Projeto 9

UNIVERSIDADE DE AVEIRO

Gonçalo Silva, número mecanográfico: 103244
Samuel Teixeira, número mecanográfico: 103325

VERSAO 1.0

Agradecimentos

Queremos agradecer aos professores da cadeira de LABI por fornecerem um projeto relevante para o uso dos sistemas digitais e que nos elucida quanto à importância desta ciência no dia a dia do ser humano.

Índice

Capítulo 1

Introdução

Neste trabalho, vamos realizar o projeto 9, que corresponde a um cronómetro digital programável. O objetivo consiste na criação do cronómetro referido utilizando certas metodologias/conceitos aprendidos na cadeira Laboratório de Sistemas Digitais (LSD), de um relatório em formato "pdf"sobre o projeto e ainda de uma apresentação em formato "ppt". O cronómetro possui modo crescente e decrescente, um modo de programador e opções para parar, começar e recomeçar do ponto de partida(reset).

O cronómero tem o formato "XX:00:00:00", sendo "XX"a indicação do modo em que se encontra ("r-"ou ou "r_"ou "PP"). O sistema é feito de maneira a que se comecem a contar os segundos, seguindo-se as dezenas de segundo, minutos, dezenas de minuto, horas e dezenas de hora. Ao iniciar, o cronómetro começa a contar o tempo a partir de "00:00:00", sendo "r-"(modo crescente) o modo indicado. O cronómetro possui ainda o modo de programador, que é capaz de alterar dígitos selecionados (somente segundos e minutos) e termina com o acerto dos minutos.

Capítulo 2

Arquitetura

O circuito tem 4 botões (KEYS) e 2 interruptores (SW(1) E SW(2)) como entradas, todos eles estão ligados a debouncers para estabilizar o sinal enviado, evitando transições indesejadas. O clock de 50 MHz é enviado para o Sync Generator, que transforma este em mais 3 sinais de clock: progClk que vai para ProgCntrl 1KHz, timeClk que vai para o counter0 à frequência de 100Hz e dispClk que vai para DispCntrl à frequência de 1KHz. O clock de 50MHz também é utilizado para ser ligado aos restantes blocos que necessitam de clock. O Debouncer4 envia a sua saída para ProgCntrl, mais especificamente para ProgStart, que decide se entra em modo de programador ou não. O Debouncer2 envia a sua saída para todos os blocos com reset. Debouncer6 envia a sua saída juntamente com a saída de Debouncer5 para o UpDown de todos os counters, para indicar qual o acerto do dígito caso o cronómetro esteja em modo de programação. A saída de ProgCntrl sel é redirecionada para os 4 counters (que correspondem aos dígitos que podem ser programados) e indica, no caso de modo de programação, se esse dígito está selecionado. A saída ProgBusy vai para DispCntrl, para o Mux 8:1 e para os counters, servindo para indicar o modo de programação "PP"caso esteja nesse modo. O sinal proveniente do OR entre stop e clearStop (proveniente da máquina de estados que gere o estado do cronómetro, se está parado, a contar, em reset parado ou em reset a contar) vai para mainEn dos Counters e é o enable principal que indica se os counters estão em funcionamento.

Todos os TC's dos counters são enviados para as entradas dataIn0 até dataIn5, representando os números que aparecem nos displays, à exceção de dataIn6 e dataIn7 que indicam o modo e, como tal, não são provenientes dos counters, Para além disso, os TC's também são enviados para os Equals que daram origem a s_max_value, que indica se já foi atingido o mínimo ou máximo contável (00:00:00 ou 59:59:99). Este sinal junta-se através de um OR à saída do Debouncer1 para iniciar o a máquina de estados (a amarelo) que escolhe um dos 4 estados que indicam como se encontra o cronómetro: start, stop, cleanStart ou cleanStop. O ClkDividerN recebe a frequência de CLOCK_50 e,

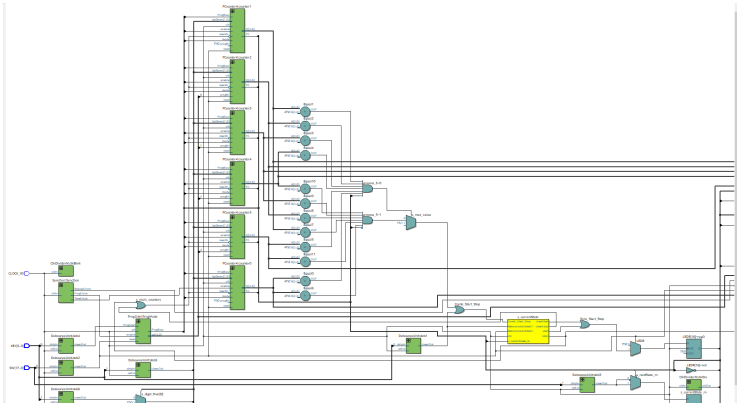

Figura 2.1: Circuito do Cronómetro

caso o cronómetro se encontre em modo de programação, diminui a frequência de modo a colocar o dígito selecionado a piscar. O sel proveniente de Progcntrl e stop da máquina que escolhe entre os referidos quatro estados vão para as entradas counter_en e start, respetivamente, de DispCntrl, que escolhe qual o display que deve funcionar. As suas saídas são sel e regSel, sel vai para o multiplexer e escolhe uma das 8 entradas correspondentes ao que é apresentado nos displays e regSel vai para os registers, dividindo os 8 sinais do vetor entre os 8 registers (o MSB vai para o último register e o LSB vai para o primeiro, os restantes vão por ordem crescente até chegar ao MSB). A saída do Mux 8:1 é encaminhada para Bin7SegDecoder, que descodifica o vetor de 3 bits para o vetor de 8 bits que é ligado aos registers, representando a informação (número) selecionada pelo Mux para aparecer num dos 8 displays.

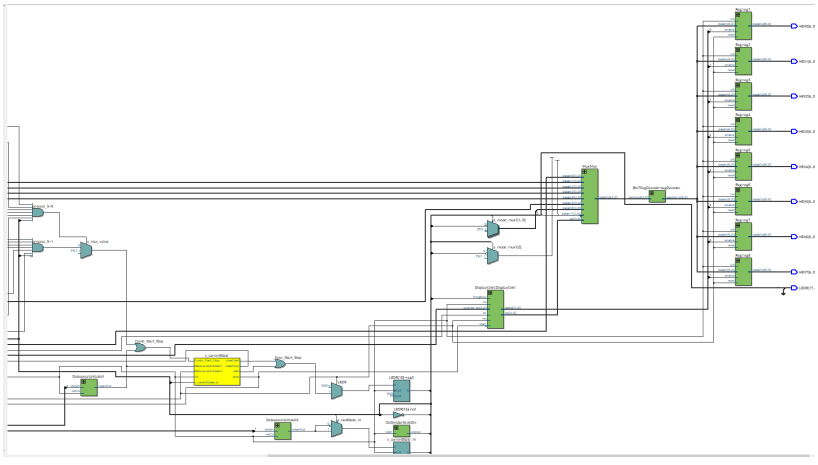

Figura 2.2: Circuito do Cronómetro

Capítulo 3

Implementação

3.1 Counters

Os Counters usados são de 4 bits pois pretendemos que o máximo seja 9 (1001). Possui como entradas clock, 4 tipos de enables(enable, mainEn, progBusy e progEn) e mode. As saídas são TC e o vetor Q.
No instante ascendente do clock, e se reset for ativado, o valor de s_count volta ao valor original (0) e TC terá o valor de '0'. Caso progBusy e progEn sejam '1' e upDown seja "10"ou "01", é somado("10") ou subtraído("01") a s_count o valor '1'. Porém, se o valor de s_count atual for o máximo para "10"ou o mínimo para "01", s_count passa a ter o valor original e TC é '1'.
Quando o clock não se encontra ascendente, se enable e mainEn tiverem o valor de '1',

Caso o enable esteja a '1', o Counter inicia funcionamento.

3.2 Debouncers

O Debouncer possui como entradas refclk, dirtyIn e como saída cleanOut.

No instante ascendente do relógio, se inPolarity estiver a 1, s_dirtyIn equivalerá a dirtyIn, caso contrário equivale à negação de dirtyIn. o sinal s_previousIn tem o valor de s_dirtyIn. Nesse mesmo instante, se s_dirtyIn for 0 ou se s_debounceCnt for menor que MIN_IN_WIDTH_CYCLES, então s_debounceCnt e s_cleanOut são 0. Se s_dirtyIn for 1 e s_previousIn for 0, então s_debounceCnt é igual a MIN_IN_WIDTH_CYCLES e s_cleanOut é 0. Caso contrário, se s_debounceCnt for maior ou igual a 1, é subtraido 1 ao seu valor. Se s_debounceCnt for 1, s_cleanOut é 1, caso contrário é 0. Por fim, cleanOut equivale a s_cleanOut quando outPolarity for 1, senão cleanOut é igual à negação de s_cleanOut.

3.3 Registers

O Register possui como entradas clk, enable, reset, e dataIn e como saída dataOut.

Quando o clock estiver no instante ascendente, se o enable e o reset forem '1', dataOut tem o valor de "0000000". Se o enable for '1' e se o resete for '0', dataOut regista dataIn.

3.4 Geradores de Pulso

O Gerador de Pulso tem como entradas clkIn e como saídas pulseOut0 e pulseOut1.

O sinal s_counter vai contar até alcançar o valor do genérico numberSteps. Desta forma, pulseOut0 só é 1 quando s_counter for um múltiplo de out0CompVal e pulseOut1 só é 1 quando s_counter for múltiplo de out1CompVal.

3.5 Multiplexer

O multiplexer tem 8 dataIn's de 0 até 7 (ex: dataIn0, dataIn1, ... dataIn7) e sel e como saídas dataOut.

O funcionamento não difere em nada de um típico Multiplexer 8:1, o dataOut terá o valor de um dos dataIn's dependendo de sel. Se o sel for "0001", seleciona dataIn0, se for "0010", seleciona dataIn1 e assim sucessivamente por ordem binária NBCD.

3.6 Clock Divider

O Clock Divider tem como entrada clkIn e como saída clkOut. Existe também um genérico divFactor.

o Clock Divider vai dividir a frequência de clkIn por um número, indicado pelo genérico divFactor. No instante ascendente do clock, s_divCounter vai ter incrementos de 1 enquanto não for igual a divFactor - 1, depois disso o seu valor é 0. Nesse mesmo instante, clkOut só tem o valor '1' quando s_divCounter for maior ou igual a divFactor/2 - 1.

3.7 Sync Generator

Sync Generator tem como entradas clKin e como saídas DisplayClock e TimeClock.

O Sync Generator vai contar até numberSteps, que no caso é a frequência de 50 milhões. TimeClock tem valor '1' apenas quando s_counter for múltiplo de out0CompVal e DisplayClock, da mesma maneira, terá valor '1' quando s_counter for múltiplo de out1CompVal.

3.8 Máquinas de Estados

3.8.1 DispCntrl

Esta máquina de estados é constituída por clk, res, en, start, ProgBusy e counter_en como entradas e sel, regSel e busy como saídas.

A máquina possui 8 estados(INIT, E1, E2, E3, E4, E5, E6, E7, E8). No instante ascendente de clk, se en e res tiverem valor de '1', o estado atual passa a ser o inicial(INIT), caso seja apenas en com esse valor, o estado atual poderá ser qualquer um dos 8, dependendo do estado atual e de comb_proc. Quando o estado é INIT, que é o estado em que sel e regSel são "0000"e "0000 0000"respetivamente, o estado (s_nextState) só muda para E1 (estado seguinte) se start for '1'. Quando o estado é E1, o estado só muda para E2 se, novamente, start for '1'. Este funcionamento é idêntico para todos os outros estados, diferindo apenas na possível mudança de estado(que avança no sentido INIT->E1->E2->E3->...->E8->E1->E2...) e nos valores de sel e regSel(que mudam no sentido crescente binário).

3.8.2 MainCntrl

MainCntrl é uma máquina de estados contida no ficheiro top level (ChronometerTop.vhd). É responsável por gerir todas as funcionalidades do cronómetro.

Gestão de Modos

Para gerir os modos, MainCntrl possui dois estados (growing ou crescente e decreasing ou decrescente. Se o estado atual for growing, o sinal s_mode_final é '0', indica que o modo é crescente. Apenas quando s_mode for '1' é que o estado muda para decreasing. No estado decreasing, somente s_mode_final muda pra '1' e o estado volta para growing se s_mode for '0'. No caso de ocorrer outra condição imprevista, o cronómetro volta para o estado growing por defeito.

Gestão de Paragem da contagem

De forma a parar ou reinicar a contagem, MainCntrl possui quatro estados(start de iniciar, stop de parar, clearStart de reset a iniciar e clearStop de reset parado). No instante ascendente de CLOCK_50, se houver reset e se o estado atual for start ou clearStart, o estado atual muda para clearStop se s_mode_final for '1', caso contrário o estado atual muda para clearStart. Na

circunstância de apenas s_reset ser '1', o estado atual muda para clearStop. Se nem essa condição se verificar, o estado atual é o estado seguinte.

O estado seguinte é escolhido através de Comb_Start_Stop. Quando o estado atual for stop,s_main_counters e s_main_Display são '0'(contadores e displays desligados) e o estado seguinte é start se s_start_stop(KEY(3)) for '1' ou stop caso contrário. No estado atual start, os contadores e os displays estão ligados e só se s_start_stop estiver ligado é que o estado seguinte muda e essa mudança é para start. No estado cleanStart, tanto os displays como os counters estão ligados e há mudança do estado seguinte se s_start_stop se for '1' para stop. Em cleanStop, apenas os displays estão ligados e, caso s_start_stop seja '1', há mudança de estado para start. Outra condição imprevista faz com que o estado seguinte seja stop por definição.

3.8.3 ProgCntrl

ProgCntrl possui como entradas clk, enable, reset e ProgStart e sel, OeSel e ProgStart como saídas.

No instante ascendente do clock, se enable e ProgStart forem 1 e se reset for 1, s_currentState(estado atual) é INIT(inicial), caso não haja reset o estado atual é o seguinte. Nesse mesmo instante, quando o estado atual é INIT, sel é igual "0000"ProgBusy é 0. O estado seguinte só muda se ProgStart for 1, então o estado seguinte é C5. No caso do estado atual ser um dos outros estados(C5, C4, C3, C2), o funcionamento é idêntico, mudando apenas ProgBusy, que é sempre 1 e a possível mudança do estado seguinte, que segue esta ordem (INIT->C5->C4->C3->C2->INIT->...). No caso de uma condição inesperada, o estado seguinte é por definição INIT.

Capítulo 4

Validação

Foram realizadas testbenches para testar os componentes que achávamos pertinente testar, que podem ser encontradas na pasta do projeto. Para além disso, também é possível encontrar os ficheiros VWF que estão feitos especificamente para confirmar o funcionamento correto de cada componente utilizado.

Capítulo 5

Manual do Utilizador

Nesta secção vamos explicar como utilizar o cronómetro digital.

A manipulação do relógio resume-se somente a 4 botões (KEY(0), KEY(1), KEY(2) e KEY(3)) e 2 interruptores(SW(1) e SW(2)). As funcionalidades são as seguintes:

- Iniciar/Parar Contagem
- Reset
- Modo Crescente/Decrescente
- Modo de Programação

5.1 Iniciar/Parar Contagem

Para iniciar ou parar o cronómetro, prima KEY(0).

5.2 Reset

Para dar reset, basta empurrar SW(0) para cima (assumindo que está a ver o cronómetro à sua frente). Se o cronómetro se encontrar parado, o tempo permanece 00:00:00. caso esteja a contar tempo, a contagem inicia automaticamente em 00:00:00.

5.3 Modo Crescente/Decrescente

O modo crescente é um modo em que a contagem é feita de uma maneira típica, almegando o máximo do tempo (59:59:99), enquanto que o modo decrescente faz com que a contagem se inverta e passe a aproximar-se de 00:00:00. Se pretende

usar este modo pela primeira vez, empurre SW(1) para cima(assumindo que está a ver o cronómetro à sua frente). Qualquer alternância de modos posterior à primeira, terá de ser feita empurrando SW(1) para baixo e novamente para cima, quer seja de modo decrescente para crescente ou de crescente para decrescente.

5.4 Modo de Programação

O modo de Programador permite ao utilizador alterar o número de um display na direção dos minutos para os segundos (os centésimos de segundo não são programáveis). Basta premir KEY(0) para entrar no modo e, com KEY(1) E KEY(2) aumenta ou decresce o número pretendido.

Capítulo 6

Conclusão

O projeto foi concluído com algum sucesso. Infelizmente, não fomos capazes de fazer a fase 3 e de completar a fase 2. Porém, construímos um cronómetro funcional, essas pequenas falhas, mas capaz de contar o tempo de forma crescente e decrescente, podendo pausar e reiniciar a contagem e podendo também voltar ao princípio.

A construção deste cronómetro fez os nossos horizontes expandirem-se, pois fazê-lo não é o mesmo que dar a teoria nas aulas. De facto, agora conseguimos ver com clareza como podemos usar os conteúdos lecionados nas aulas para criar sistemas digitais úteis, apesar da "simplicidade"do que fizemos. Este projeto trouxe mais conhecimento sobre como programar em FPGA e de todos os blocos presentes neste.

O Gonçalo Silva fez o DispCntrl, o ProgCntrl, quase todo o ChronometeTop, o DebounceUnit, o ClkDividerN, o PulseGeneratorN, o SyncGen e o Bin7SegDecoder e todas as testbench's.O Samuel Teixeira fez o restante código, que é o Mux, o Counter, o processo que escolhe o valor de s_max_value (limites do cronómetro), os ficheiros VWF e o relatório. Fizemos o trabalho sempre juntos, quase sempre nas salas do DETI. Por este motivo, ambos fizemos sugestões de algoritmos, testámos e resolvemos bugs de código que não foi feito por nós e estivemos sempre presentes para ajuda um do outro. A autoavaliação do projeto é de 15.5. O Gonçalo Silva fez 55% do trabalho, enquanto que o Samuel Teixeira fez 45%.

Acrónimos

LSD Laboratório de Sistemas Digitais