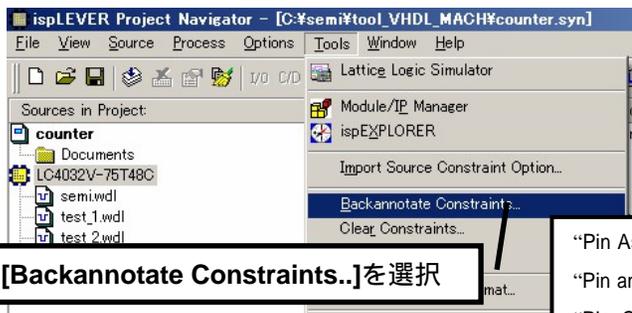
1. 右図の最下部にあるウィンドウに表示された信号名をハイライトした状態で、[Delete]を押せば信号を削除できます。
2. 修正したい場合は、ハイライトした状態で[Modify]を押し、修正後のピンを決定した後で[Update]を押してください。



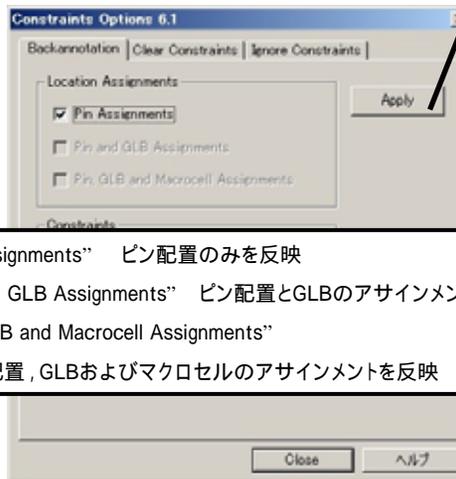
5.2 ピン固定 (Compile後できる便利な設定)

一度ピンフリーでコンパイルした結果を反映することが可能です

1. コンパイルが終了した段階で、[Tools] [Backannotate Project Assignment]を選択して下さい。
2. Backannotationタグにある[Pin Assignments]にチェックをして[Apply]を押します。



[Backannotate Constraints...]を選択



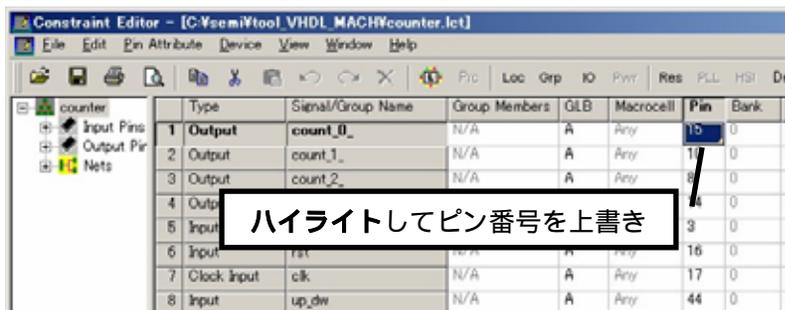
必ず[Apply]を押して下さい。

“Pin Assignments” ピン配置のみを反映
 “Pin and GLB Assignments” ピン配置とGLBのアサインメントを反映
 “Pin, GLB and Macrocell Assignments”
 ピン配置, GLBおよびマクロセルのアサインメントを反映



「はい(Y)」を押して下さい。

3. Constraint Editorを開き(起動方法はP.18参照)、ピン番号が書かれている部分をハイライトもしくはダブルクリックして、固定したい番号を上書きします。



ハイライトしてピン番号を上書き

4. 割付したくないピンがある場合は、右クリックで[Clear Selected]を選択して下さい。

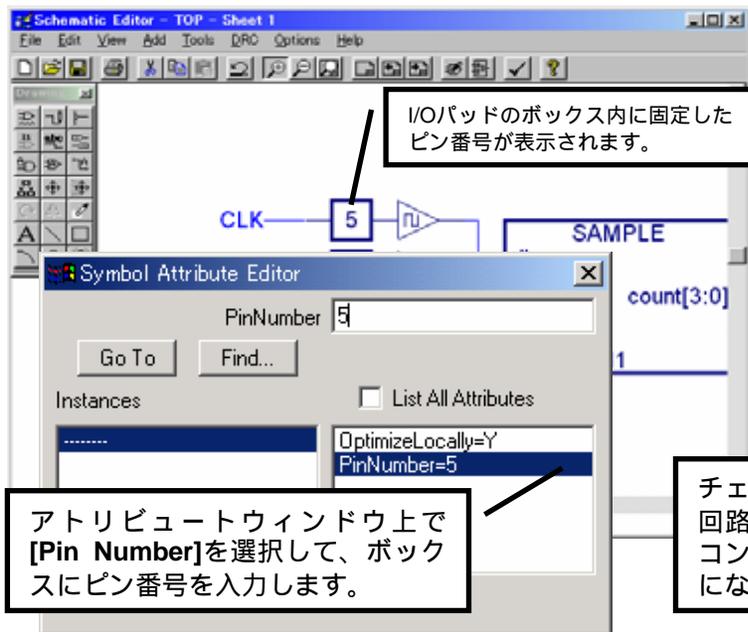
Pin	Bank	IO Types	Slew
15	0	LVTTTL	FAS

10	Clear Selected
8	Delete Row(s)
14	Location Assignment...
3	Pin Number...

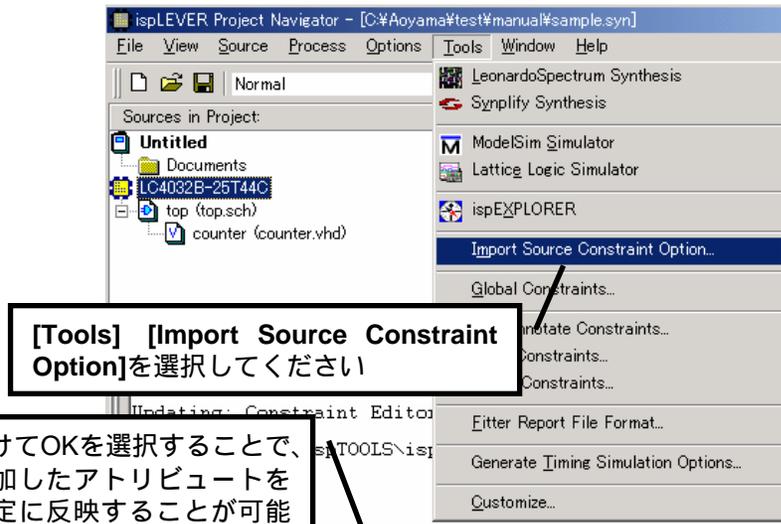
5.3 ピン固定 (回路図上での設定)

回路図エディタ上で実際にピンの配置を行います

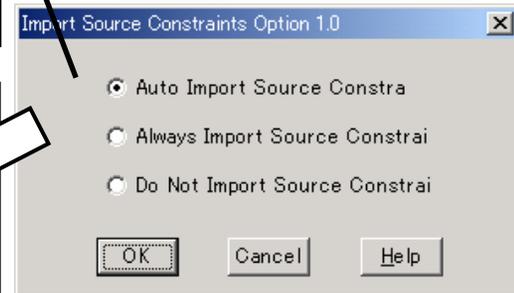
1. I/Oパッドを用いてピン固定を行ないます。ツールバーより **[Edit Symbol Attribute]** を選択して、ピンを固定したいI/Oパッドを選択してください。 **[Pin Number]** という項目を選択して固定したいピン番号を入力しウィンドウを閉じます。



2. 回路図エディタ上で固定したピンを実際のコンパイルに反映するためには、 **Project Navigator** 上で以下のような設定を行なってください。



チェックを付けてOKを選択することで、回路図上で付加したアトリビュートをコンパイル設定に反映することが可能になります。



- “Auto Import Source Constraints”
確認のためのダイアログが表示されます。(デフォルト設定)
- “Always Import Source Constraints”
ソース式のコンストレイントをインポートします。 Constraint Editorの設定は反映されません。
- “Do Not Import Source Constraints”
コンストレイント・エディタでの設定インポート。 ソース式のコンストレイントは反映されません。

IO周りの設定を行います。

まず、ispLEVERのConstraint Editorを起動します。

以下の画面が表示されます。

Constraint Name	Constraint Value
1 Pull	Up
2 Security	Off
3 Usercode	
4 Usercode_format	Hex
5 Balanced_partitioning	Yes
6 Zero_hold_time	No
7 Auto_buffering_for_high_glb_fanin	Off
8 Auto_buffering_for_low_bonded_io	Off
9 Spread_placement	Yes
10 Max_macrocell_percent	100
11 Max_glb_input_percent	100
12 Fitter_effort_level	Low
13 User_max_glb_fanin	36
14 Adjust_input_assignments	Off
15 Svf_erase_program_verify	Off
16 Svf_erase_program_verify_secure	On
17 Svf_verify_only	Off

次に、Global Constraintsを選択します。

内部pull-up等の設定はCPLDの場合、ピン毎ではなく、一括設定になることに注意して下さい。
ただし、MACH4000ZEシリーズのみ、ピン毎に設定することが可能です。

各項目で以下の設定を行うことができます。

	Constraint Name	Constraint Value
1	Pull	Up
2	Security	Off
3	Usercode	
4	Usercode_format	Hex
5	Balanced_partitioning	Yes
6	Zero_hold_time	No
7	Auto_buffering_for_high_glb_fanin	On
8	Auto_buffering_for_low_bonded_io	Off
9	Spread_placement	Yes
10	Max_macrocell_percent	100
11	Max_glb_input_percent	100
12	Fitter_effort_level	Low
13	User_max_glb_fanin	36
14	Adjust_input_assignments	Off
15	Svf_erase_program_verify	Off
16	Svf_erase_program_verify_secure	Off
17	Svf_verify_only	Off

Pull (UP / DOWN / HOLD / OFF)

Pull UP … I/Oピンを内部Pull UPに設定します。
 Pull DOWN … I/Oピンを内部Pull DOWNに設定します。
 Bus HOLD … I/Oピンを内部HOLDに設定します。最後の値が保持されます。
 OFF … OFFに設定します。

MACH4000ZEでは『Global Constraints』ではPull設定項目は表示されません。
 『Pin Attributes』シート内よりピン毎に設定を行います。
 (資料内P.30～P.31『ピン属性の設定』をご覧ください。)

Security (ON / OFF)

書き込みデータの読み出しが出来ないようにセキュリティの設定をします。

Usercode

ユーザーコードをJEDECファイルに追加します。

Usercode_format (Hex / Bin / ASCII / Checksum)

ユーザーコードで使用するコード形式を設定します。

Balanced_partitioning (Yes / No)

デバイス内のパーティションを設定します。有効の場合、設計回路の論理合成がデバイスの一箇所に固まる事なく、効率のよい論理合成を行います。デバイスの使用率が上がるとパーティションに納まりきらずワーニングで強制的にNoに変更されます。

Zero_hold_time (Yes / No)

入力レジスタにZero hold timeヒューズを設定します。入力レジスタのHoldが0になります。

各項目で以下の設定を行うことができます。

	Constraint Name	Constraint Value
1	Pull	Up
2	Security	Off
3	Usercode	
4	Usercode_format	Hex
5	Balanced_partitioning	Yes
6	Zero_hold_time	No
7	Auto_buffering_for_high_glb_fanin	Off
8	Auto_buffering_for_low_bonded_io	Off
9	Spread_placement	Yes
10	Max_macrocell_percent	100
11	Max_glb_input_percent	100
12	Fitter_effort_level	Low
13	User_max_glb_fanin	36
14	Adjust_input_assignments	Off
15	Svf_erase_program_verify	Off
16	Svf_erase_program_verify_secure	Off
17	Svf_verify_only	Off

Auto_buffering_for_high_glb_fanin (ON / OFF)

自動バッファ追加の設定を行います。AND入力本数が制限値を超えた際 (GLBオーバー)にONに設定するとバッファを追加し、AND入力 がGLBに分散します。

Auto_buffering_for_low_bonded_io (ON / OFF)

自動バッファ追加の設定を行います。入力レジスタが一箇所のGLBに固まっている際にONに設定するとバッファを追加し、入力レジスタを別のGLBに分散させます。

Spread_placement (Yes / No)

論理合成時のデバイスへのリソースの分散を設定します。Yesの場合、ロジックはデバイスに均等になるよう分散されます。またデバイスのFitting率が向上します。Noの場合、ロジックは一箇所に固まり構成されるため、スピードが向上します。

Max_macrocell_percent

論理合成時に使用するマクロセルの許容範囲を設定します。設定を超えた場合はエラーが出ます。

Max_glb_input_percent

論理合成時に使用するGLBへの入力本数の許容範囲を設定します。

Fitter_effort_level (Low / Medium / High)

論理合成のレベルを設定します。Highに設定すると論理合成結果が向上する事もあります。

各項目で以下の設定を行うことができます。

	Constraint Name	Constraint Value
1	Pull	Up
2	Security	Off
3	Usercode	
4	Usercode_format	Hex
5	Balanced_partitioning	Yes
6	Zero_hold_time	No
7	Auto_buffering_for_high_glb_fanin	Off
8	Auto_buffering_for_low_bonded_io	Off
9	Spread_placement	Yes
10	Max_macrocell_percent	100
11	Max_glb_input_percent	100
12	Fitter_effort_level	Low
13	User_max_glb_fanin	36
14	Adjust_input_assignments	Off
15	Svf_erase_program_verify	Off
16	Svf_erase_program_verify_secure	Off
17	Svf_verify_only	Off

User_max_glb_fanin

GLBに出来る信号の最大数を設定します。

Adjust_input_assignments (ON / OFF)

GLBへの信号の分散を設定します。GLBへの本数が多い場合、ONに設定すると固定されていない信号が別のGLBに分散されます。

Svf_erase_program_verify (ON / OFF)

SVFファイルにErase、Program、Verifyの設定を行います。JEDECファイル生成時に生成されるSVFファイルに書き込みオペレーションを指定します。

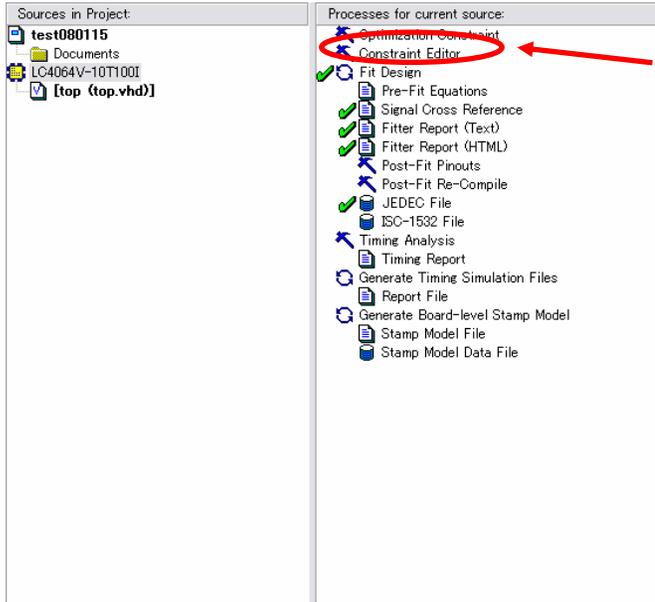
Svf_erase_program_verify_secure (ON / OFF)

SVFファイルにErase、Program、Verify、Secureの設定を行います。JEDECファイル生成時に生成されるSVFファイルに書き込みオペレーションを指定します。

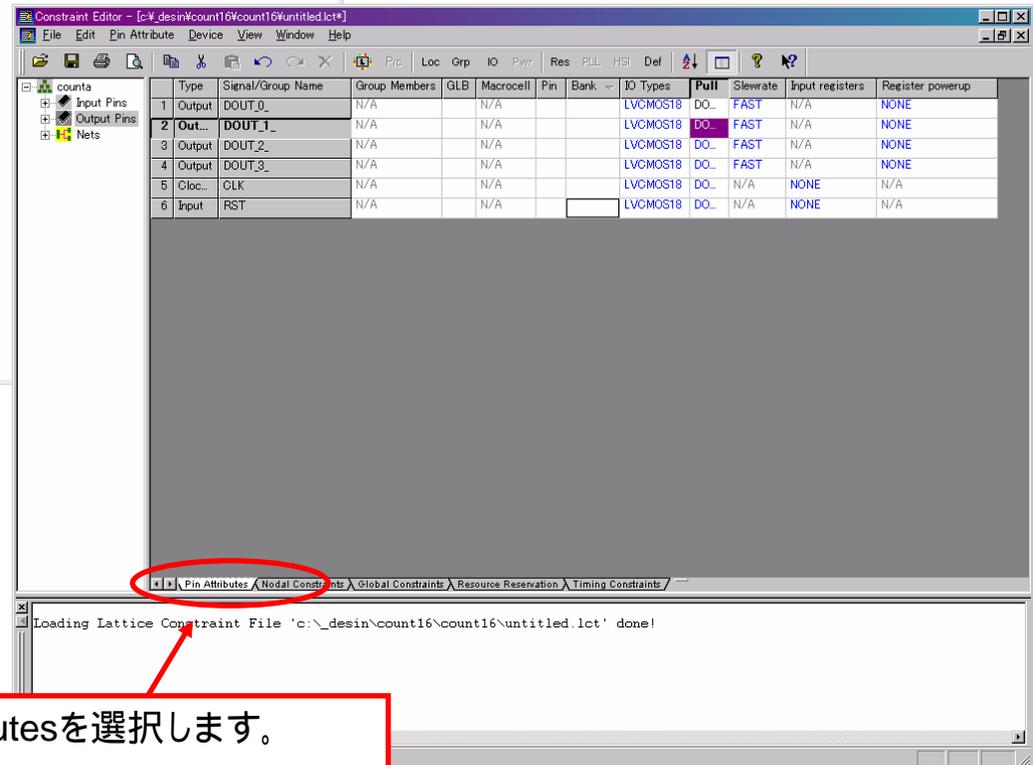
Svf_verify_only (ON / OFF)

SVFファイルにVerify onlyの設定を行います。JEDECファイル生成時に生成されるSVFファイルに書き込みオペレーションを指定します。

IOの属性設定を行います。



まず、ispLEVERのConstraint Editorを起動します。



次に、PIN Attributesを選択します。

5.5. ピンの属性の設定について

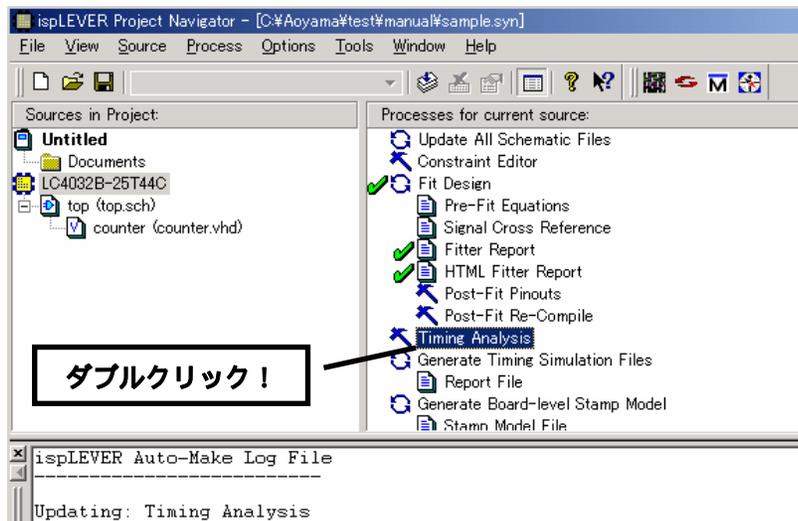
この項目では以下の設定を行うことが可能です。

	Type	Signal/Group Name	Group Members	GLB	Macrocell	Pin	Bank	IO Types	Pull	Slewrate	Input registers	Register powerup
1	Output	DOUT_0_	N/A		N/A			LVC MOS18	DO...	FAST	N/A	NONE
2	Out...	DOUT_1_	N/A		N/A			LVC MOS18	DO...	FAST	N/A	NONE
3	Output	DOUT_2_	N/A		N/A			LVC MOS18	DO...	FAST	N/A	NONE
4	Output	DOUT_3_	N/A		N/A			LVC MOS18	DO...	FAST	N/A	NONE
5	Cloc...	CLK	N/A		N/A			LVC MOS18	DO...	N/A	NONE	N/A
6	Input	RST	N/A		N/A			LVC MOS18	DO...	N/A	NONE	N/A

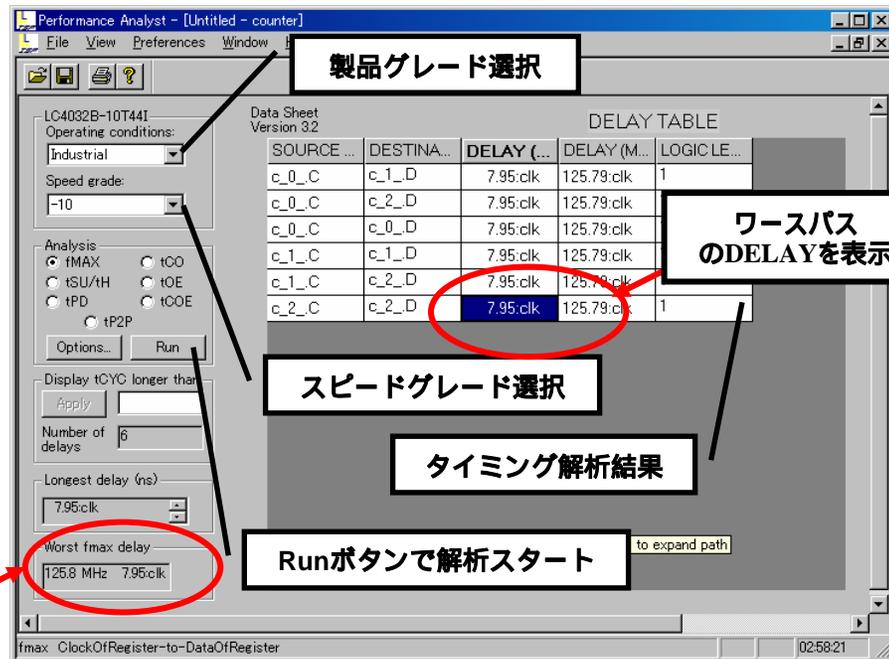
- Group Members ...ピンをグループ化させることができます。
- GLB ...ピンフリーの際、GLBを指定することができます。
- Macrocell ...ピンを設定するとマクロセルナンバーが表示されます。
- Pin ...ピン番号を直接入力出来ます。
- Bank ...バンクを指定出来ます。
- IO Types ...Type IOレベルを決定することができます。
- Pull ...MACH4000ZEでは、この項目によりピン毎にUP/DOWN/HOLD/OFF (プルアップ/プルダウン/前置保持/未設定)から選択出来ます。
この設定はMACH4000ZEにだけ対応しています。
- Slewrate ...スルーレートをFAST/SLOWから選択出来ます。
- Input registers ...INREGを選択することで、マクロセルへの速い入力経路を設定できます。
これによりセットアップの時間を早く出来ます。
- Register powerup ...NONE/RESET/SETから選択出来ます。
電源がONになったときに自動的にRESETやSETをかけることができます。
ただし、電源の立ち上がり方等の条件によってはリセットされないことがありますので、必ず外部からリセットをかけれる構造をとることをお勧めします。

タイミング解析を行います。

1. [Timing Analysis]をダブルクリックして下さい。



2. [Performance Analyst]が起動します。



ワースト値を表示

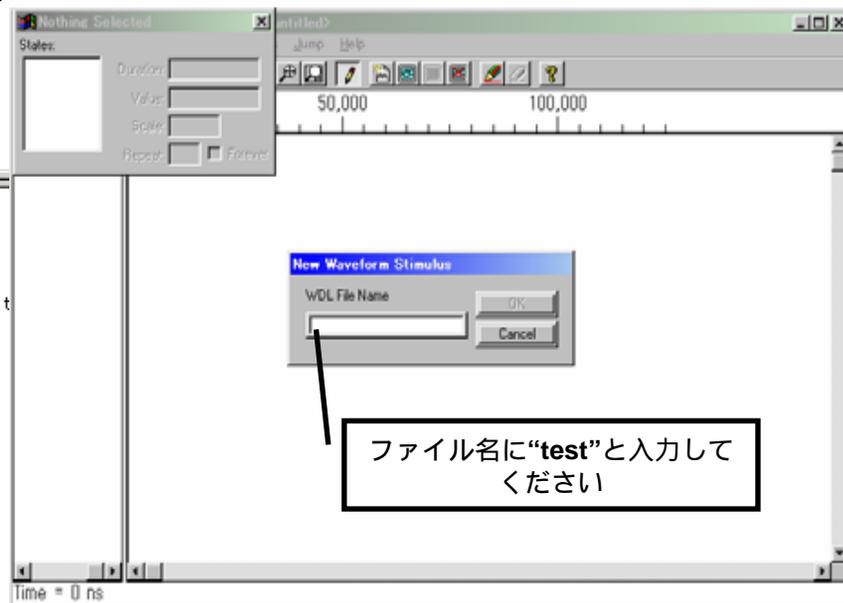
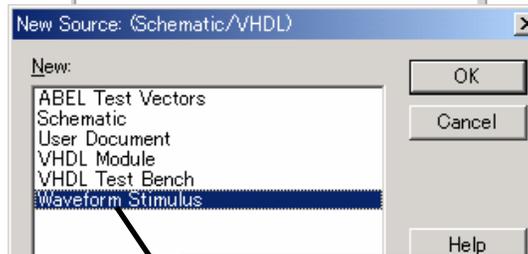
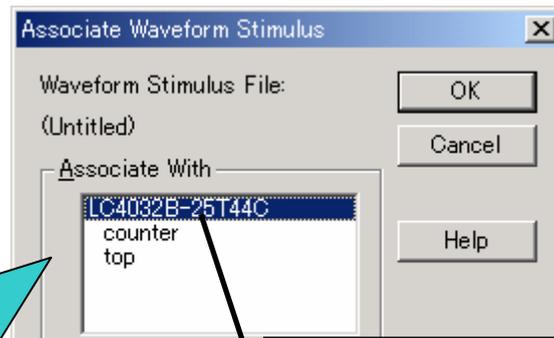
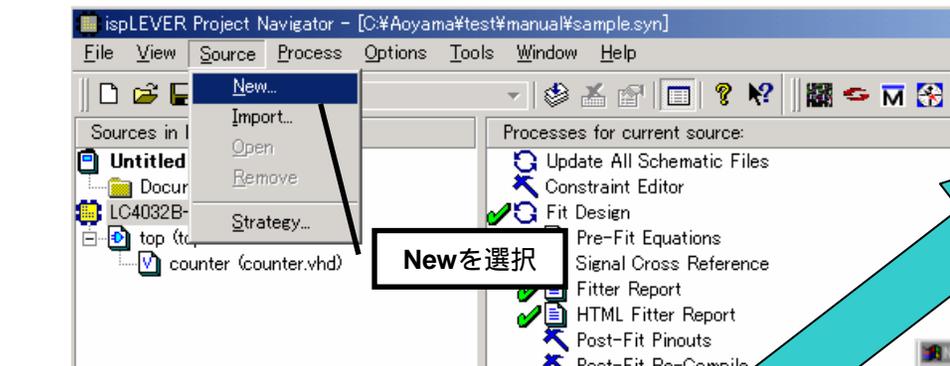
タイミング解析ツールの優位点
ispLEVERにバンドルされているタイミング解析ツールは**スピードグレードの異なる解析結果を再コンパイルすることなく表示することができる**にあります。是非お試しください。

- fMAX ... クロック最大動作周波数
- tSU/tH ... セットアップ/ホールドタイム
- tPD ... TPD (ピン間の遅延時間)
- tCO ... TCO (クロック to アウト)
- tOE ... OE信号のタイミング (組み合わせOE)
- tCOE ... OE信号のタイミング (クロックOE)
- tP2P ... 指定したピン間の遅延時間

8. 波形シミュレーション

最初にシミュレーションに用いる波形を作成します

1. Project Navigatorから[Source] [New...]を選択します。続いて、表示されたウィンドウから“Waveform Stimulus”を選択しクリックします。



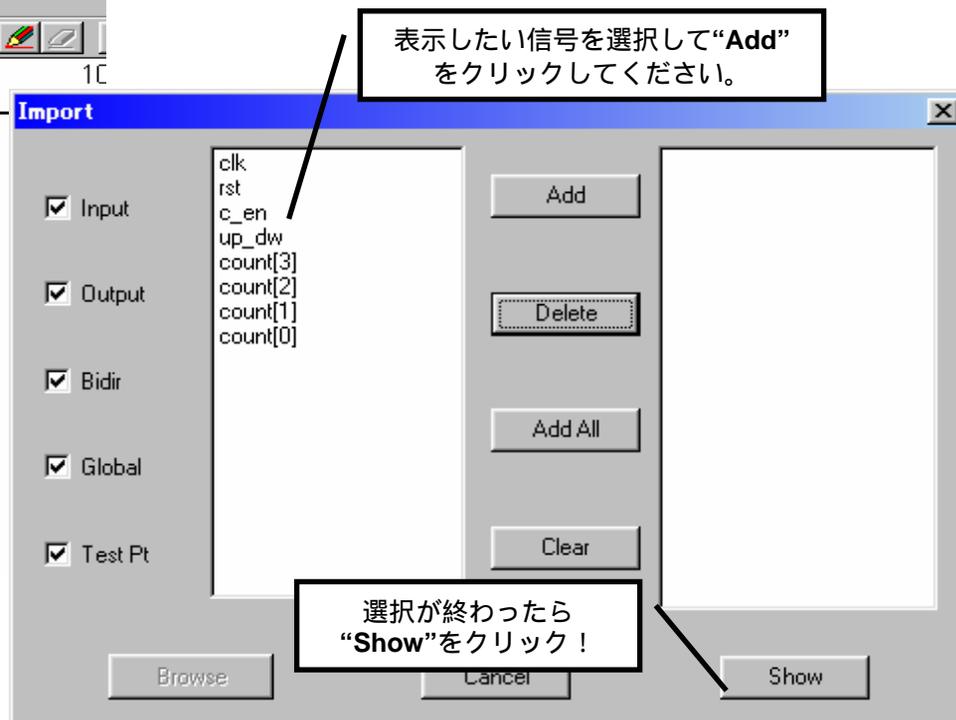
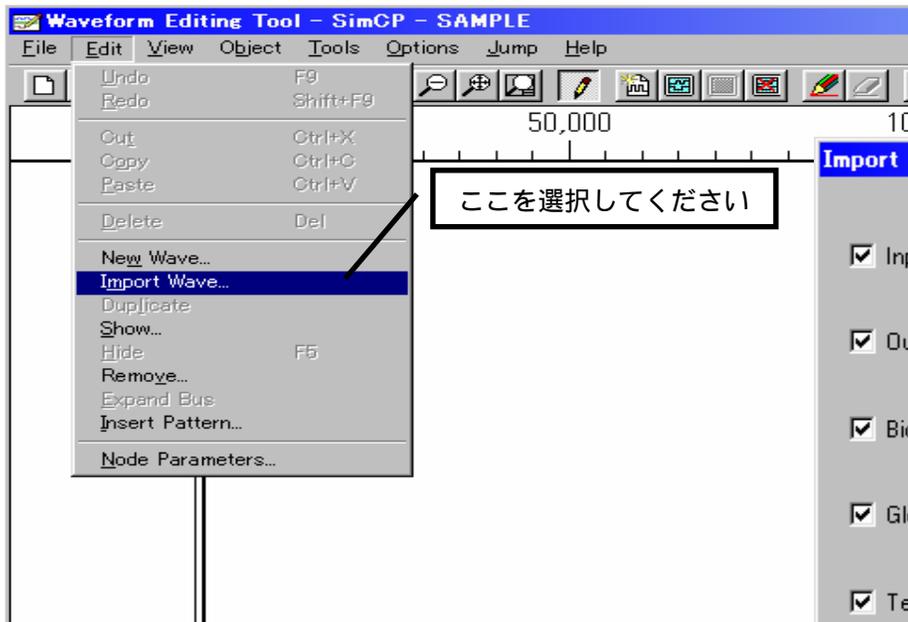
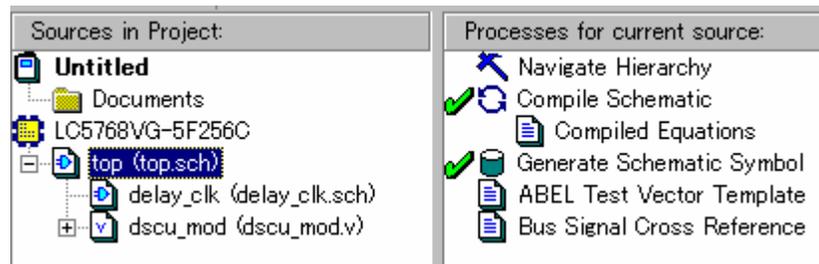
シミュレーションファイルの関連付けについて

[Associate Waveform Stimulus]ウィンドウで、波形ファイルをデバイスに関連付けることによりタイミングシミュレーション(デバイス固有の遅延値を含んだシミュレーション)も実行することが可能になります。ソース名に関連付けをするとファンクションシミュレーション(ゲートレベルのシミュレーション)のみ実行可能になります。

ソースファイルから信号をインポートします

1. トップソースをハイライトし、右ウインドに表示される **[Generate Schematic Symbol]** を実行して下さい。

(階層設計の場合、トップソースについてはモジュールのI/F部分を管理している*.NAFファイルが自動生成されないため、上記作業によりファイルを生成する必要があります。下位階層のソース及びスタンダードアローンで設計を行なっている場合にはこの作業は必要ありません)

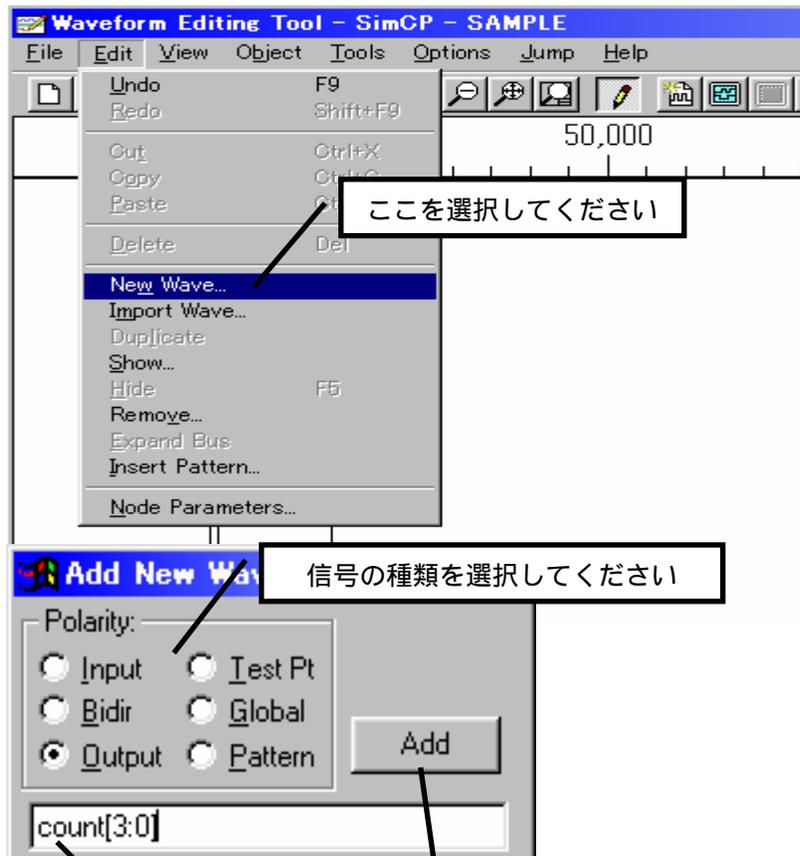


2. [Edit] [Import Wave]を選択してソースファイルで記述した信号をシミュレーターにインポートします。

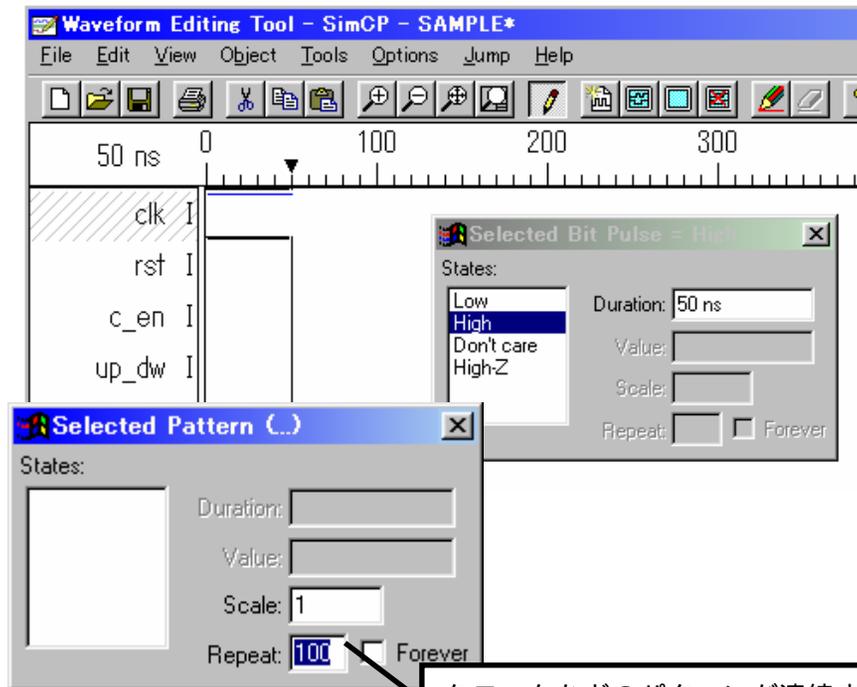
8. 波形シミュレーション

信号をバスでまとめ、実際に波形を入力します

1. [Edit] [New Wave]を選択して 入力信号をバスでまとめます(入力バス化にはこの作業が必要です!!)。



2. 実際に波形を入力する際には、信号名をハイライト(斜線が表示されます)して、必要なパルス幅の部分でクリックします。すると、**Selected Bit Pluse**ウィンドウに信号の極性と幅が表示されます。



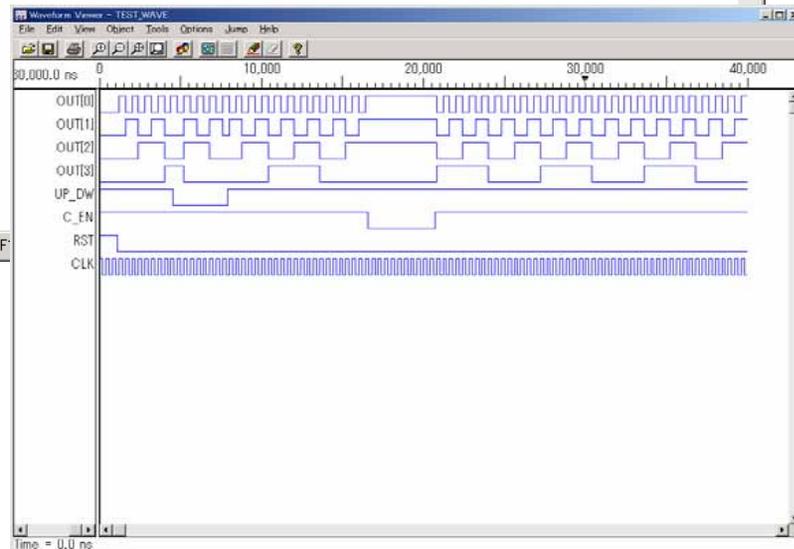
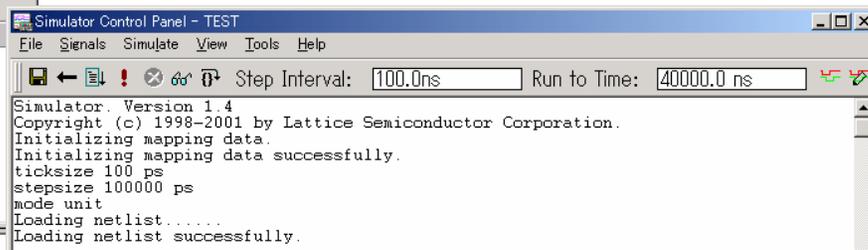
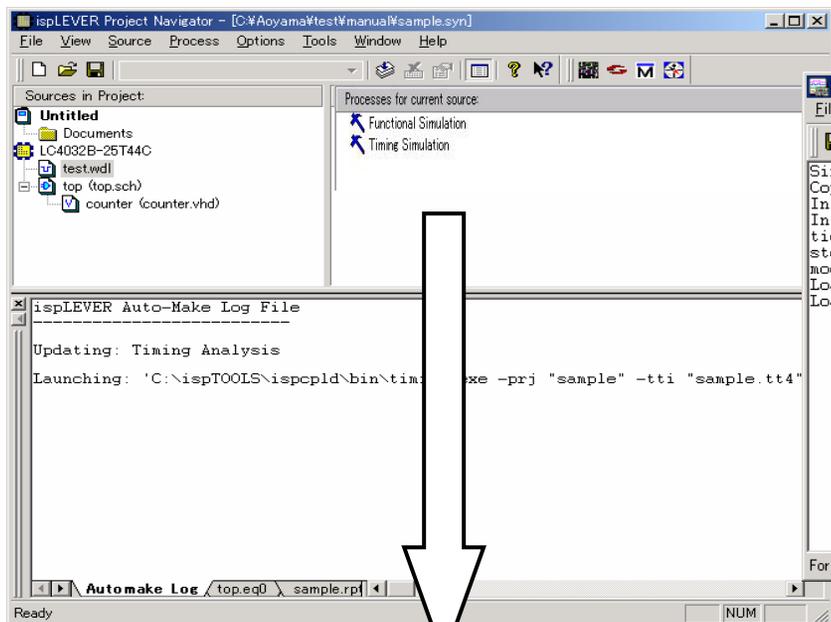
クロックなどのパターンが連続する波形については、信号を選択してRepeatダイアログに数値を入力してください。

8. 波形シミュレーション

それでは実際にシミュレーションをしてみましょう

1. シミュレーション波形ファイルをハイライトして右画面の **Function Simulation** をダブルクリックしてください。

2. 下図のような画面が表示されたらアイコン  をクリックしてください。



シミュレーションには以下の2種類があります。

- Functional Simulation **波形シミュレータが起動します**
- Timing Simulation **波形シミュレータが起動します**

Functionalは論理レベルで早期に波形シミュレーション可能
Timing Simulationはfitting後遅延も考慮した波形シミュレーション可能

プロジェクトのタイプによって表示されないものもあります。詳細については該当の各マニュアルを参照してください。また、上記波形シミュレーション以外(ActiveHDL)も使用可能ですが、オペレーションについては別途マニュアルを参照してください。

8. 波形シミュレーション

表示されたシミュレーション結果を観察してみましょう(信号のバス化)

1. [Edit] [Show]を選択して、“Show Waveform”というウィンドウを表示してください。
2. の順番にボタンをクリックしてください。
[Save]を忘れずに。

Showを選択してください

AddNet(s)をクリック

ShowBusをクリック

Showをクリック

出力バス化したい信号
を選択してください

選択したらクリック!

バス化された信号が
確認できます

上記ウィンドウ内の“D”とある部分ダブルクリックすると、内部の信号 (F/F、State-Machine等)を観察することができます。但し、内部の組み合せ論理を観察することはできません。

以上でispLEVER Clasic 1.2 Startup Manual for Mach 4000は終了です。

より詳細なお問合せ、ご質問等に関しましては、技術サポート貴社担当FAE
または下記技術サポート窓口までお気軽にお問い合わせ下さい。

株式会社 マクニカ テクスターカンパニー ラティス製品 技術サポート窓口

電話 045-470-9841/FAX 045-470-9844

Email lattice@macnica.co.jp

URL <http://www.tecstar.macnica.co.jp/contact/index.html>



