

Prova Finale di Reti Logiche

Leonardo Airoidi

A. A. 2021-22

Indice

1	Introduzione	2
1.1	Specifiche	2
1.2	Strumenti	3
1.3	Esempio	3
2	Architettura	4
2.1	Serializzatore	4
2.2	Convolutore	5
2.3	Parallelizzatore	6
2.4	Datapath	7
3	Risultati sperimentali	10
3.1	Sintesi	10
3.1.1	Componenti	10
3.1.2	Timing	10
4	Conclusione	12

Capitolo 1

Introduzione

La specifica della Prova finale (Progetto di Reti Logiche) 2021 richiede di implementare un **codificatore convoluzionale** usando il linguaggio VHDL.

Un codificatore convoluzionale è un dispositivo che tramite l'applicazione di un *codice convoluzionale* codifica uno stream di bit in ingresso in uno stream di bit in uscita ridondante che permette a un dispositivo che legge lo stream codificato di risalire allo stream originale anche in presenza di errori applicando tecniche di *error-correction*. Codificatori convoluzionali elettronici sono molto comuni nell'ambito delle telecomunicazioni, in quanto conferiscono a una rete un certo grado di robustezza: un dispositivo ricevente può ricostruire il segnale trasmesso, che potrebbe risultare corrotto a causa del rumore sulla linea, senza richiedere un ulteriore invio dell'informazione.

1.1 Specifiche

Nel caso specifico di questo progetto, è richiesto lo sviluppo di un circuito integrato con le seguenti caratteristiche:

1. il *codice convoluzionale* ha **tasso di trasmissione** $\frac{1}{2}$.
2. si interfaccia con una memoria sincrona a word di 8bit e indirizzi a 16bit.
3. lo stream in ingresso è di dimensione n words memorizzate a partire dall'indirizzo 0x01.
4. la dimensione dello stream è memorizzata all'indirizzo 0x00 ed è al massimo di 255 word.
5. il dispositivo viene sempre inizializzato con un segnale di **reset**.
6. la computazione inizia al segnale di **start** e termina quando il dispositivo alza il segnale **done**. Il dispositivo deve essere in grado di gestire computazioni successive senza aver bisogno di un segnale di **reset**. Si presuppone che il segnale di **start** rimanga alto durante tutta la computazione e che non possa essere rialzato prima che il segnale di **done** sia stato riportato a 0.
7. il dispositivo deve funzionare con un periodo di clock di almeno 100ns

1.2 Strumenti

Il progetto sarà sviluppato in **VHDL** (VHSIC Hardware Description Language) che permette di descrivere circuiti integrati. Per la simulazione e la sintesi della scheda il software utilizzato è **Xilinx Vivado** (*v2021.1*). Il dispositivo sarà poi implementato su una **FPGA Artix 7 xc7a200tfbg484-1**.

1.3 Esempio

Capitolo 2

Architettura

Il dispositivo è stato implementato come una macchina a stati che controlla un datapath, che si interfaccia con diversi componenti più piccoli. La scelta di dividere il progetto in moduli più piccoli è stata presa per non rendere troppo complessa e articolata la gestione del dispositivo nella sua interezza. In questo modo la macchina a stati si limita a gestire la comunicazione e coordinazione dei vari componenti, che eseguono ciascuno una funzione definita anch'essa da una macchina a stati interna.

2.1 Serializzatore

Il **serializzatore** è il componente addetto alla serializzazione in un flusso continuo di bit dei byte caricati dalla ram. Il serializzatore si interfaccia, oltre al segnale di *clock* e al segnale di *reset*, con il segnale di ingresso da serializzare [**input:8bit**] e dal segnale di controllo *start*. Produce in uscita un segnale di [**output:1bit**] che corrisponde al flusso serializzato del byte in input e un segnale di *done* che indica la fine della serializzazione.

Al segnale di **start** della serializzazione il componente passa dallo stato STOPPED allo stato attivo. Quando il componente è attivo produce in uscita per **due cicli di clock** l'*n-esimo* bit del segnale in ingresso, incrementando poi il contatore per produrre in uscita l'*n+1-esimo* bit del byte in ingresso nei prossimi due cicli di clock. La serializzazione termina quando il contatore interno raggiunge il valore 111: il registro viene incrementato tornando quindi a 0 ma il serializzatore torna nello stato STOPPED, che alza il segnale di **done** a 1.

Il flusso in uscita ha periodo di due cicli di clock in quanto al convolutore server produrre due bit sequenzialmente in uscita per ogni bit in entrata. In questo modo la velocità di serializzazione è ridotta della metà ma il convolutore riesce a produrre il flusso risultante sullo stesso ciclo di clock della macchina.

all'interno del registro di *count* supera il numero di parole. Questo sfrutta il fatto che il numero della parola corrente corrisponde al contenuto del registro *count*, in quando l'indirizzo della parola *n*-esima è *n*. Inoltre, se dovessimo essere nel caso in cui la lunghezza dello stream fosse 255, il confronto funzionerebbe ancora perchè fino a 255 il processo continuerebbe regolarmente mentre al raggiungimento di 256, valore possibile nel registro degli indirizzi a 16bit *count*, il confronto finirebbe correttamente per segnalare la fine dell'esecuzione.

```
if mux_count <= ("00000000" & reg_words) then
    o_done_signal <= '0';
else
    o_done_signal <= '1';
```

Multiplexer read write. Il registro *count* non è direttamente collegato con il segnale *address* della memoria RAM. Questo perchè in scrittura devono essere utilizzati gli indirizzi contigui partendo dall'indirizzo 1000. Questo multiplexer permette di selezionare quale indirizzo viene passato al segnale di indirizzo della memoria:

```
if i_rst = '1' or mux_rw_addr_sel = "00" or mux_rw_addr_sel = "11" then
    o_address <= reg_count;
elsif mux_rw_addr_sel = "01" then
    o_address <= (mux_count + mux_count) + "0000001111100110";
else --if mux_rw_addr_sel = "10" then
    o_address <= (mux_count + mux_count) + "0000001111100111";
end if;
```

Il mux seleziona di default il contenuto del registro *count* usato in lettura dello stream. Per la scrittura dei due byte corrispondenti all'elaborazione dell'*n*-esima parola, viene selezionato l'indirizzo $2n + 998$ per il primo byte e $2n + 999$ per il secondo. Questo permette di scrivere nell'indirizzo giusto i due byte derivati dall'elaborazione dell'*n*-esimo byte. Per esempio, la convoluzione terza parola verrà scritta negli indirizzi 1004 1005, in quanto la prima parola è scritta all'interno di 1000 1001 e la seconda in 1002 1003, come da specifica.

Capitolo 3

Risultati sperimentali

Il dispositivo progettato è stato sintetizzato e testato con il tool Vivado di Xilinx. Vengono riportate ora le analisi sugli aspetti di sintesi del dispositivo e sui test effettuati su di esso.

3.1 Sintesi

La sintesi è eseguita con un constraint sul periodo di clock di **100ns** e con una fpga target **FPGA Artix 7 xc7a200tfbg484-1**.

3.1.1 Componenti

Il report di sintesi riassume l'elenco dei componenti della FPGA utilizzati per sintetizzare il dispositivo.

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	109	0	0	134600	0.08
LUT as Logic	109	0	0	134600	0.08
LUT as Memory	0	0	0	46200	0.00
Slice Registers	81	0	0	269200	0.03
Register as Flip Flop	81	0	0	269200	0.03
Register as Latch	0	0	0	269200	0.00
F7 Muxes	1	0	0	67300	<0.01
F8 Muxes	0	0	0	33650	0.00

Si può notare come tutti i registri utilizzati siano **sincroni** (flip flop).

3.1.2 Timing

Il report di timing riassume i parametri della scheda in relazione al segnale di clock

Slack (MET) : 95.783ns (required time - arrival time)
Source: DATAPATH0/reg_count_reg[3]/C
{rise@0.000ns fall@50.000ns period=100.000ns}

```

Destination:          FSM_onehot_cur_state_reg[4]/D
                      {rise@0.000ns fall@50.000ns period=100.000ns}
Path Group:           clock
Path Type:            Setup (Max at Slow Process Corner)
Requirement:          100.000ns (clock rise@100.000ns - clock rise@0.000ns)
Data Path Delay:      4.066ns (logic 1.496ns (36.793%) route 2.570ns (63.207%))
Logic Levels:         4 (CARRY4=2 LUT4=1 LUT6=1)
Clock Path Skew:      -0.145ns (DCD - SCD + CPR)
  Destination Clock Delay (DCD):    2.100ns = ( 102.100 - 100.000 )
  Source Clock Delay (SCD):         2.424ns
  Clock Pessimism Removal (CPR):    0.178ns
Clock Uncertainty:     0.035ns ((TSJ^2 + TIJ^2)^1/2 + DJ) / 2 + PE
  Total System Jitter (TSJ):        0.071ns
  Total Input Jitter (TIJ):         0.000ns
  Discrete Jitter (DJ):             0.000ns
  Phase Error (PE):                0.000ns

```

Si può notare come lo **slack** (parametro che indica il tempo tra il delay massimo della logica combinatoria e il prossimo rising-edge del ciclo di clock) sia particolarmente elevato. Questo a grazie al delay della logica combinatoria (**4.066ns**) che risulta più di un ordine di grandezza inferiore al periodo del ciclo di clock (100ns). Questo dato indica quindi che la scheda sarebbe in grado di girare a 10 volte la velocità attuale, ovvero con un ciclo di clock di periodo 10ns (100Mhz).

Capitolo 4

Conclusione