Prova finale (Progetto di reti logiche)

Professore: William Fornaciari

Tutor: Davide Zoni

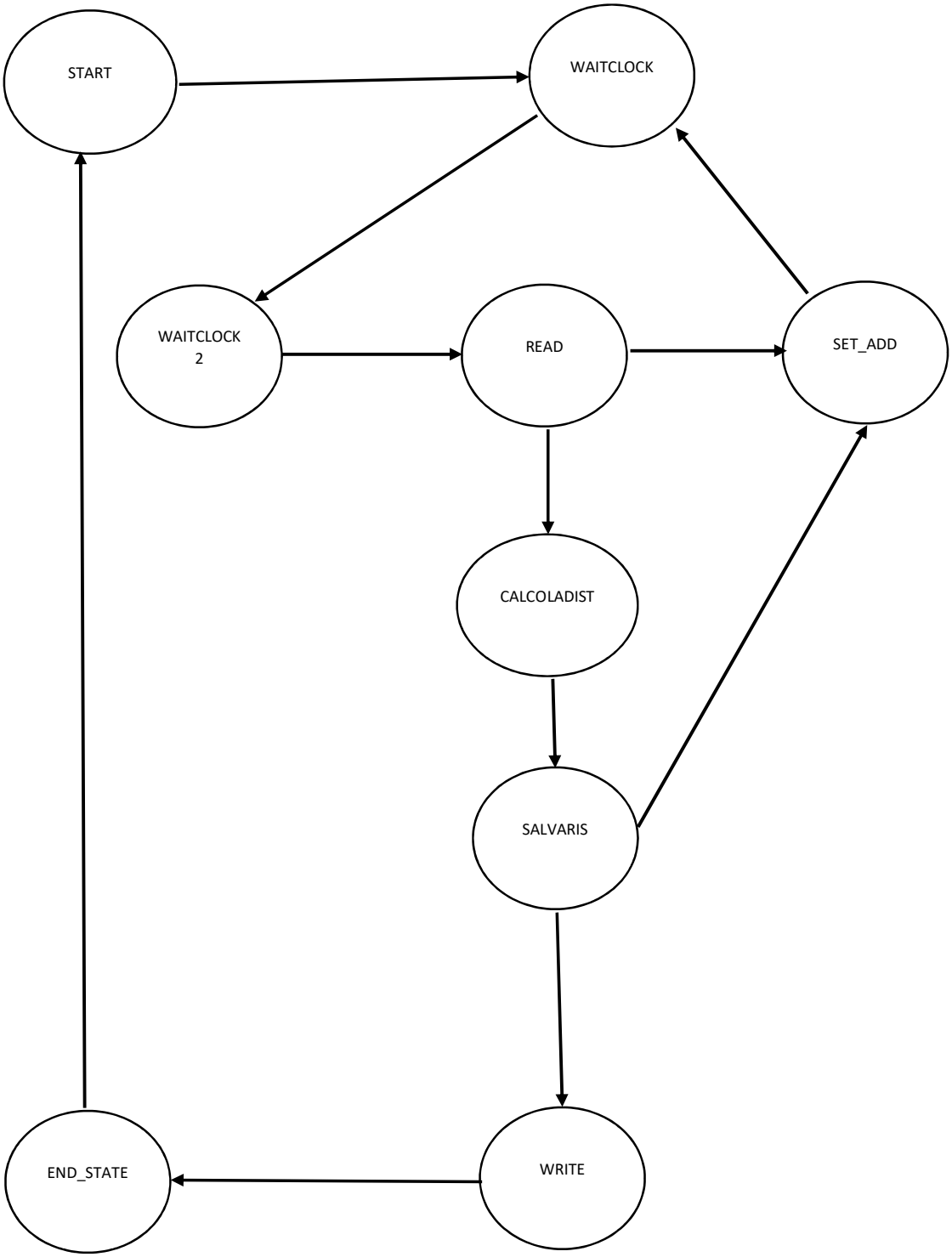
Anno accademico 2018/2019

Componenti:

Cognome	Nome	Matricola	Codice persona
Camnasio	Samuele	869764	10566482
Castelli	Francesco	870478	10563276

Indice:

Macchina a stati



Scelte progettuali

All'interno dell'architecture i valori sono gestiti tramite signal. Per ogni segnale contenente un valore corrente è presente un relativo segnale di "next", che serve a contenere il successivo valore da assegnargli.

Nell'architecture sono presenti nove stati:

- START: stato iniziale in cui vengono inizializzati i valori dei segnali next,
- SET_ADD: stato in cui viene settato il nuovo address sulla base del valore corrente,
- WAITCLOCK, WAITCLOCK2: stati di attesa per consentire di leggere correttamente dalla memoria,
- READ: stato in cui viene letto e assegnato ad un determinato segnale il valore della memoria,
- CALCOLADIST: stato in cui avviene il calcolo della distanza tra il centroide corrente e quello di riferimento,
- SALVARIS: stato in cui vengono effettuati i controlli sul risultato in base alla maschera,
- WRITE: stato in cui viene scritto il valore nella memoria,
- END_STATE: stato necessario per attendere che il dato venga salvato in memoria, per poi riportare la macchina nello stato iniziale;

Sono presenti tre process:

- Il primo sensibile a: i_clk, i_rst e i_start, che si occupa di aggiornare tutti i signal assegnandogli il valore contenuto nel relativo segnale di "next",
- Il secondo, che contiene la logica combinatoria, sensibile a tutti i signal, gestisce la transizione degli stati e l'aggiornamento dei segnali "next",
- Il terzo, sensibile a curr_state, i_start, curr_address e curr_data, che si occupa di aggiornare i segnali di uscita;

Casi di test

Abbiamo effettuato i seguenti casi di test:

- Esempio test bench fornito dal professore,

- Centroidi tutti coincidenti, ma diversi dal centroide da confrontare, e andiamo a verificare che la maschera di uscita sia tutta a 1 nelle posizioni attive secondo la maschera di ingresso, 0 nelle rimanenti

- Centroidi tutti diversi ma equidistanti, verifichiamo che la maschera di uscita sia tutta a 1 nelle posizioni attive nella maschera di ingresso, 0 nelle rimanenti

- Maschera tutta a 0, indipendentemente dai valori assegnati ai centroidi, maschera di uscita tutta a 0,

- Tutti i centroidi coincidenti con il punto da valutare, maschera di ingresso tutta a 1, maschera di uscita tutta a 1,

- Un centroide coincidente con il punto da valutare, maschera tutta a 1, maschera di uscita a 1 solo nella posizione relativa al centroide coincidente,

- Centroide da valutare in uno degli angoli della matrice, altri centroidi negli altri angoli, per testare il funzionamento con valori limite per le coordinate X e Y;

Possibili ottimizzazioni

La macchina a stati è ottimizzata utilizzando un numero limitato di stati:

- stato di lettura per leggere tutti i diversi valori dalla memoria indipendentemente dal fatto che siano la maschera, centroidi da valutare, o da confrontare. Una stessa sequenza di stati è in grado di gestire tutti i valori letti indipendentemente dal loro significato.

La macchina dopo START passa subito in WAITCLOCK risparmiando un ciclo di clock per settare il primo indirizzo da leggere.

Un possibile miglioramento è quello di ottimizzare la sincronizzazione con la memoria riducendo ad un solo stato di attesa il tempo necessario per leggere correttamente i valori dalla memoria.

Differenze tra le consegne

Versione del codice consegnata il 15/05/2019:

```
when CALCOLADIST =>
            if(X >= X_temp AND Y >= Y_temp) then
                                next_dist <= (X - X_temp) + (Y - Y_temp);
            elsif(X >= X_temp AND Y <= Y_temp) then
                                next_dist <= (X - X_temp) + (Y_temp - Y);
            elsif(X <= X_temp AND Y <= Y_temp) then
                                next_dist <= (X_temp - X) + (Y_temp - Y);
            elsif(X >= X_temp AND Y <= Y_temp) then
                                next_dist <= (X_temp - X) + (Y_temp - Y);
            end if;

            next_state <= SALVARIS;
            if(to_integer( unsigned(curr_address)) = 16) then
              address <="0000000000010011";
            end if;
```

Versione del codice aggiornata:

```
when CALCOLADIST =>
            if(X >= X_temp AND Y >= Y_temp) then
                                next_dist <= (X - X_temp) + (Y - Y_temp);
            elsif(X >= X_temp AND Y <= Y_temp) then
                                next_dist <= (X - X_temp) + (Y_temp - Y);
            elsif(X <= X_temp AND Y >= Y_temp) then
                                next_dist <= (X_temp - X) + (Y- Y_temp);
            elsif(X <= X_temp AND Y <= Y_temp) then
                                next_dist <= (X_temp - X) + (Y_temp - Y);
            end if;

            next_state <= SALVARIS;
            if(to_integer( unsigned(curr_address)) = 16) then
              address <="0000000000010011";
            end if;
```

Nella versione del 15/05/2019 manca la condizione: X <= X_temp AND Y >= Y_temp.