Implementare Generator Semnal PWM

Ciobanu Florinela-Bianca Călimoceanu Răzvan Grupa 333AA

1 Introducere

Proiectul vizează dezvoltarea unui Periferic Generator de Semnal PWM (Pulse Width Modulation) format din cinci componente majore: **Bridge-ul SPI**, **Decodorul de Instrucțiuni**, **Fișierul de Registri**, **Numărătorul** și **Generatorul PWM**. Această documentație prezintă implementarea modulară a perifericului, concentrându-se pe interfața de comunicație și pe structura logică a întregului sistem.

2 Bridge-ul de Comunicație SPI (`spi_bridge`)

Modulul `spi_bridge` acționează ca interfață **Slave SPI**, convertind datele seriale primite (MOSI) în date paralele și pregătind datele interne paralele pentru transmisia serială (MISO).

2.1 Reguli SPI Implementate

Implementarea respectă următoarele reguli de timing:

- **CPOL=0, CPHA=0:** Datele sunt **scrise** pe frontul descrescător al ceasului (`sclk`) și sunt **citite** pe frontul crescător.
- **MSB First:** Bitul cel mai semnificativ este transmis primul.
- **Sincronizare ceas:** S-a considerat că ceasul SPI (`sclk`) și ceasul perifericului (`clk`) sunt **sincrone** (`10MHz`).

2.2 Logica de Recepție (MOSI)

Logica de recepție rulează pe frontul crescător al ceasului SPI.

- **Shift Register (`shift_in`):** La fiecare front crescător al ceasului (`always @(posedge sclk)`), bitul `mosi` este preluat și shiftat la stânga.
- **Generare Sincronizare Byte (`byte_sync`):** După 8 biți primiți, semnalul `byte_sync` este pulsat timp de un ciclu.
- **Ieșirea Paralelă (`data_in`):** Byte-ul recepționat este pus la dispoziția decodorului.

2.3 Logica de Transmitere (MISO)

Logica de transmisie rulează pe frontul descrescător al ceasului SPI.

- **Tri-state:** `miso` este deconectat (`1'bZ`) când `cs_n` este High.
- **Preload:** Când slave-ul este inactiv, registrul `shift_out` este încărcat cu `data_out`.
- **Transmisie:** La fiecare negedge de `sclk`, se transmite MSB-ul curent.

3 Decodorul de Instrucțiuni (`instr_dcd`)

Decodorul funcționează ca un **Automat cu Stări Finite (FSM)** cu două stări, sincronizat de pulsul `byte_sync`. Citește două faze:

3.1 Byte de Setup

Primul byte definește:

- **Bitul 7 (Read/Write)**: 1 = scriere, 0 = citire.
- **Bitul 6 (High/Low)**: segmentul MSB sau LSB al registrului.
- **Biții 5:0 (Adresa)**: adresa registrului.

3.2 Byte de Date

În funcție de operație:

- **Scriere**: `data_write` devine valid, `write` pulsează.
- **Citire**: `data_out` primește valoarea din `data_read`.

FSM-ul revine apoi în starea inițială.

4 Fișierul de Regiștri (`regs`)

Modulul `regs` implementează fișierul configurabil de registre al perifericului. Acesta reprezintă interfața principală între software (prin SPI) și hardware (counter și generatorul PWM).

4.1 Responsabilități

- Interpretarea operațiilor de citire și scriere primite de la decodor.
- Maparea fiecărui registru la adrese pe un byte (inclusiv utilizarea segmentelor LSB/MSB).
- Expunerea valorilor către modulele `counter` și `pwm_gen`.
- Implementarea registrului `COUNTER_RESET` cu auto-clear.
- Menținerea semnalului read-only `COUNTER_VAL`.

4.2 Comportament

- Scrierile se efectuează doar când `write = 1`.
- Citirile returnează valoarea corectă, altfel 0 pentru adrese invalide.
- Resetul global readuce toate registrele la valorile implicite.

5 Numărătorul (`counter`)

Modulul `counter` furnizează baza de timp pentru generarea semnalului PWM, implementând numărătoarea controlată prin registrele configurabile.

5.1 Funcționalitate

- Numărare crescătoare sau descrescătoare în funcție de `UPNOTDOWN`.
- Limitarea prin valoarea `PERIOD`.
- Gestionarea overflow-ului și underflow-ului:
  - Dacă se numără în sus și se atinge `PERIOD`, contorul revine la 0.
  - Dacă se numără în jos și se atinge 0, contorul revine la `PERIOD`.
- Prescaler implementat ca exponent: update la fiecare $2^{prescale}$ cicluri.
- Resetare instantanee a contorului la activarea `COUNTER_RESET`.

5.2 Comportament

- `EN = 0` oprește contorul.
- `EN = 1` permite avansarea contorului pe baza prescalerului.

6 Generatorul PWM (`pwm_gen`)

Modulul `pwm_gen` generează semnalul PWM final pe baza valorilor furnizate de blocul de registre și de numărător.

6.1 Moduri de funcționare

- **Mod aliniat** (`FUNCTIONS[1] = 0`):
  - Aliniere la stânga dacă `FUNCTIONS[0] = 0`.
  - Aliniere la dreapta dacă `FUNCTIONS[0] = 1`.
  - Semnalul este inițializat la începutul unui nou ciclu (când `count_val = 0`).
  - Starea se schimbă la atingerea `COMPARE1`.
- **Mod nealiniat** (`FUNCTIONS[1] = 1`):

- Semnalul începe la 0.
- Devine 1 la `COMPARE1`.
- Revine la 0 la `COMPARE2`.

6.2 Control prin PWM_EN

- Dacă `PWM_EN = 0`, ieșirea rămâne blocată în ultima stare (hold state).
- Dacă `PWM_EN = 1`, semnalul PWM urmează logica modului configurat.