Temporizador de Reação

Universidade de Aveiro

Pedro Martins, Pedro Santos

Temporizador de Reação

Departamento de Eletrónica Telecomunicações e Informática

Universidade de Aveiro

Pedro Martins, Pedro Santos
pbmartins@ua.pt, pedroamaralsantos@ua.pt

06/05/2015

Conteúdo

Capítulo 1

Introdução

Para avaliação da componente prática da disciplina de Laboratórios de Sistemas Digitais, foi-nos proposto a criação de um mini-projeto, no qual teríamos de criar uma arquitetura e os seus variados blocos em Very high speed integrated circuit Hardware Description Language (VHDL), a qual seria utilizada para programar uma Field-Programmable Gate Array (FPGA), a Terasic DE2-115. Foi determinado pelo grupo que seria implementado um simples temporizador de reação, no qual seria utilizado botões e *switches* da FPGA, assim como um comando infravermelhos, de maneira a que seja possível jogar em ambas as plataformas. Para a construção do projeto, foram utilizadas máquinas de estados, assim como blocos de lógica simples e ainda as *blackboxes* relativas às interfaces de infravermelhos e áudio, disponibilizadas pelos docentes da disciplina.

Capítulo 2

Análise

O mini-projeto em análise consiste na implementação de um temporizador de reação. A ideia pode traduzir-se como, depois de aceso um LED, medir o tempo que o utilizador demora a carregar num botão pré-definido, registando o tempo decorrido entre ambos.

Como foi implementado um descodificador de infravermelhos, pode utilizar-se tanto o comando (desde que este envie informação no formato NEC), como os botões e *switches* da FPGA, no entanto, a plataforma de infravermelhos tem funcionalidades reduzidas.

Assim sendo, existe um botão (`KEY(0)`) na FPGA e outro no comando (botão de Play) para iniciar o jogo e que também terá como função parar o cronómetro que contabilizará o tempo de reação assim que o LED é ligado.

Existirá também um botão que servirá para fazer *Reset* ao sistema em qualquer ponto do seu funcionamento (`KEY(0)` na FPGA e Return no comando), e um botão para parametrizar o tempo de espera antes de aparecer o LED verde, que simboliza o início da contagem do tempo de reação. Caso esteja ativo o `SW(0)` da FPGA, esse tempo será definido como 5 segundos, caso contrário será um valor aleatório.

Assim que o utilizador carrega no botão de iniciar o jogo, é gerado um número aleatório entre 5 e 60 (e validado), que será o tempo, em segundos, que demorará o LED a acender desde que se iniciou o jogo. De seguida, é utilizado um "semáforo de partida", onde a cada segundo, 3 LED vermelhos se apagam e onde é emitido um som (caso o som esteja ativado), até que se inicia a contagem do tempo até o LED indicador se acender.

Se o utilizador carregar no botão de jogar antes de o LED acender, é impresso nos ecrãs de 7 segmentos uma mensagem de erro. Caso o utilizador apenas clique no botão depois de aceso, é imprimido nos ecrãs de 7 segmentos o tempo percorrido desde que o LED acendeu até o utilizador carregar no botão. A FPGA manter-se-á neste estado até que se reinicie o jogo, isto é, clicar no botão de *Reset*, onde todos os painéis de 7 segmentos e todos os LED serão apagados.

2.1 Arquitetura

2.2 Arquitetura

A figura Figura 2.1 apresenta uma arquitetura do sistema em geral.

O projeto é constituído por 13 blocos.

O *Infrared_ Core* é o bloco que controla o comando de infravermelhos (com ligação à interface da FPGA) e o *Audio_ Core* (com ligações às entradas e saídas de áudio da FPGA) é o que controla a interface áudio. Ambos têm ligações diretas ao relógio de 50 MHz e são blocos construídos tendo como uma base *blackboxes* fornecidas pelos docentes da disciplina.

As entradas de sinais para o circuito são controladas pelos blocos de *debounce* e pelo controlador de infravermelhos, daí a existência de lógica simples entre as saídas das *DebounceUnit*, para que, caso uma das entradas seja ativada (quer no comando ou nos botões da FPGA), possa ser interpretada a sua ativação.

Excetuando o *pseudo_ random_ generator*, o *HexDisplay* e os blocos *FreqDivider* (divisores de frequência de relógio), todos os restantes blocos têm uma entrada de *reset* ligada ao sinal `key1` (resultante do impulso do clique num botão ou no comando de infravermelhos).

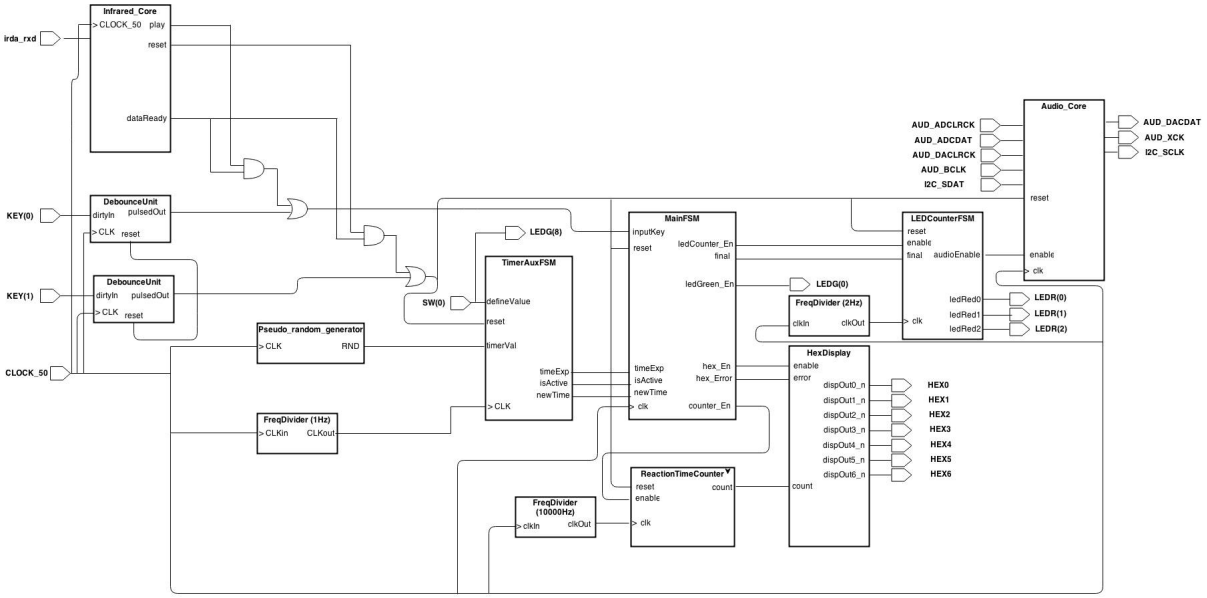

Figura 2.1: Arquitetura do temporizador de reação.

A *Main FSM* é o bloco central que coordena todo o processo. Recebe um sinal `key0` (botão de jogar, quer na placa, quer no comando de infravermelhos), para além de outros provenientes de outros blocos e máquina de estados.

A máquina principal comunica com outra máquina de estados, *LEDCounter FSM*, através da saída `LedCounter_En` e da entrada `final`. Esta última, para além destas entradas e do `reset`, está associada a um *FreqDivider* que exporta um relógio de frequência 2 Hz. Para além disso, também comunica com o bloco de áudio, pois é esta máquina que envia sinais intermitentes de ativação a este último (consoante o `SW(1)`, *mute*, esteja ativo ou não), através da saída `audioEnable`.

Assim que recebe um sinal de que a *LEDCounter FSM* acabou a sua execução (`final`), ativa a saída `ledGreen_En`, que está diretamente ligada à saída `LEDG(0)` da FPGA.

Esta máquina principal também comunica com a *TimerAux FSM* (que além do sinal de *reset*, está ligada a um *FreqDivider* que exporta um relógio de 1 Hz) através da saída `newTime` e das entradas `isActive` e `timeExp`. Esta última máquina de estados tem também uma entrada ligada ao *pseudo_ random_ generator*, um gerador pseudo-aleatório de números, que definirá o tempo a ser decrementado na máquina (caso o pretendido pelo utilizador seja um tempo de espera aleatório).

Quando o sinal `timeExp` está ativo, significa que a máquina de tempo auxiliar acabou a sua execução, ativando as saídas `counter_En`, que está ligada diretamente ao bloco *ReactionTimeCounter* (que funciona a uma frequência de 10000 Hz), e `hex_En`, ligada ao bloco `HexDisplay`, que ativa e mostra nos ecrãs de 7 segmentos o tempo do contador do tempo de reação, pois recebe um vetor de 32 *bits*, e os descodifica para mostrá-los nos 8 ecrãs de 7 segmentos. A máquina principal também tem uma ligação com este último bloco (`hex_Error`), que é ativado em caso de erro no sistema.

2.3 Implementação

Este projeto foi construído tendo como base do seu funcionamento blocos lógicos simples, assim como máquinas de estados.

Primeiramente, é importante referir que os sinais proveniente dos botões `KEY` da FPGA, foram todos passados por um *Debouncer* (bloco baseado num com a mesma função desenvolvido nas aulas), de modo a que não haja oscilações na altura do clique e seja enviado apenas um impulso.

Por outro lado, o descodificador de infravermelhos também foi baseado no fornecido pelos docentes da disciplina. Foram modificados vários aspetos, nomeadamente as funções associadas a cada botão e um analisador de impulsos, para que o sinal não seja sempre contínuo (por exemplo, para o caso em que se clica no botão Play: se o sinal fosse contínuo, quando clicássemos nesse botão, ele ia estar sempre ativo e ia, mesmo antes de aparecer o LED indicador, ele dar a mensagem de error de clicar no botão de jogar

antes de o LED acender).

2.3.1 *Main FSM*

O "cérebro"de todo o projeto é a máquina de estados *Main FSM*. É ela que gere as ativações de todos os blocos e máquinas de estados auxiliares, consoante as entradas do dispositivo (botões da FPGA e comando de infravermelhos), assim como consoante os sinais provenientes dos restantes blocos.

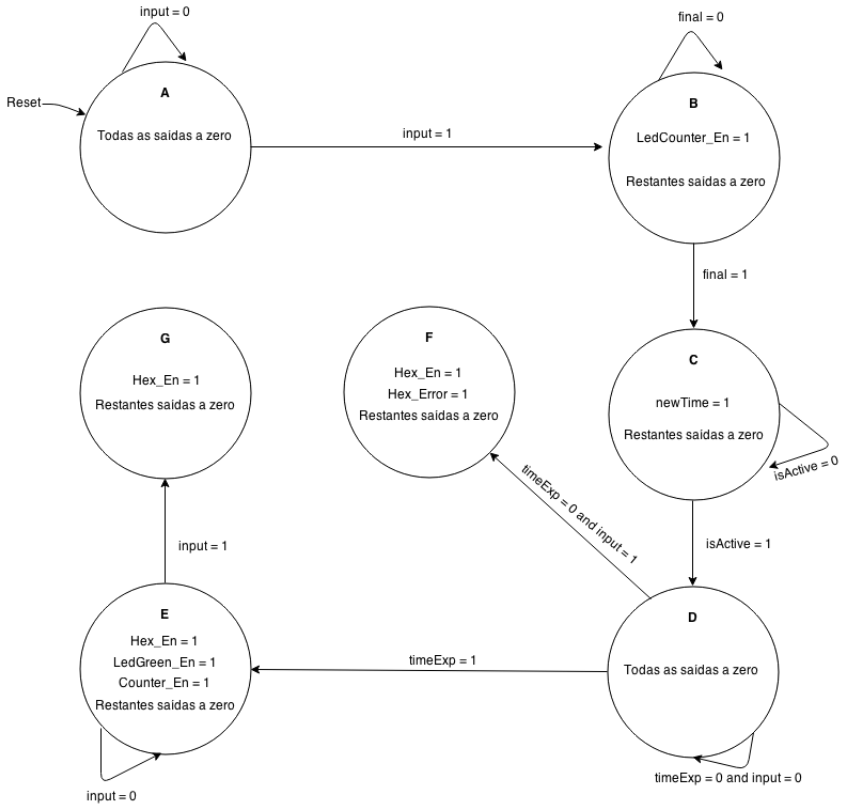

Figura 2.2: Diagrama de estados da *Main FSM*.

2.3.2 *LEDCounter FSM*

Assim que é iniciado o jogo, é dado um "sinal de partida", gerado pela *LEDCounter FSM*. Esta máquina de estados, que tem um relógio de frequência 2Hz (0.5 segundos - gerado pelo bloco *FreqDivider*), recebe um sinal que a ativa, ligando três LEDs vermelhos, e, a cada dois tiques de relógio, vai desligando-os um a um. Para além disso, a cada tique, envia um sinal de `enable`, que ativa e desativa o bloco *Audio_ Core*, responsável pela geração de um som e comunicação com a *blackbox* da interface áudio, que resultará na emissão de um som alternadamente ligado (nos dois primeiros LEDs) e continuamente ligado (no último LED), até que todos sejam desligados.

O bloco *Audio_ Core* é baseado no bloco *AudioDemo* fornecido pelos professores como exemplo de interação com a interface áudio do kit Terasic DE2-115. Apenas foi modificado canal direito de saída, para este emitir o mesmo som que o da esquerda simultaneamente.

Ou seja, esta máquina funciona como um "semáforo de partida"e que, quando desligados todos os LEDs, inicia a contagem do tempo até que o LED verde (indicador) se acenda.

No entanto, se o `SW(1)` na FPGA estiver ativo ou tiver sido pressionado o botão de Mute no comando, não será emitido qualquer som, apesar de a contagem nos LEDs ser na mesma executada.

O diagrama da *LEDCounter FSM* pode ser observado na Figura 2.3.

2.3.3 *TimerAux FSM*

De seguida, assim que a máquina de estados anterior envie um sinal à *Main FSM* de que terminou a sua ação (através do sinal `final`), a máquina principal envia um outro sinal (`newTime`) à *TimerAux*

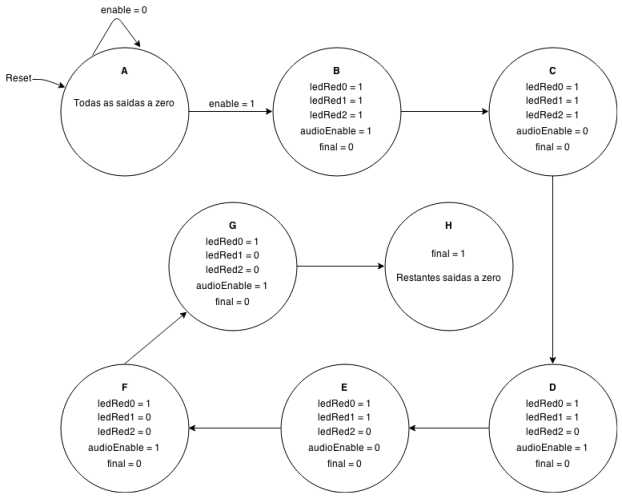

Figura 2.3: Arquitetura do *LEDCounterFSM*.

FSM, para que se comece a contar o tempo até o LED indicador se acender.

Esta máquina, para além de receber um sinal para iniciar a contagem, também recebe um sinal `defineValue` que, estando ativa, define se o tempo é parametrizado para 5 segundos, ou se, por outro lado, é um número aleatório recebido do bloco `random_number_generator` (valor este que é validado, pois apenas são aceites números entre 5 e 60 segundos).

Assim que a contagem chega a zero, a máquina envia um sinal a dizer que o tempo expirou e que já não está ativa.

O diagrama da *LEDCounter FSM* pode ser observado na Figura 2.4.

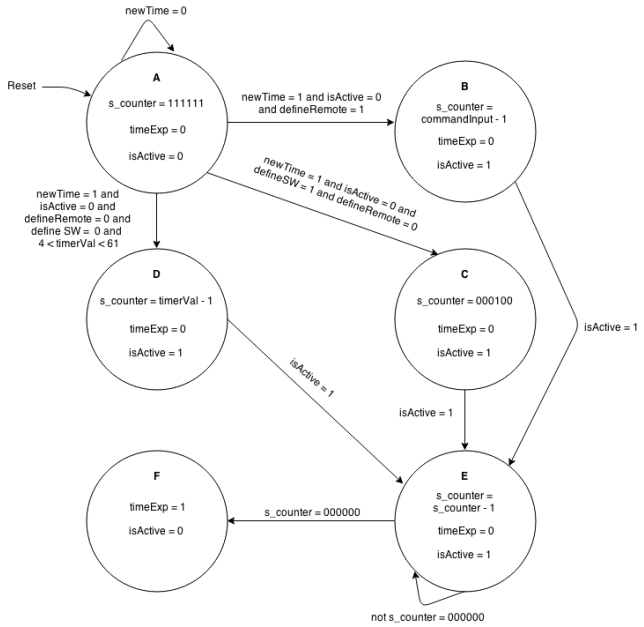

Figura 2.4: Diagrama de estados da *Timer Aux FSM*.

Por fim, assim que a máquina de estados principal recebe o sinal `timeExp` ativo, acende o LED verde (indicador) e ativa o contador do tempo de reação (*ReactionTimeCounter*), que funciona a uma frequência de 10000 Hz (também esta frequência foi gerada por um bloco *FreqDivider*), isto é, irá apresentar o resultado até às décimas de milésimas. Ao mesmo tempo, também é ativado o bloco *HexDisplay*, que interpreta o sinal vindo do contador e vai imprimindo nos ecrãs de 7 segmentos o tempo a percorrer.

Quando o utilizador carrega no botão de jogo, o contador pára e é impresso nos ecrãs o seu tempo de reação. No entanto, se carregar depois do sinal de partida, mas antes de o LED verde se acender, é impresso nos ecrãs de 7 segmentos uma mensagem de erro.

2.4 Validação

Apesar dos diversos blocos construídos, como alguns eram apenas de lógica simples (tais como o *ReactionTimeCounter*, um simples contador com `enable`, e o *HexDisplay*, que apenas converte um número binário de 24 dígitos para um formato de ecrã hexadecimal), decidiu-se apenas para validação das máquinas de estados, isto é, apenas foram desenvolvidas TestBenches para os blocos *Main FSM*, *TimerAux FSM* e *LEDCounter FSM*.

2.4.1 *Main FSM*

A construção da TestBench da máquina de estados principal (*Main FSM*), seguiu dois caminhos: um em que não eram cometidos quaisquer erros e outro onde era cometido um erro, isto é, considerava-se que o botão de jogar estava ativo antes de o LED indicador se acender, o que iria provocar um erro.

Segue-se o gráfico resultante:

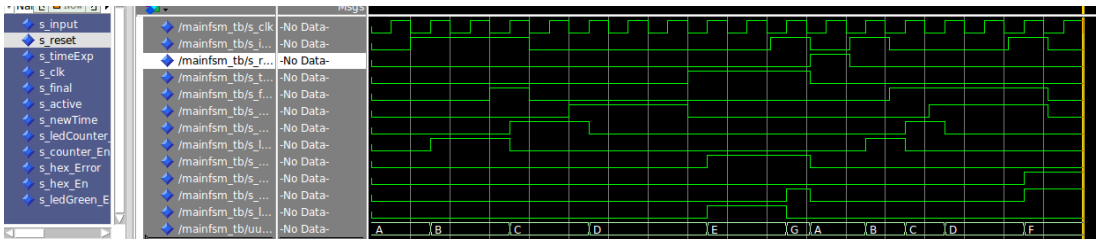

Figura 2.5: Gráfico da TestBench da *Main FSM*.

2.4.2 *TimerAux FSM*

O desenvolvimento da TestBench deste bloco exigiu a sua separação em 2 situações: quando o tempo de espera é parametrizado para 5 segundos pelo `SW(0)` ou pelo comando de infravermelhos, ou então sendo um valor aleatório entre 5 e 60 segundos.

A primeira situação foi do valor aleatório. Como estava ligado à entrada `timerVal` a saída do bloco `random_number_generator`, seria gerado um número aleatório a uma frequência de 50 MHz, e enquanto esse valor não fosse aceite (não estivesse no intervalo 5 a 60), a máquina não vai começar a subtrair o valor. Logo, neste teste, primeiro é fornecido um valor não válido e só depois um valido.

Por último, é testada a função de parametrizar o tempo.

Segue-se o gráfico resultante:

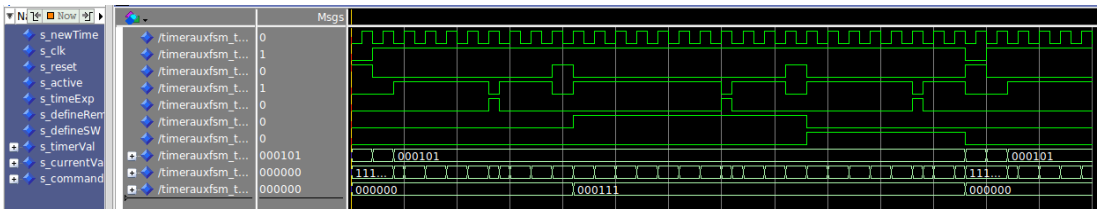

Figura 2.6: Gráfico da TestBench da *TimerAux FSM*.

2.4.3 *LEDCounter FSM*

Finalmente, esta máquina de estados apenas tem uma entrada, que é a de ativação. Logo, apenas esta foi variada.

2.5 Manual de Instruções

Para medir o seu tempo de reação, deve seguir os seguintes passos:

- Certificar-se que a FPGA está corretamente ligada e programada;

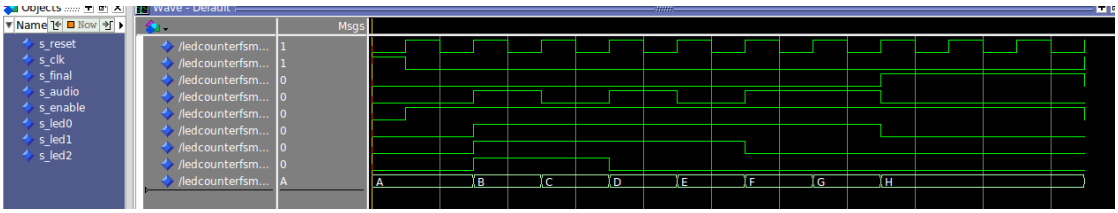

Figura 2.7: Gráfico da TestBench da *LEDCounter FSM*.

- Pressionar o botão `KEY(0)` (Play no comando de infravermelhos), para iniciar o jogo;
- Pode desligar o som da placa ativando o `SW(1)` (não tem opção no comando);
- Pode também parametrizar o tempo que demora a acender o LED verde (5 segundos), acionando o `SW(1)` (não tem opção no comando);
- Aguardar que os três LED vermelhos se apaguem;
- Clicar no botão `KEY(0)` (ou Play no comando de infravermelhos), logo depois de o LED verde se acender;
- Será impresso no ecrã hexadecimal o seu tempo de reação;
- Para reiniciar o jogo, basta pressionar `KEY(1)` (ou Return no comando de infravermelhos) e repetir todos os passos acima descritos.

Capítulo 3

Conclusões

Em suma, para realizar este projeto foi necessário criar vários blocos com o objetivo de calcular o tempo de reação do utilizador. Para além disso, para complementar o projeto foram implementados dois blocos o do som e o bloco dos infravermelhos. Assim, é possível interagir com o projeto usando o comando de infravermelhos e se for pretendido o Countdown que indica o inicio do jogo é acompanhado com som, podendo este ser omitido caso seja desejado.

Acrónimos

FPGA Field-Programmable Gate Array

VHDL Very high speed integrated circuit Hardware Description Language