# Relatório Prática 07 – 06/10/2023

# Arthur Souza/João Paulo – PN1

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

A partir disso, adicionou-se o código disponibilizado no sistema e implementado uma arquitetura comportamental para análise e compilação dos dados:

***library ieee;***

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

***entity pseudo\_mux is***

***port (***

***CLOCK : in std\_logic; -- clock input***

***RESET : in std\_logic; -- reset input***

***S : in std\_logic; -- control input***

***A,B,C,D : in std\_logic; -- data inputs***

***Q : out std\_logic -- data output***

***);***

***end pseudo\_mux;***

***architecture arch of pseudo\_mux is***

***type estado is (s0, s1, s2, s3);***

***signal est\_atual, est\_futuro : estado;***

***begin***

***process(CLOCK, RESET) is***

***begin***

***if(RESET= '1') then***

***est\_atual<=s0;***

***elsif(rising\_edge(CLOCK)) then***

***est\_atual<=est\_futuro;***

***end if;***

***end process;***

***process(S, est\_atual, A, B, C, D) is***

***begin***

***case est\_atual is***

***when s0 =>***

***Q<=A;***

***if(S='1') then***

***est\_futuro<=s1;***

***else***

***est\_futuro<=s0;***

***end if;***

***when s1 =>***

***Q<=B;***

***if(S='1') then***

***est\_futuro<=s2;***

***else***

***est\_futuro<=s1;***

***end if;***

***when s2 =>***

***Q<=C;***

***if(S='1') then***

***est\_futuro<=s3;***

***else***

***est\_futuro<=s2;***

***end if;***

***when s3 =>***

***Q<=D;***

***if(S='1') then***

***est\_futuro<=s0;***

***else***

***est\_futuro<=s3;***

***end if;***

***end case;***

***end process;***

***end arch;***

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:

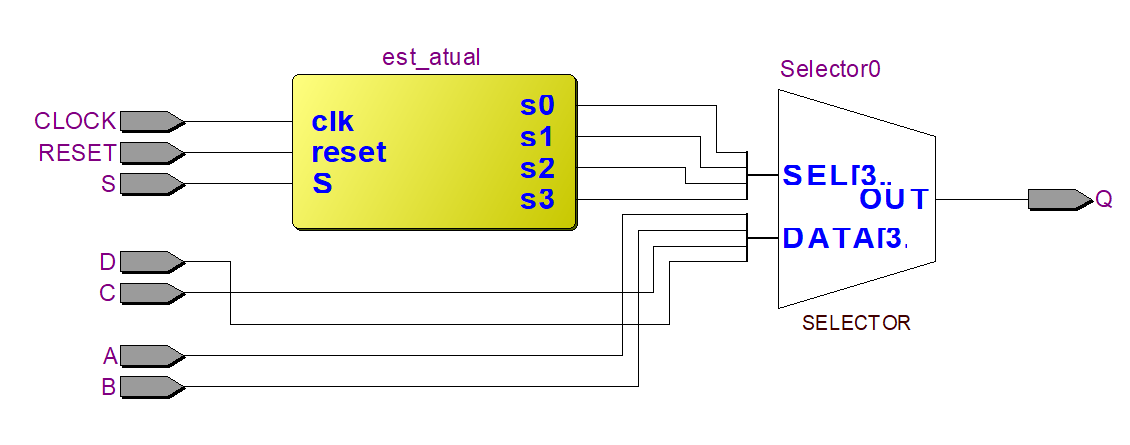


Figura 1: Circuito Pseudo Mux

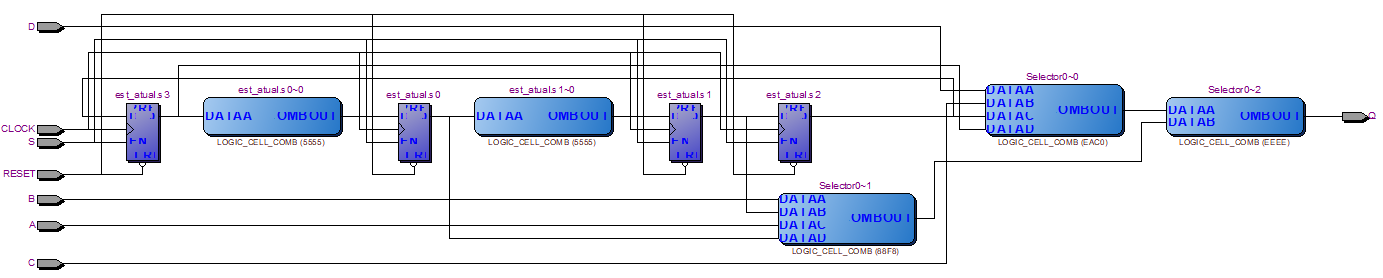


Figura 2: Diagrama do circuito Pseudo Mux

Seguiu-se para a compilação do testbench criado, este chamado de tb\_mean\_4\_clocks, e que irá definir os testes do projeto. Ele está descrito abaixo:

***library IEEE;***

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

***use ieee.numeric\_std.all;***

***entity tb\_pseudo\_mux is***

***end tb\_pseudo\_mux;***

***architecture teste of tb\_pseudo\_mux is***

***component pseudo\_mux is***

***port (***

***CLOCK : in std\_logic; -- clock input***

***RESET : in std\_logic; -- reset input***

***S : in std\_logic; -- control input***

***A,B,C,D : in std\_logic; -- data inputs***

***Q : out std\_logic -- data output***

***);***

***end component;***

***signal fio\_clk : std\_logic :='0';***

***signal fio\_res : std\_logic ;***

***signal fio\_s : std\_logic ;***

***signal fio\_a,fio\_b,fio\_c,fio\_d : std\_logic;***

***signal fio\_q : std\_logic :='0';***

***begin***

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

***instancia\_pseudo\_mux: pseudo\_mux port map(CLOCK=>fio\_clk, RESET=>fio\_res,A=>fio\_a,B=>fio\_b,C=>fio\_c,D=>fio\_d,S=>fio\_s,Q=>fio\_q);***

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

***fio\_clk <= not fio\_clk after 5ns;***

***fio\_res <= '1', '0' after 5ns;***

***fio\_s <= '0', '1' after 20ns, '0' after 30ns,'1' after 50ns, '0' after 60ns,'1' after 80ns, '0' after 90ns;***

***fio\_a <= '0', '1' after 10ns, '0' after 20ns;***

***fio\_b <= '0', '1' after 20ns, '0' after 50ns;***

***fio\_c <= '0', '1' after 50ns, '0' after 80ns;***

***fio\_d <= '0', '1' after 80ns, '0' after 90ns, '1' after 100ns, '0' after 110ns;***

***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.

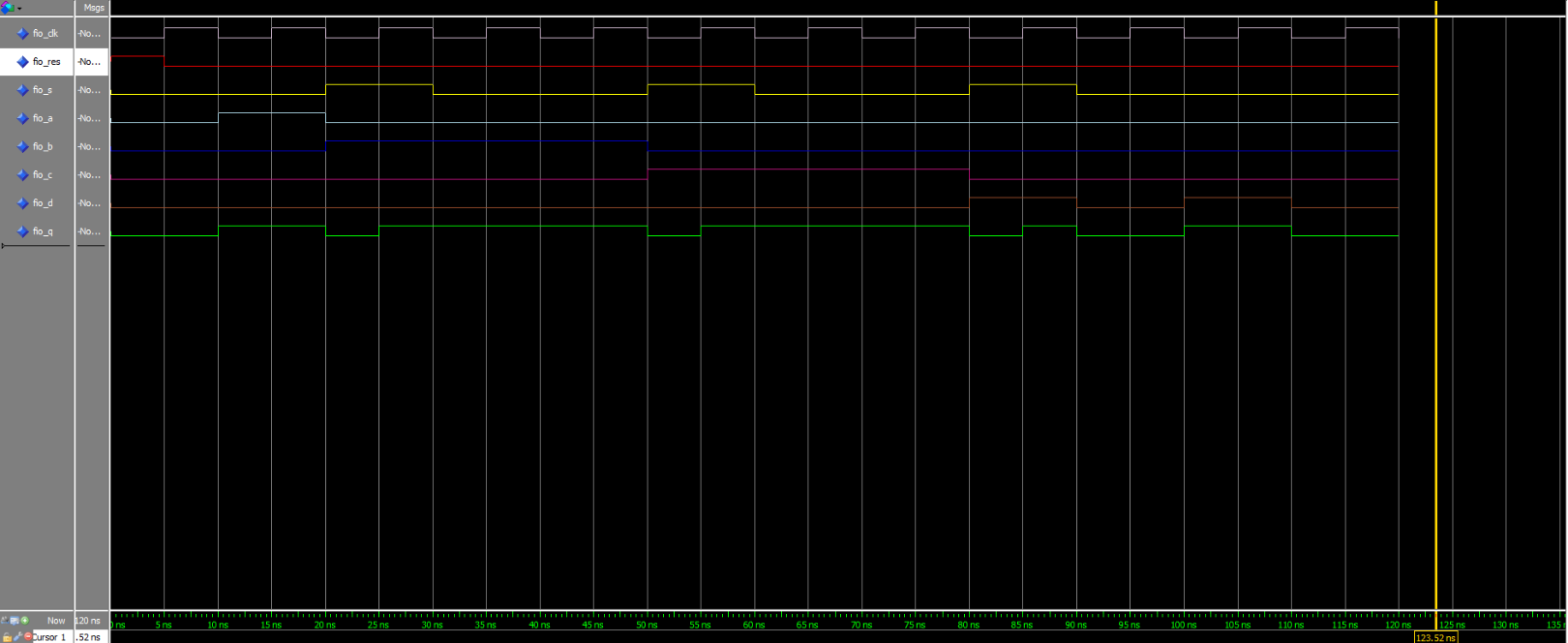


Figura 3: Simulação Pseudo Mux no Multisim

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

***A <=*** ***PIN\_N25***

***B <= PIN\_N26***

***C <= PIN\_P25***

***D <= PIN\_AE14***

***CLOCK <= PIN\_AC13***

***RESET <= PIN\_C13***

***S <= PIN\_B13***

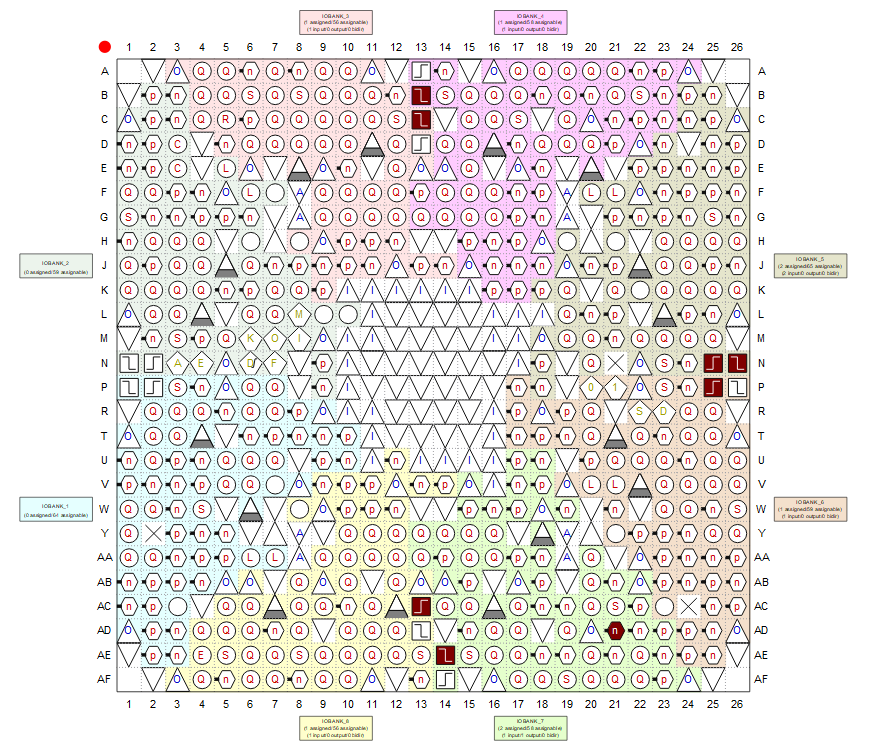


Figura 4: Pin Planner

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

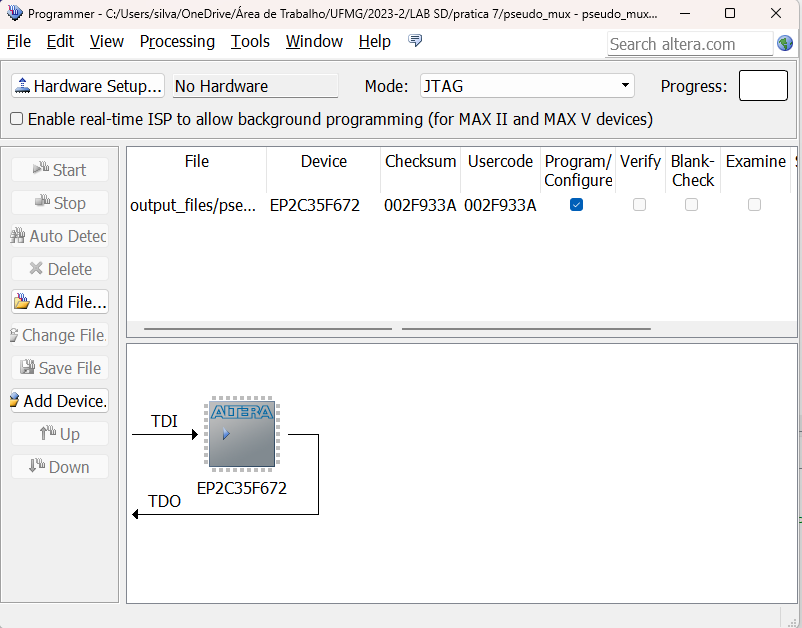


Figura 5: Programmer