

Projeto com Circuitos Reconfiguráveis

Projeto lógico sequencial

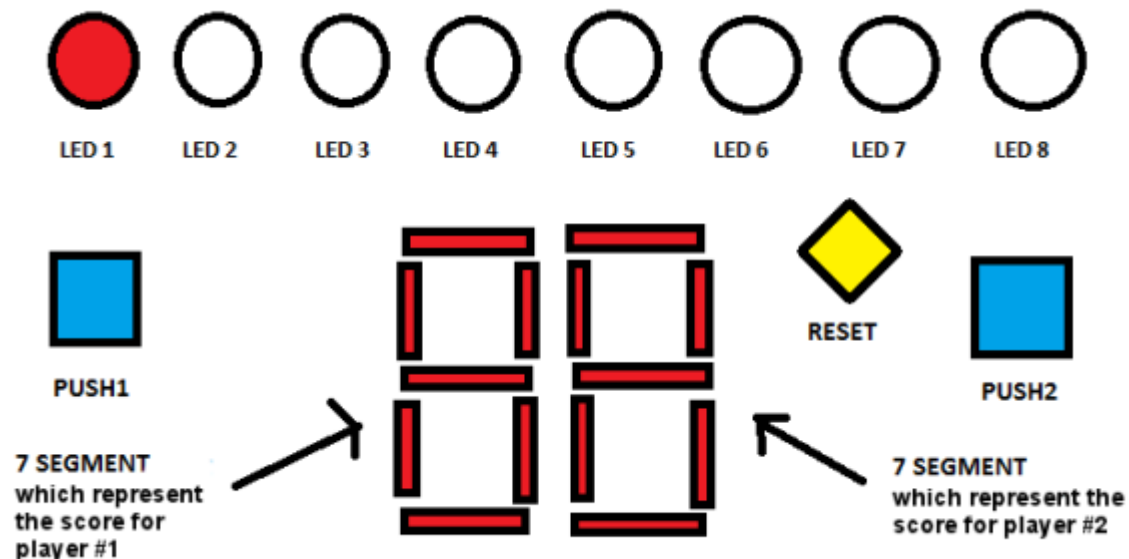
Aula demonstrativa ping-pong leds

Prof. Daniel M. Muñoz Arboleda

FGA - UnB

Projeto ping-pong leds

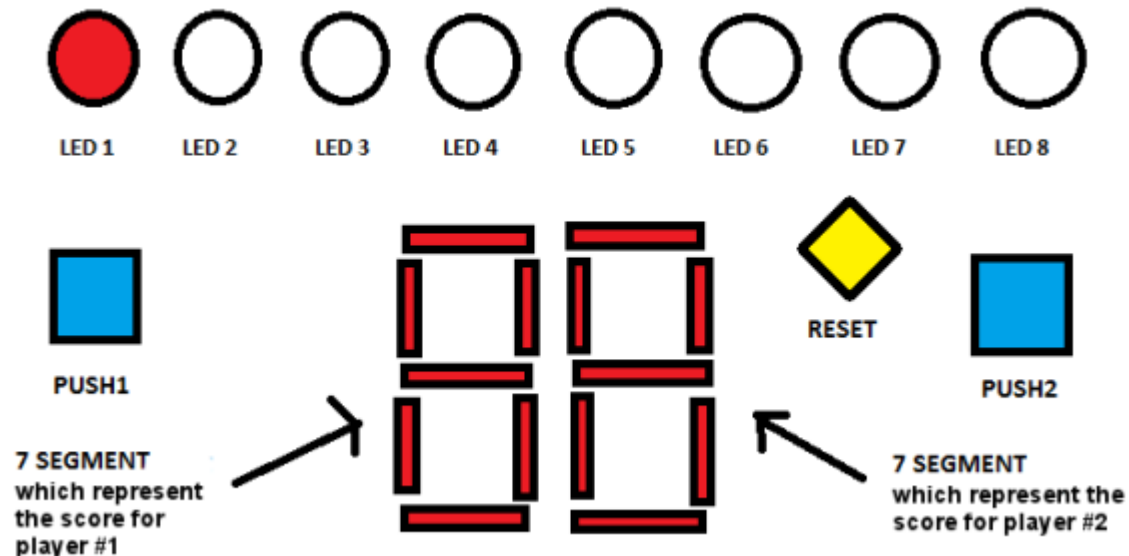
Adaptação para Basys3: a bolinha se movimenta a 100 ms; usar os 16 leds; o player1 deve rebater usando o sw0 e o player 2 usando o sw15; quando um player anota ponto a bolinha espera a ser rebatida; o jogador que chegar primeiro a 9 pontos ganha o jogo. O placar do jogo é apresentado nos displays 7 segmentos 1 e 4 (player1 e player2). Os displays 2 e 3 devem ficar em zero ou desligados.



Projeto ping-pong leds. Especificações

Módulos combinacionais: mux 4 para 1, decodificador binário para 7 segmentos.

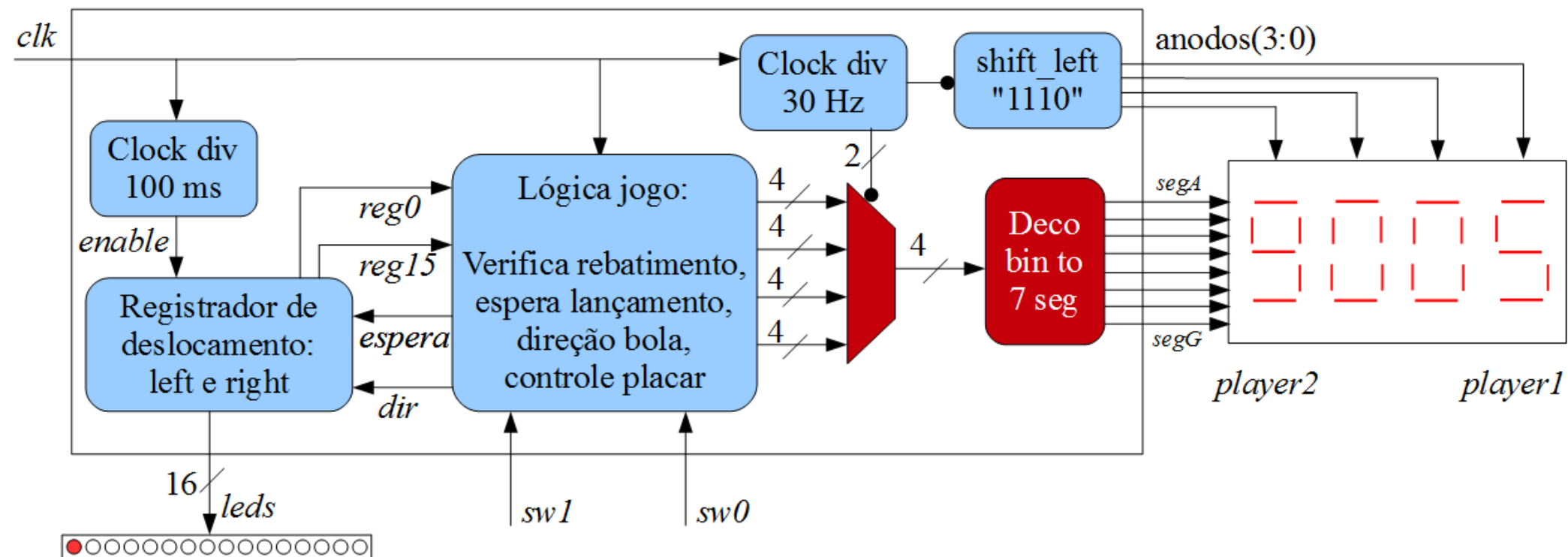
Módulos sequenciais: contador de 100 ms, registradores de deslocamento à direita e esquerda, lógica de detecção do ponto e direção da bolinha, contador a 30Hz para multiplexação dos anodos e seleção do mux 4 para 1 dos displays de 7 segmentos.



E agora?



Projeto ping-pong leds. Diagrama de blocos



E agora ... VHDL !



Projeto ping-pong leds. Bibliotecas e entidade

```
21
22 library IEEE;
23 use IEEE.STD_LOGIC_1164.ALL;
24 use ieee.numeric_std.all;
25 use IEEE.STD_LOGIC_unsigned.ALL;
26 use IEEE.STD_LOGIC_arith.ALL;
27
28 entity ping_pong_leds is
29     Port ( reset : in STD_LOGIC;
30           clk : in STD_LOGIC;
31           sw : in STD_LOGIC_VECTOR (1 downto 0);
32           led : out STD_LOGIC_VECTOR (15 downto 0);
33           an : out STD_LOGIC_VECTOR (3 downto 0); -- anodos
34           seg : out STD_LOGIC_VECTOR (6 downto 0));
35 end ping_pong_leds;
36
```

NUNCA use numeric_std e
std_logic_arith ao mesmo tempo !

Dica1: procure usar mesmo nome das portas do arquivo de constrains Basys3_Master.xdc

Dica2: nunca use numeric_std e std_logic_arith ao mesmo tempo.

Dica3: use biblioteca aritmética numeric_std (padrão da IEEE).

Dica4: está ok se usar numeric_std e std_logic_unsigned juntas

Projeto ping-pong leds. Mux4to1 e decodificador bin7seg

```
-- criando mux 4 to 1
with sel_mux select
    out_mux <= pp1 when "00",
               disp2 when "01",
               disp3 when "10",
               pp2 when others;
```

```
1 architecture Behavioral of ping_pong_leds is
2     signal out_mux: std_logic_vector(3 downto 0) := "0000";
3 begin
4     -- decodificador bin to 7 seg
5     process(out_mux)
6     begin
7         case out_mux is
8             when "0000" => seg <= "1000000";
9             when "0001" => seg <= "1111001";
10            when "0010" => seg <= "0100100";
11            when "0011" => seg <= "0110000";
12            when "0100" => seg <= "0011001";
13            when "0101" => seg <= "0010010";
14            when "0110" => seg <= "0000010";
15            when "0111" => seg <= "1111000";
16            when "1000" => seg <= "0000000";
17            when "1001" => seg <= "0010000";
18            when "1010" => seg <= "0001000";
19            when "1011" => seg <= "0000011";
20            when "1100" => seg <= "1000110";
21            when "1101" => seg <= "0100001";
22            when "1110" => seg <= "0000110";
23            when "1111" => seg <= "0001110";
24            when others => seg <= "1111111";
25        end case;
26    end process;
27
```


Projeto ping-pong leds. Contador 30 Hz para multiplexar anodos

```
131
132 -- contador de 1/30 segundos para multiplexar anodos e selecionar mux 4 to 1
133 process(clk,reset)
134 variable anodo_var: bit_vector(3 downto 0) := "1110";
135 begin
136     if reset='1' then
137         anodo_cnt <= (others=>'0');
138         anodo_var := "1110";
139         sel_mux <= "00";
140     elsif rising_edge(clk) then
141         if anodo_cnt = "101000101100001010" then
142             anodo_var := anodo_var rol 1;
143             sel_mux <= sel_mux + '1';
144             anodo_cnt <= (others=>'0');
145         else
146             anodo_cnt <= anodo_cnt + '1';
147         end if;
148         an <= to_stdlogicvector(anodo_var);
149     end if;
150 end process;
```

Nota: ver slides e video aula sobre diferença entre sinal e variável

<https://www.youtube.com/watch?v=1DEjDqnhbxs&list=PLKIWpQ56tY7KeqdSf36lrdsVTm2TGvdFq&index=6&t=357s>

Projeto ping-pong leds. Contador 100 ms

```
55 -- contador de 100 ms e criacao do enable
56 process(clk,reset)
57 begin
58     if reset='1' then
59         count <= (others=>'0');
60     elsif rising_edge(clk) then
61         if count = "10011000100101101000000" then
62             enable <= '1';
63             count <= (others=>'0');
64         else
65             enable <= '0';
66             count <= count + '1';
67         end if;
68     end if;
69 end process;
70
```

Projeto ping-pong leds. Registradores de deslocamento

```
70
71 -- registradores de deslocamento
72 process(clk,reset)
73 begin
74     if reset='1' then
75         reg_ball <= "0000000000000001";
76     elsif rising_edge(clk) then
77         if espera_p1 = '1' then
78             reg_ball <= "0000000000000001";
79         elsif espera_p2 = '1' then
80             reg_ball <= "1000000000000000";
81         elsif enable = '1' and dir_ball = '1' then
82             reg_ball <= reg_ball(14 downto 0) & '0';
83         elsif enable = '1' and dir_ball = '0' then
84             reg_ball <= '0' & reg_ball(15 downto 1);
85         end if;
86     end if;
87 end process;
88
89 -- atribuicao concorrente dos leds
90 led <= reg_ball;
```

Nota: também pode usar diretivas sll e srl

Projeto ping-pong leds. Lógica do jogo

```
93 process(clk,reset)
94 variable v_pp1 : integer range 0 to 9 := 0;
95 variable v_pp2 : integer range 0 to 9 := 0;
96 begin
97     if reset='1' then
98         espera_p1 <= '0';
99         espera_p2 <= '0';
100         pp1 <= (others=>'0');
101         pp2 <= (others=>'0');
102         v_pp1 := 0;
103         v_pp2 := 0;
104         dir_ball <= '1';
105     elsif rising_edge(clk) then
106         if reg_ball(15) = '1' and sw(1) = '1' then -- player 2 rebate a bola
107             espera_p2 <= '0';
108             dir_ball <= '0';
109         elsif reg_ball(15) = '1' and sw(1) = '0' and espera_p2 = '0' then -- ponto p1 espera player 2 rebater
110             espera_p2 <= '1';
111             dir_ball <= '0';
112             v_pp1 := v_pp1 + 1;
113             if v_pp1 = 9 then
114                 v_pp1 := 0;
115             end if;
116         elsif reg_ball(0) = '1' and sw(0) = '1' then -- player 1 rebate a bola
117             espera_p1 <= '0';
118             dir_ball <= '1';
119         elsif reg_ball(0) = '1' and sw(0) = '0' and espera_p1 = '0' then -- ponto p2 espera player 1 rebater
120             espera_p1 <= '1';
121             dir_ball <= '1';
122             v_pp2 := v_pp2 + 1;
123             if v_pp2 = 9 then
124                 v_pp2 := 0;
125             end if;
126         end if;
127         pp1 <= conv_std_logic_vector(v_pp1,4);
128         pp2 <= conv_std_logic_vector(v_pp2,4);
129     end if;
130 end process;
```

Projeto ping-pong leds. Mapeando pinos de IO

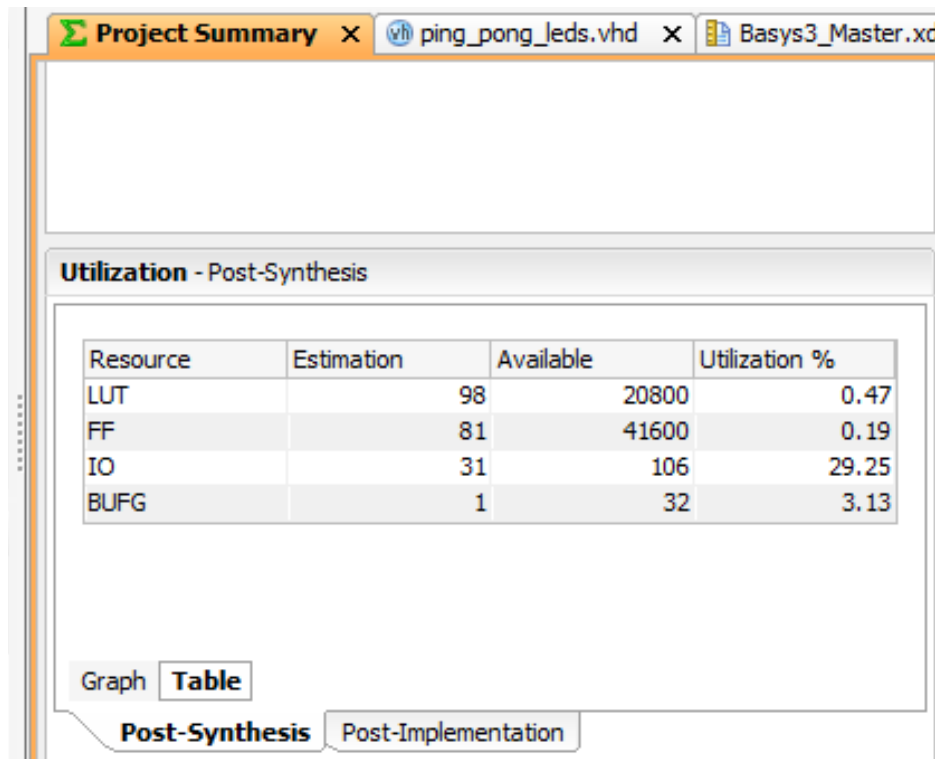
```
lanager - ping_pong_leds
Project Summary x ping_pong_leds.vhd x Basys3_Master.xdc* x ping_pong_leds_v2.vhd x
C:/FPGAprojects/PED2/ping_pong_leds/ping_pong_leds.srcs/constrs_1/imports/PED2/Basys3_Master.xdc
5
6 # Clock signal
7 set_property PACKAGE_PIN W5 [get_ports clk]
8     set_property IOSTANDARD LVCMOS33 [get_ports clk]
9     create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
10
11 ## Switches
12 set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
13     set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
14 set_property PACKAGE_PIN R2 [get_ports {sw[1]}]
15     set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
16
17 ## LEDs
18 set_property PACKAGE_PIN U16 [get_ports {led[0]}]
19     set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
20 set_property PACKAGE_PIN E19 [get_ports {led[1]}]
21     set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
22 set_property PACKAGE_PIN U19 [get_ports {led[2]}]
23     set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
24 set_property PACKAGE_PIN V19 [get_ports {led[3]}]
25     set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
26 set_property PACKAGE_PIN W18 [get_ports {led[4]}]
27     set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
```

Dica: adicione ao projeto o arquivo de constrains Basys3_Master.xdc e descomente as portas usadas.

E agora ... análise !



Projeto ping-pong leds. Reporte de consumo de recursos



The screenshot shows the 'Project Summary' window with the 'Utilization - Post-Synthesis' tab selected. It displays a table with resource utilization data. At the bottom, there are tabs for 'Post-Synthesis' and 'Post-Implementation', with 'Post-Synthesis' being the active tab. A 'Table' button is also visible.

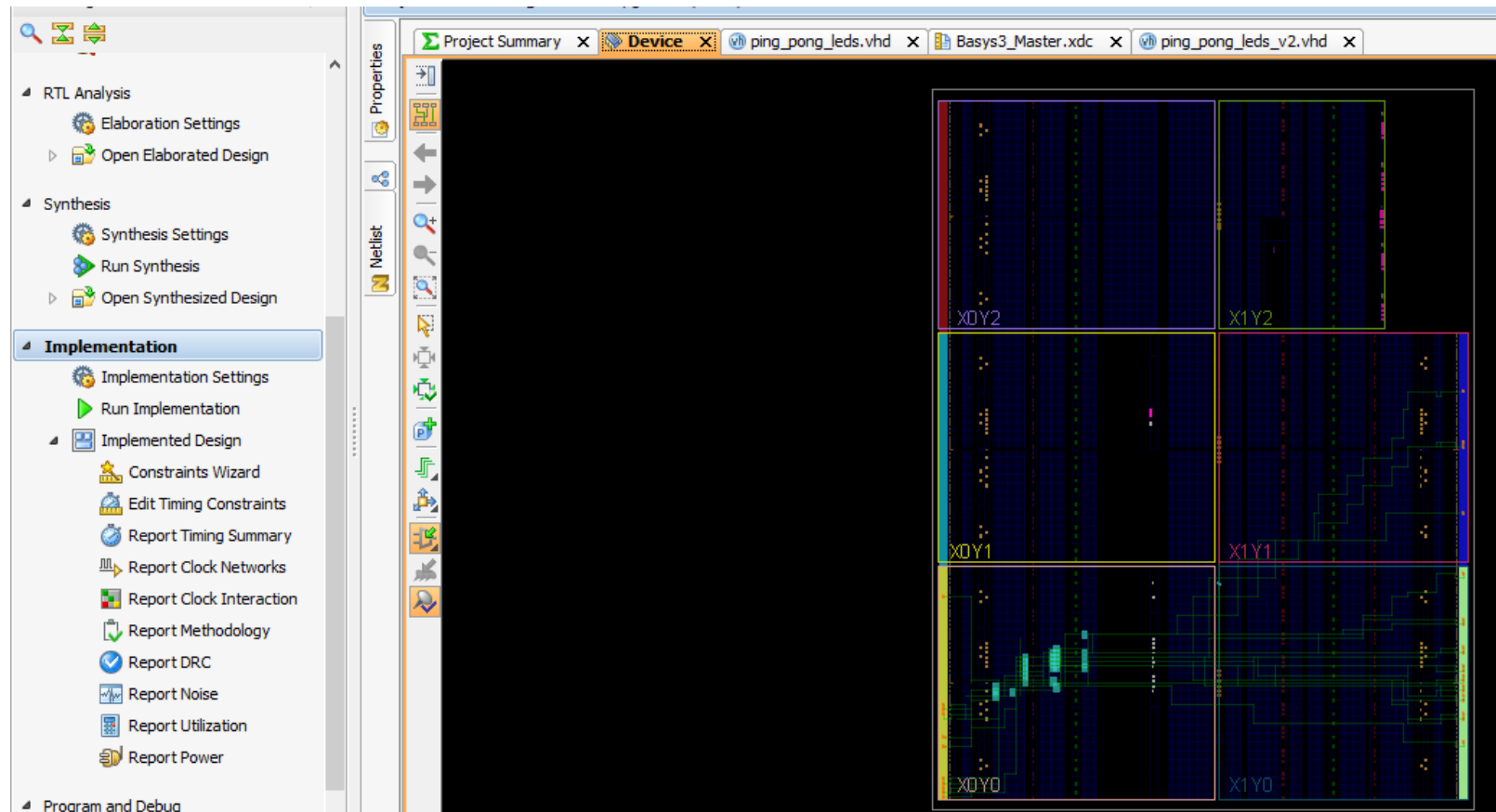
Resource	Estimation	Available	Utilization %
LUT	98	20800	0.47
FF	81	41600	0.19
IO	31	106	29.25
BUFG	1	32	3.13

Nota1: Após a síntese lógica é obtida uma netlist que contém uma descrição do circuito em termos de equações lógicas booleanas, registradores, muxes, etc.

Nota2: é possível estimar o consumo de recursos após a síntese lógica.

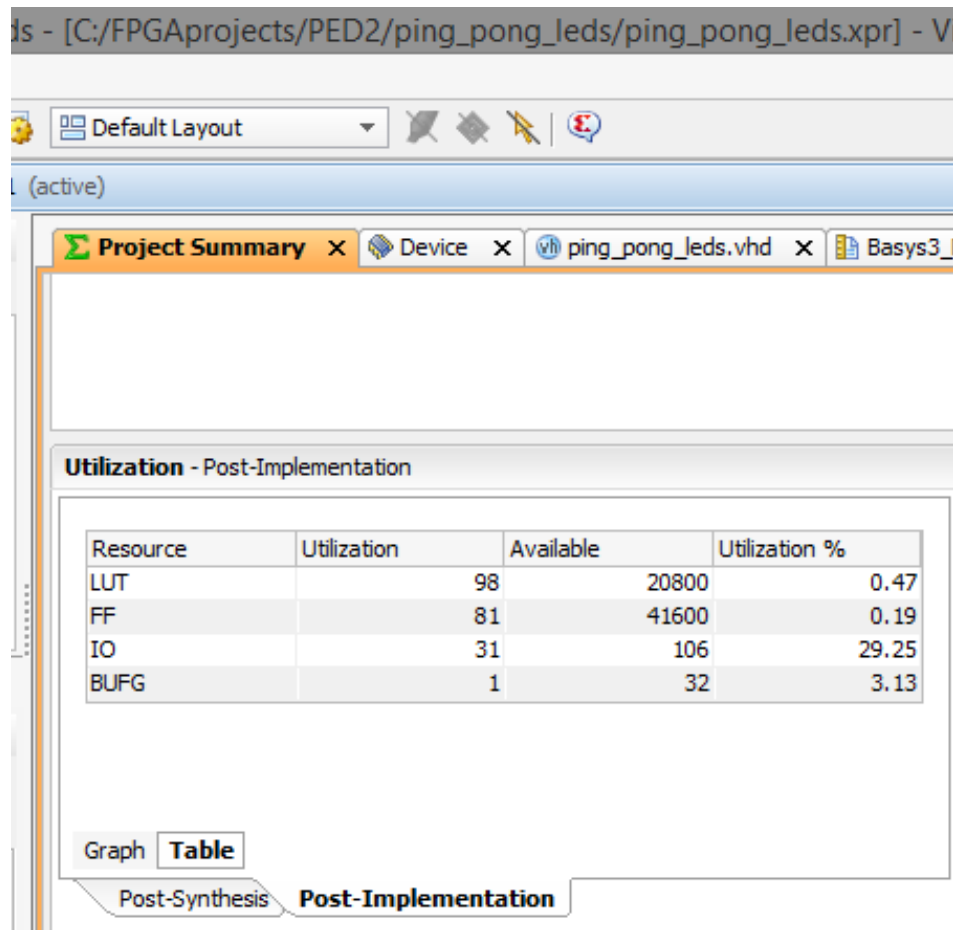
Dica: use o arquivo .rpt disponível na pasta .runs/synth_1 do diretório de trabalho. Esse arquivo tem maior detalhe sobre a ocupação do circuito.

Projeto ping-pong leds. Implementação e layout do circuito



Nota1: Após o processo de *Place and Route* - PAR o layout do circuito pode ser visualizado selecionando “*Implemented Design*”. Para visualizar o roteamento selecione o botão “Routing resources”. Faça zoom sobre a área roteada, encontre um *slice* usado e verifique suas propriedades.

Projeto ping-pong leds. Reporte final de consumo de recursos



Utilization - Post-Implementation

Resource	Utilization	Available	Utilization %
LUT	98	20800	0.47
FF	81	41600	0.19
IO	31	106	29.25
BUFG	1	32	3.13

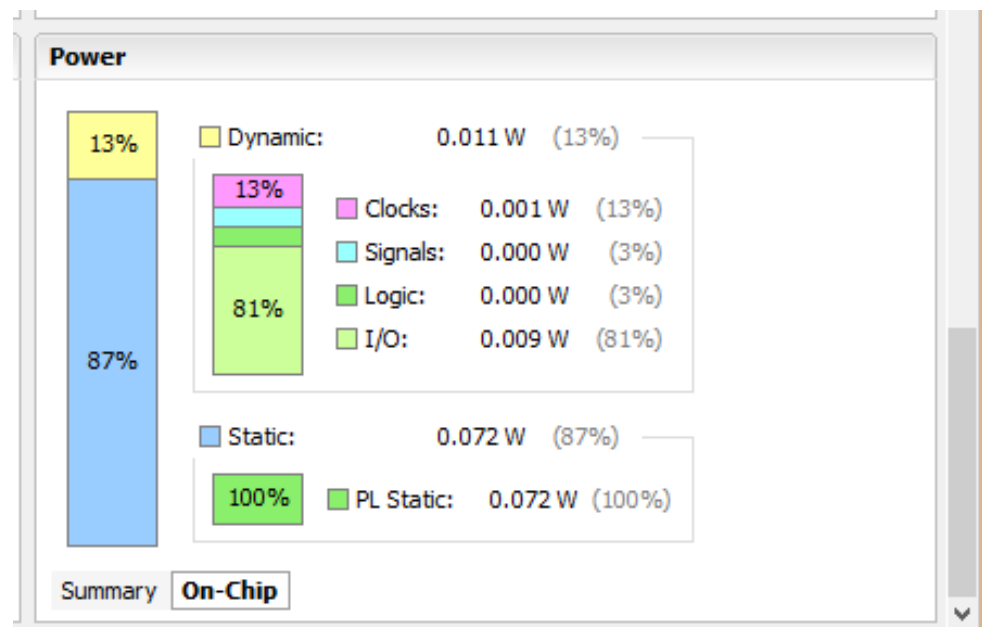
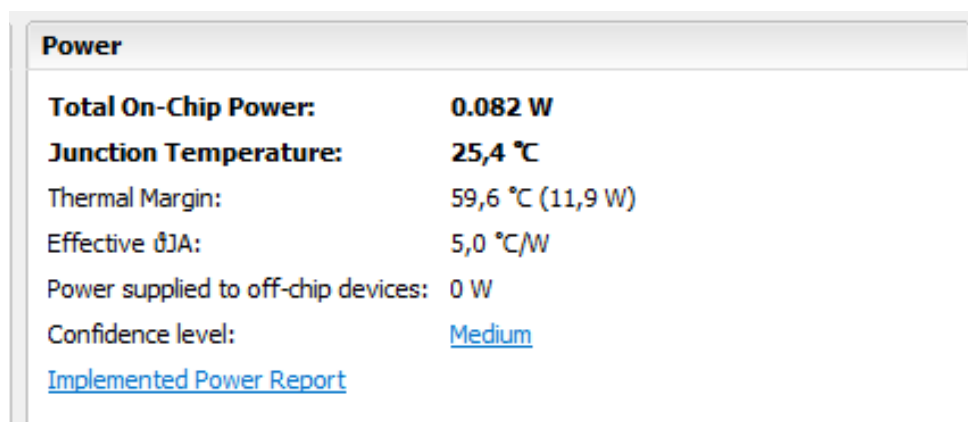
Graph **Table**

Post-Synthesis **Post-Implementation**

Nota1: Após a implementação do circuito (pós PAR) é obtida uma netlist que contém uma descrição do circuito final em termos de LUTs, Ffs, muxes, slices, blocos DSP, blocos BRAM, e o respectivo roteamento usando switch blocks.

Dica: use o arquivo nomeprojeto_utilization_placed.rpt disponível na pasta .runs/impl_1 do diretório de trabalho. Esse arquivo tem maior detalhe sobre a ocupação do circuito.

Projeto ping-pong leds. Reporte de consumo de energia



Nota1: O grau de ativação de cada elemento do circuito, a temperatura ambiente, a temperatura de junção, entre outros fatores determinam o consumo de energia do circuito.

Nota2: a potência estática é devido principalmente ao fato de manter o chip FPGA alimentado (sem lógica implementada). A potência dinâmica se refere ao consumo devido à lógica implementada no FPGA.

E agora ...
Gerar bitstream, testar e jogar !

Questionário

1. O circuito funciona adequadamente? Como resolver a oscilação dos switches?
2. Implemente a lógica para evitar que os players mantenham switches em '1'.
3. Implemente o mesmo circuito usando diretivas sll e srl para os registradores de deslocamento e a biblioteca numeric_std.
4. Implemente a lógica do jogo usando uma máquina de estados finitos

Conversões de tipos de dados usando numeric_std

