POLITECNICO
MILANO 1863

Prova Finale di Reti Logiche

Mattia Siriani

Matricola: 892550

Anno accademico 2019-2020

Indice

1 Introduzione

1.1 Descrizione generale

La specifica della Prova finale "Progetto di Reti Logiche" 2019 è ispirata al metodo di codifica a bassa dissipazione di potenza chiamato "Working Zone".
Questo metodo è per il Bus indirizzi: esso viene utilizzato per codificare in maniera differente il valore di un indirizzo (7 bit) trasmesso, quando questo è compreso tra certi intervalli di valori, denominati working-zone[1] (WZ).
Ci sono due possibili codifiche:

- Se l'indirizzo da trasmettere non appartiene ad alcuna Working Zone, esso viene trasmesso invariato, con un bit addizionale impostato a 0 inserito precedentemente all'indirizzo stesso con un'operazione di concatenamento.
- Se l'indirizzo da trasmettere appartiene ad una Working Zone, il bit addizionale viene messo a 1 mentre i bit di indirizzo vengono rappresentati come segue:
  - I primi tre rappresentano il numero della Working Zone al quale l'indirizzo appartiene.
  - Gli ultimi quattro bit indicano l'offset rispetto all'indirizzo base della Working Zone (codificato in one-hot[2]).

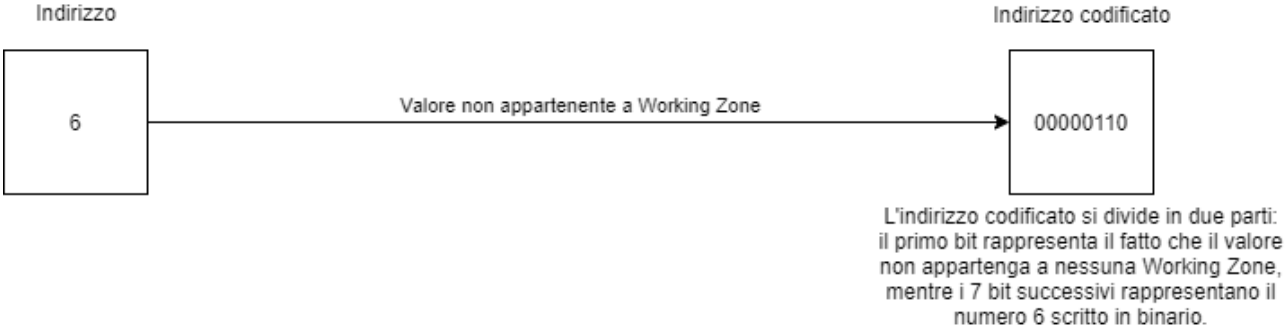
[1]working-zone: è un intervallo di indirizzi di dimensione fissa con partenza da un indirizzo base; i suddetti intervalli non possono essere sovrapposti.

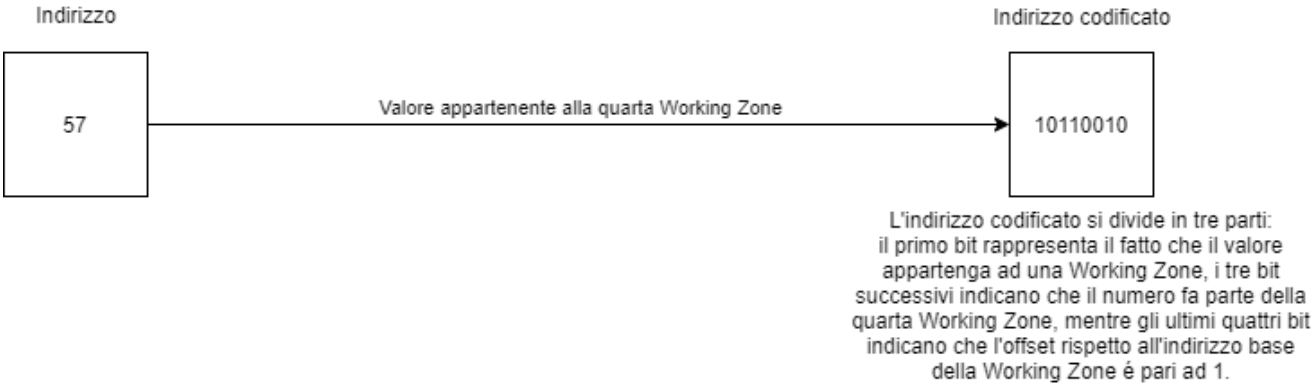
[2]Codifica one-hot: è presente un solo 1 per codifica ed in base alla sua posizione, esso rappresenta un valore differente (esempio: 0 -> 0001 1 -> 0010 2 -> 0100 3 -> 1000).

1.2 Schema di funzionamento

Di seguito sono presenti due esempi di funzionamento della specifica: nel primo viene illustrato il caso in cui il valore non appartiene ad una Working Zone, mentre il secondo rappresenta un caso diametralmente opposto.

WZ1	WZ2	WZ3	WZ4	WZ5	WZ6	WZ7	WZ8
1 - 4	11-14	23-26	56-59	89-92	72-75	60-63	100-104





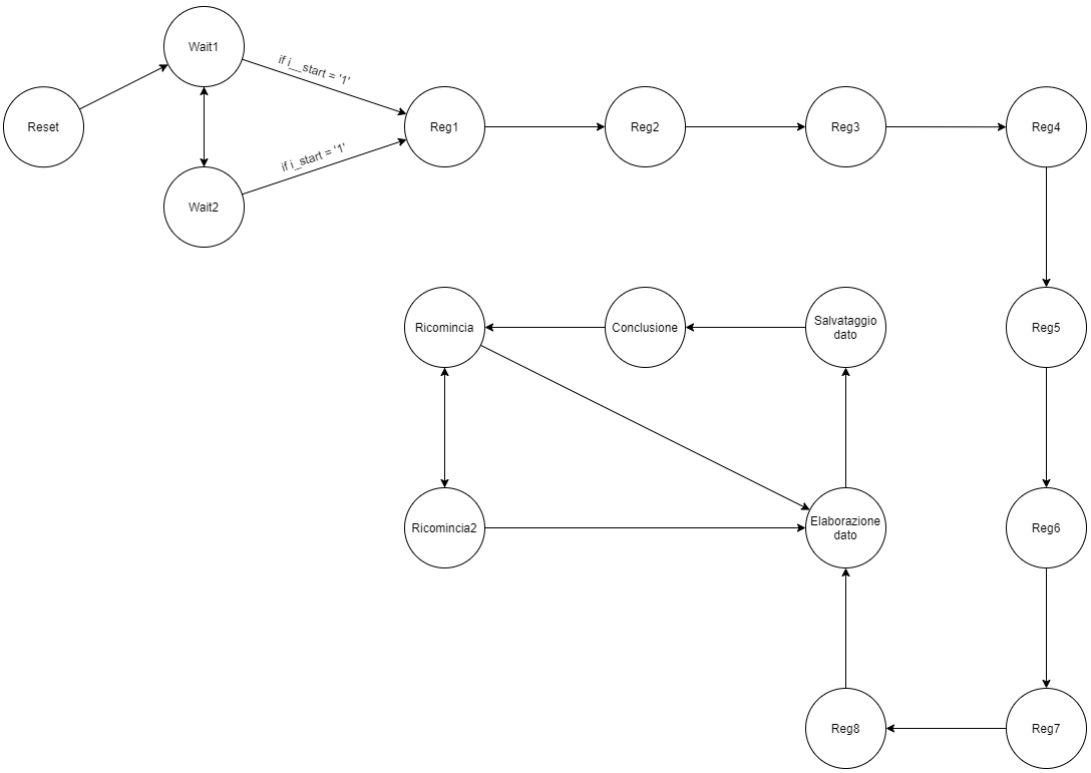
2 Architettura

2.1 Descrizione generale

La specifica da me elaborata è una FSM (Finite State Machine), che si struttura come di seguito descritto:
ho iniziato creando otto segnali, che hanno lo scopo di memorizzare al loro interno i valori delle Working Zone: questo salvataggio comincia quando il segnale *i_start* viene portato alto. In seguito, il componente verifica se il valore presente in RAM(8) appartiene o meno ad una Working Zone e a seconda dei casi subisce una determinata codifica. Alla fine della FSM sono presenti due stati che hanno lo scopo di attendere che *i_start* venga reimpostato ad 1 per poi far ripartire il componente dallo stato "Elaborazione_dato", ovvero il punto in cui si elabora l'appartenenza o meno ad una Working Zone, del valore presente in RAM(8).

2.2 Raffigurazione grafica della FSM proposta

Nell'immagine che segue ho rappresentato la FSM da me elaborata.
Per ogni stato dello schema proposto avrei dovuto inserire una freccia che porta a Reset a significare che il valore di *i_rst* viene alzato ad 1, ma per una maggiore visibilità ho preferito ometterle.



2.3 Descrizione degli stati

Numero	Nome stato	Comportamento
1	Reset	Inizializza tutti i segnali al valore corretto al fine di far funzionare la componente.
2	Wait1	Attende, insieme a Wait2, che il segnale di i_start venga alzato.
3	Wait2	Attende, insieme a Wait1, che il segnale di i_start venga alzato.
4	Reg1	Registra il valore della prima Working Zone all'interno del segnale: "registro1".
5	Reg2	Registra il valore della seconda Working Zone all'interno del segnale: "registro2".
6	Reg3	Registra il valore della terza Working Zone all'interno del segnale: "registro3".
7	Reg4	Registra il valore della quarta Working Zone all'interno del segnale: "registro4".
8	Reg5	Registra il valore della quinta Working Zone all'interno del segnale: "registro5".
9	Reg6	Registra il valore della sesta Working Zone all'interno del segnale: "registro6".
10	Reg7	Registra il valore della settima Working Zone all'interno del segnale: "registro7".
11	Reg8	Registra il valore dell'ottava Working Zone all'interno del segnale: "registro8".
12	Elaborazione_Dato	In questa fase inizialmente si eseguono 8 operazioni matematiche: **RAM(8) − registro(i)** con il valore i compreso tra 1 e 8 (per rappresentare i valori iniziali delle Working Zone). Questi valori vengono memorizzati in 8 variabili differenti denominate *temp(i)* | 1≤i≤8. In seguito, viene elaborato il confronto matematico: **0≤temp(i)≤3** per ogni variabile e se il valore è compreso nell'intervallo, allora il corrispondente registro apparterrà alla medesima Working Zone; in caso contrario, il valore non apparterrà a nessuna Working Zone. Infine, in base alla presenza o meno del valore in una Working Zone, verranno impostati i segnali che andranno poi a codificare il dato opportunamente.
13	Salvataggio_Dato	Il dato viene codificato e salvato nella posizione corretta (RAM(9)).
14	Conclusione	Una volta che il dato è stato correttamente salvato, viene alzato il segnale di *o_done*, fino a che il segnale di *i_start* non si è abbassato, concludendo così l'esecuzione.
15	Ricomincia1	Attende, insieme a Ricomincia2, che il segnale di *i_start* venga alzato e che quindi l'esecuzione possa ricominciare.
16	Ricomincia2	Attende, insieme a Ricomincia1, che il segnale di *i_start* venga alzato e che quindi l'esecuzione possa ricominciare.

2.4 Segnali utilizzati dalla FSM

Nome segnale	Funzionamento
Registro1	Si salva il valore corrispondente alla prima Working Zone.
Registro2	Si salva il valore corrispondente alla seconda Working Zone.
Registro3	Si salva il valore corrispondente alla terza Working Zone.
Registro4	Si salva il valore corrispondente alla quarta Working Zone.
Registro5	Si salva il valore corrispondente alla quinta Working Zone.
Registro6	Si salva il valore corrispondente alla sesta Working Zone.
Registro7	Si salva il valore corrispondente alla settima Working Zone.
Registro8	Si salva il valore corrispondente alla ottava Working Zone.
Wz_bit	Bit che, nella codifica, si utilizza per descrivere se il valore in RAM(8) appartiene (1) ad una Working Zone o meno (0).
Wz_num	Tre bit che, nella codifica, si utilizzano per descrivere a quale Working Zone il valore in RAM(8) appartenga.
Wz_offset	Quattro bit (codificati in one-hot) che, nella codifica, si usano per specificare l'offset del valore in RAM(8), rispetto a quello presente nella Working Zone corrispondente.
Valore	Valore a cui viene assegnato il contenuto di RAM(8), nel caso in cui quest'ultimo valore non appartenesse a nessuna Working Zone.
Is_in	Bit per determinare se il valore in RAM(8) è contenuto o meno in una Working Zone.

2.5 Modifiche progettuali

Durante la realizzazione del componente erano presenti degli stati aggiuntivi (nominati da U1 a U8), posizionati dopo gli stati di memorizzazione dei registri, in modo che il dato venisse salvato correttamente.
Nel risolvere i *warning* di *inferred latch* sono stati aggiunti dei segnali nominati "SegnaleGenerico_next": in questo modo gli stati dei registri vengono eseguiti due volte e quindi il valore di ogni Working Zone viene correttamente registrato nel corrispondente segnale.
Con questa modifica ho ridotto la FSM di otto stati.

3 Sintesi

3.1 Risoluzione warning Post-Synthesis

Ho risolto i seguenti *warning* di post sintesi:

- *Inferred latch*: ho inserito dei segnali denominati: "SegnaleGenerico_next", per permettere l'assegnazione di un valore ad ogni segnale, per tutti gli stati della FSM.
- *Warning* per segnali presenti nel processo, che però non comparivano nella *sensitivity list*.

3.2 Area Occupata

Data la quantità utilizzata, notevolmente inferiore, sia delle Look Up Table (LUT) sia dei Flip Flop (FF) rispetto alla disponibilità totale fornita dalla FPGA scelta (xc7a200tfbg484-1), ho deciso di non ottimizzare ulteriormente l'area.

SiteType	*Used*	*Available*	*Util%*
LUT	207	134600	0.15%
FF	85	269200	0.03%

4 Casi di Test

Di seguito sono presenti i risultati ricavati dal testing della mia componente.
Ho deciso di creare degli appositi "Test Bench" per testare casi limite o determinati comportamenti anormali; sono presenti anche test per esaminare il normale funzionamento del componente.
Ho testato il componente per diversi valori, per quanto riguarda la durata del ciclo di clock: nei casi di test riportati successivamente è stato inserito il valore minimo (in termini di nanosecondi) a cui il componente dava esito positivo ed il valore richiesto da specifica (100 ns).

1. Tb_rst_10_clock_fine_cambio_ram8: questo test bench verifica che, in seguito ad un'elaborazione, se ne esegua un'altra con modificato il valore presente in RAM(8), aspettando 10 cicli di clock.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
4	13	22	31	37	45	77	91	33	1-011-0100
RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)*	RAM(9)
4	13	22	31	37	45	77	91	42	0-42

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1950 ns	TEST 1 PASSATO
Post-Synthesis Functional	1950100 ps	TEST 1 PASSATO
Behavioural	4450 ns	TEST 2 PASSATO
Post-Synthesis Functional	4450100 ps	TEST 2 PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	235 ns	TEST 1 PASSATO
Post-Synthesis Functional	235100 ps	TEST 1 PASSATO
Behavioural	383 ns	TEST 2 PASSATO
Post-Synthesis Functional	383100 ps	TEST 2 PASSATO

Il simbolo di RAM(8)* significa che quel valore è stato modificato.

2. tb_wz_0: questo test bench verifica il caso limite di avere una Working Zone in 0 e che il valore contenuto in RAM(8) sia appartenente alla medesima Working Zone.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
0	13	22	31	37	45	77	91	0	1-000-0001

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1550 ns	TEST PASSATO
Post-Synthesis Functional	1550100 ps	TEST PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	129 ns	TEST PASSATO
Post-Synthesis Functional	129100 ps	TEST PASSATO

3. Tb_wz_124_ram8_max: questo test bench verifica il caso limite opposto al precedente, ovvero di trovarsi nella situazione in cui il valore in RAM(8) sia il massimo possibile ed il valore presente in una Working Zone sia anch'esso il più alto possibile.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
4	13	22	31	37	45	77	91	124	1-111-1000

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1550 ns	TEST PASSATO
Post-Synthesis Functional	1550100 ps	TEST PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	129 ns	TEST PASSATO
Post-Synthesis Functional	129100 ps	TEST PASSATO

4. Tb_wz_disordinate_in: questo test bench verifica che se le Working Zone sono disordinate, il valore viene codificato correttamente.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
109	72	24	43	87	12	99	33	110	1-00-0010

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1550 ns	TEST PASSATO
Post-Synthesis Functional	1550100 ps	TEST PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	129 ns	TEST PASSATO
Post-Synthesis Functional	129100 ps	TEST PASSATO

5. Tb_wz_disordinate_not_in: questo test bench verifica la stessa proprietà vista sopra, solo che in questo caso il valore in RAM(8) non fa parte di una Working Zone.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
109	72	24	43	87	12	99	33	42	0-42

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1550 ns	TEST PASSATO
Post-Synthesis Functional	1550100 ps	TEST PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	129 ns	TEST PASSATO
Post-Synthesis Functional	129100 ps	TEST PASSATO

6. Tb_rst_cambio_valori: questo test bench verifica la proprietà di avere un reset con cambio di tutti i valori, in seguito ad una conclusione di un'elaborazione.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
0	13	22	31	37	45	77	91	42	0-42

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1550 ns	TEST 1 PASSATO
Post-Synthesis Functional	1550100 ps	TEST 1 PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	129 ns	TEST 1 PASSATO
Post-Synthesis Functional	129100 ps	TEST 1 PASSATO

Dopo una prima elaborazione vengono inseriti nuovi valori all'interno della RAM.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
10	16	25	33	38	42	70	99	42	1-101-0001

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	3050 ns	TEST 2 PASSATO
Post-Synthesis Functional	3050100 ps	TEST 2 PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	159 ns	TEST 2 PASSATO
Post-Synthesis Functional	159100 ps	TEST 2 PASSATO

7. Tb_start_10_clock: questo test bench verifica che il segnale di *i_start* potrebbe essere impostato ad 1 in ogni momento, quindi in questo caso si aspettano 10 cicli di clock dopo che il segnale di *i_rst* è stato impostato a 0.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
4	13	22	31	37	45	77	91	33	1-011-0100

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	2550 ns	TEST PASSATO
Post-Synthesis Functional	2550100 ps	TEST PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	247 ns	TEST PASSATO
Post-Synthesis Functional	247100 ps	TEST PASSATO

8. Tb_pfrl_2020_no_wz: questo test bench verifica il normale funzionamento della componente, nel caso il valore non sia contenuto in una Working Zone.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
4	13	22	31	37	45	77	91	42	0-42

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1550 ns	TEST PASSATO
Post-Synthesis Functional	1550100 ps	TEST PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	129 ns	TEST PASSATO
Post-Synthesis Functional	129100 ps	TEST PASSATO

9. Tb_pfrl_2020_in_wz: questo test bench verifica il normale funzionamento della componente, nel caso il valore sia contenuto in una Working Zone.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
4	13	22	31	37	45	77	91	42	1-011-0100

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1650 ns	TEST PASSATO
Post-Synthesis Functional	1650100 ps	TEST PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	229 ns	TEST PASSATO
Post-Synthesis Functional	229100 ps	TEST PASSATO

10. Tb_rst_prima_o_done_cambio_ram8: questo test bench verifica il caso limite in cui il valore codificato viene salvato in RAM(9) con *o_done='0'*; nel frattempo arriva un segnale di *i_rst='1'* che fa ripartire l'esecuzione, ma con RAM(8) modificata, quindi si avrá una codifica differente in RAM(9).

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
4	13	22	31	37	45	77	91	42	0-42
RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)*	RAM(9)
4	13	22	31	37	45	77	91	46	1-101-0010

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	2950 ns	TEST PASSATO
Post-Synthesis Functional	3050100 ps	TEST PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	255 ns	TEST PASSATO
Post-Synthesis Functional	257100 ps	TEST PASSATO

Il simbolo di RAM(8)* significa che quel valore è stato modificato.

11. Tb_restart_dopo_10_clock_fine: questo test bench verifica che un segnale di *i_start* può arrivare dopo n cicli di clock in seguito alla conclusione di un'elaborazione, aspettando 10 cicli di clock.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
4	13	22	31	37	45	77	91	33	1-011-0100

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1950 ns	TEST 1 PASSATO
Post-Synthesis Functional	1950100 ps	TEST 1 PASSATO
Behavioural	3450 ns	TEST 2 PASSATO
Post-Synthesis Functional	3450100 ps	TEST 2 PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	235 ns	TEST 1 PASSATO
Post-Synthesis Functional	235100 ps	TEST 1 PASSATO
Behavioural	363 ns	TEST 2 PASSATO
Post-Synthesis Functional	363100 ps	TEST 2 PASSATO

12. Tb_valore_127_no_wz: questo test bench verifica che il valore in RAM(8) sia il massimo possibile (127) e non appartenga a nessuna Working Zone.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
0	13	22	31	37	45	77	91	127	127

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1550 ns	TEST PASSATO
Post-Synthesis Functional	1550100 ps	TEST PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	129 ns	TEST PASSATO
Post-Synthesis Functional	129100 ps	TEST PASSATO

13. Tb_valore_0_no_wz: questo test bench verifica che il valore in RAM(8) sia il minimo possibile (0) e non appartenga a nessuna Working Zone.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
1	13	22	31	37	45	77	91	0	0

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1550 ns	TEST PASSATO
Post-Synthesis Functional	1550100 ps	TEST PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	129 ns	TEST PASSATO
Post-Synthesis Functional	129100 ps	TEST PASSATO

14. tb_rst_1_clock: questo test bench alza il segnale di *i_rst* ad 1 dopo un ciclo di clock da quando il segnale di *i_start* è stato messo ad 1.
Ho controllato questa condizione anche per 0,2,3,4,5 cicli di clock.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
4	13	22	31	37	45	77	91	33	1-011-0100

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1950 ns	TEST PASSATO
Post-Synthesis Functional	1950100 ps	TEST PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	235 ns	TEST PASSATO
Post-Synthesis Functional	235100 ps	TEST PASSATO

I valori dei test per 0,2,3,4,5 clock sono praticamente uguali tranne per il tempo di esecuzione che cambia ogni volta di 100 o 2 ns in base al clock che utilizzo.

15. Tb_0_clock_rst: questo test bench verifica che, in seguito ad un'elaborazione, venga cambiato il valore in RAM(8) istantaneamente.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
4	13	22	31	37	45	77	91	33	1-011-0100
RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)*	RAM(9)
4	13	22	31	37	45	77	91	42	0-42

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1650 ns	TEST 1 PASSATO
Post-Synthesis Functional	1650100 ps	TEST 1 PASSATO
Behavioural	3350 ns	TEST 2 PASSATO
Post-Synthesis Functional	3350100 ps	TEST 2 PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	229 ns	TEST 1 PASSATO
Post-Synthesis Functional	229100 ps	TEST 1 PASSATO
Behavioural	361 ns	TEST 2 PASSATO
Post-Synthesis Functional	361100 ps	TEST 2 PASSATO

Il simbolo di RAM(8)* significa che quel valore è stato modificato.

16. Tb_rst_10_clock_fine: questo test bench verifica che, in seguito ad un'elaborazione si aspettino 10 cicli di clock prima di avere un *i_rst='1'* che, in questo caso, modifichi tutti i valori in memoria.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
4	13	22	31	37	45	77	91	33	1-011-0100

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	1950 ns	TEST 1 PASSATO
Post-Synthesis Functional	1950100 ps	TEST 1 PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	235 ns	TEST 1 PASSATO
Post-Synthesis Functional	235100 ps	TEST 1 PASSATO

Dopo una prima elaborazione vengono inseriti nuovi valori all'interno della RAM.

RAM(0)	RAM(1)	RAM(2)	RAM(3)	RAM(4)	RAM(5)	RAM(6)	RAM(7)	RAM(8)	RAM(9)
10	16	25	33	38	42	70	99	98	0-98

Clock 100 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	4450 ns	TEST 2 PASSATO
Post-Synthesis Functional	4450100 ps	TEST 2 PASSATO

Clock 2 ns

Tipologia	Tempo di esecuzione	Esito
Behavioural	383 ns	TEST 2 PASSATO
Post-Synthesis Functional	383100 ps	TEST 2 PASSATO

5 Conclusioni

Il componente funziona correttamente come da specifica: é stato da me sottoposto ad una grande quantità di test, che hanno dato tutti esito positivo, sia in Behavioural, sia in Post-Synthesis Functional, sia in Post-Implementation Functional, anche se quest'ultimo non veniva richiesto in specifica.
Sarebbe stato possibile minimizzare la FSM, ma ho deciso di tenere alcuni stati separati per favorirne la leggibilità e mantenere una chiara distribuzione logica.
Ho deciso di non ottimizzare ulteriormente l'area in quanto la quantità di Look Up Table (0,15% *used*) e di Flip Flop (0,03% *used*) è notevolmente inferiore a quella totale.

Il componente funziona correttamente fino ad un periodo di clock di 2 ns, cioè il 98% più veloce del clock voluto in specifica (100 ns).