universidade de aveiro
theoria poiesis praxis

Relatório - Master Mind

Turma P17 - Diogo Daniel Soares Ferreira e Eduardo Reis Silva
Universidade de Aveiro - Laboratórios de Sistemas Digitais

26 de Maio de 2015

Conteúdo

Introdução

O objetivo deste projeto seria o de programar a *FPGA* de forma a que fosse possível interagir de forma a jogar o jogo *Master Mind*. O sistema deveria gerar automaticamente um número aleatório com quatro dígitos. O utilizador deveria selecionar um número, também com quatro dígitos entre 0 e 9, para tentar adivinhar o número aleatório gerado pela máquina. Caso atingisse as 99 tentativas sem acertar, a *FPGA* deveria mostrar o número de tentativas e impedir o utilizador de inserir mais números. Caso contrário, o número de tentativas só seria mostrado quando o utilizador acertasse no número gerado, fazendo piscar a uma frequência de 2 Hz o número inserido. Finalmente, para auxiliar o utilizador, dois *displays* hexadecimais devem mostrar o número de dígitos certos na posição errada e o número de dígitos certos na posição certa.

Manual do utilizador

O utilizador deverá interagir com a *FPGA* através dos quatro botões (*KEY0* para alterar o dígito escolhido, *KEY1* para incrementar o dígito, *KEY2* para decrementar e *KEY3* para confirmar o número introduzido) e através de um interruptor (*SW0*) deverá poder reiniciar todo o sistema. Os oito *displays* hexadecimais deverão mostrar, da esquerda para a direita (de HEX7 a HEX0), o número de dígitos certos na posição certa (HEX7), o número de dígitos certos na posição errada (HEX6), o número de tentativas (HEX5 e HEX4) e o número inserido pelo utilizador (HEX3, HEX2, HEX1 e HEX0).

Arquitetura

O diagrama de blocos do sistema, por ser demasiado extenso, encontra-se no anexo (ficheiro "Anexo 1 - Diagrama de Blocos.jpeg"). Seguidamente iremos explicar passo a passo a função de cada entidade descrita nesse diagrama, da esquerda para a direita.

O sistema tem como entradas três botões (no diagrama *Key(3), Key(2), Key(1) e Key(0)*), um interruptor (*reset*) e um relógio (*clock*). Ligado a cada botão está um *debouncer*, para evitar possíveis transições indesejadas.

Ligado ao relógio estão dois divisores de frequência, necessários para entidades seguintes. Assim, para além do relógio de frequência 50 *Mhz*, temos também relógios a 100 *Hz* e a 2 *Hz*.

Acima (a cor-de-laranja), podemos ver a entidade *RandomNumber*, destinada a gerir o *enable* de um contador para gerar o número aleatório (acima, a vermelho). As saídas deste contador estão ligadas à entidade *compare*, que comparará este número com o introduzido pelo utilizador.

A saída do primeiro *debouncer*, ligado ao botão *Key(3)*, serve para confirmar a entrada de um novo número pelo utilizador (*confirm*). Os dois botões seguintes servem para incrementar ou decrementar um certo dígito (*inc, dim*), escolhido pelo botão *Key(0)(set)*.

O sinal *confirm* irá ser ligado a um contador de 0 a 99, para contar o número de tentativas introduzidas, que estará ligado à entidade *CheckEnd*, que falaremos mais à frente. Os sinais *dim* e *inc* estão ligado a contadores de 0 a 9, com *enable* e *up/down*, capazes de somar ou decrementar certo dígito. O sinal *set* irá estar ligado a um contador de 0 a 3, que por sua vez estará ligado a um *Demultiplexer*, para controlar o *enable* de cada contador. Assim, podemos ter a certeza que apenas um dígito está a ser alterado de cada vez. As saídas do *Demultiplexer* estão ainda ligadas ao *enable* da entidade *blink*, que fará piscar o dígito em causa, para o utilizador saber qual dígito está a alterar.

A saída de cada contador está ligada a um *register*, que tem como *clock* o sinal *confirm*. Assim, quando o utilizador confirmar o número que deseja, passa para a saída do *register*. Caso contrário, mantém-se o anterior. A saída dos contadores também está ligada à entidade *blink*, para o utilizador poder ver o número que está a introduzir. Os sinais vindos do *register* são também ligados ao módulo *compare*, que comparará os números vindos do utilizador e o número aleatório, e dirá quantos dígitos há nas posições certas e quantos dígitos certos há, nas posições erradas.

A entidade *CheckEnd* irá receber o número de dígitos certos nas posições certas e erradas e o número de tentativas. Se o número de tentativas for 99, deverá deixar de mostrar o número do utilizador, e o número de dígitos certos e errados mostrado deverá ser 0. Caso o número de dígitos nas posições certas seja 4, deverá mostrar o número de tentativas e o número do utilizador deve piscar. Em qualquer um destes cenários, o sistema só pode sair deste estado com o *reset*. Caso nenhum destes cenários aconteça, deverão ser mostrados o número de dígitos nas posições

certas e erradas e o número que o utilizador está a selecionar. Finalmente, as últimas entidades (a amarelo) são conversores de binário para código *BCD* de 7 segmentos.

Implementação

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

Optámos por criar uma entidade *Debouncer* ao invés de usar a fornecida pelo professor nos guiões. O nosso *Debouncer* é um *shift-register* com *reset* assíncrono. Após cinco pulsos de relógio, se não houver nenhuma variação do sinal de entrada, o *debouncer* irá ter saída '1'. Caso contrário, irá ter a sua saída a '0'.

Os divisores de frequência são os fornecidos no guião da aula prática. A cada pulso de *clock*, um contador soma um valor a um sinal. Caso esse valor seja menor do que metade de um número n, a sua saída será 0. Caso seja igual ou maior, será '1'. Assim, podemos dividir uma frequência recebida por qualquer número N inserido.

A entidade *RandomNumber* é uma máquina de estados com três estados (ver "Anexo 2 - RandomNumber (Máquina de Estados)") com três estados. No primeiro estado (S0), a saída count fica a '1' e *resetOut* a '0'. Depois, caso *stop_signal* seja '1', o estado seguinte vai ser S1, caso *reset* seja '1', o estado seguinte vai ser S2. No estado S1, *resetOut* fica a '0' e *stop_signal* a '1'. Caso *reset* seja '1', o estado seguinte será S2. Finalmente, para o estado S2, *resetOut* irá ser '1' e *count* irá ser '0'.

O *resetOut* irá ligar-se ao *reset* do contador de 0 a 9999, e o *count* irá ser o *enable* do contador. O contador somará sempre que o seu *enable* for '1'. Quando o *reset* for '1', deverá voltar a '0'.

Ligado ao último *debouncer*, temos o sinal *set*, ligado a um contador simples de 0 a 3. Esse contador está ligado a um *Demultiplexer*, para selecionar apenas uma saída a '1'. Cada saída do *Demultiplexer* está ligada a contadores de 0 a 9, com entradas para incrementar e decrementar, e *enable*.

No diagrama de blocos a verde, o *Register*, tem como *clock* o sinal *set*, e *reset* assíncrono (interruptor SW0). A entidade só deverá passar para a sua saída a entrada nos flancos ativos do *clock*. Em baixo, podemos ver também um contador similar aos apresentados anteriormente.

A entidade *blink* tem como entrada um *clock*, uma entrada de *load*, um *enable* e um número. Caso o *enable* seja '1', no flanco ativo do relógio, o módulo irá piscar um número guardado num sinal interno *s_nin*, à frequência do *clock*. Caso *enable* seja '0', a entidade põe na sua saída o sinal *s_nin*, sem piscar. Assincronamente, sempre que a entrada *load* é '1', *s_nin* toma o valor do número de entrada. A entidade *Binary to Decimal* irá simplesmente receber um número e converter em BCD de sete segmentos. Cada entidade *Binary to Decimal* irá estar ligada a um *display* hexadecimal.

A entidade *Compare* é uma máquina de estados (imagem em "Anexo 4 - Compare (Máquina de Estados)"). No primeiro estado, inicializa todas as variáveis e sinais a "0000". No estado seguinte, compara cada dígito da palavra gerada com o seu respetivo dígito da palavra inserida pelo utilizador. Se forem iguais, a variável soma '1' ao seu valor. Há também duas variáveis que servem de *flag*, para saber se aquele dígito já foi comparado ou não. Se forem todos os dígitos certos, vai para o estado seguinte. Caso contrário, irá comparar dígito a dígito se há algum dígito igual na palavra correspondente onde o *flag* está a '0', ou seja, não foi comparada. O estado seguinte é um estado que não faz atribuições. Se nalgum estado, o *reset* for '1', volta para o seu estado inicial.

A entidade *CheckEnd* é também uma máquina de estados (a sua imagem pode ser encontrada em "Anexo 3 - CheckEnd (Máquina de Estados)"), com quatro estados. No estado inicial, fazem-se atribuições essenciais para inicializar o diagrama de estados e os seus sinais. Depois, passamos para o último estado, onde irá não irá piscar o número do utilizador, e também não irá mostrar o número de tentativas. Caso o utilizador acerte nos quatro dígitos, passa para segundo o estado. Caso o utilizador chegue às 99 tentativas, passa para o terceiro estado.

No segundo estado, o utilizador acertou no número gerado. Logo, irá ativar a saída *blink*, que fará piscar o dígito. Irá fixar os números inseridos pelo utilizador, sem os poder alterar, e irá também mostrar o número de tentativas. No terceiro estado, o utilizador chegou às 99 tentativas. Logo, irá mostrar o número de tentativas e não irá mostrar os dígitos inseridos pelo utilizador. Irá também forçar as saídas que apontam para o número de dígitos certos no local certo e no local errado a zero.

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 *RandomNumber*, para simular o seu comportamento quando os seus sinais de entrada variavam. Depois, aplicámos outra *Testbench* à entidade *Counter9999*, para garantirmos a sua sanidade e a validade da contagem. Como não tivemos a certeza de que o comportamento conjunto fosse o correto, também criámos uma *Testbench* simulando as duas entidades ligadas, tal como no diagrama de blocos.

Seguidamente, devido a alguns erros aquando da execução do projeto, criámos uma *Testbench* para os contadores de 0 a 3 (*counter4*) e outra para a sua interligação com o *Demultiplexer*.

Como tivemos bastantes dificuldades em completar a entidade *compare*, também criámos uma *Testbench* para ela, que usámos bastante para garantir a sanidade da entidade.

Finalmente, também criámos *Testbenches* para as entidades *CheckEnd* e *Blink*.

Conclusão

O projeto realizado cumpriu o definido anteriormente. 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. De tal maneira que, depois de feito, passámos largos minutos apenas a "jogar"no que tínhamos construído.

A maior dificuldade sentida foi na construção da entidade com a máquina de estados *compare*. Despendemos bastante tempo, primeiro a planear o que tínhamos de construir, depois na sua programação, de tal maneira que tivemos, a certa altura, que construir a entidade a partir do zero.

Conseguimos com que o projeto fosse feito de forma faseada, cumprindo assim as datas impostas.