Progetto di Reti Logiche

Lancini Davide 10675347

Introduzione

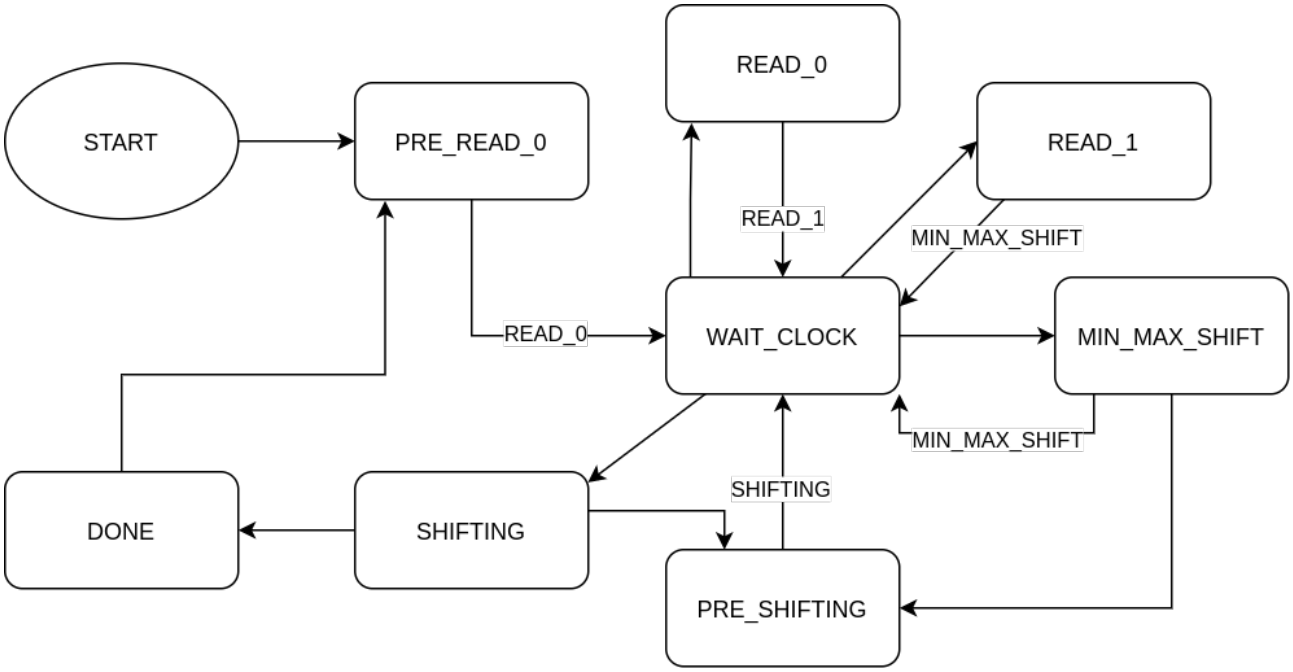
La prima fase del progetto è stata passare ca uno pseudo-codice ad alto livello a uno pseudo codice più dettagliato che permette di scandire gli stati della macchina. Nello pseudo-codice di livello medio sono separate le istruzioni di preparazione alla lettura/scrittura da quelle di lettura poiché l'informazione ha bisogno di un ciclo di clock per essere pronta.

PSEUDO-CODICE DI ALTO LIVELLO	PSEUDO-CODICE DI LIVELLO MEDIO		STATO
Lettura di riga e colonna	Preparazione lettura Riga		1
	Lettura Riga Preparazione lettura Colonna		2
	Lettura Colonna		3
	Calcolo Pixel Totali		
Ricerca Minimo e Massimo	Preparazione Ricerca Minimo e Massimo		
	Ciclo 1	Preparazione lettura Pixel	3 & 4
		Lettura Pixel Confronto Pixel con Minimo Temporaneo	
		Confronto Pixel con Massimo Temporaneo	4
Calcolo dello Shift	Calcolo del Delta		
	Calcolo dello Shift		
Elaborazione dell'Immagine	Preparazione dello Shifting		
	Ciclo 2	Preparazione lettura Pixel	4 & 5
		Lettura Pixel Shifting del Pixel Preparazione scrittura del Nuovo Pixel	6
		Scrittura del nuovo Pixel	

La divisione degli stati della macchina è vista nel dettaglio nella sezione riguardante l'architettura. Le istruzioni utilizzate da 2 stati sono frutto di un'ottimizzazione, anch'essa spiegata nella sezione dell'Architettura. Ci sono inoltre 2 stati aggiuntivi: WAIT_CLOCK (0) e DONE (8).
Le scelte riguardanti il numero e la complessità dei singoli stati sono state fatte cercando di ridurre al minimo il "Total On-Chip Power" stimato da *Vivado 2016.1* utilizzando la parte consigliata *xc7a200tfbg484-1* senza creare un circuito troppo complesso. Il consumo finale totale stimato è di 2.993 W.

Architettura

L’architettura scelta è composta da 8 stati. Gli stati potrebbero essere ridotti ulteriormente, le motivazioni riguardanti le scelte fatte sono analizzate stato per stato.



(0) WAIT_CLOCK

Questo stato (non rappresentato nello pseudo-codice) non fa operazioni logiche, l’unica funzione che ha è utilizzare un ciclo di clock per fare in modo che la lettura abbia il tempo necessario per generare l’input corretto.

(1) PRE_READ_0

Questo è il primo stato della macchina, attende il segnale di start e prepara l’indirizzo della riga da leggere.

(2) READ_0

Questo stato legge la riga dall’indirizzo 0 e ne inserisce il valore nella variabile totalPixel, dopodiché prepara la lettura della colonna nello stesso modo in cui PRE_READ_0 ha preparato quello della riga. L’unione della lettura con la preparazione della lettura successiva è un’ottimizzazione che permette di condensare 2 stati.

(3) READ_1

Questo stato legge la colonna dall’indirizzo 1 e ne moltiplica il valore nella variabile totalPixel, poi inizializza le variabili maximum, che conterrà il valore del pixel più grande dell’immagine, e minimum che conterrà il minimo. Maximum viene inizializzata a 0 e minimum a 255, così da poter far convergere i due valori durante la ricerca. Viene infine preparato il primo pixel da leggere per il calcolo di minimo e massimo.
NOTA: Lo stato READ_0 e READ_1 non sono stati uniti in un unico stato (nonostante siano logicamente molto simili) poiché lo stato così creato sarebbe stato più complesso, avrebbe dovuto avere accesso a una variabile di uscita aggiuntiva e avrebbe reso più complessa l’istruzione di “Preparazione Ricerca Minimo e Massimo” condensata nella READ_1.

(4) MIN_MAX_SHIFT

Questo stato controlla di stare puntando a un pixel dell'immagine di input (dall'indirizzo 2 a totalPixel+1), se sta puntando a un pixel legale legge quel pixel e lo confronta con minimum e poi con maximum e eventualmente aggiorna i valori delle variabili. Una volta confrontato il pixel l'indirizzo di memoria viene avanzato. Se sta puntando a totalPixel + 2 (quindi il primo dell'immagine di output) la ricerca del massimo e del minimo è terminata, quindi lo stato procede a calcolare lo Shift e a preparare gli indirizzi per lo Shifting dei pixel.
NOTA 1: Questo stato conduce a se stesso finché non ha analizzato tutti i pixel, se l'immagine ha 0 pixel il massimo e il minimo non vengono mai aggiornati.
NOTA 2: Per lo Shifting dei pixel vengono preparati 2 indirizzi, uno punta al primo pixel dell'immagine di input e uno al primo pixel dell'immagine di output. L'utilizzo di un indirizzo unico è possibile, al posto di usare 2 registri si userebbe 1 registro e 2 componenti per sommare e sottrarre totalPixel quando necessario. È stato favorito l'uso di 2 registri per ridurre il consumo elettrico stimato di 0.008W, senza considerare il costo di realizzazione delle due versioni. Le due versioni sono identiche in velocità in tutti i casi di test provati.

(5) PRE_SHIFTING

La memoria viene impostata in lettura (la prima volta non è strettamente necessario) e i due indirizzi usati dallo Shifting vengono avanzati, preparando così la lettura del pixel successivo.

(6) SHIFTING

Lo stato di Shifting è lo stato più complesso ed è lo stato che ha subito più ottimizzazioni durante la realizzazione del progetto. Se *address* sta puntando all'immagine di input (come MIN_MAX_SHIFT) lo Shifting non è finito. Viene letto il pixel e viene inserito negli 8 bit meno significativi di un registro temporaneo a 9 bit (il nono bit è 0). A questo punto si entra in un loop che dura "Shift" iterazioni. A ogni iterazione viene controllato il bit 8, se esso è un 1 il pixel sforerebbe lo Shifting e il pixel di output viene impostato a 255, se il pixel finisce lo Shifting senza sforare viene scritto nell'indirizzo di uscita.

NOTA: è possibile utilizzare un registro a 16 bit e fare lo Shifting completo controllando solo alla fine se il pixel ha sforato 255. Questa soluzione utilizzerebbe un registro più grande in funzione di un circuito lievemente più semplice. È stato favorito l'utilizzo di un registro a 9 bit poiché il consumo totale stimato del componente aumenterebbe di più di 1W. Le due versioni sono identiche in velocità in tutti i casi di test provati.

(7) DONE

Questo stato imposta il segnale *done* a 1 e attende che lo *start* venga riportato a 0. Una volta portato a 0 l'esecuzione ritorna a PRE_READ_0 e il componente è pronto per essere riutilizzato.

I Registri

- nxt è un registro che memorizza lo stato successivo della macchina dopo WAIT_CLOCK. Viene impostato prima di entrare in WAIT_CLOCK.
- totalPixel è un intero (a 15 bit) e memorizza il numero di pixel dell'immagine i valori legali vanno da 0 a 16384 (128x128).
- minimum è un registro a 8 bit che memorizza il pixel più piccolo.
- maximum è un registro a 8 bit che memorizza il pixel più grande.

- delta è un intero a 8 bit che memorizza maximum-minimum. Potrebbe essere ricalcolato ogni volta ma il progetto è stato ottimizzato in funzione del consumo elettrico ignorando i costi del registro.
- shift è un intero a 4 bit, i valori legali vanno da 0 a 8, 9-15 sono logicamente impossibili
- i è un intero a 4 bit, i valori legali vanno da 0 a Shift(8) e viene utilizzato per iterare in SHIFTING
- exceed è un booleano usato in SHIFTING per indicare che il pixel ha superato 255
- address è un registro a 16 bit che punta ai pixel dell'immagine di input
- addressShifted è un registro a 16 bit che punta ai pixel dell'immagine di output, quando è in uso è sempre totalPixel più avanti rispetto a address
- temp è un registro a 9 bit e viene usato in SHIFTING per memorizzare il pixel

Risultati Sperimentali

Utilization Post-Syntesis: 19%

Utilization Post-Implementation: 19%

Total On-Chip Power: 2.993 W

Junction Temperature: 32.4 °C

Schematic Cells: 232

Schematics I/O Ports: 38

Schematics Nets: 639

Simulazioni

Prima di indicare i test fatti è opportuno capire quali parti del componente si possono stressare:

- il segnale di reset
- i pixel totali dell’immagine
- il delta dei pixel

I casi limite del reset sono: 0, 1, 2+

I casi limite del numero di pixel sono: 0, 1, 16384

I casi limite del delta sono ai confini dei range di Shifting: 0, 1, 2, 3, 6, 7, 14, 15, 30, 31, 62, 63, 126, 127, 254, 255

I test forniti non coprono tutti questi casi, ogni caso limite è stato testato singolarmente (22 test) e in combinazione tra di loro (48 test). Oltre ai test forniti e ai casi limite sono stati fatti test generati casualmente con uno script in python. I test casuali sono stati fatti tutti con un reset e una dimensione fissa di 6 pixel , poiché essendo valori aleatori se la dimensione è troppo grande le probabilità di delta si avvicinano sempre a 255 (ci sarà quasi sempre un pixel molto piccolo e uno molto grande).

NOTA: i test che hanno 0 o 1 pixel non hanno un delta da calcolare, i casi di test si riducono da 144 a 48.

Conclusioni

Il progetto si è dimostrato abbastanza lineare nonostante qualche problema con il linguaggio e con il software. Quella presentata è la versione del progetto con il consumo minore che sono riuscito a ottenere.