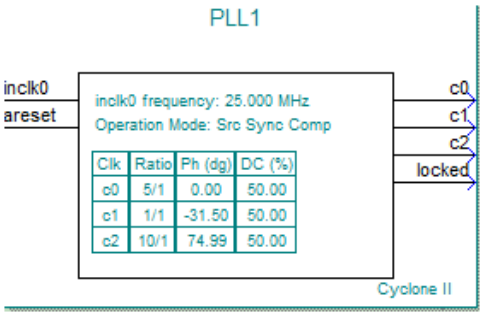
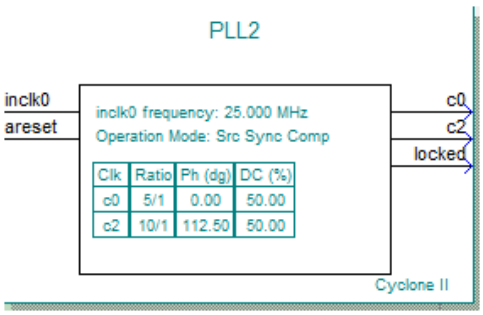
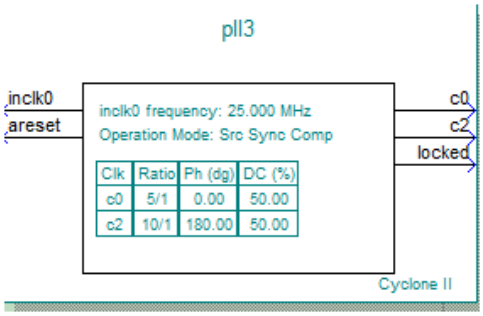
Slog V1.3 Timing-PLL



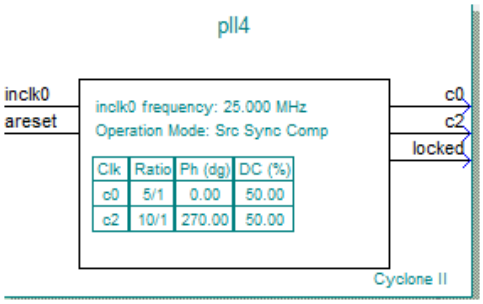
C0 – 125 MHZ 0ns delay
C1- 25 MHz -3,5ns
C2- 250 MHz 0,83 ns



C0 – 125 MHZ 0ns delay
C1- n.a.
C2- 250 MHz 1,25 ns



C0 – 125 MHZ 0ns delay
C1- n.a.
C2- 250 MHz 2,0ns



C0 – 125 MHZ 0ns delay
C1- n.a.
C2- 250 MHz 3,0ns

PLL1 (C0) -> ddin:CH1In1 und ddin:Ch2In1 und Clk125
PLL1 (C1) -> ClkCPU und Clk125
PLL1 (C2) -> CLK_ADC1

PLL2 (C0) -> ddin:CH1In2 und ddin:Ch2In2
PLL2 (C2) -> CLK_ADC2

PLL3 (C0) -> ddin:CH1In3 und ddin:Ch2In3
PLL3(C2) -> CLK_ADC3

PLL4 (C0) -> ddin:CH1In4 und ddin:Ch2In4
PLL4 (C2) -> CLK_ADC4

```
1
2   ------------------------------------------------------------------
    ----------
3   --  firmware for FPGA CycloneII EP2C35F484. For
    W2000                        --
4   --  file:       
    InADC.vhd                                                --
5   --  Version:    
    0.0.4                                                       --
6   --  Date:       may - june
    2008                                          --
7   --  Autor:      
    slog                                                        --
8   --                                                                
            --
9   ------------------------------------------------------------------
    ----------
10  --                                                                
            --
11  --  This file is part of the HelloW2000
    project                                 --
12  --                                                                
            --
13  --  Copyright (C) 2008  slog
    (s.loginov@yahoo.com)                            --
14  --                                                                
            --
15  --  This program is free software: you can redistribute it
    and/or modify  --
16  --  it under the terms of the GNU General Public License as
    published by  --
17  --  the Free Software Foundation, either version 3 of the
    License, or     --
18  --  (at your option) any later
    version.                                        --
19  --                                                                
            --
20  --  This program is distributed in the hope that it will be
    useful,       --
21  --  but WITHOUT ANY WARRANTY; without even the implied warranty
    of        --
22  --  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
    the         --
23  --  GNU General Public License for more
    details.                                --
24  --                                                                
            --
25  --  You should have received a copy of the GNU General Public
    License     --
26  --  along with this program.  If not, see
    <http://www.gnu.org/licenses/>. --
27  ------------------------------------------------------------------
    ----------
28
29
30  -- Inputs From ADC
31  -- 32 bits 250MHz SDR -> 64 bits 125MHz
```

```
32
33  library ieee;
34  use ieee.std_logic_1164.all;
35  --use ieee.numeric_std.all;
36  use ieee.std_logic_arith.all;
37  use ieee.std_logic_unsigned.all;
38
39  ENTITY InADC IS
40
41  PORT (
42  -- Inputs
43      --CLK
44      clk1    : in std_logic;
45      clk2    : in std_logic;
46      clk3    : in std_logic;
47      clk4    : in std_logic;
48      Reset   : IN STD_LOGIC;
49      --ADC
50      ADC1CLK : out std_logic;
51      ADC2CLK : out std_logic;
52      ADC3CLK : out std_logic;
53      ADC4CLK : out std_logic;
54      Ch1ADC1 : in std_logic_vector (7 DOWNTO 0);
55      Ch1ADC2 : in std_logic_vector (7 DOWNTO 0);
56      Ch1ADC3 : in std_logic_vector (7 DOWNTO 0);
57      Ch1ADC4 : in std_logic_vector (7 DOWNTO 0);
58      Ch2ADC1 : in std_logic_vector (7 DOWNTO 0);
59      Ch2ADC2 : in std_logic_vector (7 DOWNTO 0);
60      Ch2ADC3 : in std_logic_vector (7 DOWNTO 0);
61      Ch2ADC4 : in std_logic_vector (7 DOWNTO 0);
62
63  -- Outputs
64      Locked  : out std_logic;
65      clk125  : out std_logic;
66      Ch1Data : out std_logic_vector (63 DOWNTO 0);
67      Ch2Data : out std_logic_vector (63 DOWNTO 0)
68      );
69  END InADC;
70
71  ARCHITECTURE RTL OF InADC IS
72
73  component pll1
74      PORT
75      (
76      areset  : IN STD_LOGIC  := '0';
77      inclk0  : IN STD_LOGIC  := '0';
78      c0      : OUT STD_LOGIC ;
79      c2      : OUT STD_LOGIC ;
80      locked  : OUT STD_LOGIC
81      );
82  end component;
83
84  component pll2
85      PORT
86      (
87          areset  : IN STD_LOGIC  := '0';
88          inclk0  : IN STD_LOGIC  := '0';
```

```
 89          c0      : OUT STD_LOGIC ;
 90          c2      : OUT STD_LOGIC ;
 91          locked  : OUT STD_LOGIC
 92  );
 93  end component;
 94
 95  component pll3
 96      PORT
 97      (
 98          areset  : IN STD_LOGIC  := '0';
 99          inclk0  : IN STD_LOGIC  := '0';
100          c0      : OUT STD_LOGIC ;
101          c2      : OUT STD_LOGIC ;
102          locked  : OUT STD_LOGIC
103      );
104  end component;
105
106  component pll4
107      PORT
108      (
109          areset  : IN STD_LOGIC  := '0';
110          inclk0  : IN STD_LOGIC  := '0';
111          c0      : OUT STD_LOGIC ;
112          c2      : OUT STD_LOGIC ;
113          locked  : OUT STD_LOGIC
114      );
115  end component;
116
117  component ddin
118      PORT
119      (
120          datain      : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
121          inclock     : IN STD_LOGIC ;
122          dataout_h   : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
123          dataout_l   : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
124      );
125  end component;
126
127
128  signal locked1      : std_logic;
129  signal locked2      : std_logic;
130  signal locked3      : std_logic;
131  signal locked4      : std_logic;
132  signal clk125_1     : std_logic;
133  signal clk125_2     : std_logic;
134  signal clk125_3     : std_logic;
135  signal clk125_4     : std_logic;
136  signal Ch1Reg       : std_logic_vector (63 downto 0);
137  signal Ch2reg       : std_logic_vector (63 downto 0);
138  signal Ch1ADC1H     : std_logic_vector (7 downto 0);
139  signal Ch1ADC1L     : std_logic_vector (7 downto 0);
140  signal Ch1ADC2H     : std_logic_vector (7 downto 0);
141  signal Ch1ADC2L     : std_logic_vector (7 downto 0);
142  signal Ch1ADC3H     : std_logic_vector (7 downto 0);
143  signal Ch1ADC3L     : std_logic_vector (7 downto 0);
144  signal Ch1ADC4H     : std_logic_vector (7 downto 0);
145  signal Ch1ADC4L     : std_logic_vector (7 downto 0);
```

```
146   signal Ch2ADC1H     : std_logic_vector (7 downto 0);
147   signal Ch2ADC1L     : std_logic_vector (7 downto 0);
148   signal Ch2ADC2H     : std_logic_vector (7 downto 0);
149   signal Ch2ADC2L     : std_logic_vector (7 downto 0);
150   signal Ch2ADC3H     : std_logic_vector (7 downto 0);
151   signal Ch2ADC3L     : std_logic_vector (7 downto 0);
152   signal Ch2ADC4H     : std_logic_vector (7 downto 0);
153   signal Ch2ADC4L     : std_logic_vector (7 downto 0);
154
155
156   BEGIN
157
158   PLL_1 : PLL1 PORT MAP (
159           areset  => Reset,
160           inclk0  => clk1,
161           c0      => clk125_1,
162           c2      => ADC1CLK,
163           locked  => locked1
164       );
165
166   PLL_2 : PLL2 PORT MAP (
167           areset  => Reset,
168           inclk0  => clk2,
169           c0      => clk125_2,
170           c2      => ADC2CLK,
171           locked  => locked2
172       );
173
174   PLL_3 : PLL3 PORT MAP (
175           areset  => Reset,
176           inclk0  => clk3,
177           c0      => clk125_3,
178           c2      => ADC3CLK,
179           locked  => locked3
180       );
181
182   PLL_4 : PLL4 PORT MAP (
183           areset  => Reset,
184           inclk0  => clk4,
185           c0      => clk125_4,
186           c2      => ADC4CLK,
187           locked  => locked4
188       );
189
190       Locked <= locked1 OR locked2 OR locked3 OR locked4;
191       clk125 <= clk125_1;
192
193   Ch1In1: ddin    PORT MAP (
194           datain      => Ch1ADC1,
195           inclock     => clk125_1,
196           dataout_h   => Ch1ADC1H,
197           dataout_l   => Ch1ADC1L
198       );
199   Ch1In2: ddin    PORT MAP (
200           datain      => Ch1ADC2,
201           inclock     => clk125_2,
202           dataout_h   => Ch1ADC2H,
```

```
203           dataout_l   => Ch1ADC2L
204       );
205   Ch1In3: ddin    PORT MAP (
206           datain      => Ch1ADC3,
207           inclock     => clk125_3,
208           dataout_h   => Ch1ADC3H,
209           dataout_l   => Ch1ADC3L
210       );
211   Ch1In4: ddin    PORT MAP (
212           datain      => Ch1ADC4,
213           inclock     => clk125_4,
214           dataout_h   => Ch1ADC4H,
215           dataout_l   => Ch1ADC4L
216       );
217   Ch2In1: ddin    PORT MAP (
218           datain      => Ch2ADC1,
219           inclock     => clk125_1,
220           dataout_h   => Ch2ADC1H,
221           dataout_l   => Ch2ADC1L
222       );
223   Ch2In2: ddin    PORT MAP (
224           datain      => Ch2ADC2,
225           inclock     => clk125_2,
226           dataout_h   => Ch2ADC2H,
227           dataout_l   => Ch2ADC2L
228       );
229   Ch2In3: ddin    PORT MAP (
230           datain      => Ch2ADC3,
231           inclock     => clk125_3,
232           dataout_h   => Ch2ADC3H,
233           dataout_l   => Ch2ADC3L
234       );
235   Ch2In4: ddin    PORT MAP (
236           datain      => Ch2ADC4,
237           inclock     => clk125_4,
238           dataout_h   => Ch2ADC4H,
239           dataout_l   => Ch2ADC4L
240       );
241
242
243       PROCESS (clk125_1, Reset, Ch1Reg, Ch2Reg)
244       BEGIN
245
246       Ch1Data <= Ch1Reg;
247       Ch2Data <= Ch2Reg;
248
249           IF (rising_edge(clk125_1)) THEN
250               IF (reset = '1') THEN
251                   Ch1Reg <= (OTHERS =>'0');
252                   Ch2Reg <= (OTHERS =>'0');
253               ELSE
254                   Ch1Reg <= Ch1ADC4H & Ch1ADC3H & Ch1ADC2H & ⏎
      Ch1ADC1H & Ch1ADC4L & Ch1ADC3L & Ch1ADC2L & Ch1ADC1L;
255                   Ch2Reg <= Ch2ADC4H & Ch2ADC3H & Ch2ADC2H & ⏎
      Ch2ADC1H & Ch2ADC4L & Ch2ADC3L & Ch2ADC2L & Ch2ADC1L;
256               END IF;
257           END IF;
```

```
258       END PROCESS;
259   END RTL;
260
```