# Raport Tehnic: Proiectarea si Implementarea unui Periferic PWM Controlat prin Interfata SPI

**Echipa de proiect:**
Mirica Alin-Marian
Marin-Sirbu Alex-Florin
Badea Mihai

**Grupa:** 332AA
*Facultatea de Automatica si Calculatoare*

November 23, 2025

# Contents

# 1 1. Introducere

Acest document prezinta in detaliu solutiile ingineresti si deciziile de arhitectura adoptate pentru dezvoltarea unui modul hardware generator de PWM (Pulse Width Modulation) pe FPGA. Proiectul a fost realizat respectand specificatiile stricte ale temei, punand accent pe modularitate, eficienta resurselor si robustetea interfetei de comunicatie.

Obiectivul principal a fost crearea unui sistem capabil sa comunice cu un master extern prin protocolul SPI (Serial Peripheral Interface), sa interpreteze comenzi complexe de configurare si sa genereze forme de unda precise, variabile in factor de umplere si frecventa. In cele ce urmeaza, vom analiza structura interna a fiecarui modul component, evidentiind optimizarile realizate la nivel de Register Transfer Level (RTL).

# 2 2. Arhitectura Generala a Sistemului

Sistemul a fost descompus in cinci blocuri functionale interconectate, fiecare avand o responsabilitate bine definita. Aceasta abordare modulara ne-a permis testarea izolata a fiecarei componente inainte de integrarea in modulul de top (`top.v`).

Fluxul de date este urmatorul: semnalele seriale brute intra prin interfata fizica, sunt deserializate de bridge-ul SPI, decodificate de unitatea de control, stocate in banca de registri si utilizate apoi de numarator si generatorul PWM pentru a sintetiza semnalul de iesire.

# 3 3. Analiza Detaliata a Implementarii

## 3.1 3.1. Adaptarea Interfetei Fizice si Sincronizarea (spi_bridge.v)

Modulul `spi_bridge.v` reprezinta prima linie de interfata cu exteriorul. Aici am intampinat si rezolvat cea mai critica problema de compatibilitate a proiectului: definirea pinilor in scheletul de testare.

In specificatiile initiale ale fisierului `top.v`, directia pinilor pentru interfata SPI a fost definita invers fata de standardul unui Slave: pinul `miso` era declarat ca intrare (input), iar `mosi` ca iesire (output). Intr-o arhitectura standard, Master-ul comanda linia MOSI (Master Out Slave In), iar Slave-ul raspunde pe MISO (Master In Slave Out).

Pentru a rezolva acest conflict fara a modifica antetul modulului Top (actiune interzisa de regulamentul temei), am recurs la o strategie de rutare incrucisata (cross-routing) in momentul instantierii:

- Pinul fizic de intrare al FPGA-ului (etichetat eronat `miso`) a fost conectat la portul de intrare `mosi` al logicii interne.
- Iesirea logicii interne `miso` a fost conectata la pinul fizic de iesire al FPGA-ului (etichetat eronat `mosi`).

O alta functie vitala a acestui modul este **sincronizarea domeniilor de ceas**. Semnalele externe `sclk` (ceasul SPI) si `cs_n` (Chip Select) sunt asincrone fata de ceasul de sistem al FPGA-ului (10 MHz). Utilizarea lor directa in logica secventiala ar duce la violari de timp si metastabilitate. Am implementat un etaj de sincronizare folosind registri de deplasare cu 3 biti. Acestia au rol dublu: 1. Trecerea semnalului prin multiple flip-flop-uri pentru a stabiliza valoarea. 2. Detectia fronturilor (rising edge si falling edge) pentru a esantiona datele exact in momentul valid.

## 3.2 3.2. Decodificarea Instructiunilor si Adresarea Extinsa (instr_dcd.v)

Unitatea de decodificare, implementata in `instr_dcd.v`, actioneaza ca un automat de stari finit (FSM) cu doua stari principale: `ST_SETUP` si `ST_DATA`.

O provocare specifica a fost gestionarea spatiului de adrese. Registrii perifericului (precum `PERIOD`, `COMPARE1`) au o latime de 16 biti, insa protocolul SPI transfera date in pachete de cate 8 biti. Pentru a permite accesul complet fara a complica protocolul, am implementat o logica de adresare dependenta de bitul 6 al octetului de comanda:

- Daca Bit 6 = 0, decodorul selecteaza octetul inferior (LSB) al registrului tinta.
- Daca Bit 6 = 1, decodorul adauga automat un offset de +1 la adresa de baza, selectand octetul superior (MSB).

Aceasta abstractie permite software-ului de control sa gandeasca in termeni de "registre logice", lasand hardware-ul sa gestioneze fragmentarea datelor.

## 3.3 3.3. Banca de Registri si Logica de Auto-Clear (regs.v)

Modulul `regs.v` gestioneaza stocarea configuratiei. Pe langa functiile standard de citire/scriere, am implementat comportamente specifice pentru anumiti registri critici.

Cel mai notabil este registrul `COUNTER_RESET` (adresa 0x07). Acesta a fost proiectat cu o functionalitate de "auto-clearing". In momentul in care master-ul scrie valoarea '1' la aceasta adresa, semnalul de reset intern catre numarator este activat pentru exact un ciclu de ceas. Imediat in ciclul urmator, hardware-ul reseteaza automat bitul din registru la '0'. Aceasta abordare elimina necesitatea ca master-ul sa trimita o a doua comanda SPI pentru a dezactiva resetul, reducand latenta si incarcarea magistralei.

## 3.4 3.4. Numaratorul si Optimizarea Prescaler-ului (counter.v)

Baza de timp a sistemului este asigurata de modulul `counter.v`. Pentru a oferi flexibilitate in alegerea frecventei PWM, am integrat un prescaler configurabil.

Formula specificata pentru prescaler este $Limit = 2^{prescale}$. Implementarea directa a ridicarii la putere intr-un circuit digital este ineficienta, consumand multe resurse logice (LUT-uri). Am optimizat acest calcul folosind operatorul de deplasare logica pe biti:

$$\texttt{limit} = 1 \ll \texttt{prescale} \tag{1}$$

Aceasta operatie este triviala pentru un FPGA, fiind realizata prin multiplexare simpla. Numaratorul principal este actualizat doar atunci cand prescaler-ul atinge aceasta limita, permitand generarea unor frecvente foarte joase pornind de la ceasul de 10 MHz.

De asemenea, numaratorul suporta moduri de numarare "up" (crescator) si "down" (descrescator), controlate de semnalul `upnotdown`, esentiale pentru generarea formelor de unda simetrice sau asimetrice.

### 3.5 3.5. Generatorul de Forme de Unda (pwm_gen.v)

Ultimul etaj, `pwm_gen.v`, este responsabil de sinteza semnalului de iesire. Acesta este implementat folosind logica pur combinationala pentru a asigura un raspuns instantaneu la schimbarile starii numaratorului.

Am implementat logica pentru trei moduri de functionare distincte:

1. **Left Aligned:** Impulsul incepe la startul perioadei.
2. **Right Aligned:** Impulsul se termina la finalul perioadei.
3. **Unaligned:** Impulsul este pozitionat liber in interiorul perioadei, definit de intervalul $[Compare1, Compare2)$.

Pentru siguranta sistemului, am inclus o logica de "gating" conditionata de semnalul `pwm_en`. Daca acest bit de activare este 0, iesirea fizica este fortata la nivel logic 0, suprascriind orice rezultat al comparatoarelor interne. Acest lucru previne aparitia unor pulsuri eronate (glitches) in timpul configurarii sistemului.

## 4 4. Concluzii

Proiectul a demonstrat capacitatea echipei de a livra o solutie hardware complexa, care integreaza comunicatii asincrone, logica secventiala avansata si procesare de semnal. Solutiile adoptate, in special "hack-ul" ingenios pentru corectia pinilor in `top.v` si sincronizarea robusta din `spi_bridge.v`, au fost esentiale pentru trecerea testelor automate. Sistemul rezultat este unul modular, eficient din punct de vedere al resurselor si complet functional conform specificatiilor.