Politecnico di Milano
Corso di Laurea in Ingegneria Informatica
Scuola di Ingegneria Industriale e dell'Informazione

POLITECNICO
MILANO 1863

Prova Finale

Progetto di Reti Logiche

Relazione

Autore: Simone Corbo
Matricola: 955854
Codice Persona: 10727140

Prof.: William Fornaciari

Milano, Maggio 2023

Contents

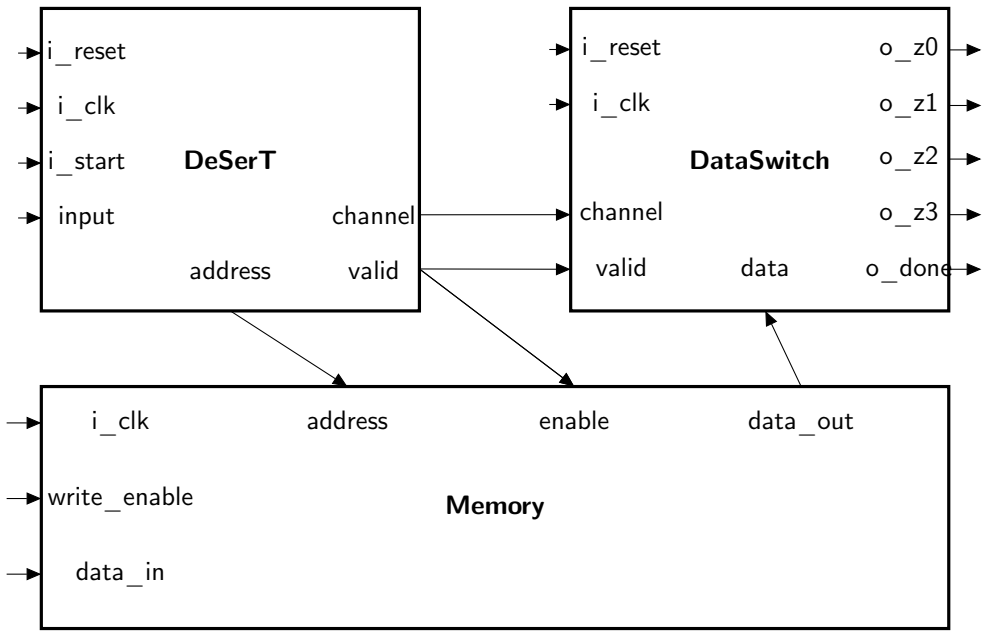
1 Introduzione

Il componente progettato salva in un registro selezionato, su 4 disponibili, il valore passato per indirizzo di memoria in input.
Il segnale di input (`i_w`) è da considerare valido esclusivamente quando `i_start` vale `1`, il passaggio dal valore 0 al valore 1 segna l'inizio dell'elaborazione.
Un input valido è composto da due sezioni:

- **Canale**: i primi due bit contengono il canale selezionato
- **Indirizzo**: i bit rimanenti, da 0 a 16, rappresentano la parte meno significativa dell'indirizzo del dato presente in memoria.

Il termine della lettura dall'input, quando `i_start` torna al valore `0`, attiva il segnale di `o_mem_en`, per la lettura del dato in memoria, utilizzando l'indirizzo ricevuto e deserializzato.

Il dato ricevuto dalla memoria viene salvato nel canale corrispondente tramite un multiplexer, pilotato dal segnale `channel`.
Infine i valori dei 4 canali vengono mostrati simultaneamente per un ciclo di clock, con `done` posto ad `1`, mentre in tutte le altre fasi le uscite sono fissate al valore `0000 0000`.
In assenza di un segnale di reset, i canali mantengono internamente il valore dell'elaborazione precedente .



2 Architettura

Il componente è realizzato tramite 2 sotto componenti, denominati DeSerializeTransform (in seguito DeSerT) e DataSwitch.
Entrambi i sottocomponenti presentano in ingresso i segnali di clock e di reset.

2.1 DeSerialize Transform

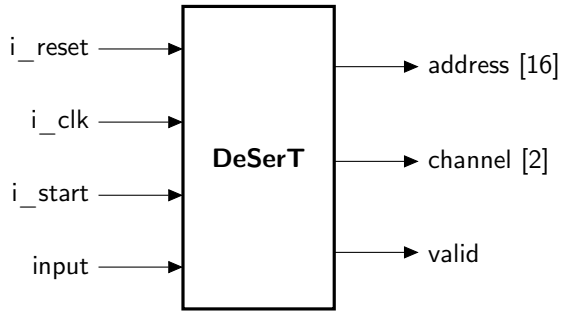


Figure 1: DeSerialize Transform

Il primo componente, DeSert, svolge la funzione di deserializzare e trasformare l'input `i_w` in due output paralleli da 2 e 16 bit, rispettivamente `channel` e `address`, rappresentanti il canale selezionato per il salvataggio del dato proveniente dalla memoria e l'indirizzo del dato.
La validità degli output è segnalata con il segnale `valid`, che assume il valore 1 per due cicli di clock, al termine dell'elaborazione.

2.1.1 FSM DeSerT

Il funzionamento viene implementato con una macchina a stati finiti, composta dai seguenti stati:

- **RST**: Stato di reset, interrompe la lettura dell'input
- **Wait Input**: Attesa di un input valido
- **Get CH[1]**: Lettura del primo bit del canale.
- **Get CH[0]**: Lettura del secondo bit del canale.
- **Get Address**: Lettura dell'indirizzo seriale
- **Send Input**: Trasmissione dell'input alla memoria.

DeSerT viene inizializzato nello stato `RST`, nel quale rimane finché `i_rst=1`.
Al termine della fase di reset, `i_start` può valere `1` o `0`, di conseguenza la macchina può passare direttamente alla prima fase dell'elaborazione quando `i_start=1`, entrando nello stato di `GET_CH1`, oppure rimanere in attesa di un input valido attendendo nello stato di `WAIT_INPUT` finché `i_start=0` e `i_reset=0`.
L'elaborazione inizia nonappena `i_start=1`, portando la macchina allo stato `GET_CH1`, permettendo il salvataggio del valore di `i_w` in `channel[1]`.
Considerando un input ben formato, lo stato successivo `GET_CH0` viene raggiunto dopo un ciclo di clock ed è responsabile del salvataggio del valore in input in `channel[0]`.
Dopo la lettura del canale, la trasmissione dell'input può interrompersi oppure continuare fino a 16 cicli di clock, quindi la macchina può passare dallo stato `GET_CH0` allo stato `SEND_INPUT` se il segnale di `i_start` vale

0, altrimenti passa allo stato GET_ADDRESS fintantoche i_start vale 1.
Per produrre una lettura corretta è stato implementato uno shift register, che ad ogni ciclo di clock trasla verso sinistra (SLL) i valori presenti all'interno del registro inserendo il valore letto in input nel bit meno significativo. Inizializzando il registro al valore 0, si permette di gestire il caso in cui la parte dell'indirizzo abbia lunghezza 0, assumendo in tal caso l'indirizzo 0000 0000 0000 0000
Successivamente al parsing dell'input, la macchina si porta nello stato di SEND_INPUT, nel quale attiva la memoria per il recupero del dato richiesto.
Dopo aver terminato l'invio alla memoria, segnalato dall'output valid=1, il componente si pone nello stato WAIT_INPUT oppure in GET_CH1 se è immediatamente presente un nuovo input.
Dalla specifica si evince che tutti gli input testati sono ben formati, quindi per semplicità non sono stati considerati i casi in cui i_start sia uguale a 0 prima del termine della lettura.

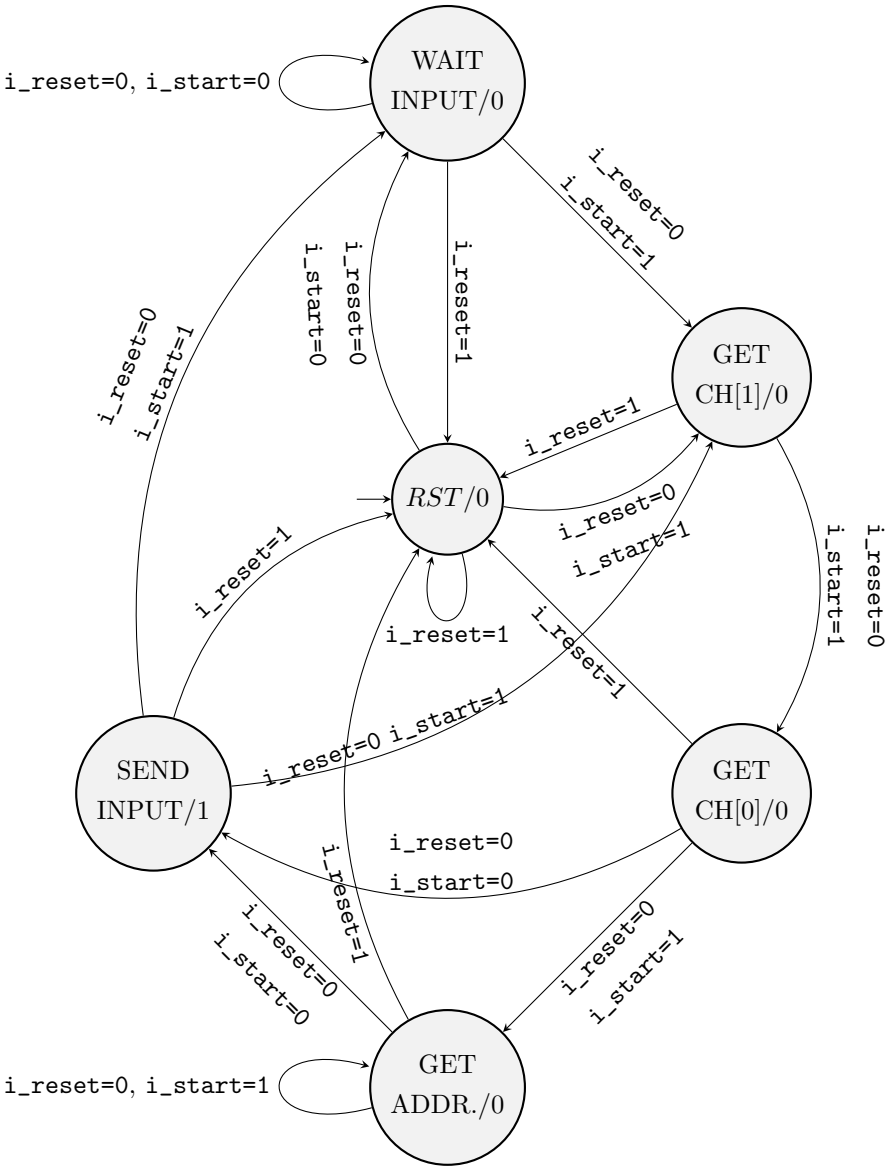
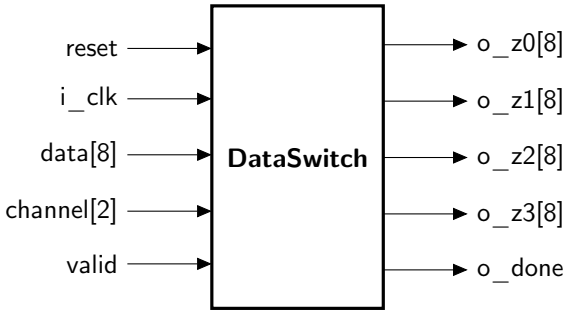


Figure 2: FSM DeSerT, Output considerato: valid

2.2 DataSwitch



Il secondo componente (DataSwitch) si occupa della comunicazione con la memoria e la gestione dei registri di memoria interni.
Più specificatamente, riceve in ingresso il canale selezionato, i dati provienienti dalla memoria, il segnale di validità degli input e il segnale di reset, mentre in uscita presenta i 4 canali e il segnale di fine elaborazione.

2.2.1 FSM DataSwitch

Come nel componente precedente, il suo funzionamento è rappresentato da una macchina a stati finiti, composta dai seguenti stati:

- **RST**: Stato di reset, inizializzazione dei registri interni, interruzione richiesta alla memoria
- **Wait Input**: Attesa di un input valido
- **Save**: Salvataggio dei dati ricevuti dalla memoria
- **Show**: Trasmissione dell'input alla memoria.

Analogamente al primo componente, esso parte nello stato di `RST`, inizializzando tutti i registri interni (uno per ciascun canale di output, denominati `internal_z0`, `internal_z1`, `internal_z2` e `internal_z3`) al valore `0000 0000`.
Al successivo ciclo di clock, in assenza del segnale di reset, il componente si pone in attesa (`WAIT_INPUT`) di un input valido proveniente da `DeSerT` che viene segnalato da `valid_input=1`, altrimenti rimane nello stato corrente finché `valid_input=0`.
Lo stato successivo `WAIT_DATA` viene utilizzato per attendere i dati dalla memoria, che richiedono un tempo di attesa pari a 2 ns, a seguito del ciclo di clock di richiesta dati.
Nello stato di `SAVE`, il dato ricevuto in input viene salvato nel relativo registro interno, selezionato tramite un multiplexer controllato dall'input `channel`.
La fase di `SHOW` permette il collegamento tra le uscite e i rispettivi registri interni, mostrando i valori salvati per un ciclo di clock, a differenza degli altri stati, in cui le uscite valgono sempre `0000 0000`.
Successivamente allo stato `SHOW` il componente può tornare in attesa di un input valido (stato `WAIT_INPUT` con `valid_input=0`) oppure può cominciare una nuova elaborazione, gestendo input ravvicinati, entrando nello stato `WAIT_DATA` con `valid_input=1`
In ogni stato, al presentarsi della condizione `i_reset=1`, si ha la transizione allo stato di `RST`.

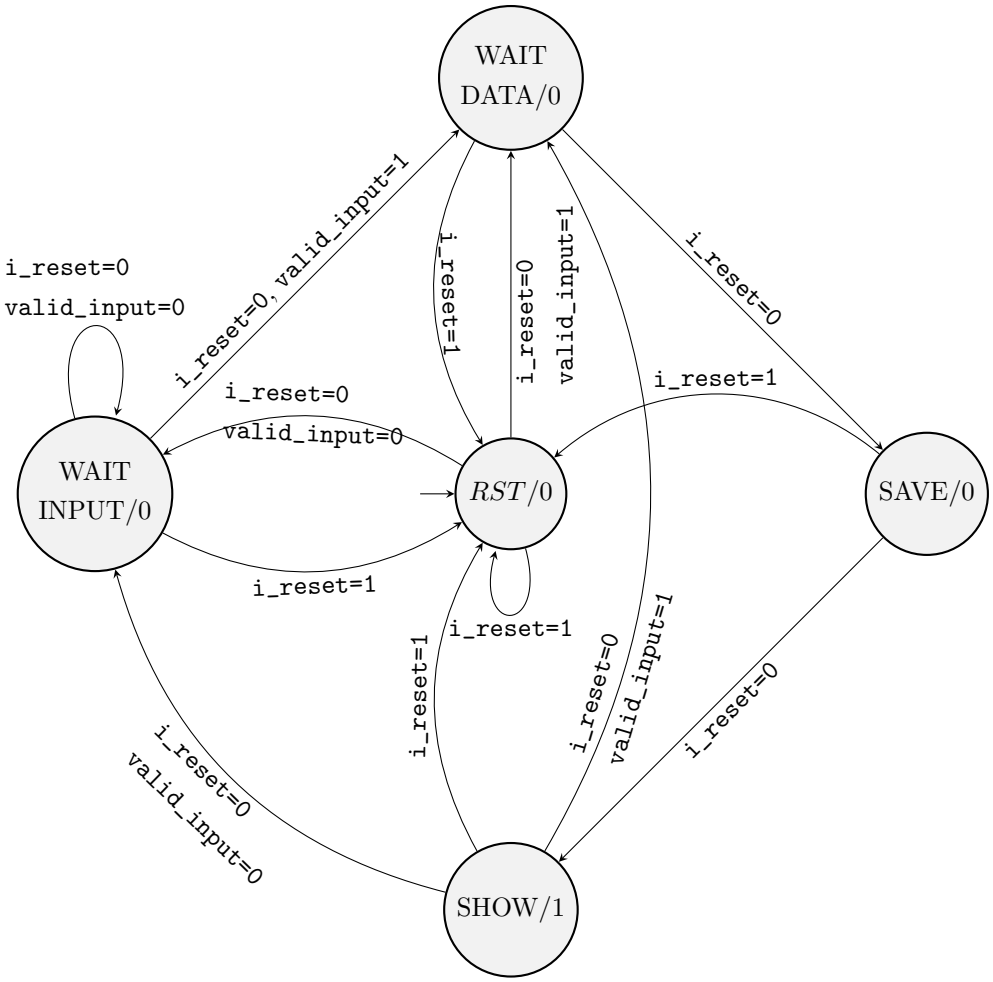


Figure 3: FSM DataSwitch, Output considerato `done`

3 Risultati Sperimentali

3.1 Report di Sintesi

Il componente è stato sintetizzato e testato, passando 17 batterie di test.
Per l'implementazione in post sintesi sono stati utilizzati 58 Registri Flip Flop e 0 Latch.
Lo schematico prodotto da VIVADO è in allegato alla presente relazione.

3.2 Simulazioni

Le simulazioni svolte hanno l'obiettivo di testare condizioni particolari e casi limite che possono verificarsi. Di seguito sono riportati i risultati relativi ai test più significativi del componente in post sintesi, realizzati tramite VIVADO.

3.2.1 Lunghezza variabile dell'indirizzo e sovrascrittura

Questa simulazione verifica che i due casi limite della lunghezza dell'indirizzo in input, cioé quando è lungo 16 e 0 bit, venga riconosciuta correttamente.
Nella simulazione sono rispettivamente il primo ed il terzo input.

Tutti i registri vengono scritti e visualizzati correttamente e si può notare come il segnale di `reset`, inizializzi a `0000` `0000` i registri interni.

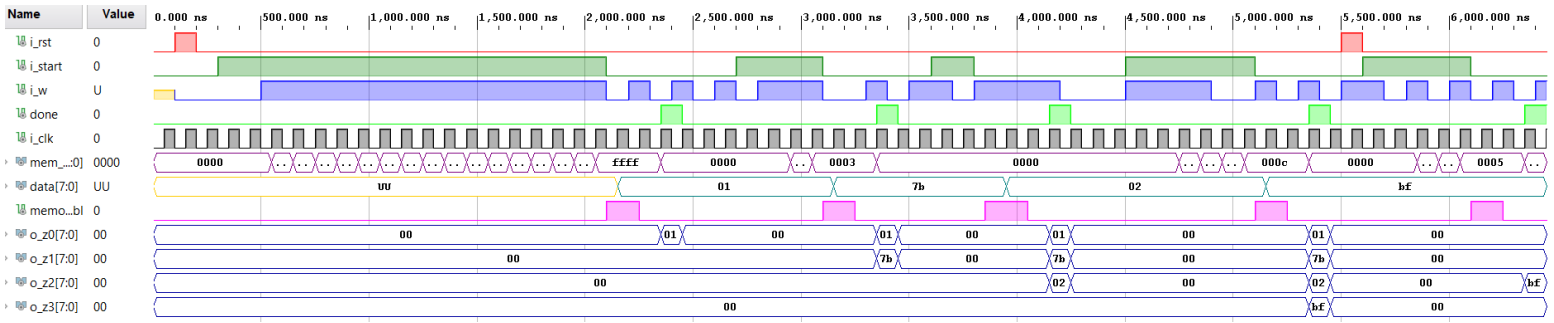


Figure 4: Test Indirizzo

3.2.2 Interruzione da `reset`

Questo test è stato creato per verificare la risposta al segnale di `reset` nei seguenti casi:

- **Reset prima del termine dell'input**: se il segnale di reset avviene prima del termine dell'input dei dati allora non viene inviata la richiesta alla memoria e i registri vengono inizializzati.
- **Reset durante input**: se il segnale di reset viene ricevuto e termina prima della fine della trasmissione dell'input, la lettura viene fatta ripartire nonappena il segnale di reset torna a 0 e prosegue ordinariamente.
- **Reset dopo l'input**: se il segnale di reset avviene dopo la fine della trasmissione, quando la richiesta alla memoria è già stata avviata, essa viene interrotta, i dati eventualmente ricevuti vengono scartati correttamente e i registri inizializzati.
- **Reset dopo ricezione dati**: se il segnale di reset avviene dopo la ricezione dei dati dalla memoria, questi vengono scartati e i registri inizializzati.

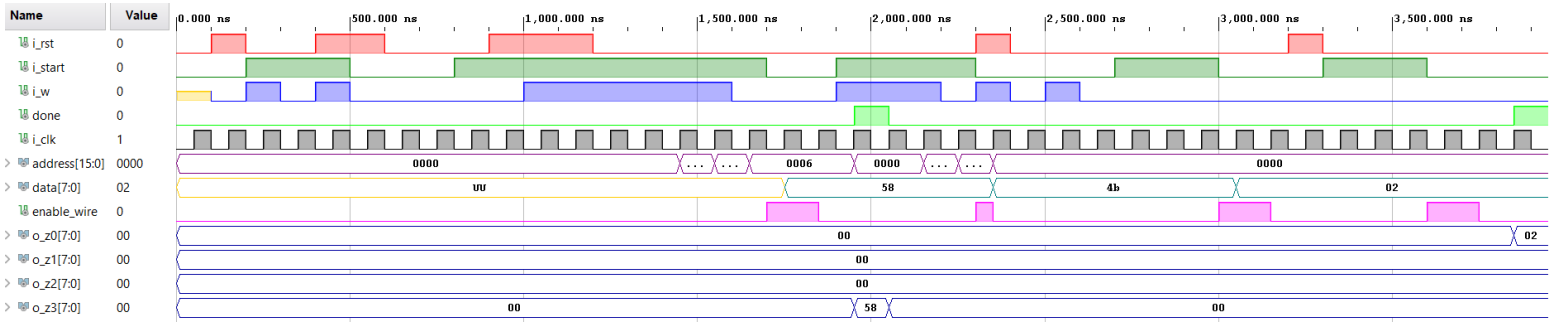


Figure 5: Test Reset

3.2.3 Trasmissioni ravvicinate

Nonostante nella specifica fosse esplicitato che il segnale di start sia sempre successivo alla discesa del segnale di done, nell'implementazione svolta il componente è in grado di gestire correttamente sequenze di input separati da almeno 2 cicli di clock, a prescindere dallo stato di done.

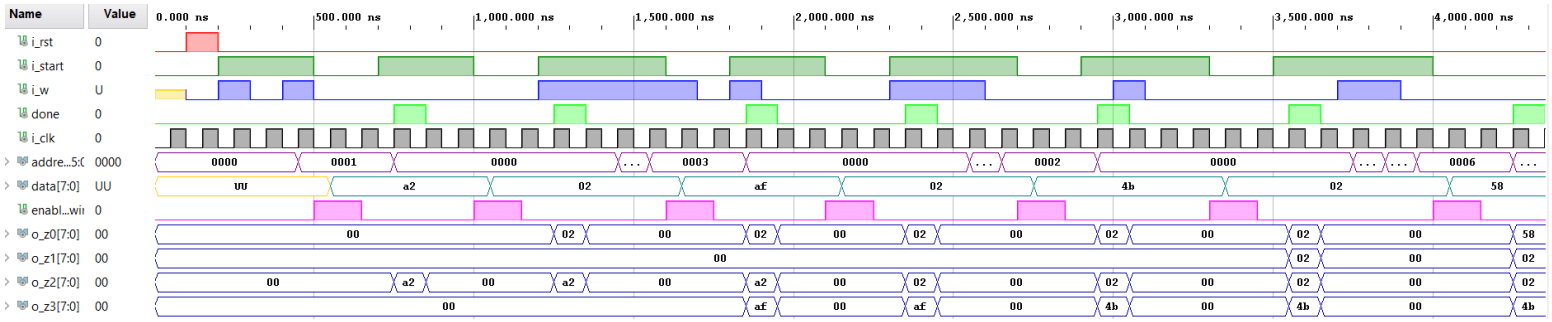


Figure 6: Test con input ravvicinati

4 Conclusioni

Il componente rispetta completamente la specifica, sia in pre che in post sintesi.
È in grado di elaborare correttamente un input in 3 cicli di clock dalla fine della trasmissione, a fronte del limite massimo dei 20 contenuti nella specifica.
In caso di interruzione da reset, tutte le richieste alla memoria vengono interrotte e gestite correttamente.

5 Allegati

5.1 Schematico prodotto