Prova finale di reti logiche

Riccardo Zappa

anno accademico 2021/2022

matricola: 935741

codice persona: 10658440

Contents

1 Introduzione

La specifica della Prova Finale (Progetto di Reti Logiche)2021/2022 chiede di implementare un modulo HW che preleva da una memoria RAM una sequenza di parole, ognuna da 8 bit, le serializza in uno stream di bit u_k su cui viene applicato il codice convoluzionale $\frac{1}{2}$, mostrato in figura 1 in forma di macchina a stati finiti, producendo uno stream y_k di bit, formato come concatenazione delle uscite p_1k e p_2k del convolutore. Lo stream di bit y_k viene infine parallelizzato su parole di 8 bit e scritto in memoria.

2 Architettura

2.1 Descrizione ad alto livello

Da un'ottica ad alto livello, l'implementazione viene divisa in 5 macro-parti:

1. Prelevare il numero di parole e le parole da codificare interfacciandosi con la RAM.
2. Serializzare le parole in uno stream di bit da inviare al convolutore.
3. implementare l'algoritmo del convolutore.
4. parallelizzare il flusso di uscita dal convolutore in parole da 8 bit.
5. scrivere in memoria le parole prodotte

La scelta implementativa ricade su una macchina a stati finiti che viene gestita da un unico process.

2.2 Macchina a stati finiti

Di seguito una descrizione degli stati appartenenti alla macchina a stati finiti implementata:

- **RESET:** stato in cui si trova la fsm quando viene alzato a 1 il reset e in cui vengono inizializzati tutti i segnali.
- **IDLING:** stato di IDLE in cui viene aspettato il segnale di start.
- **READ-NWORD:**stato che salva il numero di parole da prelevare dalla ram.
- **SETTING-ADDRESS:**stato in cui vengono calcolati gli indirizzi per prelevare le parole da codificare dalla RAM.
- **READ-WORD:**stato in cui vengono prelevate le parole da codificare.
- **PRE-CONV:** stato in cui viene effettuata la serializzazione delle parole in uno stream di bit.

- **CONVOLUTION:** stato in cui viene effettuata la codifica dello stream di bit attraverso l'utilizzo di un ulteriore msf, caratterizzata dagli stati S0,S1,S2 e S3.
- **POST-CONV:** stato adibito ad assegnare i bit in uscita dal convolutore ad un vettore di 16 bit che verrà poi inviato alla memoria separato in due parole di 8 bit.
- **SENDER1:** stato in cui viene inviata alla memoria la prima parola delle due prodotte, i bit inviati sono quelli dal 15 all'8.
- **SENDER2:** stato in cui si invia alla RAM la seconda parola prodotta, i bit inviati sono quelli dal 7 allo 0. In questo stato inoltre si effettua il controllo di fine elaborazione, nel caso in cui sia stato prelevato il numero di parole contenuto in n-word.
- **FINALIZE:** stato in cui si aspetta che il segnale i-start venga abbassato a 0 e si riporta la fsm in stato di reset.

Lo schema della macchina a stati viene riportato nella pagina successiva.

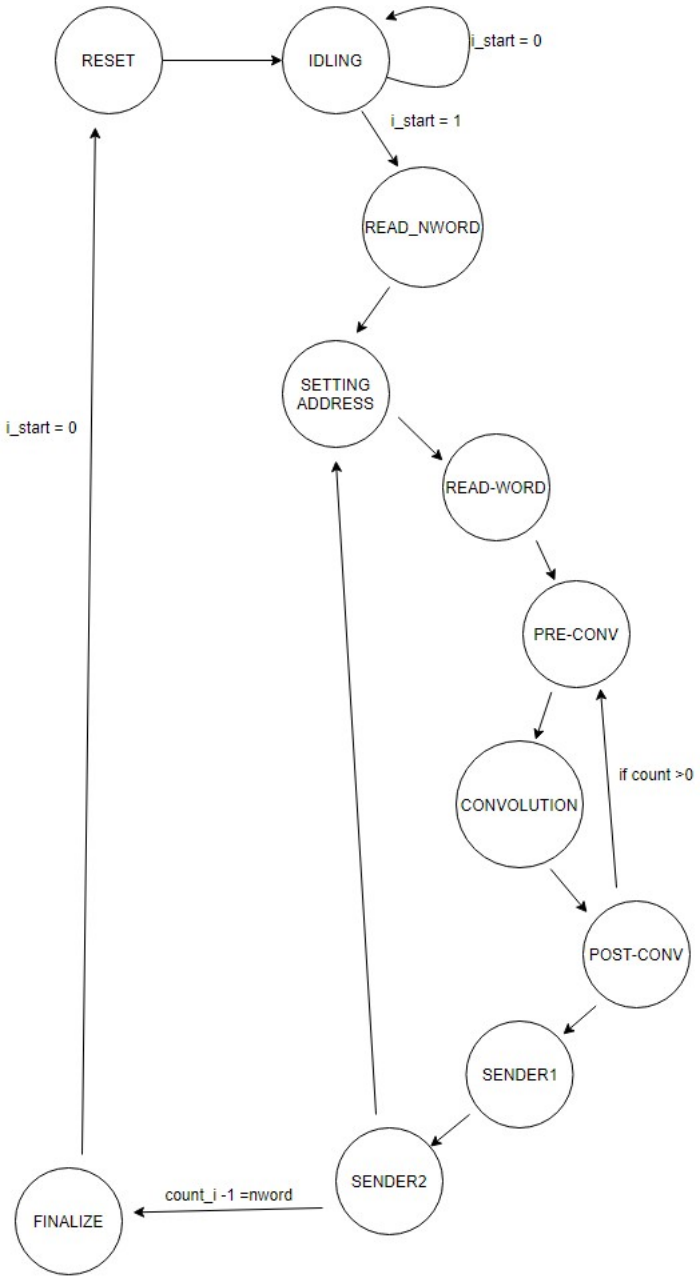

Figure 1: diagramma degli stati della Macchina a Stati Finiti

3 Risultati sperimentali

3.1 Sintesi

Per la sintesi del componente, come indicato dalla tabella 3.1 , sono stati utilizzati 132 flip flop e non sono stati inferiti latch che avrebbero precluso la correttezza del funzionamento delle simulazioni post-synthesis.

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	242	0	134600	0.18
LUT as Logic	242	0	134600	0.18
LUT as Memory	0	0	46200	0.00
Slice Registers	132	0	269200	0.05
Register as Flip Flop	132	0	269200	0.05
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

Anche per quanto riguarda il timing report è stata rispettato il costraint di 100 ns del ciclo di clock, avendo un arrival time di 5.958ns.
Slack (MET) : 94.042ns (required time - arrival time)

3.2 simulazioni

Tutte le simulazioni dei test bench utilizzati sono state passate sia in behevioral che in post-synthesis simulation.

1. **Testbench controllo risultati** La prima funzionalità testata è stata quella della corretta produzione degli output e del salvataggio degli stessi nei giusti indirizzi della RAM. Per farlo il componente è stato sottoposto a vari testbench con assegnamenti di input differenti, fino a testare anche la sequenza massima di 255 parole come da specifica.

2. **Testbench controllo funzionamento reset** La seconda funzionalità testata è il corretto funzionamento del reset, nel componente. Oltre a testare il corretto funzionamento del reset prima di ogni computazione, come da specifica di progetto, sono state testate anche situazioni in cui il reset viene alzato durante una computazione.

3. **Testbench caso limite N-word = 0** Con questo testbench viene testato la non scrittura sulla RAM in caso di numero di parole dato in input uguale a 0.

4. **Testbench computazioni successive** Con questo testbench è stata testata la capacità del componente di codificare più flussi uno dopo l'altro, come da specifica.

4 conclusioni

La scelta progettuale è ricaduta sull'utilizzo di un solo process senza ricorrere a component esterni, con particolare attenzione al non inferire latch che avrebbero precluso il funzionamento post-synthesis del modulo. Il componente realizzato si ritiene dunque che soddisfi tutte le specifiche e i vincoli assegnati dal progetto, fatto assodato tramite il testing behevioral e post-synthesis di tutti i casi limite individuati.