Esercizio Firmware in Verilog

Una unità U_a accede mediante interfaccia di memoria standard ad una memoria esterna **M** e comunica con una unità U_b. La memoria esterna, indirizzata alla parola, contiene un vettore **Y** di record di tipo **struct {int cod; int tot}** con **cod** e **int** da 32 bit. La posizione I-esima del vettore contiene in **cod** una chiave e in **tot** un intero. U_a interagisce con un'unità U_a che le manda una serie di richieste di operazione. Per ogni richiesta, Ua riceve da U_a un intero **I**, che utilizza come indice per accedere al vettore. *Successivamente*, riceve una chiave **K**. Se la chiave ricevuta **K** coincide con la chiave alla posizione **I** del vettore, U_a invia un riscontro positivo ad U_a e attende un terzo valore **V** da U_a, altrimenti segnala un errore e la comunicazione termina. Il terzo valore viene confrontato con **Y[I].tot** e si restituisce un **1** o uno **0** ad U_a a seconda se il valore di **Y[i].tot** è minore o maggiore o uguale al terzo valore ricevuto. L'interfaccia di U_a verso U_a è costituita dai soli registri in ingresso **IN** (32 bit) e in uscita **OUT** (1 bit), oltre agli indicatori a transizione di livello necessari all'implementazione del protocollo di comunicazione. Si forniscano:

- una implementazione della unità Ua, che minimizzi il numero di cicli di clock necessari ad eseguire l'operazione esterna che essa implementa;
- la lunghezza del ciclo di clock, nell'ipotesi di avere a disposizione ALU che operano in 6 t_p e porte logiche a 4 ingressi;
- il tempo medio di elaborazione, nell'ipotesi che la segnalazione dell'errore avvenga nel 25% dei casi.

Bozza di soluzione

Codice Unità

Gli indicatori a transizione di livello sono gestiti a programma:

- Quelli in uscita sono registri che alternano fra il valore 0 e il valore 1. Per esempio ACKin viene resettato con una `RESET(ACKin).
- Quelli in ingresso sono registri che hanno lo stesso comportamento di quelli in ingresso. Il valore vero o falso si ottiene confrontandone il valore con la corrispondente linea in ingresso: se sono uguali allora l'uscita sarà 0, altrimenti 1. Per esempio, RDYin è la linea in ingresso RDYinR è il registro di stato dell'indicatore (contatore modulo 2) e si usa una macro `RESET(RDYinR) per rimettere a falso una condizione vera.

```
//
// unità A : compito (vecchio programma) del 6/2/2020
//

module UB(// interfaccia richieste
      input         RDYin,
      output reg          ACKin,
      input [31:0]          IN,
      output reg [31:0] OUT,
      // interfaccia memoria
      output reg          RDYoutM,
      output reg          OP,
      output reg [31:0] IND,
      output reg [31:0] DATAOUT,
      input         RDYinM,
```

```
      input [31:0]         DATAIN,
      input [2:0]         ESITO,
      input         CLOCK);

// macro di comodo per indicatori a transizione di livello
`define   SET(A) A <= ~A;
`define RESET(A) A <= ~A;

   // tre stati
   reg [1:0]             Stato;

   // stato interno (registri temporanei): valore per confronto, indirizzo da leggere
   reg [31:0]      Chiave;
   reg [31:0]      Indirizzo;
   reg [31:0]      Valore;

   // gestione transizione di livello: contatori modulo due degli
   // indicatori in ingresso
   reg          RDYinR;
   reg          RDYinMR;

   initial  // inizializzazioni dello stato
     begin
    // inizializzazione registri temporanei
    Stato = 2'd0;

    // inizializzazione degli indicatori di uscita
    ACKin = 0;
    RDYoutM = 0;

    // inzializzazione dei reg interni degli indicatori di ingresso
    RDYinMR = 0;
    RDYinR = 0;
     end

   always @ (posedge CLOCK)
     begin
    case (Stato)
      //
      // MICROISTRUZIONE 0.
      //
      2'd0: begin
         // attesa di richiesta da parte dell'unità A
         if(RDYin != RDYinR )  // (RDYin = 1)
           begin
          $display("Istruzione 0. frase 1\nGestione interfaccia Ub");
          Indirizzo <= IN+1;
          `RESET(RDYinR)
          `SET(ACKin)
          $display("Richiesta lettura indirizzo : %d", IN);
          IND       <= IN;        // indirizzo da leggere
          OP        <= 1'b0;      // 0 lettura 1 scrittura
          `SET(RDYoutM)
          Stato     <= 2'd1;
           end // if (RDYin == RDYinR )
         else
           begin
          // NOP
```

```
     end
end
//
// MICROISTRUZIONE 1.
//
2'd1: begin
   // prima frase
   if(RDYin == RDYinR || RDYinM == RDYinMR)
     begin
    $display("Istruzione 1. frase 0 (NOP)");
    // NOP
     end
   // seconda frase
   if(RDYin != RDYinR &&
  RDYinM != RDYinMR &&
  DATAIN == IN &&
  ESITO == 0)
     begin
    $display("Istruzione 1, frase 1 DATAIN = %d == IN = %d",DATAIN,IN);
    // comunicazione con Ub
    $display("Gestione Ub");
    OUT <= 1'b1;
    `RESET(RDYinR)
    `SET(ACKin)
    // richiesta di lettura in memoria
    $display("Lettura in memoria IND = %d",Indirizzo);
    IND <= Indirizzo;
    OP <= 1'b0;
    `RESET(RDYinMR)
    `SET(RDYoutM);
    // prossimo stato 2.
    Stato = 2'd2;
     end // if (RDYin != RDYinR &&...
   // terza frase
   if(RDYin != RDYinR &&
  RDYinM != RDYinMR &&
  DATAIN != IN &&
  ESITO == 0)
     begin
    $display("Istruzione 1. frase 2");
    $display("Gestione Ub: esito negativo");
    OUT <= 1'b1;
    `SET(ACKin)
    `RESET(RDYinR)
    $display("Gestione memoria");
    `RESET(RDYinMR)
    // prossimo stato 0.
    Stato = 2'd0;
     end // if (RDYin != RDYinR &&...
   // quarta frase
   if(ESITO != 0)
     begin
    // ESITO negativo
    $display("Istruzione 1. frase 4: esito negativo lettura in memoria");
    OUT <= 1'b1;
    `SET(ACKin)
    `RESET(RDYinR)
    // NOP
```

```
    Stato = 2'd0;
    // torno alla zero
     end
end
//
// MICROISTRUZIONE 2.
//
2'd2: begin
   // prima frase : nessun rdy
   if(RDYinM == RDYinMR || RDYin == RDYinR)
     begin
    $display("Istruzione 2. prima frase: NOP");
    // nop
     end
   // seconda frase: entrabi i RDY, esito positivo
   if(RDYin != RDYinR && RDYinM == RDYinMR && ESITO == 0)
    $display("Istruzione 2. frase 1");
     begin
    if(DATAIN > IN)
      begin
         $display("Gestione Ub : esito positivo ");
         OUT <= 1'b0;
      end
    else
      begin
         $display("Gestione Ub : esito negativo ");
         OUT <= 1'b1;
      end
    `SET(ACKin)
    `RESET(RDYinR)
    `RESET(RDYinMR)
    // prossimo stato è 0
    Stato     <= 2'd0;
     end
   // seconda frase
   if(RDYin != RDYinR && RDYinM == RDYinMR && ESITO == 0)
     begin
    $display("Istruzione 2. frase 2");
    if(DATAIN > IN)
      begin
         $display("Gestione Ub : esito positivo ");
         OUT <= 1'b0;
      end
    else
      begin
         $display("Gestione Ub : esito negativo ");
         OUT <= 1'b1;
      end
    `RESET(RDYinR)
    `SET(ACKin)
    `RESET(RDYinMR)
     end // if (RDYin != RDYinR && RDYinM == RDYinMR && ESITO == 0)
   // terza frase
   if(RDYin != RDYinR && RDYinM == RDYinMR && ESITO == 0)
     begin
    $display("Istruzione 3. frase 2");
    OUT <= 1'b0;
    `RESET(RDYinR)
```

```
          `SET(ACKin)
          `RESET(RDYinMR)
          //
          Stato     <= 2'd0;
           end
      end
    endcase // case Stato

     end
endmodule
```

Codice TEST

```
module test();

`define RESET(A) A <= ~A;
`define   SET(A) A <= ~A;

   reg RDYin,RDYinM,CLOCK;

   reg [31:0] IN;
   reg [31:0] DATAIN;
   reg [2:0]  ESITO;

   wire       ACKin, RDYoutM, OP;

   wire [31:0] OUT;
   wire [31:0] IND;
   wire [31:0] DATAOUT;

   // system under test
   UB a(RDYin,ACKin,IN,OUT,RDYoutM,OP,IND,DATAOUT,RDYinM,DATAIN,ESITO,CLOCK);

   always
     begin
    #2 CLOCK = 1;
    #1 CLOCK = 0;
     end

   initial
     begin
    $dumpfile("test.vcd");
    $dumpvars;

    // condizioni iniziali
    RDYin <= 0;
    RDYinM  <= 0;
    CLOCK <= 0;
    IN <= 0;

    #6 // invio indirizzo
      IN <= 16;
    RDYin <= 1;

    #3 // memoria risponde
      DATAIN <= 123;
    ESITO <= 0;
```

```
    `SET(RDYinM)
    // e unità manda chiave
    `SET(RDYin);
    IN <= 123;

    #3 // memoria risponde V
      DATAIN <= 64;
    ESITO <= 0;
    `SET(RDYinM)
    // e unità manda il valore
    `SET(RDYin)
    IN <= 128;

    #6 // stesso giro ma con esito positvo alla fine
      IN <= 16;
    `SET(RDYin)

    #3 // memoria risponde
      DATAIN <= 123;
    ESITO <= 0;

    `SET(RDYinM)
    // e unità manda chiave
    `SET(RDYin);
    IN <= 123;

    #3 // memoria risponde V
      DATAIN <= 64;
    ESITO <= 0;
    `SET(RDYinM)
    // e unità manda il valore
    `SET(RDYin)
    IN <= 32;

    #6 // stesso giro ma con esito negativo alla chiave
      IN <= 16;
    `SET(RDYin)

    #3 // memoria risponde
      DATAIN <= 123;
    ESITO <= 0;

    `SET(RDYinM)
    // e unità manda chiave
    `SET(RDYin);
    IN <= 63;

    #50
      $finish;
     end


endmodule // test
```

Risultati

Il codice di test fa tre operazioni:

- Nella prima, l’unità B chiede la chiave all’indirizzo 16, passa come valore della chiave quello effettivamente trovato in memoria e un terzo valore V più grande (128) di quanto letto in memoria (64). L’operazione si conclude dopo tre cicli di clocl con esito negativo (OUT = 1)
- Nella seconda, il terzo valore è più piccolo (32) di quanto letto in memoria e l’operazione si conclude con successo (OUT = 0);
- Nella terza operazione, la chiave fornita (63) è diversa da quella letta in memoria (64) e l’operazione termina dopo due cicli soli, con OUT = 0.