Politecnico di Milano

Progetto di Reti Logiche 2022-2023

Fabrizio Sordetti
[redacted]

Maggio 2023

Indice

1. Introduzione 1
2. Architettura 2
2.1 Serial to Parallel 2
2.2 Memory Interface 2
2.3 Out Switcher 3
2.3.1 Register 8bit 3
3. Risultati sperimentali 3
3.1 Sintesi (Report di Sintesi) 3
3.2 Simulazioni 5
3.2.1 Test Bench generici 5
3.2.2 Reset Asincroni e Multipli 5
3.2.3 Borderline W 6
3.2.4 Heavy Test 6
3.2.5 W e START Asincroni 6
4 Conclusioni 6

1. Introduzione

L'obiettivo dell'architettura qui trattata è l'estrazione di dati dalla memoria e il loro indirizzamento verso uno dei 4 canali di uscita (Z0, Z1, Z2, Z3). Un approccio di questo tipo si basa su un'idea di switcher per la gestione dei dati della memoria in maniera intelligente.
Come da specifica funzionale del progetto, il sistema riceve le informazioni relative al canale da utilizzare e all'indirizzo di memoria tramite un ingresso seriale W da un bit. Le operazioni di estrazione e indirizzamento dei dati, dunque, iniziano dopo un'analisi dell'ingresso seriale; il quale è coadiuvato da un ulteriore segnale di ingresso primario START da 1 bit che definisce l'intervallo di validità di W. I primi due bit della sequenza valida di W definiscono l'uscita e i rimanenti bit formano l'indirizzo di memoria eventualmente esteso fino al sedicesimo bit con 0 sui bit più significativi.
Ad esempio, in Figura 1.1 si può vedere come il segnale START definisca la sequenza utile di W (evidenziata in rosso) e come questa venga suddivisa nelle due parti sopra descritte. Quindi, 01 identifica l'uscita Z1 e 0110110 identifica l'indirizzo di memoria 0x0036. Il modulo, visualizzabile come una blackbox in questo esempio, si interfaccia con la memoria comunicandogli l'indirizzo e ottenendo il dato desiderato che proporrà all'uscita corretta.
Quando il risultato è pronto all'uscita: il segnale DONE va ad 1 e le altre uscite mostrano i loro valori precedenti. L'esposizione delle uscite dura un ciclo di CLK alla fine del quale le uscite e DONE vanno a 0. Il segnale di RST (re)inizializza il modulo (portando a zero tutte le uscite ed eliminando dalla memoria del modulo i valori precedenti salvati).
A partire da queste premesse, si è voluto aggiungere come specifica non funzionale l'ottimizzazione della velocità di interfacciamento, per quanto possibile, con la RAM, dispendiosa da un punto di vista energetico e temporale.

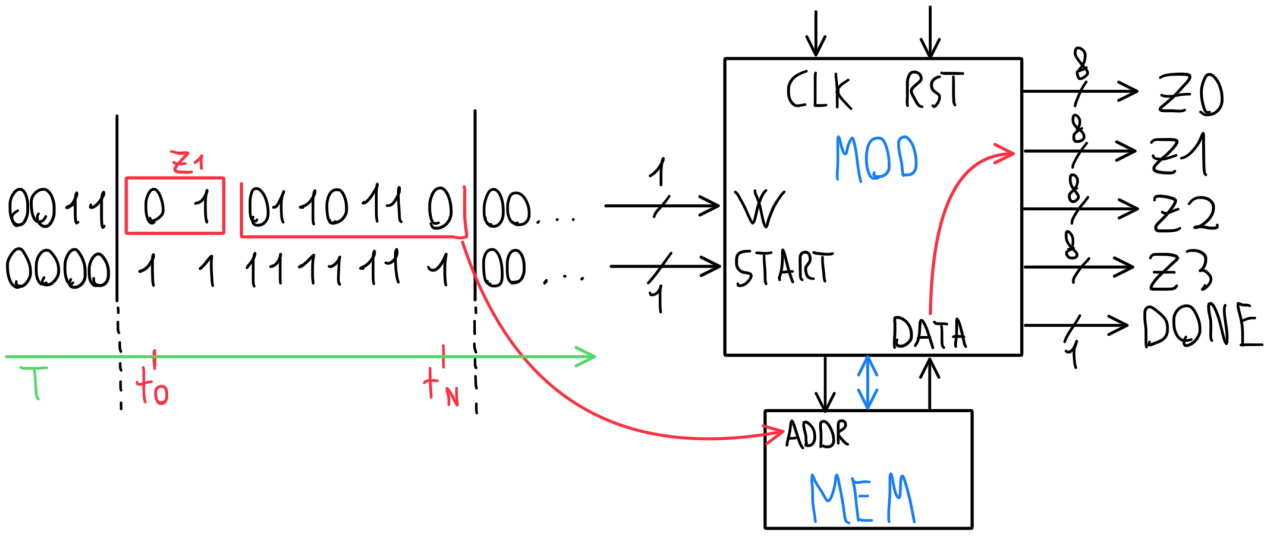


Figura 1.1: Esempio di esecuzione

2. Architettura

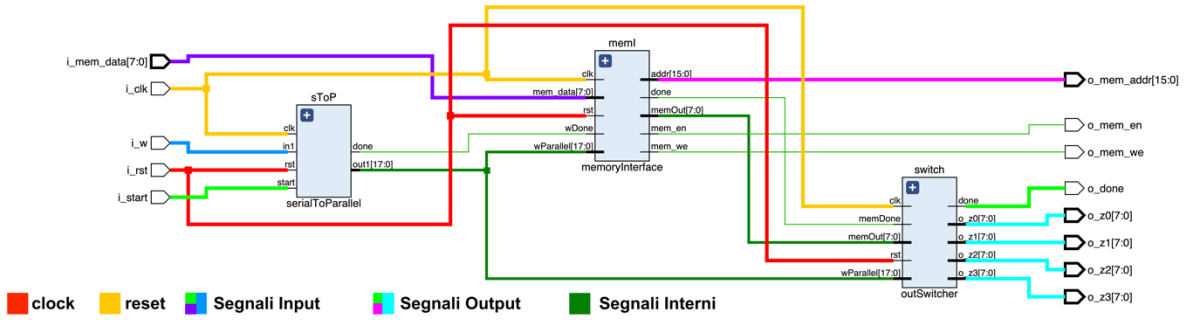


Figura 2.1: Schema funzionale dell'architettura, ingrandimento in appendice

A partire dalle specifiche funzionali e non, si è voluto individuare blocchi logici con compiti ben definiti, in modo da facilitare la fase di progettazione e favorire una migliore riusabilità. In particolare, le soluzioni più interessanti implementate nel design per rispondere ai requisiti non funzionali consistono nell'adozione di quattro registri a 8 bit (situati all'interno del modulo outSwitcher in Figura 2.1) che permettono di fare caching dei dati estratti dalla memoria.

2.1 Serial to Parallel

Questo componente sequenziale è realizzato con architettura behavioral ed è incaricato di elaborare i segnali d'ingresso (W e Start) e fornire in uscita il segnale W parallelizzato nell'intervallo di interesse. Tale segnale d'uscita è coadiuvato da un ulteriore segnale "done" che serve a segnalarne la validità. Questa importante funzione si esplicita in un architettura con all'interno tre ulteriori segnali per la memorizzazione di segnali di interesse (quali lo Start e il W paralellizato) e un macro processo CONVERSION_SERIAL_TO_PARALLEL al cui interno sono presenti 3 variabili di servizio per poter svolgere l'intero compito di paralellizzazione.

2.2 Memory Interface

È un componente sequenziale realizzato tramite una semplice architettura behavioral e fa da intermediario tra la RAM e i componenti che necessitano di accedervi (serialToParallel e outSwitcher), garantendo la corretta comunicazione dei componenti interni con la RAM. All'interno è presente un solo segnale di memorizzazione e tutto viene gestito da un processo che elabora e propone in ingresso alla memoria il segnale W parallelo (prendendo solo i bit per formare l'indirizzo) e in fine attende la risposta da quest'ultima con i dati d'interesse.

2.3 Out Switcher

Questo componente è combinato a una serie di 4 registri da 8 bit (vedere sottoparagrafo successivo); è realizzato da una coppia di processi e si occupa di caricare dati dalla memoria esterna e renderli disponibili alle uscite del modulo. In particolare, in primo luogo l'outSwitcher tramite il processo FIND_EXIT estrae i bit relativi all'uscita dal segnale W parallelo trovando quella designata affinché i dati vengano salvati nel suo registro interno. Successivamente, il processo PROPOSE_EXIT si occupa di gestire le uscite impostandole con il valore dei loro relativi registri quando il segnale done è alto (per un solo cilco di clock) e a zero in tutti gli altri casi.

2.3.1 Register 8bit

È un registro parallelo-parallelo a 8 bit con reset asincrono come quelli visti a lezione. Svolge la funzione di memorizzazione dei dati dalla memoria e ce n'è uno per ognuna delle uscite. Inoltre, tale componente è utile per adempiere ai requisiti non funzionali preposti in quanto permette di salvare i dati senza dover ricorrere ogni volta alla memoria, il quale utilizzo richiede tempo e risorse.

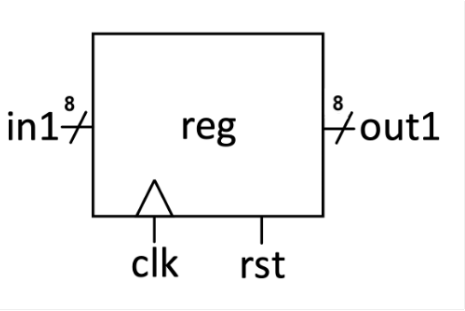


Figura 2.2: Registro parallelo-parallelo a 8 bit

3. Risultati sperimentali

3.1 Sintesi (Report di Sintesi)

La sintesi secondo strategia standard effettuata dal software Vivado 2022.2 su board xc7a200tfbg484-1 vede l'impiego di 60 LUTs e di 98 FF. La maggior parte delle risorse in uso viene impiegata per realizzare la codifica in parallelo del segnale W e il salvataggio dei dati (ad esempio, i registri interni che memorizzano i dati per ognuna delle uscite).
Si nota inoltre che tutti i registri utilizzati risultano essere dei Flip Flop. Non vi è quindi la presenza di Latches che avrebbero potuto creare problemi nei test post-sintesi.

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	60	0	0	134600	0.04
LUT as Logic	60	0	0	134600	0.04
LUT as Memory	0	0	0	46200	0.00
Slice Registers	98	0	0	269200	0.04
Register as Flip Flop	98	0	0	269200	0.04
Register as Latch	0	0	0	269200	0.00
F7 Muxes	0	0	0	67300	0.00
F8 Muxes	0	0	0	33650	0.00

Figura 3.1 Utilization Report

Nel Timing Report si trovano le informazioni riguardanti le tempistiche del clock, in particolare si è individuato lo Slack Time (WNS) di 96.701 ns; esso ci dice quanto tempo rimane al completamento del ciclo di clock nel peggiore dei paths, possiamo dedurre quindi che rispetta la condizione del clock minimo di 100 ns. Possiamo inoltre calcolare la massima frequenza alla quale il componente può funzionare:

$$T_{min} = T_{clk} - WNS = 100ns - 96.701ns = 3.299ns$$

$$f_{MAX} = \frac{1}{3.299ns} = 303{,}1\ MH$$

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	96,701 ns	Worst Hold Slack (WHS):	0,221 ns	Worst Pulse Width Slack (WPWS):	49,500 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns	Total Pulse Width Negative Slack (TPWS):	0,000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	81	Total Number of Endpoints:	81	Total Number of Endpoints:	67

All user specified timing constraints are met.

Figura 3.2 Timing Report

Infine, si può notare il Data Path Delay di 2.464ns e il Clock Constraint (richiesto dalla specifica e precedentemente citato) di 100ns all'interno del Timing Summary:

```
Max Delay Paths
--------------------------------------------------------------------------------------
Slack (MET) :             96.701ns  (required time - arrival time)
  Source:                 sToP/oldStartX2_reg/C
                            (rising edge-triggered cell FDCE clocked by clock  {rise@0.000ns fall@50.000ns period=100.000ns})
  Destination:            sToP/out1_reg[0]/R
                            (rising edge-triggered cell FDRE clocked by clock  {rise@0.000ns fall@50.000ns period=100.000ns})
  Path Group:             clock
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            100.000ns  (clock rise@100.000ns - clock rise@0.000ns)
  Data Path Delay:        2.464ns  (logic 0.806ns (32.706%)  route 1.658ns (67.294%))
  Logic Levels:           1  (LUT4=1)
```

Figura 3.3 Timing Summary

3.2 Simulazioni

Tutti i test benches riportati di seguito sono superati con successo dall'implementazione proposta. I test sono stati eseguiti in behavioural, post-sintesi e post-implementazione (sia funzionale sia timing, in entrambi i casi di simulazione). Sono stati applicati dei constraint al clock di 100ns come richiesto dalla specifica.

3.2.1 Test Bench generici

Il primo test bench generico assegnatoci dal professore viene passato in tutte le simulazioni senza problemi di alcun tipo. Esso esegue due reset su fronte di discesa e procede alla verifica della correttezza degli output dati in ingresso i segnali W e START come si vede in Figura 3.4. Sono stati effettuati alti sette test bench generici che vanno ad appesantire il carico (sulla lunghezza dei segnali e sul numero di iterazioni) rispetto al primo ma che si concludono anch'essi senza problemi.

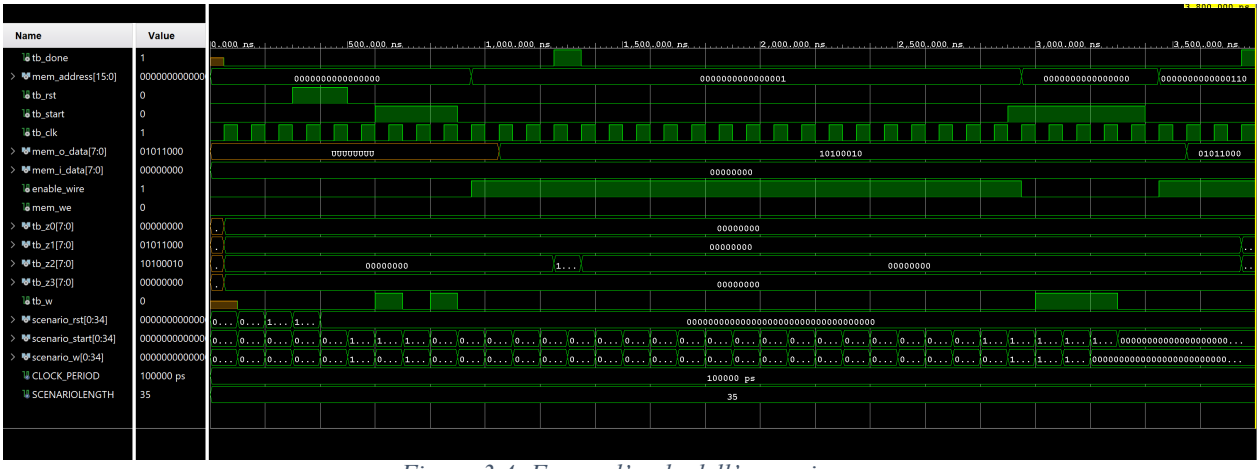


Figura 3.4: Forme d'onda dell'esecuzione

3.2.2 Reset Asincroni e Multipli

I test bench consistenti di reset Singoli e reset Multipli sono passati rispettando le richieste della specifica per questi eventi. In particolare, sono stati testati reset anche durante le varie fasi di elaborazione del segnale (conversione in parallelo di W, estrazione di dati dalla memoria e proposizione delle uscite), nessuno di questi ha generato problemi e il modulo è riuscito a re-inizializzarsi in tempi più che soddisfacenti.

3.2.3 Borderline W

Due test bench per verificare che in casi di W ai limiti (tutti 1 o tutti 0) il modulo si comporti in maniera corretta. Lo scopo di questi test è verificare anche che venga preso il giusto intervallo di bit segnalato da START. Entrambi i test sono passati con successo.

3.2.4 Heavy Test

Test bench con una serie varia di iterazioni per verificare il corretto funzionamento anche in caso di molte sovrascritture delle uscite. In particolare, si è voluto testare anche la velocità e l'efficienza del modulo messo sotto stress per molto tempo. Il risultato ha soddisfatto le aspettative, reggendo egregiamente il carico.

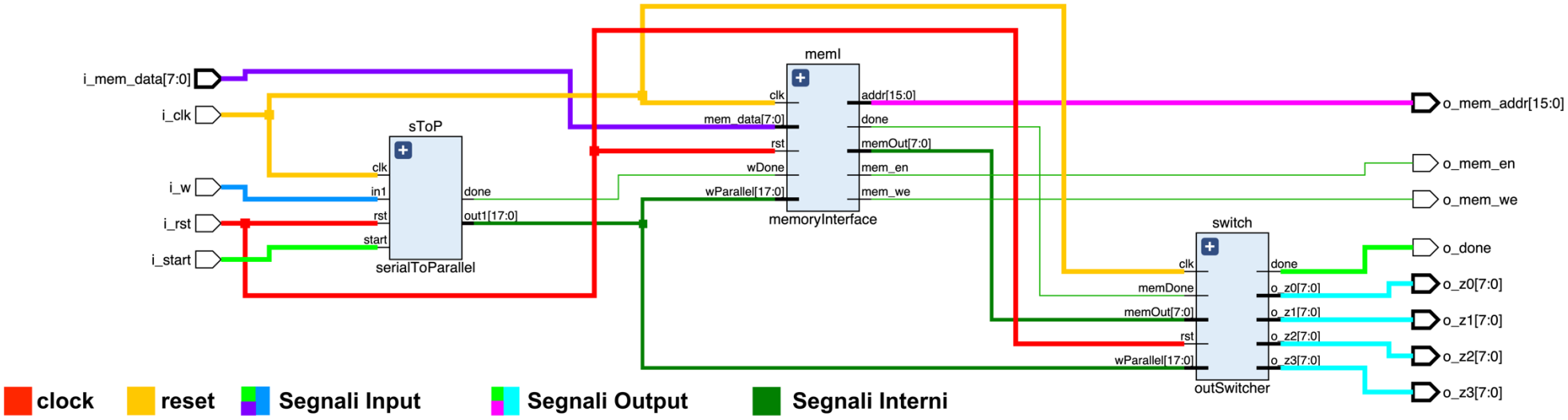
3.2.5 W e START Asincroni

Singolo test che verifica il comportamento del modulo in presenza di segnali di ingresso asincroni. Il modulo ha evidenziato una capacità di gestire tale situazione al meglio, nel limite della fisica dei collegamenti. In particolare, si nota come vengano presi i segnali su fronte di salite del clock (come richiesto da specifica) con molta accuratezza anche sotto il nanosecondo.

4 Conclusioni

Come ben dimostrato dai risultati sperimentali e dal report di sintesi, l'architettura qui descritta vanta prestazioni considerevoli, grazie alla già trattata adozione di registri interni e alla parallelizzazione dell'ingresso. Si è inoltre sperimentalmente verificato che, quando sintetizzata sulla board di riferimento, l'architettura proposta è in grado di operare a frequenze di clock elevate, permettendo quindi ottimi tempi di esecuzione.
Si ritiene, quindi, di aver descritto un componente hardware conforme alle specifiche in grado di rispondere correttamente ad ogni tipo di richiesta fornitagli.

Le problematiche (quali, ad esempio, l'iniziale fallimento dei test bench) sono state affrontate con un approccio metodico per risolverle al meglio e sviluppare così un modulo solido in gradi di rispettare gli standard preposti.
Infine, si può affermare che l'esperienza di questo progetto è stata indubbiamente positiva in quanto formativa ed interessante.


i_mem_data[7:0]
i_clk
i_w
i_rst
i_start
sToP
clk
in1
rst
start
done
out1[17:0]
serialToParallel
memI
clk
mem_data[7:0]
rst
wDone
wParallel[17:0]
addr[15:0]
done
memOut[7:0]
mem_en
mem_we
memoryInterface
switch
clk
memDone
memOut[7:0]
rst
wParallel[17:0]
done
o_z0[7:0]
o_z1[7:0]
o_z2[7:0]
o_z3[7:0]
outSwitcher
o_mem_addr[15:0]
o_mem_en
o_mem_we
o_done
o_z0[7:0]
o_z1[7:0]
o_z2[7:0]
o_z3[7:0]
clock
reset
Segnali Input
Segnali Output
Segnali Interni