Laboratório de Sistemas Digitais

Trabalho Prático nº 9

Modelação, simulação e síntese de Máquinas de Estados Finitos - Modelo de *Mealy* MEFs comunicantes

Objetivos

- Domínio dos procedimentos fundamentais no processo de simulação, síntese, implementação em FPGA e teste de Máquinas de Estados Finitos (MEF) segundo o modelo de *Mealy*.
- Utilização de diagramas de estados e da linguagem VHDL para modelação ao nível comportamental de MEFs.
- Exploração e estensão de um exemplo de MEFs comunicantes.

Sumário

Este trabalho prático aborda a modelação comportamental, simulação e síntese de MEFs baseadas no modelo de *Mealy*. Pretende-se também introduzir as MEFs comunicantes através de um exemplo de aplicação. Na primeira parte apresenta-se um exercício típico de um detetor de sequências, como um caso muito frequente da utilização de MEFs, modelado segundo o modelo de *Mealy* (primeiro diretamente em VHDL a partir do diagrama de estados e recorrendo a dois processos interdependentes e posteriormente através da introdução do diagrama de estados com a aplicação *"State Machine Editor"*). Na segunda parte pretende-se estender o exemplo de MEFs comunicantes, para um sistema de semáforos apresentado na aula teórico-prática, corrigindo um problema comportamental intencionalmente deixado no projeto fornecido. Pretende-se também com este exemplo mostrar mais algumas funcionalidades e ferramentas do IDE *"Altera Quartus Prime"*.

Parte I

1. Elabore um diagrama de estados/saídas, segundo o modelo de *Mealy*, duma MEF, com uma entrada "*Xin"* e saída "*Yout"*, capaz de detetar a sequência 1001. Admita sobreposição tal como ilustrado no exemplo seguinte:

```
Xin:  001000100100011010010010001001100100010010010010
Yout: 000000000100000000010010000001000100000010010010
```

2. Abra a aplicação *"Altera Quartus Prime"* e crie um novo projeto para a FPGA Altera Cyclone IV EP4CE115F29C7. Designe o projeto e a entidade *top-level* como "SeqDetector".

3. Crie um ficheiro VHDL, chamado "SeqDetFSM.vhd", para modelar a MEF com base no diagrama de estados/saídas elaborado no ponto 1. Adote um estilo de codificação baseado numa descrição comportamental com dois processos interdependentes (um processo atualiza o estado atual da MEF e o outro determina o estado seguinte e efetua as atribuições à saída). Designe a arquitetura da entidade **"SeqDetFSM"** por **"MealyArch"**.

4. Crie um ficheiro VHDL, chamado "SeqDetFSM_Tb.vhd", com uma *testbench* para fazer a simulação comportamental do detetor de sequências. Use a aplicação *"Modelsim"* e o fluxo de compilação e simulação abordado no guião prático 7. Avalie adequadamente o funcionamento da MEF para a sequência de entrada apresentada no ponto 1. Observe na

ferramenta de simulação o comportamento de todos os sinais internos da arquitetura. Note que estes sinais permitem fazer o seguimento dos estados da MEF perante uma determinada sequência de entrada. Para tal deve selecionar no *"ModelSim"* o separador *"sim"* (ao lado separador *"library"*). De seguida, ao selecionar a arquitetura da unidade em simulação na janela *"objects*", surgirão também os sinais internos que poderão ser adicionados à janela de visualização das formas de onda.

5. Crie um ficheiro *top-level* em VHDL ("SeqDetector.vhd") para instanciar a MEF e associar a entrada e a saída da MEF a pinos da FPGA. Sugere-se o seguinte mapeamento com as interfaces do *kit*:

$$\text{Xin} \rightarrow \text{SW(0)} \qquad \text{Yout} \rightarrow \text{LEDR(0)}$$

O sinal de *clock* da MEF deverá ter uma frequência relativamente baixa (e.g. 0.2 Hz) para que o teste no *kit* seja mais simples. Para tal utilize um módulo de divisão da frequência do sinal de relógio configurado e instanciado adequadamente para gerar o sinal com a frequência pretendida, a partir do sinal de entrada de *clock* de 50 MHz disponível no *kit*. Disponibilize também o sinal de *clock* da MEF na saída "LEDG(0)" do *kit*.

6. Compile o projeto e teste-o no *kit* (não se esqueça de importar o ficheiro "DE2_115.qsf").

[TPC] Altere o diagrama de estados/saída elaborado no ponto 1 de forma a descrever o comportamento do detetor segundo o modelo de *Moore*. Repita todos os pontos desta parte do guião (agora na perspetiva do modelo de *Moore*). Acrescente à entidade "**`SeqDetFSM`**" a arquitetura "**`MooreArch`**" resultante da codificação em VHDL do diagrama de estados/saída segundo o modelo de *Moore*. Note que pode reutilizar o mesmo ficheiro *testbench* para fazer a simulação desde que instancie a arquitetura correta.

Parte II

1. O exemplo de MEFs comunicantes com base no sistema de semáforos apresentado na aula teórico-prática 8 e disponibilizado no *site* de LSDig possui um problema comportamental:

> Quando se comuta o funcionamento dos semáforos de "amarelo intermitente" para modo normal, ambos os semáforos transitam para vermelho, quando deveriam primeiro transitar ambos para "amarelo permanente" durante alguns segundos, e só depois ambos para vermelho.

2. Com o auxílio dos slides da aula teórico-prática 8, analise todo o código fonte do projeto fornecido, compile-o e teste-o no *kit*, tendo o cuidado de avaliar os vários modos de operação, estados e respetivas temporizações. Em particular, analise se as temporizações obtidas experimentalmente correspondem às especificadas no ficheiro "TrafficLightsFSM.vhd". Caso verifique algum desvio, identifique a respetiva causa e aponte eventuais soluções.

3. Altere o sistema fornecido de forma a introduzir a correção comportamental referida no ponto 1. Para tal deverá primeiro construir no seu *log book* o diagrama de estados/saídas corrigido e só depois editar o código VHDL em conformidade. Corrija também o eventual desvio das temporizações mencionado no ponto anterior.

4. Volte a compilar o projeto e a testá-lo no *kit*, tendo o cuidado de verificar o comportamento correto do sistema de semáforos para a alteração introduzida.

5. Através da análise do código fonte fornecido, explique porque razão após a programação da FPGA, o sistema inicia sempre no mesmo estado, independentemente da ativação (ou não) do sinal de *reset* principal do sistema.

6. Agora que o sistema se comporta corretamente, vamos concluir o guião com a exploração de algumas funcionalidades do IDE *"Altera Quartus Prime"*. Em primeiro lugar analise a estrutura hierárquica do projeto (após síntese) com o comando "*Tools → Netlist Viewers → RTL Viewer*", o qual deve abrir uma janela semelhante à da Figura 1.

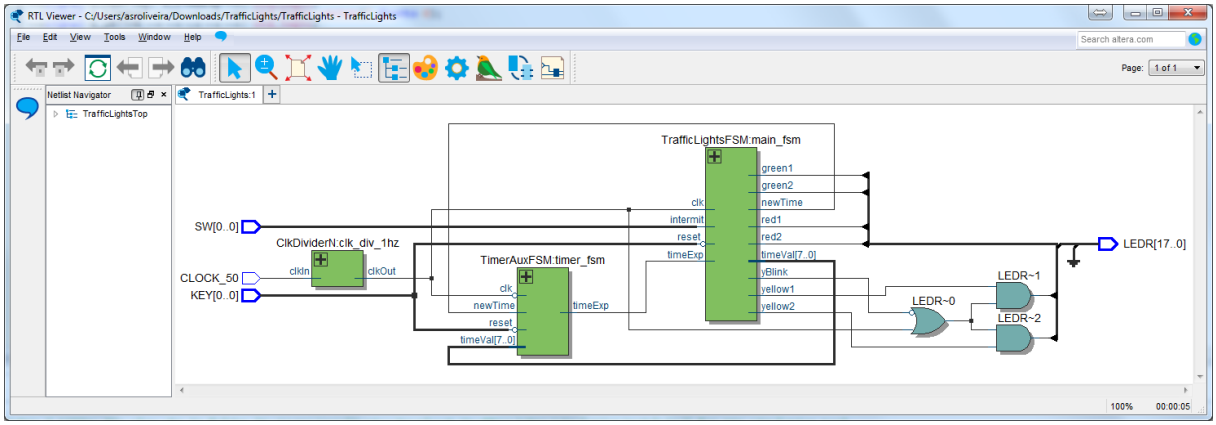

Figura 1- Diagrama esquemático do *top-level* do sistema.

7. De seguida visualize graficamente o diagrama de estados da MEF principal (*TrafficLightsFSM*) com o comando *"Tools → Netlist Viewers → State Machine Viewer"* que deverá abrir uma janela semelhante à da Figura 2.

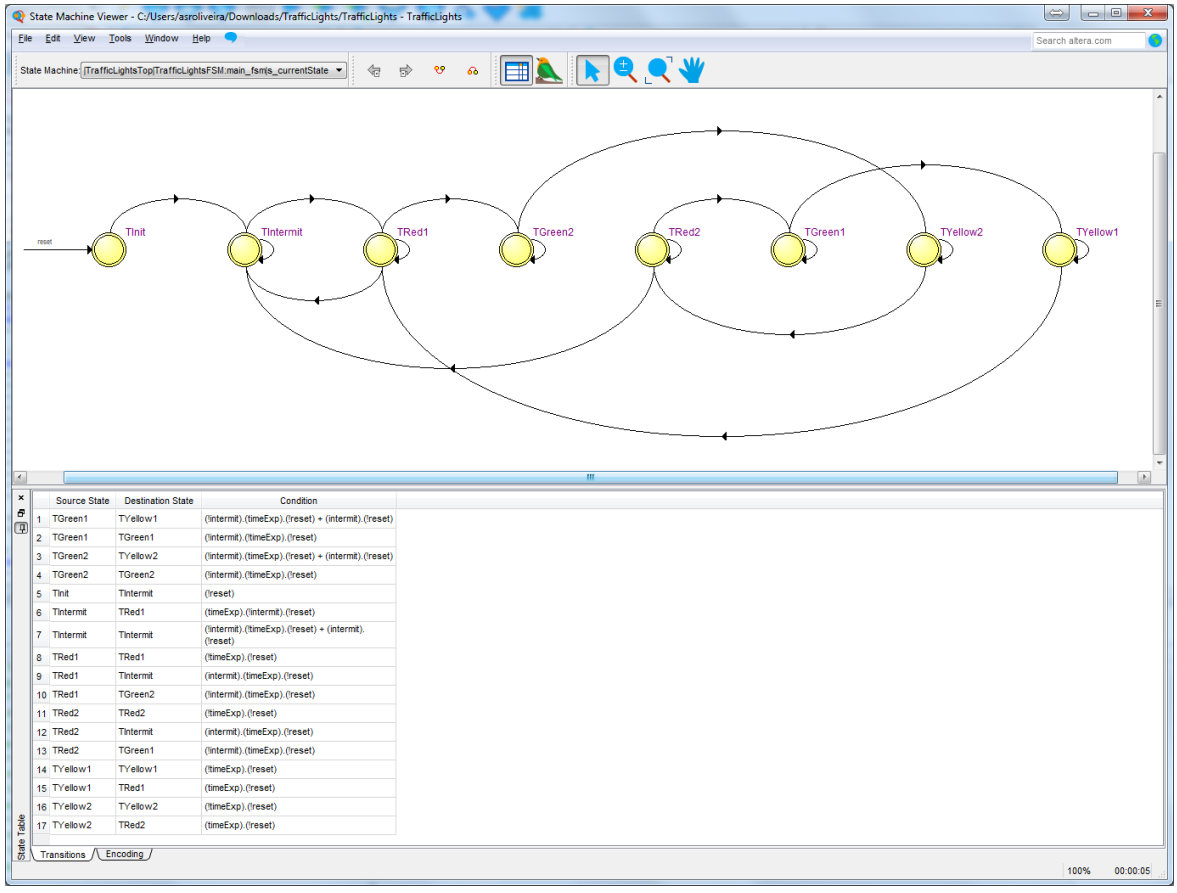

Figura 2 – Diagrama de estados e transições da MEF "TrafficLightsFSM".

8. A codificação de estados da MEF pode ser observada através do relatório mostrado na Figura 3 (disponível nos relatórios *"Analysis and Synthesis → State Machines"* do separador *"Compilation Reports"*). Nesta MEF é utilizada codificação *one-hot*. Analise e justifique o código binário atribuído a cada estado.

9. Finalmente as taxas de ocupação de cada um dos tipos de recursos da FPGA podem ser observadas após a implementação do sistema no *"Compilation Report"* (Figura 4). Constate as baixas taxas de ocupação da FPGA neste projeto.

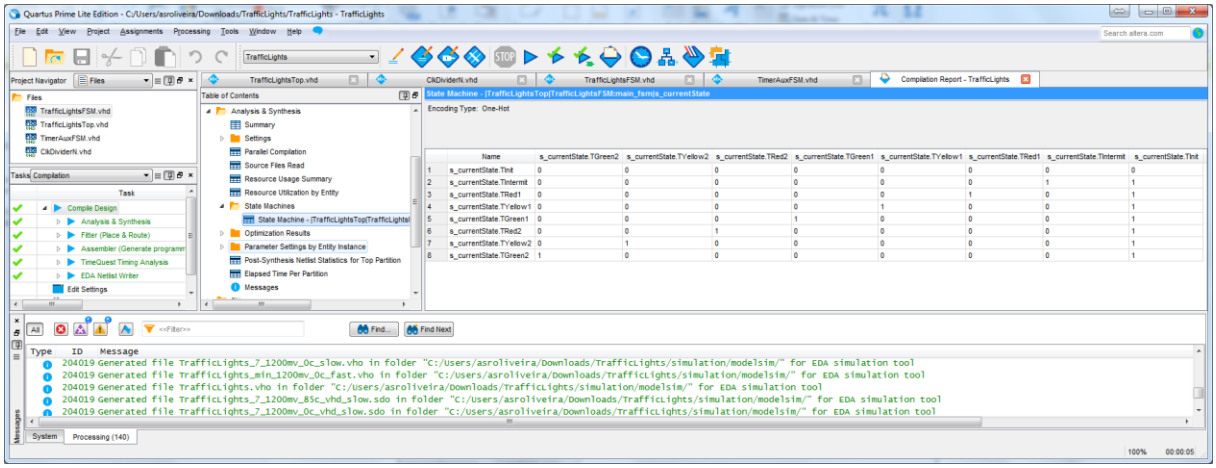

Figura 3 – Codificação de estados da MEF "TrafficLightsFSM".

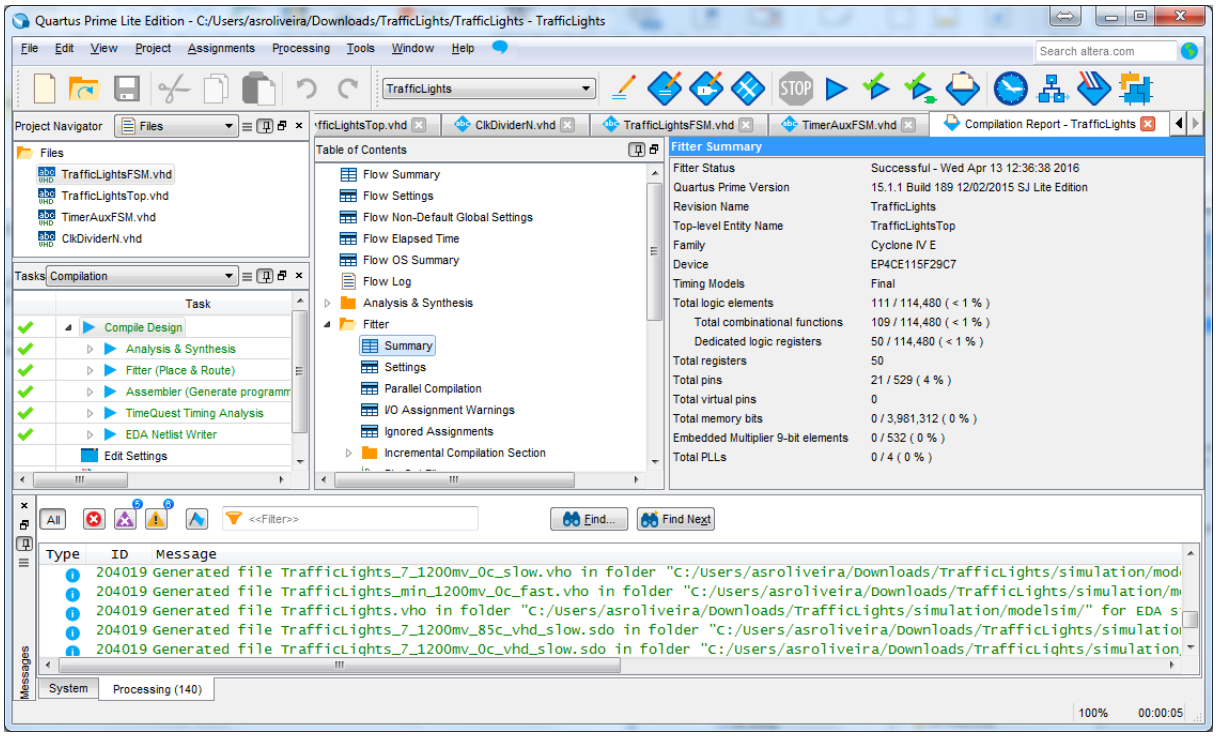

Figura 4 – Relatório sumário da ocupação de recursos da FPGA.

PDF criado em 29/03/2021 às 15:05:13