Progetto Reti Logiche

AA 2019-2020 Politecnico di Milano

Indice

Analisi della specifica

La specifica del progetto richiede la creazione in VHDL di un componente in grado di riconoscere e codificare indirizzi di 7 bit appartenenti a delle working-zone.

Il componente deve supportare 8 working-zone formate da intervalli come: [wz_addr, wz_addr+3], dove *wz_addr* è l'indirizzo base della zona.

Se l'indirizzo da elaborare appartiene a una delle working-zone va ricodificato nel seguente modo:

1 | n. working-zone (3bit) | offset one hot (4bit)

Se l'indirizzo non appartiene a nessuna zona questo deve rimanere invariato e avrà una codifica simile a:

0 | indirizzo originale (7bit)

Per funzionare, il componente deve leggere gli indirizzi delle working-zone dalla RAM. Gli indirizzi delle working-zone sono caricati nelle prime otto posizioni (index 0-7) della RAM, mentre l'indirizzo da codificare si trova nella nona posizione (index 8). Il risultato elaborato deve essere infine scritto alla decima posizione della RAM (index 9).

Il processo di codifica comincia unicamente quando viene alzato il segnale di ***i_start,*** che viene mantenuto alto finché il componente non alza a sua volta l'uscita ***o_done*** ad indicare che ha terminato l'elaborazione.

La specifica impone la seguente interfaccia del componente:

SEGNALE	DESCRIZIONE	TIPO
i_clk	Clock del sistema fornito esternamente (100ns)	Input
i_start	Segnale di avvio della codifica	Input
i_rst	Segnale di reset del componente	Input
i_data	Indirizzo di 8 bit proveniente dalla RAM	Input
o_address	Indirizzo a 16 bit della RAM in cui si vogliono leggere i dati	Input
o_done	Segnale che indica il termine del processare di codifica dell'indirizzo	Output
o_en	Segnale che abilita la RAM	Output
o_we	Segnale che abilita la scrittura in RAM	Output
o_data	Risultato della codifica dell'indirizzo svolto dal componente	Output

Non è presente alcuna porta di uscita per segnalare eventuali errori.

Design del componente

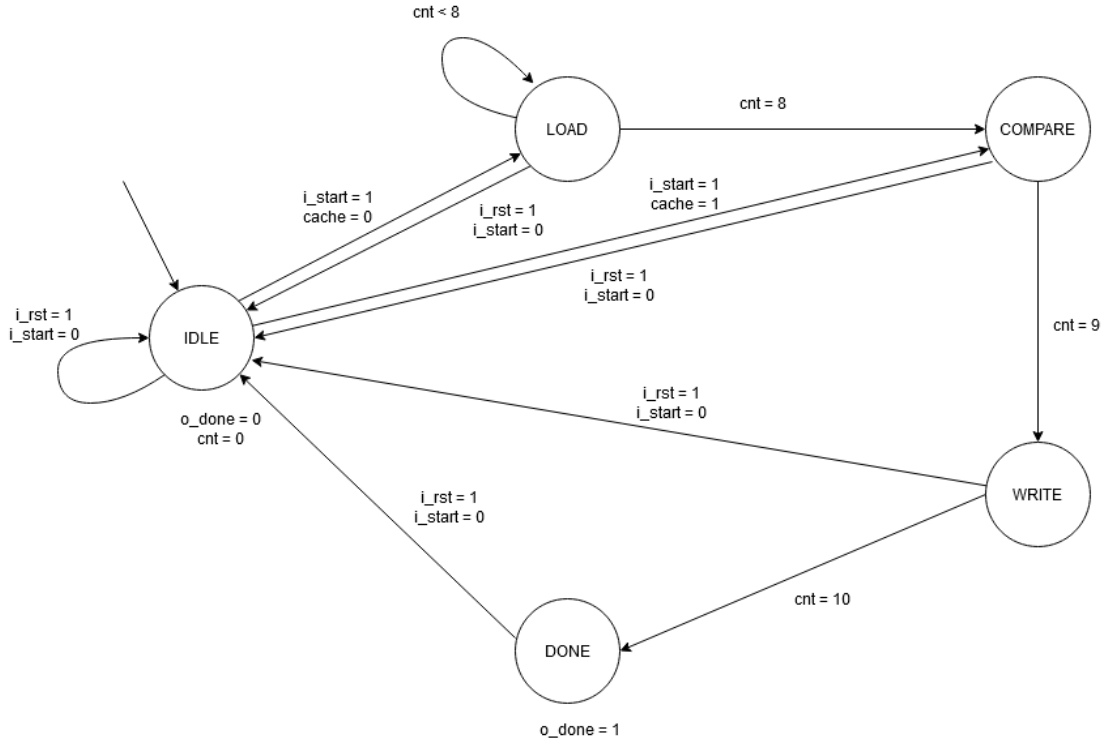
Dato che la specifica afferma che le working-zone non cambiano durante la stessa esecuzione, è possibile utilizzare una cache per evitare tutta la fase di caricamento delle zone e ottimizzare cosi i tempi di codifica.

L'intero funzionamento del componente si può basare su un contatore binario a 4bit che conta da 0 e 9. Tutti i valori del contatore possono essere usati per indicare a che indirizzo si vuole

leggere/scrivere la RAM (da specifica è richiesto di operare solo sulle prime 10 locazioni di memoria). Inoltre, è possibile assegnare univocamente un'operazione ad ogni valore.

L'assegnazione univoca tra valori del contatore e operazioni da svolgere è molto utile per realizzare il funzionamento della cache. Infatti, per saltare l'intera fase di caricamento delle working-zone è sufficiente iniziare a conteggiare dal valore a cui è assegnata la fase di "processo" (in questo caso da 8).

L'intero funzionamento del componente può essere diviso in stati:



STATO	**OPERAZIONI SVOLTE**
INDLE	Stato di default del componente in cui attende lo start per avviare la codifica. Questo stato è raggiunto in caso di reset o se il segnale start viene abbassato. In questo stato la cache viene conservata o meno a seconda del valore del segnale di reset.
LOAD	Stato in cui il componente carica gli otto indirizzi delle working-zone in un registro. Al termine del caricamento viene attivato un "flag" che indica la possibilità di usare la cache nelle successive codifiche. Questa fase è mappata sui valori del contatore tra 0 e 7.
COMPARE	Stato in cui il componente calcola l'offset dell'indirizzo da codificare rispetto ad ogni working-zone per controllarne l'appartenenza o meno. Al termine delle sottrazioni viene preparato il risultato con la codifica richiesta dalla specifica. Questa fase è mappabile sul contatore per il valore 8.
WRITE	Abilita la scrittura in RAM (alzando il segnale ***o_we***) e scrive il risultato calcolato nella fase precedente. Questa fase è mappabile sul contatore per il valore 9.

DONE	Alza il segnale ***o_done*** e attende che ***i_start*** sia abbassato così da poter ricevere nuovi comandi.

Da specifica, la RAM rende disponibile un indirizzo al fronte di salita del clock con un ritardo di 1ns. Per evitare di leggere indirizzi non ancora aggiornati è conveniente sincronizzare il componente sul fronte di discesa del clock.

Per utilizzare la RAM il componente deve tenere ***o_en*** alto. Visto che la specifica garantisce che ***i_start*** è sempre alto durante tutta si può assegnare direttamente ***i_start*** a ***o_en***.

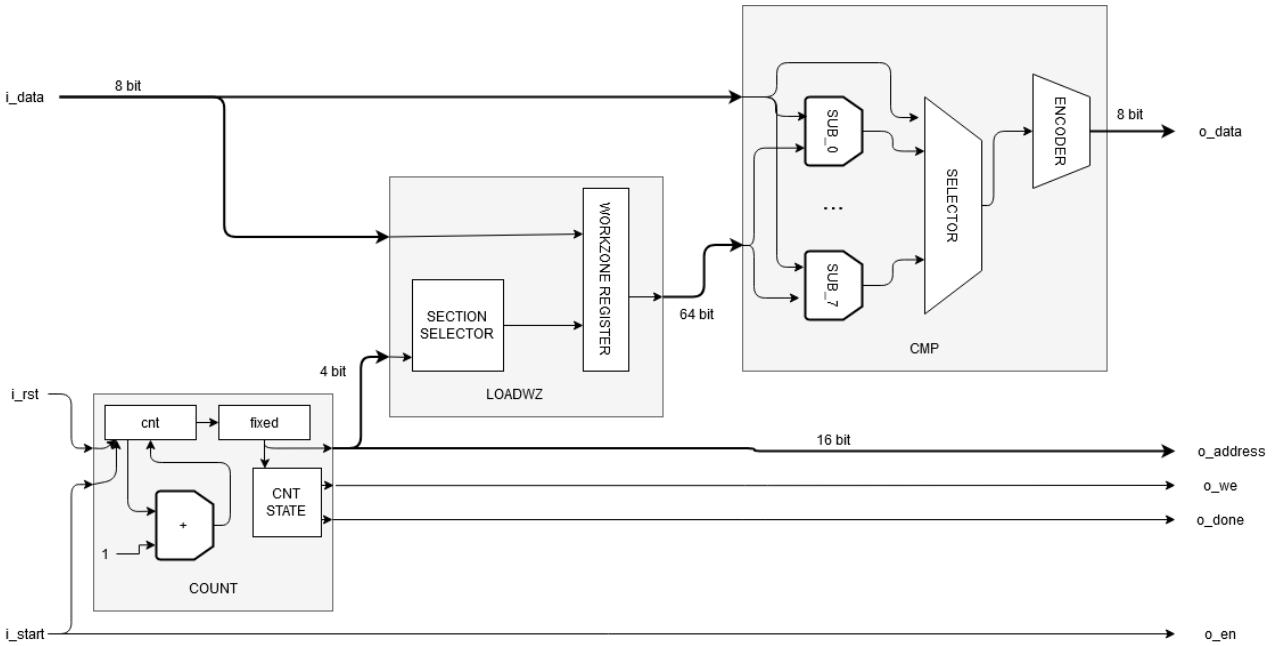
I segnali di reset e start, a differenza del resto del componente, sono entrambi asincroni perché è più semplice gestirli insieme. Entrambi i segnali possono essere immaginati come dei "suggerimenti" per scegliere il valore inziale del contatore. In particolare, il reset forza il contatore a partire da 0, mentre start lascia al contatore la possibilità di usare la cache e quindi di partire da 8.

Architettura

Il componente realizzato è stato diviso in tre processi.

Non essendo presente alcuna porta per indicare errori, si presume che i valori forniti siano sempre corretti. In ogni caso tutto il componente è stato dimensionato per gestire indirizzi di 8 bit che garantiscono la robustezza ad eventuali errori a patto di violare la specifica sulla codifica.

Schematicamente possiamo immaginare la struttura del componente come segue:



COUNT

Questo processo realizza principalmente il contatore binario a 4bit. Da questo consegue che il processo deve gestire sia il segnale start sia il reset visto che questi sono dei "suggerimenti" per

scegliere il valore da cui iniziare a contare. Essendo il funzionamento della cache basato sul modificare il valore di partenza del contatore, questo processo ingloba anche la logica della gestione di quest'ultima.

Infine, gestisce i segnali ***o_we*** e ***o_done*** che sono anch'essi mappati su valori univoci del contatore.

LOADWZ

Questo processo rappresenta la fase di LOAD delle working-zone. Esegue operazioni unicamente se il contatore è tra 0 e 7. Il suo scopo è quello di caricare gli indirizzi in un registro a 64bit (che ha anche la funzione di cache). La dimensione del registro è stata scelta per poter contenere 8 indirizzi da 8bit l'uno. Il sovradimensionamento serve per dare più robustezza al componente nel caso in cui vengano forniti erroneamente indirizzi a 8bit e non a 7.

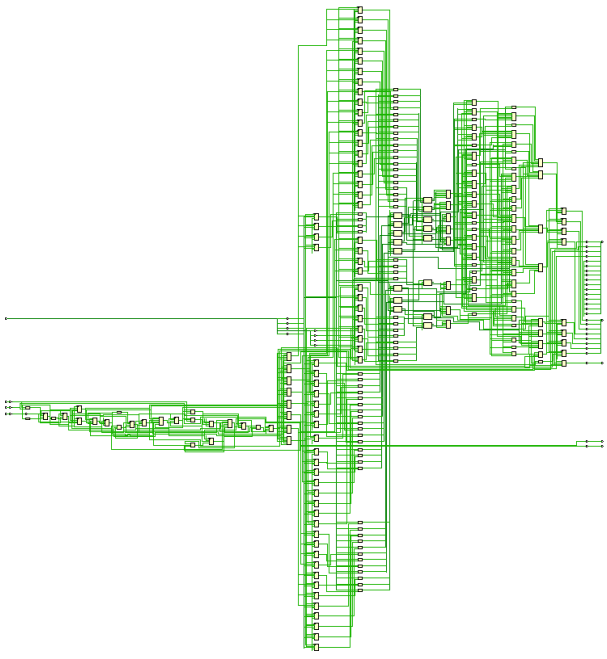
CMP

Questo processo rappresenta la fase di COMPARE ed è eseguito al termine del ciclo di clock in cui il contatore è pari ad 8. Vengono lette tutte le working-zone dalla cache e per ognuna di queste viene calcolato l'offset con l'indirizzo da elaborare. In caso l'offset sia minore di 4, si è trovata la working-zone a cui apparitene l'indirizzo. Il calcolo dell'offset è svolto in modo tale che in caso di working-zone sovrapposte venga scelta sempre quella trovata all'indirizzo più basso in RAM.

Infine, a seconda che sia stata trovata o meno una working-zone di appartenenza viene codificato l'indirizzo secondo la specifica e salvato in un registro apposito.

Sintesi Componente

Il componente viene sintetizzato con successo.



Vengono segnalate delle warning riguardanti che parte dell'indirizzo ***o_address*** è costante. Queste warning sono ignorabili visto che è necessario usare unicamente 4bit per poter accede a tutti gli indirizzi necessari al corretto funzionamento del componente.

Dai report risulta che il componente, quando sintetizzato, utilizza molti elementi sia di memoria che logici. Il risultato della sintesi è coerente col design usato in quanto si è preferito ottimizzare la velocita a discapito della semplicità della rete e della sua occupazione di area.

Site Type	Used
Slice LUTs*	134
LUT as Logic	134
LUT as Memory	0
Slice Registers	84
Register as Flip Flop	83
Register as Latch	1
F7 Muxes	0
F8 Muxes	0

```
Detailed RTL Component Info :
+---Adders :
      3 Input      8 Bit       Adders := 8
      2 Input      4 Bit       Adders := 1
+---Registers :
                  64 Bit    Registers := 1
                   8 Bit    Registers := 1
                   4 Bit    Registers := 2
                   1 Bit    Registers := 2
+---Muxes :
     10 Input     64 Bit        Muxes := 1
     20 Input      8 Bit        Muxes := 1
      2 Input      8 Bit        Muxes := 4
      4 Input      8 Bit        Muxes := 3
      2 Input      4 Bit        Muxes := 3
      2 Input      1 Bit        Muxes := 25
```

Dal report sul timing si può inoltre dedurre che il tempo di clock dato dalla specifica è abbondantemente superiore a quello richiesto dal componente sintetizzato. I test di timing indicano un WNS (Worst Negative Slack) di 93.265ns. Ciò significa che il componente realmente impiega circa 7ns a svolgere le operazioni più lunghe ed è quindi possibile ridurre a 15ns il clock mantenendo un buon margine di sicurezza per eventuali ritardi.

Worst Negative Slack (WNS):	93,265 ns
Total Negative Slack (TNS):	0,000 ns
Number of Failing Endpoints:	0
Total Number of Endpoints:	102

Verifica funzionamento

Per garantire il corretto funzionamento del componente sono stati eseguiti con successo tutti i vari tipi test elencati in seguito usando Vivado 2019.2 e 2016.1. I test sono stati eseguiti in tutte le modalità di simulazione disponibili: pre-sintesi, post-sintesi, implementazione.

TEST	COMPORTAMENTO TESTATO
No working-zone	Testa la capacità del componente di riconosce che l'indirizzo fornito non appartiene a nessuna delle working-zone lasciandolo inalterato come richiede la specifica.
Match working-zone	Quattro test dedicati a comprendere se il componente è in grado di riconosce e codificare correttamente degli indirizzi che appartengono a una working-zone. Inoltre, viene verificata la corretta codifica per tutti i possibili offset.
Cache	Testa il caricamento, il ricaricamento (dopo un reset) e l'uso della cache delle working-zone.
Reset asincrono	Testa che il reset sia asincrono e che una volta ricevuto il componente termina immediatamente l'esecuzione ed elimina la cache che possiede.
Start non costante	Verifica che in caso il segnale start venga portato a 0 durante un'elaborazione, il componente interrompe la sua esecuzione mantenendo la cache intatta (se pronta) per successivi usi.

Working-zone overlap	Verifica che in caso due o più working-zone siano sovrapposte, il componente codifica l'indirizzo usando la working-zone trovata all'indirizzo più basso in RAM.
Done	Testa che il componente mantiene l'uscita ***o_done*** alta finché non viene abbassato il comando di start.

Alternative progettuali

In alternativa al design scelto, per ottimizzare l'utilizzo dell'area a discapito della velocita, si può realizzare un componente che legge e salva il valore che si trova nell'ottava posizione della RAM in un registro a 8bit. Nei seguenti otto cicli di clock si leggono, uno alla volta, gli indirizzi delle working-zone dalla RAM e immediatamente si calcola l'offset tra il valore letto e l'indirizzo da codificato (salvato in precedenza su un registro). Se l'offset risulta minore di quattro si procede alla codifica secondo specifica dell'indirizzo e si prepara il ciclo di clock per la scrittura del risultato terminando il processo di codifica. Se si giunge all'ultimo ciclo senza aver trovato una working-zone di appartenenza si procede alla scrittura dell'indirizzo senza modifiche e si termina il processo.

Anche in questo caso risulta comodo utilizzare un contatore per gestire l'intero funzionamento del componente, tuttavia data la natura "progressiva" del componente vengono usati pochi elementi di memoria e logici poiché viene eseguita una singola operazione per ciclo di clock.