Arquitecturas Avançadas de Computadores
2009/10

Guia de Laboratório II

Simulação ao nível estrutural do processador μRISC

Demonstração: 11/5/2010 e 12/5/2010

Departamento de Engenharia Electrotécnica e de Computadores
Instituto Superior Técnico
Março de 2010

1 Objectivos

O objectivo deste trabalho é realizar uma simulação ao nível estrutural dos circuitos de **controlo e dados** do processador `uRISC`. As especificações deste processador foram definidas no enunciado do trabalho de laboratório anterior.

Neste trabalho, deverá continuar a admitir que cada instrução demora 4 ciclos de relógio a ser executada, ou seja, que o processador não usa um *pipeline*. Neste trabalho, fica ao critério dos alunos a adopção ou não da técnica de saltos retardados. Chama-se a atenção de que no trabalho de laboratório seguinte vai introduzir técnicas de resolução de conflitos de dados e controlo por *hardware* na descrição estrutural do processador desenvolvida neste segundo trabalho.

2 Simulação a nível estrutural

Tal como no 1^o trabalho, o simulador estrutural pode ser desenvolvido em VHDL ou em C, podendo-se usar apenas as funções e componentes apresentadas na tabela 1.

C/[VHDL]	function	atraso	custo
XO2(a,b,z)	$z = a \oplus b$	5	6
XNO2(a,b,z)	$z =!(a \oplus b)$	5	6
NA2(a,b,z)	$z =!(a\&b)$	3	4
NA3(a,b,d,z)	$z =!(a\&b\&d)$	4	6
A2(a,b,z)	$z = a\&b$	5	6
A3(a,b,d,z)	z=a&b&d	6	8
NO2(a,b,z)	$z =!(a\|b)$	3	4
NO3(a,b,d,z)	$z =!(a\|b\|d)$	4	6
O2(a,b,z)	$z = a\|b$	5	6
O3(a,b,d,z)	$z = a\|b\|d$	6	8
NOT[1](a,z)	$z =!a$	2	2
MUX2(sel,a0,a1,z)	$z = (sel)?(a1) : (a0)$	6	12
LATCH_EN(en[able],d,q)	if $(en == 1\&\&q! = d)q = d$	3	12

Tabela 1: Funções/componentes a utilizar na simulação estrutural.

Poderão ser usadas primitivas mais complexas, desde que sejam definidas a partir das primitivas descritas acima.

Para quem utilizar programação sequencial, a simulação a nível estrutural deverá ser feita em dois passos, por cada ciclo de relógio:

1. Uma avaliação, pela ordem devida, de toda a lógica combinatória definida com as primitivas enumeradas acima.

2. Uma avaliação da lógica sequencial, onde os valores são guardados nos registos.

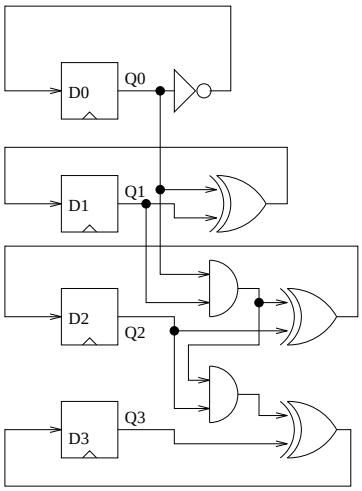
Para a avaliação do trabalho, será considerada fundamental a frequência de funcionamento do processador, atendendo-se tanbém à área de circuito ocupada.

3 Relatório

O relatório deverá incluir um esquema dos circuitos de dados e de controlo do processador, e descrever brevemente, justificando, as opções de projecto que foram tomadas. Deverá tambem incluir estatistícas com os tempos de execução dos programas utilizados para a demonstração do funcionamento do simulador. O relatório não deve exceder as 15 páginas, com excepção dos apêndices.

4 Exemplo

Considere o exemplo de um contador, cujo esquema é o seguinte:



Este contador pode ser “realizado” em C com o seguinte código:

```
#include <stdio.h>
#include "defines.h"

/* Valores de registos devem ser inicializados */

 static bit Q[4] ;
 static bit D[4] ;

void init_ffs() {
  int i;

  for(i=0; i<4; i++)
    Q[i] = ZERO();
}

 void
 contador_estrutural_combinatorio()
 {
   bit r0,r1;
   int i; /* Variavel de iteracao */
```

```
   bit one;

   one = ONE(); /* Inicializacao de bits - primitivas ONE() e ZERO() */

   XO2(Q[0],one,D[0]);

   XO2(Q[0],Q[1],D[1]);

   A2(Q[1],Q[0],r0);
   XO2(Q[2],r0,D[2]);

   A2(Q[2],r0,r1);
   XO2(Q[3],r1,D[3]);

   for(i=0; i<4; i++) {
     LATCH\_EN(1,D[i],Q[i]);
   }
}


 void
 contador_estrutural(void)
 {
   contador_estrutural_combinatorio();
   activate_clock();    /* Aqui e' dado o impulso de relogio  */
 }


main() {
  init_ffs();
  while (1) {
    contador_estrutural();

    printf("%d %d %d %d\n",
           IS_ONE(Q[3]),
           IS_ONE(Q[2]),
           IS_ONE(Q[1]),
           IS_ONE(Q[0]));
  }
}
```

ou em VHDL com o seguinte código (contador.vhd):

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

entity contador is
        port(
                reset : in STD_LOGIC;
                clk : in STD_LOGIC;
                OutPut : out STD_LOGIC_VECTOR(3 downto 0)
                );
end contador;

architecture contador of contador is
        signal Q, D : STD_LOGIC_VECTOR(3 downto 0);
        signal one, zero, r0, r1 : STD_LOGIC;
        signal enable : STD_LOGIC;

begin
        one <= '1';
        zero <= '0';
        enable <= '1';

        xor21 : xo2
        port map (
                A => Q(0),
                B => one,
                Z => D(0)
                );
        xor22 : xo2
        port map (
                A => Q(0),
                B => Q(1),
                Z => D(1)
                );
        a21 : a2
        port map (
```

```
        A => Q(1),
        B => Q(0),
        Z => r0
        );
xor23 : xo2
port map (
        A => Q(2),
        B => r0,
        Z => D(2)
        );
a22 : a2
port map (
        A => Q(2),
        B => r0,
        Z => r1
        );
xor24 : xo2
port map (
        A => Q(3),
        B => r1,
        Z => D(3)
        );

FF_Out0 : ffe port map (
        CLK => CLK,
        Reset => reset,
        Enable => enable,
        D => D(0),
        Q => Q(0)
 );
FF_Out1 : ffe port map (
        CLK => CLK,
        Reset => reset,
        Enable => enable,
        D => D(1),
        Q => Q(1)
 );
FF_Out2 : ffe port map (
        CLK => CLK,
        Reset => reset,
```

```
                Enable => enable,
                D => D(2),
                Q => Q(2)
         );
        FF_Out3 : ffe port map (
                CLK => CLK,
                Reset => reset,
                Enable => enable,
                D => D(3),
                Q => Q(3)
         );

OutPut <= Q;
end contador;
```

e o *Teste Bench* (contador_TB.vhd):

```
library ieee;
use work.gates.all;
use ieee.std_logic_1164.all;

entity contador_tb is
end contador_tb;

architecture TB_ARCHITECTURE of contador_tb is
        -- Component declaration of the tested unit
        component contador
        port(
                reset : in std_logic;
                clk : in std_logic;
                OutPut : out std_logic_vector(3 downto 0) );
        end component;

        -- Stimulus signals - signals mapped to the input and inout ports of tested
ty
        signal reset : std_logic;
        signal clk : std_logic;
        -- Observed signals - signals mapped to the output ports of tested entity
```

```
        signal OutPut : std_logic_vector(3 downto 0);

begin

        -- Unit Under Test port map
        UUT : contador
                port map (
                        reset => reset,
                        clk => clk,
                        OutPut => OutPut
                );

        CLK_GEN: process
        begin
                clk <= '0';
                wait for 20ns;
                clk <= '1';
                wait for 20ns;
        end process;

        RST_GEN: process
        begin
                reset <= '0';
                wait for 30ns;
                reset <= '1';
                wait for 40ns;
                reset <= '0';
                wait;
        end process;

end TB_ARCHITECTURE;

configuration TESTBENCH_FOR_contador of contador_tb is
        for TB_ARCHITECTURE
                for UUT : contador
                        use entity work.contador(contador);
                end for;
        end for;
end TESTBENCH_FOR_contador;
```