1　目的

ハードウェア記述言語 VHDL を用いて、ストップウオッチの機能あるいは回路構成を記述し、それを FPGA 上に実現することにより、ディジタル回路の動作原理ならびにその設計手順を理解する。また、設計・シミュレーション・実機検証の一連の工程を通じて、FPGA 開発の実践的なスキルを習得し、効率的なデバッグ手法や設計の最適化についても学ぶ。

2　原理

2.1　FPGA の原理

FPGA（Field Programmable Gate Array）は、ユーザーが回路構成を自由に書き換えられる集積回路である。FPGA 内部には多数のロジックブロック（論理素子）と、それらを相互接続する配線（インターコネクト）、そして設定情報を保持する構成メモリが組み込まれている。ユーザーは HDL（ハードウェア記述言語）を使って回路設計を行い、そのデータを FPGA に書き込むことで、加算器やカウンタ、プロセッサなど多様なデジタル回路を構成することができる。これにより、専用 IC のようなハードウェアの高速性と、ソフトウェアの柔軟性を両立できる。(1)

FPGA のチップの内では、論理機能を実現することができ、EmbeddedArrayBlock（EAB）および LogicArrayBlock（LAB）、ならびにそれらを接続するための FastTrack 列配線および行配線が、図 2.1 に示すように規則的に配置されている。

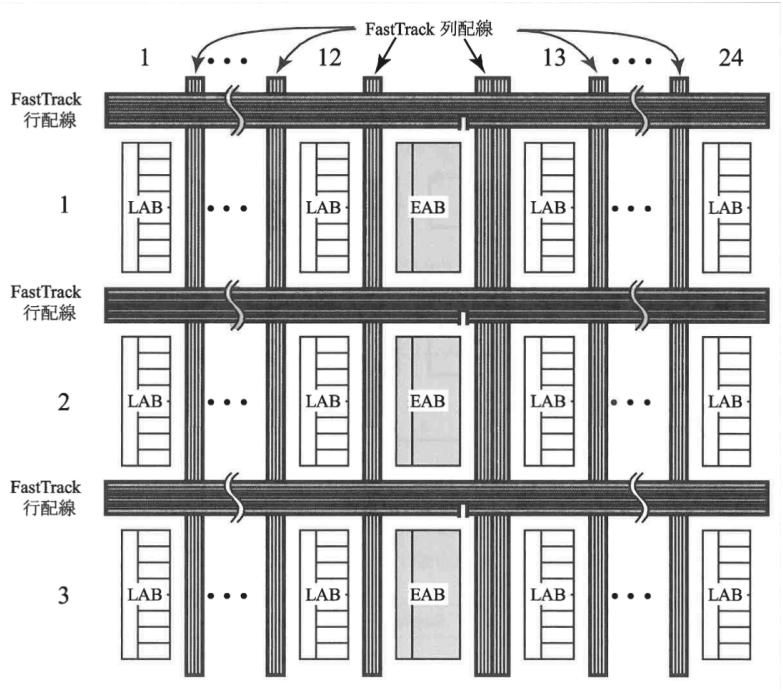

図 2.1: EP1K10 の構造

EP1K10 は 3 行 25 列構造で、各行に EAB が 1 個、LAB が 24 個配置されている。FastTrack 配線は高速伝送が可能で、行配線は 144 本、列配線は通常 24 本だが、EAB 横は 48 本となっている。EAB は 8 入力 16 出力の任意関数を実現可能で、RAM や FIFO などのメモリ機能やデータ処理にも利用できる。例えば、1 つの EAB で 256x16 ビットや 512x8 ビットの RAM を構成可能で、複数の EAB を組み合わせることでより大容量の RAM も実現できる。

2.2 実験で作成した StopWatch のモジュール構成と動作原理

今回の実験で作成する StopWatch のモジュールは、FPGA ボード「DE10-Lite」を用いて構成されている。ボードの主な構成要素としては、ストップウォッチの始動・停止用となるスイッチ SW1（上側のボタンスイッチ)、押されたら計測時間および 7 セグメント LED の表示を 0 に戻すリセット用の SW2（下側のボタンスイッチ)、時間の表示をする 3 つの 7 セグメント LED がある。

また、今回作成する StopWatch 回路の構造を図 2.2 に示す。

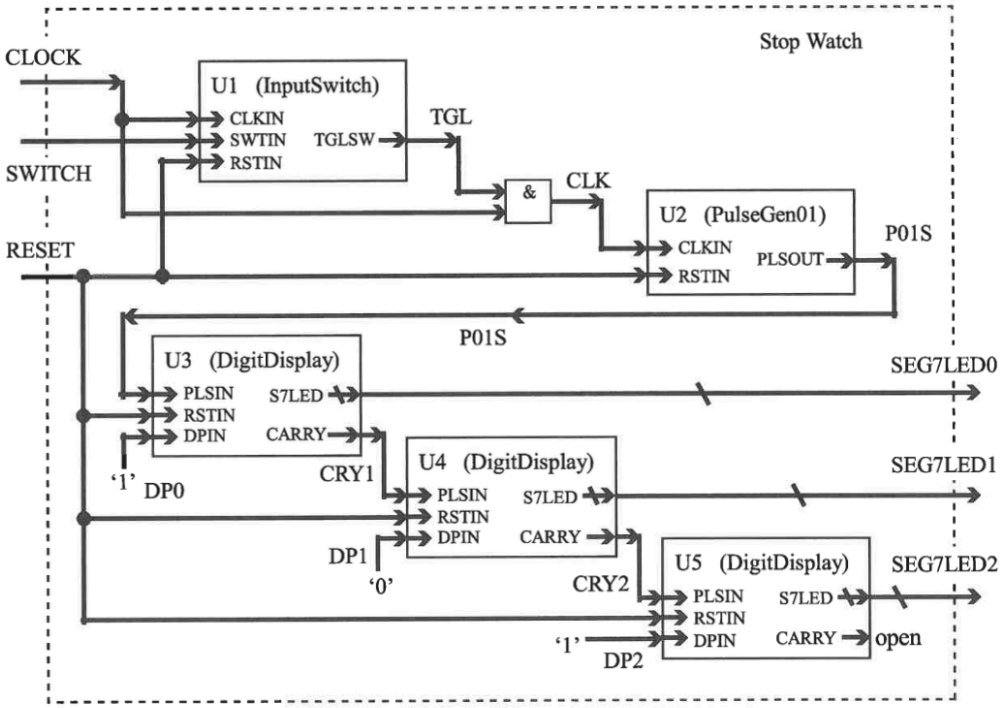

図 2.2: StopWatch 回路の構造

ユニット U1 では、クロック信号（今回実験で使用するボードでは 50MHz）とスイッチ入力（SW1,SW2）を受け取り、値に応じて U2 に情報を送っている。U2 ではクロック信号をもとに、適切なパルスを出力するようになっており、U3 から U5 ではパルス信号の入力を元に 7 セグメント LED への出力および小数点の表示、桁上げ処理などを行っている。

3 実験結果

3.1 実験手順

3.1.1 VHDL コードの作成

サブモジュールである周期 0.1 秒のパルス発生回路（PulseGen01）を VHDL で記述し、PulseGen01.vhd）として作成する。

3.1.2　コンパイルとデバッグ

Quartus Prime を用いて VHDL コードをコンパイルし、FPGA に実装可能な回路へ変換する。コンパイル時にエラーが発生した場合は、デバッグを行い記述を修正する。

3.1.3　シミュレーションによる動作確認

Quartus Prime のシミュレーション機能を利用して、設計した回路が仕様通り動作するか確認する。この際、実際の時間単位ではシミュレーションに時間がかかりすぎるのでパルス周期等を調整し、シミュレーションをしやすくする。

3.1.4　FPGA への書き込みとボード上での実機動作確認

Quartus Prime から実機（ボード）に書き込み、実際のハードウェア上で動作を確認する。

3.2　実験結果

3.2.1　使用した 2 種類の PulseGen01 のプログラムリストと処理フローおよびその役割

使用した 2 種類の PulseGen01 のリストを図 3.1,3.2 に示す。

```
-- ================================================
-- 周期 0.1 秒のパルス生成 --
-- ================================================
library ieee;
use ieee.std_logic_1164.all ;
use ieee.std_logic_unsigned.all ;
-- ================================================
entity  PulseGen01  is
        port (
                CLKIN : in std_logic;                   -- クロック入力
                RSTIN : in std_logic;                   -- reset 入力
                PLSOUT : out std_logic );   -- 0.1 秒周期のパルス
end PulseGen01 ;
-- ================================================
architecture  Some_Description  of  PulseGen01  is

    signal C_CNT : integer range 0 to 4999999;

begin
   process(CLKIN,RSTIN)
    begin
      if (CLKIN'event and CLKIN='1')then
        if(RSTIN='0')then
          C_CNT<=0;
          PLSOUT<='0';
        elsif(C_CNT = 4999999)then
          C_CNT <= 0;
          PLSOUT <= '1';
        else
          C_CNT<=C_CNT+1;
          PLSOUT<='0';
        end if;
      end if;
    end process;

end Some_Description ;
================================================
```

図 3.1: PulseGen01 のプログラムリスト（同期式）

```
-- ================================================
-- 周期 0.1 秒のパルス生成 --
-- ================================================
library ieee;
use ieee.std_logic_1164.all ;
use ieee.std_logic_unsigned.all ;
-- ================================================
entity   PulseGen01   is
          port (
                    CLKIN : in std_logic;                    -- クロック入力
                    RSTIN : in std_logic;                    -- reset 入力
                    PLSOUT : out std_logic );    -- 0.1 秒周期のパルス
end PulseGen01 ;
-- ================================================
architecture   Some_Description   of   PulseGen01   is

     signal C_CNT : integer range 0 to 4999999;

begin
   process(CLKIN,RSTIN)
     begin
       if(RSTIN='0')then
         C_CNT<=0;
          PLSOUT<='0';
        else
         if (CLKIN'event and CLKIN='1')then
           if(C_CNT = 4999999)then
             C_CNT <= 0;
             PLSOUT <= '1';
           else
             C_CNT<=C_CNT+1;
             PLSOUT<='0';
           end if;
         end if;
       end if;
   end process;

 end Some_Description ;
==================================================
```

図 3.2: PulseGen01 のプログラムリスト（非同期式）

またこれらのプログラムの処理フローを図 3.3,3.4 に示す

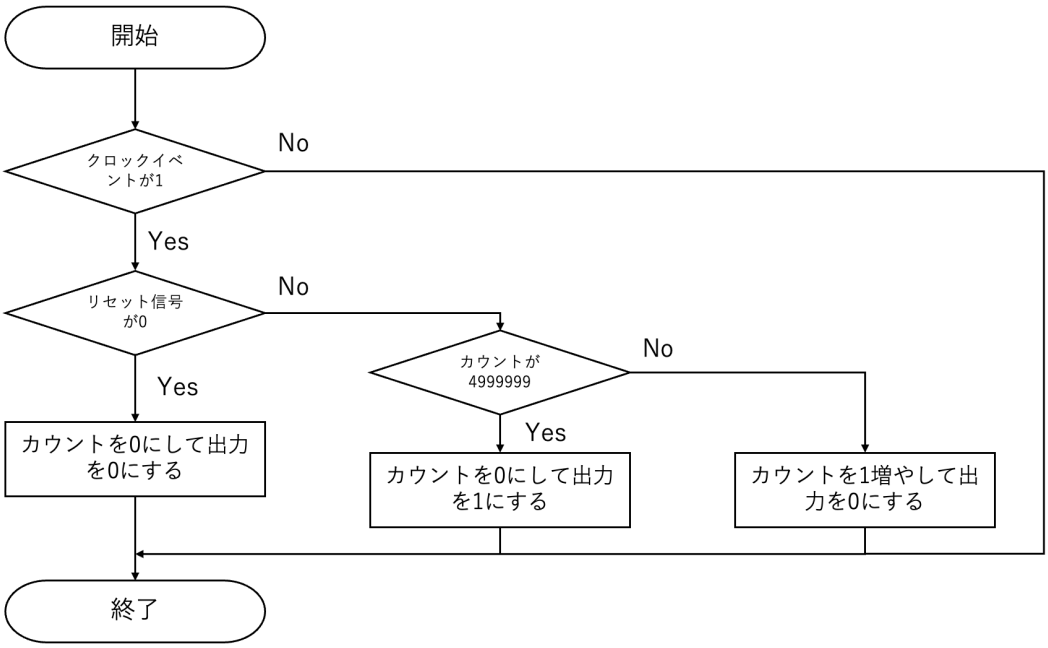

図 3.3: PulseGen01 の処理フロー（同期式）

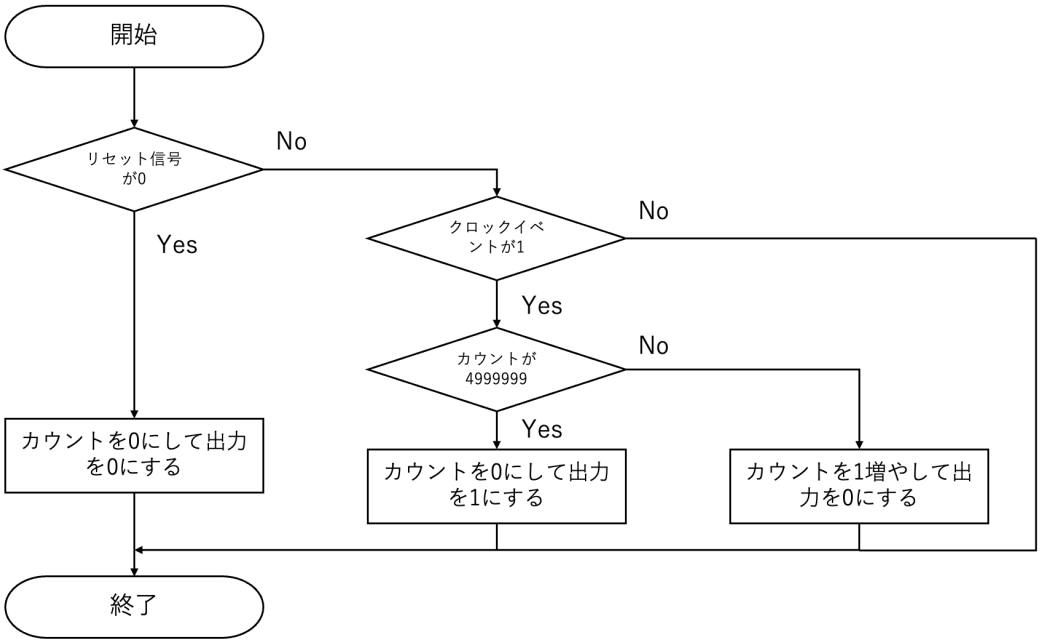

図 3.4: PulseGen01 の処理フロー（非同期式）

図 3.3 に示されているのは、PulseGen01 の同期式処理フローである。このフローでは、すべての処理がクロックイベントに同期して実行される。すべての条件分岐や処理がクロックイベントによって制御されており、タイミングの一貫性が保たれているため、ハードウェア的に安定した動作が期待できる構成である。

一方、図 3.4 に示されているのは、PulseGen01 の非同期式処理フローである。この処理フローでは、まずクロックとは無関係にリセット信号の状態が判定される。リセット信号がアクティブである場合、ただちにカウント値と出力は 0 に初期化される。リセット信号がアクティブでない場合にのみ、クロックイベントの有無が判定され、以降の処理は同期式と同様にカウント値に応じて動作する。非同期式の利点は、リセット処理がクロックに依存せず即座に反映される点にあるが、一方でタイミング制御が難しくなる可能性もあるため、安定性の観点では同期式に劣る場合がある。

表 3.1 に、同期式と非同期式のシミュレーション結果の比較を示す。

表 3.1: 同期式と非同期式の解析結果の比較

項目	同期式	非同期式
論理素子の使用数	92/49760 (0.18%)	100/49760 (0.2%)
ピンの使用数	27/360 (8%)	35/360 (10%)
最高クロック周波数	241.08MHz	196.85MHz

これより、同期式は非同期式より論理素子やピンの使用数が少なく、最高クロック周波数が高いことがわかる。

3.2.2 StopWatch と PulseGen01 の RTL Viewer

RTL Viewer は、今回の実験で使用した Quartus Prime に搭載されている機能であり、HDL（ハードウェア記述言語）で記述したデジタル回路設計を「レジスタ転送レベル（RTL）」の抽象度で図式的に可視化するためのツールである (2)。今回の実験では、3.1.1 節で示したような同期式と非同期式の回路を比較する。StopWatch 回路の全体像を図 3.5 に示す。また PulseGen01(同期式) の回路を図 3.6 に、PulseGen01(非同期式) の回路を図 3.7 に示す。

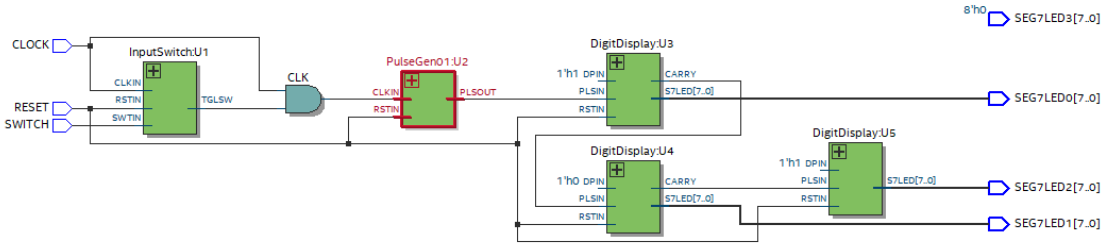

図 3.5: StopWatch の全体像の回路

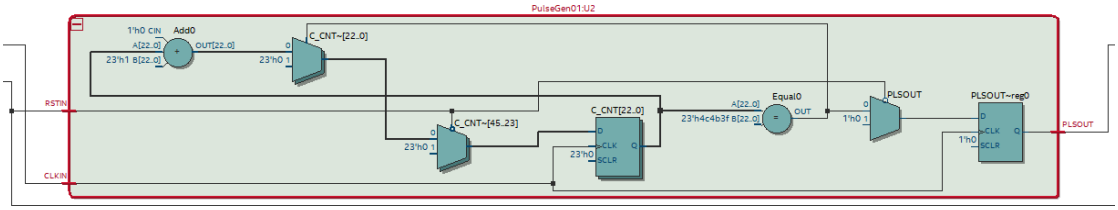

図 3.6: PulseGen01（同期式）の回路

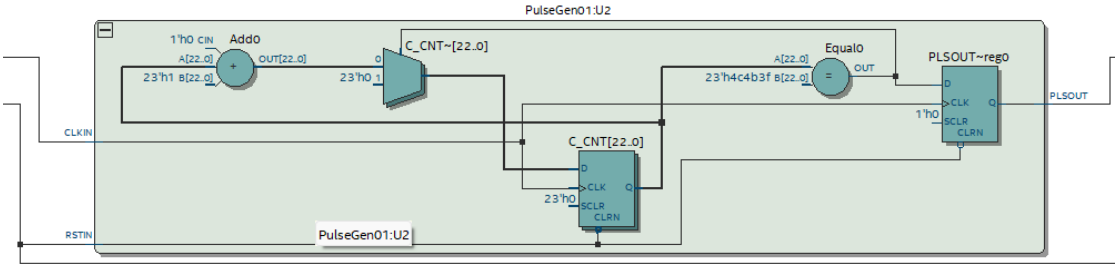

図 3.7: PulseGen01（非同期式）の回路

図 3.5 における PulseGen01:U2 の内部構造が図 3.6,3.7 のようになっている。同期式の方が非同期式の回路より構成素子が多いことがわかる。

3.2.3 StopWatch の Simulation 結果

図 3.8, 3.9, 3.10 に StopWatch のシミュレーション結果を示す。各図のグラフには、上から順に CLOCK、INIT_RESET、SWITCH、SEG7LED0、SEG7LED1、SEG7LED2 が表示されている。横軸（時間軸）を段階的に縮小することで、各 SEG7LED の動作タイミングを詳細に確認できるようにしている。

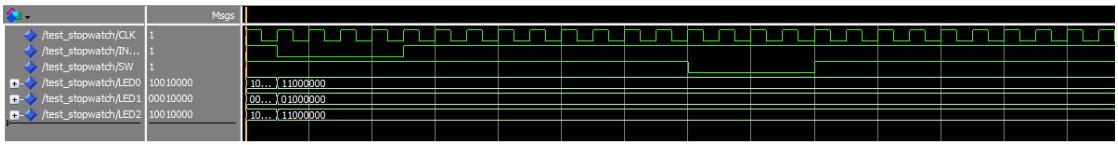

図 3.8: StopWatch のタイミングチャート (RESET と SWITCH のタイミング)

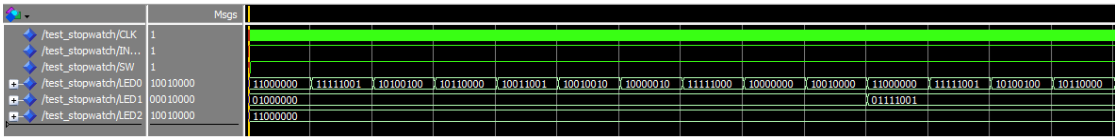

図 3.9: StopWatch のタイミングチャート (CLK と SEG7LED0,SEG7LED2 のタイミング)

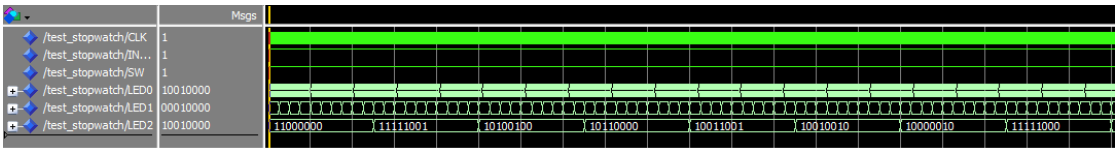

図 3.10: StopWatch のタイミングチャート (SEG7LED2 と SEG7LED3 のタイミング)

これらのタイミングチャートから、SEG7LED0 は図 3.11 に示す SEG7LED の数字表示に対応するように 0 から 9 までカウントを繰り返し、10 回カウントされるごとに SEG7LED1 に 1 つ桁上げされることが確認できる。同様に、SEG7LED1 が 10 回カウントされると SEG7LED2 に 1 つ桁上げされる動作が見られる。

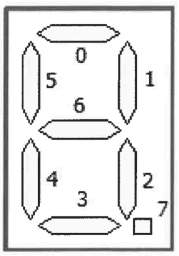

図 3.11: SEG7LED の表示

4 考察

5 使用機材

表 5.1 に示す。

表 5.1: 使用機材一覧

名称	型式	製造元	管理番号
PC		DELL	
FPGA ボード	TERASIC-DE10-LITE-A P0466		

6 参考文献

1. FPGA とは, フィールドでプログラムできる論理回路

   https://jp.mathworks.com/discovery/fpga.html

   閲覧日：2025 年 5 月 21 日

2. 新人エンジニアの赤面ブログ 『 RTL Viewer と Technology Map Viewer の使用用途を比べてみましょう』

   https://www.macnica.co.jp/business/semiconductor/articles/intel/206/

   閲覧日：2025 年 5 月 24 日