電子工学科 実験報告書

実験題目：　各種計算ハードウェアの活用 VHDL によるディジタル回路の設計
担当教員：　木場 隼介 先生
実験開始日：　令和 5 年 10 月 12 日
実験終了日：　令和 5 年 10 月 19 日
提出日：　令和 5 年 10 月 24 日
再提出日：

学年：　5 年
出席番号：　12
実験班：　B 班
氏名：　河合 将暉

共同実験者名：

コメント欄

1 目的

本実験では，業界標準の VHDL と Altera(Intel) 社の Quartus Prime Lite Edition を使用し，HDL，FPGA を用いたディジタル回路設計の基本的な考え方と手法を習得することを目的とする。

2 解説

2.1 VHDL・FPGA とは

VHDL とは IEEE で標準化されたデジタル回路設計用のハードウェア記述言語 (HDL：Hardware Description Language) である。従来の電子回路設計はプリント回路基板設計用の CAD などを用いて多数の電子部品を回路図記号で表記することが一般的で製造後に回路構成を変更できなかったが，現場で論理回路の構成をプログラムできる論理回路を集積したデバイス (FPGA：Field Programable Gate Array) が登場すると，HDL を用いてその論理ゲートをプログラミングのように記述することが可能になった．HDL には VHDL，VerilogHDL の 2 種類が存在し，VHDL は FPGA が登場した初期から存在し，Ada 言語や Pascal 言語を参考に記法が作られている [2]。VerilogHDL は比較的新しい言語で C 言語ベースの記法で作られている [2]。明確な違いの例を挙げると，論理演算子が VHDL では and or not であるが VerilogHDL では & | ~などの記号で表されている。

2.2 Altera 社について

Altera 社は 1983 年に設立された PLD(Programable Logic Device) の代表的企業で，システムオンプログラマブルチップを可能とするべく，様々な技術を開発し，その中ではチップ内にメモリやマイクロプロセッサ，トランシーバを埋め込んだものも存在するという。現在では Intel 社に買収され，FPGA 部門として活動している。

2.3 Quartus Prime とは

3 実験内容

3.1 使用器具

表 1: test

No	機器名	型番	シリアル No	備考
1	FPGA ボード	Cyclone V E FPGA Development Kit	2	シリアル No は外箱の番号を記載
2	PC	ASUS		

3.2 実験準備

Altera(Intel) 社から発売されている評価ボード Cyclone V E FPGA Development Kit を使用した。このボードに搭載されている FPGA の Cyclone V E FPGA (5CEFA7F31I7N) と周辺機器の LED4 個, 押しボタンスイッチ 4 個，クロック発振器 (50MHz), キャラクタ液晶などが搭載されている。準備として,Quartus Prime の操作手順について以下に示す。

1) New Project Wizard の起動
2) プロジェクト名の指定
3) テンプレートの設定
4) 使用する設計ファイルの設定
5) ターゲットデバイスファミリの指定
6) EDA Tool の指定

7) 設定の確認

これらの設定を行ったあと, カウンタの数字によって点灯する LED を変えるプログラムを保存した。そして, HDL で記述された回路は論理合成と配置配線を行うことで LSI に実装できるようになる．FPGA 設計用のツールでは，これらの処理はひとまとめにされ，コンパイルと呼ばれている．以下にコンパイルの処理手順について示す.

1) プリフィット
2) ピン配置
3) ポストフィット

上記によりコンパイルが行われる．次に，コンパイル後に出力されたコンフィグレーションデータを FPGA ボードに書き込む．この際に，Window のツールバー上「Tools」→「Programmer」を選択し，Programmer を開く．ここで書き込むデバイスの指定を行い，書き込みを行った.

以上の準備を以降の実験でも同様に行い，基本的な論理回路の動作確認をした.

3.3 LED の点灯と消灯

FPGA に搭載されている LED は FPGA の I/O ピンが “L” のときに LED が点灯し，“H” のときに消灯する．LED を点灯・消灯させるプログラムをソースコード 1 に示す.

ソースコード 1: sample1

```
1       library ieee;
2       use ieee.std_logic_1164.all;
3
4       entity sample01 is
5
6       port (
7       led_out : out std_logic);
8
9       end sample01;
10
11      architecture rtl of sample01 is
12
13      begin
14
15      -- LEDはLOWで点灯するように回路が構成されているため
16      -- std_logic型の0を出力する。これを1にすると消灯する。
17      led_out <= '0';
18
19      end rtl;
```

FPGA のピン配置を表 2 に示す.

表 2: LED 点灯回路のピン配置

port の名前	ピン名称
led_out	PIN_AK3

ここで，VHDL 記述の基本事項として，ライブラリ宣言部とパッケージ宣言部で始まり，エンティティ宣言で入出力ポートを定義し，アーキテクチャ宣言に動作を記述していく．コメントアウトは “–” で記述できる．各事項についての詳細を以下に示す.

1) ライブラリ宣言

2) パッケージ呼び出し
3) エンティティ宣言
4) アーキテクチャ宣言

3.4 LED でのバイナリ表示

3.3 項で行った一つの LED 表示を応用して，複数の LED を制御可能なプログラムを作成した．そして，4 つの LED を用いてバイナリ表示を表現した．ソースコード 2 を以下に示す.

ソースコード 2: sample2

```
1       library ieee;
2       use ieee.std_logic_1164.all;
3
4       entity sample02 is
5
6       port (
7       led_out : out std_logic_vector(3 downto 0));
8
9       end sample02;
10
11      architecture rtl of sample02 is
12
13      begin
14
15      -- 10進数の5を表現する。LEDはLOWで点灯するため、
16      -- ビット反転した値を出力する
17      led_out <= "1010";
18
19      end rtl;
```

FPGA のピン配置を表 3 に示す.

表 3: 4 つの LED で 2 進数表現する回路のピン配置

port の名前	ピン名称
led_out[0]	PIN_AK3
led_out[1]	PIN_AJ4
led_out[2]	PIN_AJ5
led_out[3]	PIN_AK6

上記のプログラムを実行した結果を確認し，点灯パターンが異なる場合についても動作確認を行った．この際，4 ビットの信号は std_logic_vector によってバスとして定義している．そのため，led_out <= ”1010”; のようにまとめて記述することが可能である.

3.5 LED でのグレイ符号表示

本項では，2 進数の進み方が 1 ビットずつしか変化しないグレイ符号を表示するプログラムを作成した．ソースコード 3 を以下に示す.

ソースコード 3: sample3

```
1       library ieee;
2       use ieee.std_logic_1164.all;
```

```
3
4       entity sample03 is
5
6       port (
7       led_out : out std_logic_vector(3 downto 0));
8
9       end sample03;
10
11      architecture rtl of sample03 is
12
13      signal counter : std_logic_vector(3 downto 0):= (others=>'0');
14
15      begin
16
17      counter <= "0010"; -- ①
18
19      process (counter) -- ②
20      begin
21      case counter is
22      when "0000" =>
23      led_out <= "1111"; -- ③
24      when "0001" =>
25      led_out <= "1110";
26      when "0010" =>
27      led_out <= "1100";
28      when "0011" =>
29      led_out <= "1101";
30      when "0100" =>
31      led_out <= "1001";
32      when "0101" =>
33      led_out <= "1000";
34      when "0110" =>
35      led_out <= "1010";
36      when "0111" =>
37      led_out <= "1011";
38      when "1000" =>
39      led_out <= "0011";
40      when "1001" =>
41      led_out <= "0010";
42      when "1010" =>
43      led_out <= "0000";
44      when "1011" =>
45      led_out <= "0001";
46      when "1100" =>
47      led_out <= "0101";
48      when "1101" =>
49      led_out <= "0100";
50      when "1110" =>
51      led_out <= "0110";
52      when "1111" =>
53      led_out <= "0111";
54      when others => null;
55      end case;
```

```
56      end process;
57      end rtl;
```

上記のプログラムで 2 進数がグレイ符号に変換して出力されているか確認した．また，それぞれ異なる数字を変換した場合やグレイ符号を 2 進数に変換した場合についてもプログラムを変更し，同様に確認した．

3.6 LED のクロック同期動作

sample3 で使用したグレイ符号へのエンコーダ/デコーダ回路を自動的に一定周期で変化させるプログラムをソースコード 4 に示す．

ソースコード 4: sample4

```
1       library ieee;
2       use ieee.std_logic_1164.all;
3       use ieee.std_logic_unsigned.all;
4
5       entity sample04 is
6
7       port (
8       clk : in std_logic;
9       led_out : out std_logic_vector(3 downto 0));
10
11      end sample04;
12
13      architecture rtl of sample04 is
14
15      signal counter : std_logic_vector(3 downto 0);
16
17      -- 以下は分周用のカウンタとクロック
18      signal div_counter : std_logic_vector(25 downto 0);
19      signal div_clk : std_logic;
20
21      begin
22
23      -- 以下は50MHzのクロックから1秒弱のクロックを作成する
24      -- 分周回路です。
25      process (clk)
26      begin -- process
27      if clk'event and clk = '1' then
28      div_counter <= div_counter + 1;
29      end if;
30      end process;
31
32      -- div_clkが1Hzのクロックで、デコーダへのクロック入力
33      div_clk <= div_counter(19);
34      -- ここまでが分周回路
35
36      process (div_clk)
37      begin
38      if div_clk'event and div_clk = '1' then
39      counter <= counter + 1;
40      end if;
41      end process;
42
```

```
43      process (counter)
44      begin
45      case counter is
46      when "0000" =>
47      led_out <= "1111";
48      when "0001" =>
49      led_out <= "1110";
50      when "0010" =>
51      led_out <= "1100";
52      when "0011" =>
53      led_out <= "1101";
54      when "0100" =>
55      led_out <= "1001";
56      when "0101" =>
57      led_out <= "1000";
58      when "0110" =>
59      led_out <= "1010";
60      when "0111" =>
61      led_out <= "1011";
62      when "1000" =>
63      led_out <= "0011";
64      when "1001" =>
65      led_out <= "0010";
66      when "1010" =>
67      led_out <= "0000";
68      when "1011" =>
69      led_out <= "0001";
70      when "1100" =>
71      led_out <= "0101";
72      when "1101" =>
73      led_out <= "0100";
74      when "1110" =>
75      led_out <= "0110";
76      when "1111" =>
77      led_out <= "0111";
78      when others => null;
79      end case;
80      end process;
81      end rtl;
```

3.7 FPGA の論理演算

ソースコード 5: sample5

```
1       library ieee;
2       use ieee.std_logic_1164.all;
3
4       entity sample05 is
5
6       port (
7       led_out : out std_logic;
8       pb0: in std_logic;
9       pb1: in std_logic);
```

```
10
11      end sample05;
12
13      architecture rtl of sample05 is
14
15      signal operation_result : std_logic;
16      signal pb0_positive : std_logic;
17      signal pb1_positive : std_logic;
18
19      begin
20
21      pb0_positive <= not pb0; -- スイッチ入力は負論理なので not で正論理に変換
22      pb1_positive <= not pb1; -- スイッチ入力は負論理なので not で正論理に変換
23
24      operation_result <= (pb0_positive or pb1_positive);
25
26      led_out <= not operation_result; -- LEDは負論理動作なので not で正論理から変換
27
28      end rtl;
```

3.8 I/O 機器による LED 点消灯

ソースコード 6: sample6

```
1       library ieee;
2       use ieee.std_logic_1164.all;
3       use ieee.std_logic_unsigned.all;
4
5       entity sample06 is
6
7       -- div_bitsは分周回路に用いられているカウンタのビット数
8       generic (
9       div_bits : integer := 16);
10
11      port (
12      clk : in std_logic;
13      sw_in : in std_logic;
14      led_out : out std_logic);
15
16      end sample06;
17
18      architecture rtl of sample06 is
19
20      signal led_node : std_logic := '0';
21      signal div_counter : std_logic_vector(div_bits-1 downto 0) := (others => '0');
22      signal sw_in_node : std_logic;
23      signal sw_latch_on : std_logic := '0';
24      begin
25
26      -- 入力クロックを分周し、チャタリングを除去する回路
27      process (clk)
28      begin
29      if clk'event and clk = '1' then -- rising clock edge
```

```
30      div_counter <= div_counter + 1;
31      end if;
32      end process;
33
34      -- 分周したクロックでスイッチからの入力をラッチする回路
35      -- 分周用カウンタの最上位ビットをクロックとして用いる
36      process (div_counter(div_bits-1))
37      begin -- process
38      if div_counter(div_bits-1)'event and div_counter(div_bits-1) = '1' then
39      sw_in_node <= sw_in;
40      end if;
41      end process;
42
43      -- 基本クロックに同期してDFF(led_node)をトグルする。
44      -- チャタリングが発生すると、この回路が複数回アクティブになり、
45      -- パルスが何度も発生し、LEDがついたまま、あるいは、
46      -- 消えたままの場合がある。このような時はdiv_counterのビット数を増やす。
47      process (clk)
48      begin -- process
49      if clk'event and clk = '1' then -- rising clock edge
50      if sw_in_node = '0' and sw_latch_on = '0' then
51      led_node <= not led_node;
52      sw_latch_on <= '1';
53      elsif sw_in_node = '1' and sw_latch_on = '1' then
54      sw_latch_on <= '0';
55      elsif sw_in_node = '0' and sw_latch_on = '1' then
56      sw_latch_on <= '1';
57      end if;
58      end if;
59      end process;
60
61      led_out <= led_node;
62
63      end rtl;
```

3.9 2 進数ルーレット

ソースコード 7: sample7

```
1       library ieee;
2       use ieee.std_logic_1164.all;
3       use ieee.std_logic_unsigned.all;
4
5       entity sample07 is
6       generic (
7       div_bits : integer := 15);
8
9       port (
10      clk : in std_logic;
11      sw_in : in std_logic;
12      led_out : out std_logic_vector(3 downto 0));
13
14      end sample07;
```

```
15
16      architecture rtl of sample07 is
17
18      -- ステートマシンのためのノード宣言
19      signal counter_curr, counter_next : std_logic_vector(3 downto 0):= (others=>'0');
20
21      -- チャタリング除去のための分周クロックとスイッチ入力のためのラッチ
22      signal clk_div_sw, sw_node, sw_dff : std_logic := '0';
23      -- LED出力用の分周カウンタ。このカウンタのビット数を増やすと
24      -- ルーレットの変化がゆっくりになります。
25      signal clk_div_count : std_logic_vector(22 downto 0) := (others => '0');
26      signal div_counter : std_logic_vector(div_bits-1 downto 0) := (others => '0');
27      signal sw_latch_on : std_logic := '0';
28      begin
29
30      -- チャタリング除去のためのクロック分周
31      process (clk)
32      begin
33      if clk'event and clk = '1' then
34      div_counter <= div_counter + 1;
35      end if;
36      end process;
37      -- 最上位をチャタリング除去クロックとして用いる
38      clk_div_sw <= div_counter(div_bits-1);
39
40      -- チャタリング除去してスイッチ入力値を採る
41      process (clk_div_sw)
42      begin
43      if clk_div_sw'event and clk_div_sw = '1' then
44      sw_node <= sw_in;
45      end if;
46      end process;
47
48      -- ルーレットの回る速さをゆっくりにするための分周回路
49      -- clk_div_countの最上位ビットをステートマシンのクロックとして用います。
50      process (clk)
51      begin
52      if clk'event and clk = '1' then
53      clk_div_count <= clk_div_count + 1;
54      end if;
55      end process;
56
57      -- ルーレットをストップ・スタートさせるためのラッチ
58      -- スイッチ入力によってトグルされます。
59      process (clk)
60      begin -- process
61      if clk'event and clk = '1' then -- rising clock edge
62      if sw_node = '0' and sw_latch_on = '0' then
63      sw_dff <= not sw_dff;
64      sw_latch_on <= '1';
65      elsif sw_node = '1' and sw_latch_on = '1' then
66      sw_latch_on <= '0';
67      elsif sw_node = '0' and sw_latch_on = '1' then
```

```
68      sw_latch_on <= '1';
69      end if;
70      end if;
71      end process;
72
73      -- LED出力を制御するステートマシン
74      process (clk)
75      begin
76      if clk_div_count(22)'event and clk_div_count(22) = '1' then -- rising clock edge
77      if sw_dff = '1' then
78      counter_curr <= counter_next;
79      end if;
80      end if;
81      end process;
82
83      process (counter_curr)
84      begin
85      case counter_curr is
86      when "1011" =>
87      counter_next <= "0000"; -- (a)
88      when "0000" =>
89      counter_next <= "0001";
90      when "0001" =>
91      counter_next <= "0010";
92      when "0010" =>
93      counter_next <= "0011";
94      when "0011" =>
95      counter_next <= "0100";
96      when "0100" =>
97      counter_next <= "0101";
98      when "0101" =>
99      counter_next <= "0110";
100     when "0110" =>
101     counter_next <= "0111";
102     when "0111" =>
103     counter_next <= "1000";
104     when "1000" =>
105     counter_next <= "1001";
106     when "1001" =>
107     counter_next <= "1010";
108     when "1010" =>
109     counter_next <= "1011";
110     when others => null;
111     end case;
112     end process;
113
114
115     process (counter_curr)
116     begin
117     led_out <= not counter_curr;
118     end process;
119     end rtl;
```

3.10 4 色スロットマシン

ソースコード 8: sample8

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4
5  entity sample08 is
6  generic (
7  div_bits : integer := 15);
8
9  port (
10 clk : in std_logic;
11 resetn : in std_logic;
12 sw_in : in std_logic;
13 led_out0, led_out1, led_out2 : out std_logic_vector(3 downto 0));
14
15 end sample08;
16
17 architecture rtl of sample08 is
18
19 component roulette
20
21 port (
22 clk : in std_logic;
23 resetn : in std_logic;
24 sw_in : in std_logic;
25 led_out : out std_logic_vector(3 downto 0));
26
27 end component;
28
29 signal clk_div, clk_roulette, sw_node : std_logic := '0';
30 signal div_counter : std_logic_vector(22 downto 0) := (others => '0');
31 signal sw0, sw1, sw2 : std_logic;
32 signal sw_latch, sw_latch_on : std_logic := '0';
33 begin
34
35 roulette_unit0 : roulette port map (
36 clk => clk_roulette,
37 resetn => resetn,
38 sw_in => sw0,
39 led_out => led_out0);
40
41 roulette_unit1 : roulette port map (
42 clk => clk_roulette,
43 resetn => resetn,
44 sw_in => sw1,
45 led_out => led_out1);
46
47 roulette_unit2 : roulette port map (
48 clk => clk_roulette,
49 resetn => resetn,
```

```
50          sw_in => sw2,
51          led_out => led_out2);
52
53          process (clk)
54          begin
55          if clk'event and clk = '1' then -- rising clock edge
56          div_counter <= div_counter + 1;
57          end if;
58          end process;
59
60          clk_div <= div_counter(div_bits-1);
61          clk_roulette <= div_counter(22);
62
63          -- チャタリング除去
64          process (clk_div)
65          begin -- process
66          if clk_div'event and clk_div = '1' then -- rising clock edge
67          sw_node <= sw_in;
68          end if;
69          end process;
70
71          -- スイッチ入力を1パルスにして出力する回路
72          process (clk)
73          begin -- process
74          if clk'event and clk = '1' then -- rising clock edge
75          if sw_node = '0' and sw_latch_on = '0' then
76          sw_latch <= '1';
77          sw_latch_on <= '1';
78          elsif sw_node = '1' and sw_latch_on = '1' then
79          sw_latch <= '0';
80          sw_latch_on <= '0';
81          elsif sw_node = '0' and sw_latch_on = '1' then
82          sw_latch <= '0';
83          sw_latch_on <= '1';
84          end if;
85          end if;
86          end process;
87
88          -- スイッチによるパルスを受け取ってルーレットを止めていく回路
89          process (clk, resetn)
90          begin -- process
91          if resetn = '0' then
92          sw0 <= '1';
93          sw1 <= '1';
94          sw2 <= '1';
95          elsif clk'event and clk = '1' then -- rising clock edge
96          if sw_latch = '1' then
97          if (sw0 = '1' and sw1 = '1' and sw2 = '1') then
98          sw0 <= '0';
99          elsif (sw0 = '0' and sw1 = '1' and sw2 = '1') then
100         sw1 <= '0';
101         elsif (sw0 = '0' and sw1 = '0' and sw2 = '1') then
102         sw2 <= '0';
```

```
103         end if;
104         end if;
105         end if;
106         end process;
107         end rtl;
```

4　実験結果

参考文献

[1]「各種計算ハードウェアの活用〜VHDL によるディジタル回路の設計〜」神戸高専電子工学科 pp.01-33
[2] トーマスイッチ「Verilog と VHDL の違いとはわかりやすく解説」https://toumaswitch.com/5q0shy1sod/