Documentatie periferic PWM controlat prin SPI

Alin-Marian Mirica
Alex-Florin MARIN-SIRBU
Pavel-Mihai BADEA

1 Ideea solutiei

Perifericul este format din mai multe blocuri care comunica intre ele. Comunicarea cu exteriorul se face prin SPI, iar in interior datele sunt prelucrate pe un bus paralel.

Fluxul de functionare este urmatorul:

- datele vin serial prin SPI
- sunt transformate in octeti in modulul SPI bridge
- comenzile sunt interpretate de decodor
- registrele sunt citite sau scrise
- numaratorul si PWM-ul folosesc valorile din registre

Activarea numaratorului (COUNTER_EN) si activarea PWM-ului (PWM_EN) sunt complet separate, conform cerintei. Doar atunci cand ambele sunt active, la iesirea pwm_out apare un semnal PWM.

2 SPI Bridge

Am modificat antetul:

- `output reg miso`
- `output reg byte_sync`
- `output reg [7:0] data`$_in$
- `input wire [7:0] data`$_out$

Avem CPOL=0 si CPHA=0, adica:

- masterul pune datele pe linie pe frontul descrescator

- slave-ul (perifericul nostru) citeste datele pe frontul crescator

Enuntul spune ca SCLK si ceasul perifericului sunt considerate sincrone (ambele 10MHz). Din motivul asta, implementarea SPI bridge a fost facuta pe ceasul `clk`, iar semnalul `sclk` este doar folosit pentru a detecta fronturile (rising/falling). Practic, esantionam `sclk` pe `clk` si obtinem doua semnale interne: front crescator si front descrescator.

Pentru receptie, pe fiecare front crescator de `sclk` citim un bit de pe MOSI. Dupa 8 biti, formam un byte complet si generam un puls `byte_sync` de un singur ciclu de `clk`. In acel moment, `data_in` contine byte-ul primit si poate fi folosit de decodor.

Pentru transmisie, pe fiecare front descrescator de `sclk` scoatem urmatorul bit pe MISO (MSB primul). La inceputul tranzactiei (cand `cs_n` devine 0), pregatim primul bit imediat, ca sa fie stabil pana la primul front crescator (cand masterul citeste).

Cat timp `cs_n` este 0, pot exista oricate secvente de cate 8 biti, iar modulul continua sa genereze `byte_sync` pentru fiecare byte complet receptionat.

3 Decodorul de instructiuni

Modulul `instr_dcd` interpreteaza comenzile primite prin SPI. Implementarea foloseste o masina de stari simpla, cu doua stari.

In prima stare se primeste comanda, care contine:

- tipul operatiei (read / write)
- adresa registrului

Daca operatia este de tip read, semnalul `read` este activat imediat, astfel incat registrul sa poata pune datele pe magistrala inainte de al doilea byte SPI.

In a doua stare se primesc datele propriu-zise, folosite doar in cazul unei operatii de write. Semnalul `write` este activ un singur ciclu de ceas, pentru a evita scrieri multiple.

4 Registri

Modulul `regs` contine toate registrele configurabile ale perifericului.

Scrierea in registre se face sincron cu ceasul `clk`, doar atunci cand semnalul `write` este activ. Citirea este combinationala si depinde de semnalul `read` si de adresa selectata.

Registrul `COUNTER_VAL` este read-only si reprezinta valoarea curenta a numaratorului. Registrul `COUNTER_RESET` reseteaza doar numaratorul si este implementat ca un puls de un singur ciclu, fara a afecta ceilalti parametri.

5 Numaratorul

Modulul `counter` implementeaza un numarator pe 16 biti, care poate numara crescator sau descrescator. Incrementarea sau decrementarea se face in functie de semnalul `upnotdown`.

Pentru controlul vitezei de numarare, am implementat un *prescaler*. Acesta este un contor separat, care ajunge pana la valoarea $2^{prescale} - 1$. Doar cand prescalerul ajunge la aceasta valoare, numaratorul principal isi modifica valoarea.

Numaratorul functioneaza doar atunci cand `COUNTER_EN` este activ. Valorile din registre pot fi modificate in timpul functionarii, dar efectul lor se vede doar la overflow, underflow sau reset.

6 Generatorul PWM

Modulul `pwm_gen` foloseste valoarea curenta a numaratorului pentru a genera semnalul PWM. Modul de functionare este ales prin registrul `FUNCTIONS`.

Sunt implementate urmatoarele moduri:

- PWM aliniat la stanga
- PWM aliniat la dreapta
- PWM activ intre doua valori de compare

PWM-ul este generat doar daca `PWM_EN` este activ. In cazuri speciale, cum ar fi `compare1 == compare2` sau un interval invalid, iesirea este fortata la 0 pentru a evita comportamente neasteptate.

7 Testare

Am adaugat un test suplimentar pentru dezactivarea PWM-ului in timpul functionarii, pentru a verifica faptul ca iesirea este fortata la 0 indiferent de starea numaratorului.

In mare parte, testele din testbench-ul dat au trecut - ce am facut pana acum functioneaza conform cerintei *(hopefully)*.