Davide Foini

Matricola 909710

Codice Persona 10674448

PROVA FINALE

Progetto di Reti Logiche

A.A. 2020/2021

Scaglione Prof. Fabio Salice

SOMMARIO

1. INTRODUZIONE

Il progetto ha come scopo quello di implementare un componente hardware tramite il linguaggio VHDL che sia in grado di equalizzare l'istogramma di un'immagine in scala di grigi a 256 livelli.
Al componente è richiesto di interfacciarsi con una memoria sequenziale già fornita da cui leggere un'immagine e i dati sulle sue dimensioni, per poi scrivere l'immagine equalizzata subito dopo quella originale. Una schematizzazione della memoria è riportata di seguito, con gli indirizzi espressi in decimale.

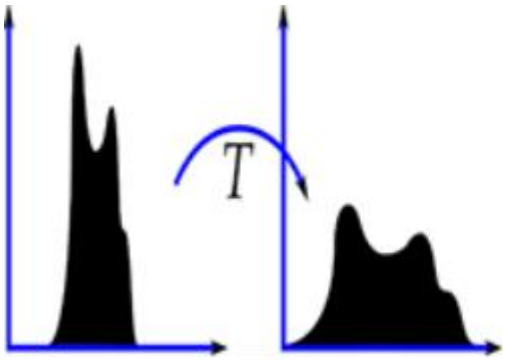
Indirizzo	0	1	2	...	1 + n° pixel immagine
	numero di colonne	numero di righe	primo pixel immagine	...	ultimo pixel immagine

L'algoritmo di equalizzazione dei pixel è fornito dalla specifica:

```
DELTA_VALUE = MAX_PIXEL_VALUE – MIN_PIXEL_VALUE
SHIFT_LEVEL = (8 – FLOOR(LOG2(DELTA_VALUE +1)))
TEMP_PIXEL = (CURRENT_PIXEL_VALUE - MIN_PIXEL_VALUE) << SHIFT_LEVEL
NEW_PIXEL_VALUE = MIN( 255 , TEMP_PIXEL)
```

All'algoritmo è necessario provvedere un'equivalente codifica in VHDL. In primo luogo, è necessario calcolare la differenza fra il valore massimo e minimo dei pixel, dopodiché è possibile ottenere lo *shift* che sarà applicato ad ogni pixel. Nel caso in cui il valore del pixel equalizzato sia maggiore di 255, in memoria esso viene memorizzato con valore 255 per rispettare il limite dei 256 livelli di grigio.

Un esempio di equalizzazione è riportato nelle figure sottostanti.



FONTE

FONTE

2. ARCHITETTURA

Nelle specifiche di progetto viene fornita l'interfaccia del componente, riportata di seguito.

```
entity project_reti_logiche is
      port (
            i_clk     : in std_logic;
            i_rst     : in std_logic;
            i_start   : in std_logic;
            i_data    : in std_logic_vector(7 downto 0);
            o_address : out std_logic_vector(15 downto 0);
            o_done    : out std_logic;
            o_en      : out std_logic;
            o_we      : out std_logic;
            o_data    : out std_logic_vector (7 downto 0)
      );
end project_reti_logiche;
```

- *i_clk* rappresenta il segnale di clock in ingresso;
- *i_rst* è il segnale di reset che inizializza la macchina pronta a ricevere il segnale di start;
- *i_start* è il segnale che da inizio all'elaborazione;
- *i_data* è il segnale proveniente dalla memoria e contenente il valore richiesto in precedenza all'indirizzo specificato;
- *o_address* contiene l'indirizzo da cui il componente richiede di leggere o in cui scrivere;
- *o_done* è il segnale che il componente alza al termine dell'equalizzazione e scrittura in memoria di un'immagine;
- *o_en* serve ad abilitare la memoria;
- *o_we* rappresenta il metodo di accesso alla memoria: '0' per la lettura e '1' per la scrittura;
- *o_data* è il segnale contenente il valore da scrivere all'indirizzo *o_address*.

Vista l'assenza di particolari requisiti in termini di efficienza nella specifica, si è ritenuto soddisfacente optare per una soluzione algoritmica semplice, senza una particolare focalizzazione sull'ottimizzazione.

È stata utilizzata una Macchina a Stati Finiti a dieci stati, il cui funzionamento è stato simulato tramite un solo processo. Il dispositivo utilizzato è l'FPGA xc7a200tfbg484-1.

L'algoritmo si compone principalmente di cinque fasi:

1. Inizio dell'elaborazione;
2. Richiesta e memorizzazione delle dimensioni dell'immagine;
3. Scorrimento dei pixel dell'immagine per trovare i valori massimo e minimo dei pixel per calcolare il valore di shift;
4. Equalizzazione dei pixel e scrittura in memoria;
5. Fine dell'elaborazione e preparazione per una possibile nuova immagine.

2.1 SEGNALI DELLA MACCHINA

L'architettura del componente sviluppata fa uso di alcuni segnali interni e di un tipo di segnale *state* appositamente creato per rappresentare lo stato della macchina. Si riporta la sezione dichiarativa dei segnali.

```
type state is (INIT, RESET, ASK_SIZE, GET_SIZE, ASK_VALUE, WAIT_MEMORY, COMPARE, EQUALIZE, WRITE, DONE);
signal current_state : state;
signal current_address : std_logic_vector (15 downto 0);
signal max : std_logic_vector (7 downto 0);
signal min : std_logic_vector (7 downto 0);
signal max_set : std_logic;
signal min_set : std_logic;
signal shift : std_logic_vector(7 downto 0);
signal cols : std_logic_vector(7 downto 0);
signal rows : std_logic_vector(7 downto 0);
signal eq_value : std_logic_vector(15 downto 0);
signal cols_set : std_logic;
signal rows_set : std_logic;
signal shift_set : std_logic;
```

Come già descritto in precedenza, *current_state* indica lo stato corrente, *current_address* rappresenta l'indirizzo di memoria raggiunto durante l'elaborazione. *Max, min, shift, cols, rows* contengono rispettivamente il valore massimo e minimo dei pixel dell'immagine, il valore dello shift calcolato ed il numero di colonne e righe di cui è composta l'immagine da equalizzare. *Max_set, min_set, cols_set, rows_set* servono come indicatori della fase in cui si trova la macchina: se il *set* ha valore '1' la macchina ha già richiesto alla memoria o calcolato il valore e l'ha già attribuito al segnale corrispondente, '0' altrimenti. *Eq_value* rappresenta il valore del pixel equalizzato.

Si è scelto di utilizzare tutti i segnali (oltre a *current_state*) di tipo *std_logic_vector* e *std_logic* perché in memoria i valori sono memorizzati in binario e per non dover utilizzare eccessivamente funzioni di conversione.

2.2 STATI DELLA MACCHINA

Di seguito sono descritti in modo più dettagliato gli stati della macchina, che non inizia l'elaborazione fintanto che viene alzato il segnale *i_rst.*

INIT

Lo stato in cui resta la macchina in attesa del primo segnale di reset.

RESET

Lo stato in cui la macchina si porta all'alzarsi del segnale *i_rst* oppure al termine dell'elaborazione di un'immagine. In questo stato avviene l'inizializzazione dei segnali interni e di output.

ASK_SIZE

Viene richiesto alla memoria il numero di colonne o quello di righe, memorizzati agli indirizzi 0 e 1 (espressi in decimale), a seconda del *current_address.*

GET_SIZE
In base ai valori di *cols_set* e *rows_set* si riconosce se si sta ricevendo dalla memoria il numero di colonne/righe e di conseguenza si memorizza il valore in *cols*/*rows*, dopodiché si setta a '1' cols_set/rows_set, a seconda del caso.

ASK_VALUE
Viene richiesto alla memoria il valore del pixel a seconda del *current_address* e viene calcolato il valore di shift una volta raggiunto l'ultimo pixel dell'immagine originale.

WAIT_MEMORY
Questo stato funge da "stato di attesa" del valore richiesto alla memoria e indica il prossimo stato utilizzando i segnali di *set* indicati in precedenza:

- Se non sono ancora stati memorizzati *cols* o *rows*, il prossimo stato sarà ASK_SIZE;
- Se non sono ancora stati memorizzati *max* o *min,* il prossimo stato sarà COMPARE;
- Altrimenti il prossimo stato sarà EQUALIZE.

COMPARE
In questo stato avviene il confronto tra il valore del pixel appena letto dalla memoria e i valori di *max* e *min* fino a quell'istante e li aggiorna con i valori appena letti nel caso in cui il pixel corrente sia massimo o minimo. Se il *current_address* raggiunto è l'ultimo dell'immagine, allora *current_address* viene riportato al primo pixel e i *set* del massimo e del minimo vengono portati a '1'.

EQUALIZE
Eq_value viene aggiornato con il valore equalizzato del pixel corrente. Viene utilizzata la funzione *shift_left* per effettuare lo shift dei bit.

WRITE
Viene inviato alla memoria il valore di *eq_value* nel caso in cui esso sia inferiore a 255, altrimenti il valore 255 (in binario). Se la macchina ha raggiunto l'ultimo pixel dell'immagine il prossimo stato della macchina sarà DONE, altrimenti ASK_VALUE dove verrà chiesto il valore del prossimo pixel.

DONE
L'ultimo stato in ordine cronologico, nel quale viene portato a '1' il segnale *o_done* per indicare il termine dell'elaborazione. La macchina resta in questo stato fino a che il segnale *i_start* viene riportato a '0', momento del quale ritorna allo stato RESET per prepararsi ad una nuova immagine.

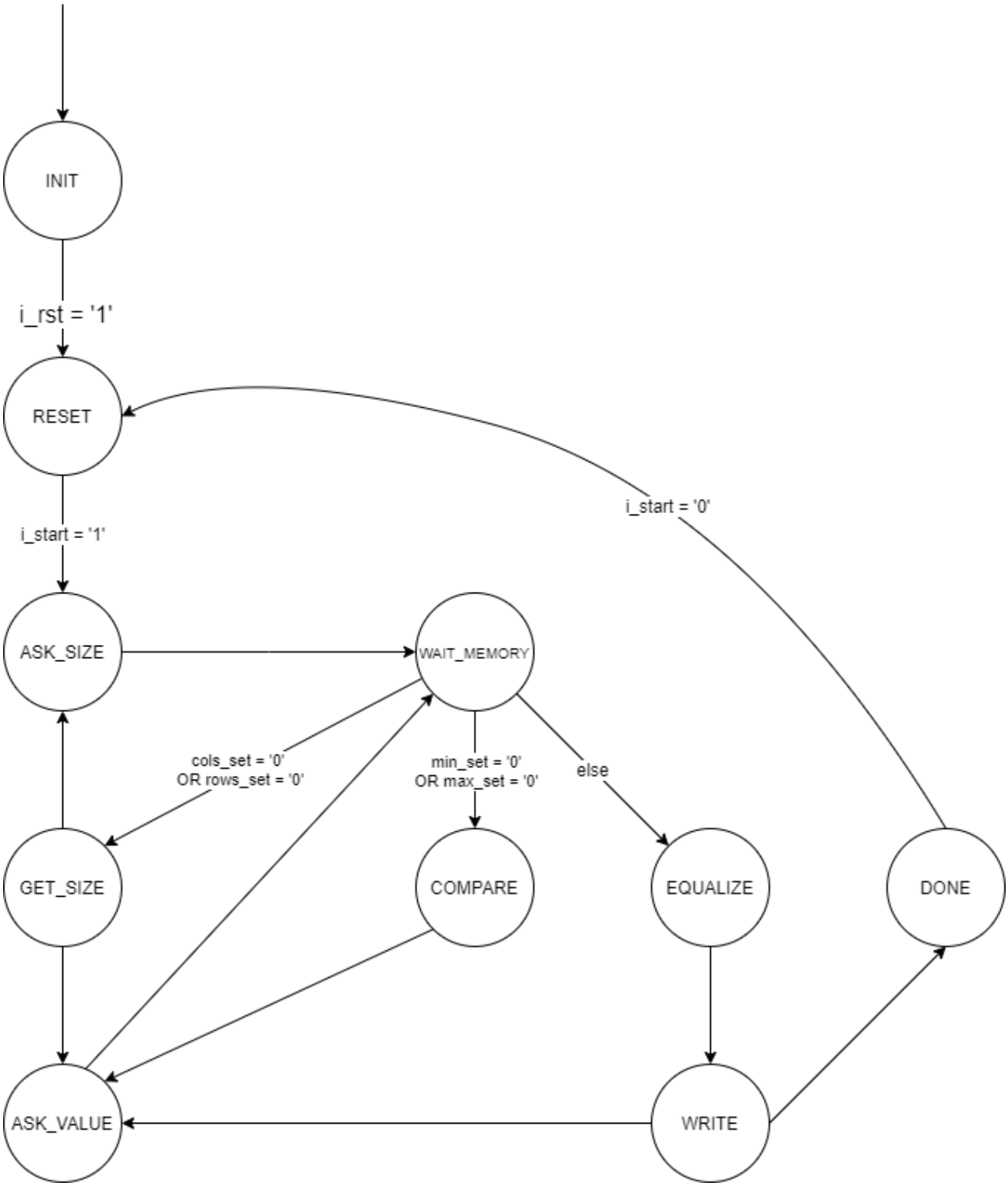


DIAGRAMMA DEGLI STATI DELLA FSM

NOTA: per semplicità non sono state indicate le frecce che da ogni stato portano allo stato di RESET in caso di reset asincrono.

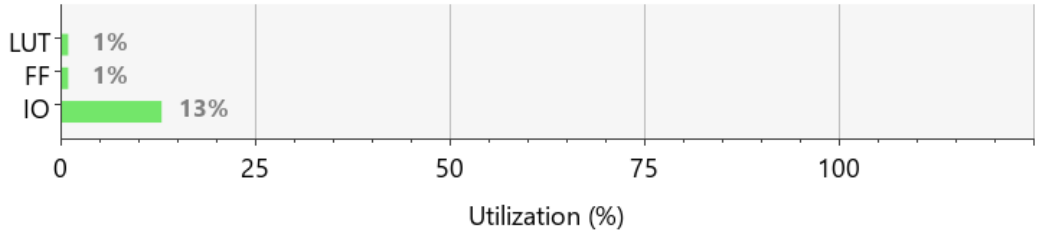
3. RISULTATI SPERIMENTALI

Il componente viene sintetizzato senza nessun errore o avviso. I componenti RTL risultati dopo l'elaborazione del design da Vivado sono riportati nella seguente tabella.

COMPONENTE	NUMERO DI INPUT	NUMERO DI BIT	TOTALE
ADDER	2	16	1
	3	8	2
REGISTRO	-	16	3
	-	8	7
	-	4	1
	-	1	7
MULTIPLEXER	11	16	3
		8	3
		1	19
	3	4	1
		3	1
	2	16	1
		8	1
		4	1
		1	3
Totale ADDER	-	-	3
Totale REGISTRI	-	-	18
Totale MULTIPLEXER	-	-	33
TOTALE COMPONENTI	**-**	**-**	***54***

L'utilization report prodotto dalla sintesi mostra inoltre come le risorse disponibili nell'FPGA siano utilizzate in percentuali molto basse.

Resource	Utilization	Available	Utilization %
LUT	204	134600	0.15
FF	103	269200	0.04
IO	38	285	13.33



4. SIMULAZIONI

Per quanto riguarda i casi di test, sono stati scelti otto casi per testare nel modo più completo possibile il dispositivo:

1. Immagine di dimensioni zero;
2. Immagine composta da un solo pixel;
3. Immagine di dimensioni massime;
4. Immagine con valore massimo dei pixel;
5. Immagine con valore minimo dei pixel;
6. Immagine con delta massimo;
7. Reset asincrono;
8. Più elaborazioni consecutive;

Per tutti i casi di test si è scelto di utilizzare un periodo di clock di 10 ns al fine di verificare il corretto funzionamento anche con periodi di clock inferiori a quello minimo richiesto di 100 ns.

Per alcuni casi di test verrà inoltre riportato il grafico delle forme d'onda per la simulazione *Post-Synthesis Functional* a titolo illustrativo.

IMMAGINE DI DIMENSIONI ZERO

Controlla la condizione limite di un'immagine con dimensioni zero. Fornisce un'immagine con numero di colonne e/o di righe uguale a 0 e verifica quindi che l'esecuzione termini lasciando inalterata la memoria.

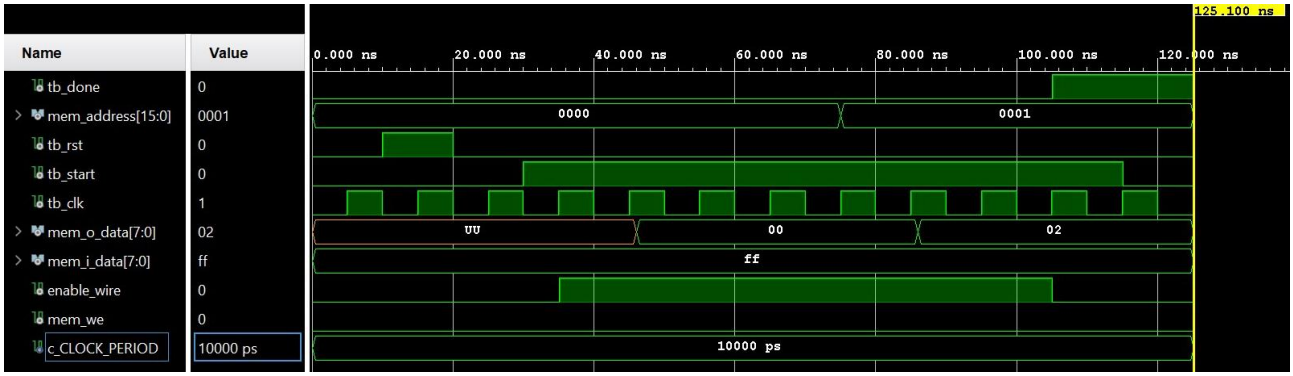


IMMAGINE COMPOSTA DA UN SOLO PIXEL

Verifica la condizione limite di un'immagine formata da un singolo pixel. Il test verifica che il pixel equalizzato sia quindi uguale a 0.

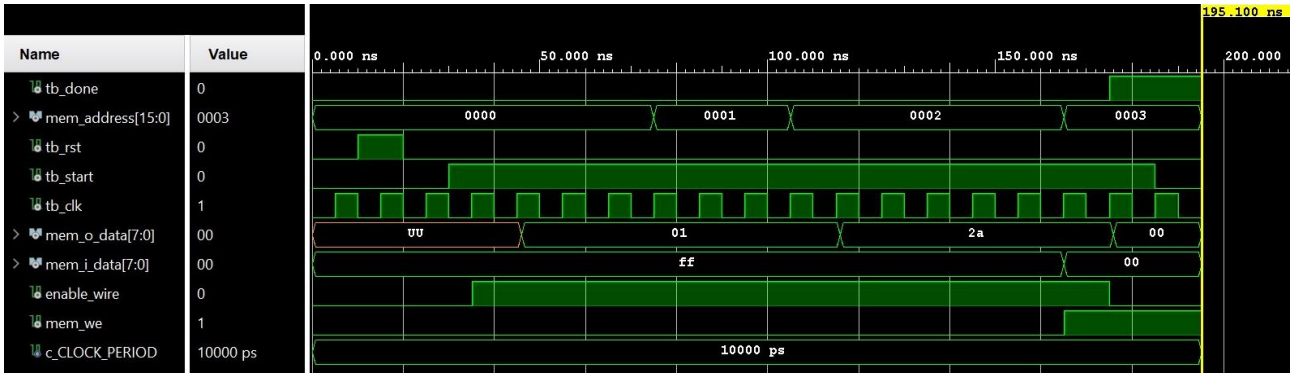


IMMAGINE DI DIMENSIONI MASSIME

Testa la condizione limite di un'immagine di dimensione massima, cioè 128x128 pixel. Il test verifica che tutta l'immagine sia equalizzata correttamente e scritta in memoria.

IMMAGINE CON VALORE MASSIMO DEI PIXEL

Controlla il caso limite di un'immagine composta da tutti i pixel con valore 255.

IMMAGINE CON VALORE MINIMO DEI PIXEL

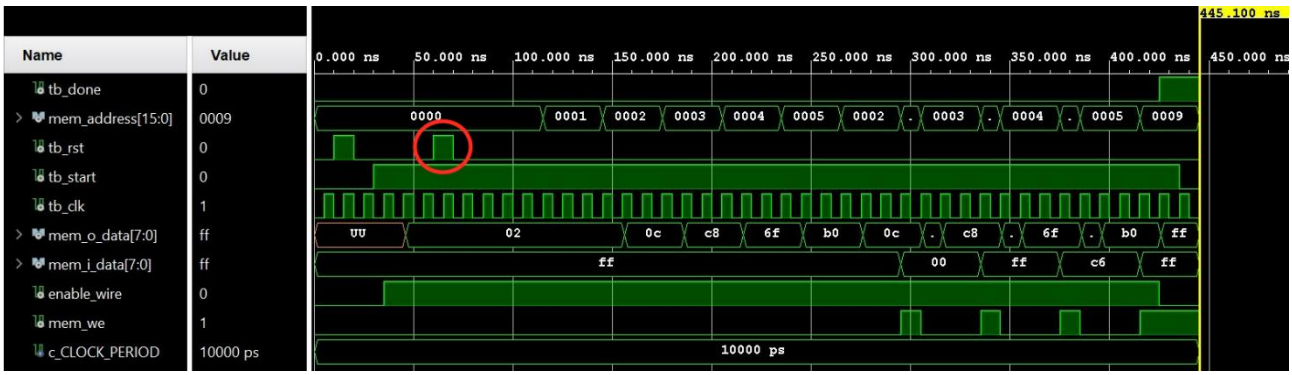
Controlla il caso limite di un'immagine composta da tutti i pixel con valore 0.

IMMAGINE CON DELTA MASSIMO

Verifica la condizione limite di un'immagine con un delta tra i pixel massimo, vale a dire contenente almeno un pixel con valore 0 e almeno uno con valore 255. L'immagine deve essere scritta in memoria inalterata, cioè "copiata" senza modificare il valore dei pixel.

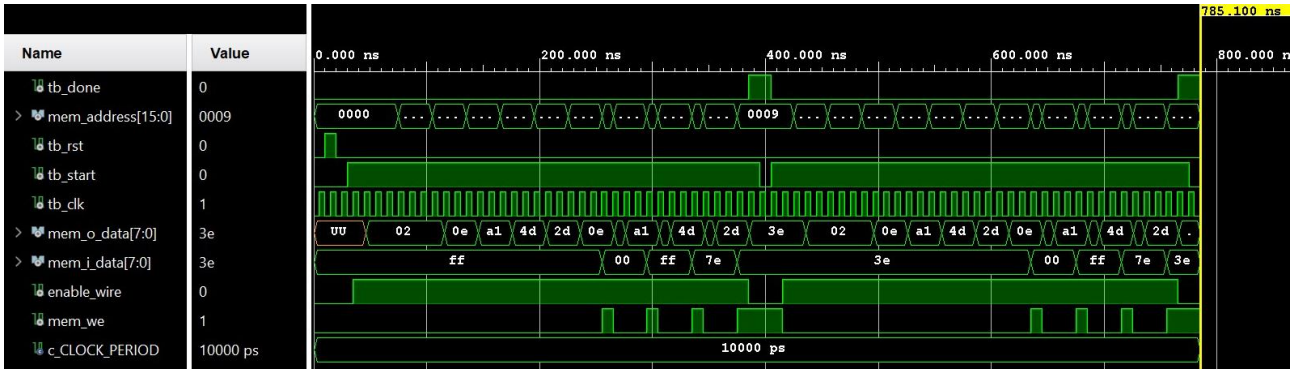
RESET ASINCRONO

Testa il caso in cui durante l'esecuzione venga alzato il segnale di RESET una seconda volta oltre al RESET iniziale. La macchina deve riportarsi allo stato RESET.



PIÙ ELABORAZIONI CONSECUTIVE

Verifica il corretto funzionamento nel caso di più elaborazioni consecutive. Ad ogni elaborazione conclusa il dispositivo alza il segnale di DONE, attende che quello di START si abbassi e si riporta allo stato di RESET, per poi iniziare una nuova elaborazione con una nuova salita del segnale di START. Nel caso sottostante sono state utilizzate due immagini.



Il componente è stato inoltre testato con numerosi test generati casualmente senza riscontrare errori.

5. CONCLUSIONI

Il componente sviluppato viene sintetizzato correttamente ed inoltre supera le simulazioni in *Behavioral* e *Post-Synthesis Functional* per tutti i casi di test con un periodo di clock inferiore a quello richiesto nelle specifiche. I tempi di esecuzione variano tra un minimo di 125.1 ns, nel caso di immagine con dimensioni 0, fino a 1.15 ms nel caso di dimensioni massime. Inoltre, nei casi di dimensioni medie (64x64) il tempo di esecuzione è 286.86 µs.

Si ritiene quindi di aver soddisfatto tutti i requisiti prediligendo un'architettura semplice ma, nonostante ciò, non meno efficace.