Trabalho Prático OC2:

I2O2

Segunda entrega : Pipeline & Multiplicação.

Lucas Starling, Semar Augusto, Pedro Militão, Igor Marques.

Introdução:

Para a segunda entrega do trabalho prático foi pedido a implementação em verilog do processador I2O2 com pipeline e operação de multiplicação em quatro ciclos.

Desenvolvimento:

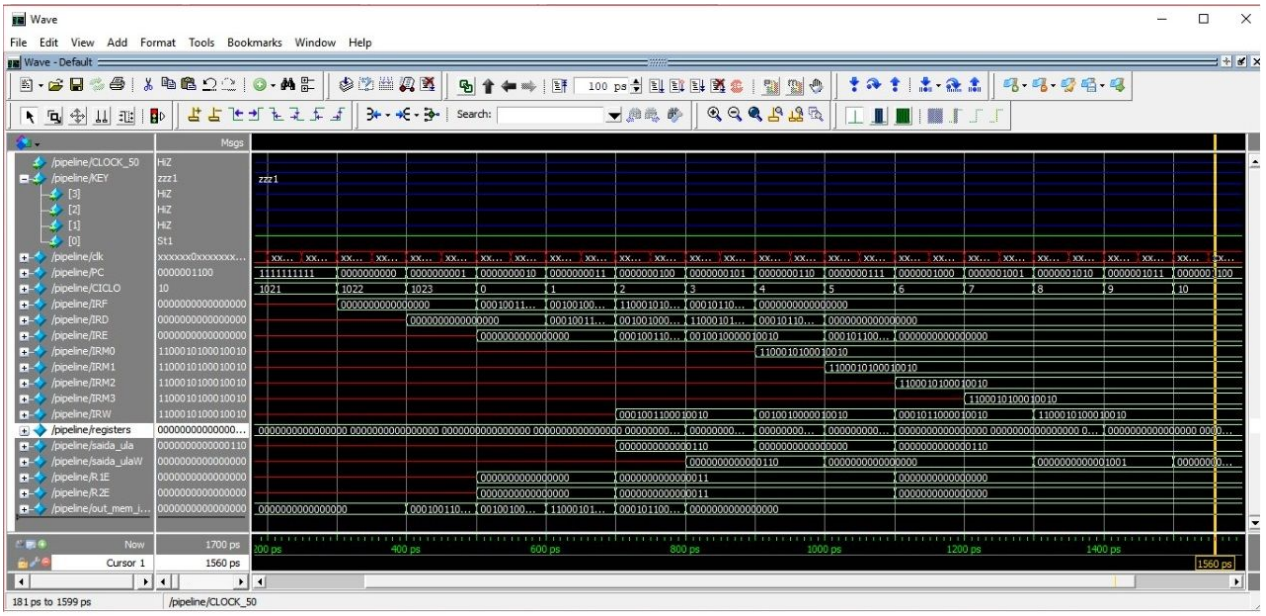
Para alterar o I2O2 básico, sem memória de instruções ou pipeline, a estrutura do código principal foi alterada. Para essa entrega o código não mais conta com uma máquina de estados que controla os estágios do processamento, visto que devido ao pipeline todos os estágios ocorrem ao mesmo tempo para instruções diferentes. O código final foi feito a partir do *Pipeline.v*, disponibilizado no pacote de arquivos para o trabalho, com alterações para incluir as demais operações: *Slti, And, AndI, Or, OrI, Xor, XorI, AddI, SubI e Multiplicação*. As alterações para as operações *Slti, And, AndI, Or, OrI, Xor, XorI, AddI e SubI* são simplesmente adição de testes condicionais dos bits 15 a 12 da instrução e atribuição em *saida_alu* da operação correta.

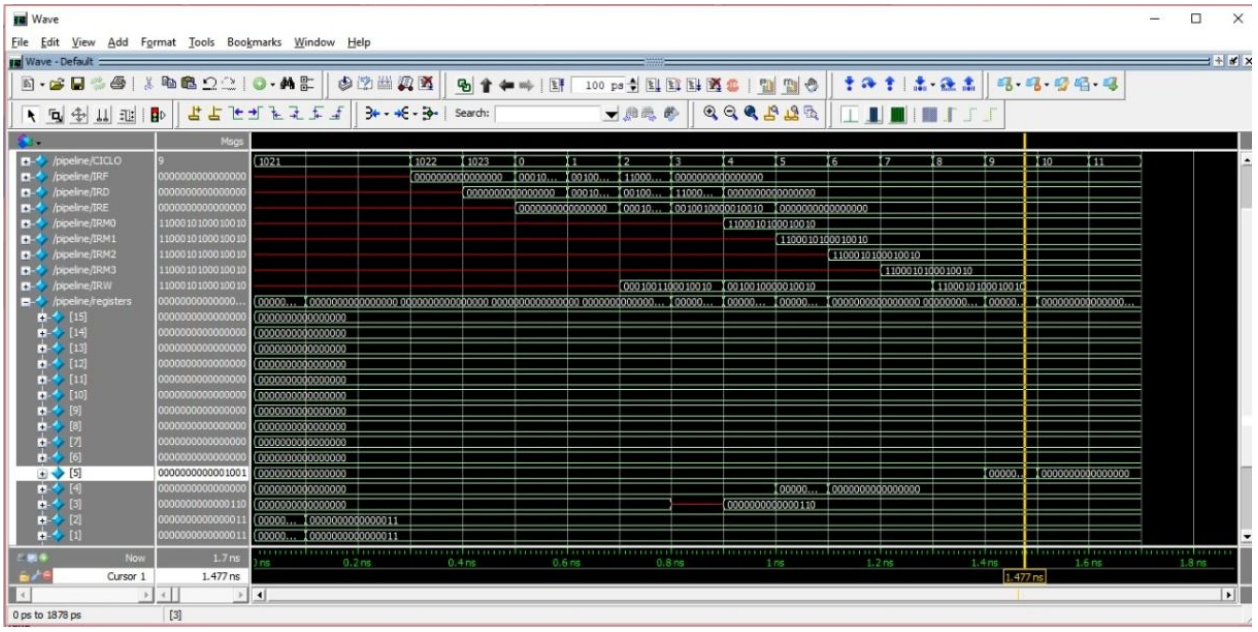
Por fim foi implementada a multiplicação, devido a essa ser uma operação que dura quatro ciclos, em um módulo diferente de *Execute.* Quando uma multiplicação é detectada o Decode atualiza a nova variável *IRM0 e* uma *flag "mult",* e quando isso ocorre uma série de quatro novos *always* são ativados. Os novos laços rodam quando o clock está em posedge e o estágio anterior da multiplicação foi concluído. No último estágio *saida_ulaW* recebe o valor da multiplicação entre os operandos. Esse valor segue para a o writeback normalmente como modelado no projeto base.

```
122
123         if(IRD[15:12] == 4'b1100)
124         begin
125             mult <= 1;
126             IRM0 <= IRD;
127         end
128         else
129         begin
130             IRE <= IRD;
131         end
132
133     end
134 end
135
```

```
185     always@(posedge clk[25]) // Y0
186   begin
187       if(START == 1 && KEY[0] == 1 && mult == 1)
188       begin
189           mult1 <= 1;
190           IRM1 <= IRM0;
191       end
192       else if(START == 1 && KEY[0] == 1 && mult == 0)
193       begin
194           mult1 <= 0;
195       end
196   end
197
198     always@(posedge clk[25]) // y1
199   begin
200       if(START == 1 && KEY[0] == 1 && mult1 == 1)
201       begin
202           mult2 <= 1;
203           IRM2 <= IRM1;
204       end
205       else if(START == 1 && KEY[0] == 1 && mult1 == 0)
206       begin
207           mult2 <= 0;
208       end
209   end
210
211     always@(posedge clk[25]) // y2
212   begin
213       if(START == 1 && KEY[0] == 1 && mult2 == 1)
214       begin
215           mult3 <= 1;
216           IRM3 <= IRM2;
217       end
218       else if(START == 1 && KEY[0] == 1 && mult2 == 0)
219       begin
220           mult3 <= 0;
221       end
222   end
223
224     always@(posedge clk[25]) // y3
225   begin
226       if(START == 1 && KEY[0] == 1 && mult3 == 1)
227       begin
228           IRW <= IRM3;
229           saida_ulaW <= R1E * R2E;
230       end
231   end
232
```

Simulação em ModelSim:


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
/pipeline/CLOCK_50
/pipeline/KEY
/pipeline/CICLO
/pipeline/IRF
/pipeline/IRD
/pipeline/IRE
/pipeline/IRM0
/pipeline/IRM1
/pipeline/IRM2
/pipeline/IRM3
/pipeline/IRW
/pipeline/registers
/pipeline/saida_ula
/pipeline/saida_ulaW
/pipeline/R1E
/pipeline/R2E
/pipeline/out_mem_i...
Now
1700 ps
Cursor 1
1560 ps
100 ps
400 ps
600 ps
800 ps
1000 ps
1200 ps
1400 ps
181 ps to 1599 ps
/pipeline/CLOCK_50


Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Msgs
/pipeline/CICLO
/pipeline/IRF
/pipeline/IRD
/pipeline/IRE
/pipeline/IRM0
/pipeline/IRM1
/pipeline/IRM2
/pipeline/IRM3
/pipeline/IRW
/pipeline/registers
[15]
[14]
[13]
[12]
[11]
[10]
[9]
[8]
[7]
[6]
[5]
[4]
[3]
[2]
[1]
Now
1.7 ns
Cursor 1
1.477 ns
0.2 ns
0.4 ns
0.6 ns
0.8 ns
1 ns
1.2 ns
1.4 ns
1.6 ns
1.8 ns
0 ps to 1878 ps
[3]