Disciplina: PCS 3335 – Laboratório Digital A	
Prof.: *Kechi Hirama*	**Data:** *17/06*
Turma: *02*	**Bancada:** *01*
Membros:	
10336379 Henrique Ludovico Pretel	
11260791 Kim Amaral Luna	

Experiência *9*

Duelo de Jokempô

1. Introdução

Como conclusão da disciplina, foi proposta aos alunos a elaboração de um projeto desenvolvido pelos mesmos para o exercício dos conhecimentos e habilidades obtidos ao longo do semestre. Pensando nisso, decidimos nos basear em um projeto passado da disciplina de Sistemas Digitais I, desenvolvido em VHDL por um dos membros do grupo, e adaptá-lo para que seja possível implementá-lo com as ferramentas que temos agora, isto é, o painel de montagens e a FPGA, de forma a criar uma interface dinâmica e interativa para os usuários. Assim, definimos que faremos a construção e montagem de um jogo de jokempô, baseado no projeto 1 de Sistemas Digitais I do oferecimento de 2021: Jokempô Triplo, adaptando-o para que esse possa ser de fato jogado por dois jogadores reais. Do projeto original foi reaproveitado o código da entidade jokempotriplo desenvolvido por um dos integrantes do grupo (autor: Kim Amaral Luna), que sofreu poucas alterações para se remover a possibilidade de empate, e acrescentadas novas entidades que tratam as entradas das partidas transformando-as de paralelas para em série, e saídas convertidas em um placar de pontuação, adequando o projeto a um jogo que possa ser jogado de fato por 2 jogadores físicos de forma dinâmica e interativa.

2. Objetivo

O objetivo do projeto é construir um sistema que aceite a entrada das jogadas clássicas da brincadeira popular "jokenpo", “pedra”, “papel” ou “tesoura” de ambos os jogadores e contabilize as vitórias de cada jogador em um placar, sendo que mostre a pontuação de cada jogador, sendo cada ponto uma vitória, e também informe qual jogador ganhou a disputa obtendo a maior pontuação em 3 partidas (empates não contam como partida), de forma a tornar possível um “melhor-de-três”.

3. Planejamento

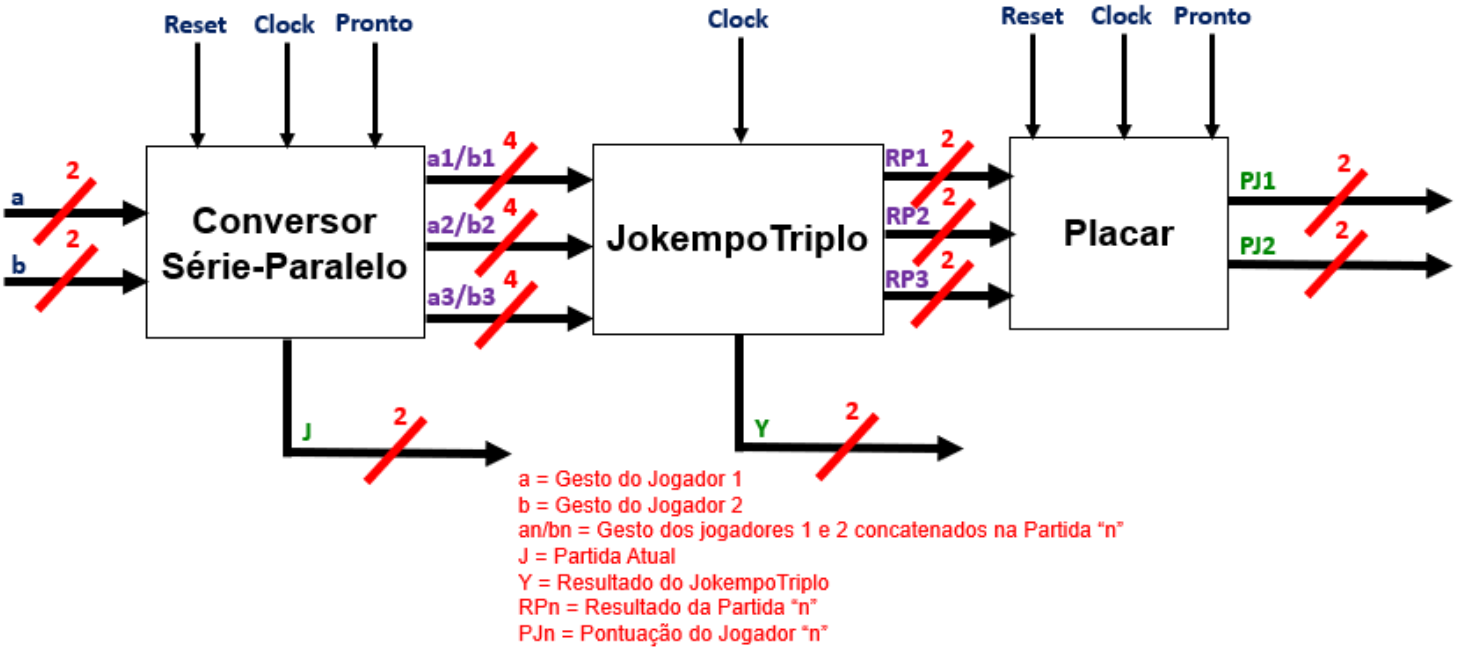
a) Descrição funcional

O circuito receberá duas entradas de dois bits cada, sendo elas o gesto de cada jogador, e armazenará, um por vez, os gestos enviados por ambos os jogadores em cada uma das 3 partidas. O gesto "00" corresponde a um estado de espera, ou seja, entrada padrão quando nenhum gesto foi computado, "01" equivalente a "Pedra", "10" a "Papel", e "11" a tesoura. Também receberá uma entrada "Clear" de 1 bit responsável por reiniciar o jogo, isto é fazer o resultado dos placares voltar a 0 e voltar a espera dos gestos da primeira partida, e uma entrada "Pronto" de 1 bit que atua como sinalizador de que os gestos do confronto em questão já foram escolhidos, nenhum deles está em estado espera, e que logo estão prontos para serem enviados, e para prosseguir para o próximo confronto.

O circuito deverá fornecer como saídas, de dois bits todas, a pontuação do jogador 1, e a pontuação do jogador 2, podendo ir ambas de 0 a 3, a partida em questão, indo ela de 0 a 3, sendo a partida 0 aquela após o reset, antes de quaisquer gestos serem enviados, e o campeão do jogo, isto é, aquele que atingiu os 2 pontos primeiro, sendo este "10" quando o jogador 1 ganha, "01" quando o jogador 2 ganha, e "11" enquanto nenhum dos dois tiver atingindo 2 pontos (indeciso);

b) Diagrama de blocos

Figura 1 – Diagrama de blocos do circuito.



c) Códigos VHDL dos Blocos

c.1) Bloco “JokempoTriplo”:

```
library IEEE;
use IEEE.numeric_bit.all;

entity jokempo is
port(
  a: in bit_vector(1 downto 0);                    --! gesto do jogador A
  b: in bit_vector(1 downto 0);                    --! gesto do jogador B
  y: out bit_vector(1 downto 0)                    --! resultado do jogo
);
end jokempo;

architecture jokempoArch of jokempo is
begin
  y <= "10" when ((a = "01" and b = "11") or (a = "10" and b = "01") or (a = "11" and b = "10")) else
            --! jogador A venceu
              "01" when ((b = "01" and a = "11") or (b = "10" and a = "01") or (b = "11" and a = "10")) else
            --! jogador B venceu
       "11" when (a = b) else                  --! Empate
       "00" when (a = "00" or b = "00");       --! Estado de espera
end jokempoArch;



entity melhordetres is
port(
resultado1 : in bit_vector(1 downto 0);          --! resultado do jogo 1
resultado2 : in bit_vector(1 downto 0);          --! resultado do jogo 2
resultado3 : in bit_vector(1 downto 0);          --! resultado do jogo 3
        z : out bit_vector(1 downto 0)                                  --! resultado da disputa
);
end melhordetres;

architecture melhordetresArch of melhordetres is
            signal vitorias_a: bit_vector(1 downto 0);
   signal vitorias_b: bit_vector(1 downto 0);
begin

                        vitorias_a <= "01" when (resultado1 = "10" and resultado2 /="10" and resultado3 /="10" ) or
                                                                                                (resultado2   =
"10" and resultado1 /="10" and resultado3 /="10" ) or
                                                                                                (resultado3   =
"10" and resultado1 /="10" and resultado2 /="10" ) else
                                                                        "10" when (resultado1 = "10" and resultado2
="10" and resultado3 /="10" ) or
                                                                                                (resultado2   =
"10" and resultado1 /="10" and resultado3 ="10" ) or
                                                                                                (resultado3   =
"10" and resultado1 ="10" and resultado2 /="10" ) else
                                                                        "11" when (resultado1 = "10" and resultado2
="10" and resultado3 ="10" )  else
                                                                        "00";

                        vitorias_b <=  "01" when (resultado1 = "01" and resultado2 /="01" and resultado3 /="01" ) or
                                                                                                (resultado2   =
"01" and resultado1 /="01" and resultado3 /="01" ) or
                                                                                                (resultado3   =
"01" and resultado1 /="01" and resultado2 /="01" ) else
                                                                        "10" when (resultado1 = "01" and resultado2
="01" and resultado3 /="01" ) or
                                                                                                (resultado2   =
"01" and resultado1 /="01" and resultado3 ="01" ) or
                                                                                                (resultado3   =
"01" and resultado1 ="01" and resultado2 /="01" ) else
```

```
                                                        "11" when (resultado1 = "01" and resultado2
="01" and resultado3 ="01" )  else
                                                        "00";

                        z <= "10" when vitorias_a > "01" else
                                        "01" when vitorias_b > "01" else
                                        "11";
end melhordetresArch;


entity jokempotriplo is
port(
            a1, a2, a3 : in bit_vector(1 downto 0);
   b1, b2, b3 : in bit_vector(1 downto 0);
                clock : in bit;
            g1, g2, g3 : out bit_vector(1 downto 0);
          z : out bit_vector(1 downto 0)
    );
end jokempotriplo;

architecture jokempotriploArch of jokempotriplo is

component jokempo is
port(
  a: in bit_vector(1 downto 0);
  b: in bit_vector(1 downto 0);
  y: out bit_vector(1 downto 0)
);
end component;

component melhordetres is
port(
resultado1: in bit_vector(1 downto 0);          --! resultado do jogo 1
resultado2: in bit_vector(1 downto 0);          --! resultado do jogo 2
resultado3: in bit_vector(1 downto 0);          --! resultado do jogo 3
z: out bit_vector(1 downto 0)                           --! resultado da disputa
);
end component;

signal RJ1, RJ2, RJ3: bit_vector(1 downto 0);

begin
            xRJ1 : jokempo port map (a1, b1, RJ1);
   xRJ2 : jokempo port map (a2, b2, RJ2);
   xRJ3 : jokempo port map (a3, b3, RJ3);
            g1 <= RJ1;
            g2 <= RJ2;
            g3 <= RJ3;
   xZ : melhordetres port map(RJ1, RJ2, RJ3, z);

end jokempotriploArch;
```

c.2) Bloco “Conversor Série-Paralelo”:

```
library IEEE;
use IEEE.numeric_bit.all;

entity conversor is
        port(           a, b : in bit_vector(1 downto 0);
                reset, pronto : in bit;
                        clock : in bit;
j, a1, a2, a3, b1, b2, b3 : out bit_vector(1 downto 0));
end conversor;

architecture conversor_arch of conversor is
        signal j1, j2, j3 : bit;
        signal sustain : bit;
begin
        process(a, b, reset, pronto)
        begin
                if reset = '1' then
                        j1 <= '0';
                        j2 <= '0';
                        j3 <= '0';
                        a1 <= "00";
                        a2 <= "00";
                        a3 <= "00";
                        b1 <= "00";
                        b2 <= "00";
                        b3 <= "00";
                        sustain <= '0';
                elsif pronto'event and pronto = '1' then
                        if j1 = '0' then
                                if (a /= "00" and b /= "00") and (a /= b) then
                                        a1 <= a;
                                        b1 <= b;
                                        j1 <= '1';
                                end if;
                        elsif j2 = '0' then
                                if (a /= "00" and b /= "00") and (a /= b) then
                                        a2 <= a;
                                        b2 <= b;
                                        j2 <= '1';
                                end if;
                        elsif j3 = '0' then
                                if (a /= "00" and b /= "00") and (a /= b) then
                                        a3 <= a;
                                        b3 <= b;
                                        j3 <= '1';
                                end if;
                        end if;
                end if;
                if j3 = '1' then j <= "11";
                elsif j2 = '1' then j <= "10";
                elsif j1 = '1' then j <= "01";
                else j <= "00";
                end if;
        end process;
end conversor_arch;
```

c.3) Bloco "Pontuação/Placar"

```
library IEEE;
use IEEE.numeric_bit.all;


entity pontuacao is --! máquina de estados para registrar a pountuação da partida
        port(jogo1, jogo2, jogo3 : in bit_vector(1 downto 0);
                pronto : in bit;
              clock, reset : in bit;
      placar1, placar2 : out bit_vector(1 downto 0));
end entity;

architecture pontuacao_arch of pontuacao is
begin
        process(jogo1, jogo2, jogo3, pronto)
        begin
                if rising_edge(clock) then
                        if reset = '1' then
                                placar1 <= "00";
                                placar2 <= "00";
                        elsif pronto = '1' then
                                if jogo3 = "10" then       --! três jogos
                                        if jogo2 = "10" then
                                                if jogo1 = "10" then  --! 1, 1, 1
                                                        placar1 <= "11";
                                                        placar2 <= "00";
                                                else                                    --! 2, 1, 1
                                                        placar1 <= "10";
                                                        placar2 <= "01";
                                                end if;
                                        else
                                                if jogo1 = "10" then  --! 1, 2, 1
                                                        placar1 <= "10";
                                                        placar2 <= "01";
                                                else              --! 2, 2, 1
                                                        placar1 <= "01";
                                                        placar2 <= "10";
                                                end if;
                                        end if;
                                elsif jogo3 = "01" then
                                        if jogo2 = "10" then
                                                if jogo1 = "10" then  --! 1, 1, 2
                                                        placar1 <= "10";
                                                        placar2 <= "01";
                                                else                                    --! 2, 1, 2
                                                        placar1 <= "01";
                                                        placar2 <= "10";
                                                end if;
                                        else
                                                if jogo1 = "10" then  --! 1, 2, 2
                                                        placar1 <= "01";
                                                        placar2 <= "10";
                                                else              --! 2, 2, 2
                                                        placar1 <= "00";
                                                        placar2 <= "11";
                                                end if;
                                        end if;
                                else
                                        if jogo2 = "10" then     --! dois jogos
                                                if jogo1 = "10" then  --! 1, 1, 0
                                                        placar1 <= "10";
                                                        placar2 <= "00";
                                                else                                    --! 2, 1, 0
                                                        placar1 <= "01";
                                                        placar2 <= "01";
```

```
                        end if;
                    elsif jogo2 = "01" then
                        if jogo1 = "10" then  --! 1, 2, 0
                            placar1 <= "01";
                            placar2 <= "01";
                        else                --! 2, 2, 0
                            placar1 <= "00";
                            placar2 <= "10";
                        end if;
                    else                --! um jogo
                        if jogo1 = "10" then  --! 1, 0, 0
                            placar1 <= "01";
                            placar2 <= "00";
                        elsif jogo1 = "01" then --! 2, 0, 0
                            placar1 <= "00";
                            placar2 <= "01";
                        else                --! 0, 0, 0
                            placar1 <= "00";
                            placar2 <= "00";
                        end if;
                    end if;
                end if;
            end if;
        end if;
    end process;
end pontuacao_arch;
```

c.4) Circuito completo:

```
library IEEE;
use IEEE.numeric_bit.all;

entity projeto is
        port(   a, b : in bit_vector(1 downto 0);
        reset, pronto : in bit;
                clock : in bit;
            pj1, pj2 : out bit_vector(1 downto 0);
                        j, y : out bit_vector(1 downto 0));
end projeto;

architecture projeto_arch of projeto is
        component conversor is
                port(       a, b : in bit_vector(1 downto 0);
             reset, pronto : in bit;
             clock : in bit;
j, a1, a2, a3, b1, b2, b3 : out bit_vector(1 downto 0));
        end component;
        component pontuacao is
                port(jogo1, jogo2, jogo3 : in bit_vector(1 downto 0);
                        pronto : in bit;
                clock, reset : in bit;
          placar1, placar2 : out bit_vector(1 downto 0));
        end component;
        component jokempotriplo is
                port(a1, a2, a3 : in bit_vector(1 downto 0);
      b1, b2, b3 : in bit_vector(1 downto 0);
                clock : in bit;
            g1, g2, g3 : out bit_vector(1 downto 0);
          z : out bit_vector(1 downto 0));
        end component;
        signal a1, a2, a3, b1, b2, b3, g1, g2, g3, z          : bit_vector(1 downto 0);
        signal placar1, placar2 : bit_vector(1 downto 0);
begin
        entrada : conversor port map(a, b, reset, pronto, clock, j, a1, a2, a3, b1, b2, b3);
        jogos : jokempotriplo port map(a1, a2, a3, b1, b2, b3, clock, g1, g2, g3, y);
        placar : pontuacao port map(g1, g2, g3, pronto, clock, reset, pj1, pj2);
end projeto_arch;
```

d) Diagramas RTL Viewer

Figura 2 – Diagrama RTL Viewer do circuito "JokempoTriplo".

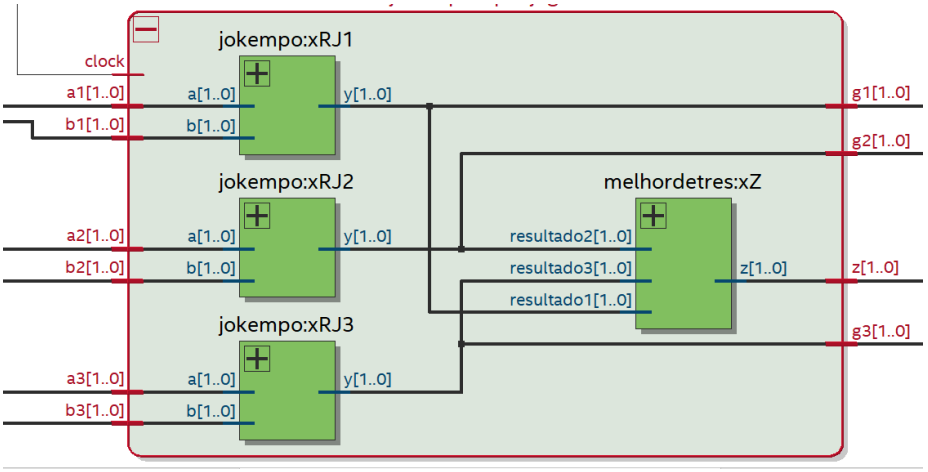


Figura 3 – Diagrama RTL Viewer do circuito “Jokempo” do "JokempoTriplo".

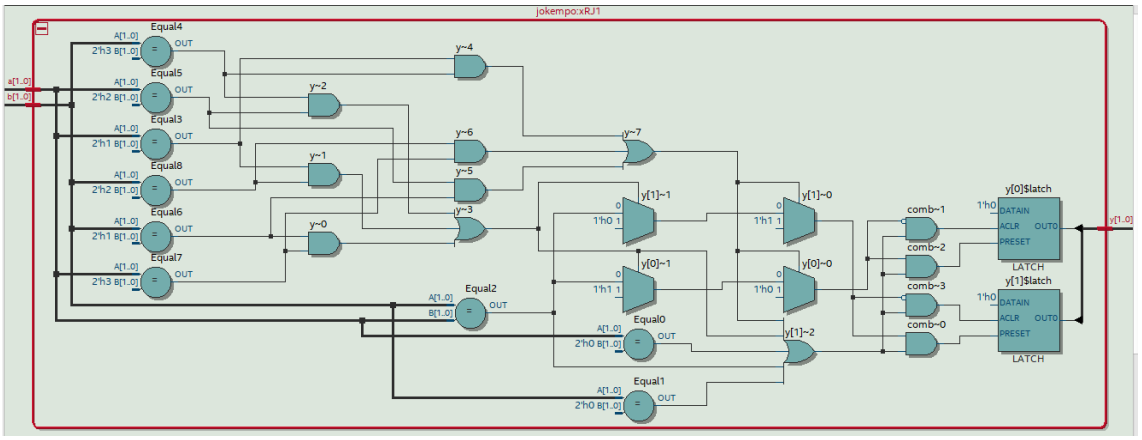


Figura 4 – Diagrama RTL Viewer do circuito “melhor de 3” do "JokempoTriplo".

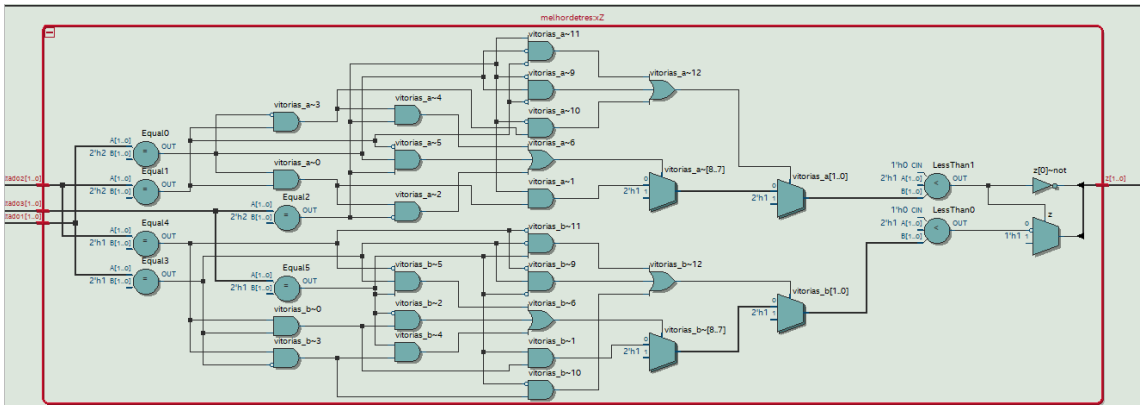


Figura 5 – Diagrama RTL Viewer do circuito "Conversor".

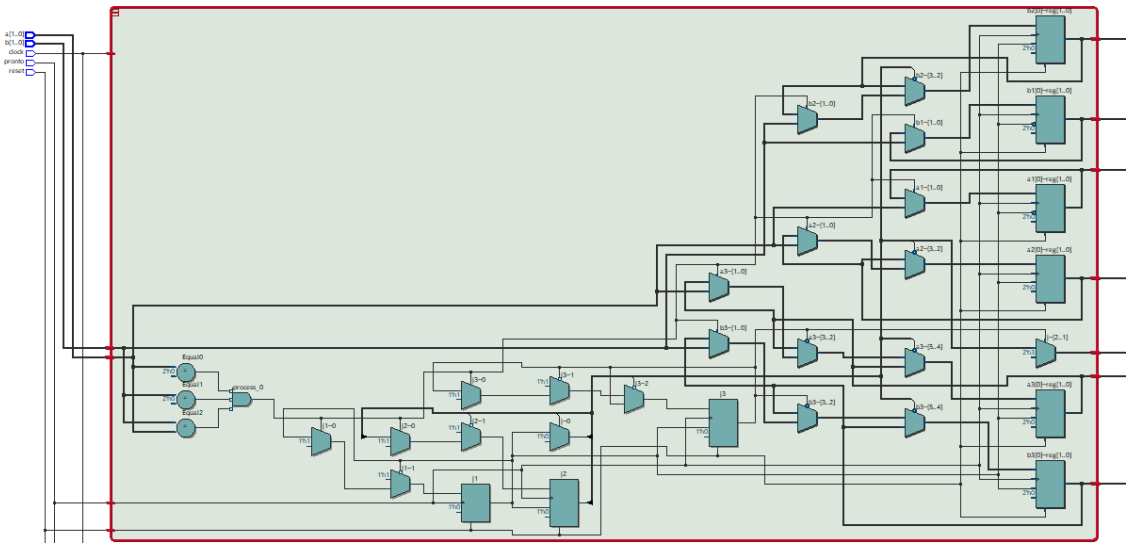
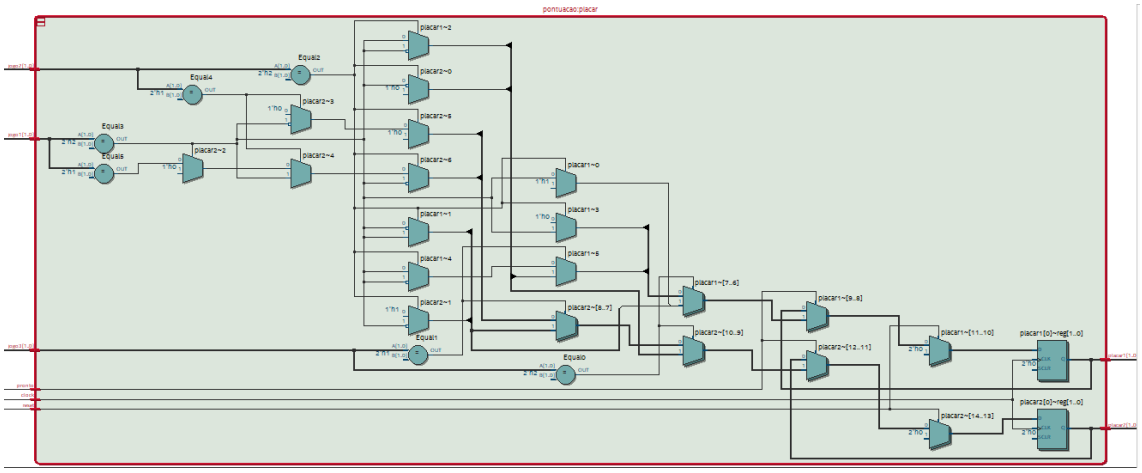
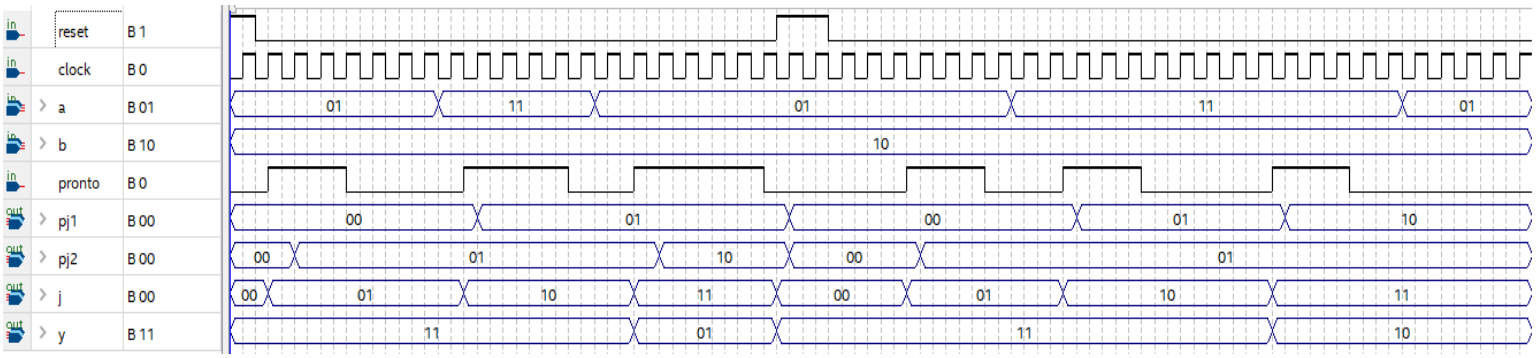


Figura 6 – Diagrama RTL Viewer do circuito "Pontuação".



e) Simulações

Figura 7 – Carta de tempos do circuito.



f) Tabela de testes

Tabela de Testes - Duelo de Jokenpô											
Entradas				Saídas Esperadas/Obtidas							
Clear	Pronto	jogador1	jogador2	Partida		Pontuação1		Pontuação2		Campeão	
1	0	X	X	00	00	0	0	0	0	11	11
0	1	01	11	01	01	1	1	0	0	11	11
0	1	10	10	01	01	1	1	0	0	11	11
0	1	11	10	10	10	2	2	0	0	10	10
0	1	10	11	11	11	2	2	1	1	10	10
1	0	X	X	00	00	0	0	0	0	11	11
0	1	11	01	01	01	0	0	1	1	11	11
0	1	00	10	01	01	0	0	1	1	11	11
0	1	10	11	10	10	0	0	2	2	01	01
0	1	01	10	11	11	0	0	3	3	01	01

4. Relatório

a) Resultados obtidos

O circuito funcionou majoritariamente conforme o planejado, entretanto foram necessárias alterações para que o empate não fosse computado e para que a saída do "campeão" funcionasse devidamente. Além disso, o projeto deveria ter sido adaptado a um modelo de Fluxo de Dados (MSI) e Unidade de Controle (VHDL), e foi feito sem diferenciação dos dois, também foi feito integralmente em VHDL, sem uso de unidades MSI.

Referências

1. Apostilas e documentos fornecidos para a experiência.
2. Apostilas disponíveis na plataforma e-Disciplinas.