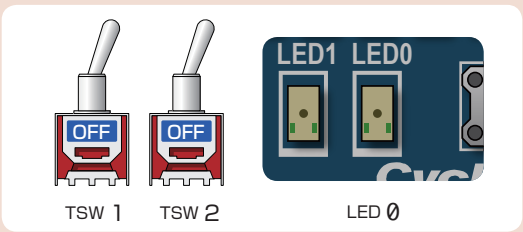
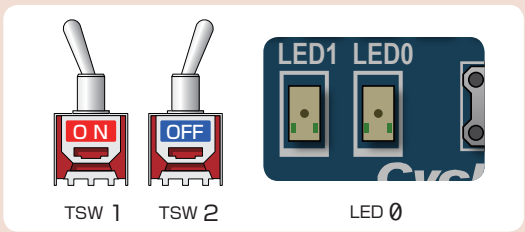
STEP 05

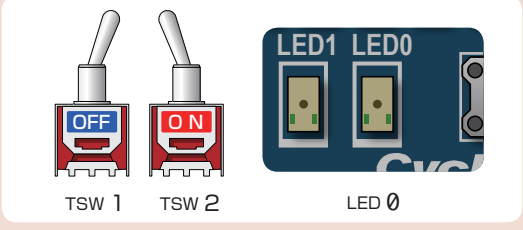
AND 回路

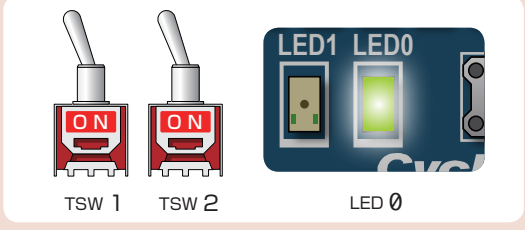
課題 05

AND 回路 – トグルスイッチ 1 とトグルスイッチ 2 の両方を ON したときだけ LED0 点灯









1. 組合せ回路の設計

組合せ回路とは

STEP 03 で設計した回路は、KEY0 が押されていないときは LED0 は消灯していて、KEY0 が押されているときだけ LED0 が点灯します。このように、同じ入力に対して出力が必ず同じになる回路を組合せ回路と言います。

組合せ回路とは記憶素子を持たない回路で、入力値にのみ依存して出力値が決定される回路です。

（組合せ回路 ⇔ 順序回路　STEP 08）

まずは、この組合せ回路を設計しながら、Verilog HDL の基礎を学んでいきましょう。

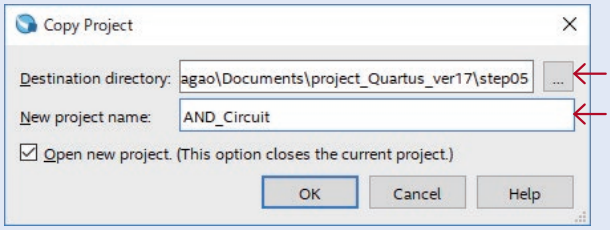
2. プロジェクトの作成

STEP 03 を参考に新しくプロジェクトを作成してください。

もしくは、右ページを参考にプロジェクトを複製してください。

Quartus Prime　プロジェクトの複製

STEP 05 のプロジェクトを、STEP 03 と同様に「New Project Wizard」で新規に作成してもかまいませんが、Quartus Prime で STEP 03 のプロジェクトを複製することもできます。
コピー元となるプロジェクト（STEP 03 の ON_Circuit.qpf）を開いた状態で、メニューから、Project ＞ Copy Project を選択します。
Copy Project ダイアログで以下のように入力します。



OK をクリックすると、ディレクトリを作成する確認ダイアログが表示されます。再度 OK をクリックするとプロジェクトが複製されます。

モジュール名を変更する場合は、同時に Top-level entity 名も変更する必要があります。
Top-level entity 名は ①②のいずれかの方法で Settings ウィンドウを開き、
① Quartus Prime のメニューから Assignment ＞ Settings を選択。
② ツールバーの「Settings」ボタン をクリック。
General の Top-level entity 欄にモジュールと同一名を入力してください。

プロジェクトコピー後の .v ファイルはリンクが切れて ON_Circuit のように表示されます。
この状態ではダブルクリックしても .v ファイルを表示することはできません。一度「文法チェック」かけるとリンクが確立し ON_Circuit のように表示されます。

コンパイルして生成される .sof 名を変更したい場合は、Project ＞ Revisions を選択し、<<new revision>> をダブルクリックし任意のリビジョン名で新規作成します。その後コンパイルすると、新規 Revision 名で .sof が作成されます。不要なリビジョンは削除してかまいません。

.v ファイル名を変更する場合は、まず、エクスプローラで .v ファイル名を変更します。
次に Quartus Prime の Settings ウィンドウを開き、Files で変更した .v ファイルを追加してください。不要な .v ファイルは Remove してかまいません。

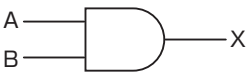
AND 回路

3. AND（論理積）回路とは

AND（論理積）とは、すべての入力が 1 のときにのみ出力が 1 であり、他の場合の出力はすべて 0 となる演算のことです。

右に示すのが２入力AND演算の回路記号となります。入力 A と B を AND 演算して、結果を X へ出力します。ディジタル回路設計では基本ゲートと呼ばれる基本素子の一種です。

AND 演算の回路記号



右に２入力の場合のAND演算の真理値表を示します。入力 A と入力 B の値を積算した結果が出力 X になります。

AND 演算の真理値表

入力 A	入力 B	出力 X
0	0	0
0	1	0
1	0	0
1	1	1

4. プログラム（回路）の記述

STEP 03 を参考に新しく Verilog ファイルを作成します。
.v ファイルに課題を実現するプログラムを記述していきます。以下に課題とサンプルを示します。エディタ画面に以下に記すサンプルを記述しましょう。

step05.v　弊社サイトに解答例ソースをご用意しています。http://www.adwin.com/product/AKE-1104.html

```
1  module AND_Circuit(TSW1, TSW2, LED0);
2
3    input TSW1, TSW2;
4    output LED0;
5    wire TSW1, TSW2, LED0;
6
7    assign LED0 = TSW1 & TSW2;
8
9  endmodule
```

これが、TSW1 と TSW2 の両方 ON したときにのみ LED0 が点灯する AND 回路を Verilog HDL で記述したものです。TSW1 と TSW2 を AND 演算した結果を LED0 に出力する回路です。

5. Verilog HDL の文法

Verilog HDL の文法を AND_Circuit.v を参考に解説していきます。
どんなに複雑な回路もこのような回路モジュールから構成されています。

> ⚠ AND_Circuit.v のようなゲート回路は、Verilog HDL ではあらかじめ基本ゲート回路として用意され、定義や宣言なしで使えます。ですから本書の例は学習用の参考として考えください。

```
1  module AND_Circuit(TSW1, TSW2, LED0); ……❶ モジュール宣言
2
3    input TSW1, TSW2;  }
4    output LED0;       }……❷ ポート宣言
5    wire TSW1, TSW2, LED0; ……❸ ネット宣言
6
7    assign LED0 = TSW1 & TSW2; ……❹ 回路記述（組合せ回路）
8
9  endmodule
```

❶ モジュール宣言　module( ) ～ endmodule

```
module 回路名（入出力ポート名）;
    …
    …
    …
    …
endmodule
```

モジュールとは回路ブロックのことを示します。プログラム（回路）の記述を行うときは必ず宣言します。モジュールの最後は必ず endmodule と記述します。

回路名の部分にプロジェクト作成時に指定した回路名（Top-Level Entity）を記述します。この名前が違っている場合、後の論理合成時にエラーが発生してしまいます。

入出力ポート名の部分には、この回路に入力される信号名と回路から出力される信号名をすべて記述しておきます。
今回の例では、TSW1、TSW2、LED0 を宣言しています。
module には必ず「;（セミコロン）」が必要ですが、endmodule には必要ありません。

> ⚠ Top-level design entity 名とモジュール名は一致していなければなりません。

AND 回路

❷ ポート宣言　input, output

```
input 入力信号名 ;
output 出力信号名 ;
```

モジュール宣言時に記述した入出力ポート名を、入力（input）と出力（output）に分けて宣言します。
モジュール宣言時に記述した入出力ポート名はすべてポート宣言を行わなければなりません。
それぞれカンマ区切りで続けて宣言できます。宣言順は問いません。
また、モジュール宣言時に存在しないポートをポート宣言することはできません。
今回の例では、TSW1 と TSW2 を入力、LED0 を出力として宣言しています。

❸ ネット宣言　wire

```
wire 信号名 ;
```

wire 宣言ではモジュール内で使用する信号線、配線を定義します。カンマ区切りで続けて宣言できます。
組合せ回路で用いる信号線などは wire 宣言を行います。なお、ポート宣言（input, output）を行ったものは自動的にネット宣言されるので省略可能です。

❹ 組合せ回路　assign

```
assign ネット型変数 = 論理式など ;
```

継続的代入を示します。電気回路の配線接続に相当するものです。
組合せ回路を記述するときには assign 文を用います。 assgin 文の左辺は、ネット型変数（ポート宣言された変数）でなければなりません。

```
assign a = b;
assign b = c;
assign c = d;
```

すべての assign 文は同時に処理される

assign 文は物理的な配線なので、記述の順番に関係なく同時に処理されます。ですから同じ出力に対して 2 つの文で assign することはできません。

❹ 回路記述内のその他の構文

ブロッキング代入文　=

```
変数 = 論理式など ;
```

「=」を使用した代入文をブロッキング代入文といいます。ブロッキング代入文が並んで記述されている場合、上の行から順番に代入が行われます。例えば、

```
b = c;
a = b;
```

は、c の値が b に代入された後、b の値が a に代入されるので a に代入されるのは c の値となります。一般的に、組合せ回路を設計する場合はブロッキング代入文を用います。P.63 参照

ビット反転　~

```
~( 変数や論理式 )
```

指定した値をビット毎に反転させます。
今回の例では、DE0-Nano ボードのプッシュキーが負論理（ON：0、OFF：1）、LED は正論理（消灯：0、点灯：1）なので、入力 KEY0 の値を反転して出力 LED0 に代入しています。

AND 演算子　&

```
入力信号 1 & 入力信号 2
```

入力信号 1 と入力信号 2 を AND 演算します。

AND 回路

Verilog HDL　記述上の注意

Verilog HDL は C 言語とよく似ています。

・大文字小文字を区別　小文字が標準
・ステートメントの終了にはセミコロン；をつける
・２行以上のステートメントは begin - end で囲む
・ダブルスラッシュ // から行末まで、及び /*　*/ で囲まれた範囲がコメント文

Verilog HDL の演算子

Verilog HDL で使える演算子は C 言語と似ています。
++（インクリメント）演算や、−−（デクリメント）演算はありません。
Verilog HDL 特有の演算子もあります。

算術演算	
＋	加算
−	減算
＊	乗算
/	除算
%	剰余
論理演算	
!	論理否定
&&	論理 AND
||	論理 OR
シフト演算	
<<	左シフト
>>	右シフト

ビット演算	
~	NOT
&	AND
|	OR
^	Ex-OR
~^	Ex-NOR
リダクション演算	
&	AND
~&	NAND
|	OR
~|	NOR
^	Ex-OR
~^	Ex-NOR

等号演算	
==	等しい
!=	等しくない
===	等しい（x,z も）
!==	等しくない
関係演算	
<	小
<=	以下
>	大きい
>=	以上
その他	
? :	条件演算
{ }	連接

6. コンパイル（論理合成）

> ⚠ 本書では、文法チェック→ピン配置→コンパイル（コンフィギュレーションファイルの生成）までをまとめて「コンパイル（論理合成）」と総称しています。

文法チェック
STEP 03 と同様に Analysis & Synthesis を行い文法チェックを行ってください。

ピン配置（配置結線）
STEP 03 と同様にピン配置を行ってください。STEP 05 のピン配置例は以下のようになります。

ノード名	ピン番号	パーツ名
LED0	PIN_A15	LED [0]
TSW1	PIN_A6	トグルスイッチ 1
TSW2	PIN_B7	トグルスイッチ 2

コンパイル（コンフィギュレーションファイルの生成）
ピン配置が終わったら、STEP 03 と同様にコンパイルを行ってください。

7. コンフィギュレーションファイル転送

STEP 03 と同様に FPGA に .sof を転送してください。転送したら、動作を確認してみましょう。
トグルスイッチ 1 とトグルスイッチ 2 の両方を ON すると LED0 が点灯します。

Quartus Prime　回路図の表示

Quartus Prime では、VerilogHDL で記述したプログラム（回路）を回路図形式で表示確認することができます。
Tool ＞ Netlist Viewers ＞ RTL Viewer を実行します。