Relatório 3 MIPS Multiciclo

Mikael Luan da Silva Saraiva,
João Vitor Maia Neves Cordeiro,
Paola de Oliveira Abel

20 de Novembro de 2019

1 Introdução

Neste projeto desenvolveremos um PROCESSador MIPS multiciclo, criando seus arquivos de descrição em VHDL e realizando as devidas simulações além de analisar seu funcionamento.

2 Descrição do Sistema

Eu preciso completar isso

3 Principais características

Contem um unidade de memoria, banco de registradores com 32 registradores, um a ULA, possui PROCESSamento multiciclo é mais rápido que o monociclo, pois pode realizar mais de uma etapa de um PROCESSo por ciclo de clock, desde que usem áreas diferentes. No projeto multiciclo consideramos que um ciclo de clock pode acomodar no máximo uma das seguintes operações: Um acesso a memoria, um acesso ao banco de registradores (duas leituras e uma escrita, ou uma operação da ULA). Consequentemente quaisquer dados produzidos por uma dessas três unidades funcionais precisam ser salvos em um registrador temporário para uso em um ciclo posterior, se eles não forem salvos poderia haver a possibilidade de uma disputa de sincronização, levando ao uso de um valor incorreto.

3.1 Circuitos

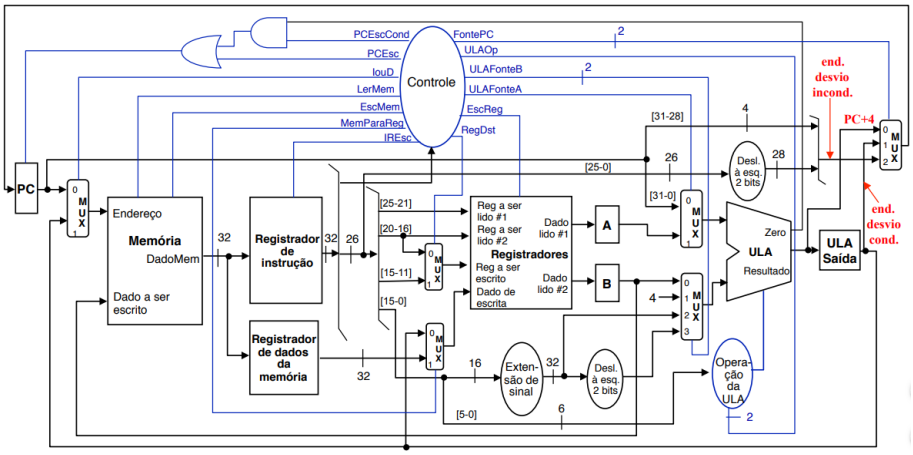


Figura 1: Circuito do MIPS multiciclo.

3.2 Diagramas

3.3 Transição de Estados

Pegar nos slides.

3.4 Maquinas de Estado

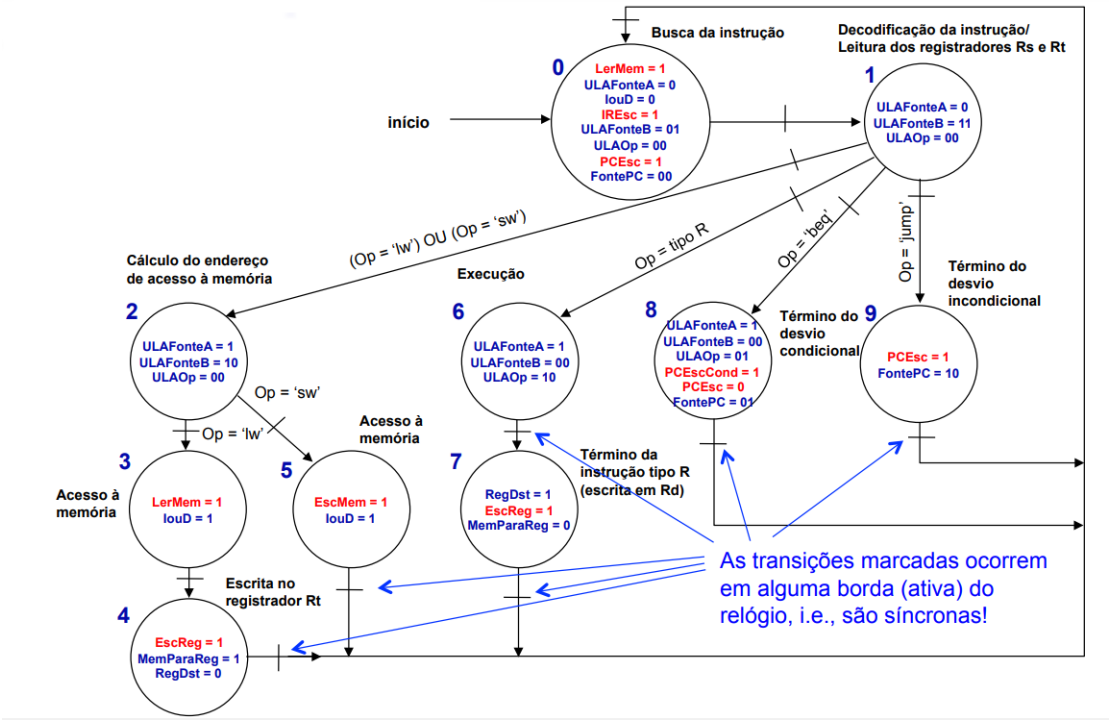


Figura 2: Maquina de estados do MIPS multiciclo.

4 Resultados de atrasos

A seguir é apresentado os dados adquiridos após a compilação do MIPS multiciclo.

Clock to Output Times

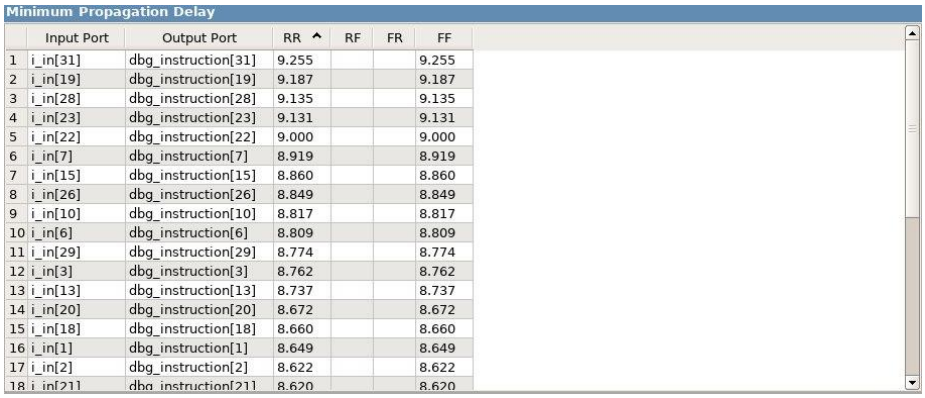
	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock R
1	dbg_data_3_EntradaMemDados[*]	Controle:control|currentState.S4	20.600	20.600	Fall	Controle:control
1	dbg_data_3_EntradaMemDados[12]	Controle:control|currentState.S4	20.600	20.600	Fall	Controle:control
2	dbg_data_3_EntradaMemDados[25]	Controle:control|currentState.S4	20.221	20.221	Fall	Controle:control
3	dbg_data_3_EntradaMemDados[31]	Controle:control|currentState.S4	20.178	20.178	Fall	Controle:control
4	dbg_data_3_EntradaMemDados[20]	Controle:control|currentState.S4	20.111	20.111	Fall	Controle:control
5	dbg_data_3_EntradaMemDados[4]	Controle:control|currentState.S4	20.057	20.057	Fall	Controle:control
6	dbg_data_3_EntradaMemDados[26]	Controle:control|currentState.S4	19.926	19.926	Fall	Controle:control
7	dbg_data_3_EntradaMemDados[10]	Controle:control|currentState.S4	19.891	19.891	Fall	Controle:control
8	dbg_data_3_EntradaMemDados[8]	Controle:control|currentState.S4	19.886	19.886	Fall	Controle:control
9	dbg_data_3_EntradaMemDados[16]	Controle:control|currentState.S4	19.882	19.882	Fall	Controle:control
10	dbg_data_3_EntradaMemDados[5]	Controle:control|currentState.S4	19.860	19.860	Fall	Controle:control
11	dbg_data_3_EntradaMemDados[27]	Controle:control|currentState.S4	19.691	19.691	Fall	Controle:control
12	dbg_data_3_EntradaMemDados[28]	Controle:control|currentState.S4	19.690	19.690	Fall	Controle:control
13	dbg_data_3_EntradaMemDados[3]	Controle:control|currentState.S4	19.643	19.643	Fall	Controle:control
14	dbg_data_3_EntradaMemDados[7]	Controle:control|currentState.S4	19.624	19.624	Fall	Controle:control
15	dbg_data_3_EntradaMemDados[1]	Controle:control|currentState.S4	19.598	19.598	Fall	Controle:control
16	dbg_data_3_EntradaMemDados[6]	Controle:control|currentState.S4	19.585	19.585	Fall	Controle:control

Figura 3: Clock to Output Times.

Hold Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	i_in[*]	Controle:control|currentState.S0	-2.642	-2.642	Rise	Controle:control|currentState.S0
1	i_in[0]	Controle:control|currentState.S0	-2.642	-2.642	Rise	Controle:control|currentState.S0
2	i_in[1]	Controle:control|currentState.S0	-6.473	-6.473	Rise	Controle:control|currentState.S0
3	i_in[2]	Controle:control|currentState.S0	-6.228	-6.228	Rise	Controle:control|currentState.S0
4	i_in[3]	Controle:control|currentState.S0	-6.923	-6.923	Rise	Controle:control|currentState.S0
5	i_in[4]	Controle:control|currentState.S0	-6.362	-6.362	Rise	Controle:control|currentState.S0
6	i_in[5]	Controle:control|currentState.S0	-5.997	-5.997	Rise	Controle:control|currentState.S0
7	i_in[6]	Controle:control|currentState.S0	-6.673	-6.673	Rise	Controle:control|currentState.S0
8	i_in[7]	Controle:control|currentState.S0	-6.100	-6.100	Rise	Controle:control|currentState.S0
9	i_in[8]	Controle:control|currentState.S0	-5.686	-5.686	Rise	Controle:control|currentState.S0
10	i_in[9]	Controle:control|currentState.S0	-5.697	-5.697	Rise	Controle:control|currentState.S0
11	i_in[10]	Controle:control|currentState.S0	-6.011	-6.011	Rise	Controle:control|currentState.S0
12	i_in[11]	Controle:control|currentState.S0	-6.835	-6.835	Rise	Controle:control|currentState.S0
13	i_in[12]	Controle:control|currentState.S0	-6.092	-6.092	Rise	Controle:control|currentState.S0
14	i_in[13]	Controle:control|currentState.S0	-6.739	-6.739	Rise	Controle:control|currentState.S0
15	i_in[14]	Controle:control|currentState.S0	-5.514	-5.514	Rise	Controle:control|currentState.S0
16	i_in[15]	Controle:control|currentState.S0	-7.078	-7.078	Rise	Controle:control|currentState.S0
17	i_in[16]	Controle:control|currentState.S0	-4.425	-4.425	Rise	Controle:control|currentState.S0

Figura 4: Hold Times.



Minimum Propagation Delay

	Input Port	Output Port	RR	RF	FR	FF
1	i_in[31]	dbg_instruction[31]	9.255			9.255
2	i_in[19]	dbg_instruction[19]	9.187			9.187
3	i_in[28]	dbg_instruction[28]	9.135			9.135
4	i_in[23]	dbg_instruction[23]	9.131			9.131
5	i_in[22]	dbg_instruction[22]	9.000			9.000
6	i_in[7]	dbg_instruction[7]	8.919			8.919
7	i_in[15]	dbg_instruction[15]	8.860			8.860
8	i_in[26]	dbg_instruction[26]	8.849			8.849
9	i_in[10]	dbg_instruction[10]	8.817			8.817
10	i_in[6]	dbg_instruction[6]	8.809			8.809
11	i_in[29]	dbg_instruction[29]	8.774			8.774
12	i_in[3]	dbg_instruction[3]	8.762			8.762
13	i_in[13]	dbg_instruction[13]	8.737			8.737
14	i_in[20]	dbg_instruction[20]	8.672			8.672
15	i_in[18]	dbg_instruction[18]	8.660			8.660
16	i_in[1]	dbg_instruction[1]	8.649			8.649
17	i_in[2]	dbg_instruction[2]	8.622			8.622
18	i_in[21]	dbg_instruction[21]	8.620			8.620

Figura 5: Minimum Propagation Delay.

Propagation Delay

	Input Port	Output Port	RR	RF	FR	FF
1	i_in[31]	dbg_instruction[31]	9.255			9.255
2	i_in[19]	dbg_instruction[19]	9.187			9.187
3	i_in[28]	dbg_instruction[28]	9.135			9.135
4	i_in[23]	dbg_instruction[23]	9.131			9.131
5	i_in[22]	dbg_instruction[22]	9.000			9.000
6	i_in[7]	dbg_instruction[7]	8.919			8.919
7	i_in[15]	dbg_instruction[15]	8.860			8.860
8	i_in[26]	dbg_instruction[26]	8.849			8.849
9	i_in[10]	dbg_instruction[10]	8.817			8.817
10	i_in[6]	dbg_instruction[6]	8.809			8.809
11	i_in[29]	dbg_instruction[29]	8.774			8.774
12	i_in[3]	dbg_instruction[3]	8.762			8.762
13	i_in[13]	dbg_instruction[13]	8.737			8.737
14	i_in[20]	dbg_instruction[20]	8.672			8.672
15	i_in[18]	dbg_instruction[18]	8.660			8.660
16	i_in[1]	dbg_instruction[1]	8.649			8.649
17	i_in[2]	dbg_instruction[2]	8.622			8.622
18	i_in[21]	dbg_instruction[21]	8.620			8.620

Figura 6: Propagation Delay.

Setup Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	⊟ i_in[*]	Controle:control|currentState.S0	7.991	7.991	Rise	Controle:control|currentState.S0
1	i_in[0]	Controle:control|currentState.S0	3.459	3.459	Rise	Controle:control|currentState.S0
2	i_in[1]	Controle:control|currentState.S0	7.166	7.166	Rise	Controle:control|currentState.S0
3	i_in[2]	Controle:control|currentState.S0	7.226	7.226	Rise	Controle:control|currentState.S0
4	i_in[3]	Controle:control|currentState.S0	7.938	7.938	Rise	Controle:control|currentState.S0
5	i_in[4]	Controle:control|currentState.S0	7.042	7.042	Rise	Controle:control|currentState.S0
6	i_in[5]	Controle:control|currentState.S0	7.015	7.015	Rise	Controle:control|currentState.S0
7	i_in[6]	Controle:control|currentState.S0	7.360	7.360	Rise	Controle:control|currentState.S0
8	i_in[7]	Controle:control|currentState.S0	6.790	6.790	Rise	Controle:control|currentState.S0
9	i_in[8]	Controle:control|currentState.S0	6.509	6.509	Rise	Controle:control|currentState.S0
10	i_in[9]	Controle:control|currentState.S0	6.583	6.583	Rise	Controle:control|currentState.S0
11	i_in[10]	Controle:control|currentState.S0	6.843	6.843	Rise	Controle:control|currentState.S0
12	i_in[11]	Controle:control|currentState.S0	7.527	7.527	Rise	Controle:control|currentState.S0
13	i_in[12]	Controle:control|currentState.S0	6.779	6.779	Rise	Controle:control|currentState.S0
14	i_in[13]	Controle:control|currentState.S0	7.614	7.614	Rise	Controle:control|currentState.S0
15	i_in[14]	Controle:control|currentState.S0	6.393	6.393	Rise	Controle:control|currentState.S0
16	i_in[15]	Controle:control|currentState.S0	7.991	7.991	Rise	Controle:control|currentState.S0
17	i_in[16]	Controle:control|currentState.S0	5.104	5.104	Rise	Controle:control|currentState.S0

Figura 7: Setup Times.

5 Utilização da placa

Utilizando os dados de compilação do Quartus, podemos ver que quanto ao uso de placa, o MIPS faz uso de uma quantidade muito pequena de registradores e funções combinacionais se comparado com o máximo suportado pela placa. Entretanto, o uso de pinos por parte dessa implementação é elevado, chegando a ultrapassar 50% da capacidade total da placa.

Flow Summary

Flow Status	Successful - Thu Nov 14 17:30:52 2019
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	mips
Top-level Entity Name	datapath
Family	Cyclone II
Device	EP2C35F672C6
Timing Models	Final
Total logic elements	462 / 33,216 (1 %)
Total combinational functions	462 / 33,216 (1 %)
Dedicated logic registers	138 / 33,216 (< 1 %)
Total registers	138
Total pins	279 / 475 (59 %)
Total virtual pins	0
Total memory bits	0 / 483,840 (0 %)
Embedded Multiplier 9-bit elements	0 / 70 (0 %)
Total PLLs	0 / 4 (0 %)

Figura 8: Flow Summary.

6 Discussão dos resultados

Durante os testes realizados com essa implementação do MIPS, a percepção geral do grupo é que uma implementação de um sistema digital complexo como um processador exige muito mais do que apenas a descrição do sistema em VHDL (ou outra linguagem de descrição de hardware). É necessário a aplicação de padrões de projeto e metodologias de teste mais ágeis do que o uso de arquivos .do com ModelSim.

A simulação das instruções implementadas foi a parte mais complicada em questão de complexidade, além de não ter ficado muito claro como é feito o uso do Testbench, o software padrão de simulação da disciplina (ModelSim Altera) tem pouca documentação online e consome quantidades elevadas de recursos de hardware. Quanto maior a simulação, mais hardware consumido, até o ponto onde a simulação não consegue continuar, isso limitou significativament o teste de um bloco grande de instruções e nos levou a optar por executar isoladamente cada instrução implementada.

A criação e validação de uma memória de instruções também foi um problema, já que para forçar os sinais necessários para testes das instruções, esses testes deveriam ser inseridos diretamente na memória de instruções, porém a inserção de dados na memória de instruções da maneira atual só pode ser feita a partir de uma instrução (store word). Assim, foi decisão do grupo criar sinais "mímicos"para realizar os testes, estes sinais irão substituir a saída da memória de instruções e irão copiar a saída dos registradores mais importantes da implementação. Apesar de funcionar como uma metodologia de teste, para a implementação completa do MIPS, outra solução deve ser escolhida.

As instruções que foram implementadas: Tipo R, Jump, Beq, Load Word, Store Word. As instruções que foram validadas: Tipo R, Jump, Beq, Store Word

A instrução Load Word não pode ser validada (apesar de ter sido implementada) justamente pela limitação de testes usando um sinal que imita a saída da memória de dados.

7 Testes

Instrução tipo R: esse teste em específico realizou a instrução add a seguir, com os valores dos registradores base setados para 8 e 4, resultando em uma saída 12.

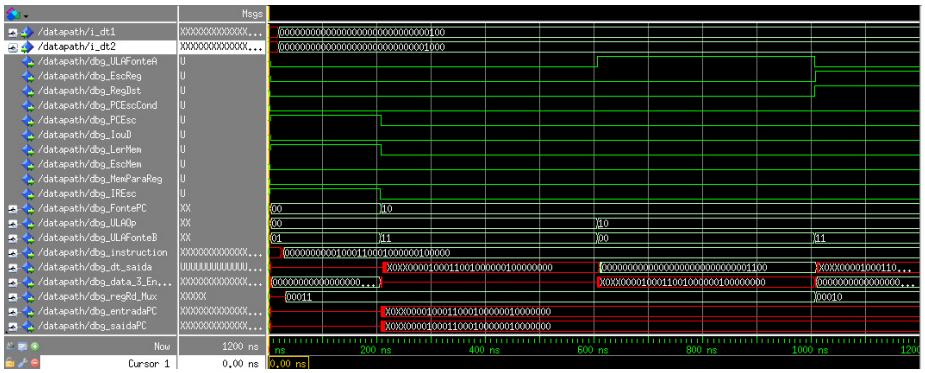


Figura 9: Onda da simulação de uma instrução R

Instrução tipo J: esse teste em específico realizou a instrução jump a seguir, com um deslocamento do PC no valor de 24 unidades

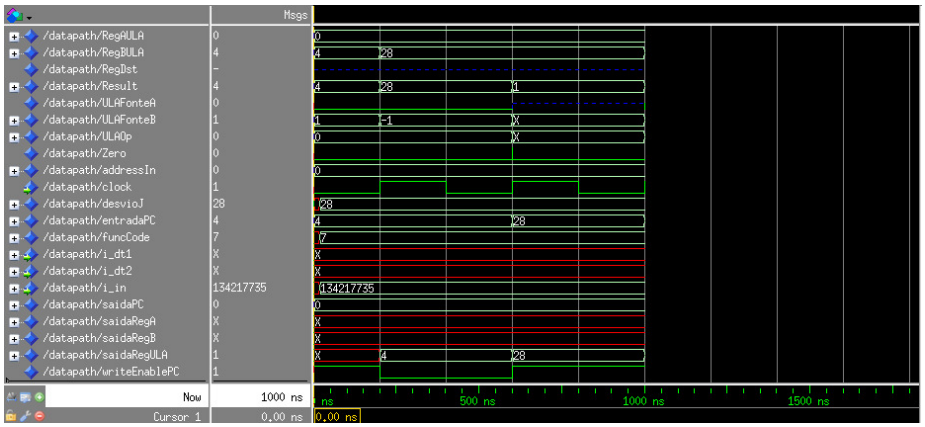


Figura 10: Onda da simulação de uma instrução J

8 Conclusões

Apesar de ser apenas uma versão resumida do MIPS, a implementação realizada contribui com a fixação de conhecimentos adquiridos tanto durante as aulas práticas quanto teóricas. Como processador com fins didáticos o MIPS cumpre um bom papel, sendo a compreensão do seu funcionamento intuitiva e simples. Entretanto, o uso comercial do MIPS, que já foi explorado por algumas empresas, exige a implementação de todas as features do processador, incluindo também aumento nas memórias e quantidade de instruções.

A gama limitada de instruções dessa versão do processador ainda consegue realizar blocos de instruções moderadamente complexos, mas pelo tamanho reduzido do banco de registradores e da memória RAM desse projeto, torna-se

inviável escrever um programa completo para rodar nessa implementação reduzida.