Progetto finale di reti logiche

AA 2017/2018

Documentazione

Componenti Gruppo:

- Luca Terracciano, Cod.Pers.: 10493487
- Manuel Trivilino, Cod.Pers.: 10503183

Obiettivo:

Implementare un componente HW descritto in VHDL che, data un'immagine in scala di grigi in un formato descritto successivamente, calcoli l'area del rettangolo minimo che circoscrive totalmente una figura di interesse, presente nell'immagine stessa.

Strumenti di Sintesi Utilizzati:

- XILINX VIVADO WEBPACK
- Target FPGA (xc7a200tfbg484-1)

Soluzione e Scelte implementative:

- Macchina a Stati Finiti (FSM)

Introduzione

La soluzione è stata progettata sulla base di una macchina a stati finiti che descrive il funzionamento e le diverse fasi dell'algoritmo implementato: dall'inizializzazione dei valori, alla lettura dei dati da memoria, fino al calcolo del risultato e alla sua scrittura in memoria.

Per quanto riguarda l'implementazione in VHDL sono stati utilizzati tre processi: il primo (state_sequence process) aggiorna ad ogni ciclo di clock (fronte di salita) lo stato corrente con lo stato prossimo; il secondo (lambda process) determina la sequenza di stati da seguire, assegna a next_state lo stato successivo in base allo stato corrente; l'ultimo processo (MAIN_PROCESS) contiene le istruzioni da eseguire nei diversi stati, ad ogni esecuzione, in corrispondenza del fronte di discesa del clock, viene eseguita la porzione di codice relativa allo stato corrente.

Descrizione Implementazione

L'algoritmo implementato fa uso di tre registri (column, lines, treshold) nei quali vengono salvati i valori di header dell'immagine: larghezza (numero colonne), altezza (numero righe) e valore di soglia, che vengono letti da memoria e salvati durante i primi e omonimi stati di esecuzione.

La lettura dei dati relativa all'immagine invece viene eseguita tramite l'ausilio di un contatore (column_counter_c) per l'incremento delle colonne e del registro lines per il decremento delle righe (vedi immagine 1). Inoltre si fa uso di ulteriori quattro registri (north, south, west, east) per salvare le coordinate parziali dei pixel di immagine che si trovano agli estremi dell'immagine: north e south salvano rispettivamente le righe con indice più alto e più basso mentre east e west le colonne più a destra e più a sinistra (immagine 3).

Terminata la lettura dell'immagine si procede al calcolo dell'area con i dati raccolti nei registri (immagine 3).

(n,0)	(n,1)	(n,2)	(n,…)	(n,m-1)	(n,m)
(n-1,0)	(n-1,1)	(n-1,2)	(n-1,…)	(n-1,m-1)	(n-1,m)
(1,0)	(1,1)	(1,2)	(1,…)	(1,m-1)	(1,m)
(0,0)	(0,1)	(0,2)	(0,…)	(0,m-1)	(0,m)

Immagine 1

		North(n)			
				East(m-1)	
	West(1)				
		South(0)			

Immagine 2: le celle in rosso rappresentano una figura in rilievo di cui bisogna calcolare l'area

		North			
				East	
	West				
		South			

Immagine 3: area complessiva da calcolare

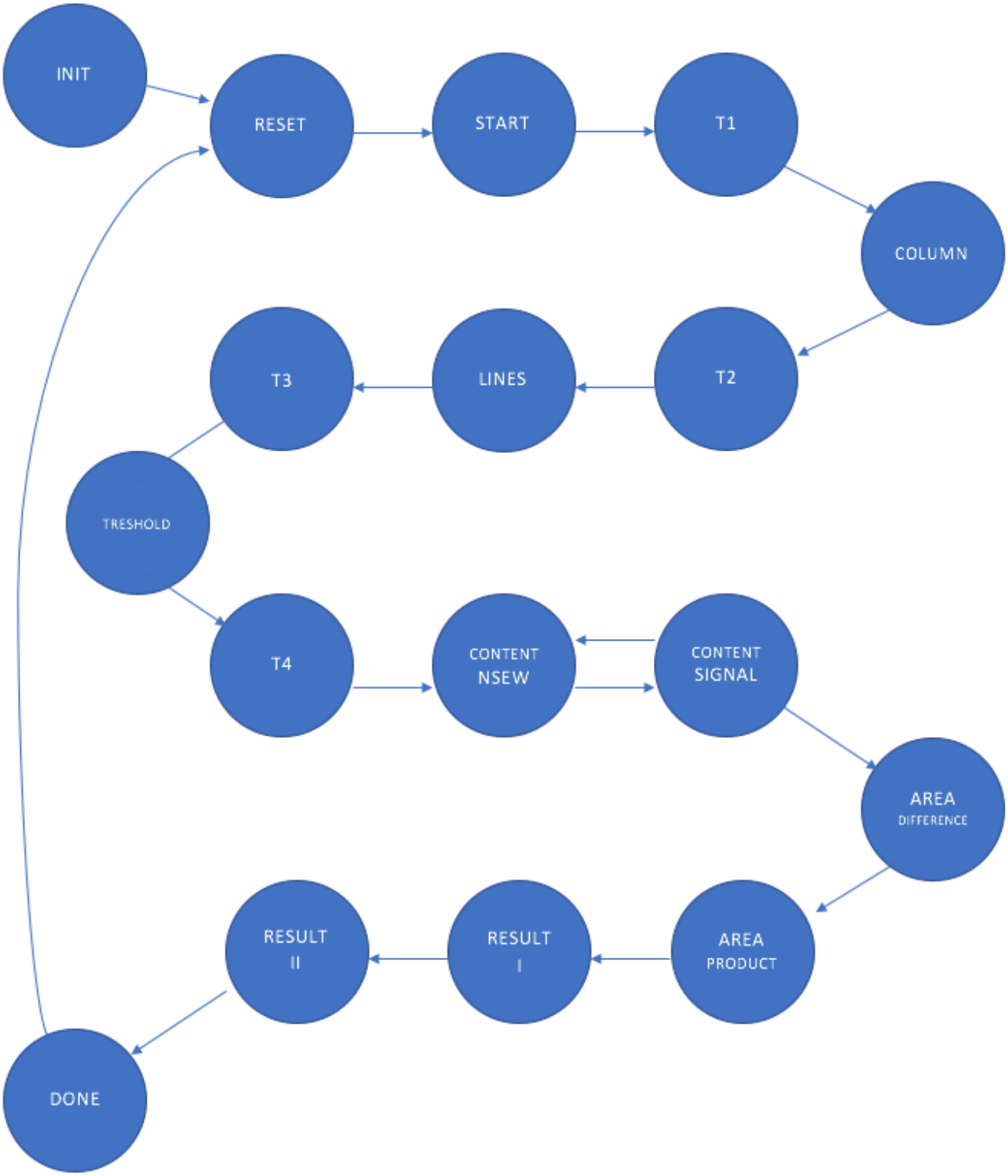
Descrizione Stati

- INIT: stato di inizializzazione delle variabili di stato, avviato il componente le variabili di stato vengono inizializzate a questo stato in attesa del segnale di RESET, dopo il primo reset non si torna più in questo stato.
- RESET: stato di reset, vengono inizializzati tutti i segnali con i valori iniziali necessari all'esecuzione della macchina, in attesa di ricevere il segnale di START.
- START: stato di start, attiva la memoria per dare il via all'esecuzione.
- T1: segnala l'indirizzo di lettura della memoria per iniziare la lettura del primo dato contenuto nell'header.
- COLUMN: legge il valore relativo al numero delle colonne, cioè alla larghezza in pixel dell'immagine e lo salva nel rispettivo registro.
- T2: incrementa l'indirizzo di memoria per la lettura delle righe.
- LINES: legge il valore relativo alle righe, cioè all'altezza in pixel dell'immagine e lo salva nel rispettivo registro.
- T3: incrementa l'indirizzo di memoria per la lettura della soglia.
- TRESHOLD: legge il valore relativo alla soglia e lo salva nel rispettivo registro.
- T4: incrementa l'indirizzo di memoria per la lettura del primo pixel dell'immagine.
- CONTENT_NSEW: viene letto il valore del pixel in posizione corrente (column_counter_c, lines), viene verificato se è parte dell'immagine o dello sfondo. Se è parte dell'immagine vengono effettuati i controlli per aggiornare le coordinate parziali (north, south, weast, east) relative al minimo rettangolo che inscrive l'immagine.
- CONTENT_SIGNAL: viene incrementato il contatore column_counter_c controllando che non sia uguale al valore contenuto in column: in tal caso azzero il contatore ,decremento lines e si ricomincia a contare fino a che anche lines non è uguale a 1 (che corrisponde alla fine dell'immagine).
- AREA_DIFFERENCE: viene eseguita la sottrazione tra le coordinate di riga (north, south) e le coordinate di colonna (weast, east) per calcolare

la base e l’altezza del rettangolo cercato.

- AREA_PRODUCT: viene calcolata l’area del rettangolo.
- RESULT_FIRST: viene restituita la parte meno significativa del risultato, gli 8 bit meno significativi (da 0 a 7) .
- RESULT_SECOND: viene restituita la parte più significativa del risultato, gli 8 bit più significativi (da 8 a 15).
- DONE: vengono spenti i pin della memoria e viene inviato il segnale di fine (done). Dopo ciò si torna nello stato di reset, in attesa di un nuovo segnale di start.

FSM


INIT
RESET
START
T1
COLUMN
T2
LINES
T3
TRESHOLD
T4
CONTENT NSEW
CONTENT SIGNAL
AREA DIFFERENCE
AREA PRODUCT
RESULT I
RESULT II
DONE

TEST

La soluzione è stata testata sia con i testbench forniti, sia con altri testbench creati per coprire le altre casistiche di test (casi con valori limite e altri casi generici non coperti dai testbench forniti). Sono stati eseguiti quindi 8 testbench forniti (i quattro originali e i quattro con i ritardi, forniti successivamente su beep) e 6 creati da noi.

Il componente passa correttamente tutti i testbench a livello di:

- Pre-sintesi
- Post-sintesi
- Post-implementazione funzionale

TEMPO DI COMPUTAZIONE:

Il tempo necessario alla computazione del caso pessimo (immagine di 255x255 pixel) è circa 2 s.

- Max Bound Time ~ 2 s

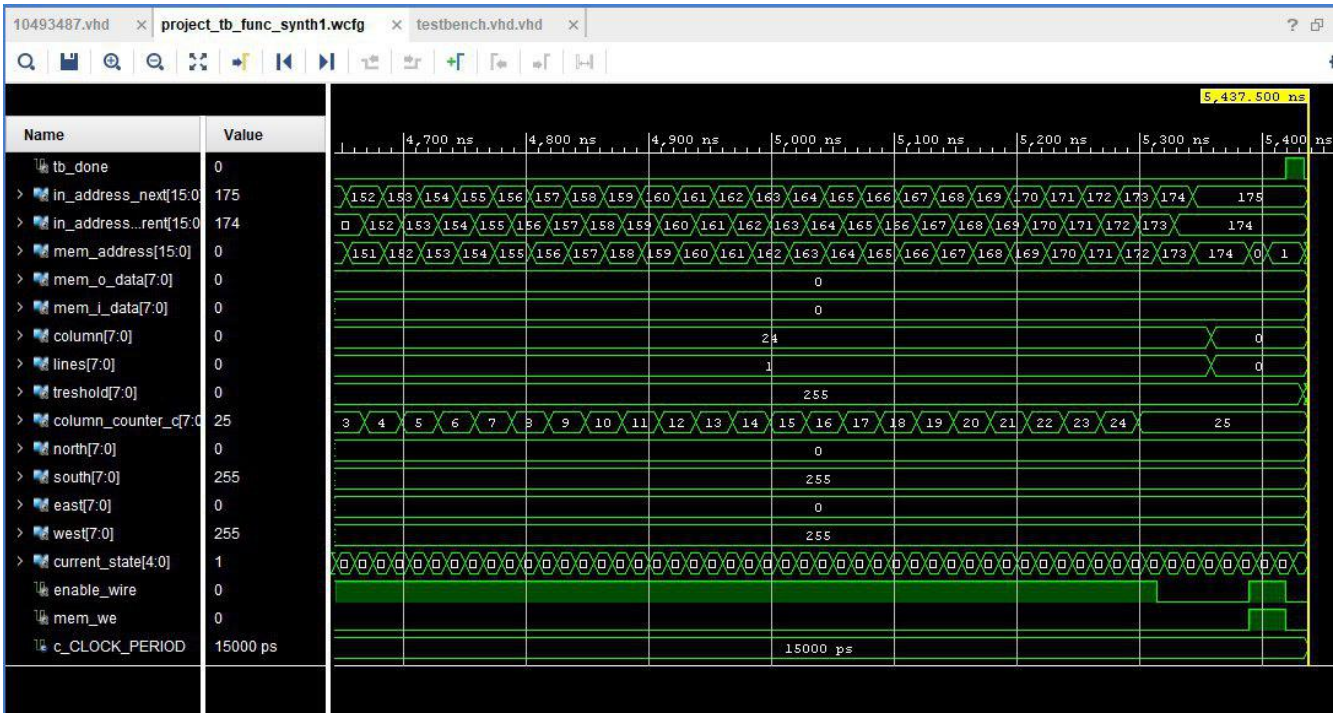
TESTBENCH FORNITI (1/8)

-Test 1:

10493487.vhd | project_tb_func_synth1.wcfg | testbench.vhd.vhd

C:/Users/terra/OneDrive - Politecnico di Milano/Progetto di Reti Logiche/testbench.vhd.vhd

```
92      end process;
93
94
95
96      test : process is
97      begin
98      wait for 100 ns;
99      wait for c_CLOCK_PERIOD;
100     tb_rst <= '1';
101     wait for c_CLOCK_PERIOD;
102     tb_rst <= '0';
103     wait for c_CLOCK_PERIOD;
104     tb_start <= '1';
105     wait for c_CLOCK_PERIOD;
106     tb_start <= '0';
107     wait until tb_done = '1';
108     wait until tb_done = '0';
109     wait until rising_edge(tb_clk);
110
111     assert RAM(1) = "00000000" report "FAIL high bits" severity failure;
112     assert RAM(0) = "00000000" report "FAIL low bits" severity failure;
113
114
115     assert false report "Simulation Ended!, test passed" severity failure;
116     end process test;
117
118     end projecttb;
119
```

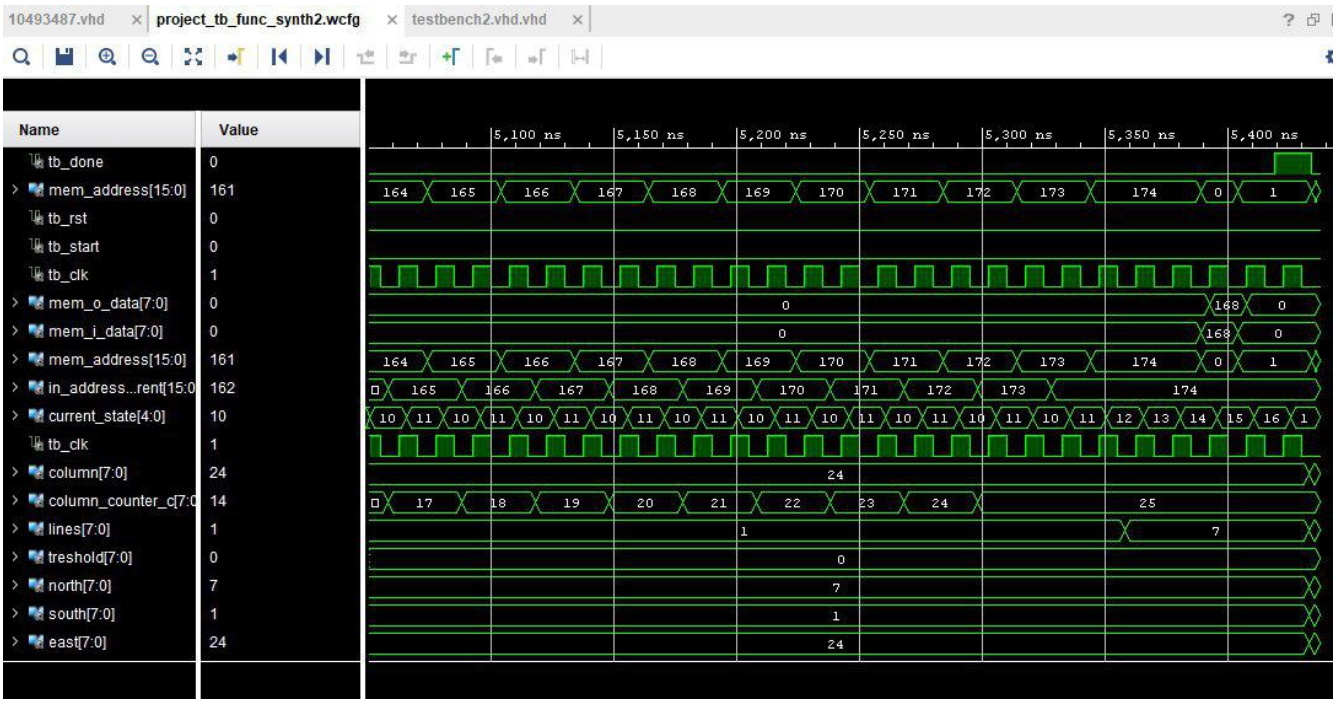


-Test 2:

10493487.vhd | project_tb_func_synth2.wcfg | testbench2.vhd.vhd

C:/Users/terra/OneDrive - Politecnico di Milano/Progetto di Reti Logiche/testbench2.vhd.vhd

```
92      end process;
93
94
95
96      test : process is
97      begin
98      wait for 100 ns;
99      wait for c_CLOCK_PERIOD;
100     tb_rst <= '1';
101     wait for c_CLOCK_PERIOD;
102     tb_rst <= '0';
103     wait for c_CLOCK_PERIOD;
104     tb_start <= '1';
105     wait for c_CLOCK_PERIOD;
106     tb_start <= '0';
107     wait until tb_done = '1';
108     wait until tb_done = '0';
109     wait until rising_edge(tb_clk);
110     assert RAM(1) = "00000000" report "FAIL high bits" severity failure;
111     assert RAM(0) = "10101000" report "FAIL low bits" severity failure;
112
113
114     assert false report "Simulation Ended!, test passed" severity failure;
115
116     end process test;
117
118     end projecttb;
```

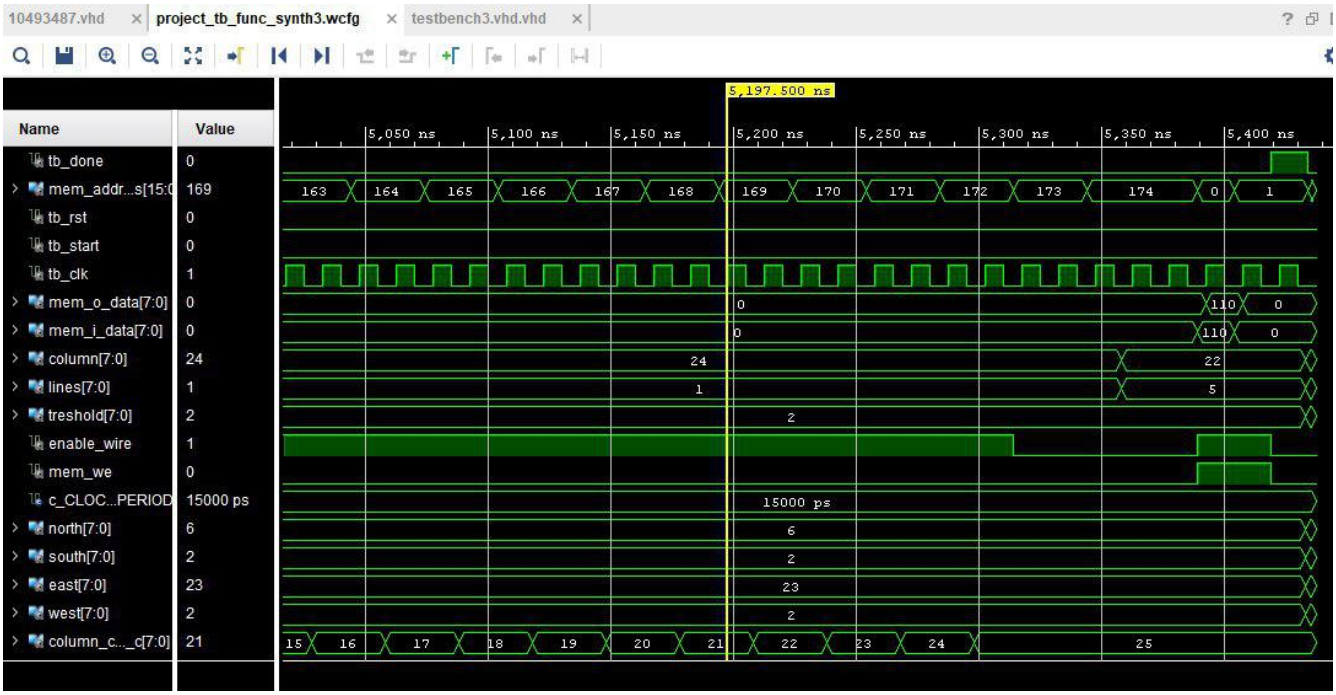


-Test 3:

10493487.vhd | project_tb_func_synth3.wcfg | testbench3.vhd.vhd

C:/Users/terra/OneDrive - Politecnico di Milano/Progetto di Reti Logiche/testbench3.vhd.vhd

```
92      end if;
93    end process;
94
95
96    test : process is
97    begin
98    wait for 100 ns;
99    wait for c_CLOCK_PERIOD;
100   tb_rst <= '1';
101   wait for c_CLOCK_PERIOD;
102   tb_rst <= '0';
103   wait for c_CLOCK_PERIOD;
104   tb_start <= '1';
105   wait for c_CLOCK_PERIOD;
106   tb_start <= '0';
107   wait until tb_done = '1';
108   wait until tb_done = '0';
109   wait until rising_edge(tb_clk);
110
111   assert RAM(1) = "00000000" report "FAIL high bits" severity failure;
112   assert RAM(0) = "01101110" report "FAIL low bits" severity failure;
113
114
115   assert false report "Simulation Ended!, test passed" severity failure;
116   end process test;
117
118   end projecttb;
```

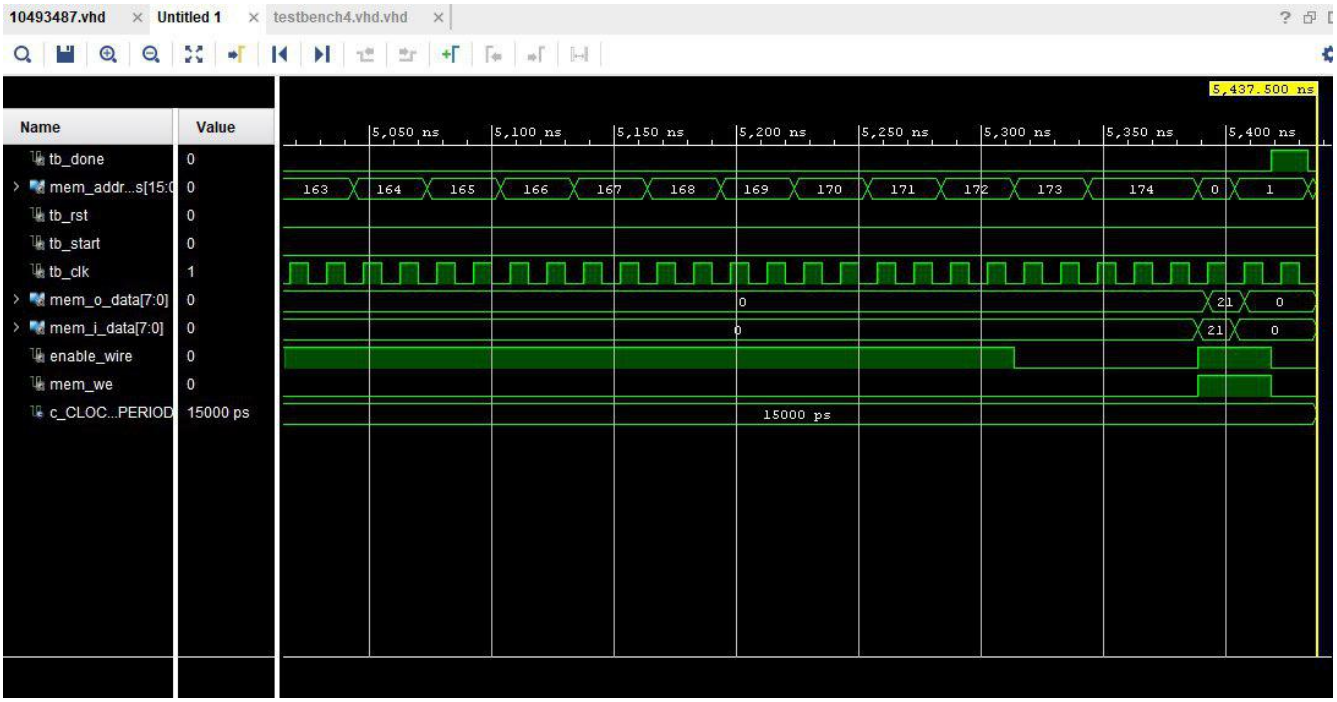


-Test 4:

10493487.vhd | Untitled 1 | testbench4.vhd.vhd

C:/Users/terra/OneDrive - Politecnico di Milano/Progetto di Reti Logiche/testbench4.vhd.vhd

```
91      end if;
92    end process;
93
94
95    test : process is
96    begin
97    wait for 100 ns;
98    wait for c_CLOCK_PERIOD;
99    tb_rst <= '1';
100   wait for c_CLOCK_PERIOD;
101   tb_rst <= '0';
102   wait for c_CLOCK_PERIOD;
103   tb_start <= '1';
104   wait for c_CLOCK_PERIOD;
105   tb_start <= '0';
106   wait until tb_done = '1';
107   wait until tb_done = '0';
108   wait until rising_edge(tb_clk);
109
110   assert RAM(1) = "00000000" report "FAIL high bits" severity failure;
111   assert RAM(0) = "00010101" report "FAIL low bits" severity failure;
112
113
114   assert false report "Simulation Ended!, test passed" severity failure;
115   end process test;
116
117   end projecttb;
118
```



-Test 5/8:

Ulteriori testbench forniti, uguali ai primi quattro, con l'aggiunta di una componente di delay.

TESTBENCH AGGIUNTIVI (9/14)

-Test 9:

N° Colonne: 1

N° Righe: 1

Soglia: 255

Immagine formata da una sola cella con valore di 255 posizionata in (0,0).

Risultato atteso: 1

-Test 10:

N° Colonne: 24

N° Righe: 7

Soglia: 255

Immagine formata da una sola cella con valore 255 in posizione (0,0).

Risultato atteso: 1

-Test 11:

N° Colonne: 24

N° Righe: 7

Soglia: 255

Immagine formata da due elementi di valore 255 in posizione (0,0) e (6,0).

Risultato atteso: 7

-Test 12:

N° Colonne: 24

N° Righe: 7

Soglia: 255

Immagine formata da due elementi di valore 255 in posizione (0,0) e (0,23).

Risultato atteso: 24

-Test 13:

N° Colonne: 24

N° Righe: 7

Soglia: 255

Immagine formata da due elementi di valore 255 in posizione (0,0) e (6,23).

Risultato atteso: 168

-Test 14:

N° Colonne: 255

N° Righe: 255

Soglia: 255

Immagine formata da due elementi d valore 255 in posizione (0,0) e (254,254).

Risultato atteso: 65 025

Link ai Testbench aggiuntivi:

https://drive.google.com/drive/folders/1ZHzzul1q3wWhw9oPhvZg38MZyKVrZJwy?usp=sharing