



POLITECNICO DI MILANO

PROGETTO FINALE DI RETI LOGICHE

# Codifica Working Zone

*Nicolò Sonnino*

supervisore  
Prof. Gianluca PALERMO

2 agosto 2020

# Indice

<b>1</b>	<b>Specifiche</b> .....	<b>2</b>
1.1	Descrizione .....	2
1.2	WZE (Working Zone Encoding) .....	2
1.3	Esempi .....	3
<b>2</b>	<b>Architettura</b> .....	<b>4</b>
<b>3</b>	<b>Risultati Sperimentali</b> .....	<b>5</b>
3.1	Sintesi .....	5
3.2	Simulation .....	6
3.3	Schematic .....	7
<b>4</b>	<b>Test Benches</b> .....	<b>8</b>
4.1	tb_pfrl_2020_in_wz .....	8
4.2	tb_pfrl_2020_no_wz .....	8
<b>5</b>	<b>Conclusioni</b> .....	<b>8</b>

# 1 Specifiche

## 1.1 Descrizione

Il progetto ha come scopo la realizzazione della codifica **working zone**, questa strategia viene utilizzata soprattutto nello sviluppo di microprocessori. Il consumo di energia dei pin è una parte integrante nella loro progettazione, per ridurlo, si suppone, che determinati programmi favoriscano poche zone all'interno del loro spazio di memoria in ogni istante.

Quindi vengono identificate tali zone e con un riferimento dell'indirizzo scelto viene inviato anche un offset codificato "one-hot" per trovare la sua posizione rispetto all'indirizzo base della working zone.

## 1.2 WZE (Working Zone Encoding)

Il progetto presenta una memoria RAM da 65536 indirizzi con valore base di 8 bit per ogni cella di memoria; di questi i primi otto sono riempiti con indirizzi base, mentre il nono contiene il valore da codificare e il decimo è quello riservato alla scrittura del risultato codificato.

A questo punto abbiamo due possibili casi: il valore rientra in una delle working zones, oppure non appartiene a nessuno dei sette indirizzi; nel primo caso si identifica la working zone (WZ\_NUM) e l'offset rispetto ad essa (WZ\_OFFSET), si pone WZ\_BIT = 1 e si scrive nell'indirizzo di posizione  $9 \text{ WZ\_BIT} \& \text{ WZ\_NUM} \& \text{ WZ\_OFFSET}^{(1)}$ , mentre nel secondo caso viene posto WZ\_BIT = 0, il valore (ADDR) viene salvato e l'output risulta WZ\_BIT & ADDR.

L'offset **one-hot** associa all'unico 1 il valore da rappresentare nel seguente modo:

- $\text{WZ\_OFFSET} = 0 \Rightarrow \text{"0001"}$
- $\text{WZ\_OFFSET} = 1 \Rightarrow \text{"0010"}$
- $\text{WZ\_OFFSET} = 2 \Rightarrow \text{"0100"}$
- $\text{WZ\_OFFSET} = 3 \Rightarrow \text{"1000"}$

---

<sup>(1)</sup>& simbolo di concatenazione

### 1.3 Esempi

WZ_BIT	WZ_NUM	WZ_OFFSET
1	3 bits	4 bits

**1|010|0100**

0	0000000000000000	17
1	:	9
2	:	<b>33</b>
3	:	42
4	:	54
5	:	88
6	:	69
7	:	180
8	:	<b>35</b>
9	00000000000001001	<b>164</b>

Figura 1: Valore contenuto in posizione 2

WZ_BIT	ADDR
0	7 bits

**0|10110**

0	0000000000000000	17
1	:	9
2	:	23
3	:	42
4	:	115
5	:	88
6	:	69
7	:	180
8	:	<b>54</b>
9	00000000000001001	<b>54</b>

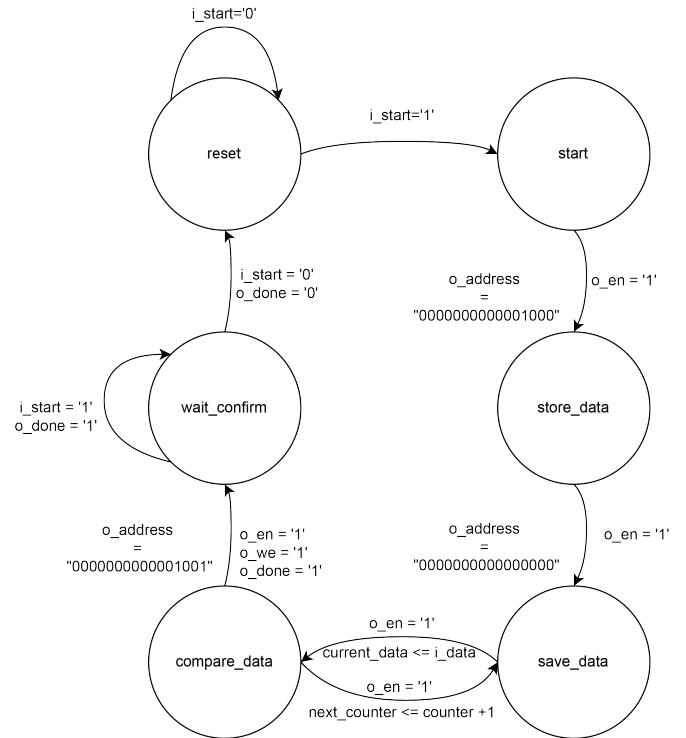
Figura 2: Valore non contenuto in nessuna working zone

## 2 Architettura

Di seguito viene mostrata la FSM (Final State Machine) in versione semplificata, sono omessi tutti gli anelli che da qualsiasi stato tornano a **reset** quando **i\_rst='1'**.

Descrizione di ogni stato:

- **reset**: stato di inizializzazione, se il segnale **i\_rst** viene alzato si ritorna a questo stato. Fintanto che **i\_start** rimane basso si ritorna a reset, se viene alzato si passa a start.
- **start**: stato di preparazione, i segnali **o\_we**, **wz\_bit** vengono abbassati, l'**offset** viene posto a 0000; per permettere la lettura dell'indirizzo in posizione 8, **o\_en** viene alzato a 1 e si assegna 0000000000001000 a **o\_address**.
- **store\_data**: stato di memorizzazione del valore da codificare, **i\_data** letto precedentemente viene salvato nel registro **data** e viene inizializzato il valore **next\_counter**, responsabile di iterare gli indirizzi di memoria.
- **save\_data**: stato di salvataggio, il valore di **i\_data** viene salvato per essere utilizzato per confrontarlo successivamente.
- **compare\_data**: stato di confronto dei valori, attraverso un contatore interno allo stato si confronta il valore di **data** con **current\_data** con sommato il contatore; se risultano uguali viene abilitata la scrittura tramite l'assegnazione di **o\_en** e **o\_we** a 1 e **o\_address** viene posto a 9, infine **o\_data** viene codificato come discusso nelle specifiche e **o\_done** viene alzato a 1. Nel caso in cui non appartenga alla working



zone, se non è al ottavo indirizzo, il contatore viene incrementato e si ritorna allo stato precedente, altrimenti viene abilitata la scrittura del valore codificato e si alza **o\_done**.

- **wait\_confirm**: stato di attesa, se allo stato precedente è stato alzato **o\_done** si arriva a quest'ultimo; vengono quindi azzerati i segnali di scrittura e lettura, **o\_done** rimane alzato fino a quando non viene ricevuto un segnale **i\_start** = 0 e allora in quel caso **o\_done** viene abbassato e si ritorna a reset.

## 3 Risultati Sperimentali

### 3.1 Sintesi

Il progetto è stato testato sulla versione di **Vivado 2016.4**, utilizzando come FPGA target **xc7a200tfbg484-1**, e ha generato il seguente report di sintesi:

Site Type	Used	Fixed	Available	Util%
Slice LUTs*	46	0	134600	0.03
LUT as Logic	46	0	134600	0.03
LUT as Memory	0	0	46200	0.00
Slice Registers	19	0	269200	<0.01
Register as Flip Flop	5	0	269200	<0.01
Register as Latch	14	0	269200	<0.01
F7 Muxes	0	0	67300	0.00
F8 Muxes	0	0	33650	0.00

Per gli stati e il consumo invece:

State	New Enc.	Previous Enc.	Type	Power	Util%
reset	000	000	Signals	0.448 W	16%
start	001	001	Logic	0.285 W	10%
store_data	010	010	I/O	2.059 W	74%
save_data	011	011	Device Static	0.142 W	5%
compare_data	100	100			
wait_confirm	101	101			

### 3.2 Simulation

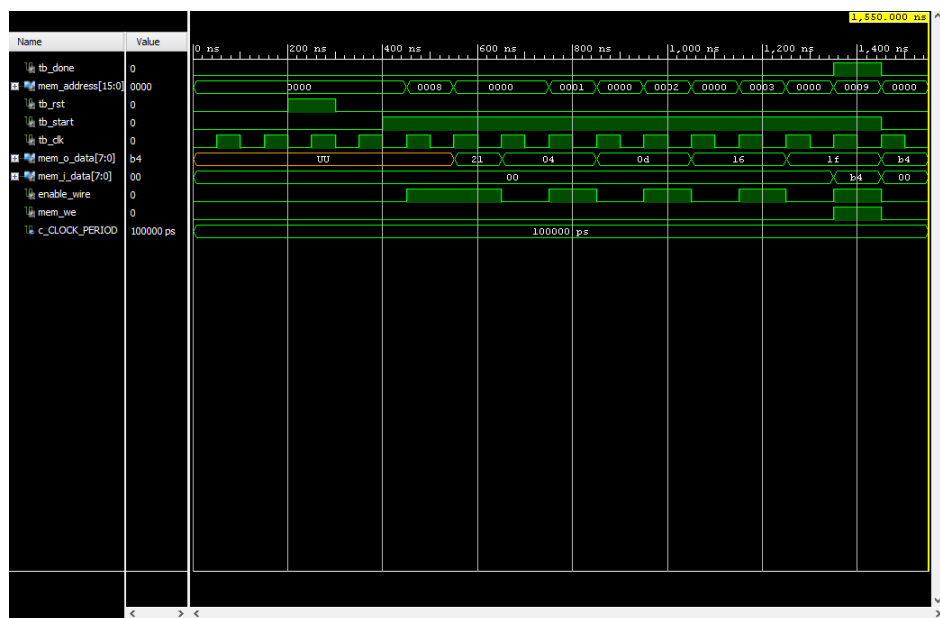


Figura 3: tb\_pfrl\_2020\_in\_wz

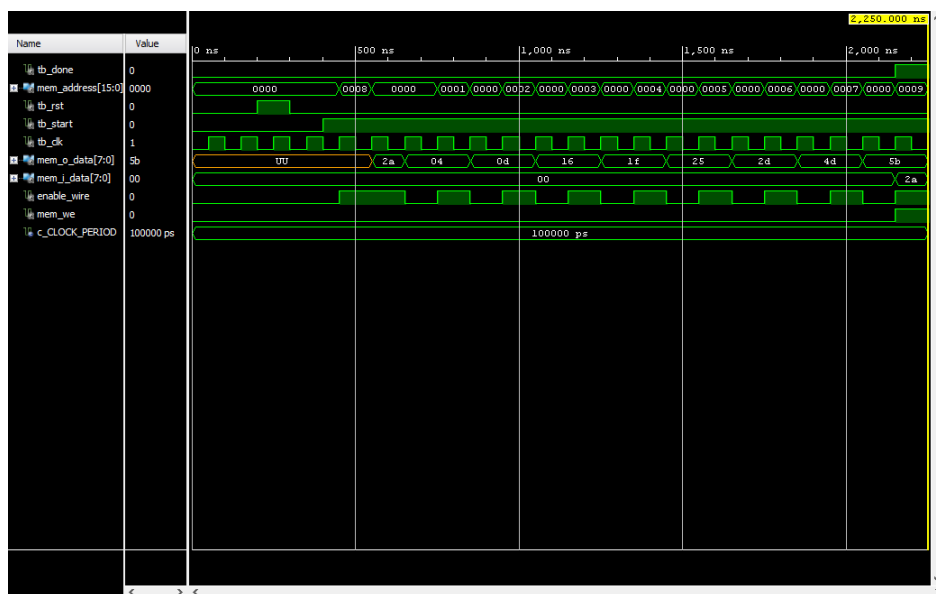
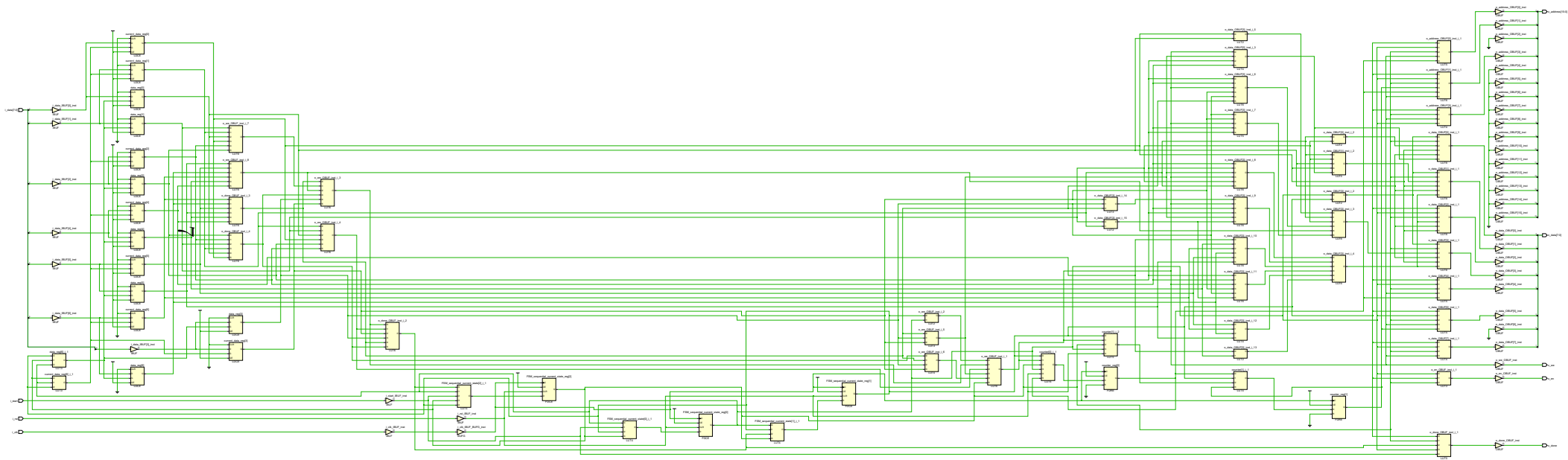


Figura 4: tb\_pfrl\_2020\_no\_wz

### 3.3 Schematic





## 4 Test Benches

### 4.1 tb\_pfrl\_2020\_in\_wz

In questo test bench si vuole verificare la correttezza del progetto nel caso di un valore appartenente a una working zone, in particolare il valore da codificare è 33. La posizione 3 di memoria ha il valore 31, quindi l'output corretto per passare questo testbench è 1 (WZ\_BIT) & 011 (WZ\_NUM) & 0100(OFFSET). Il testbench risulta passato con n cicli di clock da n ns.

### 4.2 tb\_pfrl\_2020\_no\_wz

Lo scopo di questo test bench è verificare che nel caso di un valore non appartenente a una working zone, l'output risulti corretto. Poichè il valore da codificare è 42 e la working zone più vicina ad esso è 37, quindi non esiste  $0 \leq \text{offset} \leq 3$  tale da raggiungere il valore, l'output risulta 0 (WZ\_BIT) & 0101010 (42). Il testbench risulta passato con n cicli di clock da n ns.

## 5 Conclusioni

Dai risultati sperimentali e dalla sintesi si può notare che si sia studiata la codifica working zone con offset one-hot nella sua interezza. Una possibile ottimizzazione potrebbe essere gestire il valore di o\_data ad ogni stato con un registro apposito per evitare di avere valori undefined, anche quando non ci interessa osservarlo, e quindi abbassare il numero di LUT ed eliminare un inferring latch.