Contatore binario

Lorenzo Bartolozzi Luca Pacioselli Giuseppe Prudente

Università di Perugia
A.A. 2018/2019

Sommario

La presente esperienza è volta a studiare il funzionamento di: un contatore a 4 bit, implementato mediante l'utilizzo di 4 flip-flop D in cascata; un contatore a 16 bit, tramite codice in Verilog scritto per un dispositivo FPGA.

I. Introduzione

i. Strumenti a disposizione

- alimentatore in CC (fondoscala utilizzato: 10V);
- oscilloscopio digitale RIGOL;
- basette sperimentali;
- scheda NI USB-6008;
- LED a luce gialla (funzionamento tra 2,10V-2,18V);
- resistenze da 100Ω e 10kΩ;
- condensatori da 22μF e 47μF;
- un interruttore;
- circuito integrato LM555;
- due circuiti integrati 74HC (contenenti ciascuno 2 flip-flop di tipo D);
- FPGA (tipo Artix prodotta dalla Xilinx) integrata in una evaluation board (Diligent).

ii. Cenni di teoria

Flip-Flop

Un flip-flop è un circuito elettronico sequenziale, cioè tale per cui il valore istantaneo delle uscite dipende non solo dai valori degli ingressi, ma anche dalla storia passata del circuito, usato come unità fondamentale di memoria poiché in grado di memorizzare le informazioni di base, ovvero i bit 0 e 1. Nonostante ne esistano di vari tipi, in questa esperienza sono stati usati flip-flop di tipo D (Data). I flip-flop che sono stati utilizzati sono contenuti nei circuiti integrati, ovvero circuiti elettronici miniaturizzati contenenti più componenti elettroniche, 74HC. Il clock, elemento fondamentale per qualunque circuito sequenziale, è stato realizzato mediante il circuito integrato LM555 in modalità astabile, cioè come oscillatore.

FPGA

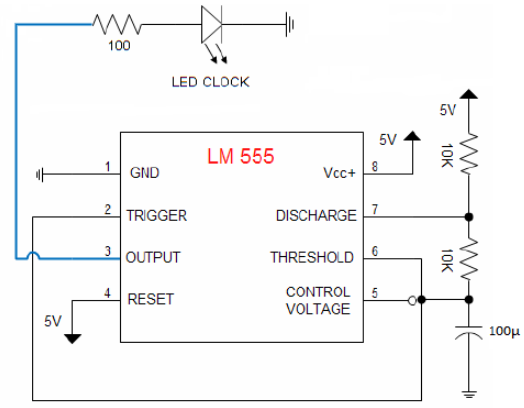
Un FPGA, Field Programmable Gate Array, è un circuito integrato a logica programmabile tramite un linguaggio di descrizione hardware. In generale un FPGA è composto da una matrice di blocchi logici configurabili, chiamati CLB (Configurable Logic Blocks) collegati tramite interconnessioni programmabili. I CLB sono formati di solito da due o quattro celle logiche che sono costituite da:
-LUT (*Look Up Table*): mappa una qualsiasi funzione combinatoria con 4 segnali in ingresso in 1 segnale in uscita;
-un *Flip-Flop* di tipo D;
-un *multiplexer* che dati 2 segnali in entrata, li restituisce in uscita uno alla volta.
Il linguaggio di descrizione hardware usato nell'esperienza è il *Verilog*, che supporta la progettazione, l'implementazione e la verifica di circuiti digitali.

II. Procedura sperimentale

Contatore a 4 bit (Flip-Flop)

Al fine di implementare un contatore a 4 bit si è partiti dall'assemblaggio del sistema di clock tramite il seguente schema circuitale del circuito integrato LM555 astabile:

Figura 1: *Le due resistenze a destra valgono* $10k\Omega$*, la resistenza prima del LED vale* 100Ω *(serve per evitare che si bruci) ed il condensatore ha capacità di* $14,98\mu F$



Quando questo circuito è alimentato, il LED, che corrisponderà al clock, comincerà ad accendersi e spegnersi con periodo dato dalla formula:

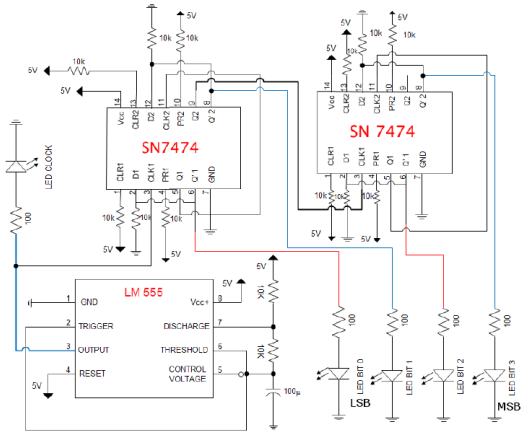
$$T = 0,693(R_1 + 2R_2)C$$

Il tempo in cui il LED sta acceso, t_1, e il tempo in cui sta spento, t_2, sono invece:

$$t_1 = 0,693R_2C \qquad t_2 = 0,693(R_1 + R_2)C$$

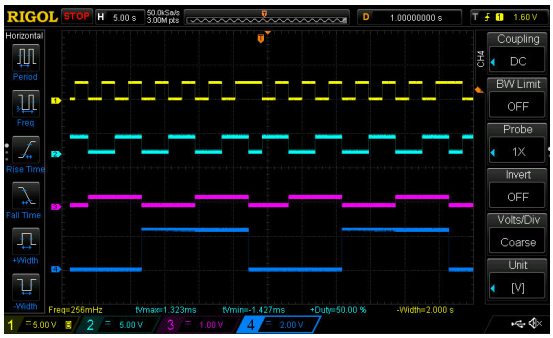
Scelti i valori delle componenti come riportato nella didascalia della Figura 1, il periodo è di circa $T = 311,4ms$. Si prosegue poi collegando il clock ai flip-flop in cascata seguendo lo schema circuitale seguente:

Figura 2



Avendo alimentato il circuito con un alimentatore in CC impostato a $V_{CC} = 5V$ i flip-flop utilizzati ricevono in entrata una d.d.p. di $V_{IH} > 2,0V$ per lo stato logico 1 e di $V_{IL} < 0,8V$ per lo stato logico 0 (valori adatti ai transistor di tipo TTL interni ai flip-flop); in uscita invece si avranno per lo stato logico 1, $V_{OH} > 3,3V$ e per lo stato logico 0 $V_{OL} < 0,35V$. All'uscita di ogni flip-flop è stato collegato un LED con cui effettuare la lettura dei conteggi, infatti ad ogni LED corrisponde una cifra di un numero in codice binario: quando il LED è acceso lo stato logico è 1, quando è spento lo stato logico è 0. Ogni LED si accende e si spegne con una frequenza pari alla metà di quella del LED precedente, come si vede dalla Figura 3.

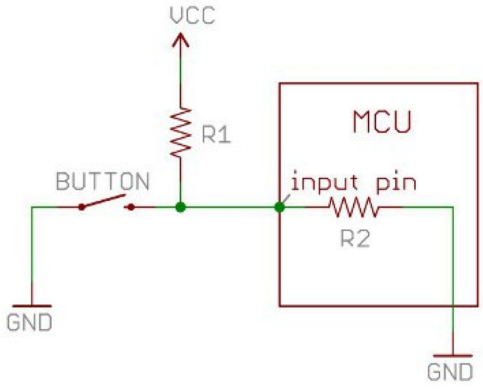
Figura 3: *Divisore di frequenza*



Ad ogni ciclo di clock il numero di conteggi aumenta di 1.

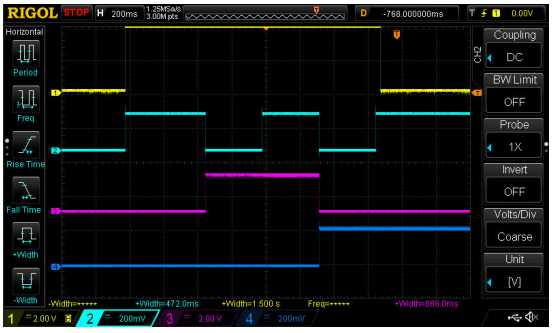
Si prosegue implementando, tramite programma "Labview", il VI in Figura 12 che invia al contatore, attraverso la scheda NI USB-6008, un segnale che sta a $5V$ per un tempo che si andrà a cambiare volta per volta e a $0V$ per un tempo sufficiente a vedere quanti LED sono accesi e prendere i dati. Infine, dato che alla partenza del conteggio si verificavano dei *glitch* che casualmente fanno accendere dei LED, si è aggiunto un interruttore collegato, schematicamente come nella seguente Figura 4

Figura 4: *Schema interruttore*



ai *preset*, PR1/PR2/PR3/PR4 Figura 2, dei flip-flop in modo da far resettare il contatore, ovvero spegnere tutti i LED. Una volta finita l'implementazione del contatore lo si è utilizzato per misurare la durata di un impulso noto di $1,5s$ con risultato in Figura 5.

Figura 5: *Il primo segnale è l'impulso, mentre gli altri sono in successione i bit 0, 1 e 2 del contatore*



Abbiamo notato, durante i tentativi di presa dati per la misura di un impulso noto, che il nostro circuito presenta un'anomalia, ovvero quando il contatore va in *overflow*, invece di far tornare il numero di conteggi a 0, questo riparte da 1. Quindi se si volesse utilizzare il contatore per misurare un intervallo di tempo maggiore di 15 conteggi bisognerebbe tenere conto di un errore sistematico per ogni "ciclo" di 15 conteggi che il contatore effettuerà.

Contatore a 16 bit (FPGA)

Si è implementato un contatore a 16 bit che funziona allo stesso modo del precedente, scrivendo il codice in *Verilog* in Figura 11 e fornendo il bitstream alla scheda FPGA. La frequenza del clock interno alla FPGA è di $100MHz$, ovvero un periodo di $10ns$, quindi per avere il bit meno significativo con una frequenza accettabile anche per l'occhio umano si è associato al clock del cronometro una frequenza di $100MHz/2^{15} = 3052Hz$, cioè un periodo di $0,33ms$, mentre il massimo numero di conteggi che il contatore può fare è $2^{16} - 1 = 65536(conteggi)$ che corrisponde ad un intervallo di tempo di $(0,00033s) * 65536(conteggi) = 21,62688s$. Per la caratterizzazione del contatore si è deciso di dare alla scheda FPGA un segnale tramite VI identico a quello precedente, con l'unica accortezza di diminuire la tensione da $5V$ a $3V$ perché la scheda non regge in entrata sopra ai $3,3V$ (Figura 13).

III. Analisi dati

Contatore a 4 bit (Flip-Flop)

I valori ottenuti per il seguente grafico dei conteggi in funzione del tempo sono riportati in Tabella 1.

Figura 6: *Grafico conteggi in funzione del tempo*

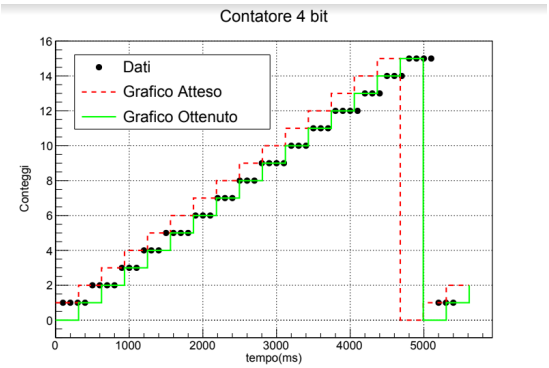
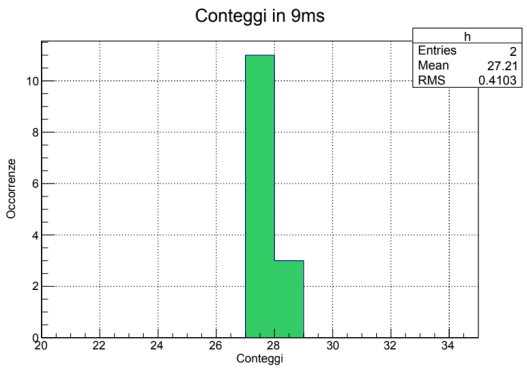
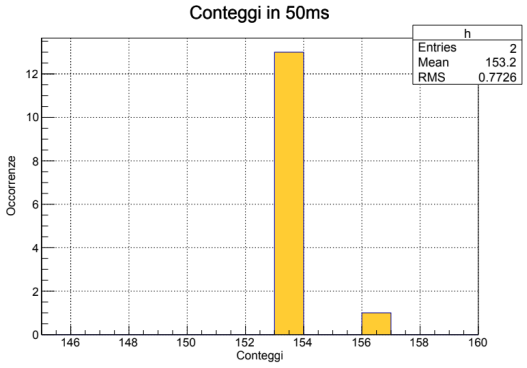


Figura 7: *Istogramma a 9ms fissati*



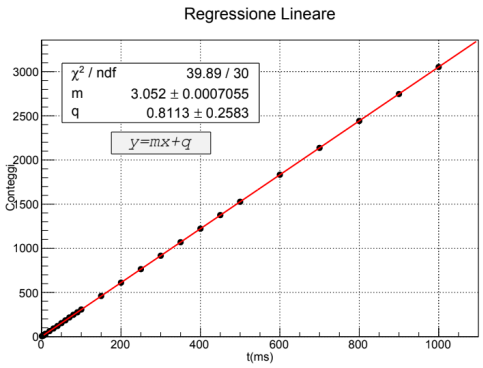
Si nota dunque che rispetto all'andamento che teoricamente ci si dovrebbe aspettare si ha un ritardo di circa $312ms$; questo valore di delay è stato ottenuto considerando il grafico teorico e traslandolo finché non si ottenga, qualitativamente, un buon fit con i dati che abbiamo raccolto.
Per verificare quantitativamente la validità della misura dell'impulso noto bisogna moltiplicare il numero di conteggi letto dal LED, (Figura 5), per la durata del clock, quindi: $5(conteggi) * 0,3114s = 1,557s$. La sensibilità dello strumento è di più o meno un conteggio, perciò, in secondi, di più o meno un periodo di clock $(311,4ms)$.

Figura 8: *Istogramma a 50ms*



I valori necessari alla verifica della risposta lineare del contatore sono riportati in Tabella 2. Il grafico della regressione lineare è il seguente (Figura 9):

Figura 9: *Regressione lineare FPGA*

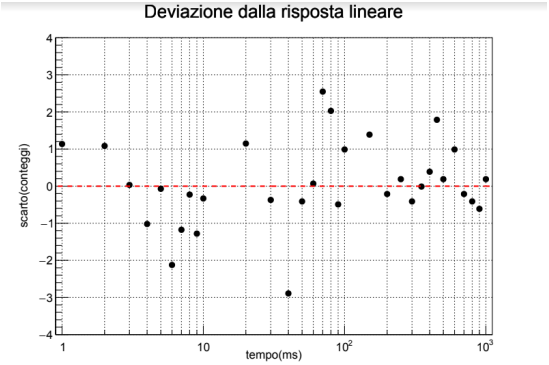


Contatore a 16 bit (FPGA)

I dati per quantificare l'oscillazione dei conteggi ad un tempo fissato, dovuta ad una instabilità in tensione nel momento in cui il segnale in output dalla scheda NI USB-6008 comincia, sono riportati in Tabella 3, a $9ms$ fissati, e Tabella 4, a $50ms$ fissati. I grafici corrispondenti sono rispettivamente i seguenti (Figura 7 e 8):

da cui si ottiene un valore della frequenza del cronometro di $(3052,0 \pm 0,7)Hz$.
La deviazione dalla risposta lineare si quantifica, invece, dalla seguente Figura 10:

Figura 10: *Deviazione dalla risposta lineare*



Tale grafico è stato ottenuto facendo la differenza, per ogni valore di tempo, fra il conteggio misurato nei nostri dati e l'ordinata della retta allo stesso valore di tempo.

IV. Conclusioni

Per quanto riguarda il contatore a 4 bit possiamo dire che la misura della durata dell'impulso noto di $1,5s$ risulta coerente con quanto misurato tramite il circuito realizzato, in quanto si è ottenuto $(1,6 \pm 0,3)s$.
Inoltre il delay del grafico in Figura 6 risulta molto simile al periodo del clock che abbiamo utilizzato; questo risultato, aggiunto al fatto che il contatore aumenta di uno il conteggio ad ogni fronte di salita del clock, invece che ad un istante qualsiasi in cui il livello del clock sia positivo o negativo (*level triggering*), ci porta ad ipotizzare che i flip-flop D utilizzati siano composti da due latch D messi in modalità Master-Slave *rising edge triggered*, quindi con ENABLE negato al primo latch e non negato al secondo. In definitiva ipotizziamo che il delay sia dovuto all'architettura Master-Slave che compone i flip-flop D utilizzati.
Dal grafico in Figura 9 abbiamo ottenuto, dal fit, che la frequenza del contatore a 16 bit è $(3052,0 \pm 0,7)Hz$ che è un valore coerente con la frequenza teorica $100MHz/2^{15} = 3052Hz$, inoltre abbiamo ottenuto che la deviazione dalla risposta lineare è di circa 3 conteggi, perciò di $1ms$, e lo scarto quadratico medio a $9ms$ e a $50ms$ è di circa $0,6$ conteggi, quindi circa $0,2ms$, il che ci porta a considerare la sensibilità del nostro strumento pari a, tramite somma in quadratura, $1ms$, che risulta essere migliore della sensibilità del cronometro a 4 bit, che era $0,3s$ cioè un ciclo di clock.

Tabella 1: *Dati per osservare delay nel contatore a 4 bit*

$Tempo(ms)$	$Conteggi$	$Tempo(ms)$	$Conteggi$
100	1	2800	9
200	1	2900	9
300	1	3000	9
400	1	3100	9
500	2	3200	10
600	2	3300	10
700	2	3400	10
800	2	3500	11
900	3	3600	11
1000	3	3700	11
1100	3	3800	12
1200	4	3900	12
1300	4	4000	12
1400	4	4100	12
1500	5	4200	13
1600	5	4300	13
1700	5	4400	13
1800	5	4500	14
1900	6	4600	14
2000	6	4700	14
2100	6	4800	15
2200	7	4900	15
2300	7	5000	15
2400	7	5100	15
2500	8	5200	1
2600	8	5300	1
2700	8	5400	1

Tabella 2: *Dati per regressione lineare FPGA*

$Tempo(ms)$	$Conteggi$
1	5
2	8
3	10
4	12
5	16
6	17
7	21
8	25
9	27
10	31
20	63
30	92
40	120
50	153
60	184
70	217
80	247
90	275
100	307
150	460
200	611
250	764
300	916
350	1069
400	1222
450	1376
500	1527
600	1833
700	2137
800	2442
900	2747
1000	3053

Tabella 3: *Dati istogramma a 9ms fissati*

Misura	*Conteggi*
1	27
2	27
3	27
4	27
5	27
6	28
7	28
8	27
9	27
10	27
11	27
12	27
13	28
14	27
15	27

Tabella 4: *Dati istogramma a* 50*ms fissati*

Misura	*Conteggi*
1	153
2	153
3	153
4	153
5	153
6	153
7	153
8	153
9	153
10	153
11	153
12	153
13	153
14	153
15	156

Figura 11: *codice in Verilog*

```
module esercizio1(
    input clk,
    output reg [15:0] led,
    input [15:0] sw,
    input [7:0] JA
);

    reg [30:0] counter;
    initial
        counter = 0;
    always @ (posedge clk) begin
            if (JA[0] != 0)
            begin
                led[0] <= counter[15];
                led[1] <= counter[16];
                led[2] <= counter[17];
                led[3] <= counter[18];
                led[4] <= counter[19];
                led[5] <= counter[20];
                led[6] <= counter[21];
                led[7] <= counter[22];
                led[8] <= counter[23];
                led[9] <= counter[24];
                led[10] <= counter[25];
                led[11] <= counter[26];
                led[12] <= counter[27];
                led[13] <= counter[28];
                led[14] <= counter[29];
                led[15] <= counter[30];
                counter <= counter + 1;
            end
            if (sw[0] == 0) begin
                led[0] <= 0;
                led[1] <= 0;
                led[2] <= 0;
                led[3] <= 0;
                led[4] <= 0;
                led[5] <= 0;
                led[6] <= 0;
                led[7] <= 0;
                led[8] <= 0;
                led[9] <= 0;
                led[10] <= 0;
                led[11] <= 0;
                led[12] <= 0;
                led[13] <= 0;
                led[14] <= 0;
                led[15] <= 0;
                counter <= 0;
            end
    end
endmodule
```

Figura 12: *VI contatore a 4 bit*

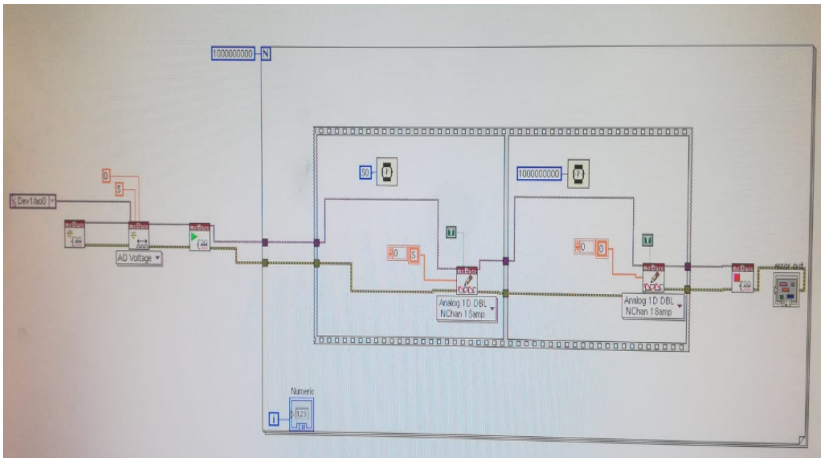


Figura 13: *VI contatore a 16 bit*