# Relatório - Marcador de Tênis

Universidade de Aveiro

Guilherme Santos (107961), João Gaspar (107708)

Laboratórios de Sistemas Digitais

# Conteúdo



## Introdução

O objetivo deste projeto é programar uma FPGA de forma a que seja possível fazer a contagem de marcação de pontos em tênis. O sistema faz a contagem de pontos/points (15, 30, 40), jogos/games, e conjuntos de jogos/sets. O utilizador deve informar qual jogador marcou ponto até a máquina avaliar um vencedor. O vencedor final é aquele que ganha o número pré-definido de sets, por norma 3.

## Manual do Utilizador

O utilizador deverá interagir com a FPGA através de três botões, KEY3 para marcar ponto/point ao jogador A, KEY0 para marcar ponto ao jogador B e KEY1 que deverá reiniciar todo o sistema após um tempo de atuação de 3 segundos. Os seis displays hexadecimais deverão mostrar o número de pontos/points no game em curso do jogador A (HEX7-HEX6) e o número de pontos/points do jogador B (HEX5-HEX4), o número de games no set em curso de A (HEX1) e de B (HEX0). Por fim os LEDG's de 6 a 4 e de 2 a 0 monstram o número de sets respetivamente do jogador A e B.

## Arquitetura

O diagrama de blocos do sistema, por se tornar extenso, encontra-se no anexo (ficheiro "DiagramadeBlocos.png"). Seguidamente iremos explicar passo a passo a função de cada entidade descrita nesse diagrama.

O sistema tem como entradas três botões (no diagrama KEY[3..0] que se divide em KEY(3), KEY(1) e KEY(0)) e um relógio de frequência 50 (CLOCK_50). Ligado a cada botão está um debouncer, para evitar possíveis transições indesejadas.

A saída do s_debounceunit_reset, ligado ao botão KEY(1), serve para fazer reset, ou seja para reinicializar o sistema. Os dois botões restantes (KEY(3) e KEY(0)) servem para incrementar pontos ao jogador A e B.

Analisando o diagrama podemos ver a entidade s_normal_fsm, destinada a gerir a contagem de pontos/points no decorrer do game. As saídas displayA[13..0] e displayB[13..0] desta máquina de estados estão ligadas aos displays hexadecimais e as saídas pointA e pointB (que se referem ao caso de A marcar ou B marcar respetivamente) estão conectadas a duas entidades diferentes porém que funcionam de maneira igual.

As entidades s_set_fsm_a e s_set_fsm_b irão receber um input cada, caso A ou B ganhem um game. Quando o número de games de qualquer jogador alcança o limite (7), a entidade do jogador respetivo ativa a saída pointOut que se liga a s_debounceunit_aS ou s_debounceunit_bS e depois s_match_fsm.

Esta entidade (s_match_fsm), irá receber os números de sets e mostrá-los nos LEDG's. O jogador que ganhar terá as suas LEDG's todas ligadas e o perdedor desligadas.

## Implementação

Nesta secção, iremos explicar o que faz cada entidade e como, entidade a entidade.

Optámos por usar o Deboucer fornecido pelos professores. Esta entidade tem o objetivo de tornar o sinal vindo dos botões num sinal mais "limpo".

A entidade NormalGameFSM é uma máquina de estados que só funciona quando apresenta a entrada enable ligada e começa no estado A0B0, com os displays ligados às constantes zeroP e às saídas pointA e ponitB com o valor '0'. Caso seja pressionado KEY3, o estado seguinte será A1B0 caso contrário o estado seguinte será A0B1 e assim sucessivamente para os outros estados. Se KEY3 for pressionado durante os estados A3B0, A3B1 e A3B2, ele irá passar ao estado PA, onde irá mudar o valor da saída pointA para '1' e consecutivamente volta para o estado A0B0. Se o KEY0 for pressionado durante os estados A0B3, A1B3 e A2B3, ele irá passar ao estado PB, onde irá mudar o valor da saída pointB para '1' voltando para o estado A0B0. Caso a máquina chegue ao estado A3B3 e algum jogador marque, a máquina entra nos estados de deuce e, apartir de deuce deve assinalar advantage para o jogador que conquistar o próximo ponto. A partir de advantage o jogador que marcar ganha '1' point e a máquina reinicia.

O Set funciona principalmente por duas entidades SetFSM, cada entidade é ligada a cada saída da entidade NormalGameFSM (pointA e pointB). A entidade SetFSM é uma máquina de estados auxiliada com um contador. Esta máquina tem como entradas um clk e um point. A entrada "point" é acionada se o pointA ou o pointB (pointX, em que X poderá representar A ou B) do NormalGameFSM estiver ligada. Esta máquina começa no estado G0 e sempre que a entrada pointX é ligada, passará para o estado seguinte que, neste caso será G1. Quando chegar a G6 e caso o s_cntValue seja 7, a máquina irá reiniciar e começar denovo pelo G0 e sempre assim consecutivamente. Quando as saídas seisD dos dois SetFSM (s_set_fsm_a e s_set_fsm_b) estiverem ligadas, elas desligam o enable presente no NormalGameFSM e ativam o enable do TieBreak que, como não está implementado apenas "trava" o resultado em 6-6. Quando uma das saídas seteD dos dois SetFSM ligar, irá provocar um reset nestas duas entidades, voltado todo ao estado G0.

O botão KEY1 (reset) está ligado a um Debouncer (s_debounceunit_reset),

em conjunto com um clock, com a variável mSecMinInWidth apresentando o valor 3000, assim, a saída pulsedOut só será ativada se o utilizador pressionar o botão KEY1 durante 3 segundos. A saída pulsedOut está ligada a todas as outras entidades presentes no circuito.

A entidade TieBreak é uma máquina de estados que apresenta o mesmo funcionamento do NormalGame, com a única diferença sendo o aumento do limite de pontos necessários para completar um game. Esta máquina só seria ligada caso as duas entidades SetFSM estivessem no estado G5.

A entidade MatchFSM também é uma máquina de estados tendo como entradas aIn, bIn e um clock. Esta entidade irá começar no estado A0B0, se aIn for ligado, a máquina passa para o estado A1B0, se bIn for ligado, a máquina passa para o estado A0B1. Caso aIn seja ligado nos estados A2B0, A2B1 e A2B2 a máquina passa para o estado GA e o jogador A ganha o jogo. Caso bIn seja ligado nos estados A0B2, A1B2 e A2B2, a máquina passa para o estado GB e o jogador B ganha o jogo. Cada saída, displayMatchesA e displayMatchesB é apresentado um código binário de 3 bits, que representam os LED's verdes da FPGA, podendo haver apenas três possibilidades: 001 (1 LED ligado), 011 (2 LED's ligados) e 111 (3 LED's ligados).

A entidade ServiceFSM é uma máquina de estados que define aleatóriamente quando jogador efetua o serviço primeiro. Sempre que um dos dois marca um ponto, o serviço passará automáticamente para o outro jogador.

## Validação

Ao longo do projeto, realizámos Testbenches às entidades que considerámos válidas testar, devido à sua complexidade ou a possíveis erros de interligação com outras entidades. Todas as Testbenches acabaram por ser válidas, e por estar de acordo com o esperado.

Começámos pela Testbench aplicada à máquina de estados NormalGameFSM, para simular o seu comportamento quando os seus sinais de entrada variavam. Depois, aplicámos outra Testbench à entidade SetFSM, para garantirmos a sua sanidade e a validade da contagem e uma também à entidade MatchFSM.

## Conclusão

O projeto realizado não cumpriu com todas as especificações pretendidas. Sentimos que foi um projeto que nos fez aprender bastante sobre sistemas digitais e programação para FPGA's. Escolhemos este projeto porque queríamos um desafio, queríamos construir algo que nos divertisse, que tivesse alguma funcionalidade interessante. O nosso objetivo foi completamente alcançado.

A maior dificuldade sentida foi na compreensão das regras do jogo visto que encontrámos várias versões online. Despendemos bastante tempo, primeiro a planear o que tínhamos de construir e depois na sua programação.

Consideramos que o trabalho foi bem distribuído entre nós. A comunicação foi algo indispensável neste projeto, pois assim conseguimos ajudarmo-nos mutuamente nos problemas que eventualmente nos iria aparecendo. Posto isto, entendemos que ambos realizaram 50% do projeto.

Tal como referido anteriormente, o trabalho não foi concluído na sua totalidade. Foi feito de forma faseada, tendo conseguido terminar as fases 1, 3 e 4. Também encontrámos algumas dificuldades na interligação dos blocos. Com isto e reconhecendo o tempo perdido, sentimo-nos merecedores de 12 valores.