Prova finale di Reti Logiche

a.a. 2022/23

Prof. Fabio Salice

POLITECNICO

MILANO 1863

Progetto a cura di

Irene Lo Presti e Matteo Lussana

Ingegneria informatica - corso di laurea triennale

Introduzione

La memoria RAM (Random Access Memory) è la memoria centrale del calcolatore, viene rappresentata come una sequenza di celle identificate da una sequenza di bit detta indirizzo di memoria.
La dimensione della cella dipende dal calcolatore utilizzato (solitamente è di 8 bit), così come la dimensione dell'indirizzo (ad oggi può essere di 32 bit o 64 bit).
Per accedere alle varie celle, ovvero per scriverne o leggerne il contenuto, a livello di programmazione software (per esempio nel linguaggio C) ci si avvale dei puntatori.
Un puntatore è un tipo di variabile che contiene l'indirizzo di una cella di memoria e nel linguaggio C si può accedere al contenuto di tale cella tramite l'operatore di dereferenziazione (*).

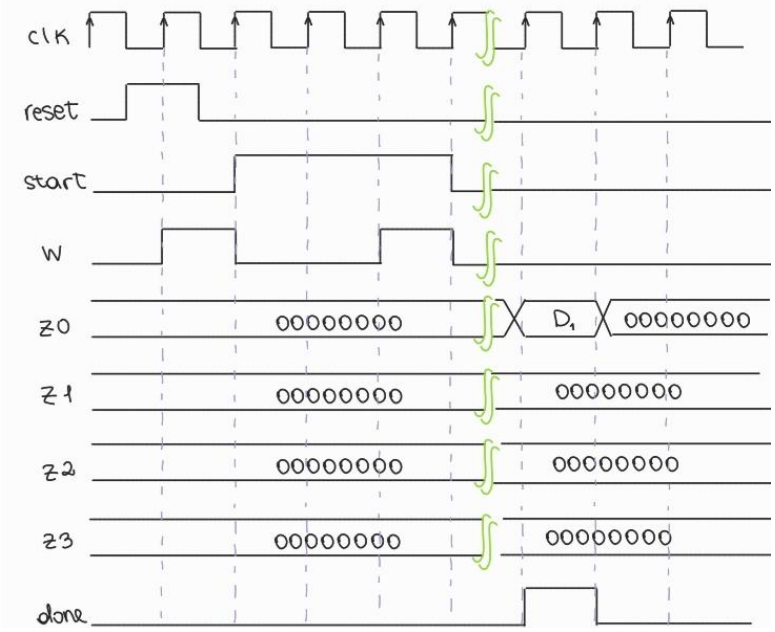
Esempio

```c
#include <stdio.h>

int main(){
   int var1 = 4;
   int var2;
   int *pointer;
   pointer = &var1; //pointer contiene l'indirizzo di var1

   var2 = *pointer; //operazione di differenziazione
                    //var2 contiene il valore 4
}
```

Il modulo hardware da implementare deve svolgere la funzione di deferenziazione, in particolare il sistema riceve in ingresso informazioni rispetto al canale di uscita su cui bisogna mostrare il contenuto della cella di memoria e l'indirizzo di tale cella. Questi dati sono ottenuti come sequenze sul segnale di ingresso W da 1 bit. In particolare i primi 2 bit ricevuti in sequenza indicano il canale d'uscita, mentre i successivi sono destinati all'indirizzo che deve essere di 16 bit (se i bit letti sono meno, la sequenza deve essere estesa con 0 sui bit più significativi).
Il sistema, inoltre, dispone di un secondo segnale di ingresso da 1 bit: START, quando alto rende valida la sequenza in ingresso su W, quando basso la lettura su W è terminata. START rimane alto per almeno 2 cicli di clock e non più di 18 cicli di clock.
Il modulo presenta 5 uscite: Z0, Z1, Z2, Z3 (ovvero i canali di uscita da 8 bit) e DONE (segnale da 1 bit alto in corrispondenza della scrittura sui canali per un solo ciclo di clock).
Le uscite sono inizialmente a "0000 0000" e così rimangono quando il segnale di DONE è basso, mentre mostrano l'ultimo valore da loro stampato quando DONE è alto.

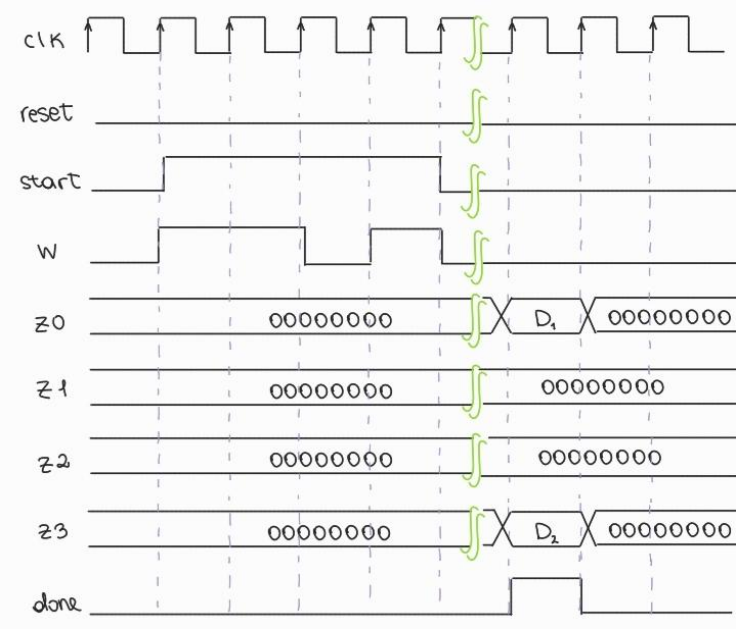
Il sistema ha un unico segnale di RESET, dopo la sua attivazione le uscite devono essere a 0000 0000 e il segna di DONE a 0;

Esempio 1:



- START rimane alto per tre cicli di clock;
- il canale di uscita è Z0 perché i primi due bit di W sono “00”;
- l’indirizzo di memoria è “0000000000000001” perché è stato necessario estendere aggiungendo 15 zeri.

Esempio 2:

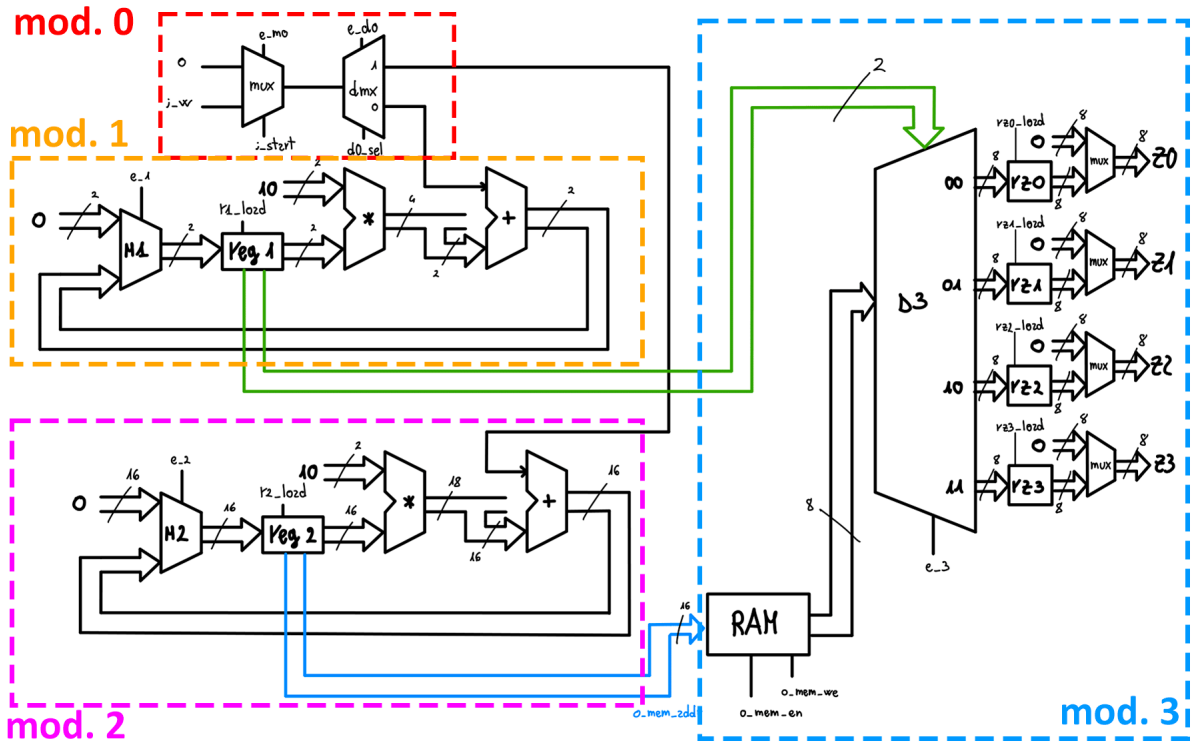


- START rimane alto per 4 cicli di clock;
- il canale di uscita è Z3 perché i primi due bit di W sono “11”;

- l'indirizzo di memoria è "0000000000000001" perché è stato necessario estendere aggiungendo 14 zeri;
- quando DONE è alto, Z0 mostra il valore letto da memoria precedentemente, Z1 e Z2 restano ancora a zero, Z3 mostra il nuovo valore letto da memoria.

Architettura

Datapath



Mod. 0: modulo designato alla ricezione del segnale di ingresso, incaricato di smistare il segnale di ingresso al mod. 1 (se sono i primi due bit) o al mod. 2.

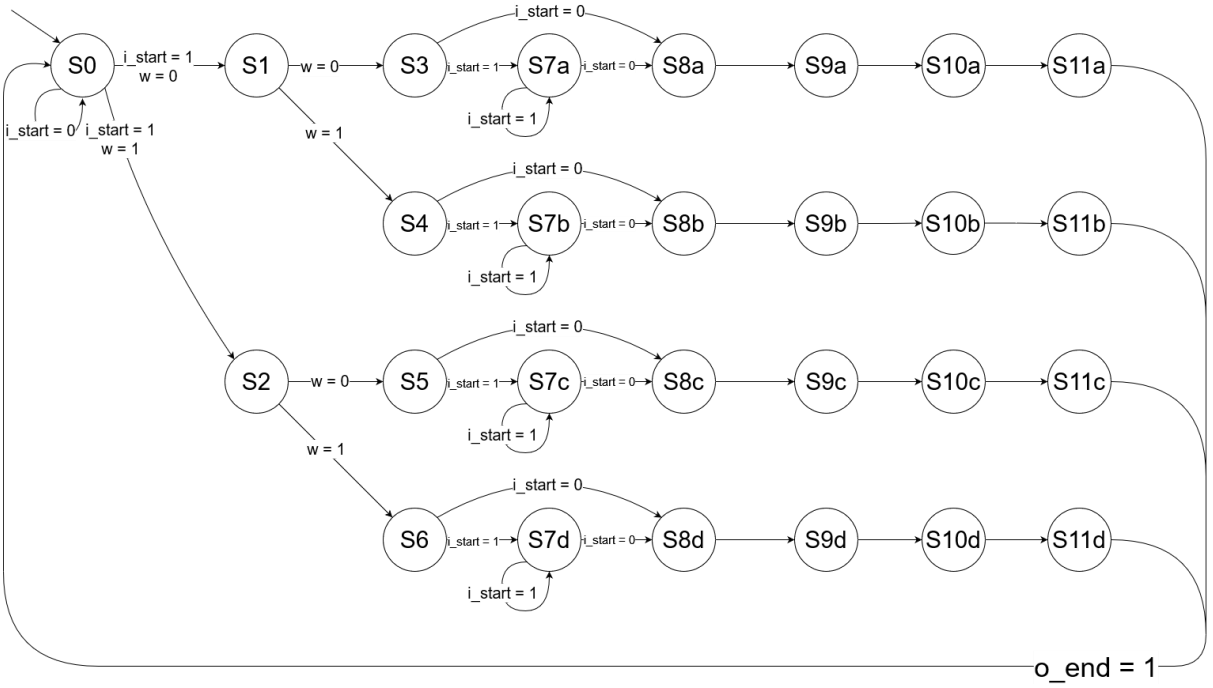
Mod. 1: modulo per l'accorpamento dei due bit di identificazione del canale di uscita. L'accorpamento viene fatto nel seguente modo: si parte dal codice binario 00 a cui viene sommato il primo bit in ingresso, il nuovo codice verrà moltiplicato per 10 in modo tale che il primo bit in ingresso subisca uno shift verso sinistra. Il secondo bit in ingresso viene sommato al codice che verrà salvato nel registro 1.

Mod. 2: modulo che si occupa della composizione dell'indirizzo di memoria, la composizione avviene esattamente come nella composizione per identificare il canale di uscita nel mod. 2.

Mod. 3: modulo che serve per visualizzare il valore letto dalla memoria nel giusto canale di lettura. Esso è formato da un demultiplexer D3 per indirizzare il valore letto nel corretto registro (infatti il selettore è l'uscita del registro 1). Successivamente ad

ogni registro si trova un multiplexer il cui selettore è il segnale o_done: se tale segnale è alto allora viene mostrato sui canali di uscita il contenuto dei registri corrispondenti, altrimenti viene mostrato “00000000”.

Macchina a stati



Nota: Per alleggerire il disegno e semplificarne la lettura non è stato rappresentato che se il segnale i_rst si alza, indipendentemente dallo stato corrente, si torna allo stato iniziale S0.

S0 (start): è lo stato iniziale e di reset, nel quale si rimane finché i_start è basso.

S1, S2, S3, S4, S5, S6: stati per la codifica del canale di uscita, al fine di permettere unicamente il caricamento del registro corretto. I primi due stati sono per il primo bit mentre gli altri sono per il secondo.

S7 (a,b,c,d): stato per la lettura dell’indirizzo di memoria.

S8 (a,b,c,d): è lo stato in cui si inizia a leggere il contenuto della memoria perciò viene attivato il segnale i_mem_en.

S9 (a,b,c,d): stato successivo alla lettura che serve per attivare il demultiplexer D3 in modo tale che il valore letto venga inserito nel registro corrispondente al canale scelto.

S10 (a,b,c,d): in questo stato viene caricato il contenuto della memoria del registro designato.

S11 (a,b,c,d): ultimo stato che serve per alzare il segnale di done, il quale verrà abbassato dopo un ciclo di clock nello stato S0.

Risultati sperimentali

Sintesi

Utilization Design Information

1. Slice Logic

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	76	0	134600	0.06
LUT as Logic	76	0	134600	0.06
LUT as Memory	0	0	46200	0.00
Slice Registers	105	0	269200	0.04
Register as Flip Flop	105	0	269200	0.04
Register as Latch	0	0	269200	0.00
F7 Muxes	3	0	67300	<0.01
F8 Muxes	0	0	33650	0.00

Inizialmente il progetto contava quasi il doppio dei registri attualmente presenti, di cui molti latch. Dopo un'attenta analisi, si è riuscito a ridurre il numero dei registri ed a eliminare tutti i latch grazie all'inizializzazione di tutti i segnali della macchina a stati e ad una corretta compilazione della sensitivity list dei process.

Timing report:

```
Slack (MET) :             46.850ns  (required time - arrival time)
  Source:                 cur_state_reg[0]/C
                            (rising edge-triggered cell FDCE clocked by clock  {rise@0.000ns fall@50.000ns period=100.000ns})
  Destination:            DATAPATH0/o_mem_addr_reg[0]/R
                            (rising edge-triggered cell FDRE clocked by clock'  {rise@0.000ns fall@50.000ns period=100.000ns})
  Path Group:             clock
  Path Type:              Setup (Max at Slow Process Corner)
  Requirement:            50.000ns  (clock fall@50.000ns - clock rise@0.000ns)
  Data Path Delay:        3.109ns  (logic 0.875ns (28.144%)  route 2.234ns (71.856%))
  Logic Levels:           2  (LUT2=1 LUT5=1)
```

Il periodo di clock non utilizzato è di 46.850 ns, questo significa che si potrebbe avere un periodo di clock nettamente inferiore e il sistema funzionerebbe ugualmente.

Simulazioni

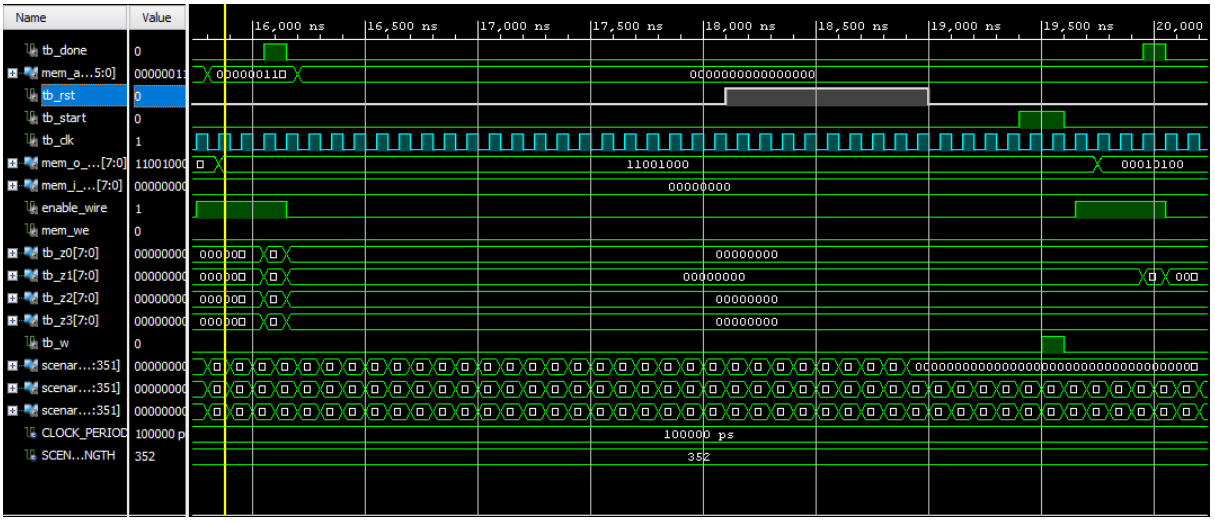
Test bench 1

Risultati:

- launch_simulation: Time (s): cpu = 00:00:02 ; elapsed = 00:00:19 . Memory (MB): peak = 756.016 ; gain = 0.000
- test passato
- Time: 32500 ns Iteration: 0

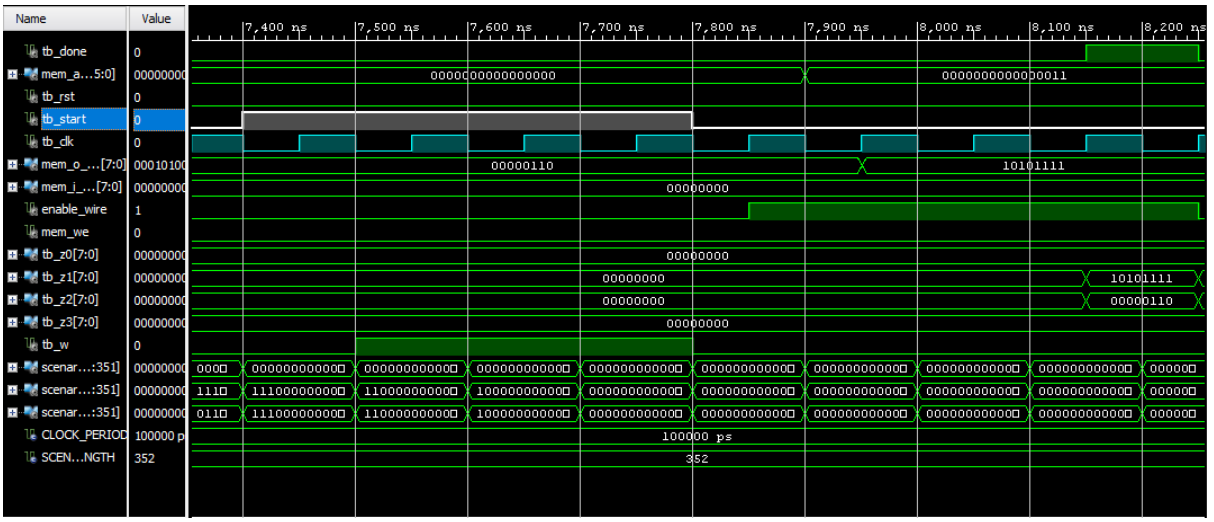
Il test bench 1 controlla la risposta del circuito ai segnali di reset, start e done. In particolare:

1. Dopo il segnale di reset tutte le uscite devono essere a zero.

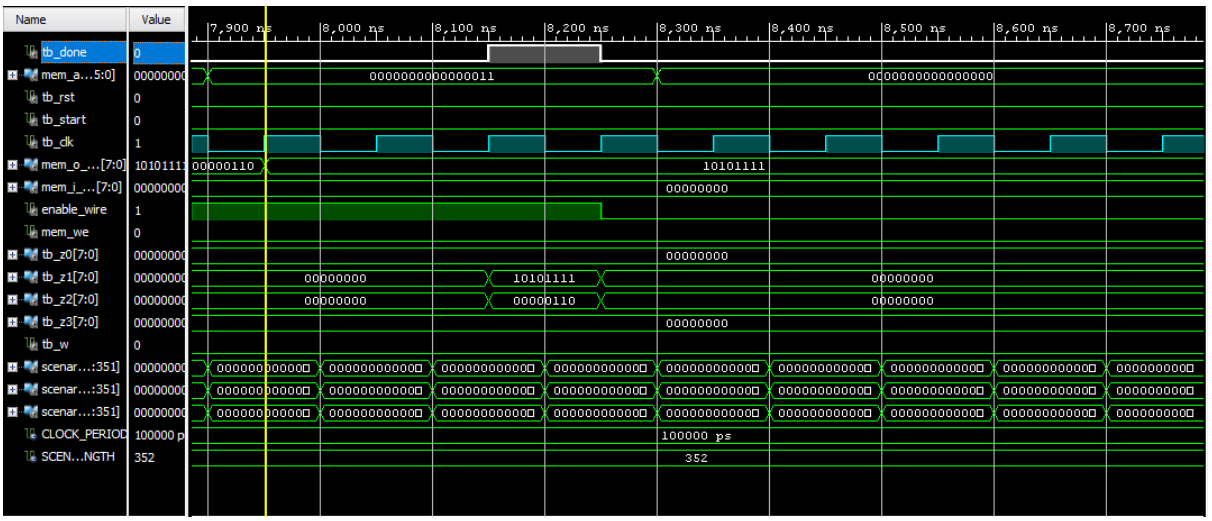


In questo esempio si può notare che, prima del segnale di reset, tutte le uscite contenevano un valore letto da memoria diverso da zero. Dopo la lettura dell'input avvenuta successivamente al segnale di reset, solo l'uscita Z1 contiene il valore letto da memoria, mentre le altre sono tutte a zero.

2. Durante la lettura del segnale d'ingresso (quando i_start = '1') , tutte le uscite devono essere a zero:

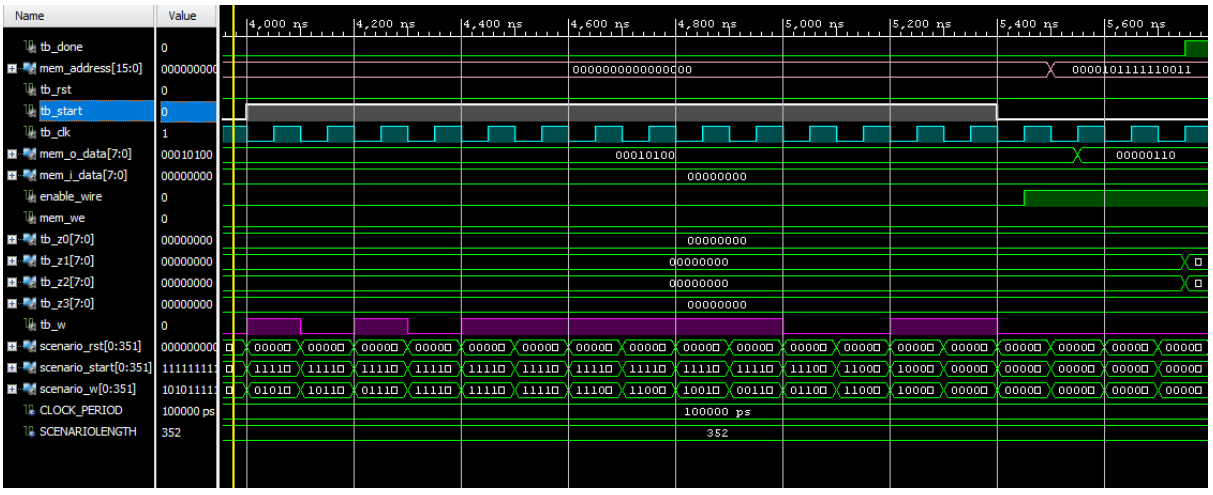


3. Quando il segnale o_done è alto, solo la corretta uscita deve contenere il valore letto da memoria mentre le altre devono continuare a mostrare il valore stampato al segnale di done precedente (ovvero un valore letto da memoria oppure zero se non è mai stato stampato nulla):



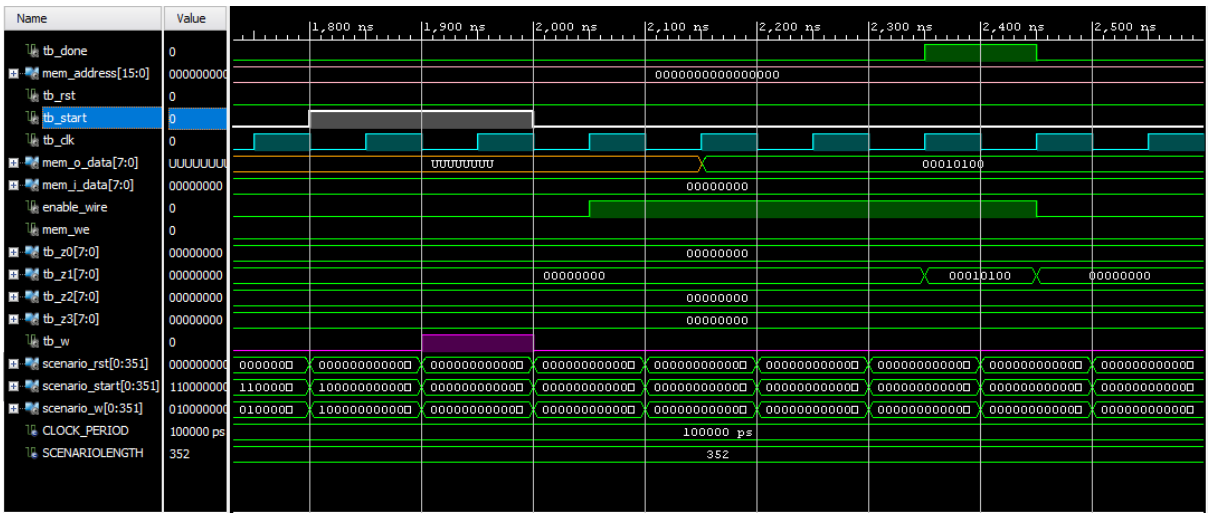
In questo esempio si può notare che il segnale di done è alto per un ciclo di clock, durante il quale vengono mostrati due valori letti da memoria sulle uscite Z1 e Z2. Il valore di Z1 è stato appena letto mentre quello di Z2 è stato conservato dalla lettura precedente. Le uscite Z0 e Z3 mostrano tutti zeri perché non hanno ancora stampato alcun valore della memoria.
Quando o_done è basso, le uscite devono essere tutte a zero, si può notare anche questo particolare nell'esempio precedente.

Questo test bench controlla, inoltre, il corretto funzionamento dell'estensione dell'indirizzo della memoria a sedici bit, per esempio nel seguente caso:



Qui si può notare che il segnale i_start rimane alto per 14 cicli di clock. I primi 2 sono riservati alla codifica dell'uscita quindi i restanti 12 servono per l'indirizzo della memoria. I 12 bit letti devono essere estesi a 16 tramite l'aggiunta di 4 zeri alla sinistra del codice.

Corner case: il segnale start rimane alto per due cicli di clock, quindi viene letto solo il codice corrispondente all'uscita, dando zero bit per l'indirizzo di memoria il quale sarà "0000000000000000".

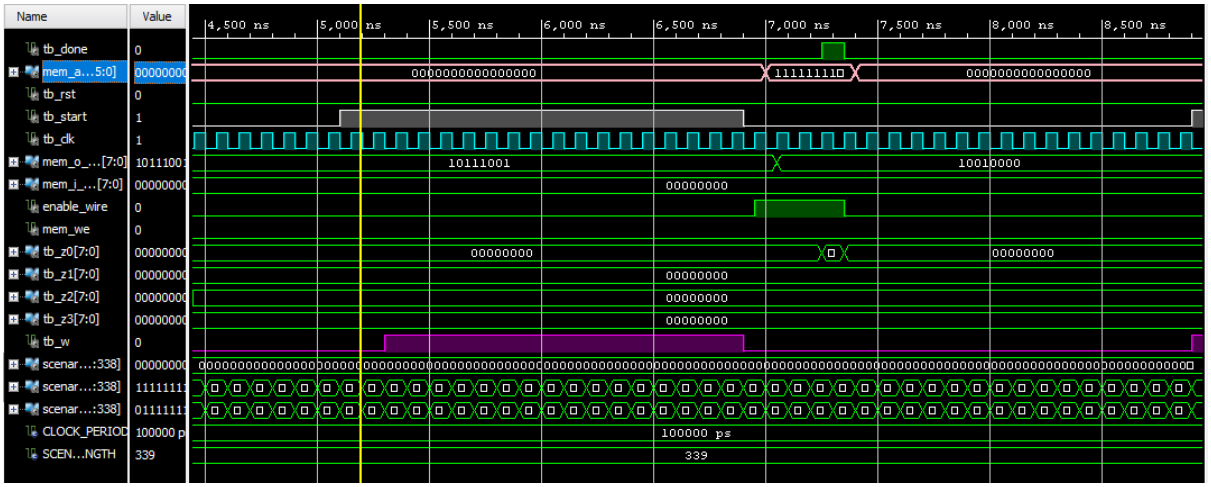


Test bench 2
Risultati:

- launch_simulation: Time (s): cpu = 00:00:01 ; elapsed = 00:00:11 . Memory (MB): peak = 855.043 ; gain = 0.000
- test passato
- Time: 32500 ns Iteration: 0

Il test bench 2 controlla che il circuito si comporti in maniera corretta rispetto alle regole base: così come nel test bench 1 si controlla il funzionamento del circuito in relazione ai segnali di reset, start e done, e se l'indirizzo di memoria viene esteso in maniera corretta.

Corner case: il segnale di start rimane alto per il massimo numero di cicli di clock (ovvero 18).



Di questi 18 cicli di clock, i primi due servono, come sempre, alla codifica dell'uscita, mentre gli altri 16 servono per l'indirizzo della memoria. Quest'ultimo viene quindi interamente letto da input e di conseguenza non è necessario estendere il segno.

Test bench 3
Risultati:

- launch_simulation: Time (s): cpu = 00:00:03 ; elapsed = 00:00:14 . Memory (MB): peak = 812.195 ; gain = 0.000
- test passato
- Time: 58 us Iteration: 0

Il test bench 3 controlla il funzionamento corretto del circuito, facendo riferimento alle regole imposte dalla specifica, non aggiunge casi particolari rispetto ai primi due test bench, l'unica differenza è la quantità superiore di letture della memoria.

Test bench 4
Risultati:

- launch_simulation: Time (s): cpu = 00:00:01 ; elapsed = 00:00:10 . Memory (MB): peak = 855.043 ; gain = 0.000
- test passato
- Time: 12600 ns Iteration: 0

Il test bench 4 testa il funzionamento dei segnali di reset, start e done. La sua particolarità è che stampa i valori della memoria unicamente sull'uscita Z2, lasciando le altre sempre a zero.

Test bench 5
Risultati:

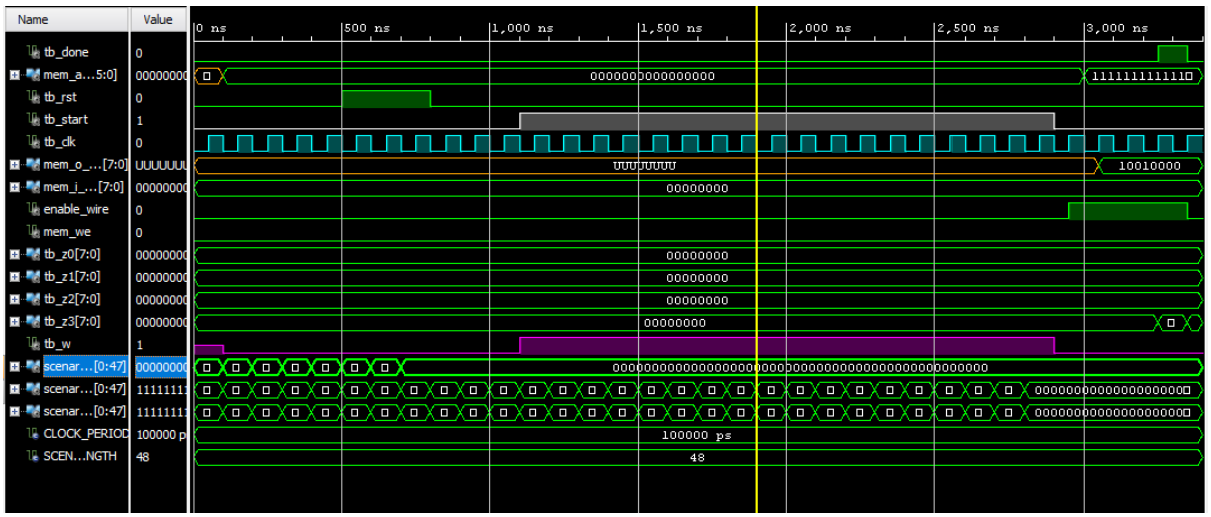
- launch_simulation: Time (s): cpu = 00:00:01 ; elapsed = 00:00:12 . Memory (MB): peak = 855.043 ; gain = 0.000
- test passato
- Time: 62400 ns Iteration: 0

Il test bench 5 testa il corretto funzionamento del circuito senza aggiungere particolarità rispetto agli altri test bench.

Test bench 6
Risultati:

- launch_simulation: Time (s): cpu = 00:00:02 ; elapsed = 00:00:13 . Memory (MB): peak = 882.898 ; gain = 0.000
- test passato
- Time: 3400 ns Iteration: 0

Il test bench 6 fa un unico test: dopo il reset, il segnale di start rimane alto per 18 cicli di clock durante i quali l'ingresso i_w rimane sempre alto. Per cui l'uscita sarà Z3 mentre l'indirizzo di memoria sarà "1111111111111111".

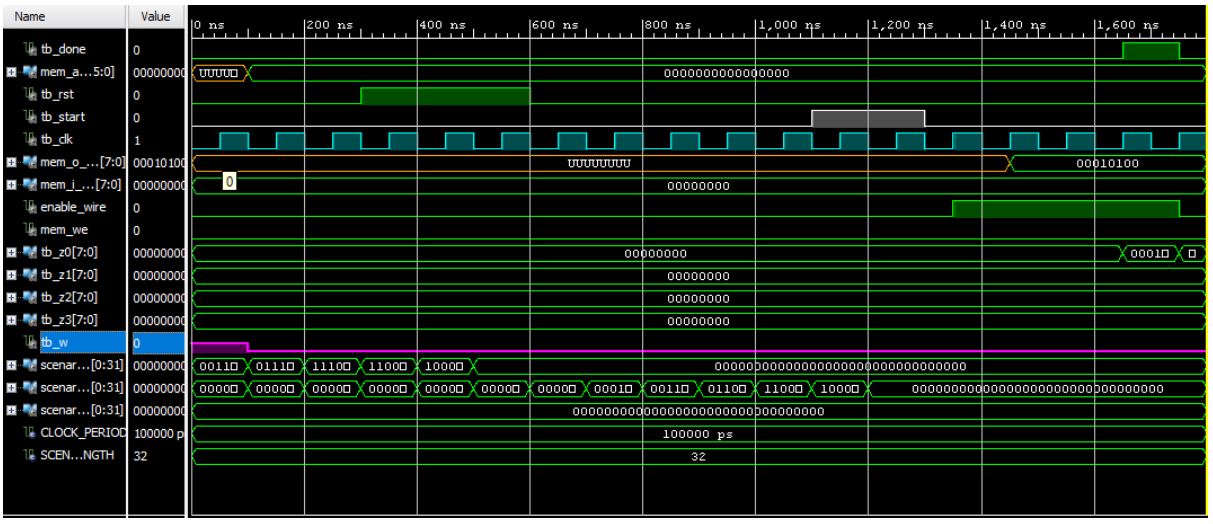


Test bench 7

Risultati:

- launch_simulation: Time (s): cpu = 00:00:02 ; elapsed = 00:00:16 . Memory (MB): peak = 882.898 ; gain = 0.000
- test passato
- Time: 1800 ns Iteration: 0

Il test bench 7 fa un unico test: dopo il reset, il segnale di start rimane alto per 18 cicli di clock durante i quali l'ingresso i_w rimane sempre basso. Per cui l'uscita sarà Z0 mentre l'indirizzo di memoria sarà "0000000000000000".



Conclusioni

Il progetto soddisfa tutti i requisiti richiesti dalla specifica, superando in pre e post sintesi tutti i test bench forniti. Inoltre è stato progettato in maniera efficiente in quanto necessita un periodo di clock minore di quello richiesto dalla specifica.

La macchina a stati creata è di semplice lettura e molto scalabile, in quanto è possibile aggiungere funzionalità per esempio specifiche per ogni canale di uscita o aumentare il numero delle uscite.

Anche il datapath è scalabile, per esempio è necessario modificare unicamente la dimensione dei segnali per poter utilizzare indirizzi di memoria superiori a 16 bit.