POLITECNICO
MILANO 1863

Relazione della prova finale di Reti Logiche

Alexandru Gabriel Bradatan
Codice persona: 10658858
Docente di riferimento: Prof. Fornaciari
Data: 5 giugno 2022

a.a 2021-2022

Indice

1 Introduzione **2**

2 Architettura **2**
2.1 Datapath 4
2.1.1 Elaborazione di un byte 4
2.1.2 Gestione della memoria 5
2.1.3 Generazione del segnale di fine elaborazione 6
2.2 Macchina a stati 6

3 Test effettuati **7**

4 Report di sintesi **7**

5 Conclusione **9**

1 Introduzione

Il progetto consiste nella realizzazione di un componente hardware descritto in VHDL che dato uno stream di byte in input gli applica il codice convoluzionale 1/2 e restituisce uno stream in output.

Modello di memoria Il modulo si interfaccerà con una memoria RAM con 16 bit di address space, indirizzata al byte. La memoria, non implementata all'interno del componente, comunicherà tramite il seguente protocollo:

- Un segnale di enable `mem_en` attiverà la memoria.
- La modalità di accesso verrà determinata dal segnale `mem_we`: 0 per indicare la lettura e 1 la scrittura.
- L'indirizzo di lettura/scrittura viene impostato tramite il il vettore `mem_address`.
- Nel caso di una scrittura, verranno salvati i dati presenti in `mem_in`; nel caso di una lettura, il risultato dell'operazione sarà disponibile in `mem_out`.

I dati saranno distribuiti in RAM seguendo la seguente convenzione:

- All'indirizzo 0 sarà salvata la lunghezza totale in byte dello stream in ingresso. Ciò implica che la lunghezza massima dell'input sarà di 255 byte.
- Nel range di indirizzi $[1; 255]$ sarà salvato l'input stream
- Nel range di indirizzi $[1000; 65535]$ sarà salvato l'output dell'elaborazione.

2 Architettura

Il componente è suddiviso in due entità: il datapath e la macchina a stati che governa l'elaborazione.

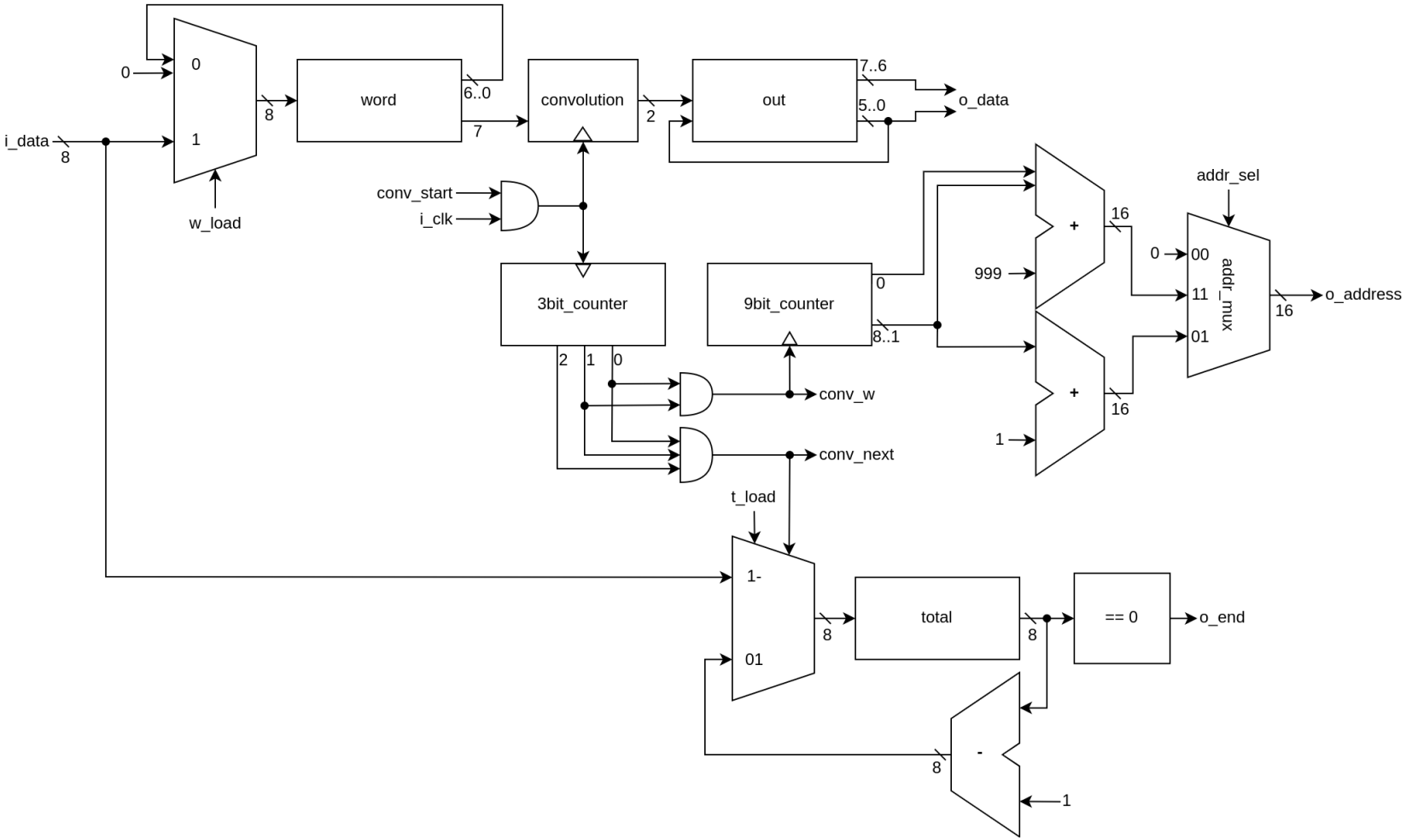

Figura 1: Schematica completa del datapath

2.1 Datapath

Il datapath è suddiviso in 3 gruppi funzionali interconnessi che si occupano ognuno di un aspetto dell'elaborazione: elaborazione di un byte, generazione del segnale di fine elaborazione dell'input e generazione dei segnali che gestiscono l'accesso alla memoria. Lo schema completo del datapath è riportato in figura 1, ciascun modulo verrà affrontato nel dettaglio nella sua sezione corrispondente.

Nel suo complesso, il datapath esporrà verso la macchina a stati che lo governa la seguente interfaccia:

- `i_clk` (in): Segnale di clock (passthrough dalla macchina a stati).
- `i_rst` (in): Segnale di reset (passthrough dalla macchina a stati).
- `i_data` (in): Byte letto dalla RAM (passthrough dalla macchina a stati).
- `addr_sel` (in): Vettore di 2 bit di selezione tra 3 indirizzi. Il valore viene così interpretato:
  - `00` si riferisce all'indirizzo 0
  - `01` si riferisce all'indirizzo di un byte nel range $[1; input\_length]$ con $input\_length$ la lunghezza in byte dello stream in input
  - `11` si riferisce all'indirizzo di un byte nel range $[999; 999 + 2input\_length]$.

  Gli indirizzi prodotti tramite `01` e `11` corrispondono rispettivamente al byte corrente in lettura e a quello corrente in scrittura.
- `t_load` (in): Segnale di enable per il registro che salverà la lunghezza totale dello stream in input.
- `w_load` (in): Segnale di enable per il registro che salverà la parola fornita in ingresso tramite `i_data`.
- `conv_start` (in): Segnale di enable per il processo di elaborazione di un byte.
- `conv_rst` (in): Segnale di reset secondario generato dalla macchina a stati.
- `conv_w` (out): Segnale che indica alla macchina a stati che un byte è pronto per la scrittura in RAM.
- `conv_next` (out): Segnale che indica alla macchina a stati che l'elaborazione di un byte è terminata.
- `o_address` (out): Vettore di 16 bit che contiene l'indirizzo generato tramite `addr_sel`.
- `o_end` (out): Segnale che indica la fine dell'elaborazione dello stream in input.
- `o_data` (out): Byte da scrivere in RAM.

Ogni componente sequenziale del datapath, ad esclusione di dove indicato diversamente, riceverà in ingresso come clock `i_clk` e come segnale di reset `i_rst` + `conv_rst`.

2.1.1 Elaborazione di un byte

Il processo di convoluzione può essere visto come una macchina a stati che prende in ingresso un bit dello stream in entrata e ne produce due in uscita. La macchina a stati è rappresentata in figura 2. Lo stream in uscita avrà, quindi, il doppio della lunghezza rispetto allo stream in entrata.

La rete di elaborazione è invece rappresentata in figura 3. Il byte letto dalla RAM viene salvato in uno shift register R-to-L a 8 bit governato dal segnale `w_load`. Ad ogni ciclo di clock, il MSB viene fornito alla macchina a stato di convoluzione. L'output della convoluzione viene salvato nei 2 bit più bassi di un altro shift register R-to-L a 8 bit (per convenzione ordiniamo i bit in ordine discendente da sinistra a destra). Il byte da scrivere in RAM `o_data` è composto dalla lettura dei bit in parallelo dell'ultimo shift register. La macchina a stati di convoluzione riceve come clock `conv_start` × `i_clk`.

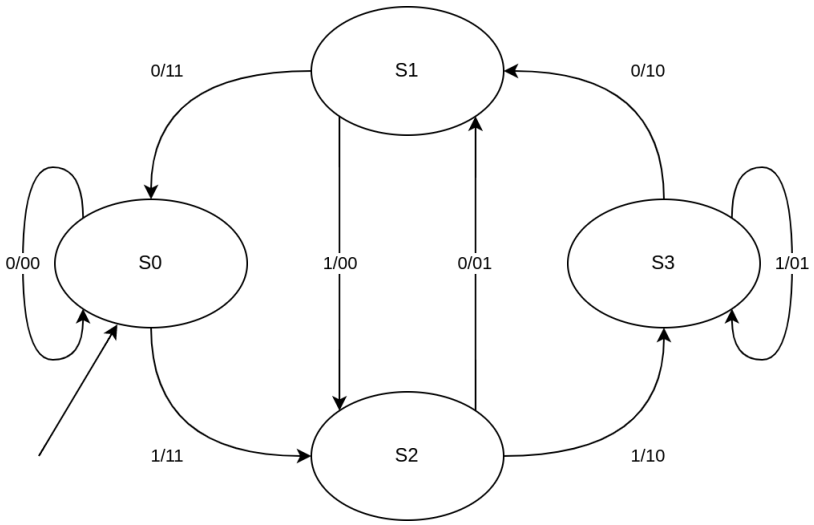

Figura 2: Macchina di Mealey rappresentante il processo di convoluzione di un byte

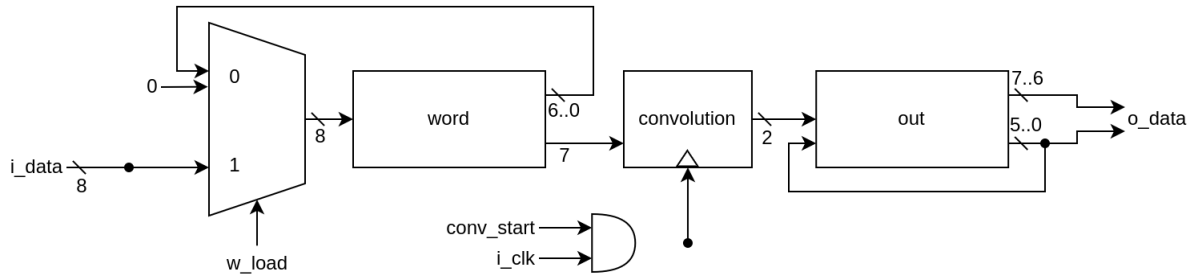

Figura 3: Modulo di elaborazione di un byte

2.1.2 Gestione della memoria

Il modulo di gestione della memoria è responsabile della generazione dei segnali `o_address`, `conv_w` e `conv_next`. La schematica del modulo è riportata in figura 4

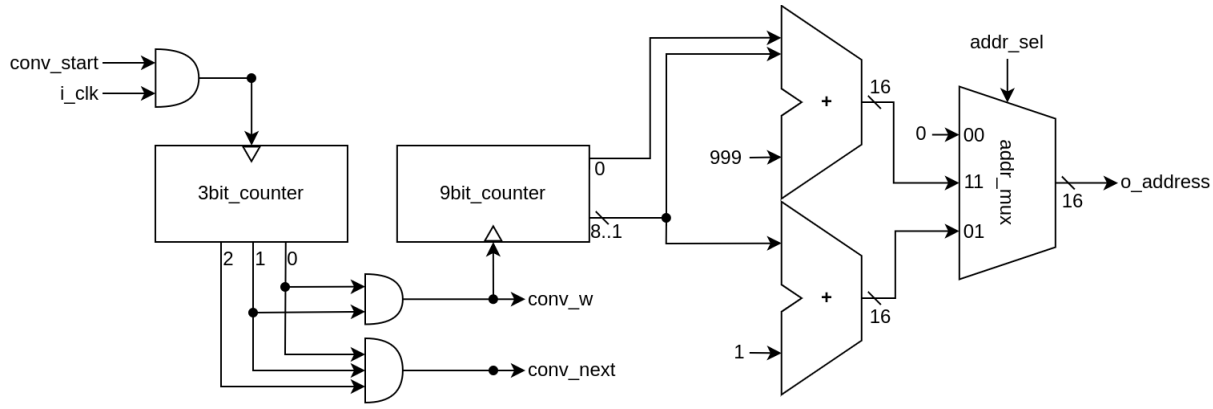

Figura 4: Modulo di gestione della memoria

Il modulo utilizza due contatori, uno a 3 bit e uno a 9 bit. Il primo viene incrementato per ogni ciclo di clock in cui la macchina di convoluzione lavora. Esso conta il numero di coppie prodotte:

- Per 4 coppie (3) il segnale `conv_w` viene alzato
- Per 8 coppie (7) il segnale `conv_next` viene alzato

Il secondo contatore conta invece i byte scritti in memoria. Poiché i dati in uscita hanno il doppio della lunghezza di quelli in entrata, possiamo riutilizzare lo stesso contatore per gestire sia l'indirizzo di lettura che quello di scrittura:

- Tutti i 9 bit vengono utilizzati per calcolare l'indirizzo di scrittura
- Gli 8 bit più significativi vengono utilizzati per calcolare l'indirizzo di lettura

Il valore del secondo contatore viene infine sommato a degli offset e passato al multiplexer di selezione dell'indirizzo.

2.1.3 Generazione del segnale di fine elaborazione

Il modulo di generazione del segnale di fine elaborazione genera il segnale `o_end`. La schematica è rappresentata in figura 5.

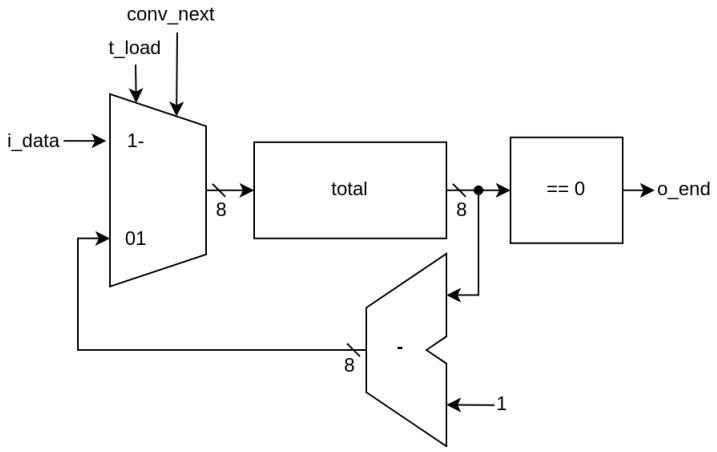

Figura 5: Modulo di generazione del segnale di fine

Esso è composto da un contatore con preload in cui viene caricato ad inizio elaborazione la lunghezza dello stream in entrata. Il valore salvato viene decrementato ogni volta che `conv_next` è alto. Il segnale di `o_end` viene impostato a 1 quando il valore salvato all'interno del contatore è pari a 0.

2.2 Macchina a stati

La macchina a stati che governa il datapath è una macchina di Mealey che espone ai morsetti i seguenti segnali:

- `i_clk` (in): Segnale di clock.
- `i_rst` (in): Segnale di reset.
- `i_start` (in): Segnale di inizio elaborazione
- `i_data` (in): Byte letto dalla RAM.
- `o_address` (out): Vettore di 16 bit che contiene l'indirizzo di lettura/scrittura in RAM (passthrough da datapath).
- `o_done` (out): Segnale di fine elaborazione
- `o_en` (out): Segnale di enable della RAM
- `o_we` (out): Segnale write-enable della RAM

- `o_data` (out): Byte da scrivere in RAM (passthrough da datapath).

Una diagramma ad alto livello della macchina a stati è presente in figura 6. Il diagramma completo invece è riportato in figura 7.

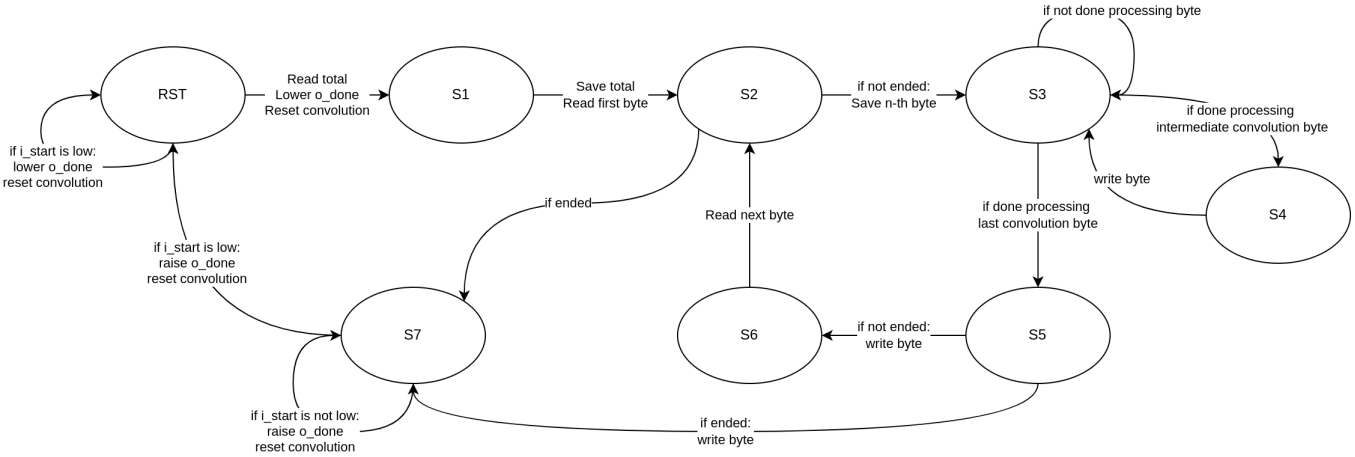

Figura 6: Macchina di Mealey che controlla il datapath (semplificata)

3 Test effettuati

Sono stati usati due testbench, ognuno dei due in grado di leggere ed eseguire i casi di test da file. La differenza tra i due moduli sta nella modalità di reset tra vari test-case: uno usa la funzionalità di restart che il circuito deve implementare come per specifica, l'altro usa il segnale di RESET `i_rst`.

I casi limite testati sono stati:

1. Lunghezza dell'input pari a 0
2. Lunghezza dell'input diversa da 0
3. Restart/reset tra sequenze diverse
4. Restart/reset tra sequenze uguali

Tutti i test sono stati eseguiti in modalità *behavioural* e *functional post-synthesis*. I file letti dai testbench sono stati scritti tramite un generatore di test scritto in python dal sottoscritto.

4 Report di sintesi

Nelle tabelle 1 e 2 sono stati riportati i dati di sintesi più importati. È stato utilizzato Vivado 2021.2 come software di sintesi e xc7a200tfbg484-1 come FPGA di riferimento.

Site type	Used	Available	Utilization%
LUT as Logic	53	134600	0.04
LUT as Memory	0	134600	0.00
Register as Flip Flop	41	269200	0.02
Register as Latch	0	269200	0.00
F7 Muxes	0	67300	0.00
F8 Muxes	0	33650	0.00

Tabella 1: Alcune metriche del report di utilizzo come riportato da `report_utilization`

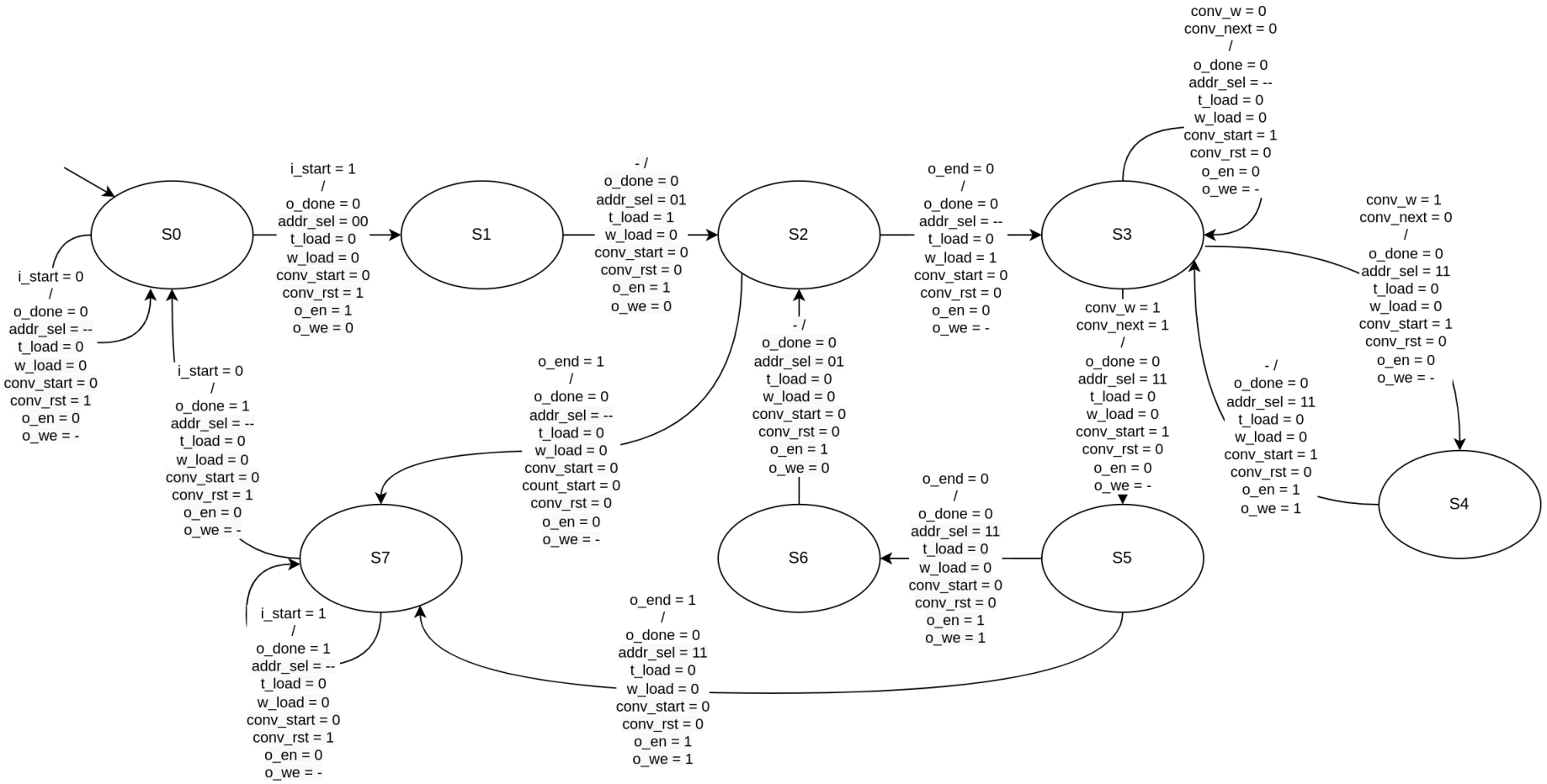

Figura 7: Macchina di Mealey che controlla il datapath (completa)

Slack (MET)	`90.910ns (required time - arrival time)`
Requirement	`100.000ns (clock rise@100.000ns - clock rise@0.000ns)`
Data Path Delay	`2.939ns (logic 0.999ns (33.991%) route 1.940ns (66.009%))`

Tabella 2: Alcune metriche del report di timing come riportato da `report_timing`

5 Conclusione

Si ritiene che il componente sviluppato rispetti tutti i vincoli funzionali e non indicati nel documento di specifica.

L'architettura è stata sviluppata secondo il modello a due componenti FSM + Datapath. Particolare attenzione è stata posta sull'efficienza: si è riuscito ad ottenere uno slack di circa 90 nanosecondi e un utilizzo di soli 41 flip flop. Si è inoltre evitato l'utilizzo di Latch. Alcuni dei componenti sequenziali più complessi, come i contatori e la macchina a stati del convolutore, sono stati specificati usando `process` per dare piena libertà al tool di sintesi di implementarli nella maniera più efficiente.