FPGA tutorial

Lecture 9

06.01.2021

Jochen Steinmann

III. Physikalisches Institut B

RWTH AACHEN UNIVERSITY

HAPPY NEW YEAR
2021

Organisatorical

Evaluation

Please take your time and evaluate this course.

https://www.campus.rwth-aachen.de/evasys/online.php?pswd=G7SF9

This link is for the FPGA course only.

Solution

ToDo from before Christmas

- A
  - Implement a matched filter for
    - 4 Samples of 8 bit ones
  - Test the result of this filter using different patterns
    - Response to
      - 1x 8bit
      - 2x 8bit
      - 3x 8bit
      - 4x 8bit
- B
  - Modify the matched filter, that you can change the coefficients from the testbench
  - **Do not change the array at once!**
  - Use:
    - One signal for enabling the „programming mode“
    - 3 bits for the address of the array
    - single 8 bit for the content of the array

    - Keep in mind: **Data should be transferred at the rising edge of the clock**

Test the response of the Matched Filter

```
architecture behavior of MatchedFilter is
    signal shift_reg : slv8_array_t(3 downto 0) := (others=>(others=>'0'));

    signal coeff_reg : slv8_array_t(3 downto 0) := (
                                                    0 => (others => '1'),
                                                    1 => (others => '1'),
                                                    2 => (others => '1'),
                                                    3 => (others => '1')
                                                    );
begin

    process(i_sl_CLK)
        -- 8 bit x 8 bit = 16 bit
        -- 3 bit (4) x 16 bit = 19 bit
        variable average : unsigned(18 downto 0) := (others => '0');
    begin
        if rising_edge(i_sl_CLK) then
            if (i_sl_en = '1') then

                -- do shifting of all input values
                for I in 3 downto 1 loop
                    shift_reg(I) <= shift_reg( I - 1);
                end loop;
                shift_reg(0) <= i_slv_data;

                -- reset avarage
                average := (others => '0');
                for I in 3 downto 0 loop
                    average := average + unsigned( shift_reg(I) ) * unsigned( coeff_reg(I) );
                end loop;

            end if;
        end if;

        o_slv_average <= std_logic_vector(average);

    end process;

end behavior;
```

one loop for the shift register

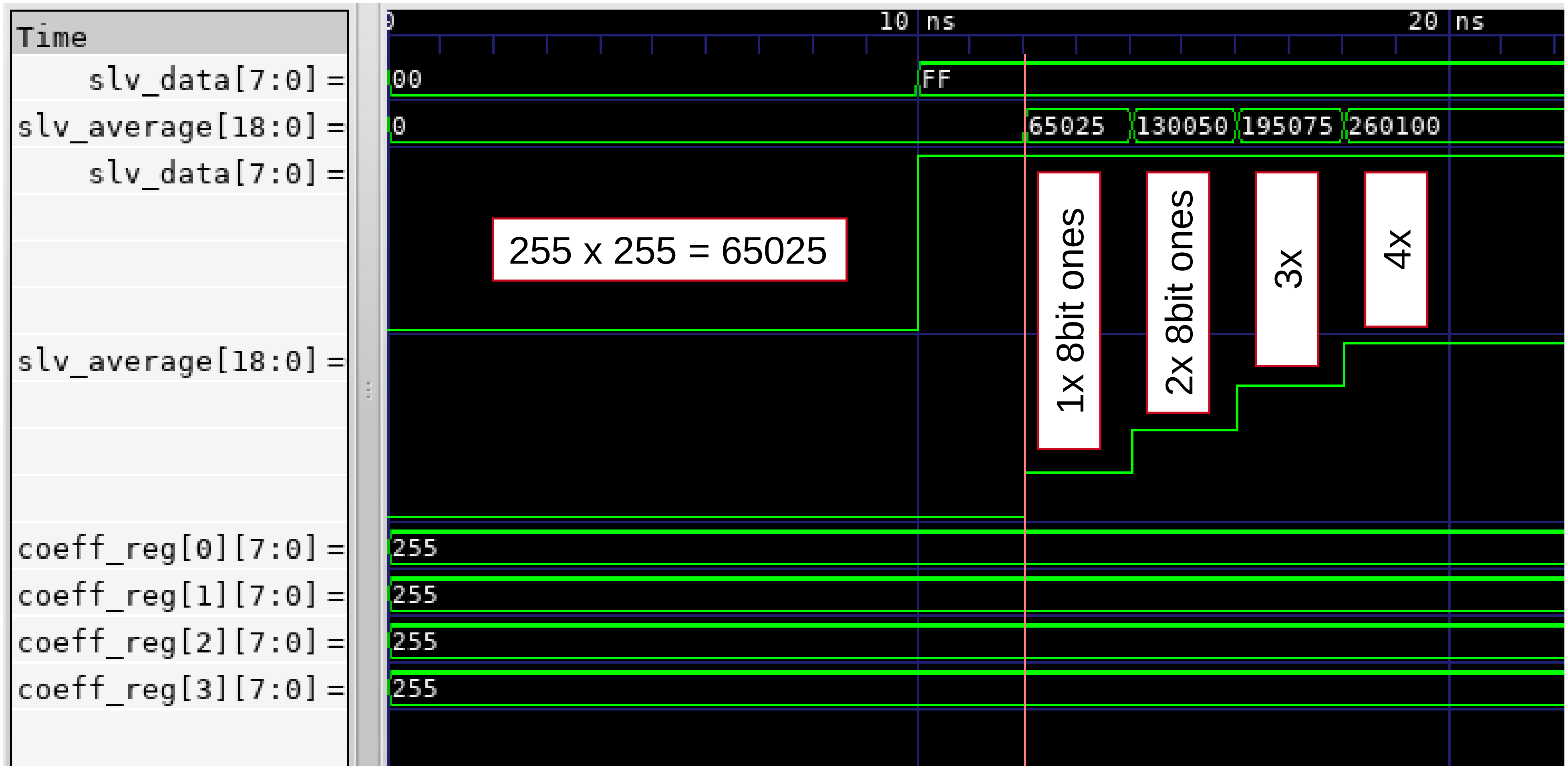
one loop for the FIR filter

Why 19 bits?

GHDL is assuming, that all variables can have their maximal value

Bit 18 will never been set to 1 in our application.

Test the response of the Matched Filter



Lecture 08b

Write coefficients during runtime…

```
    process(i_sl_CLK)
        -- 8 bit x 8 bit = 16 bit
        -- 3 bit (4) x 16 bit = 19 bit
        variable average : unsigned(18 downto 0) := (others => '0');
    begin
        if rising_edge(i_sl_CLK) then
            if (i_sl_en = '1') then

                -- do shifting of all input values
                for I in 3 downto 1 loop
                    shift_reg(I) <= shift_reg( I - 1);
                end loop;
                shift_reg(0) <= i_slv_data;

                -- reset avarage
                average := (others => '0');
                for I in 3 downto 0 loop
                    average := average + unsigned( shift_reg(I) ) * unsigned( coeff_reg(I) );
                end loop;
            else
                if i_sl_prog = '1' then
                    coeff_reg( to_integer(unsigned( i_slv_Taddr ) ) ) <= i_slv_Tvalue;
                end if;
            end if;
        end if;

        o_slv_average <= std_logic_vector(average);

    end process;

end behavior;
```

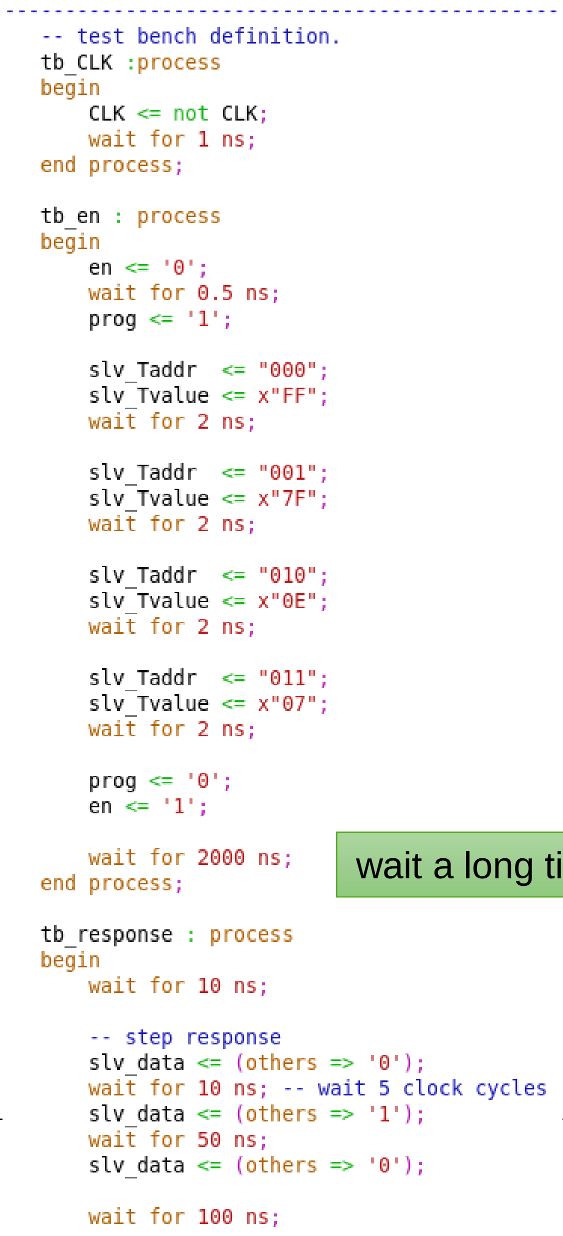
same as before

„Programming“ part

III. Physikalisches Institut B

RWTH AACHEN UNIVERSITY

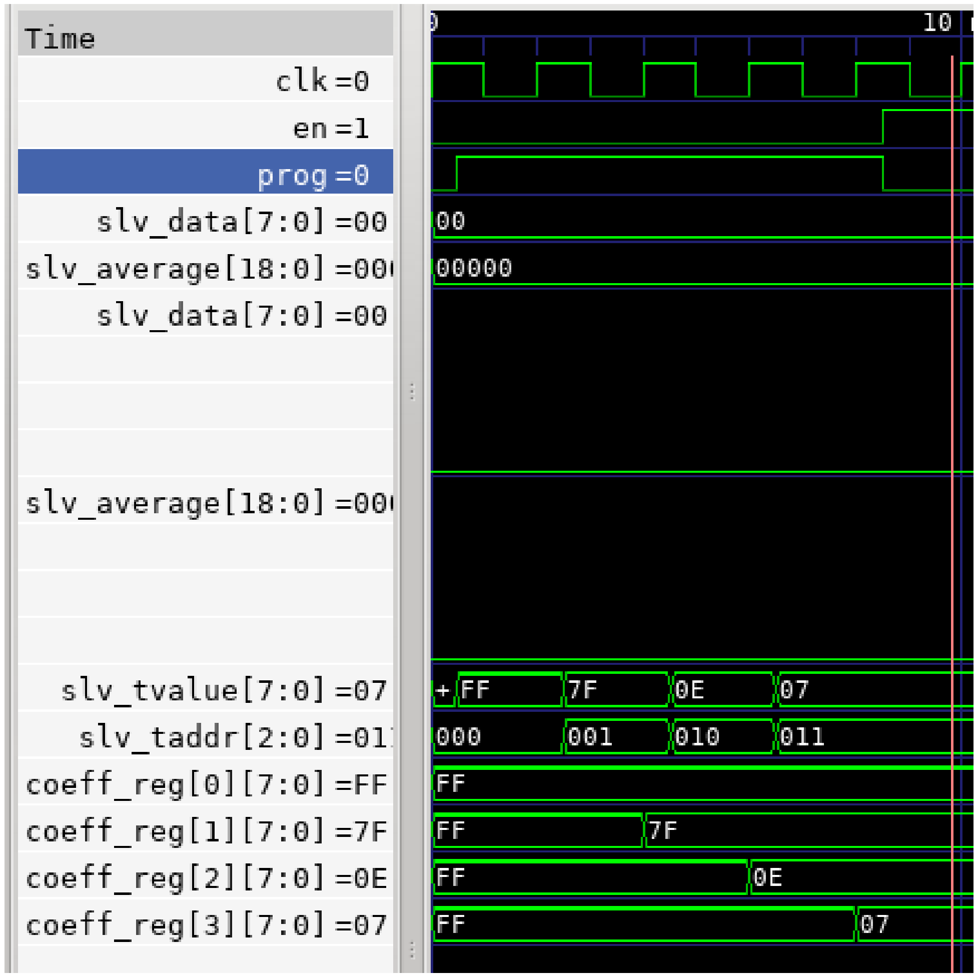
Write coefficients during runtime…



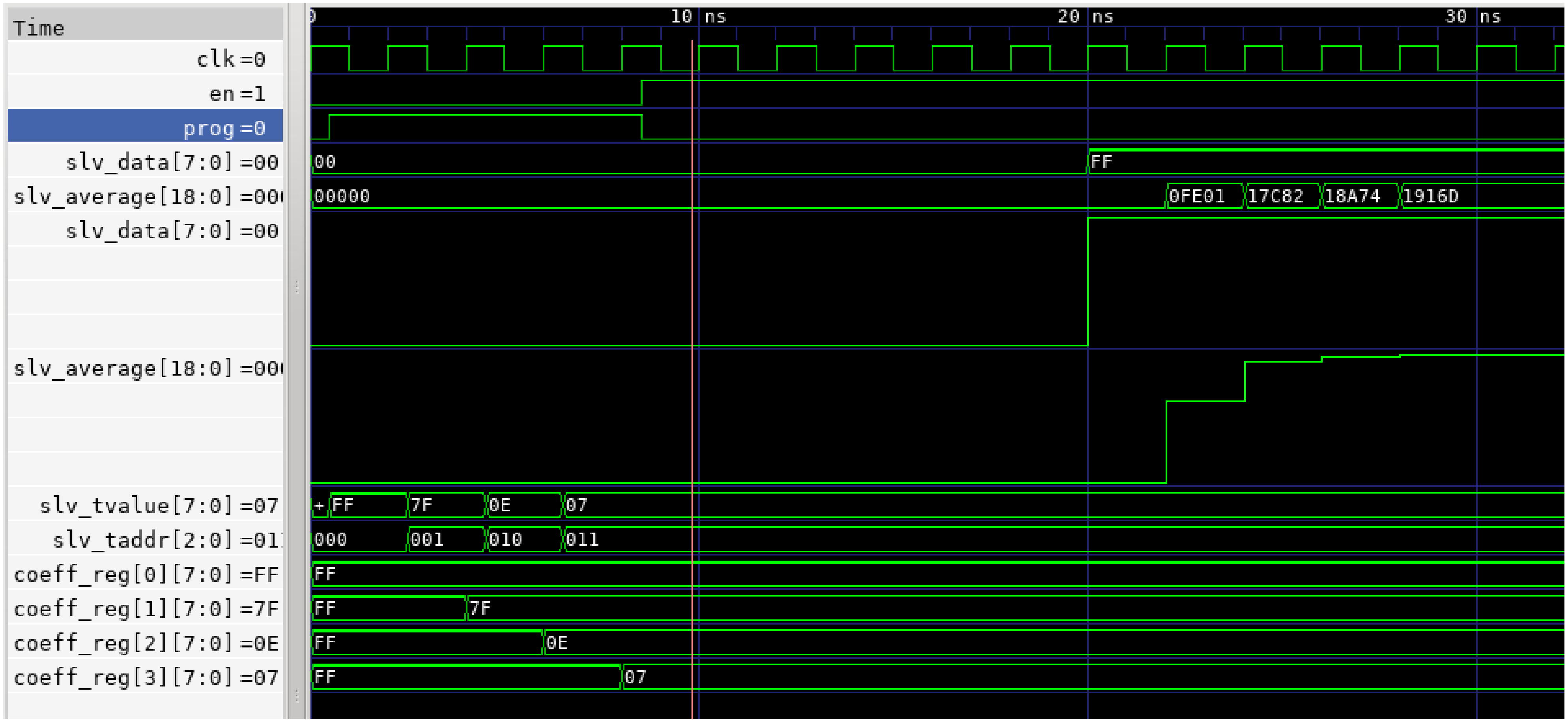
```
---------------------------------------------
  -- test bench definition.
  tb_CLK :process
  begin
      CLK <= not CLK;
      wait for 1 ns;
  end process;

  tb_en : process
  begin
      en <= '0';
      wait for 0.5 ns;
      prog <= '1';

      slv_Taddr  <= "000";
      slv_Tvalue <= x"FF";
      wait for 2 ns;

      slv_Taddr  <= "001";
      slv_Tvalue <= x"7F";
      wait for 2 ns;

      slv_Taddr  <= "010";
      slv_Tvalue <= x"0E";
      wait for 2 ns;

      slv_Taddr  <= "011";
      slv_Tvalue <= x"07";
      wait for 2 ns;

      prog <= '0';
      en <= '1';

      wait for 2000 ns;
  end process;

  tb_response : process
  begin
      wait for 10 ns;

      -- step response
      slv_data <= (others => '0');
      wait for 10 ns; -- wait 5 clock cycles
      slv_data <= (others => '1');
      wait for 50 ns;
      slv_data <= (others => '0');

      wait for 100 ns;
```

wait a long time to run this process „only once“

Write coefficients during runtime…



Write coefficients during runtime…



NEW

From signal to physics

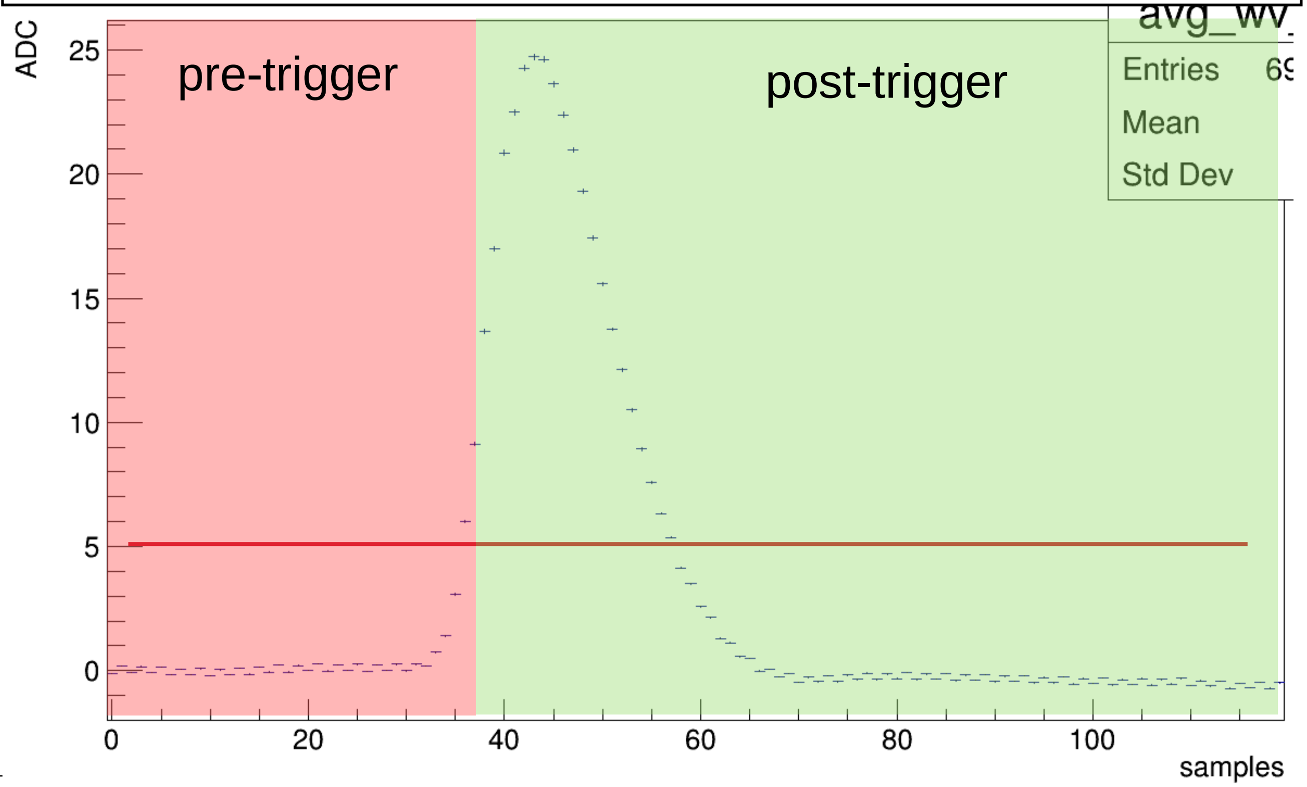
III. Physikalisches Institut B

RWTH AACHEN UNIVERSITY

Storing waveform data

- If we want to store data from our analog stream
  (or at least from a stream, which we are simulating)
- We want to select the interesting data (=> trigger)
- We want to have some pre trigger data

Average waveform, baseline corrected from OSIRIS iPMTs



III. Physikalisches Institut B

RWTH AACHEN UNIVERSITY

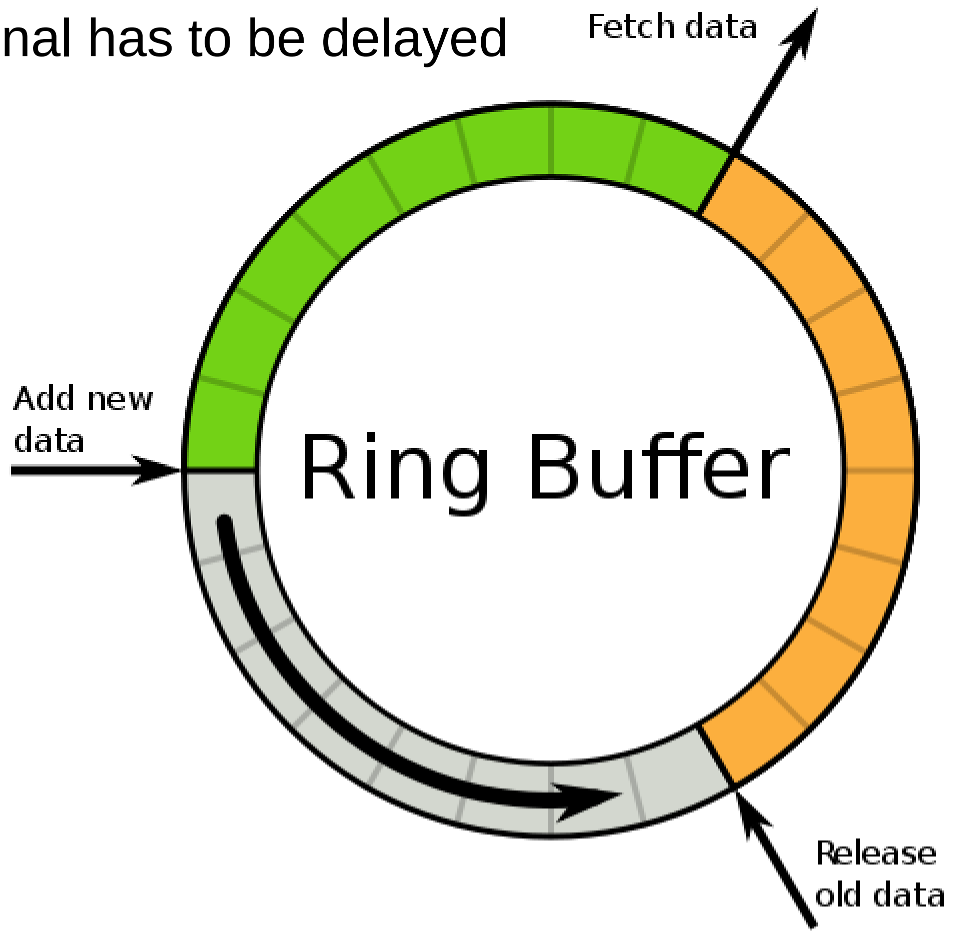
Solution

Ring Buffer

Always store data and overwrite old data

If trigger is raised, stop writing

To save pre-trigger data the trigger signal has to be delayed



How to implement this?

Step by Step

- Think about, what do we need?
  - Data input
    - Clock
    - Write Enable
    - Data 8 bit wide
  - Data output
    - Clock (different from data input)
    - Read Enable
    - Data 8 bit wide

Internal:

- Write pointer
- Read pointer
- Storage array

ToDo:

A. Create a RingBuffer, which can
   - store 8x 8bit values
   - Two different clocks (read and write) each with its own enable

B. Change the RingBuffer that it provides these features:
   - Readout only, if write disabled
   - Output in the correct order (read pointer has to start from write pointer)
     Hint: Between read and write are some clock cycles of each clock.

Keep in mind:
It is not possible to set variables in different processes.

C. Start with a simple trigger module
   - If reset, run = 1
   - If data is for one clockcycle above threshold run = 0 until reset

Thank you!