VHDL Demo 3

Processes, sequential logic, Vivado simulator (xsim)

NOTE! This demo is supposed to be presented after lecture 3 (synchronous/sequential logic), before moving to lab 3.

V 1.4, 27.9.2021
Jarno Tuominen

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Turning your design into processes

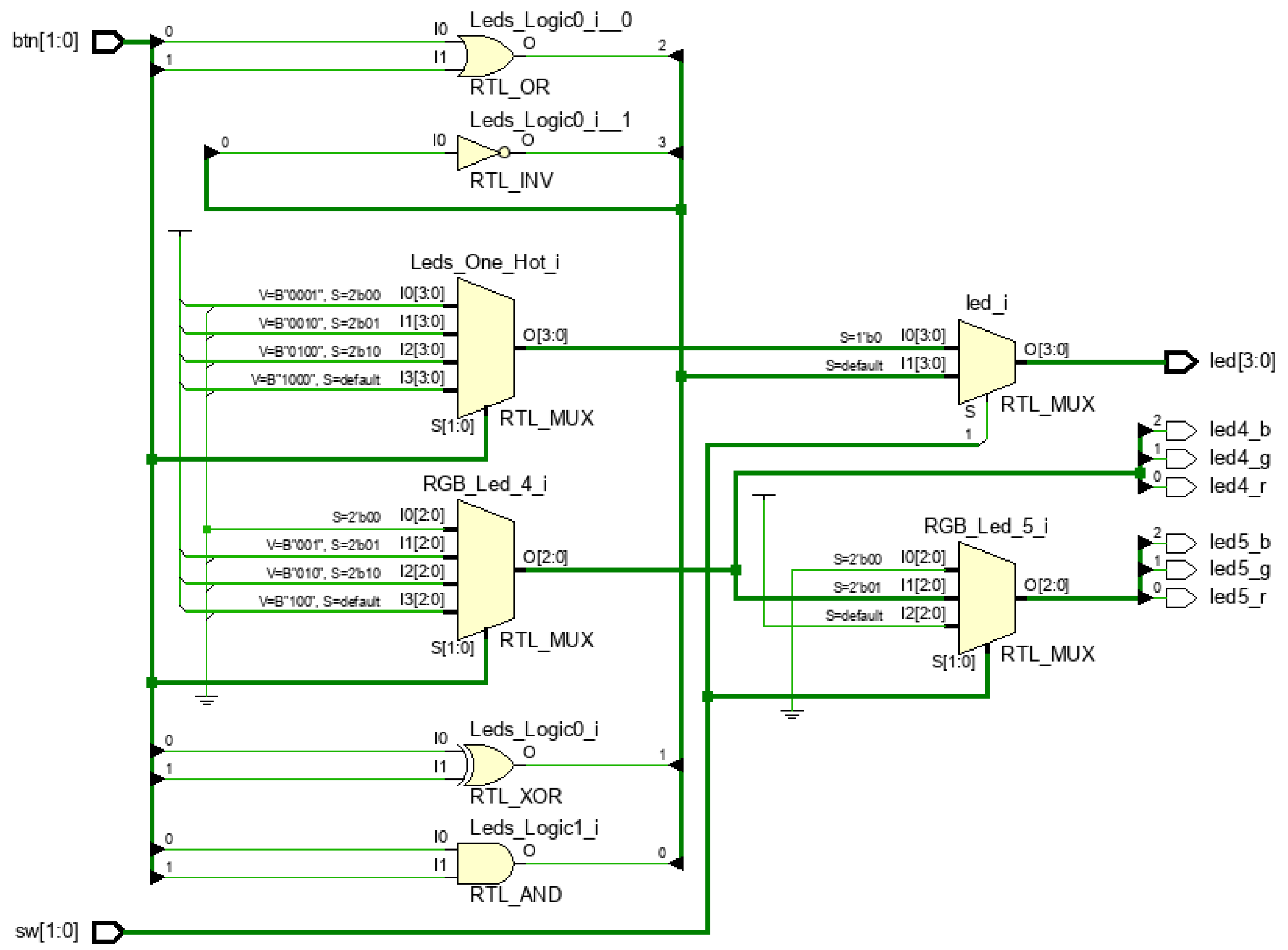
- Recalling the “green led” controller in LAB2: we used
  - concurrent signal assignments
  - conditional signal assignments
  - selected signal assignments
- The concurrent signal assignment is allowed within the process, the other 2 aren’t
- Note: term “process” has nothing to do with processes in computer science / operating systems

```
-- Green leds stuff here

 -- Control green leds, 2 to 4 decoder
with btn(1 downto 0) select
   Leds_One_Hot  <= "0001" when "00",
                    "0010" when "01",
                    "0100" when "10",
                    "1000" when others;

-- Control green leds, some logic stuff
Leds_Logic(0) <= btn(0) and btn(1);
Leds_Logic(1) <= btn(0) xor btn(1);
Leds_Logic(2) <= btn(0) or btn(1);
Leds_Logic(3) <= btn(0) nand btn(1);

-- output mux for green leds
led <= Leds_One_Hot when sw(1)='0' else Leds_Logic;
```

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

LAB 2 as schematics

- Green leds controller implemented as mux (2-to-4 decoder part), individual gates (the logic part), and a 2 to 1 mux
- RGB4 and RGB5 controlled by multiplexers, no registers
- Note: Path from btn to RGB-led5 contains 2 multiplexers



Turning your... (2)

- Now, we have used
  - concurrent signal assignment for the “logic functions” part
  - Sequential if-statement for the output mux
  - Sequential case-statement for the 2-to4 decoder
- The process is sensitive to all the input signals
- Still, after synthesis we have plain combinatorial logic design (no clocks involved)

```vhdl
green_led_ctrl_p: process( btn(1), btn(0), sw(1) )
begin

  if sw(1) = '0' then
    case btn(1 downto 0) is
      when "00" =>
        led <=  "0001";
      when "01" =>
        led <=  "0010";
      when "10" =>
        led <=  "0100";
      when others =>
        led <=  "0100";
    end case;
  else
    led(0) <= btn(0) and btn(1);
    led(1) <= btn(0) xor btn(1);
    led(2) <= btn(0) or btn(1);
    led(3) <= btn(0) nand btn(1);
  end if; -- sw(1)

end process green_led_ctrl_p;
```

Turning your… (3)

- After synthesis, we have (or at least should have) exactly similar design as earlier – you can check that by looking at the netlist
- The number of used resources is also the same (make note of them right now)
- Next, let’s make a clocked process
  - let’s modify the RGB led 4 controller (2-to-4 decoder)

```
+-------------------------+------+-------+-----------+-------+
|        Site Type        | Used | Fixed | Available | Util% |
+-------------------------+------+-------+-----------+-------+
| Slice LUTs*             |    5 |     0 |     53200 | <0.01 |
|   LUT as Logic          |    5 |     0 |     53200 | <0.01 |
|   LUT as Memory         |    0 |     0 |     17400 |  0.00 |
| Slice Registers         |    0 |     0 |    106400 |  0.00 |
|   Register as Flip Flop |    0 |     0 |    106400 |  0.00 |
|   Register as Latch     |    0 |     0 |    106400 |  0.00 |
| F7 Muxes                |    0 |     0 |     26600 |  0.00 |
| F8 Muxes                |    0 |     0 |     13300 |  0.00 |
+-------------------------+------+-------+-----------+-------+
```

```
-- Control of RGB LED 4
with btn(1 downto 0) select
   RGB_Led_4 <=   "000" when "00", --off
                   "001" when "01", --red
                   "010" when "10", --green
                   "100" when others; --"11" -> blue
```

Turning your… (4)

- The process is sensitive to clock and reset
  - n_Reset : just a naming convention to highlight that the reset is active low
- Buttons are sampled every time there is a **rising edge** in the clock signal
  - Alternative (VHDL93) form:

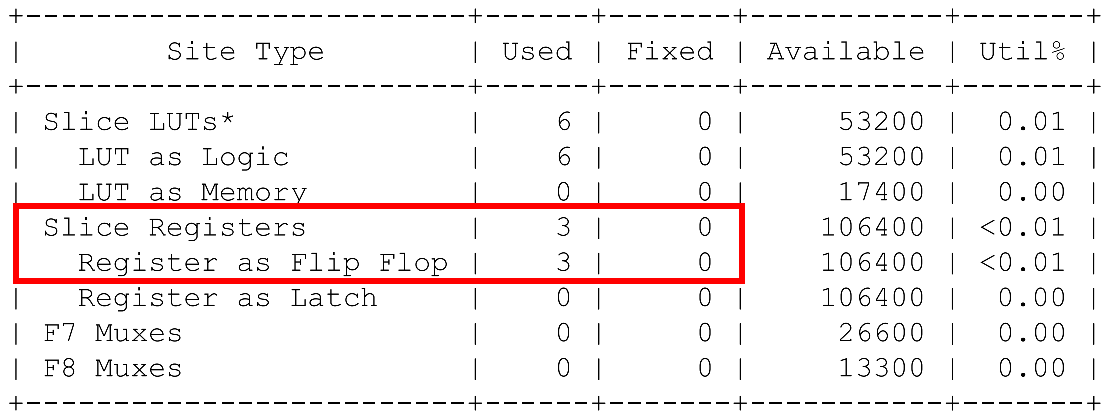
```
if rising_edge(sysclk) then
```

- The rgb4 led is “updated” after a clock edge – so this design became **synchronous** (to sysclk)

```
rgb4_ctrl_p: process(sysclk, n_Reset)
begin
        if n_Reset = '0' then
                RGB_Led_4 <= (others => '0');
        elsif sysclk'event and sysclk = '1' then
          case btn(1 downto 0) is
                when "00" =>
                        RGB_Led_4 <=  "000"; --off
                when "01" =>
                        RGB_Led_4 <=  "001"; --red
                when "10" =>
                        RGB_Led_4 <=  "010"; --green
                when others =>
                        RGB_Led_4 <=  "100"; --blue
          end case;
        end if; --clk/rst
end process rgb4_ctrl_p;
```

Turning your... (5)

- Now, look at the resource utilization report
- Same number of LUT's, but 3 more registers, as expected
- Do you notice any differences in functionality?
- There is! After pressing the button, you might need to wait up to 8 nanoseconds until the led changes color! ☺
- -> Time to simulate!

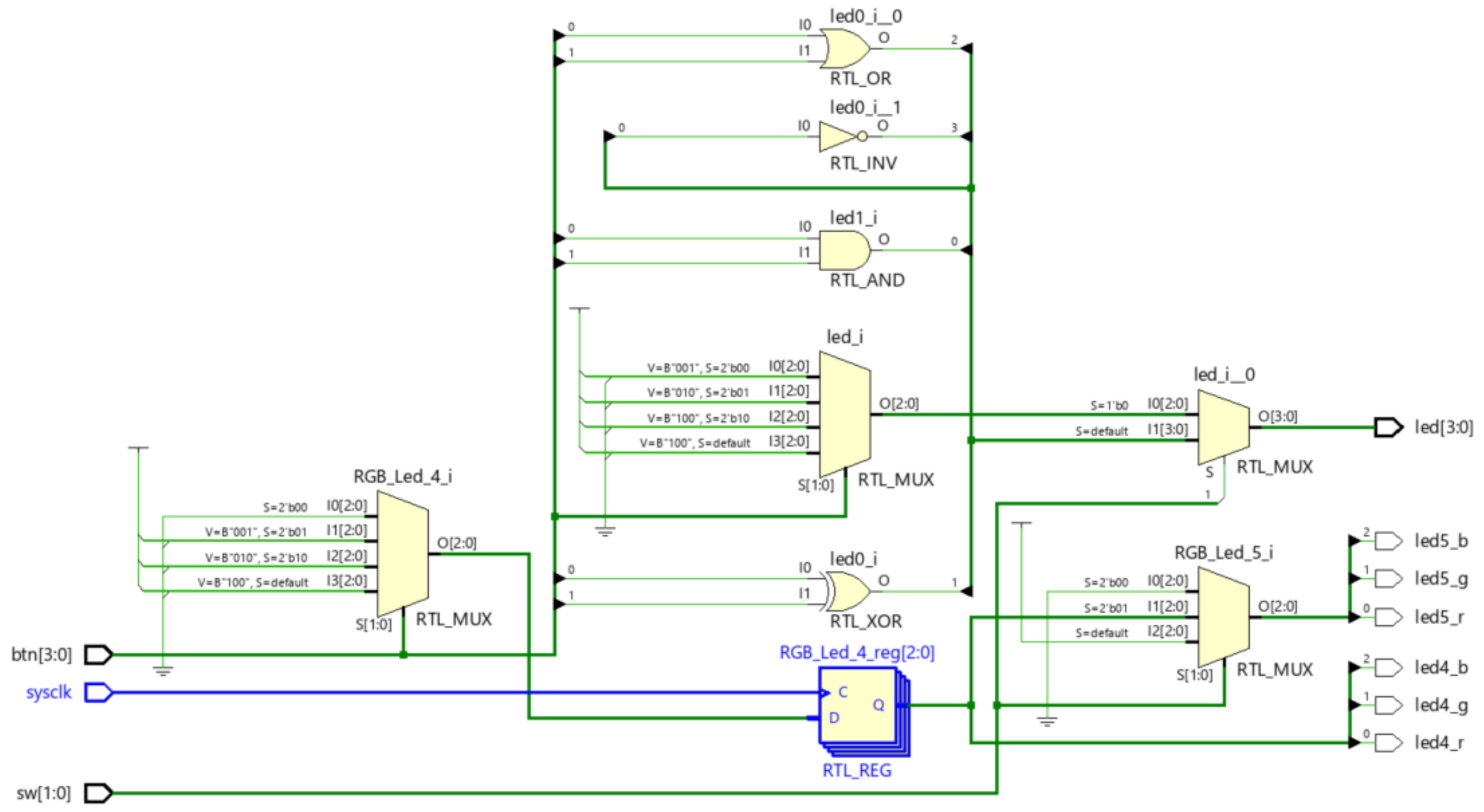


```
+--------------------------+------+-------+-----------+-------+
|         Site Type        | Used | Fixed | Available | Util% |
+--------------------------+------+-------+-----------+-------+
| Slice LUTs*              |    6 |     0 |     53200 |  0.01 |
|   LUT as Logic           |    6 |     0 |     53200 |  0.01 |
|   LUT as Memory          |    0 |     0 |     17400 |  0.00 |
| Slice Registers          |    3 |     0 |    106400 | <0.01 |
|   Register as Flip Flop  |    3 |     0 |    106400 | <0.01 |
|   Register as Latch      |    0 |     0 |    106400 |  0.00 |
| F7 Muxes                 |    0 |     0 |     26600 |  0.00 |
| F8 Muxes                 |    0 |     0 |     13300 |  0.00 |
+--------------------------+------+-------+-----------+-------+
```

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

RTL schematics after modifications

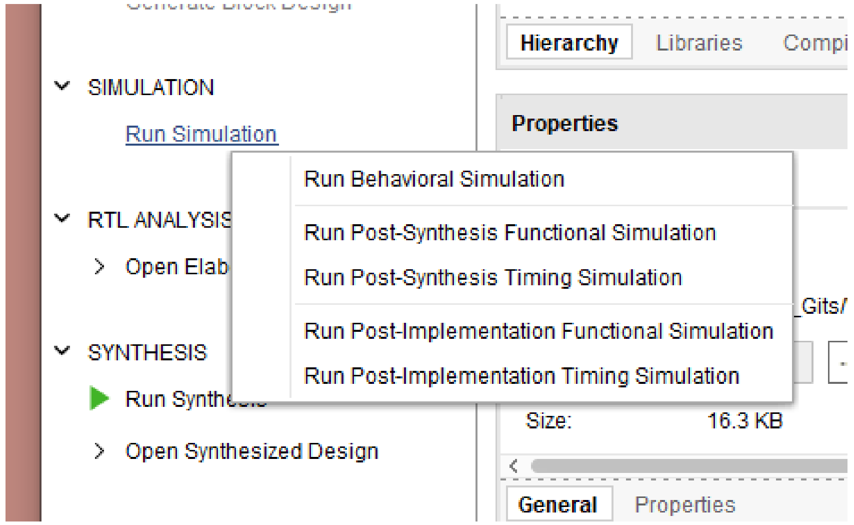
- Identical to lab2, except we have now the (3-bit) register holding RGB4 value
- Timing-wise this runs also faster: the path from btn to RGB5 is cut by a register (but of course it adds 1 clock cycle latency)

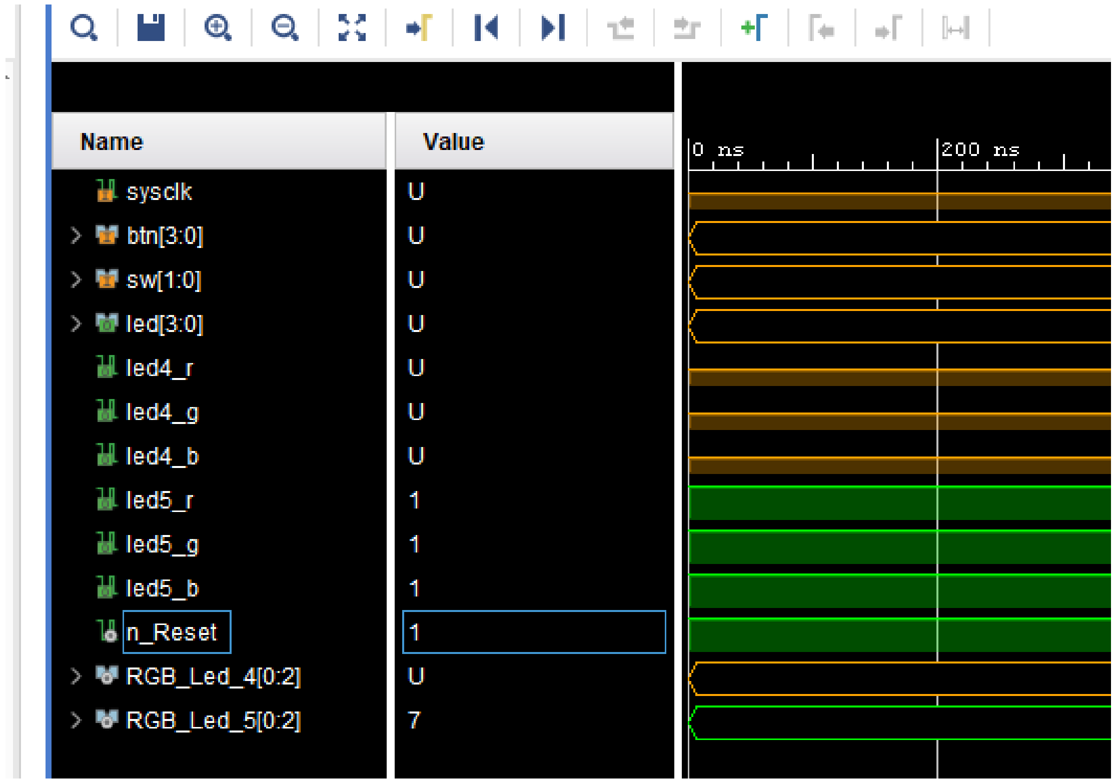


TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Using Vivado simulator (xsim)

- 1. Launch simulator
  - Disable your anti-virus software first! (Simulator will create a new executable to speed-up the simulation, which is guarantined, or at least the execution is delayed)
  - Run behavioral simulation
- 2. By default, it will run for 1 us
  - No input signals are driven, you really don't see anything
  - Restart your simulation now (as we have 1us "worth nothing", since there is no clock or any input stimulus

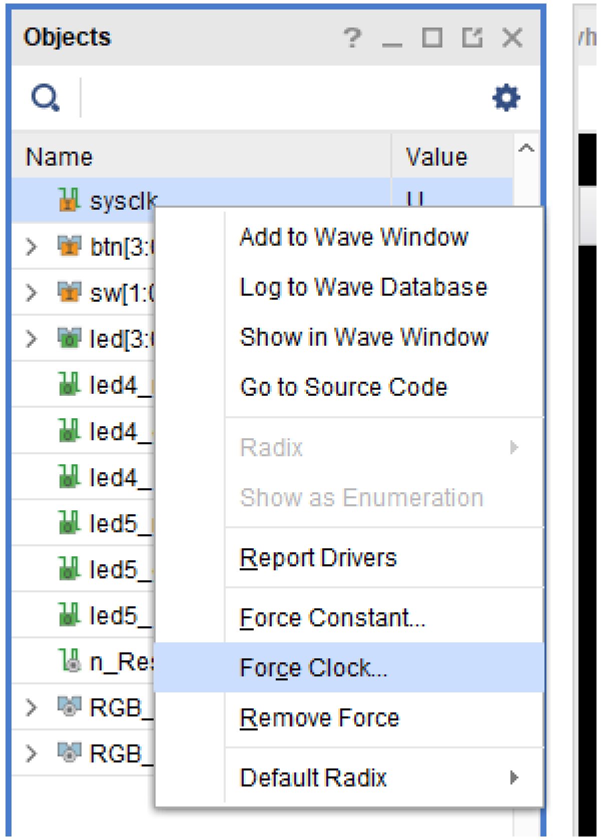

SIMULATION
Run Simulation
RTL ANALYSIS
Open Elab
SYNTHESIS
Run Synthesis
Open Synthesized Design
Hierarchy
Libraries
Compi
Properties
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation
Run Post-Implementation Functional Simulation
Run Post-Implementation Timing Simulation
Size:
16.3 KB
General
Properties

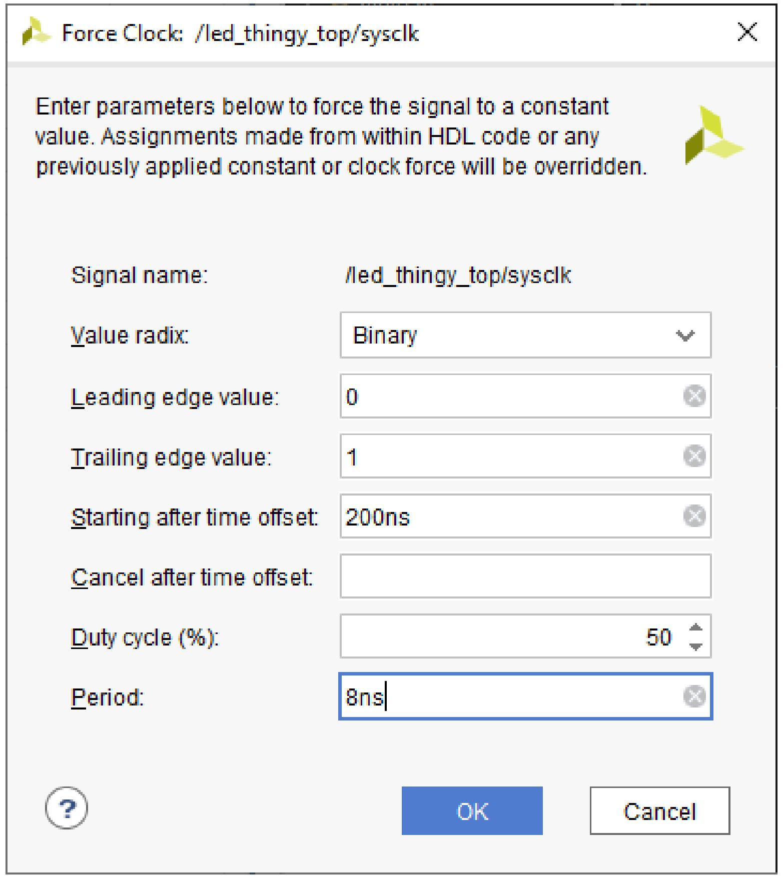


Name	Value
sysclk	U
btn[3:0]	U
sw[1:0]	U
led[3:0]	U
led4_r	U
led4_g	U
led4_b	U
led5_r	1
led5_g	1
led5_b	1
n_Reset	1
RGB_Led_4[0:2]	U
RGB_Led_5[0:2]	7

xsim (2)

- Restart your simulation (ctrl+shift+F5)
- In objects-window
- Force clock signal to sys clk
  - after 200 ns, 125MHz clock starts ticking
- Force zero constant to btn and sw
- Run simulation for 500 ns
- Force non-zero constant to btn (like 0x2 = "0010")
- Run simulation for 500 ns




Force Clock: /led_thingy_top/sysclk

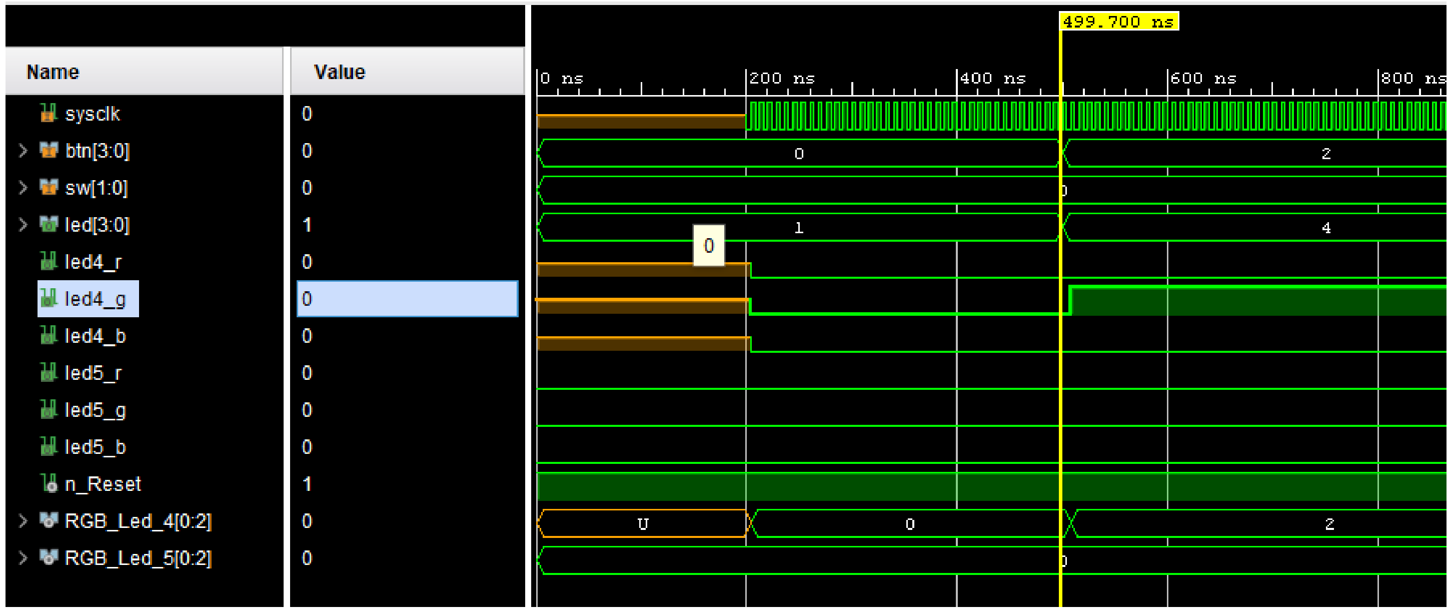
Enter parameters below to force the signal to a constant value. Assignments made from within HDL code or any previously applied constant or clock force will be overridden.

Signal name:	/led_thingy_top/sysclk
Value radix:	Binary
Leading edge value:	0
Trailing edge value:	1
Starting after time offset:	200ns
Cancel after time offset:	
Duty cycle (%):	50
Period:	8ns

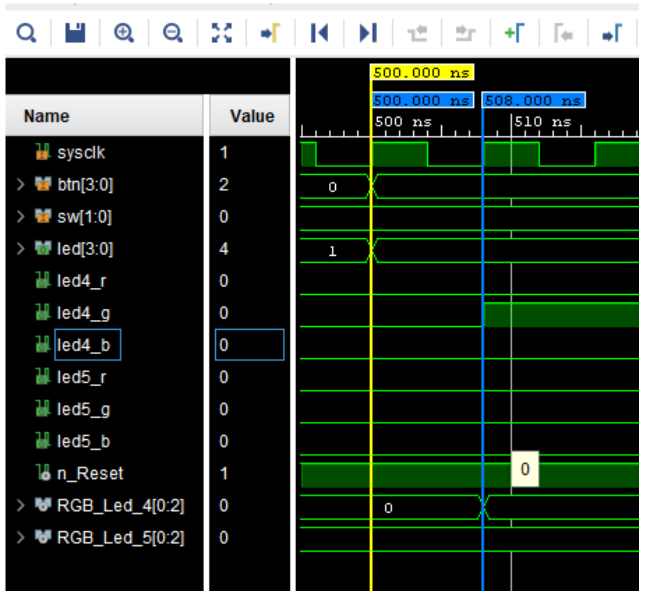
TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

xsim (3)

- See how RGB led 4 is updated only after clock starts ticking
- Note 8 ns delay from button press to RGB4 led state change. (I told you!)
- Note: Coverage of this simulation is really poor and this is really clumsy to use!
- Later on, we will create a **testbench** for our design to automate things



Name	Value
sysclk	0
btn[3:0]	0
sw[1:0]	0
led[3:0]	1
led4_r	0
led4_g	0
led4_b	0
led5_r	0
led5_g	0
led5_b	0
n_Reset	1
RGB_Led_4[0:2]	0
RGB_Led_5[0:2]	0



Name	Value
sysclk	1
btn[3:0]	2
sw[1:0]	0
led[3:0]	4
led4_r	0
led4_g	0
led4_b	0
led5_r	0
led5_g	0
led5_b	0
n_Reset	1
RGB_Led_4[0:2]	0
RGB_Led_5[0:2]	0

TURKU AMK
TURKU UNIVERSITY OF APPLIED SCIENCES

Your turn!

- Perform this demo on your own machine
- Then, move on to Lab 3