Laboratório de Sistemas Digitais

Trabalho Prático nº 7

Modelação em VHDL, Simulação e Síntese de Máquinas de Estados Finitos
Modelo de Moore

Objetivos

- Domínio dos procedimentos fundamentais no processo de síntese de máquinas de estados finitos (MEF).
- Utilização de VHDL para modelação ao nível comportamental de MEF.
- Desenvolvimento de estratégias de simulação de MEF.

Sumário

Este trabalho pretende proporcionar uma introdução à modelação comportamental em VHDL, simulação e síntese de MEFs, com enfoque no modelo de *Moore*. Na primeira parte apresenta-se um sistema completo baseado num cronómetro digital, para análise inicial e posterior extensão das funcionalidades implementadas. Na segunda parte é apresentado um problema de modelação e implementação duma máquina de venda de bebidas com especificações muito restritas e simples. Em ambas as partes deste trabalho, a modelação da MEF em VHDL é realizada de forma comportamental, recorrendo a dois processos interdependentes.

Parte I

No *site* de LSDig é disponibilizado o código fonte de um cronómetro digital, cuja estrutura se encontra na figura 1. O cronómetro utiliza 4 *displays* de 7 segmentos e apresenta a informação na forma "ss:cc", em que "ss" corresponde ao contador dos segundos e "cc" ao contador dos centésimos de segundo (ambos entre 00 e 99).

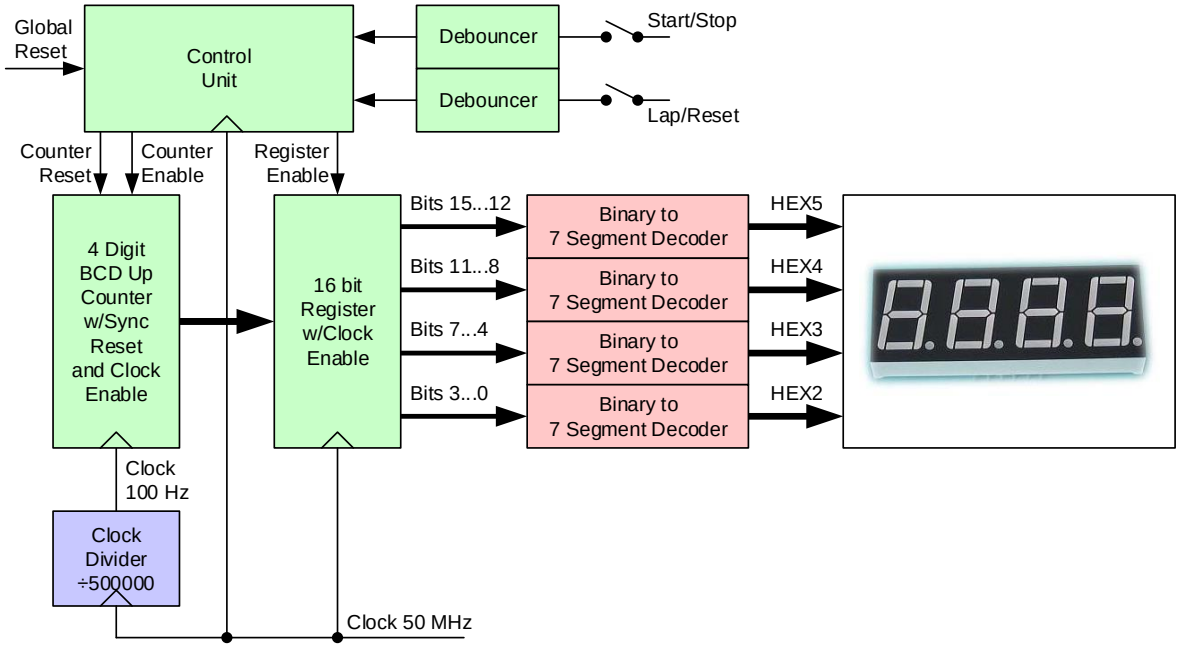

Figura 1 – Arquitetura do cronómetro fornecido.

1. Crie um novo projeto no IDE Altera Quartus II para a FPGA do *kit* DE2-115 (o nome do projeto poderá ser *Chronometer*).
2. Importe os ficheiros VHD fornecidos (com código fonte) para o projeto criado no ponto anterior.
3. Selecione o ficheiro "Chronometer.vhd" como o *top-level* do projeto.
4. Compile o projeto e teste-o no *kit* DE2-115 (não se esqueça de importar o ficheiro "DE2_115.qsf").
5. Verifique o funcionamento do cronómetro. As teclas do *kit* DE2-115 para controlo do cronómetro são as seguintes:

- KEY0 – *start/stop*
- KEY1 – *lap/reset*
- KEY3 – *global reset*

O funcionamento implementado no código fonte fornecido pode ser resumido da seguinte forma:

- Após o *reset* global, o cronómetro é colocado a "00:00", assim permanecendo até que seja premido o botão *start/stop*.
- Uma vez premido o botão ***start****/stop*, o cronómetro começa a contar. Nesta situação, caso seja premido o botão:
  a) *start/**stop*** – o cronómetro para.
  b) ***lap****/reset* – o cronómetro continua a contar (em *background*), mas o *display* deixa de ser atualizado (permitindo observar e registar um tempo parcial).
- Na situação **a)**, caso seja premido o botão:
  - ***start****/stop*, o cronómetro continua a contagem a partir do valor em que foi suspenso.
  - *lap/**reset***, o cronómetro é colocado a "00:00".
- Na situação **b)**, caso seja premido o botão ***lap****/reset*, o *display* volta a ser atualizado, permitindo visualizar novamente o cronómetro a ser incrementado.

6. O cronómetro é controlado pelo módulo *ControlUnit*, no qual está implementada uma máquina de estados finitos segundo o modelo de *Moore* e cujo diagrama se encontra na figura 2. Analise o respetivo código VHDL para verificar se corresponde ao comportamento descrito no diagrama de estados da figura 2.

7. Estabeleça a relação entre cada um dos estados do cronómetro descrito acima em linguagem natural, o diagrama de estados da figura 2 e o código VHDL fornecido.

Entrada/Saída	Sinal
Global Reset	reset
Start/Stop	statop
Lap/Reset	laprst
Counter Reset	cnrst
Counter Enable	cnenb
Register Enable	rgenb

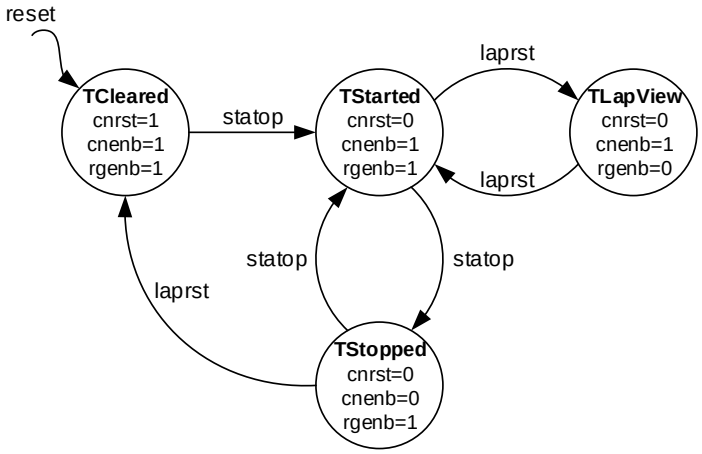

Figura 2 – Diagrama de estados do cronómetro fornecido (por simplicidade da figura apenas são apresentadas as transições relativas às mudanças de estado).

8. Apesar do cronómetro fornecido já estar funcional, este não implementa uma funcionalidade comum neste tipo de dispositivos:

   Quando o sistema se encontra na situação **b)** do ponto cinco e é premido o botão *start/**stop***, o cronómetro deve parar a contagem que está a fazer em *background*. O valor do cronómetro nesse instante deve ser visualizado nos *displays* após ser premido o botão ***lap**/reset*.

9. Altere o diagrama de estados fornecido de forma a adicionar a funcionalidade descrita no ponto anterior.

10. Edite o ficheiro VHDL do módulo *ControlUnit* de forma a refletir no código as alterações que introduziu no diagrama de estados no ponto anterior.

11. Volte a compilar o projeto e a testá-lo no *kit* DE2-115, tendo o cuidado de verificar todas as funcionalidades implementadas.

12. Realize o *bypass* entre o sinal de entrada e de saída de cada *Debouncer*, de forma a aplicar à unidade de controlo as entradas diretas provenientes dos botões de pressão.

13. Compile o projeto e a teste-o no *kit* DE2-115, verificando o seu comportamento incorreto devido ao *bounce* nos sinais de controlo.

14. Reponha as ligações iniciais (remova o *bypass*) dos *Debouncers* para que voltem a estar intercalados entre os sinais dos botões de pressão e as entradas da unidade de controlo.

[TPC] Altere o cronómetro para que passe a usar 6 displays, mostrando a informação no formato “mm:ss:cc”, em que “mm” (minutos) e “ss” (segundos) variam entre “00” e 59 e “cc” (centésimos de segundo) varia entre “00” e “99”.

Parte II

Pretende-se implementar uma MEF que permita controlar uma máquina de venda de bebidas, semelhante à que se descreve na aula TP7 com “ligeira” inflação no preço que passa agora para 0.90€.

1. Elabore o diagrama de estados/saídas segundo o modelo de Moore. Evite estados redundantes e associe um estado a cada quantia acumulada.

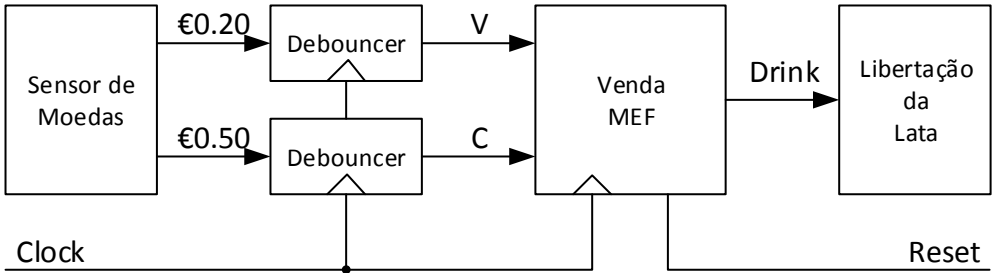

Figura 3 - Diagrama de blocos da máquina de venda de bebidas.

2. Crie um novo projeto no IDE Altera Quartus II para a FPGA do *kit* DE2-115 (o nome do projeto poderá ser *DrinksMachine*).

3. Crie um ficheiro VHDL "DrinksFSM.vhd" para modelar o funcionamento da MEF. Adote um estilo de codificação VHDL baseado numa descrição comportamental com dois processos interdependentes (um processo que atualiza o estado presente da MEF e outro onde se definem as atribuições do estado seguinte e das saídas).

4. Simule funcionalmente a MEF a partir dum ficheiro de estímulos ".VWF" criado na aplicação "Altera Quartus II".

5. Crie o ficheiro *top-level* para associar as entradas e saídas da MEF a pinos da FPGA. Sugere-se o seguinte mapeamento das entradas da MEF com as seguintes interfaces do *kit* DE2-115:

   Reset => KEY0

   V (€0.20) => SW1

   C (€0.50) => SW2

   Drink => LEDG0

Deverá também efetuar o *debounce* dos sinais provenientes do sensor de moedas. O componente a usar para este efeito deverá ser o *Debouncer* da parte I deste guião, uma vez que este garante a ativação da saída apenas durante um ciclo de relógio por cada "moeda inserida". De notar que o *debounce* adequado é fundamental para evitar múltiplas transições incorretas da máquina de estados (quer provocadas por *glitches* dos contactos mecanicos, quer por uma duração incorreta dos sinais de entrada da MEF).

6. Compile o projeto, programe a FPGA e teste-o.

A introdução duma moeda deve ser concretizada (simulada) do seguinte modo: **"SW1 on"->"SW1 off"** ou **"SW2 on"->"SW2 off"**. Como é evidente apenas um SW deverá estar ativo de cada vez.

[TPC] Altere o projeto de forma que seja possível visualizar em 3 displays de 7 segmentos a quantia acumulada no ato de compra de cada bebida. Recomendação fundamental: elabore um diagrama de blocos completo do sistema antes de escrever qualquer linha de código.

PDF criado em 22/04/2015 às 11:08:27