

Progetto finale di Reti Logiche

Weger Marco - Matricola nr 888201

Anno Accademico 2019/2020

Contenuto

1	Introduzione	1
1.1	Obiettivi aggiuntivi	1
1.2	Obiettivo velocità	1
1.3	Obiettivo codice semplice	2
1.4	Funzionamento in sintesi	2
1.5	Note aggiuntive sulla specifica	2
2	Architettura	3
2.1	Macchina a stati finiti	3
3	Sintesi	3
4	Simulazioni	3
5	Conclusione	3

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 rispettare 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

Da scrivere...

Tabella 1: stati della macchina a stati finiti implementata.

Stato	Descrizione
RST	
MEM	
ADDR	
WZ	
WRITE	
DONE	

1.3 Obiettivo codice semplice

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 pertanto 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'indirizzo 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

Come consigliato sul regolamento per la sintesi è stata scelta l'FPGA xc7a200tfbg484-1.

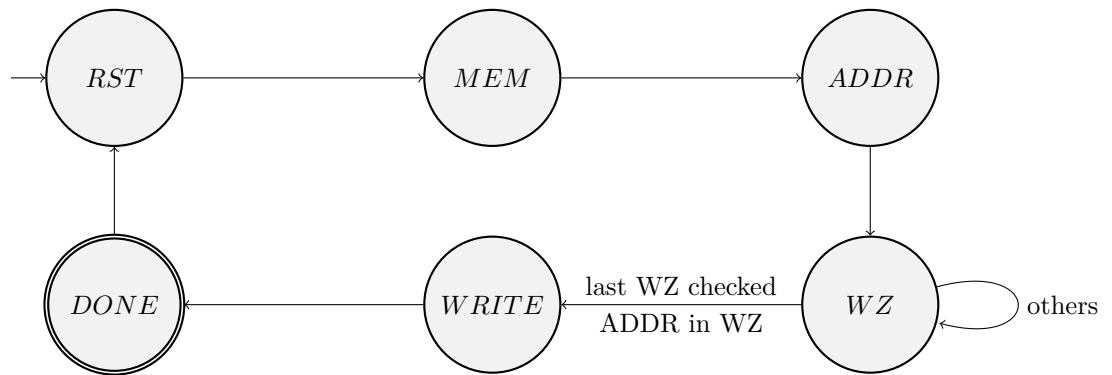


Figura 1: macchina a stati finiti implementata.

2 Architettura

2.1 Macchina a stati finiti

3 Sintesi

4 Simulazioni

5 Conclusione