Universidade de Aveiro

Modelação de um processador simplificado

Laboratórios de Sistemas Digitais

31-05-2020

Rafael Santos – 98466

Introdução

No âmbito da unidade curricular de laboratório de sistemas digitais, foi proposto a realização de uma modelação de um processador simplificado.

Este trabalho tem como objetivo avaliar e testar competências adquiridas pelos alunos durante as aulas de LSD ao longo do semestre.

Fase 1-Implementação e teste da unidade de execução

A fase 1 é constituída por diversos componentes tais como: PC (Program counter), uma memória de instrução (IMemory), SignExtend, Registers,mutliplexers e uma memória de dados(DMemory).

O PC é apenas um contador que vai guardar o endereço da próxima instrução a ser executada. Se a sua entrada En estiver ativa então o processador funcionará, caso contrário não funcionará.

A memória de instrução (IMemory) é uma memória rom. Este componente vai ‘’lendo’’ a saída do PC e vai executando as instruções conforme previamente estabelecido no código.

Conclui-se assim a fase fetch da unidade de execução.

A próxima fase é chamada de Decode, onde vão ser lidos os conteúdos dos registos.

Esta fase inclui um Mux2_1 e um registo. Registo esse constituído também por um decoder 3_8 e dois Mux8_1.

No registo, RA1 vai funcionar como select de um Mux8_1 e RA2 do outro. WD tem a função de receber e armazenar dados vindos da DMemory nos seus registos. Não é permitido ao registo 0 armazenar qualquer tipo de dados. O decoder vai funcionar como enable dos registos descodificando a entrada de modo a ativar o registo que é pretendido.

A próxima fase é o execute que envolve o bloco ALU.

Este realiza as várias operações dadas entre op1 e op2. As operações a realizar serão dadas pelo ALUOp (sinal enviado pela unidade de controlo). op1 será a saída RD1 do register e op2, dependendo da instrução opCode, poderá ser RD2 ou então RD da memória de instrução. Nesta fase participa também o bloco SignExtend que transforma os 7 bits menos significativos, que saíram do RD da IMemory ,em

8 bits para que possa ser usado na DMemory.

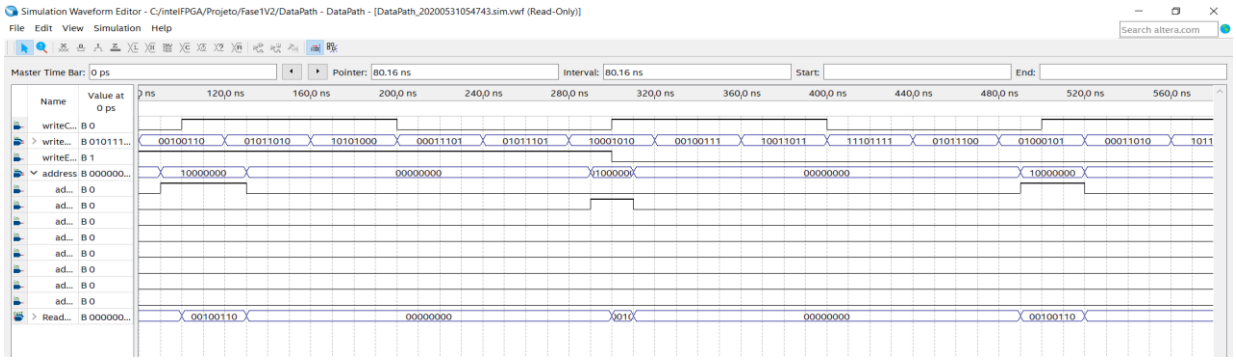

Fig1- waveform DMemory

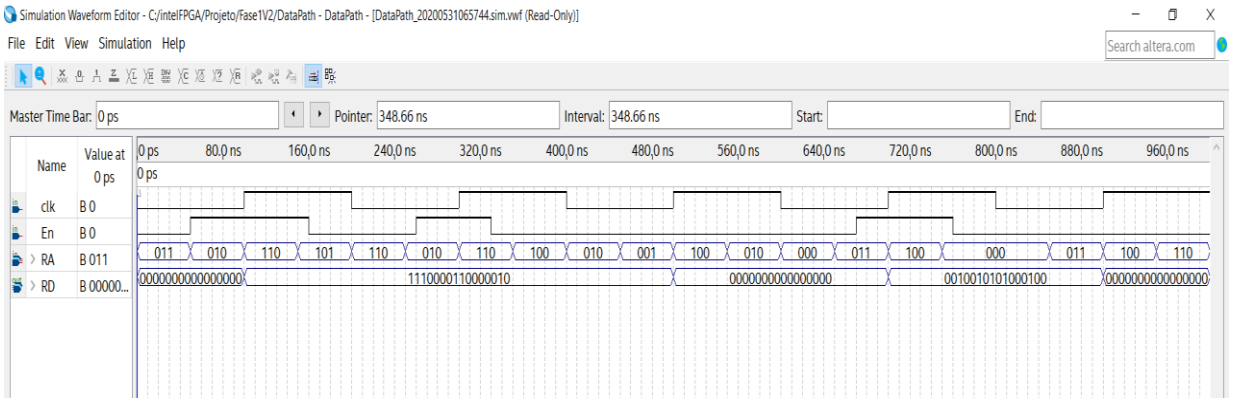

Fig2 – Waveform IMemory

As últimas duas fases são RegUpdate e WriteMem. WriteMem apenas será atingido aquando a instrução SW.

O RegUpdate é o estado que corresponde à atualização do registo enquanto que o WriteMem corresponde à escrita de dados na DMemory.

Fase 2 – Implementação e teste da unidade de controlo

A fase 2 tem como objetivo controlar o sistema.

Possui duas entradas opCode e func que vão determinar o rumo do sistema. Possui outras duas, uma de clock e uma de reset, que vão determinar quando o sistema está ativo ou não.

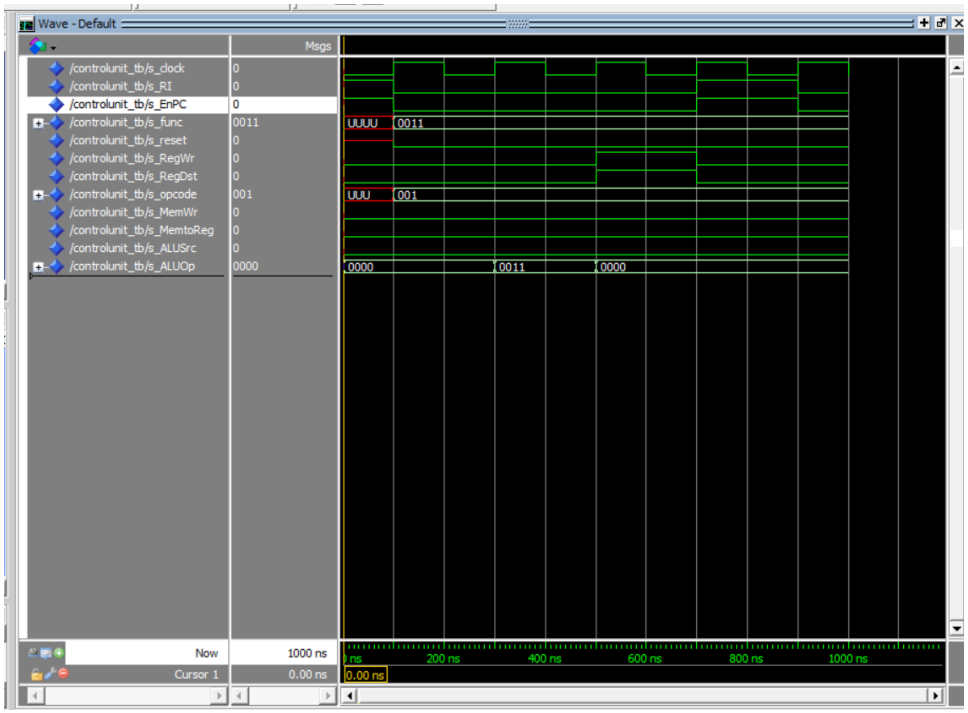

Fig3 – TestBench da Control Unit.

Na figura acima está demonstrado a minha TestBench da ControlUnit, não está correta visto não encontrar os erros.

Fase 3 – Interligação e teste do processador completo

Nesta fase já só falta interligar o control unit e o datapath.

Devemos instanciar num único ficheiro vhdl tanto o datapath como o control unit e atribuir os sinais aos respetivos outputs e inputs de todos os componentes.

```
begin
   Control_Unit : entity work.ControlUnit(behavioral)
      port map(clk => clk,
               reset => reset,
               opCode => IMemOut(15 downto 13),
               func =>IMemOut(3 downto 0),
               RI =>s_RI,
               RegWr => s_RegWr,
               RegDst => s_RegDst,
               ALUSrc => s_ALUSrc,
               ALUOp => s_ALUOp,
               MemWr => s_MemWr,
               Memtoreg => s_MemtoReg);

   Pc: entity work.PC(behavioral)
         generic map (N => 4)
         port map(clk   => clk,
                  reset => reset,
                  En    => s_EnPc,
                  cnt   => s_count);

   Mux2_1_0 : entity work.Mux2_1(behavioral)
         generic map(N => 8)
         port map(sel    => s_ALUSrc,
                  input0 => RData2,
                  input1 => signExtendOut,
                  y      => OutMux);

   Mux2_1_1 : entity work.Mux2_1(behavioral)
         generic map(N => 3)
         port map(sel    => s_RegDst,
                  input0 => IMemOut(6 downto 4),
                  input1 => IMemOut(9 downto 7),
                  y      =>Mux2_1Out);

   ALU : entity work.ALU(behavioral)
         port map(op1   => RData1,
                  op2   => outMux,
                  res   => ALUOut,
                  ALUOp =>s_ALUOp);

   DMemory : entity work.DMemory(behavioral)
         port map(writeEnable => s_MemWr,
                  writeClk  => clk,
                  address   => outMux,
                  writeData => RData2,
                  ReadData  => DMemOut);

   IMemory : entity work.IMemory(behavioral)
         port map(clk => clk,
                  En => s_RI,
                  RA => s_count,
                  RD => IMemOut);

   SignExtend : entity work.SignExtend(behavioral)
         port map(input  => IMemOut(6 downto 0),
                  output => signExtendOut);

   Regs : entity work.Regs(behavioral)
         port map(clk   => clk,
                  reset => reset,
                  RA1   => IMemOut(12 downto 10),
                  RA2   => IMemOut(9 downto 7),
                  WA    => Mux2_1Out,
                  WD    => DMemOut,
                  WE    => s_RegWr,
                  RD1   => RData1,
                  RD2   => RData2);
```

Fig4 – Instanciação da control unit e do dataPath.

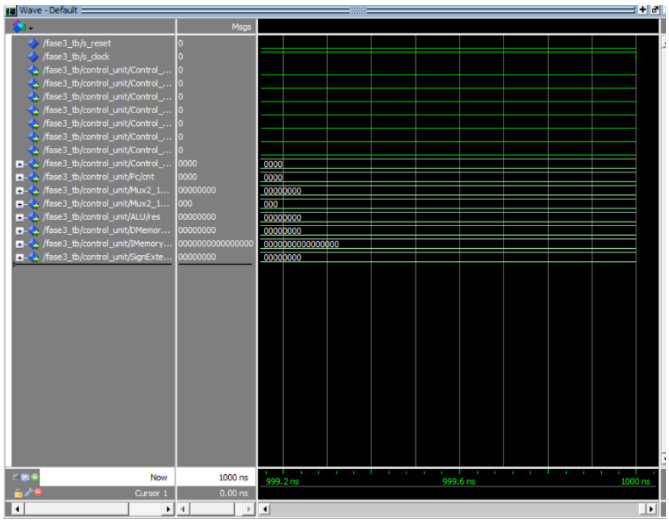

Fig5- TestBench da fase 3 mal, pois não consegui identificar os erros.

Conclusão

Este trabalho fez com que eu percebesse melhor a arquitetura de computadores, exigindo me trabalho, perseverança e espírito crítico.

Por pena minha, não consegui pôr em prática a fase 4.

Quanto à autoavaliação acho que mereço um 10 porque fiz o projeto até à fase 3 mas, no entanto, não consegui realizar a parte 4 nem realizar de forma correta os testes das fases 2 e 3.