Relatório

Link do GitHub: https://github.com/HeitorgOliveira/labSD

Heitor Gomes de Oliveira - 15458350
Dante Brito Lourenço - 15447326
João Gabriel Peroli da Silva - 15678578

20 de Agosto de 2024

1 Introdução:

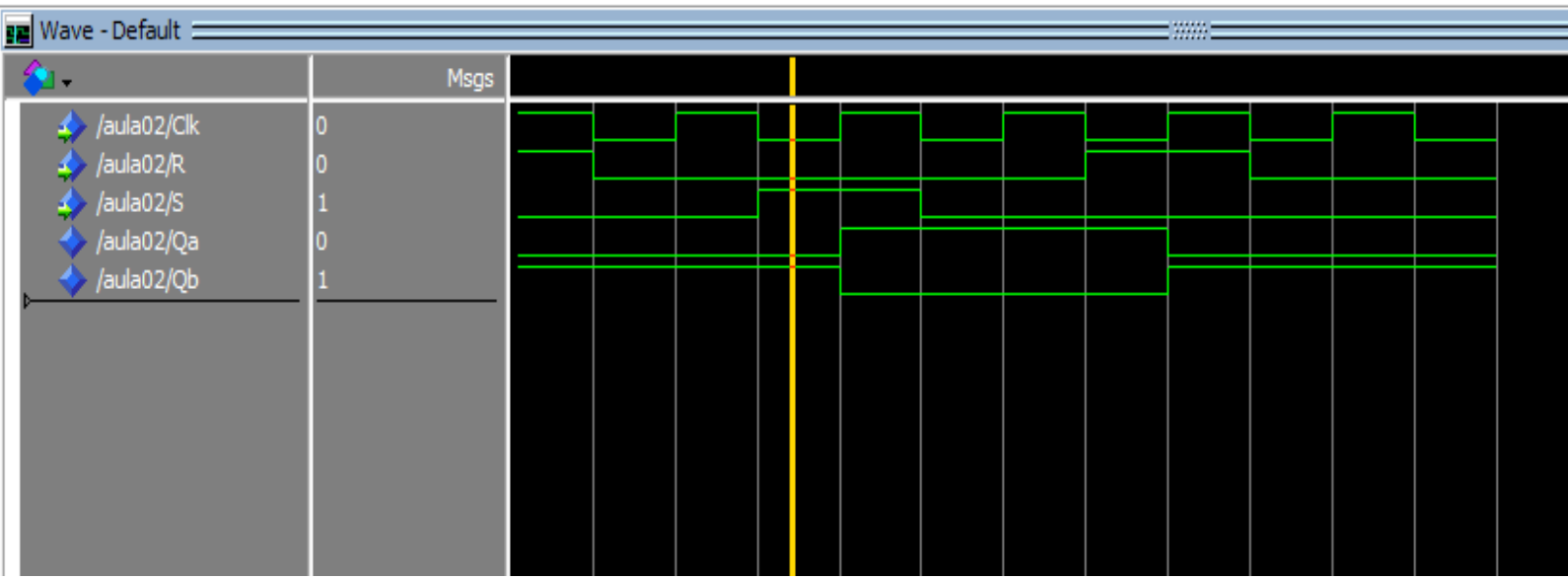
Segue o relatório do trabalho de laboratório realizado neste dia.

Parte I:

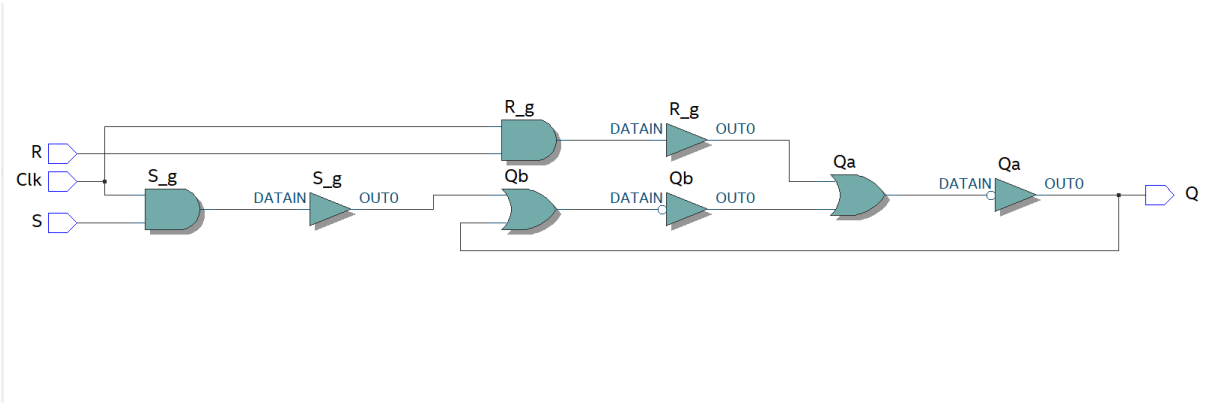
Na parte I simulamos o código em VHDL que estava presente no PDF fornecido. Segue o código:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY part1 IS
PORT ( Clk, R, S : IN STD_LOGIC;
Q : OUT STD_LOGIC);
END part1;
ARCHITECTURE Structural OF part1 IS
SIGNAL R_g, S_g, Qa, Qb : STD_LOGIC ;
ATTRIBUTE KEEP : BOOLEAN;
ATTRIBUTE KEEP OF R_g, S_g, Qa, Qb : SIGNAL IS TRUE;
BEGIN
R_g <= R AND Clk;
S_g <= S AND Clk;
Qa <= NOT (R_g OR Qb);
Qb <= NOT (S_g OR Qa);
Q <= Qa;
END Structural;
```

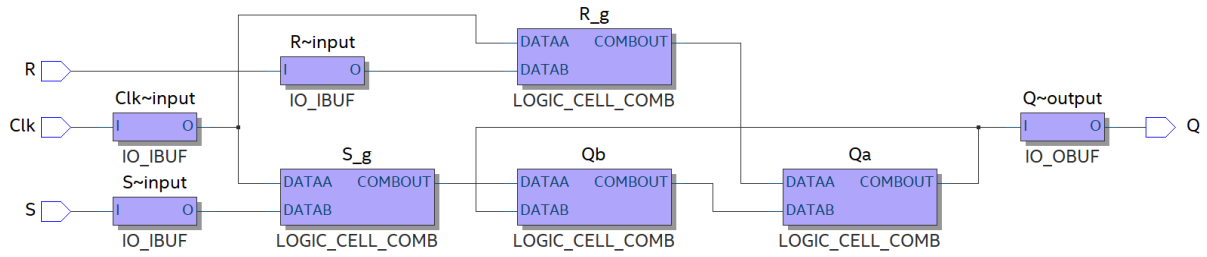
1 - Código VHDL Parte I



2 - Captura de tela do simulador da **parte I**.



3 - Representação das portas lógicas do código I.



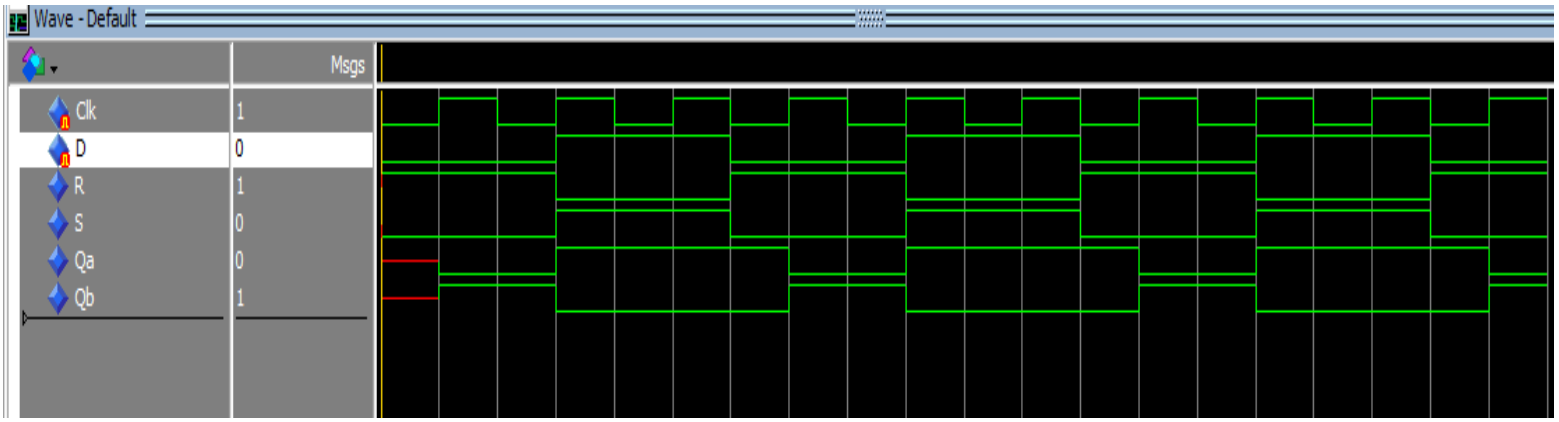
4 - Representação pelo Technology Map Viewer

Parte II:

Nesta parte trocamos as portas por portas NAND no arquivo VHDL e testamos para todas as combinações de entrada.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY parte3 IS
PORT ( Clk, D : IN STD_LOGIC;
Q : OUT STD_LOGIC);
END parte3;
ARCHITECTURE Structural OF parte3 IS
SIGNAL  R, S, R_g, S_g, Qa, Qb : STD_LOGIC ;
ATTRIBUTE KEEP : BOOLEAN;
ATTRIBUTE KEEP OF R_g, S_g, Qa, Qb, R, S : SIGNAL IS TRUE;
BEGIN
R <= NOT(D);
S <= D;
R_g <= NOT(R AND Clk);
S_g <= NOT(S AND Clk);
Qa <= NOT(R_g AND Qb);
Qb <= NOT(S_g AND Qa);
Q <= Qa;
END Structural;
```

1 - Código VHDL Parte II



2 - Captura de tela do monitor da parte II.

3 - Teste FPGA

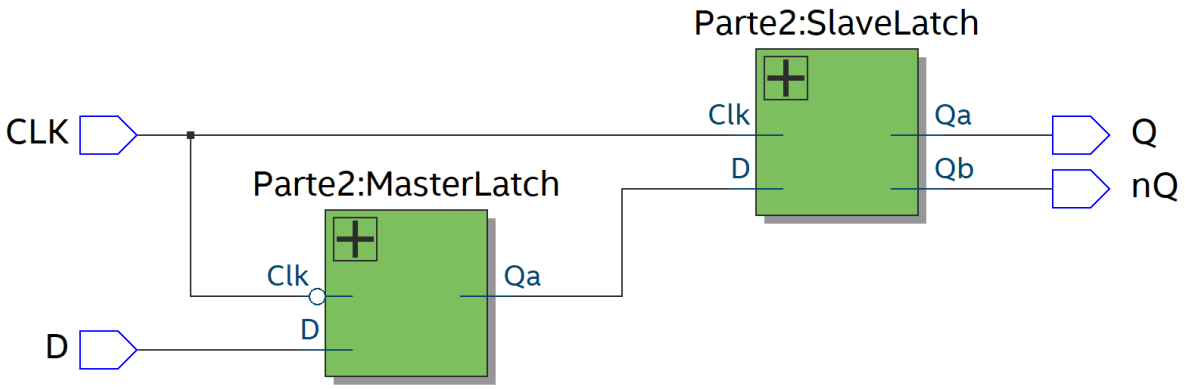
Parte III:

Nesta parte testamos o circuito master-slave D flip-flop na placa FPGA, que funciona da seguinte forma: a cada borda de subida do Clock, o Q final assume o valor que D está assumindo naquele momento.

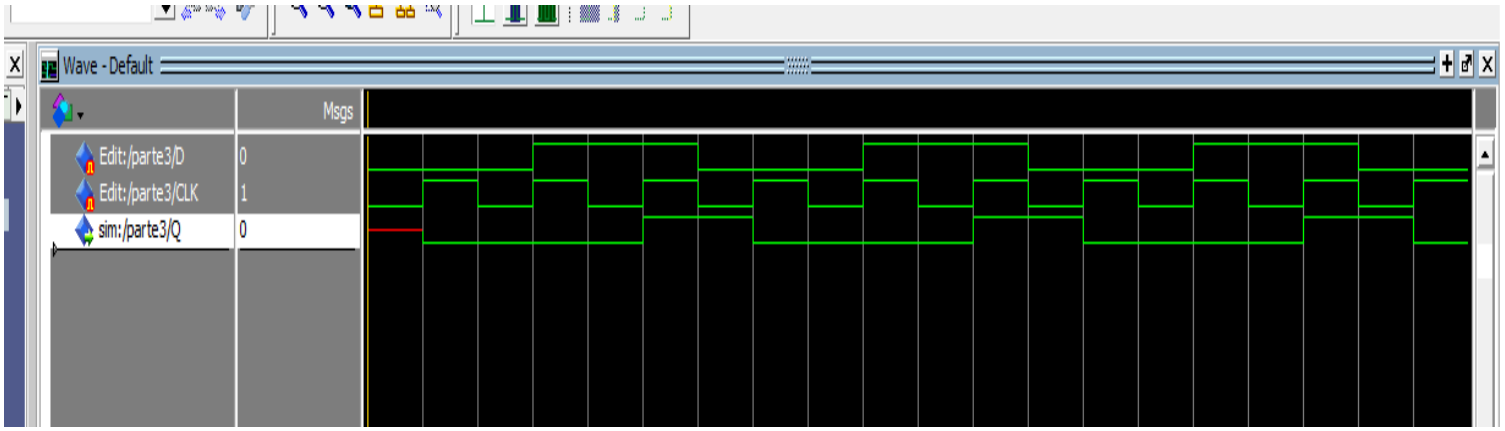
```
library ieee;
use ieee.std_logic_1164.all;

-- Definição do módulo do flip-flop mestre-escravo
entity parte3 is
   Port ( D : in STD_LOGIC;
          CLK : in STD_LOGIC;
          Q : out STD_LOGIC);
end parte3;

architecture Behavioral of parte3 is

   -- Declaração do componente do D latch (Parte2)
   component Parte2
      Port ( Clk : in STD_LOGIC;
             D : in STD_LOGIC;
             Q : out STD_LOGIC);
   end component;

   -- Sinais internos para conectar o latch mestre e escravo
   signal Q_master : STD_LOGIC;
   signal CLK_inv : STD_LOGIC;

begin

   -- Inverter o sinal de clock para o latch escravo
   CLK_inv <= not CLK;

   -- Instanciação do latch mestre
   MasterLatch : Parte2
      Port map ( Clk => CLK_inv,
                 D => D,
                 Q => Q_master);

   -- Instanciação do latch escravo
   SlaveLatch : Parte2
```

```
        Port map ( Clk => CLK,
                   D => Q_master,
                   Q => Q);

end Behavioral;
```

1 - Código VHDL Parte III



2 - Representação pelo Technology Map Viewer



3 - Representação pelo Technology Map Viewer

4 - Teste FPGA

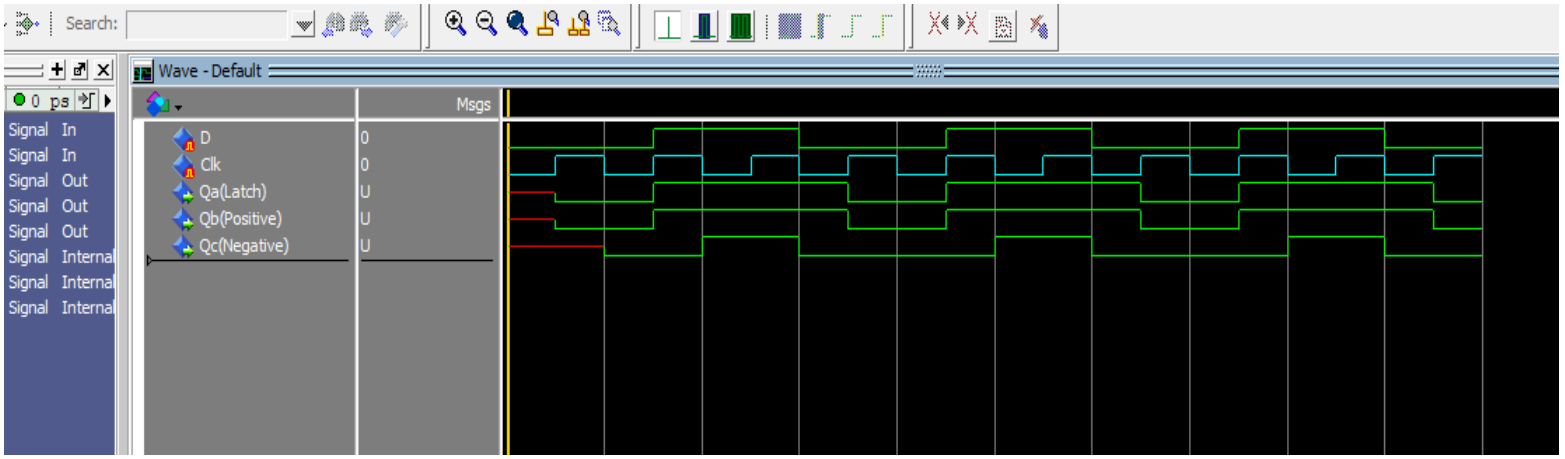
Parte IV:

Nessa parte implementamos um circuito que junta a gated D latch, o positive-edge triggered D flip-flop e o negative-edge triggered D flip-flop. Sua implementação deve seguir de acordo com o gráfico de sinais a seguir:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity storage_elements is
   port (
      D     : in  std_logic;
      Clock : in  std_logic;
      Qa    : out std_logic;
      Qb    : out std_logic;
      Qc    : out std_logic
   );
end entity storage_elements;

architecture behavioral of storage_elements is
   -- Gated D latch
   signal latch_Q : std_logic;

   -- Positive-edge triggered D flip-flop
   signal flip_flop_b_Q : std_logic;

   -- Negative-edge triggered D flip-flop
   signal flip_flop_c_Q : std_logic;

begin
   -- Gated D Latch
   process(D, Clock)
   begin
      if Clock = '1' then
         latch_Q <= D;
      end if;
   end process;

   -- Positive-edge triggered D Flip-Flop
   process(D, Clock)
   begin
      if rising_edge(Clock) then
         flip_flop_b_Q <= D;
      end if;
   end process;
```

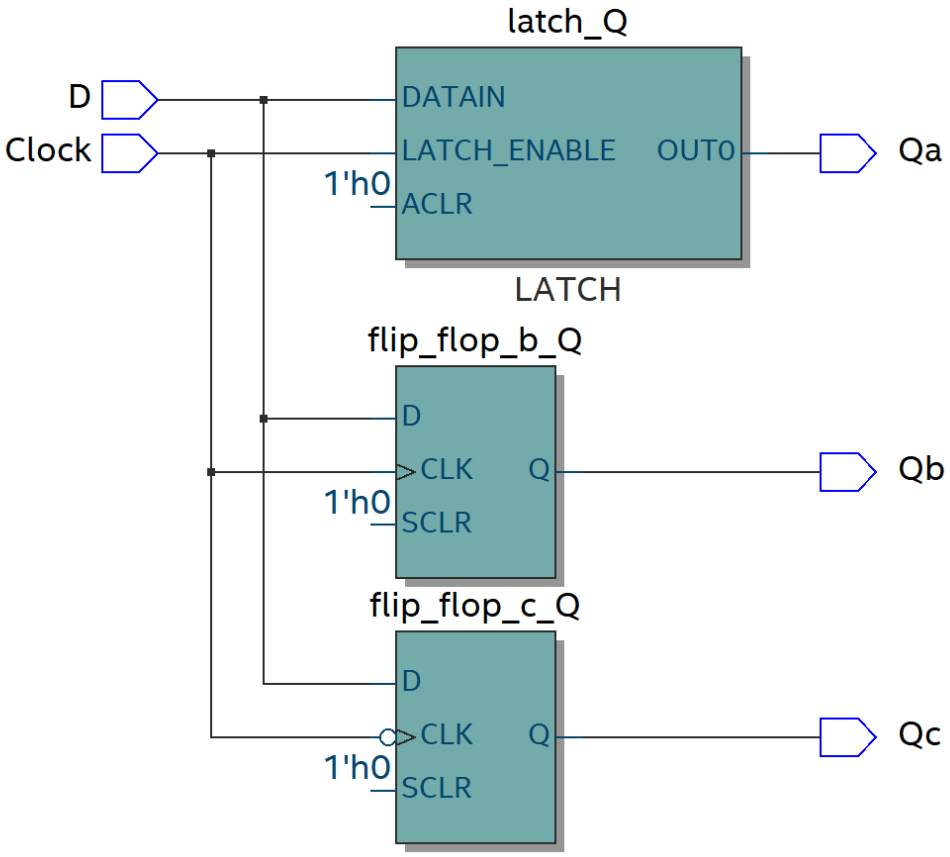
```
    -- Negative-edge triggered D Flip-Flop
    process(D, Clock)
    begin
        if falling_edge(Clock) then
            flip_flop_c_Q <= D;
        end if;
    end process;

    -- Outputs do circuito
    Qa <= latch_Q;
    Qb <= flip_flop_b_Q;
    Qc <= flip_flop_c_Q;

end architecture behavioral;
```

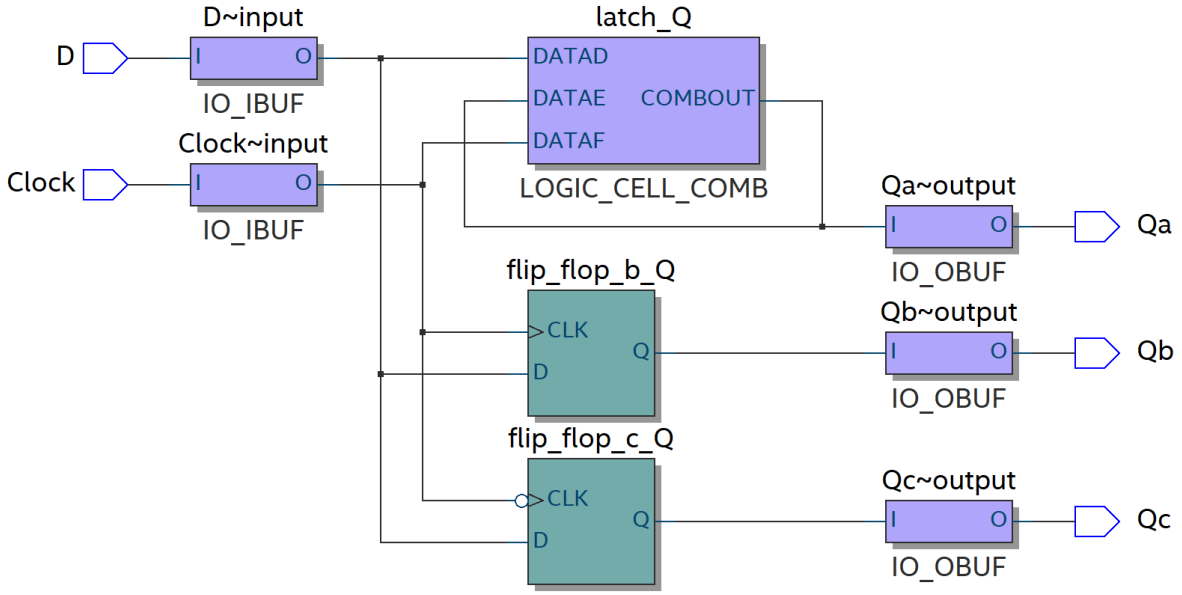
1 - Código VHDL Parte IV

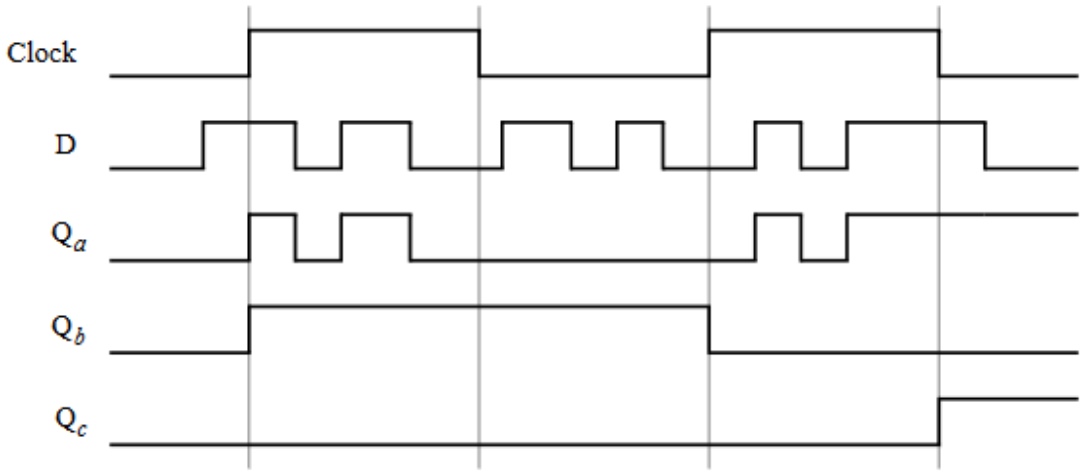


2 - Captura de tela do monitor

3- Map Technology Viewer







4- Comportamento esperado