# Relazione di vhdl

Verdicchio Giacomo e Zarbo Nicola

24 marzo 2022

## 1 Introduzione

//////////////////////////////da inserire
fatto da Nicola Zarbo e Giacomo Verdicchio

## 2 Architettura

Architettura: L'obiettivo è quello di riportare un schema funzionale (lo schema in moduli... un bel disegno chiaro... i segnali i bus, il/i clock, reset... );

### 2.1 Modulo 1

———-————disegno di alto livello del funzionamento———————
Il modulo gestisce la lettura e scrittura da memoria e i segnali del modulo 2 'codificatore convoluzionale'.

#### 2.1.1 Segnali

i_clk : segnale di clock
i_rst : segnale di reset asincrono
i_start : segnale di enable '1'=> operativo,'0' => fermo (quindi il modulo viene resettato)
i_data : bus 8 bit, dati di lettura da ram
o_address : bus 16 bit, comunica alla ram l'indirizzo su cui eseguire lettura/scrittura
o_done : seganale di finita elaborazione '1' => flusso elaborato/scritto in memoria
o_en : segnale di enable per ram
o_we : segnale per comunicare alla ram quale operazione svolgere, '0'=> read, '1' => write
o_data : bus 8 bit, dati in scritttura per ram

#### 2.1.2 Registri

stato_att, st_prox (8 bit): registri di stato per fsm
in_value (4 bit): dove viene copiato la parola di 8 bit letta da i_data
out_value_buffer (8 bit) : dove viene scritta la parola da scrivere, collegato a o_data
in_addr (16 bit): per mantenere address per lettura e per controllo terminazione codifica
in_a_prox (16 bit): per incrementare l'address per l'operazione di read
out_addr, out_a_prox (16 bit):registri per mantenere e incrementare address per write
nTerminazione (9 bit): usato per controllo terminazione, mantiene il valore della cella ram '0000' incrementato di 1

#### 2.1.3 Segnali per componente interno

fU : flusso di bit in lettura da codificare
fY : flusso di 2 bit in uscita da codificatore
stop_en : segnale per fermare la macchina a stati del codificatore al di fuori dei clock di codifica

### 2.1.4 FSM

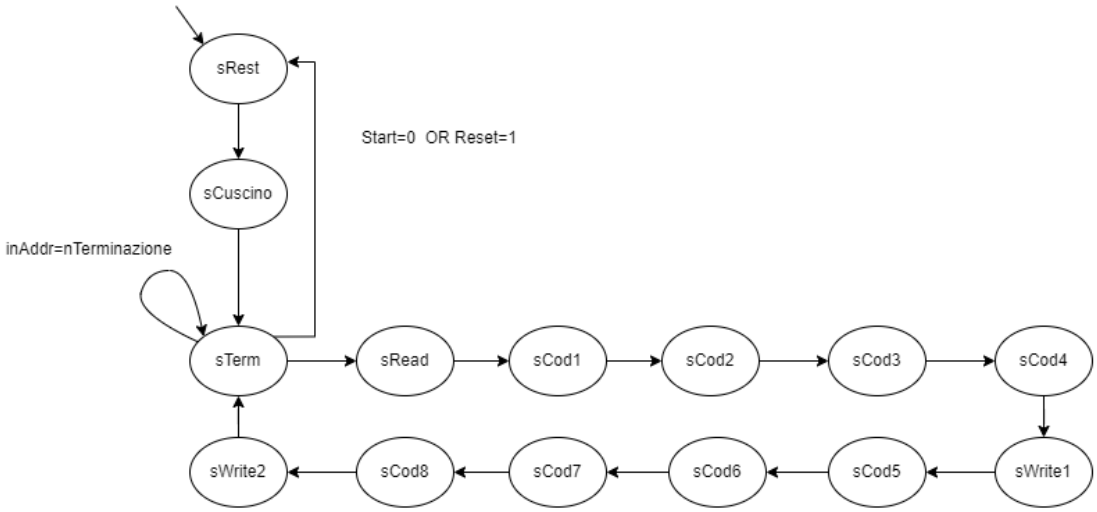


Figura 1: Disegno esplicativo della FSM

Descrizione stati:

sReStart : stato di reset, viene letto la cella all'indirizzo '0000' e il suo valore incrementato di uno viene salvato in nTerminazione
sCuscino : stato attraversato solo una volta in tutta l'operzione di codifica, serve per consentire funzionamento con n di parole nullo ( per ulteriori info vedere test seq_min)
sTerm : controllo terminazione codifica, confronta il numero totale di parole da leggere (+ 1) con il prossim indirizzo di lettura
sRead : stato di lettura, fornisce alla ram i segnali per leggere la prossima parola da elaborare
sCod1 : codificatore in funzionamento, viene salvata la parola appena letta da i_data nel registro in_value, viene inserito in fU il primo bit dalla parola letta
sCod2 to sCod4 : inserito in fU il nuovo bit da leggere preso da in_value, bit codificati da fY salvati in out_value_buffer nell'apposita posizione
sWrite1 : inseriti bit da fY negli ultimi due bit del registro
out_value_buffer, forniti segnali alla ram per scrivere la parola appena codificata, codificatore bloccato
sCod5 to sCod8 : codificatore in funzionamento, funzionamento equivalente a sCod1-sCod4, usando gli utlimi 4 bit di in_value
sWrite2 : equivalente a sWrite1

## 2.2 Modulo 2 : Codificatore Convoluzionale

Il codificatore convoluzionare è il modulo che si occupa dell'effettiva codifica dei dati in ingresso

### 2.2.1 Segnali

i_U : flusso di bit in ingresso
i_start : segnale di enable del componente, se off il componente viene resettato
i_rst : segnale di reset asincrono
stop_en : segnale di enable del componente, se off viene mantenuto in standby
o_Y : flusso di 2 bit da codifica in uscita

### 2.2.2 Registri

st_att, s_pros : regsitri per stati della fsm di mealy

### 2.2.3 FSM

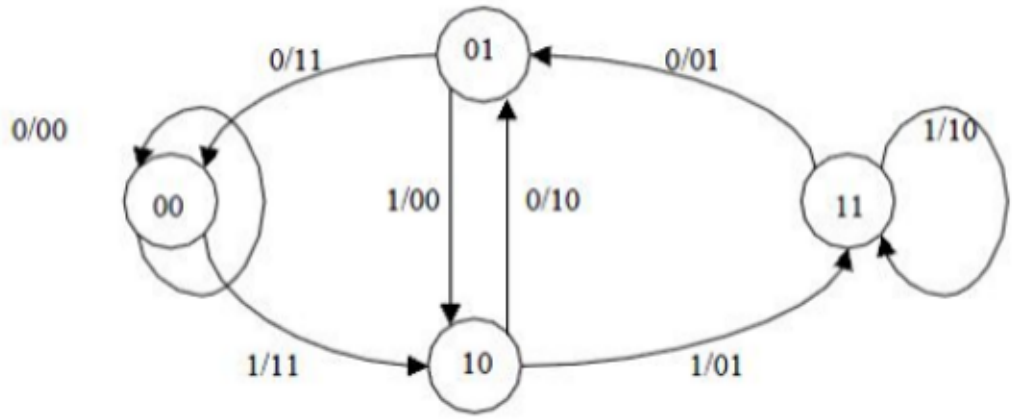


Figura 2: Disegno esplicativo del codificatore convoluzionare

# 3 Risultati sperimentali

## 3.1 Sintesi

(report di sintesi)

## 3.2 Simulazioni

Le simulazioni che abbiamo creato servono a testare i casi limite che potrebbero mandare in loop, blocco o crash la macchina a stati da noi svilupata

### 3.2.1 Simulazione max

test bench 1 (cosa fa e perchè lo fa e cosa verifica; per esempio, controlla una condizione limite)

### 3.2.2 Simulazione min

test bench 2 (cosa fa e perchè lo fa e cosa verifica; per esempio, controlla una condizione limite)

#### 3.2.3 Simulazione reset multipli

#### 3.2.4 Simulazione start multipli

# 4 Conclusione

leggibilità over velocità

tutto facilmente moidificabile e leggibile

## 4.1 note particolari modulo 1

La fsm usa molti stati equivalenti, scelta adottata per favorire la comprensibilità usando un approccio simile agli automi a stati finiti, evitando quindi il più possibile dei controlli per la scelta del prossimo stato, tranne ovviamente in sTerm, dove viene verificata la terminazione della codifica.

## 4.2 note particolari modulo 2

La codifica avviene tramite questo componente, pensato per rispecchiare il più possibile quello descritto dalla specifica e per essere facilmente utilizzabile in contesti diversi senza bisogno del Modulo 1. Questo tipo di codificatori sono usati nelle telecomunicazioni, quindi potrebbe essere utilizzato collegando l'uscita ad un componente per trasmettere direttamente il segnale codificato invece di salvarlo su una memoria.