Esercizio 5: cronometro

L’obiettivo di questo esercizio è quello di progettare un cronometro tramite un approccio strutturale, visualizzando i valori di ore, minuti e secondi sul display presente sulla board.

Il progetto prevede sicuramente l’utilizzo di una serie di contatori, i quali dovranno scandire il tempo.

Componente contatore:

Immagine che contiene testo

Descrizione generata automaticamente

Tale contatore è stato implementato tramite un approccio comportamentale, in modo da incrementare il valore di conteggio solo in presenza di un opportuno segnale di abilitazione.

Immagine che contiene testo

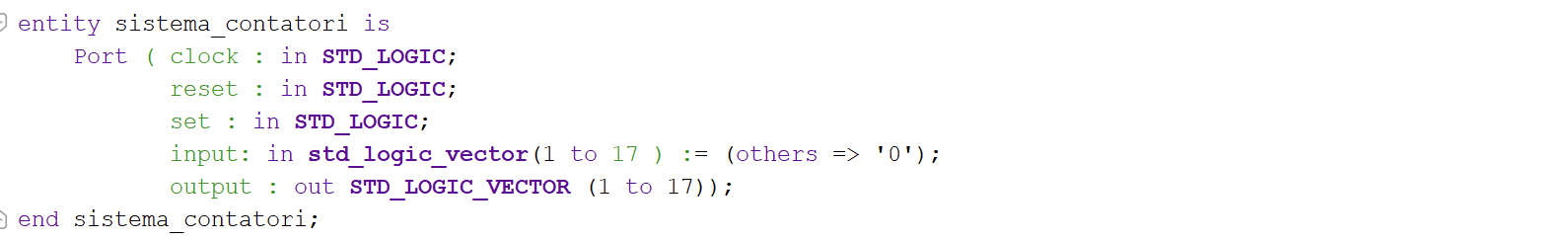
Descrizione generata automaticamente

Data la richiesta di poter settare i valori di secondi, minuti e ore da cui il cronometro deve partire, si è deciso di introdurre due ingressi aggiuntivi: un segnale di set ed un vettore d’ingresso.

Sostanzialmente, quando il segnale di set è alto, il contatore non rileva il segnale di abilitazione e carica il valore presente in ingresso; quando il segnale di set si abbassa, il contatore ritorna al suo normale funzionamento.

Una volta definiti tre contatori differenti per scandire ore, minuti e secondi, vediamo come questi sono stati combinati tra loro.

Sistema Contatori:



tale sistema di contatori è formato da 4 contatori, 1 blocco di delay ed un blocco di and.

Il primo contatore che ritroviamo è un divisore di frequenza, il quale ha il compito di passare dalla frequenza del clock della board a quella di 1HZ, la cui uscita di div funge da abilitazione epr il contatore dei secondi.

L’uscita di div del primo contatore entra come abilitazione del contatore relativo ai minuti; entrambi i contatori sono definiti modulo 60.

Leggermente più complessa è la connessione con il contatore relativo alle ore, il quale dovrà effettuare un conteggio solo quando sia il contatore dei secondi che quello dei minuti sono arrivati al termine del conteggio. Come segnale di abilitazione, dunque, si utilizza l’uscita di una porta and, che presenta come ingresso entrambi i segnali di div dei contatori precedenti.

Si è presentato però un problema di tempificazione: il segnale di div del contatore relativo ai secondi, fungendo da abilitazione al contatore dei minuti e rimanendo alto per 1 solo colpo di clock, si alzerebbe appunto un colpo di clock prima di quello dei minuti; si è realizzato pertanto un blocco di delay, il quale ritarda tale segnale di 1 colpo di clock, in modo che la porta and possa vedere entrambi i segnali alti. Se non avessimo usato tale blocco, il contatore delle ore non avrebbe mai visto il segnale di abilitazione alto.

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

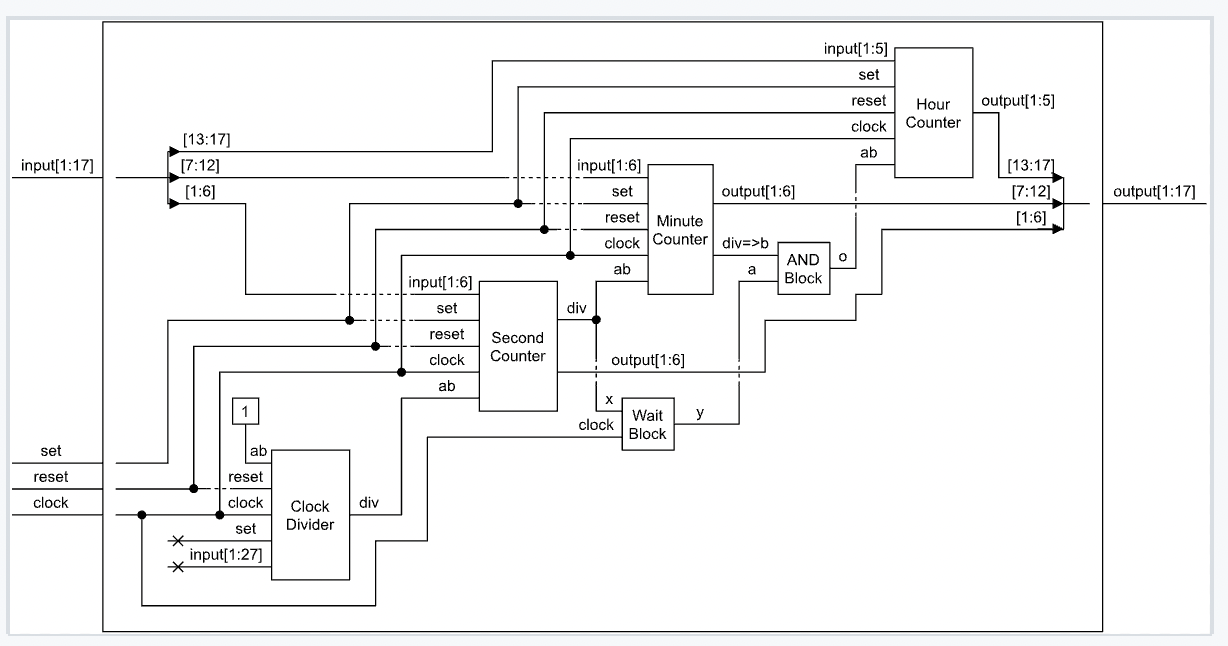
Component wait\_block:

è stato progettato mediante un approccio comportamentale, con lo scopo di ritardare di 1 colpo di clock il segnale in ingresso.

Immagine che contiene testo

Descrizione generata automaticamente

Schema sistema contatori:



Definita la parte relativa al conteggio, possiamo ora vedere i componenti necessari al set di un valore iniziale dei contatori, in modo da far partire il cronometro da un orario scelto.

Component gestore\_set:

Immagine che contiene testo

Descrizione generata automaticamente

Il segnale di set in ingresso al contatore viene processato da un gestore di set, il quale ha il compito di mantenere quel segnale sempre alto finché non vengono settate ore minuti e secondi.

Il segnale di set viene presentato in uscita in base ad un segnale fornito dall’esterno: la prima volta che si riceve tale segnale, il valore di set viene posto ad 1.

A questo punto, tutti i contatori fermano il conteggio e rimangono in attesa di un vettore in ingresso e, tramite una variabile di conteggio chiamata count, si gestisce quale contatore inizializzare ad ogni passo.

Il vettore in ingresso, infatti, contiene solo 6 bit e viene ogni volta aggiornato con il valore di ore, minuti e secondi che si vuole caricare; si è preferita una scelta di questo tipo rispetto ad un unico vettore contenente 17 ingressi dato che, quando si passerà alla sintesi su fpga, non saranno presenti 17 switch per gli input.

Tornando alla variabile count, vediamo come gestisce gli ingressi:

* Count=1 : i valori in ingresso vengono caricati nel counter dei secondi;
* Count=2 : i valori in ingresso vengono caricati nel counter dei minuti;
* Count=3 : i valori in ingresso vengono caricati nel counter delle ore;

quando il count arriva a 3, inoltre, il suo valore viene portato a 0, indicando la fine dei dati in input da ricevere.

Il valore di count viene incrementato ogni volta che arriva il segnale esterno che definisce il set; la quarta volta che il gestore riceve tale segnale, il valore di count sarà pari a 0 e, dunque, il segnale di set in uscita verrà abbassato e il cronometro riprende il suo normale funzionamento.

È bene notare che, tutte le operazioni del gestore modo vengono eseguite quando il segnale di ingresso ab\_modo è pari a 0; il motivo verrà spiegato nel seguito.

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

L’implementazione riportata è già adattata alla sintesi su fpga, introducendo dei valori di output relativi all’accensione di alcuni led, i quali indicano quale counter stiamo inizializzando.

Sintesi su fpga:

Così come richiesto nell’esercizio 5.2, il componente cronometro realizzato in precedenza verrà sintetizzato su fpga. In particolare, verrà utilizzato il display a 7 segmenti presente sulla board, per la visualizzazione dell’orario, e 2 bottoni, per inviare i segnali di set e reset.

Tuttavia, tale sintesi non può essere immediata, ma si necessita di 2 componenti aggiuntivi:

* Debouncer: componente che serve a ripulire il segnale inviato dal bottone, che sarà sicuramente caratterizzato da rumore.
* Convertitore per display: l’uscita dei contatori deve essere tradotta in un formato leggibile dal display.

Componente debouncer:

tale componete ha il compito di trasformare un segnale rumoroso in un segnale pulito.

Immagine che contiene testo

Descrizione generata automaticamente

Quando un segnale arriva da un bottone sarà sicuramente effetto da rumore, come riportato in figura. Ad un occhio umano, tali oscillazioni non vengono percepite ma, quando tale segnale viene analizzato ad un microcontrollore, queste vengono rilevate a pieno e potrebbero creare problemi (se tale segnale va in ingresso ad un contatore si traduce in conteggi spuri).

Il debouncer viene progettato come una macchina a stati:

* Stato not pressed: il sistema permane in questo stato finchè non vede il segnale in ingresso alzarsi. Passa così nello stato pressed.
* Stato pressed: il sistema permane in questo stato finchè il segnale non si abbassa. Quando ciò accade, si utilizza una variabile di conteggio, la quale fa si che passi un periodo di tempo pari a D2 prima di riportare il segnale come alto in uscita e tornare nello stato not pressed.

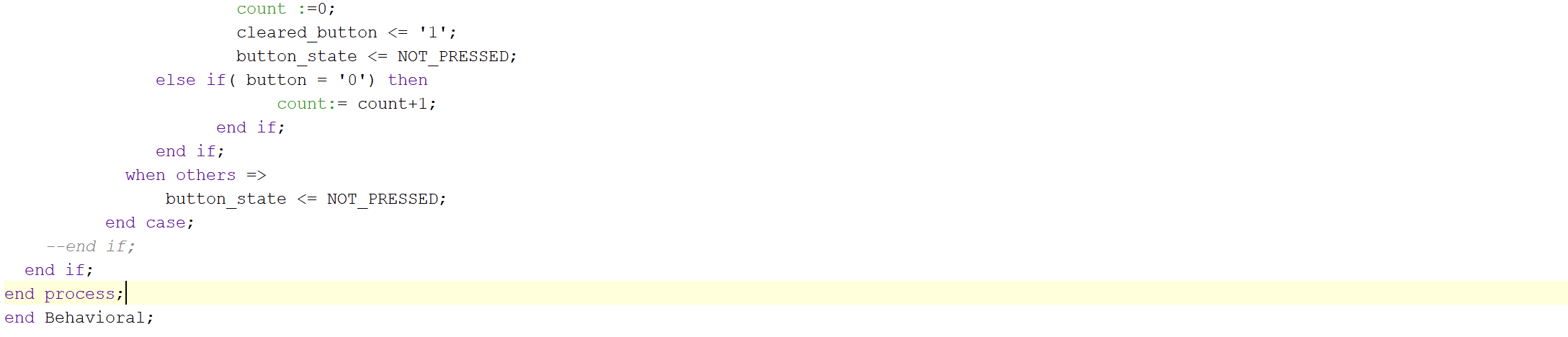
Il segnale in uscita rimane alto per un colpo di clock, dato che quando il sistema ritorna nello stato not pressed, l’uscita viene abbassata nuovamente. A differenza di un debouncer classico, l’uscita viene riportata alta solo dopo che il segnale si è abbassato e non dopo un tempo D1 dal fronte di salita; la scelta è stata fatta poiché, altrimenti, se il bottone venisse premuto troppo a lungo potrebbe essere rilevato nuovamente ed in uscita si produrrebbero 2 segnali puliti invece di 1.

Immagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente



Component conv\_per\_display:

tale componente è stato progettato con un approccio dataflow: prende in ingresso i vettori in uscita dai contatori e traduce ognuno in vettori 2 vettori da 4 bit (richiesti dal display in ingresso), i quali rappresentano i valori di decine e unità.

Immagine che contiene testo

Descrizione generata automaticamente

Le uscite di questo componente andranno poi in ingresso al display.

Estensione con intertempi:

per far si che il sistema sia in grado di memorizzare degli intertempi, sono stati aggiunti ulteriori componenti:

* Gestore modo: necessario per la visualizzazione degli intertempi salvati.
* Memoria: componente nel quale salvare gli intertempi.

Componente Gestore\_modo:

Immagine che contiene testo

Descrizione generata automaticamente

è stato realizzato come una macchina a 2 stati

* Stato cronometro: a tale stato è associata l’uscita ab\_modo=0, la quale indica dunque che l’uscita sul display è quella del sistema dei contatori, ovvero stiamo visualizzando il cronometro che scorre. Quando riceve in ingresso il segnale di cambio modo (corrispondente al cleared\_button, preso dall’uscita di un debouncer che ripulisce il bottone di cambio modo), si passa allo stato intertempo.
* Stato intertempo: a tale stato è associata l’uscita ab\_modo=1, la quale indica dunque che l’uscita sul display è quella della memoria, ovvero stiamo visualizzando gli intertempi salvati.

Per implementare la modalità intertempi, dunque, le uscite dei contatori e della memoria entrano nel gestore modo il quale, in base alla modalità di funzionamento del sistema, decide quale dei due ingressi riportare in uscita. Tale uscita andrà poi in ingresso al convertitore per display.

L’uscita ab\_modo verrà inoltre riportata in ingresso alla memoria ed al gestore set, in quanto ne limita le funzionalità:

* Gestore set: quando siamo nella modalità intertempo, non è possibile settare un valore nei contatori.
* Memoria: quando si è nella modalità cronometro e si invia un segnale di write, allora l’intertempo viene scritto in memoria, altrimenti no. Inoltre, i dati possono essere letti dalla memoria e riportati sullo schermo, in corrispondenza di un segnale di read, solo quando il sistema è in modalità cronometro.

È bene notare che il segnale di ab\_modo non va in ingresso al sistema di contatori. Si è infatti supposto che, anche quando si è in modalità intertempo, il cronometro continua a lavorare, ma il tempo non viene mostrato sul display. Quando si ripassa alla modalità cronometro, infatti, si noterà che il valore di conteggio sarà avanzato.

Immagine che contiene testo

Descrizione generata automaticamente



Componente memoria:

Immagine che contiene testo

Descrizione generata automaticamente

tale componente è stato progettato tramite un approccio comportamentale e varia le sua funzionalità in base alla modalità di lavoro del sistema:

* Modalità intertempo: permette, in corrispondenza di un segnale di read, di scorrere gli intertempi salvati. Non si possono salvare nuovi intertempi in tale modalità.
* Modalità cronometro: permette, in corrispondenza di un segnale di write, di salvare l’intertempo in memoria. Non si possono visualizzare gli intertempi salvati in tale modalità.

Per come è stata progettata, se si salvano più di N intertempi, vengono sovrascritti i primi che sono stati salvati.

Immagine che contiene testo

Descrizione generata automaticamente

Schema finale: