論理回路入門（１）

論理回路基礎１：論理演算と論理ゲート

李　亜民

2024 年 9 月 24 日 (火)

論理回路入門

授業の内容

- 論理回路の基本: ブール代数、真理値表、カルノー図、論理式の積和形と和積形、回路の設計と動作検証の方法
- 組合せ回路: 半加算器、全加算器、リップルキャリーアダー、キャリールックアヘッドアダー、ツリー型桁上げ先見加算器、加算器を利用した減算器、マルチプレクサ、7 セグメント LED、ALU、デコーダ、エンコーダ
- 順序回路: RS ラッチ、D ラッチ、D フリップフロップ、T フリップフロップ、JK フリップフロップ、レジスタ・ファイル、状態遷移図、交通信号機制御システム、N 進カウンター

論理回路入門

- 成績評価
  課題レポート成績 × 60% +
  期末試験成績 × 40% (期末試験: 参照不可)
- 課題レポートの提出：次回の授業開始前までに
  いかなる理由でも遅延提出は認めない (遅刻厳禁)
  課題レポートの提出：学習支援システム (Hoppii)
- 教科書: 授業のスライドの PDF ファイル
- 参考書: ディジタル回路設計とコンピュータアーキテクチャ 第 2 版 2017
- その他: ノート PC をクラスに持ち込んでください

論理回路入門

回路の設計と動作検証の方法について

- C 言語でソフトウェアを開発する場合、C プログラムをコンパイルするために C コンパイラが必要である。コンパイルされたプログラムを実行して、期待通りの結果が得られるかどうかを確認する。
- 同様に、設計した回路が期待通りの結果を出せるかどうかを検証することも必要である。
- 検証する方法はいくつかある。たとえば、Xilinx ISE / Vivado、Intel Quartus / ModelSim などである。ただし、これらのツールは M1 Mac PC では実行できない。
- この講義では、Verilog HDL を使用して論理回路を実装する。講義スライドでは、Verilog HDL 回路の例とすべての検証用のテストベンチを用意する。

今日のポイント

論理演算と論理ゲート

ポイント

- ０と１の表現（電圧の低／高）
- 論理演算と論理ゲート
  - 基本的な論理演算と論理ゲート: AND、OR、NOT
  - ほかのゲート: NAND、NOR、XOR、XNOR
- 論理演算の理解（電球を点灯する回路）
- 真理値表、論理式と論理回路
- ハードウェア記述言語
  - Verilog HDL (Hardware Description Language)
- シミュレーションのためのテストベンチ (Test bench)
- iverilog と gtkwave

論理回路とは

- 論理回路は論理演算を行う電気回路である。
- デジタル回路とも呼ばれる。
- 論理回路を使う場所
  - コンピューター、携帯電話、ゲーム機
  - テレビ、DVD レコーダー、デジタルカメラ
  - 人工衛星、ナビゲーション
  - 冷蔵庫、洗濯機、電子レンジ
  - コピー機、プリンタ
  - 電卓、時計、リモコン
  - 自動運転車、・・・・・

論理演算とは

- コンピュータのハードウェアは、電圧の高／低または電圧の有／無の状態を動作の基本としている。
- これら二つの状態を数値化して表現するには、１と０の二つの数値を組み合わせる２進数が最適である。
- コンピュータでは、

  電圧が高いまたは電圧がある状態を２進数の１に、
  電圧が低いまたは電圧が無い状態を２進数の０に
  割り当てている。
- ２進数は１０進数と同じような四則演算 (和、差、積、商) のほかに、２進数特有な論理演算がある。
- 最も基本的な論理演算は論理積 (AND) と論理和 (OR) 及び否定 (NOT) である。

基本的な論理演算 (AND、OR、NOT)

基本的な論理演算 1. AND

1. AND (アンド)
   論理積 (かつ)
   『例』 $F = A \cdot B = A\ B$

   (Verilog HDL の表現)
   (F = A & B)

$0 \cdot 0 = 0$ 偽
$0 \cdot 1 = 0$ 偽
$1 \cdot 0 = 0$ 偽
$1 \cdot 1 = 1$ ..... 真 かつ 真 = 真

論理積は、入力値がすべて 1 のときに 1 を出力する。
それ以外の入力値のときは 0 を出力する。

基本的な論理演算 2. OR

2. OR (オア)
論理和 (または)
『例』 $F = A + B$

(Verilog HDL の表現)
(F = A | B)

0 + 0 = 0	..... 偽 または 偽 = 偽	
0 + 1 = 1		真
1 + 0 = 1		真
1 + 1 = 1		真

論理和は、入力値がすべて 0 のときに 0 を出力する。
それ以外の入力値のときは 1 を出力する。

基本的な論理演算 3. NOT

3. NOT (ノット)
否定
『例』 $F = \overline{A}$

(Verilog HDL の表現)
(F = ~A)

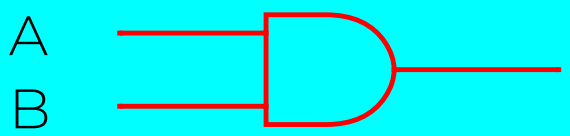
$\overline{0} = 1$ ..... 偽の否定 = 真
$\overline{1} = 0$ ..... 真の否定 = 偽

論理否定は、入力された値が 0 なら 1 に、
1 なら 0 に反転する。

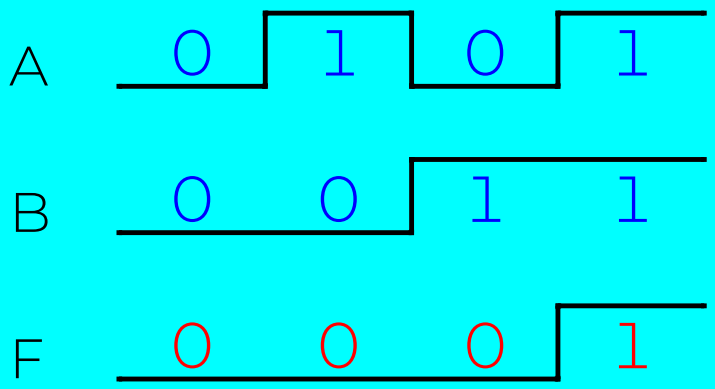
基本的な論理ゲート (AND、OR、NOT)

基本的な論理ゲート — 1. ANDゲート

論理積



波形



論理式 $F = A \cdot B = AB$

Verilog HDLの表現：`F = A & B`

論理積 (AND) は、入力値がすべて1のときに1を出力する。それ以外の入力値のときは0を出力する。

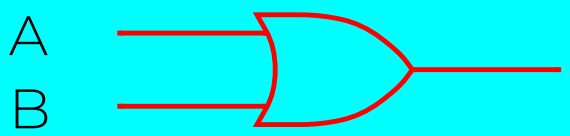
真理値表

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

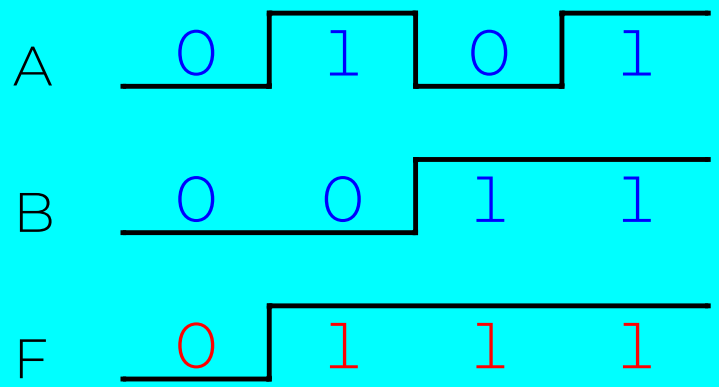
論理積は、入力値がすべて1のときに1を出力する。それ以外の入力値のときは0を出力する。

基本的な論理ゲート — 2. ORゲート

論理和



波形



論理式 $F = A + B$

Verilog HDL の表現：`F = A | B`

論理和 (OR) は、入力値にいずれか 1 が入力されたときに 1 を出力する。それ以外の入力値のときは 0 を出力する。

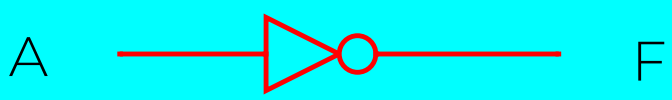
論理和は、入力値がすべて 0 のときに 0 を出力する。それ以外の入力値のときは 1 を出力する。

真理値表

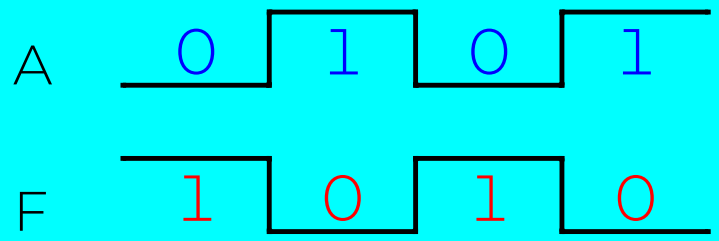
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

基本的な論理ゲート — 3. NOTゲート

否定



波形



論理式 $F = \overline{A}$

Verilog HDL の表現：`F = ~A`

否定 (NOT) は、入力された値が 1 なら 0 に、0 なら 1 に反転する。

真理値表

A	F
0	1
1	0

論理式と Verilog HDL の書き方

集合論	命題論理	論理式	Verilog HDL
積集合 $A \cap B$	連言 $A \wedge B$	論理積 $A \cdot B$	A & B
和集合 $A \cup B$	選言 $A \vee B$	論理和 $A + B$	A | B
補集合 A^c	否定 $\neg A$	否定 $\overline{A}$	~A
全体集合 U	真 T	論理 1 1	1
空集合 $\varnothing$	偽 F	論理 0 0	0

他の論理ゲート (XOR、XNOR)

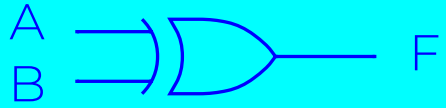
排他的論理和 (XOR)

排他的論理和 XOR　(Exclusive　　Or)

$A \quad XOR \quad B = A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$

Verilog HDL の表現：`F = A ^ B`

排他的論理和は、二つの入力値が違うとき 1 を、それ以外（入力値が同じとき）は、0 を出力する。



$A \oplus A = 0$
$A \oplus \overline{A} = 1$
$A \oplus 0 = A$
$A \oplus 1 = \overline{A}$

XOR　真理値表

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

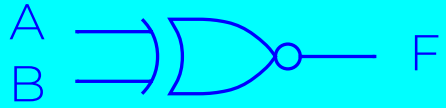
否定排他的論理和 (XNOR)

否定排他的論理和 XNOR (Exclusive Not Or)

$A \ \mathrm{XNOR} \ B = A \odot B = \overline{A} \cdot \overline{B} + A \cdot B = \overline{A \oplus B}$

Verilog HDL の表現：`F = ~(A ^ B)`

否定排他的論理和は、二つの入力値が同じとき 1 を、
それ以外（入力値が違うとき）は、0 を出力する。



$A \odot A = 1$

$A \odot \overline{A} = 0$

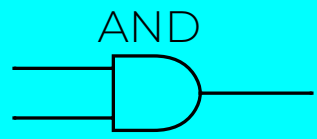
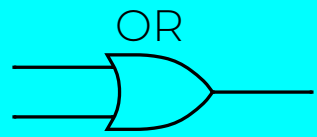
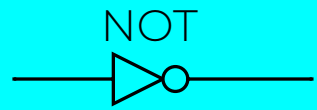
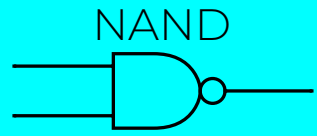
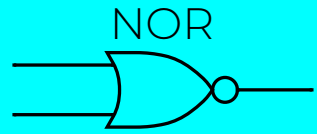
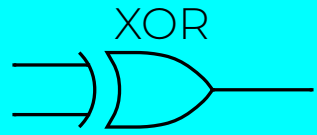
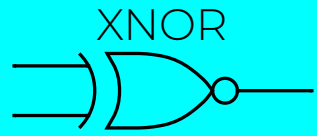
$A \odot 0 = \overline{A}$

$A \odot 1 = A$

XNOR 真理値表

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

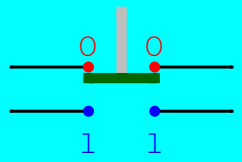
論理式と Verilog HDL の書き方

	演算	論理式	Verilog HDL	ゲート
(1)	AND:	$F = A \cdot B$	`F = A & B`	
(2)	OR:	$F = A + B$	`F = A | B`	
(3)	NOT:	$F = \overline{A}$	`F = ~A`	
(4)	NAND:	$F = \overline{A \cdot B}$	`F = ~(A & B)`	
(5)	NOR:	$F = \overline{A + B}$	`F = ~(A | B)`	
(6)	XOR:	$F = A \oplus B$	`F = A ^ B`	
(7)	XNOR:	$F = \overline{A \oplus B}$	`F = ~(A ^ B)`	

論理演算の理解

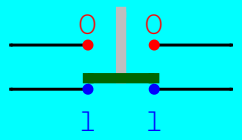
電球を点灯回路

論理演算を理解（AND）



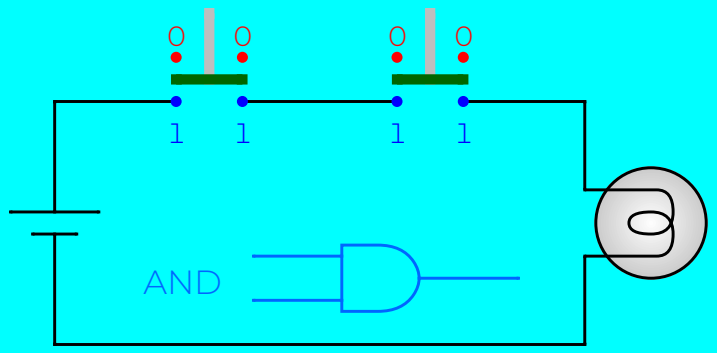
0: Switch Off

論理 0 （偽）



1: Switch On

論理 1 （真）



0 ・ 0 ＝ 0　　偽（消灯）

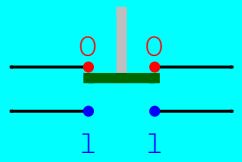
0 ・ 1 ＝ 0　　偽（消灯）

1 ・ 0 ＝ 0　　偽（消灯）

1 ・ 1 ＝ 1 ……真　かつ　真＝真（点灯）

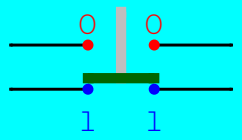
論理積は、入力値がすべて 1 のときに 1 を、
それ以外の入力値のときは 0 を出力する。

論理演算を理解（OR）



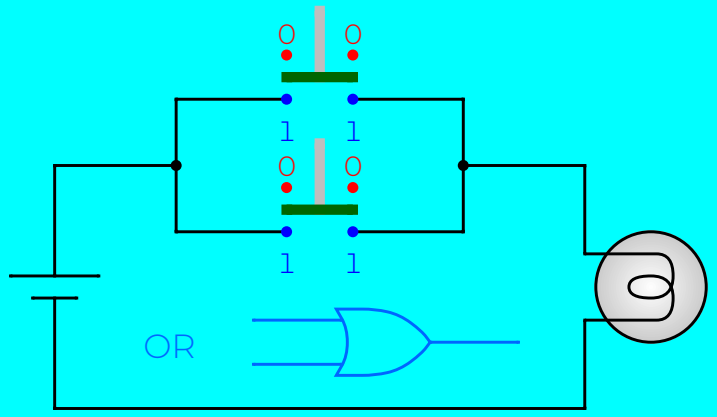
0: Switch Off

論理 0 （偽）



1: Switch On

論理 1 （真）



0 + 0 = 0 ・・・・・・・・・偽 または 偽 = 偽 （消灯）

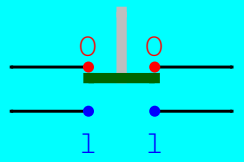
0 + 1 = 1 真 （点灯）

1 + 0 = 1 真 （点灯）

1 + 1 = 1 真 （点灯）

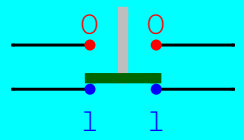
論理和は、入力値がすべて 0 のときに 0 を、
それ以外の入力値のときは 1 を出力する。

論理演算を理解（XOR）



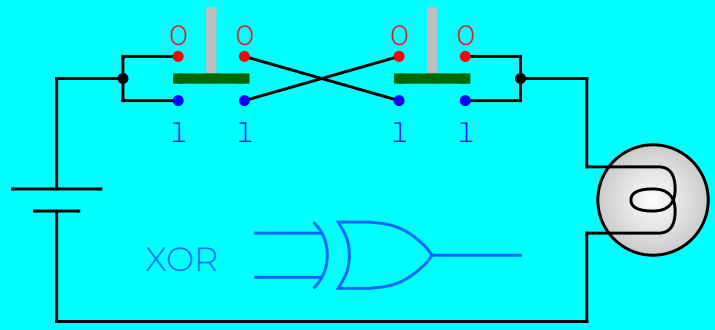
0: Switch Off

論理 0 （偽）



1: Switch On

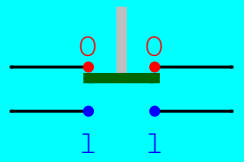
論理 1 （真）



0	⊕	0	=	0	・・・・・・・・・・・・	偽 ⊕ 偽 =	偽 （消灯）
0	⊕	1	=	1			真 （点灯）
1	⊕	0	=	1			真 （点灯）
1	⊕	1	=	0	・・・・・・・・・・・・	真 ⊕ 真 =	偽 （消灯）

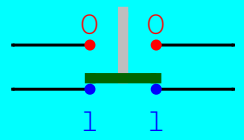
排他的論理和は、二つの入力値が違うとき 1 を、それ以外（入力値が同じとき）は、0 を出力する。

論理演算を理解（XNOR）



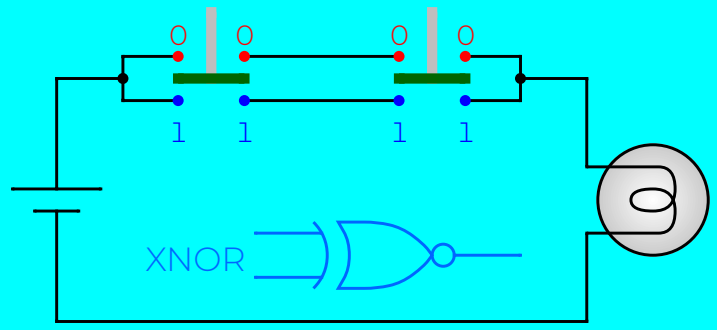
0: Switch Off

論理 0 （偽）



1: Switch On

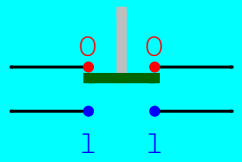
論理 1 （真）



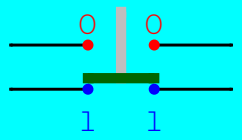
0 ⊙ 0 = 1 ・・・・・・・・・・・・	偽 ⊙ 偽 = 真	（点灯）
0 ⊙ 1 = 0	偽	（消灯）
1 ⊙ 0 = 0	偽	（消灯）
1 ⊙ 1 = 1 ・・・・・・・・・・・・	真 ⊙ 真 = 真	（点灯）

否定排他的論理和は、二つの入力値が同じとき 1 を、
それ以外（入力値が違うとき）は、0 を出力する。

論理演算を理解（NOT）

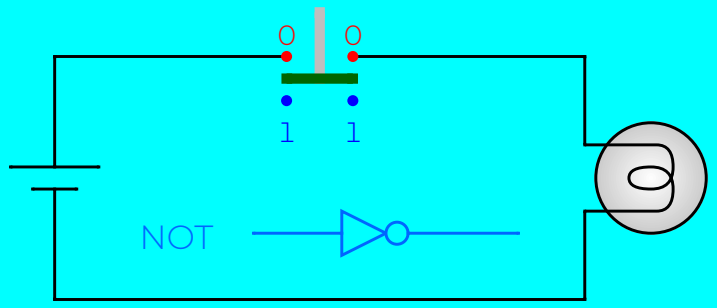


0: Switch Off



1: Switch On

論理 0 （偽）　　論理 1 （真）

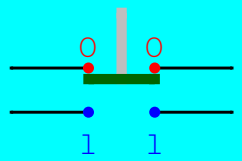


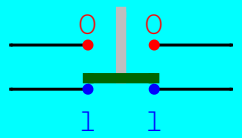
$\overline{0} = 1$ ・・・・・・・・・・・・・・・　偽 の否定 = 真（点灯）

$\overline{1} = 0$　　偽（消灯）

論理否定は、入力された値が 0 なら 1 に、
1 なら 0 に反転する。

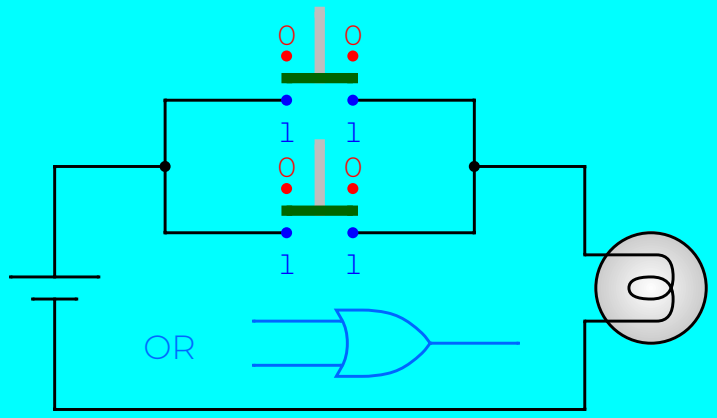
論理演算を理解（電球を点灯する）

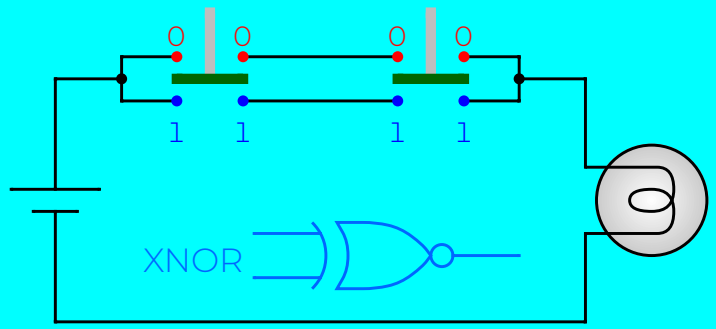


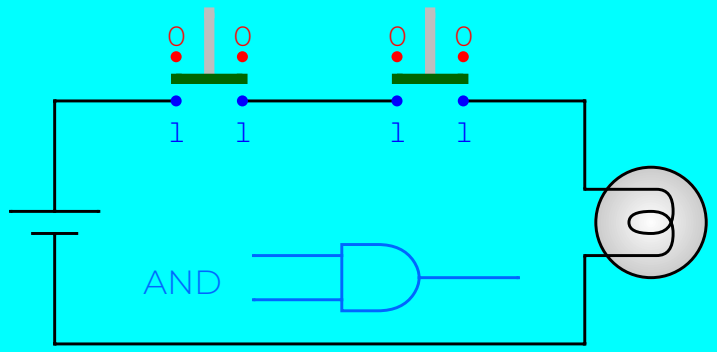


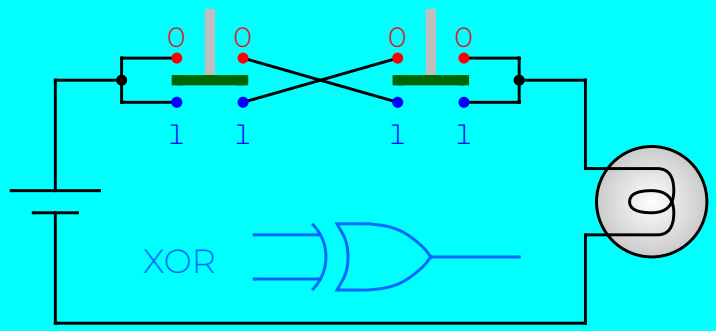
論理 0 （偽）

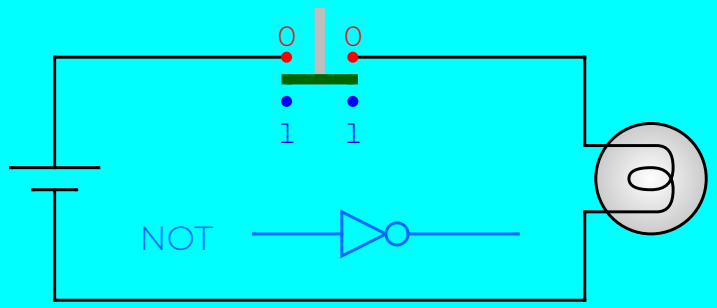
論理 1 （真）





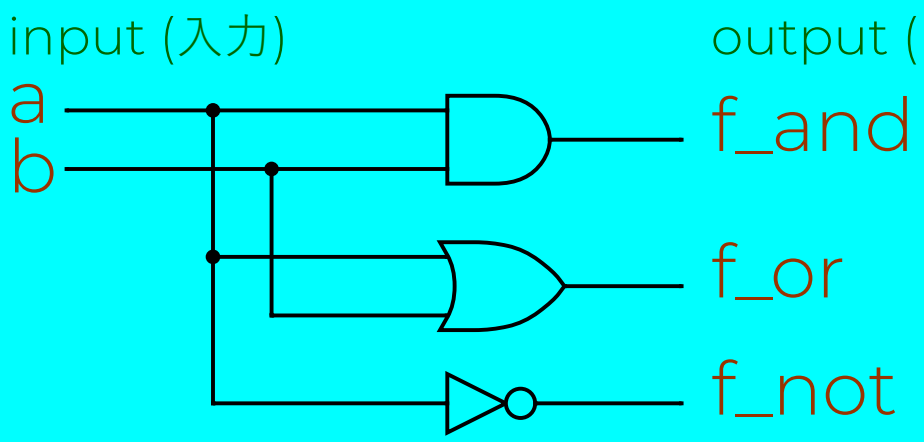






回路の例とその Verilog HDL による実装

回路と Verilog HDL による実装



$f\_and = a \cdot b$ AND

$f\_or = a + b$ OR

$f\_not = \bar{a}$ NOT

and_or_not.v

```
`timescale 1ns/1ns

module and_or_not (a, b, f_and, f_or, f_not);
    input  a, b;
    output f_and, f_or, f_not;

    assign f_and = a & b; // AND
    assign f_or  = a | b; // OR
    assign f_not = ~a;    // NOT

endmodule
-:---  and_or_not.v   All L10    (Verilog)
```

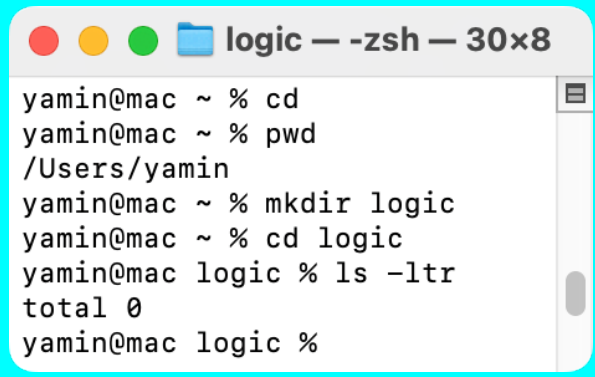
[and_or_not.v]

リンク先のファイルをダウンロード

logic フォルダを作る

Finder → アプリケーション → ユーティリティ → ターミナル

```
yamin@mac ~ % ls -l
yamin@mac ~ % mkdir logic
yamin@mac ~ % cd logic
yamin@mac logic % ls -l
yamin@mac logic % touch and_or_not.v
yamin@mac logic % open -e and_or_not.v
```



Safari ウィンドウで command + a, command + c

Text editor で command + v, command + s, command + w

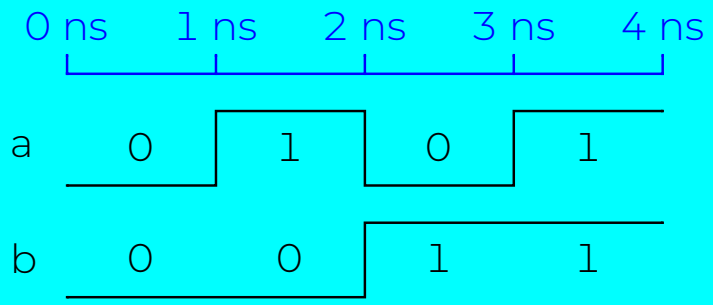
```
yamin@mac logic % ls -l
yamin@mac logic % cat and_or_not.v
```

テストベンチ — シミュレーションのため

and_or_not_tb.v

```
`timescale 1ns/1ns

module and_or_not_tb;
    reg  a, b;
    wire f_and, f_or, f_not;

    and_or_not i0 (a, b, f_and, f_or, f_not);

    initial begin
        #0 a = 0;
        #0 b = 0;
        #4 $finish;
    end

    always #1 a = ~a;
    always #2 b = ~b;

    initial begin
        $dumpfile ("and_or_not.vcd");
        $dumpvars;
    end

endmodule
-:---  and_or_not_tb.v   All L22    (Verilog)
Beginning of buffer
```

入力信号の値を指定する



and_or_not_tb.v

リンク先のファイルをダウンロード

テストベンチを作る

logic のターミナルで

```
yamin@mac logic % touch and_or_not_tb.v
yamin@mac logic % open -e and_or_not_tb.v
```

Safari ウィンドウで command + a, command + c

Text editor で command + v, command + s, command + w

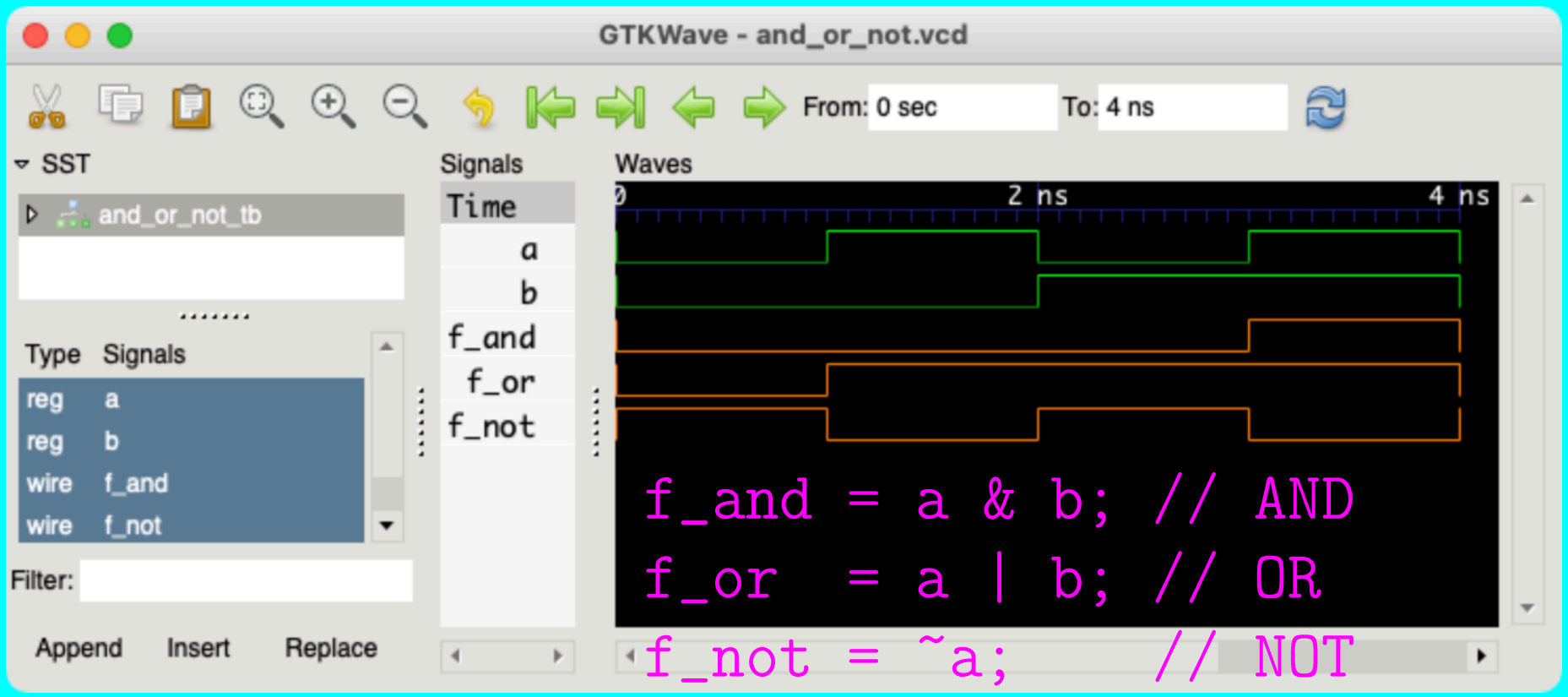
```
yamin@mac logic % ls -l
yamin@mac logic % cat and_or_not_tb.v
```

emacs がお勧めである

```
yamin@mac logic % emacs and_or_not_tb.v &
```

シミュレーションと波形

```
logic — open -n -W -a gtkwave -args -chdir — 76×5
[yamin@mac logic % iverilog -Wall -o and_or_not and_or_not_tb.v and_or_not.v ]
[yamin@mac logic % ./and_or_not                                              ]
 VCD info: dumpfile and_or_not.vcd opened for output.
[yamin@mac logic % gtkwave and_or_not.vcd                                    ]
```



iverilog と gtkwave のインストール

Homebrewのインストール

iverilog と gtkwave をインストールするには、まず Homebrew をインストールする必要がある。

Homebrew をインストールするには、次の Web ページを参照してください。

M1 MacBook Air Package Installation

そのあと、brew コマンドを使用してパッケージをインストールできる。

iverilog と gtkwave のインストール

iverilog をインストールする:

```
% brew install icarus-verilog
```

```
% iverilog -Wall -o and_or_not and_or_not_tb.v and_or_not.v
% vvp and_or_not
VCD info: dumpfile and_or_not.vcd opened for output.
```

gtkwave をインストールする:

```
% brew install --HEAD randomplum/gtkwave/gtkwave
```

```
% gtkwave and_or_not.vcd &
```

論理式と回路の例

論理式と論理回路の例 (真理値表)

論理式の例: $Y = \overline{S} \cdot A0 + S \cdot A1$

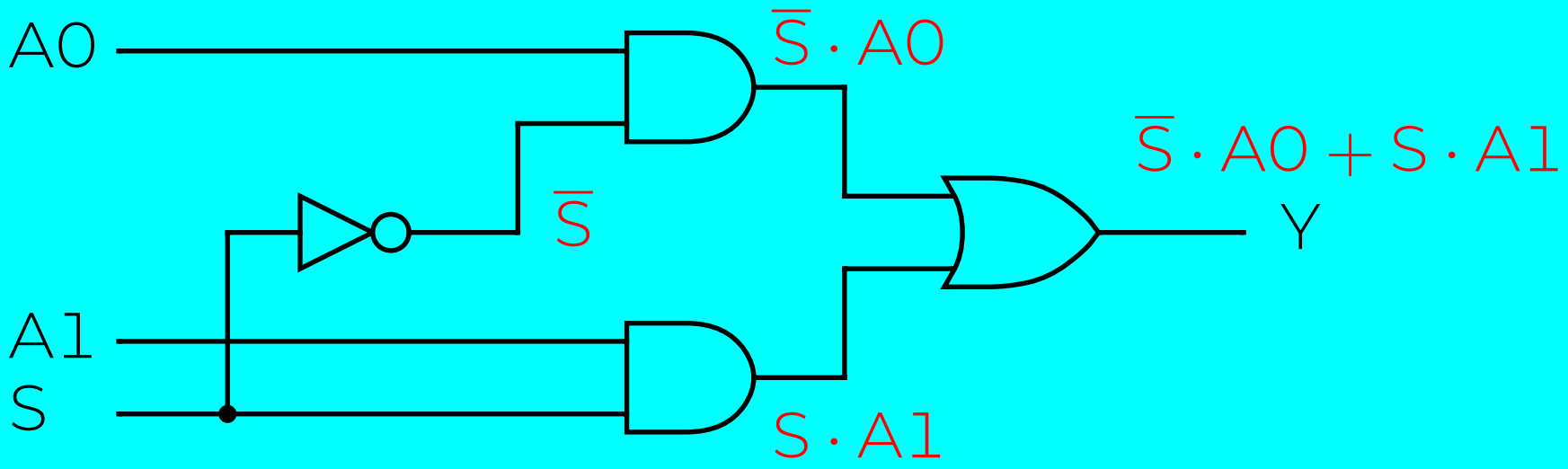
その論理式の真理値表

入力						出力
S	A1	A0	$\overline{S}$	$\overline{S} \cdot A0$	$S \cdot A1$	Y
0	0	0	1	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	0	0
0	1	1	1	1	0	1
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	0	1	1
1	1	1	0	0	1	1

論理式と論理回路の例 (回路図)

論理式の例: $Y = \overline{S} \cdot A0 + S \cdot A1$

その論理式の論理回路



優先順位: (高い) 否定 → 論理積 → 論理和 (低い)

論理式と論理回路の例 (Verilog HDL)

論理式の例: $Y = \overline{S} \cdot A0 + S \cdot A1$

その論理式の Verilog HDL による回路

```
‘timescale 1ns/1ns

module example (A0, A1, S, Y);
    input  A0, A1, S;
    output Y;

    assign Y = ~S & A0 | S & A1;

endmodule
```

優先順位: (高い) 否定 → 論理積 → 論理和 (低い)

論理式と論理回路の例 (Verilog HDL)

論理式の例: $Y = \overline{S} \cdot A0 + S \cdot A1$

その論理式の Verilog HDL による回路

example.v

```
`timescale 1ns/1ns

module example (A0, A1, S, Y);
    input  A0, A1, S;
    output Y;

    assign Y = ~S & A0 | S & A1;

endmodule
-:---  example.v      All L8     (Veril
```

[example.v](#)

論理式: $Y = \overline{S} \cdot A0 + S \cdot A1$

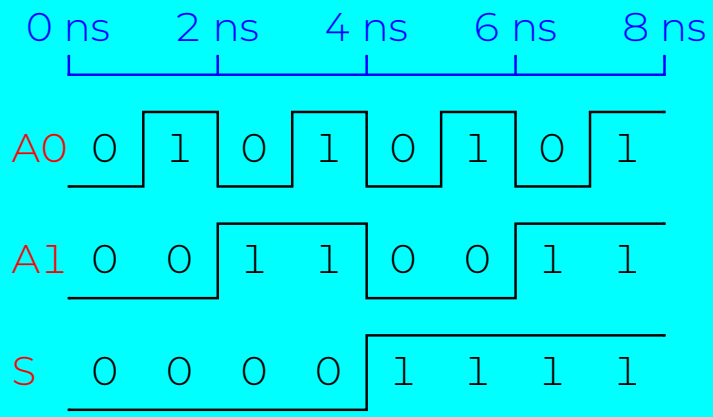
Verilog HDL: `Y = ~S & A0 | S & A1`

論理式と論理回路の例 (テストベンチ)

example_tb.v

```
`timescale 1ns/1ns

module example_tb;
    reg  A0, A1, S;
    wire Y;

    example i0 (A0, A1, S, Y);

    initial begin
        #0 A0 = 0;
        #0 A1 = 0;
        #0 S  = 0;
        #8 $finish;
    end

    always #1 A0 = ~A0;
    always #2 A1 = ~A1;
    always #4 S  = ~S;

    initial begin
        $dumpfile ("example.vcd");
        $dumpvars;
    end

endmodule
-:---  example_tb.v   All L24    (Veril
```

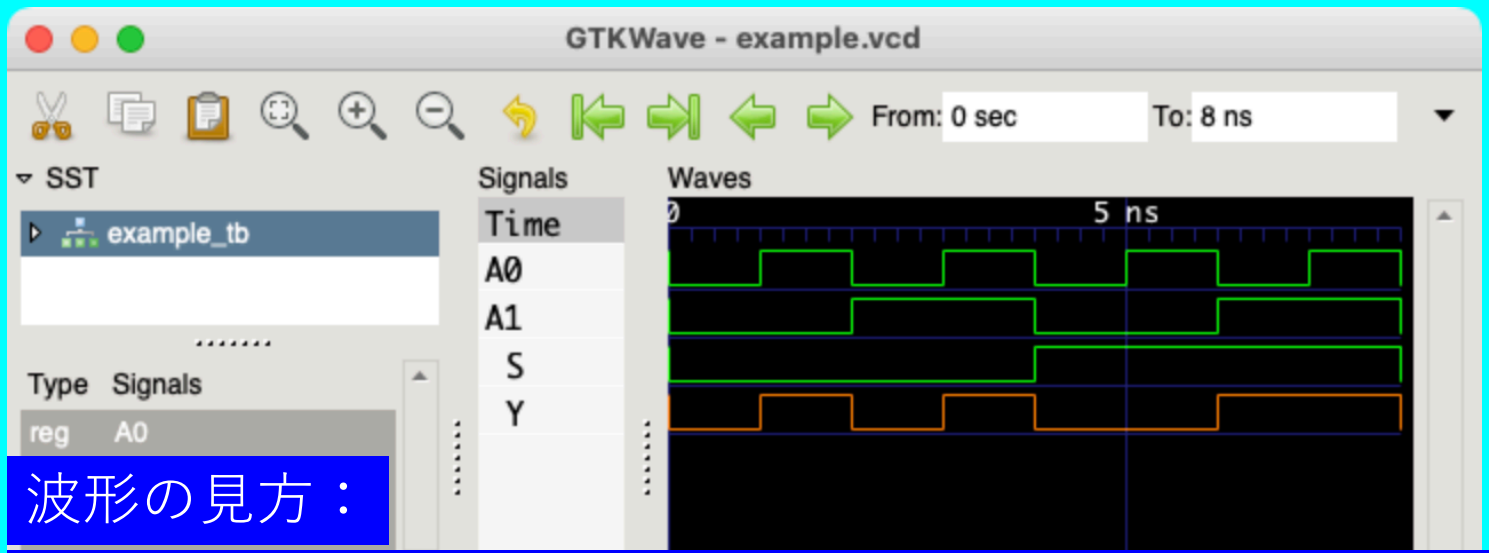
入力信号の値を指定する



example_tb.v

論理式と論理回路の例 (波形)

```
logic — open -n -W -a gtkwave -args -chdir — 76×5
yamin@mac logic % iverilog -Wall -o example example_tb.v example.v
yamin@mac logic % ./example
VCD info: dumpfile example.vcd opened for output.
yamin@mac logic % gtkwave example.vcd
```



波形の見方：

$S = 0$ の時、$Y = \overline{S} \cdot A0 + S \cdot A1 = 1 \cdot A0 + 0 \cdot A1 = A0 + 0 = A0$

$S = 1$ の時、$Y = \overline{S} \cdot A0 + S \cdot A1 = 0 \cdot A0 + 1 \cdot A1 = 0 + A1 = A1$

まとめ

論理演算と論理ゲート

まとめ

- ０と１の表現（電圧の高／低）
- 論理演算と論理ゲート
  - 基本的な論理演算と論理ゲート: AND、OR、NOT
  - ほかのゲート: NAND、NOR、XOR、XNOR
- 論理演算の理解（電球を点灯する回路）
- 真理値表、論理式と論理回路
- ハードウェア記述言語
  - Verilog HDL (Hardware Description Language)
- シミュレーションのためのテストベンチ (Test bench)
- iverilog と gtkwave

課題

課題 I (100 点)

問題 1 ：排他的論理和 XOR：$A \oplus B = \overline{A} \cdot B + A \cdot \overline{B}$
問題：$A \oplus B \oplus C$ の真理値表を完成して下さい。
注意：$A \oplus B \oplus C = (A \oplus B) \oplus C$

入力								出力
A	B	C	$\overline{A}$	$\overline{B}$	$\overline{A} \cdot B$	$A \cdot \overline{B}$	$A \oplus B$	$A \oplus B \oplus C$
0	0	0	1	1				
0								
0								
0								
1								
1								
1								
1	1	1	0	0				

課題 I (100 点)

問題 2 ：iverilog と gtkwave を使用して、問題 1 の Verilog HDL 回路をシミュレーションし、波形を問題 1 の真理値表と比較してください。 用意したものは、(1) 回路 xor3.v、(2) テストベンチ xor3_tb.v。 実行：

```
iverilog -Wall -o xor3 xor3_tb.v xor3.v
vvp xor3
gtkwave xor3.vcd &
```

レポート pdf：logic-01-24k0000-法政花子.pdf
レポート tex：logic-01-24k0000-法政花子.tex

TeX のインストール：Mac Package Installation

課題 I (100 点)

If you cannot install gtkwave, please use the testbench xor3_testbench.v. Then the inputs and output can be shown as follows.

```
iverilog -Wall -o xor3 xor3_testbench.v xor3.v
vvp xor3
VCD info: dumpfile xor3.vcd opened for output.
time      a   b   c   f
 0 ns:    0   0   0   0
 1 ns:    0   0   1   1
 2 ns:    0   1   0   1
 3 ns:    0   1   1   0
 4 ns:    1   0   0   1
 5 ns:    1   0   1   0
 6 ns:    1   1   0   0
 7 ns:    1   1   1   1
 8 ns:    0   0   0   0
```

Windows PC

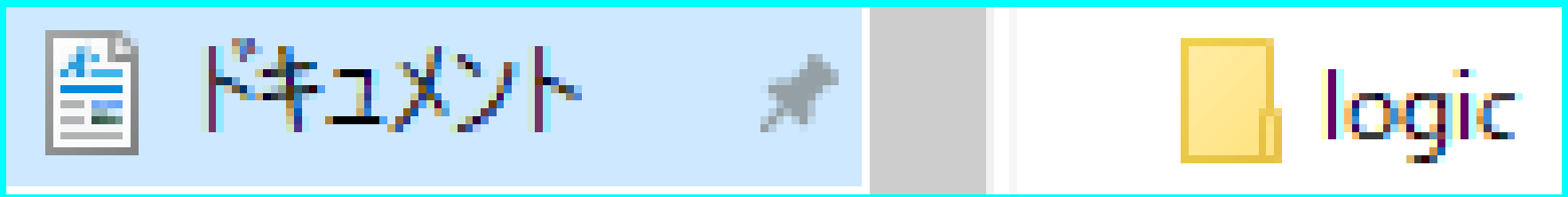
でのやり方

1. 作業フォルダーを作成する

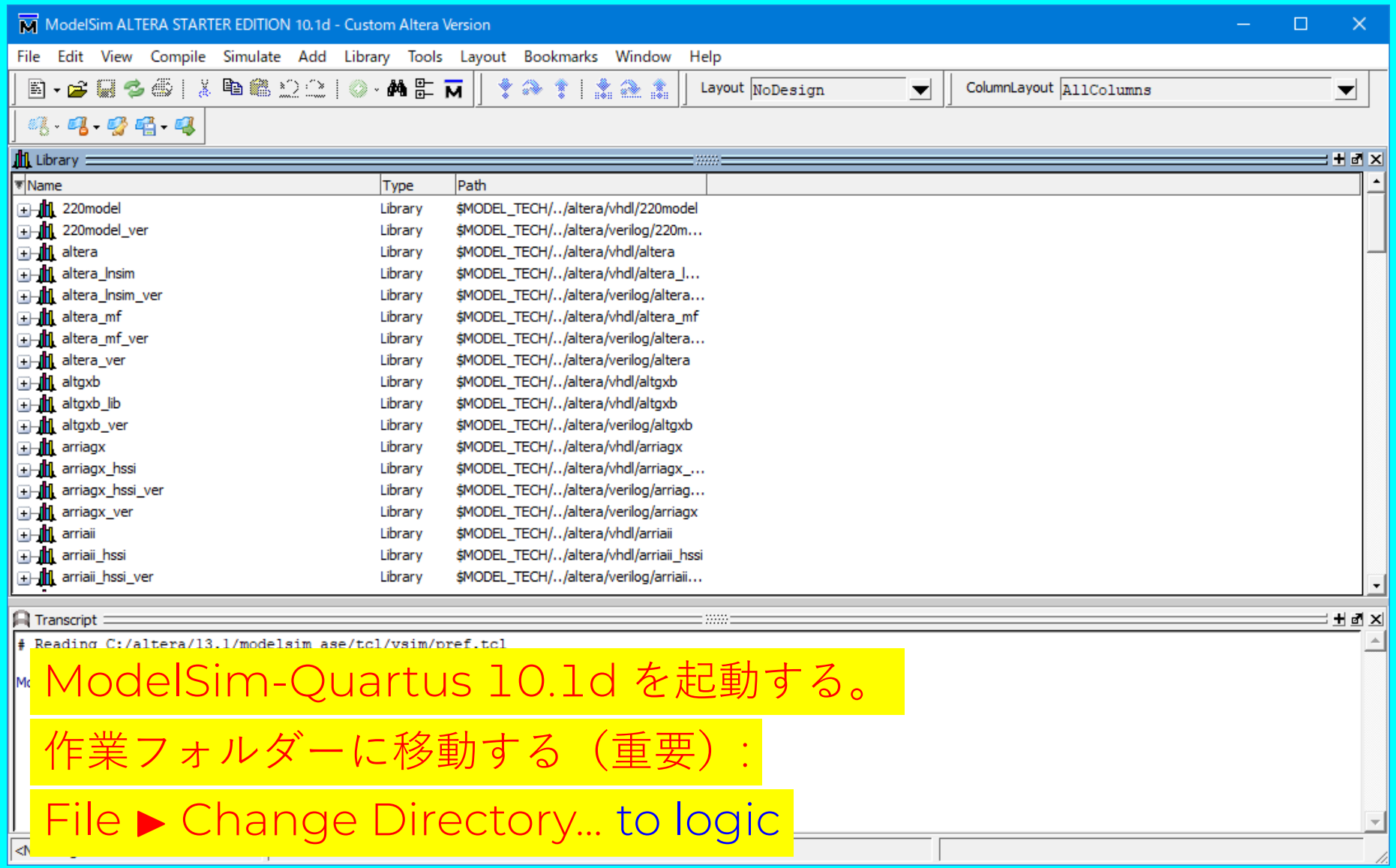
まず最初に、作業フォルダーを作成する。例えば

C:\Users\your_account\Documents\logic

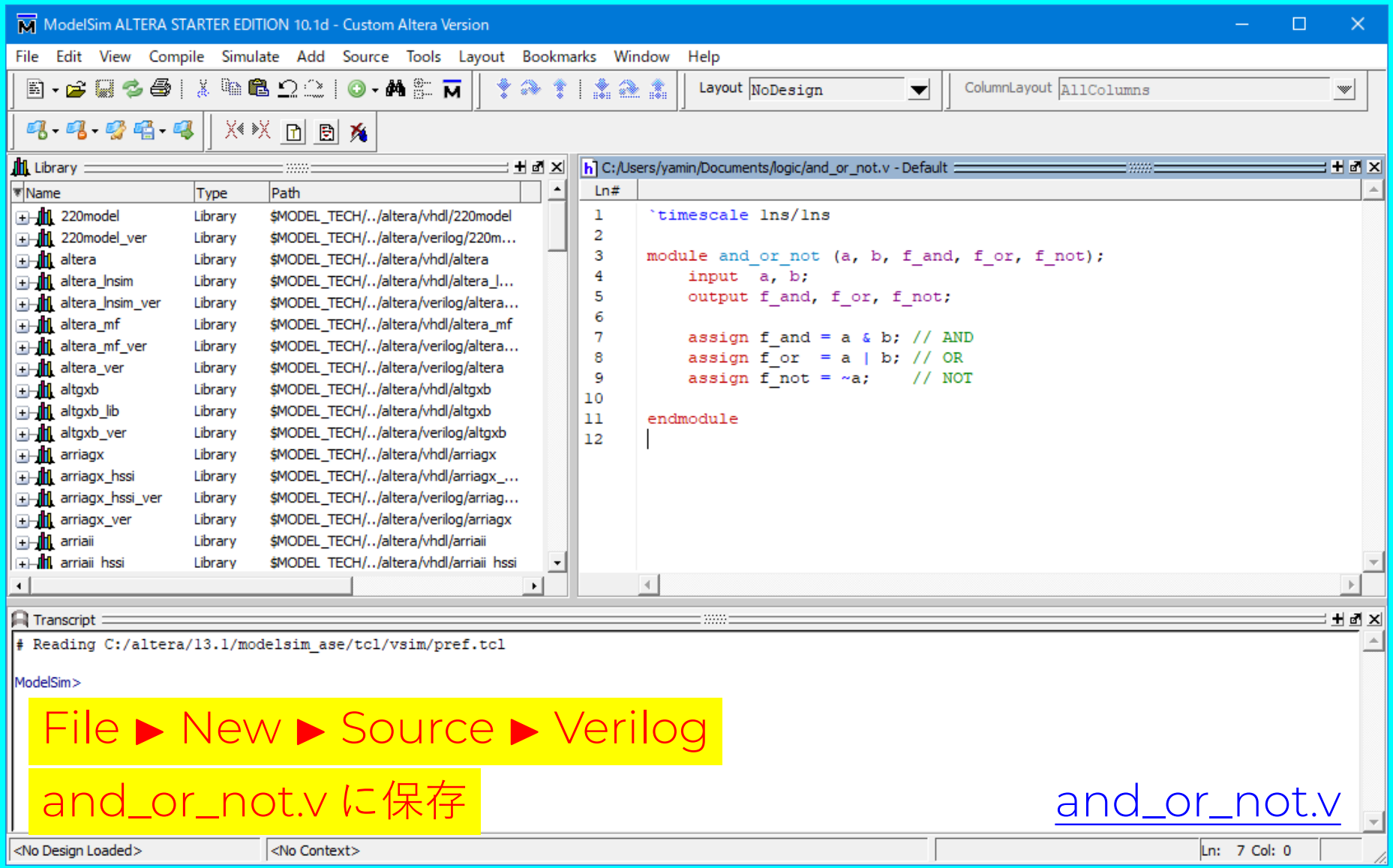
これからのすべてのデザインはこの作業フォルダーで行われなければならない。



2. ModelSim を起動



3. Verilog HDLファイルを入力

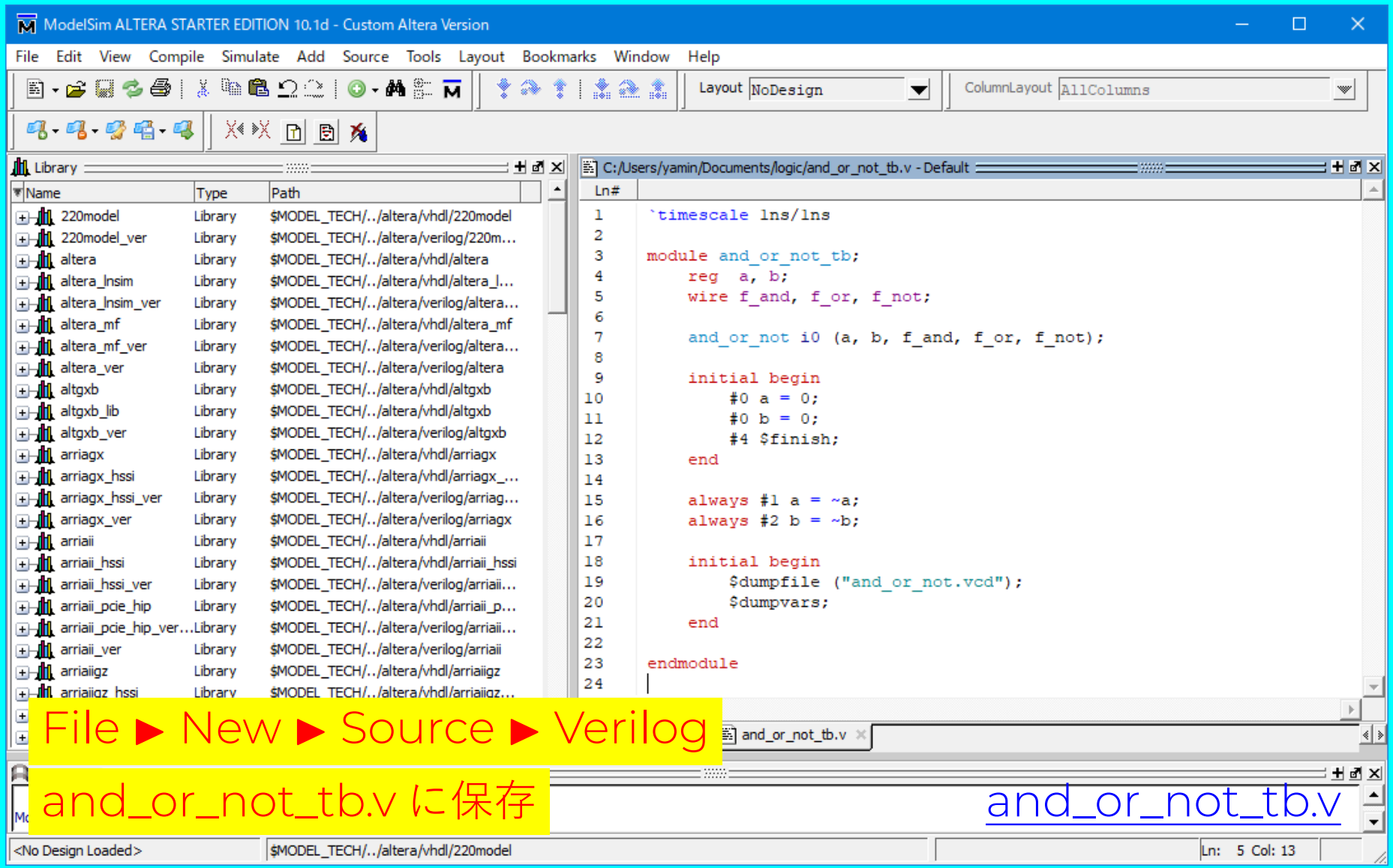


File ▶ New ▶ Source ▶ Verilog

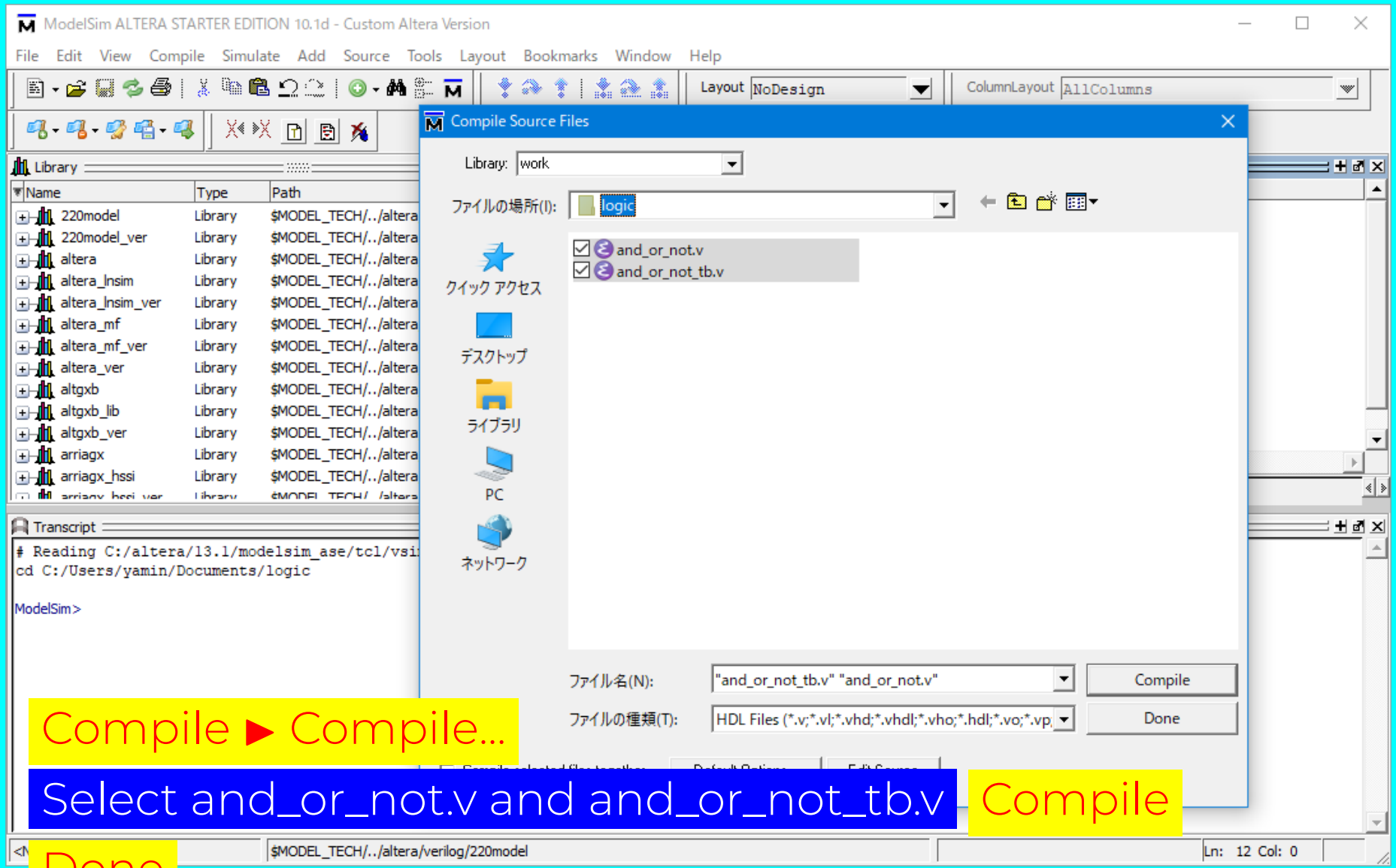
and_or_not.v に保存

and_or_not.v

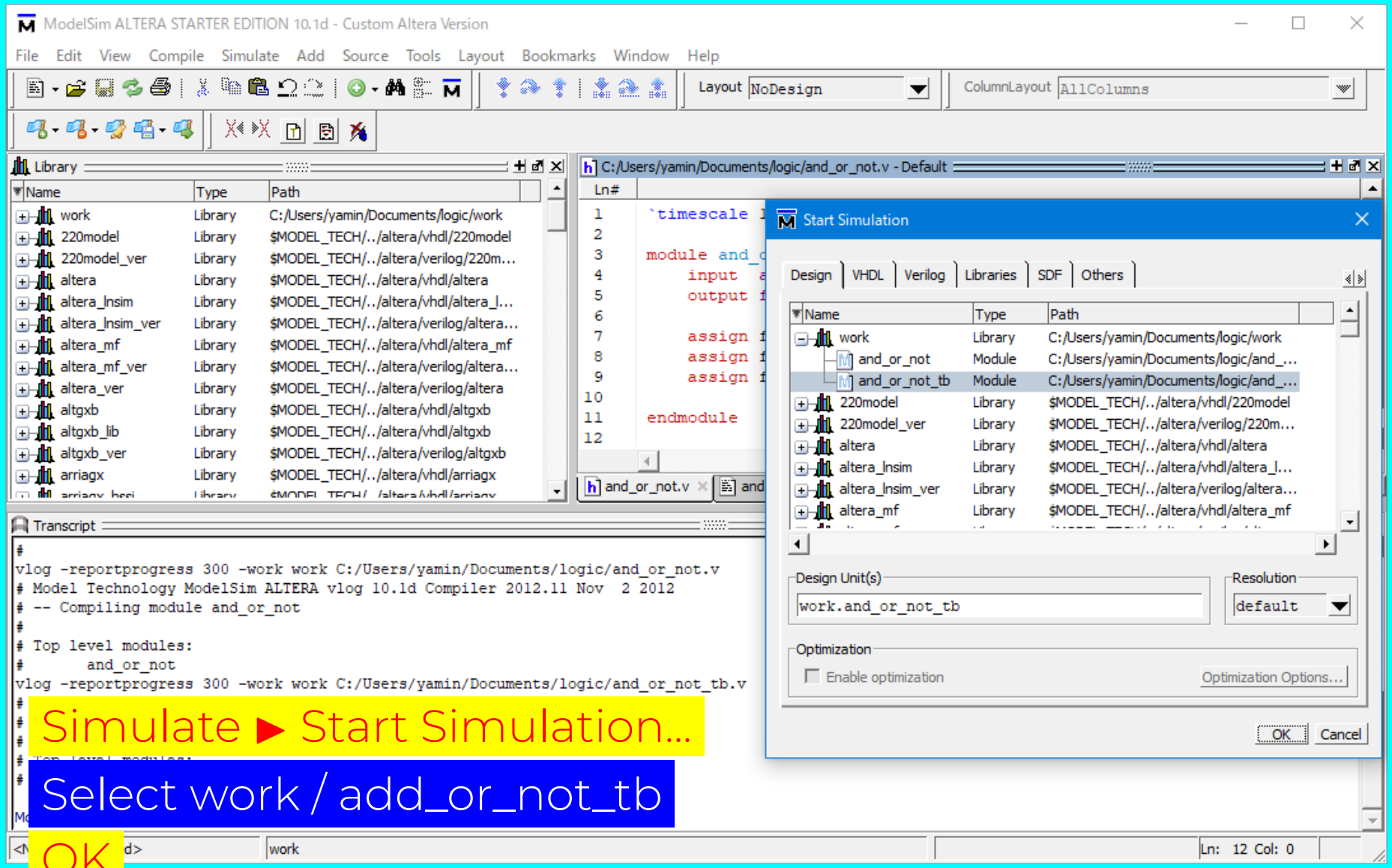
4. テストベンチを入力


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Source Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Name Type Path
220model Library $MODEL_TECH/../altera/vhdl/220model
220model_ver Library $MODEL_TECH/../altera/verilog/220m...
altera Library $MODEL_TECH/../altera/vhdl/altera
altera_lnsim Library $MODEL_TECH/../altera/vhdl/altera_l...
altera_lnsim_ver Library $MODEL_TECH/../altera/verilog/altera...
altera_mf Library $MODEL_TECH/../altera/vhdl/altera_mf
altera_mf_ver Library $MODEL_TECH/../altera/verilog/altera...
altera_ver Library $MODEL_TECH/../altera/verilog/altera
altgxb Library $MODEL_TECH/../altera/vhdl/altgxb
altgxb_lib Library $MODEL_TECH/../altera/vhdl/altgxb
altgxb_ver Library $MODEL_TECH/../altera/verilog/altgxb
arriagx Library $MODEL_TECH/../altera/vhdl/arriagx
arriagx_hssi Library $MODEL_TECH/../altera/vhdl/arriagx_...
arriagx_hssi_ver Library $MODEL_TECH/../altera/verilog/arriag...
arriagx_ver Library $MODEL_TECH/../altera/verilog/arriagx
arriaii Library $MODEL_TECH/../altera/vhdl/arriaii
arriaii_hssi Library $MODEL_TECH/../altera/vhdl/arriaii_hssi
arriaii_hssi_ver Library $MODEL_TECH/../altera/verilog/arriaii...
arriaii_pcie_hip Library $MODEL_TECH/../altera/vhdl/arriaii_p...
arriaii_pcie_hip_ver... Library $MODEL_TECH/../altera/verilog/arriaii...
arriaii_ver Library $MODEL_TECH/../altera/verilog/arriaii
arriaiigz Library $MODEL_TECH/../altera/vhdl/arriaiigz
C:/Users/yamin/Documents/logic/and_or_not_tb.v - Default
Ln#
1 `timescale 1ns/1ns
2
3 module and_or_not_tb;
4 reg a, b;
5 wire f_and, f_or, f_not;
6
7 and_or_not i0 (a, b, f_and, f_or, f_not);
8
9 initial begin
10 #0 a = 0;
11 #0 b = 0;
12 #4 $finish;
13 end
14
15 always #1 a = ~a;
16 always #2 b = ~b;
17
18 initial begin
19 $dumpfile ("and_or_not.vcd");
20 $dumpvars;
21 end
22
23 endmodule
24
and_or_not_tb.v
File ▶ New ▶ Source ▶ Verilog
and_or_not_tb.v に保存
and_or_not_tb.v
<No Design Loaded>
$MODEL_TECH/../altera/vhdl/220model
Ln: 5 Col: 13

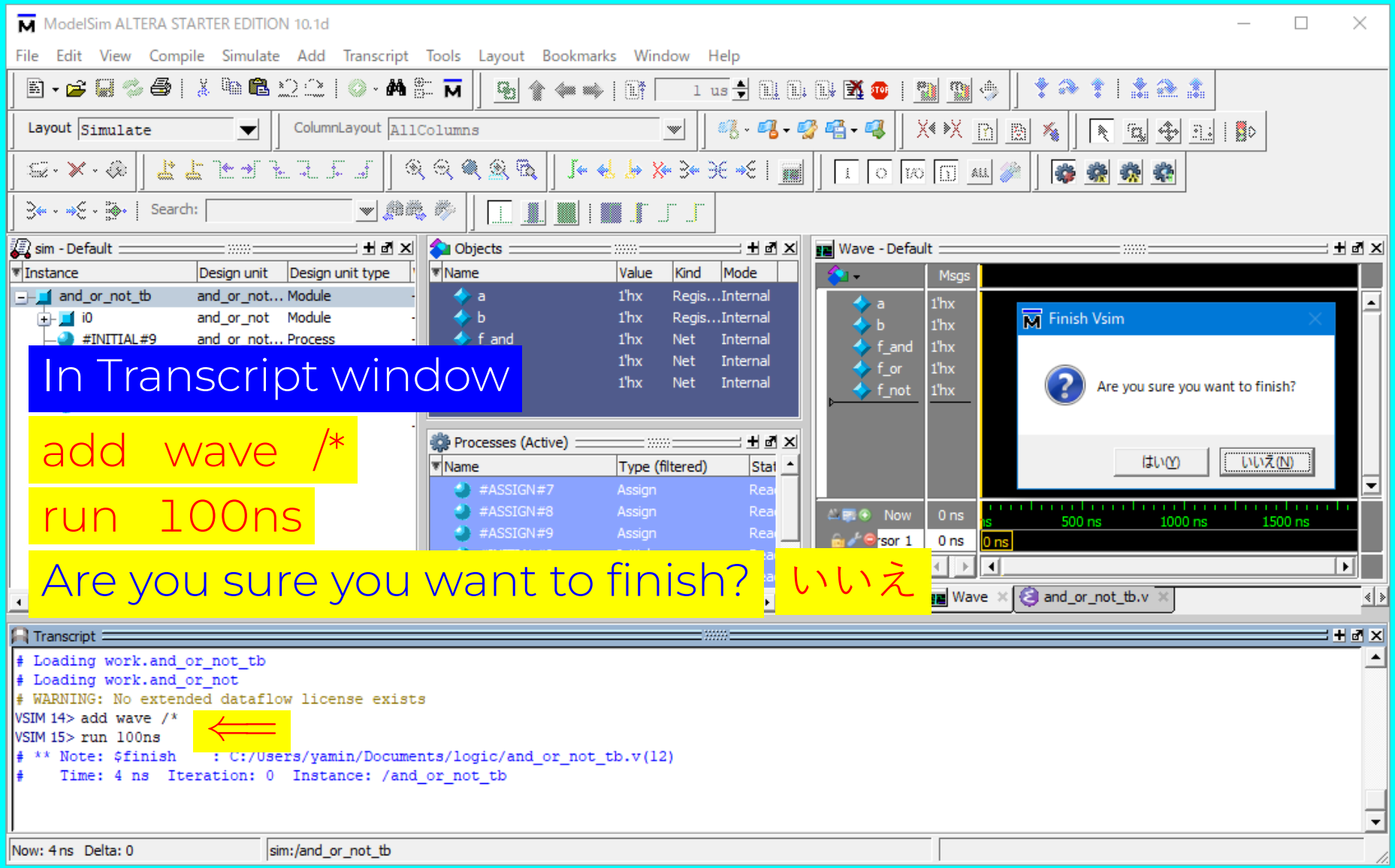
5. 論理合成


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Source Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Name Type Path
220model Library $MODEL_TECH/../altera
220model_ver Library $MODEL_TECH/../altera
altera Library $MODEL_TECH/../altera
altera_lnsim Library $MODEL_TECH/../altera
altera_lnsim_ver Library $MODEL_TECH/../altera
altera_mf Library $MODEL_TECH/../altera
altera_mf_ver Library $MODEL_TECH/../altera
altera_ver Library $MODEL_TECH/../altera
altgxb Library $MODEL_TECH/../altera
altgxb_lib Library $MODEL_TECH/../altera
altgxb_ver Library $MODEL_TECH/../altera
arriagx Library $MODEL_TECH/../altera
arriagx_hssi Library $MODEL_TECH/../altera
Transcript
Reading C:/altera/13.1/modelsim_ase/tcl/vsi
cd C:/Users/yamin/Documents/logic
ModelSim>
Compile Source Files
Library: work
ファイルの場所(I): logic
and_or_not.v
and_or_not_tb.v
クイック アクセス
デスクトップ
ライブラリ
PC
ネットワーク
ファイル名(N): "and_or_not_tb.v" "and_or_not.v"
Compile
ファイルの種類(T): HDL Files (*.v;*.vl;*.vhd;*.vhdl;*.vho;*.hdl;*.vo;*.vp
Done
Compile ▶ Compile...
Select and_or_not.v and and_or_not_tb.v
Compile
Done
$MODEL_TECH/../altera/verilog/220model
Ln: 12 Col: 0

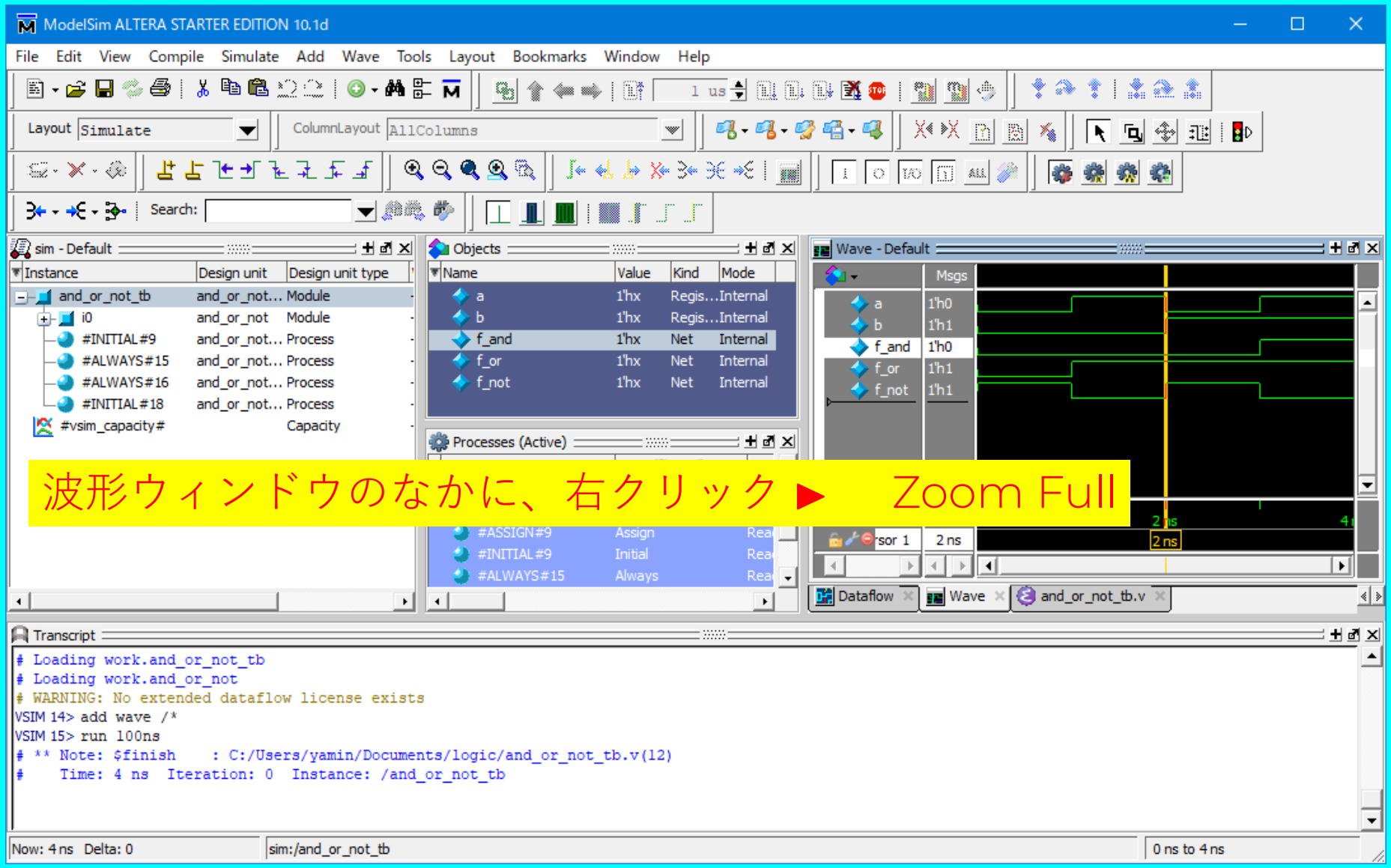
6. シミュレーション


ModelSim ALTERA STARTER EDITION 10.1d - Custom Altera Version
File Edit View Compile Simulate Add Source Tools Layout Bookmarks Window Help
Library
Transcript
C:/Users/yamin/Documents/logic/and_or_not.v - Default
Start Simulation
Design VHDL Verilog Libraries SDF Others
Design Unit(s)
work.and_or_not_tb
Resolution
default
Optimization
Enable optimization
Optimization Options...
OK
Cancel
Simulate ▶ Start Simulation...
Select work / add_or_not_tb
OK

7. コマンドを実行


In Transcript window
add wave /*
run 100ns
Are you sure you want to finish?
いいえ
Finish Vsim
Are you sure you want to finish?
はい(Y)
いいえ(N)
Loading work.and_or_not_tb
Loading work.and_or_not
WARNING: No extended dataflow license exists
VSIM 14> add wave /*
VSIM 15> run 100ns
** Note: $finish : C:/Users/yamin/Documents/logic/and_or_not_tb.v(12)
Time: 4 ns Iteration: 0 Instance: /and_or_not_tb

8. 波形を確認



波形ウィンドウのなかに、右クリック ▶ Zoom Full