Laboratório de Sistemas Digitais

Trabalho Prático nº 9

Modelação, simulação e síntese de Máquinas de Estados Finitos - Modelo de *Mealy* *Datapath*, *Controlpath* e FSMs comunicantes
Aborgadens alternativas de modelação em VHDL de Máquinas de Estados Finitos

Objetivos

- Domínio dos procedimentos fundamentais no processo de simulação, síntese, implementação em FPGA e teste de Máquinas de Estados Finitos (MEFs) / *Finite State Machines (FSMs)* segundo o modelo de *Mealy*.
- Utilização de diagramas de estados e da linguagem VHDL para modelação ao nível comportamental de FSMs.
- Exploração e extensão de um exemplo de um sistema computacional com *datapath*, *controlpath* e FSMs comunicantes segundo o modelo de *Moore*.
- Familiarização com aborgadens alternativas de modelação em VHDL de FSMs.

Sumário

Este trabalho prático começa por abordar a modelação comportamental, simulação e síntese de FSMs baseadas no modelo de *Mealy*. Depois disso aborda também a arquitetura e projeto de sistemas computacionais compostos por um *datapath* e um *controlpath*. Na primeira parte apresenta-se um exercício típico de um detetor de sequências, como um caso muito frequente da utilização de FSMs, neste caso segundo o modelo de *Mealy*. Na segunda parte é explorado um exemplo de um sistema computacional que determina de forma iterativa o resultado da multiplicação de dois números sem sinal. Finalmente na terceira parte pretende-se estender o sistema da parte II de forma a incluir também a divisão iterativa de dois números sem sinal.

Parte I

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

```
Xin:  00100100100011001001001000100110010010011001100
Yout: 00000000100000001000010010000001000100000100100010100
```

2. Abra a aplicação *"Quartus Prime"* e crie um novo projeto para a FPGA Cyclone IV EP4CE115F29C7. Designe o projeto e a entidade *top-level* como "SeqDet1001".

3. Crie um ficheiro VHDL, chamado "SeqDetFSM.vhd", para modelar a FSM 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 FSM 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 Intel FPGA Starter Edition"* e o fluxo de compilação e simulação abordado no guião prático 7. Avalie adequadamente o funcionamento da FSM 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 FSM perante uma determinada sequência de entrada. Para tal deve selecionar no *"ModelSim"* o separador *"sim"* (ao lado do 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 (*"wave*").

5. Crie um ficheiro *top-level* em VHDL ("SeqDet1001.vhd") para instanciar a FSM e associar a entrada e a saída da FSM a pinos da FPGA. Sugere-se o seguinte mapeamento com as interfaces do *kit*:

Xin $\rightarrow$ SW(0) Yout $\rightarrow$ LEDR(0)

O sinal de *clock* da FSM 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 FSM na saída "LEDG(8)" do *kit*.

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

Na aula teórico-prática 8 é apresentado um exemplo de um sistema computacional que implementa de forma iterativa a multiplicação de dois números inteiros sem sinal. O respetivo código fonte é fornecido juntamente com este guião prático. Nesta parte do trabalho pretende-se que o analise e compreenda na integra. A arquitetura do projeto fornecido é apresentada na Figura 1, onde são mostrados os módulos principais, as suas interligações, os portos do *top-level* e respetivas conexões aos pinos da FPGA. Apesar da operação de divisão não estar implementada, o esqueleto do respetivo bloco "IterDivCore" já é fornecido.

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

2. Adicione ao projeto os ficheiros fonte contidos no ficheiro "IterMultDiv.zip" fornecido juntamente com este trabalho prático, nomeadamente: "IterMultDiv.vhd", "ClkDividerN.vhd", "MainCtrlUnit.vhd", "IterMultCore.vhd", "MultCtrlUnit.vhd", "ShiftRegN.vhd", "AddSubRegN.vhd", "IterDivCore.vhd" e "IterMultDiv_Tb.vhd".

Foram incluídos os seguintes aspetos relacionados com boas prática de projeto, não discutidos na aula teórica-prática:

- Os sinais de entrada provenientes dos interruptores (SWs) e botões (KEYs) são registados com o sinal CLOCK_50 de forma a manterem-se estáveis durante todo o ciclo do sinal de relógio do sistema e também para simplificar a análise temporal.

- O sinais de saída ligados a LEDs são também registados de forma a simplificar a análise temporal e cumprimento de restrições. Este aspeto será abordado posteriormente.

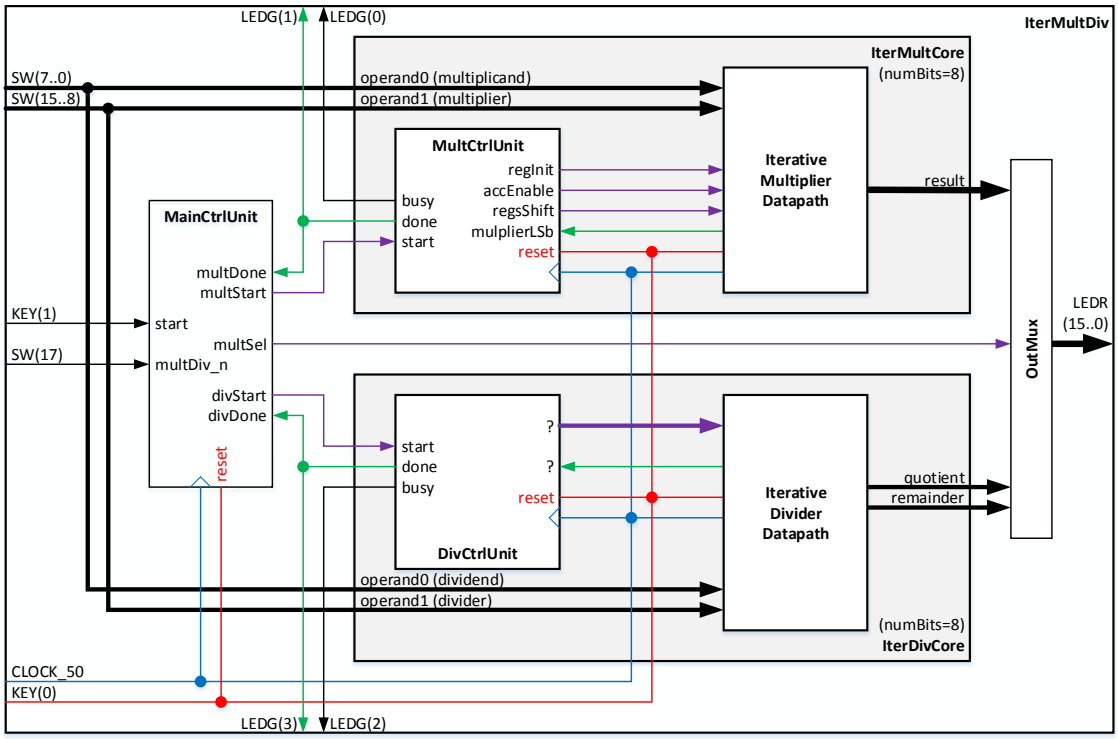

Figura 1 – Arquitetura do projeto "IterMultDiv" fornecido.

3. Analise todo o código fonte fornecido, incluindo os seguintes aspetos:

- De forma a construir um modelo VHDL compacto e legível de uma unidade de multiplicação, todos os componentes base são descritos comportamentalmente.
- Para que a unidade de multiplicação possa ser usada com operandos de diferentes tamanhos (fixado em *synthesis time*), o número de bits é parametrizável através de uma constante genérica no modelo em VHDL. No projeto fornecido esta constante (**`numBits`**) toma o valor 8 no *top-level*, sendo propagada hierarquicamente para todos os módulos que dela dependem.
- O módulo "MainCtrlUnit" corresponde à unidade de controlo principal, cujo diagrama de estados é apresentado na Figura 2. Esta FSM é descrita apenas com um processo em VHDL. Nesta abordagem de modelação todas as atribuições ao sinal de estado e às saídas são realizadas num único processo e síncronas com o sinal de relógio. Em função do estado atual e das entradas é determinado o estado seguinte e as respetivas saídas, ou seja a forma de pensar é: cada vez que ocorre um flanco ativo do sinal de relógio, se o estado atual é "este" e as entradas são "estas", então o estado seguinte e as respetivas saídas serão… . Esta abordagem só pode ser aplicada a máquinas de *Moore*, mas possui a vantagem das saídas serem registadas e atualizadas de forma síncrona com o *clock*, não apresentando *glitches*. Os *glitches* nas saídas nem sempre são críticos, mas podem ocorrer quando estas não são registadas e são produzidas por um circuito combinatório com diferentes atrasos entre as suas entradas e a saída.

A arquitetura do módulo "IterMultCore" constituído por um *datapath* e unidade de controlo (*controlpath*) é apresentada na Figura 3. Relativamente a este módulo:

- Os registos usados para armazenar o multiplicando e multiplicador são baseados no mesmo módulo (entidade-arquitetura) que implementa um registo de deslocamento muito flexível, mas em que algumas das entradas estão ligadas a níveis lógicos constantes. Apesar de isto poder à primeira vista parecer uma utilização pouco eficiente de recursos da FPGA, tal não acontece porque a ferramenta de síntese otimiza o circuito gerado, removendo todos os componentes (lógica) não usados, redundantes ou cujos sinais não variam.
- O registo de resultado juntamente com o somador estão integrados num acumulador também descrito comportamentalmente.

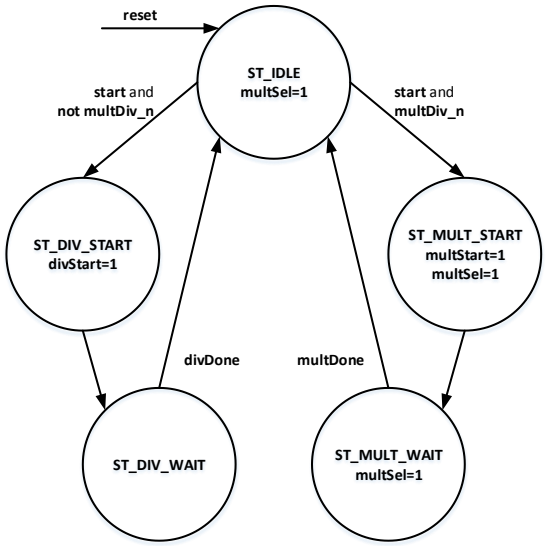

Figura 2 – Diagrama de estados e das saídas da FSM "MainCtrlUnit".

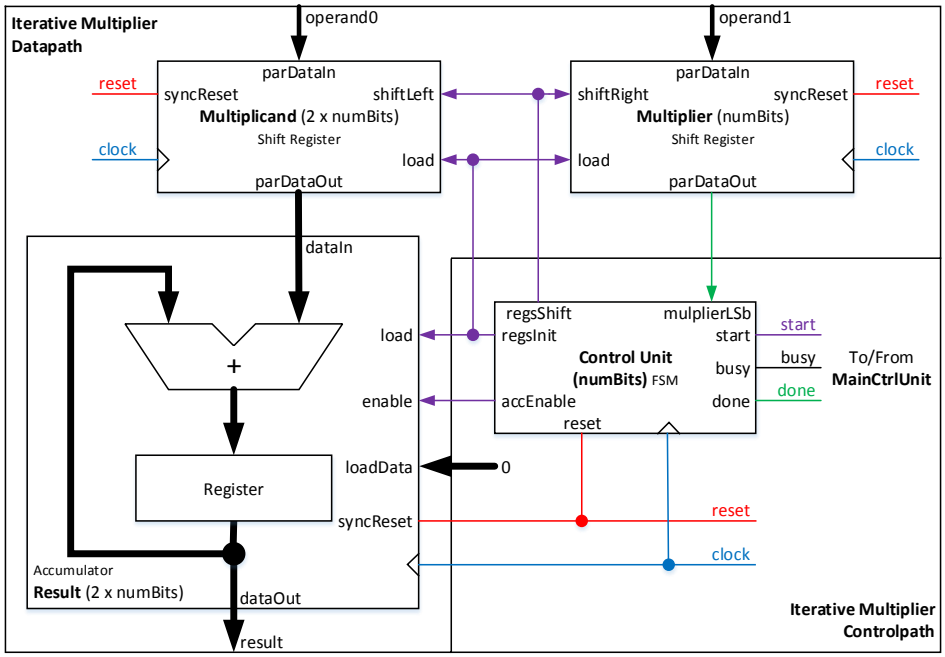

Figura 3 – Arquitetura do módulo "IterMultCore" constituído por um *datapath* e unidade de controlo (*controlpath*).

São fornecidas duas implementações (arquiteturas) da unidade de controlo da multiplicação (módulo "MultCtrlUnit"), modelada segundo uma máquina de *Moore* e cujo diagrama de estados é apresentado na Figura 4:

- Uma arquitetura com 4 processos em VHDL (apresentada na aula teórico-prática e designada **`Behavioral_4Proc_AsyncOut`**), correspondendo ao registo de estado, ao contador do número de iterações e aos circuitos combinatórios que determinam o estado seguinte e as saída.
- Uma arquitetura com apenas 1 processo em VHDL (com uma abordagem semelhante ao módulo "MainCtrlUnit" e designada **`Behavioral_1Proc_SyncOut`**), em que todas as atribuições (estado e saídas) são síncronas com o sinal de *clock*.

As unidades de controlo "MainCtrlUnit" e "MultCtrlUnit" podem ser vistas como duas FSMs comunicantes: a primeira para controlo global do sistema e a segunda para controlo das operações específicas da multiplicação.

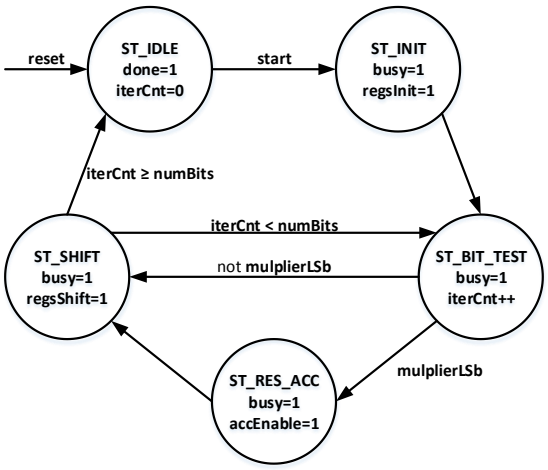

Figura 4 – Diagrama de estados e das saídas da FSM "MultCtrlUnit".

4. Selecione o ficheiro "IterMultDiv.vhd" como o *top-level* do projeto.

5. Efetue a simulação do sistema com a aplicação *"ModelSim ALTERA STARTER EDITION"* e usando a *testbench* fornecida. Realize todos os passos necessários tal como indicado no guião prático 7. Adicione à janela *"Wave"* do "*ModelSim*" todos os sinais dos módulos "IterMultDiv", "MainCtrlUnit", "IterMultCore" e "MultCtrlUnit". Analise cuidadosamente os resultados da simulação, verificando todos os passos realizados até à obtenção do resultado final.

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

7. Verifique o funcionamento do sistema com diversos valores de multiplicando e multiplicador (**sugestão:** utilize os mesmos valores definidos na *testbench* fornecida). As teclas e interruptores do *kit* para interação com o sistema são as seguintes:

- KEY0 – *reset*
- KEY1 – *start*
- SW0 – seleção entre divisão (0) ou multiplicação (1)

8. Intercale no *top-level* um divisor de frequência entre o sinal **`CLOCK_50`** e o **`s_globalClk`** de forma a que o segundo possua uma frequência de 1Hz.

9. Volte a compilar o projeto e a testá-lo no *kit*, observando, para diversos valores dos operandos, a forma como o resultado vai sendo gerado ao longo do tempo. Recomenda-se que acrescente os portos necessários para visualizar em LEDs o sinal **`s_globalClk`** e o estado da FSM "MultCtrlUnit".

[TPC] Remova o divisor de frequêcia adicionado no ponto 6 de forma a que todo o circuito volte a funcionar a 50 MHz. Utilize a ferramenta *"SignalTap II Logic Analyser"*de forma a capturar em tempo real os sinais mais relevantes do circuito (operandos, resultado, sinais de controlo e de estado e sinais de saída dos componentes do *datapath*).

Parte III

O sistema explorado na parte II não implementa a divisão de quantidades *unsigned* de forma iterativa, apesar da arquitetura global ter sido definida já a pensar nessa funcionalidade. Assim, nesta parte pretende-se que estenda o projeto fornecido de forma a incluir a divisão de números inteiros sem sinal. Após o estudo do algoritmo iterativo da divisão deverá projetar, integrar e testar um *datapath* e uma unidade de controlo (*controlpath*) para o módulo que implementa a divisão.

1. Estude os slides sobre o algoritmo iterativo de divisão fornecidos juntamente com este guião.

2. Conceba um *datapath* capaz de realizar as operações de adição, subtração e deslocamento necessários para implementação do algoritmo iterativo da divisão (**sugestão:** utilize como blocos base os mesmos componentes usados na multiplicação – registos de deslocamento e acumulador). Desenhe no seu *log book* o *datapath* à medida que o vai concebendo, indicando todos os módulos, portos e sinais relevantes.

3. Modele em VHDL o *datapath* projetado no ponto anterior usando o esqueleto fornecido no ficheiro "IterDivCore.vhd".

4. Construa o diagrama de estados e saídas que descreve a unidade de controlo da divisão segundo o modelo de *Moore*.

5. Modele em VHDL a FSM descrita no ponto anterior seguindo uma abordagem baseada em quatro processos, correspondendo ao registo de estado, ao contador do número de iterações e aos circuitos combinatórios que determinam o estado seguinte e as saída.

6. Simule com a aplicação *"ModelSim ALTERA STARTER EDITION"* a unidade de divisão completa, adaptando a *testbench* "IterMultDiv_Tb.vhd" fornecida.

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

8. Verifique o funcionamento do sistema com diversos valores de dividento, divisor, multiplicando e multiplicador (**sugestão:** utilize os mesmos valores definidos na *testbench*). As teclas e interruptores do *kit* para interação com o sistema são as mesmas da parte II.

[TPC] Modele em VHDL a unidade de controlo da divisão com apenas um processo, de forma análoga à arquitetura **`Behavioral_1Proc_SyncOut`** do módulo "MultCtrlUnit". Simule a unidade de divisão com esta implementação da respetiva unidade de controlo e teste em FPGA o sistema completo.

PDF criado em 04/04/2018 às 16:42:40