Politecnico di Milano

CORSO DI RETI LOGICHE – PROF. SALICE FABIO

PROGETTO DI RETI LOGICHE

Simone Gabrielli | 10679234 | A.A. 2020-2021

INTRODUZIONE

DESCRIZIONE GENERALE

La specifica della Prova finale (Progetto di Reti Logiche) 2020 è ispirata al metodo di equalizzazione dell'istogramma di una immagine. Il metodo di equalizzazione dell'istogramma di una immagine è un metodo pensato per ricalibrare il contrasto di una immagine quando l'intervallo dei valori di intensità sono molto vicini effettuandone una distribuzione su tutto l'intervallo di intensità, al fine di incrementare il contrasto.

fig.1 - Esempi di immagine pre e post equalizzazione (sorgente Wikipedia)

Il modulo da implementare dovrà leggere l'immagine da una memoria in cui è memorizzata, sequenzialmente e riga per riga, l'immagine da elaborare. Ogni byte corrisponde ad un pixel dell'immagine. La dimensione della immagine è definita da 2 byte, memorizzati a partire dall'indirizzo 0. Il byte all'indirizzo 0 si riferisce alla dimensione di colonna; il byte nell'indirizzo 1 si riferisce alla dimensione di riga. La dimensione massima dell'immagine è 128x128 pixel. L'immagine è memorizzata a partire dall'indirizzo 2 e in byte contigui. Quindi il byte all'indirizzo 2 è il primo pixel della prima riga dell'immagine.

L'immagine equalizzata deve essere scritta in memoria immediatamente dopo l'immagine originale

ANALISI DELLA SPECIFICA

Il processo di equalizzazione dell'immagine richiederà una sequenza di operazioni che possono essere raggruppate in 4 fasi:

1. *Preparazione alla ricezione dell'immagine*
   Nella prima parte dell'esecuzione il componente attenderà di un segnale di avvio (i_start) che avvierà una nuova equalizzazione. Alla sua ricezione verrà avviata la procedura di preparazione alla ricezione, in cui verranno letti dalla memoria il numero di righe e quello di colonne e saranno collocati in due registri adibiti.

2. *Individuazione degli estremi*
   Tramite il prodotto delle righe per le colonne, ricavate nel passaggio precedente, il modulo potrà scorrere i pixel dell'immagine fino al loro termine: in tal modo sarà in grado di identificare e registrare il pixel con valore intero maggiore e il minore.

3. *Equalizzazione*
   Una volta individuati gli estremi si procede alla vera e propria equalizzazione dell'immagine: il componente leggerà di nuovo tutti i pixel dell'immagine e, tramite un processo specifico descritto in seguito, per ognuno di essi calcolerà il pixel equalizzato per poi scriverlo in memoria.

4. *Reset*
   Al termine della procedura il dispositivo restituirà un segnale che indica il completamento della procedura (o_done) e resterà in attesa di un nuovo segnale di avvio, che ne reimposterà la condizione interna e comincerà una nuova equalizzazione.

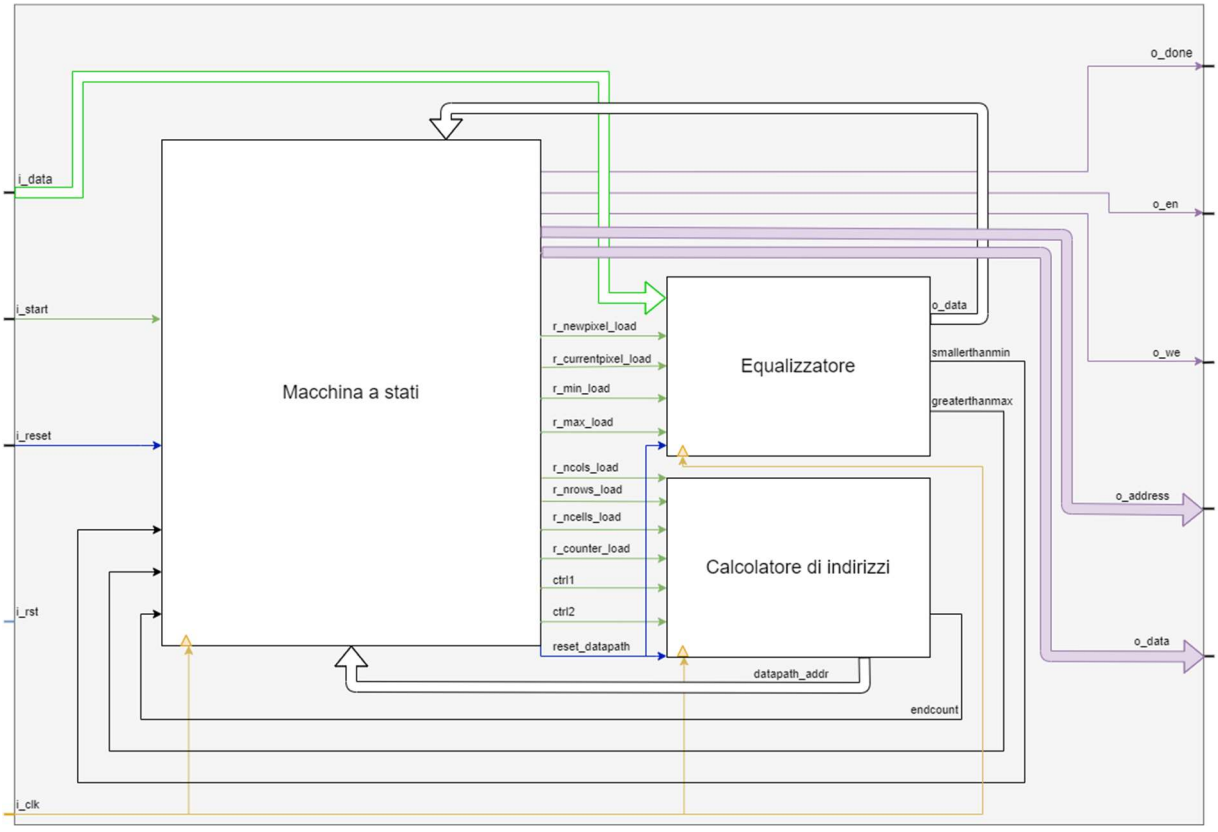
Le singole sottofasi che compongono le quattro fasi saranno suddivise assegnate ai vari stati di una Macchina a Stati interna, mentre i calcoli necessari saranno svolti da due componenti: l'*equalizzatore* e il *calcolatore di indirizzi*. Il loro funzionamento è spiegato in seguito.

ARCHITETTURA

PROJECT_RETI_LOGICHE

Come richiesto dalle specifiche è stato realizzato un modulo denominato *project_reti_logiche* con la seguente interfaccia:

```
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;
```



- Descrizione Funzionale:

  Il modulo è composto di tre componenti logiche principali: la macchina a stati, l'*equalizzatore* e il *calcolatore di indirizzi*.
  La macchina a stati si occupa di calcolare, in base ai parametri in input, lo stato successivo. Il registro contenente lo stato corrente sarà aggiornato al successivo ciclo di clock con lo stato successivo calcolato.

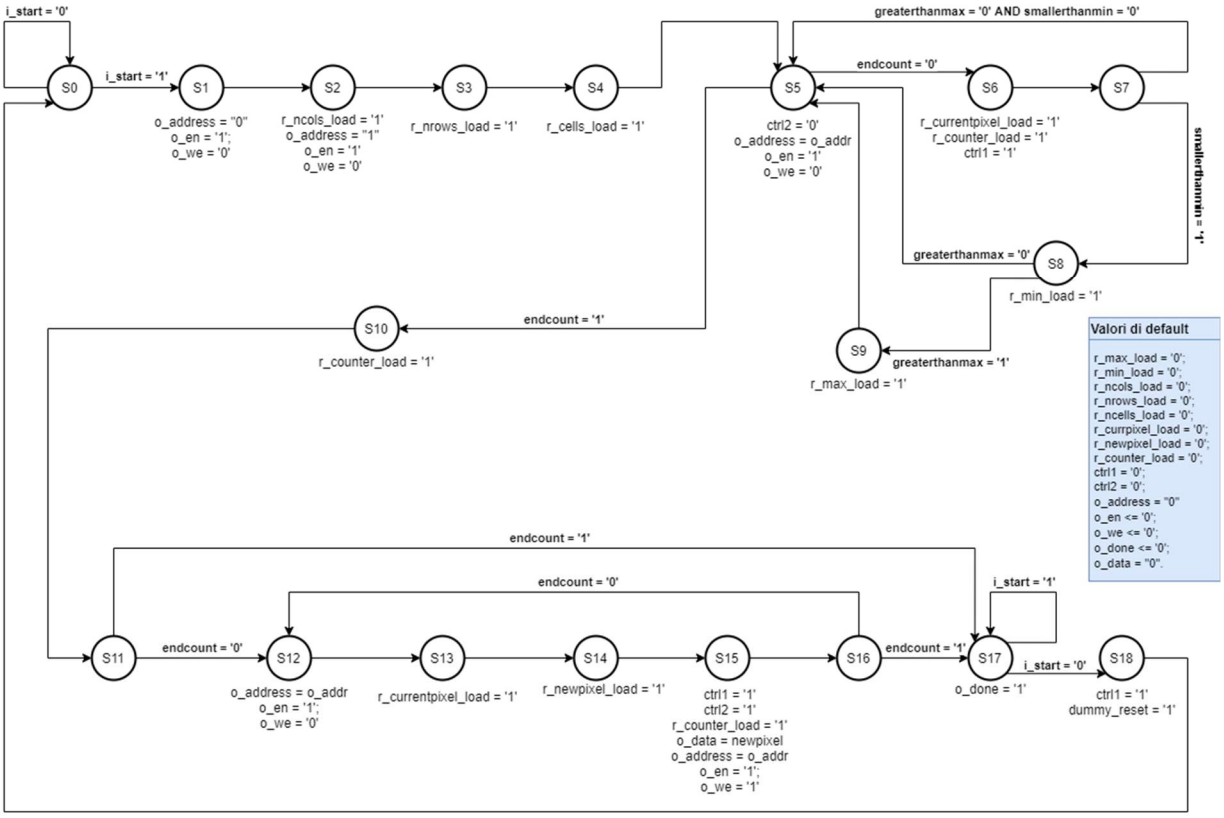
In base allo stato attuale verranno generati dei valori per dei segnali di controllo sia per l'*equalizzatore*, sia per il *calcolatore di indirizzi*.
Questi ultimi, grazie ai segnali generati dalla macchina a stati, modificano le loro impostazioni interne e generano così i segnali in uscita richiesti, che saranno poi nuovamente "filtrati" e rielaborati dalla macchina a stati.

MACCHINA A STATI

La macchina a stati prevede 19 stati, che eseguono le 4 fasi descritte in precedenza.

- Input:
  - **i_start**: indica l'avvio di una equalizzazione;
  - **i_rst**: resetta lo stato di States Machines a So e le condizioni dei registri;
  - **i_clk:** il clock del sistema;
  - **greaterthanmax**: segnale che indica se la cella di memoria appena letta contiene un valore maggiore rispetto alla più grande identificata finora; utile nella fase di *Individuazione degli estremi.*
  - **smallerthanmin**: segnale che indica se la cella di memoria appena letta contiene un valore minore rispetto alla più piccola identificata finora; utile nella fase di *Individuazione degli estremi.*
  - **endcount**: segnale che indica se abbiamo concluso la lettura dell'immagine.

- Output:
  - **o_done**: indica la terminazione dell'equalizzazione corrente
  - **o_en**: apre la comunicazione con la memoria;
  - **o_we**: attiva la modalità scrittura della memoria;
  - **o_address**: indica l'indirizzo di memoria su cui effettuare l'operazione;
  - **o_data:** il dato da scrivere in memoria, ossia il valore del pixel equalizzato;
  - **r_newpixel_load**: quando posto a 1 il registro newpixel salverà il valore in ingresso;
  - **r_currpixel_load**: quando posto a 1 il registro currpixel salverà il valore in ingresso;
  - **r_min_load**: quando posto a 1 il registro min salverà il valore in ingresso;
  - **r_max_load**: quando posto a 1 il registro max salverà il valore in ingresso;
  - **r_ncols_load**: quando posto a 1 il registro ncols salverà il valore in ingresso;
  - **r_nrows_load**: quando posto a 1 il registro nrows salverà il valore in ingresso;
  - **r_ncells_load**: quando posto a 1 il registro ncells salverà il valore in ingresso;
  - **r_counter_load**: quando posto a 1 il registro counter salverà il valore in ingresso;
  - **ctrl1**: indica l'ingresso che il multiplexer che precede il contatore del *calcolatore di indirizzi* dovrà restituire in output;
  - **ctrl2**: indica l'ingresso che il multiplexer degli indirizzi del datapath dovrà restituire in output;
  - **reset_datapath**: effettua il reset interno sia dell'*equalizzatore* sia del *calcolatore di indirizzi* viene posto a 1 sia quando ricevo un reset dall'esterno, sia appena finita l'equalizzazione, permettendo così al sistema di prepararsi per la prossima equalizzazione.

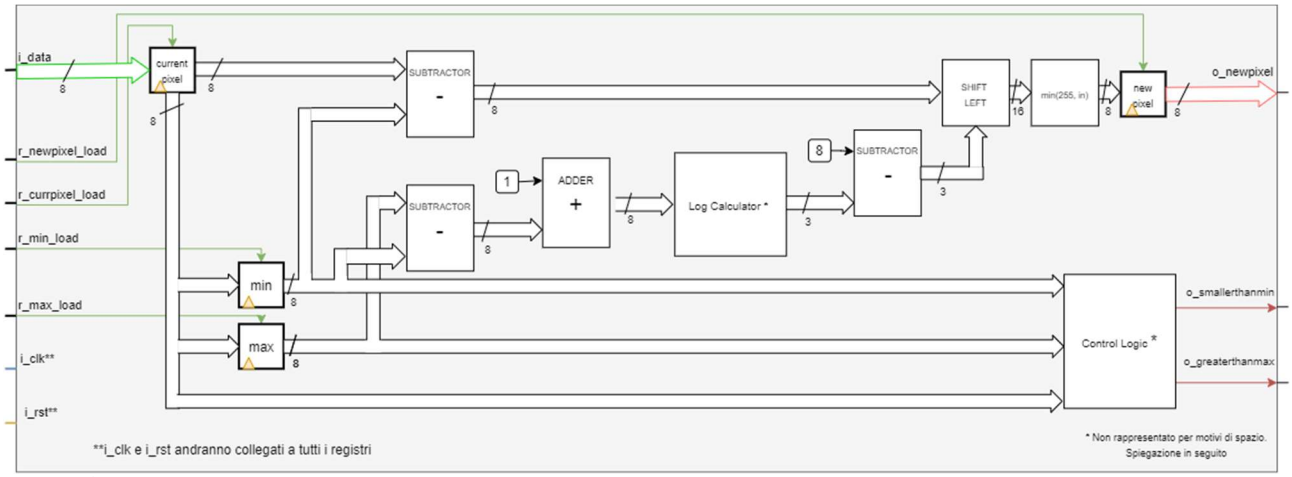


- Descrizione Funzionale:
  1. **Gli stati da So a S4** realizzano la fase di *preparazione alla ricezione dell'immagine:*
     Si attende l'input che indicherà l'avvio della procedura (i_start=1), dopodichè si procede a leggere dalla memoria il numero di righe e il numero di colonne (rispettivamente negli indirizzi 0 e 1). Con questi nuovi dati sarà possibile calcolare il numero di celle totali.
  2. **Gli stati da S5 a S9** realizzano la fase di *Individuazione degli estremi:*
     vengono letti uno ad uno i pixel dell'immagine e a sua volta l'*equalizzatore*, tramite i segnali greaterthanmax e smallerthanmin, indica se il pixel appena letto è maggiore o minore rispetto a tutti quelli letti in precedenza e qualora il pixel letto rientri in uno di questi due casi, viene salvato nel rispettivo registro (max o min). L'operazione si ripete finchè endcount assume valore 1; a questo punto si passa alla fase successiva.
  3. **Gli stati da S10 a S16** realizzano la fase di *Equalizzazione:*
     il contatore viene azzerato e si ricomincia a leggere tutti i pixel. L'*equalizzatore* si occuperà dell'equalizzazione vera e propria, mentre il *calcolatore di indirizzi* restituirà il pixel equalizzato e l'indirizzo su cui scrivere. Quando la lettura sarà terminata, ossia quando endcount restituirà il valore 1, avremo terminato l'equalizzazione.
  4. **Gli stati S17 e S18** realizzano la fase di *Reset:*
     viene restituito un segnale o_done dal valore 1, che indica al modulo esterno il termine dell'operazione, dopodichè viene inviato un segnale di reset sia

all'*equalizzatore* sia al *calcolatore di indirizzi* (tramite reset_datapath), così da prepararli per un'eventuale nuova equalizzazione.

EQUALIZZATORE

È il componente che realizza la vera e propria equalizzazione.



- Input:
  - **i_start**: indica l'avvio di una equalizzazione;
  - **i_rst**: resetta le condizioni dei registri;
  - **i_clk:** il clock del sistema;
  - **i_data:** (8 bit) il dato letto da memoria;
  - **r_newpixel_load**: segnale di enable del registro newpixel: quando posto a 1 il registro newpixel salverà il valore in ingresso;
  - **r_currpixel_load**: segnale di enable del registro currpixel: quando posto a 1 il registro currpixel salverà il valore in ingresso;
  - **r_min_load**: segnale di enable del registro min: quando posto a 1 il registro min salverà il valore in ingresso;
  - **r_max_load**: segnale di enable del registro max: quando posto a 1 il registro max salverà il valore in ingresso;

- Output
  - **o_data:** il dato da scrivere in memoria, ossia il valore del pixel equalizzato;
  - **o_greaterthanmax**: segnale che indica se la cella di memoria appena letta contiene un valore maggiore rispetto alla più grande identificata finora; utilizzato per definire il next state nella macchina a stati nella fase di *Individuazione degli estremi.*
  - **o_smallerthanmin**: segnale che indica se la cella di memoria appena letta contiene un valore minore rispetto alla più piccola identificata finora; utilizzato per definire il next state nella macchina a stati nella fase di *Individuazione degli estremi.*

- Descrizione Funzionale:
  L'*equalizzatore* svolge ha un ruolo importante nelle fasi di *Individuazione degli estremi* e di *equalizzazione.*

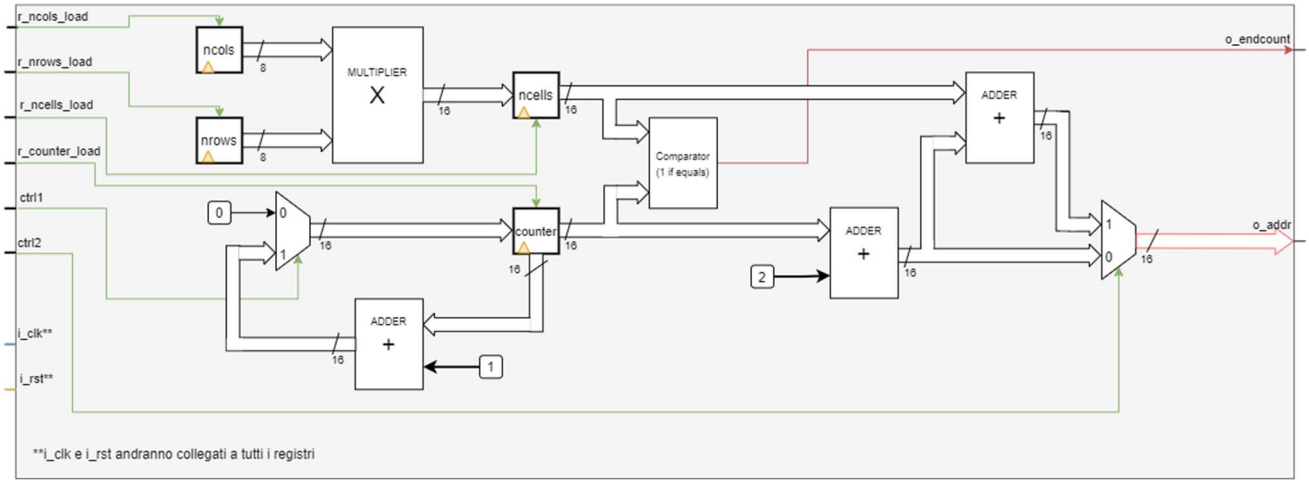
Nella fase di *Individuazione degli estremi* controlla il valore di tutti i pixel e a mano a mano li confronta con il massimo e il minimo trovati fino a quel momento; quando individua un nuovo massimo (minimo) lo segnala alla macchina a stati tramite o_greaterthanmax (o_smallerthanmin), che provvederà a settare lo stato interno dell'*equalizzatore* in modo da aggiornare il valore del registro Max (Min).
Nella fase di *equalizzazione* invece riceve un pixel per volta e si occupa di equalizzarlo in un unico ciclo di clock; nel ciclo successivo, tramite il registro new_pixel metterà a disposizione il valore del pixel equalizzato, della cui scrittura si occuperà la macchina a stati.
Lo stato dei registri verrà poi pulito nella fase di *reset*.

- Log Calculator
  È un semplice controllo a soglie: [Input Range]→Output
  [0,1]→0; [4,7]→2; [16,31]→4; [64,127]→6;
  [2,3]→1; [8,15]→3; [32,63]→5; [128,255]→7;

- Control Logic
  Una coppia di comparatori: se il valore di currpixel è maggiore del max attuale o_greaterthanmax vale 1, altrimenti vale 0.
  Analogamente se il valore di currpixel è minore del min attuale o_smallerthanmin vale 1, altrimenti vale 0.

CALCOLATORE DI INDIRIZZI

È il componente che calcola gli indirizzi delle celle di memoria da cui leggere e scrivere



- Input:
  - **i_rst**: resetta le condizioni dei registri;
  - **i_clk:** il clock del sistema;
  - **r_nrows_load**: segnale di enable del registro nrows: quando posto a 1 il registro nrows salverà il valore in ingresso;
  - **r_ncols_load**: segnale di enable del registro ncols: quando posto a 1 il registro ncols salverà il valore in ingresso;

- **r_ncells_load**: segnale di enable del registro ncells: quando posto a 1 il registro ncells salverà il valore in ingresso;
- **r_counter_load**: segnale di enable del registro counter: quando posto a 1 il registro counter salverà il valore in ingresso;
- **ctrl1:** controlla il multiplexer del counter; quando vale 1 il valore in ingresso al registro sarà il suo valore attuale +1, quando vale 0 il valore in ingresso al registro sarà un array di 0;
- **ctrl2:** controlla il multiplexer di o_addr; quando vale 1 verà restituito in output l'indirizzo della cella di memoria in cui scrivere (address_write = indirizzo lettura + ncells), quando vale 0 verà restituito l'indirizzo della cella di memoria da leggere (address_read = counter + 2).

❖ Output
- **o_addr:** (16 bit) l'indirizzo del dato da leggere/scrivere.
- **o_endcount**: indica che il contatore ha raggiunto il valore di ncells, la lettura dei pixel è giunta al termine.

❖ Descrizione Funzionale:
A differenza dell'*equalizzatore* il *calcolatore di indirizzi* interviene in tutte le fasi: Nella *preparazione* salva nei registri adibiti il numero di righe, di colonne e di celle totali. Nella fase di *Individuazione degli estremi* si occuperà di aggiornare un contatore interno e restituire l'indirizzo del nuovo pixel da leggere, fino a quando il contatore avrà raggiunto il valore del numero di celle totali (+2); in tal caso il segnale endcount verrà alzato a 1.
Nella fase di *Equalizzazione* il contatore viene resettato e si ricomincia la lettura, ma stavolta il *calcolatore di indirizzi* restituisce alternativamente l'indirizzo da cui leggere il pixel da equalizzare (contatore + 2) e l'indirizzo in cui andremo a scriverlo (contatore + celle totali + 2) una volta processato.
Lo stato dei registri verrà poi pulito nella fase di *reset*.

RISULTATI SPERIMENTALI

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	215	0	134600	0.16
LUT as Logic	215	0	134600	0.16
LUT as Memory	0	0	46200	0.00
Slice Registers	99	0	269200	0.04
Register as Flip Flop	99	0	269200	0.04
Register as Latch	0	0	269200	0.00
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

I registri utilizzati sono 99, nessuno dei quali è un latch. L'assenza di latch indica che tutti i registri sono sincroni e vengono aggiornati solo in presenza di uno specifico valore di ingresso e solo al rising edge del clock

Slack (MET): 93.855 ns (required time - arrival time)

In base al tempo di slack è possibile stabilire che la frequenza a cui opera la macchina rientra abbondantemente nel margine dei 100ns del periodo di clock.

SIMULAZIONI

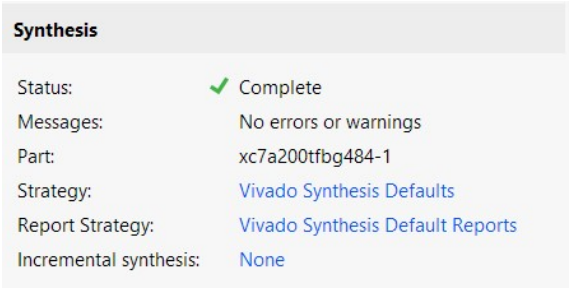
Tutti i testbench proposti sono stati superati entro i tempi stabiliti sia nelle simulazioni behavioral, sia nelle simulazioni post-sintesi.

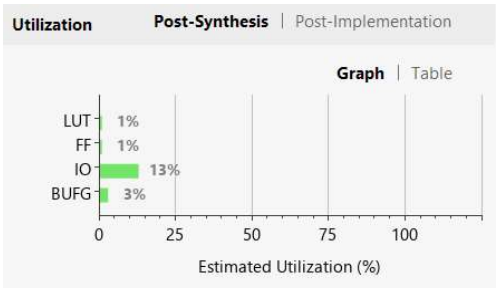
In particolare sono stati effettuati test sia con una singola immagine, sia con più immagini in sequenza.

Sono stati inoltre verificati, tramite opportuni testbench, i seguenti casi limite:

CASO LIMITE	NOTE
Immagine 0x0	Completato in 12 cicli di clock
Immagine 0x1	Completato in 12 cicli di clock
Immagine 1x0	Completato in 12 cicli di clock
Immagine 128x128	Necessari circa 2ms complessivi (con periodo di clock = 15ns)
Max pixel = 255, Min pixel =255	
Max pixel = 0, Min pixel =0	
Max pixel = 255, Min pixel =0	
Max pixel = 1, Min pixel =0	

CONCLUSIONI





Il codice risulta funzionante nei modi e tempi previsti dal punto di vista Behavioral; allo stesso modo la sintesi sembra essere efficiente ed efficace.