Introdução

O nosso projeto baseia-se num registador de pontuações de um jogo de ténis.

A máquina é capaz de detetar as situações de *deuce* (quando ambos os jogadores possuem 40 pontos), *tie-break* (quando ambos os jogadores ganharam 6 *games*), *set point* (momento em que um jogador precisa de ganhar uma jogada para ganhar um set) e *match point* (momento em que um jogador precisa de ganhar uma jogada para ganhar a partida), logo o utilizador não precisa de se preocupar com estas situações, basta-lhe clicar nos botões definidos para adicionar pontos.

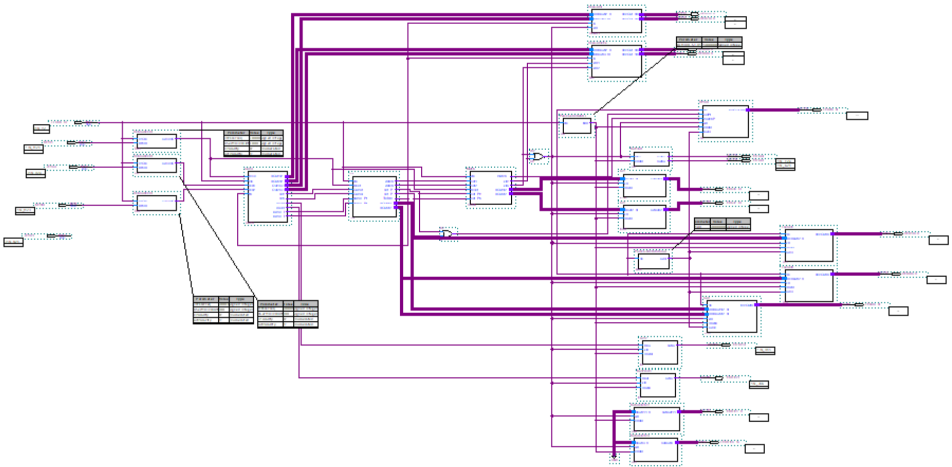
É possível observar nos "LEDG's" o registo dos *sets* ganhos por cada jogador e nos "HEX's" é mostrado a pontuação de cada jogador durante um *game,* assim como o número de *games* jogados. Quando um dos jogadores ganhar a *match*, o utilizador observará um efeito especial que realça o jogador vencedor.

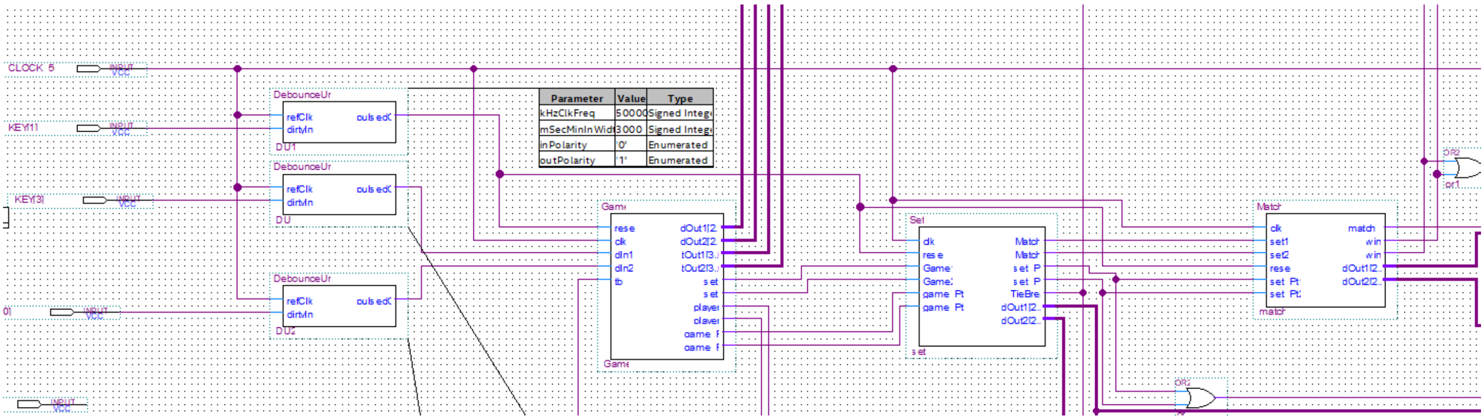
Arquitetura

A arquitetura do nosso marcador de ténis baseia-se na utilização de duas máquinas de estados. A primeira e mais utilizada é a máquina de estados que regista os pontos marcados por cada jogador num game. Ela é capaz de detetar de forma autónoma situações de *deuce* e *advantage*, assim como perceber se um jogador marcou pontos suficientes para vencer o game e registar quando um jogador está prestes a ganhar o mesmo, pelo que lhe demos o nome de "Game". A máquina de estados seguinte (de nome "Set") é responsável por calcular os sets. Ela recebe o sinal da máquina anterior e procede a registá-los de modo que possa verificar se possui games suficientes para ganhar o set. Também é responsável por detetar situações de tie-break e de *set point*. O último bloco fundamental (cujo nome é "Match") é capaz de constatar se o jogador está prestes a ganhar a partida, determinar se um jogador possui os sets suficientes para poder ser declarado vencedor e enviar um sinal que ativa o efeito especial que salienta o vencedor da mesma.

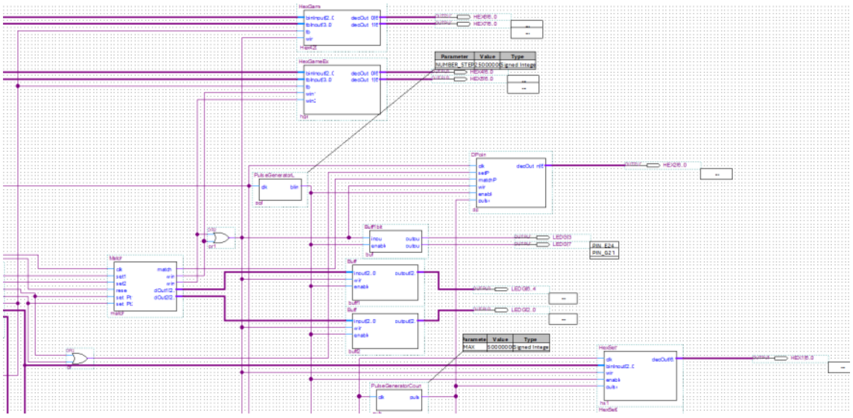
Os blocos "HexGame" e "HexSetExp" são responsáveis por exibir nos "HEX's" os pontos de cada jogador durante um *game* e os games ganhos por cada jogador durante um *set,* respetivamente.

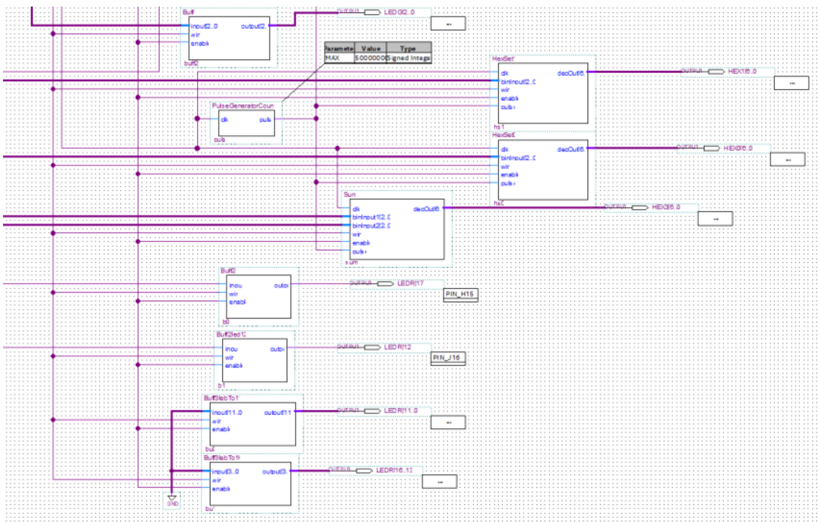
Os blocos "Sum" e "DPoint" apenas transmitem o número total de *sets* jogados e uma mensagem que mostra ao utilizador se o jogo se encontra em *set point* ou *match point.*





CLOCK_5
KEY[1]
KEY[3]
INPUT
VCC
DebounceUr
refClk
dirtyIn
pulsedC
DU1
DU
DU2
Parameter | Value | Type
kHzClkFreq | 50000 | Signed Intege
mSecMinInWidt | 3000 | Signed Intege
inPolarity | '0' | Enumerated
outPolarity | '1' | Enumerated
Game
rese
clk
dIn1
dIn2
tb
dOut1[2
dOut2[2
tOut1[3..
tOut2[3..
set
player
game
Set
Game
game Pt
Match
set P
TieBre
dOut1[2..
dOut2[2..
Match
set1
set2
set Pt
match
win
dOut1[2..
dOut2[2..
OR2
or1





Implementação

Tendo em conta as máquinas de estado criadas anteriormente, precisamos de implementá-las de modo a colocar o circuito a funcionar da maneira desejada. Quanto á máquina "Game", ela tem 5 entradas e 6 saídas, em que as entradas correspondem ao registo do ponto ganho pelo jogador 1 ou pelo jogador 2, um *reset*, um *clock* e ao estado de *tie-break* que provém da saída "TieBreak" do bloco "Set". As saídas correspondem ao registo do *game point* (conectado às entradas "game_Pt1" e "game_Pt2" do bloco "Set") e do jogador que ganhou o set e os pontos que cada jogador possui durante o *game* (que será apresentado nos "HEX's").

O "Set" receberá do bloco "*Game"* o registo do jogador que ganhou o *set* e o estado de *game point,* possui um *clock* e um *reset*. Após realizar as diversas operações, passará para o próximo bloco o registo do jogador que ganhou o set e o registo do *set point*. Passará para um bloco diferente, o registo dos sets que cada jogador ganhou até ao momento (que apresentará nos "HEX's" 0 e 1) e uma saída que se ligará ao bloco "HexSetExp", o qual será responsável por apresentar o estado de *set point* ou de *match point* no "HEX2". Este último bloco é ligado a um "PulseGenerator" para ser possível mostrar, por exemplo, um "S" a "piscar" no "HEX2". Para ser possível colocar o HEX a piscar a uma frequência de 1Hz, foi preciso mudar a componente genérica do "PulseGeneratorLed" para 50_000_000 pois a frequência do "clk" era de 50MHz.

"Match" receberá o registo do *set point, do* jogador que ganhou o *set* e possui um *reset* e um *clock*. De seguida, realizará as poucas operações que possui e, através de duas saídas ligará o número de "LEDG'S" correspondente ao número de *sets* ganhos pelo respetivo jogador.

O efeito especial é desencadeado por um sinal enviado pela "Match" que através de um multiplexer, altera o comportamento dos blocos de "Hex" criados anteriormente para mostrar o jogador vencedor. Também foi utilizado "PulseGenerators" para ligar e desligar os "LED's" da FPGA.

O bloco "Sum" terá como entrada as saídas "dOut1" e "dOut2", que correspondem aos pontos de cada jogador, e este mostrará no HEX3 o valor da soma destes dois valores.

O bloco "DPoint" terá como entrada os valores de "set_Pt1" e "set_Pt2" provenientes do "Set" e mostra no HEX2 um "S", caso se verifica a existência de set point, e um símbolo que substitui o "m", pois não é possível representá-lo no visor.

Validação

Para testar os módulos principais (“Game”, “Set” e “Match”), utilizámos *testbenches,* nas quais utilizamos código adequado para verificar o comportamento das máquinas de estado nas diversas situações possíveis. Por exemplo, verificar se o código do bloco “Game” seguia para o estado de *deuce* e de *advantage* nos momentos corretos ou se o “Set” verificava a situação de *tie break*. Utilizamos também a própria FPGA pois é nesta que o nosso projeto incidirá. Deste modo, verificamos, por exemplo, se a máquina recebia o *reset* ou os pontos corretamente.

Manual do utilizador

- KEY3 – adicionar ponto ao jogador A;
- KEY1 – *reset*;
- KEY0 – adicionar ponto ao jogador B;
- HEX7 e HEX6 – mostra os pontos do jogador A;
- HEX5 e HEX4 – mostra os pontos do jogador B;
- HEX3 – mostra o número de *games*
- HEX2 – mostra a situação de *match point* ou de *set point;*
- HEX1 – mostra os “games” ganhos pelo jogador A;
- HEX0 – mostra os “games” ganhos pelo jogador B;
- LEDG[6..4] - mostra os *sets* ganhos pelo jogador A;
- LEDG[2..0] - mostra os *sets* ganhos pelo jogador B;

Conclusão

Este projeto exigiu uma quantidade de tempo maior do que esperávamos inicialmente. Houve vários problemas que tivemos de resolver que nos levaram várias horas a entender o que os causavam, pois muitas vezes resultavam das interligações entre blocos ou de pequenos enganos que passavam despercebidos. Acredito que houve algumas

Apesar de o tempo ter parecido escasso, conseguimos implementar as especificações com sucesso e estão todas funcionais.

Tendo em conta o trabalho realizado por cada um de nós, acreditamos que cada um realizou a mesma quantidade de trabalho, logo a divisão de pontos deverá ser 50/50.

Trabalho realizado por

Daniel Emídio: nmec: 108986
Gabriel Costa : nmec : 109050