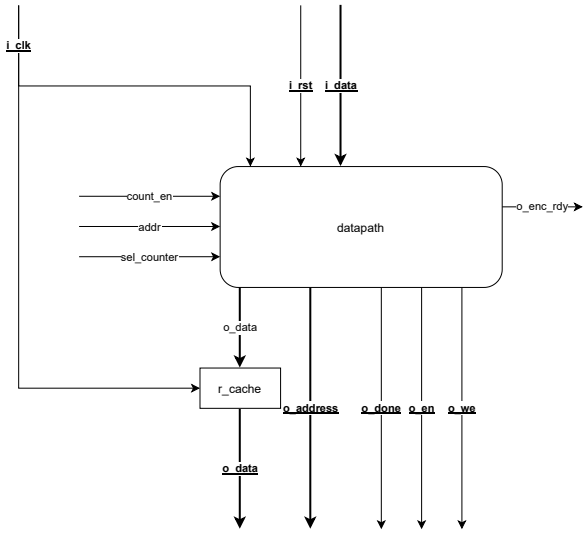
Progetto finale Reti Logiche

Luca De Martini

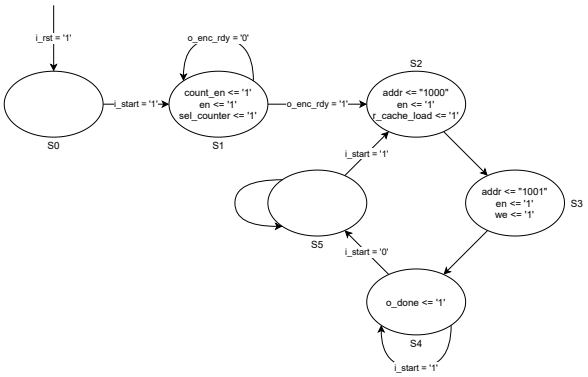
1 Introduzione

2 Architettura

Al livello più alto il componente presenta un modulo `project_reti_logiche` che contiene tutti i moduli interni, un'automa a stati finiti e un registro di cache dell'indirizzo in output.



L'automa a stati finiti è implementato con una collezione di process:
Due process descrivono il comportamento del registro di cache e quello del registro di stato, entrambi sono realizzati con flip-flop sincroni attivati sul fronte di discesa, la scelta di usare il falling edge è stata fatta per ridurre il numero di cicli di clock necessari per la codifica.
Un altro process aggiorna il valore del segnale `next_state` in base allo stato attuale, al segnale in input di `i_start` e al segnale `enc_rdy`, output del modulo interno `datapath` che verrà descritto nel dettaglio successivamente.
Infine un ultimo process aggiorna i valori dei segnali in input ai moduli interni e in output dal componente necessari al funzionamento basandosi sullo stato attuale.



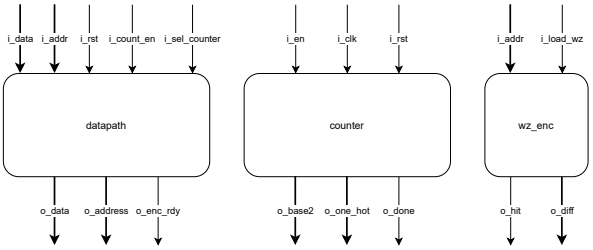
L'FSA ha 6 stati ed è caratterizzato da due zone di funzionamento, la prima zona è quella iniziale di setup costituita dagli stati 'S0' e 'S1': una volta ricevuto il segnale di `i_start` il componente si prepara alla codifica rimanendo nello stato 'S1' mentre sta caricando gli indirizzi delle working zones nei suoi registri interni.

La fine di questa fase è segnalata dal segnale `o_enc_rdy` proveniente dai componenti interni che viene alzato a '1' quando è pronto a codificare, avanzando così l'automa al prossimo stato 'S2' e iniziando la fase di codifica.

La fase di codifica iniza nello stato 'S2' dove il componente legge l'indirizzo da codificare, lo codifica e lo carica nel registro di cache in un unico ciclo di clock. Nello stato seguente viene scritto in RAM il risultato e in quello dopo viene alzato il segnale di `o_done` segnando la fine della codifica.

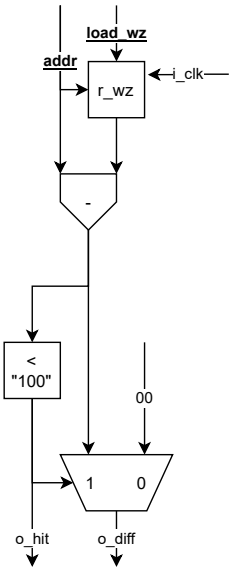
Dopo di che il componente attenderà l'abbassarsi del segnale di `i_start` per abbassare `o_done` e poi mettersi in attesa di un nuovo segnale di start nello stato 'S5', pronto per una nuova codifica.

2.1 Moduli interni



Il componente contiene un modulo `datapath` che ha al suo interno un modulo `counter` e 8 moduli `wz_enc` identici (uno per ogni indirizzo della working zone).

2.1.1 wz_enc

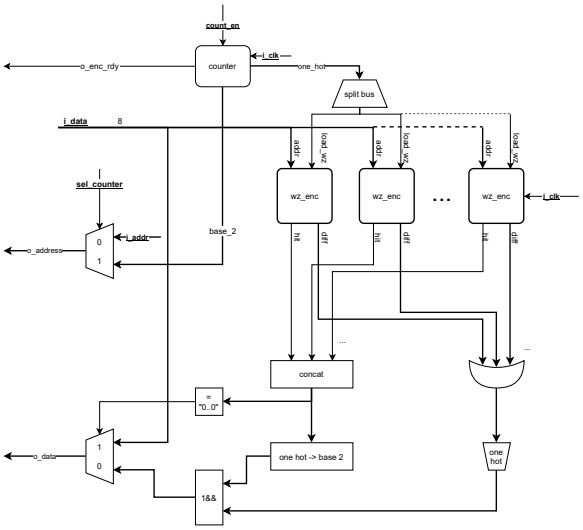


I moduli `wz_enc` si occupano di stabilire se un indirizzo sia all'interno di una determinata working zone, segnalandolo con `o_hit`, e di calcolarne l'offset in caso positivo. L'offest rispetto alla working zone è `o_diff` che è da considerarsi valido solo se `o_hit` è a '1', altrimenti avrà tutti i bit a '0'

2.1.2 counter

Il modulo `counter` è un contatore da 0 a 7 che scatta sul falling edge. Esso inizia a contare quando riceve un segnale di enable, dando l'output sia in codifica binaria, sia in codifica one hot, dopo aver sorpassato il 7 tutti i bit degli output vengono messi a '0'

2.1.3 datapath



Il `datapath` divide il bus contenente la codifica one hot in uscita dal `counter` collegando ogni bit del segnale al segnale di load di un `wz_enc`, in questo modo, usando il segnale in codifica binaria per l'indirizzo in uscita permette di caricare l'indirizzo delle working zone nei rispettivi registri.
Utilizzando l'output degli `o_hit` dei `wz_enc` il `datapath` può rilevare se un indirizzo faccia parte di una working zone o meno, poi combinando gli output degli `o_diff` può calcolare la codifica completa dell'indirizzo.

3 Risultati sperimentali

Dal report di sintesi si può osservare che il design del componente utilizza in minima parte le risorse disponibili sulla superficie dell'FPGA, non sono presenti black-box e non sono state sintetizzate strutture con un numero di componenti indesiderato.

Site Type	Used	Util%
Slice LUTs*	98	0.07
LUT as Logic	98	0.07
LUT as Memory	0	0.00
Slice Registers	71	0.03
Register as Flip Flop	71	0.03
Register as Latch	0	0.00

Il numero di nodi utilizzato scala linearmente con il numero di working zone e visto l'ampio margine disponibile sarebbe possibile creare un componente

che supporti più working zones con questo stesso design senza incorrere in problemi di eccessiva utilizzazione della superficie dell'FPGA

Ref Name	**Used**	**Functional Category**
FDRE	71	Flop & Latch
LUT2	58	LUT
OBUF	27	IO
LUT4	16	LUT
CARRY4	16	CarryLogic
LUT5	15	LUT
LUT6	13	LUT
IBUF	11	IO
LUT3	7	LUT
LUT1	1	LUT
BUFG	1	Clock

Primitive utilizzate

4 Simulazioni

5 Conclusioni