Zasnova naključnega bita

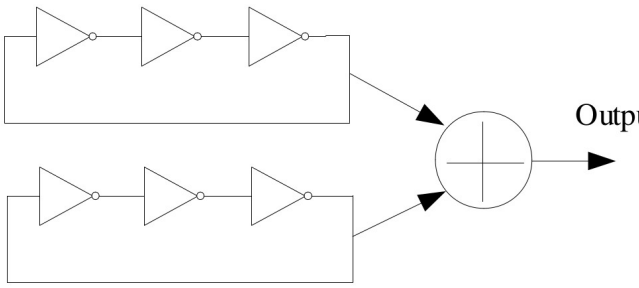


Figure 14: Two rings XOR'ed

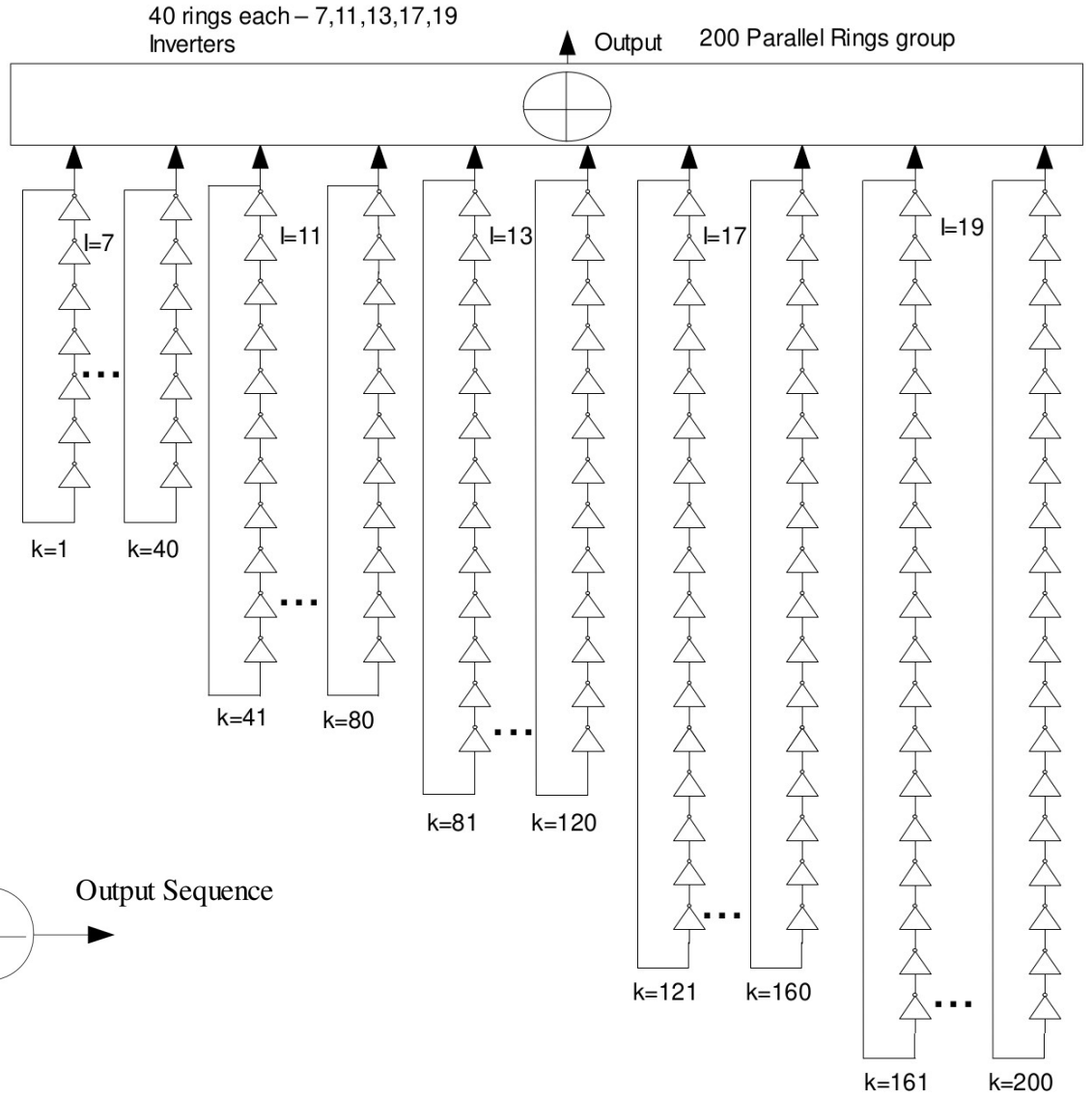
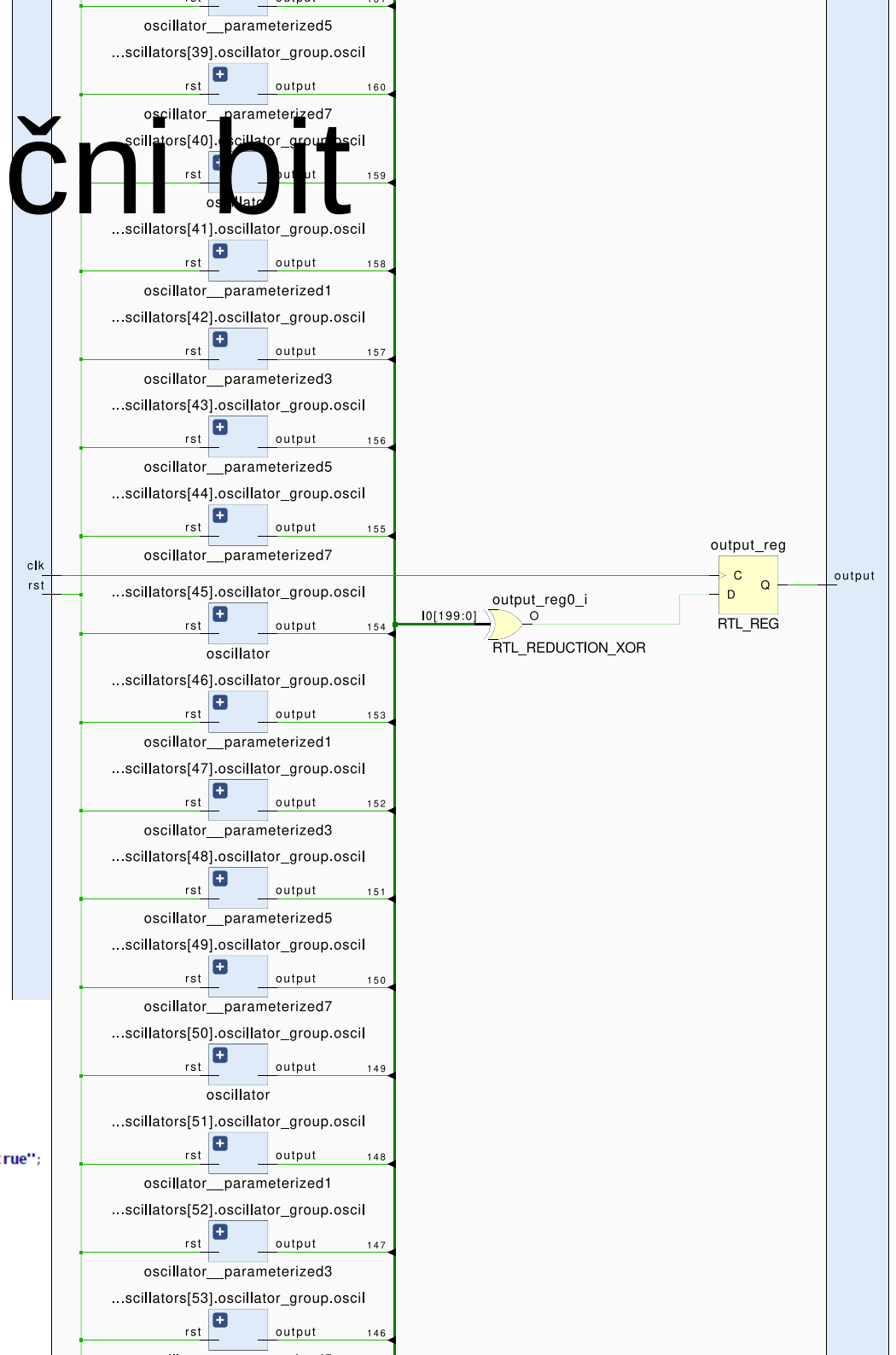


Figure 15 : Multiple ring design - ring 200

Naključni bit

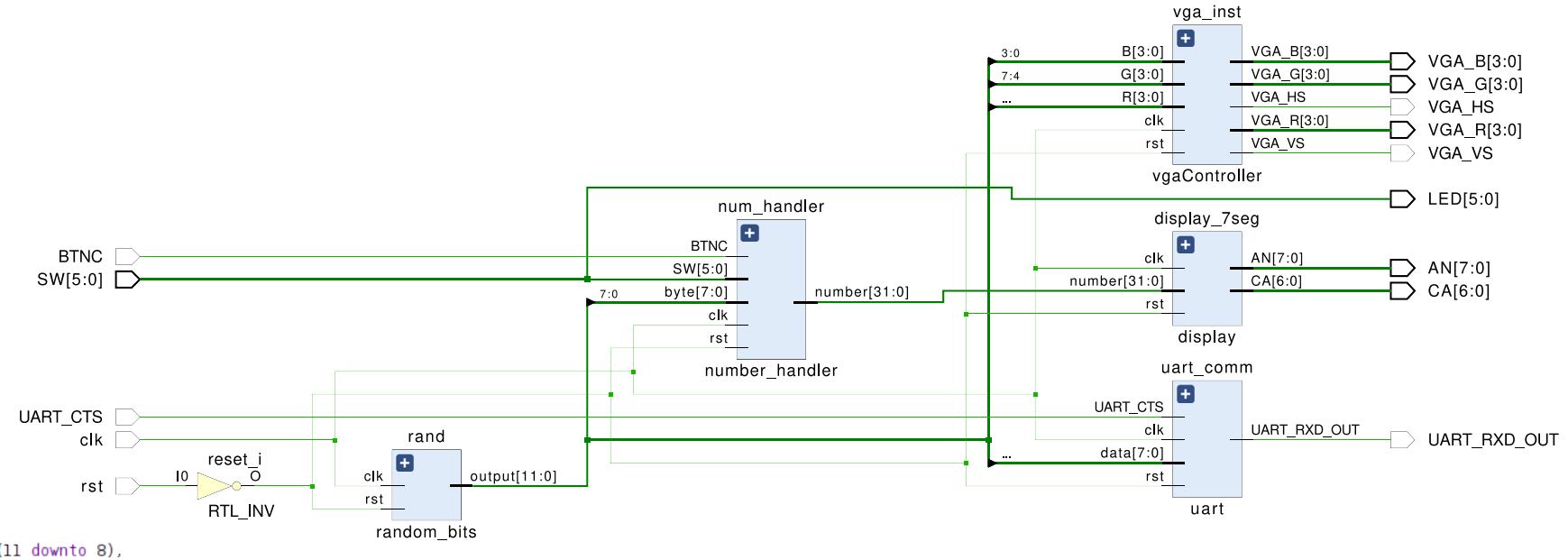
```
begin

    generate_oscillators: for i in 0 to 199 generate

        oscillator_group: if i mod 5 = 0 generate
            oscil : oscillator
                generic map (
                    length => 5
                )
                port map(
                    rst => rst,
                    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(
                    rst => rst,
                    output => oscillators(i)
                );
        end generate oscillator_group ;

    end generate generate_oscillators;

    digitalize : process(clk)
    begin
        if rising_edge(clk) then
            output <= xor oscillators;
        end if;
    end process;

end Behavioral;
```

```
architecture Behavioral of oscillator is
    signal chain: STD_LOGIC_VECTOR(length-1 downto 0);

    attribute DONT_TOUCH : string;
    attribute ALLOW_COMBINATORIAL_LOOPS : string;

    attribute DONT_TOUCH of chain  : signal is "true";
    attribute DONT_TOUCH of output : signal is "true";
    attribute ALLOW_COMBINATORIAL_LOOPS of output : signal is "true";
begin

    generate_chain: for i in 1 to length-1 generate
        chain(i) <= not chain(i-1);
    end generate;
    chain(0) <= not chain(length-1) or rst;

    output <= chain(0);

end Behavioral;
```



Top – vse komponente

```
begin

    reset <= not rst;

    LED <= SW;

    rand : random_bits
        generic map (
            width => 12
        )
        port map (
            clk    => clk,
            rst    => reset,
            output => rand_bits
        );

    display_7seg : display
        port map (
            clk    => clk,
            rst    => reset,
            AN     => AN,
            CA     => CA,
            number => number
        );

    vga_inst : vgaController
        port map (
            clk    => clk,
            rst    => reset,
            VGA_HS => VGA_HS,
            VGA_VS => VGA_VS,
            VGA_R  => VGA_R,
            VGA_G  => VGA_G,
            VGA_B  => VGA_B,
            R      => rand_bits(11 downto 8),
            G      => rand_bits( 7 downto 4),
            B      => rand_bits( 3 downto 0)
        );

    uart_comm : uart
        port map (
            clk          => clk,
            rst          => reset,
            UART_RXD_OUT => UART_RXD_OUT,
            UART_CTS     => UART_CTS,
            data         => rand_bits(11 downto 4)
        );

    num_handler : number_handler
        port map (
            clk    => clk,
            rst    => reset,
            BTNC   => BTNC,
            SW     => 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);

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='1' 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;
            end case;
            index <= index + 1;
        end if;

        BTNC_prev <= BTNC;
    end if;
end process;
```