Relatório da Prática V - Diretório

Alunos: Bruno Rocha Ribeiro e Cristhian Sala Minoves

1. Introdução

Em contrapartida ao protocolo Snooping - MSI, visando auxiliar a reduzir os problemas por ele trazidos, foi elaborado um outro protocolo, o Diretório. A sua principal diferença é a existência de pequenas tabelas, tanto anexas às caches de cada processador, como nas suas memórias externas. Com isso, é possível ter um maior controle dos dados, como em qual processador ele se encontra, quais outras caches locais determinado dado está compartilhado ou se ele se encontra em um processador externo.

Apesar da grande diferença, o diretório e o snooping guardam uma semelhança: as suas transições de estados também são geridas por uso de uma máquina de estados — os quais são os mesmos na máquina de emissão e há uma troca na máquina receptora do estado inválido para o estado uncached. No entanto, suas transições são diferentes, possuindo, também, mensagens diferentes tanto de entrada como do barramento.

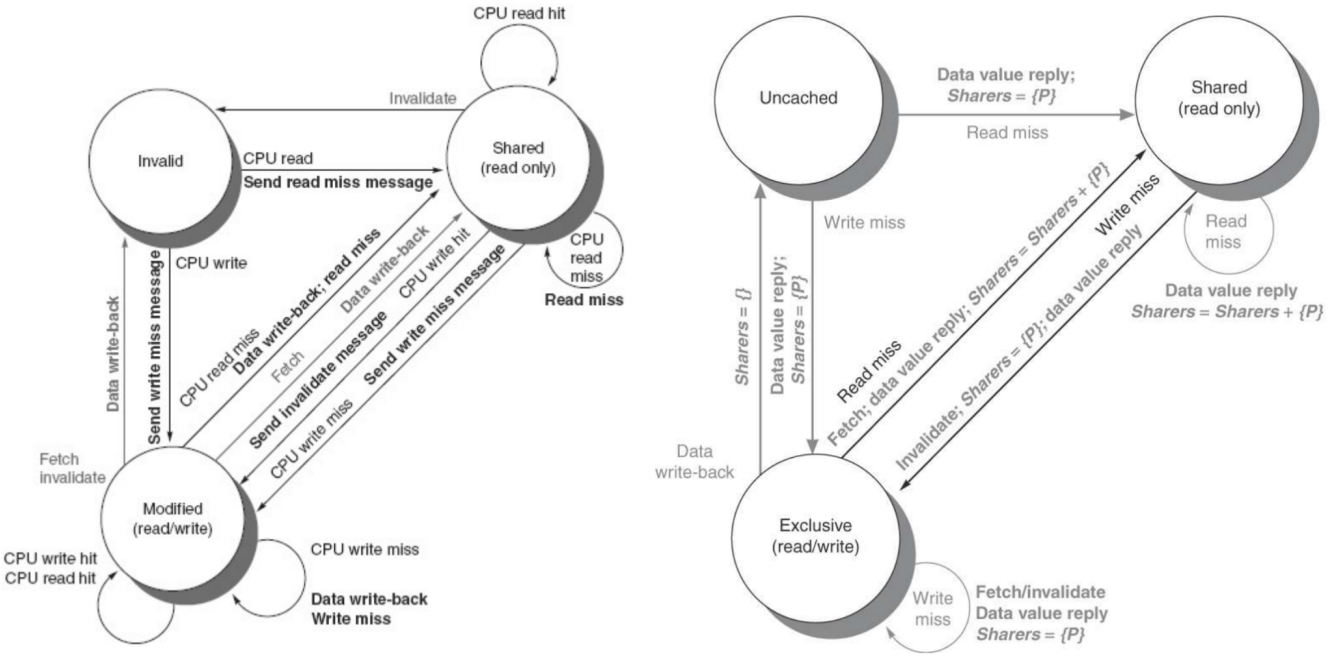
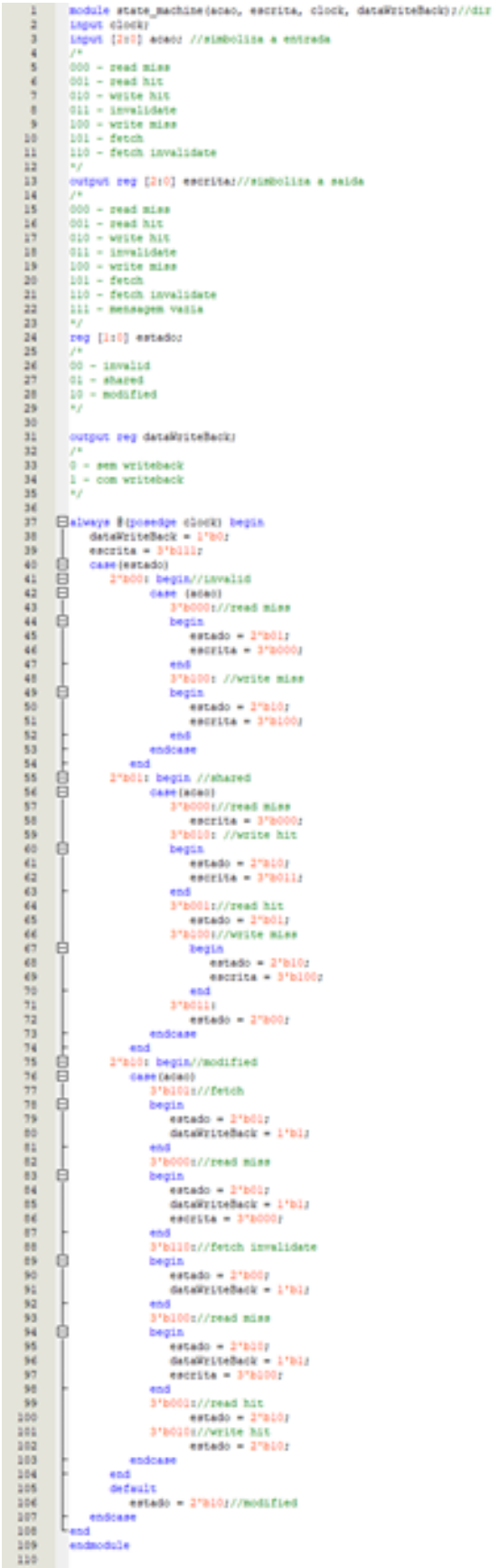


Figura 1: Funcionamento do protocolo Diretório com as máquinas emissora(esquerda) e receptora(direita)

2. Projeto e seus testes

Em nosso trabalho, confeccionamos apenas a máquina de estados do protocolo Diretório, sem a implementação de caches, memórias, processadores, etc. Abaixo será listado como foi feita essa parte.

Máquina emissora



```
module state_machine(acao, escrita, clock, dataWriteBack);//dir
input clock;
input [2:0] acao; //simbolisa a entrada
/*
000 - read miss
001 - read hit
010 - write hit
011 - invalidate
100 - write miss
101 - fetch
110 - fetch invalidate
*/
output reg [2:0] escrita;//simbolisa a saida
/*
000 - read miss
001 - read hit
010 - write hit
011 - invalidate
100 - write miss
101 - fetch
110 - fetch invalidate
111 - mensagem vazia
*/
reg [1:0] estado;
/*
00 - invalid
01 - shared
10 - modified
*/

output reg dataWriteBack;
/*
0 - sem writeback
1 - com writeback
*/

always @(posedge clock) begin
   dataWriteBack = 1'b0;
   escrita = 3'b111;
   case(estado)
      2'b00: begin//invalid
            case (acao)
               3'b000://read miss
               begin
                  estado = 2'b01;
                  escrita = 3'b000;
               end
               3'b100: //write miss
               begin
                  estado = 2'b10;
                  escrita = 3'b100;
               end
            endcase
         end
      2'b01: begin //shared
            case(acao)
               3'b000://read miss
                  escrita = 3'b000;
               3'b010: //write hit
               begin
                  estado = 2'b10;
                  escrita = 3'b011;
               end
               3'b001://read hit
                  estado = 2'b01;
               3'b100://write miss
                  begin
                     estado = 2'b10;
                     escrita = 3'b100;
                  end
               3'b011:
                  estado = 2'b00;
            endcase
         end
      2'b10: begin//modified
         case(acao)
            3'b101://fetch
            begin
               estado = 2'b01;
               dataWriteBack = 1'b1;
            end
            3'b000://read miss
            begin
               estado = 2'b01;
               dataWriteBack = 1'b1;
               escrita = 3'b000;
            end
            3'b110://fetch invalidate
            begin
               estado = 2'b00;
               dataWriteBack = 1'b1;
            end
            3'b100://read miss
            begin
               estado = 2'b10;
               dataWriteBack = 1'b1;
               escrita = 3'b100;
            end
            3'b001://read hit
                  estado = 2'b10;
            3'b010://write hit
                  estado = 2'b10;
         endcase
      end
      default
         estado = 2'b10;//modified
   endcase
end
endmodule
```

Figura 2: Implementação da máquina emissora

Seguindo a máquina de estados exposta anteriormente, foi feita a codificação dela. Foi observado os estados, as transições e as mensagens colocadas em cada uma das transições

Testes

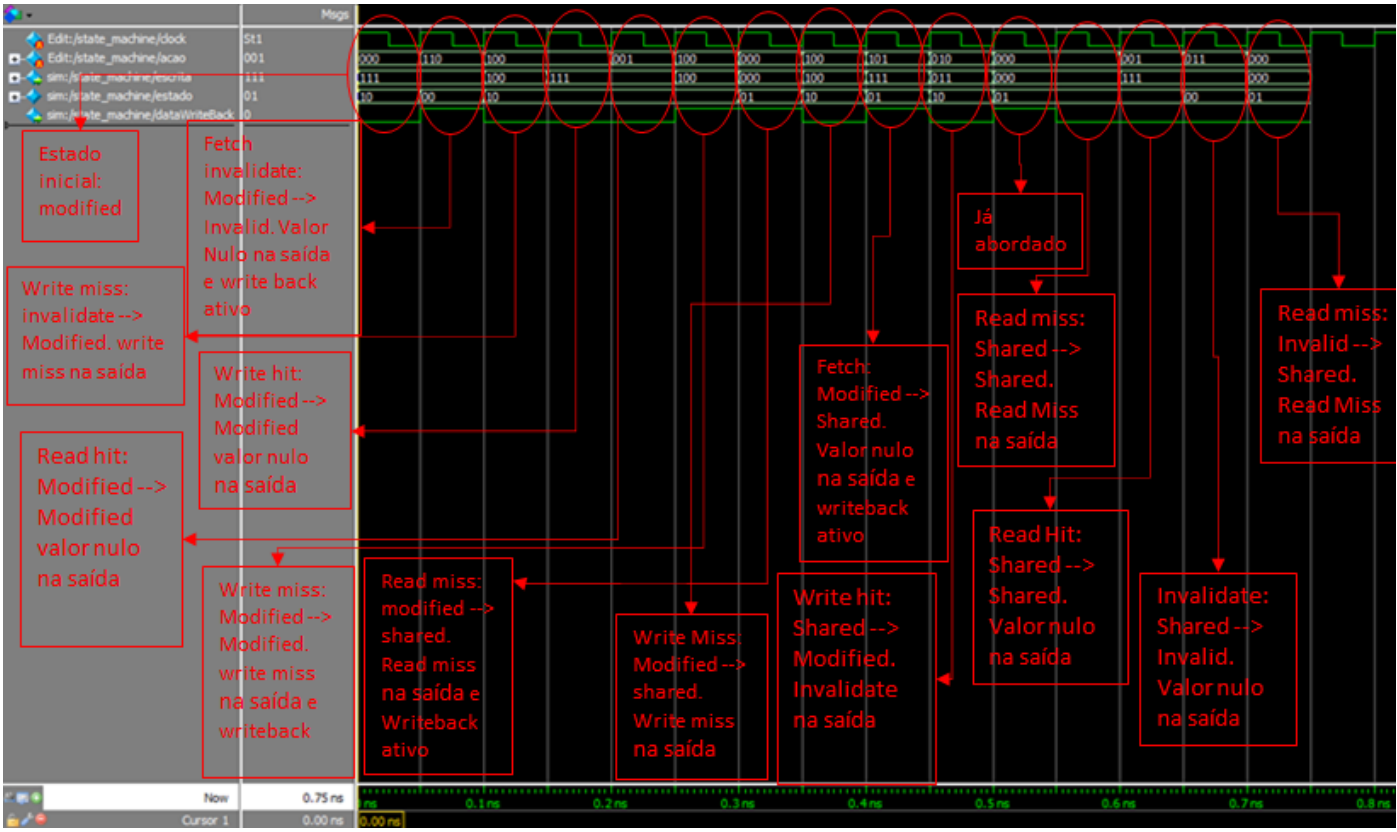


Figura 3: Testes da máquina emissora

Verifica-se nessa imagem o funcionamento correto de todas as transições da máquina emissora.

Máquina receptora

```
module state_machine2(acao, escrita, clock, dataWriteBack, dataValueReply,sharers);//dir
input clock;
input [1:0] acao; //simboliza a entrada
/*
00 - read miss
01 - read hit
10 - Data Writeback
11 - write miss
*/
output reg [1:0] escrita;//simboliza a saida
/*
00 - valor nulo
01 - invalidate
10 - fetch
11 - fetch invalidate
*/
output reg [1:0] sharers;
/*
00 - Nulo
01 - sharers = {}
10 - sharers = {p}
11 - Sharers = sharers + {p}
*/
reg [1:0] estado;
/*
00 - uncached
01 - shared
10 - exclusive
*/
input dataWriteBack;
/*
0 - sem writeback
1 - com writeback
*/
output reg dataValueReply;

always @(posedge clock) begin
   escrita = 2'b00;
   sharers = 2'b00;
   dataValueReply = 1'b0;
   case(estado)
      2'b00: begin//uncached
            case (acao)
               2'b00://read miss
                  begin
                     estado = 2'b01;
                     dataValueReply = 1'b1;
                     sharers = 2'b10;
                  end
               2'b11://write miss
                  begin
                     estado = 2'b10;
                     dataValueReply = 1'b1;
                     sharers = 2'b10;
                  end
            endcase
         end
      2'b01: begin //shared
            case(acao)
               2'b00://read miss
                  begin
                     estado = 2'b01;
                     sharers = 2'b11;
                     dataValueReply = 1'b1;
                  end
               2'b11://write miss
                  begin
                     estado = 2'b10;
                     escrita = 2'b01;
                     dataValueReply = 1'b1;
                     sharers = 2'b10;
                  end
            endcase
         end
      2'b10: begin//exclusive
         case(acao)
            2'b11://write miss
               begin
                  estado = 2'b10;
                  dataValueReply = 1'b1;
                  sharers = 2'b10;
                  escrita = 2'b11;
               end
            2'b00: //read miss
               begin
                  estado = 2'b01;
                  dataValueReply = 1'b1;
                  sharers = 2'b11;
                  escrita = 2'b10;
               end
            2'b10://data writeback
               begin
                  estado = 2'b00;
                  sharers = 2'b01;
               end
         endcase
      end
      default
      estado = 2'b00;
   endcase
end
endmodule

```

Figura 4: Implementação da máquina receptora

Testes

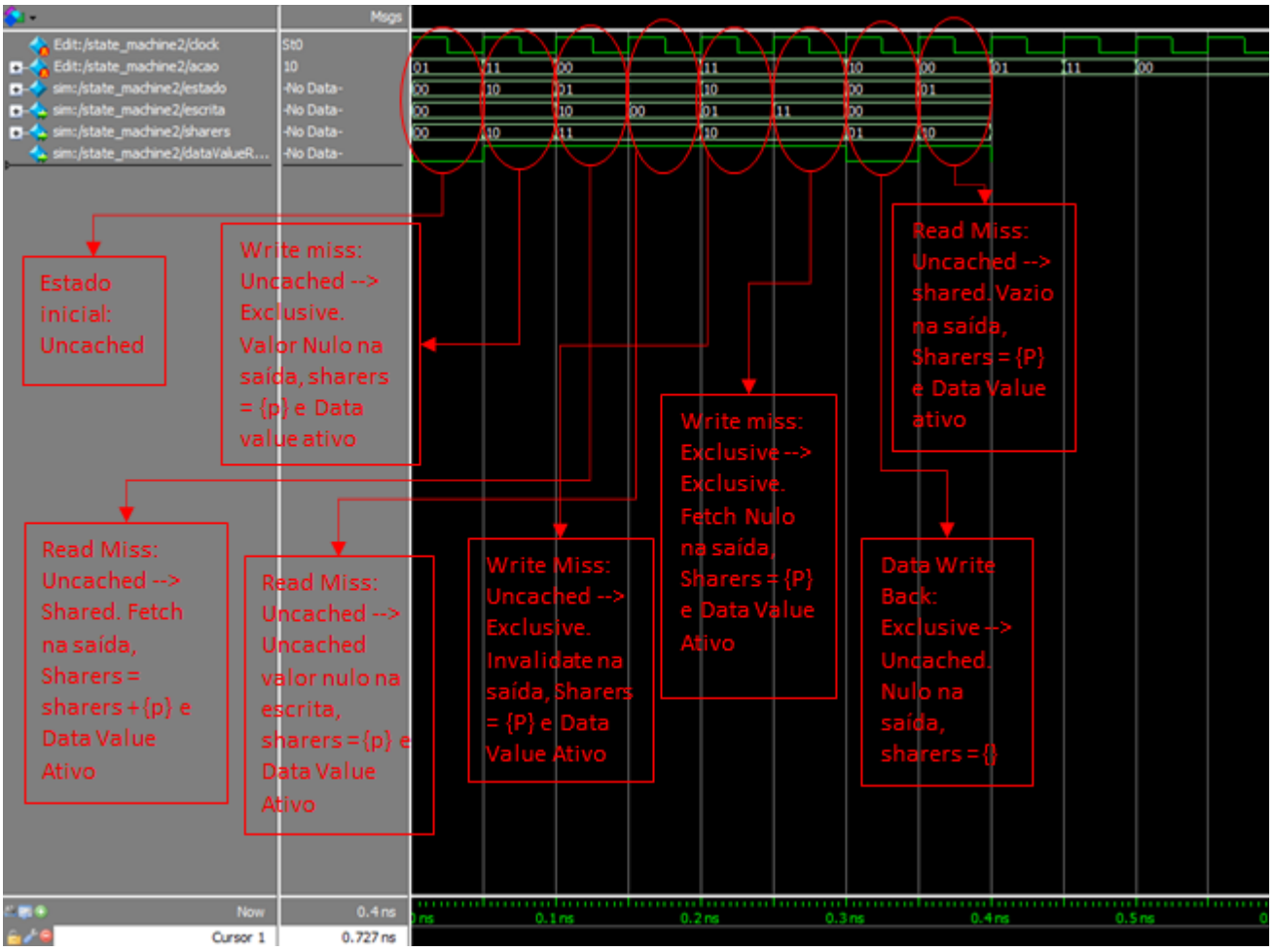


Figura 5: Testes da máquina receptora

3. Dificuldades encontradas

Como o projeto foi focado somente na primeira etapa, a da confecção das máquinas de estado, as dificuldades foram bem pequenas, só mesmo a confusão entre zeros e uns,situação que ocorreu também na confecção da máquina de estados do Snooping

4. Sugestões para melhoria da prática

Para essa parte 1, os métodos estão de acordo com o conteúdo da disciplina e aulas ministradas em sala, como sugestão apenas a questão da falta de um monitor que poderia nos auxiliar com questões mais complexas ou eventuais.

5. Comentários adicionais

Não é necessário comentar nada além do que já foi apresentado.