

Progetto Reti Logiche

Francesco Ostidich: 10716798

A.A. 2022/2023

Indice

1	Introduzione	2
2	Architettura	3
2.1	Registri	4
2.2	Input reader	4
2.3	Selettore	5
2.4	Output writer	5
2.5	Accesso memoria e selezione uscita	6
2.6	Segnali di controllo	6
3	Risultati sperimentali	7
4	Conclusioni	8

1 Introduzione

Il circuito da implementare vuole, in seguito alla lettura di un indirizzo da input sequenziale, accedere ad un organo di memoria e riportarne il valore vettoriale su un'uscita prestabilita.

L'architettura qui proposta è in grado di riportare il valore richiesto sull'uscita corretta senza tempi di attesa. Infatti, in coincidenza del falling-edge del segnale di start, il segnale di done si trova sul rising-edge, con il valore da mostrare esposto sull'uscita designata.

Il processo è da suddividere in vari passaggi. Partendo dalla lettura degli ingressi w e $start$, il primo obiettivo è suddividere l'input sequenziale in selezione della porta di uscita (z_1, z_2, z_3, z_4) e indirizzo di memoria (mem_addr). Se il segnale di start diventa basso prima che l'indirizzo di memoria sia comunicato per intero, si necessita estendere il valore con zero. Tuttavia, essendo l'input letto partendo dall'MSB, un registro a scorrimento SIPO inizializzato a zero svolge tale compito senza richiedere operazioni aggiuntive.

Successivamente l'indirizzo è letto dalla memoria, la quale ha come segnale di mem_enable lo stesso $start$: tale combinazione permette alla memoria di avere l'output pronto per il fronte di discesa del segnale che rende l'input idoneo.

Il valore è inserito in un demultiplexer che, in base ai bit di selezione, spedisce il vettore al registro da 8 bit indicato. Essendo questo registro composto da flip-flop, l'informazione scritta non può essere visualizzata in modo immediato. Un multiplexer intercetta l'output del registro segnalato per la scrittura, imponendo direttamente il segnale mem_data sull'uscita proposta (un and-gate tra tale valore e il segnale di done nasconde l'output quando non richiesto). L'enable del registro e la selezione del MUX sono specificati da un decoder.

2 Architettura

Il circuito qui proposto si avvale di molteplici componenti, di seguito elencati.

- Demultiplexer n°2: instrada l'input nei registri a scorrimento.
- Registro 15 bit: contiene il valore dell'indirizzo.
- Registro 3 bit: racchiude i bit di selezione.
- Multiplexer n°2: stabilisce i bit corretti da leggere per la selezione.
- Demultiplexer n°1: convoglia l'informazione sulla corretta porta.
- Decoder: gestisce segnali di controllo per l'uscita eletta.
- Memory: organo di memoria, già istanziato.
- Registro 8 bit: ricorda i valori richiesti in precedenza.
- Multiplexer n°1: anticipa il registro selezionato esponendo direttamente l'informazione dalla memoria.

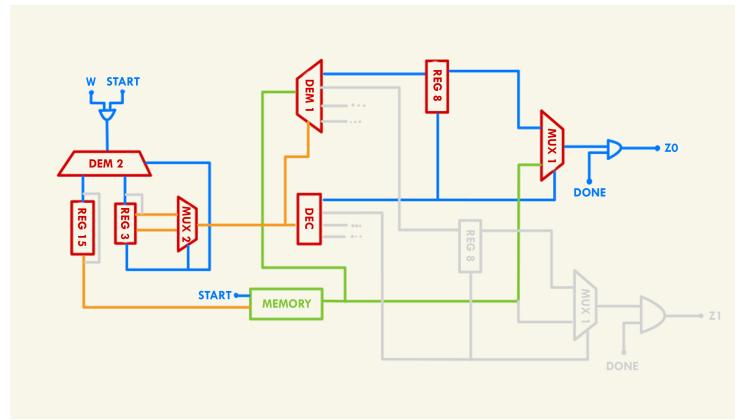


Figura 1: Architettura completa.

2.1 Registri

I registri utilizzati sono in totale sei: un registro SIPO da 15 bit per salvare l'indirizzo di memoria, un registro SIPO da 3 bit per designare una porta di output, e quattro registri PIPO da 8 bit contenenti i valori di output.

Concetto fondamentale per garantire la prontezza della macchina è la lettura senza intermedi dell'input, nel registro a 15 bit. Il 16° bit è infatti l'input stesso proveniente dal demultiplexer. Il registro a 3 bit gode della stessa proprietà, nell'edge-case di due soli cicli di start.

Come si può vedere in figura 2, gli shifting-registry (a sinistra) sono costruiti con flip-flop in serie: ad ogni ciclo di clock l'informazione scorre di una posizione. Un registro parallelo (a destra) è invece un banco di flip-flop che si pone da ponte tra entrata vettoriale e uscita di dimensione congruente.

Reset, clock ed enable sono segnali necessari da trasmettere come direttive al registro. Il clock utilizzato dai registri coincide con il clock unico del sistema, mentre reset ed enable variano da registro a registro.

In alcune sezioni successive (2.3, 2.5, 2.6) si espongono alcuni segnali generati e qui utilizzati (*done*, *done1*, *done2*, *sel*, *dec*); i registri utilizzano tali segnali ai fini di una corretta sincronizzazione collettiva, in particolare:

- registro 15 bit: il reset è un *or* tra *rst* e il segnale di *done2*; l'enable è sempre alto;
- registro 3 bit: il reset coincide con quello del precedente, mentre l'enable è *rst or start and not sel*. Si noti la configurazione di reset a "100";
- registro 8 bit: il reset è congruente a *rst*; l'enable è *rst or done1 and dec*.

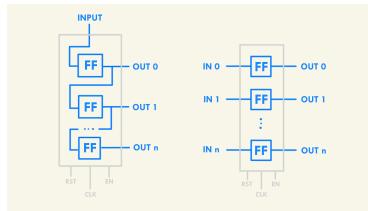


Figura 2: Registri da 15, 3 e 8 bit.

2.2 Input reader

Il segnale di input *w*, letto in *and* con *start*, entra in "input reader", il quale restituisce l'indirizzo a 16 bit e l'identificazione del canale d'uscita a 2 bit.

In ingresso un demultiplexer scinde le due informazioni, facendo combaciare il bit di controllo con l'MSB del registro a 3 bit (contenuto nell'entità "select"). Quest'ultimo è stato infatti inizializzato a "100", permettendo, facendo scorrere il primo bit, di compiere una funzione analoga ad un "counter one-hot"; con il riempimento del registro, la selezione del DMUX si inverte e il registro non cicla più, mantenendo il valore.

L'indirizzo è di 16 bit pur avendo un registro a 15 poiché l'input è esso stesso il bit mancante. La stessa strategia sarà poi applicata, in parte, nel componente "select", approfondito in seguito.

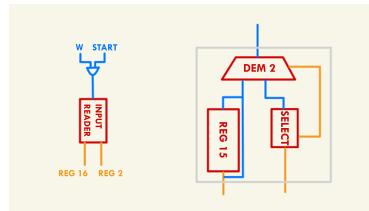


Figura 3: Input.

2.3 Selettore

Il selettore, che circoscrive il registro a 3 bit, è indispensabile per ovviare un caso limite ben preciso: se il segnale *start* ha durata minima (2 cicli) il registro non esporrebbe per tempo il valore domandato. Il multiplexer valuta quindi se sia necessario scorrere la lettura, leggendo direttamente l'input.

Il MUX adotta lo stesso segnale di selezione del DMUX della sezione precedente: l'ultimo bit del registro (*sel*) diviene alto solo se il registro si completa.

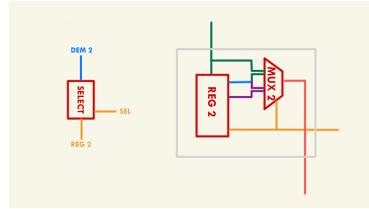


Figura 4: Selettore di uscita.

2.4 Output writer

Un settore, dedicato ad esporre i vettori in uscita, regola l'output da scrivere, in base che questo sia da ricevere dal registro parallelo, o necessiti un bypass che esibisca direttamente *mem_data*.

Nel caso la porta in analisi sia la designata alla modifica, nella prima metà di ciclo con *done* alto (*done1*), la selezione del multiplexer è l'uscita disposta dal decoder, e si legge quindi direttamente dalla memoria, non essendo il registro pronto; nella seconda metà (*done2*), quando il decoder si aggiorna, si vuole di conseguenza leggere dal registro parallelo: una porta *and* allaccia il controllo del MUX con *not done2*.

Infine un and-gate nasconde l'uscita con il done basso.

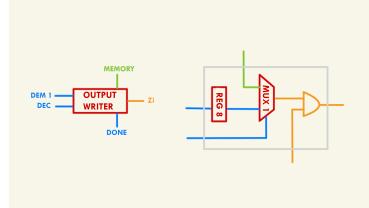


Figura 5: Output.

2.5 Accesso memoria e selezione uscita

In seguito all'inserimento dei valori sequenziali nei registri (sezione "input reader"), la memoria riceve l'indirizzo e presenta il valore. Tale valore è inoltrato ad un demultiplexer, il quale lo instrada al registro parallelo dell'uscita appropriata, che lo memorizzerà.

I bit di selezione sono invece diramati al vettore di controllo del DMUX, e al decoder che indica (segnalet *dec*) la porta corretta.

Demultiplexer, decoder e memoria immettono le proprie uscite in ciascuno dei quattro componenti "output writer", i quali si prendono carico di inserire l'informazione da scrivere sui canali di uscita.

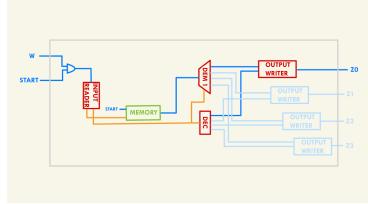


Figura 6: Percorso da input ad output.

2.6 Segnali di controllo

Per moderare i componenti, alcuni segnali di controllo vengono generati da una porzione di circuito adibita a tale finalità.

Il fronte di discesa dello *start* è captato da un falling-edge-detector che esprime l'evento con un segnale *done1*. Tale segnale resta tuttavia attivo per una durata dimezzata rispetto al *done* effettivo che è richiesto; un secondo falling-edge-detector, attivo sul ciclo di clock invertito e che legge *done1*, permette di introdurre un segnale *done2* complementare, al fine di comporre un *done* completo (tramite un or-gate).

Riproponendo le finalità, a grandi linee, dei succitati, *done1* si occupa di sincronizzare la prima partizione di circuito, adibita alla comunicazione con la memoria ed ai reset dei registri, mentre *done2* autorizza l'informazione ad essere mostrata sui canali di output. Il *done* di fine elaborazione è composto come *done1 or done2*.

In generale, un qualsiasi segnale di controllo viene ricavato da un segmento di circuito, il quale promette una variazione in vista di un evento. Operando e modellando quest'ultima ai fini dell'utilizzo che si esige, si può regolamentare un altro segmento che richiede supervisione.

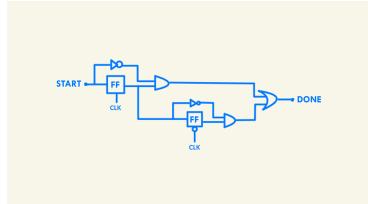


Figura 7: Generazione segnali di controllo.

3 Risultati sperimentali

Il circuito è stato sottoposto a svariati testbench, variegati per livello di stress prodotto e per edge-case esaminati. Il circuito ha soddisfatto ogni genere di simulazione: supera i TB behavioral, post-synthesis (functional e timing) e post-implementation (functional e timing).

L'utilizzo delle simulazioni è fondamentale per ricercare opzioni risolutive alle complicanze che sorgono, spesso a causa della sensibilità agli edge-case.

Di seguito alcuni waveform relativi ai testbench adoperati, in particolare: la prima coppia è fornita dal professore, la seconda è stata trovata in rete, mentre la terza è stata costruita personalmente.

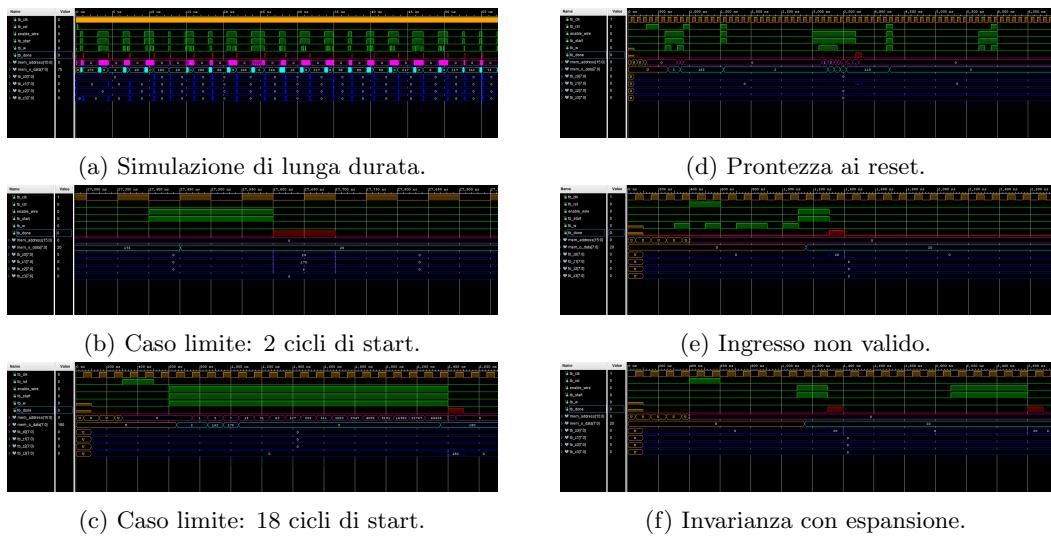


Figura 8: Waveform estratti da testbench.

4 Conclusioni

L'architettura proposta è efficiente per quanto riguarda il tempo di risposta. Infatti, indipendentemente dalla combinazione d'ingresso inserita, il circuito è sempre in grado di completare l'elaborazione istantaneamente. Il dato è riportato infatti nel primo istante disponibile, ovvero sul fronte di discesa del segnale di start.

Nella totalità, la macchina è stata concegnata trattando i fronti di salita del clock, eccetto in un singolo frammento di circuito. Il segnale di controllo, che suddivide il done nelle sue metà, usa in parte il clock invertito; grazie a questa caratteristica è stato possibile operare sul ritardo che, essendo l'input letto sui fronti di salita, sarebbe apparso.

Inoltre, simulando con testbench ed esaminando i rispettivi waveform, il circuito realizzato ha dimostrato di riportare esiti positivi costantemente.

Se l'immediatezza della risposta non giustifica i costi richiesti, è possibile modificare l'architettura per completare la stessa mansione, sebbene introducendo del ritardo. Potenzialmente la nuova architettura, priva di alcune porzioni, sarà in grado di svolgere l'elaborazione con mezzo ciclo di clock di attesa. Innanzitutto perde di significato un uso della memoria così esaustivo; è implementabile quindi una configurazione di *mem_en* e *start* non saldati. Inoltre, il design potrebbe essere alleggerito in area coperta dai componenti logici: alcune entità interne possono essere rimosse.

In particolare l'unità "selettore" non è più essenziale (il registro agisce autonomamente senza MUX), così come il multiplexer necessario all'aggiramento del registro parallelo (da parte di *mem_data*, in "output writer"). Anche i segnali di controllo sono semplificabili, non essendo più di rigore la scissione delle due metà di *done*.

Essendo la progettistica prevalentemente di livello strutturale, il conseguimento di alcuni obiettivi richiede la gestione di numerosi dettagli, permettendo però di realizzare alcune qualità più settoriali. Operare per livelli di astrazione è di prassi per comprendere l'istanza più alta e generale.