Laboratório de Sistemas Digitais

Trabalho Prático nº 6

Modelação em VHDL e implementação de registos e módulos combinatórios de deslocamento

Objetivos

- Modelação em VHDL, simulação, implementação em FPGA e teste de circuitos combinatórios e sequenciais elementares de deslocamento e rotação.

Sumário

Um bloco funcional de deslocamento permite deslocar um vetor um certo número de *bits*. O deslocamento pode efetuar-se no sentido do *bit* mais significativo (*Most Significant bit* – MSb) – *à esquerda* – ou no sentido do *bit* menos significativo (*Least Significant bit* – LSb) – *à direita*. Um bloco deste tipo pode ser realizado de forma puramente combinatória; neste caso, a saída corresponde ao deslocamento de um vetor colocado à entrada. Pode também ser realizado por um circuito sequencial; neste caso, a saída é deslocada relativamente ao seu próprio valor no anterior ciclo de relógio. Quando o vetor é interpretado como uma quantidade numérica inteira (com ou sem sinal), um deslocamento de *n bits* à esquerda realiza uma operação de multiplicação por 2^n; um deslocamento de *n bits* à direita realiza uma operação de divisão por 2^n. No caso da multiplicação, pressupõe-se que os *bits* acrescentados à direita são '0'. No caso da divisão, pressupõe-se que os *bits* acrescentados à esquerda replicam o MSb original (para quantidades em complemento para dois), ou são '0' (para quantidades representadas como inteiros positivos – sem sinal).

Este trabalho prático está dividido em três partes. Na primeira, é implementado um bloco de deslocamento sequencial (*shift-register*) do tipo *serial in / parallel out* com dimensão parametrizável. A segunda parte é dedicada à descrição comportamental e implementação de um *shift-register* genérico, com *parallel load*, capaz de efetuar operações de deslocamento e rotação à esquerda e à direita, e suportando igualmente o deslocamento aritmético à direita e à esquerda. A terceira parte é dedicada à implementação de módulos combinatórios de deslocamento.

Parte I

1. Crie um novo projeto para a FPGA Cyclone IV EP4CE115F29C7. Poderá designar o projeto e a entidade *top-level* como "ShiftRegister_Demo".

2. Descreva em VHDL um s*hift register* de 4 *bits* com deslocamento à esquerda, de acordo com a interface representada na Figura 1. Guarde o seu código num ficheiro com o nome "ShiftRegister4.vhd".

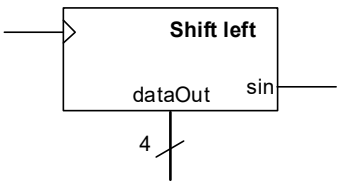

Figura 1 – Interface do módulo "ShiftRegister4".

3. Efetue a simulação do componente modelado.

4. Copie o ficheiro "ShiftRegister4.vhd" para "ShiftRegisterN.vhd" e altere-o de modo a que a dimensão do *shift register* seja parametrizável, definida com uma constante genérica "**`size`**" a fixar aquando da instanciação do registo. Por omissão, "**`size`**" poderá assumir o valor 4.

5. Crie um novo ficheiro VHDL, chamado "ShiftRegister_Demo.vhd", que instancie o *shift register* parametrizável com uma dimensão de 8. Para tal, deve atribuir esse valor à constante genérica "**size**" usando a construção **generic map**. Ligue a entrada "**clk**" à saída de um divisor de frequência que gere um relógio com a frequência de 1Hz e as restantes entradas a interruptores; ligue as saídas a LED. O divisor de frequência pode ser baseado na instanciação do módulo apresentado na Figura 2, ligando a entrada "**clkIn**" ao pino "CLOCK_50" da FPGA e atribuindo um valor adequado à constante "**divFactor**".

6. Efetue a síntese e implementação do projeto, programe a FPGA e teste o funcionamento do componente.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;

entity ClkDividerN is
  generic(divFactor : positive := 10);
  port(clkIn        : in  std_logic;
       clkOut       : out std_logic);
end ClkDividerN;

architecture Behavioral of ClkDividerN is

  subtype TCounter is natural range 0 to divFactor - 1;
  signal s_divCounter : TCounter := 0;

begin
  assert(divFactor >= 2);

  process(clkIn)
  begin
    if (rising_edge(clkIn)) then
      if (s_divCounter >= (divFactor - 1)) then
        clkOut       <= '0';
        s_divCounter <= 0;
      else
        if (s_divCounter = (divFactor / 2 - 1)) then
          clkOut <= '1';
        end if;
        s_divCounter <= s_divCounter + 1;
      end if;
    end if;
  end process;
end Behavioral;
```

Figura 2 -Descrição em VHDL de um divisor de frequência parametrizável.

[TPC] Copie o ficheiro "ShiftRegisterN.vhd" para o ficheiro "ShiftRegisterLoadN.vhd" e altere-o de modo a modelar um s*hift register* de N *bits* com entradas de carregamento paralelo, de acordo com o bloco lógico da Figura 3, em que as entradas de "**reset**" e "**load**" são síncronas. Efetue a simulação, instanciação, implementação e teste do novo componente.

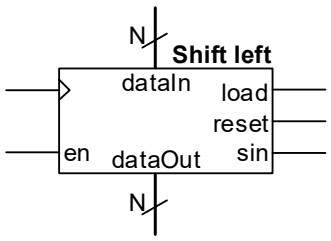

Figura 3 – Interface do módulo "ShiftRegisterLoadN".

Parte II

1. Crie um projeto para a FPGA Cyclone IV EP4CE115F29C7. Poderá designar o projeto e a entidade *top-level* como "SeqShiftUnit_Demo".

2. Dentro deste projeto crie um novo ficheiro VHDL com o nome "SeqShiftUnit.vhd" e transcreva para este ficheiro o código que se apresenta na Figura 4. Complete-o de forma que, para além do *parallel load*, possa efectuar as seguintes operações:

- *shift left*/*right* lógico
- *shift left*/*right* aritmético
- *rotate left/right*

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity SeqShiftUnit is
   port(clk      : in  std_logic;
        dataIn   : in  std_logic_vector(7 downto 0);
        siLeft   : in  std_logic;
        siRight  : in  std_logic;
        loadEn   : in  std_logic;
        rotate   : in  std_logic;
        dirLeft  : in  std_logic;
        shArith  : in  std_logic;
        dataOut  : out std_logic_vector(7 downto 0));
end SeqShiftUnit;

architecture Behavioral of SeqShiftUnit is

  signal s_shiftReg : std_logic_vector(7 downto 0);

begin
  process(clk)
  begin
    if (falling_edge(clk)) then
       if (loadEn = '1') then
          s_shiftReg <= dataIn;

       elsif (rotate = '1') then
          if (dirLeft = '1') then
            s_shiftReg <= s_shiftReg(6 downto 0) & s_shiftReg(7);
          else
            s_shiftReg <= s_shiftReg(0) & s_shiftReg(7 downto 1);
          end if;

       elsif (shArith = '1') then
       --
       --   Complete o código aqui
       --
       end if;
    end if;
  end process;

  dataOut <= s_shiftReg;
end Behavioral;
```

Figura 4 – Esqueleto da descrição em VHDL do módulo "SeqShiftUnit".

NOTA: "**siLeft**" e "**siRight**" são as entradas série para os *shifts* lógicos à esquerda e à direita, respetivamente (ver tabela do lado). Relembre a matéria de ISDig e das aulas TP de LSD, e anote no seu *log book* uma breve descrição funcional de cada uma das operações indicadas neste ponto.

Shift	À Esquerda	À Direita
Lógico	Entra "**siLeft**" do lado direito	Entra "**siRight**" do lado esquerdo
Aritmético	Entra '0' do lado direito	Entra (replica) MSb do lado esquerdo

3. Sintetize o circuito criado, verifique a sua correção sintática e efetue a simulação do componente modelado.

4. Crie um novo ficheiro, chamado "SeqShiftUnit_Demo.bdf", para instanciar de forma gráfica o *shift register* que descreveu. Ligue a entrada "**clk**" à saída do divisor de frequência que usou na *Parte I*, as restantes entradas a interruptores e as saídas a LED à sua escolha. Teste o funcionamento do circuito na sua placa de desenvolvimento.

Parte III

1. Crie um projeto para a FPGA Cyclone IV EP4CE115F29C7. Poderá designar o projeto e a entidade *top*-level como "CombShiftUnit_Demo".

2. Dentro deste projeto, crie um novo ficheiro VHDL com o nome "CombShiftUnit.vhd".

3. Escreva o código necessário para instanciar um módulo de deslocamento **combinatório** de 8 *bits*, capaz de efetuar o mesmo tipo de operações que o *shift register* da *Parte II.*

NOTA: utilize as seguintes funções, definidas na *package* **NUMERIC_STD** da biblioteca **IEEE**, para realizar as operações de deslocamento e rotação de forma combinatória:

Deslocamentos lógicos
```
function SHIFT_LEFT (ARG: UNSIGNED; COUNT: NATURAL) return UNSIGNED;
function SHIFT_RIGHT (ARG: UNSIGNED; COUNT: NATURAL) return UNSIGNED;
```
Deslocamentos aritméticos
```
function SHIFT_LEFT (ARG: SIGNED; COUNT: NATURAL) return SIGNED;
function SHIFT_RIGHT (ARG: SIGNED; COUNT: NATURAL) return SIGNED;
```
Rotações
```
function ROTATE_LEFT (ARG: UNSIGNED; COUNT: NATURAL) return UNSIGNED;
function ROTATE_RIGHT (ARG: UNSIGNED; COUNT: NATURAL) return UNSIGNED;
```

4. Sintetize o circuito criado, verifique a sua correção sintática e efetue a simulação do componente modelado.

5. Crie um novo ficheiro VHDL, chamado "CombShiftUnit_Demo.vhd", onde deverá instanciar o *shifter* anteriormente modelado e associar os respetivos portos aos seguintes pinos concretos da FPGA do *kit* de desenvolvimento:

dataIn->SW[7..0] **rotate->**KEY[0] **dirLeft->**KEY[1]
shArith->KEY[2] **shAmount->**SW[17:15] **dataOut->**LEDR[7:0]

6. Efetue a síntese e compilação do projeto, programe a FPGA e teste o funcionamento do componente.

PDF criado em 27/02/2024 às 13:03:31