パルス波高値計測回路の製作

吉田久史、○豊田朋範

自然科学研究機構　分子科学研究所　装置開発室

概要

分子科学研究所の極端紫外光実験施設(UVSOR)の自由電子レーザー(FEL : Free Electron Laser)実験において、FEL 強度モニターからの信号の波高値を計測する装置が必要になった。信号の波高値は最終的に TDC(Time to Digital Converter)によりパソコンに取り込むため、製作する計測装置には波高値の検出と同時にその値に比例したインターバルを持つ 2 つのパルスを出力する機能が必要となった。我々は、40MSPS の 12bitA/D 変換器と Xilinx の CPLD を使用して、それらの機能を順次実行する計測装置を製作した。本報告では回路ブロックの機能とハードウェア記述言語による構築、並びにシミュレーションによる動作検証手法等について報告する。

1　計測装置の構成

本装置の回路構成を図 1 に示す。CPLD で構築したのは図 1 における青色の部分である。CPLD で構築した理由は、(1)40MHz の高速クロックに同期したシーケンス動作を実現する(2)NIM1 巾モジュールという限られたスペースに実装するーの 2 つが挙げられる。シーケンス動作については別記事で詳述されているので、ここでは割愛する。

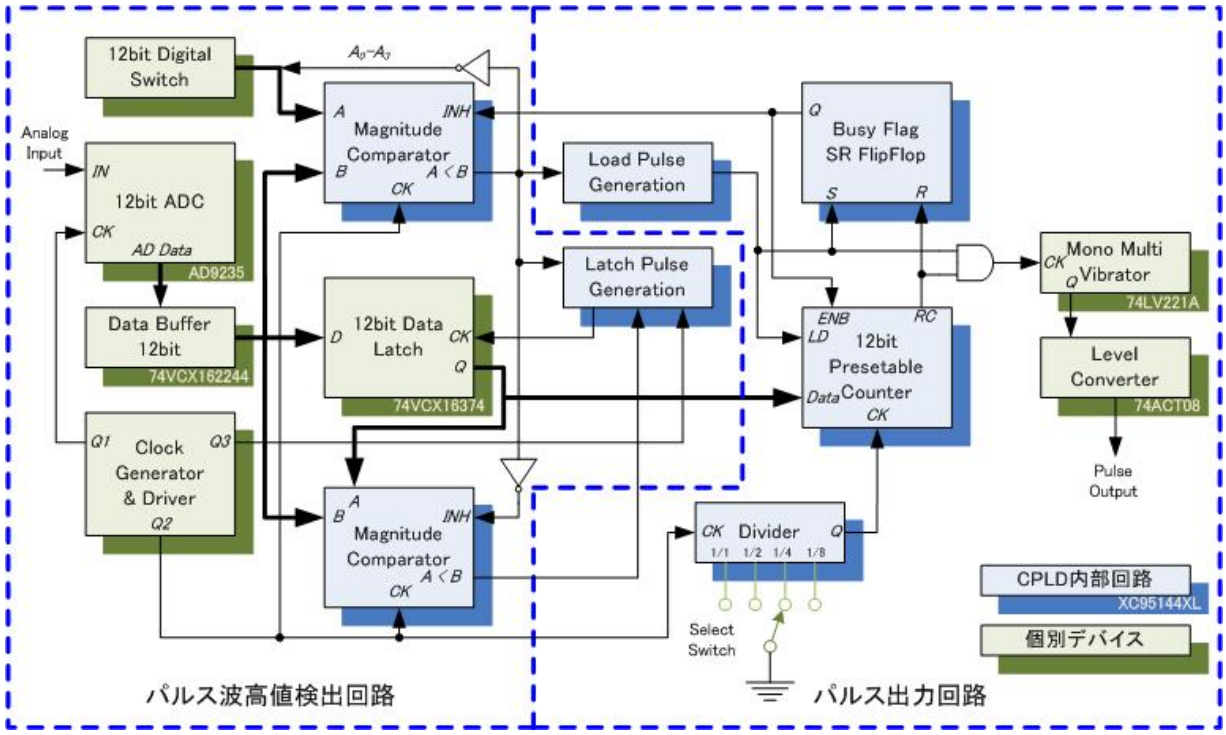

図 1.　パルス波高値計測回路のブロック図

CPLD の機能を構築するには、ハードウェア記述言語が用いられる。今回は筆者が何度か製作経験を有する[1]Xilinx 社の IDE(統合開発環境)である ISE Webpack と、XC95144XL-10TQ144[2]を搭載したカメレオン USB 基板[3](図 2)を用いた。ハードウェア記述言語は VHDL を選択した。

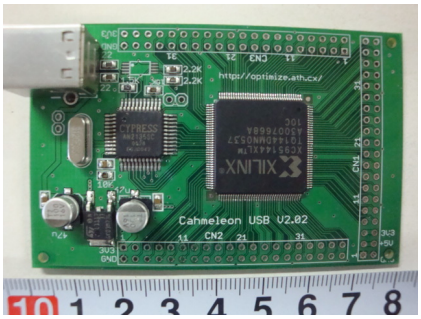

図 2.　カメレオン USB

2　VHDL によるディジタル回路構築

2.1　マグニチュード・コンパレータ

12bitA/D 変換器 AD9235[4]は、40MHz のシステムクロックに同期して、12bit ディジタルデータを逐次出力する。図 1 上段のコンパレータ(コンパレータ 1)は、この 12bit ディジタルデータ B とスイッチで設定するしきい値 A を比較して、A<B の時出力を”1”にする。図 1 下段のコンパレータ(コンパレータ 2)は、ラッチされたデータ A と 12bit ディジタルデータ B を比較して、A<B であれば出力を”1”にする。いずれのコンパレータもシステムクロック CK の立ち下がりで比較判定する。また、INH 入力が”1”だと比較動作を停止して出力を保持する。

コンパレータ 1 の VHDL ソースを図 3 に示す。比較演算のために 2 つの入力を bit 列から Integer に変換する(241-246 行)。クロックもしくは INH の変化で駆動する process 構文(250-265 行)では、INH が”1”であれば直前の出力を保持する(252-255 行)。INH が”0”であれば、クロックの立ち下がりで比較演算を行う。しきい値より 12bit ディジタルデータが大きければ出力を”1”にして、小さければ”0”にする。それ以外では出力を”0”にする(257-263 行)。

```
237  ---************************************************---
238  -- Comparator1 (Function "A < B", "A>=B" with 'Inhibit') --
239  -- Transform 'Integer' section
240      --Conversion binary to decimal(Comparator1-A)
241      int_cmp1_a <= CONV_INTEGER(not(cmp1_in1));  -- also 'int_value' for normal
242  -->  int_cmp1_a <= CONV_INTEGER((cmp1_in1));--also 'int_value' for testing
243      --Conversion binary to decimal(Comparator1-B&2-A)
244  -->  int_cmp1_b <= int_tvalue1;      -- for testing
245      int_cmp1_b <= CONV_INTEGER(cmp1_in2);
246      int_cmp2_a <= int_cmp1_b;
247
248  -- Synchronous Comparator (Function "A < B",and "A >= B")
249  -- Main(comparating) section
250  process (clk,inhibit) begin
251      -- Priority 'Inhibit(Fixing logics)' function --
252      if (inhibit = '1') then
253          pre_outla <= pre_outla;
254          pre_outlb <= pre_outlb;
255      elsif (clk'event and clk = '0') then
256      -- Comparation 'int_cmp1_a'(A) and 'int_cmp1_b'(B)
257          if (int_cmp1_a < int_cmp1_b) then
258              pre_outla <= '1';
259              pre_outlb <= '0';
260          else
261              pre_outla <= '0';
262              pre_outlb <= '1';
263          end if;
264      end if;
265  end process;
266  ---************************************************---
```

図 3.　コンパレータ 1 の VHDL ソース

2.2　12bit プリセッタブル・ダウンカウンタ

TDC に接続する出力パルスのインターバルを生成する 12bit プリセッタブル・ダウンカウンタは、ロード入力 LD が”0”の時に 12bit ラッチデータ Data をプリセット値とする。イネーブル入力 ENB が”1”であればクロック CK の立ち上がりでダウンカウントする。カウント値が 0 になるとキャリー出力 RC を”0”にする。

12bit プリセッタブル・ダウンカウンタの VHDL ソースを図 4 に示す。

```
214  ---************************************************---
215  -- 12bit Presetable Down Counter with Carry --
216  -- Asynchronous preset and down counting section--
217  process (int_clk,load,int_bfq) begin
218      if (load = '0') then
219          int_value <= int_cmp2_b;
220      elsif (int_bfq = '1') then
221          if (int_clk'event and int_clk = '1') then
222              int_value <= int_value - 1;
223          end if;
224      end if;
225  end process;
226
227  -- Carry control section --
228  process (int_value) begin
229      if ((int_value = 0) and (int_bfq = '1')) then
230          int_precarry <= '0';
231      else
232          int_precarry <= '1';
233      end if;
234  end process;
235  ---************************************************---
```

図 4.12bit プリセッタブル・ダウンカウンタの VHDL ソース

後述の分周器で分周されたシステムクロックかロード入力かイネーブル入力のいずれかの変化で駆動する process 構文(217-225 行)では、ロード入力が”0”であればパルス波高値をプリセット値としてロードする。イネーブル入力が”1”の時、クロックの立ち上がりで 1 つダウンカウントする。カウンタの値の変化で駆動する process 構文(228-234 行)では、カウンタの値が 0 且つイネーブル入力が”1”の時にキャリー出力を”0”に、その他の状態では”1”にする。

2.3　分周器(Divider)

12bit プリセッタブル・ダウンカウンタに供給するクロックを生成する分周器は、入力されるシステムクロック CK を 1/1、1/2、1/4、1/8 分周し、外部スイッチで選択した分周値のクロックを出力する。

分周器の VHDL ソースを図 5 に示す。システムクロックと 4bit 分周値入力で駆動する process 構文において、システムクロックの立ち上がりで 1 つアップカウントする 4bit カウンタを構成する(169～176 行)。また、4bit 分周値入力の値に応じて、出力をシステムクロックもしくは 4bit カウンタのいずれかの bit に接続するセレクタ回路を構成する(179-191 行)。

分周値入力が想定する 4 つ以外の場合、出力が不定になる恐れがある。それを回避するため、セレクタ回路で現在の分周値を示すフラグ int_flagdiv を設定しておく。分周値入力が想定以外の値では、int_flagdiv の値に応じて出力をシステムクロックもしくは 4bit カウンタのいずれかの bit に接続する(192-203 行)。

```
166    ---************************************************************---
167    -- 4bit Selectable Divider --
168    -- Dividing section --
169    process (clk,div_selin) begin
170        if (clk'event and clk = '1') then
171            if (int_divcnt = "1111") then
172                int_divcnt <= "0000";
173            else
174                int_divcnt <= int_divcnt + '1';
175            end if;
176        end if;
177
178    -- Seleting (pre)output section
179        case (div_selin) is
180            when "1110" =>
181                pre_divout <= clk;
182                int_flagdiv <= 0;   -- Select 'clk'(1/1 dividing)
183            when "1101" =>
184                pre_divout <= int_divcnt(0);
185                int_flagdiv <= 1;   -- Select 1/2 dividing
186            when "1011" =>
187                pre_divout <= int_divcnt(1);
188                int_flagdiv <= 2;   -- Select 1/4 dividing
189            when "0111" =>
190                pre_divout <= int_divcnt(2);
191                int_flagdiv <= 3;   -- Select 1/8 dividing
192            when others =>
193                case (int_flagdiv) is
194                    when 0 =>
195                        pre_divout <= clk;
196                    when 1 =>
197                        pre_divout <= int_divcnt(0);
198                    when 2 =>
199                        pre_divout <= int_divcnt(1);
200                    when 3 =>
201                        pre_divout <= int_divcnt(2);
202                end case;
203                int_flagdiv <= int_flagdiv;
204        end case;
205
206    end process;
207    ---************************************************************---
```

図 5.分周器の VHDL ソース

3　シミュレータによるディジタル回路の動作検証

試作したカメレオン USB(CPLD)の動作検証では、ファンクション・ジェネレータや外部スイッチ回路を基板に接続し、入力条件を変化させてその応答を観測する手法を取る。ハードウェア記述言語で構成したディジタル回路はソースの変更で容易に回路構成を変更できるが、そのたびに前述の手法で動作検証を繰り返すのは非効率的である。また、観測したい信号線はすべて CPLD の I/O ピンを通して外部に引き出す必要がある。今回は従来の手法に加えて、シミュレーションによるパソコン上での動作検証を行った。

シミュレータには、Xilinx 社から無償提供されている ModelSim(図 6)[5]を用いた。電子回路シミュレータでは Spice が有名であるが、ModelSim も回路に入力する信号を定義してその出力を検証するツールである。Spice では回路構成を示すファイルであるネットリストをテキストエディタや回路図エディタで作成するのに対し、ModelSim は入力信号を定義するテストベンチと称するテキストファイルと、同じくテキストファイルである検証対象の VHDL(もしくは Verilog-HDL)ソースを使用する。

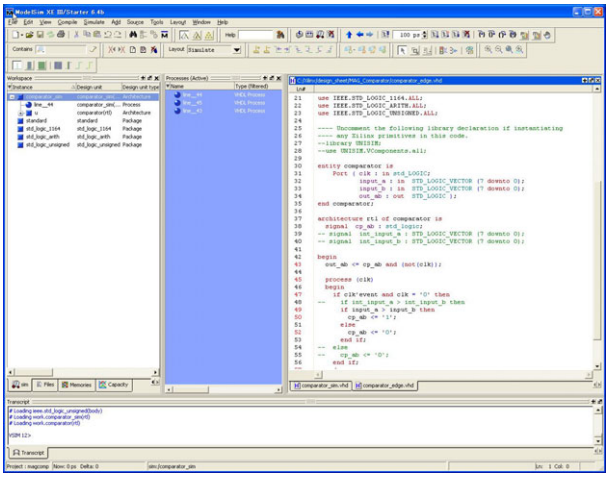

図 6.ディジタル回路シミュレータ ModelSim

今回製作したマグニチュード・コンパレータについてのテストベンチを図 7 に示す。テストベンチは VHDL もしくは Verilog-HDL の文法を使用する。図 9 は VHDL の文法を使用したもので、まずライブラリを読み込み(6-9 行)、回路モジュールを定義する(12-17 行)。続いて回路モジュールの入出力を定義して(21-27 行)、内部信号の定義や(29-32 行)信号の振る舞い(45 行-)を記述する。

図 7 のテストベンチをシミュレーションした結果を図 8-1 と 8-2 に示す。波形は上から順にシステムクロック、出力、データ入力 A、データ入力 B である。A>B が成立する時、システムクロックの立ち下がりで出力が”1”になり、システムクロックの次の立ち上がりで”0”になる出力が得られている。このように、テストベンチを使用することで、あたかも各種外部入力を接続してロジックアナライザを使用しているように回路の入出力を表示・検証できる。

C:/Xilinx/design_sheet/MAG_Comparator/comparator_sim.vhd

File Edit View Tools Window

```
Ln#
 3  -- Module Name:    comparator - rtl - simulation file
 4  --
 5  ----------------------------------------------------------------------------
 6  library IEEE;
 7  use IEEE.STD_LOGIC_1164.ALL;
 8  use IEEE.STD_LOGIC_ARITH.ALL;
 9  use IEEE.STD_LOGIC_UNSIGNED.ALL;
10
11
12  entity comparator_sim is
13  --    Port ( clk : in std_LOGIC;
14  --     input_a : in  STD_LOGIC_VECTOR (7 downto 0);
15  --           input_b : in  STD_LOGIC_VECTOR (7 downto 0);
16  --           out_ab : out  STD_LOGIC );
17  end comparator_sim;
18
19  architecture rtl of comparator_sim is
20
21    component comparator
22      port (
23        clk  : in std_LOGIC;
24        input_a : in  STD_LOGIC_VECTOR (7 downto 0);
25        input_b : in  STD_LOGIC_VECTOR (7 downto 0);
26        out_ab : out  STD_LOGIC );
27    end component;
28
29    signal  cclk : std_logic;
30    signal  out_ab : std_logic;
31    signal  int_input_a : STD_LOGIC_VECTOR (7 downto 0);
32    signal  int_input_b : STD_LOGIC_VECTOR (7 downto 0);
33
34  begin
35  U: comparator
36    port map (
37      clk  => cclk,
38      out_ab => out_ab,
39      input_a => int_input_a,
40      input_b => int_input_b
41      );
42
43    process
44    begin
45      int_input_a <= "00001111";
46      int_input_b <= "00001111";
47      cclk <= '1'; wait for 10ns;
48      cclk <= '0'; wait for 10ns;
49
50      cclk <= '1'; wait for 5ns;
51      int_input_b <= "00001110"; wait for 5ns;
52      cclk <= '0'; wait for 10ns;
53
54      cclk <= '1'; wait for 5ns;
55      int_input_b <= "00010000"; wait for 5ns;
```

Ln: 19 Col: 0 READ

図 7.マグニチュード・コンパレータのテストベンチ

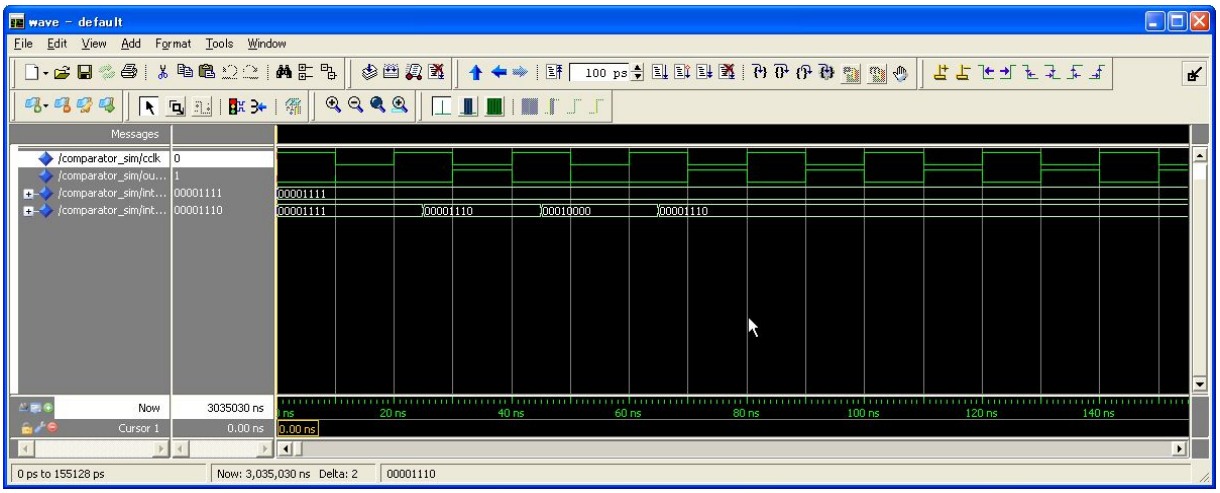

図 8-1.　ModelSim でのシミュレーションの様子(1)

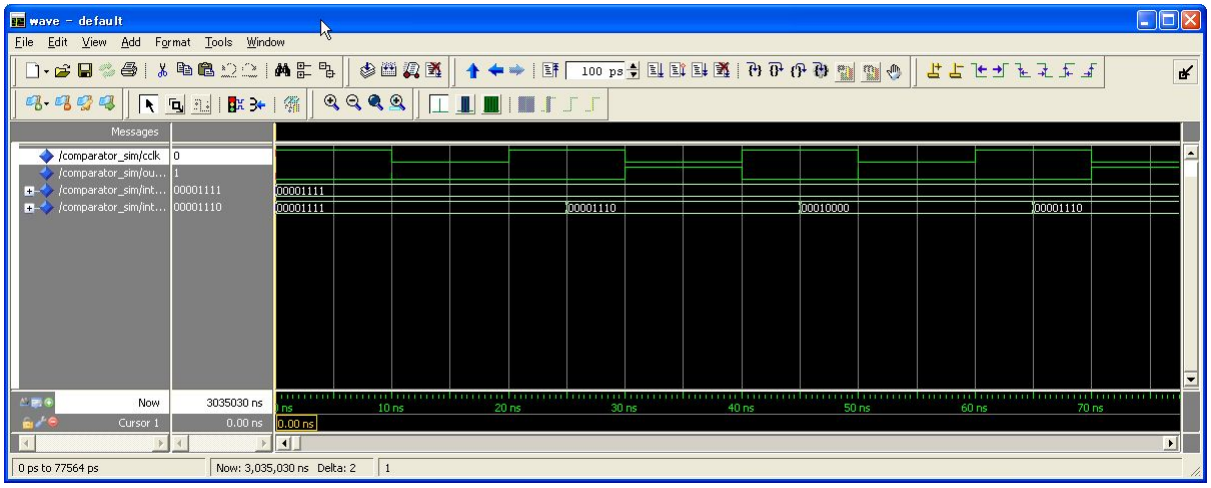

図 8-2.　ModelSim でのシミュレーションの様子(2)

4　総括

CPLD を用いることで、標準ロジック IC では実現が難しい特別な機能を有するディジタル回路を VHDL により柔軟に構築できた。また、シミュレーションを導入することで動作検証をより効率的に行えた。

複数の回路ブロックを同時に駆動させたため、ある回路ブロックの出力論理のミスが回路全体に波及して動作に支障をきたす事態に遭遇した。複数の回路ブロックを正確に動作させるためには、ソースの一部を修正するにしても常に回路全体の動作を俯瞰する必要性があると実感した。

参考文献

[1]　豊田朋範、河本充司、”USB 接続 NMR 用高速高分解能データロガーの開発”、平成 20 年度京都大学総合技術研究会報告集第一分冊、平成 20 年 3 月、p190-191

[2]　XC95144XL Data Sheet, Xilinx Inc.

[3]　http://optimize.ath.cx/cusb/index.html

[4]　AD9235 Data Sheet, Analog Devices Inc.

[5]　芹井滋喜、黒毛利　学、”はじめよう！ディジタル回路シミュレーション”、トランジスタ技術 2003 年 5 月号別冊付録、平成 15 年 5 月