Schema di soluzione per esercizio firmware con Verilog

Domanda

Un'unità firmware B è collegata ad un'altra unità firmware A ed a una memoria esterna M da 16K parole di 32 bit. B riceve da A un indirizzo di M e preleva da M la parola il cui indirizzo è stato fornito da A, e la invia ad A. Successivamente, preleva da M le parole che occupano posizioni consecutive, a partire da quella successiva indicata sopra, ed invia ad A la generica parola i-esima solamente se l'operazione di lettura è stata eseguita correttamente ed il suo contenuto è maggiore di quello dell'ultima parola inviata ad A. Ad esempio, se A invia a B l'indirizzo della parola che contiene "2" della seguente sequenza di parole 2,3,3,1,5,4,… allora B invierà ad A i valori 2,3,5,… B continua ad operare come su descritto finché non riceve un nuovo indirizzo da A. A quel punto, B ripete la procedura su descritta, a partire dal nuovo indirizzo.

Si implementi l'unità B e se ne fornisca la lunghezza del ciclo di clock, facendo le seguenti assunzioni:

- Sono a disposizione ALU che fanno operazioni fra interi in $5t_p$
- La prima parola prelevata ("2" nell'esempio) è sempre corretta
- Le porte logiche hanno al massimo 8 ingressi

Soluzione Verilog

Automa

L'unità firmware in oggetto deve sostanzialmente implementare un automa a tre stati:

- Nello stato 1 (iniziale), attende richieste dall'unità A. Se ne riceve una, manda la richiesta di lettura alla memoria, si tiene l'indirizzo in un registro temporaneo e passa nello stato 2
- Nello stato 2, attende una risposta dall'unità memoria. Quando la riceve, passa la parola letta all'unità A, la salva in un registro temporaneo, incrementa di uno l'indirizzo e richiede una nuova lettura alla memoria, passando nello stato 3. In questo stato, non viene controllato ne' ESITO (come da specifica) né quanto vale la parola letta.
- Nello stato 3, si attende un generico evento fra i due eventi possibili:
    - termina una lettura in memoria senza che l'unità A abbia richiesto una nuova operazione: in questo caso si controlla ESITO e valore letto. Se l'esito è positivo e il valore letto è maggiore del valore precedentemente salvato nel temporaneo, si manda il valore letto ad A e si ordina la lettura del prossimo indirizzo alla memoria, altrimenti si procede a richiedere la lettura del prossimo indirizzo senza comunicare alcunchè alla unità A
    - L'unità A richiede una nuova operazione: si procede inoltrando la richiesta di lettura del nuovo indirizzo alla memoria. Se la memoria non ha segnalato il completamento dell'operazione precedente, si passa direttamente allo stato 2. Se lo ha fatto, si scarta il valore letto e si passa comunque allo stato 2.

Realizzazione in Verilog

Interfaccia

L'unità ha la seguente interfaccia:

```
module UB(// interfaccia richieste
      input         RDYreqA,
      output reg          ACKreqA,
      input [13:0]          INDinA,
      // interfaccia risposte
      input         ACKansA,
      output reg          RDYansA,
      output reg [31:0] ANSWER,
      // interfaccia memoria
      output reg          RDYoutM,
      output reg          OP,
      output reg [13:0] IND,
      output reg [31:0] DATAOUT,
      input         RDYinM,
      input [31:0]          DATAIN,
      input [2:0]          ESITO);
```

I commenti indicano chiaramente le tre parti: interfaccia in ingresso da A (richieste di operazione), interfaccia in uscita verso A (parole lette dalla memoria), interfaccia verso la memoria (a domanda risposta, una sola coppia di indicatori).

Indicatori a transizione di livello

Per gli indicatori in uscita si usa un registro da 1 bit, inizializzato a 0. L'operazione di **set** consiste nell'assegnamento al registro del suo valore negato. Ad esempio:

```
RDYoutM   <= ~RDYoutM;
```

Per gli indicatori in ingresso, si usa un wire in ingresso e un registro interno inizializzato a 1. Il test dell'indicatore avviene confrontando il wire con il registro, e il **reset** avviene assegnando al registro interno il suo valore negato. Ad esempio:

- `RDYreqA != RDYreq` equivale a testare che RDYreqA valga 1
- `RDYreqA == RDYreq` equivale a testare che RDYreqA valga 0
- `RDYreqA = ~RDYreqA` equivale ad un `reset(RDYreqA)`

Automa

Prevediamo di avere un registro di stato da 2 bit. Ogni volta che viene segnalato un evento esterno (RDYreqA o RDYinM) si procede analizzando le condizioni in atto, eseguendo il codice relativo a quello stato e si passa infine allo stato successivo, che può essere lo stesso o diverso.

Codice completo dell'Unità

Il codice completo è riportato di seguito:

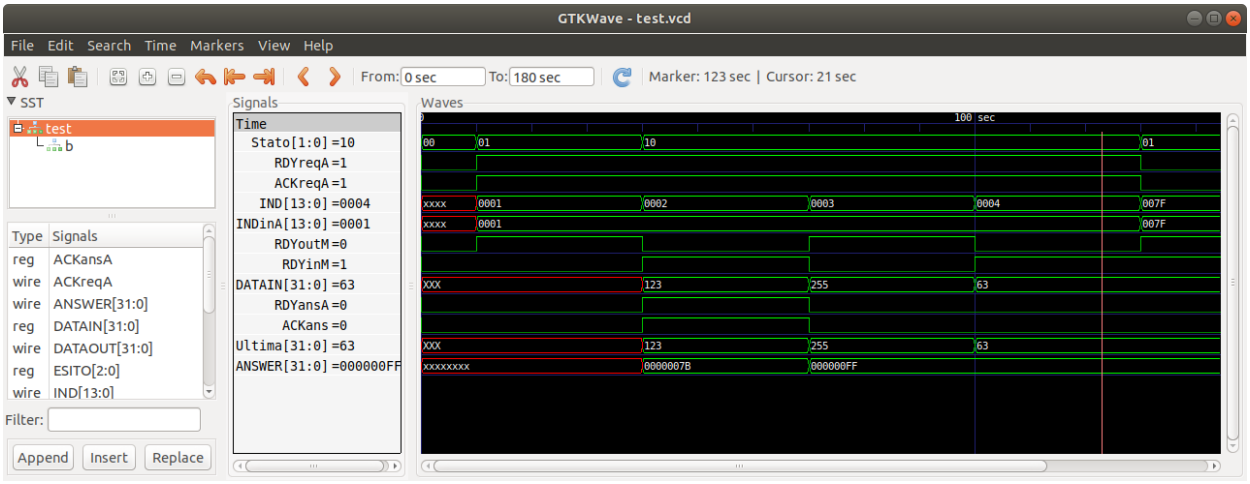
```
//
// // unità B
//

module UB(// interfaccia richieste
      input         RDYreqA,
      output reg          ACKreqA,
      input [13:0]          INDinA,
      // interfaccia risposte
```

```
   input         ACKansA,
   output reg          RDYansA,
   output reg [31:0] ANSWER,
   // interfaccia memoria
   output reg          RDYoutM,
   output reg          OP,
   output reg [13:0] IND,
   output reg [31:0] DATAOUT,
   input         RDYinM,
   input [31:0]          DATAIN,
   input [2:0]         ESITO);

// tre stati : attesa richiesta, attesa primo dato memoria, attesa evento
reg [1:0]      Stato;

// stato interno: valore per confronto, indirizzo da leggere
reg [31:0]      Ultima;
reg [13:0]      Indirizzo;

// gestione transizione di livello
reg          RDYreq;
reg          RDYin;
reg          ACKans;


initial
  begin
 Stato = 2'd0;
 Indirizzo = 32'd0;
 ACKreqA = 0;
 RDYoutM = 0;
 RDYreq = 0;  // per confronto
 RDYin = 0;   // per confronto
 ACKans = 0;  // per confronto
 RDYansA = 0;
 ACKreqA = 0;
 RDYoutM = 0;


  end

always @ (RDYreqA, RDYinM)
  begin
 case (Stato)
   2'd0: begin
      // attesa di richiesta da parte dell'unità A
      if(RDYreqA != RDYreq)
        begin
       $display("lettura richiesta unità A: ind %d", INDinA);
       Indirizzo <= INDinA;    // salvo indirizzo
       IND       <= INDinA;    // indirizzo da leggere
       OP        <= 1'b0;      // 0 lettura 1 scrittura
       RDYoutM   <= ~RDYoutM;  // set RDYoutM
       RDYreq    <= ~RDYreq;   // reset RDYreqA
       ACKreqA   <= ~ACKreqA;  // set ACKreqA
       Stato     <= 2'd1;      // passo ad attesa lettura
        end
   end
   2'd1: begin
      // attesa della lettura della prima parola dalla memoria
      if(RDYinM != RDYin && ACKansA != ACKans)
        begin
       $display("Letta prima parola da memoria : %d", DATAIN);
       ANSWER    <= DATAIN;    // parola letta ad unità A
       RDYin     <= ~RDYin;    // reset RDYinM
       RDYansA   <= ~RDYansA;  // set RDYansA
```

```
      ACKans    <= ~ACKans;   // reset ACKansA
      Ultima    <= DATAIN;    // salvataggio parola letta
      Indirizzo <= Indirizzo+1;   // agggiornamento indirizzo
      IND       <= Indirizzo+1;   // da leggere
      OP        <= 1'd0;      // operazione di lettura
      RDYoutM   <= ~RDYoutM;  // set RDYoutM
      Stato     <= 2'd2;      // prossimo stato steady state
       end
  end
  2'd2: begin
     if(RDYreqA != RDYreq && RDYinM == RDYin)
       begin // solo nuova richiesta da A
      Indirizzo <= INDinA;    // salvo indirizzo
      IND       <= INDinA;    // indirizzo da leggere
      OP        <= 1'b0;      // 0 lettura 1 scrittura
      RDYoutM   <= ~RDYoutM;  // set RDYoutM
      RDYreq    <= ~RDYreq;   // reset RDYreqA
      ACKreqA   <= ~ACKreqA;  // set ACKreqA
      Stato     <= 2'd1;      // passo ad attesa lettura
       end
     if(RDYreqA == RDYreq && RDYinM != RDYin &&
    (ESITO[0]==0 && ESITO[1]==0 && ESITO[2]==0) &&
    !(DATAIN < Ultima))       // devo spedire e richiedere nuova
       begin
      ANSWER    <= DATAIN;    // parola letta ad unità A
      RDYin     <= ~RDYin;    // reset RDYinM
      RDYansA   <= ~RDYansA;  // set RDYansA
      ACKans    <= ~ACKans;   // reset ACKansA
      Ultima    <= DATAIN;    // salvataggio parola letta
      Indirizzo <= Indirizzo+1;   // agggiornamento indirizzo
      IND       <= Indirizzo+1;   // da leggere
      OP        <= 1'd0;      // operazione di lettura
      RDYoutM   <= ~RDYoutM;  // set RDYoutM
      Stato     <= 2'd2;      // prossimo stato steady state
       end
     if((RDYreqA == RDYreq && RDYinM != RDYin &&
     (ESITO[0]==0 && ESITO[1]==0 && ESITO[2]==0) &&
     (DATAIN < Ultima))
    ||
    (RDYreqA == RDYreq && RDYinM != RDYin &&
     (ESITO[0]!=0 || ESITO[1]!=0 || ESITO[2]!=0)))
    // devo richiedere nuova
       begin
      RDYin     <= ~RDYin;    // reset RDYinM
      Ultima    <= DATAIN;    // salvataggio parola letta
      Indirizzo <= Indirizzo+1;   // agggiornamento indirizzo
      IND       <= Indirizzo+1;   // da leggere
      OP        <= 1'd0;      // operazione di lettura
      RDYoutM   <= ~RDYoutM;  // set RDYoutM
      Stato     <= 2'd2;      // prossimo stato steady state
       end // if ((RDYreqA == RDYreq &&...
     if(RDYreqA != RDYreq && RDYinM != RDYin)
       begin
      RDYin <= ~RDYin;         // scarto dato dalla memoria
      Indirizzo <= INDinA;    // salvo indirizzo
      IND       <= INDinA;    // indirizzo da leggere
      OP        <= 1'b0;      // 0 lettura 1 scrittura
      RDYoutM   <= ~RDYoutM;  // set RDYoutM
      RDYreq    <= ~RDYreq;   // reset RDYreqA
      ACKreqA   <= ~ACKreqA;  // set ACKreqA
      Stato     <= 2'd1;      // passo ad attesa lettura
       end
  end
endcase // case Stato
```

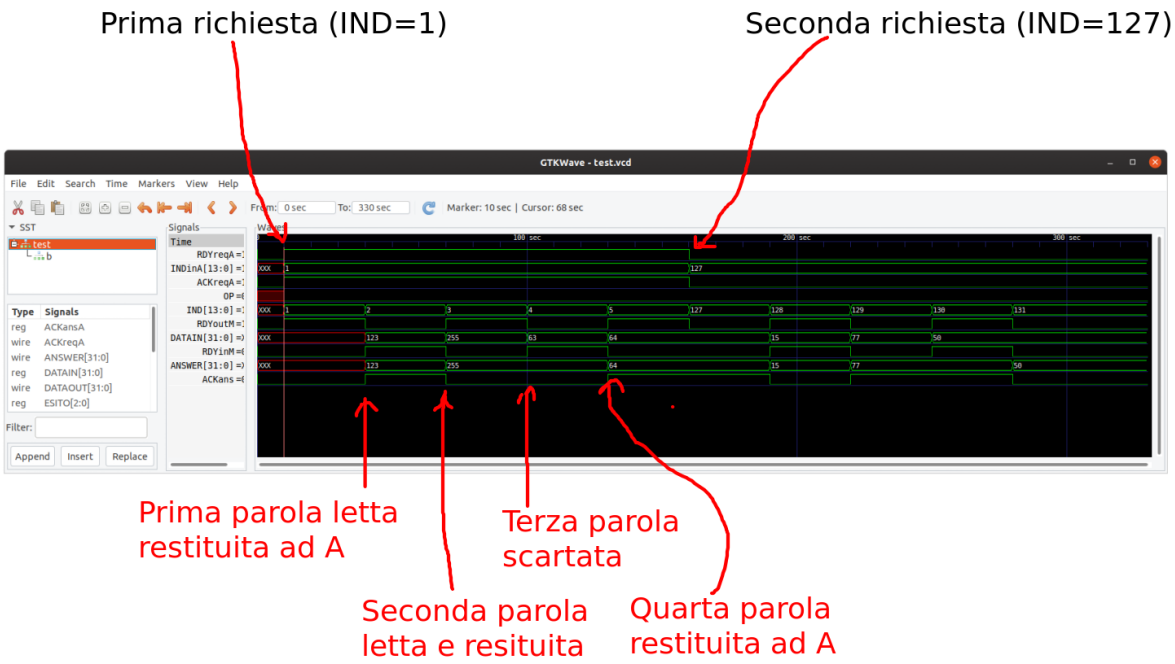
```
    end
endmodule
```

Utilizzando un programma di test che invia una richiesta di lettura dell'indirizzo 1 con una memoria che ha all'indirizzo 1 la parola 123, al 2 la parola 255, al 3 la parola 63, otteniamo un comportamento come quello riportato in Figura:



Come si vede, 123 e 255 vengo restituiti alla unità A (registro ANSWER) mentre non viene restituito 63 che risulta più piccolo di 255.

Anche se consideriamo più richieste consecutive, con valori diversi restituiti dalla memoria, otteniamo un comportamento corrispondente alla specifica:



Il programma di test utilizzato è il seguente:

```
module test();
```

```
reg RDYreqA, ACKansA,RDYinM;
reg [13:0] INDinA;
reg [31:0] DATAIN;
reg [2:0]  ESITO;

wire       ACKreqA, RDYansA, RDYoutM, OP;
wire [31:0] ANSWER;
wire [13:0] IND;
wire [31:0] DATAOUT;

// system under test
UB b(RDYreqA, ACKreqA, INDinA, ACKansA, RDYansA, ANSWER, RDYoutM,
 OP, IND, DATAOUT, RDYinM, DATAIN, ESITO);

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

 // condizioni iniziali
 RDYreqA <= 0;
 ACKansA <= 0;
 RDYinM  <= 0;

 // Adesso segnalo richiesta indirizzo 1
 #10
   INDinA = 14'd1;
 RDYreqA = ~RDYreqA;
 ACKansA = ~ACKansA;
 // la memoria risponde 123
 #30
   DATAIN = 32'd123;
 RDYinM = ~RDYinM;
 ESITO = 3'd0;
 // la memoria risponde 255
 #30
   DATAIN = 32'd255;
 RDYinM = ~RDYinM;
 ESITO = 3'd0;
 // la memoria risponde 63
 #30
   DATAIN = 32'd63;
 RDYinM = ~RDYinM;
 ESITO = 3'd0;
 // la memoria risponde 64
 #30
   DATAIN = 32'd64;
 RDYinM = ~RDYinM;
 ESITO = 3'd0;

 // la unità A chiede un altro indirizzo
 #30
   INDinA = 14'd127;
 RDYreqA = ~RDYreqA;
 ACKansA = ~ACKansA;
 // la memoria risponde 15
 #30
   DATAIN = 32'd15;
 RDYinM = ~RDYinM;
 ESITO = 3'd0;
  // la memoria risponde 77
 #30
   DATAIN = 32'd77;
 RDYinM = ~RDYinM;
 ESITO = 3'd0;
```

```
     // la memoria risponde 50
    #30
      DATAIN = 32'd50;
    RDYinM = ~RDYinM;
    ESITO = 3'd0;
      // la memoria risponde 55
    #30
      DATAIN = 32'd50;
    RDYinM = ~RDYinM;
    ESITO = 3'd0;
    #50
      $finish;
     end
endmodule // test
```

Il codice presentato non fa uso di CLOCK ma assume che le operazioni dell'unità siano dettate dai cambiamenti degli indicatori a transizione di livello.