# Relatório Prática 05 – 22/09/2023

# Arthur Souza/João Paulo – PN1

Iniciou a prática criando uma entidade nomeado FlipFlopD no software QUARTUS II 13.0, especificando qual chip está sendo utilizado, no caso o EP2C35F672C6.

A partir disso, criou o primeiro arquivo em VHDL de mesmo nome que a entidade no qual foi colado uma parte do código disponibilizado:

***LIBRARY IEEE;***

***use ieee.std\_logic\_1164.all;***

***entity FlipFlopD is***

***port( clock: in std\_logic;***

***D: in std\_logic;***

***Q: out std\_logic***

***);***

***end FlipFlopD;***

***architecture RTL of FlipFlopD is***

***begin***

***process(clock)***

***begin***

***if (clock='1' and clock'event) then***

***Q <= D;***

***end if;***

***end process;***

***end RTL;***

Utilizou-se a estrutura logica WHEN-ELSE em architecture para alterar a descrição de comportamental para fluxo de dados, como visto abaixo:

***architecture RTL of FlipFlopD is begin***

***Q <= '0' when(reset='0') else***

***D when(clock='1' and clock'event);***

***end RTL;***

Simulado e atestado ausência de erros, verificou-se a esquematização do código em circuito pela opção RTL viewer e Technology Map Viewer. Segue abaixo as imagens:

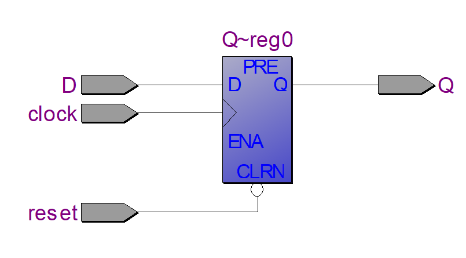


Figura 1: Circuito FlipFlopDr

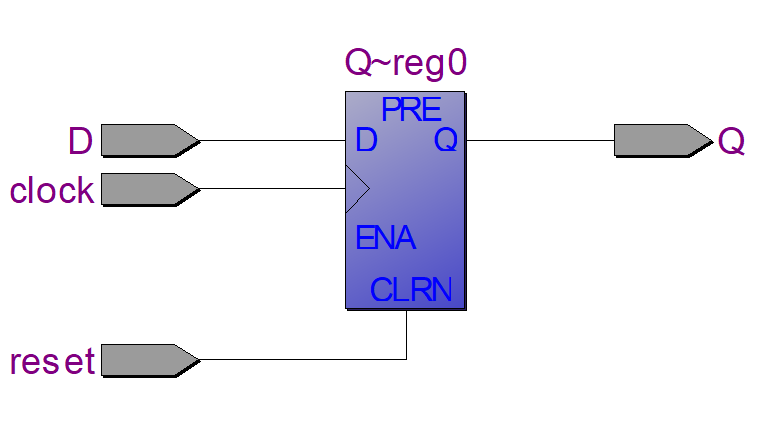


Figura 2: Diagrama do circuito FlipFlopD

Seguiu-se para a compilação do testbench criado, este chamado de tb\_FlipFlopD, e que irá definir os testes do projeto. Ele está descrito abaixo:

***library IEEE;***

***use IEEE.STD\_LOGIC\_1164.all;***

***entity tb\_FlipFlopD is***

***end tb\_FlipFlopD;***

***architecture teste of tb\_FlipFlopD is***

***component FlipFlopD is***

***port(***

***clock: in std\_logic;***

***D: in std\_logic;***

***reset: in std\_logic;***

***Q: out std\_logic***

***);***

***end component;***

***signal fio\_Cl, fio\_D, fio\_Re, fio\_Q: std\_logic;***

***begin***

***-- Note que o componente é instanciado com apenas 4 bits nas entradas para facilitar a simulação:***

***instancia\_FlipFlopD: FlipFlopD port map(clock=>fio\_Cl, D=>fio\_D, reset=>fio\_Re, Q=>fio\_Q);***

***-- Dados de entrada de 4 bits são expressos em "hexadecimal" usando "x":***

***fio\_Cl<='0', '1' after 50ns,'0' after 100ns, '1' after 150ns,'0' after 200ns,'1' after 250ns,'0' after 300ns;***

***fio\_D<='0','1' after 25ns,'0' after 75ns, '1' after 175ns;***

***fio\_Re<='0','1' after 2ns, '0' after 200ns;***

***end teste;***

Com ele, podemos simular o funcionamento do circuito através do software MULTISIM. Basta apenas indica o arquivo como testbench em simulation e começar a simulação em RTL. Irá abrir um gráfico de sinais com valores determinados pelo testbench e irá auxiliar na verificação da lógica.

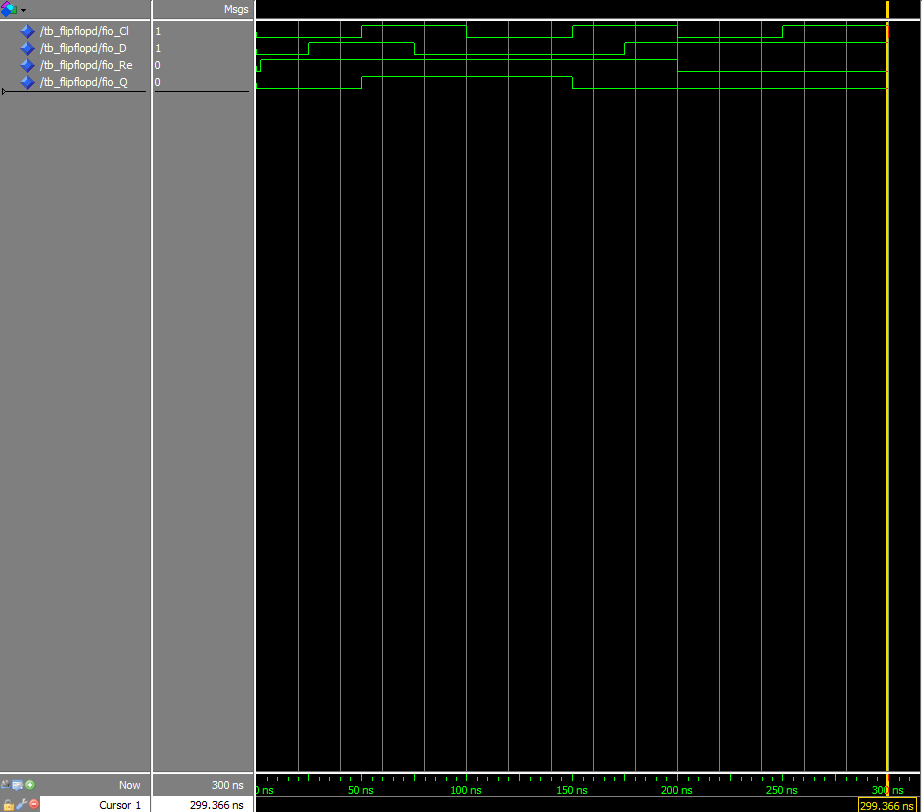


Figura 3: Simulação FlipFlopD no Multisim

Em seguida, foi feito o circuito fulladder de 4 bits, esse criado a partir de um fulladder com descrição de fluxo de dados disponibilizado no sistema, que segue abaixo:

***LIBRARY ieee;***

***USE ieee.std\_logic\_1164.all;***

***entity fulladder is***

***port ( Cin : in std\_logic;***

***x : in std\_logic;***

***y : in std\_logic;***

***s : out std\_logic;***

***Cout : out std\_logic***

***);***

***end fulladder;***

***architecture RTL OF fulladder is***

***begin***

***s <= x XOR y XOR Cin;***

***Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y);***

***end RTL ;***

O novo código, agora em 4 bits e com descrição comportamental, segue abaixo:

***LIBRARY ieee;***

***USE ieee.std\_logic\_1164.all;***

***entity fulladder is***

***generic***

***(***

***DATA\_WIDTH : natural :=4***

***);***

***port (***

***Cin : in std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***x : in std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***y : in std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***s : out std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***Cout : out std\_logic\_vector ((DATA\_WIDTH-1) downto 0)***

***);***

***end fulladder;***

***architecture RTL OF fulladder is***

***begin***

***process(x, y, cin)***

***begin***

***s <= x XOR y XOR Cin;***

***Cout <= (x AND y) OR (Cin AND x) OR (Cin AND y);***

***end process;***

***end RTL ;***

Simulado e atestado ausência de erros, verificou-se a esquematização do código em circuito pela opção RTL viewer e Technology Map Viewer. Segue abaixo as imagens:

Diagrama, Esquemático

Descrição gerada automaticamente

Figura 4: Circuito FullAdder 4bits

Diagrama

Descrição gerada automaticamente

Figura 5: Diagrama circuito FullAdder 4bits

Seguiu-se para a compilação do testbench criado, este chamado de tb\_fulladder, e que irá definir os testes do projeto. Ele está descrito abaixo:

***library IEEE;***

***use IEEE.STD\_LOGIC\_1164.all;***

***entity tb\_fulladder is***

***end tb\_fulladder;***

***architecture teste of tb\_fulladder is***

***component fulladder is***

***generic***

***(DATA\_WIDTH : natural :=4***

***);***

***port (***

***Cin : in std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***x : in std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***y : in std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***s : out std\_logic\_vector ((DATA\_WIDTH-1) downto 0);***

***Cout: out std\_logic\_vector ((DATA\_WIDTH-1) downto 0)***

***);***

***end component;***

***signal fio\_Cin: std\_logic\_vector (3 downto 0);***

***signal fio\_x: std\_logic\_vector (3 downto 0);***

***signal fio\_y: std\_logic\_vector (3 downto 0);***

***signal fio\_s, fio\_Cout: std\_logic\_vector (3 downto 0);***

***begin***

***instancia\_fulladder: fulladder generic map (DATA\_WIDTH => 4) port map(Cin=>fio\_Cin, x=>fio\_x, y=>fio\_y, s=>fio\_s,Cout=>fio\_Cout);***

***fio\_Cin <= x"0", x"2" after 30ns, x"3" after 70ns, x"7" after 100ns;***

***fio\_x <= x"0", x"5" after 50ns, x"3" after 90ns, x"7" after 110ns;***

***fio\_y <= x"0", x"4" after 40ns, x"5" after 100ns, x"7" after 150ns;***

***end teste;***

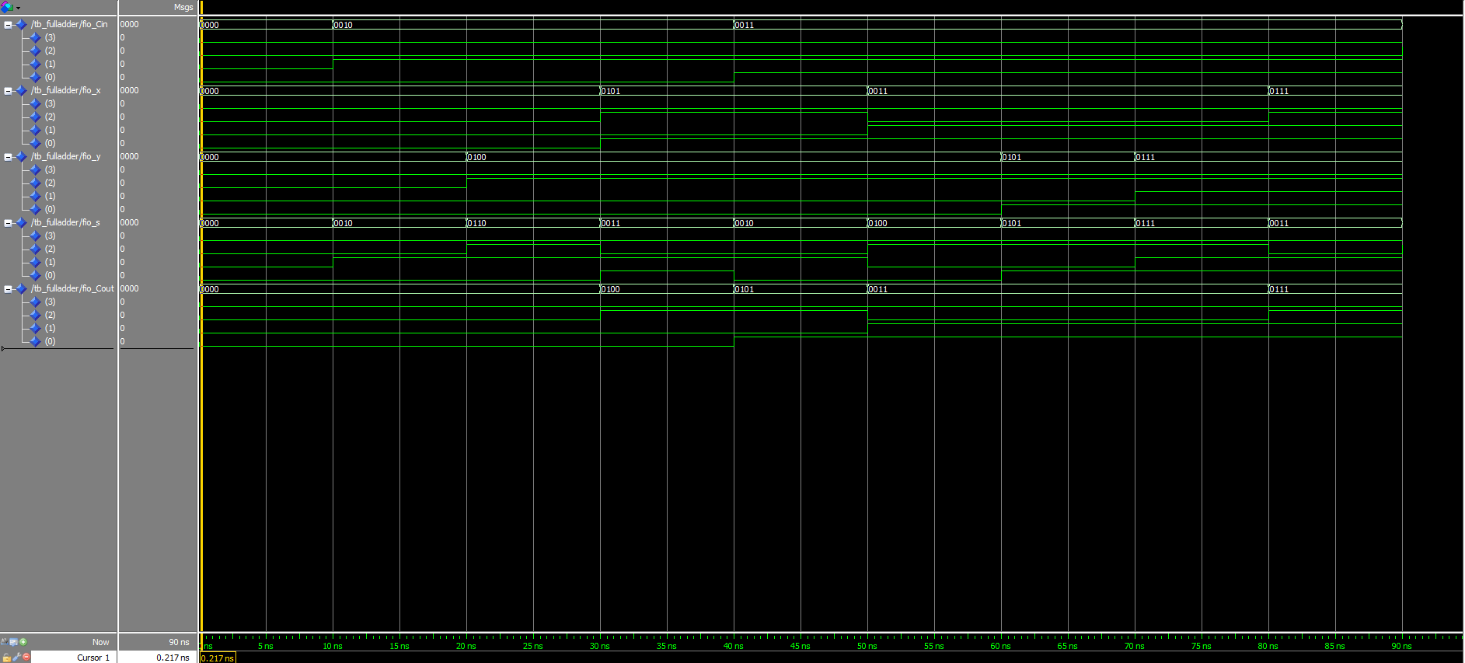
******Com ele, podemos simular o funcionamento do circuito através do software MULTISIM. Basta apenas indica o arquivo como testbench em simulation e começar a simulação em RTL. Irá abrir um gráfico de sinais com valores determinados pelo testbench e irá auxiliar na verificação da lógica.

Figura 6: Simulação FullAdder 4bits no Multisim

Foi utilizado o arquivo csv do comparador para facilitar a pinagem desse projeto.

***x[3] <=*** ***PIN\_N25***

***x[2] <= PIN\_N26***

***x[1] <= PIN\_P25***

***x[0] <= PIN\_AE14***

***y[3] <= PIN\_AC13***

***y[2] <= PIN\_C13***

***y[1] <= PIN\_B13***

***y[0] <= PIN\_A13***

***cin[3] <= PIN\_P2***

***cin[2] <= PIN\_T7***

***cin[1] <= PIN\_U3***

***cin[0] <= PIN\_U4***

***cout[3] <= PIN\_AE23***

***cout[2] <= PIN\_AF23***

***cout[1] <= PIN\_AB21***

***cout[0] <= PIN\_AC22***

***s[3] <= PIN\_AD21***

***s[2] <= PIN\_AC21***

***s[1] <= PIN\_AA14***

***s[0] <= PIN\_Y13***

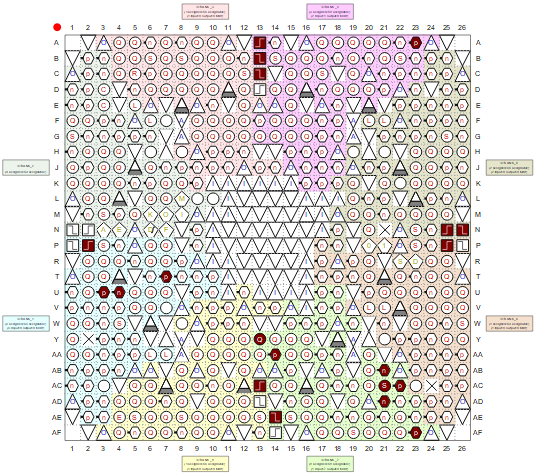


Figura 7: Pin Planner

Simulando novamente para verificação dos pinos, podemos enviar o projeto para o FPGA usando a função PROGRAMMER e o usb blaster.

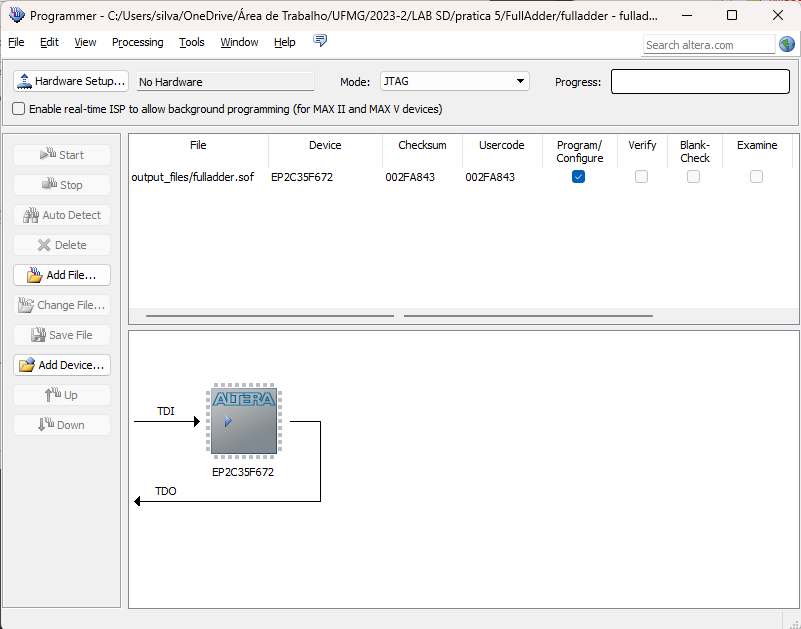


Figura 8: Programmer