Universidade Federal de Santa Catarina
EEL5105: Circuitos e Técnicas Digitais
Semestre: 2018/2 – Projeto

Batalha Naval

O projeto final consiste na implementação de um circuito na placa de desenvolvimento DE1 fazendo uso das estruturas e conhecimentos obtidos durante o curso. O circuito vai implementar um jogo interativo que simula um jogo de batalha naval. O comportamento do jogo está definido a seguir:

- O usuário começa no estado *Init* e dá início ao jogo pressionando o botão de pressão enter ($KEY(1)$). Uma vez no estado *Setup* o usuário deve escolher uma das 4 combinações possíveis com os *Switches* 1 a 0, $SW(1..0)$, de modo a selecionar a dificuldade do jogo, expressa pela frequência de contagem do relógio. As diferentes frequências serão geradas por um conversor de frequências, podendo ser de 0.5Hz, 1Hz, 2Hz ou 4Hz. Neste estado, os displays $HEX0$ e $HEX1$ mostrarão o nível selecionado (de 0 a 3) e a letra $'L'$, de level, respectivamente. Os demais displays estarão desligados.
- Uma vez pressionado enter de novo o jogo passa ao estado *Play* e se inicia o jogo. No estado *Play*, o usuário tem até a contagem chegar a 10 para selecionar uma linha de 0 a 7, usando os Switches $SW(9..7)$, e uma única coluna, usando os *Switches* $SW(6..0)$. Neste estado, os displays $HEX5$ e $HEX4$ mostrarão a letra $'t'$ de time e uma contagem regressiva de 9 a 0 com a frequência escolhida no estado Setup, respectivamente. Os displays $HEX3$ e $HEX2$ mostrarão, respectivamente, a linha e a coluna escolhidas, sendo que a linha será o endereço a ser carregado de uma de duas memórias ROM para posterior verificação. Tais memórias possuem 8 linhas de informação de 7-bits ($2^3 \times 7$), e o aluno pode encontrar um exemplo de memória parcialmente descrito no Moodle da disciplina. Cabe ao aluno preencher as memórias com oito $'1's$ lógicos representando os barcos (um navio de comprimento três bits, dois navios de comprimento dois bits, um navio de comprimento um bit).
- No estado *Play*, os displays $HEX1$ e $HEX0$ devem mostrar, respectivamente, a letra $'U'$ de user, e o jogador atual. É importante destacar que o jogador unicamente pode introduzir um $'1'$ lógico referente à coluna escolhida nos $SW(6..0)$. Se o jogador não pressiona fire ($KEY2$) antes do fim da contagem, um sinal de status chamado *end_time* é ativado e o jogo vai para o estado *Result*, e o outro jogador vence automaticamente. Uma vez selecionadas as coordenadas, o usuário deve pressionar fire e o jogo passará ao estado Check onde será avaliado se:

  1) O jogador não introduziu um único $'1'$ lógico nos Switches onde um sinal de status chamado *sw_erro* será ativado, e o outro jogador vence automaticamente;

  2) O jogador acabou o número de rodadas máximo, onde caso tinha chegado a 30 rodadas, um sinal de status chamado end_round será ativado, e o outro jogador vence automaticamente;

  3) O jogador adivinhou uma a posição de um $'1'$ lógico do mapa, em que será contabilizado um ponto em seu contador;

  4) O jogador adivinhou as posições dos oito $'1'$ lógicos do mapa, onde um sinal de status chamado *end_game* será ativado. Caso um dos três sinais de status esteja ativo, o jogo vai para um estado Result, em caso contrário vai para um estado *Next_Round* onde é preparada a próxima rodada.
- No estado *Next_Round*, será somado um ao contador de rodadas, e passa ao estado *Wait*. Em *Wait*, os displays $HEX5$ e $HEX4$ mostrarão as letras $'r'$ e $'d'$ de round e os $HEX3$ e $HEX2$ a contagem da rodada; os displays $HEX1$ e $HEX0$ mostrarão o jogador da próxima rodada. Quando o jogador pressiona enter, passa de novo ao estado *Play* para a seguinte rodada. Os jogadores possuem 30 rodadas para limpar o mapa. O placar será mostrado nas $LEDR(8..5)$ para jogador 0 e $LEDR(3..0)$ para jogador 1 ao longo dos estados *Play*, *Next_Round* e *Result*.
- No estado *Result*, é mostrado o resultado do jogo. Os displays $HEX5$ a $HEX2$ mostrarão as letras $'U'$, $'S'$, $'E'$ e $'r'$ e o $HEX0$ mostrará o usuário ganhador (0 ou 1). Para iniciar outro jogo, os usuários podem pressionar *enter*, e retorna ao estado *Init*.
- Um usuário pode em qualquer momento parar o jogo usando o botão de pressão *reset* (KEY0) zerando o sistema, para assim re-iniciar de novo.
- Visando evitar problemas de temporização em função do aperto de um KEY por um ser humano durar muitos ciclos de *clock*, o *Button Press Synchronizer* (ButtonSync) será fornecido em conjunto com o projeto deve ser utilizado. O *ButtonSync* converte apertos das KEYS em pulsos com período de um ciclo de clock. Assim, em seu projeto, as KEYS devem ser ligadas nas estradas do *ButtonSync*, e as saídas BTN0 a BTN3 do *ButtonSync* deverão ser utilizadas para controlar o projeto.
- O projeto deverá ser implementado **obrigatoriamente** usando a abordagem *datapath-control* vista nas aulas de teoria. Um modelo de datapath parcialmente completo pode ser encontrado no Moodle da disciplina.