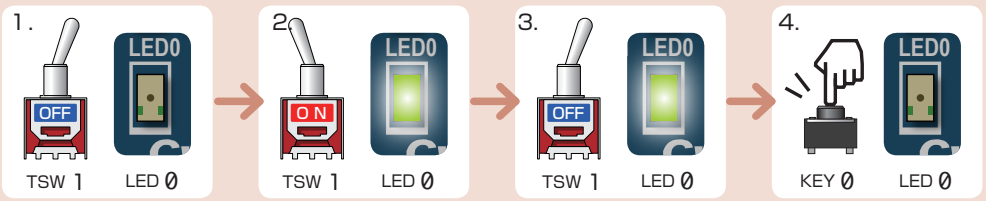
課題 08-1

以下の動作を行うフリップフロップ回路を作成してください

1．TSW1 が OFF の状態で、LED0 は消灯.
2．TSW1 を ON すると、LED0 が点灯.
3．TSW1 を OFF しても、LED0 は点灯したまま（データ保持）
4．LED0 点灯中に、KEY0 を押すとフリップフロップ内のデータがリセットされ消灯.



1. 順序回路の設計

順序回路とは回路内に記憶素子を持ち、過去の信号の入力に依存して出力が決まる回路です。このため、順序回路では過去の信号を記憶するフリップフロップという回路が基本となってきます。このステップでは Verilog でのフリップフロップの設計方法を学んでいきます。

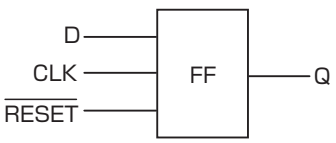
フリップフロップとは

フリップ・フロップは 1 ビットの情報を保持することができる回路です。
コンピュータのメインメモリやキャッシュメモリ、レジスタなどで使われている基本回路です。
情報は通電中のみ保持され、電源が遮断されると保持していた情報は失われます。
ちなみに原語の flip flop はパタパタという擬音を表しています。

2. フリップフロップ回路

図に、今回作成するフリップフロップ（FF）を示します。入力信号は入力 D、RESET 信号と CLK（クロック）信号で、出力信号は Q となるようなフリップフロップです。CLK 信号の立ち上がりで入力信号 D を取り込み保持します。Q にはフリップフロップが保持しているデータが出力されます。

フリップフロップの回路記号



フリップフロップ

3. プロジェクトの作成 課題 08-1

STEP 08-1 用のプロジェクトを作成してください。

4. Verilog HDL の文法 課題 08-1

課題 08-1 を実現するのに必要な文法を紹介します。

❶ レジスタ宣言 reg

```
reg　信号名；
```

reg 宣言では、値を保持する信号線を定義します。順序回路で用いるフリップフロップは reg 宣言する必要があります。また、always 文（後述）内で代入される側の信号は reg 宣言しておく必要があります。信号名はカンマ区切りで続けて宣言できます。

wire と reg は文法上、以下の制約があります。

ネット変数（wire）への代入は assign 文のみ可能
レジスタ変数（reg）への代入文は always 文の中で可能

❷ always 文

```
always @（イベント式）
```

フリップフロップなどの順序回路は always 文で記述します。イベント式の内容が発生したときに always 以降の命令を実行します。イベント式は or を用いて複数指定することもできます。
イベント式には以下の 3 種類があります。

イベント式	イベントのタイミング
信号名	信号の変化（立ち上がりまたは立下り）
posedge 信号名	信号の立ち上がり（positive edge の略）
negedge 信号名	信号の立ち下がり（negative edge の略）

大抵の場合、always 以降の命令文は複数行にわたるので begin ～ end 文（後述）を用います。

❸ begin ～ end ブロック

```
begin  end
```

always 文や if 文（後述）の後の順次処理が複数行にわたる場合に使用します。C 言語の{ }と同じ働きです。

❹ if 文

```
if（条件式） 処理 1  else  処理 2
```

C 言語と同じく、条件が成立する（真）とき処理 1 を実行します。条件が成立しない（偽）ときは処理 2 を実行します。

❺ ノンブロッキング代入文 <=

```
信号名 1  <=  信号名 2
```

「<=」を使用した代入文をノンブロッキング代入文といいます。ノンブロッキング代入文が並んで記述されている場合、それらの処理は同時に実行されます。
一般的に、順序回路を設計する場合はノンブロッキング代入文を用います。

ブロッキング代入 と ノンブロッキング代入

ブロッキング代入

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

・1 つの代入文の処理が終了するまで、
　次の処理が実行されない
・記述の順序が動作に影響する

a = b = c = d になる

ノンブロッキング代入

```
b <= a;
c <= b;
d <= c
```

・すべての右辺の処理が終了後、always 文
　end で同時に代入が実行される（同時処理）
・記述の順序が動作に影響しない

a = c や b = d にならない

フリップフロップ

5. プログラム（回路）の記述　課題 08-1

.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。

step08-1.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 Flip_Flop(CLK, RESET, D, Q);
2
3    input CLK, RESET, D;
4    output Q;
5    reg Q;                                    ❶ レジスタ宣言
6
7    always @(posedge CLK or negedge RESET)    ❷ always 文
8    begin                                     ❸ begin ～ end 文
9      if(!RESET) Q <= 0;                      ❹ if 文
10     else Q <= D;
11   end
12
13 endmodule                                   ❺ ノンブロッキング代入文
```

これが、フリップフロップを Verilog で記述したものです。CLK（クロック）の立ち上がりで値を取り込み保持します。RESET 信号が入力されると保持している値を 0 にします。

DE0-Nano ボードのクロック

DE0-Nano ボードは基板上に 50 MHz オシレータが実装されており、Cyclone IV FPGA の専用のクロック入力端子（PIN_R8）に直接接続されています。
50MHz のクロック入力は、PLL 回路を駆動するソースクロックとして使用することができます。

6. コンパイル（論理合成） 課題 08-1

文法チェック

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）

ピン配置を行ってください。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
CLK	PIN_R8	50MHz オシレータ
D	PIN_A6	トグルスイッチ 1
Q	PIN_A15	LED [0]
RESET	PIN_J15	プッシュキー [0]

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、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の転送 課題 08-1

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。

8. データが保持されない？

トグルスイッチ 1 を ON すると、LED0 が点灯します。このとき、プッシュキー 0 を押すとフリップフロップ内のデータがリセットされ LED0 が消灯します。ここまでは OK です。しかし、トグルスイッチ 1 を OFF すると、LED0 はすぐに消えてしまいます。値を保持するはずなのに、なぜでしょう。

プログラムを確認すると always 文のイベント式はクロックの立ち上がりになっています。今回使用したクロックは 50MHz でなので、フリップフロップの保持する値は 1 秒間に 5 千万回という速さで更新されることになります。このため、スイッチを切り替えた直後に LED0 が消えてしまうように見えるのです。

では、どうやってフリップフロップが値を保持しているかを確認すればよいのでしょう。このような場合にはシミュレーションソフトを使用して、回路の動作をシミュレーションすることで確認できます。シミュレーションについては STEP 09 で解説していきます。

フリップフロップ

課題 08-2

以下の変更で課題 08-1 と動作がどのように異なるでしょう。

9. プログラム（回路）の記述 課題 08-2

.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。

step08-2.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 Flip_Flop(CLK, RESET, D, Q);
2
3    input CLK, RESET, D;
4    output Q;
5    reg Q;
6
7    always @(posedge CLK)
8    begin
9      if(!RESET) Q <= 0;
10     else Q <= D;
11   end
12
13 endmodule
```

step08-1.v との違い
always 文のイベント式に negedge RESET がないだけ

10. コンパイル（論理合成）とコンフィギュレーションファイルの転送 課題 08-2

ピン配置は課題 08-1 と同じです。
コンパイルを行い、FPGA に .sof を転送し、動作を確認してみましょう。

11. 課題 08-1 とどう違う？

トグルスイッチ 1 を ON すると、LED0 が点灯します。このとき、プッシュキー 0 を押すと LED0 が消灯します。トグルスイッチ 1 を OFF すると、LED0 はすぐに消えてしまいます。一見すると課題 08-1 と動作が同じに見えますが、保持しているデータがリセットされるタイミングが課題 08-1 と違います。これについても STEP 09 のシミュレーションで解説していきます。