A L U

ARITHMETIC LOGIC UNIT

ELE0518

21. Jun 2019

COMPONENTES

BRUNO Matias de S.
LEVY Gabriel da S. G.
PEDRO Henrique de S. F. dos S.

Prof. CARLOS Yuri F. S.

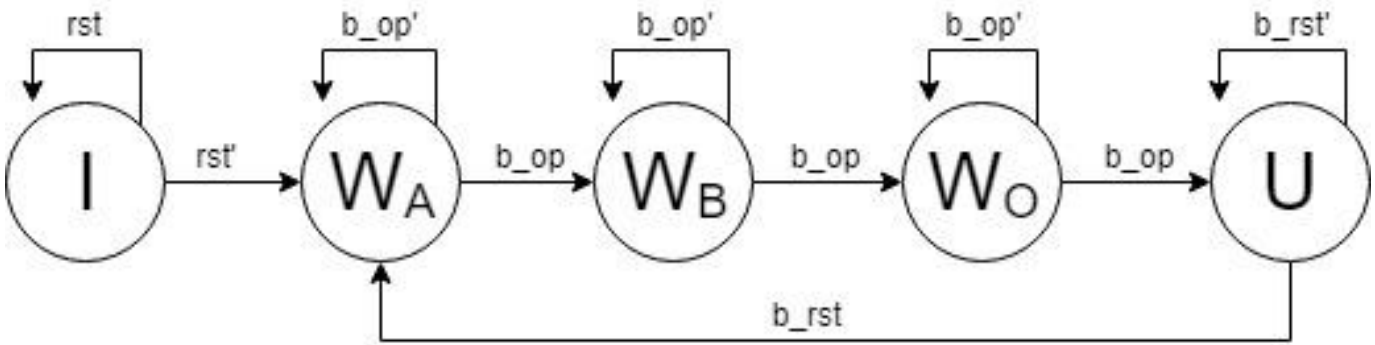
SUMÁRIO

- Proposta do projeto;
- Máquina de estados;
- Discussão do código;
  - Entidade principal;
  - Componentes;
  - Sinais;
  - Funcionamento das arquiteturas;
- Simulações;
- Desafios;
- Conclusões;

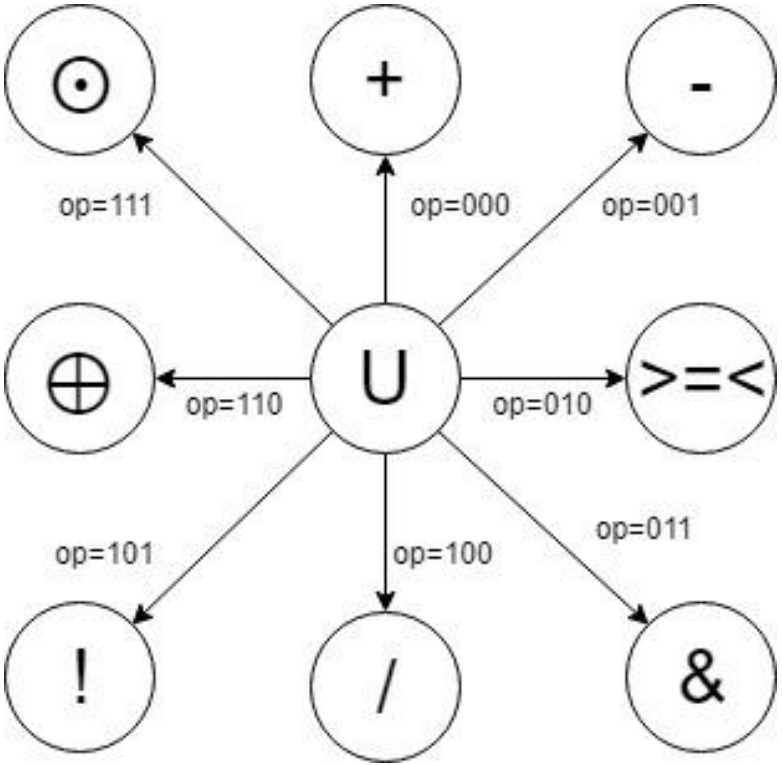
Proposta do projeto

Número mostrado pelo display	Operação
1	SOMA
2	SUBTRAÇÃO
3	COMPARAÇÃO (A>B, A<B e A=B)
4	AND
5	OR
6	NOT
7	XOR
8	XNOR

Máquina de estados



Máquina de estados



Discussão do código

- Entidade principal;
- Componentes;
- Sinais;
- Arquiteturas.

Entidade principal (project3)

```
entity project3 is
     port(
          num_in : in std_logic_vector (4 downto 0);
          b_op   : in std_logic;
          b_rst  : in std_logic;
          rst    : in std_logic;
          op_in  : in std_logic_vector (2 downto 0);
          s      : buffer std_logic_vector (4 downto 0);
          csig   : buffer std_logic;
          clk_in : in std_logic;
          clk    : buffer std_logic;
          dis_0, dis_1, dis_2, dis_3, dis_4 : out std_logic_vector (6 downto 0)
     );

end project3;
```

Componente (ClockDiv)

```
component ClockDiv is
    port(
        clkIn  : in std_logic;
        clkOut : out std_logic
    );
end component;
```

Componente (ALU)

```
component ALU is
    port(
        a    : in std_logic_vector (4 downto 0);
        b    : in std_logic_vector (4 downto 0);
        o    : in std_logic_vector (2 downto 0);
        show : in integer;
        s    : out std_logic_vector (4 downto 0);
        csig : out std_logic
    );
    end component;
```

Componente (divider)

```
component divider is
     port(
          a    : in std_logic_vector (4 downto 0);
          show : in integer;
          csig : in std_logic;
          op   : in std_logic_vector(2 downto 0);
          s    : in std_logic_vector (4 downto 0);
          dis_0, dis_1, dis_2, dis_3, dis_4: out std_logic_vector (3 downto 0)
          );
     end component;
```

Componente (BCD27)

```
entity BCD27 is
    port(
        bcd: in std_logic_vector (3 downto 0);
        seg: out std_logic_vector (6 downto 0)
        );
end BCD27;
```

Sinais

project3

```
type state is (init, wait_a, wait_b, wait_o, unit);
signal actual_s, next_s: state;
signal a_temp, b_temp: std_logic_vector (4 downto 0) := "00000";
signal o_temp: std_logic_vector (2 downto 0) := "000";
signal show: integer := 0;
signal d0, d1, d2, d3, d4: std_logic_vector (3 downto 0) := "0000";
```

Sinais

ALU

```
signal vta, vtb: std_logic_vector (5 downto 0) := "000000";
signal temp_s, temp_a, temp_b : signed (5 downto 0) := "000000";
```

divider

```
signal temp_s, d1, d0: signed (4 downto 0) := "00000";
signal vta: std_logic_vector (4 downto 0) := "00000";
signal opt, optt: std_logic_vector (3 downto 0) := "0000";
```

ClockDiv

```
signal count : integer range 0 to 50000001;
```

Funcionamento das arquiteturas

- Insere-se cada variável (**`a, b, op`**);
- Em cada estado a variável `show` assume certo valor;
- As variáveis `show` e `op` são avaliadas:
  - O componente **`ALU`** avalia se deve fazer alguma operação e qual deve fazer;
  - O componente **`divider`** gera o código BCD para cada display de acordo com o que deve ser mostrado;
- O componente **BCD27** converte o código BCD oriundo do **`divider`** para o código dos displays de 7 segmentos.

Funcionamento das arquiteturas (ALU)

```
ula: process (show)
    begin
    if (show=3) then
        vta(3 downto 0) <= a(3 downto 0);
        vtb(3 downto 0) <= b(3 downto 0);

        if (a(4)='1') then
            temp_a <=  signed(not vta)+1;
        else
            temp_a <= signed(vta);
        end if;
```

Funcionamento das arquiteturas (ALU)

```
if (b(4)='1') then
    temp_b <=  signed(not vtb)+1;
else
    temp_b <= signed(vtb);
end if;

temp_a(5) <= a(4);
temp_b(5) <= b(4);
```

Funcionamento das arquiteturas (ALU)

```
case o is
    when "000" => -- soma
        temp_s <= temp_a + temp_b;
        s <= std_logic_vector(temp_s(4 downto 0));
        csig <= temp_s(5);
    when "001" => -- subtracao
        temp_s <= temp_a - temp_b;
        s <= std_logic_vector(temp_s(4 downto 0));
        csig <= temp_s(5);
```

Funcionamento das arquiteturas (ALU)

```
when "010" => -- comparacao
    if(temp_a>temp_b) then
        s <= "00100"; -- maior
    elsif(temp_a=temp_b) then
        s <= "00010"; -- igual
    else
        s <= "00001"; -- menor
    end if;
```

Funcionamento das arquiteturas (ALU)

```
        when "011" => -- and
            s <= a and b;
        when "100" => -- or
            s <= a or b;
        when "101" => -- not
            s <= not a;
        when "110" => -- xor
            s <= a xor b;
        when "111" => -- xnor
            s <= a xnor b;
    end case;
```

Funcionamento das arquiteturas (ALU)

```
    else
        s <= "00000";
        csig <= '0';
    end if;
    end process ula;
```

Funcionamento das arquiteturas

```
procALU : ALU port map (a_temp, b_temp, o_temp, show, s, csig);
display : divider port map (a, show, csig, op_in, s, d0, d1, d2, d3, d4);
p_disp4 : BCD27 port map (d4, dis_4);
p_disp3 : BCD27 port map (d3, dis_3);
p_disp2 : BCD27 port map (d2, dis_2);
p_disp1 : BCD27 port map (d1, dis_1);
p_disp0 : BCD27 port map (d0, dis_0);
```

Funcionamento das arquiteturas

```
procALU : ALU port map (a_temp, b_temp, o_temp, show, s, csig);
display : divider port map (a, show, csig, op_in, s, d0, d1, d2, d3, d4);
p_disp4 : BCD27 port map (d4, dis_4);
p_disp3 : BCD27 port map (d3, dis_3);
p_disp2 : BCD27 port map (d2, dis_2);
p_disp1 : BCD27 port map (d1, dis_1);
p_disp0 : BCD27 port map (d0, dis_0);
```

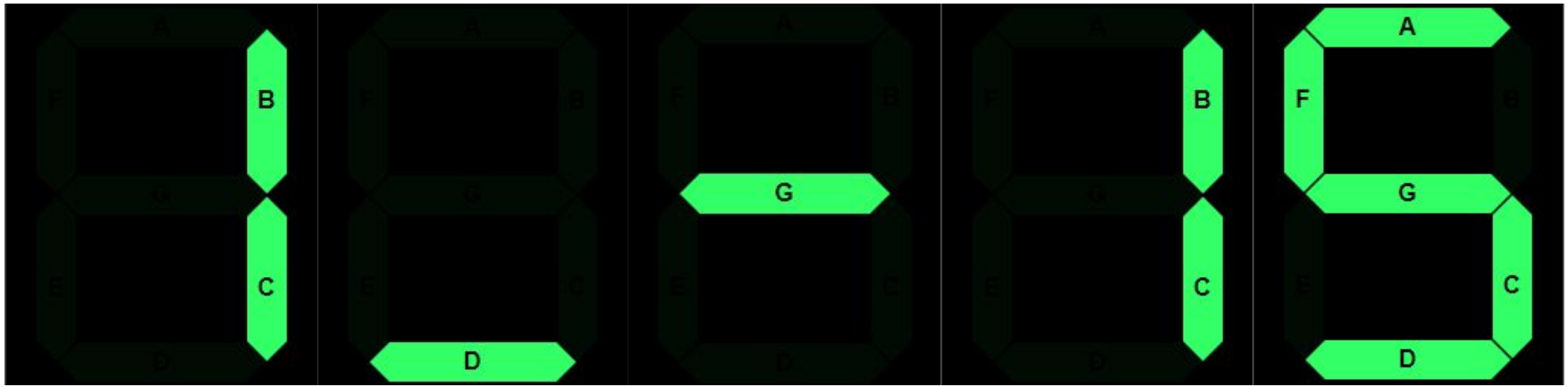
envia

recebe

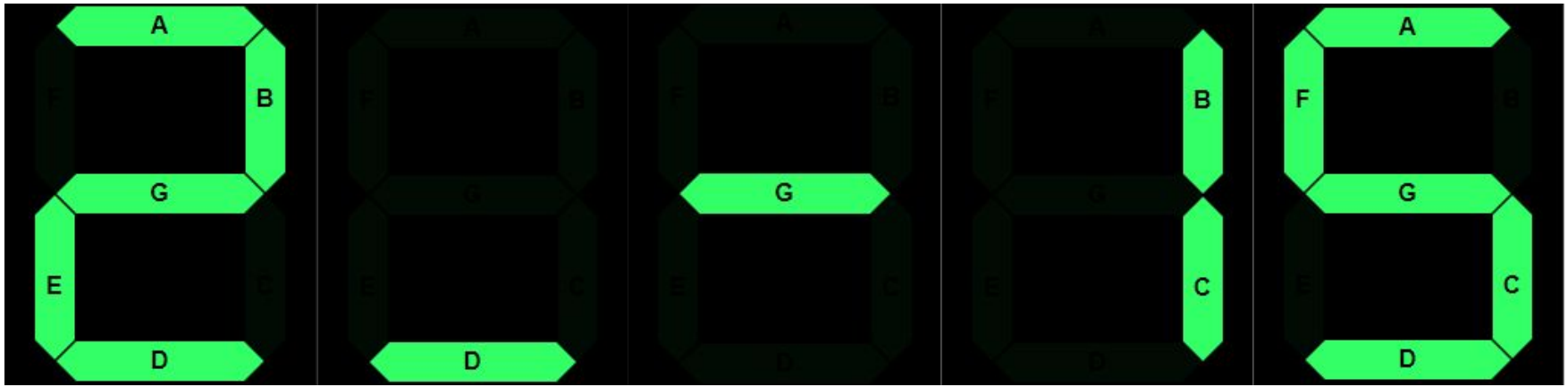
Simulações

- Soma;
- Maior, menor ou igual;
- XNOR.

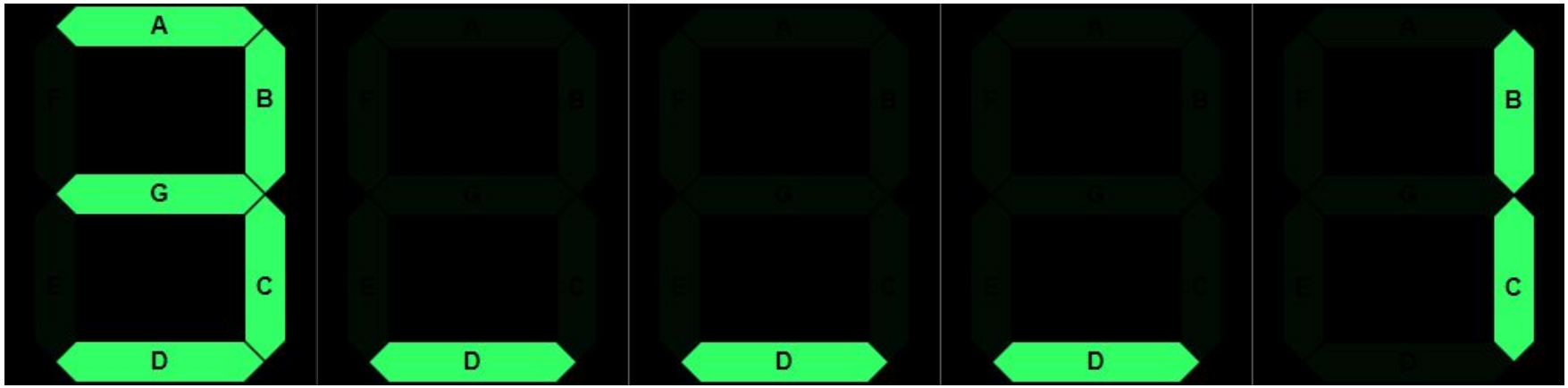
Simulações (+)



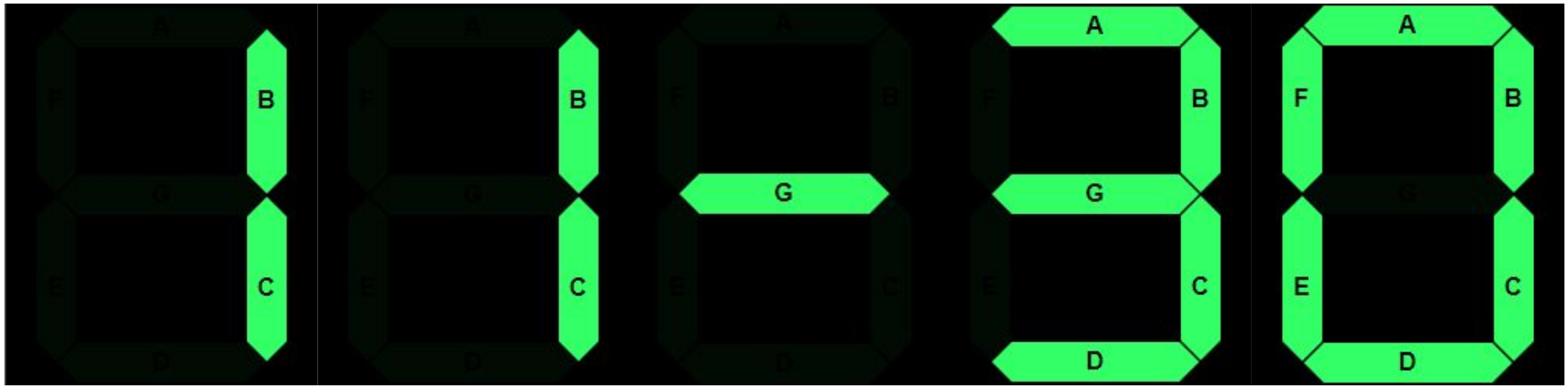
Simulações (+)



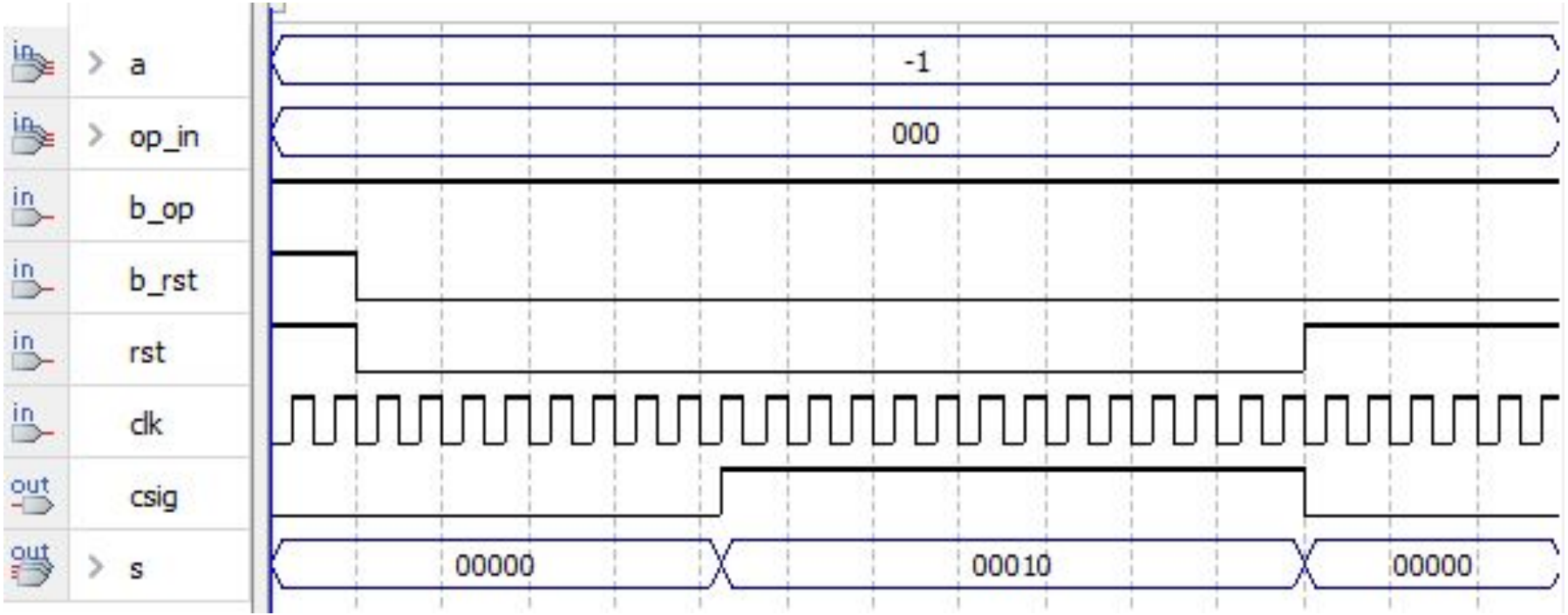
Simulações (+)



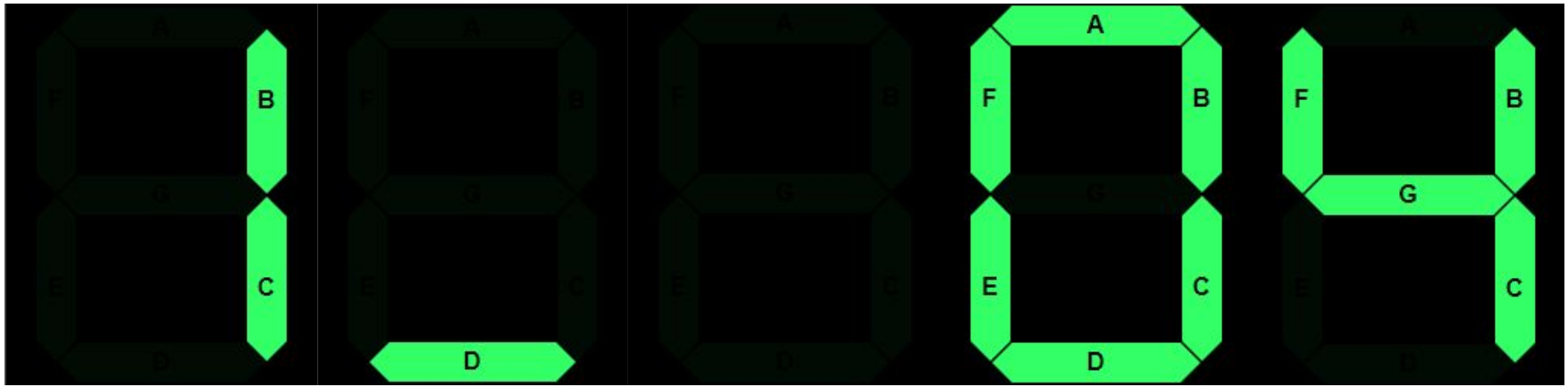
Simulações (+)



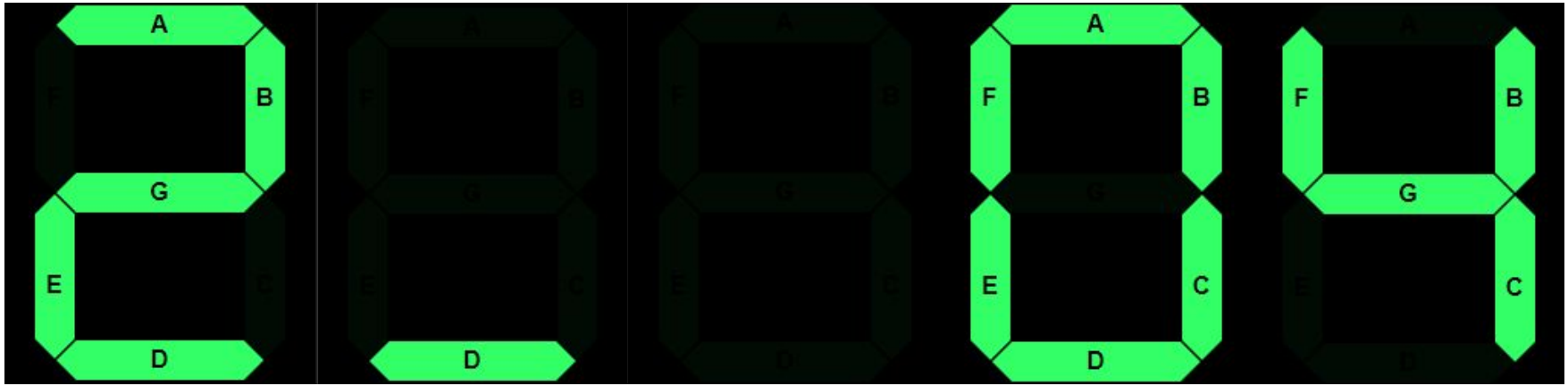
Simulações (+)



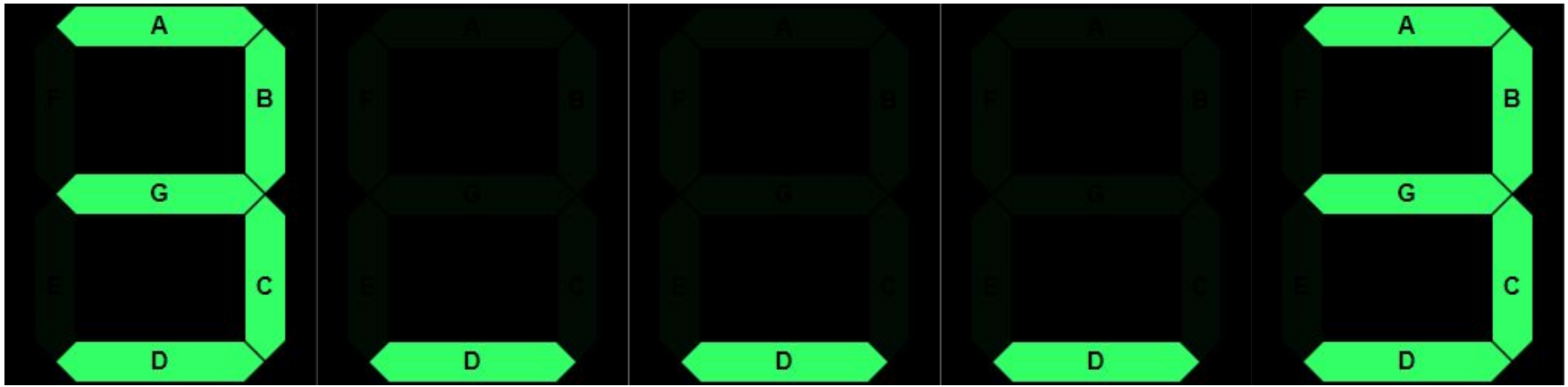
Simulações (>=<)



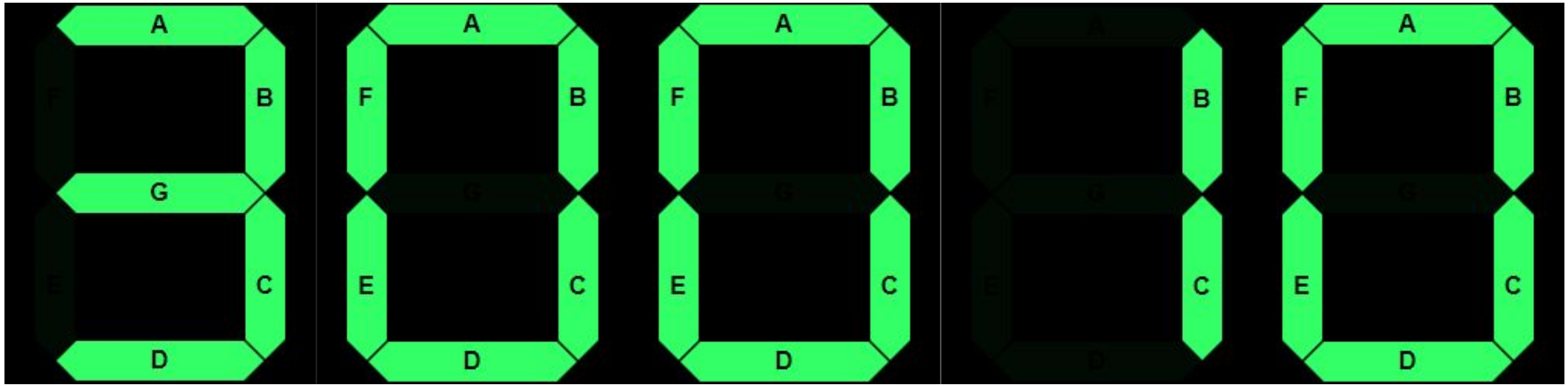
Simulações (>=<)



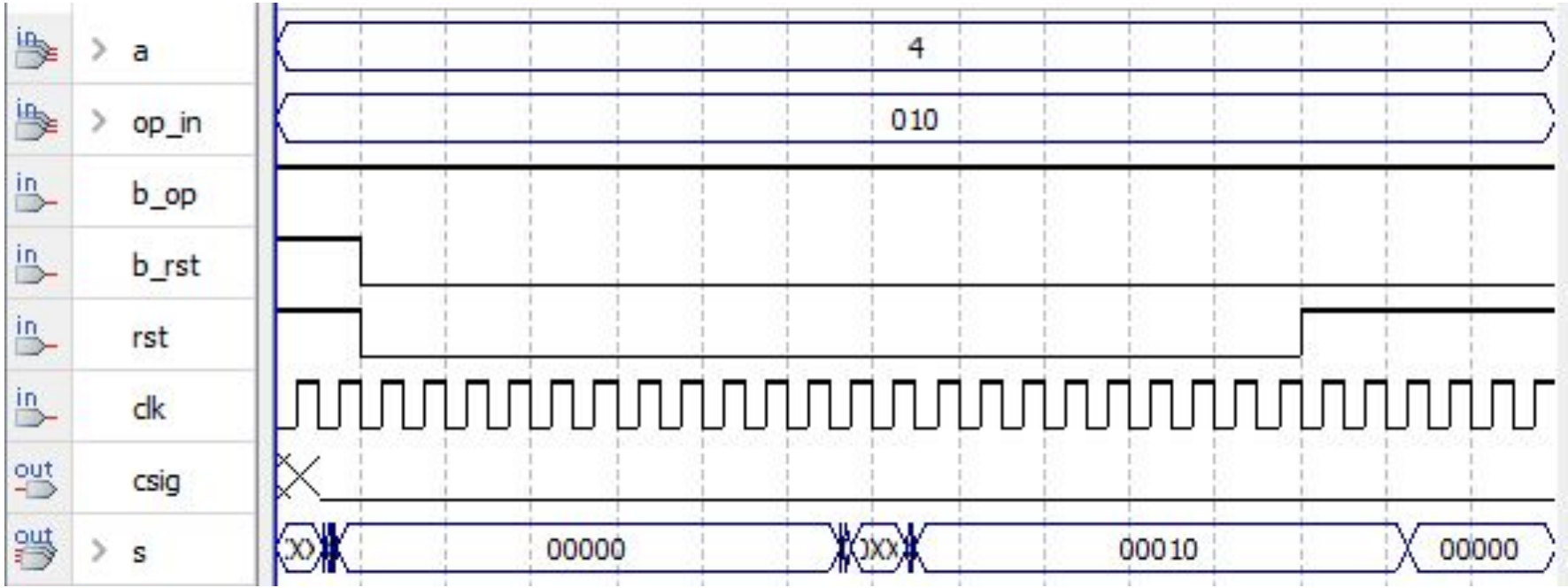
Simulações (>=<)



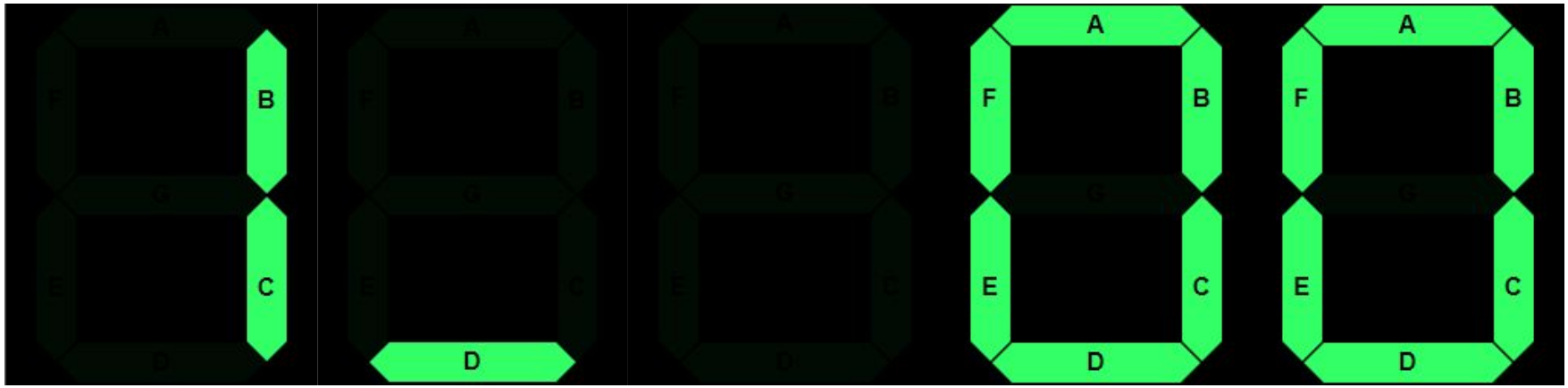
Simulações (>=<)



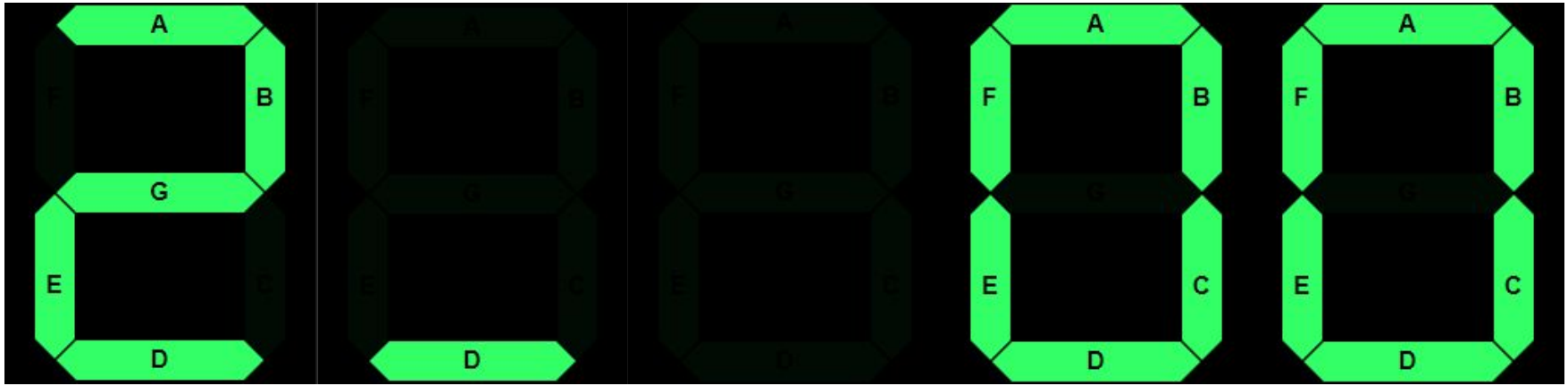
Simulações (>=<)



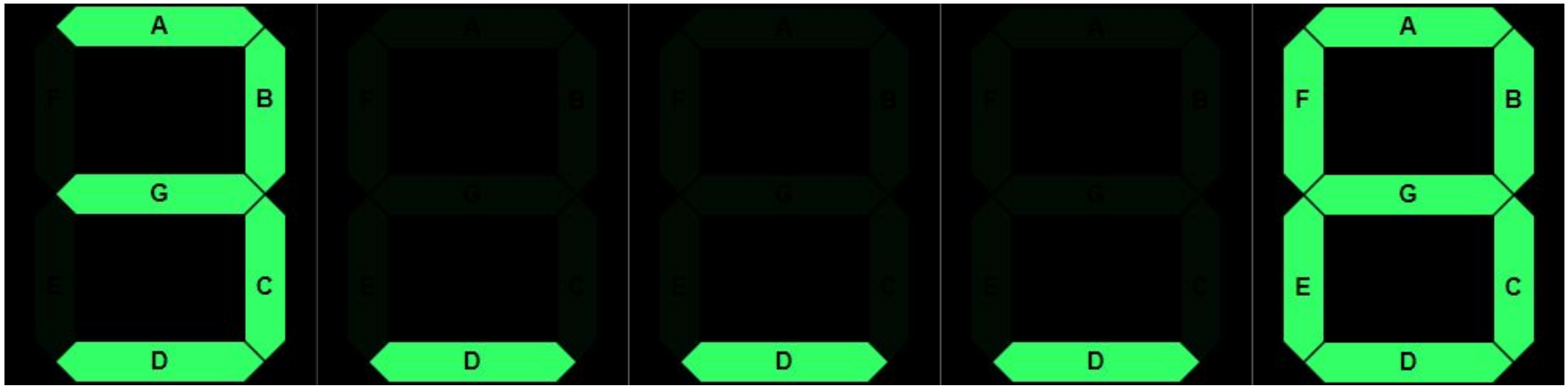
Simulações (XNOR)



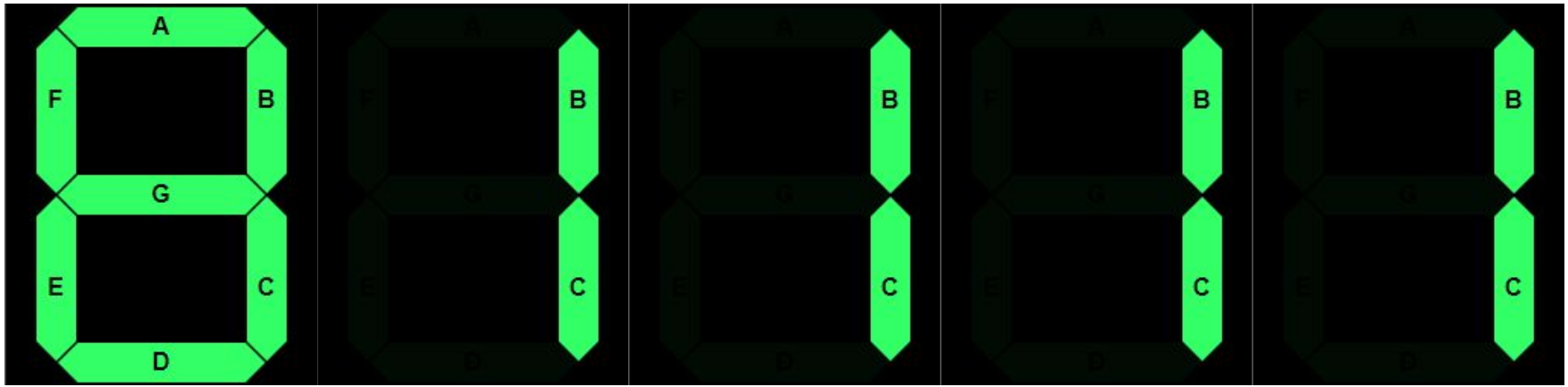
Simulações (XNOR)



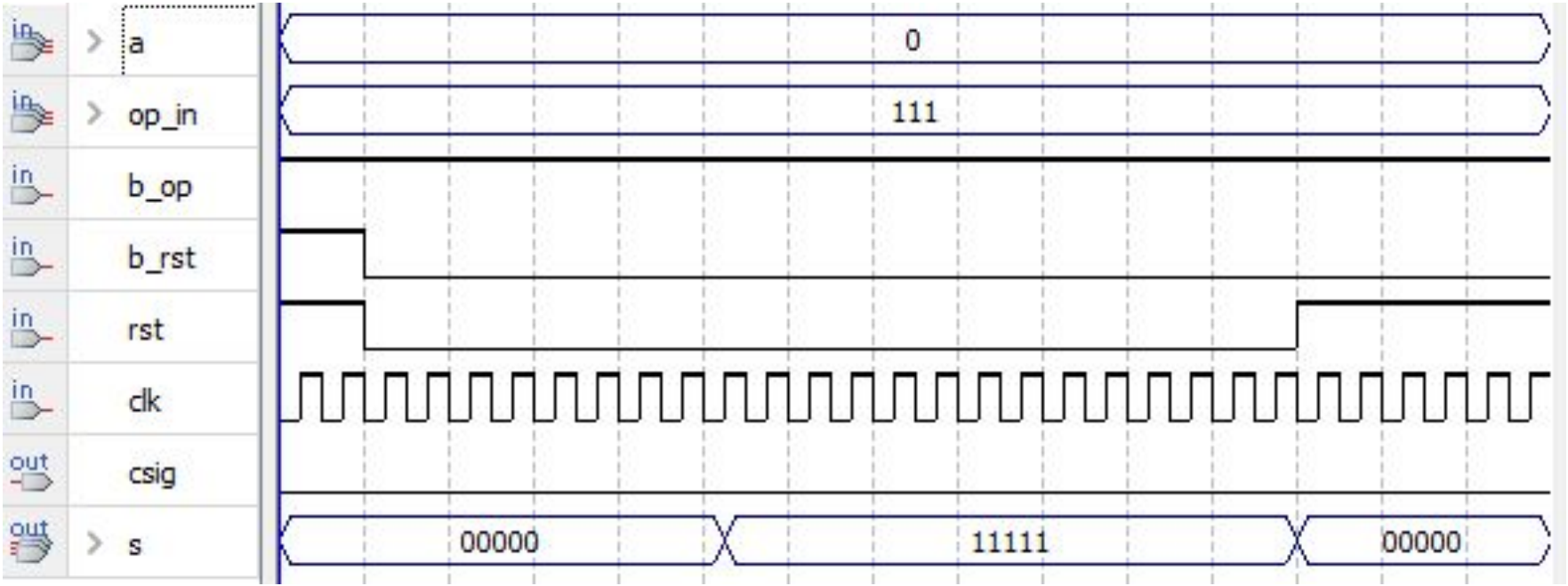
Simulações (XNOR)



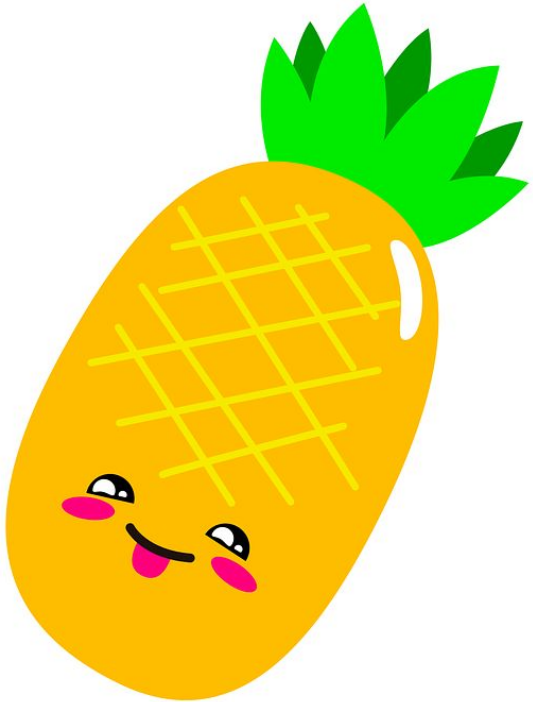
Simulações (XNOR)



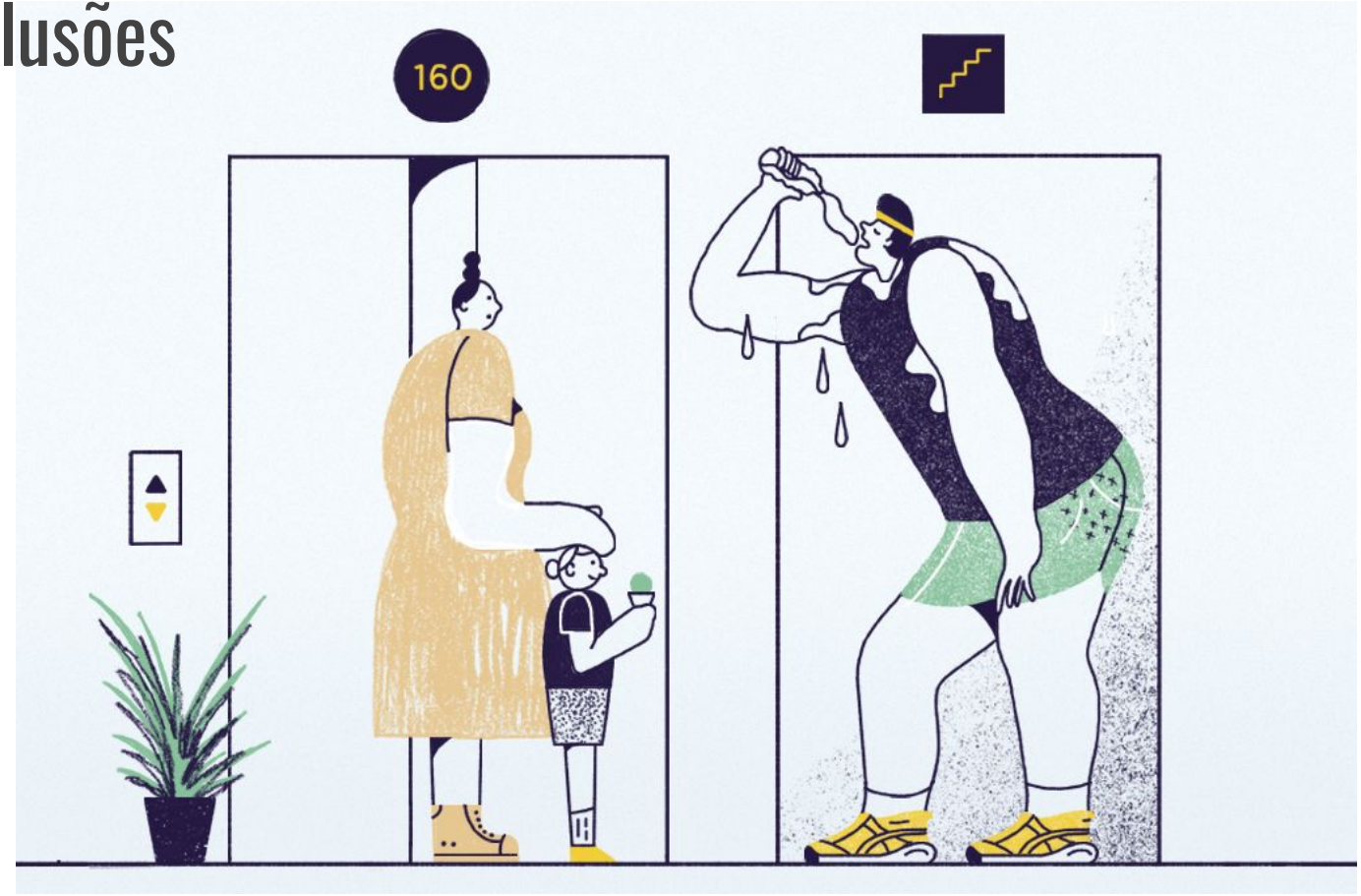
Simulações (XNOR)



Desafios



Conclusões



Referências

- ABNT, Associação Brasileira de Normas Técnicas. **NBR 10719** – Apresentação de relatórios técnico-científicos. Rio de Janeiro: ABNT, Copyright © 1989.
- MARCONI, Marina de A. & LAKATOS, Eva M. **Fundamentos de metodologia científica**. 5 ed. Editora Atlas. São Paulo, 2003.
- Slide e anotações da Aula. **Linguagem de Descrição de Hardware**. Professor: Carlos Yuri Ferreira Silva.
- TOCCI, Ronald J. **Digital Systems**: principles and applications. 11 ed. Pearson Education India, 1991.
- VAHID, Frank. Sistemas Digitais: Projetos, Otimização e HDLs. 1 ed. Editora Bookman, 2008.