Parziale Progettazione di Sistemi Elettronici 14/11/24

Marco Losso

November 19, 2024

1 Introduzione

Questo report descrive il progetto di un cronometro sviluppato in Verilog. Il cronometro conta secondi, decimi e centesimi di secondo e include funzionalità come il reset, il start/stop, la visualizzazione del tempo parziale e il conteggio alla rovescia. Inoltre, sono state implementate funzioni per contare i minuti e le ore e per commutare tra diverse modalità di visualizzazione.

Il progetto Verilog è stato simulato utilizzando ModelSim e infine caricato sulla scheda DE1-SoC di Terasic per apprezzarne il risultato finale sui sei display a sette segmenti presenti su di essa.

2 Descrizione del Cronometro

Il cronometro di base è stato progettato per contare secondi, decimi e centesimi di secondo. Sono stati implementati due tasti principali: **reset** e **start/stop**. Il tasto di reset azzera il conteggio, mentre il tasto di start/stop permette di avviare e fermare il conteggio. Nel momento in cui il cronometro viene resettato, rimane fermo sul valore zero fino a quando il tasto di start/stop non viene premuto nuovamente per avviare il conteggio.

3 Clock Divider

Il progetto utilizza un **clock divider** per generare un segnale di clock a 100 Hz partendo da un segnale a 50 MHz interno alla scheda De1-SoC. Questo assicura una frequenza di aggiornamento adeguata per il conteggio dei centesimi di secondo. Ciò viene implementato tramite il conteggio del numero di fronti di salita del clock a 50 MHz, pari a 500.000 in 0.01 secondi: quindi 250.000 per il fronte alto e altrettanti per il fronte basso. Al termine del conteggio il valore del clock a 100 Hz in uscita viene negato per generare, appunto, i due stati alto e basso. Di seguito viene riportato per intero il modulo che implementa il **clock divider** ed è facile riconoscere le operazioni appena descritte.

Listing 1: Clock Divider

```
module clock_divider (
    input clk_in,     // Clock a 50 MHz
    output reg clk_out    // Clock a 100 Hz
);
    reg [18:0] counter;
    initial begin
        counter = 0;
        clk_out = 0;
    end

    always @(posedge clk_in) begin
        if (counter == 250000) begin
            counter <= 0;
            clk_out <= ~clk_out;
        end else begin
            counter <= counter + 1;
        end
    end
endmodule
```

3.1 Descrizione delle Variabili

- **clk_in**: Segnale di clock in ingresso a 50 MHz.
- **clk_out**: Segnale di clock in uscita a 100 Hz.
- **counter**: Contatore che tiene traccia dei cicli di clock per generare il clock in uscita.

4 Contatore

Successivamente si è passati alla realizzazione del conteggio vero e proprio, il tutto è stato realizzato con degli `if` annidati che introducono anche il conteggio dei minuti e delle ore.

Di seguito è stato riportato solo un estratto del modulo che implementa il **contatore** mettendo in luce proprio la struttura degli `if`. Inoltre, si evidenzia come all'attivarsi del `reset` questo pone il valore del conteggio a 0, e l'utilizzo dell'`enable` per fermare o far ripartire il conteggio implementando la funzione di start/stop.

Listing 2: Estratto del Modulo del Contatore

```
always @(posedge clk_100Hz or posedge rst) begin
    if (rst) begin
        // Reset del conteggio
        centisec <= 0;
        sec <= 0;
        min <= 0;
        hr <= 0;
        running <= 0;
    end else begin
        if (enable && !enable_reg) begin
            running <= ~running; // Cambia lo stato del contatore
        end
        enable_reg <= enable; // Aggiorna lo stato del tasto enable
        if (running) begin
            if (centisec == 99) begin
                centisec <= 0;
                if (sec == 59) begin
                    sec <= 0;
                    if (min == 59) begin
                        min <= 0;
                        if (hr == 23)
                            hr <= 0;
                        else
                            hr <= hr + 1;
                    end else
                        min <= min + 1;
                end else
                    sec <= sec + 1;
            end else
                centisec <= centisec + 1;
        end
    end
end
```

4.1 Descrizione delle Variabili

- **centisec**: Contatore dei centesimi di secondo.
- **sec**: Contatore dei secondi.
- **min**: Contatore dei minuti.
- **hr**: Contatore delle ore.
- **running**: Stato del contatore (avviato/arrestato).
- **enable_reg**: Registrazione dello stato precedente del tasto enable.

4.2 Prima simulazione

Prima di procedere al download su scheda è stata eseguita una simulazione di questa prima parte di progetto su Modelsim in modo da verificare il corretto funzionamento del contatore e anche dei segnali di `reset` e `enable`.

La figura sottostante mostra i risulatati della simualzione e soprattutto come i segnali di `reset` e `enable` interagiscano con il contatore.

Figure 1: Simulazione contatore in ModelSim

5 Implementazione dei Tempi Parziali

La funzionalità di visualizzazione del tempo parziale è stata aggiunta utilizzando un tasto dedicato. Quando il tasto di tempo parziale viene premuto, i valori attuali del cronometro vengono congelati e visualizzati sui display a 7 segmenti per 5 secondi, senza interrompere il conteggio reale. Questo consente di visualizzare il tempo parziale in qualsiasi momento, mantenendo il cronometro in esecuzione.

L'estratto di codice Verilog seguente mostra come i valori del cronometro vengono memorizzati e visualizzati per 5 secondi quando il tasto di tempo parziale viene premuto:

Listing 3: Gestione del Tempo Parziale

```
always @(posedge clk) begin
    if (parziale_mode) begin
        if (parziale_counter < 500) begin // 5 secondi a 100 Hz
            parziale_counter <= parziale_counter + 1;
        end else begin
            parziale_mode <= 0;
            parziale_counter <= 0;
        end
    end

    if (parziale && !parziale_mode) begin
        // Attiva la visualizzazione del tempo parziale
        // quando il tasto e' premuto
        parziale_mode <= 1;
        parziale_counter <= 0;
        // Memorizza i valori attuali del contatore
        parziale_centisec <= centisec;
        parziale_sec <= sec;
        parziale_min <= min;
        parziale_hr <= hr;
    end
end
```

5.1 Descrizione delle Variabili

- **parziale_mode**: Stato del tempo parziale.
- **parziale_counter**: Contatore che tiene traccia del tempo di visualizzazione del tempo parziale.
- **parziale_centisec**, **parziale_sec**, **parziale_min**, **parziale_hr**: Memorizzano i valori correnti del cronometro quando il tasto di tempo parziale viene premuto.

6 Implementazione del Conteggio all'Indietro

È stata aggiunta una funzionalità per consentire al cronometro di contare all'indietro utilizzando uno switch dedicato. Quando lo switch è attivato, il cronometro conta all'indietro; quando lo switch è disattivato, il cronometro conta in avanti.

6.1 Codice Verilog per il Conteggio all'Indietro

L'estratto di codice Verilog seguente mostra come il conteggio viene gestito all'indietro in base allo stato dello switch:

Listing 4: Gestione del Conteggio all'Indietro

```
if (direction) begin
    // Conteggio inverso
    if (centisec == 0) begin
        centisec <= 99;
        if (sec == 0) begin
            sec <= 59;
            if (min == 0) begin
                min <= 59;
                if (hr == 0)
                    hr <= 23;
                else
                    hr <= hr - 1;
            end else
                min <= min - 1;
        end else
            sec <= sec - 1;
    end else
        centisec <= centisec - 1;
end
```

6.2 Descrizione delle Variabili

- **direction**: Switch che determina la direzione del conteggio (0 = avanti, 1 = indietro).
- **centisec**, **sec**, **min**, **hr**: Variabili che memorizzano i valori del conteggio del cronometro.

7 Simulazione Finale del Progetto

La simulazione finale del progetto è stata implementata in un modulo top entity che include l'utilizzo dei tasti e degli switch veri e propri. Questo ha permesso di testare il comportamento del sistema in condizioni realistiche, utilizzando input fisici per controllare il funzionamento del contatore e la visualizzazione sui display a 7 segmenti.

I valori del contatore sono stati convertiti in BCD (Binary-Coded Decimal) per poter essere visualizzati correttamente sui display a 7 segmenti.

7.1 Descrizione dei Segnali Utilizzati

Nella simulazione finale, sono stati utilizzati vari segnali e bus per controllare il funzionamento del sistema:

- **CLOCK_50**: Segnale di clock a 50 MHz.
- **clk_100Hz**: Segnale di clock a 100 Hz generato dal clock divider.
- **KEY[0]**: Tasto di reset.
- **KEY[1]**: Tasto di enable per avviare e fermare il conteggio.
- **KEY[2]**: Tasto di tempo parziale per fermare la visualizzazione nei display a 7 segmenti.
- **HEX**: Bus per i display a 7 segmenti.
- **disp**: Segnale che controlla i display a 7 segmenti.
- **sec, centisec, min**: Variabili che memorizzano i valori del contatore.

7.2 Risultati della Simulazione

Le seguenti immagini mostrano i risultati della simulazione:

- Nella figura 2, oltre al tasto di reset KEY[0] utilizzato all'inizio, si vede l'impiego dell'enable KEY[1] che ferma il conteggio sia del contatore (linee gialle), sia quello raffigurato nei display a 7 segmenti (linee rosa).
- Nella figura 3 viene impiegato il tasto parziale `KEY[2]` che ferma solo la visualizzazione nei display a 7 segmenti (linee rosa) ma il conteggio del contatore prosegue (linee gialle).

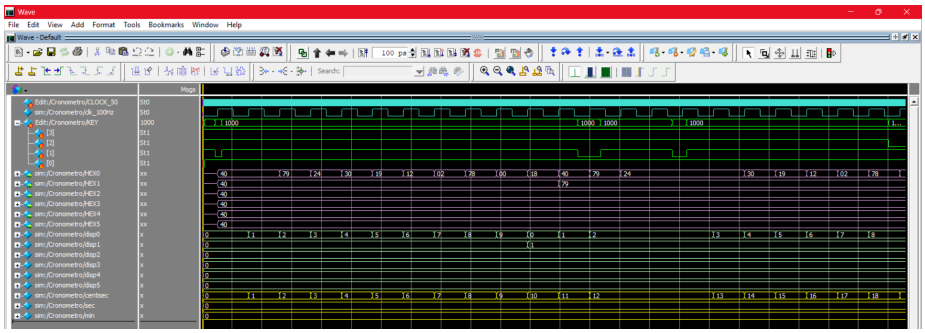

Figure 2: Simulazione che mostra l'impiego del tasto di reset (KEY[0]) e del tasto di enable (KEY[1]) per fermare il conteggio.

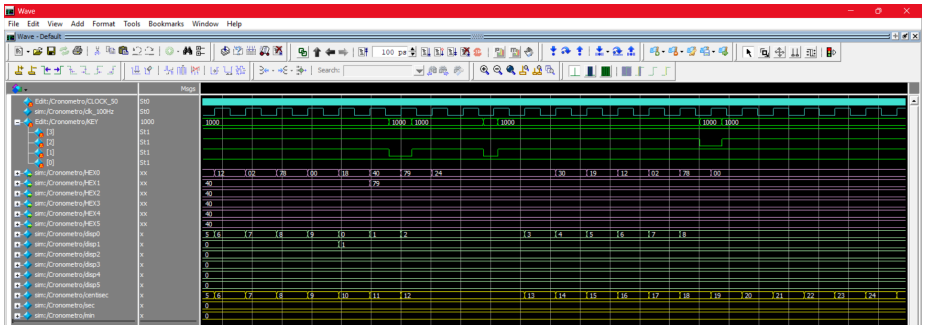

Figure 3: Simulazione che mostra l'utilizzo del tasto parziale (KEY[2]) per fermare solo la visualizzazione nei display a 7 segmenti (linee rosa), mentre il conteggio del contatore prosegue (linee gialle).

8 Modalità di Visualizzazione nei Display a 7 Segmenti

Dato che sono disponibili solo 6 display a 7 segmenti, è stata implementata una funzionalità che consente di passare tra due modalità di visualizzazione utilizzando il tasto KEY[3]. Questo tasto permette di switchare dalla visualizzazione dei centesimi di secondo, secondi e minuti (**centisec-sec-min**) a quella dei secondi, minuti e ore (**sec-min-hr**).

8.1 Codice Verilog per le Modalità di Visualizzazione

Di seguito è riportato un estratto del codice Verilog che gestisce le modalità di visualizzazione nei display a 7 segmenti:

Listing 5: Gestione delle Modalità di Visualizzazione

```
always @(posedge clk) begin
    if (key3) begin
        mode <= ~mode;
        // Cambia modalita' quando KEY[3] e' premuto
    end
end

always @(*) begin
    if (mode == 0) begin
        // Visualizza centisec-sec-min
        disp0 = centisec % 10;
        disp1 = centisec / 10;
        disp2 = sec % 10;
        disp3 = sec / 10;
        disp4 = min % 10;
        disp5 = min / 10;
    end else begin
        // Visualizza sec-min-hr
        disp0 = sec % 10;
        disp1 = sec / 10;
        disp2 = min % 10;
        disp3 = min / 10;
        disp4 = hr % 10;
        disp5 = hr / 10;
    end
end
```

La divisione per 10 viene fatta per convertire i valori in formato BCD (Binary-Coded Decimal). Nei display a 7 segmenti, i numeri devono essere visualizzati cifra per cifra in formato decimale. La divisione per 10 permette di separare le unità dalle decine per ogni valore da visualizzare, rendendo possibile mostrare ogni cifra individualmente su ciascun display.

8.2 Descrizione delle Variabili

- **key3**: Tasto utilizzato per cambiare modalità di visualizzazione.
- **mode**: Variabile che determina la modalità di visualizzazione (0 = centisec-sec-min, 1 = sec-min-hr).
- **disp0, disp1, disp2, disp3, disp4, disp5**: Variabili che controllano i display a 7 segmenti.
- **centisec, sec, min, hr**: Variabili che memorizzano i valori del conteggio.

9 Conclusioni

Il progetto sviluppato in Quartus ha raggiunto con successo l'obiettivo di creare un cronometro utilizzando Verilog. Le funzionalità implementate includono il conteggio dei centesimi di secondo, secondi, minuti e ore, l'uso di tasti per il reset e lo start/stop, la visualizzazione del tempo parziale, il conteggio all'indietro e la commutazione tra diverse modalità di visualizzazione sui display a 7 segmenti.

La simulazione finale del progetto è stata condotta utilizzando ModelSim, e l'implementazione è stata verificata sulla scheda DE1-SoC di Terasic. Sono stati effettuati tutti i test necessari sulla scheda hardware e il sistema ha funzionato perfettamente in ogni sua parte. Questo ha permesso di validare il funzionamento del cronometro in un ambiente realistico, garantendo che tutte le funzionalità operino correttamente.

Attraverso questo progetto, sono state acquisite essenziali competenze nella progettazione hardware utilizzando Verilog, nella simulazione di sistemi digitali e nell'integrazione di componenti hardware e software. La capacità di gestire segnali di clock, implementare logiche di conteggio e visualizzare dati su display a 7 segmenti sono state abilità cruciali sviluppate durante il corso del progetto.

Il successo del progetto apre la strada a potenziali miglioramenti futuri, come l'aggiunta di ulteriori modalità di conteggio, l'integrazione con altri dispositivi esterni e l'ottimizzazione delle prestazioni.

Questi miglioramenti potrebbero ulteriormente ampliare le capacità e le applicazioni del cronometro sviluppato.