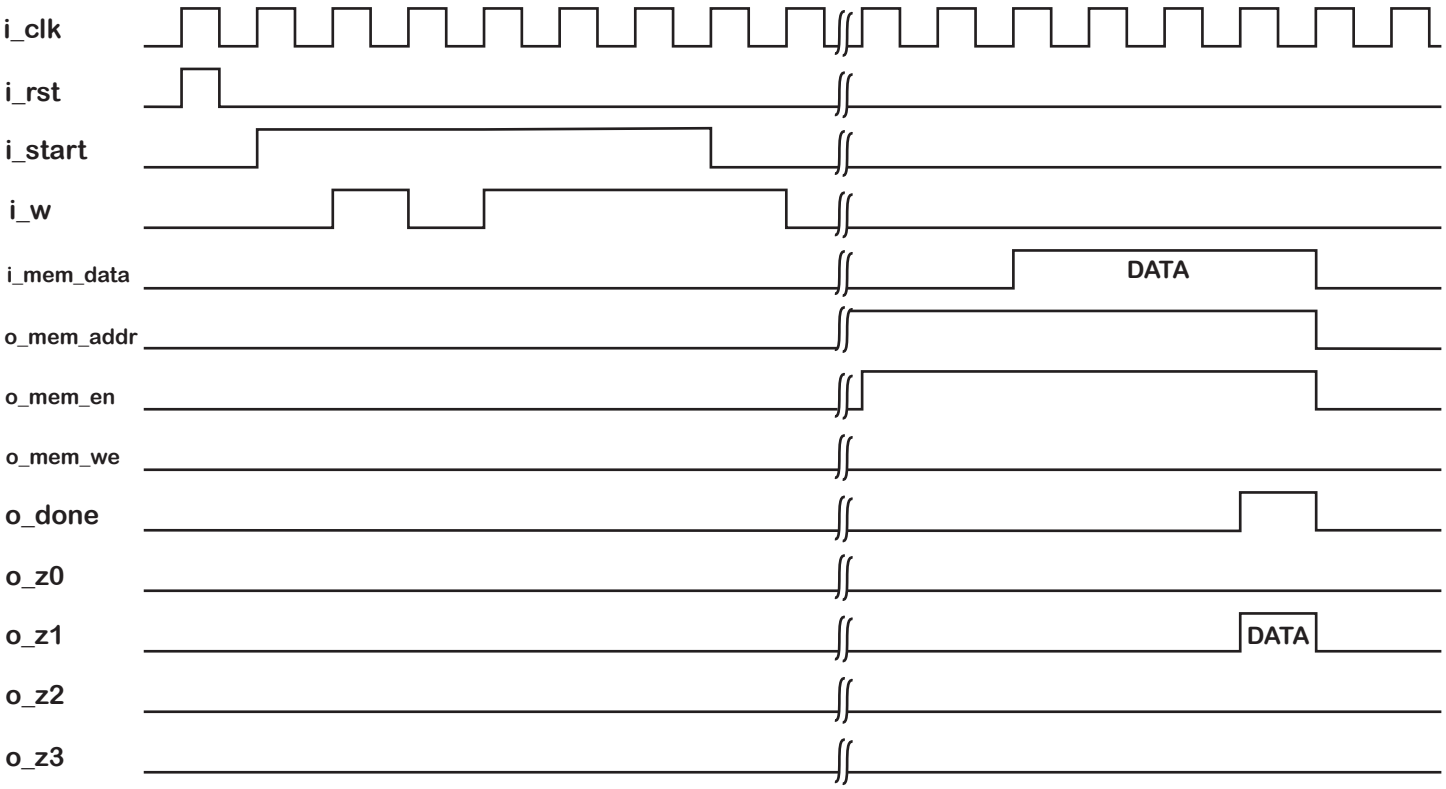
Progetto
Reti Logiche 2022/2023

Davide Vola
Cod. Persona: 10774133

Introduzione:



La rete logica da progettare deve presentare la capacità di leggere i valori di ingresso della porta i_w, solamente quando il segnale i_start è posto a 1 con il clock sul fronte di salita.
I primi due bit di i_w rappresentano la porta su cui dovrà essere fatto uscire il valore del dato, che verrà letto dalla memoria (esterna al progetto). A seconda del valore dei due bit inziali di i_w il valore del dato uscirà in una delle seguenti porte:
00 => o_z0; 01 => o_z1;
10 => o_z2; 11 => o_z3.

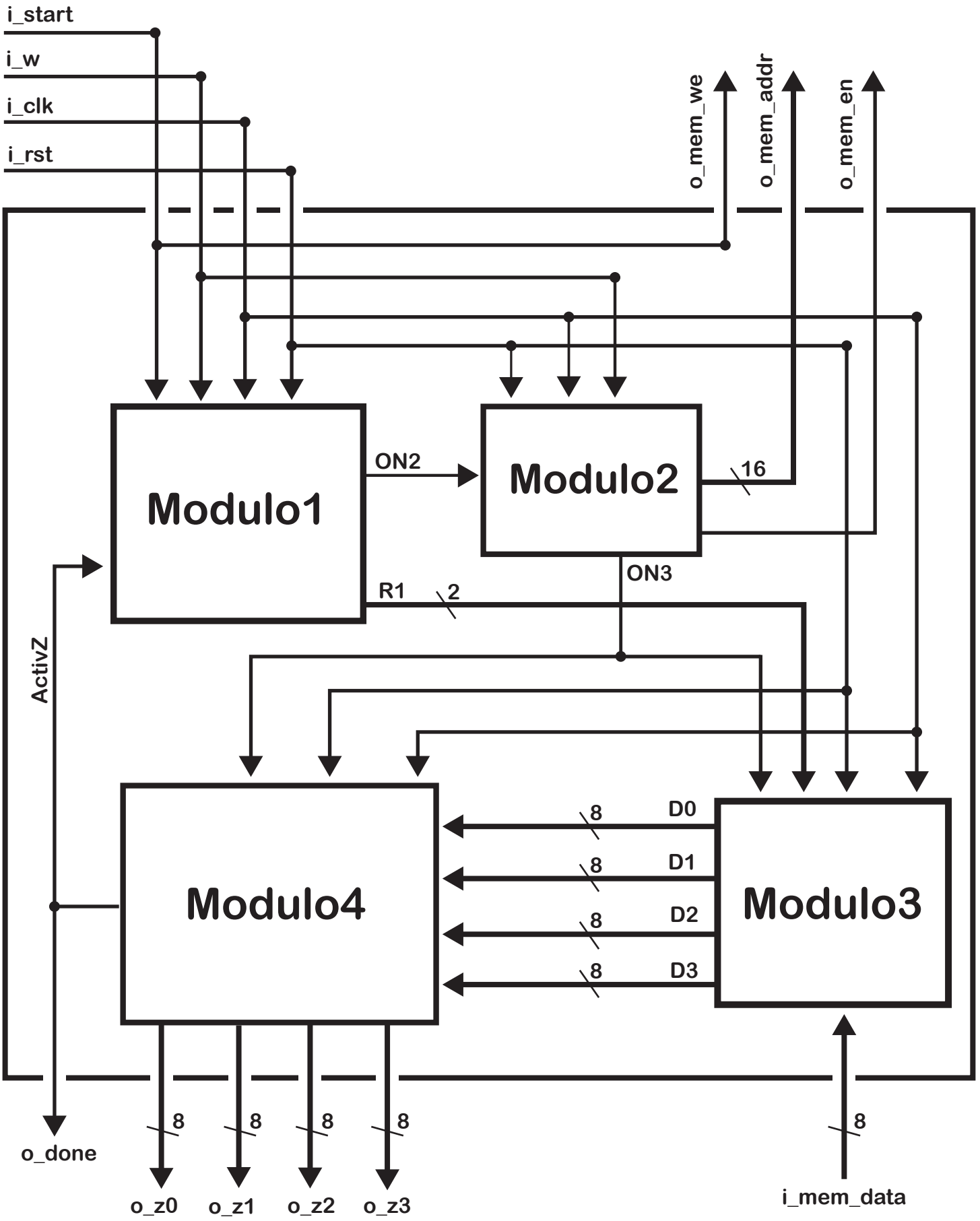
Mentre per gli altri bit di i_w, possono essere letti da un numero che varia da 0 fino a un massimo di 16 bit. Essi rappresentano l'indirizzo in cui dovrà essere letto il dato dalla memoria. Pertanto, in caso i bit letti da i_w non raggiungano i 16 bit, i restanti bit dovranno dovranno essere posti a 0. Come nel seguente esempio:
(N = 6) 101110 => 0000000000101110

La memoria per iniziare a leggere il dato, situato nell'indirizzo inviato, necessita del parametro di attivazione o_mem_en posto a 1. Mentre il parametro o_mem_we rappresenta l'attivazione in lettura o in scrittura della memoria. Per quanto riguarda il progetto si necessita solamente che la memoria legga il dato, non ci necessita di nessuna scrittura. Perciò, il paramentro o_mem_we deve essere posto a 0 quando la memoria verra arrivata. Quando essa è disattivata il suo valore non è di rilievo.
Da quando i_start torna a 0 devono passare venti cicli di clock esatti prima che esso ritorni alto. In questo frangente di tempo, l'architettura da progettare, dovrà ritornare il dato, letto sulla porta i_mem_data, portando o_done a 1 e attivando i valori di uscita dei dati sulle porte Z. Essi non dovranno restare attivi per più di un solo ciclo clock. Una volta cessato il ciclo le porte Z e il o_done dovranno di nuovo mostrare soli 0.

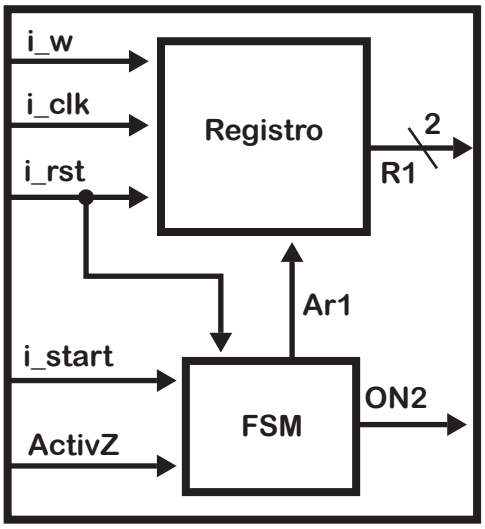
Per ogni volta che o_done va a 1 le porte di Z devono mostrare i valori sia del nuovo dato letto, sulla relativa porta, sia i valori dei dati delle altre porte.
In caso si attivi il reset i valori del dato in uscita su Z devono azzerarsi e il circuito verrà re-inizializzato. All'inizio del programma, prima che si attivi il primo start, deve attivarsi il reset..

Per soddifare tali richieste è stato pensato di suddividere il problema in quattro moduli, tutti con uno specifico compito. Il primo modulo ha il compito di salvare i primi due bit di i_w che reppresentano l'indirizzo della porta in cui dovrà essere mandato il valore della memoria. Il secondo salva i restanti bit di i_w, che reppresenteranno l'indirizzo da inviare alla memoria al fine di ricevere il dato salvato in tale indirizzo. Il terzo modulo salva il relativo dato che arriverà dalla memoria, appena pronto. Mentre l'ultimo modulo deve gestire l'uscita mostrando i valori dei dati trovati durante tutto il funzionamento rispettando la durata con cui deve mostrare i valori.
Possiamo ora vedere nel dettaglio l'architettura che è stata pensata e progettata.

Architettura:

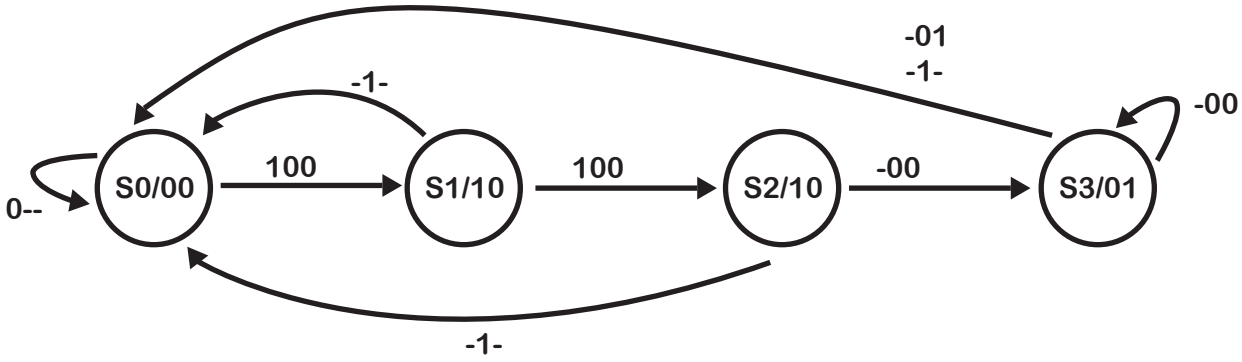


Modulo1:

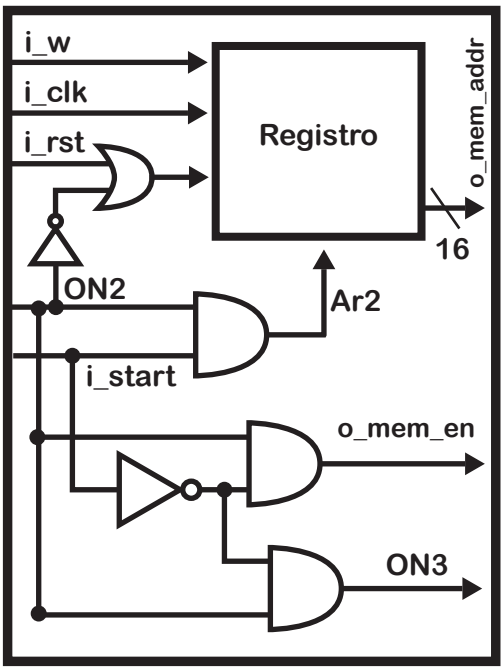


Il Modulo 1 è composto da una macchina a stati di Moore, con 4 stati. Dei quali, il primo attende semplicemente l'arrivo del primo i_start a 1, e finchè non arriva lo stato S0 deve tenere le uscite ON3 e Ar1 a 0. Quando verrà letto il primo 1 di i_start allora si passerà allo stato S1 che attiva il registro, portando Ar1 a 1, in modo che possa essere letto il bit di i_w.
Lo stesso fara lo stato S2. Visto che dalle specifiche del progetto, sappiamo che una voltà che ci sarà il primo i_start a 1 al ciclo di clock successivo resterà ancora a 1. Mentre una voltà che il FSM avrà attivato il registro per due cicli di clock è necessario che passi allo stato S3, per qualunque valore venga letto in i_start, in quanto ho già salvato i due bit di W di interesse. Pertanto Ar1 verrà portato a 0, per non sovrascivere i due bit di i_w letti precedentemente.
Lo stato S3 deve anche attivare ON2 a 1, in quanto ormai il compito del modulo 1 è compiuto e ora dovrà essere attivato il modulo 2.
Il FSM resterà nello stato S3 fino a che ActivZ non sarà posto a 1. Cioè quando il modulo 4 avrà cessato il suo compito e bisognerà riotornare nello stato di partenza: S0. Così che possa ripartire il programma, per la prossima serie di attivazione di i_start.

Si può vedere di seguito l'automa della macchina a stati che è stata pensata, con le porte d'ingresso e d'uscita: i_start, i_rst, ActiveZ/ Ar1, ON2

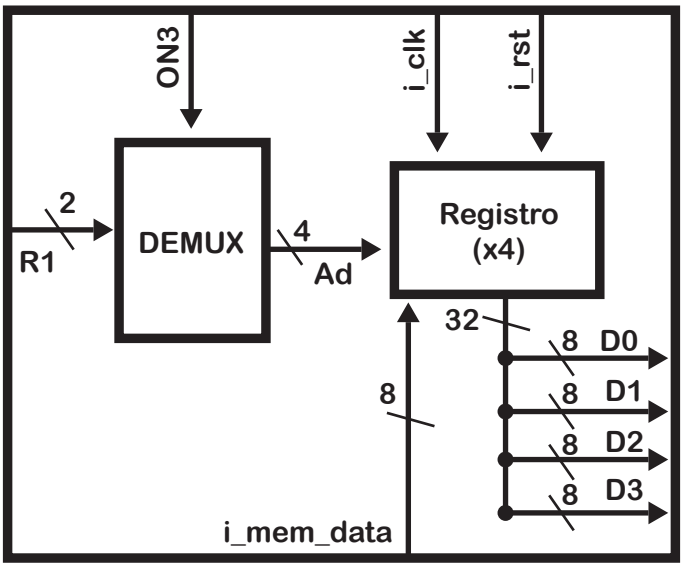


Modulo2:



Il Modulo 2 deve attivare il salvataggio dell'indirizzo in cui dovrà essere letto il dato dalla memoria. Per soddifare ciò è necessario un registro che salvi i bit di i_w, solamente quando Ar2 sarà alto. Pertanto per il segnale Ar2 è stato pensato di usare una porta AND, sui segnali ON2 e i_start, in quanto bisogna attivare il registro solamente quando entrambi i segnali sono posti a 1.
Per attivare il modulo 3 è necessario che, prima, il modulo 2 abbia terminato di salvare i bit di indirizzo nel registro. Ciò avviene solamente quando i_start ritorna a 0, ma per attivare ON3 è necessario che prima sia stato attivato ON2. Per questo è stata implementata una porta AND sui valori di i_start, che prima deve essere negato con una porta NOT, e sul segnale ON2.
Per quanto riguarda o_mem_en, è esattemante uguale per il segnale ON3, in quanto bisogna attivare la memoria solamente quando il Modulo 2 ha cessato di salvare l'indirizzo nel registro.

Modulo3:

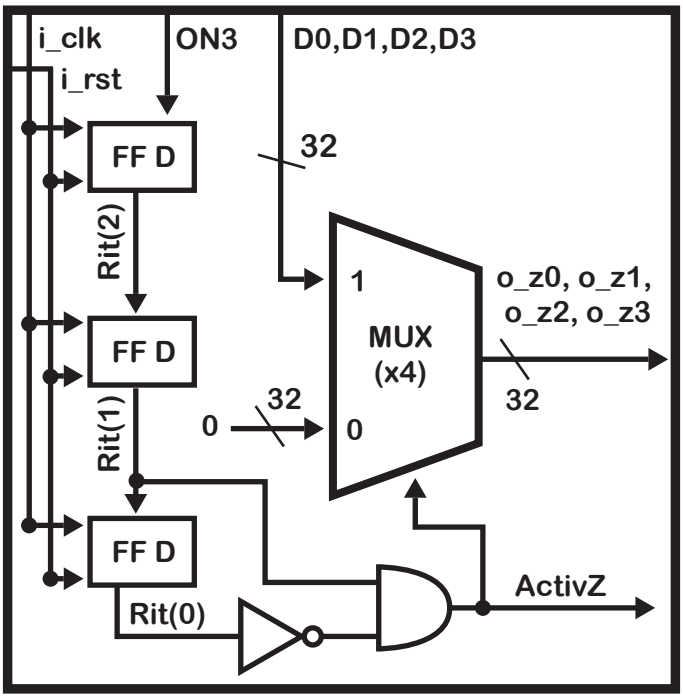


Il Modulo 3 ha il compito di memorizzare il valore del dato che arriverà dalla memoria su uno dei 4 registri, a seconda di quale registro viene attivato. L'attivazione dello specifico registro è gestita dal Demultiplexer. Il quale a seconda dei due bit di R1 verrà posto un segnale alto ad un solo dei 4 bit di Ad, i restanti restano a 0. Tale procedura viene compiuta solamente se ON3 è a 1, in caso contrario tutti i bit di Ad saranno a 0.
Una volta che la scrittura sul registro è attiva, potranno essere salvati i valori del dato che arriveranno dalla porta i_mem_data.

Per una comprensione migliore del Demultiplexer ne viene mostrata la sua tabella:

ON3	R1(0)	R1(1)	Ad(0)	Ad(1)	Ad(2)	Ad(3)
0	-	-	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

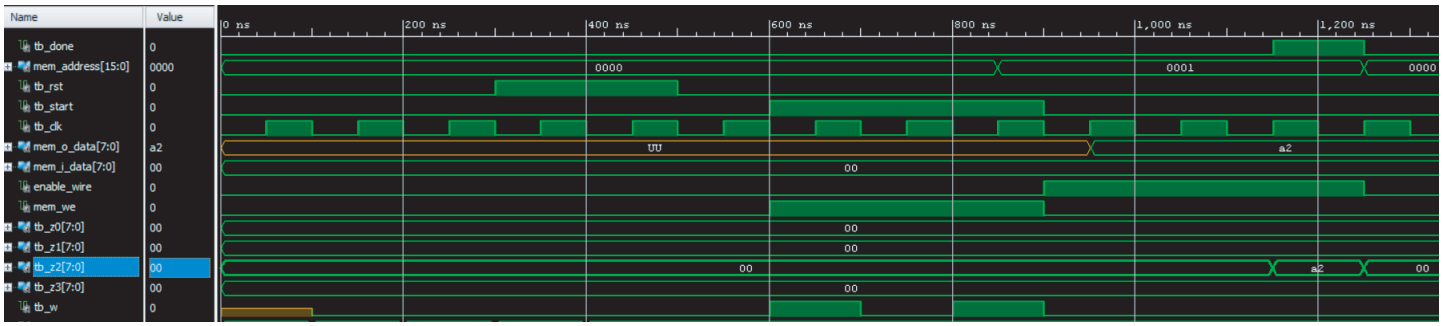
Modulo4:



Il Modulo 4 gestisce i valori d'uscita: o_z0, o_z1, o_z2, o_z3 e o_done. Una volta che ON3 è a 1 non è possibile portare istantaneamente l'uscita o_done a 1, in quanto il registro del modulo 3 non ha ancora fatto in tempo a salvare i bit di i_mem_data. Per arginare tale complicazione è stato pensato di introdurre un ritardo al segnale ON3, in modo che non ci sia il rischio di far uscire sulle porte Z qualche valore non definito o errato. pertanto son stati inseriti 3 Flip-Flop di tipo D.
Il primo ha il compito di propagare il segnale ON3 per ogni ciclo di clock, permettendo così di avere un ritardo sul segnale. Mentre gli altri due gestiscono il problema di tenere attivo il segnale o_done a 1 per un solo ciclo di clock. Gestito da una porta AND che prende i segnali del penultimo FF D e dell'ultimo FF D (negato con una porta NOT), consentendo così che il segnale di ON3 venga propagato fino al penultimo FFD e se esso sarà un segnale alto, allora anche ActivZ diverrà 1. Appena il segnale alto di ON3 arriverà sull'ultimo FFD, l'ActivZ tornerebbe di nuovo basso, permettendo così che il segnale o_done resti a 1 solamente per un ciclo di clock. In questo modo è risolto anche il caso in cui il segnale ON3 vovesse restare alto per svariati cicli di clock, in quanto non comprometterebbe il segnale di o_done.
Per quanto rigarda l'uscita del dato vengono usati 4 Multiplexer che permettono di far uscire i risultati dei 4 registri solamente quando ActivZ è posto a 1, mentre quando è a 0 i valori di uscita delle porte o_z0, o_z1, o_z2, o_z3 sono poste tutte a 0.

Risultati Sperimentali

Sintesi:

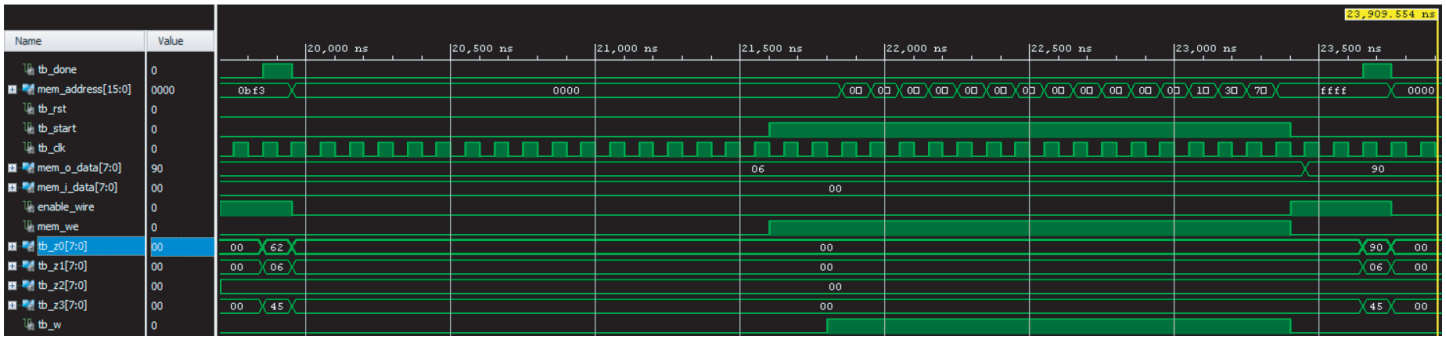


Tutti i test condivisi dal Professore sono stati passati con successo, in fase di sintesi. Dimostrando che il progetto elaborato soddisfa, finora, una buona parte dei criteri. Si può osservare con esempio esplicativo, l'andamento di uno dei test. Dall'immagine si nota che una volta che l'ultimo segnale alto di tb_start ritorna a essere 0, passano esattamente 3 cicli di clock prima che tb_done vada a 1. Restando alto solamente per un ciclo di clock, come richiesto dalla specifica. Un altro punto soddifatto è il valore del dato, letto da memoria, che viene propagato sulla porta d'uscita corretta (in questo esempiò è la porta o_z2). In aggiunta, lo stesso dato letto da memoria corrisponde al valore atteso. Dimostrando che non ci sono problemi sia sui registri che alla lettura dei dati da memoria.

Simulazioni:

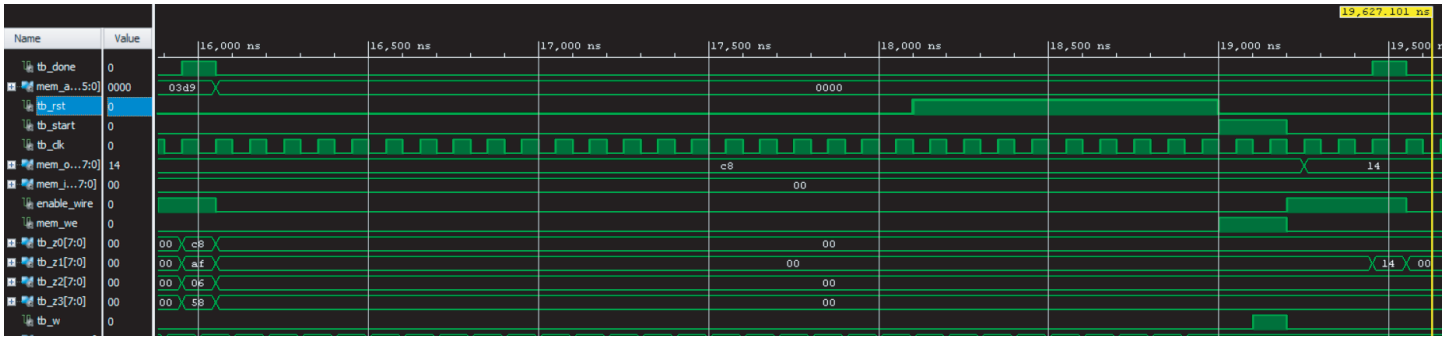
Anche in post-sintesi i test, condivisi dal Professore, vengono passati con successo. Pertanto per completezza e per assicurarsi che il progetto funzioni pienamente per ogni caso, si vedranno alcuni test banch che verifichino il funzionamento delll'architettura nei casi limiti.

Test 1:



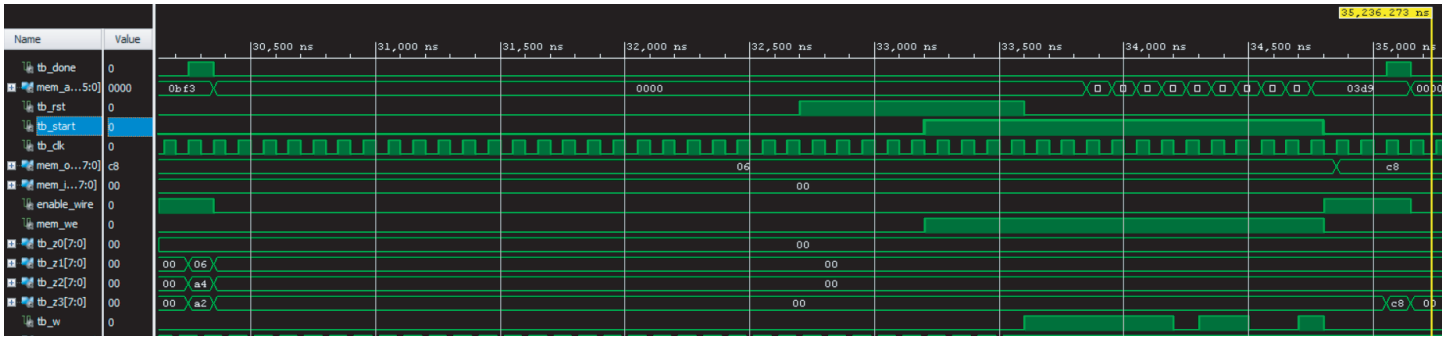
In questo Test si vuole verifica il caso in cui tb_start dovesse restare attivo per il suo valore massimo, ciò 18 cicli di clock, senza che si verifichino errori di nessun tipo.
Come si può vedere dalla simulazione il programma in questo caso non mostra errori, legge correttamente i bit di i_w per 18 cicli di cloack. Si può osservare alla fine del test come il valore di i_mem_data, inviato dalla memoria, non solo è quello atteso ma viene anche mandato alla porta corretta: o_z0. Sovrascrivendo il vecchio valore che era stato salvato precedentemente nel registro, nel funzionamento antecedente. Con questo viene mostrato definitivamente il funzionameto dei registri, sia quelli che salvano i dati che arrivano da memoria, sia quelli che leggono l'indirizzo della porta d'uscita e quelli che salvano l'indirizzo da leggere in memoria. Pertanto il test viene passato con successo.

Test 2:



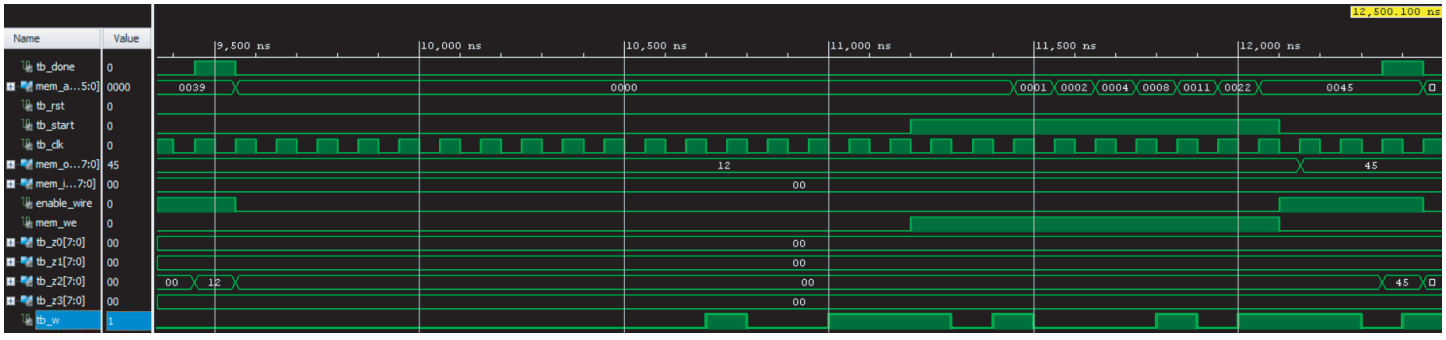
Il Test 2 ha il compito di verificare il corretto funzionameto del progetto nel caso in cui tb_rst dopo essersi attivato ritornasse a 0 e immediatamente dovesse attivarsi tb_start. Anche in questa eventualità il programma agisce senza complicazioni. Si può notare come il reset stesso funzioni nel dovuto modo, infatti i valori dei dati che erano stati salvati nel registro prima dell'attivazione del reset, vengono tutti quanti azzerati una volta attivato il reset. Dando prova del funzionamento del circuito anche per questa eventualità.

Test 3:



Per questo test è stato pensato di collaudare l'eventualità in cui tb_start dovesse attivarsi quando ancora il tb_rst è alto. Dalla simulazione si evidenzia come il modulo 1 non attiva la lettura di tb_w finché il reset è attivo. In questo modo non viene pregiudicato il corretto funzionamemto del circuito. Infatti, quando il reset è attivo viene ignorato il valore di i_start, evitando così situazioni inesatte.

Test 4:



Nell'utlimo test bench si vuole osservare il comportamento dell'architettura quando son presenti dei valori alti di tb_w anche quando tb_start si presenta al suo livello basso. In questa occorrenza si attende che i bit di W al di fuori di Start non vengano considerati. Testando questa ipoesi si è appurato l'assenza di cambiamenti al funzionameto del circuito. Anche in questo caso limite non vengono individuate nessuna gestione errata degli input; il test viene passato con sucesso.

Conclusioni:

Come visto precedentemente, sono stati passati con ampio successo sia i test in simulazione sia quelli in sintesi. Evidenziando chiaramente il funzionameto dell'architettura progettata, nei termini richiesti dalle specifiche progettuali. In più, come osservato nei casi di test limiti, non si presentano criticità, evidenti. Non solo, il progetto realizzato testimonia un agile funzionamento, con l'attitudine a processare i risultati d'uscita finali con un'attesa di soli tre cicli di clock, rispetto ai venti messi a disposizione.
A fine di ciò, si conclude che il progetto è stato realizzato con estremo successo, testimoniando la sua flessibilità e velocità di funzionamento.