Parziale Progettazione di Sistemi Elettronici 12/12/24

Marco Losso

November 12, 2024

Disclaimer

Il codice riportato in questo documento è stato semplificato per motivi di comprensione dello stesso. Esso non è funzionante così com'è e rappresenta più un'implementazione logica che sintattica. Pertanto, potrebbe essere necessario apportare modifiche per renderlo eseguibile e conforme alle specifiche reali del progetto.

1 Introduzione

Il progetto si propone di realizzare un orologio digitale con visualizzazione delle ore, dei minuti e dei secondi, regolabile tramite un encoder rotativo. Il sistema sfrutta diverse frequenze di clock per garantire una precisione elevata nel conteggio del tempo. Inoltre, sono state implementate funzionalità opzionali per migliorare l'esperienza utente, come la possibilità di scegliere tra il formato 12 ore e 24 ore, e l'aggiunta di un cronometro con rilevamento di "click brevi" e "click lunghi". Questa relazione descrive il processo di sviluppo del progetto, le sfide incontrate e le soluzioni adottate.

2 Configurazione dell'Hardware in Platform Designer

Il progetto è iniziato con una base già pronta in Platform Designer di Quartus, che ha permesso di avere moduli pre-configurati per i timer a diverse frequenze, l'interfacciamento con GPIO esterni per l'encoder rotativo, e moduli per manipolare i display a 7 segmenti. La configurazione iniziale dell'hardware è mostrata in Figura 1.

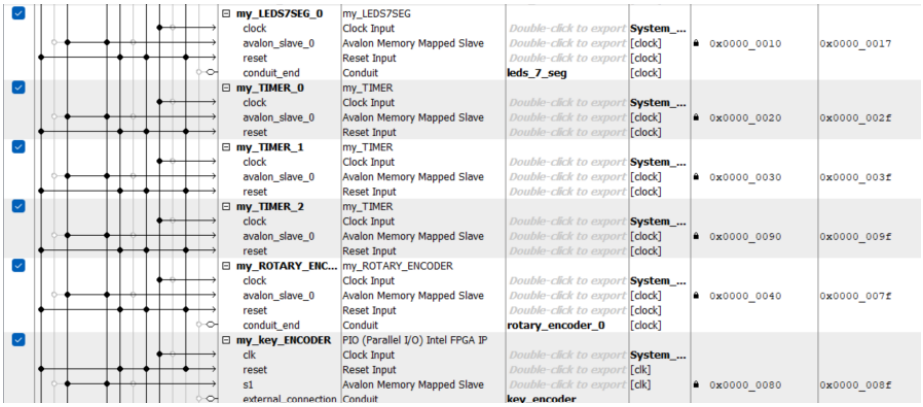

Figure 1: Estratto Struttura Platform Designer

Nella Figura 1, sono presenti vari moduli connessi tra loro:

- **my_LEDS7SEG_0**: Questo modulo controlla i 6 display a 7 segmenti. È collegato al clock, Avalon slave, reset, e conduits.
- **my_TIMER_0**: Un modulo timer che opera a 1Hz, collegato a clock, Avalon slave, e reset.
- **my_TIMER_1**: Simile a my_TIMER_0, ma opera a 2Hz.
- **my_TIMER_2**: Un altro modulo timer che opera a 100Hz, collegato a clock, Avalon slave, e reset.
- **my_ROTARY_ENCODER_0**: Modulo per interfacciarsi con un encoder rotativo, collegato a clock, Avalon slave, reset, e conduit.
- **my_key_ENCODER**: Modulo per interfacciarsi al tasto presente nell'encoder rotativo, collegato a clock, reset, s1, e connessioni esterne.

I timer menzionati (my_TIMER_0, my_TIMER_1, e my_TIMER_2) sono semplici contatori che scrivono il loro valore in un registro di memoria e operano rispettivamente a frequenze di 1Hz, 2Hz e 100Hz. Questi timer sono stati utilizzati nel programma C per gestire il conteggio preciso del tempo.

3 Modifiche al Codice Verilog HDL nella Top-Level Entity

Dopo aver configurato l'hardware in Platform Designer, sono state apportate alcune modifiche al codice Verilog HDL nella top-level entity per connettere il processore ARM con le periferiche esterne. Queste modifiche includono il collegamento dei display a 7 segmenti e dell'encoder rotativo. Il codice Verilog aggiunto è il seguente:

```
module Top_Level_Entity
...
// External Connection
.leds_7_seg_out_dispa          (HEX0),  // leds_7_seg.out_dispa
.leds_7_seg_out_dispb          (HEX1),  //           .out_dispb
.leds_7_seg_out_dispc          (HEX2),  //           .out_dispc
.leds_7_seg_out_dispd          (HEX3),  //           .out_dispd
.leds_7_seg_out_dispe          (HEX4),  //           .out_dispe
.leds_7_seg_out_dispf          (HEX5),  //           .out_dispf

.rot_encoder_data1 (GPIO1GPIO[0]),  // rotary_encoder_0.data1
.rot_encoder_data1_reg   (),        //              .data1_reg
.rot_encoder_data2 (GPIO1GPIO[1]),  //                  .data2
.rot_encoder_data2_reg   (),        //              .data2_reg

.key_encoder_export     (GPIO1GPIO[2])   // key_encoder.export

endmodule
```

3.1 Descrizione delle Modifiche

Le modifiche al codice Verilog HDL riguardano i collegamenti esterni per il controllo dei display a 7 segmenti e l'interfacciamento con l'encoder rotativo:

- **Display a 7 segmenti**:
  - **leds_7_seg_out_dispa** - Collega il display a 7 segmenti al pin HEX0.
  - **leds_7_seg_out_dispb** - Collega il display a 7 segmenti al pin HEX1.
  - **leds_7_seg_out_dispc** - Collega il display a 7 segmenti al pin HEX2.
  - **leds_7_seg_out_dispd** - Collega il display a 7 segmenti al pin HEX3.
  - **leds_7_seg_out_dispe** - Collega il display a 7 segmenti al pin HEX4.
  - **leds_7_seg_out_dispf** - Collega il display a 7 segmenti al pin HEX5.
- **Encoder rotativo**:
  - **rotary_encoder_0_data1** - Collega il pin GPIO1GPIO[0] per leggere i dati dell'encoder.
  - **rotary_encoder_0_data2** - Collega il pin GPIO1GPIO[1] per leggere i dati dell'encoder.
- **Key encoder**:
  - **key_encoder_export** - Collega il pin GPIO1GPIO[2] per leggere i dati del tasto dell'encoder.

Queste connessioni permettono di interfacciare il processore ARM con i display a 7 segmenti e l'encoder rotativo, consentendo il controllo e la visualizzazione dei dati in tempo reale.

4 Modifiche al Modulo Verilog per il Display a 7 Segmenti

Per il controllo dei display a 7 segmenti, è stato utilizzato un modulo Verilog che codifica i vari valori da visualizzare. Nella codifica del modulo, l'uscita "a" corrisponde alla situazione in cui tutti i LED del display sono spenti, mentre l'uscita "b" indica la visualizzazione della lettera "h". Il codice Verilog aggiornato del modulo è il seguente:

```
begin
    case(iDIG)
        4'h1: oSEG = 7'b1111001;    // ---t----
        4'h2: oSEG = 7'b0100100;    // |     |
        4'h3: oSEG = 7'b0110000;    // lt    rt
        4'h4: oSEG = 7'b0011001;    // |     |
        4'h5: oSEG = 7'b0010010;    // ---m----
        4'h6: oSEG = 7'b0000010;    // |     |
        4'h7: oSEG = 7'b1111000;    // lb    rb
        4'h8: oSEG = 7'b0000000;    // |     |
        4'h9: oSEG = 7'b0011000;    // ---b----
        4'ha: oSEG = 7'b1111111;    // => Tutti i LED spenti
        4'hb: oSEG = 7'b0001011;    // => Visualizzazione "h"
    endcase
end
```

4.1 Descrizione delle Modifiche

Nel codice sopra riportato, le linee **4'ha** e **4'hb** sono state modificate per gestire la situazione in cui tutti i LED del display devono essere spenti e per visualizzare la lettera "H":

- **4'ha: oSEG = 7'b1111111;** - Questa linea di codice imposta tutti i bit dell'uscita oSEG a 1, spegnendo così tutti i LED del display.
- **4'hb: oSEG = 7'b0001011;** - Questa linea di codice imposta l'uscita oSEG per visualizzare la lettera "H" sul display a 7 segmenti.

5 Implementazione del Codice C

5.1 Definizione degli Indirizzi di Memoria

La prima parte del codice definisce gli indirizzi di memoria per le diverse periferiche utilizzate nel progetto. Questi indirizzi sono necessari per interfacciarsi con l'encoder rotativo, il display a 7 segmenti, i timer e il pulsante dell'encoder.

```
// Definizione degli indirizzi di memoria
#define ENCODER_ADDR 0xff200040
#define DISPLAY_ADDR 0xff200010
#define TIMER_ADDR 0xff200020
#define BUTTON_ADDR 0xff200080
#define TIMER_LAMP_ADDR 0xff200030
#define TIMER_CRON 0xff200090
```

5.2 Implementazione dell'Orologio

Una delle principali funzionalità del progetto è l'implementazione dell'orologio. La funzione che aggiorna l'orologio è responsabile del conteggio dei secondi, minuti e ore, e della visualizzazione dei dati sui display a 7 segmenti.

```
if (old_click != new_click) {
// Se il valore del clk a 1 Hz e' cambiato, si aggiorna l'ora
    sec++;
    if (sec >= 10) {
        sec = 0;
        decisec++;
        if (decisec >= 6) {
            decisec = 0;
            min++;
            if (min >= 10) {
                min = 0;
                decimin++;
                if (decimin >= 6) {
                    decimin = 0;
                    ore++;

                    if (ore >= 10) {
                        ore = 0;
                        deciore++;
                    }
                }
            }
        }
    }
```

```
    // Verifica del caso speciale per ore >= 24 o 12
    if (formato_ora == 24) {
        if (deciore >= 2 && ore >= 4) {
            deciore = 0;
            ore = 0;
        }
    } else { // Formato 12 ore
        if (deciore >= 1 && ore >= 2) {
            // Gestisce il passaggio da 11:59:59 a 12:00:00
            ore = 0;
            deciore = 0;
        }
    }
}

if (deciore != 0) {
// Discriminazione del caso in cui le ore sono maggiori di 10
// in modo da tenere spento il display piu' significativo
    output_display = 0xAA000000 | (deciore << 20) | (ore << 16)
                                | (decimin << 12) | (min << 8)
                                | (decisec << 4) | sec;
} else {
    output_display = 0xAAA00000 | (deciore << 20) | (ore << 16)
                                | (decimin << 12) | (min << 8)
                                | (decisec << 4) | sec;
}
```

La funzione **aggiornamento_orologio** aggiorna il tempo e gestisce il formato dell'ora (12 ore o 24 ore). Se il valore del registro gestito dal timer a 1 Hz cambia, i secondi vengono incrementati, e a loro volta incrementano i minuti e le ore quando necessario. Il display viene aggiornato di conseguenza. Questa funzione è fondamentale per mantenere il conteggio preciso del tempo e aggiornare correttamente i display a 7 segmenti. Si vede, inoltre, come vengano gestiti due formati di visualizzazione dell'ora: 12 o 24.

5.3 Modifica dell'ora tramite Encoder Rotativo e implementazione lampeggio

Per permettere all'utente di modificare l'ora dell'orologio digitale utilizzando l'encoder rotativo, è stato implementato un meccanismo che rileva la pressione del pulsante e la rotazione dell'encoder. Inoltre, durante la modifica, il display della cifra in fase di regolazione lampeggia per indicare all'utente quale elemento è in corso di modifica.

5.3.1 Gestione dello Stato nel Main

Nel **main**, il codice seguente gestisce lo stato del sistema per passare tra le varie modalità di modifica (secondi, minuti, ore) e rileva la pressione del pulsante dell'encoder:

```
if (pulsante_precedente == 1 && pulsante_corrente == 0) {
// Rileva pressione del pulsante
    if (visualizzazione == 0){
        stato = (stato + 1) % 5; // aggiornamento stato
    }
}

// aggiornamento pulsante
pulsante_precedente = pulsante_corrente;

if(stato == 1){
// se lo stato e' 1 si passa alla selezione del
//formato dell'ora prima di poterla modificare
    formato_ora = seleziona_formato_ora(formato_ora);
} else if(stato != 0 && stato != 1){
// se lo stato e' diverso da 0 o da 1 si passa alla modifica
// delle variabili
    modifica_orologio(se, min, ore, formato_ora);
}
```

5.3.2 Funzione di Modifica dell'Orologio

La funzione **modifica_orologio** è responsabile della regolazione delle variabili dell'orologio (secondi, minuti, ore) e del lampeggio del display:

```
void modifica_orologio(int *clk_2Hz_ptr,
                       int *sec, int *min, int *ore,
                       int stato, int formato_ora) {

    if ((clk_2Hz_curr % 2) == 0){
    // Quando il valore nel registro e' pari i display
    // della variabile sotto modifica vengono spenti

    switch (stato) {

            case 2:
                // Spegne i display dei secondi
                output_display &= 0xFFFFFF00;
                // Lampeggia i display dei secondi
                output_display |= 0x000000AA;
                modifica_sec_min(sec);
                break;
```

```
            case 3:
                // Spegne i display dei minuti
                output_display &= 0xFFFF00FF;
                // Lampeggia i display dei minuti
                output_display |= 0x0000AA00;
                modifica_sec_min(min);
                break;

            case 4:
                // Spegne i display delle ore
                output_display &= 0xFF00FFFF;
                // Lampeggia i display delle ore
                output_display |= 0x00AA0000;
                modifica_ore(ore, formato_ora);
                break;
        }
    } else {
    // Permette la modifica delle variabili
    // mantenendo i display accesi
        switch (stato) {
            case 2:
                modifica_sec_min(sec);
                break;

            case 3:
                modifica_sec_min(min);
                break;

            case 4:
                modifica_ore(ore, formato_ora);
                break;
        }
    }
}
```

La funzione **modifica_orologio** utilizza i valori dell'encoder per regolare secondi, minuti e ore. Durante la modifica, il display della cifra in fase di regolazione lampeggia, rendendo chiaro all'utente quale elemento è in corso di modifica.

5.4 Regolazione del Formato dell'Ora

Nel progetto, l'utente ha la possibilità di scegliere tra due formati di visualizzazione dell'ora: il formato a 12 ore e quello a 24 ore. Parte di questa implementazione è visibile nella selezione dello stato nel **main** (come descritto nella sottosezione precedente) e nella funzione di aggiornamento dell'orologio classico. Di seguito viene riportata la funzione **seleziona_formato_ora**, che permette di scegliere il formato dell'ora tramite il movimento dell'encoder rotativo.

```
int seleziona_formato_ora(int formato_corrente) {

    // Aggiorna il display per mostrare il
    // formato selezionato
    if (formato_corrente == 24) {
    // Viene visualizzato h 24
        output_display = 0xAAABAA24;
    } else {
    // Viene visualizzato h 12
        output_display = 0xAAABAA12;
    }

    // Cambia formato ora tra 12 e 24 ad
    // ogni click dell'encoder
    if (encoder_reg_curr != encoder_reg_prev) {
        if (formato_corrente == 24) {
            formato_corrente = 12;
        } else {
            formato_corrente = 24;
        }
        encoder_reg_prev = encoder_reg_curr;
    }

    return formato_corrente;
}
```

La funzione **seleziona_formato_ora** rileva i movimenti dell'encoder rotativo per modificare il formato dell'ora. Ad ogni click dell'encoder, il formato dell'ora viene cambiato tra 12 ore e 24 ore. Il display viene aggiornato di conseguenza per mostrare il formato selezionato.

Questa funzionalità aggiunge flessibilità al sistema, permettendo all'utente di scegliere il formato dell'ora preferito in modo semplice e intuitivo.

5.5 Realizzazione del Cronometro

Il cronometro è stato implementato per funzionare in parallelo all'orologio. È possibile passare dalla visualizzazione dell'orologio a quella del cronometro tramite l'encoder rotativo. Un click breve sull'encoder permette di avviare o fermare il cronometro, mentre un click prolungato resetta il cronometro.

5.5.1 Gestione della Visualizzazione e Comandi del Cronometro

Nel **main**, la logica per gestire la visualizzazione e i comandi del cronometro è la seguente.

Di seguito viene riportata la funzione **aggiorna_display** che gestisce proprio la visualizzazione sui display a 7 segmenti dei dati dell'orologio o del cronometro.

```
if(stato == 1){
    // se lo stato e' 1 si passa alla selezione
    // del formato dell'ora prima di poterla modificare
} else if(stato != 0 && stato != 1){
    // se lo stato e' diverso da 0 o da 1 si passa alla
    // modifica delle variabili
} else {
    // altrimenti si rimane nello stato di default
    if (encoder_reg_curr != encoder_reg_prev) {
    // Toggle visualizzazione tra orologio e cronometro
        visualizzazione = !visualizzazione;
    }

  display_o = aggiornamento_orologio(clk_1Hz, formato_ora);
  display_c = aggiorna_cronometro(clk_100Hz, visualizzazione);

  // Aggiorna il display in base alla visualizzazione corrente
  aggiorna_display(visualizzazione, display_o, display_c);
}

void aggiorna_display(int vis, int out_o, int out_c) {
    if (vis == 0) {
        out_o; // Visualizza l'orologio
    } else {
        out_c; // Visualizza il cronometro
    }
}
```

5.5.2 Funzione di Aggiornamento del Cronometro

La funzione **aggiorna_cronometro** gestisce il conteggio del tempo per il cronometro e rileva la durata di pressione del pulsante per implementare il reset, oltre che alle funzionalità di base di start e stop.

```
int aggiorna_cronometro (int *clk_100Hz, int vis) {

    if(vis == 1) {
    // rileva il pulsante solo se siamo
    // nella visualizzazione del cronometro
        pulsante_corrente = *button_ptr;
        if (pulsante_precedente == 1
                && pulsante_corrente == 0
                && vis == 1) {
            // Reset del tempo di pressione
            tempo_premuto = 0;
        }
    }
    if (pulsante_corrente == 0) {
        if (new_clk_100Hz != old_clk_100Hz) {
            // Incrementa il tempo di pressione
            tempo_premuto++;
        }
    } else if (pulsante_precedente == 0
                    && pulsante_corrente == 1) {
        // Rileva rilascio del pulsante
        if (tempo_premuto >= 100) {
            // Pulsante premuto per piu' di un secondo
            // (100 tick di 100Hz)
            *centesimi = 0;
            *secondi = 0;
            *minuti = 0;
            // Reset del cronometro
            *cronometro_attivo = 0;
        } else {
        // Pulsante premuto per meno di un secondo
            *cronometro_attivo = !(*cronometro_attivo);
        }
    }

    if (new_clk_100Hz != old_clk_100Hz
            && *cronometro_attivo) {
        // Se il valore del clk a 100 Hz e' cambiato e
        // il cronometro e' attivo, aggiorno il cronometro
        (*centesimi)++;
        if (*centesimi >= 100) {
            *centesimi = 0;
            (*secondi)++;
```

```
            if (*secondi >= 60) {
                    *secondi = 0;
                    (*minuti)++;
            }
        }
    }

    // Reset del cronometro se raggiunge 59:59:99
    if (*minuti >= 59) {
        *minuti = 0;
        *secondi = 0;
        *centesimi = 0;
    }

    // Costruzione output display
    output_display =

      (*minuti >= 10 ? *minuti / 10 * 0x100000 : 0xAAA00000)
    | (*minuti == 0 ? 0xAA0000 : *minuti % 10 * 0x010000)
    | (*minuti == 0 && *secondi < 10 ?
           0xAAA000 : *secondi / 10 * 0x001000)
    | (*secondi % 10 * 0x000100)
    | (*centesimi / 10 * 0x000010)
    | (*centesimi % 10 * 0x000001);

    // Restituisce il valore di output_display
    return output_display;
}
```

Questa funzione aggiorna il cronometro in base ai tick del timer a 100Hz, permettendo un conteggio preciso dei centesimi di secondo. Gestisce inoltre i click brevi e prolungati del pulsante per avviare/fermare e resettare il cronometro.

Nella costruzione della variabile di output per il display a 7 segmenti vengono effettuati controllo con operatori logici sui valori delle variabili per far in modo che i display relativi alle cifre più significative stiano spenti per una visualizzazione più gradevole.

6 Conclusioni

Il progetto ha portato alla realizzazione di un orologio digitale completo di funzionalità avanzate, come la regolazione dell'ora tramite encoder rotativo, il lampeggiamento delle cifre durante la modifica e la possibilità di scegliere tra formati a 12 e 24 ore. Inoltre, è stato integrato un cronometro funzionante in parallelo all'orologio, che offre all'utente la possibilità di avviare, fermare e resettare il conteggio con semplicità.

L'utilizzo di una base hardware pre-configurata in Platform Designer di Quartus ha permesso di accelerare lo sviluppo, concentrandosi sulle funzionalità di alto livello e sull'interfacciamento con le periferiche esterne. Le modifiche al codice Verilog HDL e l'implementazione del programma in C sono state fondamentali per garantire il corretto funzionamento del sistema e per offrire un'interfaccia utente intuitiva ed efficiente.

La flessibilità e la precisione raggiunte attraverso i blocchi my_TIMERS e le varie funzioni di gestione del display dimostrano l'efficacia dell'approccio adottato. Le funzionalità opzionali implementate, come la discriminazione tra click brevi e click lunghi, arricchiscono ulteriormente l'esperienza utente, rendendo il sistema versatile e adattabile a diverse esigenze.

In conclusione, questo progetto rappresenta un esempio pratico di come combinare hardware e software per creare sistemi embedded complessi e funzionali, ponendo solide basi per future espansioni e miglioramenti.