Prova Finale (Progetto di Reti Logiche)

Prof. Gianluca Palermo – Anno 2019/2020

Xu Zewei (Codice Persona 10624970 – Matricola 892836)

Zhou Jian (Codice Persona 10601546 – Matricola 889767)

1. Introduzione

1.1 Descrizione problema

Si definisca un intervallo di indirizzi [0,127] e definiti 8 intervalli detti “Working Zone” all’interno di essa di larghezza fissa pari a 4: lo scopo del progetto è di implementare un componente hardware, descritto in VHDL, che, presi in ingresso gli indirizzi iniziali delle 8 working zone e l’indirizzo da valutare all’interno dell’intervallo considerato, calcoli l’indirizzo finale secondo un algoritmo ispirato alla codifica a bassa dissipazione di potenza denominato *Working Zone*. Dunque, la versione implementata considera gli indirizzi da codificare di 7 bit.

1.2 Algoritmo Working Zone

La codifica Working Zone è un metodo pensato per il Bus Indirizzi che si usa per trasformare il valore di un indirizzo quando viene trasmesso.

Le trasformazioni dell’indirizzo da trasmettere possibili sono 2:

- Se l’indirizzo (ADDR) in questione non appartiene a nessuna delle Working Zone, esso viene trasmesso come concatenazione di 0 & ADDR (dove & è il simbolo della concatenazione) codificato in binario
- Se l’indirizzo (ADDR) in questione appartiene ad una Working Zone, esso viene trasmesso come concatenazione di 1 & WZ_NUM & WZ_OFFSET (dove & è il simbolo della concatenazione), in particolare:
    - WZ_NUM è definita come il numero della working-zone al quale l’indirizzo appartiene in codifica binaria
    - WZ_OFFSET è definita come l’offset rispetto all’indirizzo base della working-zone a cui l’indirizzo da valutare appartiene in codifica one hot considerando il bit 0 il meno significativo, in pratica:
        - WZ_OFFSET = 0 è codificato come 0001
        - WZ_OFFSET = 1 è codificato come 0010
        - WZ_OFFSET = 2 è codificato come 0100
        - WZ_OFFSET = 3 è codificato come 1000

Quindi partendo da un indirizzo da valutare (ADDR) di 7 bit validi dopo questa trasformazione si ottiene un indirizzo di 8 bit.

1.3 Dati

I dati, ciascuno di dimensione 8 bit sono memorizzati in una memoria con indirizzamento al Byte partendo dalla posizione 0. Anche l'indirizzo che è da specifica di 7 bit viene memorizzato su 8 bit dove il valore dell'ottavo bit sarà sempre zero.

Indirizzo	Ram
0	WZ_0
1	WZ_1
2	WZ_2
3	WZ_3
4	WZ_4
5	WZ_5
6	WZ_6
7	WZ_7
8	Indirizzo_da_codificare
9	Indirizzo_codificato

Indirizzo base Working Zone

Figura 1: Rappresentazione dati in memoria

1.4 Interfaccia Componente

Il componente descritto ha la seguente interfaccia mostrata in Figura 2

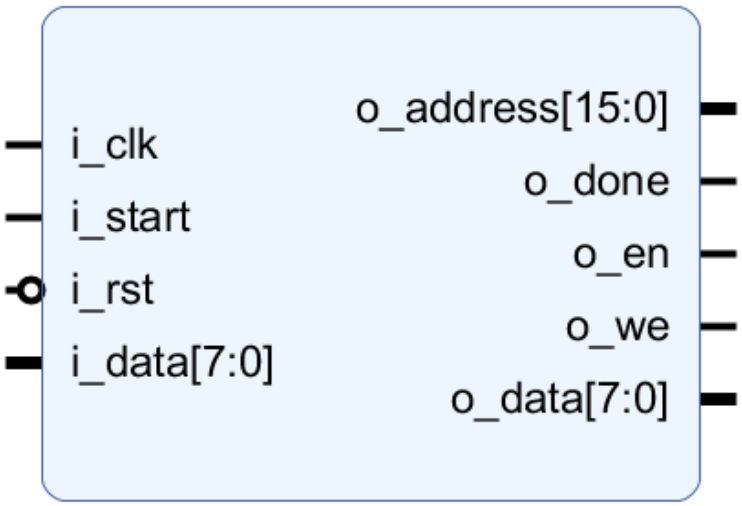

Figura 2: Interfaccia componente

In particolare:

- i_clk è il segnale di CLOCK in ingresso generato dal test banch;
- i_start è il segnale di START generato dal test banch;
- i_rst è il segnale di RESET che inizializza la macchina pronta per ricevere il primo segnale di START;
- i_data è il segnale (vettore) che arriva dalla memoria in seguito ad una richiesta di lettura;
- o_address è il segnale (vettore) di uscita che manda l'indirizzo alla memoria;
- o_done è il segnale di uscita che comunica la fine dell'elaborazione e il dato di uscita scritto in memoria;
- o_en è il segnale di ENABLE da dover mandare alla memoria per poter comunicare (sia in lettura che in scrittura);
- o_we è il segnale di WRITE ENABLE da dover mandare alla memoria (=1) per poter scriverci. Per leggere da memoria esso deve essere 0;
- o_data è il segnale (vettore) di uscita dal componente verso la memoria.

2. Design

Il modulo implementato presenta 2 processi, uno per la gestione dei registri mentre l'altro per implementare la macchina a stati finiti.

L'interazione con il modulo avviene tramite due segnali i_start e i_rst. Quando il segnale i_start in ingresso viene portato a livello logico alto il componente sviluppato inizia l'elaborazione spostandosi al primo stato della computazione. Una volta terminata la computazione e scritto il dato nella memoria del componente alza il segnale o_done, quest'ultimo viene mantenuto fino a quando il segnale i_start non viene abbassato e a questo punto la macchina passa allo stato STATE_START in attesa di un nuovo segnale i_start.

Il modulo dispone di un segnale i_rst che ha la funzionalità di portare il componente ad uno stato noto STATE_START pronto per iniziare una nuova computazione.

2.1 Stati della macchina

La macchina costruita è composta da 5 stati. Qui di seguito è fornita una breve descrizione per ciascuno di questi.

1. WAITING_FOR_START

   Questo è lo stato zero, lo stato in cui la macchina si trova quando viene avviata per la prima volta, in cui attende il primo segnale di i_rst per prepararsi ed attendere il primo segnale i_start.

2. STATE_START

   Stato in cui la macchina è pronta ed in attesa del segnale i_start passi al livello logico alto per iniziare la computazione.

3. STATE_LOAD

   Stato in cui vengono letti gli indirizzi delle working-zone dalla memoria e salvati nella cella corrispondente all'enumerazione di essa nell'array all'interno del componente.

4. STATE_WRITE

   Stato di durata fissa di un clock in cui viene letto l'indirizzo da valutare, calcolato e salvato in memoria e dunque alzato il segnale o_done al livello logico alto. Nella computazione vengono utilizzati i dati salvati precedentemente nell'array del componente.

5. STATE_DONE

Stato in cui la macchina attende l'abbassamento al livello logico basso del segnale i_start, rispondendo in seguito abbassando o_done e passando allo stato STATE_START.

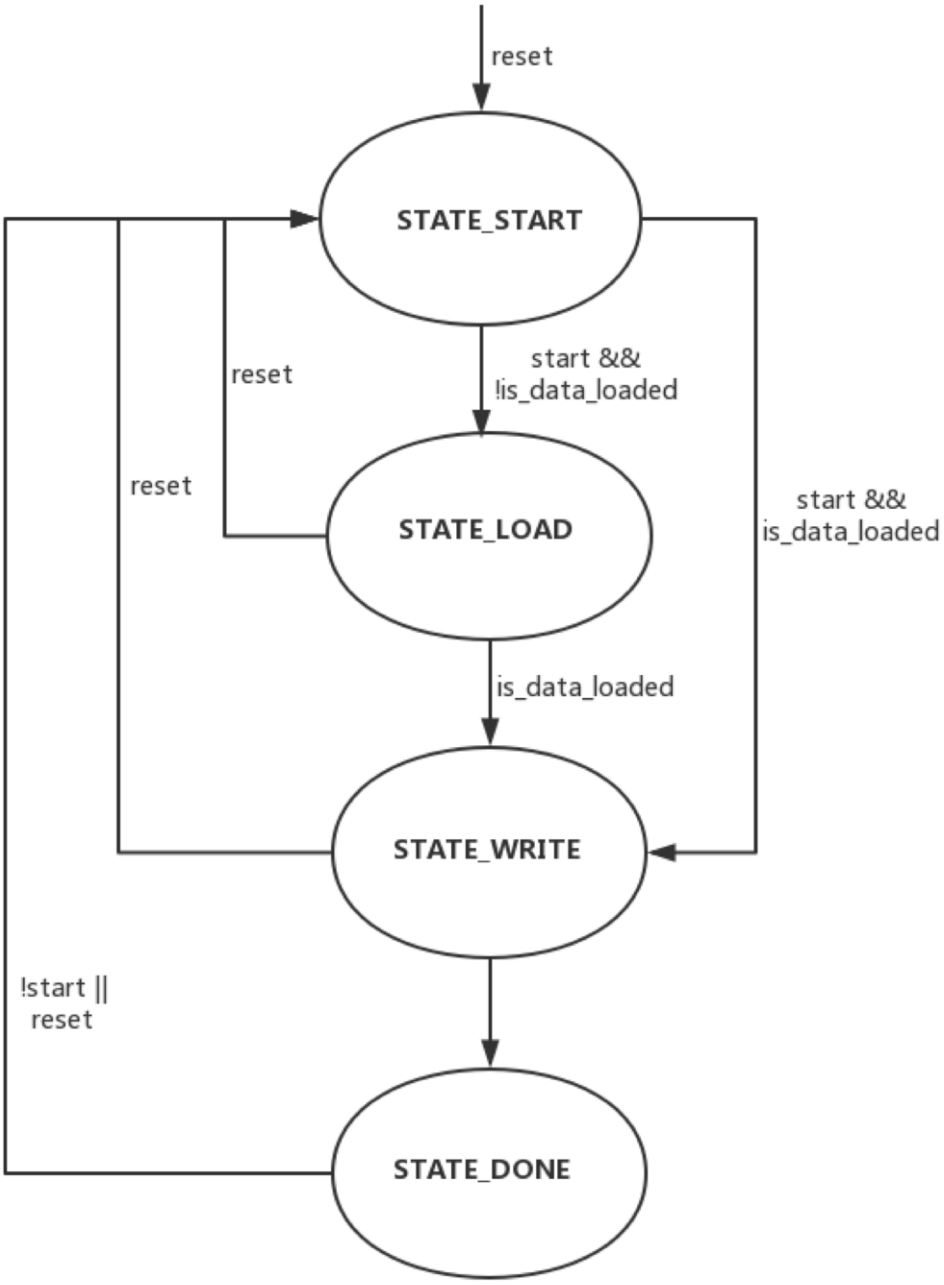

Figura 3: Diagramma degli stati della macchina

2.2 Scelte progettuali

Nella progettazione di questo modulo si è posta l'attenzione sul numero dei clock necessari per completare tutte le operazioni e quindi di generare l'indirizzo finale. In particolare, la lettura sequenziale dei dati dalla memoria in modo da ricevere un indirizzo ad ogni clock dopo il primo di attesa e il salvataggio di questi all'interno del componente in modo tale da non dover richiedere i dati dalla memoria se non dopo un segnale di reset. In particolare, questi indirizzi base delle working-zone vengono salvati in un array di interi, la ragione di questa scelta è per una migliore scalabilità del modulo nel caso in cui si volesse cambiare il numero delle working-zone da considerare.

3. Risultati

3.1 Sintesi

Il componente descritto viene correttamente sintetizzato ed implementato su FPGA (xc7a200tfbg484-1), in seguito gli *screeshot* dei risultati mostrati da Vivado.

Timing	**Setup** | Hold | Pulse Width
Worst Negative Slack (WNS):	93.257 ns
Total Negative Slack (TNS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	236

Figura 4: Vivado Timing Setup

Timing	Setup | **Hold** | Pulse Width
Worst Hold Slack (WHS):	0.207 ns
Total Hold Slack (THS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	236

Figura 5: Vivado Timing Hold

Timing Setup | Hold | **Pulse Width**

Worst Pulse Width Slack (WPWS):	49.5 ns
Total Pulse Width Negative Slack (TPWS):	0 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	115

Figura 6: Vivado Timing Pulse Width

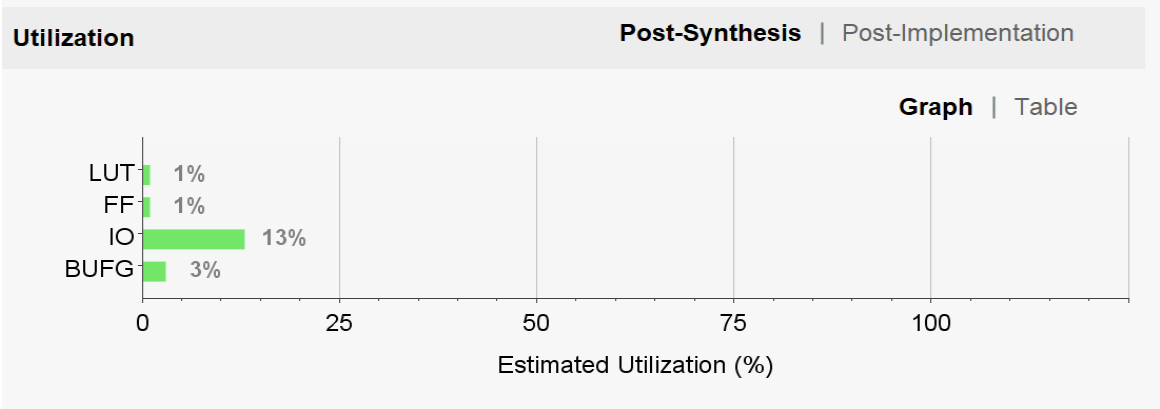

Figura 7: Vivado Utilization Post-Synthesis

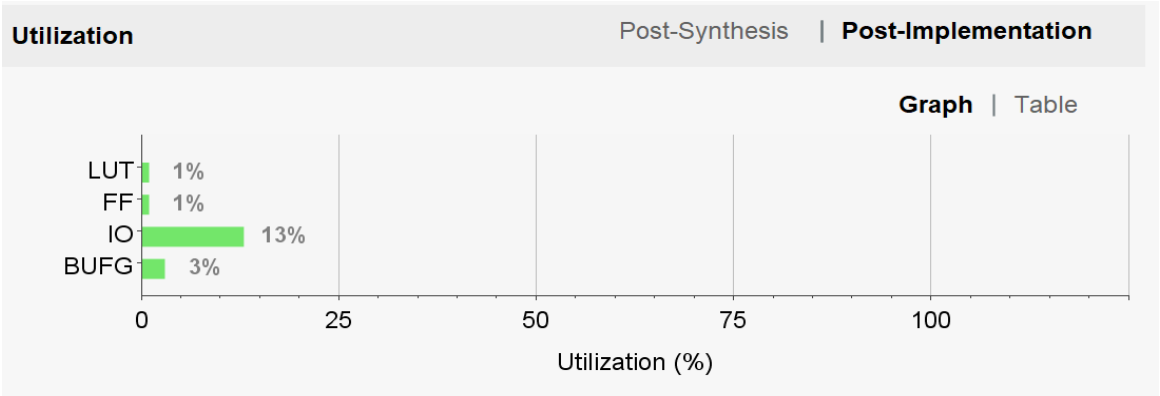

Figura 8: Vivado Utilization Post-Implementation

3.2 Test Banch

Qui in seguito vengono riportati i test bench più significativi di quelli utilizzati per verificare il corretto funzionamento del componente. In particolare, sono presenti 4 casi limiti con le relative descrizioni, motivazioni e *screenshot* dell'andamento dei segnali.

1. Doppio start senza reset tra i due. Verifica del corretto funzionamento del riutilizzo dei dati salvati nel componente dopo la prima lettura.

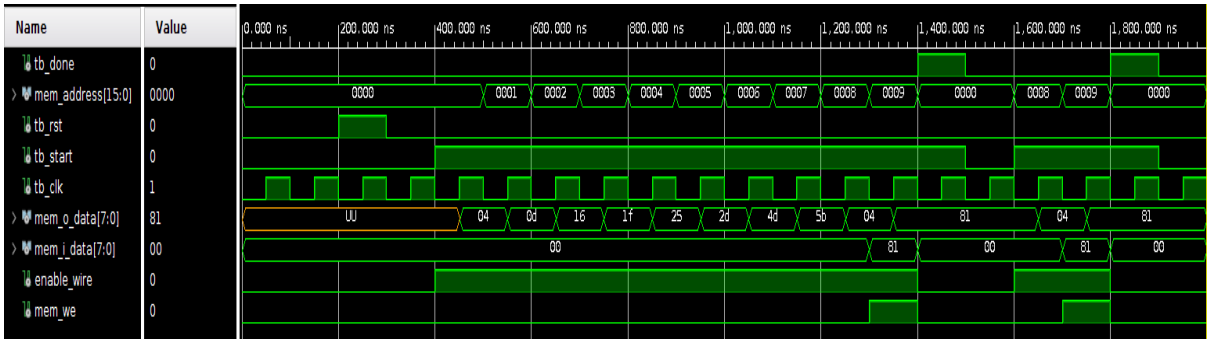

Figura 9: Doppio start senza reset tra i due

2. Doppio start con reset fra i due. Verifica del corretto funzionamento dell'invalidamento dei dati salvati precedentemente.

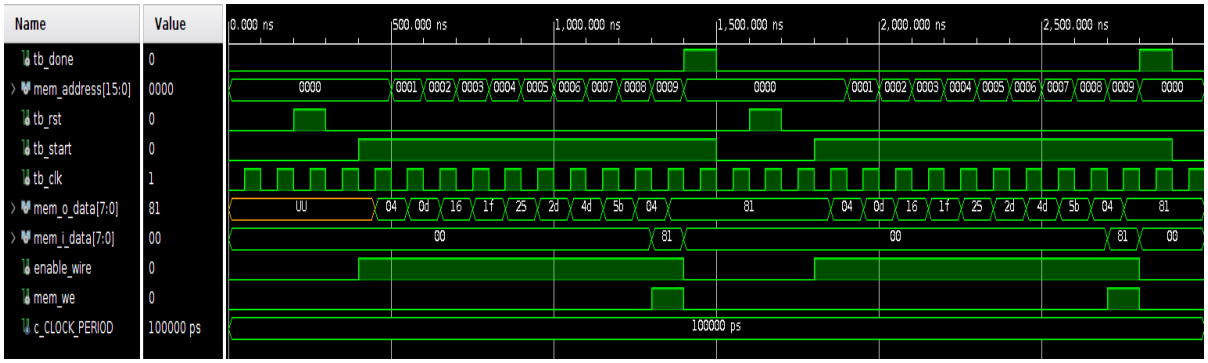

Figura 10: Doppio start con reset tra i due

4. Doppio start con reset contemporaneo insieme al secondo start. Verifica della corretta transizione dello stato e invalidamento dei dati nel caso in cui start e reset si presentano insieme.

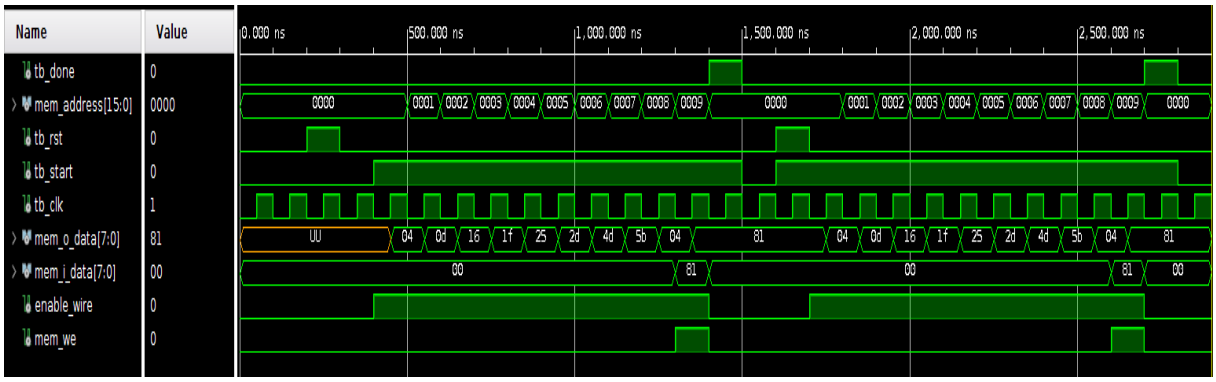

Figura 11: Doppio start con reset in contemporanea con il secondo start

5. Doppio start con reset ritardato di un clock sul secondo start. Verifica della corretta transizione dello stato e invalidamento dei dati nel caso in cui reset si presenti dopo lo start.

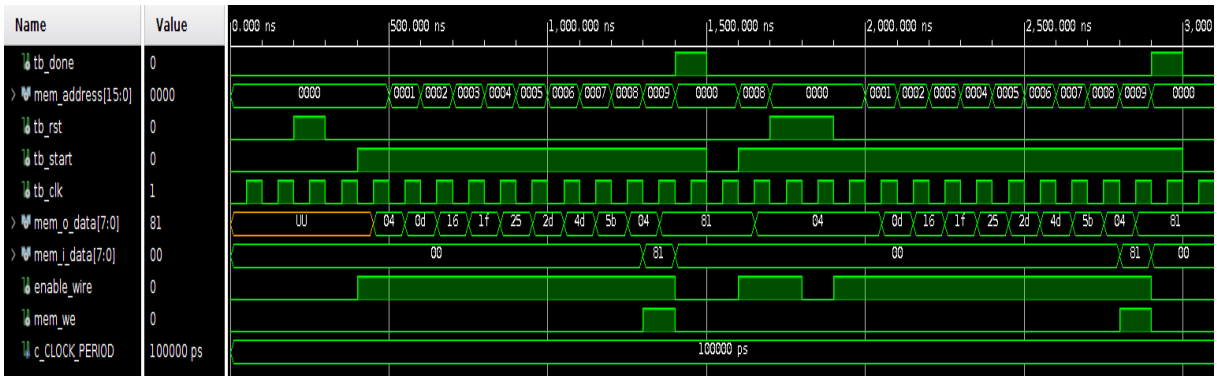

Figura 12: Doppio start con reset ritardato di un clock sul secondo start

4. Conclusioni

Le scelte di progettazione hanno portato alla riduzione da 18 a 9 cicli di clock per la lettura sequenziale degli indirizzi, incluso anche l'indirizzo da valutare, e alla riduzione da 18 a 1 cicli di clock nel caso di una computazione senza reset dei dati salvati. Di conseguenza il modulo implementato ha bisogno di un totale di 10 cicli di clock per eseguire una computazione dopo un reset o 2 cicli in assenza di quest'ultima. Comparando questi dati contro una versione senza la lettura sequenziale e il salvataggio (19 cicli di clock totali), sì ottiene una diminuzione rispetto al numero totale necessario per fare la computazione rispettivamente del 47% e 84%.