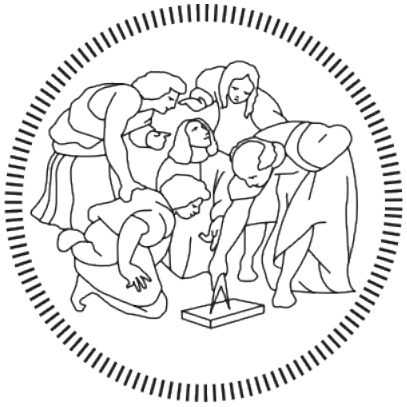
Progetto di Reti Logiche

Anno Accademico 2023/2024

Presentato da

CARMEN GIACCOTTO

ALESSIA FRANCHETTI-ROSADA



POLITECNICO

MILANO 1863

INTRODUZIONE

Con il presente documento viene presentato il progetto svolto in gruppo da Giaccotto Carmen e Franchetti-Rosada Alessia. Si è scelto di realizzarlo in forma collettiva al fine di unire e integrare le rispettive competenze, favorendo un confronto costruttivo e sfruttando le diverse prospettive per ottenere un risultato più approfondito.

1.1 Scopo del progetto

L'obiettivo di questo progetto è l'implementazione di un modulo hardware descritto in VHDL, concepito per interfacciarsi con una memoria e gestire una sequenza di dati secondo regole specifiche.
Il modulo deve leggere una sequenza di K parole, ciascuna con valori compresi tra 0 e 255, memorizzate in memoria a partire da un indirizzo di base fornito. In questa sequenza, il valore 0 rappresenta un dato non specificato e deve essere sostituito con l'ultimo valore valido (diverso da zero) precedentemente letto. Inoltre, per ogni parola della sequenza, il modulo deve calcolare e associare un valore di credibilità C, che indica l'affidabilità del dato. Il valore di C parte da 31 per ogni nuovo dato valido e diminuisce progressivamente in presenza di zeri, senza mai scendere sotto lo zero. Il modulo presenta ingressi per il segnale di clock, il reset, l'indirizzo di memoria (ADD) e la lunghezza della sequenza (K), nonché un segnale di avvio (START) che inizia l'elaborazione. L'elaborazione termina con l'alzata del segnale di fine (DONE), una volta che la sequenza elaborata e i valori di credibilità sono stati correttamente memorizzati.
Il modulo deve garantire che ogni operazione avvenga in modo sincrono al clock, tranne il reset che viene gestito in modo asincrono. La gestione della memoria, sia in lettura che in scrittura, avviene tramite segnali di abilitazione e controllo forniti dal modulo.
Lo scopo del progetto è quindi quello di sviluppare una soluzione efficiente per la gestione di sequenze di dati incomplete, implementando una logica che riempie i valori non specificati e assegna loro un indice di affidabilità, garantendo al contempo che i risultati finali siano memorizzati correttamente nella memoria esterna. Viene fornita l'interfaccia del componente, insieme alla specifica della memoria su cui deve operare e una limitazione temporale che richiede che il modulo hardware esegua correttamente con un periodo di clock di almeno 20 ns.

Per illustrare il funzionamento del modulo hardware, di seguito viene riportato un esempio pratico di elaborazione di una sequenza di dati.
Si supponga che il modulo legga una sequenza di 6 parole memorizzate in memoria a partire da un indirizzo di base fornito (ADD).

Indirizzo	Valore iniziale (Decimale)
ADD	5
ADD+1	0
ADD+2	0
ADD+3	0
ADD+4	8
ADD+5	0
ADD+6	0
ADD+7	0
ADD+8	0
ADD+9	0
ADD+10	21
ADD+11	0

Indirizzo	Valore finale (Decimale)
ADD	5
ADD+1	31
ADD+2	5
ADD+3	30
ADD+4	8
ADD+5	31
ADD+6	8
ADD+7	30
ADD+8	8
ADD+9	29
ADD+10	21
ADD+11	31

Sostituzione dei valori non specificati:

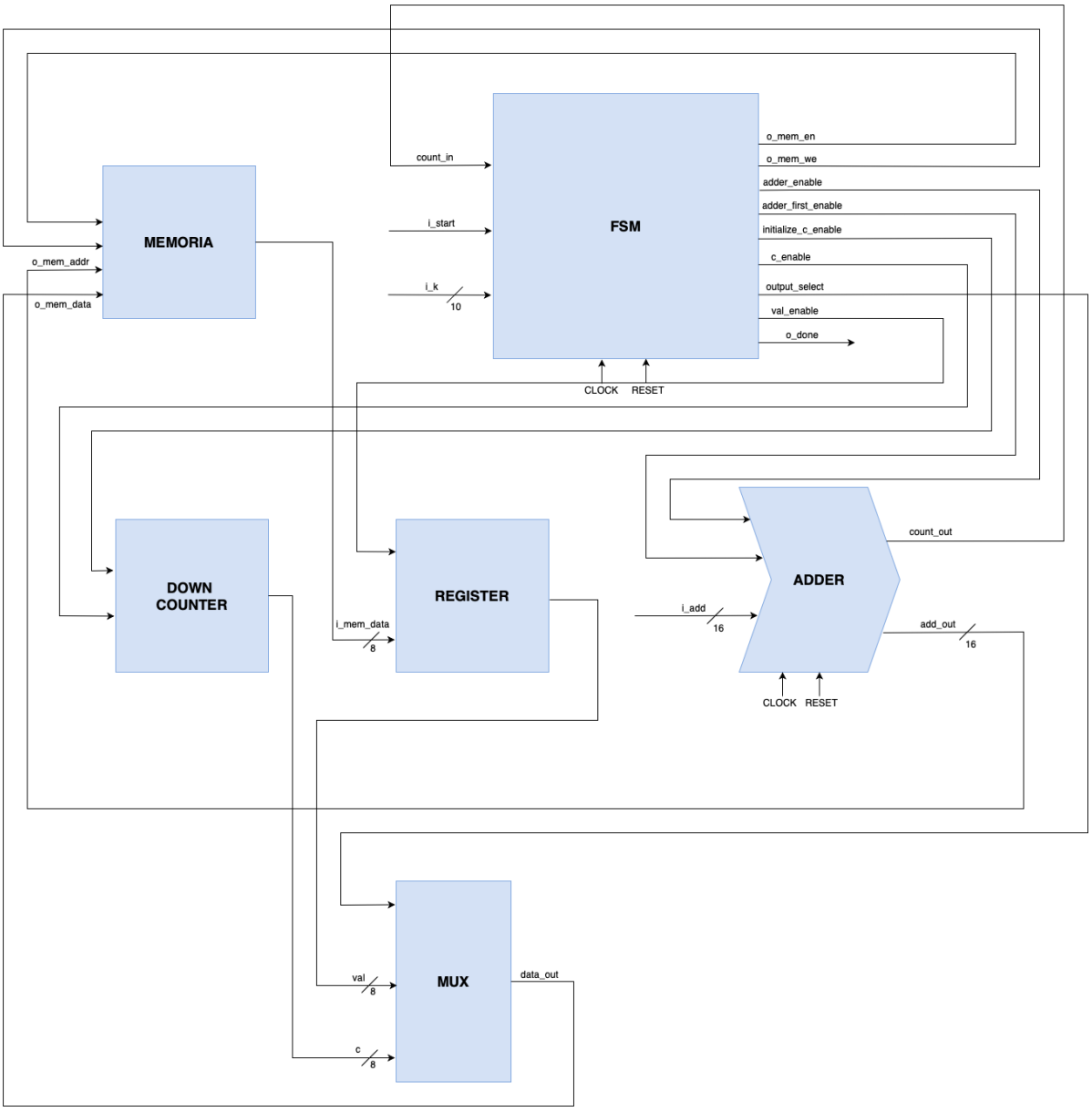
- ADD: il valore letto è 5 che è un dato valido. Pertanto, il valore rimane 5 e la credibilità C viene impostata a 31.
- ADD+2: il valore letto è 0 che rappresenta un dato non specificato. In questo caso, il modulo sostituisce il valore 0 con l'ultimo valore valido precedentemente letto, cioè 5. La credibilità C per questo valore è 30, poiché è decrementata rispetto all'ultimo valore valido.
- ADD+4: il valore letto è 8 che è un dato valido. La credibilità C viene reimpostata a 31.
- ADD+6: il valore letto è 0. Il modulo sostituisce questo valore con l'ultimo valore valido, cioè 8. La credibilità C per questo valore è 30.
- ADD+8: il valore letto è 0. Il modulo sostituisce questo valore con l'ultimo valore valido, cioè 8. La credibilità C viene ulteriormente decrementata a 29.
- ADD+10: il valore letto è 21 che è un dato valido. La credibilità C viene reimpostata a 31.

Alla fine del processo, il modulo ha elaborato la sequenza sostituendo i valori non specificati e calcolato i valori di credibilità associati. Il modulo infine memorizza i valori elaborati e i valori di credibilità nelle posizioni adiacenti.

ARCHITETTURA E SCELTE PROGETTUALI

2.1 Diagramma a blocchi del sistema digitale

In questa sezione viene analizzata l'architettura del sistema digitale progettato attraverso un diagramma a blocchi che viene riportato qui di seguito.



Il sistema è composto da cinque componenti principali: la Finite State Machine (FSM), un decrementatore (down counter), un registro, un sommatore (adder) ed un multiplexer (MUX). Ogni componente svolge una funzione specifica e interagisce con gli altri attraverso connessioni ben definite, che garantiscono il corretto funzionamento dell'intero sistema.

Di seguito si riporta una breve spiegazione delle funzionalità implementate:

- **FSM**: è stata implementata usando due processi. Il primo processo si occupa della logica di transizione tra i vari stati. Si attiva ad ogni fronte di salita del clock e in caso di reset. Il secondo invece è responsabile della generazione dei segnali di controllo in base allo stato corrente e consente la gestione della logica e dell'output del sistema. La scelta di separare la logica di transizione dello stato e la generazione dei segnali di controllo è stata fatta per rendere il codice più leggibile e facile da mantenere.

- **Down Counter**: il compito di questo modulo è quello di decrementare il valore di credibilità (rappresentato dal segnale reg_c che viene trasferito all'uscita c). Il contatore è controllato da due segnali di abilitazione (en1 e en2) gestiti dalla FSM. Quando en1 è attivo il contatore viene inizializzato al valore predefinito di 31, mentre en2, se attivo, permette di decrementare il valore attuale della credibilità. Tuttavia, il decremento avviene solo se il valore corrente è maggiore di zero, evitando così che il contatore assuma valori negativi.

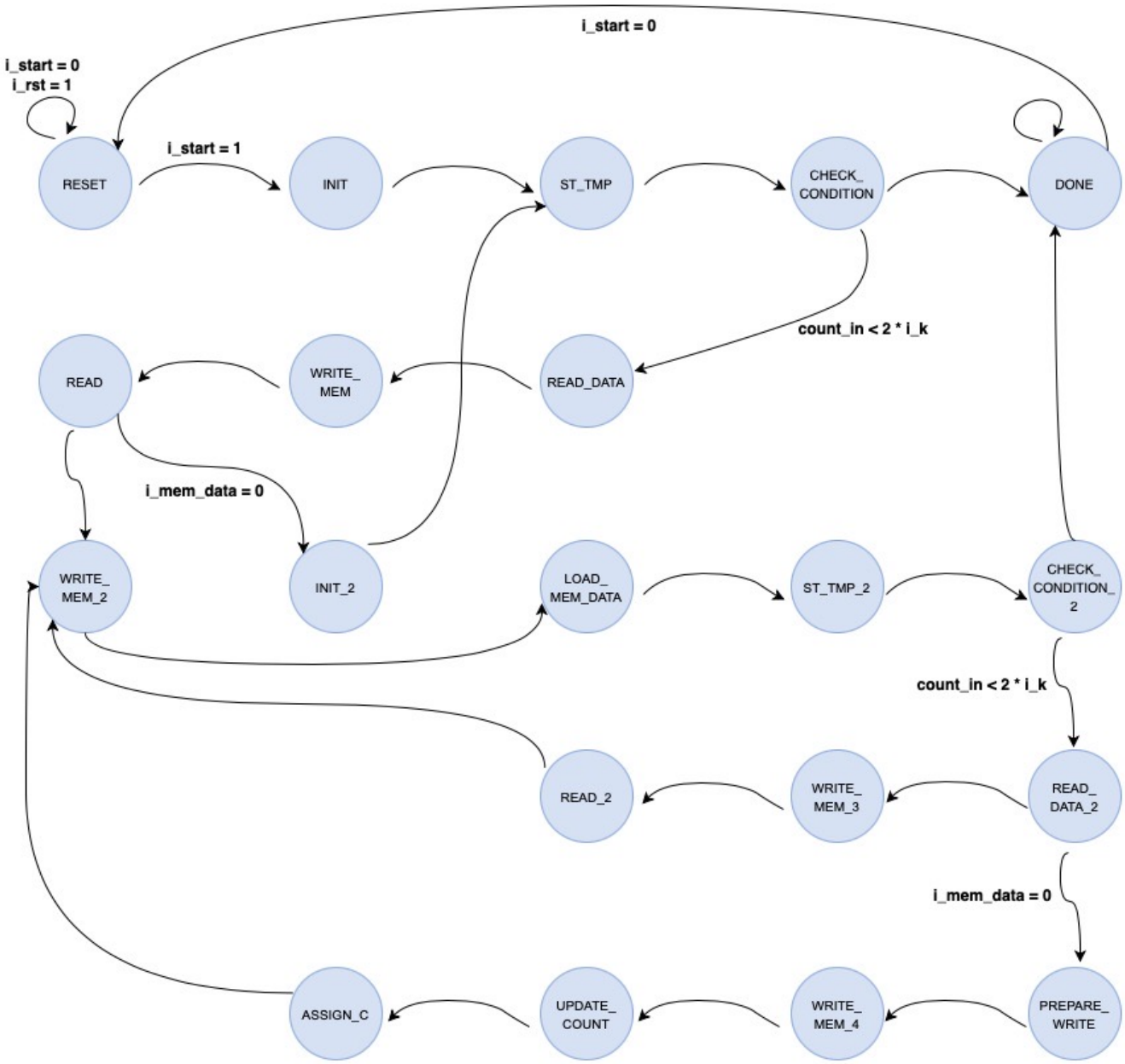
- **Register**: è stato inserito un registro per permettere alla FSM di ottenere e utilizzare sempre il valore corretto da sovrascrivere sulla sequenza di uscita. Quando il segnale di abilitazione (val_enable) fornito dalla FSM è attivo, il registro aggiorna il proprio contenuto con il valore presente in ingresso (d_in) che viene letto dalla memoria. Successivamente, il valore memorizzato viene reso disponibile in uscita. Questo meccanismo permette di evitare dei latch in quanto l'aggiornamento del valore è sincrono con il clock.

- **Adder**: questo modulo ha un duplice scopo all'interno del progetto. L'adder inizializza una variabile con l'indirizzo base fornito dalla memoria e successivamente lo incrementa di 1 a ogni ciclo di clock, quando il segnale enable è attivo. Questo garantisce che i dati vengano scritti nella posizione corretta in memoria. Il contatore associato all'adder offre invece un modo semplice per mantenere traccia di quante parole sono state processate. Questo è essenziale per la gestione della sequenza all'interno della FSM, poiché permette di sapere esattamente quando la sequenza è stata completata e quindi quando si può transitare allo stato DONE.

- **MUX**: si occupa di determinare quale valore, tra i due ingressi val e c, deve essere prodotto come output e scritto in memoria. Il segnale output_select, controllato dall'FSM, guida la scelta tra i due.

2.2 Design della FSM

L'immagine riportata sotto mostra la FSM con una panoramica dettagliata di tutti gli stati e delle transizioni tra di essi, fornendo una visione del comportamento del sistema (non sono stati riportati tutti i segnali di transizione per chiarezza illustrativa).



Di seguito una tabella descrittiva degli stati e della loro funzione.

Stato	Descrizione
RESET	Lo stato iniziale, in cui il sistema viene ripristinato. Il segnale di reset viene attivato e tutte le variabili di controllo vengono inizializzate. Se il segnale i_start è alto, il sistema passa allo stato INIT.
INIT	Prepara il modulo per l'elaborazione. Viene attivato il primo conteggio tramite il segnale adder_first_enable. Al termine, si passa allo stato ST_TMP.
ST_TMP	Stato temporaneo per stabilizzare l'abilitazione del conteggio (adder_enable), disattivandolo prima di passare a CHECK_CONDITION.
CHECK_CONDITION	Verifica se il contatore ha raggiunto il limite della sequenza (2 * K). Se la condizione è soddisfatta, si passa allo stato DONE, altrimenti continua a READ_DATA.
READ_DATA	Lo stato in cui i dati vengono letti dalla memoria e trasferiti nel registro, abilitando val_enable. Al termine, si passa a WRITE_MEM.
WRITE_MEM	Scrive i dati in memoria. Viene abilitata la scrittura in memoria tramite o_mem_we e abilitando il segnale output_select. Il modulo continua con lo stato READ.
READ	Controlla se il dato letto dalla memoria è diverso da zero. In questo caso si abilita il segnale initialize_c_enable e si passa allo stato WRITE_MEM_2. Contrariamente passa allo stato INIT_2. Questo stato è progettato per gestire situazioni in cui il dato iniziale è zero, il che comporta una gestione differente della credibilità.
INIT_2	Stato di inizializzazione secondario che riabilita il conteggio tramite adder_enable e poi ritorna allo stato ST_TMP.
WRITE_MEM_2	Scrive il valore calcolato di credibilità in memoria e disabilita il conteggio. Segue lo stato LOAD_MEM_DATA.
LOAD_MEM_DATA	Viene attivato nuovamente il conteggio tramite il segnale adder_enable. Al termine, si passa a ST_TMP_2.
ST_TMP_2	Stato temporaneo per disabilitare il conteggio, prima di verificare la condizione nel successivo stato.
CHECK_CONDITION_2	Simile a CHECK_CONDITION, verifica se il contatore ha raggiunto il limite della sequenza. Se la condizione è soddisfatta, si passa a DONE, altrimenti continua a READ_DATA_2.

Stato	Descrizione
READ_DATA_2	Controlla se il dato letto è zero o diverso da zero. Se è zero, si passa a PREPARE_WRITE, altrimenti si procede a WRITE_MEM_3.
WRITE_MEM_3	Scrive il dato elaborato in memoria e prepara il modulo per la lettura successiva nello stato READ_2.
READ_2	Abilita il conteggio e prepara i dati per una nuova iterazione, tornando a WRITE_MEM_2.
PREPARE_WRITE	Scrive i dati in memoria e disabilita temporaneamente il conteggio prima di passare a WRITE_MEM_4.
WRITE_MEM_4	Abilita nuovamente il conteggio, è seguito da UPDATE_COUNT.
UPDATE_COUNT	Aggiorna il contatore decrementando il valore di credibilità e, una volta completato, passa a ASSIGN_C.
ASSIGN_C	Assegna il valore di credibilità C per la scrittura e prepara il modulo per il ciclo successivo tornando a WRITE_MEM_2.
DONE	Stato finale in cui viene notificata la conclusione dell'elaborazione impostando o_done a 1. Il sistema rimane in questo stato fino a quando i_start non torna a zero.

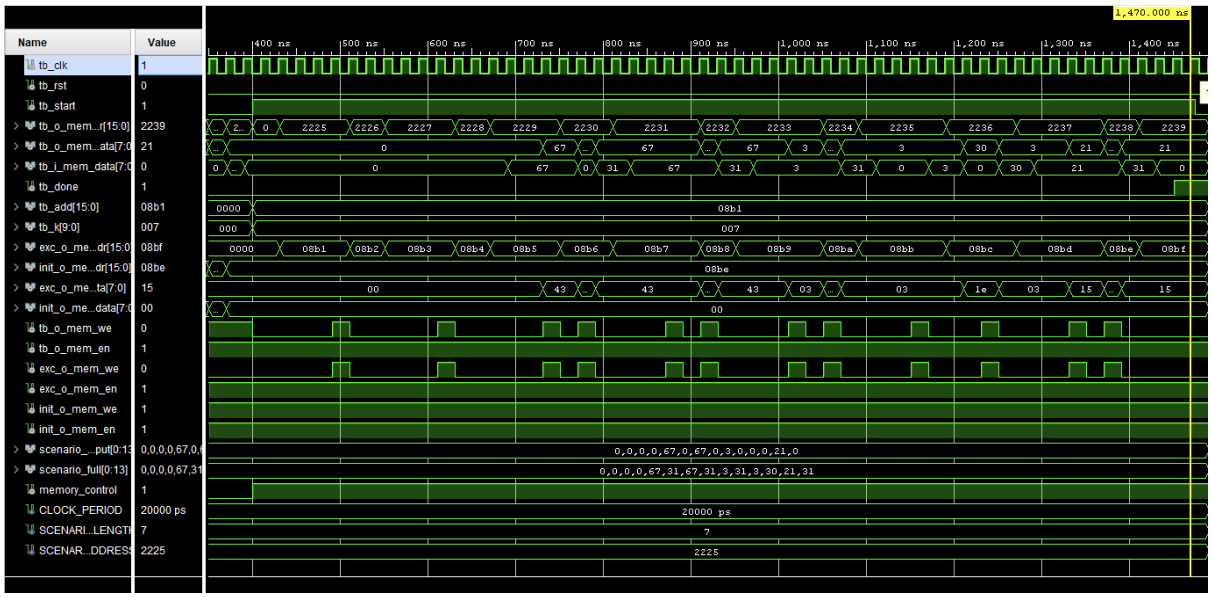
RISULTATI SPERIMENTALI

3.1 Risultati dei test

Al fine di verificare il corretto funzionamento del componente, dopo averlo testato con il test bench di esempio, sono stati definiti altri test in modo da cercare di coprire tutti i corner case possibili. Tutte le simulazioni sono state verificate sia in pre che in post sintesi.
Di seguito è fornita una breve descrizione dei test più significativi utilizzati.

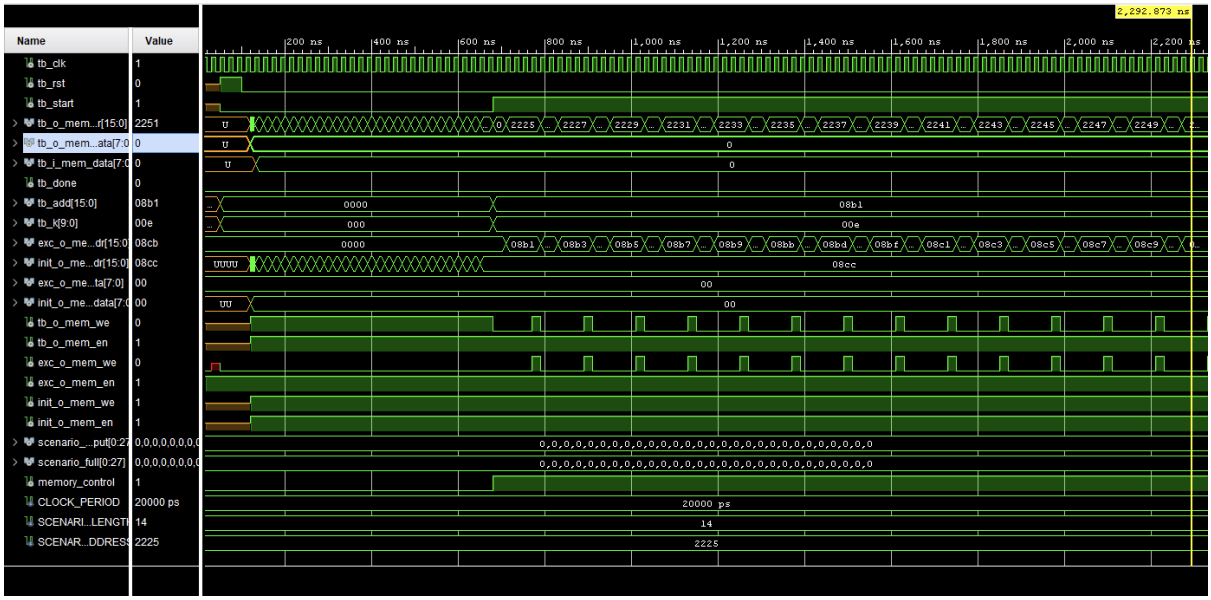
3.1.1 Primi valori della sequenza uguali a zero

Il test verifica che il componente gestisca correttamente i casi in cui i primi valori della sequenza sono pari a zero, assicurando che la credibilità associata a questi valori rimanga invariata a zero.



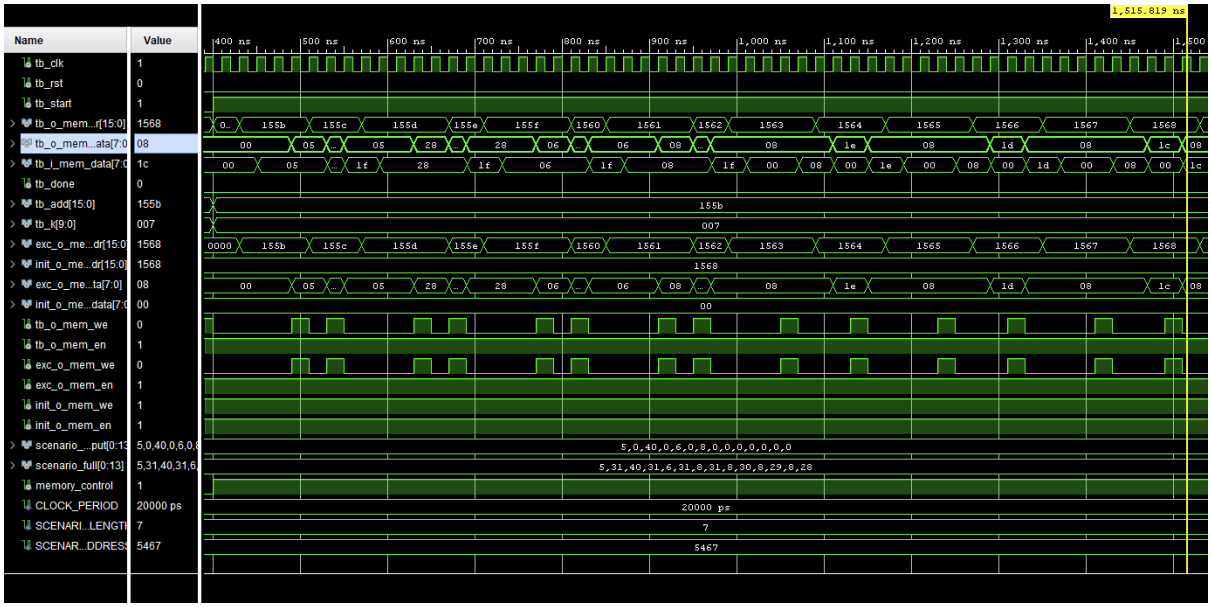
3.1.2 Sequenza composta solo da zeri

Il test verifica il comportamento del componente quando la sequenza di input è costituita esclusivamente da valori pari a zero. In questo scenario, ci si aspetta che la credibilità resti a zero per tutta la durata della sequenza e che il sistema non esegua operazioni non necessarie.



3.1.3 Gli ultimi valori della sequenza sono zeri

Il test controlla se il componente gestisce correttamente una sequenza in cui gli ultimi valori sono zero. Si verifica che, nonostante la presenza di valori validi all'inizio della sequenza, il componente imposti correttamente la credibilità per i valori finali, decrementandola correttamente.



3.1.4 Più sequenze consecutive

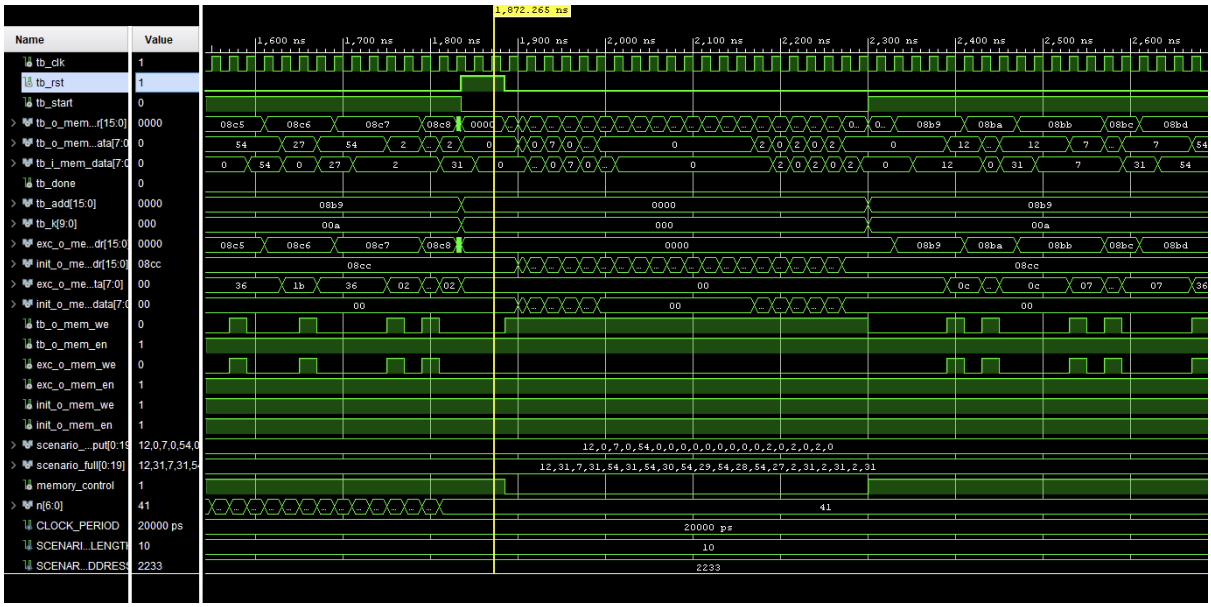
Il test verifica come si comporta la macchina nel caso in cui si abbiamo più sequenze consecutive senza passare dal reset.

3.1.5 Sequenza con solo valori validi

Il test esamina il comportamento del componente quando la sequenza di input è costituita esclusivamente da valori validi e non contiene zeri. Si verifica che la credibilità venga imposta pari a trentuno per ogni valore e che il sistema non introduca errori, garantendo un'elaborazione continua fino alla fine della sequenza.

3.1.6 Reset critico

Il test verifica che l'arrivo del segnale di reset asincrono non comprometta la computazione e che questa riinizi facendo ritornare la macchina nello stato iniziale.



3.2 Risultati della sintesi

In questa sezione vengono riportati alcuni dati sperimentali direttamente ottenuti dal tool di sintesi ed implementazione Vivado.

3.2.1 Area Occupata

Dal "report utilization" generato da Vivado, possiamo estrarre informazioni dettagliate riguardanti l'area occupata dal design sintetizzato. La tabella sottostante riporta il consumo di risorse per il design, evidenziando le risorse specifiche utilizzate, la disponibilità totale e la percentuale di utilizzo.

Risorsa	Utilizzo	Disponibilità	Utilizzo (%)
Look Up Table	65	134600	0.05
Flip Flop	47	269200	0.02
Latch	0	269200	0.00

È stato deciso di evitare l'uso dei latch nel design per diversi motivi tecnici. I latch sono elementi di memorizzazione asincroni e privi di un ingresso di clock dedicato, il che può causare problemi di sincronizzazione. Gli elementi asincroni possono comportarsi in modo imprevedibile, con possibili problemi di temporizzazione e risposte a eventi non legati al clock. Pertanto, per garantire un design più stabile e prevedibile, si è scelto di non utilizzare latch e di basarsi su altre risorse di memorizzazione sincronizzate.

3.2.2 Note aggiuntive

Per questo progetto è stato utilizzato "Vivado v.2018.3.1", e la FPGA xc7a200tfbg484-1.

CONCLUSIONE

Per concludere, il vincolo del periodo di clock di 20 ns è stato rispettato, con il modulo che ne richiede circa quattro. Durante lo sviluppo sono stati affrontati e risolti alcuni problemi, in particolare riguardanti la sintesi, eliminando i latch presenti nel circuito. I risultati ottenuti dai test e dalle simulazioni dimostrano che il progetto funziona correttamente e soddisfa tutte le specifiche richieste. L'adozione di una macchina a stati finiti, con la suddivisione del sistema in componenti autonomi e riutilizzabili, ha permesso di sviluppare un codice efficiente e facile da mantenere, raggiungendo così gli obiettivi prefissati.