FPGA IRRADIATION @ NPTC-MGH (Discussion Items)

Ray Mountain, Bin Gui,

JC Wang, Marina Artuso

Syracuse University

Outline / Discussion Items

- **Next Round Items**
  - Items
- **Test Framework**
  - Diagram
- **NIKHEF PLL Project**
  - Diagram, some clarification needed
- **PLL Project Integration**
  - Can we use current framework?

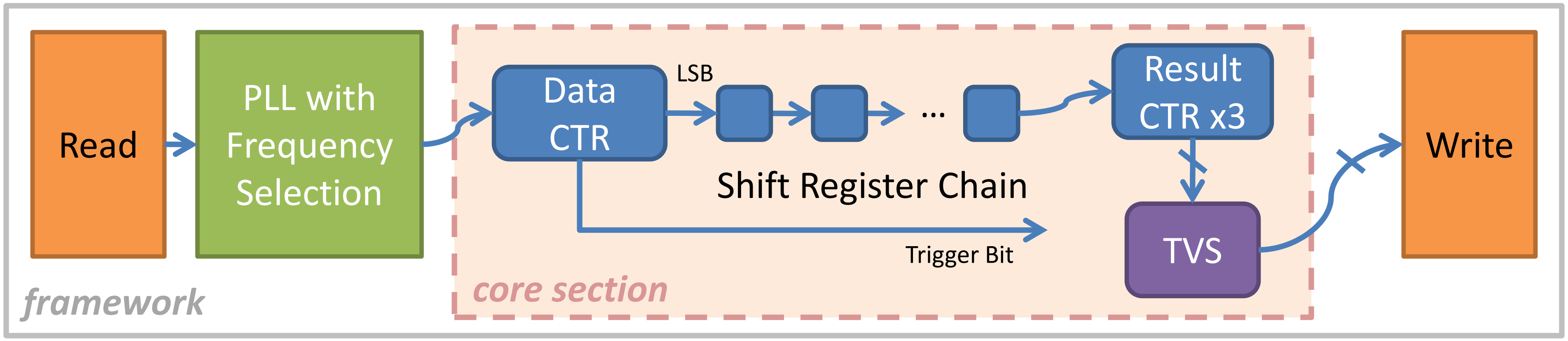
- **(NIKHEF PLL Code)**
  - For reference

Next Round – Items

- **Agenda**
  - PLL tests
  - Revisit RAM/ROM (additional statistics)
- **Need FPGAs**
  - Need to purchase additional A3PE1500s
  - Suggest: 2-4 A3PE1500s (~$150), 2-4 A3PE1500-2 (~$250) 320MHz
- **Other**
  - Modify current monitoring
  - Finish annealing

- **Schedule**
  - Possible slot early Dec, more next calendar year
- **Costs**
  - NPTC-MGH is ~700 $/beam-hour (comparable to IUCF)
  - Previous rounds: ~10K$ each (incl. travel costs)

Test Framework

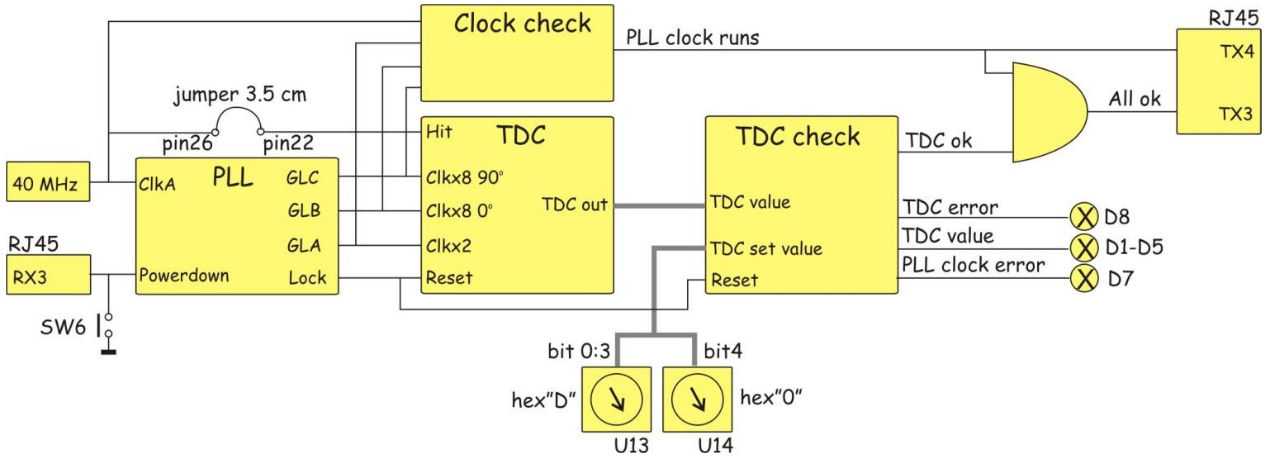


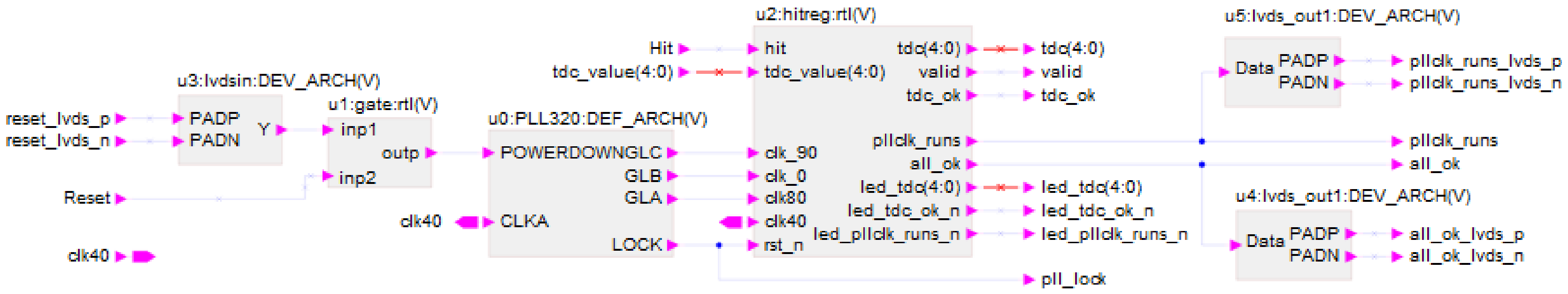
- **Framework**
  - Simple read/write to LabVIEW
  - Run for a fixed amount of time (~3.5 mins), uses a trigger bit (e.g. 2^33 MSB tunable)
  - Select different frequencies: 40, 80, 160, 240 MHz
  - Triple voting in the design with 3 result counters (internal)
- **Physical constraint to place main section in central region of FPGA**

- **General Procedure for data runs**
  - Configure (if needed)
  - From LabVIEW: Set CLK freq, and Start (enable PLL CLK)
  - Irradiate (wait for fixed dose in a given run)
  - End (disable PLL CLK), after beam off
  - Read out results (write to LabVIEW)
- **Same procedure for SEU/RAM/ROM, with different core sections to code**

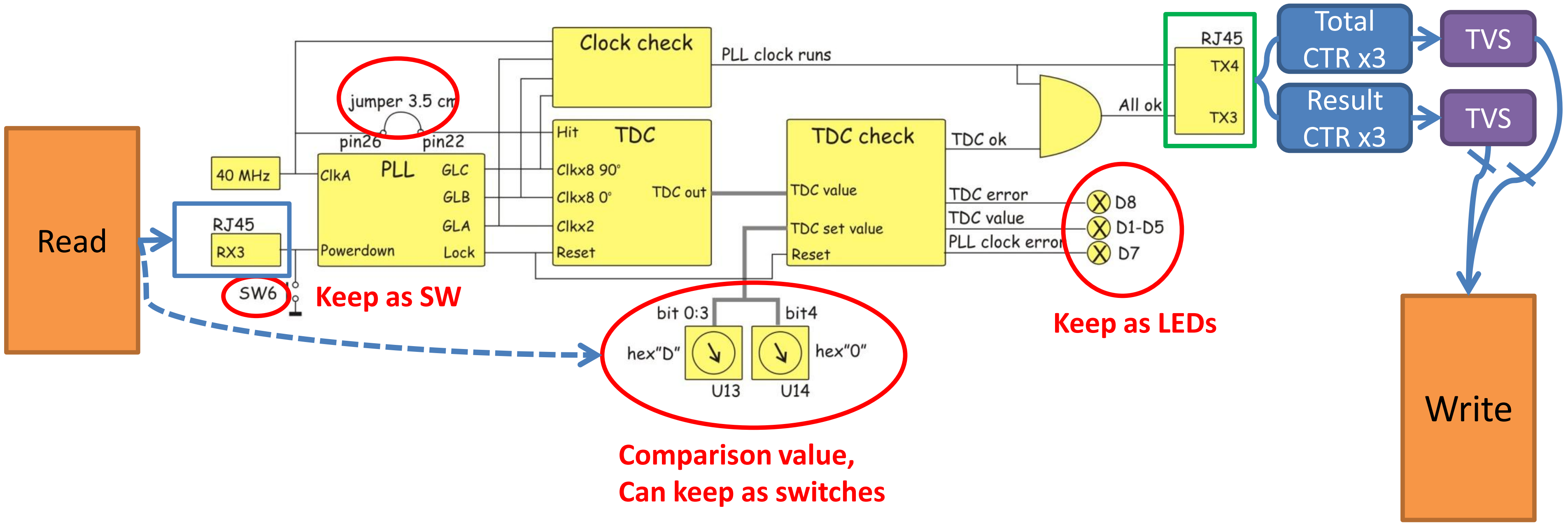
- ***Can adapt for PLL code?***

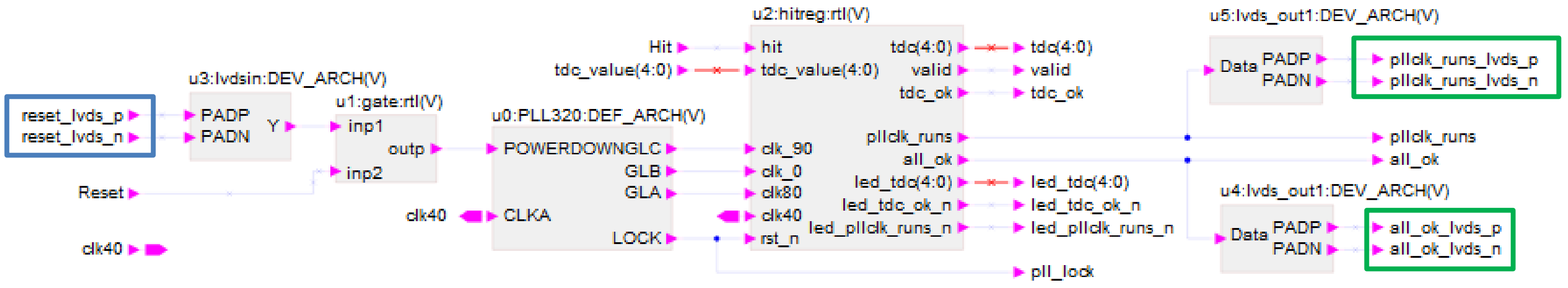
NIKHEF PLL Project



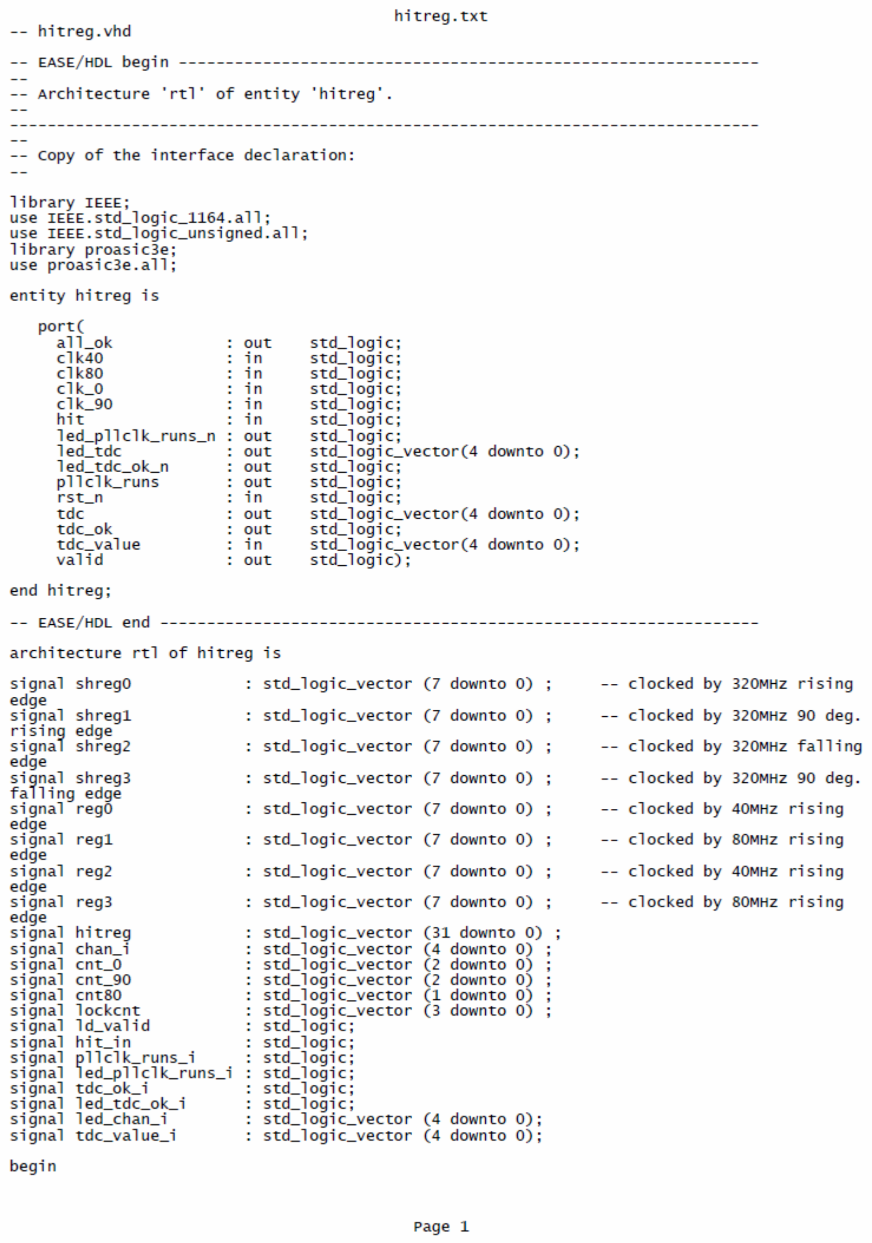


PLL Project Integration

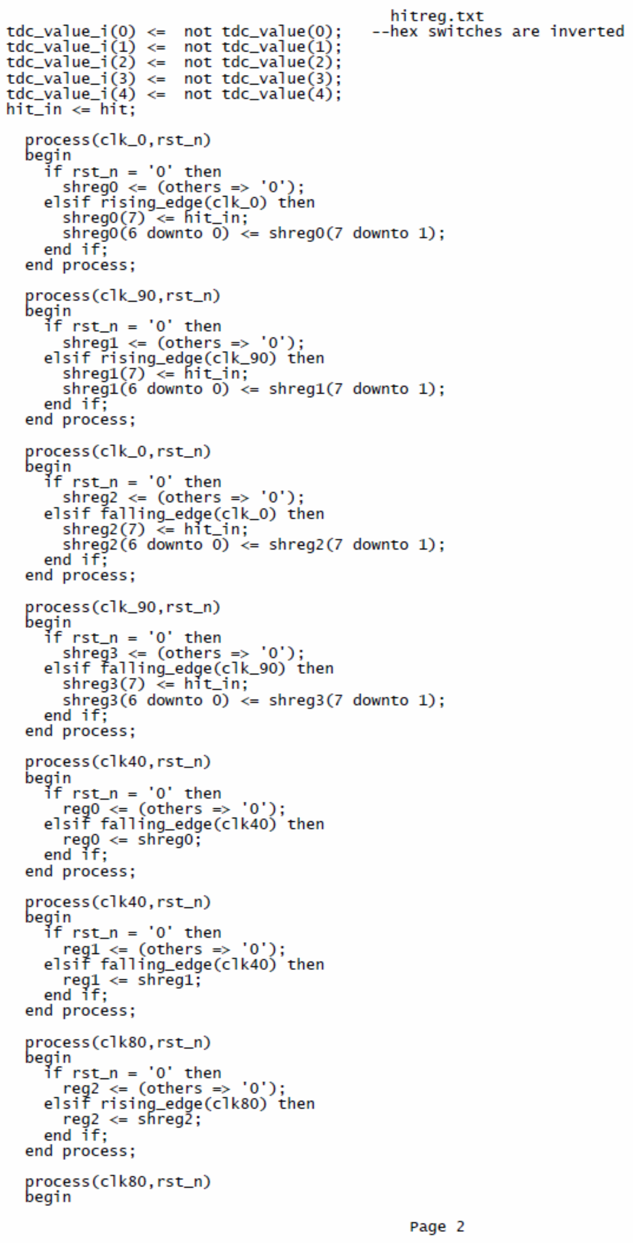




NIKHEF PLL Code (1)

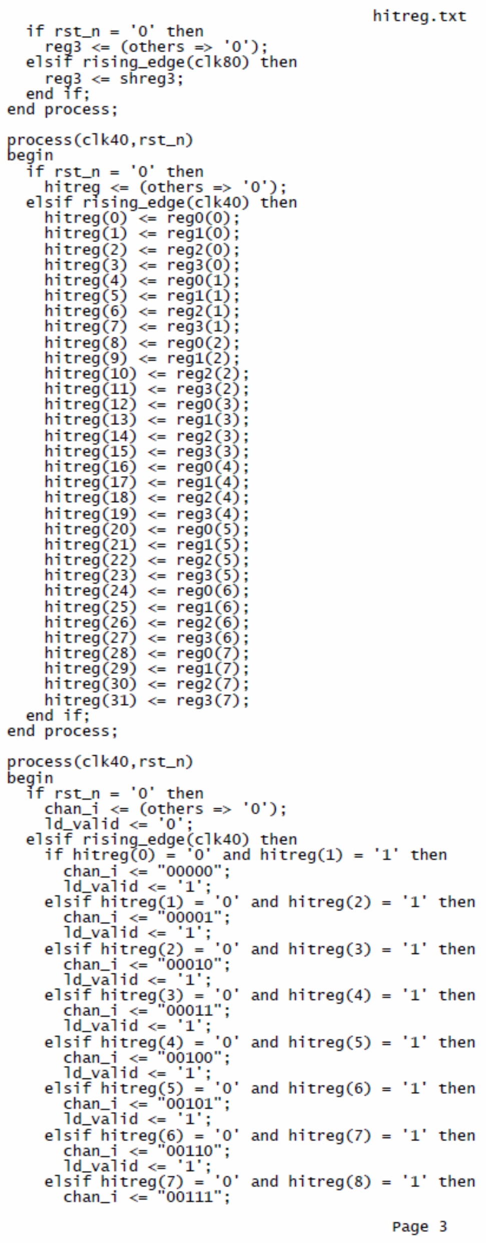


```
-- hitreg.vhd

-- EASE/HDL begin --------------------------------------------------------------
--
-- Architecture 'rtl' of entity 'hitreg'.
--
--------------------------------------------------------------------------------
--
-- Copy of the interface declaration:
--

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
library proasic3e;
use proasic3e.all;

entity hitreg is

   port(
     all_ok            : out    std_logic;
     clk40             : in     std_logic;
     clk80             : in     std_logic;
     clk_0             : in     std_logic;
     clk_90            : in     std_logic;
     hit               : in     std_logic;
     led_pllclk_runs_n : out    std_logic;
     led_tdc           : out    std_logic_vector(4 downto 0);
     led_tdc_ok_n      : out    std_logic;
     pllclk_runs       : out    std_logic;
     rst_n             : in     std_logic;
     tdc               : out    std_logic_vector(4 downto 0);
     tdc_ok            : out    std_logic;
     tdc_value         : in     std_logic_vector(4 downto 0);
     valid             : out    std_logic);

end hitreg;

-- EASE/HDL end ----------------------------------------------------------------

architecture rtl of hitreg is

signal shreg0           : std_logic_vector (7 downto 0) ;     -- clocked by 320MHz rising
edge
signal shreg1           : std_logic_vector (7 downto 0) ;     -- clocked by 320MHz 90 deg.
rising edge
signal shreg2           : std_logic_vector (7 downto 0) ;     -- clocked by 320MHz falling
edge
signal shreg3           : std_logic_vector (7 downto 0) ;     -- clocked by 320MHz 90 deg.
falling edge
signal reg0             : std_logic_vector (7 downto 0) ;     -- clocked by 40MHz rising
edge
signal reg1             : std_logic_vector (7 downto 0) ;     -- clocked by 80MHz rising
edge
signal reg2             : std_logic_vector (7 downto 0) ;     -- clocked by 40MHz rising
edge
signal reg3             : std_logic_vector (7 downto 0) ;     -- clocked by 80MHz rising
edge
signal hitreg           : std_logic_vector (31 downto 0) ;
signal chan_i           : std_logic_vector (4 downto 0) ;
signal cnt_0            : std_logic_vector (2 downto 0) ;
signal cnt_90           : std_logic_vector (2 downto 0) ;
signal cnt80            : std_logic_vector (1 downto 0) ;
signal lockcnt          : std_logic_vector (3 downto 0) ;
signal ld_valid         : std_logic;
signal hit_in           : std_logic;
signal pllclk_runs_i    : std_logic;
signal led_pllclk_runs_i : std_logic;
signal tdc_ok_i         : std_logic;
signal led_tdc_ok_i     : std_logic;
signal led_chan_i       : std_logic_vector (4 downto 0);
signal tdc_value_i      : std_logic_vector (4 downto 0);

begin
```



```
tdc_value_i(0) <=  not tdc_value(0);    --hex switches are inverted
tdc_value_i(1) <=  not tdc_value(1);
tdc_value_i(2) <=  not tdc_value(2);
tdc_value_i(3) <=  not tdc_value(3);
tdc_value_i(4) <=  not tdc_value(4);
hit_in <= hit;

  process(clk_0,rst_n)
  begin
    if rst_n = '0' then
      shreg0 <= (others => '0');
    elsif rising_edge(clk_0) then
      shreg0(7) <= hit_in;
      shreg0(6 downto 0) <= shreg0(7 downto 1);
    end if;
  end process;

  process(clk_90,rst_n)
  begin
    if rst_n = '0' then
      shreg1 <= (others => '0');
    elsif rising_edge(clk_90) then
      shreg1(7) <= hit_in;
      shreg1(6 downto 0) <= shreg1(7 downto 1);
    end if;
  end process;

  process(clk_0,rst_n)
  begin
    if rst_n = '0' then
      shreg2 <= (others => '0');
    elsif falling_edge(clk_0) then
      shreg2(7) <= hit_in;
      shreg2(6 downto 0) <= shreg2(7 downto 1);
    end if;
  end process;

  process(clk_90,rst_n)
  begin
    if rst_n = '0' then
      shreg3 <= (others => '0');
    elsif falling_edge(clk_90) then
      shreg3(7) <= hit_in;
      shreg3(6 downto 0) <= shreg3(7 downto 1);
    end if;
  end process;

  process(clk40,rst_n)
  begin
    if rst_n = '0' then
      reg0 <= (others => '0');
    elsif falling_edge(clk40) then
      reg0 <= shreg0;
    end if;
  end process;

  process(clk40,rst_n)
  begin
    if rst_n = '0' then
      reg1 <= (others => '0');
    elsif falling_edge(clk40) then
      reg1 <= shreg1;
    end if;
  end process;

  process(clk80,rst_n)
  begin
    if rst_n = '0' then
      reg2 <= (others => '0');
    elsif rising_edge(clk80) then
      reg2 <= shreg2;
    end if;
  end process;

  process(clk80,rst_n)
  begin
```

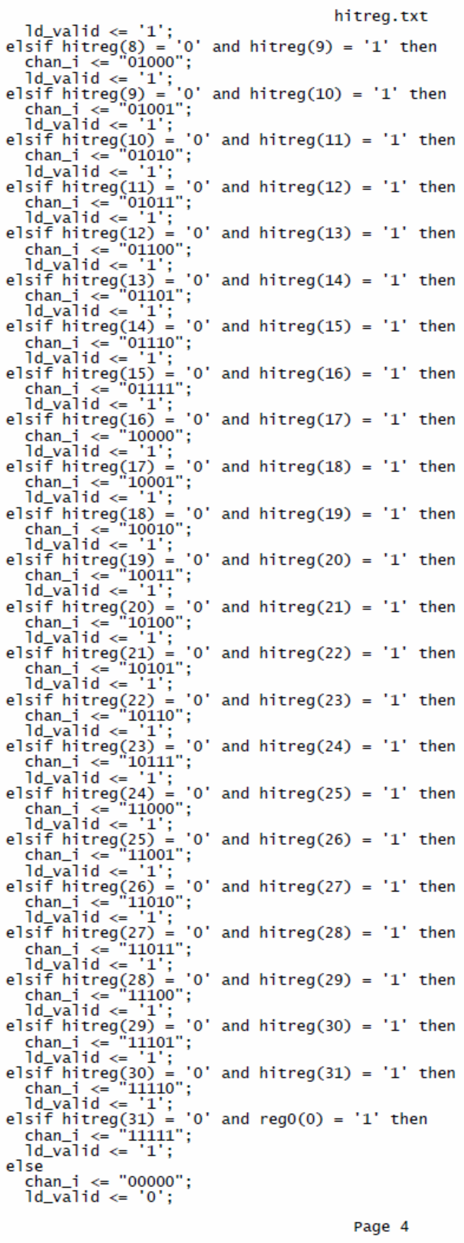
NIKHEF PLL Code (2)



hitreg.txt

```
  if rst_n = '0' then
    reg3 <= (others => '0');
  elsif rising_edge(clk80) then
    reg3 <= shreg3;
  end if;
end process;

process(clk40,rst_n)
begin
  if rst_n = '0' then
    hitreg <= (others => '0');
  elsif rising_edge(clk40) then
    hitreg(0)  <= reg0(0);
    hitreg(1)  <= reg1(0);
    hitreg(2)  <= reg2(0);
    hitreg(3)  <= reg3(0);
    hitreg(4)  <= reg0(1);
    hitreg(5)  <= reg1(1);
    hitreg(6)  <= reg2(1);
    hitreg(7)  <= reg3(1);
    hitreg(8)  <= reg0(2);
    hitreg(9)  <= reg1(2);
    hitreg(10) <= reg2(2);
    hitreg(11) <= reg3(2);
    hitreg(12) <= reg0(3);
    hitreg(13) <= reg1(3);
    hitreg(14) <= reg2(3);
    hitreg(15) <= reg3(3);
    hitreg(16) <= reg0(4);
    hitreg(17) <= reg1(4);
    hitreg(18) <= reg2(4);
    hitreg(19) <= reg3(4);
    hitreg(20) <= reg0(5);
    hitreg(21) <= reg1(5);
    hitreg(22) <= reg2(5);
    hitreg(23) <= reg3(5);
    hitreg(24) <= reg0(6);
    hitreg(25) <= reg1(6);
    hitreg(26) <= reg2(6);
    hitreg(27) <= reg3(6);
    hitreg(28) <= reg0(7);
    hitreg(29) <= reg1(7);
    hitreg(30) <= reg2(7);
    hitreg(31) <= reg3(7);
  end if;
end process;

process(clk40,rst_n)
begin
  if rst_n = '0' then
    chan_i <= (others => '0');
    ld_valid <= '0';
  elsif rising_edge(clk40) then
    if hitreg(0) = '0' and hitreg(1) = '1' then
      chan_i <= "00000";
      ld_valid <= '1';
    elsif hitreg(1) = '0' and hitreg(2) = '1' then
      chan_i <= "00001";
      ld_valid <= '1';
    elsif hitreg(2) = '0' and hitreg(3) = '1' then
      chan_i <= "00010";
      ld_valid <= '1';
    elsif hitreg(3) = '0' and hitreg(4) = '1' then
      chan_i <= "00011";
      ld_valid <= '1';
    elsif hitreg(4) = '0' and hitreg(5) = '1' then
      chan_i <= "00100";
      ld_valid <= '1';
    elsif hitreg(5) = '0' and hitreg(6) = '1' then
      chan_i <= "00101";
      ld_valid <= '1';
    elsif hitreg(6) = '0' and hitreg(7) = '1' then
      chan_i <= "00110";
      ld_valid <= '1';
    elsif hitreg(7) = '0' and hitreg(8) = '1' then
      chan_i <= "00111";
```

Page 3

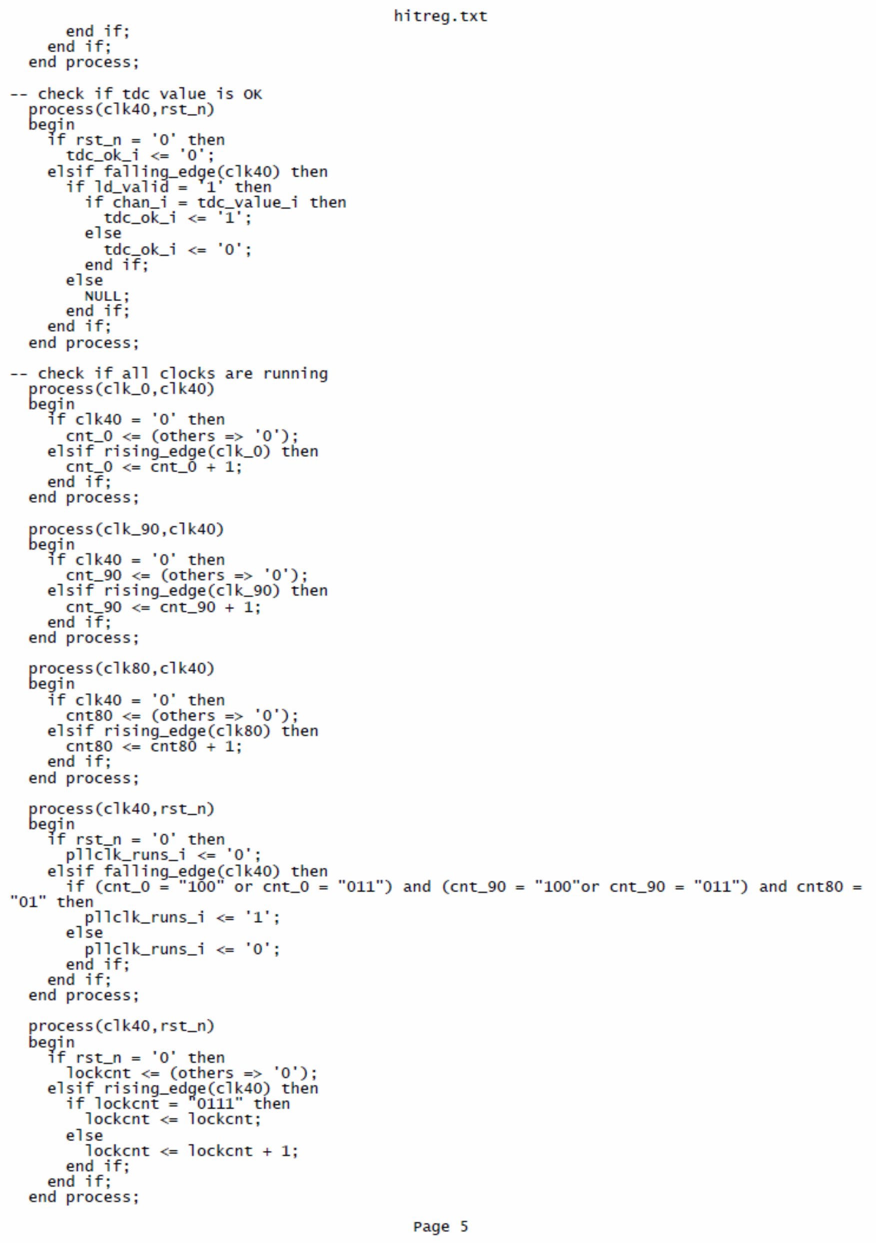


hitreg.txt

```
      ld_valid <= '1';
    elsif hitreg(8) = '0' and hitreg(9) = '1' then
      chan_i <= "01000";
      ld_valid <= '1';
    elsif hitreg(9) = '0' and hitreg(10) = '1' then
      chan_i <= "01001";
      ld_valid <= '1';
    elsif hitreg(10) = '0' and hitreg(11) = '1' then
      chan_i <= "01010";
      ld_valid <= '1';
    elsif hitreg(11) = '0' and hitreg(12) = '1' then
      chan_i <= "01011";
      ld_valid <= '1';
    elsif hitreg(12) = '0' and hitreg(13) = '1' then
      chan_i <= "01100";
      ld_valid <= '1';
    elsif hitreg(13) = '0' and hitreg(14) = '1' then
      chan_i <= "01101";
      ld_valid <= '1';
    elsif hitreg(14) = '0' and hitreg(15) = '1' then
      chan_i <= "01110";
      ld_valid <= '1';
    elsif hitreg(15) = '0' and hitreg(16) = '1' then
      chan_i <= "01111";
      ld_valid <= '1';
    elsif hitreg(16) = '0' and hitreg(17) = '1' then
      chan_i <= "10000";
      ld_valid <= '1';
    elsif hitreg(17) = '0' and hitreg(18) = '1' then
      chan_i <= "10001";
      ld_valid <= '1';
    elsif hitreg(18) = '0' and hitreg(19) = '1' then
      chan_i <= "10010";
      ld_valid <= '1';
    elsif hitreg(19) = '0' and hitreg(20) = '1' then
      chan_i <= "10011";
      ld_valid <= '1';
    elsif hitreg(20) = '0' and hitreg(21) = '1' then
      chan_i <= "10100";
      ld_valid <= '1';
    elsif hitreg(21) = '0' and hitreg(22) = '1' then
      chan_i <= "10101";
      ld_valid <= '1';
    elsif hitreg(22) = '0' and hitreg(23) = '1' then
      chan_i <= "10110";
      ld_valid <= '1';
    elsif hitreg(23) = '0' and hitreg(24) = '1' then
      chan_i <= "10111";
      ld_valid <= '1';
    elsif hitreg(24) = '0' and hitreg(25) = '1' then
      chan_i <= "11000";
      ld_valid <= '1';
    elsif hitreg(25) = '0' and hitreg(26) = '1' then
      chan_i <= "11001";
      ld_valid <= '1';
    elsif hitreg(26) = '0' and hitreg(27) = '1' then
      chan_i <= "11010";
      ld_valid <= '1';
    elsif hitreg(27) = '0' and hitreg(28) = '1' then
      chan_i <= "11011";
      ld_valid <= '1';
    elsif hitreg(28) = '0' and hitreg(29) = '1' then
      chan_i <= "11100";
      ld_valid <= '1';
    elsif hitreg(29) = '0' and hitreg(30) = '1' then
      chan_i <= "11101";
      ld_valid <= '1';
    elsif hitreg(30) = '0' and hitreg(31) = '1' then
      chan_i <= "11110";
      ld_valid <= '1';
    elsif hitreg(31) = '0' and reg0(0) = '1' then
      chan_i <= "11111";
      ld_valid <= '1';
    else
      chan_i <= "00000";
      ld_valid <= '0';
```

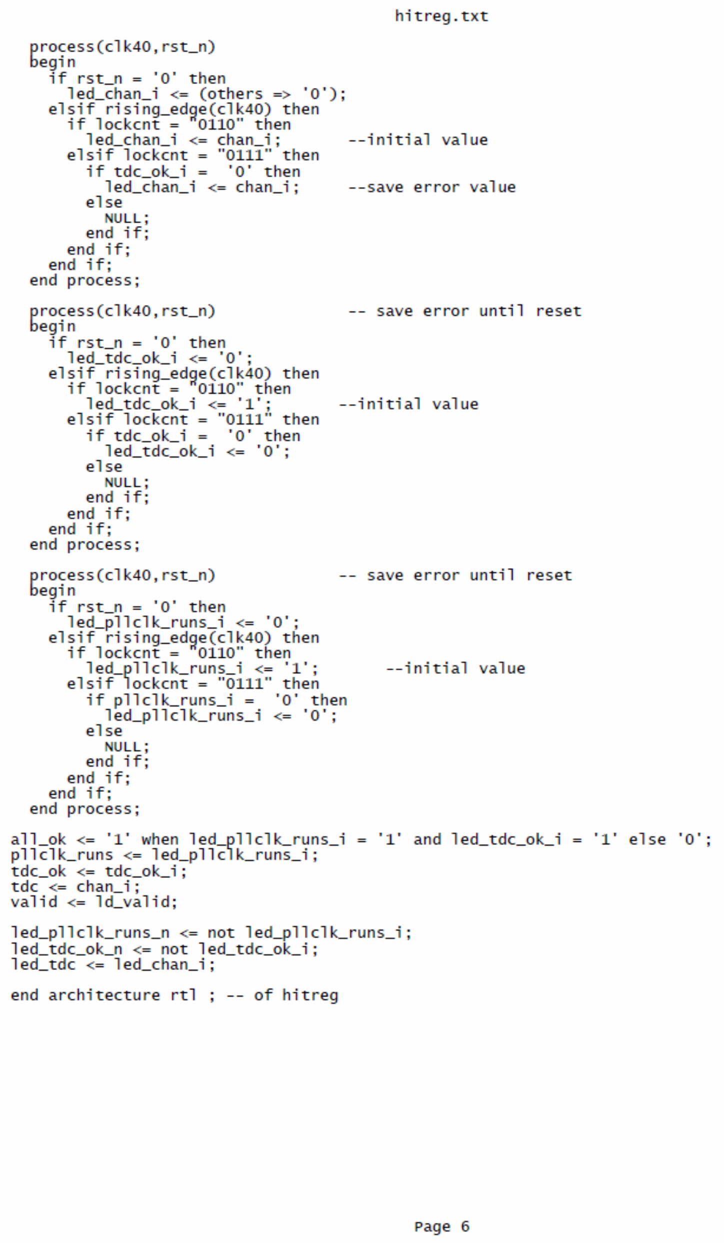
Page 4

NIKHEF PLL Code (3)


hitreg.txt

```
      end if;
    end if;
  end process;

-- check if tdc value is OK
  process(clk40,rst_n)
  begin
    if rst_n = '0' then
      tdc_ok_i <= '0';
    elsif falling_edge(clk40) then
      if ld_valid = '1' then
        if chan_i = tdc_value_i then
          tdc_ok_i <= '1';
        else
          tdc_ok_i <= '0';
        end if;
      else
        NULL;
      end if;
    end if;
  end process;

-- check if all clocks are running
  process(clk_0,clk40)
  begin
    if clk40 = '0' then
      cnt_0 <= (others => '0');
    elsif rising_edge(clk_0) then
      cnt_0 <= cnt_0 + 1;
    end if;
  end process;

  process(clk_90,clk40)
  begin
    if clk40 = '0' then
      cnt_90 <= (others => '0');
    elsif rising_edge(clk_90) then
      cnt_90 <= cnt_90 + 1;
    end if;
  end process;

  process(clk80,clk40)
  begin
    if clk40 = '0' then
      cnt80 <= (others => '0');
    elsif rising_edge(clk80) then
      cnt80 <= cnt80 + 1;
    end if;
  end process;

  process(clk40,rst_n)
  begin
    if rst_n = '0' then
      pllclk_runs_i <= '0';
    elsif falling_edge(clk40) then
      if (cnt_0 = "100" or cnt_0 = "011") and (cnt_90 = "100"or cnt_90 = "011") and cnt80 =
"01" then
        pllclk_runs_i <= '1';
      else
        pllclk_runs_i <= '0';
      end if;
    end if;
  end process;

  process(clk40,rst_n)
  begin
    if rst_n = '0' then
      lockcnt <= (others => '0');
    elsif rising_edge(clk40) then
      if lockcnt = "0111" then
        lockcnt <= lockcnt;
      else
        lockcnt <= lockcnt + 1;
      end if;
    end if;
  end process;
```

Page 5


hitreg.txt

```
  process(clk40,rst_n)
  begin
    if rst_n = '0' then
      led_chan_i <= (others => '0');
    elsif rising_edge(clk40) then
      if lockcnt = "0110" then
        led_chan_i <= chan_i;         --initial value
      elsif lockcnt = "0111" then
        if tdc_ok_i =  '0' then
          led_chan_i <= chan_i;       --save error value
        else
          NULL;
        end if;
      end if;
    end if;
  end process;

  process(clk40,rst_n)                -- save error until reset
  begin
    if rst_n = '0' then
      led_tdc_ok_i <= '0';
    elsif rising_edge(clk40) then
      if lockcnt = "0110" then
        led_tdc_ok_i <= '1';         --initial value
      elsif lockcnt = "0111" then
        if tdc_ok_i =  '0' then
          led_tdc_ok_i <= '0';
        else
          NULL;
        end if;
      end if;
    end if;
  end process;

  process(clk40,rst_n)               -- save error until reset
  begin
    if rst_n = '0' then
      led_pllclk_runs_i <= '0';
    elsif rising_edge(clk40) then
      if lockcnt = "0110" then
        led_pllclk_runs_i <= '1';         --initial value
      elsif lockcnt = "0111" then
        if pllclk_runs_i =  '0' then
          led_pllclk_runs_i <= '0';
        else
          NULL;
        end if;
      end if;
    end if;
  end process;

all_ok <= '1' when led_pllclk_runs_i = '1' and led_tdc_ok_i = '1' else '0';
pllclk_runs <= led_pllclk_runs_i;
tdc_ok <= tdc_ok_i;
tdc <= chan_i;
valid <= ld_valid;

led_pllclk_runs_n <= not led_pllclk_runs_i;
led_tdc_ok_n <= not led_tdc_ok_i;
led_tdc <= led_chan_i;

end architecture rtl ; -- of hitreg
```

Page 6