## Zasnova naključnega bita

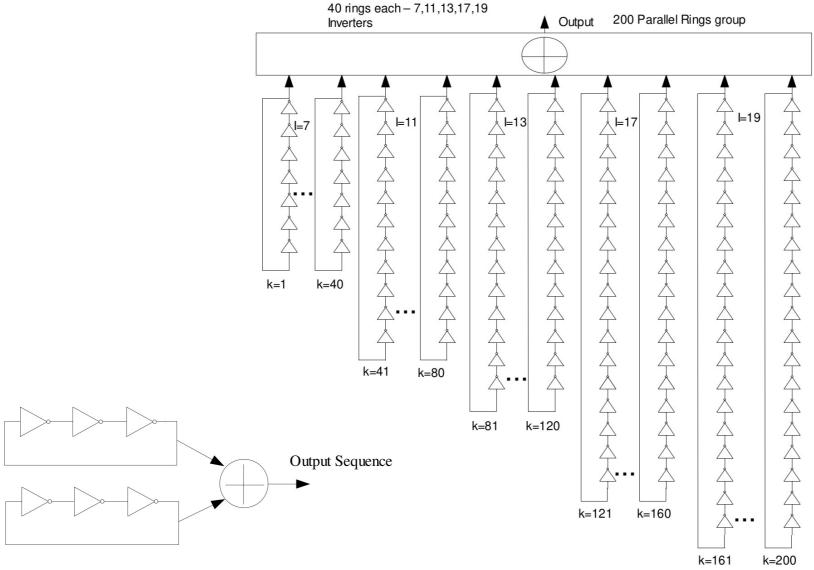
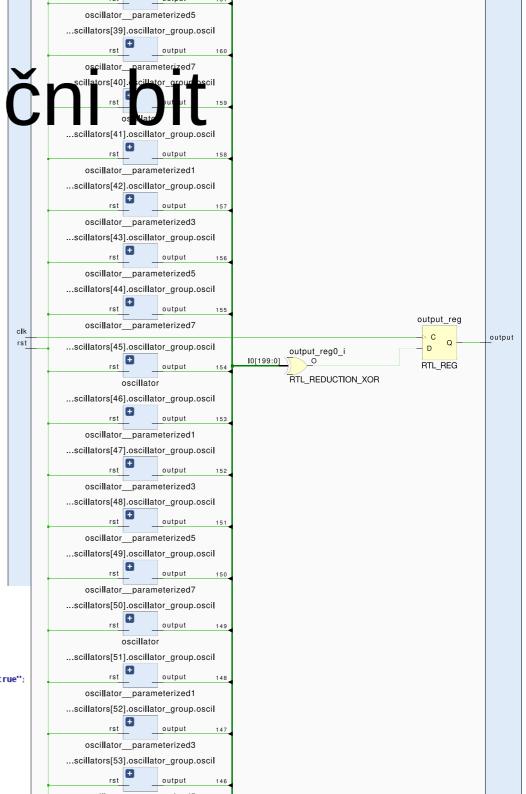


Figure 14: Two rings XOR'ed

Figure 15: Multiple ring design - ring 200

```
begin
```

```
generate oscillators: for i in 0 to 199 generate
                                                          Naključn
       oscillator group: if i mod 5 = 0 generate
           oscil : oscillator
               generic map (
                   length => 5
               port map(
                   output => oscillators(i)
       elsif i mod 5 = 1 generate
           oscil : oscillator
               generic map (
                   length => 7
               port map(
                   rst => rst,
                   output => oscillators(i)
       elsif i mod 5 = 2 generate
           oscil : oscillator
               generic map (
                   length => 11
               port map(
                   rst => rst.
                   output => oscillators(i)
       elsif i mod 5 = 3 generate
           oscil : oscillator
               generic map (
                   length => 13
               port map(
                   rst => rst,
                   output => oscillators(i)
       else generate
           oscil : oscillator
               generic map (
                   length => 17
               port map(
                   output => oscillators(i)
       end generate oscillator group ;
   end generate generate oscillators;
                                       architecture Behavioral of oscillator is
   digitalize : process(clk)
                                           signal chain: STD_LOGIC_VECTOR(length-1 downto 0);
       if rising edge(clk) then
                                           attribute DONT TOUCH : string;
           output <= xor oscillators;
                                           attribute ALLOW COMBINATORIAL LOOPS : string;
       end if:
   end process;
                                           attribute DONT TOUCH of chain : signal is "true";
                                           attribute DONT TOUCH of output : signal is "true";
end Behavioral;
                                           attribute ALLOW COMBINATORIAL LOOPS of output : signal is "true";
                                           generate_chain: for i in 1 to length-1 generate
                                               chain(i) <= not chain(i-1);</pre>
                                           end generate:
                                           chain(0) <= not chain(length-1) or rst;</pre>
                                           output <= chain(0);
                                       end Behavioral;
```



## Top – vse komponente

VGA B[3:0]

VGA G[3:0]

VGA R[3:0]

VGA VS

CA[6:0]

uart

UART\_RXD\_OUT

→ VGA B[3:0]

VGA\_G[3:0]

─ VGA\_R[3:0]

VGA HS

VGA VS

LED[5:0]

AN[7:0]

-CA[6:0]

UART RXD OUT

```
begin
    reset <= not rst:
    LED <= SW:
    rand : random bits
        generic map (
                                                                                                                                                               vga inst
            width => 12
                                                                                                                                                         B[3:0]
        port map (
                                                                                                                                            7.4
                                                                                                                                                         G[3:0]
           clk => clk,
                                                                                                                                                         R[3:0]
           rst => reset.
                                                                                                                                                            clk
            output => rand bits
                                                                                                                                                            vgaController
    display_7seg : display
                                                                                                             num handler
        port map (
                                                                                                                                                            display_7seg
                   => clk,
                                                                                                          BTNC
            rst
                   => reset.
                                      BTNC
                                                                                                        SW[5:0]
                   => AN,
                                    SW[5:0]
                                                                                                                                                   number[31:0]
            CA
                   => CA,
                                                                                                        byte[7:0]
                                                                                                                        number[31:0]
                                                                                                                                                            rst
            number => number
                                                                                                                                                               display
                                                                                                             rst
                                                                                                                                                             uart comm
    vga inst : vgaController
                                                                                                            number handler
        port map (
                                                                                                                                                      UART CTS
            clk => clk,
                                  UART CTS
            rst => reset.
                                                                            rand
            VGA HS => VGA HS.
                                                                                                                                                      data[7:0]
                                                      reset i
            VGA VS => VGA VS,
                                                                                  output[11:0]
                                                                                                                                                            rst
            VGA R => VGA R,
                                                                       rst
            VGA G \Rightarrow VGA G
                                                      RTL INV
            VGA B => VGA B,
                                                                        random_bits
                   => rand bits(11 downto 8),
                   => rand bits( 7 downto 4),
                   => rand bits( 3 downto 0)
        );
    uart comm : uart
        port map (
                         => clk.
            clk
                         => reset,
            UART RXD OUT => UART RXD OUT,
            UART CTS
                       => UART CTS,
            data
                         => rand bits(11 downto 4)
        );
    num handler : number handler
        port map (
                   => clk.
                  => reset.
            BTNC => BTNC.
                   => SW.
            byte => rand bits(7 downto 0),
            number => number
```

end Behavioral:

);

## Združevalnik števil 8-bit → 32-bit

```
-- omeji velikost števila
enable word <= std logic vector((to unsigned(1, 32) sll conv integer(SW)) - 1);</pre>
with conv integer(index) select
     enable byte <= enable word(31 downto 24) when 3,
                         enable word(23 downto 16) when 2,
                         enable word(15 downto 8) when 1,
                        enable word( 7 downto 0) when 0;
-- prikaži 4 byte na 7-segmentnem prikazovalniku
synchronous : process(clk)
begin
     if rising edge(clk) then
          if rst='l' then
                output <= (others => '0');
          elsif (BTNC = '1' and BTNC prev = '0') or (index /= "00") then
                case conv integer(index) is
                    when 3 => output(31 downto 24) <= byte and enable byte;
                     when 2 => output(23 downto 16) <= byte and enable byte;
                    when 1 => output(15 downto 8) <= byte and enable byte;
                     when 0 => output( 7 downto 0) <= byte and enable byte;
               index <= index + 1:
          end if:
                                                                                                                                          10[7:0] output0_i
                                                                                                                                              RTL AND
          BTNC prev <= BTNC;
     end if:
                                                                                                                                            output i 0
                                                                                                           enable byte i
                                                                                                                                                                     output i
                                                                                                                               V=X*FF000000*, S=2'b11 [0[31:0]
end process;
                                                                                       minusOp i
                                                                                                     S=2'b11 [0[7:0]
                                                                                                                                                              S=2'b11 [0[31:0]
                                                           12 L_i
                                                                                 l0[31:0] O[31:0]
                                                                                                    S=2'b10 [1[7:0]
                                                                                                                              V=X*00FF0000*, S=2*b10 [1[31:0]
                                                                                                                                                              S=2'b10 [1[31:0]
                                                                                                                                                 0[31:0]
                                                                                                                                                                          O[31:0]
                                                   v=x*000000001* I0[31:0]
                                                                                                     S=2'b01 [2[7:0]
                                                                                                                              V=X*0000FF00*, S=2'b01 [2[31:0]
                                                                                                                                                              S=2'b01 [2[31:0]
                               SW[5:0]
                                                                                        RTL_SUB
                                                                RTL_LSHIFT
                               byte[7:0]
                                                                                                           SI1:01 RTL_MUX
                                                                                                                                                RTL_MUX
                                                                                                                                                                    S[1:0] RTL_MUX
                                                               plusOp_i
                                                                                                                                                                                            output_reg[31:0]
                                                          [0[1:0]
                                                               RTL_ADD
                                                                                                                                                                                                          output[31:0]
                                                                                    index_reg[1:0]
                                                                                                                                                                    output i 1
                                                             index i
                                                                                                                                                               S=1'b1 [0[31:0]
                                                         S=1'b1 10
                                                                                                                                                                                       RTL_REG_SYNC
                                                                                                                                                                       RTL MUX
                                                                                                                                            10_output1_i
                                                              RTL_MUX
                                                                                     RTL_REG
                                                                                                               RTL NEQ
                                                            BTNC_prev_reg
                                                                                                               0
                                                                                    <u>I1</u> = 0
                                                                                                              RTL AND
                                                             RTL REG
                                                                                                                  number handle
```