MC542 - 2S 2011

Projeto Mips Pipeline

Cristiano J. Miranda RA 083382

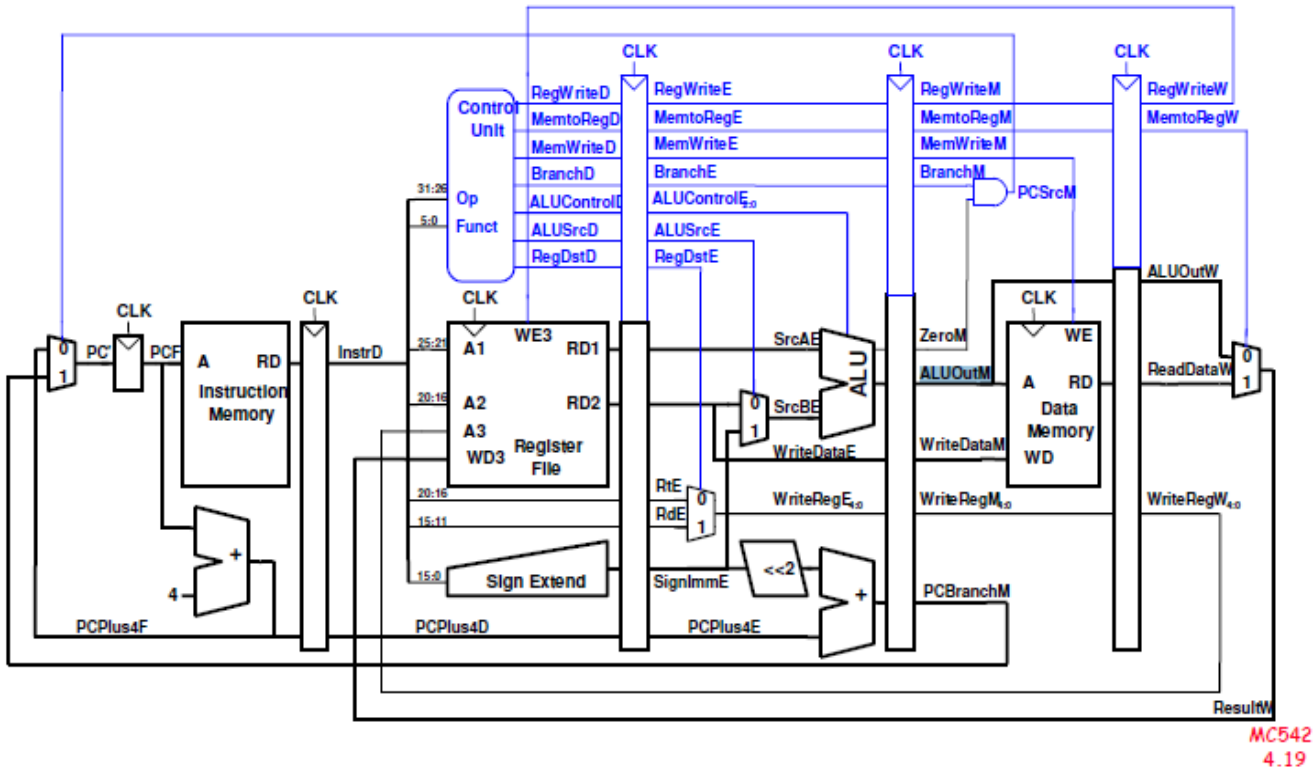
svn: https://cristianojmiranda.googlecode.com/svn/trunk/unicamp/mc542/projeto

Descrição

Implementação de um processador MIPS pipeline conforme apresentado em sala de aula (sem mecanismo de detecção de hazard e forwarding). Conforme entidae:

```
Entity mips is
        generic(nbits : positive := 32);
        port(Instruction : in std_logic_vector(nbits -1 downto 0);
                Data : in std_logic_vector(nbits -1 downto 0);
                clk : in std_logic;
                reset : in std_logic;
                PCF : out std_logic_vector(nbits -1 downto 0);
                ALUOutM : out std_logic_vector(nbits -1 downto 0));
                WriteDataM : out std_logic_vector(nbits -1 downto 0);
                MemWriteM : out std_logic);
End mips;
```

O processador deve conter uma ALU, uma unidade de controle, um register file, conforme implementações utilizadas nos laboratorios anteriormente entregues.



Implmentação

Para cada modulo do pipeline foi criado uma entidade no VHDL, conforme descrito abaixo:

Modulo	Entidade Vhdl	Observação
Instruction Fecth	instrfetch.vhd	modulo responsavel por obter a instrução apontada pelo pc na memoria de instrução.
Instruction Decode	instrdec.vhd	modulo responsavel por fazer a decodificação da instrução a ser executada.
Intruction Execute	instrexec.vhd	Modulo responsavel por executar a instrução.
Instruction Memory read/write	imem.vhd	Modulo responsavel por gerenciar a memoria de dados, gravando ou lendo valores em endereços decodificados pela instrução que esta sendo executada.
Instruction write back	instrwb.vhd	Modulo responsavel por atualizar os registradores no banco de registradores com valores lidos da memoria. Tais valores podem ou não serem atualizados na memoria novamente.
datapath	datapath.vhd	Contem todos os componente descritos anteriormente sendo linkados uns aos outros.
controller	controller.vhd	Contem as regras para decodificar uma instrução na fase de decodificação, assim como atualizar os flags de cada um dos modulos que seriam propagados posteriomente.
alu	alu.vhd	unidade logica aritimetica.
flip-flops	(flop, flopr, floprs).vhd	Registradores utilizados na implementação com as seguintes features: sem reset, com reset

		fixo para 0x0, com reset parametrizado.
mux	mux2.vhd	Multiplexador com 2 entradas, utilizado em praticamente todas os ciclos do processador.
Register file	rf.vhd	banco de registrador utilizado na fase de decode para armazenar informações da memoria ou de instruções.
Signal Extend	signext.vhd	Extende o sinal de um endereço, muito util para decodificar jump que atualizam o pc.
Shift left 2	sl2.vhd	Implementação de um shift left 2, utilizado para atualizar o pc em instruções de jump.

Test Bench

Foram realizados testes para os seguintes componentes:

1. **tb_adder**: Componente que adiciona dois sinais
    a. testando a entradaA com o valor 10 e a entradaB com valor 2 e verificando se o resultado era 12
    b. Similar ao teste anterior porem com outro valores.
2. **tb_controller**: foi implementado o controller porem na execução da implementação, gerando testes e corrigindo os defeitos não consegui garantir um cenario de tal modo que fosse possivel gerar testes para o controller, conforme explico na sessão “Considerações”.
3. **tb_datamem**: testa o cenario onde tenta se escrever e ler na memoria de dado. Foi um dos ultimos testes implementados. Ainda existe um defeito onde não esta sendo escrito corretamente no endereço fornecido.
4. **tb_flop***: para garantir que os registradores utilizados nos componentes eram confiaveis criei um teste banch para cada um testando as principais funcionalidades
    a. alteraçao de um valor em borda de subida
    b. mantendo o valor anterio em borda de descida mesmo que o valor da entrada seja alterado

c. reset assincrono
d. reset assincrono com valor default 0x0.
e. reset assincrono com valor parametrizado

5. **tb_imem**: comecei a testar esse componente mas me deparei com erros que fizeram com que fosse necessario testar os componentes que ele utiliza, porem não consegui finalizar os componente dependentes e voltar para finalizar os testes. Vide “Considerações”.
6. **tb_instrfetch**: comecei a testar esse componente mas me deparei com erros que fizeram com que fosse necessario testar os componentes que ele utiliza, porem não consegui finalizar os testes e voltar para finalizar o componente. Vide “Considerações”.
7. **tb_instrmem**: comecei a testar esse componente mas me deparei com erros que fizeram com que fosse necessario testar os componentes que ele utiliza, porem não consegui finalizar os testes e voltar para finalizar o componente. Vide “Considerações”.
8. **tb_rf**: como estou utilzando o mesmo register file feito no lab01, acabei reutilizando o teste banch desse componente.

Considerações

Professor,

como estou fazendo o projeto sozinho acabei não conseguindo conciliar todas as minhas provas de fim de semestre e outras atividades(trabalho) com o projeto, tenho trabalho nele desde domingo, creio que se tivesse mais tempo conseguiria terminá-lo. Todos os itens estão implementados e linkados, o que esta faltando são os testes. Consegui fazer teste de alguns componentes que alguns circuitos utilizam e conforme eu ia caminhando ia encontrando defeitos e corrigindo, infelizmente o prazo de entrega esta terminado e para finalizar todos os teste e corrigir todos os tempos acho que levaria pelo menos mais umas 12h.
Achei muito interessante a ideia de implementar o processador e muita coisa ficou claro com essa atividade, tenho me esforçado muito para finalizado e assim que conseguir um tempinho gostaria muito de finaliza-lo.

De qualquer forma, pelas notas que tirei nos lab (9.5 e 9.0), se eu tirasse pelo menos 1.0 nesse trabalho eu conseguiria fechar (MP = 5.5) a materia. Espero que considere o esforço.

Muito Obrigado,
Cristiano.