Esame Emanuele Riccio M63/1339

Traccia

Progettare, implementare in VHDL e simulare la seguente architettura

Un nodo A è alimentato da 2 ROM di N byte. Il nodo A trasmette al nodo B il valore ottenuto sommando gli elementi delle due ROM in posizioni omologhe. Il nodo B a sua volta è dotato di due moduli di memoria MEM1 e MEM2: i byte ricevuti da A sono memorizzati in MEM1 se positivi, mentre vengono memorizzati in MEM2 se nulli o negativi. Progettare il sistema utilizzando in ciascun nodo un componente contatore e un componente multiplexer e inserendo un sommatore strutturale in A.

Nodo A

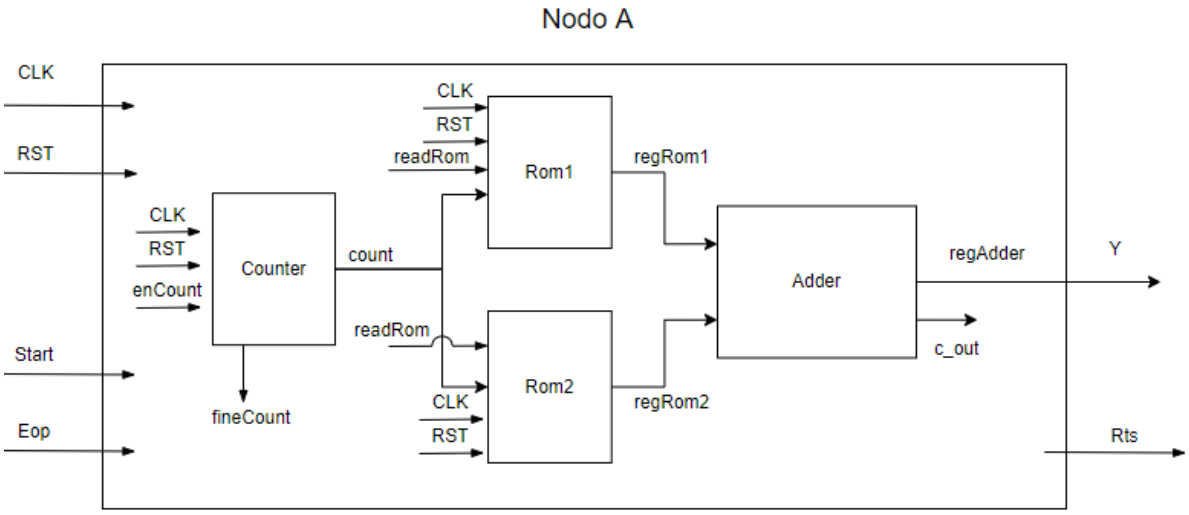

Figura 1: Schema del nodo A

Tale nodo è stato realizzato a livello strutturale, tramite la composizione di 4 blocchi elementi:

- **Rom**: sono presenti 2 Rom, le quali contengono rispettivamente N stringhe di 8 bit, precaricare nelle memorie, successivamente verrà fatta la somma algebrica delle stringhe in posizione omologa tramite l'Adder, ed inviato il risultato al Nodo B. Ogni Rom è dotata di un segnale di abilitazione per la lettura, il segnale readRom è fornito in parallelo alle due Rom e ne abilita la lettura, ogni Rom a seguito dell'abilitazione immette in output la stringa che ha memorizzato alla locazione count.
- **Counter**: questo blocco si occupa di calcolare la locazione di memoria a cui accedere per prelevare le 2 stringhe, è utilizzato come una sorta di indice. Il counter è abilitato dal segnale enCount, mentre fornisce in output il segnale di conteggio in parallelo alle 2 Rom.
- **Adder**: questo blocco è un RippleCarryAdder realizzato in modo strutturale, componendo 8 FullAdder, si occupa di effettuare la somma algebrica tra 2 stringhe selezionate ogni volta. Il Ripple Carry Adder preleva i suoi operandi dalle 2 Rom, in particolare l'operando A dal blocco Rom1 tramite il segnale regRom1, e il secondo operando dalla Rom2 tramite il segnale regRom2.

Automa di controllo Nodo A

In particola è stata realizzata tramite un automa(Figura 2) la logica di controllo dell'architettura in Figura 1

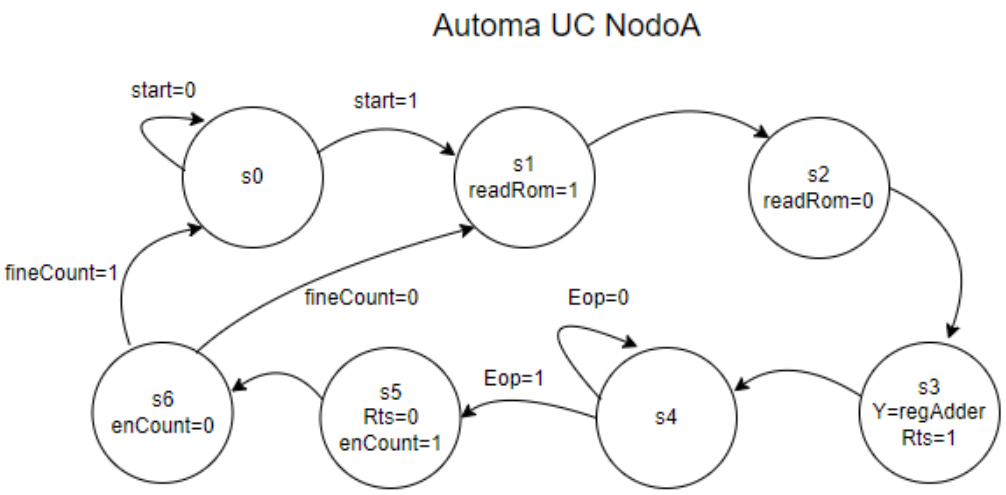

Figura 2: Automa per il controllo dell'architettura Nodo A

Analizziamo l'automa, descrivendo i suoi stati:

- **S0:** in questo stato si attende finchè non viene fornito dall'esterno il segnale di Start=1, che da il segnale di avvio alla macchina, finchè Start=0 si rimane nello stato s0
- **S1:** in questo stato di abilita la lettura delle 2 Rom, fornendo in ingresso ad entrambe il segnale di abilitazione readRom=1, ogni Rom darà in output la stringa contenuta alla locazione indicata dal valore count, le due uscite saranno messe in ingresso al RippleCarryAdder che ne farà la somma algebrica, si transita allo stato s2. **OSSERVAZIONE:** quando mettiamo readRom=1, il valore verrà davvero attribuito dopo un tempo delta(prima del clock succ)(vedi dispense drive), quindi solo nello stato S2(cioè al clock successivo) le Rom vedranno il segnale alto(perché le Rom sono sensibili solo al clock) ed immetteranno le locazioni in output Rom1 =>a, Rom2=>b, anche in questo caso i segnali a e b saranno modificati dopo un tempo delta, ma prima del clock successivo, in particolare mentre si transita da S2 ad S3 i segnali a e b saranno modificati, dato che il RippleCarryAdder è sensibile ad a e b e NON AL CLOCK(process(a,b), istantaneamente si attiverà producendo il risultato immettendolo in regAdder, arrivati in S3 quindi siamo sicuri di trovare il risultato in regAdder. Se il rippleCarryAdder non fosse stato sensibile ad a e b, ma solo al clock(process(clk)(che in questo caso non ha), lo schema non avrebbe funzionato, poiché il ripplecarryadder avrebbe visto a e b solo nello stato s3, quindi in s3 non avremmo avuto il risultato. Per ridurre il numero di stati e velocizzare il tutto si sarebbe potuta fare la Rom sensibile anche a readRom (process(readRom, clock).
- **S2:** si disabilita il segnale di abilitazione readRom, settando readRom=0, cosi' che il segnale di abilitazione abbia la durata di un singolo colpo di clock, si transita nello stato s3
- **S3:** a questo punto si avvia la trasmissione, tramite il segnale Y si da in output al nodo B il risultato fornito dall'Adder contenuto nel registro regAdder, Y<=regAdder, la comunicazione avviene in parallelo infatti Y è composto da 8 linee, infine si abilita alto Rts=1, cosi da segnalare l'inizio della comunicazione al nodo B e si transita nello stato s4

- **S4:** in questo stato si attente che il nodo B risponda di aver terminato l'elaborazione tramite il segnale Eop, finchè il segnale Eop=0 si rimane nello stato s4(autoanello), altrimenti se Eop=1 significa che B ha terminato l'elaborazione e quindi si transita nello stato s5.
- **S5:** si abbassa il segnale Rts=0, e si da il segnale di enCount=1 per abilitare il contatore, in modo da incrementare di 1 il valore di conteggio, cosi' da accedere alla locazione successiva delle Rom nella prossima trasmissione. Si transita in s6
- **S6:** per prima cosa il segnale enCount viene abbassato enCount=0, in modo da far durare il segnale di abilitazione del contatore un solo colpo di clock, in modo che venga incrementato di una sola unità il valore di conteggio, infine si verifica se sono state effettuate tutte le trasmissioni controllando il flag fineCount, il quale viene fornito in output dal contatore, ed indica se il valore di conteggio ha raggiunto il suo massimo o meno, se fineCount=0 significa che il valore di conteggio non ha ancora raggiunto il suo massimo, quindi ci sono altre comunicazioni da fare, si transita quindi nello stato s1, altrimenti se fineCount=1 il valore di conteggio è stato resettato a 0, quindi si transita in s0 poiché non ci sono più trasmissioni da fare.

Nodo B

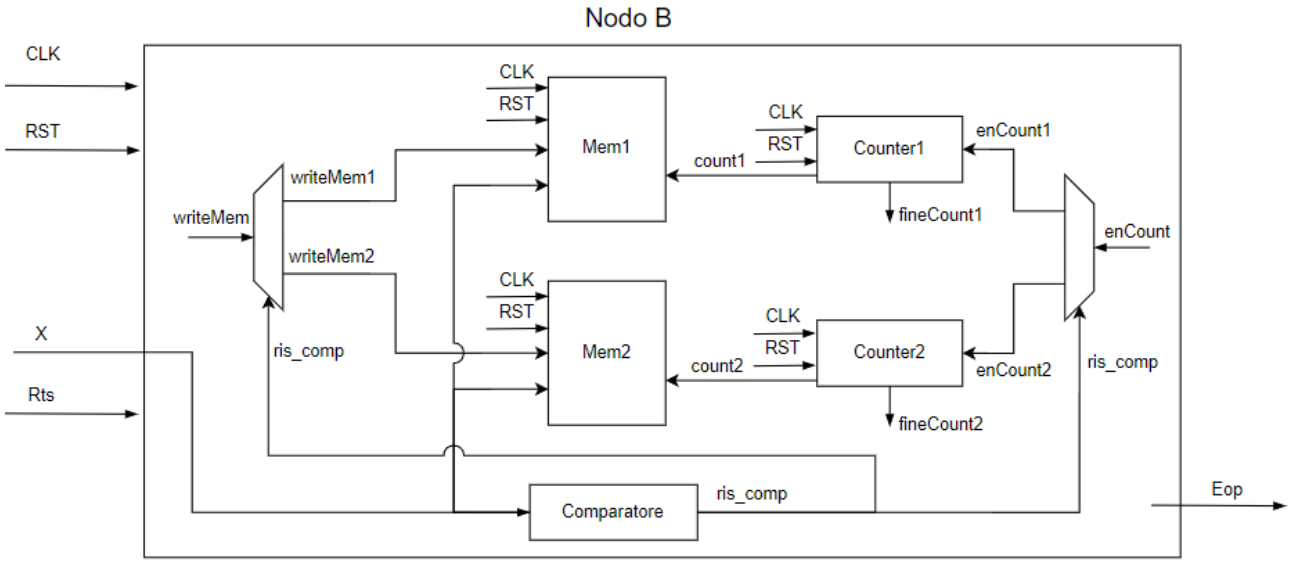

Figura 3: Schema del nodo B

Tale nodo è stato implementato tramite la composizione di 7 blocchi:

- **Memorie**: sono presenti 2 memorie diverse, poiché la stringa di 8 bit X ricevuta dal nodo A, deve essere salvata nella memoria Mem1 nel caso la seq di bit abbia segno positivo e nella Mem2 altrimenti. Infatti ogni memoria ha il suo counter che tiene traccia della locazione libera su cui eventualmente scrivere, inoltre ogni memoria prende in ingresso in parallelo la stringa di bit X, ed ha un proprio segnale di abilitazione Mem1 ha writeMem1, mentre Mem2 ha writeMem2, quindi il valore in ingresso X viene dato ad entrambe, ma tramite i due segnali viene abilitata alla scrittura solo una delle due memorie.
- **Comparatore:** questo componente prende in ingresso la stringa di 8 bit X, ricevuta dal nodo A e ne controlla il segno, questo componente mette in output ris_comp=1 se il segno della stringa X è negativo oppure se la stringa X ha tutti bit nulli, mentre il risultato è ris_comp=0 se il segno della stringa X è positivo(come descritto la seq X=00000000 viene considerata negativa). Infine il segnale di uscita del comparatore è utilizzato per pilotare due demultiplexer
- **Demux**: sono presenti due demultiplexer, pilotati entrambi dal risultato del comparatore, quindi il segnale di selezione dipende dal segno della sequenza di bit X, il primo demultiplexer(sinistra) si

occupa di inviare alla memoria selezionata il segnale di writeMem utilizzato per abilitare la scrittura, mentre il secondo demultiplexer(destra) serve per inoltrare al contatore della memoria selezionata il segnale di conteggio enCount, in modo da aumentare il valore di conteggio solo del contatore corretto. In particolare quando il segnale di selezione è ris_comp=0, viene selezionata Mem1 e Counter1, altrimenti Mem2 e Counter2

- **Contatore:** sono presenti due contatori distinti, uno per ogni memoria, sono abilitati dal secondo demultiplexer come descritto sopra.

Automa di controllo Nodo B

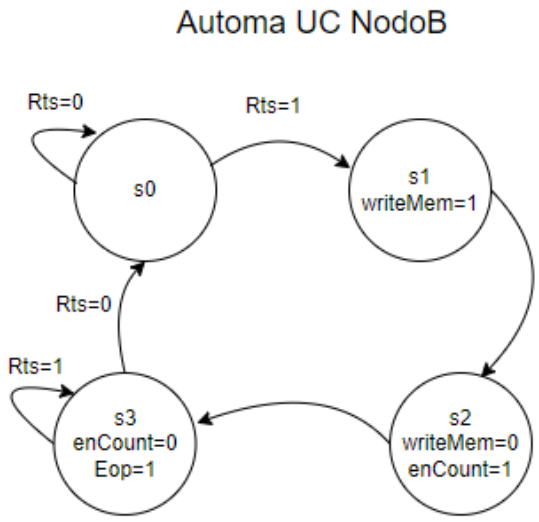

Figura 4: Automa per il controllo dell'architettura del nodo B

Analizziamo l'automa sottostante stato per stato:

- **S0**: in questo stato viene resettato il segnale Eop, si attende che sia abilitato alto Rts=1, cioè il segnale di via da parte del trasmettitore, si rimane in s0 finchè Rts=0 (autoanello), quando Rts=1 si transita in s1.
- **S1**: in questo stato è disponibile la stringa di 8 bit X, inoltre il comparatore ha fornito in output il segnale di selezione che va in ingresso ai due demultiplexer, quindi si passa ad abilitare il segnale di scrittura writeMem=1, il quale verrà inoltrato alla Memoria selezionata dal primo demultiplexer, infine si transita nello stato s2.
- **S2**: si disabilita il segnale di scrittura writeMem=0, cosi' da farlo durare un solo colpo di clock e si abilità quello per far incrementare il valore di conteggio enCount=1, anche questo segnale è inviato al Contatore selezionato dal secondo demultiplexer, infine si transita al passo S3. Si è preferito non abilitare simultaneamente il segnale writeMem ed enCount, ma di abilitare in modo sequenziale, per non far incrementare il valore di conteggio durante la scrittura.

- **S3**: si disabilita il segnale enCount, cosi' da far incrementare di una sola unità il valore di conteggio e si abilita alto Eop=1, per indicare all'unità A che l'unità ha terminato l'elaborazione. A questo punto l'unità A disabiliterà il segnale Rts=0, si rimane in S3 finchè il segnale Rts=1 (autoanello), in modo da dare il tempo all'unità A di disabilitarlo, quando Rts=0 si transita nuovamente in S0.

Simulazione

Le due Rom del nodo A, sono state caricate rispettivamente con 5 stringhe da 8 bit

Rom 1	Rom 2	Risultato
00000011 (3)	11110111 (-9)	(-6)
00000110 (6)	00000101 (5)	11
00001000 (8)	11110111 (-9)	-1
00010000 (16)	11110000 (-16)	0
00100000 (32)	00000001 (1)	33

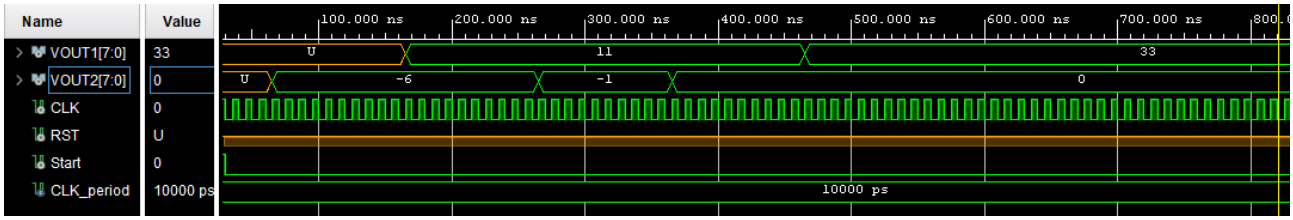

Figura 5: Simulazione

Come possiamo osservare, i due segnali VOUT1(8 bit) e VOUT2(8 bit) rappresentano rispettivamente l'ultimo risultato salvato dalla Mem1 e la Mem2, i risultati sono salvati nelle rispettive memorie correttamente, 11 e 33 nella Mem1 poiché positivi, mentre -6,-1,0 nella Mem2 poiché negativi o nulli.