

SynaptiCAD

WaveFormer Pro

Interactive Verilog Simulator の概要と操作

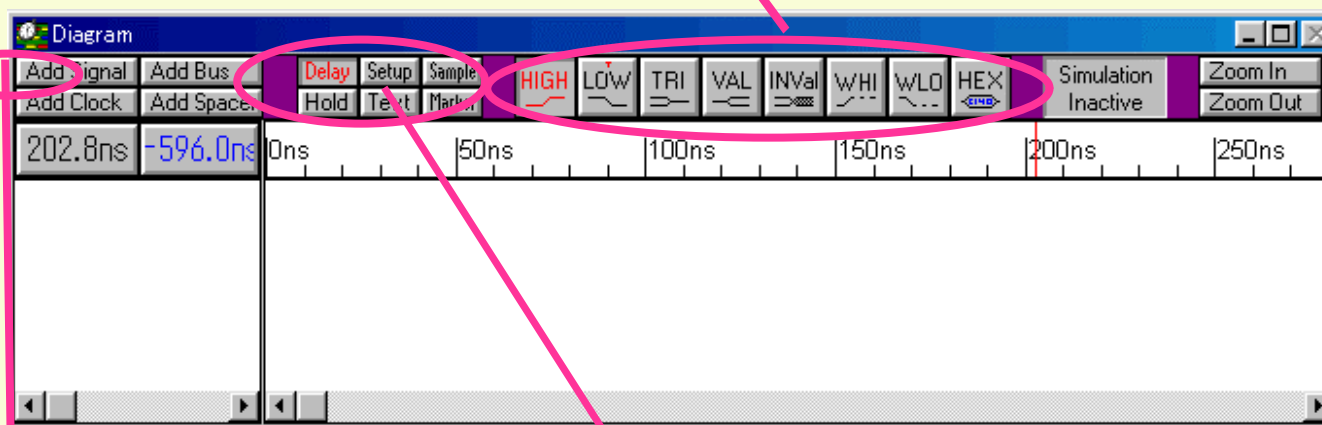
What is WaveFormer Pro ?

- タイミング・ダイヤグラムの編集と
スタティック・タイミング・アナライザでの解析
- ステイミュラスの生成とシミュレータ ATEのI/Fをサポート
VHDL , Verilog , ABEL , Minc , ViewLogic , Mentor , Aldec-
Xilinx , SPICE , VCD...
HPロジックアナライザ , STIL IEEE テスタ・フォーマット
TDML標準化タイミング記述フォーマット(SI2-ECIX)
[インターフェース仕様公開 Perl言語で追加/カスタマイズ可能]
- インタラクティブ Verilog-HDLシミュレータ
- マウス操作で入力された波形や論理式などからVerilog
コードを自動生成
- 変更ごとに再シミュレーションを自動実行
- 入力と確認が同時のため RTL設計入力が高効率

WaveFormer Pro 超！簡単な基本操作(1)

基本操作ボタン

状態の種類を表すボタン



新たな信号の追加

モード・ボタン

おもな操作は , 上記のアイコンで実行できます

WaveFormer Pro 超！簡単な基本操作(2)

まずは時間単位の設定

基本時間単位設定



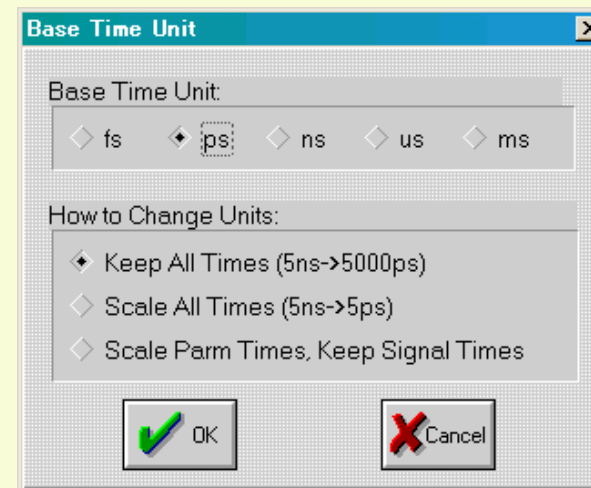
表示時間単位設定



信号の定義/追加



信号の属性を決定

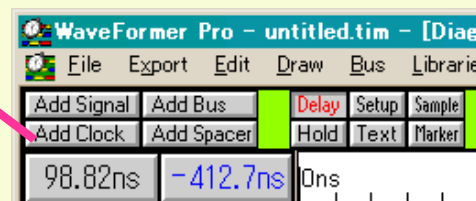


基本時間単位設定
ダイアログ・ボックス

WaveFormer Pro 超！簡単な基本操作(3)

clockの定義は...

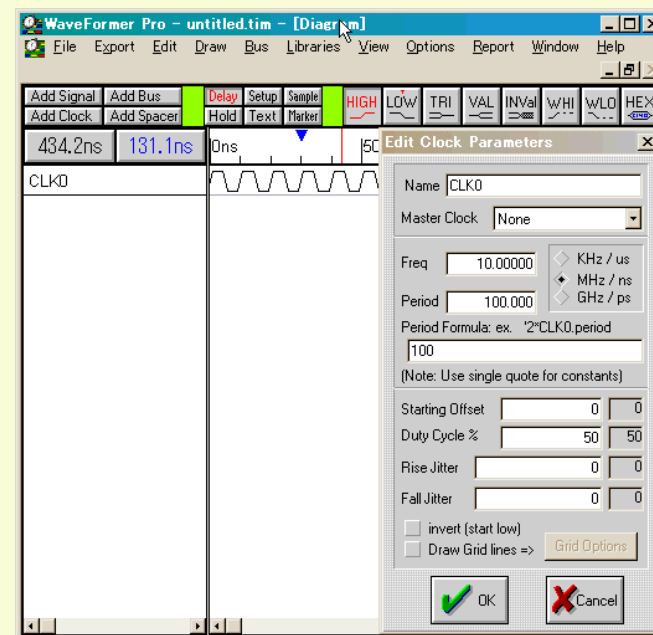
Add Clock ボタンを押す



すると,デフォルトでCLK0の波形ができ
Edit Clock Properties ダイアログが表示
れます

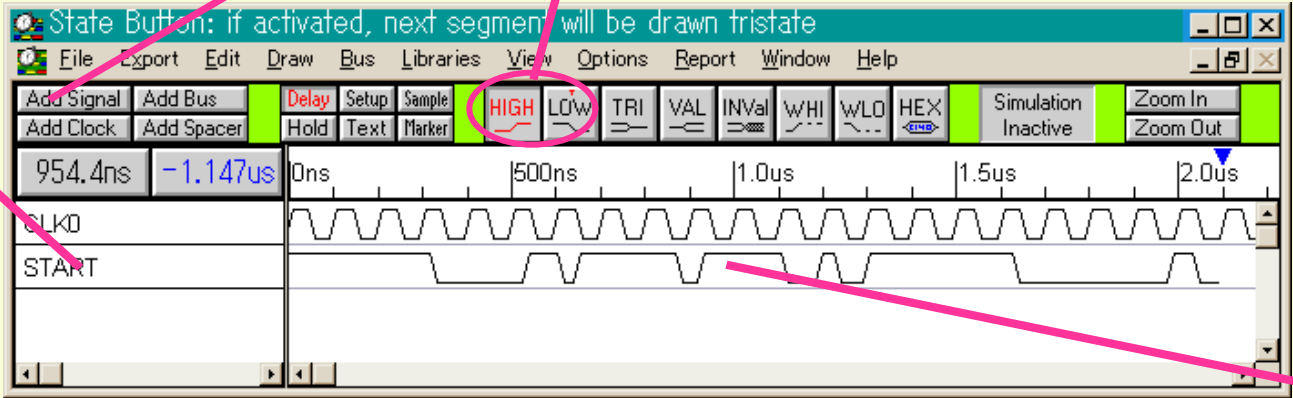
デフォルト値を確認して,必要なら変更して
OKを押します

通常の信号」も「バス」もマウス・クリック
または簡単な式で素早く入力可能



WaveFormer Pro 超！簡単な基本操作(4)

信号の定義



The screenshot shows the WaveFormer Pro software interface. The menu bar includes File, Export, Edit, Draw, Bus, Libraries, View, Options, Report, Window, and Help. The toolbar contains buttons for Add Signal, Add Bus, Delay, Setup, Sample, HIGH, LOW, TRI, VAL, INVal, WHI, WLO, HEX, Simulation Inactive, Zoom In, and Zoom Out. The HIGH and LOW buttons are circled in pink. Below the toolbar, there are time scale markers: 954.4ns, -1.147us, 0ns, 500ns, 1.0us, 1.5us, and 2.0us. The waveform plot shows two signals: CLK0 and START. The START signal is a square wave. A pink box labeled '信号名をSTARTに' points to the START signal name. Another pink box labeled '信号のHigh, Low ボタン' points to the HIGH and LOW buttons. A third pink box labeled 'Add Signal ボタン' points to the Add Signal button. A fourth pink box labeled '時間スケールを参考にマウスをクリックすると Low, High波形が交互に描かれます' points to the waveform plot area.

信号のHigh, Low ボタン

Add Signal ボタン

信号名をSTARTに

時間スケールを参考にマウスをクリックすると Low, High波形が交互に描かれます

WaveFormer Pro 超！簡単な基本操作(5)

時間式での波形入力

時間式入力
により自動生成
された波形

Wfm Eqnボタン
右に式を入力します

The screenshot displays the WaveFormer Pro interface. The main window shows a waveform for a signal named 'ADDR'. The waveform is a periodic square wave. The time scale is set to 100ns. A callout box points to the 'Wfm Eqn' field in the 'Signal Properties' dialog box, which contains the equation $20=V(5=Z10=U)*10$. Another callout box points to the 'Wfm Eqn' field in the main window, which contains the equation $26.11ns$. The 'Signal Properties' dialog box is open, showing various settings for the signal, including Name (ADDR), Boolean Equation, Clock, and Wfm Eqn.

WaveFormer Pro - untitled.tim - [Diagram]

File Export Edit Draw Bus Libraries View Options Rep...

Delay Setup Sample HIGH LOW TRI VAL

Hold Text Marker

26.11ns 0ns 50ns 100ns

ADDR

Signal Properties

Name: ADDR Properties

active low name (adds bar on top, \$BAR suffix)

Boolean Equation: ex. (SIG1 and SIG2) delay 5

Clock: Unlocked Edge/Level: neg

Clock to Out: 0 Setup: 0

Startup State: unknown Hold: 0

Boolean Equation HDL Code

Simulate Once Continuously Simulate

Wfm Eqn 20=V(5=Z10=U)*10

Export Signal Direction: shared output

VHDL Type: std_logic

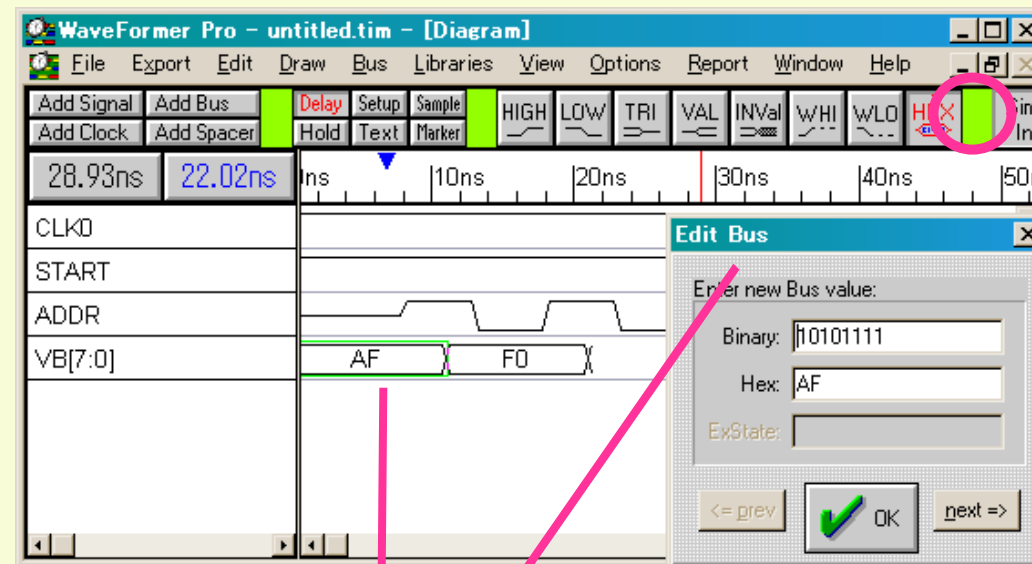
Verilog Type: wire

Radix: bin Bus MSB: 0 LSB: 0

OK Cancel Apply Prev Sig Next Sig

WaveFormer Pro 超！簡単な基本操作(6)

バス値の設定

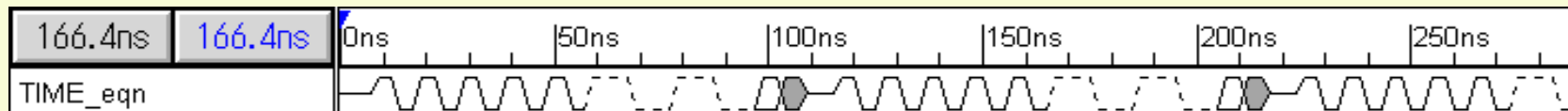
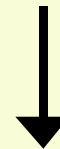
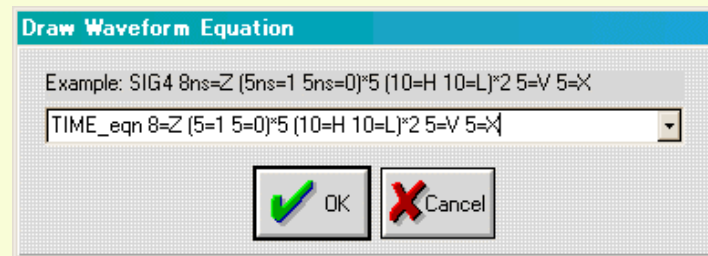


セグメント内で
左マウスクリックで選択
グリーンの枠で囲まれます

HEXボタンを押すと
ダイアログが表示され、値を
入れることができます

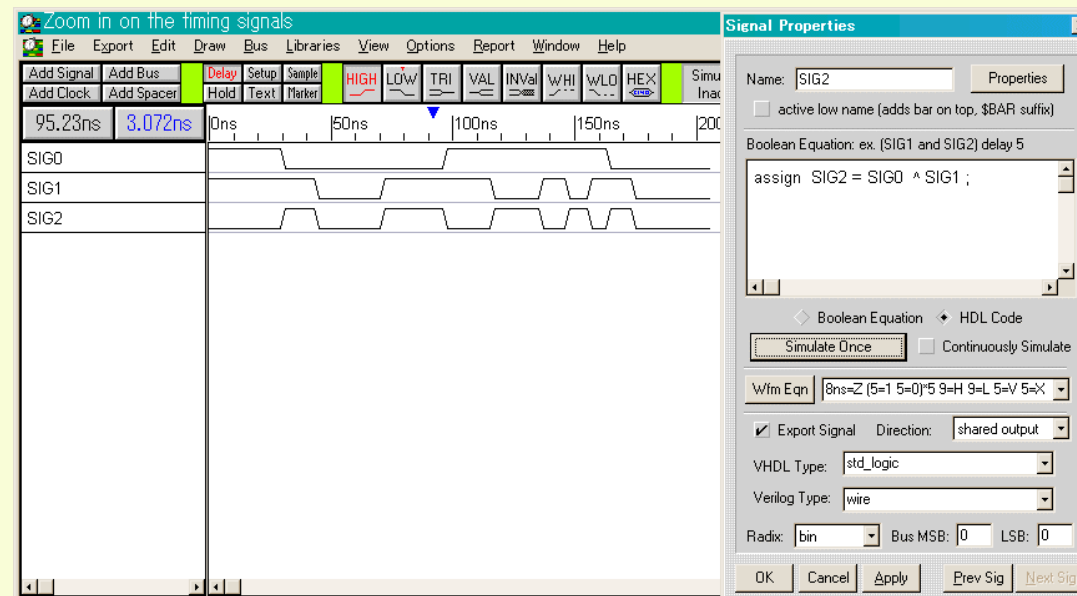
WaveFormer Pro 超！簡単な基本操作(7)

バス値 (ステート名) 入力の自動化



WaveFormer Pro 超！簡単な基本操作(8)

RTL設計入力



他の信号の論理演算結果として信号を定義

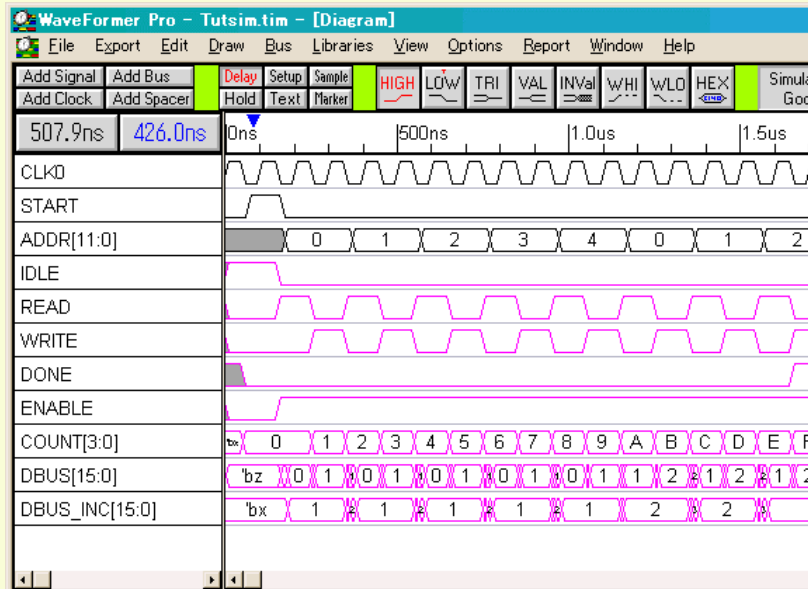
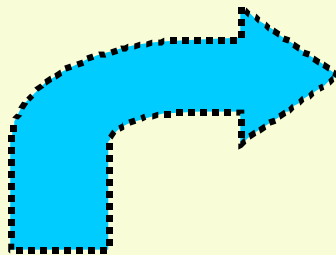
- ブール式で

- Verilog-HDLで

!! WaveFormer が Verilogコードに変換しVeriWellで自動シミュレーション

WaveFormer Pro 超！簡単な基本操作(9)

WaveFormerにより
生成されたVerilog コード



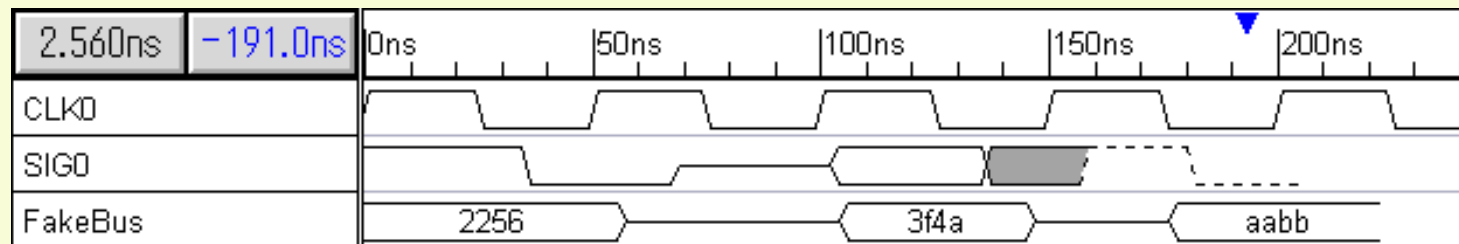
```
`timescale 1ps / 1ps
module
testbed(CLK0,START,ADDR,IDLE,READ,WRITE,DONE,ENABLE,COUNT,DBUS,D
BUS_INC);

output CLK0;
reg CLK0;
output START;
reg START;
output [11:0] ADDR;
reg [11:0] ADDR;
output IDLE;
reg IDLE;
output READ;
reg READ;
output WRITE;
reg WRITE;
output DONE;
output ENABLE;
output [3:0] COUNT;
output [15:0] DBUS;
output [15:0] DBUS_INC;
reg [15:0] DBUS_INC;

integer CLK0_stop_time;
integer CLK0_period, CLK0_duty,CLK0_offset;
integer CLK0_d1, CLK0_d2;
initial
begin
//CLOCK CLK0
CLK0_stop_time = 1999000;
CLK0_offset = 0;
CLK0_period = 100000;
CLK0_duty = 50;
CLK0_d1 = CLK0_period * CLK0_duty/100;
CLK0_d2 = CLK0_period - CLK0_d1;
```

スティミュラスの自動生成 Verilog コード(1)

Verilog-HDLスティミュラスの生成例を見てみましょう



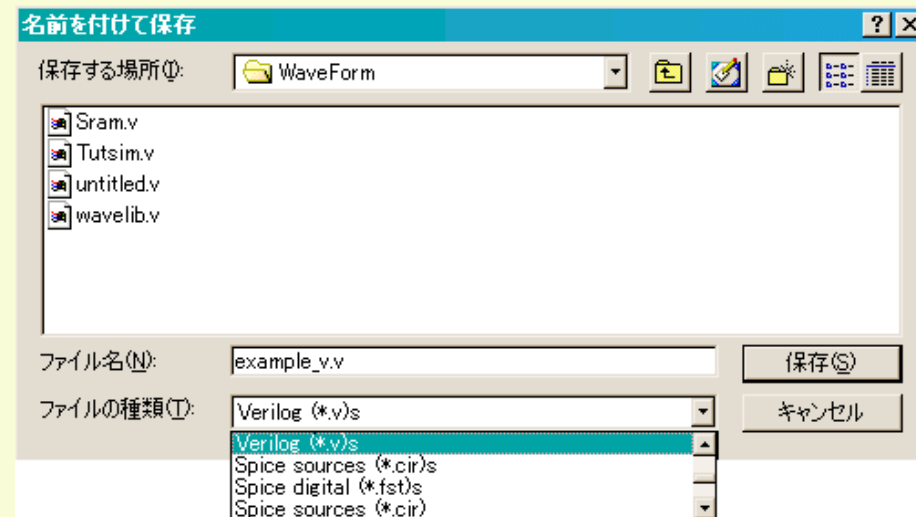
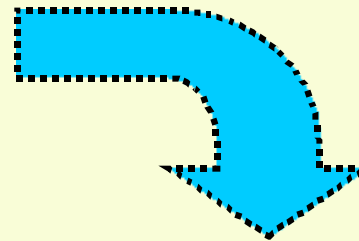
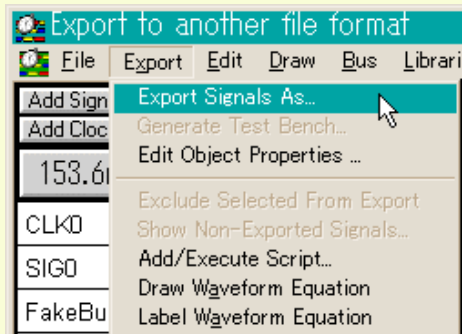
CLK0 :**周期 50[ns]のクロック**

SIG0 :WaveFormerで扱える全ての波形タイプを含んだ信号

FakeBus :**スリー・ステート・セグメントを含んだ仮想バス信号**

ステイミュラスの自動生成 Verilog コード(2)

Verilog-HDLステイミュラスの生成



ステイミュラスの自動生成 Verilog コード(3)

生成されたVerilog-HDLステイミュラス

```
`timescale 1ps / 1ps
module testbed(CLK0,SIG0,FakeBus);

output CLK0;
reg CLK0;
output SIG0;
reg SIG0;
output FakeBus;
reg FakeBus;

integer CLK0_stop_time;
integer CLK0_period, CLK0_duty,CLK0_offset;
integer CLK0_d1, CLK0_d2;
initial
begin
//CLOCK CLK0
CLK0_stop_time = 223744;
CLK0_offset = 0;
CLK0_period = 50000;
CLK0_duty = 50;
CLK0_d1 = CLK0_period * CLK0_duty/100;
CLK0_d2 = CLK0_period - CLK0_d1;
CLK0 = 1'b0;
#(CLK0_offset) CLK0 = 1'b1;
while ($time < CLK0_stop_time)
begin
#(CLK0_d1) CLK0 = 1'b0;
#(CLK0_d2) CLK0 = 1'b1;
end
end
end
```

```
initial
begin
//SIGNAL SIG0
SIG0 = 1'b1;
#35328
SIG0 = 1'b0;
#32768
SIG0 = 1'bz;
#34816
SIG0 = 1'bx;
#33280
SIG0 = 1'bx;
#21504
SIG0 = 1'bweak1;
#23552
SIG0 = 1'bweak0;
#25088
;
end
initial
begin
//SIGNAL FakeBus
FakeBus = 2256;
#56320
FakeBus = 1'bz;
#48640
FakeBus = 3f4a;
#40960
FakeBus = 1'bz;
#31232
FakeBus = aabb;
#46592
;
end
initial
#223744 $finish;

endmodule
```

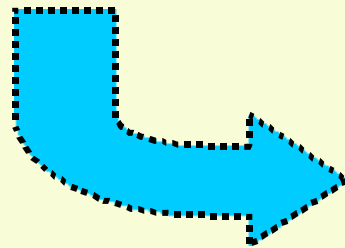
ステイミュラスの自動生成 Verilog コード(4)

WaveFormerの大きな特長

- システムのインターフェース
はすべてPerl言語で記述
- WaveFormer Pro本体との
プロトコル仕様を公開
- カスタマイズ可能

(例)

Verilog-HDLステイミュラスを生成
する Verilog.epf ファイルを見て
みましょう



```
# Copyright 1996 SynaptiCAD
# Requires TWF v3.0 or greater
#----- Export to Verilog

#(strict compliant)
$I=1;
#uses "future state" format

require 'twfsubs.pl';

%ToState = ('1' => '1',
            '0' => '0',
            'H' => 'weak1',
            'L' => 'weak0',
            'Z' => 'z',
            'V' => 'x',
            'X' => 'x',
            '!' => '!',
            " " => " ");

IF_To_Verilog();
return 1;

sub IF_To_Verilog {
    $LastOutputTime = twf::GetLastOutputTime();
    $BUnits = $twf::ToUnits[ twf::GetOptions()->GetBaseTimeUnit ( ) ];
    $DUnits = $twf::ToUnits[ twf::GetOptions()->GetBaseTimeUnit ( ) ];
    print "`timescale 1$DUnits / 1$BUnits`n";

    CreateModuleAndSignalDeclarations();

    my $sigObjIPtr = twf::NewSigObjIterator( twf::GetScrSignals ( ) );
    $sigObjIPtr->GoNextValidEx();
    my $sigObj;
```

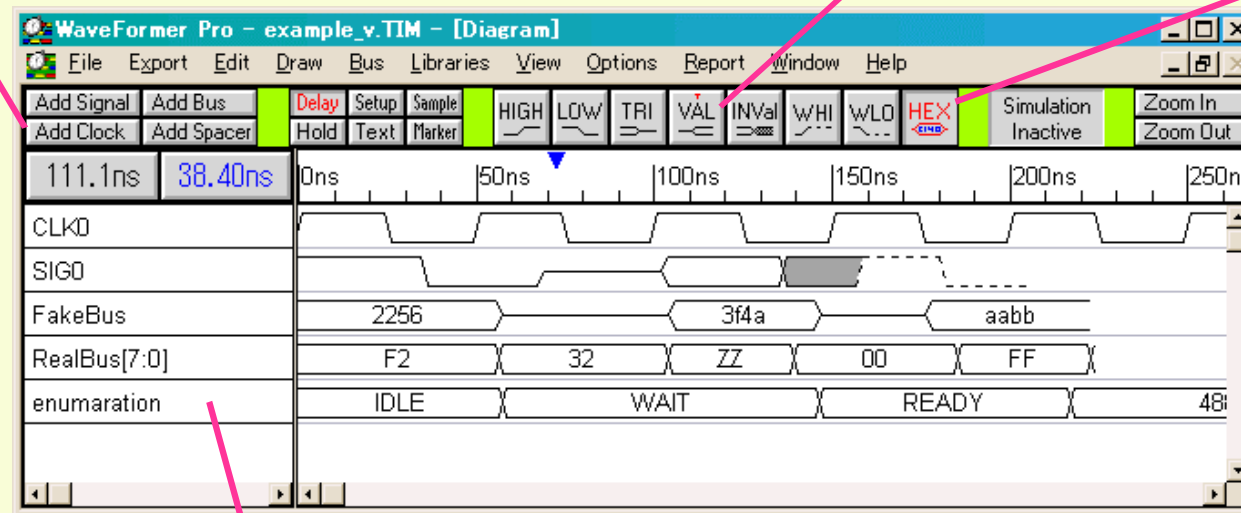
ステイミュラスの自動生成 VHDL コード(1)

VHDLでは列挙型もサポート

AddSignalボタン

VALボタン

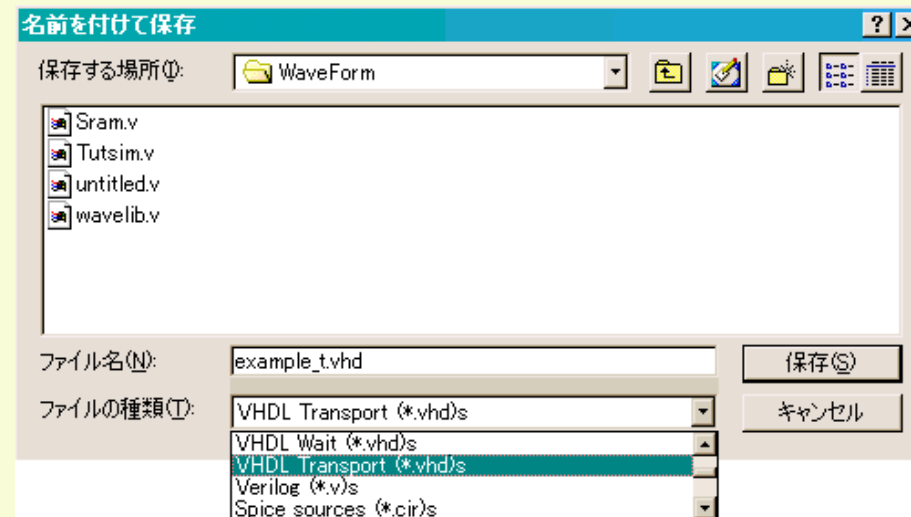
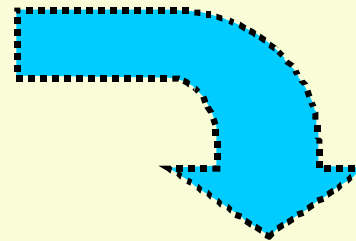
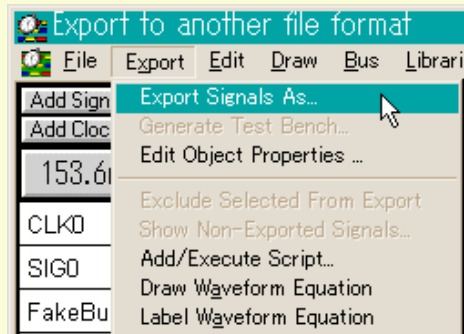
HEXボタン



IDLE, WAIT, READYは列挙型の値として

ステイミュラスの自動生成 VHDL コード(2)

TransportとWaitでステイミュラスを生成してみましよう



ステイミュラスの自動生成 VHDL コード(3)

Transport文

```
architecture test of testbench is
begin

process
begin
  CLK0 <= '0';
  wait for 0 ns;
  while true loop
    CLK0 <= '1';
    wait for 25 ns;
    CLK0 <= '0';
    wait for 25 ns;
  end loop;
end process;

process
begin
  SIG0 <=
    transport '1',
      '0' after 35.328 ns,
      'Z' after 68.096 ns,
      'X' after 102.912 ns,
      'X' after 136.192 ns,
      'H' after 157.696 ns,
      'L' after 181.248 ns;
  FakeBus <=
    transport 2256,
      'Z' after 56.32 ns,
      3f4a after 104.96 ns,
      'Z' after 145.92 ns,
      aabb after 177.152 ns;
```

Wait文

```
architecture test of testbench is
begin

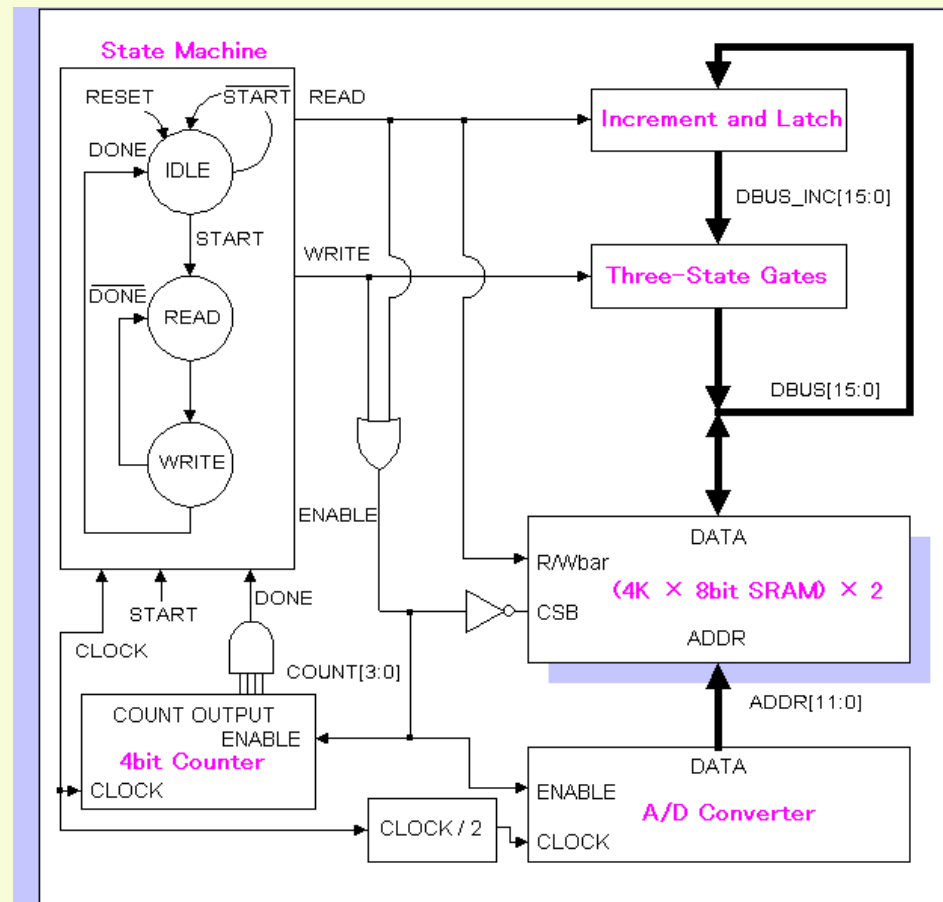
process
begin
  CLK0 <= '0';
  wait for 0 ns;
  while true loop
    CLK0 <= '1';
    wait for 25 ns;
    CLK0 <= '0';
    wait for 25 ns;
  end loop;
end process;

process
begin
  SIG0      <= '1';
  FakeBus   <= 2256;
  RealBus0  <= '0';
  RealBus1  <= '1';
  RealBus2  <= '0';
  RealBus3  <= '0';
  RealBus4  <= '1';
  RealBus5  <= '1';
  RealBus6  <= '1';
  RealBus7  <= '1';
  enumeration <= IDLE;
  wait for 35.328 ns;

  SIG0      <= '0';
  wait for 20.992 ns;
```

インタラクティブ・シミュレーション(1)

中規模の設計なら RTL入力機能ですばやくモデリングが可能



インタラクティブ・シミュレーション(2)

シミュレーション波形とVerilog-HDLコードの同時出力

The screenshot displays the WaveFormer Pro interface for a simulation named 'Tutstim.tim'. The left pane shows the Verilog-HDL code, and the right pane shows the corresponding timing diagram.

```
Report - C:\WaveForm\Tutstim.v
wire IDLE_wf3 = (WRITE & DONE) | (~START & IDLE);
registerN #(1,0,0) registerN_IDLE(IDLE,CLKD,IDLE_wf3,1'b1);

wire READ_wf1 = (IDLE & START) | (WRITE & ~DONE);
registerN #(1,0,0) registerN_READ(READ,CLKD,READ_wf1,1'b0);

wire WRITE_wf65 = READ;
registerN #(1,0,0) registerN_WRITE(WRITE,CLKD,WRITE_wf65,1'b0);

assign DONE = &COUNT;

assign ENABLE = READ | WRITE;

reg [3:0] COUNTER; //declare a 4-bit register called COUNTER
always @(negedge CLKD) //on each falling edge of CLKD
begin
if (ENABLE)
COUNTER = COUNTER + 1; // count while ENABLE is high
else
COUNTER = 0; // synchronous reset if ENABLE is low
end
assign COUNT = COUNTER; //drive wire COUNT with reg COUNTER value

wire CSB = !ENABLE;
sram BinMem1(CSB,READ,ADDR,DBUS[7:0]);
sram BinMem2(CSB,READ,ADDR,DBUS[15:8]);
assign DBUS = WRITE ? DBUS_INC : 'hz;

wire [15:0] DBUS_INC_wf2 = DBUS + 1;
latchH #(16,0,0) latchH_DBUS_INC(DBUS_INC,READ,DBUS_INC_wf2,16'bxxxxxxxxxxxxxxxx)
```

The timing diagram on the right shows the following signals:

- CLKD: Clock signal with a period of 1.638us.
- START: A single pulse.
- ADDR[11:0]: Address bus showing values 0, 1, 2, 3, 4, 0, 1.
- IDLE: Signal that is high when the system is idle.
- READ: Signal that is high during read operations.
- WRITE: Signal that is high during write operations.
- DONE: Signal that is high when a write operation is complete.
- ENABLE: Signal that is high when either read or write is active.
- COUNT[3:0]: 4-bit counter showing values 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E.
- DBUS[15:0]: Data bus showing values 'bz, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 2, 1, 2, 1.
- DBUS_INC[15:0]: Data bus showing values 'bx, 1, 1, 1, 1, 1, 1, 2, 2.