Esercizio 7: comunicazione con handshaking

Nella comunicazione tra due entità, generalmente, si individua un’entità che trasmette i dati, chiamata master, ed un’entità che riceve ed elabora i dati, chiamata slave. Per far interagire correttamente il master e lo slave bisogna prendere in considerazione 2 fattori: la presenza o meno di un criterio di asservimento e la definizione di un criterio di riferimento temporale.

Quando l’entità slave è completamente asservita all’entità master, quest’ultima non deve preoccuparsi dell’istante di tempo in cui inviare i dati, dato che l’entità slave sarà sempre pronta a riceverne di nuovi. In questo caso, quindi, non vi è la necessità di introdurre alcun segnale aggiuntivo per instaurare una comunicazione, ma basterà immettere i dati sul canale (al più si può pensare di utilizzare un segnale di strobe per evidenziare l’effettiva presenza dei dati).

Tuttavia, non sempre tra 2 entità l’asservimento è assoluto, e dunque, bisogna assicurarsi che, quando l’entità master vuole iniziare una nuova trasmissione, l’entità slave sia a sua volta disponibile.

Oltre ad un problema di asservimento, in genere, sussiste un problema di riferimento temporale, il quale nasce quando le 2 entità che devono interagire si trovano su 2 schede differenti, e dunque non possono essere fatte considerazioni sui rispettivi riferimenti temporali. Infatti, anche se le 2 schede presentano clock isofrequenziali, non è detto che i riferimenti temporali siano analoghi, dato che questi possono variare per via alla fase iniziale.

Al fine di risolvere queste problematiche e garantire una corretta comunicazione tra 2 entità, si utilizza un protocollo di trasmissione basato su handshaking, che nel senso letterale indica una “stretta di mano” prima di avviare la comunicazione.

I protocolli di trasmissione si dividono in 3 categorie:

* Protocolli sincroni: le 2 entità che vogliono comunicare presentano un riferimento temporale condiviso ed il trasferimento avviene all’interno di 1 o più cicli di clock.
* Protocolli Asincroni: le 2 entità presentano riferimenti temporali differenti, sui quali non possono essere fatte previsioni sulla fase iniziale. Tali protocolli prevedono una sincronizzazione non legata più a transizioni regolari dei fronti di un clock, ma a segnali di controllo/stato che le entità si scambiano in aggiunta ai dati, al fine di identificare l’inizio e la fine di un’operazione.
* Protocolli semi-sincroni: le 2 entità presentano un clock comune, così come nei protocolli sincroni, ma in aggiunta utilizzano un ulteriore segnale di controllo, detto wait. Tale segnale viene inviato dall’unità più lenta a quella più veloce quando non riesce ad elaborare i dati in un singolo colpo di clock. Il termine semi-sincrono deriva dal fatto che, seppur la trasmissione avviene tramite segnali di controllo/stato, questi vengono rilevati solo in presenza di un fronte del clock. (un protocollo di questo tipo è generalmente implementato nella comunicazione tra il processore e la memoria cache).

I protocolli di handshake sono protocolli di tipo asincrono e si differenziano per il numero di eventi utilizzati nelle fasi di un trasferimento; si individuano 2 principali tipologie:

* Handshake semplice: le 2 entità si sincronizzano solo all’inizio o alla fine di un trasferimento. Una volta sincronizzate, le 2 entità svolgono autonomamente le proprie operazioni e sono sempre pronte ad avviare una nuova trasmissione.
* Handshake completo: le 2 entità si scambiano eventi per definire l’inizio e la fine di ciascuna fase del protocollo, indipendentemente dalle rispettive velocità operative.

I protocolli di handshake, inoltre, si basano sulle seguenti ipotesi:

* Le 2 entità lavorano sempre una come master e l’altra come slave.
* La trasmissione può essere avviata da entrambe.
* È l’entità più lenta a definire il ritmo della trasmissione.
* Non si pongono vincoli sulla velocità delle rispettive unità operative.

Per implementare un protocollo basato su handshake sono necessari almeno 2 segnali, tramite i quali codificare 2 o 4 eventi, a seconda che sia un protocollo semplice o completo.

Approccio Utilizzato:

L’esercizio in esame prevede 2 sistemi, A e B, dove il sistema A deve inviare N valori al sistema B il quale, a sua volta, dovrà sommare tali valori a quelli contenuti in memoria per poi salvare il risultato in ulteriori locazioni della memoria stessa.

Entrambi i sistemi sono stati definiti mediante un approccio strutturale, dove l’elemento principale è l’interfaccia dei 2 sistemi, la quale è responsabile dell’implementazione del protocollo di handshaking. Prima di passare alla descrizione dei 2 sistemi, nel paragrafo successivo verrà descritto il protocollo di handshaking implementato.

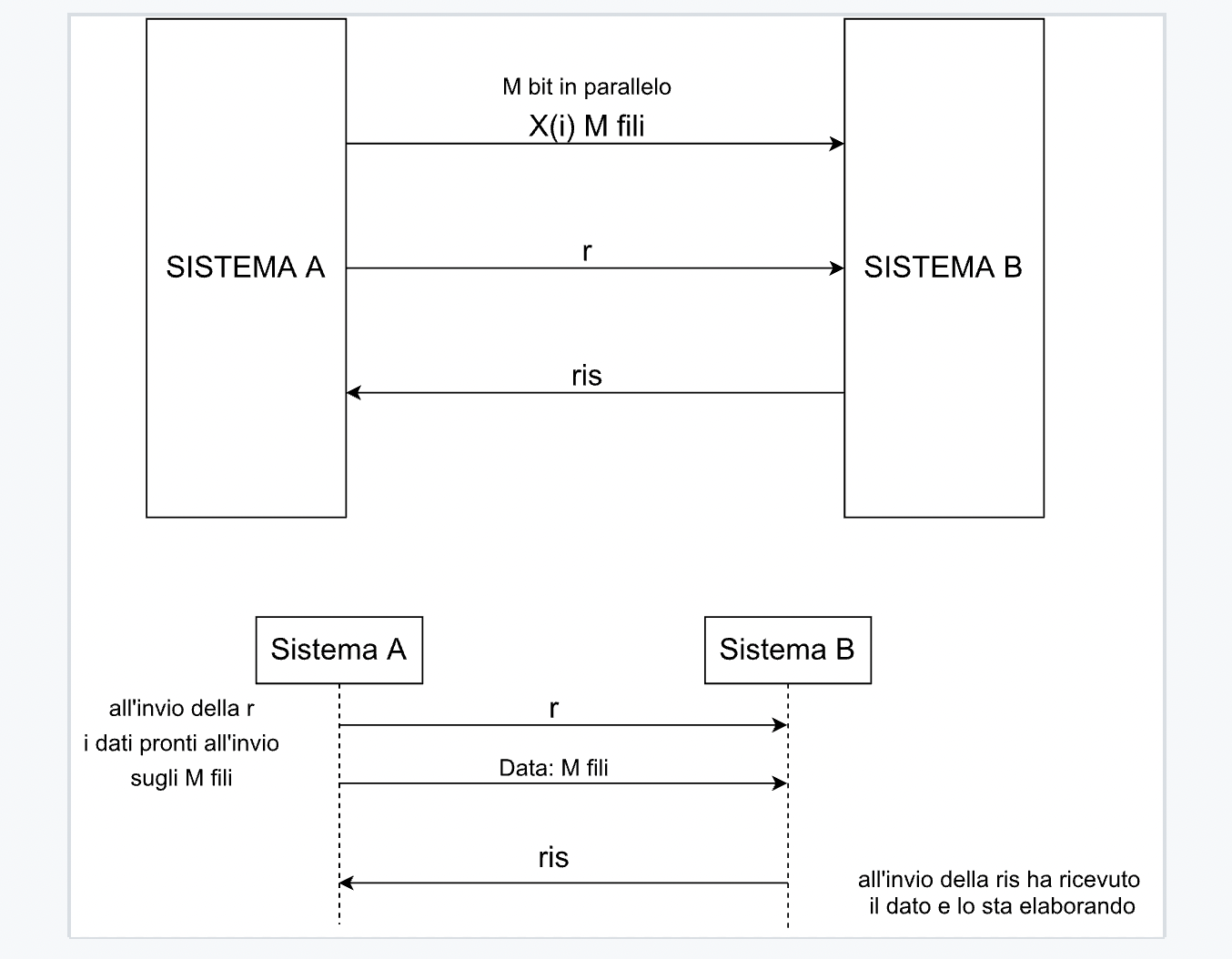
Descrizione Protocollo:

Il protocollo di handshaking implementato è di tipo completo e prevede un segnale r, ovvero di request, alzato dal sistema A (il master), il quale indica la volontà di trasmettere dei dati, ed un segnale di ris, alzato dal sistema B(lo slave) in ricezione del segnale di request, che indica la disponibilità nel ricevere i dati e che rimarrà alto fino alla completa elaborazione di quest’ultimi.

La variazione da 1 a 0 del segnale di ris indica, dunque, la completa elaborazione dei dati da parte dello slave e viene interpretata dal master come la possibilità di inoltrare nuovi dati.

L’handshake avviene prima della trasmissione di ogni dato.

Vediamo ora come i 2 sistemi implementano tale protocollo.



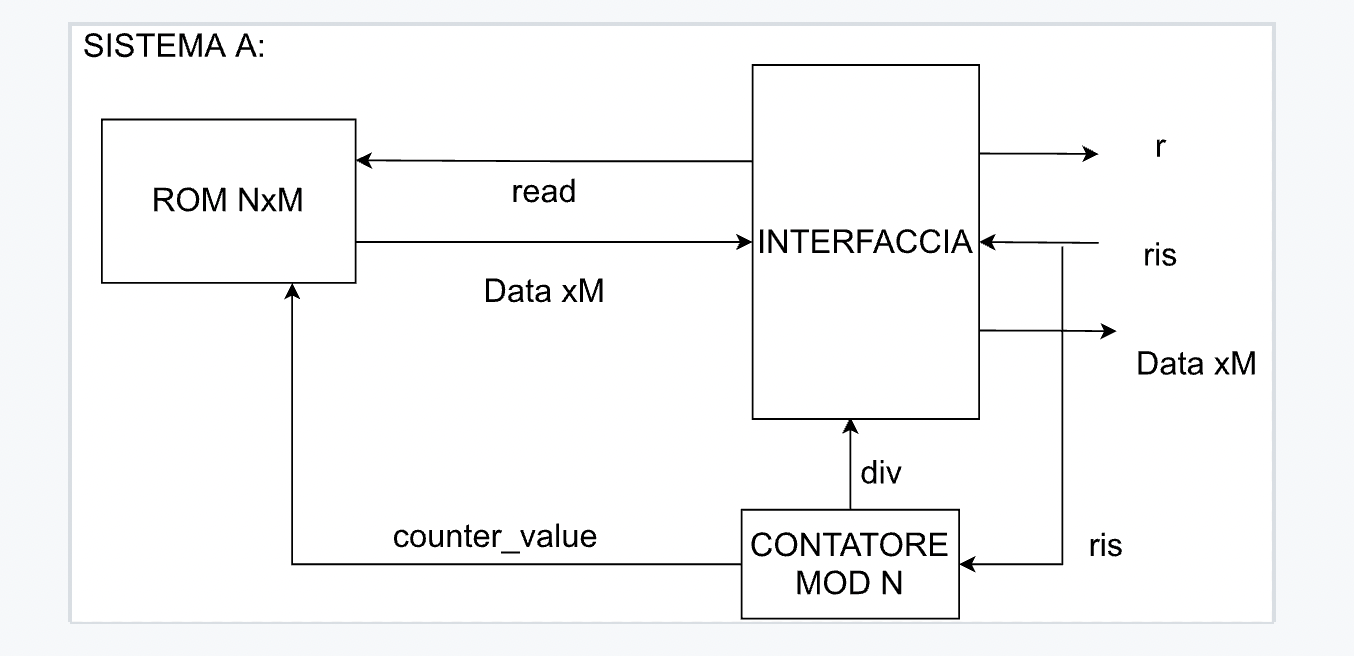
SistemaA:

Come precedentemente detto, il sistemaA è stato definito tramite un approccio strutturale, utilizzando i seguenti componenti:

-una memoria ROM dove vengono salvati i valori X(i);

-un'interfaccia per effettuare l'handshaking e la trasmissione;

-un contatore che segna il numero di comunicazioni già avvenute;



Il sistema ripete la stessa sequenza di operazioni per ogni valore X(i) contenuto in ROM.

Alla comunicazione i-esima, l'interfaccia invia un segnale di read per leggere il valore X(i) puntato dal valore del contatore. Una volta letto il valore, l'interfaccia pone in uscita i dati per iniziare la nuova comunicazione e alza il segnale di R; abbassa tale segnale solo dopo l'arrivo del RIS. Finchè RIS resta alto, il sistema A non può iniziare una nuova comunicazione, per questo rimane in attesa che il ricevitore finisca di elaborare i dati, cioè che RIS si abbassi. Appena RIS = 0 allora posso iniziare la nuova comunicazione i+1.

Appena si raggiungono le N comunicazioni il segnale di DIV in uscita al contatore si alza e termina la comunicazione complessiva tra il sistema A e B dal lato del trasmettitore.

Componente interfaccia:

Immagine che contiene testo

Descrizione generata automaticamente

l’interfaccia è il componente che gestisce l’handshaking e la trasmissione dei dati sul canale. È stata progettata come una macchina a 7 stati, dove:

q0 -> è lo stato iniziale, dove ci si ritrova prima della trasmissione oppure al termine. Per distinguere i 2 casi si utilizza il segnale di div in uscita dal contatore: quando è basso, indica che ancora non abbiamo trasferito il primo dato (dato che differenziamo la prima trasmissione dalle altre e, dunque, in q0 si valuta solo la prima), mentre, quando è alto, indica che tutti i dati sono stati trasmessi e si permane indefinitamente in q0.

Quando il segnale di div è basso, si passa il q1 e viene alzato il segnale di read, in modo da leggere dalla memoria il valore da trasmettere.

q1 -> viene abbassato il segnale di read, dato che deve durare solo 1 colpo di clock, al fine di evitare letture spurie in memoria. Si passa poi nello stato q2.

Anche se può sembrare superfluo, lo stato q1 è necessario, per dare il tempo al componente memoria di porre in uscita il dato i-esimo.

q2 -> siamo ora sicuri che il dato in uscita dalla memoria è quello corretto: possiamo dunque riporlo sul canale ed alzare il segnale r. Si passa ora nello stato q3.

q3 -> tale stato è sostanzialmente di attesa: finchè non si alza il segnale di ris in ingresso, si permane in q3. Quando invece tale segnale viene rilevato come alto, i dati sono stati acquisiti dal sistemaB e si può passare dunque allo stato q4.

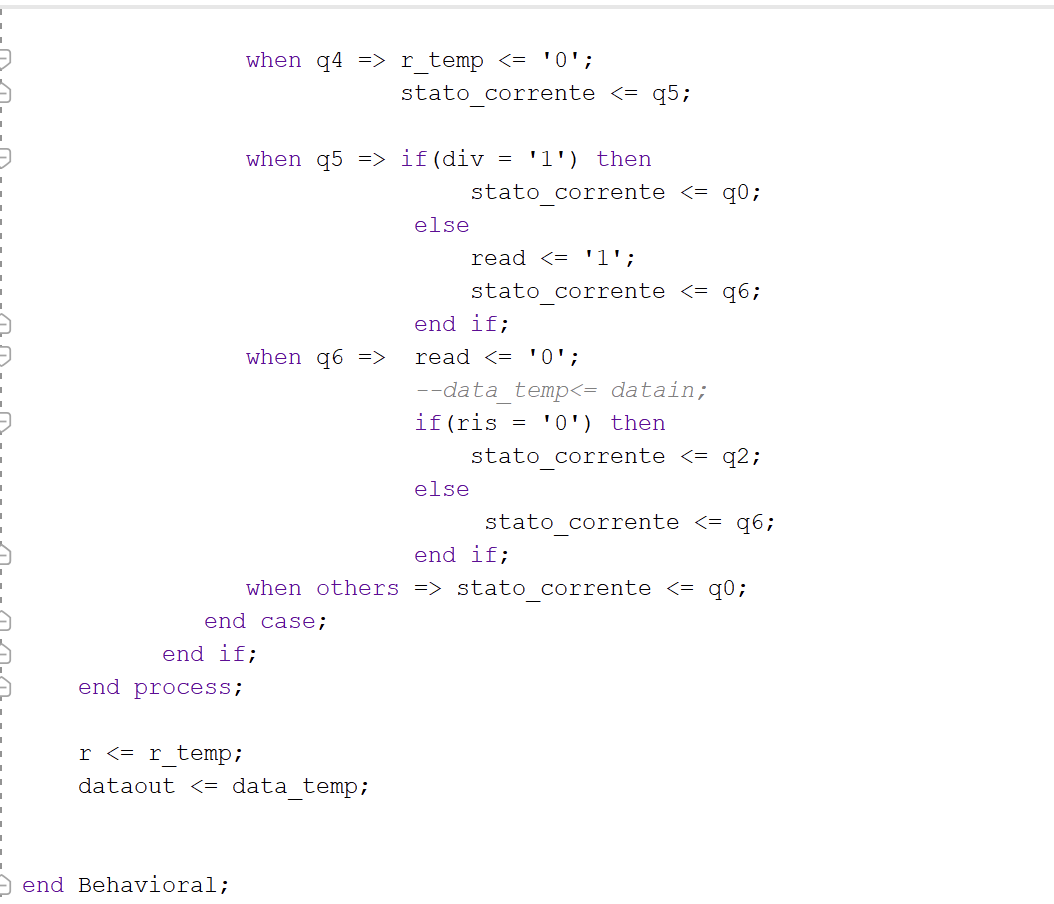
q4 -> quando ci si ritrova in questo stato, si è sicuri che il sistemaB ha ricevuto i dati inviati: si abbassa dunque il segnale di r e si passa allo stato q5.

q5 -> tale stato è sostanzialmente analogo a q0, solo che viene utilizzato per le trasmissioni successive alla prima. Si valuta quindi il segnale di div e, se alto, si procede verso q0, altrimenti si alza il segnale di read e si procede verso q6. È bene notare che, tali controlli sul segnale di div, potevano essere fatti anche in q4; tuttavia, dato che il contatore si incrementa sul segnale di ris in ingresso al sistema, si è preferito aggiungere uno stato per essere sicuri di vedere tale segnale stabile.

q6 -> se ci ritroviamo in questo stato, vuol dire che si deve procedere ad una trasmissione successiva alla prima. Per questo motivo si controlla il segnale di ris: se questo è basso, allora la precedente elaborazione è terminata e si può procedere ad inviare i nuovi dati, passando allo stato q2, mentre, se alto, si permane in q6 in attesa che si abbassi.

Immagine che contiene testo

Descrizione generata automaticamente



Componente contatore:

Immagine che contiene testo

Descrizione generata automaticamente

è descritto come una macchina a stati che conta il numero di comunicazioni effettuate. A partire da un segnale di ris, il contatore conta quante volte questo si alza, ossia i fronti di salita:

q0 -> stato in cui se il segnale in ingresso al contatore è basso allora resto in q0, altrimenti effettuo l'incremento del contatore in modulo N e mi sposto nello stato q1. Se raggiungo la fine del conteggio ossia count = N-1 allora alzo DIV (segnale di stop).

q1 -> stato in cui aspetto che il segnale in ingresso si abbassi: se rimane alto, allora resto in q1, altrimenti passo in q0;

Immagine che contiene testo

Descrizione generata automaticamente

Componente Rom:

Immagine che contiene testo

Descrizione generata automaticamente

è progettato come una semplice memoria, che sul fronte di salita del clock valuta il segnale di read: se questo è alto, pone in uscita il dato presente alla locazione indicata dal valore del counter.

Immagine che contiene testo

Descrizione generata automaticamente

Sistema B

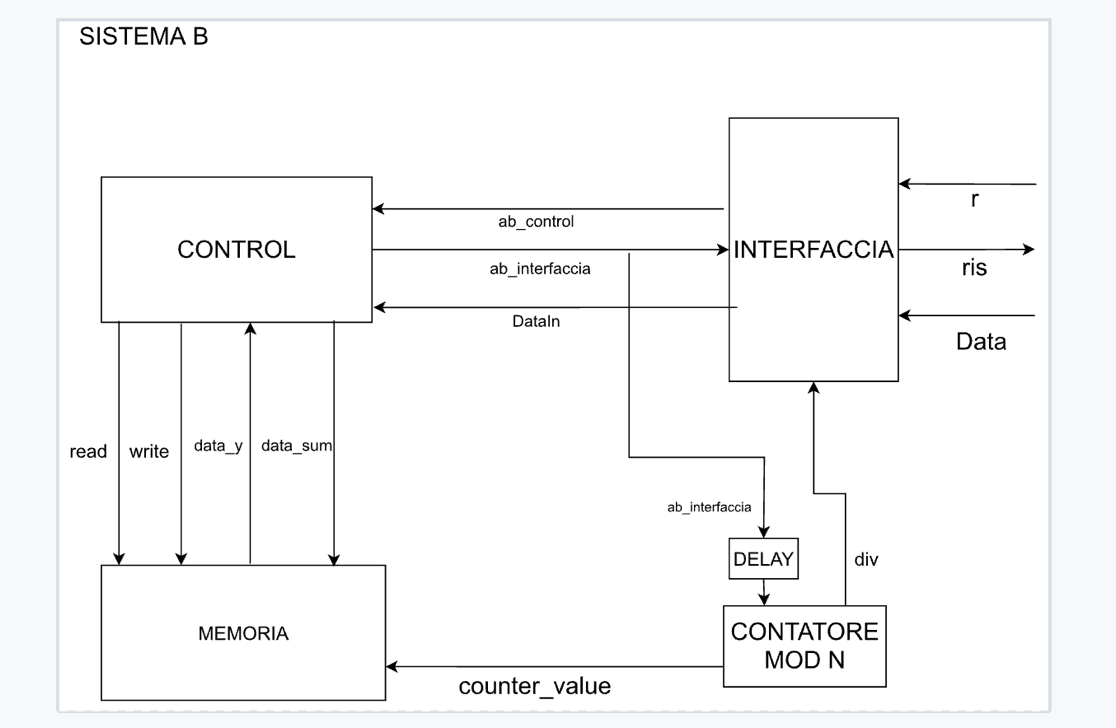
il sistemaB è stato definito tramite un approccio strutturale, utilizzando i seguenti componenti:

- un’interfaccia, la quale comunica con il sistema A, gestisce i segnali di r e ris e acquisisce i dati da elaborare;

- un contatore per indicare la terminazione dei dati da ricevere;

- una control unit, che ha il compito di effettuare le operazioni sui dati ricevuti e di salvarli in memoria;

- una memoria unica con i dati da sommare a quelli ricevuti dal sistema A e ulteriori locazioni per salvare i risultati;



Componente Interfaccia:

Immagine che contiene testo

Descrizione generata automaticamente

l’interfaccia è stata definita come una macchina a 5 stati, dove:

q0 -> è lo stato in cui si trova l’interfaccia prima di ricevere il primo dato da elaborare. In questo stato, si attende la variazione del dato in input r, ricevuto dal sistema A, la cui variazione da 0 a 1 indica la disponibilità dei dati da elaborare. Finché r è basso, l’interfaccia permane in q0 e, quando rileva la variazione sul fronte di salita del clock, acquisisce i dati, alza il segnale di ris in uscita ( il quale viene visto dal sistema A ed è indice che i dati sono stati prelevati e che l’elaborazione di B è iniziata) ed abilita il segnale di ab\_control, che mette in attesa l’interfaccia stessa durante l’elaborazione dei dati da parte della control unit. Passa infine nello stato q1.

q1 -> in tale stato l’interfaccia permane finché non vede una variazione da 0 a 1 del segnale ab\_interfaccia, il quale viene fatto variare dalla control unit una volta terminata l’elaborazione dei dati ricevuti. Quando viene rilevata la variazione su tale segnale, l’interfaccia abbassa il segnale ab\_control, disabilitando così la control unit fino alla ricezione di nuovi dati. Passa infine nello stato q2.

q2 -> la prima operazione che si effettua in questo stato è la valutazione del segnale r, verificando se questo è ancora alto o meno. Tale operazione potrebbe risultare inutile, dato che il sistema A abbassa tale segnale nel momento in cui vede il segnale di ris alzarsi, ma così non è. Bisogna ricordare infatti che il sistema B lavora in genere con un clock a frequenza maggiore (o semplicemente con fase diversa) e potrebbe quindi svolgere le operazioni in maniera molto rapida e non dare il tempo al sistema A di abbassare tale segnale. Se non venisse effettuato tale controllo, negli stati successivi si potrebbe interpretare il segnale di r pari ad 1 come indice di nuova trasmissione, portando alla rielaborazione di dati già processati.

L’interfaccia pertanto rimane in tale stato finché non rileva una variazione da 1 a 0 del segnale di r; quando tale segnale varia, abbassa il segnale di ris e passa nello stato q3.

Valutare il segnale r prima di abbassare quello di ris risolve inoltre un altro problema: dato che il segnale di ris viene valutato dal sistemaA, così facendo, si fa in modo che duri almeno 1 colpo di clock di A, in modo che quest’ultimo possa rilevarne le variazioni, indipendentemente dal numero di operazioni effettuate dal sistema B.

q3 -> tale stato è il responsabile del controllo dei dati rimanenti da elaborare. Se il segnale di div in uscita dal contatore è alto, allora vuol dire che il dato appena elaborato era l’ultimo e l’interfaccia torna dunque nello stato q0. In caso contrario si procede verso lo stato q4.

q4 -> questo stato è sostanzialmente l’analogo dello stato q0, ma quando si sono già ricevuti dei dati. È necessario introdurre questo tipo di stato per effettuare un controllo sul numero di elaborazioni effettuate per determinare la terminazione di esse.

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

Componente Contro Unit:

Immagine che contiene testo

Descrizione generata automaticamente

La control unit è responsabile dell’elaborazione dei dati ricevuti e della collocazione dei risultati all’interno della memoria. Anch’essa è stata definita come una macchina a stati, dove:

q0-> è lo stato in cui la macchina permane fino alla ricezione del primo dato da elaborare, rimanendo in attesa della variazione da 0 a 1 del segnale di ab\_control modificato dall’interfaccia.

Quando rileva tale variazione passa allo stato q1 ed abbassa l’ab\_interfaccia.

q1-> in tale stato si ripone il segnale di read a 1 e si passa allo stato q2.

q2-> dato che il segnale di read si vuole che duri un singolo colpo di clock, in questo stato viene abbassato e si passa poi allo stato q3.

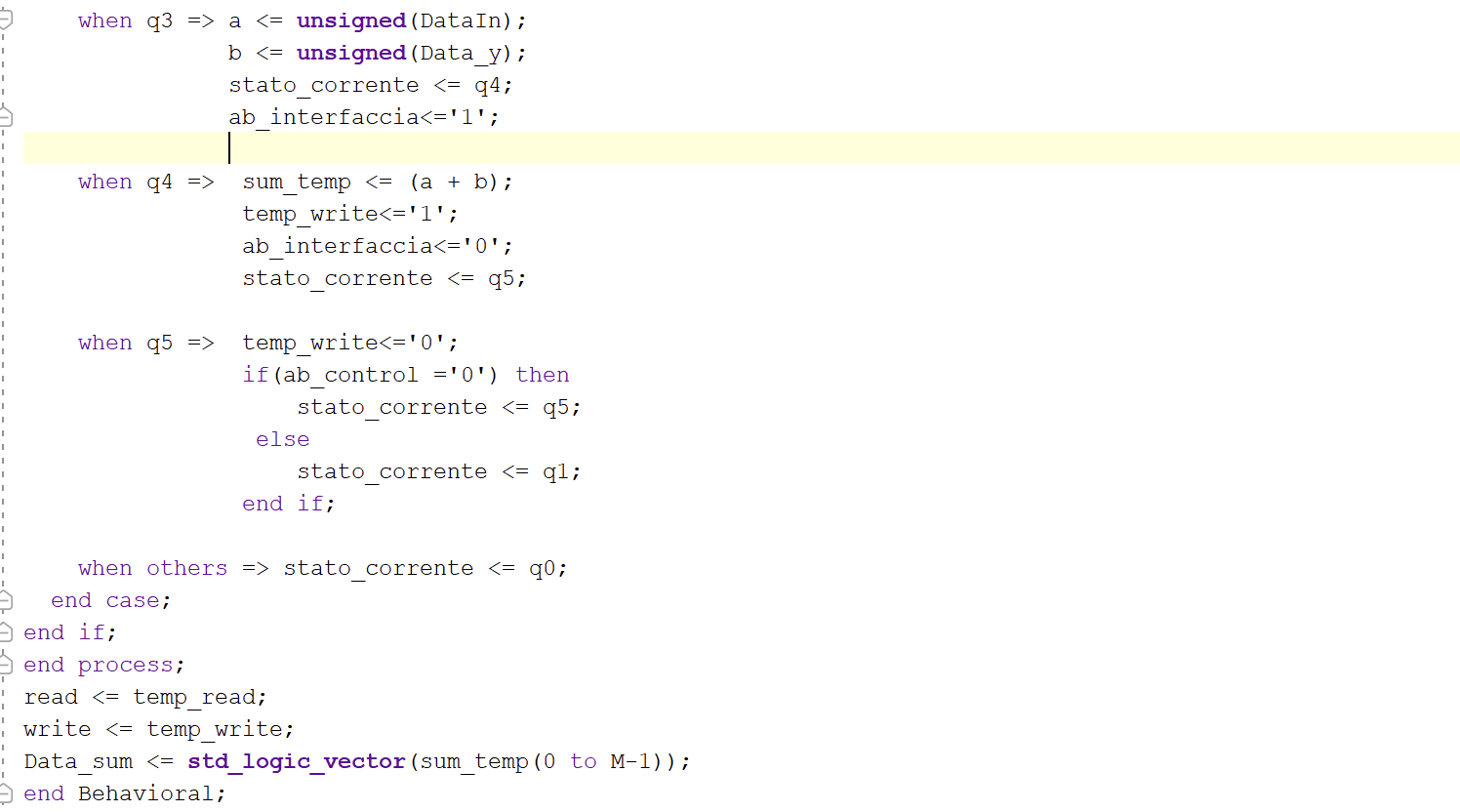
q3-> in tale stato si è sicuri che i dati da elaborare sono entrambi disponibili e corretti, motivo per cui vengono immagazzinati come unsigned all’interno di appositi signal, tramite i quali si eseguirà poi una operazione di somma. In questo stato si pone a 1 il segnale di ab\_interfaccia, il quale verrà letto al colpo di clock dopo dall’interfaccia che abbasserà il segnale di ris. Anche se l’elaborazione non è del tutto terminata, si inizia a dare tale segnale dato che al colpo di clock dopo si avrà effettivamente la completa elaborazione e, comunque, il segnale ab\_interfaccia verrà rilevato al colpo di clock successivo. Si passa infine allo stato q4.

q4-> viene alzata la write e posto sul segnale s il valore della somma, il quale verrà salvato nella memoria; si pone inoltre a 0 il segnale di ab\_interfaccia. Quest’ultima operazione può risultare ambigua, dato che porre a 0 tale segnale vuol dire disabilitare l’interfaccia, che in questo caso però deve essere attiva e pronta a ricevere ed inoltrare nuovi dati. Ciò che è bene notare, però, è che l’interfaccia valuta il segnale di ab\_interfaccia solo in alcuni stati e, la variazione di tale segnale permette comunque di proseguire verso alcuni stati, ma la mette in attesa negli stati in cui è giusto che l’interfaccia si fermi una volta ricevuti i dati. Stiamo sostanzialmente dicendo all’interfaccia che, quando arriverà nuovamente in uno di quegli stati dovrà fermarsi e non che in generale non deve lavorare. Si passa infine allo stato q5.

q5-> tale stato ha sostanzialmente lo stesso scopo dello stato q4 nell’interfaccia, ovvero quello di replicare lo stato q0 quando però non si è al primo dato da elaborare. Le operazioni sono dunque analoghe a quelle dello stato q0, con l’aggiunta della variazione da 1 a 0 del segnale di write, il quale, come il segnale di read, deve durare 1 colpo di clock.

Immagine che contiene testo

Descrizione generata automaticamente



Componente Counter:

tale contatore è impostato sul fronte di salita del clock e incrementa solo quando il segnale di ab\_interfaccia si alza, ovvero quando è terminata l’elaborazione del dato. Siccome tale segnale si alza un colpo di clock prima dell’effettiva terminazione, è stato introdotto un blocco di delay per salvare i valori nella giusta locazione di memoria.

Tale componente è analogo al contatore utilizzato nel sistema A, ovvero progettato come una macchina a stati; varia solamente il segnale che riceve in ingresso.

Componente Memoria:

Immagine che contiene testo

Descrizione generata automaticamente

Tale componente è stato descritto in modo comportamentale ed è uguale al componente utilizzato nell’esercizio 5. Tuttavia, per completezza, riportiamo nuovamente la sua implementazione.

Immagine che contiene testo

Descrizione generata automaticamente

Considerazioni finali:

Per come è stato implementato, il protocollo è efficiente con fB=N\*fA per ogni valore di N (si da per scontato che il clock di B permette l’assestamento dei segnali in 1 colpo di clock).

In questo modo, non serve che il componente master ed il componente slave si accordino sulle frequenze da utilizzare.

Nel caso in cui siano noti i riferimenti temporali dei 2 componenti, possono essere ridotti il numero di stati delle interfacce di A e di B.

Simulazione:

così come richiesto, è stata effettuata una simulazione del funzionamento del protocollo, con una frequenza fB=2\*fA:

Immagine che contiene testo, elettronico

Descrizione generata automaticamente

Immagine che contiene testo, elettronico

Descrizione generata automaticamente