

論理回路入門（１）

論理演算と論理ゲート

李 亜民

2023 年 9 月 26 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、リプルキャリーアダダー、キャリールックアヘッドアダダー、ツリー型桁上げ先見加算器、減算器、乗算器、ウォレス ツリー乗算器、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

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、期待どお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どおりの結果を生成できるかどうかを検証する必要がある。
- 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

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)

(Verilog HDL の表現)

『例』 $F = A \cdot B = A \ B$

($F = A \ \& \ B$)

0 . 0 = 0 偽

0 . 1 = 0 偽

1 . 0 = 0 偽

1 . 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)

$(F = A \mid B)$

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

(Verilog HDL の表現)

『例』 $F = \overline{A}$

($F = \sim A$)

$\overline{0} = 1$ 偽の否定 = 真

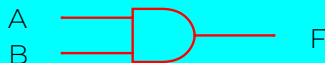
$\overline{1} = 0$ 真の否定 = 偽

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。

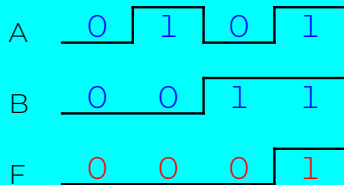
基本的な 論理ゲート (AND、OR、NOT)

基本的な論理ゲート — 1. ANDゲート

論理積



波形



論理式 $F = A \cdot B = A B$

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。

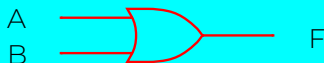
論理積は、入力値がすべて1のときに1を出力する。それ以外の入力値のときは0を出力する。

真理値表

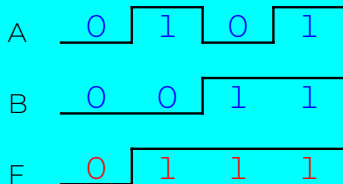
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

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: $F = A \mid B$

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。

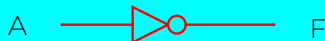
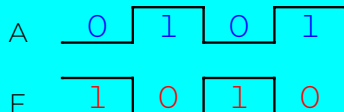
真理値表

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 $F = \bar{A}$

Verilog HDLの表現: $F = \sim A$

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}	$\sim A$
全体集合 U	真 T	論理 1 1	1
空集合 \emptyset	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)

$$A \text{ XOR } B = A \oplus B = \bar{A} \cdot B + A \cdot \bar{B}$$

Verilog HDL の表現: $F = A \wedge B$

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。



$$A \oplus A = 0$$

$$A \oplus \bar{A} = 1$$

$$A \oplus 0 = A$$

$$A \oplus 1 = \bar{A}$$

XOR 真理值表		
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)

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

Verilog HDL の表現: $F = \sim(A \oplus B)$

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

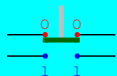
A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

論理式と Verilog HDL の書き方

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

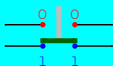
論理演算の理解 電球を点灯回路

論理演算を理解 (AND)



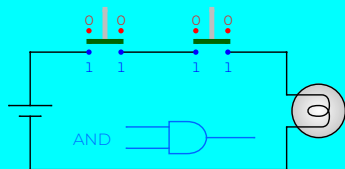
0: Switch Off

論理 0 (偽)



1: Switch On

論理 1 (真)



$$0 \cdot 0 = 0$$

偽 (消灯)

$$0 \cdot 1 = 0$$

偽 (消灯)

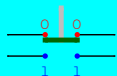
$$1 \cdot 0 = 0$$

偽 (消灯)

$$1 \cdot 1 = 1 \dots\dots\dots \text{真} \text{ かつ } \text{真} = \text{真} \text{ (点灯)}$$

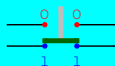
論理積は、入力値がすべて1のときに1を、それ以外の入力値のときは0を出力する。

論理演算を理解 (OR)



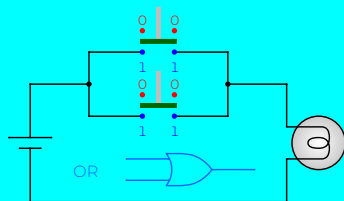
0: Switch Off

論理 0 (偽)



1: Switch On

論理 1 (真)



0 + 0 = 0 偽 または 偽 = 偽 (消灯)

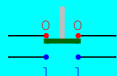
0 + 1 = 1 真 (点灯)

1 + 0 = 1 真 (点灯)

1 + 1 = 1 真 (点灯)

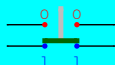
論理和は、入力値がすべて 0 のときに 0 を、
それ以外の入力値のときは 1 を出力する。

論理演算を理解 (XOR)



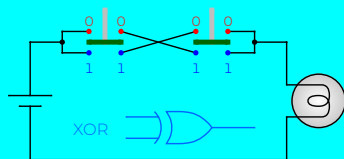
0: Switch Off

論理 0 (偽)



1: Switch On

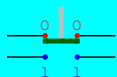
論理 1 (真)



$0 \oplus 0 = 0$	偽 \oplus 偽 = 偽 (消灯)
$0 \oplus 1 = 1$		真 (点灯)
$1 \oplus 0 = 1$		真 (点灯)
$1 \oplus 1 = 0$	真 \oplus 真 = 偽 (消灯)

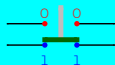
排他的論理和は、二つの入力値が違うとき 1 を、それ以外（入力値が同じとき）は、0 を出力する。

論理演算を理解 (XNOR)



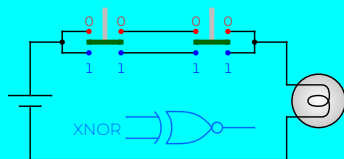
0: Switch Off

論理 0 (偽)



1: Switch On

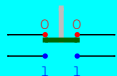
論理 1 (真)



$0 \odot 0 = 1$	偽 \odot 偽 = 真 (点灯)
$0 \odot 1 = 0$		偽 (消灯)
$1 \odot 0 = 0$		偽 (消灯)
$1 \odot 1 = 1$	真 \odot 真 = 真 (点灯)

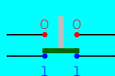
否定排他的論理和は、二つの入力値が同じとき 1 を、それ以外（入力値が違うとき）は、0 を出力する。

論理演算を理解 (NOT)



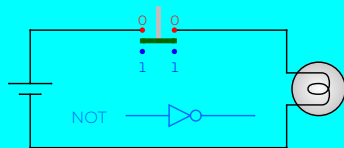
0: Switch Off

論理 0 (偽)



1: Switch On

論理 1 (真)

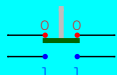


$$\overline{0} = 1 \dots\dots\dots$$
$$\overline{1} = 0$$

偽 の否定 = 真 (点灯)
真 の否定 = 偽 (消灯)

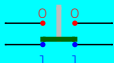
論理否定は、入力された値が 0 なら 1 に、
1 なら 0 に反転する。

論理演算を理解（電球を点灯する）



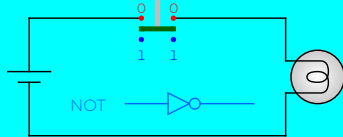
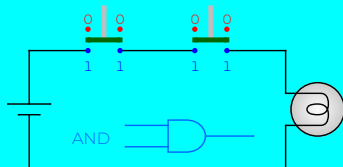
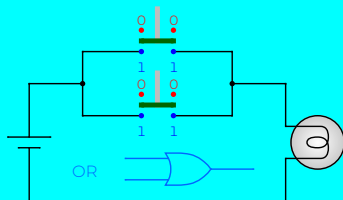
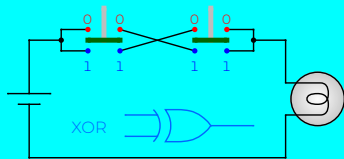
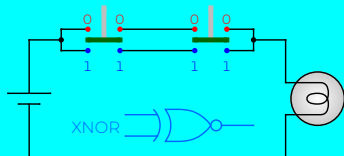
0: Switch Off

論理 0（偽）



1: Switch On

論理 1（真）

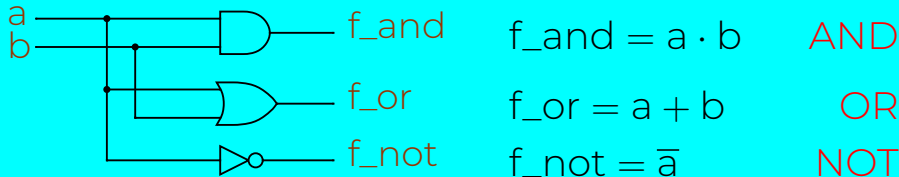


回路の例とその Verilog HDL による実装

回路と Verilog HDL による実装

input (入力)

output (出力)



```
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);

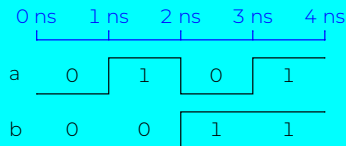
    initial begin
        #0 a = 0;
        #0 b = 0;
        #4 $finish;
    end

    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

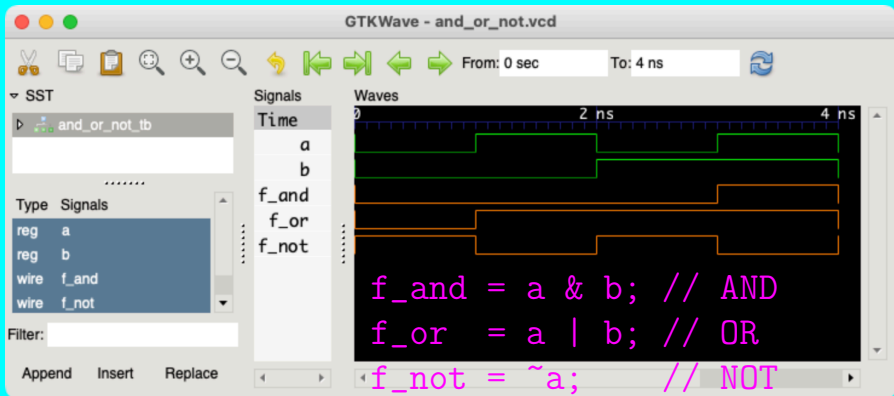
Text editor で command + v, command + s, command + w

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

シミュレーションと波形

```
logic — open -n -W -a gtkwave -args -chdir — 76x5

[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  
% ./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 --cask gtkwave  
% sudo cpan install Switch
```

Finder - Applications - **Control**-click gtkwave - Open

```
% sudo cpan install Switch
```

Finder - Applications - **Control**-click gtkwave - Open

```
% 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: $Y = \bar{S} \cdot A0 + S \cdot A1$

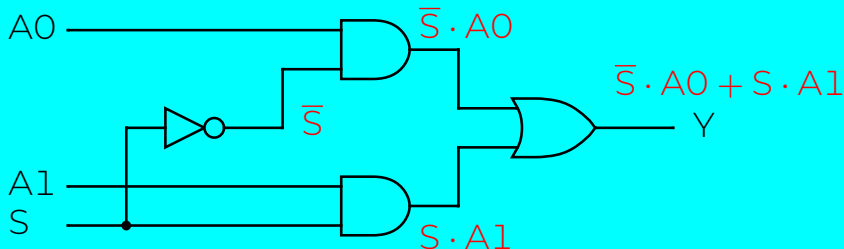
その論理式の真理値表

入力						出力
S	A1	A0	\bar{S}	$\bar{S} \cdot A0$	$S \cdot A1$	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: $Y = \bar{S} \cdot A0 + S \cdot A1$

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;

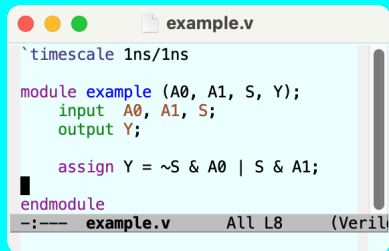
endmodule
```

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: $Y = \bar{S} \cdot A0 + S \cdot A1$

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
```

The screenshot shows a code editor window with a title bar containing three colored circles (red, yellow, green) and a file icon followed by the text 'example.v'. The editor area has a light gray background and contains the Verilog HDL code shown in the code block. The code is color-coded: backticks are blue, keywords like 'module', 'input', 'output', 'assign', and 'endmodule' are purple, and identifiers like 'example', 'A0', 'A1', 'S', 'Y' are blue. The status bar at the bottom shows 'example.v', 'All L8', and '(Verilog)'.

example.v

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

Verilog HDL: $Y = \sim S \ \& \ A0 \ | \ S \ \& \ A1$

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;

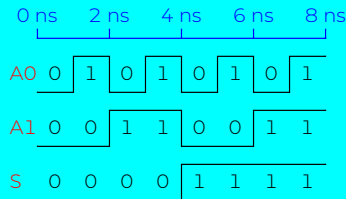
  example i0 (A0, A1, S, Y);

  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

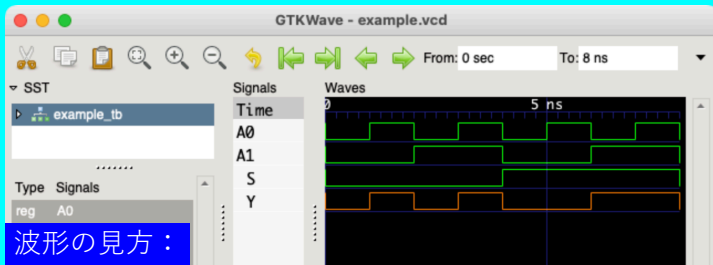


example_tb.v

論理式と論理回路の例 (波形)

```
logic — open -n -W -a gtkwave -args -chdir — 76x5

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

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

まとめ

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 : $A \oplus B = \bar{A} \cdot B + A \cdot \bar{B}$

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	\bar{A}	\bar{B}	$\bar{A} \cdot B$	$A \cdot \bar{B}$	$A \oplus B$	$A \oplus B \oplus C$
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  
./xor3  
gtkwave xor3.vcd &
```

レポート pdf : [logic-01-23k0000-法政花子.pdf](#)

レポート tex : [logic-01-23k0000-法政花子.tex](#)

TeX のインストール : [Mac Package Installation](#)

Windows PCでのやり方

Windows PC でのやり方

1. 作業フォルダーを作成する

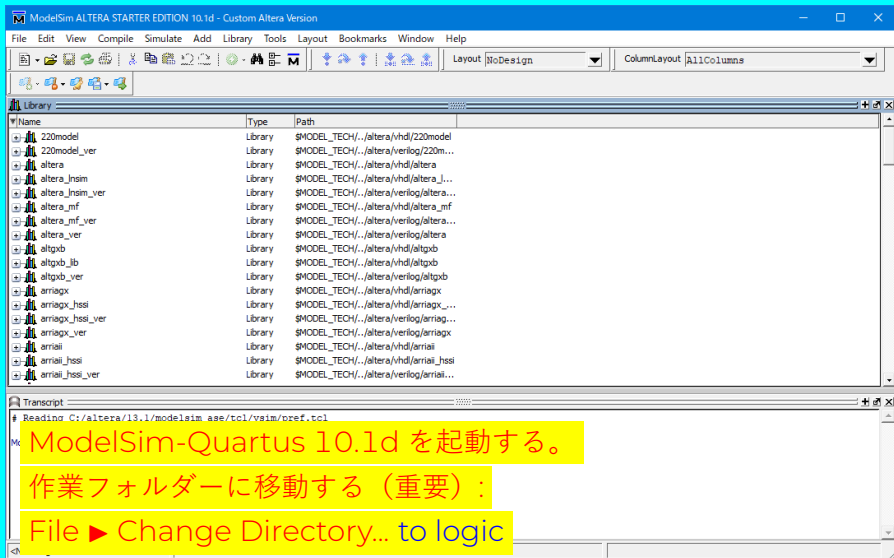
まず最初に、作業フォルダーを作成する。例えば

C:\Users\your_account\Documents\logic

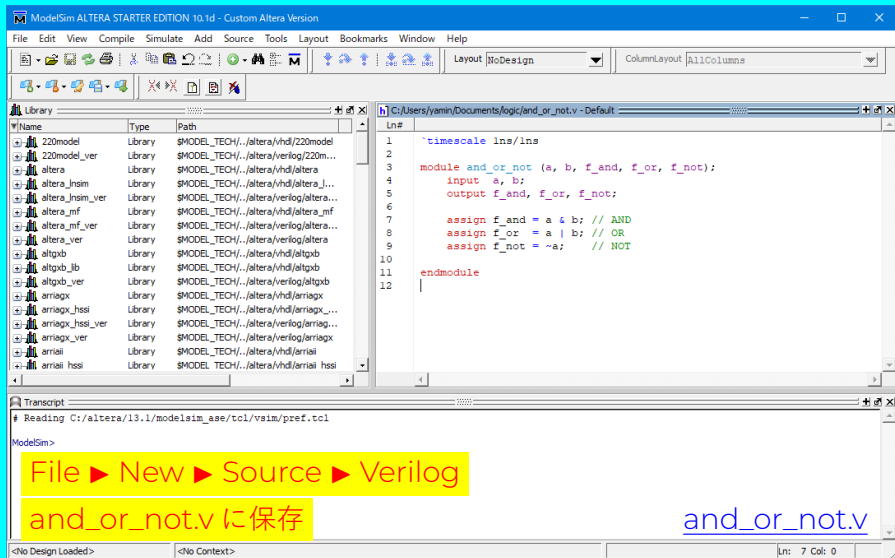
これからのすべてのデザインはこの作業フォルダーで行われなければならない。



2. ModelSim を起動



3. Verilog HDL ファイルを入力



4. テストベンチを入力

The screenshot displays the ModelSim ALTERA STARTER EDITION 10.1d interface. The top menu bar includes File, Edit, View, Compile, Simulate, Add, Source, Tools, Layout, Bookmarks, Window, and Help. The toolbar contains various icons for file operations and simulation. The left pane shows a library of components, including 220model, altera, and various Verilog models. The right pane shows the source code for the file `C:/Users/yamin/Documents/logic/and_or_not_tb.v - Default`. The code is a Verilog testbench for an AND/OR/NOT gate, defining signals `a`, `b`, `f_and`, `f_or`, and `f_not`, and performing logic operations. The status bar at the bottom indicates "<No Design Loaded>" and the current position is Ln: 5 Col: 13.

File ► New ► Source ► Verilog

and_or_not_tb.v に保存

[and_or_not_tb.v](#)

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

Compile Source Files

Library: work

ファイルの場所(I): logic

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

☒ and_or_not.v

☒ and_or_not_tb.v

Compile

Done

ファイル名(N): "and_or_not_tb.v" "and_or_not.v"

ファイルの種類(T): HDL Files (*.v;*.vlt;*.vhd;*.vhdl;*.vho;*.hdl;*.vo;*.vp)

ModelSim>

Reading C:/altera/13.1/modelsim_ase/tcl/vs...
cd C:/Users/yamin/Documents/logic

ModelSim>

Done

Compile

Select and_or_not.v and and_or_not_tb.v

Compile

Done

ModelSim>

\$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

Layout NoDesign ColumnLayout AllColumns

Library

Name	Type	Path
work	Library	C:/Users/yamin/Documents/logic/work
220model	Library	\$MODEL_TECH/./altera/vhdl/220model
220model_ver	Library	\$MODEL_TECH/./altera/verilog/220model
altera	Library	\$MODEL_TECH/./altera/vhdl/altera
altera_insim	Library	\$MODEL_TECH/./altera/vhdl/altera_insim
altera_insim_ver	Library	\$MODEL_TECH/./altera/verilog/altera_insim
altera_mf	Library	\$MODEL_TECH/./altera/vhdl/altera_mf
altera_mf_ver	Library	\$MODEL_TECH/./altera/verilog/altera_mf
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_lib	Library	\$MODEL_TECH/./altera/vhdl/arriagx

h C:/Users/yamin/Documents/logic/and_or_not.v - Default

```
1 `timescale 1ns / 1ps
2
3 module and_or_not
4   input a, b;
5   output f;
6
7   assign f = a & b;
8
9 endmodule
```

Start Simulation

Design VHDL Verilog Libraries SDF Others

Name	Type	Path
work	Library	C:/Users/yamin/Documents/logic/work
and_or_not	Module	C:/Users/yamin/Documents/logic/and_or_not.v
and_or_not_tb	Module	C:/Users/yamin/Documents/logic/and_or_not_tb.v
220model	Library	\$MODEL_TECH/./altera/vhdl/220model
220model_ver	Library	\$MODEL_TECH/./altera/verilog/220model
altera	Library	\$MODEL_TECH/./altera/vhdl/altera
altera_insim	Library	\$MODEL_TECH/./altera/vhdl/altera_insim
altera_insim_ver	Library	\$MODEL_TECH/./altera/verilog/altera_insim
altera_mf	Library	\$MODEL_TECH/./altera/vhdl/altera_mf
altera_mf_ver	Library	\$MODEL_TECH/./altera/verilog/altera_mf

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

ModelSim ALTERA STARTER EDITION 10.1d

File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help

Layout: Simulate ColumnLayout: AllColumns

Search:

sim - Default

Instance

Design unit	Design unit type
and_or_not_tb	Module
I0	Module
#INITIAL#9	Process

Objects

Name	Value	Kind	Mode
a	1hx	Regis...	Internal
b	1hx	Regis...	Internal
f_and	1hx	Net	Internal
f_or	1hx	Net	Internal
f_not	1hx	Net	Internal

Processes (Active)

Name	Type (filtered)	Status
#ASSIGN#7	Assign	Ready
#ASSIGN#8	Assign	Ready
#ASSIGN#9	Assign	Ready

Wave - Default

Msgs

Finish Vsim

Are you sure you want to finish?

はい いいえ

add wave /*

run 100ns

Are you sure you want to finish? いいえ

Transcript

```
# 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
```

Now: 4 ns Delta: 0 sim:/and_or_not_tb

8. 波形を確認

ModelSim ALTERA STARTER EDITION 10.1d

File Edit View Compile Simulate Add Wave Tools Layout Bookmarks Window Help

Layout: Simulate ColumnLayout: AllColumns

sim - Default

Instance

Instance	Design unit	Design unit type
and_or_not_tb	and_or_not... Module	Module
i0	and_or_not Module	Module
#INITIAL#9	and_or_not... Process	Process
#ALWAYS#15	and_or_not... Process	Process
#ALWAYS#16	and_or_not... Process	Process
#INITIAL#18	and_or_not... Process	Process
#vsm_capacity#	Capacity	Capacity

Objects

Name	Value	Kind	Mode
a	1hx	Regs...Internal	
b	1hx	Regs...Internal	
f_and	1hx	Net Internal	
f_or	1hx	Net Internal	
f_not	1hx	Net Internal	

Processes (Active)

Process	Mode
#ASSIGN#9	Assign
#INITIAL#9	Initial
#ALWAYS#15	Always

Wave - Default

Msgs

Signal	Value
a	1h0
b	1h1
f_and	1h0
f_or	1h1
f_not	1h1

Zoom Full

Transcript

```
# 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
```

Now: 4 ns Delta: 0 sim:/and_or_not_tb 0 ns to 4 ns