POLITECNICO
MILANO 1863

Progetto finale di Reti Logiche

Nicole Filippi

a.a. 2022 - 2023

Indice

Capitolo 1

Introduzione

Il progetto consiste nell'implementazione di un modulo hardware che si interfacci con una memoria esterna per estrarne dei dati ed è stato descritto in VHDL.

In particolare, il modulo riceverà, mediante un ingresso seriale da un bit, le indicazioni riguardanti la locazione di memoria interessata e il canale d'uscita sul quale inoltrarne il contenuto. Il dispositivo, inoltre, consente di tenere in memoria l'ultimo dato trasmesso su ogni canale d'uscita fino ad un'eventuale sovrascrittura.

1.1 Interfaccia

Il modulo ha due ingressi primari da 1 bit (W e START) e cinque uscite primarie: quattro da 8 bit ciascuna (z0, z1, z2, z3) e una da 1 bit (DONE). Vi sono poi i segnali di CLOCK e RESET comuni a tutto il sistema.

In particolare, l'interfaccia è la seguente:

```
entity project_reti_logiche is
    port (
        i_clk : in std_logic;
        i_rst : in std_logic;
        i_start : in std_logic;
        i_w : in std_logic;
        o_z0 : out std_logic_vector(7 downto 0);
        o_z1 : out std_logic_vector(7 downto 0);
        o_z2 : out std_logic_vector(7 downto 0);
        o_z3 : out std_logic_vector(7 downto 0);
        o_done : out std_logic;
        o_mem_addr : out std_logic_vector(15 downto 0);
        i_mem_data : in std_logic_vector(7 downto 0);
        o_mem_we : out std_logic;
        o_mem_en : out std_logic
    );
end project_reti_logiche;
```

I segnali d'ingresso rappresentano:

- `i_clk` è il segnale di CLOCK;
- `i_rst` è il segnale di RESET che inizializza la macchina rendendola pronta per ricevere il segnale di START;
- `i_start` è il segnale di START;
- `i_w` è l'ingresso primario seriale W, utilizzato per ricevere i dati di ingresso;
- `i_mem_data` è il segnale DATA (vettore da 8 bit) che arriva dalla memoria esterna in seguito ad una lettura.

I segnali d'uscita rappresentano:

- `o_z0, o_z1, o_z2, o_z3` sono i segnali (vettori da 8 bit) dei quattro canali di uscita Z0, Z1, Z2, Z3;
- `o_done` è il segnale DONE che comunica la fine dell'elaborazione;
- `o_mem_addr` è il segnale ADDR (vettore da 16 bit) che manda alla memoria esterna l'indirizzo a cui si vuole accedere;
- `o_mem_en` è il segnale di ENABLE della memoria esterna, va posto uguale a 1 per poter comunicare sia in lettura che in scrittura;
- `o_mem_we` è il segnale di WRITE ENABLE della memoria esterna, da porre uguale a 1 per potervici scrivere. Da notare che non dovendo mai scrivere sulla memoria, qui sarà sempre posto a 0.

1.2 Funzionamento

All'istante iniziale, relativo al RESET, i canali d'uscita Z0, Z1, Z2, Z3 e DONE sono posti a 0. Quando START = 1, inizia la lettura dei dati in ingresso in W sul fronte di salita del clock. Questi dati sono organizzati nel seguente modo:

- 2 bit di intestazione che identificano il canale d'uscita sul quale andrà indirizzato il dato successivamente prelevato dalla memoria, il primo bit ricevuto è il più significativo.
  Più in dettaglio: 00 identifica Z0, 01 identifica Z1, 10 identifica Z2 e 11 identifica Z3.
- N bit di indirizzo della memoria, con $0 \leq N \leq 16$. Per $N < 16$ l'indirizzo verrà esteso con 0 sui bit più significativi. I bit ricevuti sono in ordine dal più significativo al meno significativo. Si ricorda che ad ogni indirizzo è memorizzato un dato da 8 bit.

Quando START, dopo un minimo di 2 cicli di clock e un massimo di 18 cicli di clock, torna uguale a 0, termina la lettura di W da parte del modulo. Dopo aver memorizzato la codifica del canale d'uscita richiesto e l'indirizzo di memoria, viene attivato il segnale di ENABLE e la memoria successivamente restituisce il dato interessato. Il segnale DONE viene alzato e mentre il dato viene inoltrato sul canale d'uscita richiesto tutti gli altri canali mostreranno l'ultimo dato trasmesso, se presente. Si noti che i valori devono essere visibili sui canali d'uscita solo quando DONE = 1. Dopo un ciclo di clock DONE viene posto nuovamente a 0 e il modulo torna in attesa di un nuovo segnale di START.

1.3 Esempio

Il seguente esempio serve per mostrare la relazione tra gli ingressi START e W e le uscite DONE, z0, z1, z2, z3. Si ricorda che i canali d'uscita mostrano i dati solamente quando DONE = 1. Si noti come il valore di W viene ignorato quando START = 0.

START	1	1	1	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	1	1	1	1	0	0	0	0	0
W	0	1	1	0	0	1	0	0	0	0	1	0	1	0	0	1	1	0	0	1	1	0	0	0	0	0	0
DONE	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
Z0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	44	00	00	00	00	00	00	00	00	44
Z1	00	00	00	00	00	00	00	2E	00	00	00	00	00	00	00	00	00	2E	00	00	00	00	00	00	00	00	14
Z2	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Z3	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

In uscita sono visualizzati i dati presenti nella locazione di memoria richiesta da W secondo la seguente mappatura:

ADDR	**DATA**
0001	2E
0002	14
0005	44

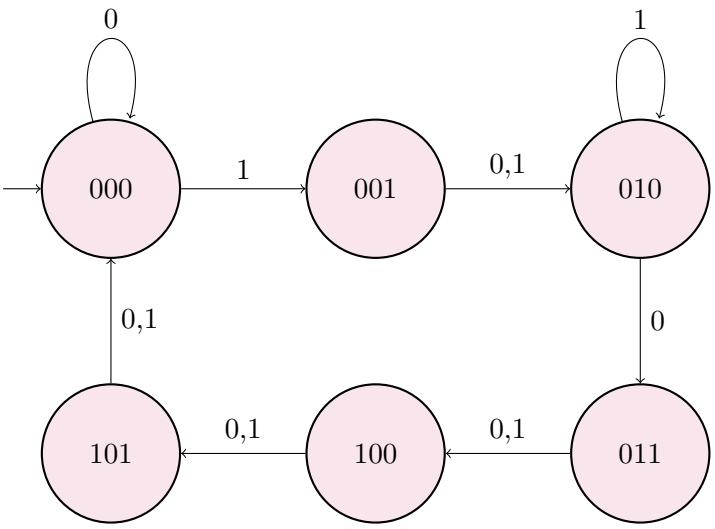
Si noti che i valori di START, W e DONE sono rappresentati con codifica binaria, mentre i valori di ADDR, DATA e dei canali d'uscita sono rappresentati in base esadecimale.

Capitolo 2

Architettura

2.1 FSM

Il modulo è rappresentabile come una macchina a stati finiti dove l'ingresso è rappresentato dal segnale START:



Gli stati della macchina sono sei, codificati in binario naturale su 3 bit:

CODIFICA	NOME
000	`Ready`
001	`Read_Channel`
010	`Read_Address`
011	`Set_Enable`
100	`Catch_Data`
101	`Show_Data`

Le azioni svolte in ogni stato sono le seguenti:

- `Ready` → stato iniziale in cui la macchina attende lo START e legge il primo valore di W, che corrisponde al bit più significativo del canale d'uscita e passa allo stato successivo;
- `Read_Channel` → stato in cui la macchina legge il secondo valore di W, che corrisponde al bit meno significativo del canale d'uscita e passa allo stato successivo;
- `Read_Address` → stato in cui la macchina, finché START = 1, legge da 0 a 16 valori da W, rappresentanti l'indirizzo della memoria a cui accedere e non appena START = 0 la macchina imposta il segnale ENABLE = 1 per permettere la lettura dalla memoria esterna e passa allo stato successivo;
- `Set_Enable` → stato in cui la macchina mantiene ENABLE = 1 e dopo un ciclo di clock passa allo stato successivo;
- `Catch_Data` → stato in cui, sul fronte di salita, la memoria esterna vede ENABLE = 1 e dopo 2 ns pone in ingresso al dispositivo il dato richiesto. A questo punto la macchina salva il dato sul canale d'uscita desiderato, riporta ENABLE = 0 per indicare la fine della lettura da memoria, pone DONE = 1 così da mostrare su tutte le uscite i dati salvati e passa allo stato successivo;
- `Show_Data` → stato finale in cui dopo un ciclo di clock la macchina riporta DONE = 0 e ritorna allo stato inziale `Ready` in attesa di un nuovo segnale di START.

In caso di RESET, qualunque sia lo stato in cui si trova la macchina, si ritorna allo stato iniziale `Ready` e il modulo viene reinizializzato .

2.2 Componenti

Il modulo è suddiviso in tre diversi componenti con funzioni specifiche per migliorare la scalabilità in caso di espansione, l'efficienza e semplificare eventuali modifiche. Questi componenti comunicano tra loro le informazioni necessarie allo svolgimento dei compiti e agiscono in parallelo.

2.2.1 State_Register

È il componente che tiene traccia dello stato in cui la macchina si trova, contando. Il suo compito principale è permettere la coordinazione interna del modulo segnalando la codifica dello stato attuale agli altri componenti, che di conseguenza sapranno quale operazione svolgere. Si occupa anche della modifica dello stato dei segnali di DONE e di ENABLE, ponendoli a 1 e a 0 quando necessario, secondo quanto indicato nella descrizione degli stati sopra.
Riceve in input i segnali di CLOCK, RESET, START e in output restituisce ENABLE, STATE e DONE. STATE è un registro da 3 bit interno al componente che indica lo stato in cui la macchina si trova secondo la codifica sopra citata, questo segnale è inviato agli altri due componenti.
In caso di RESET inizializza STATE = 000, ENABLE = 0, DONE = 0.

2.2.2 Input_Memory

È il componente che memorizza i dati in arrivo da W.
Riceve in input i segnali di CLOCK, START, W, STATE; in output restituisce CHANNEL e invia direttamente alla memoria esterna ADDR.
CHANNEL è un registro da 2 bit interno al componente che rappresenta la codifica del canale d'uscita, ovvero i primi due valori presi da W. Questo segnale è inviato al componente *channels_registers*.
ADDR è inizializzato a 0 (16 bit posti a 0) ed è utilizzato uno shift a sinistra per l'inserimento dei valori di W man mano che vengono ricevuti, in questo modo in caso di un numero di valori minori di 16 vi è già il riempimento di 0 sui bit più significativi.
START e STATE servono per la gestione della lettura.
Non necessita del segnale di RESET perché CHANNEL è semplicemente sovrascritto, mentre ADDR è inizializzato ogni volta che STATE = 000.

2.2.3 Channels_Registers

È il componente che si occupa di gestire i canali d'uscita.
Riceve in input i segnali di CLOCK, RESET, DONE, STATE, CHANNEL e l'uscita della memoria esterna DATA; in output restituisce i 4 canali d'uscita z0, z1, z2, z3.
Internamente ha quattro registri da 8 bit, inizializzati a 0 ad ogni RESET, che permettono di memorizzare dei valori. Questi quattro registri, messi ognuno in AND con DONE, creano automaticamente le rispettive uscite z0, z1, z2, z3. In questo modo viene memorizzato l'ultimo dato per ogni canale d'uscita, che verrà però mostrato solo quando DONE = 1. Il componente si occupa di memorizzare DATA nel registro corrispondente indicato da CHANNEL, eventualmente sovrascrivendo valori precedenti.
La sequenza di operazioni è controllata da STATE.

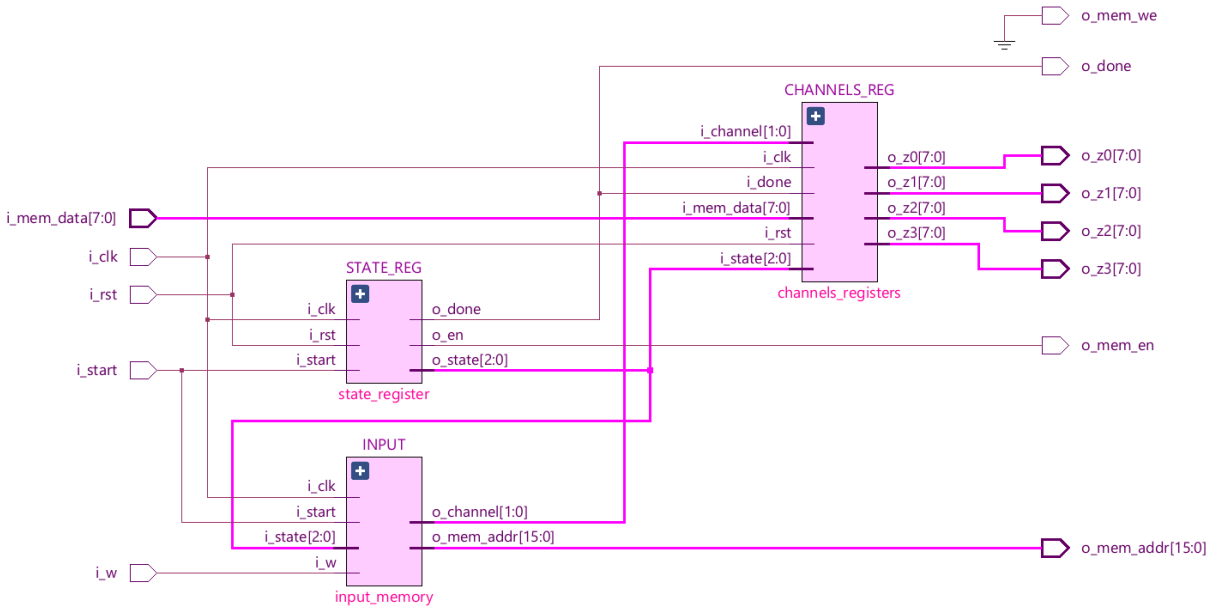


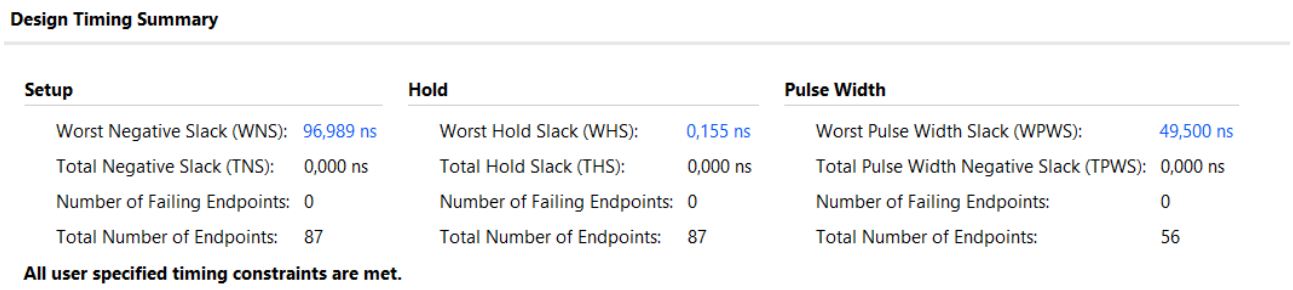
Figura 2.1: Schema dei componenti sintetizzato da Vivado

Capitolo 3

Risultati Sperimentali

3.1 Sintesi

Il modulo descritto è correttamente sintetizzabile. È stato inoltre verificato che il modulo funzioni con un periodo di clock di 100 ns aggiungendo un *timing constraint*, come si può vedere dalla *Figura 3.1*. Inoltre in post-sintesi supera gli stessi testbench a cui è stato sottoposto in pre-sintesi.



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	96,989 ns	Worst Hold Slack (WHS):	0,155 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:	87	Total Number of Endpoints:	87	Total Number of Endpoints:	56

All user specified timing constraints are met.

Figura 3.1: Estratto del timing report di sintesi di Vivado

Il report di sintesi di Vivado è uno strumento utile per estrarre statistiche e informazioni riguardanti la sintesi del modulo e permetterne un approfondimento. Tra queste informazioni sono rilevanti: la struttura della FSM (*Figura 3.2*), la descrizione ad alto livello (RTL - register-transfer level) dei componenti (*Figura 3.3*), la memoria utilizzata e il tempo impiegato per ogni fase, l'applicazione di constraint ed eventuali warnings o errori che si sono presentati:

```
INFO: [Synth 8-802] inferred FSM for state register 'r_state_reg' in module 'state_register'
---------------------------------------------------------------------------------------------------
                   State |                     New Encoding |               Previous Encoding
---------------------------------------------------------------------------------------------------
                 iSTATE0 |                              000 |                              000
                 iSTATE1 |                              001 |                              001
                 iSTATE2 |                              010 |                              010
                 iSTATE3 |                              011 |                              011
                 iSTATE4 |                              100 |                              100
                  iSTATE |                              101 |                              101
---------------------------------------------------------------------------------------------------
INFO: [Synth 8-3354] encoded FSM with state register 'r_state_reg' using encoding 'sequential' in module 'state_register'
```

Figura 3.2: Struttura della FSM dal report di sintesi di Vivado

```
Start RTL Component Statistics
---------------------------------------------------------------------------
Detailed RTL Component Info :
+---Registers :
                      16 Bit    Registers := 1
                       8 Bit    Registers := 4
                       2 Bit    Registers := 1
                       1 Bit    Registers := 2
+---Muxes :
           2 Input   16 Bit       Muxes := 1
           4 Input   16 Bit       Muxes := 1
           6 Input    3 Bit       Muxes := 2
           2 Input    2 Bit       Muxes := 1
           4 Input    2 Bit       Muxes := 1
           2 Input    1 Bit       Muxes := 1
           4 Input    1 Bit       Muxes := 1
           6 Input    1 Bit       Muxes := 4
           5 Input    1 Bit       Muxes := 4
---------------------------------------------------------------------------
Finished RTL Component Statistics
```

Figura 3.3: Descrizione ad alto livello dei componenti dal report di sintesi di Vivado

3.2 Simulazioni

Il modulo, completo di tutti i componenti, è stato testato su alcuni casi particolari per verificarne il corretto funzionamento:

- ADDR vuoto (0 bit)
- ADDR pieno (16 bit)
- Sovrascrittura di z0
- Sovrascrittura di z1
- Sovrascrittura di z2
- Sovrascrittura di z3
- RESET asincrono durante START = 1
- RESET asincrono durante le letture da W
- RESET asincrono durante l'accesso alla memoria esterna
- RESET asincrono dopo la restituzione del dato in uscita dalla memoria esterna
- RESET asincrono durante DONE = 1
- RESET asincrono dopo aver scritto tutte le uscite, per assicurarsi che vengano reinizializzate correttamente

Successivamente, è stato testato il modulo utilizzando valori di input (W), dati in memoria e RESET casuali.

3.2.1 Esempio di simulazione

Di seguito è riportato un esempio di funzionamento del modulo:

- richiesta del dato all'indirizzo di memoria esterna 0_{10} sul canale d'uscita z0
- richiesta del dato all'indirizzo di memoria esterna 8041_{10} sul canale d'uscita z1
- sovrascrittura del canale d'uscita z0 con il dato all'indirizzo di memoria esterna 27_{10}

All'interno degli indirizzi di memoria interessati si trovano i seguenti dati:

- $0_{10} \rightarrow 255_{10}$
- $8041_{10} \rightarrow 153_{10}$
- $27_{10} \rightarrow 102_{10}$

Figura 3.4: Diagramma temporale di una simulazione di Vivado

TEMPO	AZIONE
100 ns - 200 ns	reset del modulo
400 ns - 600 ns	START = 1 per 2 cicli di clock, lettura di W con CHANNEL = 00_2 e ADDR = 0_{10}
750 ns	DATA = 255_{10} viene restituito dalla memoria esterna
850 ns - 950 ns	DONE = 1: z0 = 255_{10}
1100 ns - 2600 ns	START = 1 per 15 cicli di clock, lettura di W con CHANNEL = 01_2 e ADDR = 8041_{10}
2750 ns	DATA = 153_{10} viene restituito dalla memoria esterna
2850 ns - 2950 ns	DONE = 1: z0 = 255_{10} e z1 = 153_{10}
3100 ns - 3800 ns	START = 1 per 7 cicli di clock, lettura di W con CHANNEL = 00_2 e ADDR = 27_{10}
3950 ns	DATA = 102_{10} viene restituito dalla memoria esterna
4050 ns - 4150 ns	DONE = 1: z0 = 102_{10} e z1 = 153_{10}

Capitolo 4

Conclusioni

Il modulo è efficiente ed efficace. Passa con successo tutti i test eseguiti, anche quelli con i casi più particolari. Rispetta le richieste come il periodo di clock di almeno 100 ns e si interfaccia perfettamente con il modello di memoria esterna dato. Funziona assumendo, come da specifica, che START rimanga attivo correttamente (minimo 2, massimo 18 cicli di clock) e che rimanga a 0 finché DONE non è tornato a 0. Inoltre, rispetta largamente il vincolo del tempo massimo di 20 cicli di clock per produrre il risultato.
Importante notare anche che la sintesi è priva di latch, sinonimo di corretto funzionamento del componente e della scrittura esaustiva del codice che prende in considerazione di tutti i possibili casi nei vari costrutti.