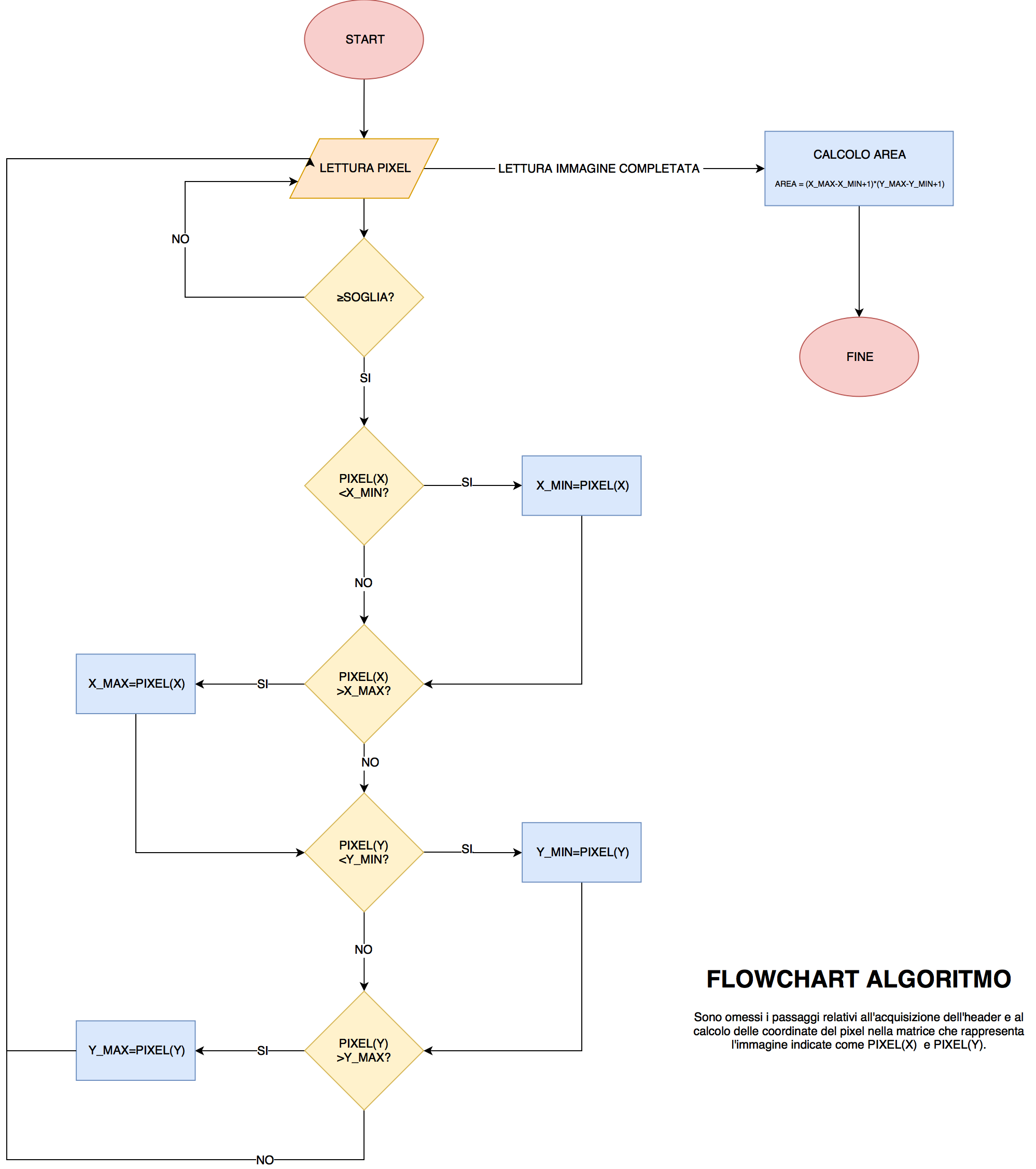
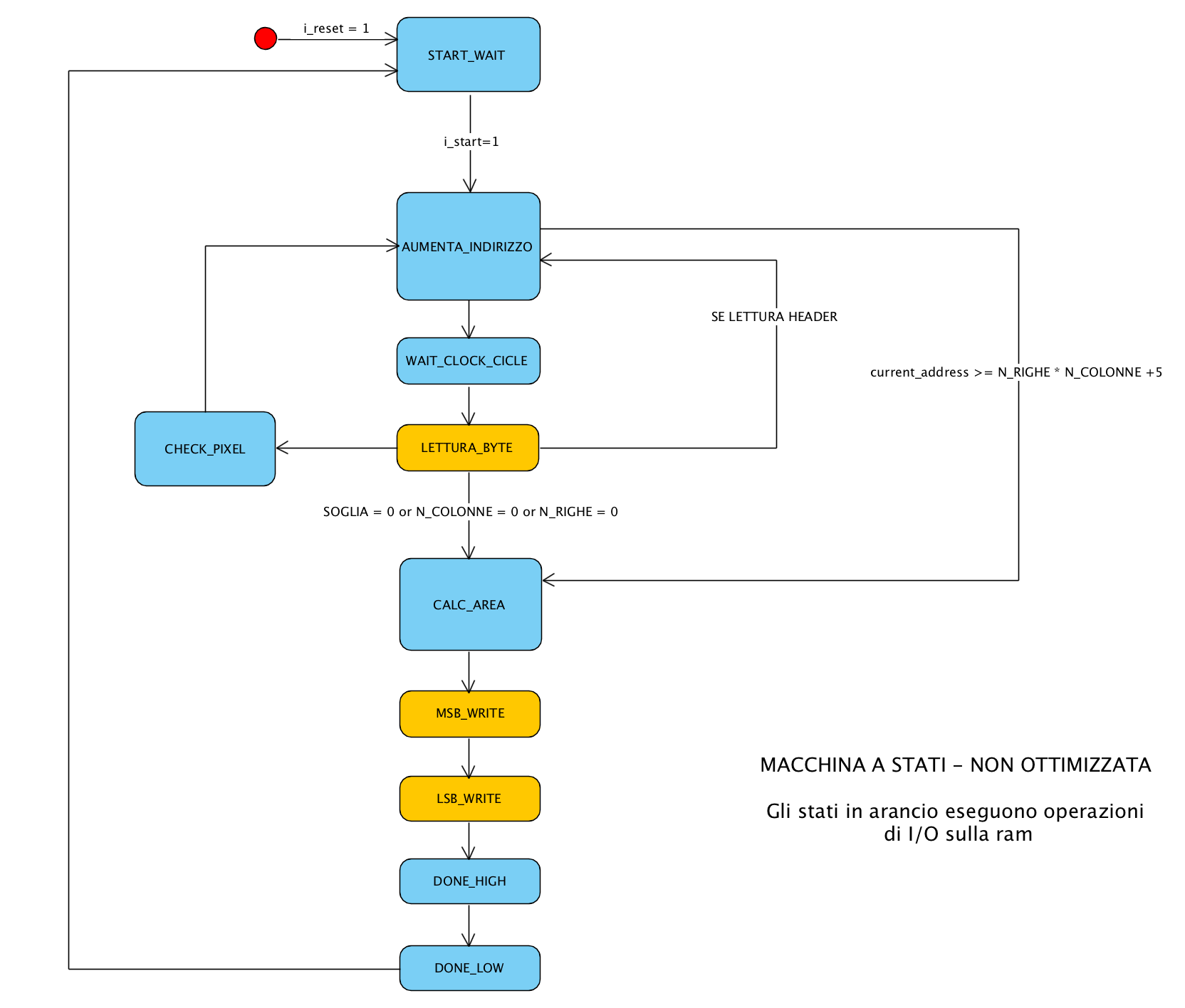
DESCRIZIONE ALGORITMO



IMPLEMENTAZIONE IN VHDL

Completata la fase di progettazione dell’algoritmo si è passati alla progettazione del componente vero e proprio in VHDL. La scelta del pattern di design è naturalmente ricaduta sul behavioural. Per poter ottenere un design sintetizzabile si è prima di tutto scomposto l’algoritmo in una macchina a stati finiti:



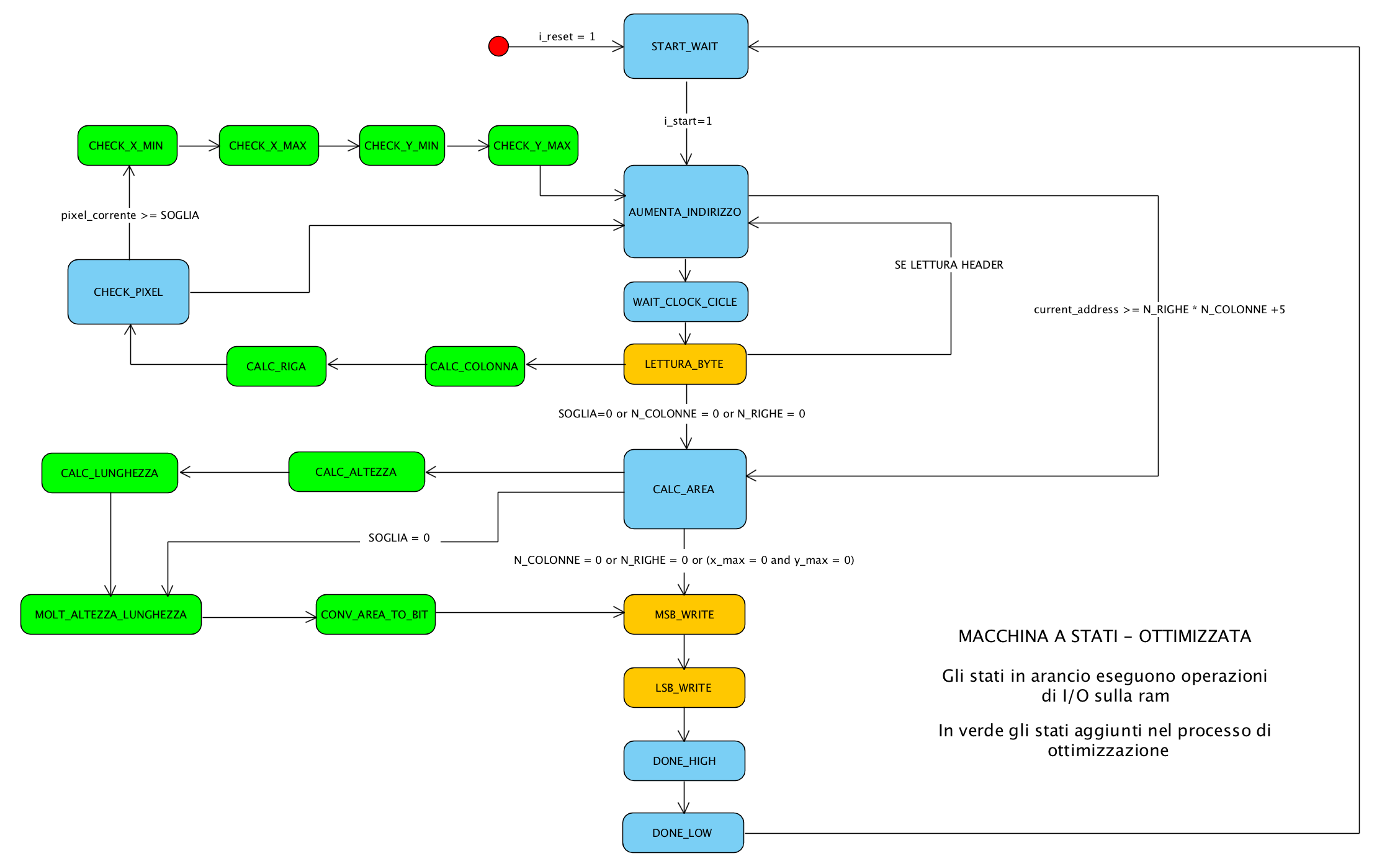
Ricevuto il segnale di start il componente entra nel primo stato, START\_WAIT, dove resta fino al segnale di start. Lo stato successivo, AUMENTA\_INDIRIZZO, si occupa di stabilire l’indirizzo della RAM da cui lo stato LETTURA\_BYTE andrà successivamente a leggere e di abilitare la lettura da memoria settando gli opportuni segnali. Tra i due è posto uno stato “vuoto” necessario per permettere alla memoria di mandare i dati. Ottenuto il pixel, questo viene processato dallo stato CHECK\_PIXEL che si occupa di stabilire se è maggiore o uguale della soglia e, in tal caso, dopo aver calcolato la riga e colonna del pixel, di controllare se è uno dei massimi/minimi utilizzati dall’algoritmo.

A questo punto il controllo ritorna ad AUMENTA\_INDIRIZZO che procede a selezionare il pixel successivo o, in caso si sia raggiunta la fine dell’immagine, a passare il controllo allo stato CALC\_AREA. Quest’ultimo, calcolata l’area, procede verso gli stati successivi, MSB\_WRITE e LSB\_WRITE, i quali effettuano la scrittura in memoria rispettivamente del bit più significativo e del meno significativo. Infine, gli stati DONE\_HIGH e DONE\_LOW alzano il segnale o\_done per un ciclo di clock, dopodiché il componente ritorna in attesa del segnale di start nello stato START\_WAIT.

La macchina così come descritta è stata implementata utilizzando come architecture del componente un processo che ha come sensitivity list il clock generato dal test bench e il segnale di i\_reset per poter ottenere reset asincroni. Gli stati sono implementati come case di un costrutto switch, una variabile provvede a tenere memoria dello stato corrente. Poiché il processo è avviato ad ogni commutazione dello stato del clock l’istruzione *rising\_edge(i\_clk)* provvede alla sincronizzazione sul fronte di salita. Altri dettagli implementativi direttamente commentati nel sorgente.

Nonostante il design così ottenuto fosse sintetizzabile, l’esecuzione di troppe operazioni complesse, specialmente negli stati CHECK\_PIXEL e CALC\_AREA, non consentivano il raggiungimento dell’obiettivo del segnale di clock a 15 ns.

Si sono quindi scomposti questi ultimi due stati in sotto-stati che eseguissero operazioni semplici:



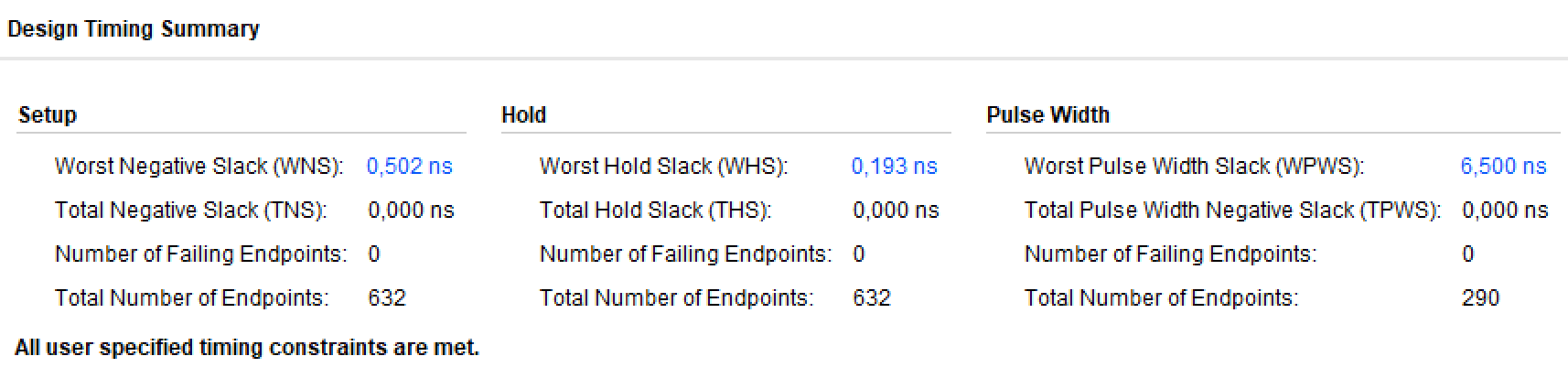
In particolare, a CHECK\_PIXEL è stato lasciato il solo compito di controllare se i pixel appartengono all’immagine mentre tutte le altre operazioni sono state delegate a sotto-stati dai nomi auto esplicativi. Stessa cosa è avvenuta per CALC\_AREA che nel design finale si occupa solamente di gestire alcuni casi particolari mentre le operazioni di calcolo vere e proprie sono svolte in stati separati.

VALIDAZIONE DEL DESIGN VHDL

Una volta ottimizzata la macchina a stati finiti e verificato che fosse ancora sintetizzabile, sono stati eseguiti i test bench forniti. Il design è risultato superare correttamente le simulazioni sia in pre-sintesi che in post-sintesi.

IMPLEMENTAZIONE E TEST DEL TIMING

Il design implementato si è dimostrato funzionante fino ad un periodo di clock di 14 ns corrispondente ad una frequenza di circa 70 MHz. Infine, sono stati ripetuti i test bench sull’implementazione per un controllo finale.



*Timing report generato da Vivado con il clock a 14 ns.*