Progetto finale di Reti Logiche

Weger Marco - Matricola n° 888201

Anno Accademico 2019/2020

Contenuto

1 Introduzione

Per questo progetto mi sono posto l'obiettivo di descrivere un componente che rispetti le specifiche sia in pre sintesi che in post sintesi. Ho voluto scrivere del codice di facile lettura e che si adatti in modo semplice e rapido a qualsiasi tipo di modifica del pattern e/o della dimensione della memoria e del suo contenuto (più dettagli in seguito). La FPGA consigliata non ci pone particolari vincoli di area nonostante ciò ho voluto dare particolare riguardo sia ai tempi di esecuzione che all'area occupata. Per quanto riguarda la frequenza di clock non sono andato alla ricerca di una massimizzazione in quanto la specifica fissa il periodo di clock a 100 ns.

1.1 Obiettivi aggiuntivi

Dopo essermi assicurato di rispettate le richieste della specifica fornita mi sono posto i seguenti obiettivi:

1. Minimizzare il tempo trascorso dal momento che il segnale di start viene ricevuto al momento di invio del segnale di done;
2. Disattivare il segnale di enable della memoria tra le varie esecuzioni;
3. Descrivere un componente in grado di funzionare anche nel caso ci fossero reset asincroni durante l'esecuzione di una codifica;
4. Rendere il componente adattabile ad un'eventuale modifica della lunghezza dell'indirizzo della cella di memoria tramite una costante;
5. Rendere il componente adattabile ad un'eventuale modifica della dimensione di una singola cella di memoria tramite una costante (ADDR);
6. Rendere il componente adattabile ad un'eventuale modifica del numero di elementi in una working-zone tramite una costante (WZ_OFFSET);
7. Rendere il componente adattabile ad un'eventuale modifica del numero di working-zone tramite una costante (WZ_NUM).

Al fine di raggiungere i sopracitati obiettivi ho assunto che l'indirizzo da codificare e l'indirizzo codificato vengano sempre salvati in successione in celle immediatamente consecutive all'ultimo indirizzo di working-zone (es. se ci fosse 16 working-zone RAM(16) conterrebbe l'indirizzo da codificare e RAM(17) l'indirizzo codificato). Tutte le ottimizzazioni descritte in seguito sono state valutate sulla base dei dati forniti dalla specifica e non tengono conto dell'eventuale crescita sproposita delle costanti sopracitate.

1.2 Obiettivo velocità

La limitazione più stringente in termini di velocità è data dall'accesso alla RAM e dai suoi ritardi. Al fine di minimizzare i tempi di letture e confronti ho

optato per un componente che al momento della lettura di un dato imposti contemporaneamente la successiva richiesta. In questo modo posso garantire l'esecuzione dei confronti tra l'indirizzo e le N working-zone in N+1 cicli di clock. Occorreranno poi 2 cicli di clock per la scrittura del dato codificato in memoria e la notificazione di elaborazione completata. Un ulteriorie ciclo di clock mi è servito a garantire che il segnale *o_mem* non rimanesse attivo tra un'esecuzione e la successiva. Ricapitolando, nel caso pessimo in cui l'indirizzo letto non si trova in nessuna delle work-zone

$$T_{\text{esecuzione}} = (N + 1) \cdot T_{\text{clock}} + 3 \cdot T_{\text{clock}}$$

Nella valutazione del tempo di esecuzione va preso in considerazione il fatto che il segnale di start potrebbe arrivare sul fronte di discesa del clock, qualora succedesse l'esecuzione ritarderà di $T_{\text{clock}}/2$. Nella specifica forntia l'esecuzione massima durerà 1200 ns (nel caso in cui *i_start* fosse allineato al fronte di salita del clock), 1250 ns altrimenti.

1.3 Obiettivo codice semplice e riutilizzabile

Da scrivere...

1.4 Funzionamento in sintesi

Una soluzione che memorizza tramite registri i valori delle working-zone non avrebbe migliorato in modo significativo i tempi di esecuzione peggiorando però l'area occupata pertando ho optato per la in memoria a ogni esecuzione. La singola esecuzione di una codifica può essere descritta attraverso un numero finito di step (che poi diventeranno una macchina a stati finiti):

1. Reset ed attesa del segnale di start (*i_start=1*);
2. Abilitazione della memoria e richiesta dell'indirizo da codificare (salvato in un registro);
3. Richiesta della i-esima working-zone e confronto con l'indirizzo salvato, eventuale codifica e passaggio a step successivo (passo ripetuto per i compreso tra 0 e il numero di working-zone);
4. Scrittura dell'indirizzo codificato in memoria;
5. Invio segnale di elaborazione completata (*o_done=1*) e attesa feedback (*i_start=0*), il dato è disponibile fin dal momento in cui *o_done* viene portato a 1;

1.5 Note aggiuntive sulla specifica

Per la sintesi è stata scelta l'FPGA xc7a200tfbg484-1.

2 Architettura

2.1 Macchina a stati finiti

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

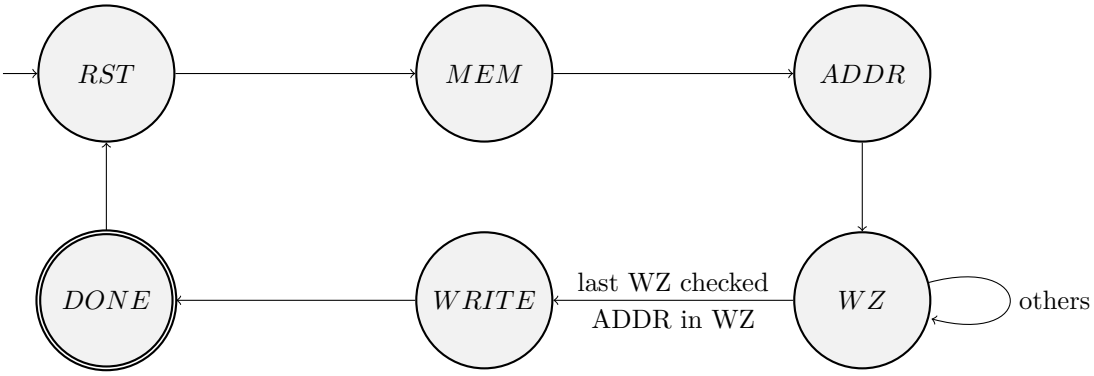

Figura 1: macchina a stati finiti implementata.

Stato	Descrizione
RST	
MEM	
ADDR	
WZ	
WRITE	
DONE	

Tabella 1: stati della macchina a stati finiti implementata.

3 Sintesi

4 Simulazioni

5 Conclusione